SOFTWARE ENGINEERING FOR ALL – WHY AND HOW

by

Mahrukh Sameen Mirza, B.E.

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Software Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

MAY, 2020

SOFTWARE ENGINEERING FOR ALL – WHY AND HOW

by

Mahrukh Sameen Mirza

APPROVED BY

Soma Datta, PhD, Chair

James Helm, PhD, Committee Member

Jana Willis, PhD, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

David Garrison, PhD, Associate Dean

Miguel Gonzalez, PhD, Dean

Dedication

This thesis is dedicated to my parents, siblings, and my very supportive husband.

Acknowledgements

Firstly, I would like to thank my parents for making me better human and always teaching me the value of time and education. Without their teachings, I wouldn’t have even thought of being where I am today. I can never pay back the sacrifices they made for me and my education. I miss you mom, dad, Chickoo (Sadaf, my sister) and Chazu (Shahzain, my brother).

I always dreamt of doing Masters from the United States of America but considering the situation of females back home, that dream looked near to impossible. Then, I got married to my most loving and supportive husband. He is that person who supported me to pursue my dream. Abrar, my Master’s degree is dedicated entirely to you. I remember the time you worked hard day and night just so you can pay my Masters tuition fee. I am blessed to have you as my soul mate.

I would like to thank Dr. Soma Datta for being an inspiration and helping me with my research work. I learnt a lot under your supervision. You are my wonder woman and I will always look up to you. Dr. Jana Willis, thank you for joining the committee and helping me out with my work. You inspire me for my next career goal. Dr. Sadhana Weerasinghe, thank you for taking time off your busy schedule in order to teach me few Mathematical concepts.

Lastly, I would also to thank Atchyutha for being a part of my thesis testing.

ABSTRACT

SOFTWARE ENGINEERING FOR ALL – WHY AND HOW

Mahrukh Sameen Mirza
University of Houston-Clear Lake, 2020

Thesis Chair: Soma Datta, PhD

A lot of the software industry has moved from the planned method of software development to an Agile development process. Agile is one of the widely used methodologies today and has several benefits over the planned method. While Agile has taken over the software industry, it is also expanding in the field of education. It is a great tool for better organization and rapid feedback. The following study starts by stating differences between planned and Agile software development processes. Next, to demonstrate how Agile can be used in other non-software related environments, this study shows a pilot study conducted with a group of online students through use of the ensemble method named “Feature Driven Scrum” (a tailor method created by the amalgamation of Feature Driven Development and Scrum). The study shows how Agile is a great tool for organization and self-assessment. Agile has several principles and manifestos, and these are similar to the ones supported by Design Thinking. A course named “Agile Design Thinking” has been proposed in this study to show how Agile and

Design Thinking support the same principles. Several games have been introduced in the course, which help students understand concepts better and retain those concepts for a longer time. In order to help students, learn and understand this course better, a pedagogy using Agile principles has been developed. This pedagogy can be used to teach students from the Engineering discipline as well as other disciplines. The pedagogy has been tested in a Data Science course, and the results are stated. Lastly, an e-learning android application that uses flashcards (using C# and XML, Xamarin platforms) was developed using Agile Design Thinking principles. This application can be used to enhance the student learning experience. The application has been validated and findings show that Agile and Design Thinking support the same principles and that great customer experience can be created using the Agile Design Thinking principles.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:
INTRODUCTION

For a long time, the planned method [1] was used for developing software. These planned models were known to be cost saving for bigger, off shore projects. Although they were widely used, they still had limitations, which resulted in software failures [1]. These failures led software practitioners to develop an “Agile Manifesto” [2]. These manifestos support individuals and interactions over processes and tools, support working process over detailed documentation, support customer collaboration over contract negotiation, and support responding to change over following a plan [3]. The software industry saw huge shift when they used Agile methodologies over planned methods such as fewer software failures, better teams and organizations, and rapid delivery. Under Agile there are methodologies like Scrum, XP, DSDM, Lean, Kanban, and Crystal that exist. The strengths and limitations of different Agile and planned methods are listed later in this study.

While Agile has gained popularity in the software industry, it can also be used in non-software related industries such as a non-software related work environment, an online study group, or any other industry such as telecommunication, education, and supply chain management. In order to overcome the limitations of one methodology, different methodologies are often tailored together. Similarly, two Agile methodologies, Scrum and Feature Driven Development, have been tailored together to create “Feature Driven Scrum”. This ensemble method has been used to demonstrate how Agile can be used in non-software related workspace. The methodology has been tested and the results are stated in Chapter V (Results).

Agile has benefits like fewer software failures and faster delivery, but are all these developed applications actually used? There are several creative applications available, but more than half are hardly used. The applications completely misalign with the outcomes users expect. Before developing anything, it is important to know what is important to the customer, what are their problems, will they use what is developed, and are they motivated to have a new solution. Developing wasteful application results in wastage of a lot of resources including time, money, and resources. In order to avoid such wastage it is important to take steps early, even before the development starts. Great user experiences can easily be created using Design Thinking framework [4]. Several benefits are associated with design thinking, which are innovation, customer satisfaction, organizational transformation, and better decision-making. While Agile is a way to solve a problem, design thinking is a way to find the problem. But are they better together? A course, which uses principles and methods of both Agile and Design Thinking, was introduced in this study. This course demonstrates how both Agile and Design Thinking support the same principles. As an example of how an Agile Design Thinking course is beneficial to develop useful software, an android mobile application was developed using the principles of this course. This application was developed in C# and XML language using the Xamarin Platform in Visual Studio 2019. It is an e-learning application where the user can add their questions and answers using flashcards. Questions are added on the front of the flashcard and answers are added on the back. With a double-tap from the user, the flashcard flips to show answers. In order to make the learning and teaching experience better, several games and a pedagogy using Agile principles were introduced. This pedagogy uses different Agile principles like daily stand up meetings, pair programming, effort estimation, and working agreement. This pedagogy was tested in a data science class and the results are stated later in this study.

CHAPTER II:
LITERATURE REVIEW

According to Mandal et. al [3], the limitations of planned software development methods are excessive documentation, too sequential, excessive planning, a lack of results until the end, late communication to stakeholders, delays in project delivery, and increased project costs. Twelve Agile principles and four Agile Manifestos were created to overcome the limitations of existing planned software development models. In spite of the fact that Agile has been successful in overcoming the limitations, it still has its own limitations [5]. Overall, the limitations of Agile as stated by Tarwani et. al [5] are miscommunication, resource increase, overall cost increase, inappropriateness for large projects, and lack of coordination.

In order to learn more about strengths and limitations of Agile and planned methodologies, a systematic literature review of 25 papers was conducted. The search strategy resulted in 91 papers initially, out of which 25 were selected as primary studies from year 2012 to 2019. Different keywords were used to search for papers. Out of those 91 papers, using keywords, 72 papers were selected by reading the abstract. Out of 72 papers, considering the inclusion and exclusion criteria, 25 papers were selected for detailed reading. The search keywords are listed below in Table 2.1.

Table 2.1

Search Keywords

Subject	Search Keywords
Traditional	Traditional software development OR traditional Agile OR software development life cycle OR SDLC OR traditional models OR traditional model OR traditional software model OR traditional software models OR waterfall Agile
Agile	Agile methodologies OR Agile software OR Agile development OR XP Agile OR eXtreme programming Agile OR Scrum Agile OR Crystal Agile OR DSDM Agile OR dynamic system development method Agile OR FDD Agile OR feature driven development Agile OR Lean Agile OR Kanban Agile OR Agile manifesto

The inclusion criteria were as follows:

1) Papers published between 2012 to 2019
2) Papers written in English
3) Papers that were scholarly & peer reviewed and journal articles
4) Papers having computer science and engineering discipline
5) Papers having search terms software engineering, software, and engineering
6) Papers where the search terms were found in the abstract
7) Papers that spoke about Agile, traditional, or core engineering design process

The exclusion criteria were:

1) Papers that are duplicates of papers already included
2) Papers that did not talk about traditional, Agile, or core engineering design process
3) Papers older than 2012

Table 2.2 summarizes the strengths and limitations of different Agile and planned methodologies.

Table 2.2

Strengths and Limitations of Agile and Planned Methodologies

Process	Strength	Limitations
Scrum	Results in good communication among team members Widely used and has one of the best management practices Continuous feedback from the customer which results in customer satisfaction Helps in growth of the team and individual through Daily Scrum and Scrum Meeting Produces quality product Can handle unclear and changing requirements	Simple to understand but difficult to master Lacks engineering practices Suitable for small projects
Extreme Programming	Has concepts like continuous integration and pair programming Works well simple and small-scale projects Improves productivity	Less focus on design Less documentation Poor architectural structure Common skillset and understanding between developers are required for pair programming
Lean	Eliminate waste Maximize value of product Reduces waste	Lack of details about implementations
Kanban	Helps in managing the product Increases communication Reduces waste	Lack of implementation details
Test Driven Development	Produces quality product Encourages testing first Removes duplication	Time consuming Requires knowledge and specific skillset
Crystal	Classification of projects is easier Increases communication in the team	Needs special training to write the requirements Only two crystal categories are defined
Feature Driven Development	Produces quality product Adaptive and incremental in nature	Less responsive to change Needs trained staff Not for small scale projects
Waterfall	Simple to use Each phase is clearly defined Detailed documentation	Not suitable for projects with changing requirements
Rational Unified Process (RUP)	Produces quality product Requires less development time	Needs trained staff Complex development process
Spiral	Handles changing requirements Delivers product frequently Lower risk of failure	Can continue without an end point Not suitable for small projects

Software Engineering Curriculum

A systematic literature review of 33 studies was conducted [6] by Garousi et al., which shows how software engineering graduates find it difficult to begin their careers after graduation. This is due to misalignment of skills taught at the university and what is actually needed in the industry. The summary of 33 studies shows that requirements, design, and testing are the most important skills required for a software engineering graduate. Apart from soft skills such as professionalism, group dynamics, and communication skills which are also desirable by the industry. Mathematics and engineering foundations are ranked the lowest desirable skills. The paper later talks about how soft skills are important along with hard skills. Soft skills contribute significantly to the growth of an individual as a professional and they are one of the most desirable skills the industry looks for when selecting an employee. Garousi's and his co-author's findings say universities place more emphasis on teaching software engineering students mathematical topics and underemphasize business topics. They suggest universities must educate their software engineering students using "real world" examples of software systems. They highly suggest universities align their curriculum with respect to topics in requirements, design, testing, and also include soft skills.

A similar study was conducted [7] where the authors investigated if there was a gap between the education provided by the universities in Software Engineering Department and what is expected from a graduate in industry. They conducted interviews of both newly graduated software engineers, having one to three years of working experience, and of hiring personnel at different companies. They investigated areas where recent graduates frequently struggled. They also sought to find out through interviews whether there were other challenges experience than the ones that mentioned by the recruiters. Finally, they investigated whether a knowledge gap existed between the university and the industry. Their findings are similar to Garousi et al., [6], whose findings suggest universities must improve their Software Engineering curriculum by including real life projects where practical knowledge is included, including soft skills and business skills, and be up to date with present technology. Also, 56% of their interviewees say that Software Testing was something they wished to improve on. The interviewees also talk about improving on software design, computing foundations, and software maintenance. It was found that the most important skills in industry were requirements, design, and testing. The least important skills were Software Engineering Economics, Mathematical Foundation, and Maintenance. The largest gap that existed was with Software Professional Practice and this skill was considered as the most important. The skills that had the largest gap but were the least important are Software Engineering Models, Methods, and Maintenance. The smallest knowledge gap existed in the area of Development and this was the most important skill required by a Software Engineer. The skill with smallest gap and least importance was Mathematical Foundations. Soft skills were also considered as one of the most important skills. The student interviewees desired that the university could add the following to their education – version control management, better and more testing courses, architecture design, better user experience courses, more cooperation, modern technologies, and more development.

A systematic mapping study of papers from 1976 to 2011 was done in "Software Engineering Curriculum: A systematic mapping study" [8]. There was huge contribution of papers towards software engineering education from 1998 to 2011. Papers were written at both the program level as well as the course level, and it was seen that there was more contribution of research towards the course level compared to the program level. The authors suggested that there needs to be improvement in Software Engineering education, and this can be done by putting more efforts towards overall Software Engineering discipline.

According to Boehm et al. [9], a T-shaped person is someone who has technical debt of systems aspect, be it at least one. He is a person who has general understanding of other systems aspects. An I-shaped person has a great understanding of software technology, but they have little understanding of other disciplines such as business, medicine, transportation, and Internet of Things, etc. The author says many graduate students fall into the I-shaped category, which leaves them with poorer understanding of multi-disciplined system thinking, thus why Boehm and Mobasser [9] have developed a curriculum for CS majors with a mission to produce more T-shaped graduates. The curriculum includes different courses such as software management, human computer interaction, software architecture, software requirements, verification and validation, and more. Apart from these courses, the curriculum also includes a long-term real-client project to provide students a platform to apply the skills learned. The validation of this curriculum was completed via feedback from students who engaged in internships and found jobs after completing the curriculum. The curriculum proved to be effective as the feedback indicated students and their employers had high rates of success in job offers and job performance.

A study conducted by Devadiga [10] compared the dynamics and engineering practices of employees working at startup companies and how their exposure to the current curriculum of Software Engineering benefitted them in their jobs. The authors indicated universities needed to adopt the latest development trends in their curriculum so Software Engineering graduates would develop better technical, interpersonal, and communication skills. Software processes, engineering, and DevOp practices should be included in the curriculum so graduates develop their abilities to take up multiple roles.

Graduates who join established companies do not have problems transitioning to the development environment as these organizations have training programs for incoming employees, but this is not the same for graduates who end up working for startup companies. The environment in startup companies is chaotic, unpredictable, and dynamic. They do not have the resources or the time to prepare their employees. It is important to bridge the gap between the university curriculum and what skills are required to be part of founding team at a startup company so that graduates have the knowledge and skills to succeed.

Software Failures Due to Lack of Understanding the Customer

Software failures can threaten the existence of a company. They result in wastage of precious resources of a company including time, money, and energy. According to Liu [11], a recent study conducted on 5,400 IT projects revealed that 17 percent of the projects threatened the existence of the companies because of their failure. The study revealed that many software failures were due to over and underestimation as well as not being able to satisfy their customers' needs. The study reporeds that 45 percent of the projects were over budget, seven percent were not scheduled, and 56 percent delivered less value than expected.

Agile is a process which supports customer satisfaction. Companies' success comes from producing quality software's, which satisfies the customers, and they start building customer centric products in order to maintain long-term relationships with their customers [12]. They know customer satisfaction will occur when developers know the customer well and understand their needs. Therefore, customer satisfaction has been identified as the central business factor [12].

Leem et. al in [13] stated many companies fail to keep up with continuous implementation of customer satisfaction. In order to help maintain customer satisfaction, they developed an evaluation system by combining traditional software process assessment models with general customer satisfaction models. However, in this paper, the authors did not specify what the actual reasons were for companies that were failing to keep up with their customer satisfaction.

Pedagogy using Agile principles

While Agile has gained popularity in the software industry, it has started gaining popularity in education as well. Several instructors from different disciplines are trying to use Agile principles in their pedagogies. Using Agile encourages students to ask questions and work collaboratively as a team [14]. Teaching with Agile provides better experiences for both teachers and students, as it encourages trust, engagement, and accountability among students [14]. With the many advantages Agile has on teaching and learning, it can also be used in online courses [14]. A major concern with online courses is facilitating collaboration and clear communication among the team members and with the instructor. Using Agile principles in online courses has yielded good results. It helped students to have better learning experiences, deliver their projects on time and with quality, and helped them use their time effectively. It helped them to keep track of work and kept them accountable for their time.

Krehbiel et. al in “Agile Manifesto in Teaching and Learning” [15] created Agile Manifesto of Teaching and Learning. These manifestos can be used to direct the work of higher education faculty in the classroom and beyond. The results indicate that using Agile principles increased student engagement, encouraged students to take responsibility of their learning, enhanced quality, increased collaboration, and produced quality deliverables.

Agile supports collaboration between team members, users, customers, and stakeholders. This is an important part of Agile and students must learn collaboration practices, which can be handy when joining an organization [16]. The authors [16] show educational projects can be set up in such a way that students understand the importance of collaboration. They use Agile collaboration techniques to accomplish this. Modern collaboration tools have been used to coach students and assess their progress. This helped students understand how Agile is used for collaboration and why collaboration is important.

In this chapter (background), strengths, and limitations of different Agile and planned methodologies has been discussed. Two methodologies were selected to create an ensemble process named “Feature Driven Scrum” to show how Agile could be used in non-software environment. This chapter also revealed that Software Engineering curriculums in universities need to change and courses that teach students about design, soft skills, and testing should be added. Students need to learn how they can contribute to the organization they work for by reducing the number of software failures because software failure is a real problem, reusulting in wastage of resources. Hence this study proposed an “Agile Design Thinking” course, which teaches students design thinking in Agile ways and teaches them how to reduce failures by empathizing with the customers. In a way, this course teaches students soft-skills that are required for collaboration and teaches them why collaboration is an important part of development. Without collaborating and understanding the customers, customer needs are clear, which in turn results in software failures. Apart from this, this chapter also revealed that Agile is beneficial when used in pedagogy for teaching and learning. Hence, this study proposes a pedagogy using Agile Manifestos and principles to teach Agile Design Thinking course.

CHAPTER III:
RESEARCH QUESTIONS

RQ – 1 Can Agile be used in a non-software environment?

RQ – 2 How can Agile and Design Thinking help to reduce software failures? Do they support same principles and manifestos?

RQ – 3 How can Agile Manifestos and principles enhance teaching and learning experience?

CHAPTER IV:
METHODOLOGY

Feature Driven Scrum

In this section, Feature Driven Scrum methodology is discussed. A new course named Agile Design Thinking was introduced, which has several games and activities to retain the concepts for a longer time. In order to make the learning and teaching experience better for both teachers and students, a new pedagogy using Agile Manifestos and principles is also introduced later in this section. Along with this, an e-learning application named Flashcards is discussed. The testing of Feature Driven Scrum and pedagogy have been done and its results are presented in the next section.

A new ensemble methodology named “Feature Driven Scrum” was developed in order to test whether Agile can be used in a non-software environment. Scrum and Feature Driven Development were tailored to create this methodology. The testing of this methodology was conducted with an online study group of students and in a non-software educational workspace.

Study groups help students to develop self-learning skills. They also help students develop confidence, collaboration, communication skills, and make them better decision makers. It teaches them how to be accountable for their own work and time. Students learn a lot from being a part of study groups, they learn from each other through discussions, and learn to respect their peers [17]. The steps involved in Feature Driven Scrum are as follows:

1. Develop an Overall Draft
2. Planning Meeting
3. Daily Scrum Meeting
4. Sprint Review Meeting

5. End of Sprint Retrospective

Each step is discussed below in details.

Develop an Overall Draft – this is the first step of Feature Driven Scrum. It aligns with the first step of Feature Driven Development (FDD). An overall draft plan is developed in this phase. This draft is a high-level plan of the work to be completed. During this step, the team can brainstorm and discuss ideas about how to achieve their final goal. This step also teaches the team about the importance of communication in Agile development and that Agile is a people centric process.

Planning Meeting – after brainstorming and developing an overall draft of how to reach the final goal, the next step is planning the meeting. This step is similar to the Sprint Planning Meeting in Scrum. Here, the team decides what their goals are for the next sprint. The goals are small and achievable in one sprint. For this study, each sprint was three days long while this methodology was tested.

Daily Scrum Meeting – the team meets every day to discuss their progress. This step aligns with the Daily Scrum Meeting step of Scrum. The team discusses "what did they do yesterday", "what are they doing today" and "what are the obstacles that they are facing?". This step shows each user how bigger chunks of work are broken down into smaller achievable goals and how progress, in Agile is measured through work completed. The progress of study groups was measured by the topics they covered each day, and the progress of workspace environment was measured by the amount of work completed.

Sprint Review Meeting – after every sprint, which was 3 days long, the team conducts a Sprint Review Meeting just the way it is done in Scrum. The main objective of this meeting is to reflect on the previous sprint and examine the work done. The team checks back to see if they reached their goal that they had decided to reach by the end of the sprint. The goals that were not achieved in the previous sprint are considered as backlogs. They learn from their experience and discuss how they can improve for the next sprint. After this meeting, plan-by-day and plan-by-week meeting is conducted to decide the goals for next sprint.

End of Study Retrospective – this step is conducted at the end of the project. The team reflects and discusses about what they learned from the process and how they can improve for the next time.

The evaluation of Feature Driven Scrum is given in the "Results" section (Chapter V).

Agile Design Thinking

What is Design Thinking and Why Do We Need It?

Design thinking is not just a methodology or a framework, but a process where empathy of the user is developed in order to know what their problem is and what is important to them. Just like problem solving, design thinking is natural and ubiquitous human activity [18]. It helps to investigate the reason you are building that software or service. According to Stanford d.school, the framework for design thinking includes five main steps – Empathize, Define, Ideate, Prototype, and Test [4]. It gives several unique and creative ways to approach a real problem. It is completely different from the normal way of business thinking where designers give solutions to a problem. While in design thinking, designers first try to understand the users and the problem better and then provide a meaningful solution to the problem. Through design thinking, designers try to investigate if users are motivated enough to use this new solution leaving behind the old one. Large companies like IDEO and IBM are moving towards design thinking. It involves the entire team along with the designers. According to [19], "design thinking is a human-centered innovation that draws from the designers toolkit to integrate the needs of people, the possibilities of technology, and the requirements for business success".

Different software companies develop different applications, but how many of them are actually used? There are several creative applications available, but more than half of them are hardly used. The applications completely misalign with the outcomes users expect. Before developing anything, it is important to know what is important to the customer, what are their problems, will they use what is developed, and are they motivated to have a new solution? Developing wasteful applications result in wastage of resources including time, money, and manpower. In order to avoid such wastage, it is important to take steps early, even before the development starts. Great user experiences can easily be created using design thinking framework. Several benefits are associated to design thinking, which are innovation, customer satisfaction, organizational transformation, and better decision making [20].

Agile and design thinking are different in that Agile is a way to solve a problem and design thinking is a way to find the problem [21]. But are they better together? Do they support the same principles and manifestos?

What is the Goal of this Course?

Several creative products often end up being something no one wants or uses. Almost more than half of the products developed by the software industry suffer this problem. It is important to know what exactly is valuable to the customer so that resources, time, and money developing wasteful products are saved. Doing so early in the development process helps. This course combines Agile and design thinking so that the budding software engineers know how important it is to develop a meaningful product. Developing meaningful software helps the company to reduce cost, time, and effort on wasteful products.

How Will Students Benefit from this Course?

At the end of this course, the students will be able to use Agile framework to understand what is valuable to the customers. They will be able to test their ideas using Agile user stories and prototype. They will understand the use of design methods. Apart from this, students will use design sprints, understand usability testing, and product architecture. They will learn how to test their value propositions to make sure it is usable before starting to build the product. Following in Table 4.1 is the Agile Design Thinking course curriculum.

Table 4.1

Agile Design Thinking Curriculum

UNITS	**TOPICS**	**SUGGESTED GAMES/ ACTIVITIES**	**ESTIMATED COMPLETION TIME**
Design Thinking Framework and Agile	Introduction to Agile Difference between traditional and Agile Agile Manifestos Different Agile Methodologies Introduction to Design Thinking Framework Steps in Design Thinking Framework		1 class
Empathize	Interviewing the customer Personas Empathy maps Customer journey maps	Making breakfast for your grandparents [22]	1 class
Define	4 W’s and 5 Y’s Epics User stories Child stories INVEST Moscow POVs		1 class
Ideate	Brainstorming Storyboarding Mind mapping	Back to high school [23]	1 class
Prototype	Types of prototyping Kano model MVP Time boxed prototypes	Spaghetti and marshmallow exercise [24]	1 class
Test Design sprints in 5 days	Usability testing Test and gather feedbacks Design Sprint 5 days		1 class

Unit I

Design Thinking Framework and Agile

Introduction to Agile and Manifestos of Agile Software Development

Agile software development is one of the most widely used methodologies in the software industry. It has changed the software development game since its evolution. The term 'Agile' was popularized by the Manifesto of Agile Software Development in 2001 [25]. It supports iterative and incremental development process and supports collaborative effort of self-organized and cross-functional teams. Compared to traditional software development processes, Agile provides early delivery, customer satisfaction, response to changing requirements, and adaptive and evolutionary development. The Manifesto's of Agile Software Development are:

1. Individuals and Interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The items on the left of the Manifesto's are valued more than the items on the right.

Introduction to Design Thinking Framework and its stages

Design thinking plays a major role in success or failure of a project. A lot of software often end up not being used by the customer because they were not what the customer needed. The customer either lacks the motivation to use that software, or the team does not really understand what the customer needs. Several software could have been successful just by developing empathy with the customers. In simple words, design thinking is a process that helps the organization gain understanding of the customer and his or her needs better. It also helps the team to stay innovative, be creative, challenge assumptions, and redefine the problems of the customer. It helps the team to think out of the box and create news ways to address customer problems [26]. Using design thinking, software can be prototyped even before they are developed so organizations know if the customer will actually use the developed software over the present alternative. It is an iterative process where testing happens frequently, starting from the earlier stages. It helps to reduce wasteful software, and in turn it also helps to reduce the cost, effort, and time wasted building the software. It does not only help to cut wastage but also is good for communication in the organization. Design thinking brings all the members of the organization together. Everyone can be a part of this team including designers, developers, managers, and sales persons. Design thinking helps to tackle ill-defined and unknown problems.

There are 5 stages to Design Thinking Framework [4]:

1. Empathize
2. Define
3. Ideate
4. Prototype
5. Test

Design Thinking Framework in Figure 4.1 below shows different stages of design thinking and the flow of the framework.

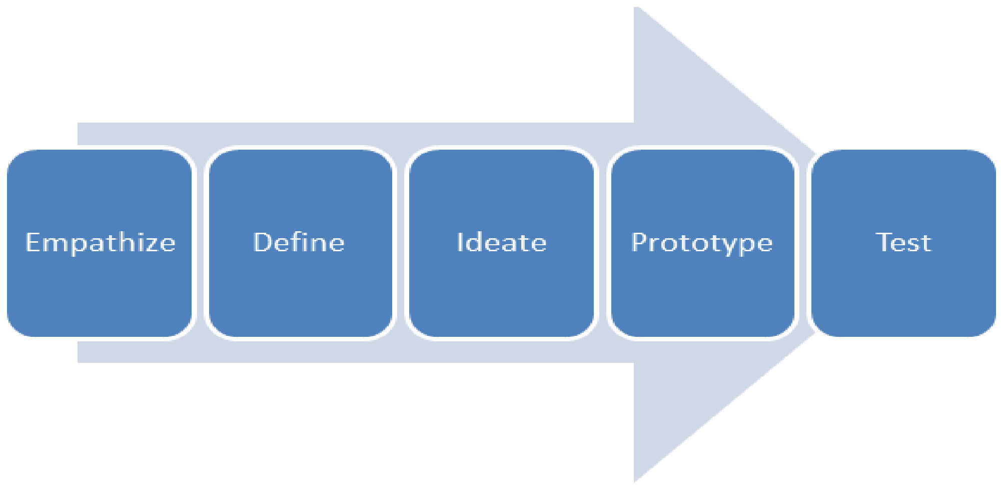


Figure 4.1
Design Thinking Framework

Let us have a brief look at each of these stages.

1. Empathize – the very first stage and the most important stage. It is this stage where the team gets to know their customer and their needs better through various means. Without gaining empathy with the customer, it is not possible to move on to the next stages.
2. Define – the next stage is where after empathy with the customer, the team defines customer needs and problems. After gathering information in the first stage, this information is then analyzed and synthesized. Defining the problem with a different point-of-view (POV) will lead to several innovative ideas.
3. Ideate – this stage is about thinking outside the box. Several methods such as brainstorming, mind spacing, and many more are used to generate as many ideas as possible.
4. Prototype – it is in this stage that the team tests if their proposition will be used by the customer over the present alternative. The team creates a minimum viable product (MVP) to experiment and identify the best possible solution. This is an iterative process where several inexpensive prototypes are created.

5. Test – They say at the Stanford d.school, “Prototype to see if you are right and test to see if you are wrong”. In this stage, prototypes are tested with real customers, and feedbacks are collected for betterment.

While design thinking practice is considered an important one by different industries, it is still new in the software development industry, especially with Agile [27]. Both Agile and design thinking are iterative in nature and focus on customer satisfaction. While Agile software development promises project success, a lot of software still fail [27]. Although all the methodologies (Scrum, XP, Lean, Kanban etc.) under Agile provide different ways to develop software, manage the development of the software, and values the customer, there are still software failures. Using Design Thinking, software can be saved from failing simply by connecting with the customers and knowing their needs better. Understanding and meeting the end users’ expectations are the main factors eto ensure software success . Thus, design thinking is a strong tool to effectively solve problems and can be used with Agile software development [28].

Throughout this study, we will discuss several methods used for different stages of design thinking while also seeing how Agile Manifestos and principles support design thinking framework.

Unit II

Empathize

Introduction

Empathy is the heart of design thinking process. It is the ability to understand the person in front of you and share their feelings [29]. In a normal company set up, teams spend hours trying to brainstorm innovative ideas on a whiteboard without paying attention to customer problems. These teams hardly connect with the customers to know what would satisfy them or if the customer will use the product they are developing.

Teams often end up getting small survey feedbacks from customers to evaluate user experience. No doubt, this will help but not completely. Customers are hard to predict, they might say yes to whatever is asked in order to save their time and the developers. Discovering the customer and his needs is the foundation of design thinking. By practicing empathy, the customer's situation is seen from the customer's point of view. Good empathy will provide a whole new perspective about the customer and his needs [30].

Interviewing the customer

By interviewing the customer and asking what they need upfront is not going to tell what they need. If the customer is asked – is that what you needed? Do you like it? The customer probably will say yes simply because they wants to save time. Customer know that if they say no then there are several other questions waiting for them in line.

It is best to interview the customer in person, but this can be done over a phone call or video chat. Interviewing the customer in person helps observe them while they are in their "everyday experience." Observing the customer helps to see if what they say they do is actually what they do or not do. Observing the customer's body language, tone of voice, and reaction reveals a lot about the customer. Interviewing this way helps to focus on the subconscious aspect of the user. This helps the team uncover the differences and to propose unexpected yet innovative, meaningful solutions to their problems [31].

- The very first step to conducting an interview is to create an interview guide. Asking straight questions to the customers should be the least priority.
- While some questions only reveal irrelevant details, others would provide great insight about the customer.
- It is always good to have one person for asking questions and other person for taking notes so that you can reflect back later and discuss.

- Ask “when was the last time” and “what are the five things” questions more. This will get the customer to give long answers which definitely would be useful.
- Try to explore emotions like “how do you feel” or “what do you think about” [32].
- Don’t forget to go deeper. Try to explore their motivation and emotions by asking “why?”

Personas vs Persona

To make valuable software that is successful, it is important to understand the customer and understand what really matters to them. Personas is an effective tool. It is nothing but a simple humanized description of the customer. It helps to understand goals and behavior of the users. They also help to make a better product, write better user stories, run better Lean startup experiment, and facilitate collaboration. Apart from this, personas help in developing a better hypothesis. Personas are created by interviewing the users. A persona should have the following four fundamentals – a name, screener, description, and perspective. The name should be a real name like Andrew the driver and Laura the florist. Next, always starting with a screening question helps to identify the subject. Example: for “Laura the florist,” ask a question like “How many flower orders have been completed in the last five months.” Ask descriptive questions like who the persona is and what keeps them motivated. Focus the perspective of a persona using empathy maps. What the persona thinks, feels, sees, and does helps to focus on the perspective.

It is desirable that a persona be vivid, testable, and durable for weeks, months and even years. If a persona is not testable or is vague, then it means it is not written correctly. Usually a persona is written in a generalized manner. An example of a bad persona is provided in Figure 4.2 –

Jeffery, the father

Figure 4.2
Google Image of a Person

- Men
- Age 35-45
- Goes to work
- Has kids
- Active on social media
- 50% said they said want to be more organized
- 70% said they would use the application to become more organized

The persona in Figure 4.2 is a generalized one. It does not really tell exactly what the customer wants.

So how do we know if our persona is good enough? A good persona always tells a story. Goodness of a persona can be determined by using a REACT list. REACT is Real, Exact, Actionable, Clear, and Testable. An example of a better persona is provided in Figure 4.3–

Figure 4.3
Google Image of a Good Persona

Adam the father

Adam is a software engineer and a single parent to his three sons. He is a successful engineer with a handsome salary. He manages to be spend quality time with his sons in spite of his busy and hectic schedule. He goes cycling with them every morning and swimming every weekend. He says being a single father is a job. While he is both a good engineer and a good father, he wants to be his best at his both jobs. He wants to be organized and keep in touch with other single fathers to know how they are doing. He likes to socialize and is active on both Facebook and Twitter.

The example Figure 4.3 has a name. The use of full name helps with vividness. The use of full sentences shows the detail of the persona. It is also desirable to use a real picture of the persona. This persona is real because it talks about a real father and its desire. It is exact and actionable. It is also testable. A good persona helps you make decisions and design ways to test whether you are producing something valuable or not.

Empathy Maps

Creating an empathy map is another way to gain empathy from the customer; like user personas. It helps gain deeper insights of the customer and their needs. What the user thinks, sees, feels, or does is uncovered by using empathy maps. Empathy maps have the following structure, shown in Figure 4.4 [33] –

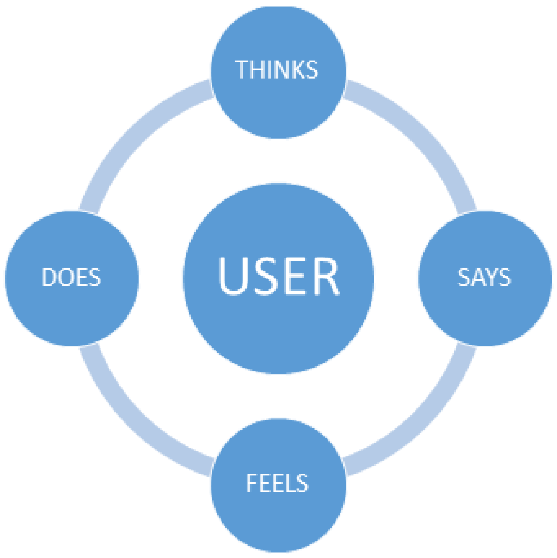


Figure 4.4
Design Thinking Framework

Thinks

Using "think" the team can examine what the customer thinks about the present situation and what does he think should change. A few example questions that might be asked are

- Tell me about your work.
- Tell me about a few things you find hard at your job.
- What would be the one thing you would change out of those things you find hard?

Example of Laura the florist:

Laura thinks the cancelation of orders should be more organized to avoid last minute cancelations. She wonders if there is a better way to know if the customers want to cancel the order before ordering them in bulk. This is important to her because this would save her time and save her unnecessary business deals, which involve losing large a initial deposit amount.

Sees

"See" is a description of how the persona arrived at that point of view, which was discussed in "think." Few example questions that might be asked are

- What do others do?
- Who do you think is doing it right?

Example of Laura the florist for see:

Laura sees that a lot of big orders are mismanaged. She sees her friend who works at another place has more staff to manage big orders. Sometimes, due to many big orders coming in, she has to work overtime.

Feels

"Feel" is emotional thoughts of the persona regarding his job. Few example questions are

- Tell me about the last time…
- What motivates you? What is the most rewarding thing and why?

Example of Laura the florist for feel:

While doing her overtime job to manage big orders, Laura feels angry and feels that the company does not care about her. She feels that the company should pay her more for the hard work she is doing.

Does

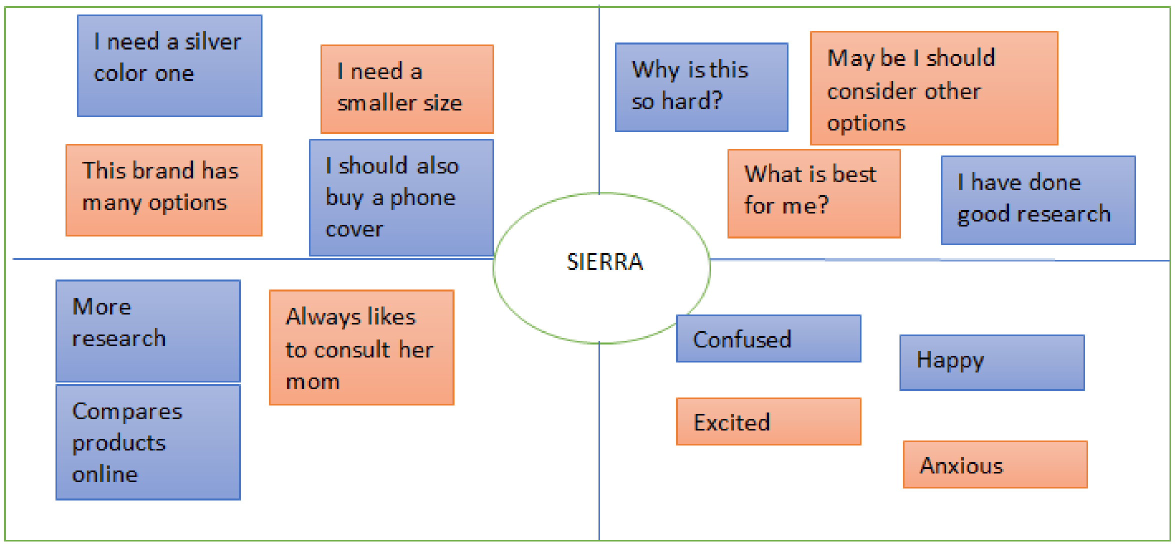
This part is an important one. We really want to know what the persona does and how much he/she does it. Questions that might be useful here are

- How many hours did you work last week?
- How many customers did you attend?
- How many orders did you take?

Example of Laura the florist for does:

Laura typically completes 5 big orders and 10 small orders in a week. Last week she has outdone herself by completing 7 big orders and 15 small orders.

Consider another example of Sierra who is a marketing manager and wants to buy a mobile phone for the first time. An empathy map for Sierra would look like as shown in Figure 4.5 –



I need a silver color one
I need a smaller size
This brand has many options
I should also buy a phone cover
Why is this so hard?
May be I should consider other options
What is best for me?
I have done good research
SIERRA
More research
Always likes to consult her mom
Compares products online
Confused
Happy
Excited
Anxious

Figure 4.5
Empathy Map for Sierra who wants to buy a mobile phone

Customer journey maps

Customer journey maps are another tool that can be used to better understand the customer. They define a process, a customer journey to achieve a particular goal with the organization. They can be used for reference by the entire team or can be used to make decisions. It is one of the best ways to visualize how the customer would react to a particular business [34]. It is a strategic tool, which is good for capturing and presenting customer goals, desires, reactions, and pain points. [31]. As mentioned earlier, it helps the organization collaborate and make collective decisions. It helps to unfold complex customers and helps the team make sound decisions and investments. Several names of customer journey maps exist, such as customer journey, journey map, and experience journey [28]. In simple words, customer journey maps are a series of touch points or emotions that a customer will experience while engaging with the application. There are several templates that exist. Below in Table 4.2 is a simple example of customer sequence of trying to book a table at the restaurant through placing an order [35]. Notice how the emotional aspect of the customer’s journey is also presented in the picture. The example is adapted from Colin Shaw and John Iven’s book named *Great Customer Experience* [35].

Table 4.2

Customer Journey Map Example

STEP	**BOOKING**	**TRAVEL**	**ARRIVE AT CAR PARKING**	**ENTER RESTAURANT**	**PLACE ORDER**
EXPECTATION	They will have availabilit y for me	I can see several forms of directions	I will get my car parking easily	They will attend me with smiles	They had several choices on the menu
THREAT	They are completely booked	Customers have hard time finding the directions	The parking is completely full	They were busy and did not have time for me	They did not have whatever I liked
OPPORTUNITY TO EXCEED PHYSICAL EXPECTATIONS	They had my details from the last time and also had what I ordered	The restaurant has sent me details about their directions	They have a reserved parking for me	They upgraded my table to the city view rooftop table	Chef himself suggested me his best seller item on menu
OPPORTUNITY TO EXCEED EMOTIONAL EXPECTATIONS	The staff remember ed me and remember ed what I ordered, wow!	Menu looks great, I am excited	They called a valet to park my car, wow!	They remembered me and greeted me	They remember ed my order from the last time and got me the same thing
EMOTION INVOKED	Surprise, feels cared	Excitement, they care!	I feel special	I feel happy	I feel cared

Suggested game for teaching empathy to students

Making breakfast for grandparents

This is a simple game to teach empathy to students. The scenario is – imagine you have a visitor; your grandparents for the weekend and you have to make a perfect breakfast for them. You know they normally like eggs. What is not known is how they would like them prepared – omelet style, fried, boiled etc. By using empathize of design thinking process what they would like can be known. Here the teacher can act as a grandparent of students. Students can be divided into teams or can play individually. Students have to empathize with the teacher to know his or her needs better. They can interview the teacher and create any of these– empathy map, persona, or customer journey map. Approximaetly30 minutes are needed for this game. Students can use first 5 minutes to brainstorm and create questions they will be asking. They can interview the teacher for 10 minutes and spend other 15 minutes creating any of the above tools. This game is adapted by Stem Family Games available on [22]. This game by Stem Family does not provide any example solutions. It uses interviewing and brainstorming to demonstrate the entire Design Thinking process. In this study, this game is used to concentrate only on the concept of empathy and different tools are used to gain empathy. An example solution is also provided below.

Before creating any empathy map, persona, or customer journey map, grandfather is interviewed. The following questions can be asked –

- What do you do as soon as you wake up?
- What does grandmother do while you are doing that?
- What time do you wake up?
- What are three things you like about being a morning person?
- Does it bother you if you wake up late?

- Do you help grandmother while she is preparing breakfast?
- Do you like broccoli or any other vegetable served along with your eggs?
- How many eggs do you think would keep a person full for the next 3-4 hours?
- Do you like any juice or coffee served with your breakfast?
- By what time do you think you should finish your breakfast and get to work?
- How do you feel when you visit your son and when breakfast is served by him?

Assuming the answers, an empathy map is prepared as an example solution found in Figure 4.6. However, personas or customer journey maps can also be used. By the empathy map below, we understand that grandfather is health conscious. We see that grandmother understands this and tries to use fresh vegetables and olive oil along with two whole eggs. Through the empathy map below, we understand that grandfather likes to have toasted bread along with his eggs and prefers fried eggs with onions and tomatoes. He feels excited and nervous at the same time because today, the breakfast is served by someone else and not by grandmother. We understand through the empathy map that grandfather feels hungry as soon as he wakes up. We do not know if he can make breakfast for himself but we know that he likes to wait for grandmother. He does not like to disturb her while she is sleeping and tries to help her out by getting other things ready like cutting vegetables and toasting the bread. They also like to work as a team.

Empathy map –

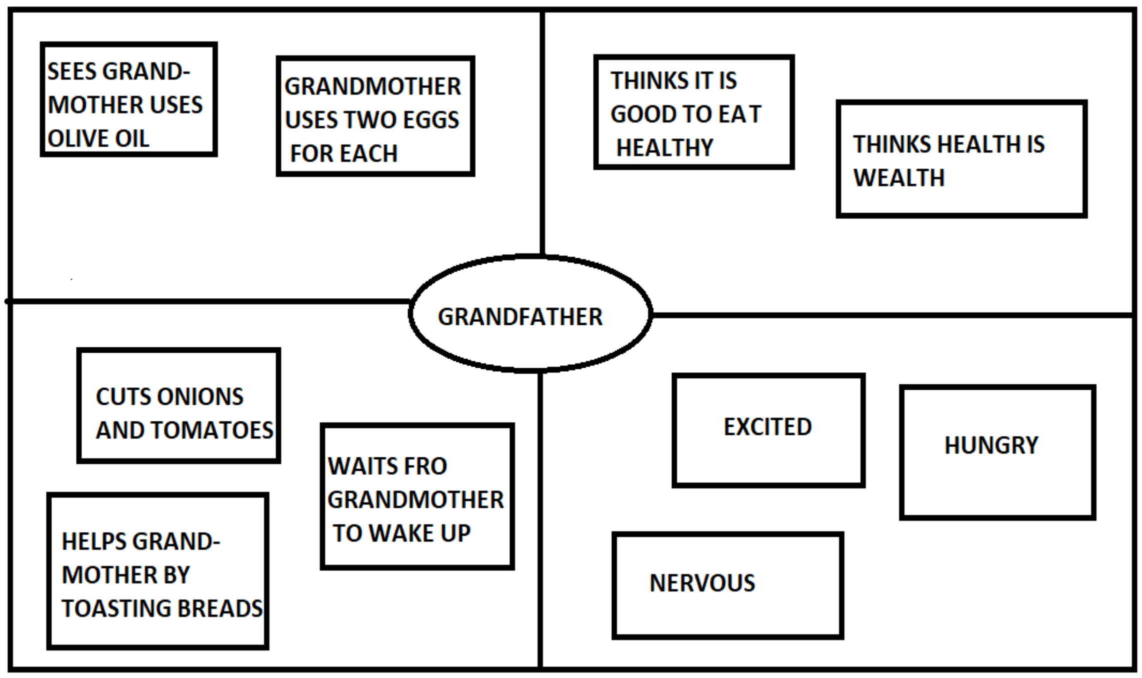

SEES GRAND-MOTHER USES OLIVE OIL
GRANDMOTHER USES TWO EGGS FOR EACH
THINKS IT IS GOOD TO EAT HEALTHY
THINKS HEALTH IS WEALTH
GRANDFATHER
CUTS ONIONS AND TOMATOES
WAITS FRO GRANDMOTHER TO WAKE UP
HELPS GRAND-MOTHER BY TOASTING BREADS
EXCITED
HUNGRY
NERVOUS

Figure 4.6
Example Empathy Map for "Making Breakfast for Grandparents"

Relation of Agile to the Empathize stage

This stage (empathize) of design thinking completely supports one of the Agile Manifesto – "Customer collaboration over contract negotiation." This statement is one of those pillars that has built the foundation of Agile Manifestos. In order to develop software that is acceptable by the customer and that meets customer requirements, regular communication with the customer is important. It is the key to create a great user experience. The empathize stage of design thinking is doing the same thing. Just like Agile, this stage involves the customer to gain insights about their needs, problems, and requirements early in the development process. It supports face-to-face communication with the customer and also brings all the team members together as a team. Agile believes in bringing the business people and the developers together and creating self-organizing teams. Empathizing with the customer is not just the designer's job but it is desirable for all the members of a team to self-organize and communicate with the customers and communicate within the team. Creation of personas, journey maps, and empathy maps encourage communication within the team and helps the team connect by making decisions together. The team becomes stronger and better as they continuously communicate and listen to each other while making decisions about their customers. They learn to value each other's decisions and also learn to value their customer. Hence, the empathize stage of design thinking completely supports Agile Manifestos and principles.

Unit III

Define

Introduction

After collecting stories and insights from the customer, the next step is defining their problems in a formal way. This step is about reframing the problem to create innovative solutions. It gives an opportunity to synthesize the problems discovered and come up with a problem statement [30]. These problem statements are meaningful and actionable in nature. Before understanding how to define a problem in the right way, let us gain an understanding about what is analysis and synthesis. Analysis is breaking down problems into smaller pieces, making it easier to understand the constituent. However, synthesis is bringing small pieces together to make a big picture. An example of analysis can be the empathy stage where a lot of information about the customer is gathered and then broken down into pieces to focus on specifics. An example of synthesis is the define stage where we try to make sense of the data we have gathered in order to make a problem statement [36, p. 2]. Good design thinkers often first analyze the problem to synthesize and later analyze it. A good problem statement should be broad enough to have innovative solutions and it should be narrow enough to be manageable.

4 W’s and 5 Y’s

After researching the customer, questions having 4 W’s (who, what, where and why) and 5 Y’s (why is the customer, why has the customer, why hasn’t the customer, why doesn’t the customer and why is it a problem) should be asked. These questions help to understand if you can move on to the next phase of design thinking – “the define stage”.

Epics, user stories and child stories

User stories help to create a good user experience for the user. They help to define the user needs in achievable goals. They have three clauses – epics, stories, and test cases. Epics are larger user stories, which are eventually broken down into child stories. For explaining epics and child stories, let us take an example of organizing a party. Organizing party is the epic here and the child stories for this could be - first create a guest list, decide on a date and time for party, call all guests to inform them, prepare food according to number of guests, and host the guests. Creating personas and user stories can help the team to debug problems if there is something wrong with the software, if it comes out to be a wasteful product, or if nobody uses the developed product. The format of a user story is as follows –

“As a [persona], I want to [do something] so that I can [realize a reward]”

Here “persona” is a humanized description of a user, “do something” is the goal that the user wants to achieve and “realize a reward” is a testable statement to know if the user has achieved his goal. The realizing a reward is the most important clause of a user story and should not be ignored.

An example of a user story can be – “as an online shopper, I want to compare prices of products so that I can buy things for the best price”. The child stories for this user story would be –

Child story 1 – “as an online shopper, I want to browse multiple options of same category so that I see what options are available”

Child story 2 – “as an online shopper, I want to compare two products of same category and different price so that I see which one is cheaper”

Every user story should be testable and must have important details. A user story must be broken down further into child stories if the story is not immediately actionable.

Testing user stories

For making a product successful, it is important to see two things – motivation and usability. Often software companies are so focused on usability that they forget about the motivation. They see that the product must provide good usability to the user, but do they think that the customer will use the product over his current alternative? That’s the motivation that is needed to be focused on along with the usability.

By usability tests on user stories, only the actual usability of the product is tested without considering the customers motivation to actually use the product. The product may work fine, but that does not mean the customer will use it. These tests are designed assuming that the user is already motivated. It always helps if the test cases are written along with the child stories. For this, consider previous example of online shopper –

Epic story –

As an online shopper I want to compare products so that I can buy them for the best price available

Child stories and test cases –

Table 4.3

Example of Test Cases for Child Stories

Child story	**Test cases**
A) As an online shopper, I want to browse multiple options of same category so that I see what options are available	Make sure it's possible to search by category
	Make sure descriptive information appears to avoid confusion of options
…	…
D) As an online shopper, I want to compare two products of same category but different price so that I see which one is cheaper	Make sure that prices of each item appear
	Make sure the website compares prices for the customer

Table 4.5 helps the team to stay organized and focused. As the team keeps completing each child story, testing of user stories is carried out simultaneously.

INVEST

Writing good user stories comes with time and practice, but one way to write a good breadth of the user stories is by using the INVEST acronym. The full form of INVEST is Independent, Negotiable, Valuable, Estimable, Small, and Testable. These are a set of questions asked to determine the quality and breadth of the written user stories. Are the user stories independent? Each user story should stand-alone without dependence on others. If they are independent, they can be completed in small iterations and can be worked upon spending a good amount of time on each one. By negotiable, it means that the user stories are not requirements. They can be used for communication between the team members and changes can be made when better ideas come up. User stories should be negotiable between collaborators. They are meant to be valuable to the customers and to the development team when they are translated. Are the stories estimable? If the stories are estimable that means they are small enough to work on. If they are not estimable then it is an indication that the user story needs to be broken down further. User stories being estimable leads to “small” acronym of INVEST. Are the user stories testable? For every user story written, tests must be written. Writing functional stories in advance always helps the team.

MoSCoW

MoSCoW is a method used for prioritizing user stories. It stands for – [37] …

M – Must have this

S – Should have this if at all possible

C – Could have this if at all it does not affect anything else

W – Won’t have this time but would like it in the future

Requirements that are in the MUST section must be included in the current delivery time box in order for the product to be successful. The SHOULD haves are those which should be included but are second in the list. They are critical to success but can be done after MUST haves are completed. The COULD haves are less critical and good to have. The WON’T haves should not be included at that particular time [38].

Point of View (POV)

The problem statement that you create in the define stage of design thinking is through POV. It is an actionable problem statement that will derive the rest of the design work. There are 3 main elements to POV – user, need and insight [30]. Three questions are answered in a POV – Who is your user? What is their unmet need? And why is this insightful? A good POV [30]

- Should focus on the problem and state it clearly
- It should inspire the team
- It should help the team to make decisions according to their high-level goals
- Capture the hearts and minds of the customers

Consider an example of a technician who says – "I am a technician and I hate getting stuck in the traffic as I get paid hourly, I feel it eats up my time which I could use for work".

The POV statement could be – "How might we create a way for this technician to avoid traffic and use that time for working peacefully?"

Notice how this statement has all the three elements of POV – user, need, and insight. Here the user is the technician, his needs are he wants to more money than he is making now and from this we gain an insight that the traffic is affecting his work hours. The amount of time he spends in traffic can be used to work and make money.

Relation of Agile to Design stage

Agile Manifesto states that it gives priority to individuals and interactions over process and tools. While several tools are available to get things done, Agile gives importance to interactions and individuals. Collaboration on different levels to focus on details can help the team to create great user experience. By asking questions of each other, new dimensions of a problem can be discovered. New ideas and new insights are always welcomed in an Agile environment. The same goes with the define stage of design thinking. Teams come together to discuss and discover customers. They together create POVs and user stories. Together, they try to think from the perspective of the customer. New insights and new needs are discovered by communication within the team. In define stage, continuous attention is given to the customer needs in order to create good design. Out of twelve Agile principles, this one principle discusses how continuous attention to useful design and technical excellence enhances agility. In a way, the define stage of design thinking totally supports Agile and strives to enhance agility by trying to create a good design.

Unit IV

Ideate

Introduction

Ideate stage is a simple and fun stage of design thinking framework. It should be remembered that this stage should be done only after you understand the customer and their needs. This phase is all about unleashing the imagination and designing creative solutions for solving the defined problem [4]. Brainstorming, storyboarding, and mind mapping are different techniques that can be used to carry out the ideate phase. Good definition of the problem leads to good ideation phase. The team should follow the, "there are no bad ideas," policy through this stage. The team is not trying to find any "correct answer" here, instead they have fun and use their creativity. A broad perspective can be gained by involving several team members from different departments in an organization. It is always good to have several good items on list rather than focusing on one great idea. If that one great idea does not work during the prototype phase, other alternative ideas that were brainstormed during ideate phase can be considered.

Storyboarding

Storyboarding is an effective design tool that helps the team visualizes what the user wants. It helps to predict a user's experience with the product. It enhances the user experience by visualizing how the user will interact with the product. Apart from that, storyboarding is a good communication initiator with team members, and it pushes them to stay creative throughout the process. Through this process the team members should think about what is necessary and, in enough detail, rather than to overdo their thinking process.

As an example, consider Mac who is a software engineer. He is a busy person and hardly finds time for shopping. He chooses to shop online as it is more convenient for him. He wants to order a new phone for himself. Mac sees that the website has many options available, but the website does not compare prices for him. We suggest creating an application which will compare prices for him from all the website he is viewing and provides him the best deal.

Storyboarding of this example in before and after situation are depicted in Figure 4.7 and Figure 4.8 respectively –

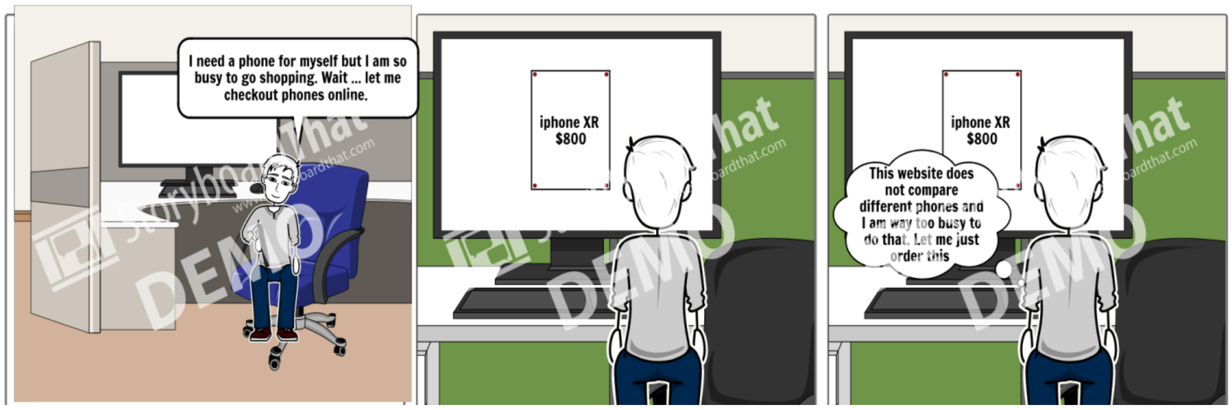

I need a phone for myself but I am so busy to go shopping. Wait ... let me checkout phones online.
iphone XR $800
iphone XR $800
This website does not compare different phones and I am way too busy to do that. Let me just order this

Figure 4.7
Storyboard Showing Before Situation

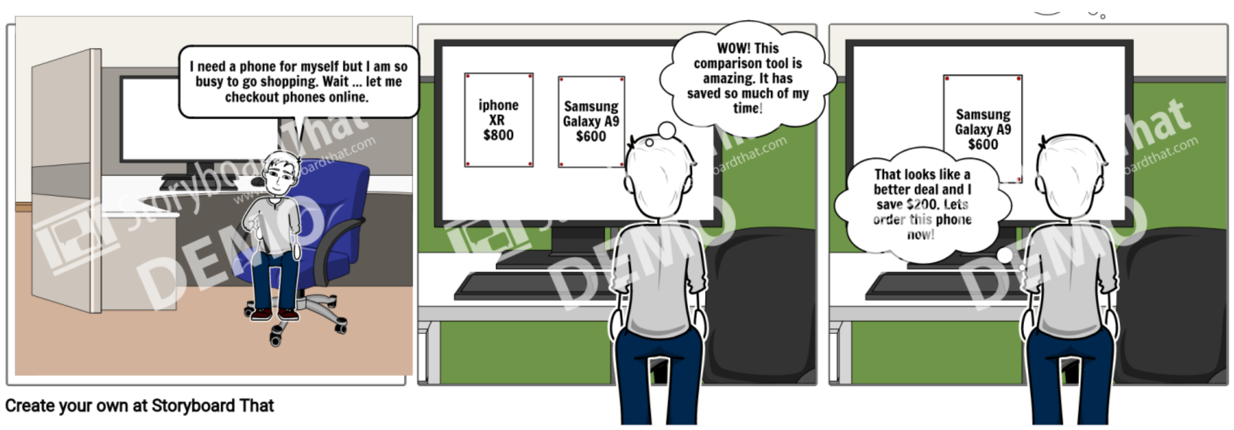

I need a phone for myself but I am so busy to go shopping. Wait ... let me checkout phones online.
iphone XR $800
Samsung Galaxy A9 $600
WOW! This comparison tool is amazing. It has saved so much of my time!
Samsung Galaxy A9 $600
That looks like a better deal and I save $200. Lets order this phone now!
Create your own at Storyboard That

Figure 4.8
Storyboard Showing After Situation

The Storyboardthat.com tool was used to create these storyboards. It is a free online storyboarding tool.

Once the before and after storyboarding situations are completed, the stories can be then broken down to describe them in detail and take notes. Each scene in the storyboard should be broken down as one and notes should be taken on them. Let us again look back at the Mac's example. Table 4.4 and Table 4.5 below show Mac's example of storyboarding along with notes.

Table 4.4

Storyboard Showing Before Situation with Notes

Notes	**Board**
Mac the software engineer is looking for a new phone because he bought his old phone 5 years ago and needs an update. Since he is busy with his work and hardly finds time to go shopping, he decides to checkout new phones online.	I need a phone for myself but I am so busy to go shopping. Wait ... let me checkout phones online.
He checks out IPhone XR at the phones4u.com website but sees that he has to scroll and go back again and again to compare prices and features of other phones with the IPhone XR.	iphone XR $800
He notices that the website does not compare phones for him. Due to lack of time, he just goes with IPhone XR.	iphone XR $800 This website does not compare different phones and I am way too busy to do that. Let me just order this

Table 4.5

Storyboard Showing After Situation with Notes

Notes	Board
Mac the software engineer is looking for a new phone because he bought his old phone 5 years ago and needs an update. Since he is busy with his work and hardly finds time to go shopping, he decides to checkout new phones online.	I need a phone for myself but I am so busy to go shopping. Wait ... let me checkout phones online.
He checks out IPhone XR at phones4u.com website. He sees that by using the comparison tool he is able to compare phones of his choice. The tool also gives him best deals by auto comparing bestselling phones on the website.	iphone XR $800 Samsung Galaxy A9 $600 WOW! This comparison tool is amazing. It has saved so much of my time!
He sees a better deal which saves him $200 and orders a Samsung Galaxy A9 for himself. This has helped him get a better deal and saved a lot of his time.	Samsung Galaxy A9 $600 That looks like a better deal and I save $200. Lets order this phone now!

Brainstorming

There are no steps for brainstorming. All you need is paper, pen, people, and their ideas. It is important to know the aim of the brainstorm session. The team should try to make each other comfortable and tell each other that they are not going to judge anyone. As mentioned earlier, there is no such thing as a "bad idea". Every out of the box idea should be honored and must be written down at least as a future consideration. It is also considered as a collaborative activity, which brings the team together for innovation [39]. Try to keep the obvious solutions aside, remember the team has to think out-of-the-box. Try to set time boxed sessions, which helps to be quick and innovative. Brainstorm session games would elevate if it has a facilitator who keeps asking the team questions like, "how about this", "how might we", or "how do we do this the other way". The team members can encourage each other by building their ideas on other member's ideas. After the brainstorming session, the team votes for a good idea to start with. Other ideas can be honored by keeping them on the future consideration list.

Mind mapping

Mind map is an excellent tool for visualization or imagination. It is a map, which talks about the idea without having much text included. All it has is pictures, colors, arrows, and keywords [40]. They can be used to convey complex information in a very simple way, using figures, tables, or charts [41]. In mind mapping, a series of ideas are created in a tree like structure, which has branches and sometimes even sub-branches. During ideation phase, mind mapping can be used as another technique to visualize the idea that the team has decided to move forward with. Any keyword related to the problem statement can be placed in between and branches can be built surrounding that keyword. The mind mapping activity can be done as a team together or can be done individually and later merged together. They are great for organization, communication, brainstorming, and for discovering connections between different things. They can be used for anything from a technical mind mapping to a very informal one. Let us consider a simple example of mind mapping a dinner party menu (Figure 4.9).

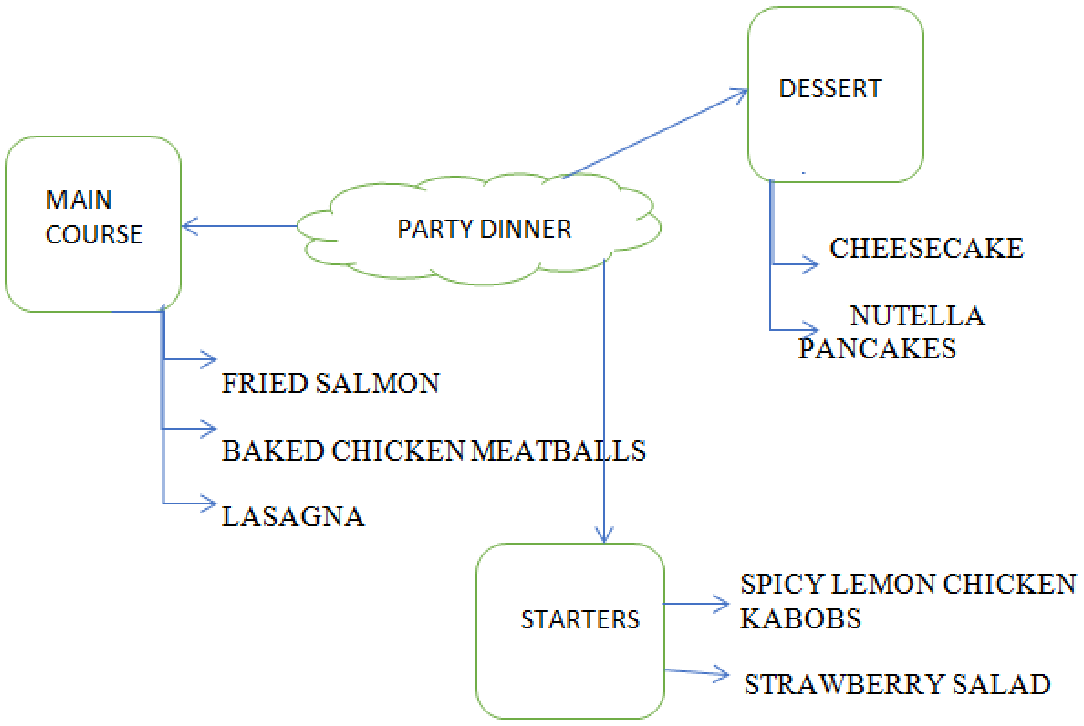


Figure 4.9
Example of Mind Mapping

The example in Figure 4.9 has two levels of mind mapping. The first level has “starters, main course and deserts”. The second level has the dishes in each of starters, deserts, and main course. However, a mind map can have any number of levels.

Relation of Agile to Ideate stage

Agile software development encourages developers to build software around motivated individuals. Individuals in a team can be motivated if they are allowed to speak their minds out. Their confidence level increases immensely and they get a sense of contribution towards the project they are involved in. Agile suggests providing an environment and support to the team members where the team members can collaborate and trust each other to get the job done. Ideate phase of design thinking attempts to provide a similar environment to the team members. It in builds enthusiasm and encourages team members to speak out without being judged. By brainstorming, mind mapping, or storyboarding, team members get a chance to trust and respect each other's ideas. They learn to honor the other team member's ideas. Agile and ideate phases of design thinking both support innovation in order to create good products for the customer. Satisfaction of the customer is the first priority for both Agile framework and design thinking framework.

Suggested game/activity to teach students about Ideate stage of design thinking

Back to High School

This activity is an individual activity where students would need a paper and a pen. A scenario is given to them, which states – "Go back to your high school memories. What was that one problem you faced while you were in your high school and could never solve ? Brainstorm five ideas you would use to solve that problem. Ideas can be crazy and will never be judged. Based on one best idea, create a storyboard."

This activity requires an estimated total time of 20 minutes. Five minutes can be given to students to brainstorm 5 ideas, and another 5 minutes to create a storyboard for one best idea. The 10 minutes resting period can be utilized to discuss these ideas with the entire class. This activity is adapted from Stanford d. Schoolgrand [23].

Unit V

Prototype

Introduction

After the Ideate stage, next is the prototype stage. This stage is entered with several good ideas that are expected to work. The ideas should be selected based on the quality and every idea will have a prototype to be developed. Ideas should be organized and sorted according to do-ability. Prototyping does not mean creating lines of code to develop a smaller version of your bigger application; it can just be something with the least minimum resources to test if that will be used by the customer. A minimum viable product is developed in this stage. Prototypes are taken to the test stage for testing customer’s motivation. An example of prototyping is given below.

There are students who want to share their skills with other students at the university and they need a platform to share their work as well as record it in an organized manner. Currently, they exchange links through Whatsapp or use Skype to share their skills with other students who are interested and use Google docs to record their work for future use. But if we offer them an application that allows them to share their work as well as store it then we will observe that the students will use this application and also share it with other friends at other universities.

For example, taking the previous student skills example, a concierge MVP can be - being on Skype with the students recording what they speak, organizing the links and documents they share, and posting it in a group email. This does not require any line of code but we get the results. We see if the students like what we did, if doing this made their work easier or not. Here a person does the job of an application. It is the cheapest prototype with minimum and available resources. The idea in prototyping phase is to avoid developing waste applications that nobody would use and to test ideas early in the process.

Types of Prototyping

Prototyping is generally divided into two types [42]. Low fidelity prototyping and high-fidelity prototyping. Low fidelity prototypes involve those prototypes, which are developed using very basic, inexpensive and minimum resources. Storyboarding is an example of low fidelity prototype. High fidelity prototypes are the prototypes, which look closer to the finished product but are not finished product itself. Example, a 3D view of a house under construction is a high-fidelity prototype. It can also be a very initial version of software with a few lines of code. It is always advisable to build from the perspective of the user. The easiest and the most used type of prototyping is low fidelity.

KANO Model

The KANO model was developed at the Tokyo Rika University in 1984 [43]. It was developed for achieving customer satisfaction. It divides the software attributes into several categories for delighting and exciting the customer. It classifies the customer requirements into 5 categories – indifferent attributes, reverse attributes, delight attributes, basic attributes, and performance attributes. The KANO model can be used while in the prototype stage to achieve customer satisfaction. Reverse attributes are those, which are supposed to be avoided duirng the development of the software. Adding these attributes will result in customer dissatisfaction. Example, adding a feature to store customer credit card details even when the app has nothing to do with buying or selling goods. Indifferent attributes are those, which add no value to the software. Adding these attributes will have no effect on the user or software. Example, coloring of buttons. Any color or any most commonly used color can be used. it does not matter to the customer unless specified by him. Basic attributes are those, which should be on the application. Not having them will dissatisfy the customer. Example, having a payment option for a shopping application. Customer might not mention these attributes but they are very basic and must be included. Performance attributes are those for which the customer is paying us. These attributes will result in immense customer satisfaction and cannot be missed. Example, HD video on Netflix. Delight attributes are those, which cause customer satisfaction when implemented but does not matter if they are not implemented. Example, giving free t-shirt on first purchase with the application.

The KANO model can play a role while prototyping. Its attributes can be used to see what matters to the customer and what can create delight for the customer. Keeping those attributes in mind, a prototype can be developed.

MVP

The team needs to be creative on how they can get the results with fastest and lowest possible efforts and cost. Sometimes the team would not even need to build anything, not even a single line of code. The results can either prove or disprove the idea. This is called as an MVP (Minimum Viable Product). An MVP is not the first version of the product; instead it is a 'proxy' of the product. The whole point of MVP is to avoid waste so that time or effort is not wasted in building something, which no one wants or something which nobody is motivated to use. Below are three different types of patterns that the MVP uses

- The concierge MVP – in concierge MVP, an experience for the user is created. It does not require any line of code but results are guaranteed.
- The wizard of Oz MVP – in this, customer experience is shown or faked.
- The sales MVP – the idea here is to see if the customer buys product before even the team actually has it. For example, using a response rate or sign up from the emails to see if the customer is interested. Unfortunately, this MVP does not give much depth on the motivation of the customer.

Few big companies like Dropbox and Zappos have used Lean and MVP concept [44]. People in those times were still not used to buying products online. An observed problem scenario by Zappos was their persona could not find the shoes they wanted at the local shoe store, they did not have the time to go out and explore, and were frustrated. An alternative that their persona used was to wait until he has time to go to a bigger market or mail the order. The value hypothesis that Zappos came up with was - make shoes available online so that people do not have to come to the store and make their user experience better so that they come back again whenever they needed new shoes. For this, Zappos made an agreement with local shoe store, put up pictures online on a very basic site and waited to see the response. As soon as they got an order, they went down to the store, bought it, and shipped it to that person. The point here was to see if people would really used what they created, if they really needed something like this, or is it something valuable to them.

Relation of Agile to Prototyping stage

Agile is a software development methodology, which is responsive to changing customer needs. Customer needs are never the same. The market keeps changing every other day. one of the biggest reason Agile methodology was chosen over traditional methodology was because it was responsive to changing customer requirements. The prototype stage of design thinking is also responsive to the same. While prototyping, if the customer decides to change the requirements then those changes are welcomed. This can be done at any stage of design thinking.

Agile prefers working software over comprehensive documentation. By developing MVPs in the prototype stage, the team has something to show the customer, other than just mere documentation. It may be something, which is not valuable to the customer but it probably will result in customer satisfaction later in the development process. Through the prototype stage, we try to understand if the customer is really going to use the proposed solution over the present alternative. Agile practices like KANO can be used to attain customer satisfaction in design thinking framework. Hence, both AgileAgile and prototype stage of design thinking support the same ideology.

Suggested game/ activity to teach students about prototype stage of design thinking

Spaghetti and marshmallow exercise

This exercise is adapted from Stanford d.school [24]. In this exercise, students have to develop a tower using very basic and simple resources – spaghetti, tape, string and a marshmallow. The estimated time required to complete this task is 10 minutes. This activity is a group activity where each group should use a minimum amount of spaghettis and one marshmallow to make their tower. After 10 minutes, the teacher should examine which team used the least number of spaghettis and also has the tallest tower. Debriefing after the exercise can be done by asking students what they found difficult in the exercise and how they could relate this to prototype stage of design thinking.

Unit VI

Test and Design Sprints in 5 Days

Introduction

This stage is the last stage of design thinking. By this point, it must be clear that design thinking framework does not need to be followed as it is. It is an interactive and incremental process. If there is a need to iterate the prototype stage then the team can freely do that. Same goes with every other stage. All the prototypes that are being developed in the prototype stage are tested in the test stage of design thinking. It is best if the team tries to get at least two to three prototypes to test stage. If after the test stage, the team discovers that the user does not like the proposed solution then they can go back to carry out the process again and determine where they went wrong.

Usability testing

It is desirable that the prototype is tested in a time boxed session and tested it in the environment in which the user will actually use the prototype [4]. There is more in the testing phase than just getting customer feedback on a survey form. The team needs to be present while the prototype is being tested as this will help the team members to observe the reactions of the user. Getting a simple feedback from the users normally does not yield actual results. The customer is smart and knows that if he or she answers a NO in the feedback then they will have to answer several other questions. This does not mean that one should not use feedbacks. Feedbacks can work great if clumped with observation of reactions. Hence it is good to have the customer and observe their reactions. Here, the team is not just testing the prototype but they are also testing if the user is motivated enough to use the proposed solution. There are three stages to usability testing. Those are –

- Exploratory test – this is the first stage of performing the usability test. Here, the team tries to figure out which testing approach fits best for a particular application
- Assessment test – during this phase, the team creates a rough functional space of the selected approach
- Validation test – this phase is where the team bring in the users, times the process and records their reaction.

Customer experience with the prototype along with its feedback should be recorded. Getting negative feedback from the customer is a good way to learn more about them and give your best again.

Design thinking sprint in 5 days

Now that all the concepts are clear, how are these concepts going to be used in a design thinking sprint of 5 days? Starting with an unknown customer on Monday, the team can finish a design thinking sprint on Friday with a tested prototype in hand. These sprints do not need to be followed as it is, they can be customized according to the needs of the team or the software being build. A suggested way to carry out these design thinking sprints is given below:

Day 0 – preparation day

Monday – empathize stage

Tuesday – define stage

Wednesday – ideate stage

Thursday – prototype stage

Friday– test stage

Day 0 – this day is the preparation day. By this day, the team should have a list of customers who are ready to be interviewed. It is preferable to have those customers who are going to use the developed software directly. It is also preferable to have more than one customer to gain a better insight. By this day, the team must know all their concepts better. A checklist can be prepared by the team to see if they are on track. An example of checklist is provided in Table 4.6

Table 4.6

Example Checklist for Day 0

DONE?	**ITEMS**
	Slides?
	Room?
	Sprint guide?
	Supplies?
	Screen sharing?
	Subjects?
	Google Docs?
	Emails to Subjects?

Slides having all the concepts well written can help the team to go back and brush up when needed. As these sprints do not only have the design team working, but will also have different members from other teams. A room with comfortable environment can help the team to be motivated. A sprint guide having all the details about the upcoming sprints can also be helpful. Screen sharing is optional. The team might need screen sharing while they are storyboarding or developing high-level fidelity prototypes. Subjects are the customers. As mentioned earlier, the team should have their customers ready and those customers should be emailed about agreed meeting time. Google Docs are needed to document the entire sprint and to reflect back if needed.

Monday – today is the first day of the sprint. Before starting this sprint, the team should have the list of customers ready. They should have carried out preparation day 0. Today, the team strives to discover the customers by interviewing them, creating personas, customer journey maps, and empathy maps. Interviews with the customer should be timed. It is suggested to prepare the interview questions before hand and to do a mock interview with the team so that they can discuss and come up with new questions that would gain more insight about the customers and their needs. During the interview, the team records the answers and the reactions of the customer. After the interview, the team sits together to draft customer personas. Personas can be drafted as a team or by every individual in the team and converge it later. The next step is to prepare empathy maps and customer journey maps to unwind what was observed and learned about the customer. Every activity can be timed as shown in Table 4.7.

Table 4.7

Example of Timeboxed Activities for Empathize Stage

MIN	**ACTIVITY**
90	Revision of concepts
15	Draft personas in pairs, individually or as a team
20	Converge personas
10	Discussion on drafting personas
15	Create Empathy map
10	Discussion on empathy map
15	Create customer journey map
10	Discussion on customer journey map

Tuesday – by today, the team knows their customer well. The team asks 4 W's and 5 Y's questions to cross check their understanding of the customer. Now it is time to define the customer needs in a formal format so that the team can comes up with good ideas to solve their problems. The team writes all the problems in a user story format and creates a POV (point of view) of customer needs and insights. User stories can be prioritized using MoSCoW and see if good user stories have been written using the INVEST acronym. Activities can be time boxed as shown below in Table 4.8.

Table 4.8

Example of Timeboxed Activities for Define Stage

MIN	ACTIVITY
10	Revision of concepts / 4 w’s and 5 y’s
10	Draft user stories
20	Discussion and converge
5	INVEST acronym
20	MoSCow to prioritize user stories
20	Create POVs
20	Discussions, review, compare and update

Wednesday – since the team knows the customers and has their needs are defined in a proper format, now is the time to come up with ideas that could solve their existing problems. The team gets together to brainstorm the ideas and create mind maps and storyboards of the ideas selected. Following in Table 4.9 is an example of how the activities can be time boxed.

Table 4.9

Example of Timeboxed Activities for Ideate Stage

MIN	**ACTIVITY**
10	Revision of concepts
15	Brainstorm ideas
20	Discus ideas
20	Rank ideas
20	Crate mind maps
20	Discuss and converge
20	Create storyboards
90	Discussion

Thursday – the team has top ideas on the list and now is the fun stage of prototyping these ideas. Ideas from previous day are prototyped to test the customers motivation. An example of time boxed sessions for this day is shown in Table 4.10.

Table 4.10

Example of Timeboxed Activities for Prototype Stage

MIN	**ACTIVITY**
10	Revision of concepts
15	Brief discussion on idea 1
20	prototype idea 1
15	Brief discussion on idea 2
20	Prototype idea 2
15	Brief discussion on idea 3
20	Prototype idea 3
90	Discussions on how to carry out next day

Friday – today is the final day of the design thinking sprint. The team members meet the customer again to test their prototypes and also to test if the customer is motivated to use their proposed solution over the present alternative. As mentioned earlier, it is desirable to allow the customer to test in the environment they will be using the actual software. Remember to time box the tests. If the ideas work well with the customer then the team moves to building of software else the team goes back to find where they went wrong.

Relation of Agile to test phase of design thinking

Agile has methodologies like Test Driven Development (TDD) whose idea is about development along with testing. In traditional methodologies, testing of software was done only after the entire software development was completed or sometimes the software were not even tested. Testing in Agile is not a phase that has to be completed instead it is a continuous process that goes along with the software development. Similarly, each prototype developed is tested in the test stage of design thinking. Testing is an integral part both in Agile and in design thinking and cannot be ignored. Testing of prototypes also encourages the team to come back with better ideas and prototypes. It encourages them to focus on little details, which would have otherwise been ignored.

Pedagogy using Agile Principle and Manifestos

In this section, different Agile principles and manifestos are used for teaching. These include working agreement, estimation, daily stand up meetings, backlogs, sprint review meeting, and pair programming. These principles were used to teach data science to software engineering students. Details of how these principles were used is provided below.

Working Agreement

Working agreements are one of the core concepts of Agile [45]. They help to create transparency and accountability among the team members along with negotiating different issues that they have. It helps the team to have a clear conversation of things they think are inappropriate. Questions that a team must ask are – who are we? What is our team name? What are our values? How do we handle our conflicts? How do we run our meetings? How will we manage our work? How will we know if we are successful? [14]

In a classroom environment, a working agreement can be used for the same reasons as it is used in Agile methodology. In the data science class, the instructor introduced the working agreement concept on the first day of class. Several questions were asked so that the students could discuss and come to one agreement. Following is the list of questions that were asked:

1. What time during the class should the stand meeting be held?
2. How assignments are going to be done – like in teams, individually or pairs?

3. How will they help another student if he or she is unable to attend the class?

After discussion, the students came to an agreement that the daily stand up meetings would be held in the last 15 minutes of the class. They agreed that the assignments would be done individually, but the activities in class would be paired. The students who missed their classes would be briefed by other students on call or by using Skype. During these discussions, the students got along with each other. This acted as an icebreaker for most of the students as they did not know each other. Apart from this, it helped the students create a sense of transparency and accountability among each other.

Estimation

Effort estimation is one of the important activities for successfully planning and executing a project. Research shows that almost 66 percent of projects run over budget and 33 percent run over the schedule [46]. This often results in great loss to the software companies in the form of time, effort, money, and resources. Underestimations result in work overload, budget overruns, schedule overruns, and affect the quality, whereas overestimation results in wastage of resources, time, and money. There are several effort estimation techniques that exist, namely, planning poker, expert judgments, story points, functional points, and the Delphi method. Out of these, expert judgment and planning poker are most commonly used [47]. In expert judgment, an estimate is provided by an expert in the particular domain. The expert uses story points for estimation. Story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work.

At the start of planning poker, each developer is given a deck of cards, which has Fibonacci numbers (0, 1, 1, 2, 3, 5 and so on.) on them. These cards are usually prepared before the estimation session starts. The customer explains all the requirements of a project. The developers ask any questions they have regarding the requirements. After the

question-answer session, the developers estimate the complexity of each requirement. The developers show their cards to other team members. The highest estimate and the lowest estimate then explain the reason of their estimate to other team members. Estimation process is again repeated until the entire team reaches a consensus.
Carrying out an estimation activity in class will help the teacher know whether students find the course easy or difficult. Depending upon the estimations of students, the course difficulty level can be increased or decreased. In the data science class, the students wrote their estimates on cards and shared with other students. Students having highest and lowest estimates were asked the reason of their estimates.

Stand Up Meetings

Daily stand up meetings are a part of scrum methodology. In this, the team members have to meet each other in order to discuss their progress. They stand in a group and answer three questions – what did they do yesterday? what are they doing today? And what are the roadblocks they are facing [48]? This meeting is supposed to be no longer than 15 minutes. Each member of the team should answer the questions.
If a particular class is every day, then daily stand up meetings can be carried out as is, but if that class is once in a week then they can be done online and can be done every other day depending upon the availability of the students. This will help students to become accountable of their work. Stand up meetings will help the students to take some time out of their schedule and give it to that particular course. It will help them to find questions while reading the course contents and discuss it with other students. Transfer of knowledge can also take place while doing the stand-up meetings. Face-to-face students as well as online students can be a part of this meeting.

While doing the working agreement activity, the students in the data science class agreed that they would do their stand up meetings face-to-face every week during the last

15 minutes of the class. They discussed what they read last week, what are they going to read this week, and what are the questions that they have in the course content so far. The teacher also became a part of this meeting and acted as a facilitator. Complicated questions that other students could not answer were answered by the teacher. If the teacher noticed any student struggling to give their answers then the teacher could contact that student to further investigate the issue. This helps in creating positive interaction between the students and between the teacher and students.

Backlogs

Scrum has two artifacts, namely product backlog and sprint backlog [48]. The product backlog is a list of items that the customer wants to be completed throughout the lifetime of a project. These tasks are prioritized according to the business value of the customer. Sprint backlog is similar to product backlog but the items in sprint backlog must be completed by the end of the sprint. Sprints usually are 1-2 weeks long. Product backlog comprises of sprint backlog [48]. If any item is not completed in the previous sprint then it is taken as a priority in the next sprint.

In a class environment, depending upon the class, the sprint length can vary. Course contents can be divided into items that are to be completed in one class, similar to sprint backlog (to be completed in one sprint). The entire course can be treated as a product backlog. Items that are not completed in the present class can be taken as a priority in the next class.

In the data science class, the sprint was one week long. As mentioned, items not completed in previous class were taken as a priority in the next class. Usage of backlogs in class environment helped the students as well as the teacher stay goal oriented. It motivated them and helped them with time management.

Sprint Review Meetings

A sprint review meeting is one of the four scrum ceremonies (scrum ceremonies include daily scrum meeting, sprint review meeting, sprint planning meeting, and sprint retrospective) [48]. This ceremony takes place at the end of each sprint. This meeting is conducted to reflect back and gather feedback on the previous sprint. Mistakes are learned from the previous sprint and improvements for the next sprint are discussed. This meeting is insightful and reflective.

Sprint review meetings in a class environment help teachers make revisions to the course content if needed. It also helps the teacher understand whether or not the students understand and enjoy the course. These reviews help increase interaction in the class. Sprint review meetings were held every other week in the data science class. They were held for 15 minutes at the end of class.

Pair Programming

Pair programming is a technique in extreme programming methodology where two programmers work sitting side by side in front of one computer system to write code. There is a driver and a navigator where the driver is the one responsible for writing the code and navigator is the one who sits beside the driver watching the code for syntax errors [49]. The programmers can switch roles after some time. There are several advantages of using pair programming such as better code, better satisfaction, and saved time.

Using pair programming in a classroom environment will encourage communication. Students who do not interact with each other might end up communicating with each other. Working together as a team will help students to build confidence and will inculcate team management skills. If one student is an expert in a particular topic and other student is a novice then this pair programming activity will really help the novice student. Pair programming activity will help the students to develop collaboration skills. In the data science class, as decided in the working agreement activity, all the activities in class were done in pairs to demonstrate pair programming. This pedagogy has been tested in the Data Science class and results are presented in the next section.

CHAPTER V:
RESULTS

In this section, results of Feature Driven Development are presented. Then e-learning application called Flashcards is introduced, which was developed using the Agile Design course. Later, results of Agile pedagogy are presented.

Feature Driven Scrum Validation

For evaluating the Feature Driven Development methodology, a study group of students in Masters of Software Engineering was formed to learn Functional C# programming language. The reason for making an online group was to include online students who were from different states and different time zones. All of the students were added on a business platform named “Slack.” All the meetings were conducted online and there were eight participants. A pre-assessment sheet was asked to be completed by the students in order to track their progress before and after the validation. The questions and average responses of the pre-assessment are provided in Table 5.1. The scale of the assessment was 1-5 or freeform answers. The Likert scale (rubrics) was defined as follows: 1 being not at all effective, 2 being not effective, 3 been somewhat effective, 4 being effective, and 5 being highly effective.

Table 5.1

Pre-assessment Sheet Questions and Averaged Answers

Questions	**Scale**	**Responses**
What is the level of your experience in C#?	1-5	2
Have you been in a study group before?	Yes/No	No
How good was your previous experience in the study group? (if applicable)	1-5	3
How familiar are you with Agile methodology?	1-5	1

The first meeting served as an icebreaker for the students. It started with the introduction of each student. Then, an overall draft plan was developed, taking into account the topics they wanted to finish by the end of this study group.

Next was the planning meeting, which was conducted the next day. A sprint length of three days was decided. The students came up with topics they wanted to finish by the end of these three days (one sprint). These topics were then divided into per day topics. The daily scrum meetings were conducted online using Slack. The students did not speak to each other but instead posted their responses to the questions online. If any student hit an obstacle while studying, then other team members came forward to help.

After one sprint (three days), a sprint review meeting was conducted. Students discussed whether they achieved the desired goals or not. They also discussed their concerns and suggestions during this meeting. They considered the topics that were not completed in this sprint as backlogs and made these backlogs as a priority in their next sprint.

This online study group was discontinued after two sprints. Since the study group had students from different regions and different time zones it was difficult for them to come online at the same time for meetings. Coordinating online interactions eventually became difficult. The students completed two sprints and realized that the study group would have been successful it was conducted face-to-face. Since the study was discontinued after two sprints, the end of study retrospective could not be conducted. One external reason that was identified was that the students had their midterm examinations when the study group was conducted. Hence, the students were occupied with other commitments. However, the feedback from the students showed that Feature Driven Scrum helped them to become organized, helped them to divide big goals into smaller achievable goals, and instilled confidence in them.

Feature Driven Scrum was also validated in an educational workspace environment. It was validated in the university setting by the Course Development and Support Team. The work of this team is to provide quality assurance for courses that are taught online at that university. Before the introduction of Agile at their office, they followed the traditional way of working. In the traditional way of working, they evaluated the course all at once and then sent it back to the professor for changes. They used red and yellow colored slips to determine mandatory changes and optional changes respectively. They had few meetings every month and followed a plan than responding to changes.

There are seven team members in the Course Development and Support Team. After the introduction of Agile in their office in 2017, they started having several meetings and started doing their work in sprints. Their feedbacks were collected in the form of a survey, which is presented in Table 5.2 below. The scale (rubric) was defined as follows: 1 being not at all effective, 2 being not effective, 3 been somewhat effective, 4 being effective, and 5 being highly effective.

Table 5.2

Survey Questions and Answers

Questions	**Scale**	**Responses**
How was the performance of the team before using Agile?	1-5	3
How was the performance of the team after implementation of Agile?	1-5	4
How difficult was it to convert from traditional way of working to Agile?	1-5	4
How effectively can you train an employee with this process?	1-5	4
How did setting a common goal for the team effect the team progress?	freeform	The time spent working on a common goal great paid off in terms of the long-term gains associated with the new methodology.
How often do you have meetings?	freeform	2-3/month
Do you have daily scrum meetings?	freeform	During peak periods of development (right before QA cycles), we will go to non-daily (but regular) SCRUM meetings.
If yes, how effective were the daily scrum meetings?	1-5	4
What did you like about Agile in the workplace	freeform	It gives us flexibility in terms of meeting a wide variety of different faculty preferences in course design. It also makes our design process more responsive to a vastly changing workplace landscape
What did you not like about Agile?	freeform	The team tends to fall back into older model-based practices, so we do have to plan for refreshers/retraining of the staff
What changes would you want?	freeform	None at this time

The results from the survey indicate that Feature Driven Scrum works well in an office environment.

Agile Design Course Validation

For the validation of the Agile Design course, an e-learning application called Flashcards was developed using C# and XML on Xamarin platform. This application was developed using the principles of Agile Design course. All stages of Agile Design were completed in five days starting from Monday to Friday.

Empathize

Designing of the application started from the empathize stage. There was one customer who was a Masters student at a university. First, the interview with the customer was conducted followed by the creation of a persona and an empathy map. The interview was conducted on a video call. This was a 20 minutes interview, which had the following questions. 4 W's and 5 Y's were used to create these interview questions.

- Which university are you studying in?
- What is your major?
- Why did you choose this university?
- Why did you choose this major?
- What are the top three things you like about this major and your university?
- What are the top three things you dislike about this major and your university?
- Which course do you like and dislike?
- What are the top five reasons you dislike that course?
- How do you try to cope up with the difficult courses?
- What do you think would make those courses fun?
- What do you do when you are not studying?
- How many hours do you study?

According to the answers of this interview, a persona was created which is shown in Figure 5.1.

Atchyutha, the student

Atchyutha is a graduate student in software engineering program at a university. She aspires to become a successful Java developer. She loves to listen to music and is a huge fan of Selena Gomez. She calls herself a “lazy perfectionist” because she is a lazy person, but when it comes to finishing something, she hits it hard. She has been a bright student since her childhood but she found her first semester at the university a little difficult. She is good at learning and understanding logic but finds theoretical courses difficult. She wants to make theory learning easy and fun.

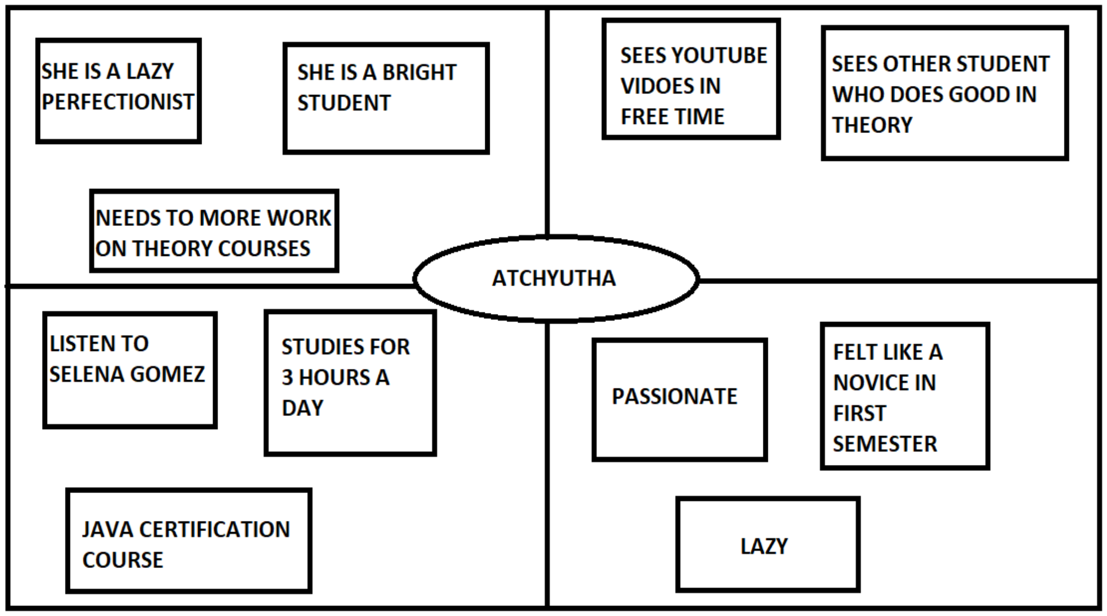

SHE IS A LAZY PERFECTIONIST
SHE IS A BRIGHT STUDENT
NEEDS TO MORE WORK ON THEORY COURSES
SEES YOUTUBE VIDOES IN FREE TIME
SEES OTHER STUDENT WHO DOES GOOD IN THEORY
ATCHYUTHA
LISTEN TO SELENA GOMEZ
STUDIES FOR 3 HOURS A DAY
JAVA CERTIFICATION COURSE
PASSIONATE
FELT LIKE A NOVICE IN FIRST SEMESTER
LAZY

Figure 5.1
Empathy Map for Flashcard Application

Define

For defining the problem, user stories and child stories were created. Following are the user stories that were written using the INVEST acronym and these user stories were sorted using MoSCoW.

As a student, I want to learn my theory subjects as well as enjoy them so that I can get good grades in them.

This user story was broken down into child stories and test cases were written for them shown in Table 5.3.

Table 5.3

User Stories and Test Cases for Flashcard Application

Child stories	Test cases
As a student, I want to create my own questions from the course material so that I can learn from them	1. Make sure add button works 2. Make sure new questions can be added on clicking the add button
As a student, I want to write answers to those questions I have created so that I can learn from them	1. Make sure answers can be added on clicking the add button
As a student, I want to hide the answers behind the questions so that I can test myself	1. Make sure the card flips when tapped once 2. Make sure the questions are added at the front of the card 3. Make sure the questions are added at the back of the card
As a student, I want to save my questions and answers so that I can get back to them whenever I want to	1. Make sure the cards are saved
As a student, I want to edit my previously saved questions so that I can learn right content	1. Make sure there is an edit button 2. Make sure the questions can be edited when the edit button is clicked
As a student, I want to delete my questions so that I can remove irrelevant questions	1. Make sure there is a delete button 2. Make sure the card can be deleted when the delete button is clicked
As a student, I want to edit my previously saved answers so that I can learn right content	1. Make sure the answers can be edited when the edit button is clicked
As a student, I want to delete my answers so that I can remove irrelevant questions	1. Make sure the card can be deleted when the delete button is clicked
As a student, I want to navigate through my previous and next questions so that I can learn better	1. Make sure there is a next button 2. Make sure there is a previous button 3. Make sure the next card appears when the next button is clicked 4. Make sure the previous card appears when the previous button is clicked

Ideate

After defining the problem, storyboarding was done. The storyboard and notes are shown below. The before situation is illustrated in Figure 5.2 and the after situation is illustrated in Figure 5.3. Notes of before and after storyboard are in shown Tables 5.4 and 5.5 respectively.

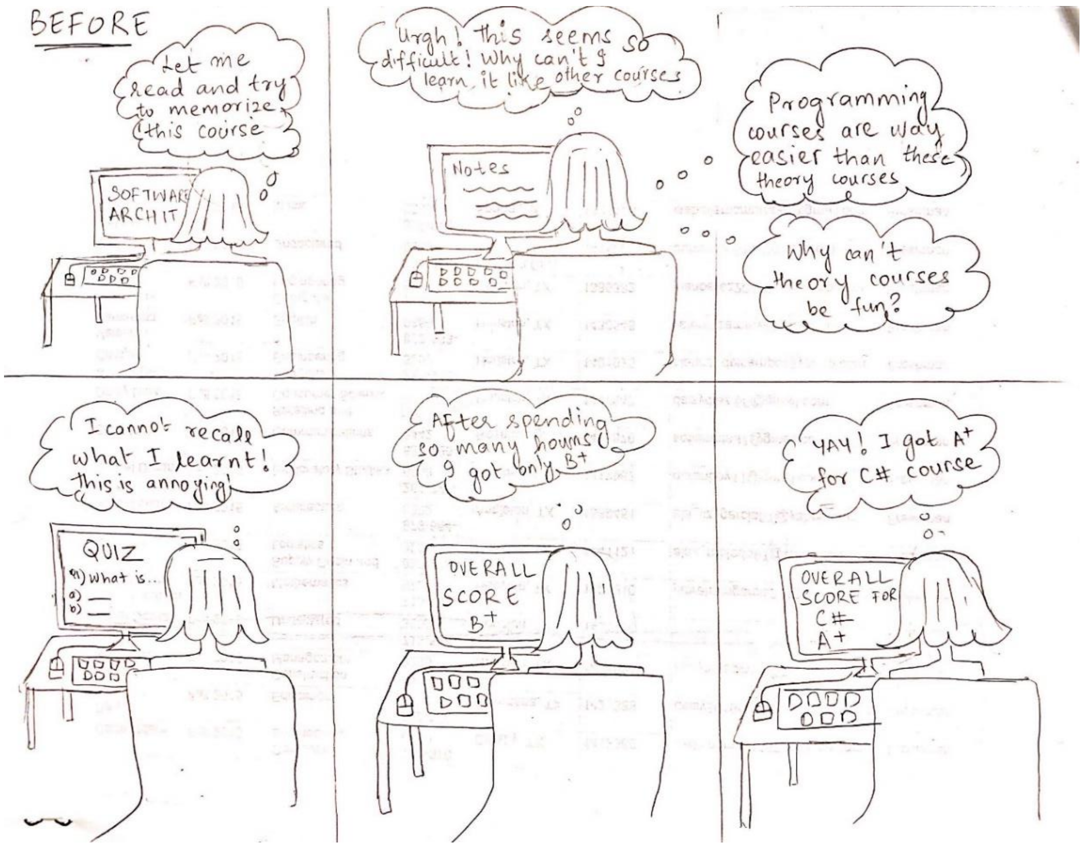

BEFORE
Let me read and try to memorize this course
SOFTWARE ARCHIT
Urgh! this seems so difficult! Why can't I learn it like other courses
Notes
Programming courses are way easier than these theory courses
Why can't theory courses be fun?
I cannot recall what I learnt! this is annoying!
QUIZ
Q) What is..
a)
b)
After spending so many hours I got only B+
OVERALL SCORE B+
YAY! I got A+ for C# course
OVERALL SCORE FOR C# A+

Figure 5.2
Storyboard Showing Before Situation for Flashcard Application

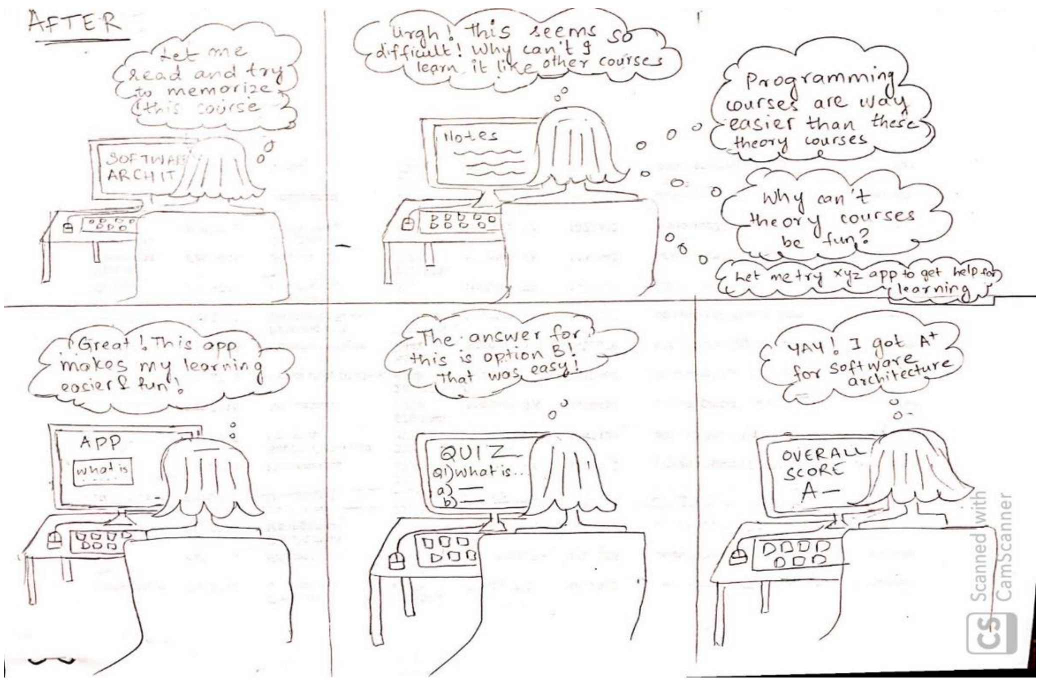
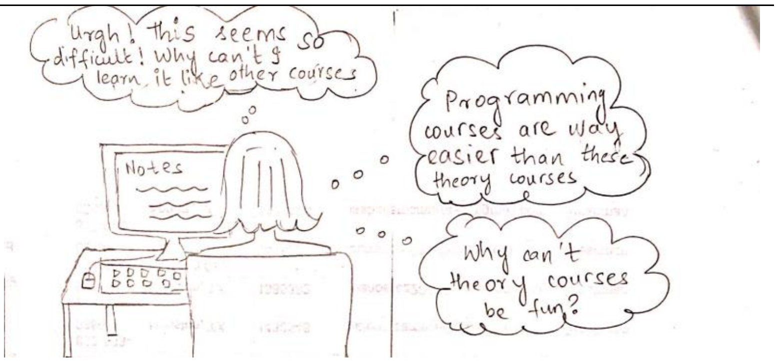
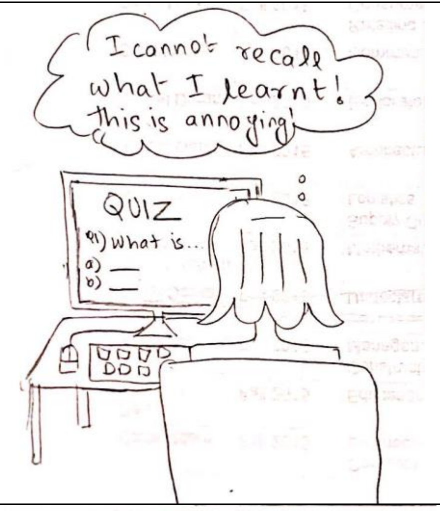
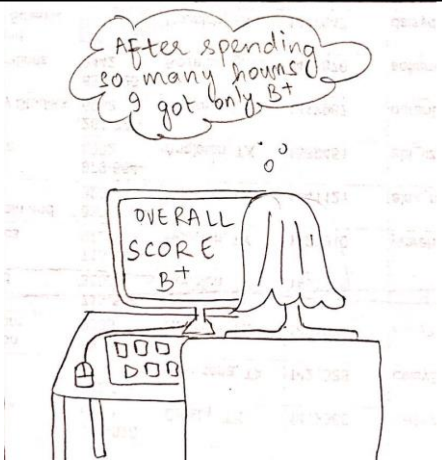

AFTER
Let me read and try to memorize this course
SOFTWARE ARCHIT
Urgh! this seems so difficult! Why can't I learn it like other courses
Notes
Programming courses are way easier than these theory courses
Why can't theory courses be fun?
Let me try xyz app to get help for learning
Great! This app makes my learning easier & fun!
APP
what is
The answer for this is option B! That was easy!
QUIZ
q1) what is...
a)
b)
YAY! I got A+ for Software architecture
OVERALL SCORE
A—

Figure 5.3
Storyboard Showing After Situation for Flashcard Application

Table 5.4

Notes for Before Situation

Notes	**Situation**
Atchyutha is a budding Software Engineer. She is trying to read and memorize Software Architecture course because she has her quizzes coming up this week.	Let me read and try to memorize this course SOFTWARE ARCHIT
She finds the theory courses difficult and boring. She instead finds the logic thinking courses much easier and fun. She struggles to memorize the course content.	 Urgh! this seems so difficult! Why can't I learn it like other courses Notes Programming courses are way easier than these theory courses Why can't theory courses be fun?

Notes	Situation
Even after studying for the quiz, she cannot recall the answers while taking the quiz. She is annoyed and frustrated.	 I cannot recall what I learnt! this is annoying! QUIZ q) what is... a) — b) —
After few days, she gets her overall score for Software Architecture course. She gets a B+ grade. She is sad and frustrated on seeing that grade because she had spent many hours behind this course.	 After spending so many hours I got only B+ OVERALL SCORE B+

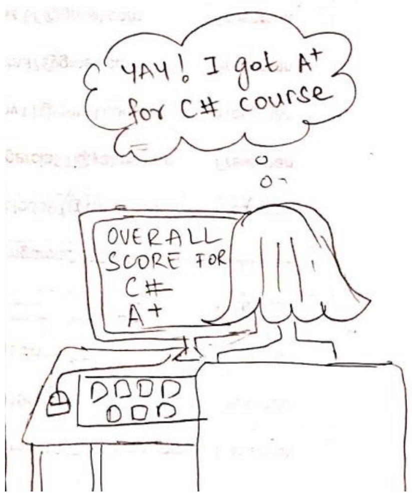
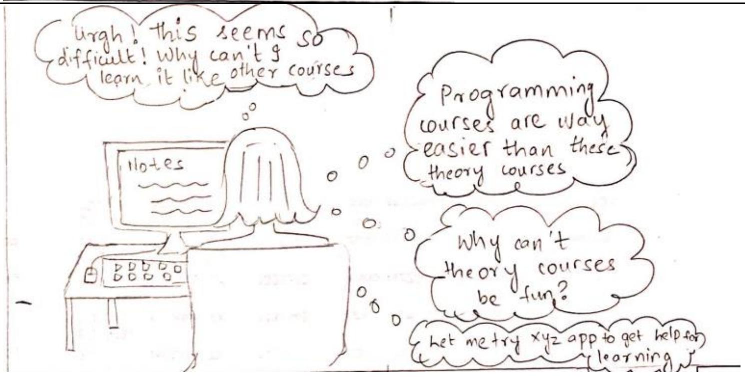
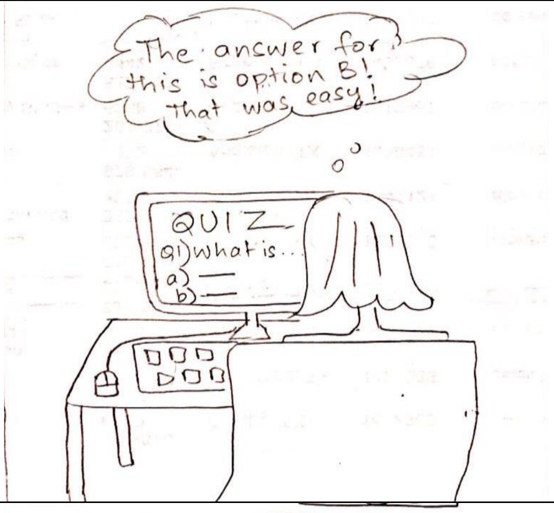
Notes	Situation
She gets her C# programming course score the same day and gets a A+. She is a bright student but needs help with theory courses. She does not want her overall GPA to drop down because of theory courses.	

Table 5.5

Notes for After Situation

Notes	Situation
Atchyutha is a budding Software Engineer. She is trying to read and memorize Software Architecture course because she has her quizzes coming up this week.	Let me read and try to memorize this course SOFTWARE ARCHIT
She finds the theory courses difficult and boring. She instead finds the logic thinking courses much easier and fun. She decided to use the "XYZ app" to get make learning easy and fun	 Urgh! this seems so difficult! Why can't I learn it like other courses Programming courses are way easier than these theory courses Why can't theory courses be fun? Let me try xyz app to get help for learning Notes
She sees that she is able to learn faster and this is also fun. The application helps her to prepare for her upcoming quizzes. She feels better.	Great! This app makes my learning easier & fun! APP what is

Notes	**Situation**
While taking her quiz for Software Architecture course, she is now able to remember what she learned. She finds answering those questions easier.	 The answer for this is option B! That was easy! QUIZ Q1) what is... a) b)
She gets an A- in Software Architeture course and this helps her maintain her GPA. She feels learning theory courses isn't that difficult or boring after all.	YAY! I got A+ for software architecture OVERALL SCORE A-

After story boarding, several ideas were written down during the brainstorming session. The top three ideas that were decided to be prototyped are as follows:

1. Write questions and answers for every unit and email the student
2. Use paper cards and let the student write questions in the front and answers on the back of the card
3. Use pictures related to the course content to help the student memorize

Prototype and Test

All of the above-mentioned ideas were taken to the prototype and testing stage. The second idea was the most liked by the customer. A deck of blank cards was created using paper. These cards were given to the customer so that she can write her questions on the front and answers on the back of the card while reading the course content. She was asked to use these cards for a day while learning her theory subject. She was able to add more cards and discard cards whenever needed. She was also able to edit the questions and answers on the cards. She liked what was prototyped. This did not require even one line of code and was done using readily available material.

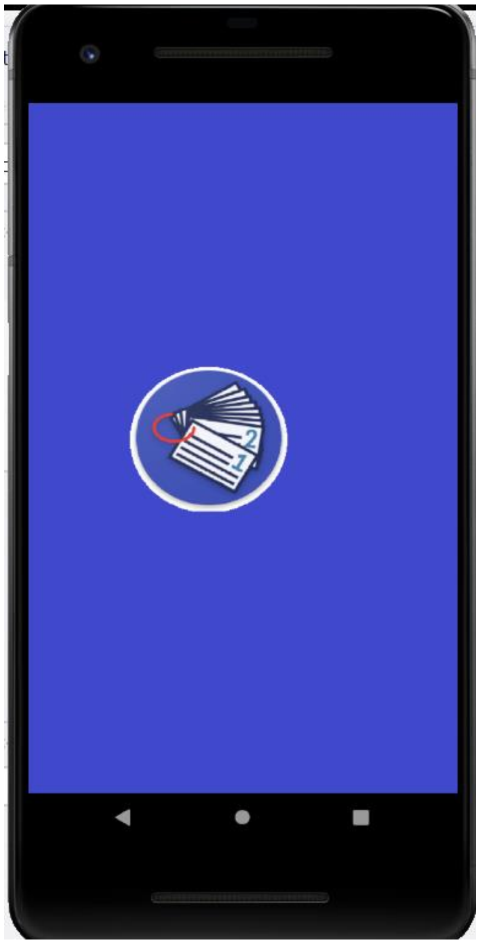


Figure 5.4
Splash Screen for Flashcard Application

Since this idea was most liked by the customer, an e-learning application having flip cards was created. This application lets the user write questions on the front of the card and answers on the back of the card. The user can add, delete, and edit cards whenever wanted. The card flips when tapped. Next and previous cards can be accessed when the next and previous buttons are clicked. The cards are saved automatically whenever a new card is created using the '+' symbol on the lower right of the application. The screenshots of the application are given below. The code of this application is in Appendix. The first screen of the application is a splash screen with a logo as shown in Figure 5.4.

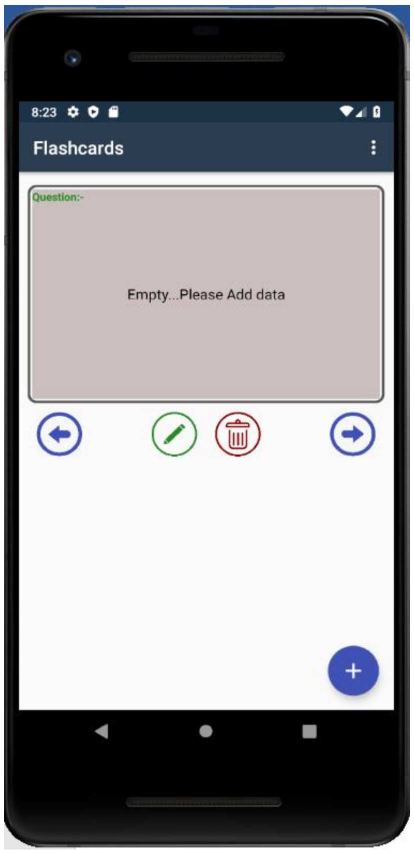


Figure 5.5
Empty Card

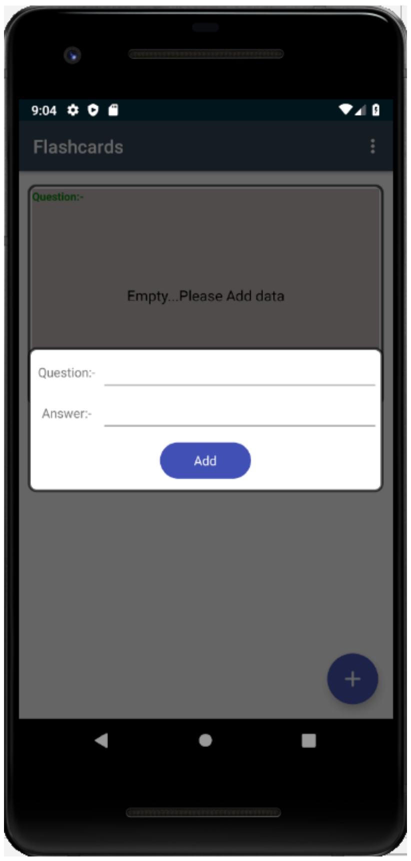


Figure 5.6
Adding First Card

Since the application is initially empty without any data on the card a message is displayed on the card which says, "Empty… Please add data" as shown in Figure 5.5. On clicking the "+" button shown on lower right, the user is asked to enter question and answer for the next card. This is shown in Figure 5.6.

The user then adds a question and answer for the card (Figure 5.7). Once the card is added a toast message is displayed which is shown in Figure 5.8. an answer added is shown in Figure 5.9.

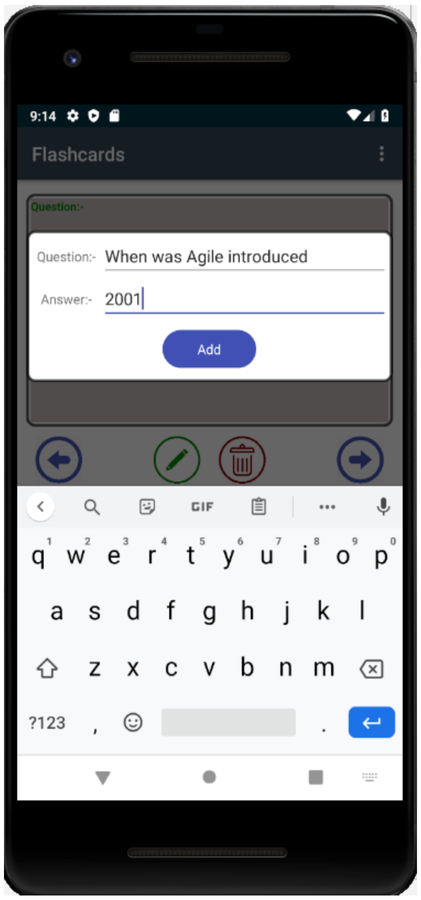


Figure 5.7
Adding Question and Answer

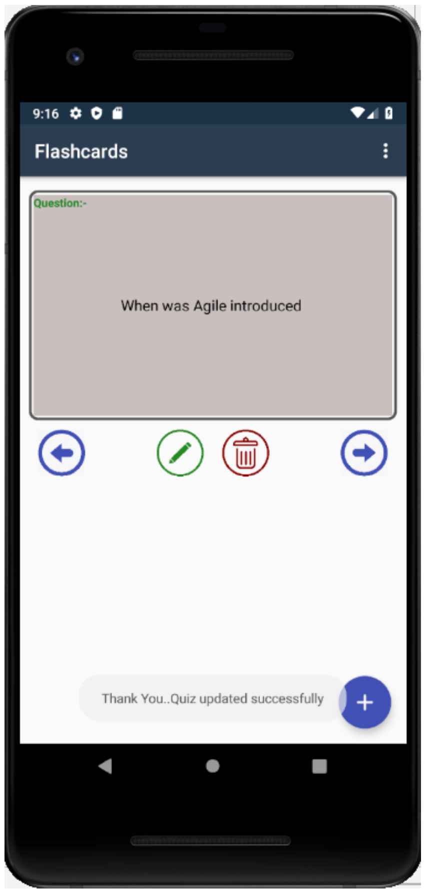

9:16
Flashcards
Question:-
When was Agile introduced
Thank You..Quiz updated successfully

Figure 5.8
Added First Question

A second card is added by clicking "+" sign. Questions are added in the front of the card and answers are added on the back of the card. This is shown in Figure 5.10. On clicking the edit button, both question and answer on the card can be edited (Figure 5.11). The edited card is shown in Figure 5.12.

On clicking delete button, an alert message appears (Figure 5.13). If the user clicks on "yes", the card is deleted (Figure 5.14) and a confirmation message appears else the user can cancel deletion of card.

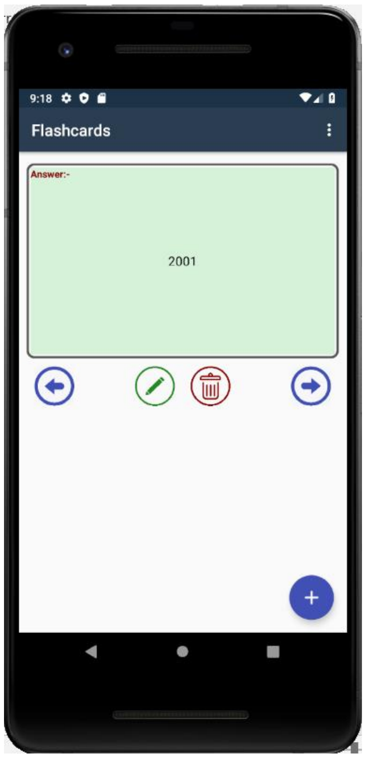


Figure 5.9
Added First Answer

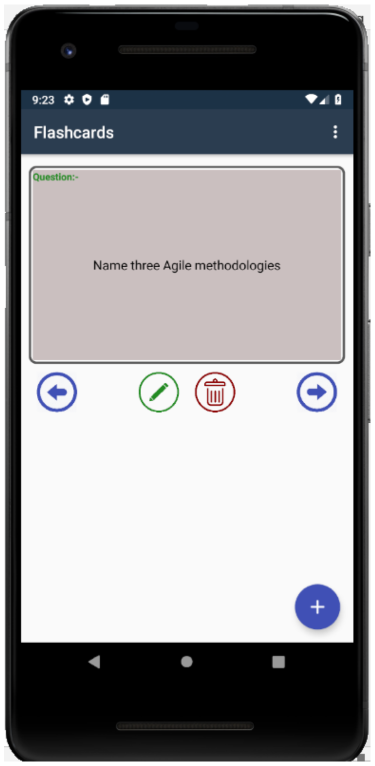


Figure 5.10
Second Card

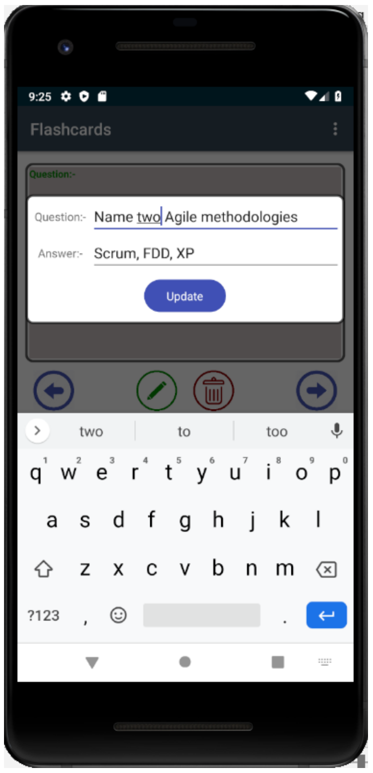


Figure 5.11
Editing Card

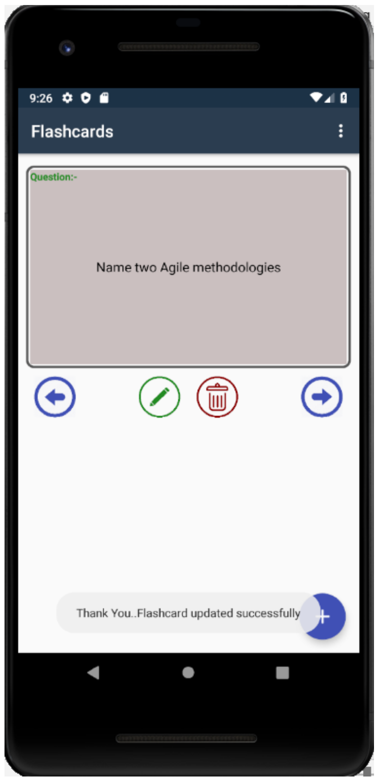

9:26
Flashcards
Question:-
Name two Agile methodologies
Thank You..Flashcard updated successfully

Figure 5.12
Edited Card

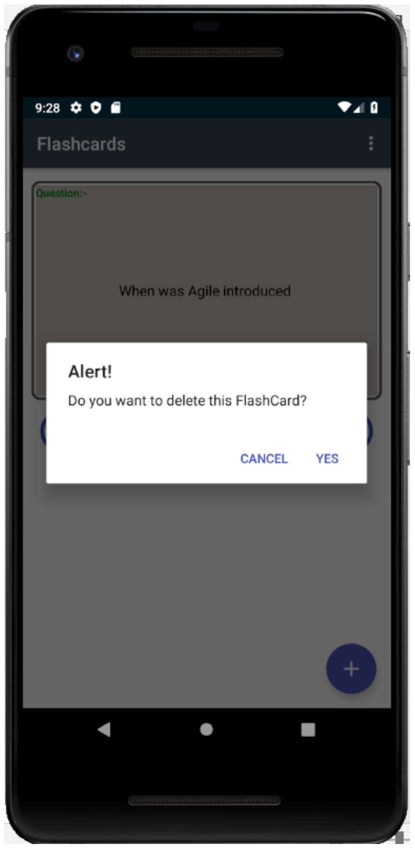

9:28
Flashcards
Question:-
When was Agile introduced
Alert!
Do you want to delete this FlashCard?
CANCEL
YES

Figure 5.13
Deleting Card

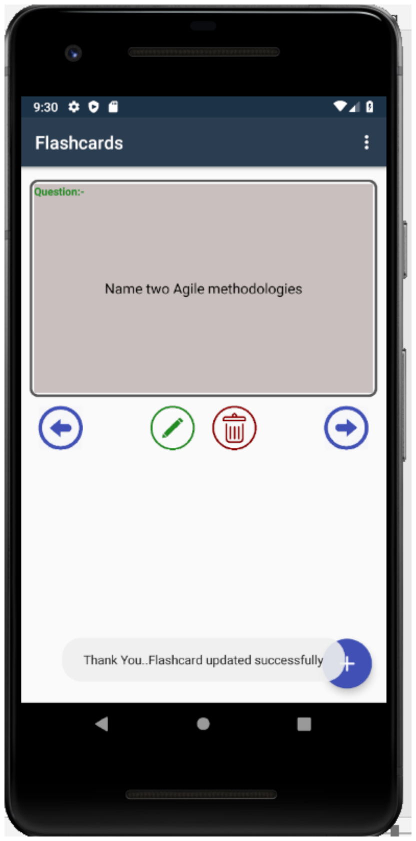

9:30
Flashcards
Question:-
Name two Agile methodologies
Thank You..Flashcard updated successfully

Figure 5.14
Card Deleted

Agile Pedagogy Validation

Validation of Agile pedagogy was done at the end of semester through an online survey using SurveyMonkey. This survey was completely anonymous. There were seven students who participated in the survey. The scale of this survey was 1-5 or freeform answers. The scale (rubric) was defined as follows: 1 being not at all effective, 2 being not effective, 3 been somewhat effective, 4 being effective, and 5 being highly effective. Following are the questions and the averaged answers in Table 5.6.

Table 5.6

Questions and Averaged Answers for Agile Pedagogy

Questions	**Averaged Answers**
Rate the Teaching Methodology of Data Science class	3.6
How effective did you find the first 15 Minutes of the class? This session included a review of the previous class and a question-answer session.	4.1
What are your feedbacks on daily stand up meetings in class and how do you think we can improve them?	Not effective. Probably writing on discussion board is more effective than actual stand up meetings.
Did you find paired activities better than those activities done by you alone?	Yes
Would you call this class as an Active Learning class or a Passive Learning class? Active learning is a form of learning in which the teacher strives to involve students in the learning process more directly. Passive learning is “teacher-directed learning” where the teacher is the leader and students only follow the instructions of the teacher without direct involvement.	Active learning
Apart from learning Data Science, did this class help you learn about the Agile Framework?	Yes
Mention one take-away from this class about Agile	Stand up meetings and paired programming

From the survey results, it can be concluded that the students liked the teaching methodology of the data science class. They found this class as an interactive and active learning class. They learned about Data Science as well as Agile in the same class. Students found the review meeting helpful and gave it a rating of 4.1 out of 5. Paired activities were liked by all seven students. They said that they would prefer working in pairs than working alone. However, most of the students did not find the daily stand up meetings effective. They felt that the daily stand up meetings could have been done on Blackboard rathan than in the class. While six students said they did not find the daily stand up meetings effective, one student said, “I personally liked the daily stand up meetings and it is very easy to get to know what are the things that we would do for the next week or the things that we had done last week.”

CHAPTER VI:
CONCLUSION AND FUTURE WORK

In this study, advantages and limitations of both planned and Agile software development methodologies have been discussed. To demonstrate thatAgile can be used in an educational as well as a workspace environment, Feature Driven Scrum have been introduced and validated. Next, Agile Design courses have been introduced to show that Agile and Design Thinking support the same principles and manifestos. This course has been validated by designing and developing an e-learning application called Flashcards. Later, an Agile pedagogy has been introduced and validated in a Data Science class at a university. This pedagogy was created using different Agile principles.

For future work, the Agile Design course could be taught as an actual university course and feedback could be collected from the students who use the course principles in their actual workspace. Agile pedagogy could be used to teach the Agile Design course. Additions or deletions could be made to the pedagogy based on the feedback. Flashcard application could be further developed.

REFERENCES

[1] University of Houston-Clear Lake,Houston, Texas, USA., M. Sameen Mirza, and S. Datta, “Strengths and Weakness of Traditional and Agile Processes - A Systematic Review,” *J. Softw.*, vol. 14, no. 5, pp. 209–219, May 2019, doi: 10.17706/jsw.14.5.209-219.

[2] P. Hohl *et al.*, “Back to the future: origins and directions of the ‘Agile Manifesto’ – views of the originators,” *J. Softw. Eng. Res. Dev.*, vol. 6, no. 1, pp. 1–27, 2018, doi: 10.1186/s40411-018-0059-z.

[3] A. Mandal and S. C. Pal, “Achieving agility through BRIDGE process model: an approach to integrate the Agile and disciplined software development,” *Innov. Syst. Softw. Eng.*, vol. 11, no. 1, pp. 1–7, Mar. 2015, doi: 10.1007/s11334-014-0239-x.

[4] B. R. Ingle, *Design Thinking for Entrepreneurs and Small Businesses Putting the Power of Design to Work*, 1st ed. 2013. Berkeley, CA: Apress, 2013.

[5] S. Tarwani and A. Chug, “Agile Methodologies in Software Maintenance: A Systematic Review,” *Inform. Ljubl.*, vol. 40, no. 4, pp. 415–426, Dec. 2016.

[6] V. Garousi, G. Giray, E. Tüzün, C. Catal, and M. Felderer, “Closing the gap between software engineering education and industrial needs,” 20181205, Accessed: Jan. 22, 2020. [Online]. Available: http://arxiv.org/abs/1812.01954.

[7] M. Karlson and F. Olsson, “Investigating the Newly Graduated StudentsExperience after University,” 2019.

[8] M. M. Qadir and M. Usman, “Software Engineering Curriculum: A systematic mapping study,” 2011, pp. 269–274, doi: 10.1109/MySEC.2011.6140682.

[9] B. Boehm and S. K. Mobasser, “System Thinking: Educating T-Shaped Software Engineers,” in *2015 IEEE 28th Conference on Software Engineering Education and Training*, Florence, Italy, May 2015, pp. 13–16, doi: 10.1109/CSEET.2015.11.

[10] N. M. Devadiga, “Software Engineering Education: Converging with the Startup Industry,” in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, Savannah, GA, Nov. 2017, pp. 192–196, doi: 10.1109/CSEET.2017.38.

[11] S. Liu, “Engineering the Success of Software Development,” *IT Prof.*, vol. 15, no. 5, pp. 4–5, 2013, doi: 10.1109/MITP.2013.76.

[12] S. Dalal and R. Chhillar, “Empirical study of root cause analysis of software failure,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 4, pp. 1–7, 2013, doi: 10.1145/2492248.2492263.

[13] C. Leem and Y. Yoon, “A maturity model and an evaluation system of software customer satisfaction: the case of software companies in Korea,” *Ind. Manag. Data Syst.*, vol. 104, no. 3/4, pp. 347–354, 2004, doi: 10.1108/02635570410530757.

[14] A. Hulshult and T. Krehbiel, “Using Eight Agile Practices in an Online Course to Improve Student Learning and Team Project Quality,” *J. High. Educ. Theory Pract.*, vol. 19, no. 3, pp. 55–67, 2019.

[15] T. C. Krehbiel *et al.*, "Agile Manifesto for Teaching and Learning," *J. Eff. Teach.*, vol. 17, no. 2, pp. 90–111, 2017.
[16] M. Kropp, A. Meier, and R. Biddle, "Teaching Agile Collaboration Skills in the Classroom," 2016, pp. 118–127, doi: 10.1109/CSEET.2016.27.
[17] M. S. Mirza, A. V. Choday, and S. Datta, "Let's Do Feature Driven Scrum," in *Proceedings of the 2019 3rd International Conference on Software and e-Business*, Tokyo, Japan, Dec. 2019, pp. 110–114, doi: 10.1145/3374549.3374573.
[18] R. Razzouk and V. Shute, "What Is Design Thinking and Why Is It Important?," *Rev. Educ. Res.*, vol. 82, no. 3, pp. 330–348, 2012, doi: 10.3102/0034654312457429.
[19] T. Kurokawa, *Service design and delivery: how design thinking can innovate business and add value to society*, First edition. New York, New York (222 East 46th Street, New York, NY 10017): Business Expert Press, 2015.
[20] D. Dunne, "Implementing design thinking in organizations: an exploratory study," *J. Organ. Des.*, vol. 7, no. 1, pp. 1–16, 2018, doi: 10.1186/s41469-018-0040-7.
[21] "Design Thinking vs. Agile: Combine Problem Finding & Problem Solving," *Mendix*, Oct. 25, 2017. https://www.mendix.com/blog/design-thinking-vs-Agile-combine-problem-finding-problem-solving-better-outcomes/ (accessed Feb. 04, 2020).
[22] "Brainstorm Eggs for Your Grandparents," *STEM Family*, Nov. 06, 2017. https://www.stem.family/2017/11/06/making-breakfast-for-your-grandparents/ (accessed Feb. 10, 2020).
[23] "Back to KG???!!!" https://dschool-old.stanford.edu/groups/k12/wiki/2bacb/The_Fun_Challenge.html (accessed Feb. 14, 2020).
[24] "Spaghetti & Marshmallow Exercise." https://dschool-old.stanford.edu/groups/k12/wiki/c6410/Spaghetti__Marshmallow_Exercise.html (accessed Feb. 15, 2020).
[25] "Agile software development," *Wikipedia*. Feb. 07, 2020, Accessed: Feb. 07, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Agile_software_development&oldid=939575369.
[26] "What is Design Thinking?," *The Interaction Design Foundation*. https://www.interaction-design.org/literature/topics/design-thinking (accessed Feb. 07, 2020).
[27] W. M. D. Ruchira Prasad, G. I. U. S. Perera, K. V. Jeeva Padmini, and H. M. N. Dilum Bandara, "Adopting Design Thinking Practices to Satisfy Customer Expectations in Agile Practices: A Case from Sri Lankan Software Development Industry," 2018, pp. 471–476, doi: 10.1109/MERCon.2018.8422006.
[28] M. Palacin-Silva, J. Khakurel, A. Happonen, T. Hynninen, and J. Porras, "Infusing Design Thinking into a Software Engineering Capstone Course," 2017, vol. 2017-, pp. 212–221, doi: 10.1109/CSEET.2017.41.
[29] G. Washington and R. Shirvani, "Towards Understanding and Modeling Empathy for Use in Motivational Design Thinking," *arXiv.org*, 2019, Accessed: Feb. 08,

2020. [Online]. Available: http://search.proquest.com/docview/2266668155/?pq-origsite=primo.

[30] "Design thinking | Design Defined | InVision," *Design thinking | Design Defined | InVision*. https://www.invisionapp.com/design-defined/design-thinking/ (accessed Feb. 04, 2020).

[31] J. Dalton and T. Kahute, "Why Empathy and Customer Closeness is Crucial for Design Thinking," *Des. Manag. Rev.*, vol. 27, no. 2, pp. 20–27, 2016, doi: 10.1111/drev.12004.

[32] "Techniques for Empathy Interviews in Design Thinking," *Web Design Envato Tuts+*. https://webdesign.tutsplus.com/articles/techniques-of-empathy-interviews-in-design-thinking--cms-31219 (accessed Feb. 09, 2020).

[33] W. L. in R.-B. U. Experience, "Nielsen Norman Group: UX Research, Training, and Consulting," *Nielsen Norman Group*. https://www.nngroup.com/articles/empathy-mapping/ (accessed Feb. 09, 2020).

[34] A. Agius, "How to Create an Effective Customer Journey Map [Examples + Template]." https://blog.hubspot.com/service/customer-journey-map (accessed Feb. 10, 2020).

[35] *Customer journey maps*. .

[36] R. F. Dam and Y. S. Teo, "Stage 2 in the Design Thinking Process: Define the Problem and Interpret the Results," *The Interaction Design Foundation*. https://www.interaction-design.org/literature/article/stage-2-in-the-design-thinking-process-define-the-problem-and-interpret-the-results (accessed Feb. 10, 2020).

[37] R. Popli, N. Chauhan, and H. Sharma, "Prioritising user stories in Agile environment," 2014, pp. 515–519, doi: 10.1109/ICICICT.2014.6781336.

[38] C. G. Cobb, *The project manager's guide to mastering Agile: principles and practices for an adaptive approach*. Hoboken, New Jersey: Wiley, 2015.

[39] C. Griffiths, "Fostering A Culture Of Creativity In A Smaller Firm," *Leadersh. Excell.*, vol. 36, no. 6, pp. 12–13, 2019.

[40] F. Rustler, *Mind mapping for dummies*. Chichester [England: John Wiley & Sons, 2012.

[41] Evan Afri and Muhammad Khoiruddin Harahap, "INCREASING TOEFL SCORE USING MIND MAPPING METHOD," *Lang. Lit. J. Linguist. Lit. Lang. Teach.*, vol. 3, no. 2, pp. 234–240, 2019, doi: 10.30743/ll.v3i2.1977.

[42] R. F. Dam and Y. S. Teo, "Stage 4 in the Design Thinking Process: Prototype," *The Interaction Design Foundation*. https://www.interaction-design.org/literature/article/stage-4-in-the-design-thinking-process-prototype (accessed Feb. 14, 2020).

[43] "Applying the KANO Model in Mobile Services World: A Report from the Frontline - University of Houston." https://uh.primo.exlibrisgroup.com/discovery/fulldisplay?docid=ieee_s6511800&context=PC&vid=01UHO_INST:CLAKE&lang=en&search_scope=UHCL_EVRYTHING&adaptor=Primo%20Central&tab=Everything&query=any,contains,kano%20model%20Agile%20%20&mode=basic (accessed Feb. 15, 2020).

[44] J. Parcell and S. Holden, “Agile policy development for digital government: an exploratory case study,” 2013, pp. 11–17, doi: 10.1145/2479724.2479731.
[45] “How Working Agreements Help Scrum Teams,” *ClearlyAgile - Agile Transformation, Certified Training, DevOps, and Agile Software Development*. https://www.clearlyAgileinc.com/Agile-blog/how-working-agreements-help-scrum-teams (accessed Mar. 18, 2020).
[46] H. H. Osman and M. E. Musa, “A Survey of Agile Software Estimation Methods,” vol. 7, no. 3, p. 5.
[47] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in Agile software development: a systematic literature review,” in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, 2014, pp. 82–91, doi: 10.1145/2639490.2639503.
[48] A. A. Albarqi and R. Qureshi, “The Proposed L-Scrumban Methodology to Improve the Efficiency of Agile Software Development,” *Int. J. Inf. Eng. Electron. Bus. Hong Kong*, vol. 10, no. 3, p. 23, May 2018, doi: http://dx.doi.org/10.5815/ijieeb.2018.03.04.
[49] K. Chen and A. Rea, “Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses,” *J. Inf. Syst. Educ. West Lafayette*, vol. 29, no. 2, pp. 53–64, Spring 2018.

APPENDIX A:

FLASHCARD APPLICATION CODE

Splash screen code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Support.V7.App;
using Android.Views;
using Android.Widget;
using Flashcards.SQLite;

namespace Flashcards.Activity
{/// <summary>
/// MainLauncher = true..starting point
/// </summary>
    [Activity(Label = "@string/app_name", Theme = "@style/SplashScreenTheme",
MainLauncher = true)]
    public class SplashScreenActivity : AppCompatActivity
    {
        private SqLiteDatabase DbDatabase = new SqLiteDatabase();
        //protected override void OnCreate(Bundle savedInstanceState)
        //{
        //    base.OnCreate(savedInstanceState);

        //    // Create your application here
        //}
        protected override void OnResume()
        {
            try
            {
                base.OnResume();
                new Handler(Looper.MainLooper).Post(new
Java.Lang.Runnable(FirstRunExcite));
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
        }

        private void FirstRunExcite()
        {
            try
            {
                DbDatabase = new SqLiteDatabase();
                DbDatabase.CheckTablesStatus();
                DbDatabase.CheckDataStatus();
                //start main page
                StartActivity(new Intent(this, typeof(MainActivity)));

                DbDatabase.Dispose();
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
        }
    }

}
```

Main Activity code

```
using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using Android.Animation;

using Android.App;

using Android.Content;

using Android.Graphics;

using Android.Graphics.Drawables;

using Android.OS;
using Android.Runtime;
using Android.Support.Design.Widget;
using Android.Support.V7.App;
using Android.Support.V7.Widget;
using Android.Util;
using Android.Views;
using Android.Widget;
using Flashcards.Anmated;
using Flashcards.Models;
using Flashcards.SQLite;
using AlertDialog = Android.App.AlertDialog;

namespace Flashcards
{
    [Activity(Label = "@string/app_name", Theme = "@style/AppTheme.NoActionBar")]
    public class MainActivity : AppCompatActivity
    {
        private SqLiteDatabase DbDatabase = new SqLiteDatabase();
        CardView cardview_question, cardview_answer;
        TextView txtQuestion, txtAnswer;
        ImageView editImageView, deleteImageView, prevImageView, nextImageView;
        List<QuizeTb> quizList;
        QuizeTb selectedQuize;
        EditText edit_answer, edit_question;
        private Dialog dialog;
        int index = 0;

        public string emptyData = "Empty...Please Add data";
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            SetContentView(Resource.Layout.activity_main);

            Android.Support.V7.Widget.Toolbar toolbar =
    FindViewById<Android.Support.V7.Widget.Toolbar>(Resource.Id.toolbar);
            SetSupportActionBar(toolbar);

            FloatingActionButton fab =
    FindViewById<FloatingActionButton>(Resource.Id.fab);
            fab.Click += FabOnClick;

            InitControl();
            FetchDataFromDB();
        }

        private void FetchDataFromDB()
        {
            try
            {
```

```
        quizList = new List<QuizeTb>();
        DbDatabase = new SqLiteDatabase();
        DbDatabase.CheckTablesStatus();
        quizList = DbDatabase.FetchData();
        DbDatabase.Dispose();
        ShowData(quizList[0]);
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }
}

private void ShowData(QuizeTb quizModel)
{
    try
    {
        selectedQuize = quizModel;
        txtQuestion.Text = quizModel.Question;
        txtAnswer.Text = quizModel.Answer;
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }
```

```
        }
        private void DeleteData(QuizeTb quizModel)
        {
            try
            {
                if (quizList.Count>1)
                {
                    if (RemoveFromDB(quizModel))
                    {
                        quizList.Remove(quizModel);
                        if (index > 0)
                        {
                            index--;
                        }
                        ShowData(quizList[index]);
                    }

                }
                else
                {
                    if (quizList.Count == 1 && quizList[0].Question.Trim() !=
emptyData.Trim())
                    {
                        selectedQuize.Question = emptyData;
                        selectedQuize.Answer = emptyData;
                        if (UpdateInDB(selectedQuize))
                        {
                            quizList[0] = selectedQuize;
                            ShowData(quizList[0]);
                        }
                    }
                    else
                    {
                        if (UpdateFirstEmptyData())
                        {
                            FetchDataFromDB();
                        }
                    }
                }

                Toast.MakeText(this, "Thank You..Flashcard updated successfully",
ToastLength.Short).Show();
                ShowData(quizList[index]);
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
            }
        }
```

```
private bool UpdateFirstEmptyData()
{
   try
   {
      DbDatabase = new SqLiteDatabase();
      DbDatabase.CheckTablesStatus();
      DbDatabase.CheckDataStatus();
      DbDatabase.Dispose();
      return true;
   }
   catch (Exception ex)
   {
      Console.Write(ex.Message);
      return false;
   }
}

private bool RemoveFromDB(QuizeTb quizModel)
{
   try
   {
      DbDatabase = new SqLiteDatabase();
      DbDatabase.DeleteRow(quizModel);
      DbDatabase.Dispose();
      return true;
```

```
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
                return false;
            }
        }

        private void EditData(QuizeTb quizModel)
        {
            try
            {
                dialog = new Dialog(this);
                dialog.SetContentView(Resource.Layout.insert_query);
                dialog.Window.SetBackgroundDrawable(new
ColorDrawable(Color.Transparent));
                TextView add =
dialog.FindViewById<TextView>(Resource.Id.textView_add);
                add.Text = "Update";
                edit_answer = dialog.FindViewById<EditText>(Resource.Id.edit_answer);
                edit_answer.Text = quizModel.Answer;
                edit_question = dialog.FindViewById<EditText>(Resource.Id.edit_question);
                edit_question.Text= quizModel.Question;
                add.Click += Update_Click;
                dialog.Window.SetLayout((int)(GetScreenWidth(this) * .95),
LinearLayout.LayoutParams.WrapContent);
                dialog.SetCancelable(true);
                dialog.Show();
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
            }
        }

        private void InitControl()
        {
            try
            {
                cardview_question =
FindViewById<CardView>(Resource.Id.cardview_question);
                cardview_question.Click += Cardview_question_Click;
                cardview_question.Visibility = ViewStates.Invisible;
                cardview_answer =
FindViewById<CardView>(Resource.Id.cardview_answer);
                cardview_answer.Visibility = ViewStates.Invisible;
                cardview_answer.Click += Cardview_question_Click;
                txtQuestion = FindViewById<TextView>(Resource.Id.txtQuestion);
                txtAnswer = FindViewById<TextView>(Resource.Id.txtAnswer);
                prevImageView = FindViewById<ImageView>(Resource.Id.prevImageView);
                prevImageView.Click += TextView_prev_Click;
                nextImageView = FindViewById<ImageView>(Resource.Id.nextImageView);
                nextImageView.Click += TextView_next_Click;
                cardview_question.Visibility = ViewStates.Visible;
                cardview_answer.Visibility = ViewStates.Invisible;
                editImageView = FindViewById<ImageView>(Resource.Id.editImageView);
                editImageView.Click += EditImageView_Click;
                deleteImageView =
FindViewById<ImageView>(Resource.Id.deleteImageView);
                deleteImageView.Click += DeleteImageView_Click;
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
            }
        }

        private void DeleteImageView_Click(object sender, EventArgs e)
        {
            try
            {
                if (quizList.Count == 1 && quizList[0].Question.Trim() == emptyData.Trim())
                {
                    Toast.MakeText(this, "Empty FlashCard...Please add data first",
ToastLength.Short).Show();
                }
                else
                {
                    // base.OnBackPressed();
                    Android.App.AlertDialog.Builder dialog = new AlertDialog.Builder(this);
                    AlertDialog alert = dialog.Create();
                    alert.SetTitle("Alert!");
                    alert.SetMessage("Do you want to delete this FlashCard?");
                    alert.SetButton("YES", (c, ev) =>
                    {
                        alert.Dismiss();
                        if (index >= 0)
                        {
                            DeleteData(quizList[index]);
                        }
                    });
                    alert.SetButton2("CANCEL", (c, ev) => { });
                    alert.Show();
                }

            }
            catch (Exception ex)
            {
```

```
            Console.Write(ex.Message);
        }
    }

    private void EditImageView_Click(object sender, EventArgs e)
    {
        try
        {
            if (quizList.Count == 1 && quizList[0].Question.Trim() == emptyData.Trim())
            {
                Toast.MakeText(this, "Empty FlashCard...Please add data first",
ToastLength.Short).Show();
            }
            else
            {
                EditData(quizList[index]);
            }
        }
        catch (Exception ex)
        {
            Console.Write(ex.Message);
        }
    }

    private void TextView_next_Click(object sender, EventArgs e)
```

```
{
   try
   {
      if (index == quizList.Count-1)
      {
         Toast.MakeText(this, "Sorry..Last Flashcard", ToastLength.Short).Show();
      }
      else
      {
         cardview_question.Visibility = ViewStates.Visible;
         cardview_answer.Visibility = ViewStates.Invisible;
         index++;
         ShowData(quizList[index]);
      }
   }
   catch (Exception ex)
   {
      Console.Write(ex.Message);
   }
}

private void TextView_prev_Click(object sender, EventArgs e)
{
   try
   {
        if (index != 0)
        {
            cardview_question.Visibility = ViewStates.Visible;
            cardview_answer.Visibility = ViewStates.Invisible;
            index--;
            ShowData(quizList[index]);
        }
        else
        {
            Toast.MakeText(this, "Sorry..First Flashcard", ToastLength.Short).Show();
        }
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }
}

private void Cardview_question_Click(object sender, EventArgs e)
{
    try
    {
        if (cardview_question.Visibility == ViewStates.Invisible)
        {
            cardview_question.Visibility = ViewStates.Visible;
```

```
                cardview_answer.Visibility = ViewStates.Invisible;
                //FlipAnimation(this, cardview_question);
                CustomAnim.AnimLeft(this, cardview_question);
            }
            else
            {
                cardview_question.Visibility = ViewStates.Invisible;
                cardview_answer.Visibility = ViewStates.Visible;
                //FlipAnimation(this, cardview_answer);
                CustomAnim.AnimRight(this, cardview_answer);
            }
        }
        catch (Exception ex)
        {
            Console.Write(ex.Message);
        }

    }

    private void FlipAnimation(Context context, View view)
    {
        try
        {
            ObjectAnimator anim =
(ObjectAnimator)AnimatorInflater.LoadAnimator(context, Resource.Animator.flipping);
```

```
                //ObjectAnimator anim =
(ObjectAnimator)AnimatorInflater.LoadAnimator(context, Resource.Animation.flip);
                anim.SetTarget(view);
                anim.SetDuration(3000);
                anim.Start();
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
            }
        }

        public override bool OnCreateOptionsMenu(IMenu menu)
        {
            MenuInflater.Inflate(Resource.Menu.menu_main, menu);
            return true;
        }

        public override bool OnOptionsItemSelected(IMenuItem item)
        {
            int id = item.ItemId;
            if (id == Resource.Id.action_settings)
            {
                return true;
            }
            return base.OnOptionsItemSelected(item);
        }

        private void FabOnClick(object sender, EventArgs eventArgs)
        {
            try
            {
                //View view = (View) sender;
                //Snackbar.Make(view, "Replace with your own action",
Snackbar.LengthLong)
                //    .SetAction("Action",
(Android.Views.View.IOnClickListener)null).Show();
                dialog = new Dialog(this);
                dialog.SetContentView(Resource.Layout.insert_query);
                dialog.Window.SetBackgroundDrawable(new
ColorDrawable(Color.Transparent));
                TextView add =
dialog.FindViewById<TextView>(Resource.Id.textView_add);
                edit_answer = dialog.FindViewById<EditText>(Resource.Id.edit_answer);
                edit_question = dialog.FindViewById<EditText>(Resource.Id.edit_question);
                add.Click += Add_Click;

                dialog.Window.SetLayout((int)(GetScreenWidth(this) * .95),
LinearLayout.LayoutParams.WrapContent);
```

```
        dialog.SetCancelable(true);
        dialog.Show();
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }

}

public static int GetScreenWidth(Context context)
{
    DisplayMetrics dm = Application.Context.Resources.DisplayMetrics;
    return dm.WidthPixels;
}

public override void OnBackPressed()
{
    // base.OnBackPressed();
    Android.App.AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    AlertDialog alert = dialog.Create();
    alert.SetTitle("Alert!");
    alert.SetMessage("Do you really want to exit?");
    alert.SetButton("OK", (c, ev) =>
    {
        alert.Dismiss();
        var a = new Intent(Intent.ActionMain);
        a.AddCategory(Intent.CategoryHome);
        a.SetFlags(ActivityFlags.NewTask);
        StartActivity(a);
    });
    alert.SetButton2("CANCEL", (c, ev) => { });
    alert.Show();
}


private void Update_Click(object sender, EventArgs e)
{
    try
    {
        if (string.IsNullOrEmpty(edit_question.Text))
        {
            Toast.MakeText(this, "Please add question", ToastLength.Short).Show();
        }
        else if (string.IsNullOrEmpty(edit_answer.Text))
        {
            Toast.MakeText(this, "Please add answer", ToastLength.Short).Show();
        }
        else
        {
```

```
                    QuizModel updatedQA = new QuizModel { Question = edit_question.Text,
Answer = edit_answer.Text };
                    selectedQuize.Question = updatedQA.Question;
                    selectedQuize.Answer = updatedQA.Answer;
                    if (UpdateInDB(selectedQuize))
                    {
                        quizList[index].Question = updatedQA.Question;
                        quizList[index].Answer = updatedQA.Answer;
                        Toast.MakeText(this, "Thank You..Flashcard updated successfully",
ToastLength.Short).Show();
                        dialog?.Dismiss();
                        ShowData(quizList[index]);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
            }

        }

        private bool UpdateInDB(QuizeTb selectedQA)
        {
            try
```

```
    {
        DbDatabase = new SqLiteDatabase();
        DbDatabase.UpdateRow(selectedQA);
        DbDatabase.Dispose();
        return true;
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
        return false;
    }
}

private void Add_Click(object sender, EventArgs e)
{
    try
    {
        if (string.IsNullOrEmpty(edit_question.Text))
        {
            Toast.MakeText(this, "Please add question", ToastLength.Short).Show();
        }
        else if (string.IsNullOrEmpty(edit_answer.Text))
        {
            Toast.MakeText(this, "Please add answer", ToastLength.Short).Show();
        }
```

```
            else
            {
                QuizeTb newQA = new QuizeTb { Question = edit_question.Text, Answer =
edit_answer.Text };

                if (quizList.Count==1 && quizList[0].Question.Trim()==emptyData.Trim())
                {
                    selectedQuize.Question = newQA.Question;
                    selectedQuize.Answer = newQA.Answer;
                    if (UpdateInDB(selectedQuize))
                    {
                        quizList[0] = newQA;
                        ShowData(quizList[0]);
                    }
                }
                else
                {
                   newQA.QuestionID = quizList[quizList.Count - 1].QuestionID + 1;
                   if (AddInDB(newQA))
                   {
                        quizList.Add(newQA);
                   }
                }
                Toast.MakeText(this, "Thank You..Quiz updated successfully",
ToastLength.Short).Show();
```

```
            dialog?.Dismiss();
        }
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }

}

private bool AddInDB(QuizeTb newQA)
{
    try
    {
        DbDatabase = new SqLiteDatabase();
        DbDatabase.InsertRow(newQA);
        DbDatabase.Dispose();
        return true;
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
        return false;
    }
}
```

```
        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
            try
            {
                Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

                base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
            }
            catch (Exception ex)
            {
                Console.Write(ex.Message);
            }

        }
    }
}
```

Database connections

Creating table

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using SQLite;

namespace Flashcards.SQLite
{/// <summary>
/// Model for Question and answer
/// </summary>
    public class QuizeTb
    {
        [PrimaryKey, AutoIncrement]
        public int QuestionID { get; set; }
        public string Question { get; set; }
        public string Answer { get; set; }
    }

}
```

Database code

```
using Android.Database.Sqlite;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;

namespace Flashcards.SQLite
{
    public class SqLiteDatabase : IDisposable
    {
        //############# DON'T MODIFY HERE #############
        private static readonly string Folder =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        public static readonly string PathCombine = Path.Combine(Folder, "flashcard.db");
        private SQLiteConnection Connection;

        public void CheckTablesStatus()
        {
            try
            {
                using (OpenConnection())
                {
                    if (Connection == null) return;
                    Connection.CreateTable<QuizeTb>();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
        }

        private SQLiteConnection OpenConnection()
        {
            try
            {
                Connection = new SQLiteConnection(PathCombine);
                return Connection;
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                return null;
            }
        }
        public void Dispose()
        {
            try
            {
                using (OpenConnection())
                {
                    if (Connection == null) return;
                    Connection.Dispose();
                    Connection.Close();
                    GC.SuppressFinalize(this);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e);

            }
        }

        internal void CheckDataStatus()
        {
            try
            {
                using (OpenConnection())
                {
                    if (Connection == null)
                    {
                        return ;
                    }
                    else
                    {
                        var dataCount = Connection.Table<QuizeTb>().Count();
                        if (dataCount == 0)
                        {
                            InsertDefaultRow();
                            //InsertDefaultMultiRow();
                        }
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                return ;
            }


        }

        internal List<QuizeTb> FetchData()
        {
            try
            {
                using (OpenConnection())
                {
                    if (Connection == null)
                    {
                        return null;
                    }
                    else
                    {
                        var data = Connection.Table<QuizeTb>().ToList();
                        return data;
                    }
                }
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                return null;
            }
        }

        private void InsertDefaultMultiRow()
        {
            try
            {

                List<object> quizList = new List<object>();
                QuizeTb q1 = new QuizeTb {QuestionID=2, Question = "What is the supreme 
law of the land?", Answer = "The Constitution" };
                QuizeTb q2 = new QuizeTb { QuestionID = 3, Question = "What does the 
Constitution do?", Answer = "protects basic rights of Americans" };
                QuizeTb q3 = new QuizeTb { QuestionID = 4, Question = "The idea of self-
government is in the first three words of the Constitution. What are these words?", 
Answer = "We the People" };
                QuizeTb q4 = new QuizeTb { QuestionID = 5, Question = "What is an 
amendment?", Answer = "An addition or change" };
                quizList.Add(q1);
                quizList.Add(q2);
                quizList.Add(q3);
                quizList.Add(q4);
                InsertListOfRows(quizList);
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
        }

        private void InsertDefaultRow()
        {
            try
            {
                var dataUser = Connection.Table<QuizeTb>().FirstOrDefault();
                if (dataUser==null)
                {
```

```
            QuizeTb newdata = new QuizeTb();
            newdata.QuestionID = 1;
            newdata.Question = "Empty...Please Add data";
            newdata.Answer = "Empty...Please Add data";
            InsertRow(newdata);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
    }
}

#region General

public void InsertRow(object row)
{
    try
    {
        using (OpenConnection())
        {
            if (Connection == null) return;
            Connection.Insert(row);
        }
    }
    catch (Exception exception)
    {
        Console.WriteLine(exception);
    }
}

public void UpdateRow(object row)
{
    try
    {
        using (OpenConnection())
        {
            if (Connection == null) return;
            Connection.Update(row);
        }
    }
    catch (Exception exception)
    {
        Console.WriteLine(exception);
```

```
            }
        }

        public void DeleteRow(object row)
        {
            try
            {
                using (OpenConnection())
                {
                    if (Connection == null) return;
                    Connection.Delete(row);
                }
            }
            catch (Exception exception)
            {
                Console.WriteLine(exception);
            }
        }

        public void InsertListOfRows(List<object> row)
        {
            try
            {
                using (OpenConnection())
                {
                    if (Connection == null) return;
                    Connection.InsertAll(row);
                }
            }
            catch (Exception exception)
            {
                Console.WriteLine(exception);
            }
        }

        #endregion
    }


}
```

Main activity layout code

```
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent">

 <android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />

</android.support.design.widget.AppBarLayout>
<LinearLayout
    android:orientation="vertical"
    android:padding="10dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="60dp"
    android:id="@+id/linearLayout2" >

<include layout="@layout/content_main" />
<RelativeLayout
     android:layout_margin="10dp"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
            android:src="@drawable/prev"
            android:tint="@color/colorAccent"
            android:layout_alignParentStart="true"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:id="@+id/prevImageView" />
    <ImageView
            android:src="@drawable/next"
            android:tint="@color/colorAccent"
            android:layout_alignParentEnd="true"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:id="@+id/nextImageView" />
        <LinearLayout
            android:orientation="horizontal"
            android:layout_centerInParent="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/linearLayout3" >
             <ImageView
            android:src="@drawable/edit"
            android:tint="@color/colorGreen"
            android:layout_alignParentStart="true"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_marginHorizontal="10dp"
            android:id="@+id/editImageView" />
             <ImageView
            android:src="@drawable/delete"
            android:tint="@color/colorRed"
            android:layout_alignParentEnd="true"
            android:layout_width="50dp"
            android:layout_height="50dp"
                android:layout_marginHorizontal="10dp"
            android:id="@+id/deleteImageView" />
          </LinearLayout>

        </RelativeLayout>

        </LinearLayout>
      <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@drawable/ic_add" />


  </android.support.design.widget.CoordinatorLayout>
```

Content page code

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="245dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main"
     android:gravity="center_horizontal"
    android:background = "@drawable/roundcorner"
    >

<android.support.v7.widget.CardView
        android:id="@+id/cardview_question"
        android:layout_width="fill_parent"
        android:layout_height="245dp"
        app:cardBackgroundColor="#cabfbf"
        android:layout_gravity="center_horizontal">
        <TextView
            android:text="Question:-"
            android:textSize="12dp"
            android:textStyle="bold"
            android:gravity="top"
            android:textColor="@color/colorGreen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />

            <TextView
            android:id="@+id/txtQuestion"
            android:text="Questions"
            android:textSize="16dp"
            android:layout_marginTop="0dp"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:textColor="@color/colorBlack"
            android:layout_centerVertical="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentEnd="true" />
</android.support.v7.widget.CardView>
<android.support.v7.widget.CardView
```

```
        android:id="@+id/cardview_answer"
        android:layout_width="fill_parent"
        android:layout_height="245dp"
        app:cardBackgroundColor="#d6f3da"
        android:layout_gravity="center_horizontal">
   <TextView
            android:text="Answer:-"
            android:textSize="12dp"
            android:textStyle="bold"
            android:gravity="top"
            android:textColor="@color/colorRed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
<TextView
            android:id="@+id/txtAnswer"
            android:text="Answer"
             android:textSize="16dp"
            android:layout_marginTop="0dp"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:textColor="@color/colorBlack"
            android:layout_centerVertical="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentEnd="true" />
</android.support.v7.widget.CardView>


</RelativeLayout>
```

Insert query code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
     android:background = "@drawable/roundcorner"
    android:padding="5dp">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <TextView
            android:text="Question:-"
            android:gravity="center"
            android:layout_gravity="center_horizontal"
            android:layout_width="75dp"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="start"
            android:id="@+id/edit_question" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
         <TextView
            android:text="Answer:-"
            android:gravity="center"
            android:layout_gravity="center_horizontal"
            android:layout_width="75dp"
            android:layout_height="match_parent" />
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="start"
            android:id="@+id/edit_answer" />
    </LinearLayout>
    <TextView
        android:text="Add"
        android:layout_gravity="center_horizontal"
        android:gravity="center"
        android:textColor="#ffffff"
        android:background="@drawable/button_bg"
        android:layout_width="100dp"
        android:layout_height="40dp"
        android:id="@+id/textView_add"
        android:layout_margin="10dp"/>

</LinearLayout>
```