

SE 216 SOFTWARE PROJECT MANAGEMENT

Servicify An E-commerce Platform That Connects Different Services

Development Plan By:

Ege Sezak 20220601059

Bariş Can Ceylan 20190601053

Buğra Yurtsever 20210601068

Alperen Demirezen 20210601021

Table of contents

1.0 Introduction	3
2.0 Objective	3
3.0 Software Requirements	3
3.1 Functional Requirements	3
3.2 Non Functional Requirements	4
4.0 Software Process	5
4.1 Necessary Needs From The Organizational Process.....	5
4.2 Selected Software Process.....	5
4.3 What is SCRUM.....	5
4.4 Software Process Model	6
4.5 Reasons to Choose This Mode	7
5.0 Software Tools	7
5.1 Project Tasks Which Require Software Tool Support	7
5.2 User Interface	8
5.3 Data Management and Database Infrastructure	8
5.4 Customer Support Tools	9
5.5 Financial Management	11
6.0 Software Measurements	12
6.1 Questions to identify measurements	12
6.2 Identified measurements	12
6.3 Measurement storage and collection	12
7.0 Project Risks	14
7.1 Risk Likelihood Ranking	15
7.2 Risk Impact Ranking	16
7.3 Combined Risk Ranking	17
8.0 Project Stakeholders	17
9.0 Project Needs	18
9.1 Software Needs	18
9.2 Hardware Needs	19
9.3 Support Needs	20
10.0 Project Schedule	21
11.0 User Interfaces	23
12.0 Conclusion.....	25

1. Introduction

In today's world, we realized that there is no application that includes the services provided by other applications. It came to our mind that if we could find a way to give customers a chance to be able to reach all of those apps inside of an application, which is called "Servicify". Servicify was designed due to the most common needs of customers. We have investigated what is needed to get the most benefits, after the application is released.

2. Objective

Mobile Service applications are essential for today's establishments to give customers what they need. However, number of the applications are too high, so that Servicify was created. Servicify aims to collect different kind of services under one roof. This way Servicify provides a faster, easier, and more efficient experience to Customers.

3. Software Requirements

3.1 Functional Requirements

1. Users should be able to create accounts and log in securely.
2. Users should be able to search services near them using location services.
3. Vendors should be able to list their products or services for sale.
4. Users should be able to add items to their shopping carts and proceed to checkout.
5. Restaurants should be able to create/modify menus.
6. Users should be able to view the status of their orders in real-time.
7. Vendors should have access to tools for managing orders including order fulfillment and tracking.
8. Users should be able to leave reviews and ratings for products/services they have purchased.

9. Vendors should be able to respond to reviews and address customer feedback.
10. The application should support multiple languages and currencies to cater to a global user base.
11. Multiple payment methods should be supported, including credit/debit cards, digital wallets, and cash on delivery.
12. System should be able to recommend services based on previous user purchases.
13. Users should be able to search for specific services or products and apply filters to narrow down their choices.
14. The application should provide analytics and reporting tools for businesses to track performance, user behavior, and overall system usage.

3.2 Non Functional Requirements

1. The application must implement robust security measures, including encryption, secure authentication, and protection against common vulnerabilities.
2. In case of an unexpected crash, the system should not lose vital information.
3. Loading times for pages and images should be minimal and intuitive to ensure a seamless user experience.
4. The system should be able to handle increasing numbers of concurrent users during peak hours without a significant decrease in performance.
5. The application should be compatible with a wide range of devices and screen sizes, including smartphones, tablets, and desktop computers.
6. Modular design principles should be followed to allow for easy addition of new features and modifications
7. The application should comply with relevant regulations and standards, ensuring legal and ethical operation.
8. Regular data backups should be performed, and the system should have a reliable data recovery mechanism in case of data loss.

4.0 Software Process

4.1 Necessary Needs From The Organizational Process

1. Having daily meetings should be held to facilitate group discussion.
2. The process must be easily adaptable to changing requirements. There will always be changes during the process, it will be nice to make it adaptable.
3. It is crucial to discover an error in the process immediately when it occurs. A small error might lead to having a bigger error if not fixed in time.
4. The project should be divided into smaller tasks to aid in development. Simpler the project is, the faster the process will be.
5. Previous increments should be considered to find potential problems. It is likely to see the same problems in the future so previous increments can help us.
6. Collaboration and accountability should be encouraged among team members. It is important to pay attention to the team's motivation.
7. Increments of the product should incorporate customer feedback throughout the development process.
8. Fixing mistakes should cost minimal time and cost. If fixing problems takes too much time and money, it will jeopardize the future of the project.

4.2 Selected Software Process

Selection of the software process is a critical decision that is taken by the team. Given our needs from the software process, we decided to choose scrum framework as our software process model.

4.3 What is SCRUM:

The Scrum software development process is an iterative and incremental agile framework designed to deliver high-value software in a flexible and adaptive manner. Scrum fosters collaboration, transparency, and continuous improvement within cross-functional teams. Since scrum is a framework, hence it can be modified. This allows the team to create a productive environment. There are several roles such as product owner, scrum master and developers.

4.4 SOFTWARE PROCESS MODEL:

Our project, unifying several services under one roof is quite a large undertaking. This makes it easy to fail in the event that the team loses focus during development. With scrum, we can divide the difficult tasks of this project into smaller pieces making the team motivated to finish the project efficiently.

The Sprints:

- 1. Login/sign up system for service providers. (25 days)**
- 2. Login/sign up system for users. (10 days)**
- 3. Database design. (20 days)**
- 4. Framework for general service providers/markets. (36 days)**
- 5. Framework for clothing stores. (51 days)**
- 6. Framework for restaurants. (56 days)**
- 7. Implementing the user rating system. (20 days)**
- 8. Payment system. (20 days)**
- 9. General UI design. (20 days)**
- 10. Implementing security features. (10 days)**

4.5 REASONS TO CHOOSE THIS MODEL:

Scrum's core principles enable the team and stakeholders to constantly see the status of the work. This helps the team make better decisions and reduce risks. In startups like ours, it is imperative that we do not make any mistakes. Since the smallest mistake can bankrupt us. Sprints are essential for scrum. Each Sprint aims to increase value. This enables continuous improvement of the product and faster delivery to customers and stakeholders. E-commerce is a competitive market and it is important to follow new trends and adapt to the changing environment. Scrum provides the ability to adapt quickly to changing requirements making it ideal for our use case. With feedback from the stakeholders, the team constantly learns and improves while delivering the product. Scrum encourages the creation of small, cross-functional teams. Such teams often work more efficiently and produce more creative solutions, and the absence of multi-layered hierarchies alleviate management overhead which speeds up the development. To conclude, scrum is a versatile framework that aids our team to make better decisions and decrease the risk of fatal mistakes, all the while making our project more flexible to changing requirements of the industry. These advantages show that Scrum provides more effective business process management for our organization. Scrum offers excellent solutions to achieve our goals, especially in complex and rapidly changing projects like ours.

5.0 Software Tools

5.1 Project Tasks Which Require Software Tool Support

1	User Interface
2	Data Management and Database Infrastructure
3	Customer Support Tool
4	Financial Management Tool

5.2 User Interface

Kotlin/Jetpack Compose:

Android recommends Jetpack Compose as a contemporary toolset for creating native user interfaces. It streamlines and expedites Android UI development. Develop your project more quickly with less code, stronger tools, and user-friendly Kotlin APIs.

Swift/SwiftUI:

SwiftUI helps you build great-looking apps across all Apple platforms with the power of Swift — and surprisingly little code. You can bring even better experiences to everyone, on any Apple device, using just one set of tools and APIs.

5.3 Data Management and Database Infrastructure

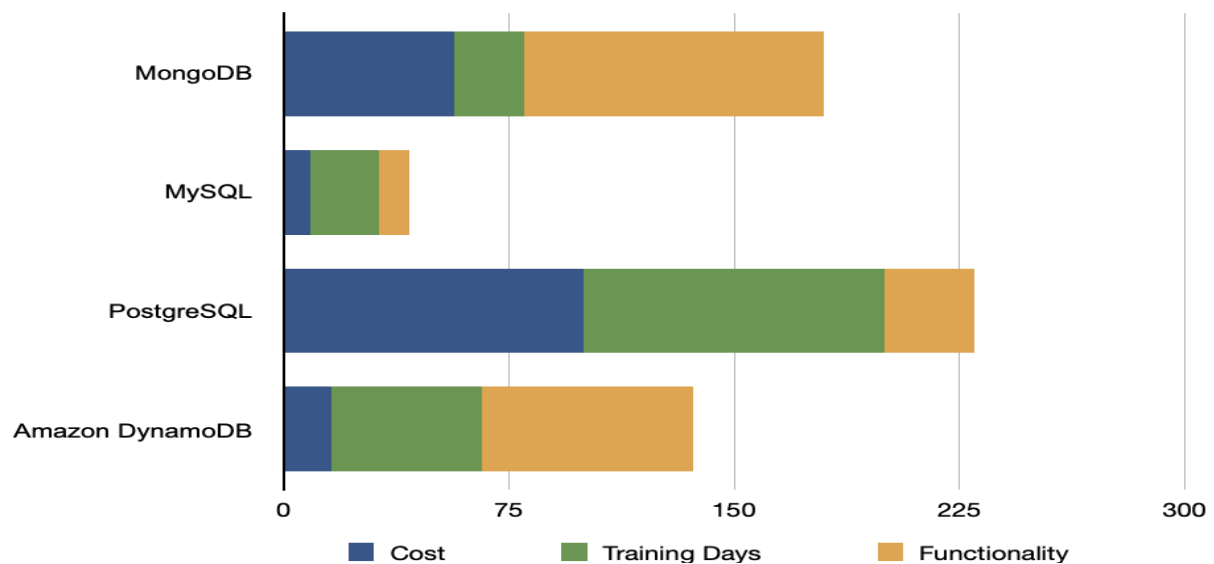
Tool Cost/Training/Functionality Data

Tool	MongoDB	MySQL	PostgreSQL	Amazon DynamoDB
Cost	191.406\$	29.327,64\$	335.373\$	54.210,24\$
Training Days	7	7	30	15
Functionality	10	1	3	7

Normalized Cost/Training/Functionality Data

Tool	MongoDB	MySQL	PostgreSQL	Amazon DynamoDB
Cost	57.1	8.7	100.0	16,2
Training Days	23.3	23.3	100.0	50.0
Functionality	100	10	30	70

Normalized Tool Graph



Which tool has been selected? Why?

MongoDB is an open-source NoSQL database management system. NoSQL databases are utilized as an alternative to conventional relational databases. When dealing with big, dispersed data sets, NoSQL databases come in handy. MongoDB is a tool for managing and storing document-oriented data. High-volume data storage is facilitated by MongoDB, which enables enterprises to store vast volumes of data while maintaining speed. In addition, MongoDB's ad hoc queries, indexing, load balancing, aggregation, server-side JavaScript execution, and other features are integral for our use case.

5.4 Customer Support Tools

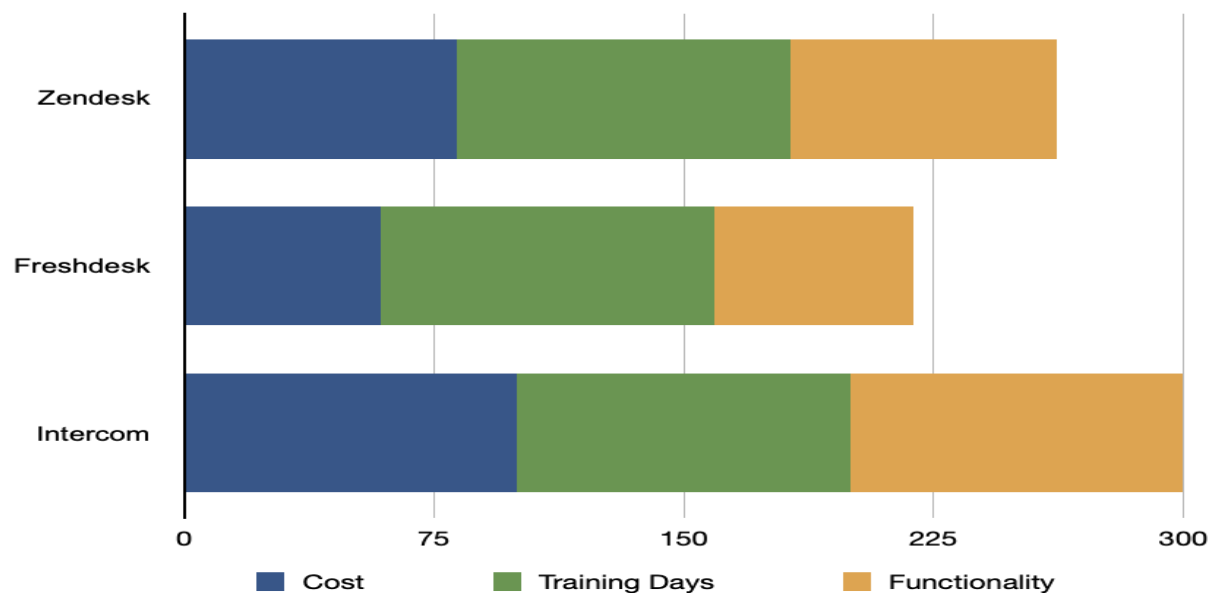
Tool Cost/Training/Functionality Data

Tool	Zendesk	Freshdesk	Intercom
Cost	1400\$	1000\$	1700\$
Training Days	7	7	7
Functionality	8	6	10

Normalized Cost/Training/Functionality Data

Tool	Zendesk	Freshdesk	Intercom
Cost	82.3	58.8	100.0
Training Days	100.0	100.0	100.0
Functionality	80.0	60.0	100.0

Normalized Tool Graph



Which tool has been selected? Why?

Intercom:

Intercom is a customer messaging platform that allows businesses to engage with customers through personalized messages, live chat, and targeted campaigns. It offers features for tracking user behavior, segmenting audiences, and measuring customer satisfaction. These features will help increase sales with actions that can be taken after analyzing users. This is the reason why we chose it even though it costs more.

5.5 Financial Management

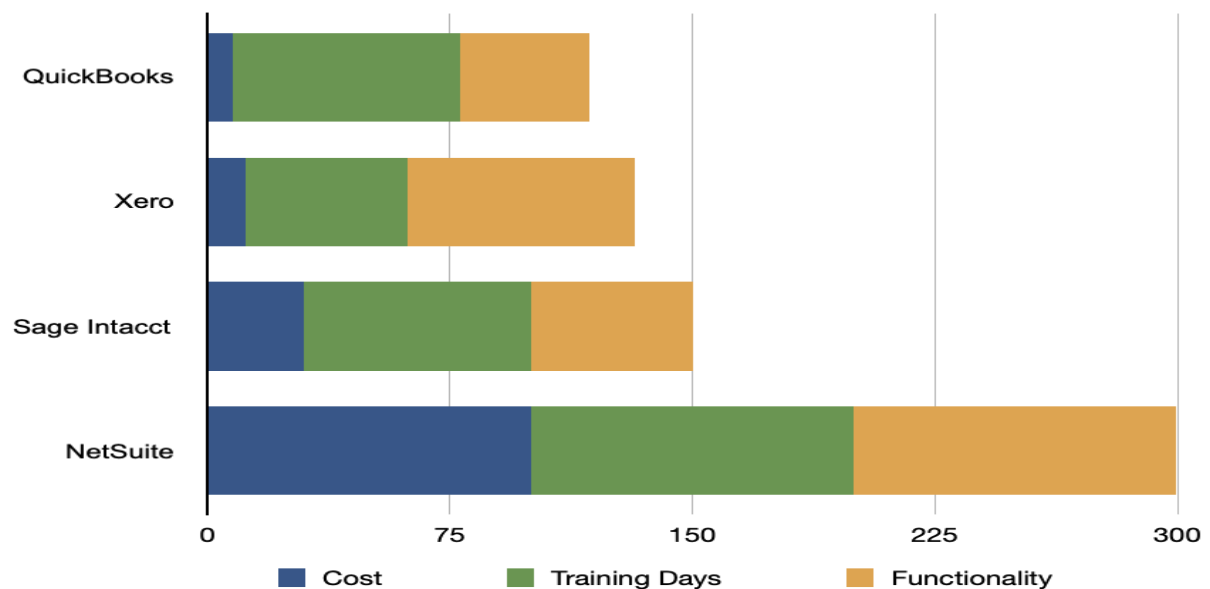
Tool Cost/Training/Functionality Data

Tool	QuickBooks	Xero	Sage Intacct	NetSuite
Cost	1000\$	1500\$	3600\$	12.000\$
Training Days	21	15	21	30
Functionality	4	7	5	10

Normalized Cost/Training/Functionality Data

Tool	QuickBooks	Xero	Sage Intacct	NetSuite
Cost	8.3	12.5	30.0	100.0
Training Days	70.0	50.0	70.0	100.0
Functionality	40.0	70.0	50.0	100.0

Normalized Tool Graph



Which tool has been selected? Why?

Xero:

Xero is a cloud-based accounting software that provides tools for invoicing, bank reconciliation, expense tracking, and financial reporting. It offers real-time collaboration with advisors. Not only is this important for businesses, but it is also cost-effective.

6.0 Software Measurements

6.1 Questions to identify measurements:

- What did the team produce? (sprints)
- How much effort did this project require?
- How much effort went to testing?
- How many times has the code been tested?
- How many commits have been done?
- What is the defect density in the code?

6.2 Identified measurements:

- Number of hours spent working.
- Number of times code committed.
- Number of sprints finished.
- Number of test cases written.
- Number of times that tests have failed.

6.3 Measurement storage and collection:

Hours spent working:

This measurement measures how much time has the team spent working on the project.

Example:

At the end of each week, a monitoring program tracks the number of hours spent working by a developer.

Number of commits:

This measurement measures how many times the code has been committed by a team member and the commit is rated by a senior developer by 0-10 depending on the features of the commit. Every member has to meet a weekly quota.

Example:

The amount of times a member committed code daily*rating of the commit.

Number of tests:

This measurement measures how many test cases the code has to pass before the code can be committed.

Example:

A development team runs unit tests regularly before every commit and the size of the test cases increases as the code base develops.

Error:

This measurement measures errors that occur during the testing phase of the software and represent undesirable behavior of software.

Example:

In the event that a developer gets more errors than normal or seen struggling with development of a feature the workload can be redistributed depending on the situation.

Defect Density:

The number of faulty components or faulty lines of code in a product. The decrease of this value may indicate the experience of the team.

Example:

During a sprint, 1000 lines of code are written and 10 of them are buggy in subsequent testing. In this case, the error density is 1%.

7.0 Software Risks

Identifying and managing risks is a critical component of project management. In this section, we will discuss the potential risks associated with the Servify project. Each risk is described in terms of its likelihood and impact on the project's success.

- 1. Communication Issues:** Communication breakdowns between developer groups can lead to misunderstandings and misaligned visions, potentially derailing the project.
- 2. Integration Challenges:** Combining various parts of a complex software can present significant challenges. Integration issues can delay the project and increase costs.
- 3. Software Problems:** Technical problems that arise during development can impede progress and affect the final product's quality.
- 4. Inadequate Planning:** Poor planning, whether due to inexperience or unforeseen issues, can result in increased time and cost for the project.
- 5. Infrastructure Problems:** Issues such as hardware failures, network outages, or data loss can disrupt project operations and lead to delays.
- 6. Cyber Attack:** Given the sensitive information stored by the app, it is a prime target for cyber attacks. Security breaches can have severe consequences.
- 7. Legal Issues:** Changes in regulations, licensing issues, or copyright infringements can pose significant legal risks to the project.
- 8. Staffing Shortages:** A lack of sufficient personnel can hinder the project's progress and affect its overall success.
- 9. Company Leaks:** The leakage of internal documents or source code can compromise the project's integrity and competitive advantage.

7.1 Risk Likelihood Ranking

The table below lists the potential risks in order of their likelihood of occurrence.

1	Communication Issues
2	Integration Challenges
3	Software Problems
4	Inadequate Planning
5	Infrastructure Problems
6	Cyber Attack
7	Legal Issues
8	Staffing Shortages
9	Company Leaks

7.2 Risk Impact Ranking

The following table ranks the risks based on their potential impact on the project.

1	Software Problems
2	Legal Issues
3	Company Leaks
4	Infrastructure Problems
5	Integration Challenges
6	Cyber Attack
7	Inadequate Planning
8	Staffing Shortages
9	Communication Issues

7.3 Combined Risk Ranking

This table combines the likelihood and impact rankings to provide an overall risk ranking for the project.

1	Software Problems
2	Integration Challenges
3	Infrastructure Problems
4	Legal Issues
5	Communication Issues
6	Inadequate Planning
7	Cyber Attack
8	Company Leaks
9	Staffing Shortages

8.0 Software Stakeholders

In this section, we will outline the key stakeholders involved in the Servicify project. Understanding the roles and interests of each stakeholder is crucial for ensuring effective communication and project success. The stakeholders are categorized based on their influence and impact on the project.

- 1. Project Owner:** The project owner is the highest authority, responsible for overseeing the entire project and making final decisions. Their role is crucial in setting the project's direction and ensuring its alignment with organizational goals.
- 2. Developers:** Developers are the individuals responsible for creating and maintaining the software. Their technical expertise and efforts directly impact the project's success. Failure of the software can lead to significant career repercussions for the developers.
- 3. Managers:** Managers oversee the project's progress, manage team dynamics, and ensure that project goals are met. They are heavily invested in the project's success.

and may have a stake in the company, making their role vital for project management and execution.

- 4. Restaurants & Other Vendors:** These stakeholders use the software to sell their goods and services. The success or failure of the project directly affects their revenue streams, making them a critical stakeholder group.
- 5. Users:** Users interact with the software, and their experience is paramount. Ensuring the security and usability of the software is crucial as it stores personal information, such as addresses and phone numbers.
- 6. Investors:** Investors provide the necessary capital for the project. Their financial stake in the project means that any failure can result in significant monetary loss.
- 7. Support Team:** The support team addresses user issues and serves as the bridge between users and developers. Their role is essential for maintaining user satisfaction and facilitating communication.
- 8. Couriers:** Couriers deliver services to customers and are paid based on their deliveries. The project's success or failure directly impacts their livelihood.

9. Project Needs

This section will outline the required project needs. These needs fall into three categories. These are: software needs, hardware needs and support needs. These are essential needs that need to be obtained in order to complete this project.

9.1 Software Needs

1. Docker: Docker is a software framework for building, running, and managing containers on servers and the cloud. It also streamlines the process of updating and deploying software on a server.

2. Kubernetes: This software manages server workloads and makes load balancing easier to manage. It also uses docker as a foundation thus making the program run in an isolated sandbox.

3. Ansible: Ansible is a computer configuration automation and management tool. System admins can write play books and run on the host machines to download necessary dependancies and software. This also makes updating and maintaining multiple servers easier by automating the process.

4. IDEs: Integrated development environments or IDEs are essential for developing software. We will use vs code as its support for plugins enable it to have support for a wide range of programming languages.

5. Katalon: Katalon is a test management tool that helps to streamline software testing processes. This tool is used by testers, developers, and team leads to manage, track, and organize software testing efforts.

6. Slack: Slack is a cloud based professional communication and collaboration tool that will be used to facilitate communication between members.

7. Microsoft Project: Microsoft Project is a time table creation and management tool. This will enable the team to plan and track their schedule.

8. Git: Git is a version control and source management software that we can use to track the changes that are done to the codebase.

9.2 Hardware Needs

1. Developer workstations: These general purpose computers will allow developers of the software to write code and test the software. It is preferable to have work specific machines that are separate from daily use, as they provide a layer of security of the source code as work computers are generally more secure.

2. Host mobile devices: These devices will allow users to interact with our program. These devices include: mobile devices, tablets, and etc.

3. Server Hardware: These devices can be around the world communicating with each other and other mobile devices. These computers are the most costly and important part of our hardware needs as most of the processing power required by our software will be handled by these machines. We will use Amazon as our service provider.

9.3 Support Need

1. Customer Support: Making customers feel safe using our app is essential. And in case of a problem they should be able to solve the problem. That is why customer support is paramount for this project. We have chosen Intercom, a customer messaging platform that provides live chat, to fill this need. As they are a customer support company, they will provide support for the foreseeable future.

2. Cloud Hardware Support: It is more cost effective to rely on cloud hardware providers than to rent space in a data center and populate it with our own hardware.

Thus it is important to be able to get support from the cloud hardware providers. We will be using the service provided by Amazon (AWS).

3. Financial Support: Having good financial advisors is important for this project, since the mismanagement of funds can cause extreme monetary loss during the lifetime of the project.

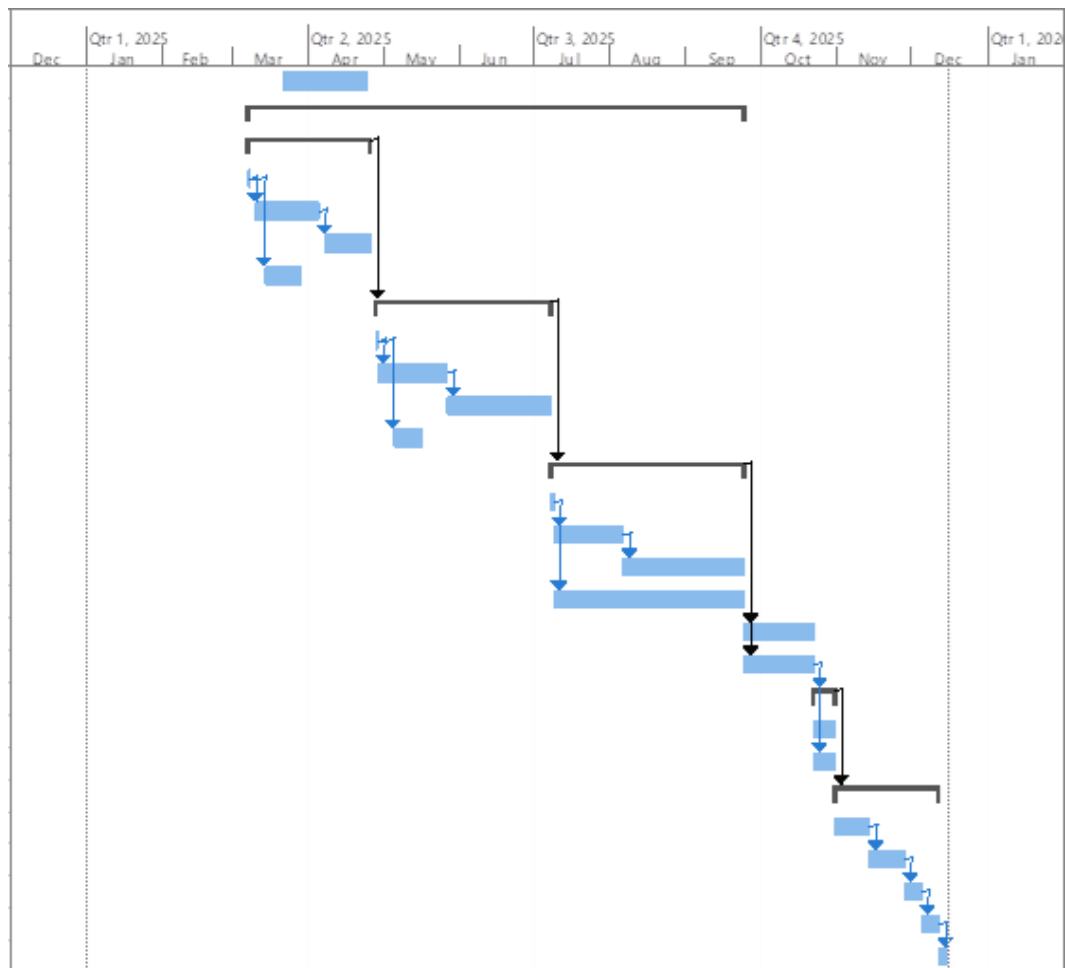
4. Legal Support: Having a good legal team is essential for the longevity of this project as there can be legal constraints that need to be considered. And a legal team is necessary in case of any potential litigation.





























5. User Support (User Feedback): Getting feedback regarding app usage and opinion from users are crucial as their satisfaction will make this project successful.

10. Project Schedule

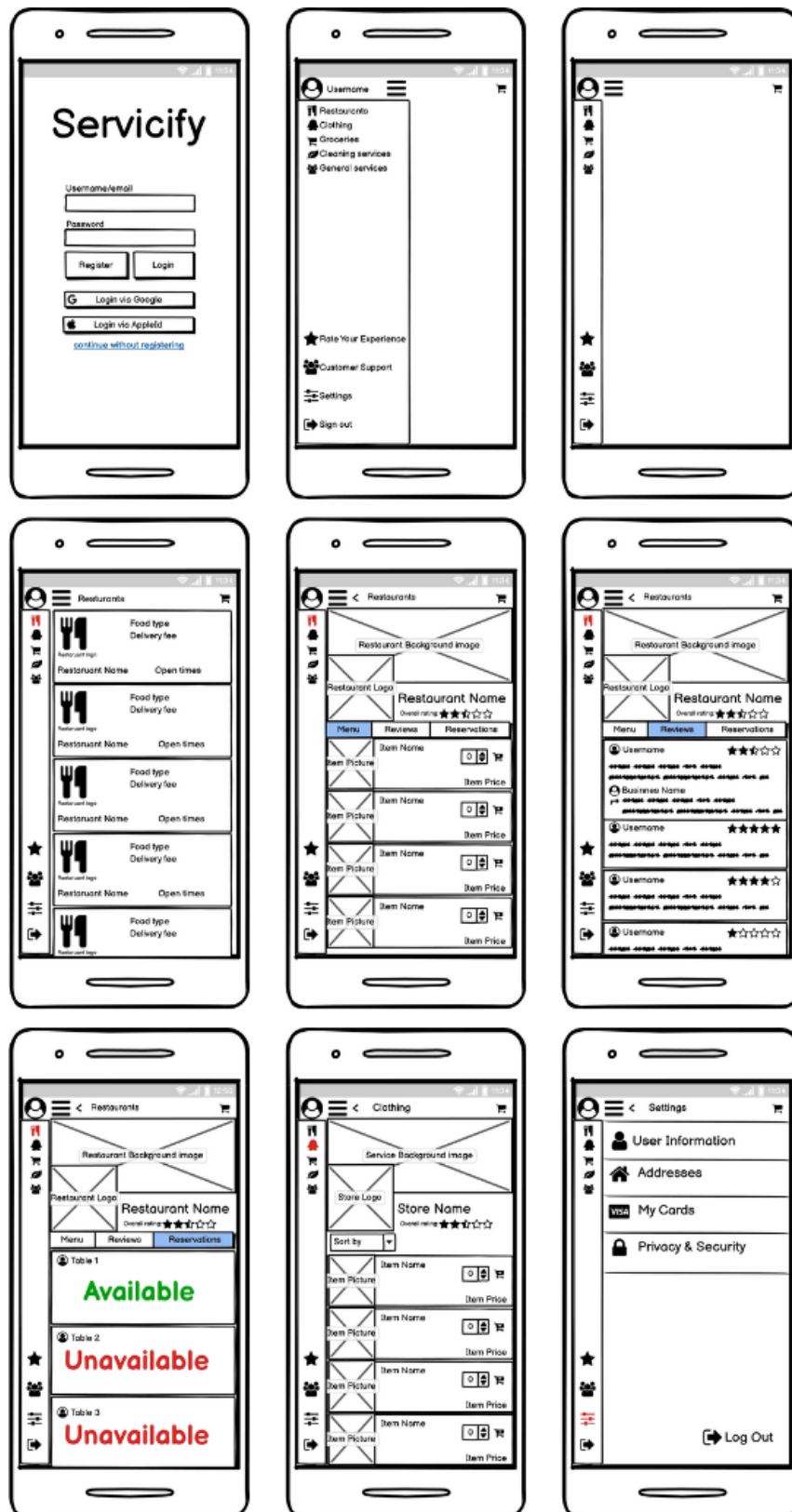
In software projects, a schedule consists of a list of project events with their respective start and finish dates. Scheduling is the most important part of the software development process. A good schedule can keep the team motivated while

keeping track of the project. Detailed project schedule and the gantt chart of the project is given below.



D	Task Mode	Task Name	Duration	Start	Finish	Predecessors
16		Testing	25 days	Fri 03/21/25	Thu 04/24/25	11FS+10 days
17		General Functionality	143 days	Fri 03/07/25	Tue 09/23/25	
18		Functionality for General service providers/markets	36 days	Fri 03/07/25	Fri 04/25/25	
19		Start General Service Providers/Markets	1 day	Fri 03/07/25	Fri 03/07/25	
20		Implementing service home page	20 days	Mon 03/10/25	Fri 04/04/25	19
21		Implement service addition functionality	15 days	Mon 04/07/25	Fri 04/25/25	20
22		Testing	11 days	Fri 03/14/25	Fri 03/28/25	19FS+4 days
23		Functionality for clothing stores	51 days	Mon 04/28/25	Mon 07/07/25	18
24		Start Clothing Stores	1 day	Mon 04/28/25	Mon 04/28/25	
25		Implement adding/removing items to cart	20 days	Tue 04/29/25	Mon 05/26/25	24
26		Implementing UI	30 days	Tue 05/27/25	Mon 07/07/25	25
27		Testing	10 days	Mon 05/05/25	Fri 05/16/25	24FS+4 days
28		Functionality for restaurants	56 days	Tue 07/08/25	Tue 09/23/25	23
29		Start Restaurants	1 day	Tue 07/08/25	Tue 07/08/25	
30		Implementing menu system	20 days	Wed 07/09/25	Tue 08/05/25	29
31		Implementing Reservation System	35 days	Wed 08/06/25	Tue 09/23/25	30
32		Testing	55 days	Wed 07/09/25	Tue 09/23/25	29
33		Implementing user rating system	20 days	Wed 09/24/25	Tue 10/21/25	28
34		Payment system	20 days	Wed 09/24/25	Tue 10/21/25	28
35		Deployment	7 days	Wed 10/22/25	Thu 10/30/25	34
36		Deploying software to cloud	7 days	Wed 10/22/25	Thu 10/30/25	
37		Deployment tests	7 days	Wed 10/22/25	Thu 10/30/25	34
38		System Testing	30 days	Fri 10/31/25	Thu 12/11/25	35
39		Security testing	10 days	Fri 10/31/25	Thu 11/13/25	
40		User Testing	10 days	Fri 11/14/25	Thu 11/27/25	39
41		Stress testing cloud servers	5 days	Fri 11/28/25	Thu 12/04/25	40
42		Final tests	5 days	Fri 12/05/25	Thu 12/11/25	41
43		Release	2 days	Fri 12/12/25	Mon 12/15/25	42

11. UI





12. Conclusion

We have explained everything that is needed to know about this project in this report. These are: Objective, software requirements, software process, software tools, software measurements, project risks, project stakeholders, project needs, project schedule and user interfaces are explained in detail. By following this project outlined in this report, a project manager can make this project into reality. This report will be very useful to the project manager if the project is followed accordingly.