Table of Contents

Section 1: Agile Methodology Implementation

Agile Methodology Overview

There were different elements in our management, especially the Scrum patterns that belong to the Agile methods. This makes it clear that due to the dynamics of the projects and the emphasis on returning value, the flexibility afforded by agile was an additional benefit. The above Agile principles of collaboration, customer feedback, and the idea of delivering working features in short cycles helped our team to effectively deliver a banking software system that can be used by everyone.

Teamwork and Agile Paradigm

Agile culture was the foundation of our team because every team member was aware of the goals and the main processes we were following. Some of the established practices that kept the team's communication transparent along with the Agile principles were daily scrums stand up, sprint planning meeting, and sprint retrospective sessions. Every single participant of the development team: developers, testers, and product owners were equally to blame to make sure that our project embraced Agile values.

Daily Stand-ups

The riot is also great to follow the progress and recognize barriers; it was conducted daily, the so-called stand-up meeting. In these meetings, the team members reported their activities, for instance the work on the deposit with and without draw features and issues encountered. This practice was instrumental in ensuring that problems were solved as they occurred, hence the continuity of the project. These stand-ups also allowed for the management of issues and decision-making in real time, thereby increasing the amount of solidarity and efficiency of the team.

Sprint Planning Sessions

During our sprint planning meetings we identified the user stories from the product backlog. Some of the user stories that emerged in the Banking application domain were things like creating and configuring new banking accounts, modifying the details of the accounts as well as incorporating optimal and secure user authentication procedures. While developing the user stories, each of the stories was split into task user stories, and the teams were assigned to team members depending on the team members' availability, qualification, and domain expertise. This made sure that every sprint had well defined goals and the work load was evenly distributed. That is why, by defining the appropriate ambitious goal and tangible outcomes for each sprint we might check how effective we are and improve the process if necessary.

Sprint Retrospectives

It also means that Sprint retrospectives were performed on the end of each Sprint within which parties identified what went well, what could be better, and what should be done for the subsequent Sprints. The scheme in (a) maintained a continuous feedback loop that was essential in defining problem areas

and adopting new measures that enhanced the organization's efficiency and performance levels. Such retrospectives helped to encourage free-speaking and constant improvement, which paved the ways for the team's growth.

Evidence of Agile Practices

Sprint Reviews and Stakeholder Surveys

During the end of every sprint, we held sprint review meetings in order to present progress and ingredients to the stakeholders. For example, it is important to demonstrate how to use filters for date, type, and amount of transactions in the transaction history feature. Seventeen stakeholders were interviewed aiming at getting feedback to incorporate into subsequent sprints such that the developed software meets the users' expectations. These reviews helped in maintaining the stakeholders' interest and keeping them informed as well as helped us in measuring our work against their specifications.

Incremental Delivery

This strategy restricted us to releasing functional increments of the software at regular intervals. It allowed the key stakeholders to try it out and see for themselves how the software works, which proved helpful for the refinement of the product based on users' requirements. Every increment had fully working and complete features so that every version that was deployed was fully working and complete to meet the client's needs.

Distribution of the tasks and the extent of its completion.

Besides, daily stand-ups, we incorporated different Agile tools concerning the distribution of the tasks and their implementation. That is why instruments such as Jira allowed us to plan out the work for the respective period, track the rates of implementation, and make sure that all the participants are moving in the direction of achieving the objectives and goals set for the sprint. Very strict communication was observed to have done with every team member being answerable for his/her assignment and everybody tracking time to see that the work was done effectively and efficiently as required.

Reflection on Alternatives

Although scrum was the main framework used, some of the related and competing Agile processes include the Kanban system and the Extreme Programming (XP). Kanban is attractive since the work continuously moves through the system, and task management is based on visualization, but we were looking for strictly defined iterations with start and end dates, which wasn't convenient for Kanban. Of course, we did not see the continuous delivery model of Kanban as matching our timely feedback and the closing of milestones.

This was done using Extreme Programming (XP) which has its focus in engineering practices like pair programming among others, test-driven development as well as continuous integration. However it was more applicable to the situations where there are large number of code builds and fast iterations. To that end, because our project was feature-driven and had a goal to deliver a most efficient and friendly banking software system, Scrum's structure that pays more attention to the process of delivering several features incrementally was more suitable for us.

Therefore, adopting Scrum allowed reaching our goal to create a complex product and coordinate large teams and numerous interactions to deliver the value continuously. Staying true to the Agile concept and using the Scrum approach, it was possible to implement a system of banking software that would address the users' needs and satisfy stakeholders' expectations.

Section 2 – Role During Development

Through the transition of building software system for banking process, i had taken multiple roles in the team but my role was multifaceted which involved many responsibilities and contributions.

1. Scrum Master:
I got an opportunity to be a Scrum Master which was a person responsible for enforcement of Agile principles, methodology and practices in all phases of the project. It encompassed delivering the meetings of sprint standups, sprint planning and reviewing, as well as retrospectives on a daily basis. Being a Scrum Master, I played a role of a servant-leader, freeing the team from the bottlenecks, boosting teamwork, and improving the quality of work done.

2. Requirement Analysis and User Story Refinement:
I have been a contributor in the requirement gathering process, which involved the collection and definition of the project needs. I helped with the user stories' refinement, splitting them into simpler and actionable tasks and making sure they were well written and attainable within a sprint. Also, my team mutually collaborated along with stakeholders to rank the backlog by business value and user needs.

3. Frontend Development:
I was indeed being part of the development team and I mainly concerned about frontend development tasks that built a useful and good-looking interface for banking software. These platforms include creating and working of adaptable pages of window application developed with C# window forms. I worked with UI/UX professionals to make sure the frontend design was in synch with users' expectation and usability standards.

4. Backend Development:
Beyond developing a front-end interface, I was involved in the backend efforts and these were in developing the financial transactions and account functionalities. I developed the APIs running Node.js

and Express.js, which help in making RESTful APIs, and authenticating and authorizing users. I also integrated the database to store and retrieve user data with the application securely.

5. Testing and Quality Assurance:

I had an active part in penning automated tests covering the frontend as well backend layers articulating the Test-Driven Development (TDD) methodology. As a part of this, it was necessary to write unit test, integration tests and end-to-end-tests to achieve the reliability as well as the robustness of the software. Moreover, I bore witness to the QA testers with whom I worked on applying manual testing, bug identification, and the verification that the software matched the acceptance criteria.

6. Documentation and Knowledge Sharing:

I keep a record of all my development projects' system architecture, design options, and coding guidelines throughout the total development cycle. I also lent a hand in creating usability instructions and technical documentation so that the regular users and other colleagues can aboard and enjoy support without any hindrances. Besides, I also engaged with other employees in interchange sessions, giving out my own knowledge, best ways of carrying out tasks and lessons learned.

7. Continuous Improvement Initiatives:

Within my role, I have included step by step collaboration processes, which have been implemented and carried out by the team. The capacity building program comprised of organizing workshops on Agile practices, conducting technical sessions of knowledge-sharing and implementation of process improvements commencing on the feedback and retrospectives conducted. Through the establishment of a learning culture and developing procedures for improving them on a continual basis, we have managed to optimize the working process and get high-quality software on the short terms.

Summarily, mine position required a mix of leadership, technology, teamwork and persistent goal towards delivering values to end users by continuous improvement and iterative development.

Section 3: Project Planning

In regards to our banking software system project we are applicable of a structured approach to project planning which has as objective the accentuating the importance of the clearly defined objectives, specific activities and these which are able to be attained. We were going to use Agile Methodology in our project management process, Scrum would be our framework. Here's how we approached project planning and broke down the work into activities and milestones: Here's how we approached project planning and broke down the work into activities and milestones:

1. Reasoning for Choosing Agile Methodology (Scrum):

Flexibility:
Scrum methodology brings in agility that can promptly react to the shifts in the developing conditions, new needs and priorities. This was very important for us when the bank software project was being

worked at, since sometimes requirements could change based on what customers are saying and also demands of a market.

Iterative Development:

Agile champions an incremental approach to development that means we roll out parts of our software at the end of every iteration. Adopting this method helped us garner initial feedback from the followers of the product and constantly enhanced the product according to their demands

Collaboration:

Agile methods put a high emphasis on collaboration between cross-cutting teams, therefore, supporting transparent communication and building a common goal in teams. With all of the various factors involved and complicated nature of our project - collaboration has proved to be great for its completion.

Customer Focus:

In Agile customer happiness becomes one of the top priorities by delivering successful succumbing value fractions by fraction. This statement really echoed our main goal of designing a user-friendly banking software done with care which fulfills the demands of them.

2. Breaking Down Work into Activities and Milestones:

Product Backlog Creation:

We commenced with developing a product backlog that had specific user stories referred to the numerous features and related functionalities that characterized the banking software system. Through of user stories that were arranged based on business value and customer needs.

Sprint Planning:

During preparation for each sprint, sprint planning meetings take place where user stories are selected from the product backlog to be finished. An estimated task size per user story was allocated to us and we intended to deliver as much work we were able to achieve in a sprint.

Daily Stand-up Meetings:

The daily meetings where we discussed the accomplishments of the day before, the difficulties we faced and the to-do tasks for the day helped us out with getting stuff done. Consequently, these the brief meetings were the team remained aligned, improved communication and ultimately to recognize early on any dilemmas that could hinder the progress.

Development and Testing:

In the middle of the sprint, team members executed the chosen user stories, in compliance with the highest development standards, like TDD and pairing programming. Tests were carried out manually to guarantee that code quality and consistency would not be compromised.

Sprint Review and Retrospective:

We held sprint review meetings at the end of each sprint, during which we presented all the completed items to stakeholders and usually invited their feedback. This feedback was then turned into a new update for the backlog and further updated for the planning of the next sprint. Next, we ran the sprint retrospective meetings to consider our approach, reveal the steps of improvements and ultimately make any edits.

Incremental Delivery:

In the course of the project, our main purpose was always to ensure a steady value and relevance by releasing fragments of the app at the end of each sprint. These opportunities gave us a sense of how to validate our guesses, get feedback, and implement these alterations even before the full development cycle has begun.

The Agile method of project management paradigm was used to break up work into tasks and deliverables. In this manner, we ensured a structured and iterative approach to planning and execution of the project. The good thing about this was that it made our work smoother, and even the processes that we did were easily adapted to. We were able to do this because we were all about team work and was always doing trial and error, which was the recipe to the success of running the banking software system.

Section 4: Prototype Design

Description of the System:

The banking software system prototype is a fully-featured platform for use in the banking sector with the goal of increasing operations efficiency. The system is equipped with simple user interface, and powerful back-end capabilities to help do multi-banking processes in a secure and convenient manner. Here's an overview of the key components and features: Here's an overview of the key components and features:

1. Account Management:
- People may mull over and, in the end, open new banking accounts, account types may also be personal and business accounts.

- This data can be altered e.g. user info or account type. It depends on what data the content contains.
- Option of Closure is available and the step is well define checked and authenticated.

2. Financial Transactions:
- Users can put cash into their accounts, including a printer option that is automatically generated for the users to have a proof.
- Recharge of account with the necessary card or digital wallet while getting real-time balance update is an important convenience that our system provides in order to keep transactions accurate.
- The funds are easy to transfer between various accounts owned at the same bank.
- The history of transactions is executed, the option which enables users to filter transactions by date, type, and amount is available to them thus helps in better tracking and analysis.

3. Account Services:
- For user's convenience, this information consists of all account types, such as: current balance, recent transactions, and account type.
- The system offers an ability to users to keep in touch with all services provided and get debit/credit cards and checkbooks as they wish, whenever they want.
- Recurring payments and direct debit management don't involve any problems recurring payments are arranged and managed smoothly.

4. Security and Compliance:
- Strong authentication mechanism involving systems like two-factor authentication are to be applied to make sure that accounts are accessible only to their legal owners.
- Data security is strengthened by encrypting confidential details to meet the data regulations.
- Our Token System follows financial administration and compliance rules to ensure the system is fair and trustworthy.

5. Automated Testing:
- Most important part of our development process is automated testing; it measures the quality of the developed software.
- To validate the functionality of the system, integration tests, unit tests and end-to-end tests are written.
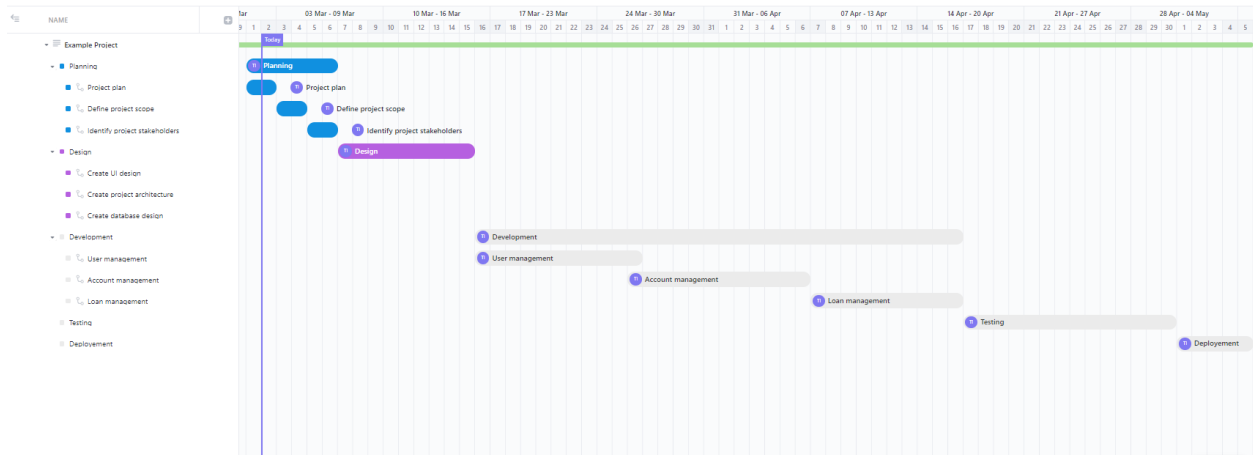
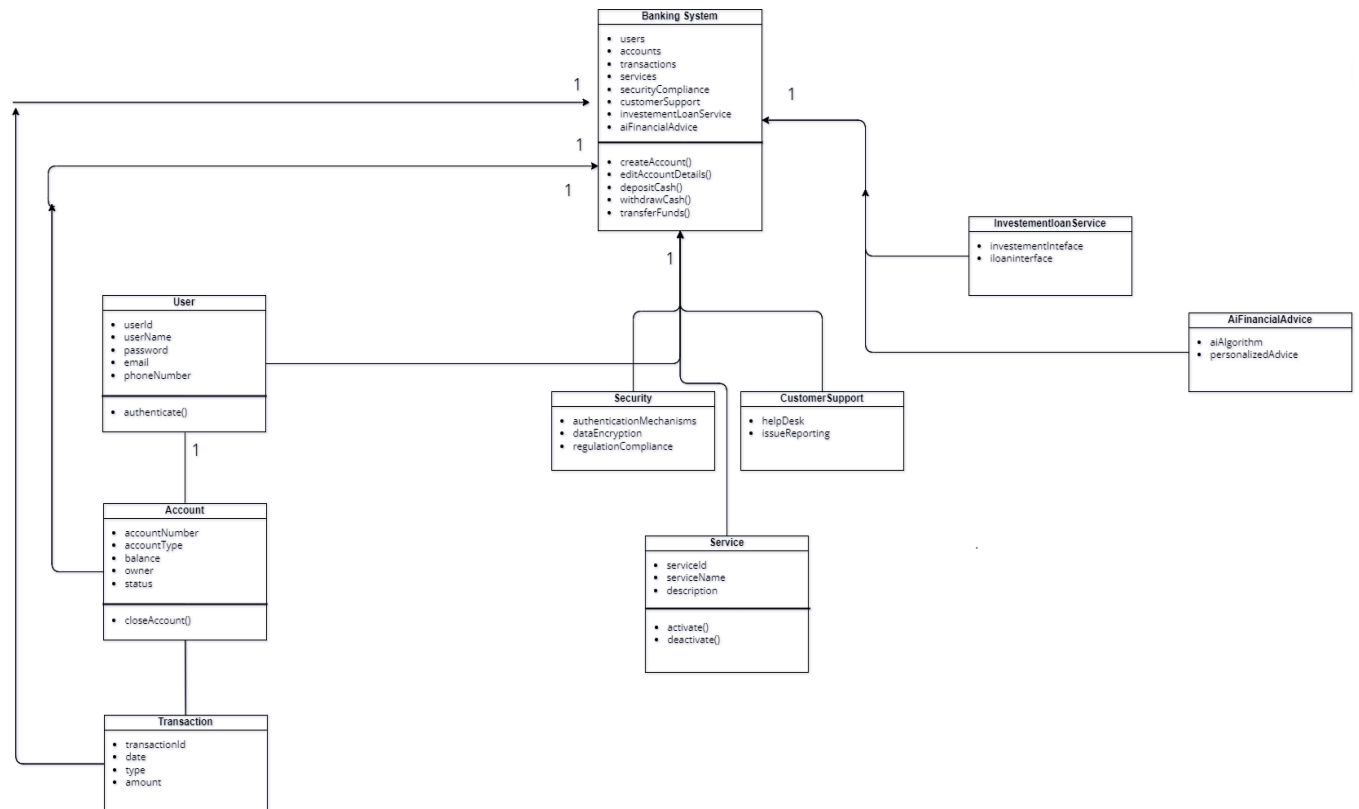*Figure 1. Gantt Chart*

6. System Design:

Figure 2. Class Diagram
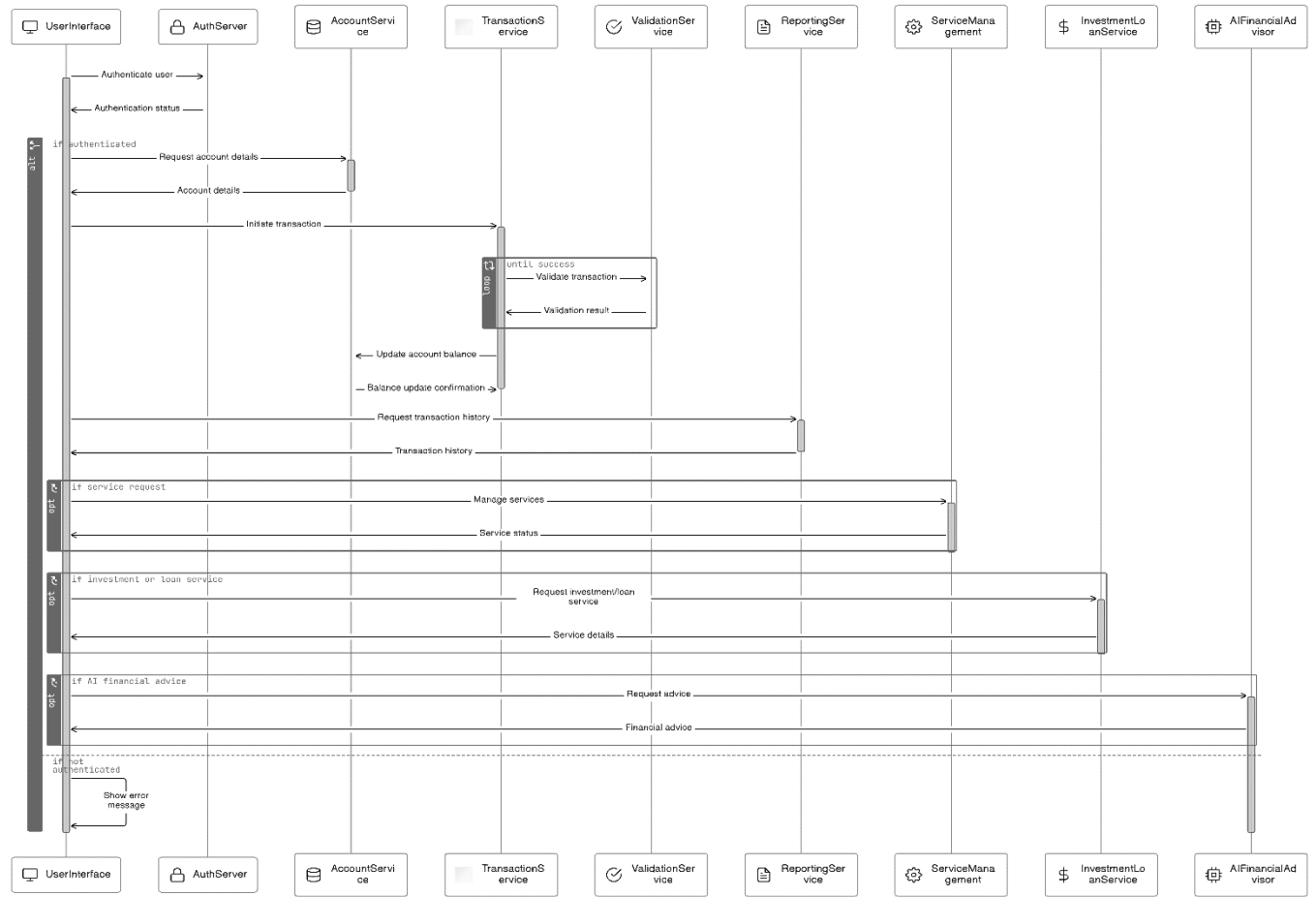
Banking System Workflow

UserInterface	AuthServer	AccountService	TransactionService	ValidationService	ReportingService	ServiceManagement	InvestmentLoanService	AIFinancialAdvisor

Authenticate user →

← Authentication status

alt [if authenticated]

Request account details →

← Account details

Initiate transaction →

loop [until success]

Validate transaction →

← Validation result

← Update account balance

→ Balance update confirmation

Request transaction history →

← Transaction history

opt [if service request]

Manage services →

← Service status

opt [if investment or loan service]

Request investment/loan service →

← Service details

opt [if AI financial advice]

Request advice →

← Financial advice

[if not authenticated]

Show error message

Figure 3. Sequence Diagram

Banking System ERD

transactionHistory
| transactionId | string | pk |
| details | | string |

transactions
transactionId	string	pk
type		string
amount		number
date		timestamp
accountId		string
recipientAccountId		string

accounts
userId	string
accountType	string
balance	number
accountId	string	pk

accountServices
| accountId | string |
| serviceType | string |

security
| userId | string |
| authenticationType | string |

users
userId	string	pk
name		string
email		string
phoneNumber		string

Figure 4. Entity Relation Diagram

7. Application Layout
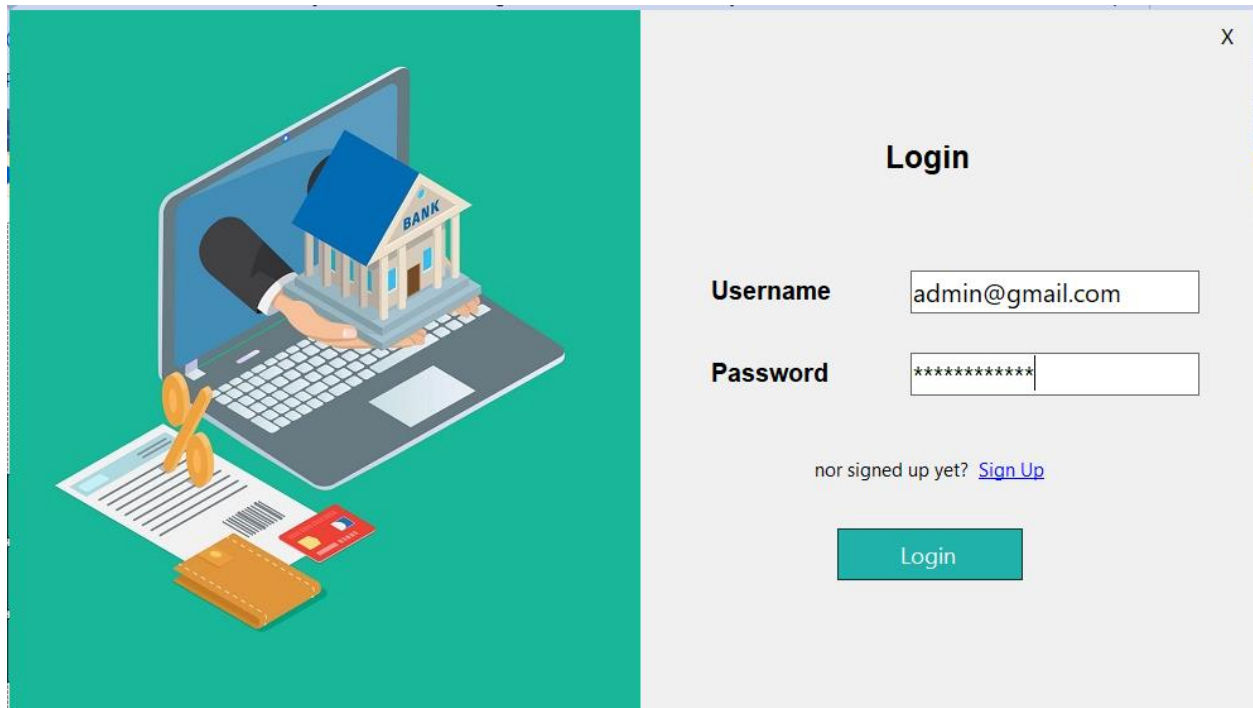
Figure 5. Login Screen

Figure 6. Add new user
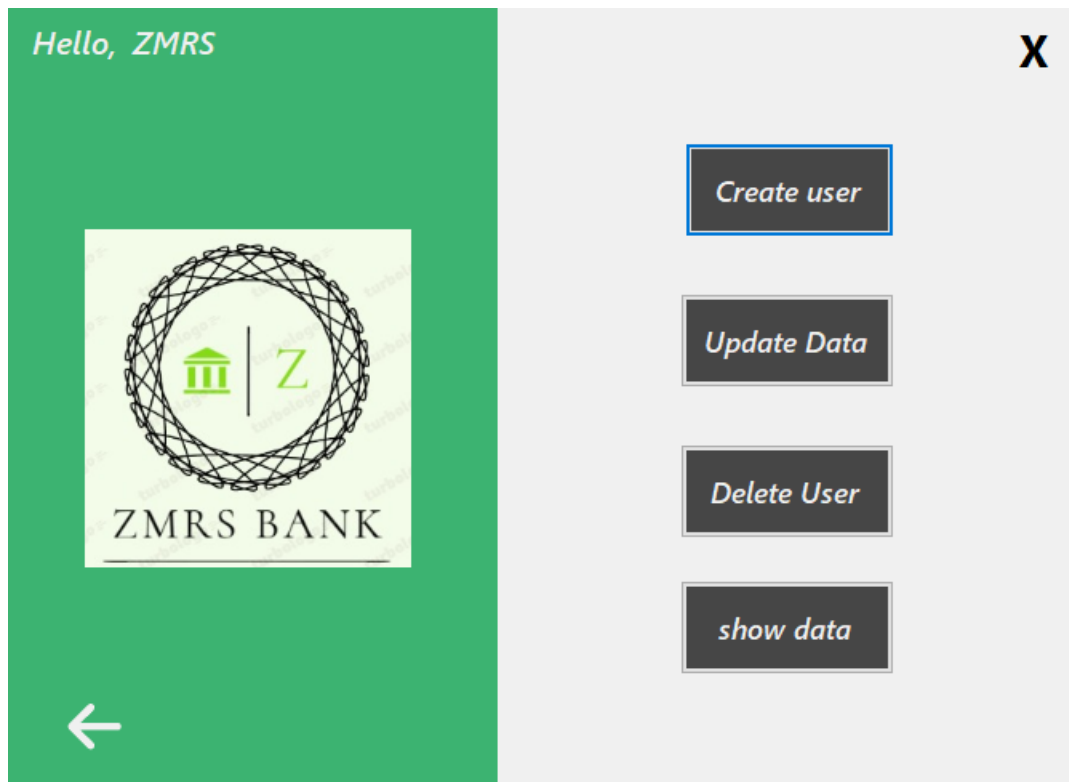
Figure 7. Admin Interface
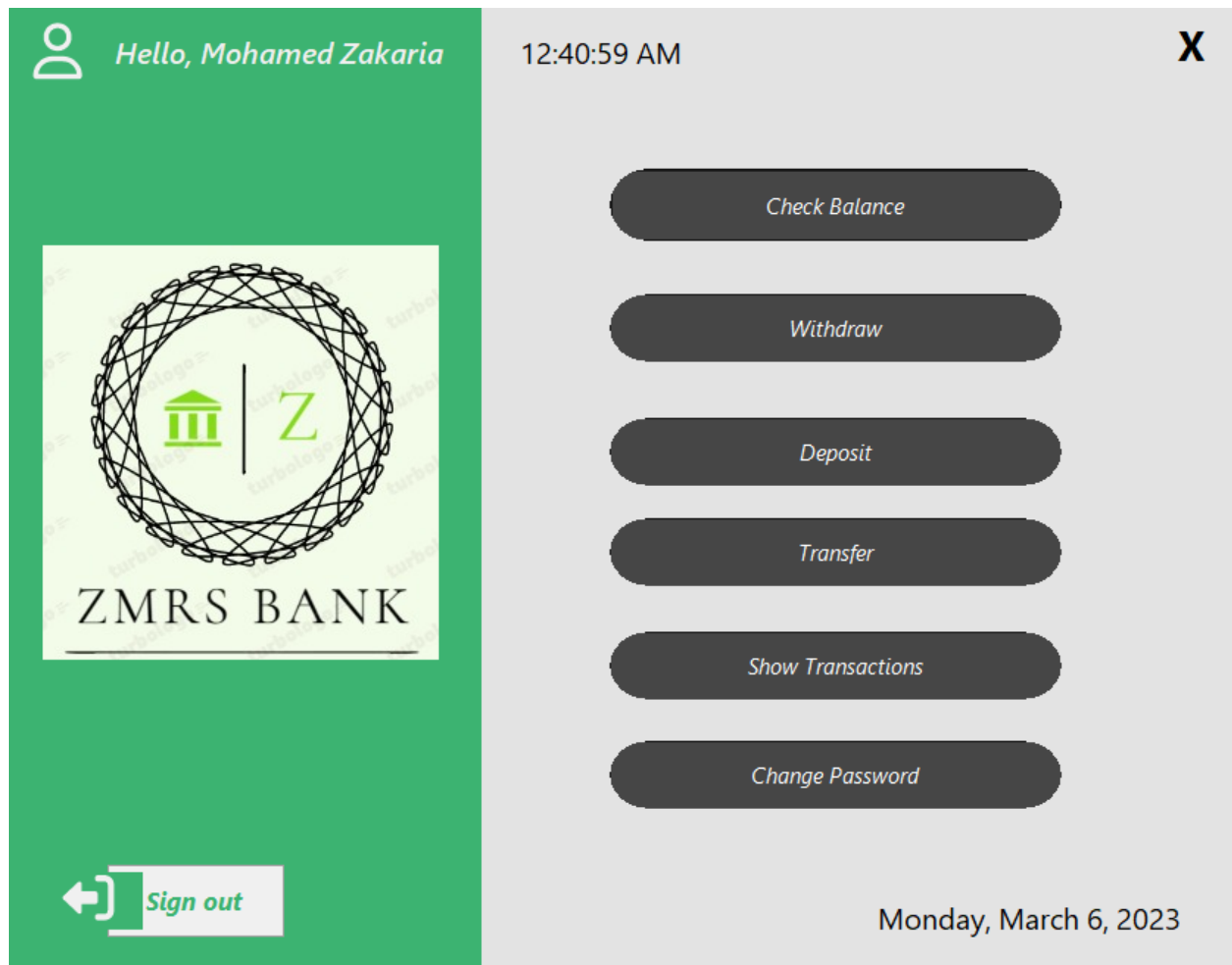
Figure 8. User interface

Id	operation	balance	amount	date
1	Withdraw	800	14	1/9/2023
1	Deposit	1800	1000	1/9/2023
1	Transfer	1600	200	1/9/2023
1	Deposit	2000	400	1/9/2023
1	Withdraw	16065	600	1/17/2023
1	Deposit	16465	400	1/17/2023
1	Deposit	17065	600	1/17/2023
1	Withdraw	1900	100	1/9/2023

Figure 9. Transaction Table

Figure 10. Transfer funds interface

Section 5: Critical Evaluation of Software Engineering Tools and Techniques

a. Advances in Software Processes:

i. Methodologies:

The Agile methodology, on the other hand, is behind all development process in our banking software product. We followed Agile principles of iterative development, collaboration, and flexibility and thus were able to keep up with the requests related to changing requirements and offer the value incrementally as our team worked progressively.

Evidence: We used the Scrum technique, therefore, our activities involved sprint planning, daily stand-up meetings, sprint reviews and retrospec-ti-ves. This enabled us to examine functionality from the point of view of various users, react to the feedback from the testers, and release the software in small stages on a regular basis.

ii. Agile Processes:

Agile methodology allowed us to fix the bugs and other issues in a short span of time, thereby us being responsive to the customer needs and market demands that result in software development with customer centric as an approach.

Evidence: The customer-inclusive approach of our Agile process was achieved through constant engagement with stakeholders, collecting feedback, and adjusting priorities and plans to best suit their needs and values. Through this rigorous and participatory approach, the software was ensuring its feasibility and efficient response to the needs of the users in regards to the software updates.

b. Software Engineering Techniques:

i. Project Management:

Project management skills that guaranteed timely execution of the tasks within the stipulated space and time was critical for coordinating all these (i.e. the tasks, resources, and timelines). Among the toolkit were project boards and task management programs that served as a window to seeing through the team and communicating effectively.

Evidence: These were enabled by tools such as Jira and Trello which were used for tracking the sprint progress, managing the product backlog, and assigning tasks to the team members. Thanks to the support of this framework, we were always staying on the versatile way, dividing high priority tasks by themselves and allowing every feature within agreed upon deadlines.

ii. Prototype Design:

Prototyping enabled us to create a user interface and user experience blueprint and then go through iterations and refinements so as to have a full-scale software system development. Prototyping tools helped executing fast feedback loops and provided comfort communication between designers, developers and stakeholders.

Evidence: Our wireframes and mockups were done using prototyping tools, such as Adobe XD and Sketch, which give us an opportunity to create various designs, interactive parts and to approach different workflows. Role of stakeholders was pivotal in the process of reviewing and contributing with useful perspectives for the revisions.

iii. Version Control:

Undoubtedly, the version control system, for instance Git, were crucial in maintaining the code consistency, collaboration with a common team, and unfailing code integrity. Payments or loans were made in a matter of clicks instead of waiting for days or even weeks during the traditional banking process.

Evidence: Version control was implemented through Git, where a feature branch is created for new developments and merged with the main branch after the code reviewing and testing was accomplished. This feature allowed us to achieve a variety of abilities, such as tracking changes, reverting to previous versions, and providing some much-needed help to geographically distributed teams.

GitHub repository Link: https://github.com/talhaiqbal-lincoln/BankSystem-resit.git

c. Impact of Advanced Software Systems and Software Engineering:

Social Impact

The system of banking software that we have designed, key values and priorities: increasing the level of financial literacy of the population guaranteeing the availability of banking products and services. Understanding the versality of the users, we focused our efforts on the friendly interfaces and mobile banking apps development. These endeavours were meant to cover the knowledge deficit on all its users, irrespective of the level of computer literacy, so as to facilitate easy usage and management of money.

In this case, we made it easier for those with banking deficiencies to attain them through the use of our software; hence, making the banking system more efficient. For example, aspects mentioned as Account Management, and Transaction interface were developed to ease the usage of the site/to-portal, depending on users' needs and enable those needs despite potential lack of formal financial services. This approach not only made banking operation to be easier and also made the user to be more involved in their operation.

Ethical Impact

The questions of ethical nature were at the core of the considerations both when designing our software system and during its implementation. In order to contribute the trustworthiness of the application

among users, we considered the protection and security of data as well as their compliance with legal standards as a top priority. Proactive and profound security features like secure authentication methods and encrypted approaches were practiced to protect the users' data from being hacked and violated.

Moreover, our software had some protective measures aside technical ones that covered GDPR and other regulations on the protection of data. We explained to the users on how their data would be processed and ensured that they agreed to it. These were not only ethical practices, but essential for the retention of our software's integrity to users and other stakeholders.

Entrepreneurial Impact

So, as an enterprise, our banking software system delivered commensurate value since it stimulated innovations as well as improved the performance of the involved financial institutions. Thus, adding up features and options like real-time transaction processing, customizable front-end, and advanced security tools, we created conditions under which banks and other similar financial institutions could stand out on the market and address a wider range of clients' needs.

Our software enabled the various financial institutions to achieve better operational performance, better customers' satisfaction and rapidly address market conditions. For instance, innovations such as the automated recurring payments, account management among others made it easier for both the banks and their customers to achieve improved value in the delivery of services and increased customers' retention.

Furthermore, it was possible to note that flexibility of our software matched the high level of financial institutions' capacities and let them satisfy the changing demands of the market and develop new technologies. Through the enhancement of efficiency of digital banking solutions' implementation, our software allowed banks to timely respond to new vistas in the developing more digital-oriented field and remain leaders.

Thus, our banking software system not only served important social objectives, which are contributing to the improvement of the general public's financial knowledge and financial inclusion but also ensured a manifest ethical standard through implementing strict policies of personal data protection. Also, it favored relevant financial institutions with innovative and competitive instruments and advantages that enhance the industry development and customers' satisfaction. This approach ensured that the requirements of our developed software system responded not only to the current market needs but also to future needs and trends for the same clients.

Section 6: References

[1] Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9), 833–859. https://doi.org/https://doi.org/10.1016/j.infsof.2008.01.006

[2] J. Yuonan and R. Mamedov, 'Agile Project Management in Banking : A study of how agile methods are modified to suit the context of a bank', Dissertation, 2020.

[3] Vejseli, S., Proba, D., Rossmann, A., & Jung, R. (2018). The agile strategies in IT Governance: Towards a framework of agile IT Governance in the banking industry. *Twenty-Sixth European Conference on Information Systems (ECIS 2018)*, 1–17. http://ecis2018.eu/wp-content/uploads/2018/09/1840-doc.pdf