Table of Contents

Section 1: Agile Methodology Implementation

Our team considerably used Agile practices starting from the inception and development of our banking software platform throughout the system's life cycle. The principle of agile methods is to develop iteratively, collaborate, have flexible approach, and get feedback from the customers. Here's how we

applied Agile principles and XP (Extreme Programming) techniques in our project: Here's how we applied Agile principles and XP (Extreme Programming) techniques in our project:

1. Scrum Framework:

We implemented the Scrum approach, one of the widely used Agile techniques, to manage our work moments for the sprint. At that time, as a rule, we had the sprint of 2 weeks including the plan phase, development, and testing of the individual conclusions of all the tasks during that period. Daily stand-up meetings were the principle collaborative instrument which ensured everyone was on track and made it possible to discuss accomplishments, issues, and army marches.

2. User Stories and Backlog Management:

Our product backlog split into user stories which stranded for certain functionality or feature of their perspective for user. Those stories were considered in virtue of their significance to both end user and the business value. During sprint planning we prioritized user stories by drawing them randomly from the backlog and incorporated the high-priority features to be addressed first.

3. Continuous Integration and Continuous Delivery (CI/CD):

We shall establish a CI/CD pipeline that will be handy in the automation of building, testing, and deployment of processes. It would be possible for us to take which is the last sprint through the software at the end of each sprint. Trying any changes to the code, it will be run for testing and merged to the main branch and this ensures that the software automatically gets updated and stable at all times.

4. Pair Programming:

The practice of pair programming, which is part of the Extreme Programming, was utilized on our side: two developers were working together on one task simultaneously. It shortened knowledge internalization course, prevented mistakes and elevated code quality as well. Pair programming as well worked in favor of constant communication and collaboration among the members of the team, this helped achieve high quality final solutions and made the problem-solving processes very fast.

5. Test-Driven Development (TDD):

All of the tests have been developed using the test-driven development approach which means that we made the automated tests before implementing any new functionality. These proofs actually run alongside the code so that apart from being documentation they also act as validation of the fact that it conforms to the given specifications. We prefer to write and maintain our test-first tracks code so that we obtain reliable, extendable, and maintainable result.

6. Iterative and Incremental Development:

Our process of being agile and our development was incremental and iterative with each sprint being a potentially receiver deliverable. We, therefore, worked to connect with all of our stakeholders and end-users at this stage as well as collecting their thoughts which contributed to possible future improvements. Through continuous and ongoing work on the software we have over time made sure that it perfectly fit the ever-evolving needs of our customers.

7. Regular Retrospectives:

On each of our sprints, we did twice a retrospective. This was a process of reflecting on our progress and what we needed to do to improve. Now with this experience, we know how to trim the product development process of all hassles and speed up the development speed. In addition, the experience gave us both an understanding of how our idea can be improved and fine-tuned in further stages and has paved the way to other aspects where we can advance.

In the end, therefore, the Agile framework and XP ways of doing things that we used, and not on time and in the deadlines. This product was of a very high quality as well as met with the users and stakeholders requirements as a results of implementing some of the value which included; collaboration, flexibility and the lean development approaches.

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.

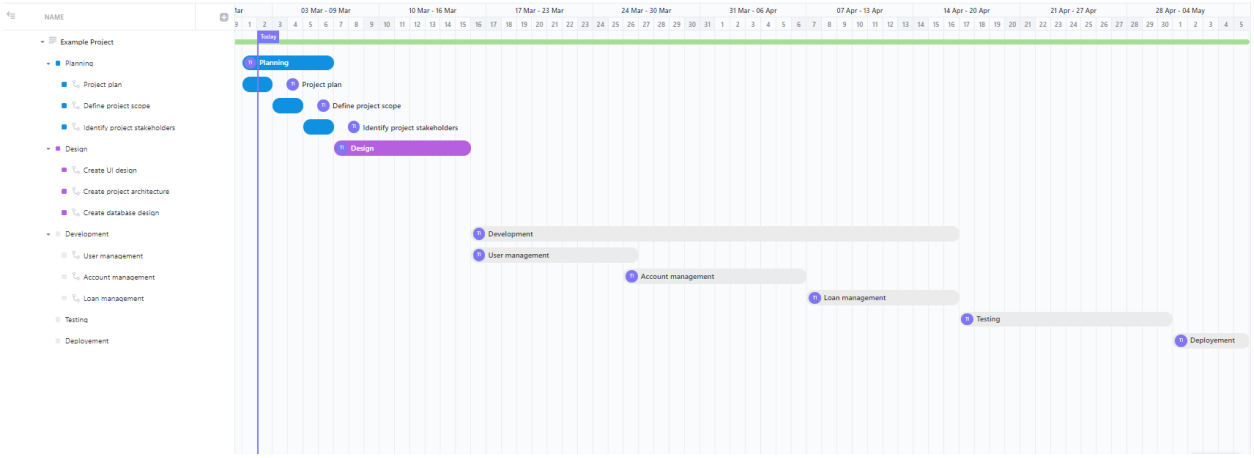


Figure 1. Gantt Chart

6. System Design:

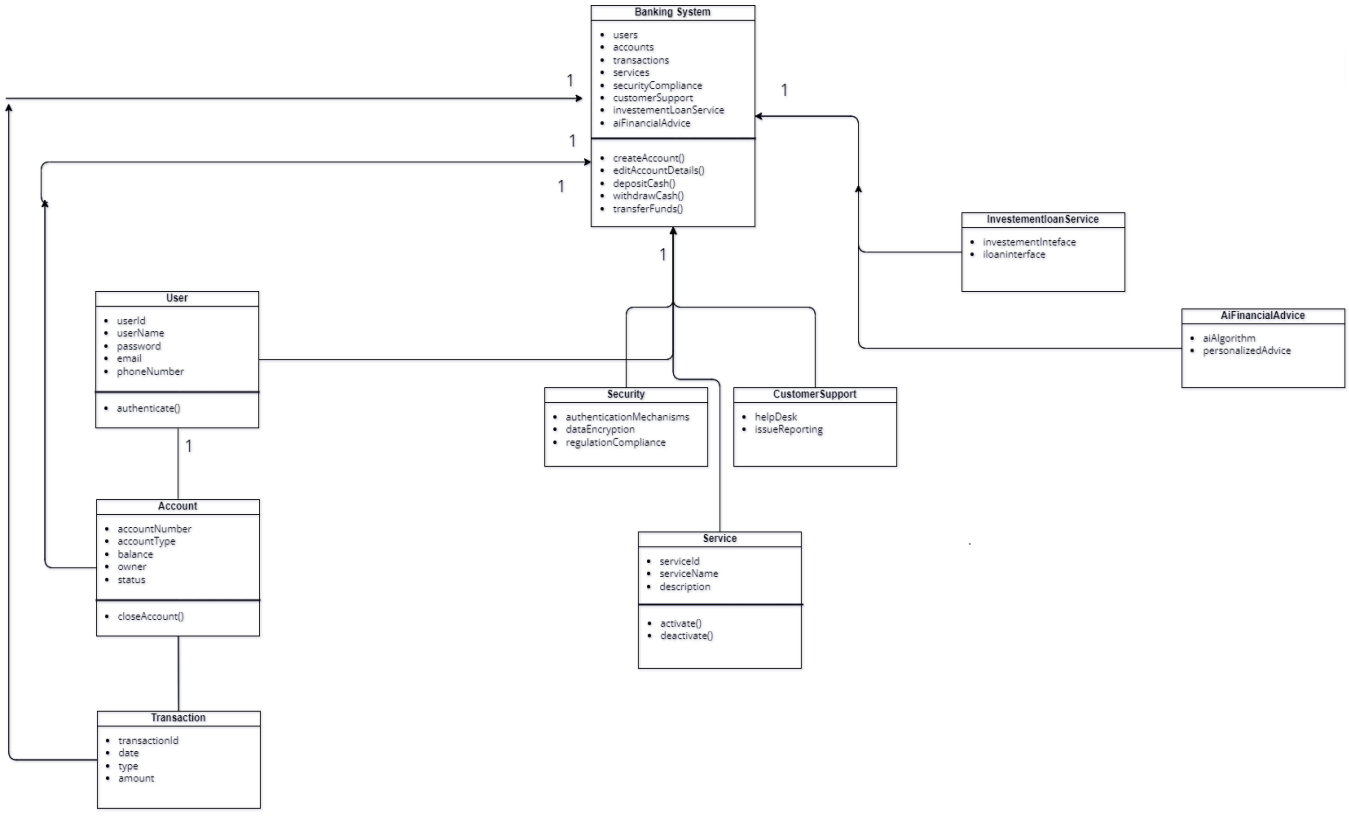


Figure 2. Class Diagram

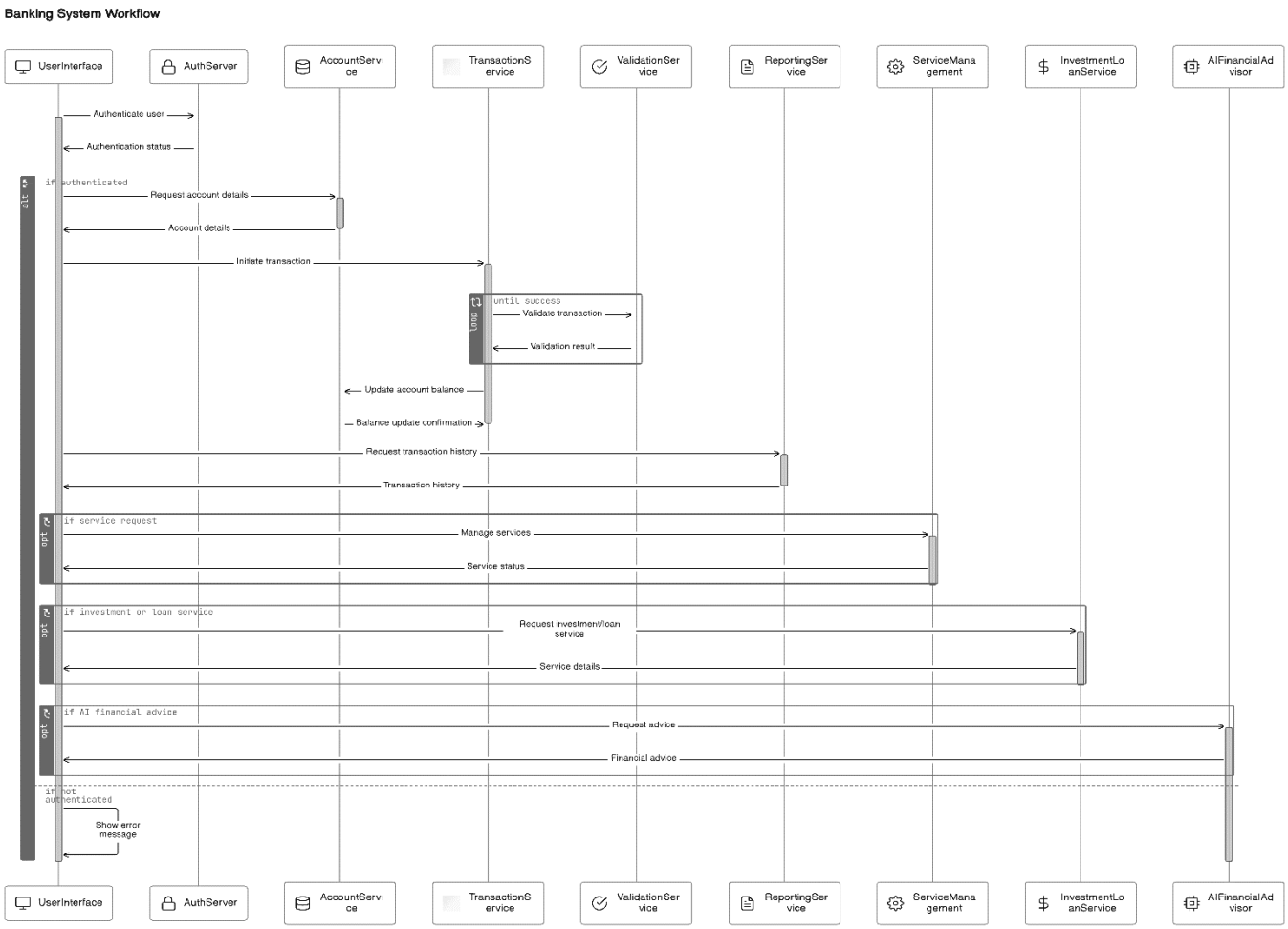


Figure 3. Sequence Diagram

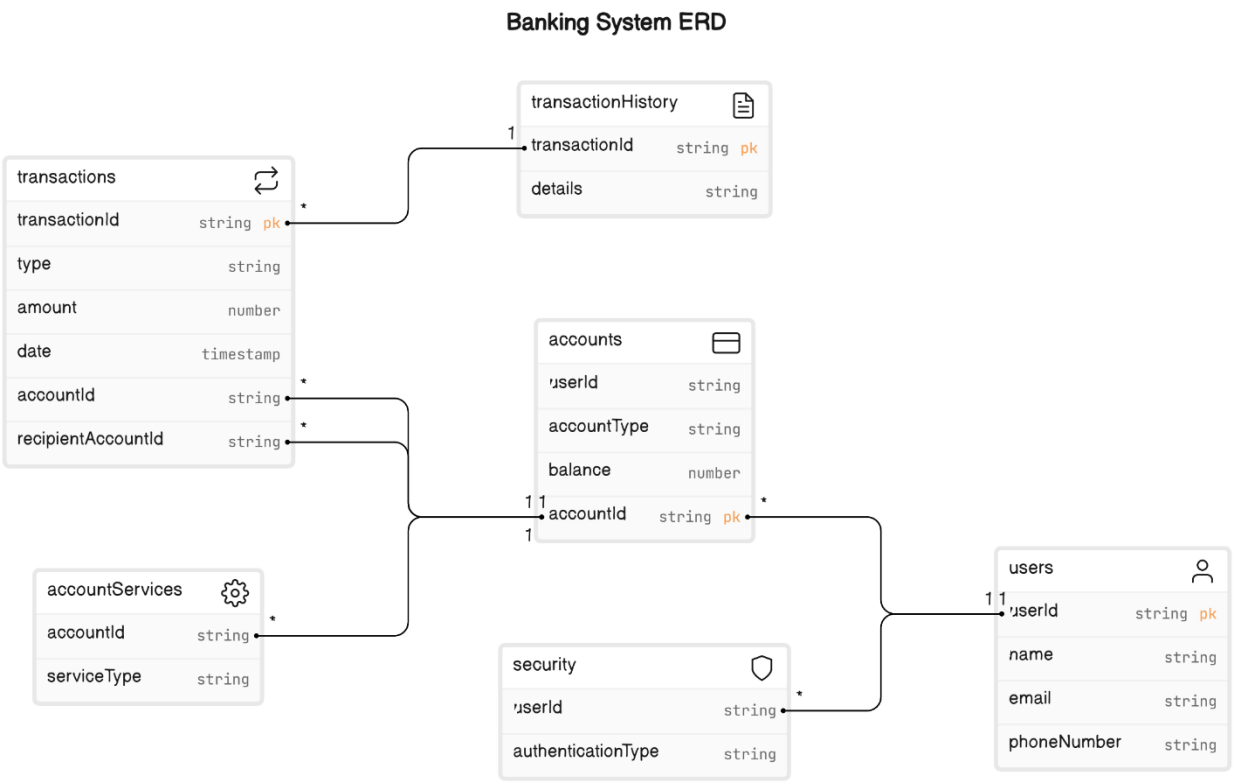


Figure 4. Entity Relation Diagram

7. Application Layout

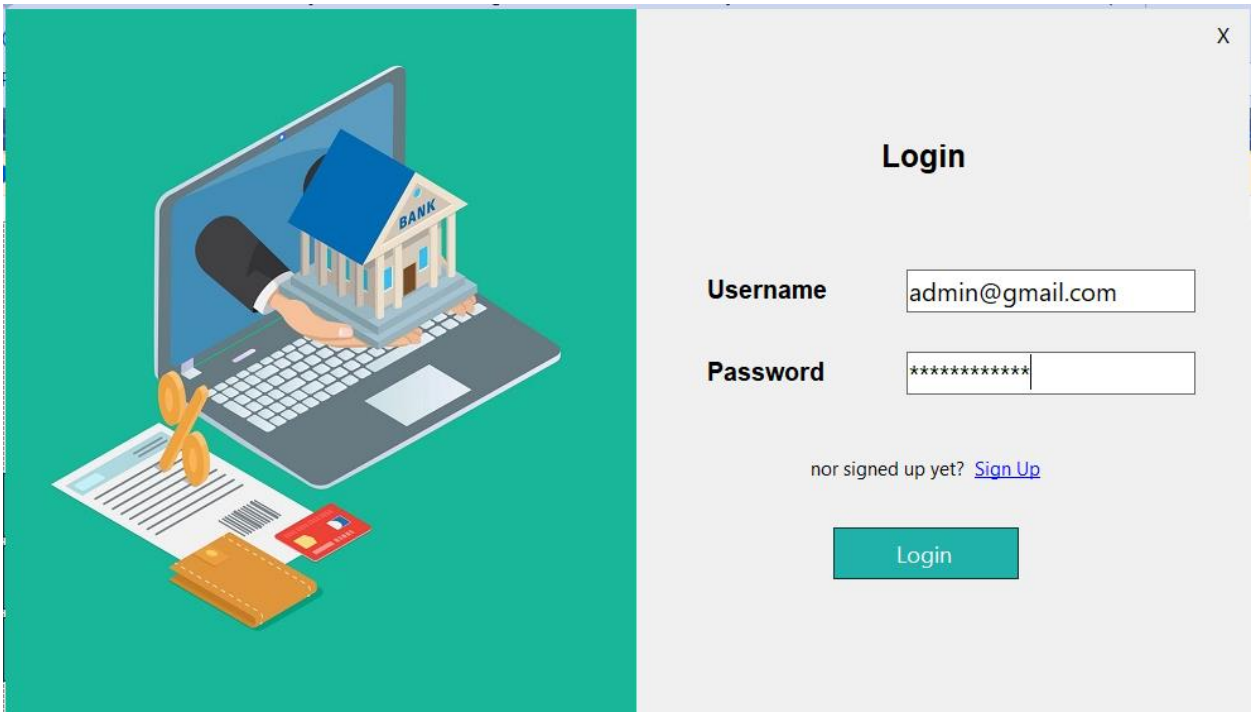


Figure 5. Login Screen

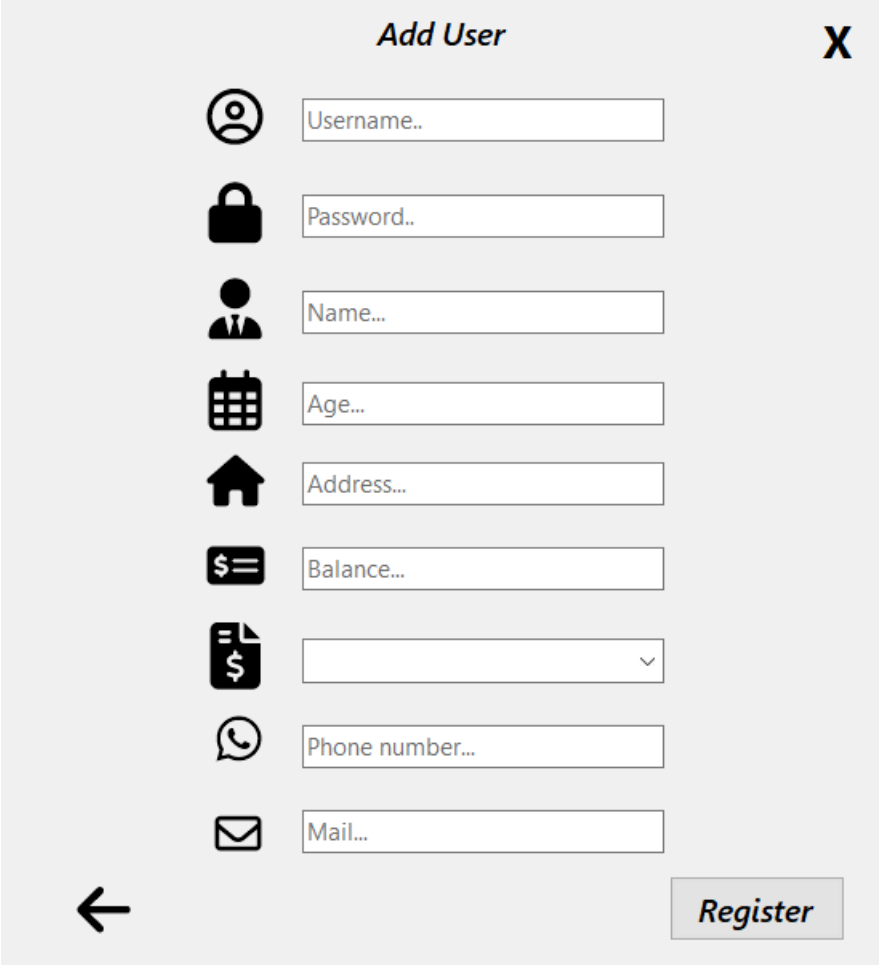


Figure 6. Add new user

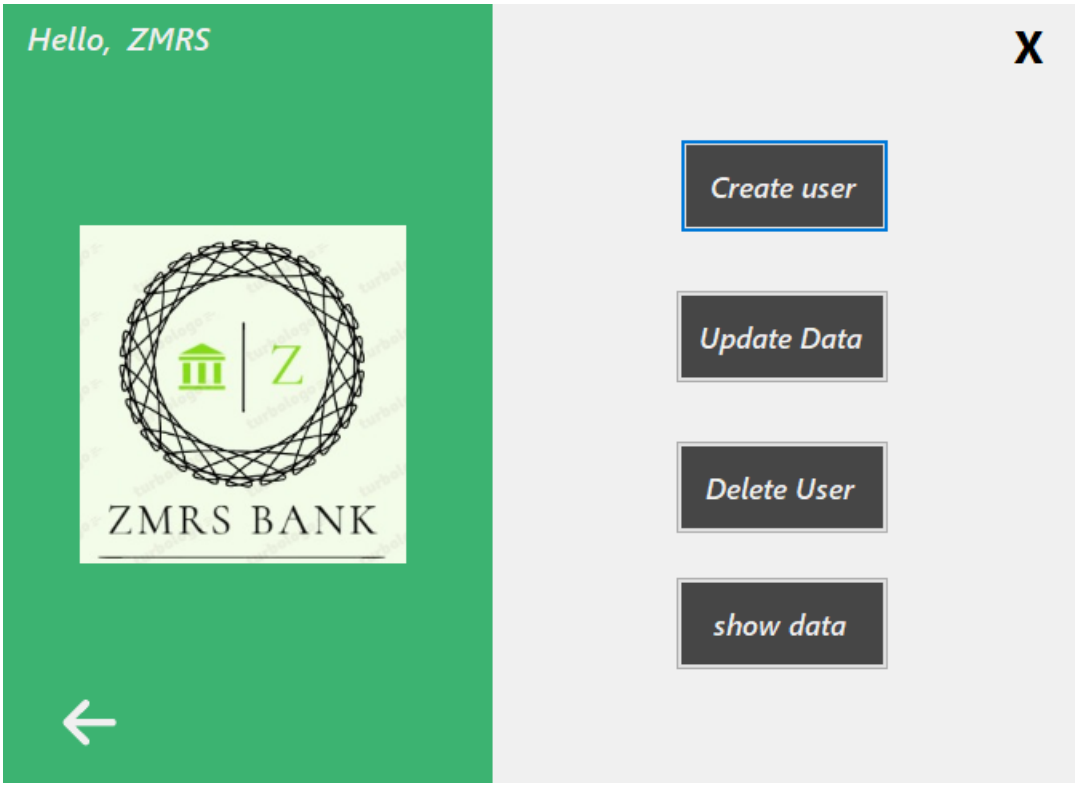


Figure 7. Admin Interface

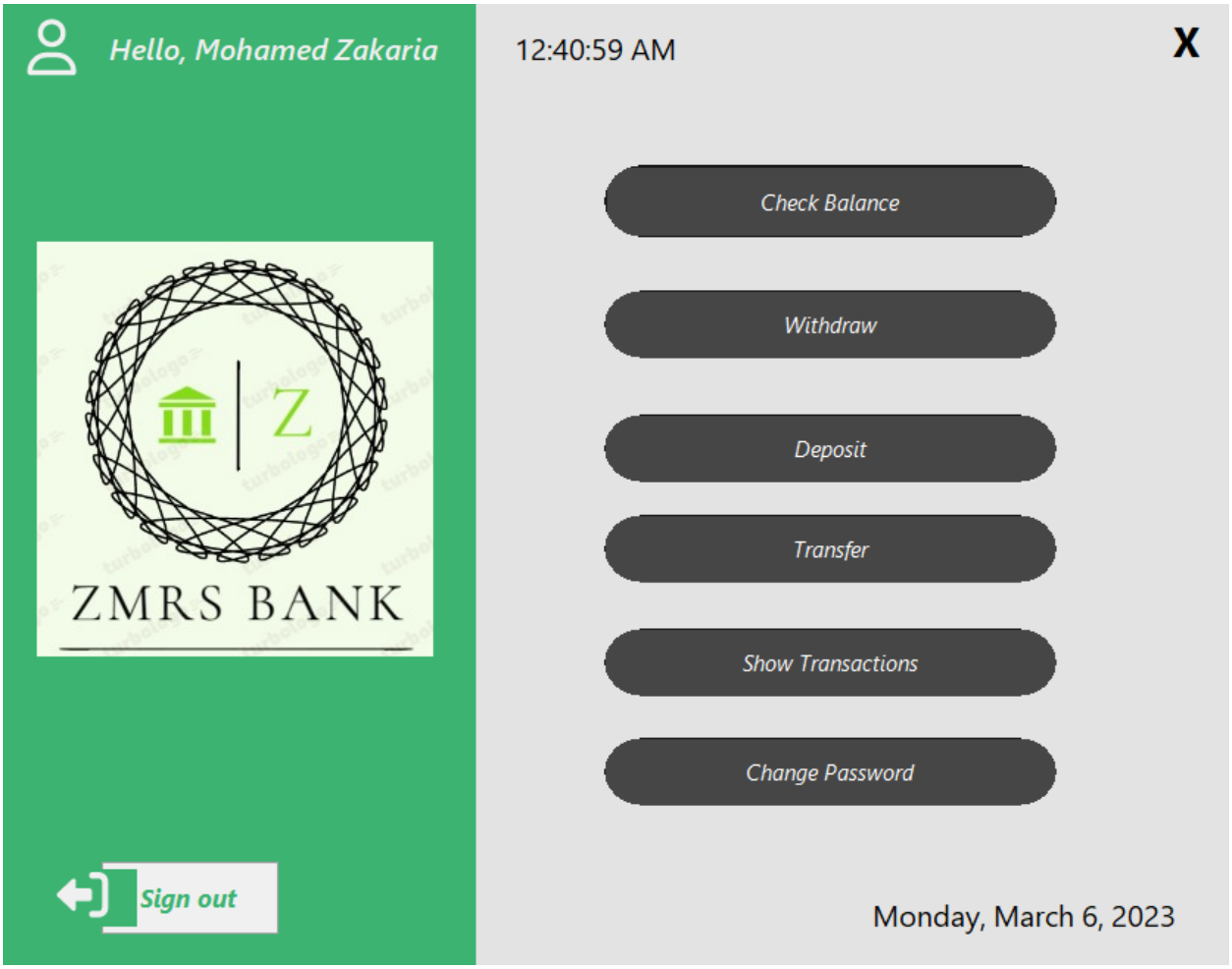
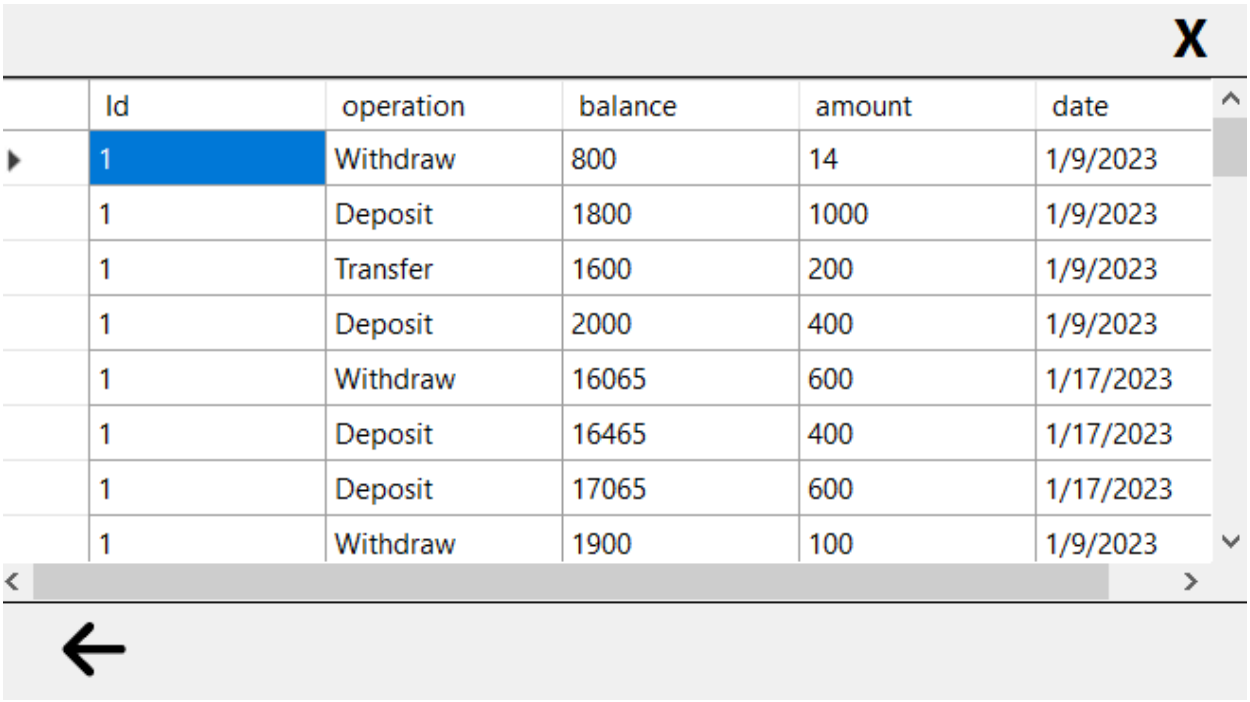


Figure 8. User interface



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

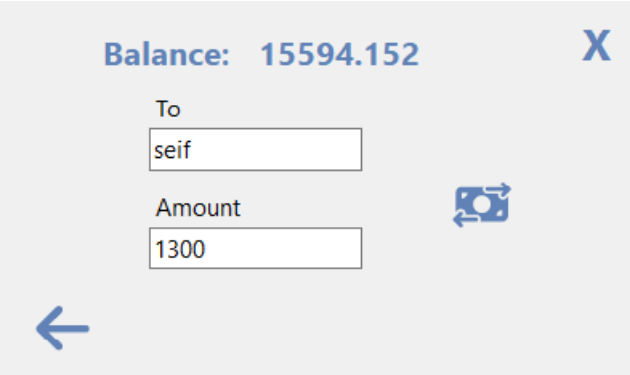


Figure 10. Transfer funds interface

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/talhaiqballincoln/BankingSystem.git

c. Impact of Advanced Software Systems and Software Engineering:

i. Social Impact:

The banking software mechanism is playing a crucial social part for the clients by offering them convenient and comprehensive banking service facilities regardless of geographical barriers and physical disability. Digital banking not only inclusivity at mobile access to banking functions worldwide but empowerment through its digital channels also.

ii. Ethical Impact:

The ethical factors concerning the protection of privacy, security and fairness in the software development and operations are extremely important for banks, among other things. We put in place rigorous security measures, encrypted the sensitive data, and kept with the laws, which made sure that the user trust and the private data are preserved.

iii. Entrepreneurial Impact:

The advent of sophisticated applications within software systems including our ("our") banking application of entrepreneurship is a door that strengthens the potential for startups. Innovation that entails reaching

for existing banking architectures via cutting-edge technology can equally breed value, income, and also address the neglected place in financial services industry. In addition to providing them with a platform where the small and medium-sized enterprises can come up with fintech startups and compete in the digital banking environment.

Synthesizing the implementation of the sophisticated software engineering tools and approaches where Agile framework was mainly weaved in produced the finished product. The wide-ranging effects of these Agile processes were not only in terms of process management and quality control but also had the most significant social, ethical, and entrepreneurial consequences that revolutionized people's interaction with online banking systems.

Section 6: References

[1] Bewak, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Guidelines for Sustainable Software Development. Agile Alliance. Retrieved from https://agilemanifesto.org/

[2] Cockburn, A. (2002). Agile Software Development: Simulation Game "The Cooperative Game" (2nd edition) Addison-Wesley.

[3] Ambler, S. W. (2002). Agile Modeling: Successful strategies that embrace iterative and incremental development (Extreme Programming) as well as integrate many aspects of development (the Unified process). Wiley.

[4] Pressman, R. S.; and Maxim, B. R. (2015). Software Engineering: Practitioner's Perspective (8th version). McGraw-Hill Education.