**Report**

**Table of content**

[**Section 1: Description and evidence of agile methods and XP techniques** 3](#_Toc162818217)

[**1.1. Agile methodology selection:** 3](#_Toc162818218)

[**1.2. Utilization of agile techniques:** 3](#_Toc162818219)

[**a)** **User stories and planning discussion:** 3](#_Toc162818220)

[**b)** **Simple design and pair programming:** 4](#_Toc162818221)

[**c)** **Testing and continuous integration:** 4](#_Toc162818222)

[**d)** **Roles and responsibilities:** 4](#_Toc162818223)

[**e)** **Documentation and retrospectives:** 5](#_Toc162818224)

[**Section 2: Role, tasks, and contributions** 6](#_Toc162818225)

[**2.1: Role assignment:** 6](#_Toc162818226)

[**2.2: Task and contributions:** 6](#_Toc162818227)

[**2.3: Contributions chart:** 7](#_Toc162818228)

[**Section 3: Project planning** 8](#_Toc162818229)

[**3.1. Reasoning:** 8](#_Toc162818230)

[**3.2. Gantt chart:** 8](#_Toc162818231)

[**3.3. Work breakdown structure:** 10](#_Toc162818232)

[**3.4. Agile artifacts:** 12](#_Toc162818233)

[**Section 4: Prototype Design** 12](#_Toc162818234)

[**4.1. Description of the system:** 12](#_Toc162818235)

[**4.2. System design:** 12](#_Toc162818236)

[**4.3. UI designs:** 15](#_Toc162818237)

[**4.3. Version Control:** 16](#_Toc162818238)

[**4.4. Developed system:** 16](#_Toc162818239)

[**4.4. Testing:** 23](#_Toc162818240)

[**Section 5: Critical evaluation** 23](#_Toc162818241)

[**5.1. Advances in software processes:** 24](#_Toc162818242)

[**5.2. Software engineering techniques:** 24](#_Toc162818243)

[**5.3. Societal impact:** 25](#_Toc162818244)

[**Section 6: References** 25](#_Toc162818245)

# **Section 1: Description and evidence of agile methods and XP techniques**

# **1.1. Agile methodology selection:**

For this project we chose to adopt the scrum framework as our agile methodology. Scrum’s iterative approach allowed us to break down our project into manageable chunks and deliver incremental value to the stakeholders. We held regular sprint planning, review, and retrospective meetings to ensure continuous improvement throughout the development process.

# **1.2. Utilization of agile techniques:**

The following agile techniques were utilized during the development of this project.

# **User stories and planning discussion:**

Table 1: User stories

Extensive planning discussions, to identify user stories and prioritize them based on business value were conducted. The following table contains the list of the user stories created during the planning of this project.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **User Story Description** | **Priority** | **Acceptance Criteria** |
| US1 | As a customer, I want to be able to create a new banking account (Personal/Business) so that I can start using banking software system. | High | * The system should provide options for creating personal or business accounts. * Users should be prompted to provide necessary account details. |
| US2 | As a customer, I want to be able to edit account details (e.g., account information, account type) so that I can keep my account information up to date. | High | * Users should be able to access and modify their account information. * Changes made to account details should be reflected in the system. |
| US3 | As a customer, I want to be able to close my account so that I can terminate my relationship with the bank if needed. | High | * Users should have the option to initiate the account closure process. * Confirmation should be obtained before closing the account. |
| US4 | As a customer, I want to be able to deposit cash with receipt generation so that I can add funds to my account securely. | High | * Users should be able to select the deposit option and enter the amount to be deposited. A receipt should be generated for the transaction. |
| US5 | As a customer, I want to be able to withdraw cash with real time balance update so that I can access my funds conveniently. | High | * Users should be able to initiate a withdrawal request and specify the amount to be withdrawn. * The system should update the account balance accordingly. |
| US6 | As a customer, I want to be able to transfer funds between different accounts so that I can manage my finances effectively. | High | * Users should be able to select accounts for transfer and specify the amount to be transferred. * The system should update balances for both accounts accordingly. |
| US7 | As a customer, I want to be able to view transaction history with filters (date, type, amount) so that I can track my financial activities. | Medium | * Users should be able to access a transaction history page and apply filters based on date, type, and amount. * The system should display relevant transactions based on applied filters. |
| US8 | As a customer, I want to be able to view detailed account information (current balance, recent transactions, account type) so that I can manage my account effectively. | Medium | * Users should have access to an account dashboard displaying current balance, recent transactions, and account type information. |
| US9 | As a customer, I want to be able to request and manage services like checkbooks and debit/credit cards so that I can access additional banking features. | Medium | * Users should have options to request additional banking services like checkbooks and debit/credit cards. * The system should process service requests and update user accounts accordingly. |

# **Simple design and pair programming:**

We emphasized simple design principles to ensure the maintainability and extensibility of our codebase. Additionally, we practiced pair programming during coding, allowing team members to collaborate closely and share knowledge effectively and efficiently. GitHub was used for version control and commits by different accounts during the development of this project are the evidence of pair programming.

# **Testing and continuous integration:**

Test-driven development (TDD) was employed to ensure that each feature implemented met its acceptance criteria. Automated unit tests were written for critical components such as database class which held the responsibility of managing the MySQL database, and continuous integration was employed to test each component on development. In the section 4 the detailed evidence of automated testing is mentioned.

# **Roles and responsibilities:**

Each team member took on multiple roles throughout the project, including customer, designer, tester, and programmer, based on their skills and the needs of this project. This flexibility allowed us to maximize our productivity. The following table shows the roles fulfilled by each team member.

Table 2: Roles fulfilled by different team members

|  |  |
| --- | --- |
| **Name** | **Roles** |
| Amaara Arshad | Planner, Designer, Programmer |
| Amina Abdulhaq | Customer, Designer, Programmer, Tester |
| Maryam | Planner, Designer, Programmer |
| Uzma Sajjad | Planner, Designer, Programmer, Tester |
| Aqsa Faraz | Customer, Designer, Programmer, Tester |

# **Documentation and retrospectives:**

We maintained detailed documentation of our sprint planning, user stories and task breakdown in Trello. Additionally, we conducted regular sprint retrospectives to reflect on our process and identify areas for improvement in subsequent sprints. The figure 1 shows a screenshot of the Trello board, showcasing the planning discussion and assignment of different activities to the team members.

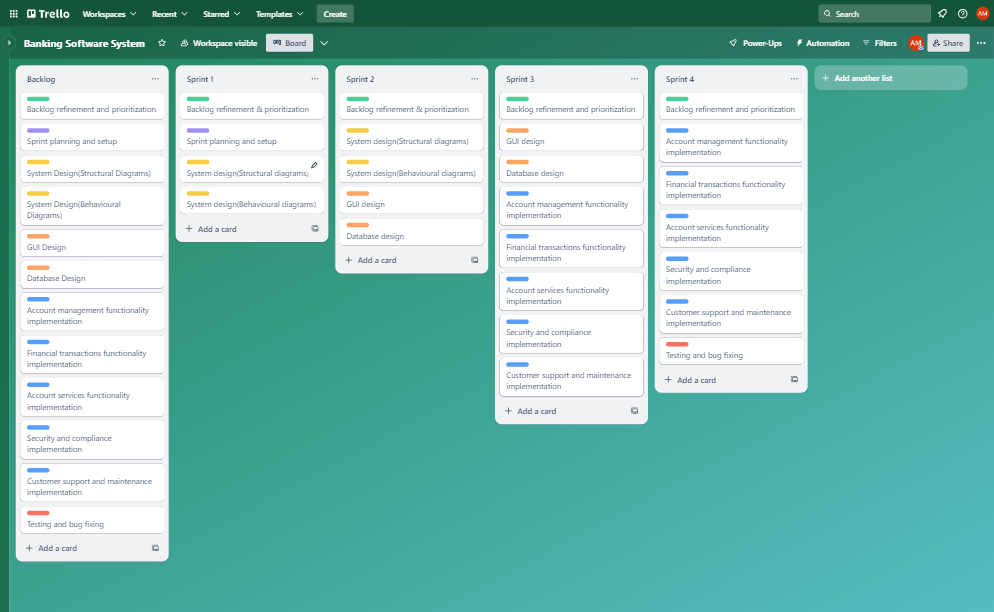


Figure 1: Screenshot of Trello Board.

# **Section 2: Role, tasks, and contributions**

For this project multiple roles were assigned to each team member. Each team member undertook multiple tasks and contributed to the completion of the project.

# **2.1: Role assignment:**

The following roles were assigned to each team member.

Table 3: Roles assigned to each team member.

|  |  |
| --- | --- |
| **Team member** | **Roles** |
| Amaara Arshad | Planner, Designer, Programmer |
| Amina Abdulhaq | Customer, Designer, Programmer, Tester |
| Maryam | Planner, Designer, Programmer |
| Uzma Sajjad | Planner, Designer, Programmer, Tester |
| Aqsa Faraz | Customer, Designer, Programmer, Tester |

# **2.2: Task and contributions:**

The tasks and contributions of each team member are summarized bellow, showcasing the distribution of responsibilities across different project phases.

1. **Sprint planning and setup:**

Table 4: Planning task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| Backlog refinement and prioritization | Amaara Arshad, Amina Abdulhaq, Maryam, Uzma Sajjad |
| Sprint planning and setup | Amina Abdulhaq, Maryam, Uzma Sajjad |

1. **System design (Structural diagrams):**

Table 5: Structural diagrams task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| Structural diagrams | Aqsa Faraz, Maryam |

1. **System design (Behavioural diagrams):**

Table 6: Behavioural diagrams task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| Behavioural diagrams | Uzma Sajjad, Amaara Arshad |

1. **GUI design:**

Table 7: GUI design task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| GUI design | Aqsa Faraz, Amaara Arshad |

1. **Database design:**

Table 8: Database design task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| Database design | Amina Abdulhaq, Maryam, Uzma Sajjad |

1. **Functionality implementation:**

Table 9: Functionality implementation task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| Account management functionality | Uzma Sajjad |
| Financial transaction functionality | Maryam |
| Account services functionality | Aqsa Faraz |
| Security and compliance implementation | Amina Abdulhaq |
| Customer support and maintenance | Amaara Arshad |

1. **Testing and bug fixing:**

Table 10: Testing and bug fixing task assignments.

|  |  |
| --- | --- |
| **Task** | **Team member** |
| Testing and bug fixing | Amina Abdulhaq, Maryam, Uzma Sajjad |

# **2.3: Contributions chart:**

The chart below provides a visual representation of each team member’s contribution to different aspects of the project, including planning, implementation testing etc.

Figure 2 : Task distribution of each team member.

The figure 2 illustrates the distribution of tasks among team members and highlights their respective areas of expertise and involvement in the project. It reflects the collaborative effort of the team in ensuring the successful completion of various project components.

# **Section 3: Project planning**

For this project we choose to follow these agile methodologies due to their iterative nature, which aligns well with the requirements of the software development. We utilized various agile techniques such as user stories, sprint planning, simple design, pair programming and testing to ensure effective collaboration and development progress tracking.

# **3.1. Reasoning:**

We reason for choosing these methodologies is that these methodologies allowed us to effectively and efficiently collaborate and deliver incremental value to stakeholders. As agile fosters close collaboration between team members and stakeholders, ensuring continuous feedback and alignment with project goals.

Our approach to project plan ensured that we had a clear roadmap for development, with well defined milestones and deliverables. This helped us stay organized, prioritize tasks effectively, and respond promptly to changes in project constraints. By adopting agile methodologies and utilizing appropriate planning techniques we were able to streamline our development process, improve collaboration among team members, and deliver high quality software solution within the specified timeframe.

# **3.2. Gantt chart:**

The project was broken down into manageable activities and milestones using a Gantt chart. This helped us visualize dependencies, allocate resources effectively, and track progress throughout the development lifecycle. We identified key deliverables and set realistic timelines to ensure timely completion of each phase. Following figure shows the Gantt Chart of the banking software system project.

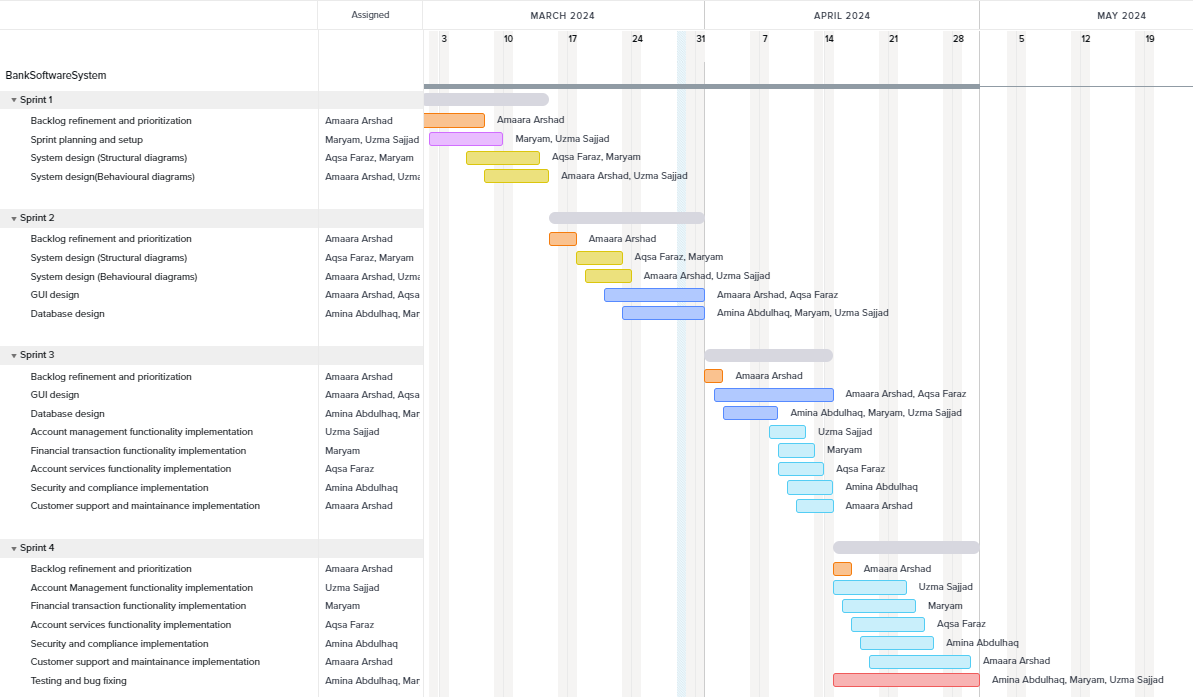


Figure 3 : Gantt chart for Bank Software System project.

# **3.3. Work breakdown structure:**

Following table shows the work breakdown structure for this project.

Table 5: Work breakdown structure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task ID** | **Task Name** | **Parent Task** | **Start Date** | **End Date** | **Resources** | **Colour** |
| **1** | **Bank Software System** |  | **01/03/2024** | **30/04/2024** |  |  |
| **1.1** | **Sprint 1** | **1** | **01/03/2024** | **14/03/2024** |  |  |
| 1.1.1 | Backlog refinement and prioritization | 1.1 | 01/03/2024 | 07/03/2024 | Amaara Arshad | orange1 |
| 1.1.2 | Sprint planning and setup | 1.1 | 02/03/2024 | 09/03/2024 | Amina Abdulhaq, Maryam, Uzma Sajjad | purple1 |
| 1.1.3 | System design (Structural diagrams) | 1.1 | 06/03/2024 | 13/03/2024 | Aqsa Faraz, Maryam | yellow1 |
| 1.1.4 | System design (Behavioural diagrams) | 1.1 | 08/03/2024 | 14/03/2024 | Uzma Sajjad, Amaara Arshad | yellow1 |
| **1.2** | **Sprint 2** | **1** | **15/03/2024** | **31/03/2024** |  |  |
| 1.2.1 | Backlog refinement and prioritization | 1.2 | 15/03/2024 | 17/03/2024 | Amaara Arshad | orange1 |
| 1.2.2 | System design (Structural diagrams) | 1.2 | 18/03/2024 | 22/03/2024 | Aqsa Faraz, Maryam | yellow1 |
| 1.2.3 | System design (Behavioural diagrams) | 1.2 | 19/03/2024 | 23/03/2024 | Amaara Arshad, Uzma Sajjad | yellow1 |
| 1.2.4 | GUI design | 1.2 | 21/03/2024 | 31/03/2024 | Aqsa Faraz, Amaara Arshad | blue1 |
| 1.2.5 | Database design | 1.2 | 23/03/2024 | 31/03/2024 | Amina Abdulhaq, Maryam, Uzma Sajjad | blue1 |
| **1.3** | **Sprint 3** | **1** | **01/04/2024** | **14/04/2024** |  |  |
| 1.3.1 | Backlog refinement and prioritization | 1.3 | 01/04/2024 | 02/04/2024 | Amaara Arshad | orange1 |
| 1.3.2 | GUI design | 1.3 | 02/04/2024 | 14/04/2024 | Aqsa Faraz, Amaara Arshad | blue1 |
| 1.3.3 | Database design | 1.3 | 03/04/2024 | 08/04/2024 | Uzma Sajjad, Amina Abdulhaq, Maryam | blue1 |
| 1.3.4 | Account management functionality implementation | 1.3 | 08/04/2024 | 11/04/2024 | Uzma Sajjad | blue2 |
| 1.3.5 | Financial transaction functionality implementation | 1.3 | 09/04/2024 | 12/04/2024 | Maryam | blue2 |
| 1.3.6 | Account services functionality implementation | 1.3 | 09/04/2024 | 13/04/2024 | Aqsa Faraz | blue2 |
| 1.3.7 | Security and compliance implementation | 1.3 | 10/04/2024 | 14/04/2024 | Amina Abdulhaq | blue2 |
| 1.3.8 | Customer support and maintenance implementation | 1.3 | 11/04/2024 | 14/04/2024 | Amaara Arshad | blue2 |
| **1.4** | **Sprint 4** | **1** | **15/04/2024** | **30/04/2024** |  |  |
| 1.4.1 | Backlog refinement and prioritization | 1.4 | 15/04/2024 | 16/04/2024 | Amaara Arshad | orange1 |
| 1.4.2 | Account Management functionality implementation | 1.4 | 15/04/2024 | 22/04/2024 | Uzma Sajjad | blue2 |
| 1.4.3 | Financial transaction functionality implementation | 1.4 | 16/04/2024 | 23/04/2024 | Maryam | blue2 |
| 1.4.4 | Account services functionality implementation | 1.4 | 17/04/2024 | 24/04/2024 | Aqsa Faraz | blue2 |
| 1.4.5 | Security and compliance implementation | 1.4 | 18/04/2024 | 25/04/2024 | Amina Abdulhaq | blue2 |
| 1.4.6 | Customer support and maintenance implementation | 1.4 | 19/04/2024 | 29/04/2024 | Amaara Arshad | blue2 |
| 1.4.7 | Testing and bug fixing | 1.4 | 15/04/2024 | 30/04/2024 | Amina Abdulhaq, Maryam, Uzma Sajjad | red1 |

# **3.4. Agile artifacts:**

* Planning discussion.
* Gantt chart.
* Work breakdown structure.
* User stories and tasks etc.

# **Section 4: Prototype Design**

## **4.1. Description of the system:**

The purpose of this assessment was to create a banking software system following an agile methodology by assigning different tasks to each team member. The system that we came up with had the following functionalities:

1. **Account management:**

Create, edit, and close a bank account.

1. **Financial transactions:**

Deposit, withdraw and transfer amount and keep history of the transactions.

1. **Account services:**

View detailed account information, request check books and credit cards and managing recurring payments.

1. **Security and compliance:**

Robust authentication mechanisms and encryption and compliance functionalities.

1. **Customer support and maintenance:**

Integrated helpdesk for customers with the functionality to post queries and reporting issues and fraudulent activities.

## **4.2. System design:**

1. **Structural diagrams:**

**Class diagram:**

Class diagram describes the structure of a system by showing the system's classes, their attributes, methods, and the relationships among objects.

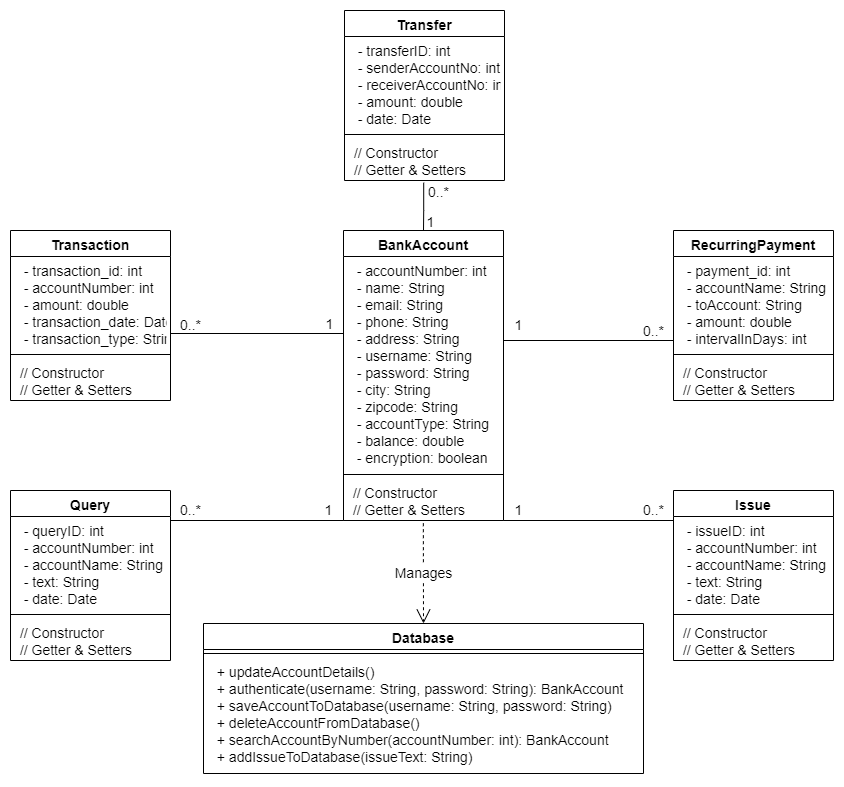


Figure 4 : Class diagram.

**Entity relationship diagram:**

The Entity Relationship Diagram explains the relationship among the entities present in the database.

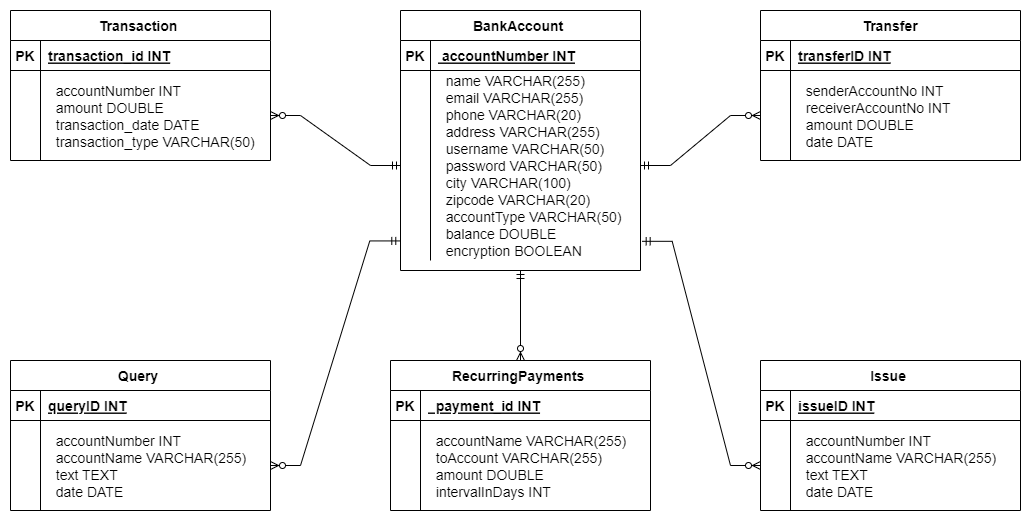
****

Figure 5: Entity relationship diagram.

1. **Behavioural diagrams:**

**Use case diagram:**

A Use Case diagram illustrates a set of use cases for a system, i.e. the actors and the relationships between the actors and use cases. The following figure shows the use case diagram of banking software systems.

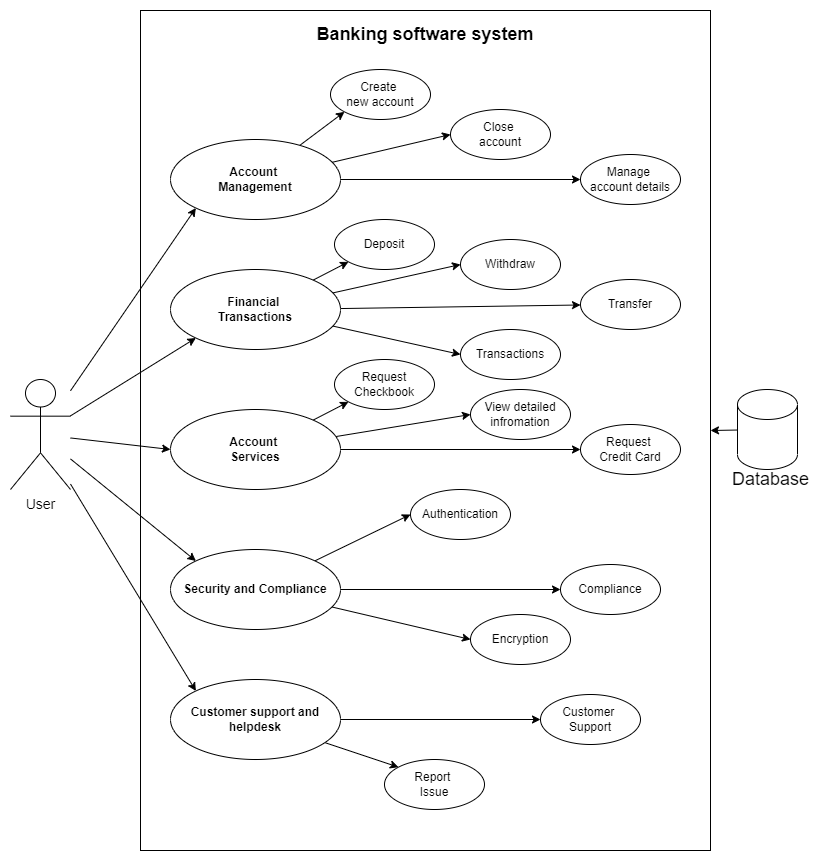
****

Figure 6: Use case diagram.

**Sequence diagram:**

A sequence diagram simply depicts the interaction between the objects in a sequential order that is the order in which these interactions occur. Sequence diagrams describe how and in what order the objects in a system function.

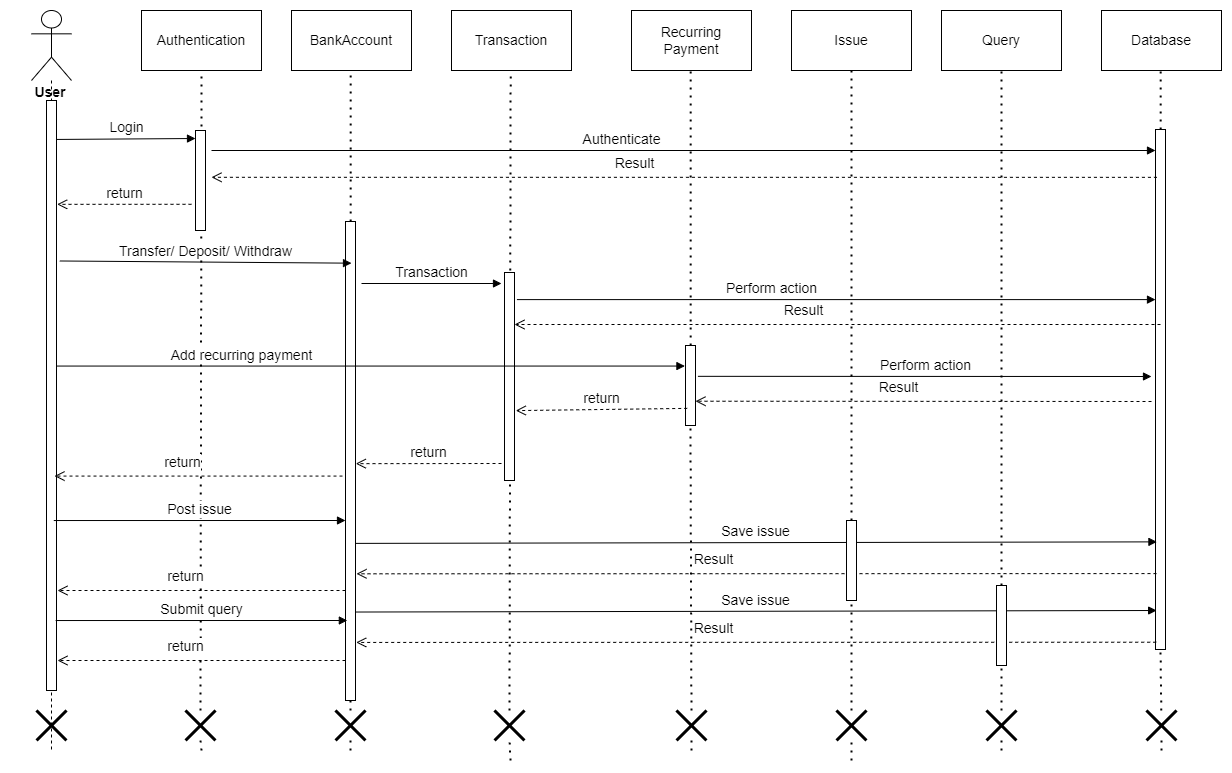
****

Figure 7: Sequence diagram

## **4.3. UI designs:**

1. **Page 1:** This is the main page of the system and contains all the options or button for performing different functions.

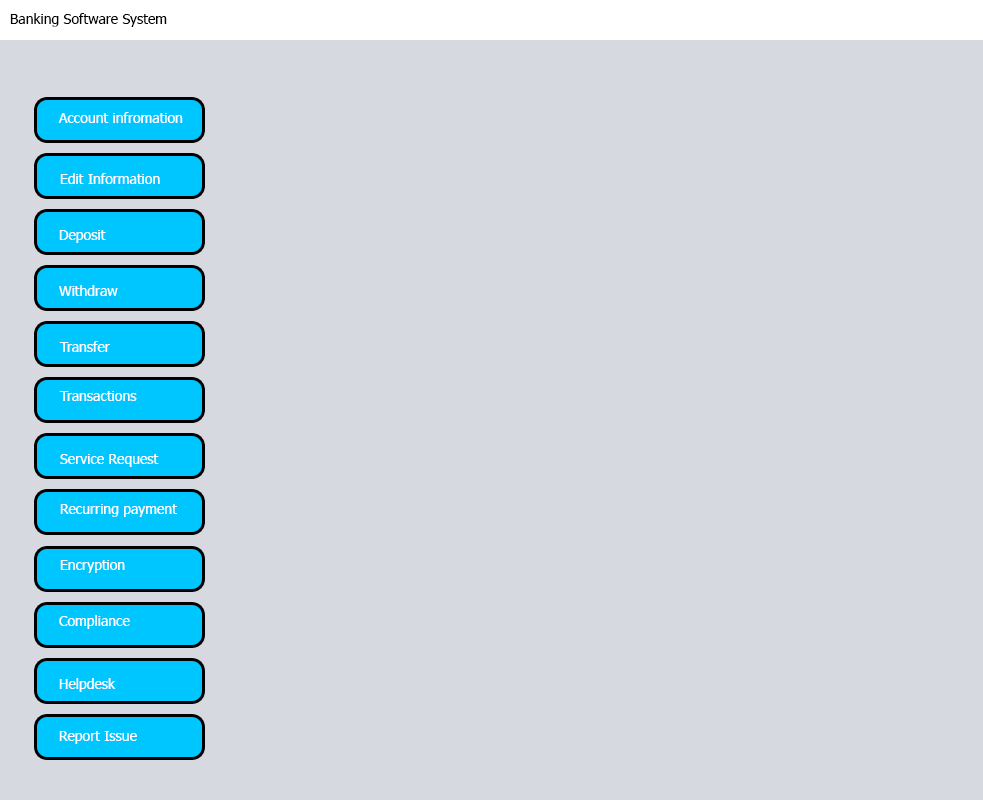
****

Figure 8: Mockup design of the main page of banking software system.

1. **Page 2:** This is the deposit page and allows the user to deposit amount and generate a report.

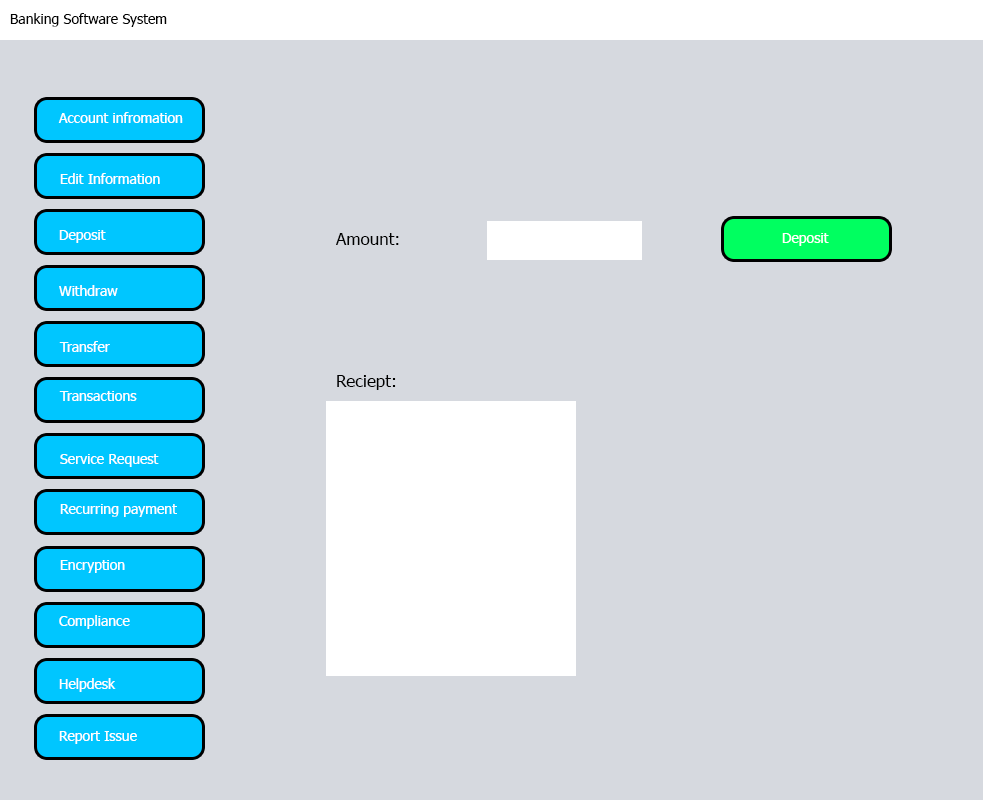
****

Figure 9: Mockup design of the deposit page of banking software system

## **4.3. Version Control:**

GitHub was used for version control, the link to the repository is:

////////////////////////////////////////////////////////////////////////

## **4.4. Developed system:**

The system has been developed in java using Eclipse IDE. The user interface of the system is developed using java swing library.

1. **Authentication page:**

A screenshot of a computer login

Description automatically generated

Figure 10: Authentication page.

1. **Account setup page:**

A screenshot of a computer

Description automatically generated

Figure 11: Account setup page.

1. **Main page:**

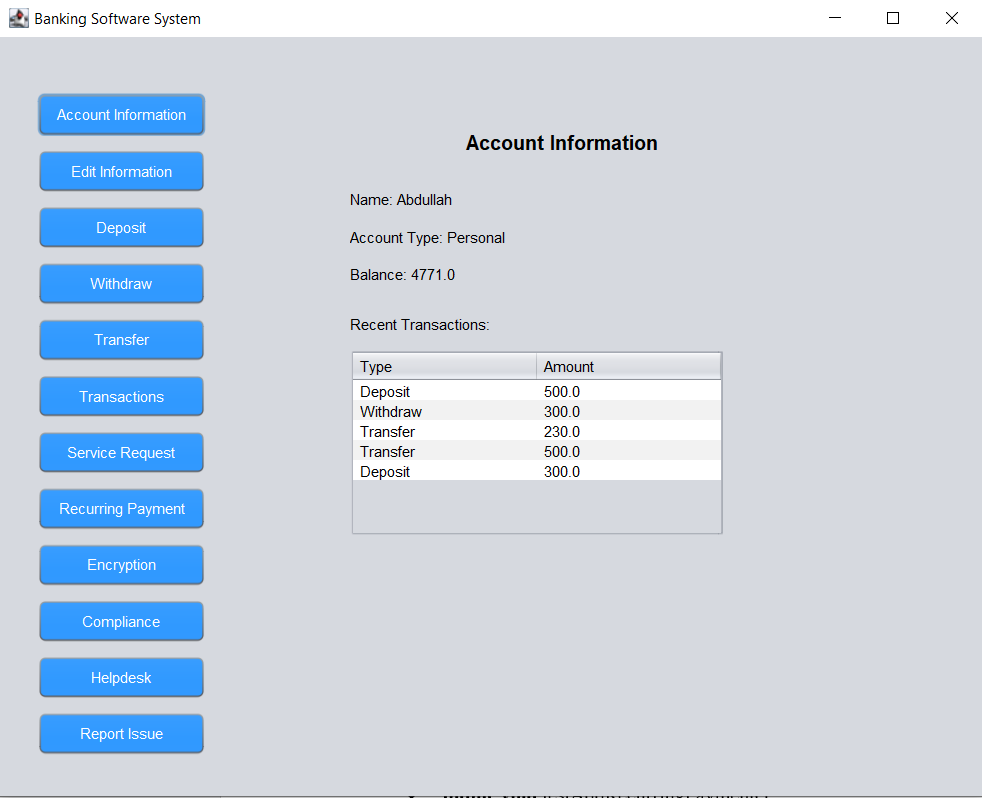


Figure 12: Detailed account information page.

1. **Edit information page:**

A screenshot of a computer

Description automatically generated

Figure 13: Edit account information page.

1. **Deposit page:**

A screenshot of a computer

Description automatically generated

Figure 14: Deposit page.

1. **Withdraw page:**

A screenshot of a computer

Description automatically generated

Figure 15: Withdraw page.

1. **History page:**

A screenshot of a computer

Description automatically generated

Figure 16: Transaction history page.

1. **Services page:**

A screenshot of a computer

Description automatically generated

Figure 17: Service request page.

1. **Recurring payments page:**

A screenshot of a computer

Description automatically generated

Figure 18: Recurring payments page.

1. **Encryption page:**

A screenshot of a computer

Description automatically generated

Figure 19: Encryption page.

1. **Compliance page:**

A screenshot of a computer

Description automatically generated

Figure 20: Compliance page.

1. **Help desk page:**

A screenshot of a software system

Description automatically generated

Figure 21: Helpdesk page.

1. **Report issue page:**

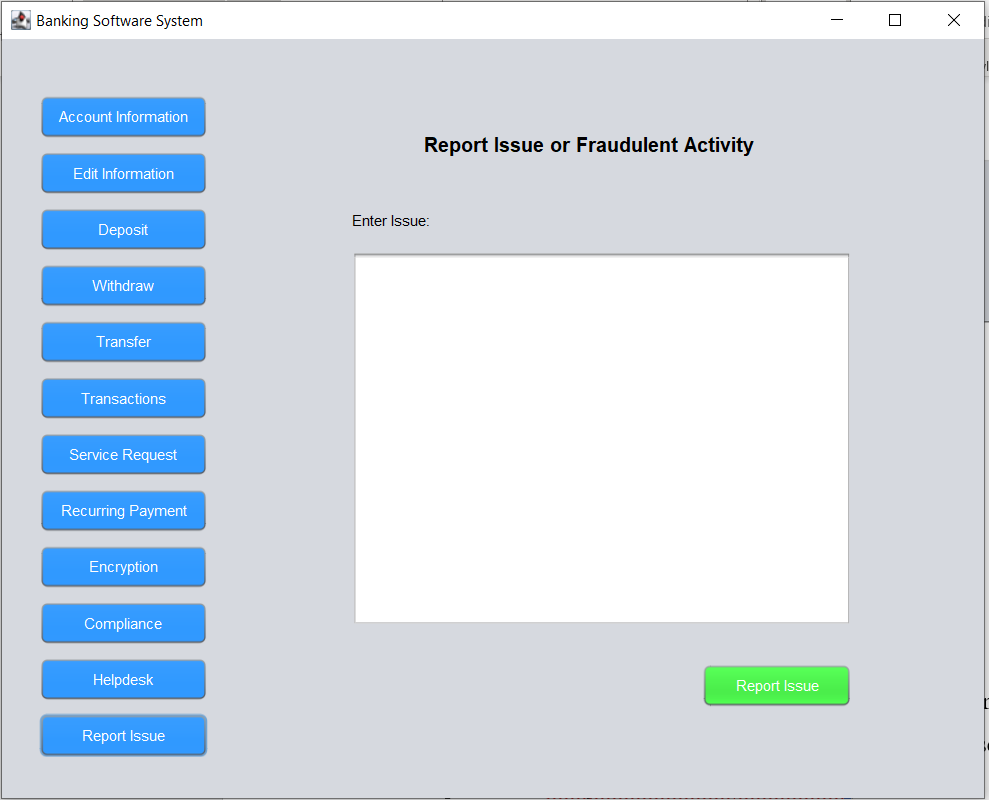


Figure 22: Report issue page.

## **4.4. Testing:**

The system was thoroughly tested by the team and the automated testing was done by JUnit testing.

These are the test methods of the Databasetest.java class which was used for automated testing:

* + - **public** **void** testUpdateAccountDetailsInDatabase()
    - **public** **void** testAuthenticate()
    - **public** **void** testAddRecurringPayment()
    - **public** **void** testGetAllRecurringPayments()
    - **public** **void** testGetTransactionsByAccountNumber()
    - **public** **void** testAddTransactionToDatabase()
    - **public** **void** testSearchAccountByNumber()
    - **public** **void** testAddIssueToDatabase()

The following figure shows a screen shot taken from Eclipse IDE that show a run of a test suite on Database class. There are a total of 8 tests and each test is successful without any error or failures.

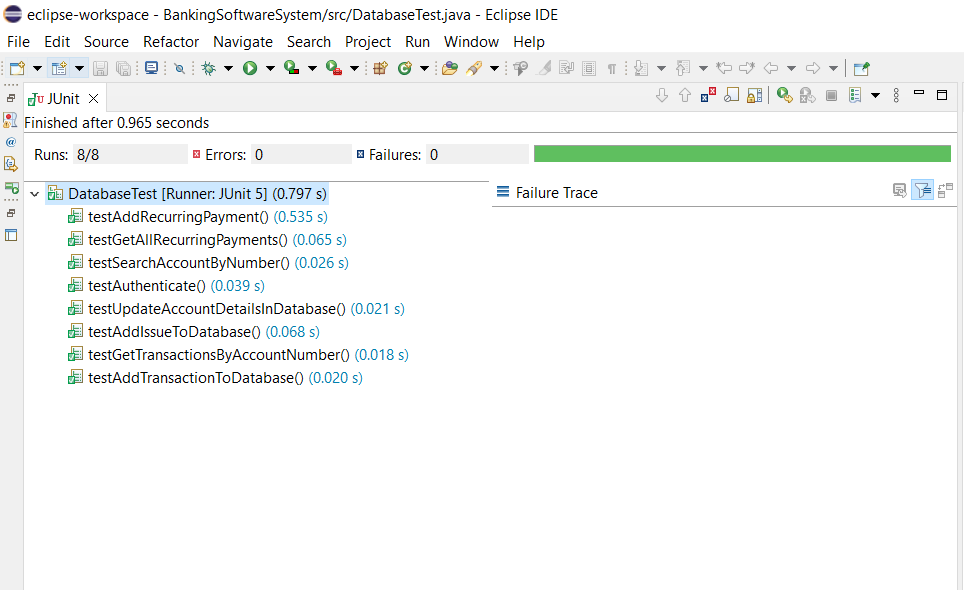


Figure 23: Screen shot of the successful run of Unit test

# **Section 5: Critical evaluation**

The development of the banking system using agile methodologies and XP techniques provided valuable insights into the effectiveness of various software engineering tools and techniques. This section critically evaluates the utilization of these tools and techniques, considering their impact on project success and their alignment with industry practices. It is imperative to note that the evaluation provided here is based on the specific utilization of tools and techniques used within out project, adhering to the directive to be specific and provide evidence from our own project.

# **5.1. Advances in software processes:**

1. **Methodologies**

The concept of agile process models has gained great popularity in software development in recent years, promoting fast development but facing challenges such as poor documentation and low quality [1]. However, the adoption of agile methodologies, particularly the scrum framework, enabled iterative and incremental development cycles, promoting flexibility and adaptability in responding to changing requirements. Regular sprint planning, review, and retrospective meetings facilitated continuous improvement and ensured alignment with stakeholder expectations.

1. **Agile practices:**

Embracing agile practices such as user stories, sprint planning, and continuous integration, have been widely recognized for fostering close collaboration among team members and stakeholders [2]. In banking software system project, the iterative nature of agile development allowed for early and frequent delivery of working software, enhancing transparency and stakeholder satisfaction.

# **5.2. Software engineering techniques:**

Throughout the development life cycle of the banking software system, a variety of software engineering tools and techniques were employed to facilitate efficient collaboration, streamline development processes, and ensure the delivery of a high-quality software system.

1. **Version control with Git:**

Version control systems like Git have been instrumental in enabling seamless collaboration, code management, and version tracking among team members [2]. The use of commits, push and pull requests, branches, and merge strategies, in this project, facilitated efficient code integration and conflict resolution, ensuring code quality and stability throughout the development process.

1. **Project planning and management tools:**

Tools such as Trello and TeamGantt.com played crucial role in project planning, task management, and milestone tracking [3]. The visualization of project workflows, allocation of resources, and monitoring of progress facilitated effective project management and decision making, contributing to the successful completion of project’s objectives.

1. **Prototype design:**

The creation of prototype designs, including structural and behavioural diagrams, provided a visual representation of system requirements and design specifications. Draw.io was used to draw the structural and behavioural diagrams, which is a prominent and useful tool in UML design. Prototyping facilitated stakeholder engagement, requirement validation, and early feedback, guiding the development process, and minimizing the risk of misunderstanding or misinterpretations.

1. **Test driven development and automated testing:**

The adoption of test-driven development (TDD) and automated testing practices ensured the reliability and functionality of the software system. Junit tests was employed for automated testing as shown in figure 23. Automated Unit tests, validated critical system functionalities, detecting defects and regressions early in the development lifecycle.

1. **Pair programming and code reviews:**

Pair programming and regular code reviews promoted knowledge sharing, code reviews promoted knowledge sharing, code quality improvement, and error detection. Collaborative code development and review practices enhanced code readability, maintainability, and adherence to coding standards, contributing to the overall quality of the codebase.

# **5.3. Societal impact:**

1. **Social impact:**

The banking software has a significant social impact, providing individuals and businesses with convenient access to financial services. The system enhances financial inclusion, empowers users to manage their finances effectively, and facilitates economic participation and growth.

1. **Ethical impact:**

Robust security measures and compliance functionalities within the system uphold ethical standards and protect user data and financial transactions. Ethical considerations, such as data privacy and confidentiality, are prioritized to ensure trust and confidence among users and stakeholders.

1. **Entrepreneurial impact:**

The entrepreneurial impact of the banking software system lies in its potential to stimulate innovation and economic growth within the banking sector. The system fosters competition, drives customer centric financial services, and creates opportunities for new business models and revenue streams.

# **Section 6: References**

[1] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, ‘A decade of agile methodologies: Towards explaining agile software development’, *J. Syst. Softw.*, vol. 85, no. 6, pp. 1213–1221, Jun. 2012, doi: 10.1016/j.jss.2012.02.033.

[2] M. Gasparic and A. Janes, ‘What recommendation systems for software engineering recommend: A systematic literature review’, *J. Syst. Softw.*, vol. 113, pp. 101–113, Mar. 2016, doi: 10.1016/j.jss.2015.11.036.

[3] M. Rizwan Jameel Qureshi and S. A. Hussain, ‘An adaptive software development process model’, *Adv. Eng. Softw.*, vol. 39, no. 8, pp. 654–658, Aug. 2008, doi: 10.1016/j.advengsoft.2007.08.001.

**Appendix**