Report

# Section 1: Description and Evidence of Agile and XP Techniques

## Project Overview:

It is a banking application that has been developed using Next.js and MongoDB providing functionality for account management, financial transactions, account services as well as security and compliance. The application provides a very simple user interface with powerful support at the back to be able to deal with many operations that take place in the banking system.

## Agile Methods Employed:

### Iterative Development:

* **Approach:** The application was developed by breaking down the project into small, manageable iterations (sprints). Every sprint aimed to deliver a subset of features or functionality with continuous feedback and improvement. This way, from each sprint output is obtained, which is demonstrated to obtain confirmation toward the end of it.
* **Evidence:** Features, such as account management and financial transactions, were developed in independent sprints. For example, if the first sprint focused on the creation and setup of accounts, then the second sprint would be about financial transactions like deposits or withdrawals.

### User Stories and Backlog Management:

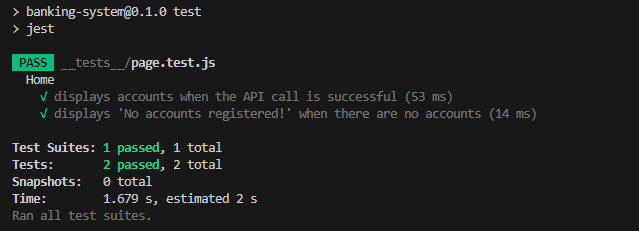
* **Approach:** The project captured requirements from an end-user perspective. Evidence, each user story was added to the product Backlog, prioritized, and refined in collaboration with stakeholders.
* **Evidence:** It had such user stories as "As a user I want to be able to deposit cash and have the ATM give me a receipt" created and added to the backlog. The backlog is reviewed regularly and adjusted due to feedback and changing priorities. See appendix for user stories.

### Sprint Planning and Review:

* **Approach:** The sprint planning meetings were done at the beginning of each sprint to determine the scope and goals. The sprint review meetings were held at the end of each sprint to display the completed features and get feedback.
* **Evidence:** During sprint planning, the tasks such as implementing deposit functionality and receipt generation were outlined and allocated. Sprint reviews were done via displaying the working deposit feature to the stakeholders and then discussing the possible improvements.

### Continuous Integration and Deployment:

### **Methodology:** To guarantee that code updates were consistently included into the primary codebase, the project used git version control. To enable seamless releases, Jest.js was used for writing automated tests.



### Pair Programming and Code Reviews:

* **Approach:** To improve code quality and promote teamwork among participants, pair programming and code reviews were encouraged.
* **Evidence:** Pair programming was used to create important code areas, such the two-factor authentication mechanism. To make sure that best practices and coding standards were followed, code reviews were carried out.

### Test-Driven Development (TDD):

* **Method:** Before deploying features, tests were written using test-driven development (TDD). This method assisted in clearly stating the requirements and making sure the code behaved as expected.
* **Evidence:** Prior to deployment, tests were established for features including fund transfers and transaction histories. This guaranteed that all of the features were functional and had undergone extensive testing.

## Hypothetical Group Development:

### Roles and Responsibilities:

1. **Product Owner:** Identified the goals of the project, set priorities for the backlog, and made sure the development team produced features that satisfied the objectives of the business.
2. **Scrum Master:** Led stand-ups, retrospectives, and sprint planning. eliminated barriers and made sure the group adhered to Agile procedures.
3. **Developers:** Wrote code, carried out code reviews, and implemented features. The application's frontend interfaces for account management and backend services for financial transactions were among the various features that developers focused on.
4. **Testers for quality assurance (QA)**: Examined the application to find errors and confirm that the features fulfilled the requirements for acceptance. They designed and carried out test scenarios covering a range of features.

### Dividing Responsibilities:

* Account creation, editing, and termination were the responsibility of a single team member.
* Financial Transactions: The deposit, withdrawal, and fund transfer processes were managed by a different team member.
* Account Services: A third team member handled checkbooks, regular payments, and account information.
* Security and Compliance: A fourth team member made sure that data encryption and strong authentication procedures were in place.

### Planning:

1. Sprint Planning: To identify tasks, assign responsibilities, and estimate effort, the team held sprint planning sessions.
2. Daily Stand-Ups: Quick meetings were conducted every day to talk about the day's goals, obstacles, and progress.
3. Sprint Retrospectives: The team convened retrospectives at the conclusion of each sprint to discuss what went well, what needed improvement, and how to improve the development process.
4. The project was built collaboratively and iteratively through the use of XP and Agile approaches, enabling continual improvement and adaptability to changing requirements.

# Section 2: Project Planning

## Software Process Selection:

Agile software development was selected for the banking application development because of its iterative and flexible nature, which fits in well with the project's changing requirements. Agile approaches make managing complicated projects with changing needs easier by enabling regular feedback, continuous improvement, and adaptability to change.

## Reasoning for Choosing Agile:

### Flexibility and Adaptability:

* **Reasoning:** Regulatory changes, user feedback, and evolving technology frequently need banking programs to alter. Agile ensures that the application is current and relevant by allowing for frequent revisions and iterations.
* **Evidence:** As user requirements or compliance standards change during the development process, they can be addressed in later sprints to reduce the risk of obsolescence.

## User-Centric Approach:

* **Reasoning:** Agile is all about delivering features that give people real value. The development team can prioritize and deploy features that directly address user needs and enhance the user experience overall by collaborating closely with stakeholders and end users.
* **Proof:** User stories, like "View detailed account information," were developed to address certain needs and were ranked according to how well they would make users happy.

## Continuous Feedback and Improvement:

* **Reasoning:** Agile places a strong emphasis on iterative development and frequent feedback, which aid in improving features and resolving problems early in the development cycle. This methodology guarantees that the application develops in response to feedback from stakeholders and real-world usage.
* **Evidence:** Sprint reviews and retrospectives offered chances to get input, evaluate development, and make required modifications.

## Breaking Down the Work into Activities and Milestones:

### Initial Planning:

* **Activities:**
  + Define project scope and objective.
  + Identify key featuers and functionalities.
  + Create a high level project roadmap.
* **Milestones:**
  + Project kickoff and initial planning completed.
  + High-level requirement and goals established.

### Requirement Analysis and Backlog Creation:

* **Activities:**
  + Gather detailed requirement from stakeholders.
  + Break down requirement into user stories and add them to the product backlog.
  + Prioritize backlog item based on importance and dependency.
* **Milestones:**
  + Product backlog created and prioritized.
  + Detailed requirement documented and reviewed.

### Sprint Planning and Execution:

* **Activities:**
  + Plan sprints, including defining sprint goals and selecting backlog item for implementation.
  + Develop features based on sprint goal, including coding, testing, and integrating functionalities.
  + Conduct daily stand-up meetins to track progress and address issues.
* **Milestones:**
  + Completion of each sprint with defined features and functionalities.
  + Successful integration and testing of developed features.

### Feature Development and Integration:

* **Activities:**
  + Implement core feature such as account management, financial transactions, and account services.
  + Integrate security and compliance measure into the application.
  + Conduct code review and ensure adherence to coding standards.
* **Milestones:**
  + Core features developed and integrateds.
  + Security and compliance measures implemented.

### Testing and Quality Assurance:

* **Activities:**
  + Conduct unit testing, integration testing, and system testing.
  + Perform user acceptance testings (UAT) with stakeholders.
  + Address bugs and issues identified during testing.
* **Milestones:**
  + Successful completion of testing phases.
  + Bug fixes and improvements based on testing feedback.

### Deployment and Release:

* **Activities:**
  + Prepare the application for deployment to production environments.
  + Conduct final checks and validation before release.
  + Deploy the application and monitor for post-release issues.
* **Milestones:**
  + Application deployed to production.
  + Post-release monitoring and support initiated.

### Post-Deployment and Maintenance:

* **Activities:**
  + Collect user feedback and monitor application performance.
  + Address any issues or bugs reported by users.
  + Plan and implement updates and enhancements based on feedback.
* **Milestones:**
  + Post-deployment feedback collected and analyzed.
  + Updates and enhancements planned for future sprints.

## Summary:

The Agile methodology was selected due of its flexibility in responding to evolving demands and its emphasis on producing value through incremental development. The project was divided into discrete tasks and completion dates in order to guarantee organized development and efficient use of resources. With well-defined goals and deliverables for every phase of the project, the team was able to stay on course and accomplish the intended results.

# Section 3: Prototype Design

## System and Interface Design Overview:

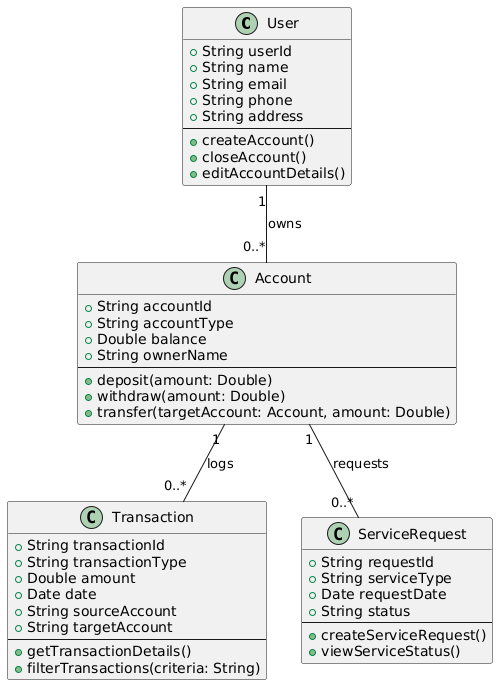
The banking application's prototype was made to offer an easy-to-use interface for handling banking procedures. The system's overall architecture, user interface components, and interaction processes are all included in the design. This part offers a thorough explanation of the interface and system designs, together with screenshots of the produced prototype and automated testing.

## System Design:

The system design is represented through class diagrams sequence diagrams and use case diagrams which illustrate the structure and behavior of the application.

### Class Diagram:

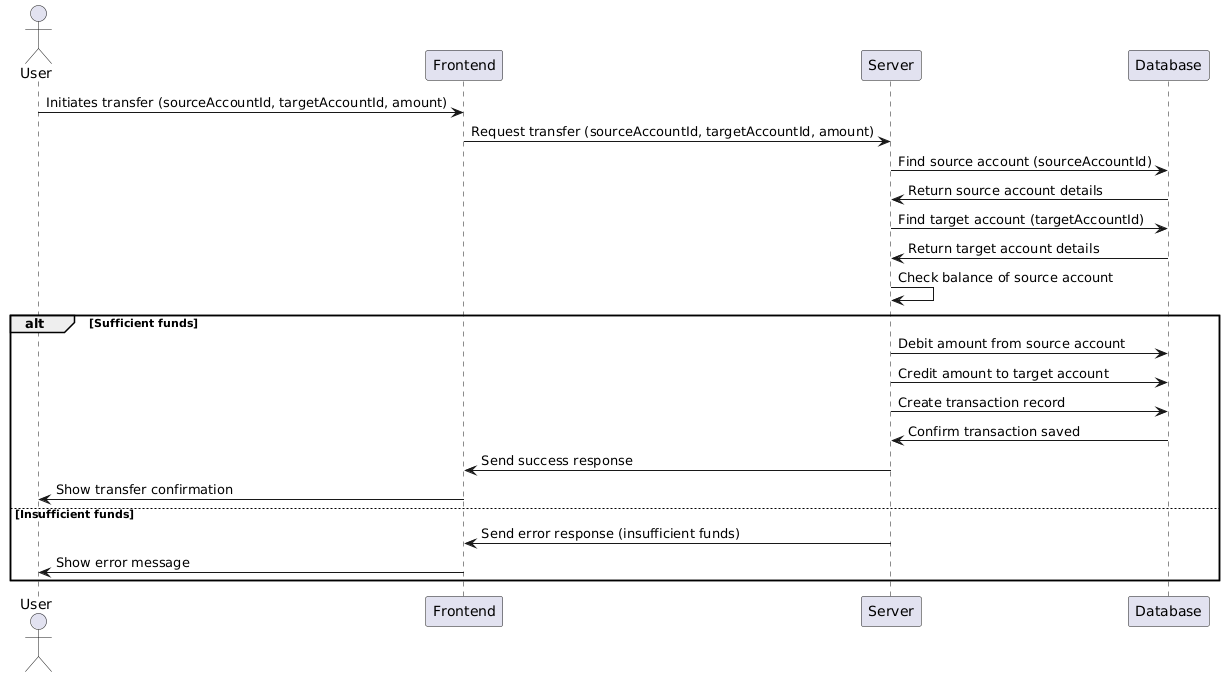
* **Description:** The primary elements of the banking application and their connections are shown in the class diagram. They contain classes for managing transactions, account services, and user accounts.



* **Details:**
  + Account class manages account details and operations.
  + Transactions class handle financial transactions.
  + User class represents users of the application and their profiles.
  + Service class manages additional account services like checkbooks and cards.

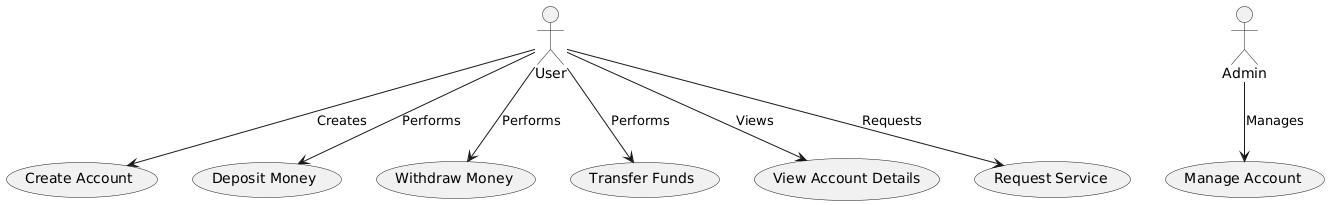
### Sequence Diagram:

* **Description:** Sequence diagrams show how various system components interact with one another during particular use cases, including financial transfers or account creation.



### Use Case Diagrams:

* **Description:** Use case diagrams outline the system's functionality from the viewpoint of the user as well as the various actors' interactions with it.

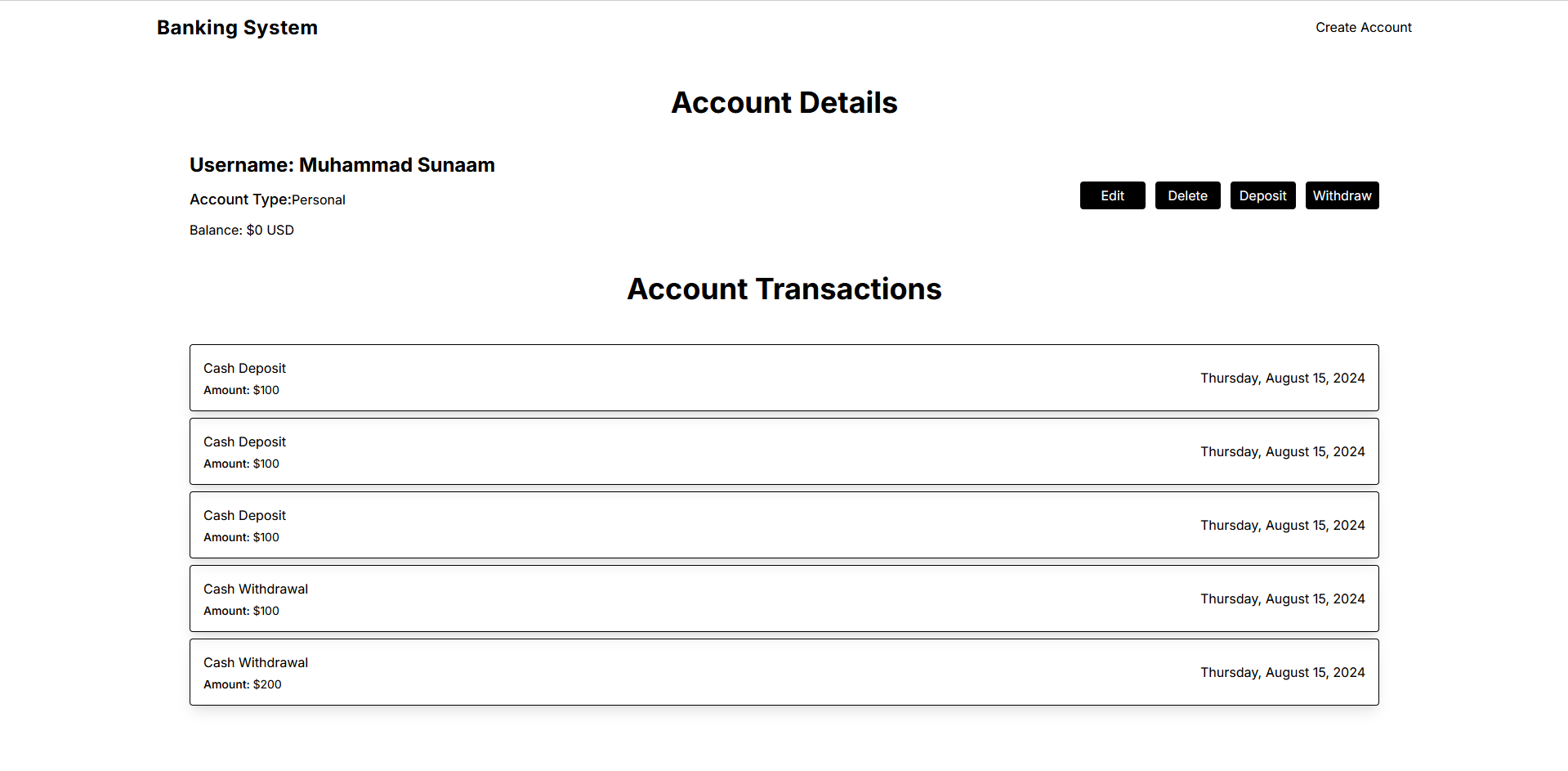
****

* **Details:**
  + Bank Customer can perform operations like account management, financial transactions, and requesting services.
  + System Administrator handles administrative tasks such as managing user accounts and ensuring system compliance.

## Interface Design:

### User Interface (UI) Design:

* **Description:** The goal of the user interface design is to make the banking application's interface simple and easy for people to use. Transaction forms and dashboards are important components.
* **Screens:**
  + **Dashboard:** Displays a summary of account information, recent transactions, and quick access to key features.



* + **Transaction Forms:** Simple and user-friendly forms for depositing, withdrawing, and transferring funds.

A screenshot of a computer

Description automatically generated

### Evidence of Developed Prototype:

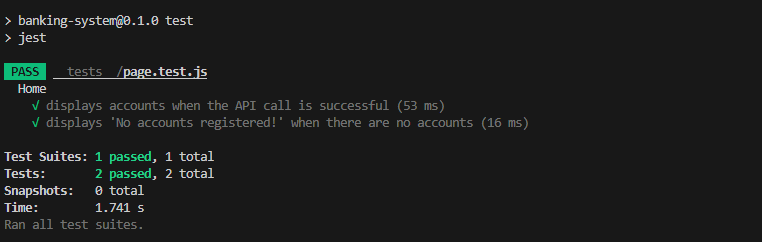
Link:

## Automated Testing:

### Automated Testing Overview:

The implementation of automated testing was done to make sure the banking application was reliable and functional. Aspects such as functional integration, and security testing were all included in the designs of the test.

* **Instrument Applied:** The tests were written and run using Jest as the testing framework, guaranteeing thorough coverage and trustworthy outcomes.



# Section 4: Critical Evaluation of Software Engineering Tools and Techniques

## Advances in Software Processes

### Methodologies:

In order to guarantee an organized and effective development process, the banking application project made use of contemporary software development approaches. Agile (Torgeir, Sridhar, VenuGopal, & Nils Brede, 2012) was the main technique employed, and its flexible and iterative nature worked well.

The project was able to swiftly incorporate comments and adjust to changes in needs by implementing Agile methodology. For instance, the account management interface was improved in terms of usability after input on the first prototype. Agile's iterative cycles made it possible to develop incrementally and frequently reevaluate priorities, which was essential for managing the banking application's dynamic requirements.

### Agile Processes:

The development of the banking application was effectively managed through the utilization of agile procedures. The project used continuous integration, sprint planning (Inayat, Salim, Marczak, Daneva, & Shamshirband, 2015), and iterative development techniques, all of which enhanced the flexibility and responsiveness of the development environment.  
Regular assessment and modification were made possible by the use of sprints in the development and refinement of features like account services and financial transactions. Continuous improvement and problem solving were enabled by sprint retrospectives and daily stand-up meetings. For example, the team determined that further security features were needed during a sprint review, and those features were quickly included to the next sprint cycle.

## Software Engineering Techniques

### Project Management:

For the banking application to be developed successfully, effective project management was essential. Backlog management, sprint planning, progress monitoring, and other agile project management approaches (Highsmith) were used to manage tasks and guarantee feature delivery on time.

The product backlog was managed and the progress of each sprint was monitored using Jira and other similar tools. Sprint goals were specified and tasks were ranked according to their dependencies and relevance. This method made guaranteed that important features, such transaction processing and secure login, were produced and tested on time.

### Prototype Design:

According to Rogers, Sharp, and Preece (2011), the prototype design phase entailed developing user-friendly interfaces and confirming them using iterative feedback. A thorough grasp of the system architecture and user interactions was made possible by the employment of class, sequence, and use case diagrams.

Key components like the account dashboard and transaction forms were incorporated in the interface design as mockups and screenshots. The design was improved with input from prototype testing, producing an interface that is more user-friendly and intuitive. For instance, changes to the transaction history page's layout to increase readability were made in response to user comments.

### Version Control:

Git and GitHub were used to manage version control, allowing for easier collaboration, code management, and change tracking during the development process.

Git was used to handle branching, merge changes, and manage code versions. GitHub offered a platform for collaborative development, problem tracking, and code review. On GitHub, pull requests and commit histories show how code modifications were applied and examined, guaranteeing the coherence and quality of the code.

## Impact of Advanced Software Systems and Engineering

### Social Impact:

Innovative digital solutions have revolutionized the way people engage with banking services, increasing the efficiency and accessibility of financial management. Users may easily conduct transactions, see account details, and manage their finances online with the help of the banking application.  
The program enhances accessibility and user experience with its user-friendly interface and features like real-time balance updates and transaction history. The convenience of banking from a mobile device is increased, and customers are more equipped to handle their money.

### Ethical Impact:

Ethical aspects were taken into account during the creation of the banking application, specifically with regard to data security and privacy. Preserving user confidence and protecting data necessitates making sure that sensitive data is secured and that the application conforms with applicable laws.  
To safeguard user information, the program encrypts sensitive data and uses strong authentication methods, such as two-factor authentication. Ensuring adherence to financial regulations and data protection legislation was given top priority in order to tackle ethical issues of data security.

### Entrepreneurial Impact:

The banking app serves as an example of the fintech industry's possibilities for business initiatives. The project demonstrates how entrepreneurs can innovate and provide fresh solutions in the financial industry by utilizing contemporary technologies and software engineering techniques.  
The application's creation underscores the possibility of developing user-centered, competitive financial services. With features like safe account management and automated transactions, the app is positioned as a useful resource in the fintech sector, providing chances for expansion and self-employment.

# References

Highsmith, J. A. (n.d.). *Adaptive Software development: a collaborative approach to managing complex systems.*

Inayat, Salim, Marczak, Daneva, & Shamshirband. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*.

Rogers, Sharp, & Preece. (2011). *Interaction Design: Beyond Human-Computer Interaction.*

Torgeir, Sridhar, VenuGopal, & Nils Brede. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*.

# Appendix

**Video Demonstration:** [Link to video demonstration of the prototype]

**Repository:** <https://github.com/AyeshaSaddiqa001/BankingSystem_1>

**User Stories:** see stories.md on Github.