**Table of Contents**

1. Abstract
2. Introduction
3. Requirements Engineering
   * Functional Requirements
   * Non-Functional Requirements
   * Requirement Gathering Techniques
   * Use Case Diagrams & User Stories
4. Version Control Setup and Collaboration
   * Git & GitHub Setup
   * Branching Strategies
   * Collaboration on GitHub
5. Software Design and System Modeling
   * UML Diagrams Overview
   * System Architecture
   * Image Analysis & Explanation
6. User Interface (UI) Prototyping
   * Wireframing & Prototyping
   * Figma Implementation
   * Image Analysis & Explanation
7. Software Project Management
   * Project Planning & Scheduling
   * Resource Allocation
   * Risk Management
   * Monitoring Progress
8. Conclusion
9. References

**Abstract**

The rapid evolution of **FinTech (Financial Technology)** has led to increased demand for secure, scalable, and user-friendly mobile financial applications. This project aims to develop a **FinTech mobile application prototype**, integrating advanced features such as **money transfers, transaction history, balance inquiries, and secure authentication**. The development follows structured phases, including **requirements engineering, version control management, system modeling, UI prototyping, and project management**. GitHub serves as the primary collaboration tool for version control, while Figma ensures a seamless user interface design. This documentation provides an in-depth breakdown of the project's framework, ensuring clarity and consistency in execution.

**Introduction**

FinTech applications revolutionize financial transactions by enhancing speed, security, and accessibility. This project focuses on developing a **mobile prototype** tailored for seamless transactions, incorporating **efficient software design, real-time data processing, and interactive UI elements**. By leveraging **UML-based system modeling**, **GitHub-based collaboration**, and **Figma-driven UI prototyping**, the project ensures structured development aligned with industry best practices.

**Requirements Engineering**

**Functional Requirements**

The core features define how users interact with the system:

* **User Registration & Authentication** – Secure login system with multi-factor authentication.
* **Balance Inquiry** – Instant account balance retrieval.
* **Transaction History** – Detailed breakdown of financial activities.
* **Money Transfers** – Secure sending and receiving of funds.
* **Payment Integration** – Third-party payment system compatibility.

**Non-Functional Requirements**

Non-functional aspects define system performance and security:

* **Performance Optimization** – Fast processing times with minimal latency.
* **Scalability** – Ability to handle increased user base.
* **Security** – Strong encryption methods for data protection.
* **Availability** – 24/7 uptime with failover mechanisms.
* **User Experience** – Smooth, intuitive interface with responsive design.

**Requirement Gathering Techniques**

To ensure the mobile application meets user needs, various techniques were employed:

* **Surveys & Questionnaires** – Direct user feedback on financial needs.
* **Interviews** – Insights from potential customers and FinTech industry experts.
* **Focus Groups** – Testing usability aspects with a diverse audience.

**Use Case Diagrams & User Stories**

Use case diagrams outline interactions between users and the system, while **user stories** define specific **feature expectations** and **workflow experiences** to ensure a **user-friendly design**.

**Version Control Setup and Collaboration**

**Git & GitHub Setup**

* Created project repositories on **GitHub (**GitHub Account**)**.
* Established **main branch**, along with **feature and bugfix branches**.
* Enabled **repository permissions** for effective collaboration.

**Branching Strategies**

A structured approach ensures smooth version control:

* **Feature Branches** – New functionalities are developed separately before merging.
* **Pull Request Reviews** – Code is reviewed before integration into the main branch.
* **Continuous Integration** – Automated checks to verify code quality.

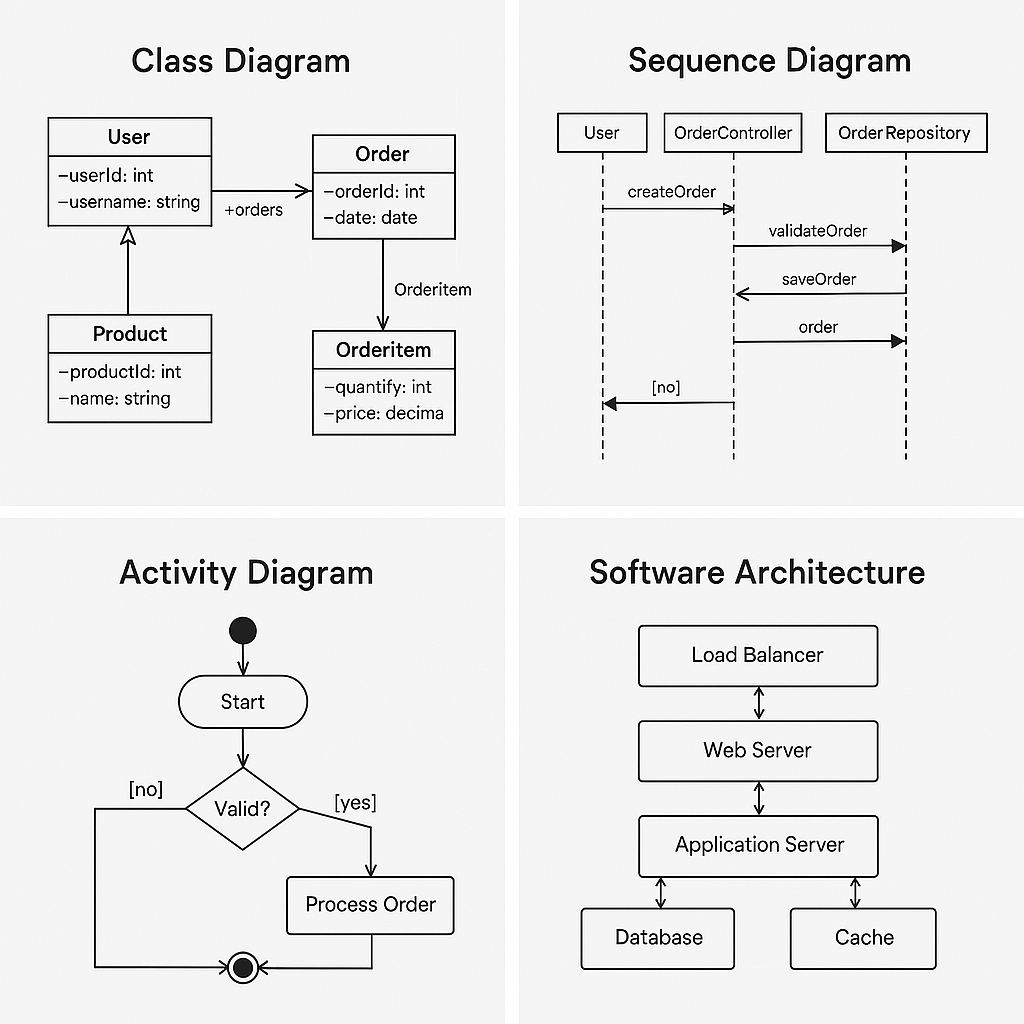
**Collaboration on GitHub**

* Managed tasks using **GitHub Issues and Kanban boards**.
* Maintained version control through **regular commits and structured merges**.
* Tracked project progress with **activity logs and milestone updates**.

**Software Design and System Modeling**

The following **UML diagrams** illustrate system architecture and interactions.

**System Modeling Image:**

****

**Image Analysis & Explanation**

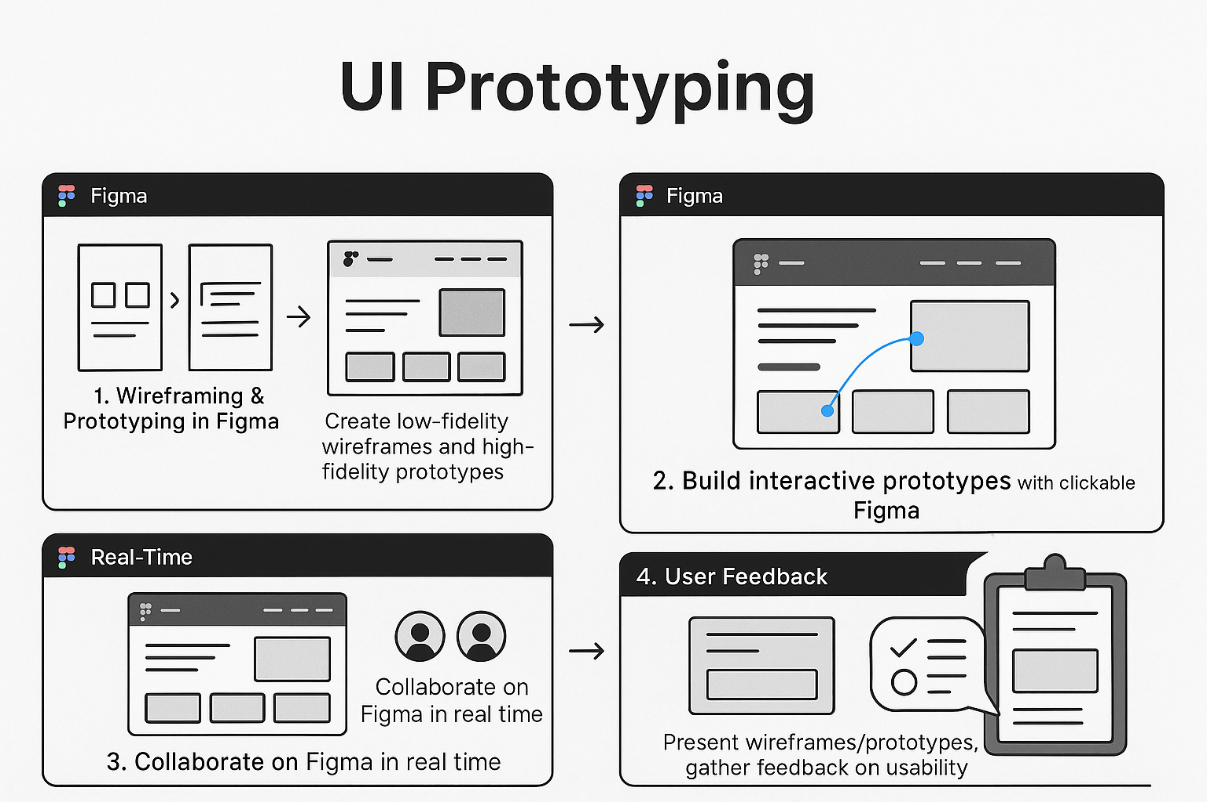
The UML diagram represents the core structure of the application, including:

* **Class Diagrams** – Defines relationships among entities (e.g., User, Account, Transaction).
* **Sequence Diagrams** – Illustrates transaction workflows and user interactions.
* **Activity Diagrams** – Ensures smooth decision-making processes within the system.
* **Software Architecture Diagrams** – Showcases system layering for scalability and security.

**User Interface (UI) Prototyping**

UI prototyping ensures an intuitive user experience using **Figma**.

**UI Prototyping Image:**

****

**Image Analysis & Explanation**

The UI wireframe showcases:

* **Navigation flow for transaction management**.
* **Interactive UI elements for seamless experiences**.
* **Responsive design principles ensuring cross-device adaptability**.

**Software Project Management**

**Project Planning & Scheduling**

Effective planning ensures timely delivery:

* Developed a **Gantt chart** mapping tasks and deadlines.
* Created a **Work Breakdown Structure (WBS)** to assign specific roles.

**Resource Allocation**

* Team members assigned based on their expertise (UI/UX, development, testing).

**Risk Management**

Potential risks and mitigation strategies:

* **Security Threats** – Implementing encryption and secure authentication.
* **Scalability Issues** – Load balancing and cloud-based expansion.
* **Development Bottlenecks** – Agile methodology for iterative improvements.

**Monitoring Progress**

Continuous monitoring through:

* **Regular sprint reviews**.
* **Task adjustments based on performance metrics**.
* **Stakeholder feedback for iterative refinements**.

**Conclusion**

This FinTech mobile application prototype integrates **software engineering principles, structured design, and intuitive UI elements**, ensuring **security, scalability, and seamless financial transactions**. By leveraging **UML modeling, GitHub collaboration, and Figma prototyping**, the project delivers an efficient and user-friendly platform optimized for mobile finance. Through structured planning and risk mitigation, the team ensures smooth execution and industry-standard development.

**References**

* Sommerville, I., 2011. Software Engineering. 9th ed. Pearson.
* Pressman, R.S., 2014. Software Engineering: A Practitioner's Approach. 8th ed. McGraw-Hill.
* Fowler, M., 2004. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley.
* Nielsen, J., 1993. Usability Engineering. Morgan Kaufmann.