Software Development Report – Online Banking System

1. Introduction

Purpose

This report outlines the design, implementation, and validation of a secure and scalable online banking system developed using Firebase. The project provides a responsive, user-friendly interface for financial transactions, enhancing accessibility and security in banking operations.

Definitions

- Firebase: A Backend-as-a-Service (BaaS) offering real-time database, authentication, and cloud functions.
- Authentication: Identity verification mechanism.
- Authorization: Determines access rights and roles.
- Transactions: Money transfers or account-related financial operations.
- UAT (User Acceptance Testing): Testing to validate user requirements.

Goals and Objectives

- Develop a secure online banking solution.
- Implement real-time features using Firebase.
- Ensure compliance with data protection standards.
- Provide intuitive UI/UX for ease of use.
- Deliver high availability and scalability.

Project Plan

1. Requirements Analysis
2. System Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

---

2. Requirements

User Requirements

- Create and manage accounts.
- View balances and transaction history.
- Transfer funds

System Requirements

Technical:

- Web-based, responsive design.
- Integration with Firebase services.
- Real-time synchronization.
- RESTful API support.

Security:

- End-to-end encryption.
- Secure session handling.
- Audit trails for all operations.

---

3. Software Process

Development Phases

- Planning: Resource and scope definition.
- Analysis: Gathering functional and non-functional requirements.
- Design: Architectural and UI design.
- Implementation: Coding and service integration.
- Testing: Multi-level validation.
- Deployment: Live system launch and documentation.

Methodology: Agile

- Iterative 2-day sprints.
- Continuous delivery and feedback loops.

- Regular reviews and retrospectives.

---

4. Architectural Design

System Architecture

Frontend:

- Developed in Svelte.js.
- Responsive, SvelteKit with dynamic routing.
- Integrated form validation and error handling.

Backend (Firebase):

- Authentication: Secure login & role-based access.
- Firestore: For storing user profiles, accounts, transactions.
- Cloud Functions: For transaction processing and logic.
- Real-time Database: Live updates for transactions and notifications.

Security Layer:

- HTTPS enforcement.
- Firebase rules for role-based data access.
- JWT session tokens.
- Monitoring via Firebase Analytics.

Data Flow

- Login Flow: User → Firebase Auth → Token → Secure Session.
- Transaction Flow: Form → Validation → Firestore update → Real-time sync.

Scalability

- Firebase's auto-scaling services handle load.
- Modular components for easy updates.
- Backup and recovery via Firebase's managed backups.

---

5. Design & Implementation

UI/UX Design

- Clean layout with Material UI components.
- Mobile-first responsive design.
- Clear navigation structure.

Database Design

- Users: Profile info, auth UID.
- Accounts: Balance, status, user reference.
- Transactions: Type, amount, timestamp, references.
- Audit Logs: Action, actor, time, metadata.

API Design

- /auth/login, /auth/signup
- /accounts/:id, /transactions/send, /transactions/history
- Consistent REST patterns with error handling.

Development Tools

- GitHub: Source control.
- Firebase Console: Backend management.
- VS Code: IDE.
- SvelteKit: Frontend framework for building fast, modern web apps with server-side rendering (SSR), routing, and optimized bundling.
- Svelte Devtools: Component inspection and reactive state tracking.

---

6. Testing

Development Testing

- Unit Testing: Jest, React Testing Library.
- Integration: Firebase Auth + Firestore tests.
- Performance: Lighthouse, Firebase Profiler.
- Security Checks: Manual input validation, Firebase rules tester.
- Form Testing

Release Testing

- System Testing: End-to-end on all user flows.
- Security Audit:
    - Firebase security rule evaluation.
    - Static code analysis.
- Load Testing: Simulated traffic with 10+ sessions.
- Regression Testing: Before every sprint release.

Documentation

- Technical guide (README, API Docs).
- User manual.
- Maintenance plan.
- Release logs and change history.

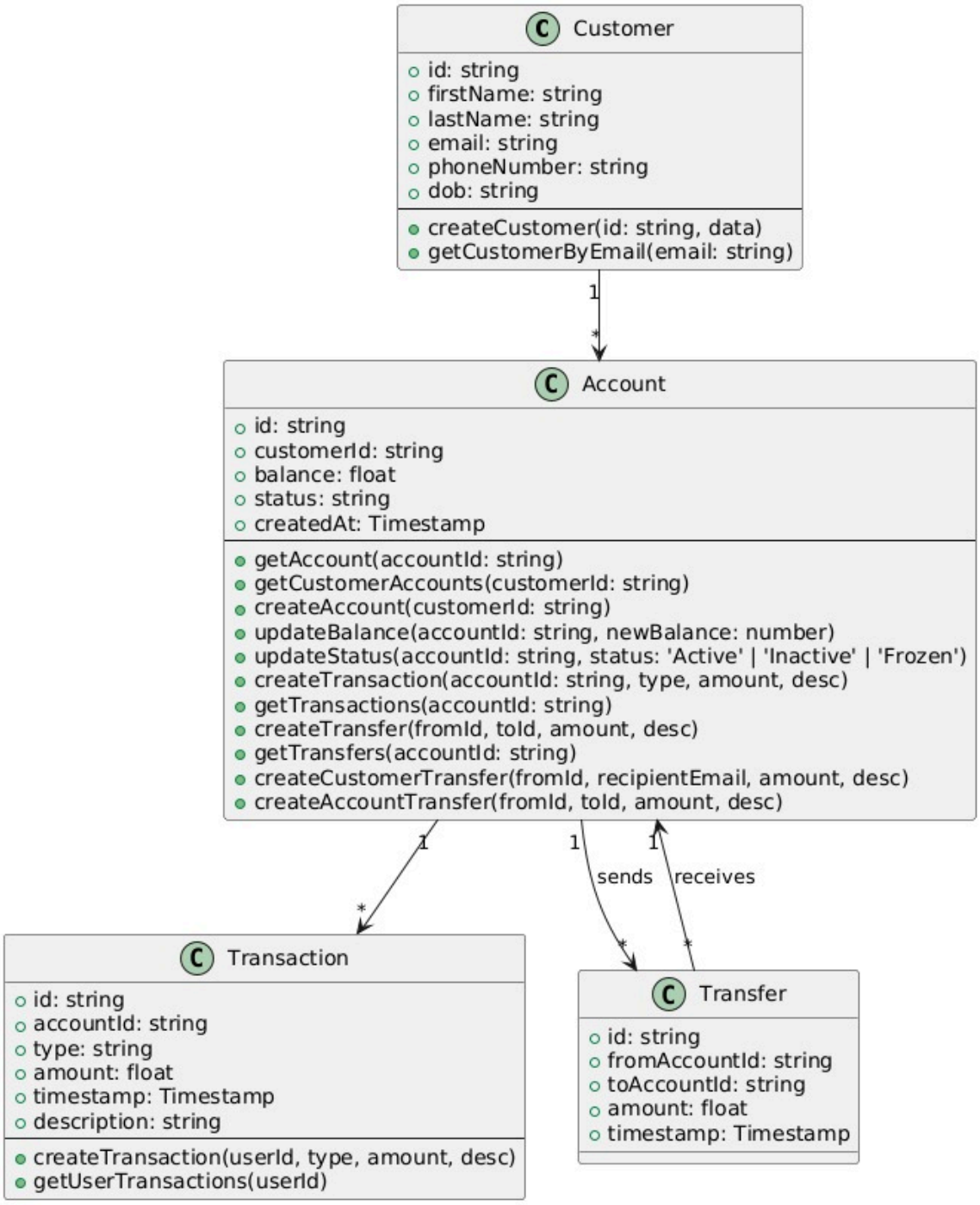
---

7. Conclusion

This banking system project demonstrates the integration of modern web development with cloud-native services like Firebase to build a secure, efficient, and user-friendly digital banking platform. The architecture supports future scalability and ongoing feature enhancements. Agile methodology ensured flexible, iterative development aligned with stakeholder needs.

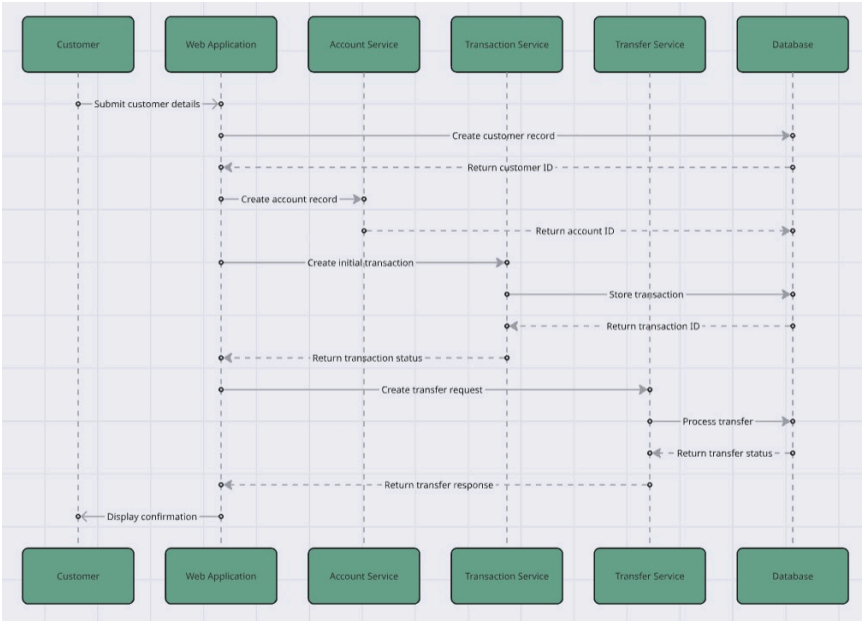
UML

1. Class Diagram



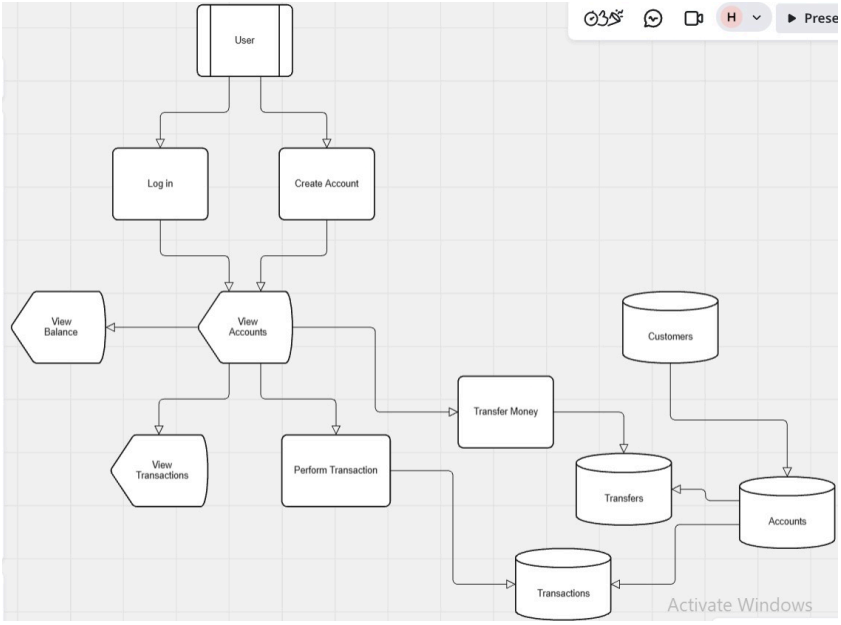
Description: This class diagram represents the backend structure of a banking system that allows user account management, balance tracking, money transactions, and transfers.

2. Sequence Diagram



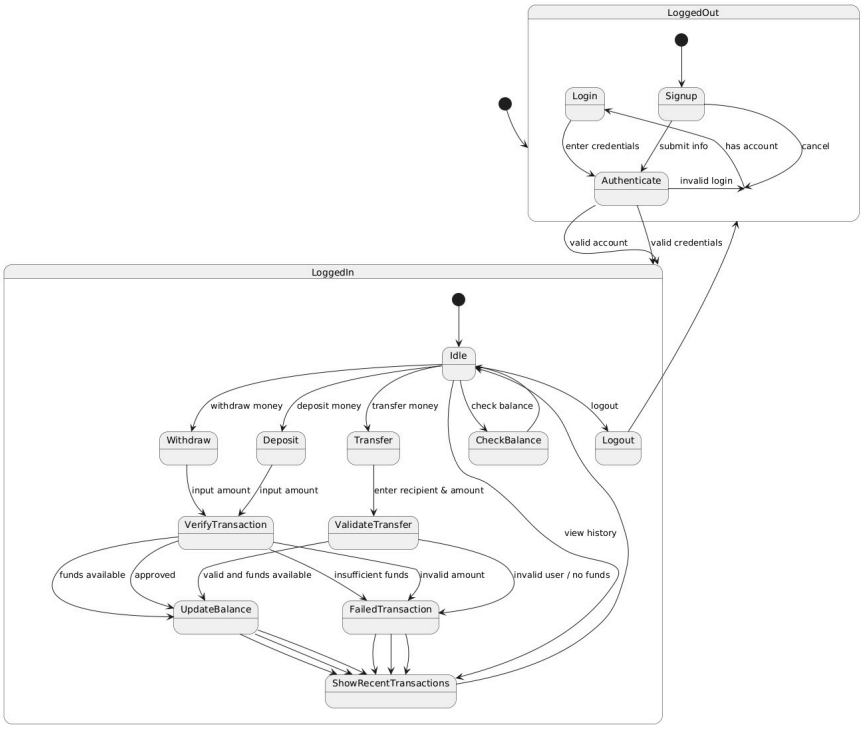
Description: This sequence diagram represents the login part of the website and how the client communicates with the server and how tokens are transferred and verified.

3. Activity Diagram



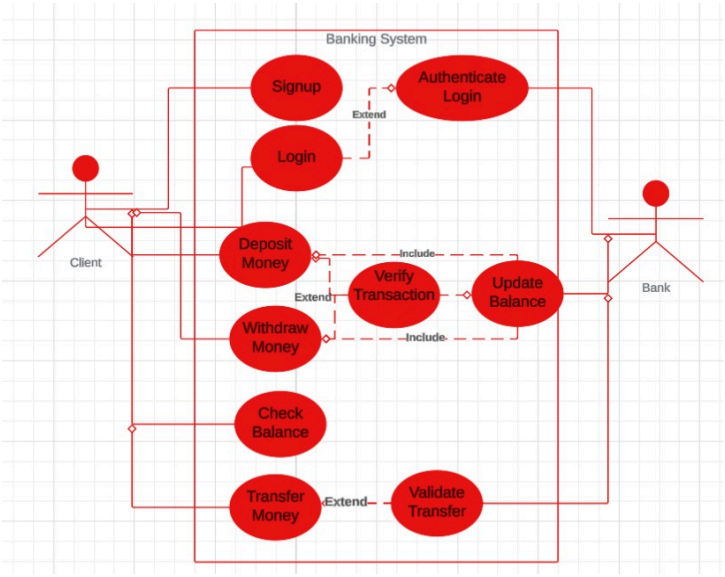
Description: This activity diagram explains the website, from the start at the landing page to logging in and all the options available once you are logged in.

4. State Machine Diagram



Description: Like the sequence diagram, this state diagram explains the difference in states from when you are just browsing without being logged in and from when you log in and what you can do and see after being logged in.

5. Use Case Diagrams



Description: This is a simplified version of the website, it shows what the user does and what the bank does and how they are connected and what they show to each other.

6. Mind Map/Flowchart

Description: This mind map explains how each part interacts with the server and what is displayed for the client.