Bank System

Requirement Document

Version	**V0.5.1**
Author	**Linshu Yang**
Group	**Team 5**
Date	**2022/5/29**

Catalog

1. Change Log

Version	Commit	Date	Author
V0.1.0	First drafts	2022/3/18	Linshu Yang
V0.2.0	Update backend requirement	2022/3/26	Linshu Yang
V0.2.1	Change client-server architecture	2022/4/15	Linshu Yang
V0.2.2	Update server requirement	2022/4/29	Linshu Yang
V0.2.3	Rewrite and polish the whole document	2022/5/15	Linshu Yang
V0.2.4	Change virtual hardware design	2022/5/16	Linshu Yang
V0.3.0	Change frontend structure	2022/5/17	Linshu Yang
V0.3.1	Create matview framework	2022/5/18	Linshu Yang
V0.3.2	Update matview framework	2022/5/19	Linshu Yang
V0.4.0	Extend matview framework	2022/5/20	Linshu Yang
V0.5.0	Change and upgrade UI design	2022/5/23	Linshu Yang
V0.5.1	Change the frontend bill design	2022/5/29	Linshu Yang

2. Introduction

The Bank System is a distributed online banking service system, providing multiplatform access and end-to-end property management services including depositions, withdrawals and transference. It eases the management of databases for bank personnel and brings convenience for clients of the bank.

2.1 Project Scope

The main objective of this project is to build a MATLAB based bank system simulating the pipelines and management of real-life banks, which involve the control of cash flow, identity identification, database safety maintenance and multiplatform user interface design. And create a convenient and easy-to-use application for bank personnel to maintain their business and clients to manage their properties.

2.2 Intended Audience

This project is a prototype for the Bank System, and it is restricted within the university premises. It will be implemented under the guidance of university professors and teaching assistants. In spite of those, this project is useful for real bank managers as well as their clients.

2.3 Concepts

Following concepts will be used in this document to give a clearer explanation of the project.

User	Client of the bank
Admin	Staffs and managers of the bank
ATM	ATM machines of the bank.
APP	Mobile phone applications of the bank.
Frontend	Client applications including ATM and APP
Backend	Database and server applications
User Id	ID number on client's card for user to login
Admin Id	ID number for admins to login

3. Requirements

3.1 Functional Requirements

3.1.1 Basic Requirements

The Bank System in its first release will contain basic functions for users to access their account and manage their properties and admins to maintain their business.

- **3.1.1.1** It should allow admins to create accounts for users.
- **3.1.1.2** It should provide access for admins to deactivate accounts when in needed.
- **3.1.1.3** Admins should be able to activate accounts when it is created or deactivated.
- **3.1.1.4** Its database should be able to run health check automatically.
- **3.1.1.5** Its server should allow multiplatform simultaneous requests.
- **3.1.1.6** It should allow users to login both on ATM and on APP.

- **3.1.1.7** It should provide users access to their bills and allow users to save them both on ATM and on APP.
- **3.1.1.8** It should be able to provide transference services both on ATM and on APP.
- **3.1.1.9** Its ATM should provide deposit and withdraw service for users.
- **3.1.1.10** Its ATM should give notifications when it runs out of cash.

3.1.2 Safety Requirements

System safety and security is of great significance in finance-related software development. Hence, strict requirements in this field will be made to our Bank System.

- **3.1.2.1** Its database should verify the validity of users' accounts.
- **3.1.2.2** It should be able to validate the user id when users making login.
- **3.1.2.3** It should be able to check the validity of cash when users making depositions.
- **3.1.2.4** It should not allow over-transference or over-withdraw at any time.
- **3.1.2.5** Its server should give notifications to admins when an error occurs.
- **3.1.2.6** Its server should send response without waiting too long or otherwise an error message.
- **3.1.2.7** Users will be required to activate their APP when they first login on their phones.
- **3.1.2.8** Its frontend should function well with and without the backend is functioning.
- **3.1.2.9** It should have upper limits for users' daily withdrawals and transferences.
- **3.1.2.10** It should not allow the user to stay logged in for too long.

3.1.3 Optional Requirements

Optional requirements cover the function that will not be accomplished in early releases and may or may not be added to the Bank System in the future.

- **3.1.3.1** It should allow admins to edit the database manually.
- **3.1.3.2** It should provide interest calculations for users.
- **3.1.3.3** It should allow users to report the loss of their cards.
- **3.1.3.4** Admins should have an interface to carry out operations.
- **3.1.3.5** Admins should have the right to check the database and manually review any accounts they are interested in.
- **3.1.3.6** Users should be able to change their passwords.
- **3.1.3.7** Its APP should provide online payment functions.

3.2 Interface Requirements

3.2.1 ATM Interface

ATM Interface will simulate both hardware and software.

- **3.2.1.1** Basic hardware including a keyboard, a card slot, a cash draw and a screen should be provided.
- **3.2.1.2** It should have a login page after a card is injected and verified valid.
- **3.2.1.3** It should have a main menu page for users to change between different pages.
- **3.2.1.4** It should have independent pages for deposition, withdrawal and transference.
- **3.2.1.5** User can check and print their bills on the ATM.
- **3.2.1.6** Users can see the result and be asked to check bills after they make deposition, withdrawal and transference.
- **3.2.1.7** Users should be notified when it loses contact with the backend.

3.2.2 APP Interface

APP interface will only have software part since it based on users' mobile phones.

- **3.2.2.1** It should have a login page and an extra password box for admins to activate users' APP on their first login.
- **3.2.2.2** It should have a main menu for users to change between different pages.
- **3.2.2.3** It should allow users to make transference and show them the results.
- **3.2.2.4** Users should be able to check their bills and download them whenever they want.

3.3 Validation Requirements

- **3.3.1.1** Model checking since the early development of it.
- **3.3.1.2** Use-case-based Testing and Specification-based Testing.
- **3.3.1.3** Code checking, development review and other static checking methods.
- **3.3.1.4** System integration check and functionality check.

4. System Architecture

4.1 System Composition

The Bank System consists of independent frontend and backend, controlled by two different controllers, using the Model-View-ViewModel structure.

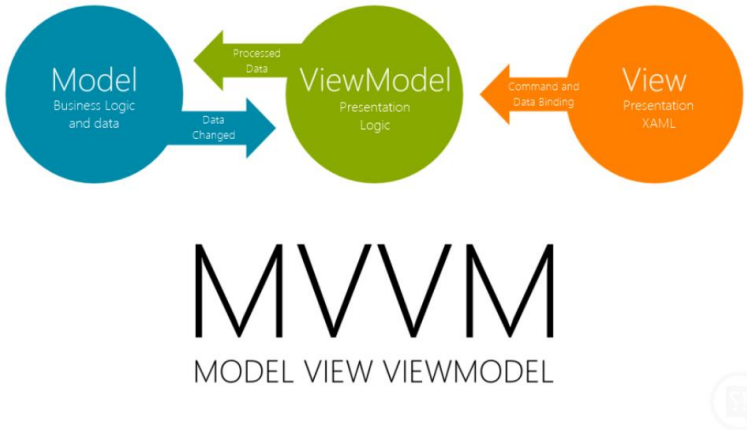

Fig. 4.1. MVVM Structure

Modules

1. Model
   The backend of the Bank System, receive requests from the frontend and maintain the database accordingly.
2. ViewModel
   The frontend controller of the Bank System, making communication with the backend and giving responses to users' requests.
3. View
   Present information and interfaces to users, make directly interactions with them.

4.2 Client-Server Architecture

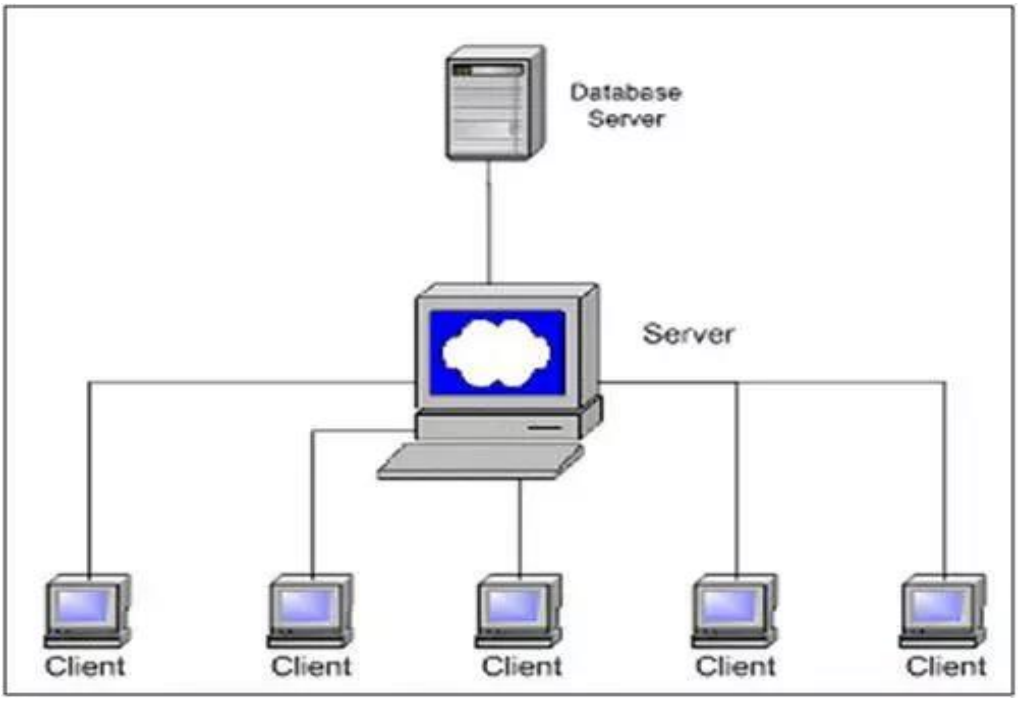

Fig. 4.2. Client-Server Architecture

The client-server model of the Bank System is a distributed application structure that partitions tasks or workloads between server, the providers of the resources and clients, the service requesters. In MATLAB environment, the client finishes the workload of user interactions and get resources from the server. And the server covers tasks related to the database, directly communicate to the database server (SQLite3 drivers) provided by MATLAB to maintain the database.

5. Design and Implementation

5.1 Product Features

The major features of the Bank System are listed below.

- Provide basic functions for users to access the bank and for admins to manage the system.
- Allow users to manage their properties and check their bills both on ATM and on APP, without going the bank.
- Use a real database to simulate the long-run data storage, able to preserve the data with and without the system working.
- User friendly interfaces provide a smooth interaction between it and users and

admins.

- Leave compositional APIs available for extensions and further development.

5.2 Class Diagram Design

Note that all the types mentioned below follow to the JavaScript type system.

5.2.1 Backend Class Design

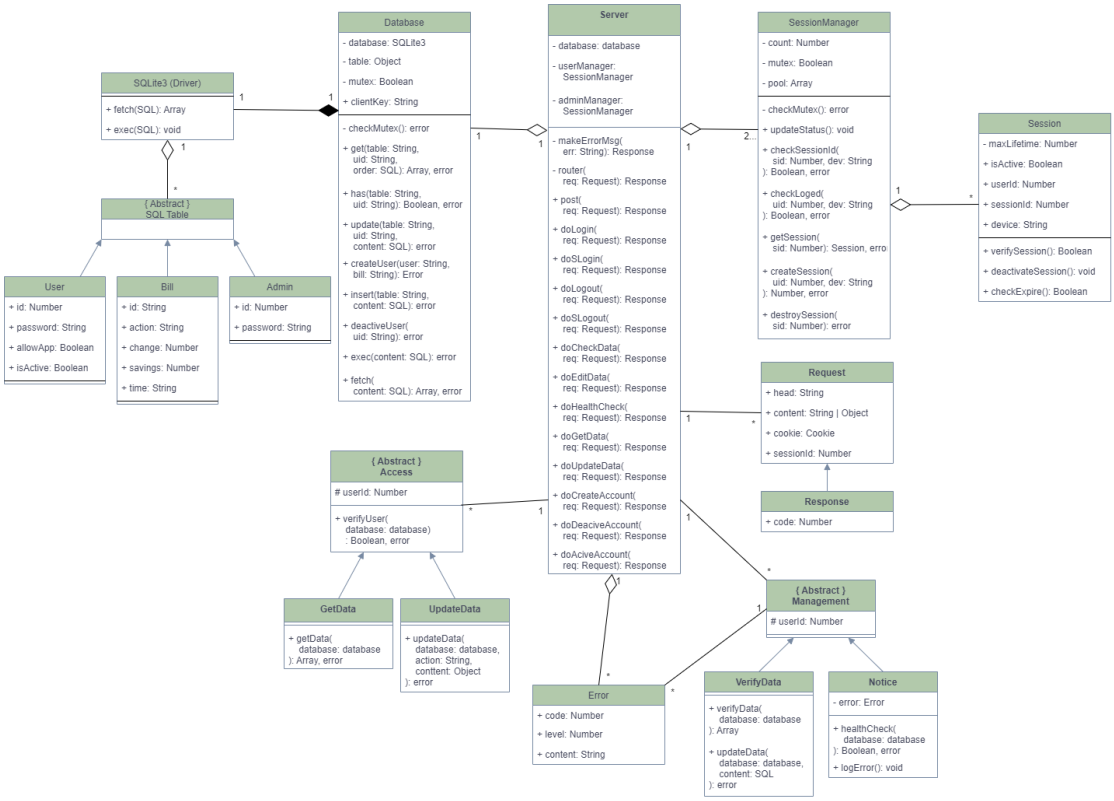

Fig. 5.1. Backend Class Diagram

The backend consists of database and session manager, as well as other middleware and function sets. In these parts, Database mainly handles its interaction with the SQLite3 embed database, and SessionManager manages users' and admins' login status. Access and Management are two function sets handling users' access to the database and admins' management to the database. Error, Request and Response mainly serves as a uniform interface for the server to interact with other components of the bank system.

5.2.2 ATM Class Design

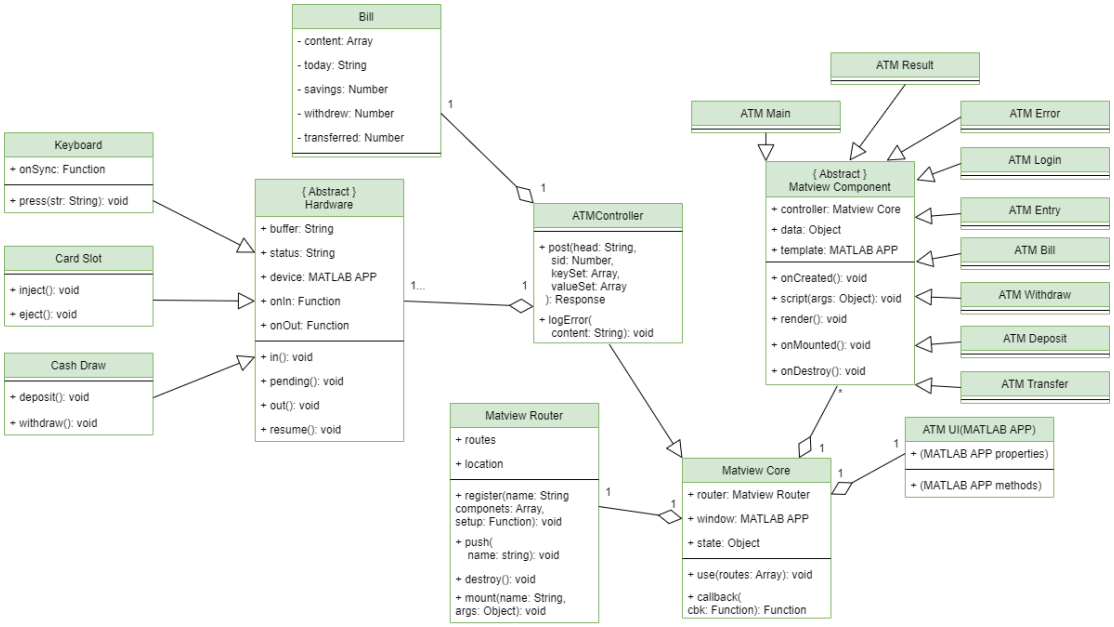

Fig. 5.2. ATM Class Diagram

ATM of the frontend is built based on the framework Matview made by us. It has Keyboard, CardSlot and CashDraw to simulate virtual hardwares, and Bill to parse raw bill data. Meanwhile, every page to be displayed to user is considered as a Matview component, created, scheduled and destroyed by MatviewRouter and MatviewController.

5.2.3 APP Class Design

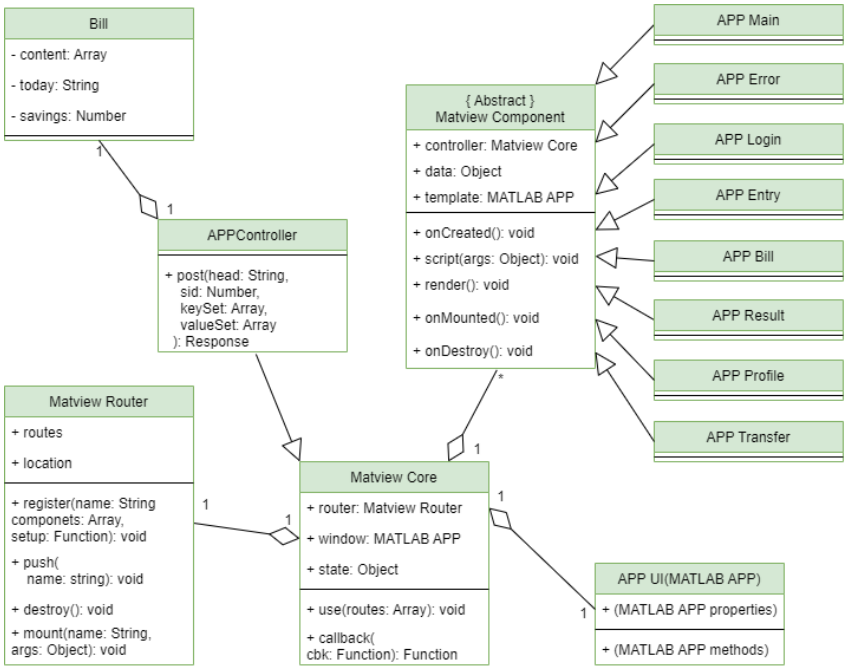

Fig. 5.3. APP Class Diagram

APP of the frontend is built based on the framework Matview made by us. It uses a Bill class to parse raw bill data. And every page to be displayed to user is considered as a Matview component, created, scheduled and destroyed by MatviewRouter and MatviewController.

5.3 Use Case Diagram

5.3.1 Backend Use Case Design

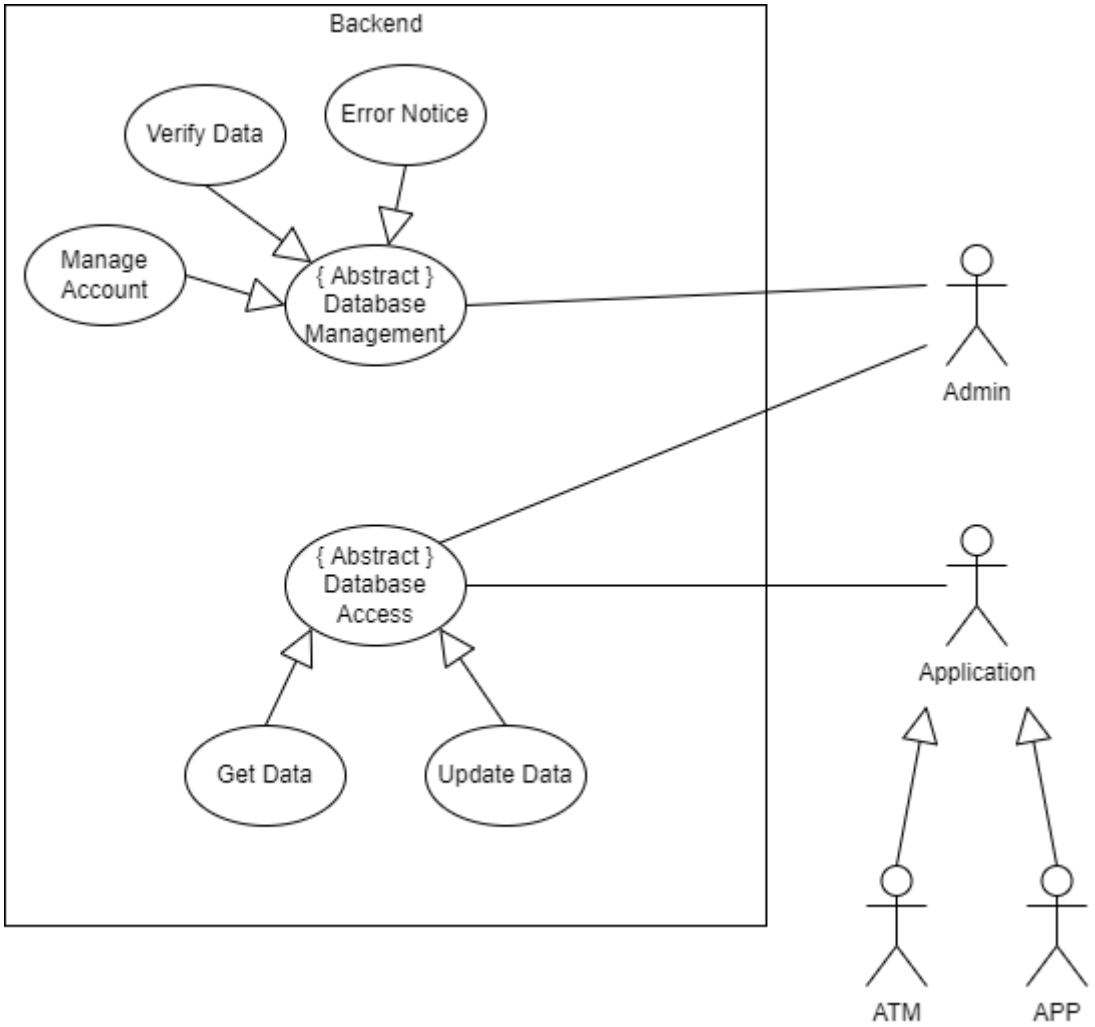

Fig. 5.4. Backend Use Case Diagram

Applications like ATM and APP access the backend to get data and update data from the database. Admins will access the backend to manage accounts and verify data, and they will receive error notice from the backend when an emergency happens. If possible, admins can access the backend by an admin application instead of raw access in future releases.

5.3.2 ATM Use Case Design

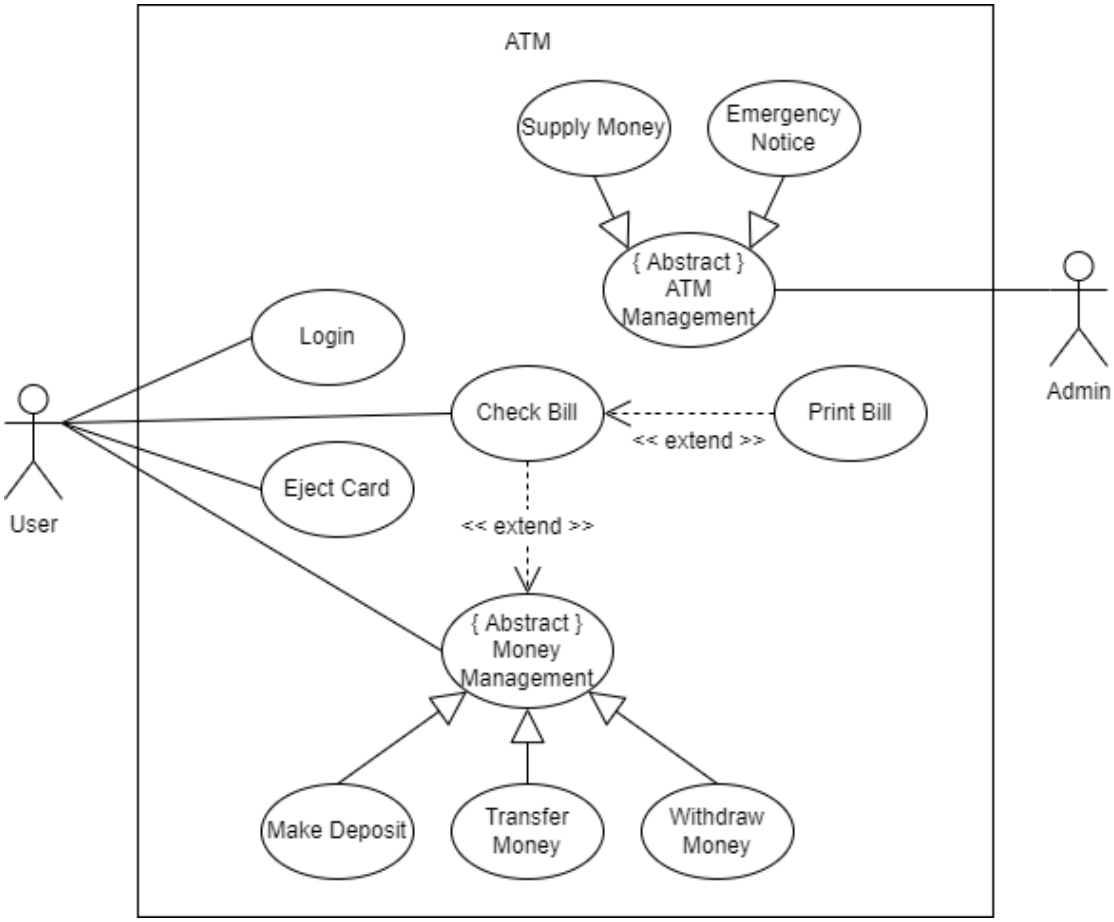

Fig. 5.5. ATM Use Case Diagram

User can login, check and print their bills, logout (eject card) and manage their properties including making depositions, transferring money and withdrawing money on the ATM. Admins will supply money to the ATM and receive emergency notice from it.

5.3.3 APP Use Case Design

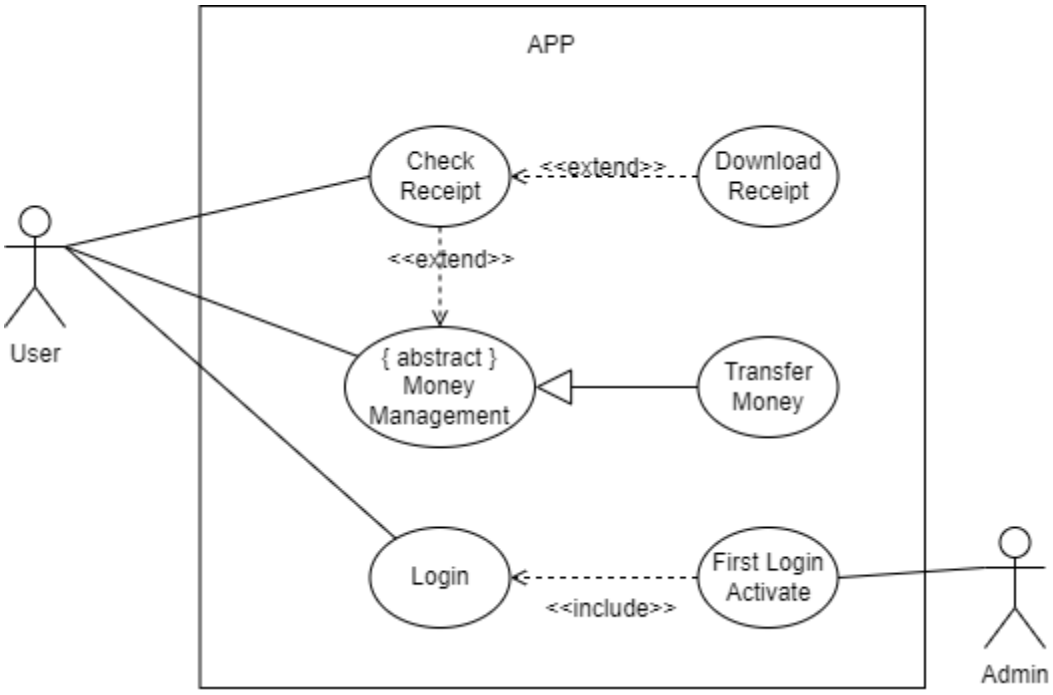

Fig. 5.6. APP Use Case Diagram

User can login, check and download their bills, logout (eject card) and making transferences. Admins will help users to make their first login activations on the APP.

5.4 Sequence Diagram Design

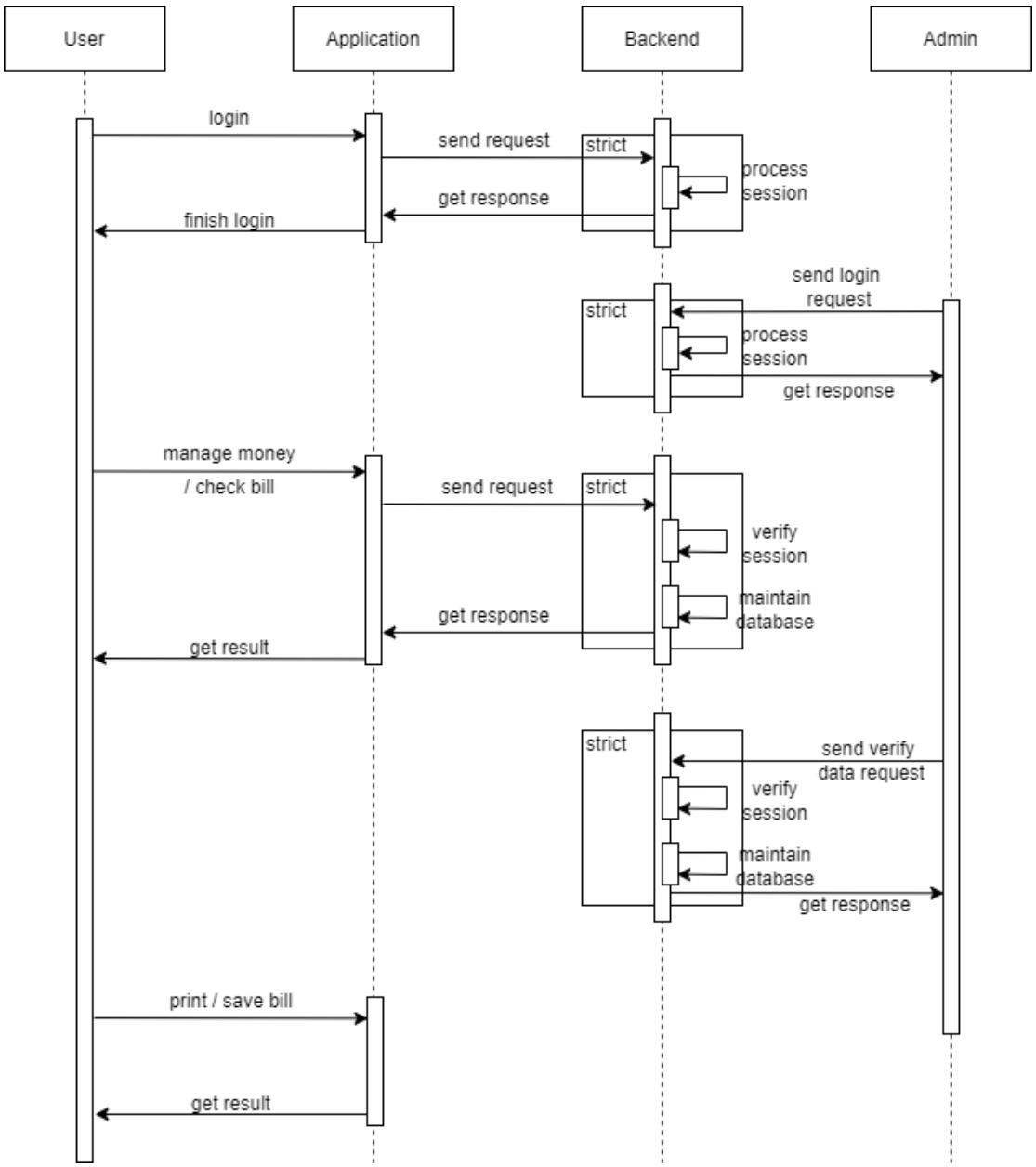

Fig. 5.7. Sequence Diagram

Users interact with the backend indirectly, with the help of the frontend. And extra services are also available for users. Admins currently interact with the backend directly.

5.5 Activity Diagram Design

5.5.1 Client-Server Activity Diagram

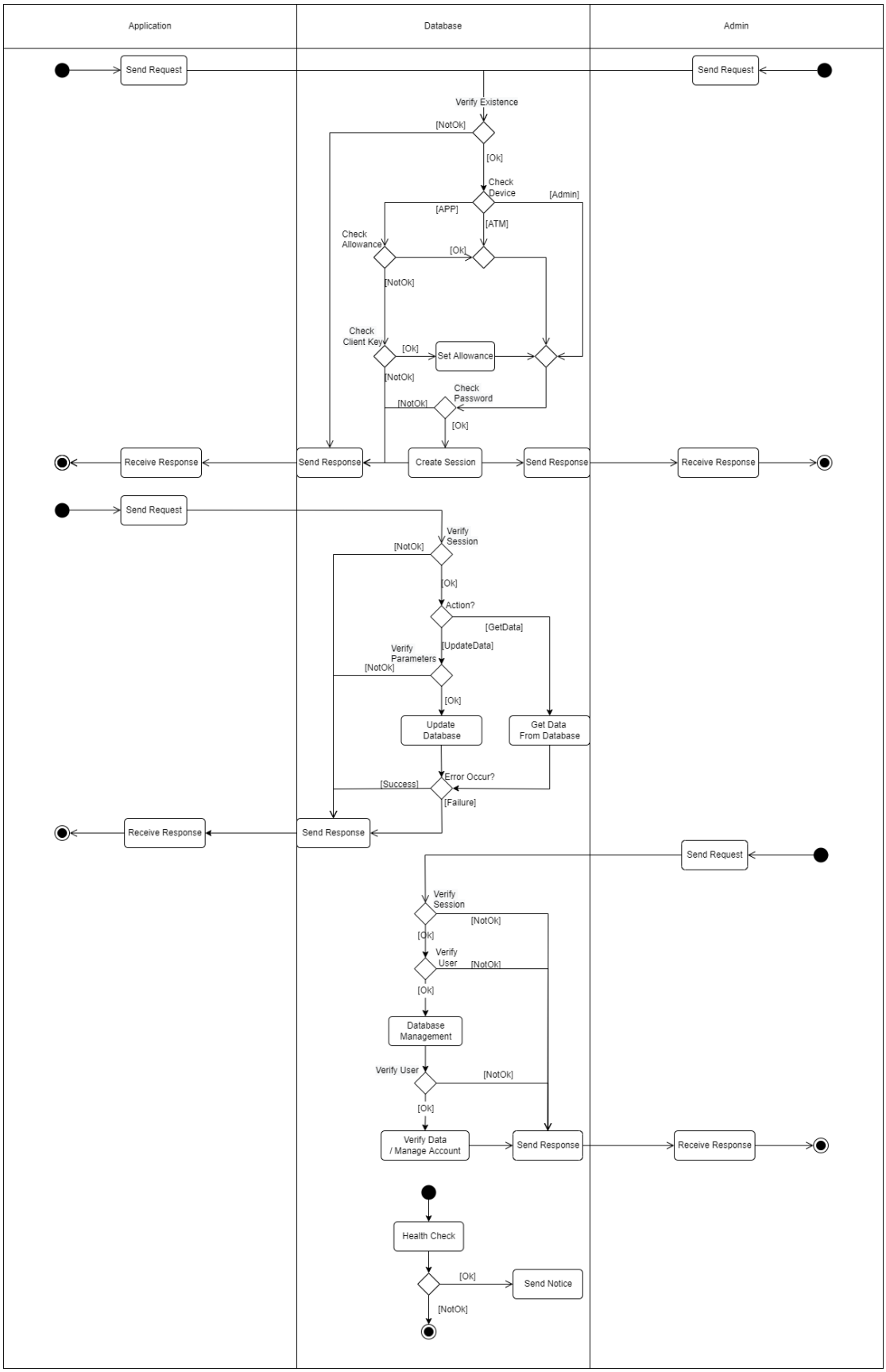

Fig. 5.8. Client-Server Activity Diagram

At the beginning of whole process, users and admins will have to login the system through restricted procedures via requests, and a session id will send back to them as a proof of identity via response. After that, users are able to get or update data with session id and get response from their actions. And admins can verify the data in the database and run health check for the system. Sessions will be deactivated when it reaches its max lifetime or user and admin logout.
Once an error occurs and some unusual incidents are detected in the automated health check, the system will send a notice via console to notify admins.

5.5.2 User-Client Activity Diagram

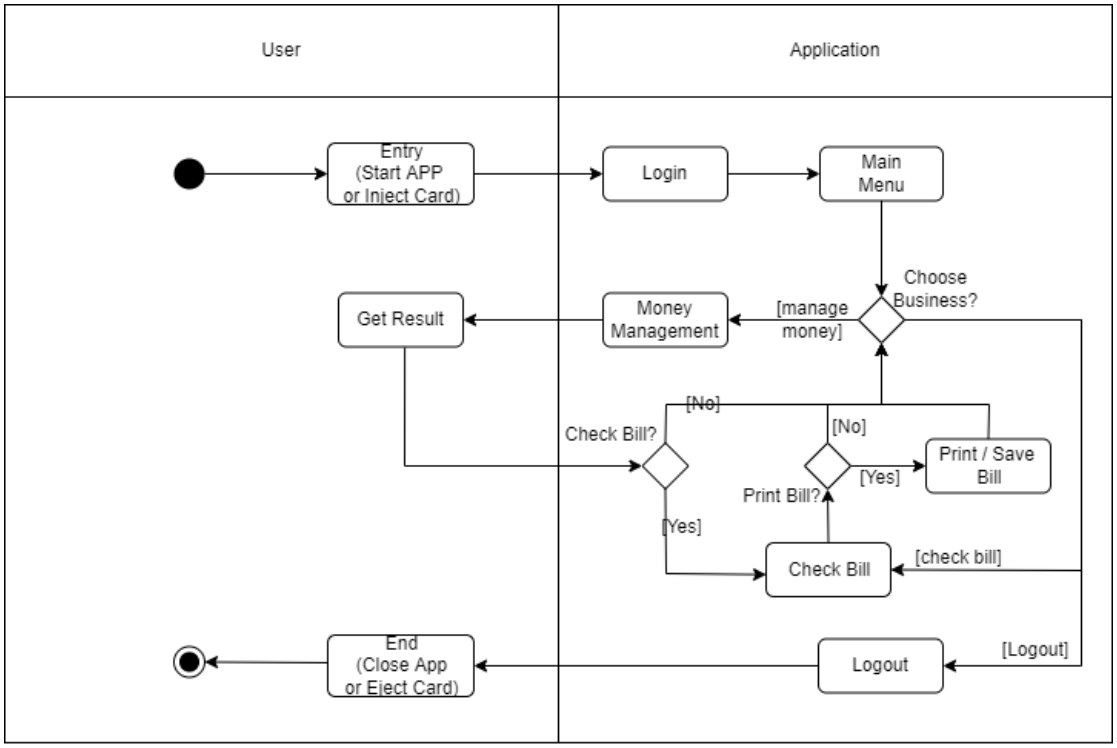

Fig. 5.9. User-Client Activity Diagram

After Inject the Card or start the APP, users are required to login first, then they can choose services they want at the main menu. If they choose to manage their properties, they can make transferences on their phone or transference, depositions and withdrawal on ATM. Then they will be asked to check their bills or return to main menu. After they checked their bills, they can choose to print or save their bills then return to main menu, or directly return to main menu. At the menu, they can also manually choose to check bills or logout.

5.6 Interface Design

Two independent user interfaces are built to provide services for users on ATM machines and on their mobile phones. Both interfaces give services on multiple pages

and are implemented on one page with a router to control the visibility of components.

5.6.1 ATM Interface

Fig. 5.10. ATM UI

5.6.2 APP Interface

Fig. 5.11. APP UI

6. Appendix

6.1 Resources

- ISO 11608-1:2014 - Needle-based injection systems for medical use — Requirements and test methods
- Software Requirements Specification Report for a Project - Krazytech
- diagrams.net

6.2 Contact

Please visit team5 / Proj1 · GitLab for more information