A V O C A D O T O A S T: A N O N L I N E

B A N K I N G P L A T F O R M

damh pham anandita dubey flaviu tamas
carlos deleon alex petros

contents

1 Planning Schedule and Peer Evaluation 2
2 Refining System 3
3 Requirements 6
3.1 User Requirements. 6
3.2 Use Cases 6
3.3 Use Case Requirements 7
3.4 System Requirements 9
4 System Modeling (Analysis) 10
4.1 Class Diagram. 10
4.2 Database Specification and analysis. 10
4.3 Architecture Modelling 10
4.4 Behavioral Modelling. 10
5 Implementation 11
5.1 Implement the Database Design (Backend): Overview 11
5.2 Implement the Class Diagram Design (Frontend): Overview 11
6 Testing 12
Appendix:Screenshots 14

list of figures

Figure 1	Screenshot of our group's Kanban board	14
Figure 2	Use case diagram	15
Figure 3	System class diagram	15
Figure 4	Architecture Modeling diagram (Client Server Pattern)	16
Figure 5	Behavioral Modeling diagram (UML sequence diagram)	17

1 Planning schedule and peer evaluation

Assignee	Email	Task	Dur.	Depends on	Due date	Evaluation
Anandita	adubey2	Refine System Architecture Modeling,	2h	Use cases	2019–03–03	Great job, did 100% of the task.
Alex	apetros1	Implementation (Backend)	2h	Frontend	2019–03–03	Great job, did 100% of the task.
Carlos	cdeleon1	Behavioral Model	2h	Use Cases	2019–03–02	Great job, did 100% of the task.
Danh	dpham16	Testing	2h	Implementation	2019–03–02	Great job, did 100% of the task.
Flaviu	ftamas1	Implementation (Frontend)	2h	None	2019–03–02	Great job, did 100% of the task.

Table 1: Work breakdown

2 Refining system

2.0.1 *What is your product, on a high level?*

Our product is an online banking application, which will allow the customers of a bank or other another financial institution to perform money-management without physically visiting the bank. It will allow the customers to open their accounts, manage them electronically, to monitor them, to make transactions, pay their bills, transfer money, make deposits, and so on.

2.0.2 *Whom is it for?*

An online banking application is beneficial to everyone but is especially useful for those people with a stringent work schedule. It will help them to manage their accounts and keep track of their activities in a quick manner with minimal costs without the need to visit a physical bank during working hours or make a phone call.

2.0.3 *What problem does it solve?*

24/7 availability, saving the customer from rushing the banks during working hours. With an online banking system, people can perform their tasks at a time that suits their work schedule.

Stringent schedules, where customers with strict working hours to perform their banking activities effectively and conveniently.

Centralized source of information, where instead of visiting different officials specialized in different tasks, the customer can use an online banking applica-tion flexible single enough to do any task in a click. Human bankers are not always available, but the application always will be, meaning there is no need to rush to the bank to get things done.

Remain informed: With the online banking system, customers can easily receive up-to-date information regarding their upcoming deadlines or dues through notifications, emails, or text messages.

Easy bill payments, so that there is no need to rush to the bank. Everything can be done at home instead.

2.0.4 *What alternatives are available?*

- Services provided by 3rd party application,

  like Mint A phone app

- In person banking

  services

- Phone calls
- ATMs

2.0.5 *Why is this project compelling and worth developing?*

- It would reduce costs for banks by reducing the amount of human labor re-quired.
- An electronic banking system would enable banks to keep stringent records
- A computerized ledger would reduce the number of mistakes when calculating interests, making transactions, and so on.
- It would increase customer satisfaction because they would have access to our banks from any place with WiFi.
- Providing flexibility to the customers to get their work done at minimal or no cost.
- Saving time of the customers (customer need not necessarily visit the bank physically)

2.0.6 *Describe the top-level objectives, differentiators, target customers, and scope of your product.*

- Objective: To create a product that performs to our requirements and have it sustainable for multiple lifecycles.
- Differentiators: Our system won't sell data to any third parties, have no hidden transaction fees, and will not participate in predatory lending practices
- Target customers: Our target customer is the average citizen, anyone that currently uses or will use a bank for their transaction needs. (Basically useful to everyone, no matter what)
- Scope: The scope of our project is building an online business ledger, making a database to store our information, developing a web app and making a UI for our customers.

2.0.7 *What are the competitors and what is novel in your approach?*

Our competitors are Finacle, nCino, Oracle, etc. What we bring new to the table is that our system will be entirely online. Additionally, our platform is customer focused and will be fully transparent. We won't sell customer data or charge hidden fees. Additionally, providing the customers with the flexibility to perform the tasks at the minimal or no cost.

2.0.8 *Make it clear that the system can be built, making good use of the available resources and technology.*

Our system is possible because it will be very simplistic and direct. Clients will connect to our application software where they will be greeted by a user-friendly interface, which will primarily be coded in HTML, with CSS used to create a beautiful design for our clients. Finally, the frontend of our application will be connected to a backend SQL database. The database will be responsible for recording transactions such as withdrawals, deposits, bill pays, and more.

2.0.9 *What is interesting about this project from a technical point of view?*

The project will use a client-server architecture in order to be accessible to clients from anywhere. Emphasis will also be placed on graphical design, using our creativity to design a nice interface for our clients. Providing flexibility to the clients to perform the non-transactional tasks through online banking. That might include:

- Viewing account balances
- Viewing recent transactions
- Downloading bank statements
- Downloading periodic account statements
- Funds transfer between the customer's linked accounts Paying third parties, including bill payments
- Credit card applications

- Register utility

  billers

3 Requirements

3.1 User Requirements:

Use Case Diagram

See Figure in Appendix

3.2 Use Cases

3.2.1 *Use Case Number: 1 Use Case Name: Register Customer*

- Actors: Banker
- Description: Add a new customer, with their own username and password
- Alternate Path: None
- Pre-Condition: Logged in

3.2.2 *Use Case Number: 2 Use Case Name: Register Customer*

- Actors: Banker
- Description: Take a certain amount of cash from the customer, adding the amount to their balance
- Alternate Path: None
- Pre-Condition: Logged in

3.2.3 *Use Case Number: 3 Use Case Name: Register Customer*

- Actors: Banker
- Description: Give the customer a certain amount of cash, subtracting the amount from their balance
- Alternate Path: None
- Pre-Condition: Logged in, account has sufficient balance

3.2.4 *Use Case Number: 4 Use Case Name: Register Customer*

- Actors: Banker, Customer
- Description: View a list of all the transactions in the account, as well as the overall account balance
- Alternate Path: None
- Pre-Condition: Logged in

3.2.5 *Use Case Number: 5 Use Case Name: Register Customer*

- Actors: Customer
- Description: Send someone else money electronically through the website
- Alternate Path: None
- Pre-Condition: Logged in, account has sufficient balance

3.2.6 *Use Case Number: 6 Use Case Name: Register Customer*

- Actors: Analyst
- Description: View the amount of money that was deposited and withdrawn in a single day, for the entire bank
- Alternate Path: Manual SQL access
- Pre-Condition: Logged in

3.2.7 *Use Case Number: 7 Use Case Name: Register Customer*

- Actors: Banker, Customer, Analyst
- Description: Authenticate themselves to the system
- Alternate Path: None
- Pre-Condition: None

3.3 Use Case Requirements

3.3.1 *Use Case Number: 1*

- Introduction: Register Customer, this system interacts with the user, authentication utility, SQL database
- Inputs: Username Password

- Requirements Description: Registers a new customer, they must input a user-name and password for their account
- Outputs: Their login information is logged into the system to be recalled later
- Alternate Path: None

3.3.2 *Use Case Number: 2*

- Introduction: Deposit Cash, this system interacts with the user SQL database
- Inputs: Cash
- Requirements Description: Insert cash from the customer, the amount is added into the balance
- Outputs: Cash amount is added into their balance
- Alternate Path: None

3.3.3 *Use Case Number: 3*

- Introduction: Withdraw cash, this system interacts with the user SQL database
- Inputs: Cash amount
- Requirements Description: Withdraw cash from the bank, the amount is de-ducted from the balance
- Outputs: Cash is given to the customer and cash amount is deducted from balance
- Alternate Path: None

3.3.4 *Use Case Number: 4*

- Introduction: View Ledger, this system interacts with the user SQL database
- Inputs: User PIN number
- Requirements Description: View a list of all the transactions in the account, as well as the overall account balance
- Outputs: Transaction record is displayed on the screen, with their account balance
- Alternate Path: None

3.3.5 *Use Case Number: 5*

- Introduction: Pay Bill, this system interacts with the user, SQL database, and the recipient
- Inputs: Cash amount and information of the recipient
- Requirements Description: Send a person or company money electronically through the website
- Outputs: Deducts cash amount from overall balance and cash is sent to recipient
- Alternate Path: None

3.3.6 *Use Case Number: 6*

- Introduction: View Transaction Volume, this system interacts with the user SQL database
- Inputs: User PIN number
- Requirements Description: Views the amount of money deposited and with-drawn in a single day
- Outputs: Shows deposit withdraw amount in specified day
- Alternate Path: Manual SQL Access

3.3.7 *Use Case Number: 7*

- Introduction: Log in, this system interacts with the user and authentication utility
- Inputs: User ID and password
- Requirements Description: Authenticate a user to their account
- Outputs: User is logged into the system and given access to use their account
- Alternate Path: None

3.4 *System Requirements:*

- CPU: Intel/AMD processor
- RAM: 2 GB(Recommended)
- OS: 64-bit Windows 7, Windows 8.1, Windows 10

4 System modeling (analysis)

4.1 Create system class diagram.

See Figure in Appendix

4.2 Database Specifications and analysis

4.2.1 Specify your system database tables (data attributes and their types) and relationship between them (Primary Keys and Foreign Keys)

- The first table would have 3 data types which include INT, DATE, and VARCHAR.
- INT Attributes: social security number, account number, account balance
- DATE Attributes: date of birth
- VARCHAR Attributes: first name, last name, email address, username, pass-word, account type
- The primary key would be the account number.
- The second table would have 2 data types which include INT and DATE.
- INT Attributes: transaction number, transaction amount, and account number
- DATE Attributes: transaction date
- The primary key is the transaction number.

4.2.2 Specify the type of database management system (MySQL, MS-SQL server, Oracle, etc.) you will use in your project

The database management system that will be used in the project will be MySQL.

4.3 Architecture Modeling

See Figure in the Appendix

4.4 Behavioral Modeling (Dynamic Modeling)

See Figure in the Appendix

5 Implementation

5.1 Implement the Database Design (Tables, Backend):

Overview

This server was generated by the swagger-codegen project.
By using the OpenAPI-Spec, you can easily generate a server stub.
This is an example of building a swagger-enabled server in Java using the SpringBoot framework.
The underlying library integrating swagger to SpringBoot is springfox
Start your server by running gradle run.
You can view the api documentation in swagger-ui by pointing to
http://localhost:8080/
Change default port value in application.properties

5.2 Implement the Class Diagram Design (Frontend and Logic): (develop/write code)

In the project directory, you can run:

`npm start`
Runs the app in the development mode.
Open http://localhost:3000 to view it in the browser.

The page will reload if you make edits.
You will also see any lint errors in the console.

`npm test`
Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

`npm run build`
Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.
The build is minified and the filenames include the hashes.
Your app is ready to be deployed!

`npm run eject`
Note: this is a one-way operation. Once you `eject`, you can't go back!
If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (Webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.

You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

6 Testing

Identify Features (Functionality):

1. Login
2. Send Money

Feature One, Login:

Partition Input**:** login(String U, string P)

- String U is a string for the user's Username
  - One possible partition is string with length < 0, string with length = 0, string with length > 0
- String P is a string for the user's Password
  - One possible partition is string with length < 0, string with length = 0, string with length > 0

Test Specification:

- Username: <String a-z, A-Z, 0-9, special characters>
- Password: <String a-z, A-Z, 0-9, special characters>

Test Case:

#1:

- Inputs:
  - Username: bobsob20!!
  - Password: po1po(@$
- Outputs:
  - Logs in

#2:

- Inputs:
  - Username:
  - Password:
- Outputs:
  - Login denied: please input username and password

Feature Two, Send Money:

Partition Input: sendMoney(String ID, int N)

- String ID is a string for the recipient's Username
    - One possible partition is string with length < 0, string with length = 0, string with length > 0
- double N is an integer between 0 and the user's maximum balance
    - One possible partition is a number: < -∞ - 0.00,0.00-∞, 0.00-…*>, …* is the user's maximum balance

Test Specification:

- Recipient's ID: <String a-Z, A-Z, 0-9, special characters>
- Amount: <0…*>, * is the user' maximum balance

Test Case:

Assume user has $10 in balance

#1:

- Inputs:
    - ID: bobross
    - Amount: $3.01
- Outputs:
    - Successful, money is sent to recipient bobross

#2:

- Inputs:
    - ID:
    - Amount: $3.01
- Outputs:
    - Denied, please identify recipient

#3:

- Inputs:
    - ID: bobross
    - Amount:$ -3.01
- Outputs:
    - Denied, value is a negative double

#4:

- Inputs:

- ID: bobross
- Amount: $500

- Outputs:
  - Denied, value exceeding user's maximum balance

Appendix: Screenshots

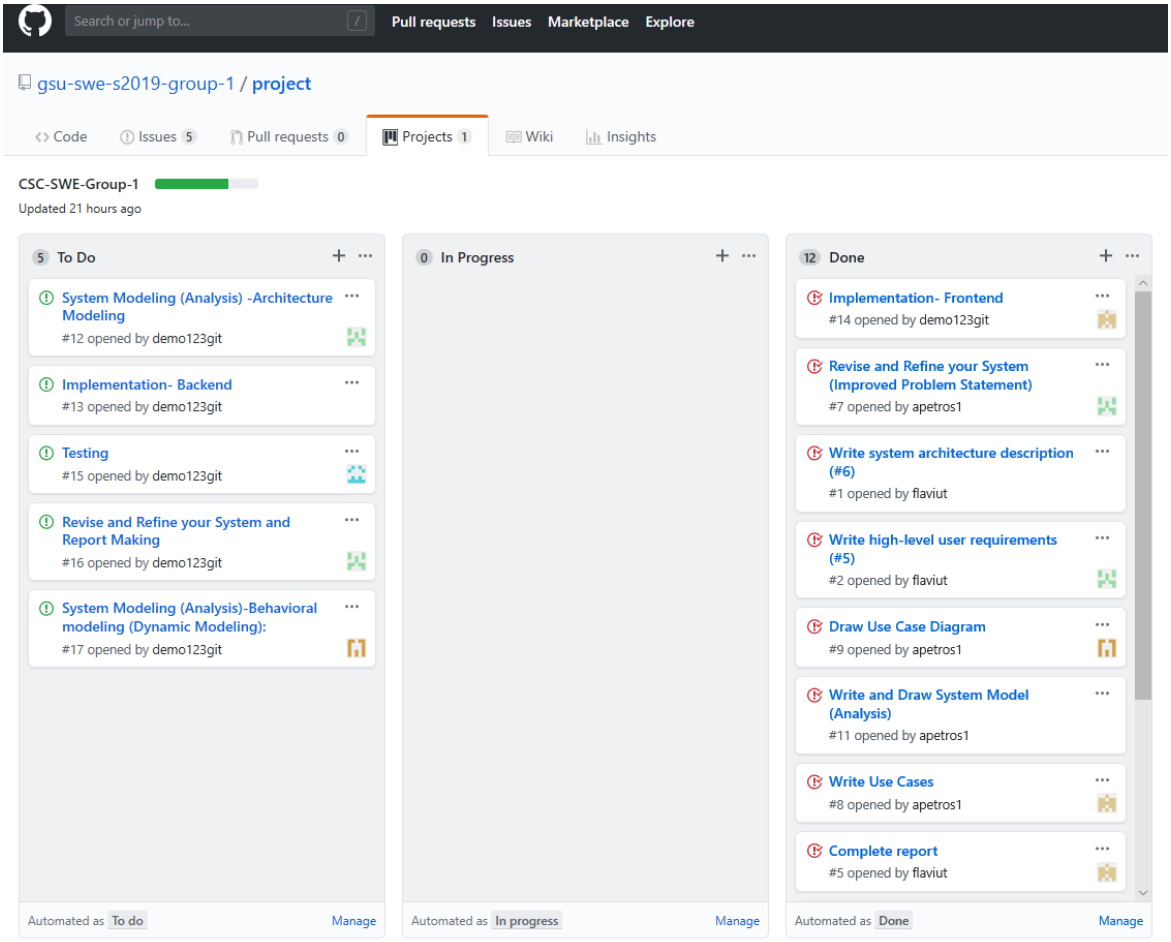

Figure 1: Screenshot of our group's Kanban board

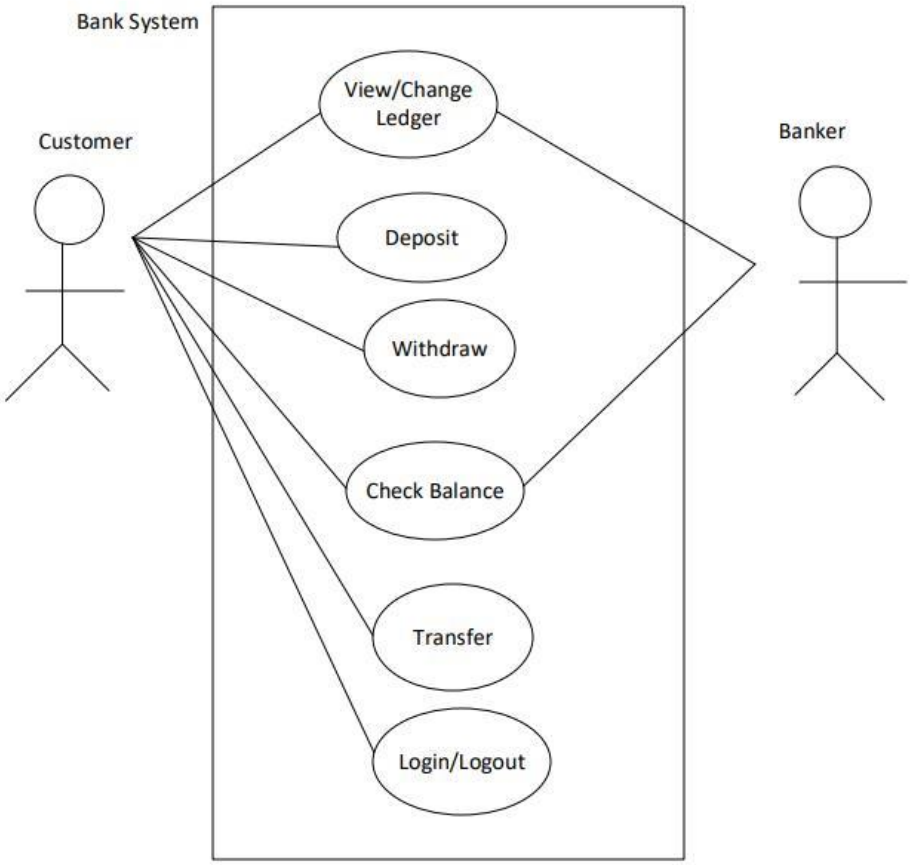

Figure 2: Use case diagram

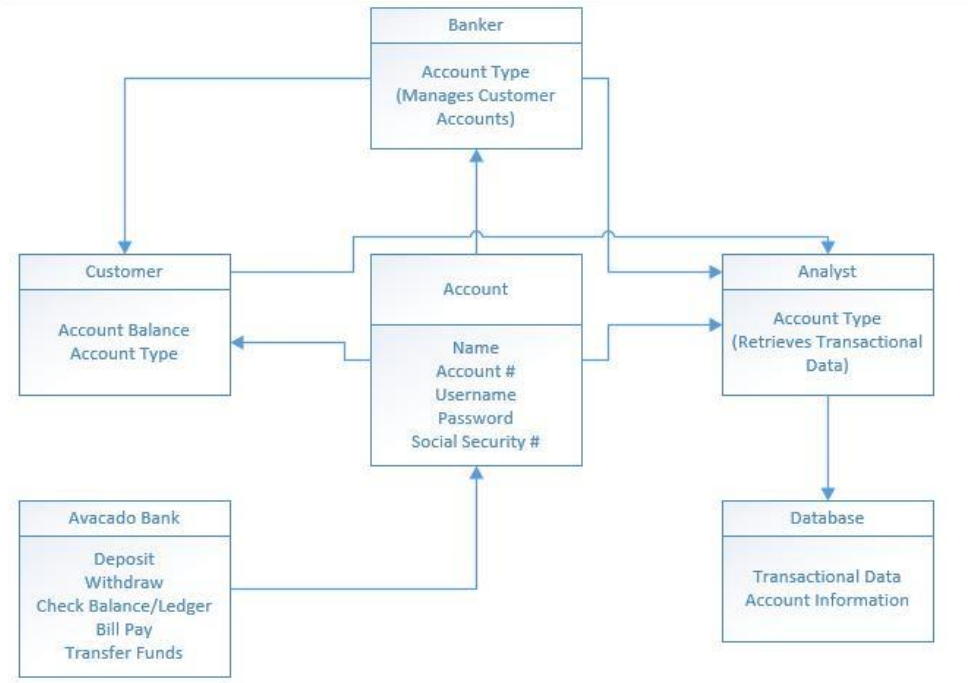

Figure 3: System class diagram

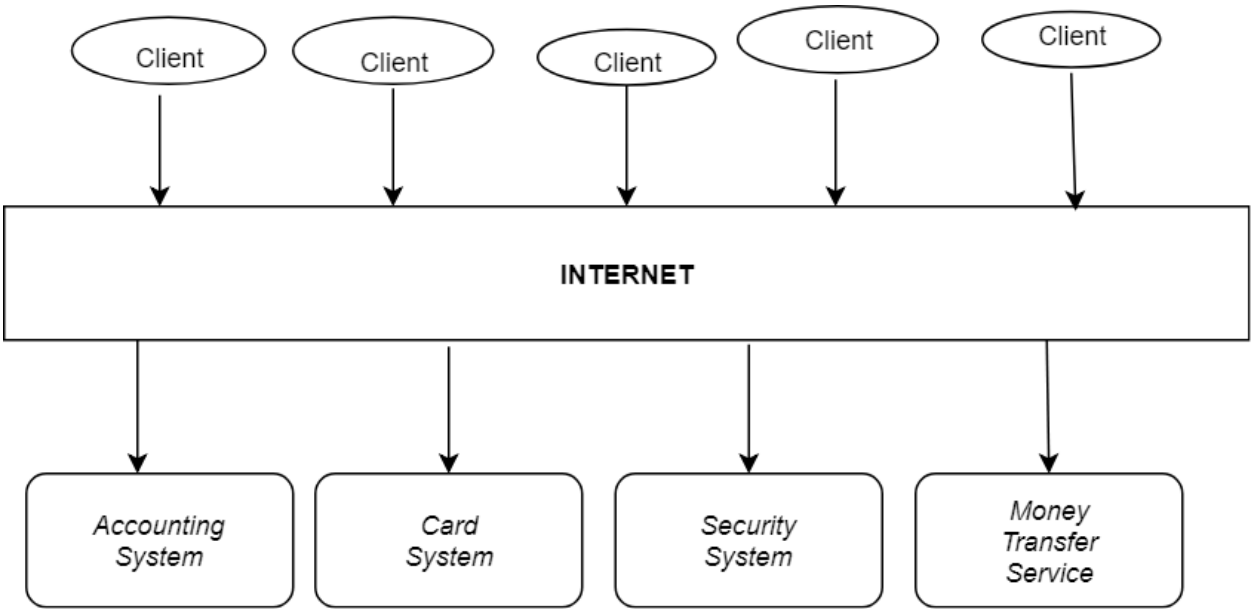

Figure 4: Architecture Modeling diagram (Client Server Pattern)

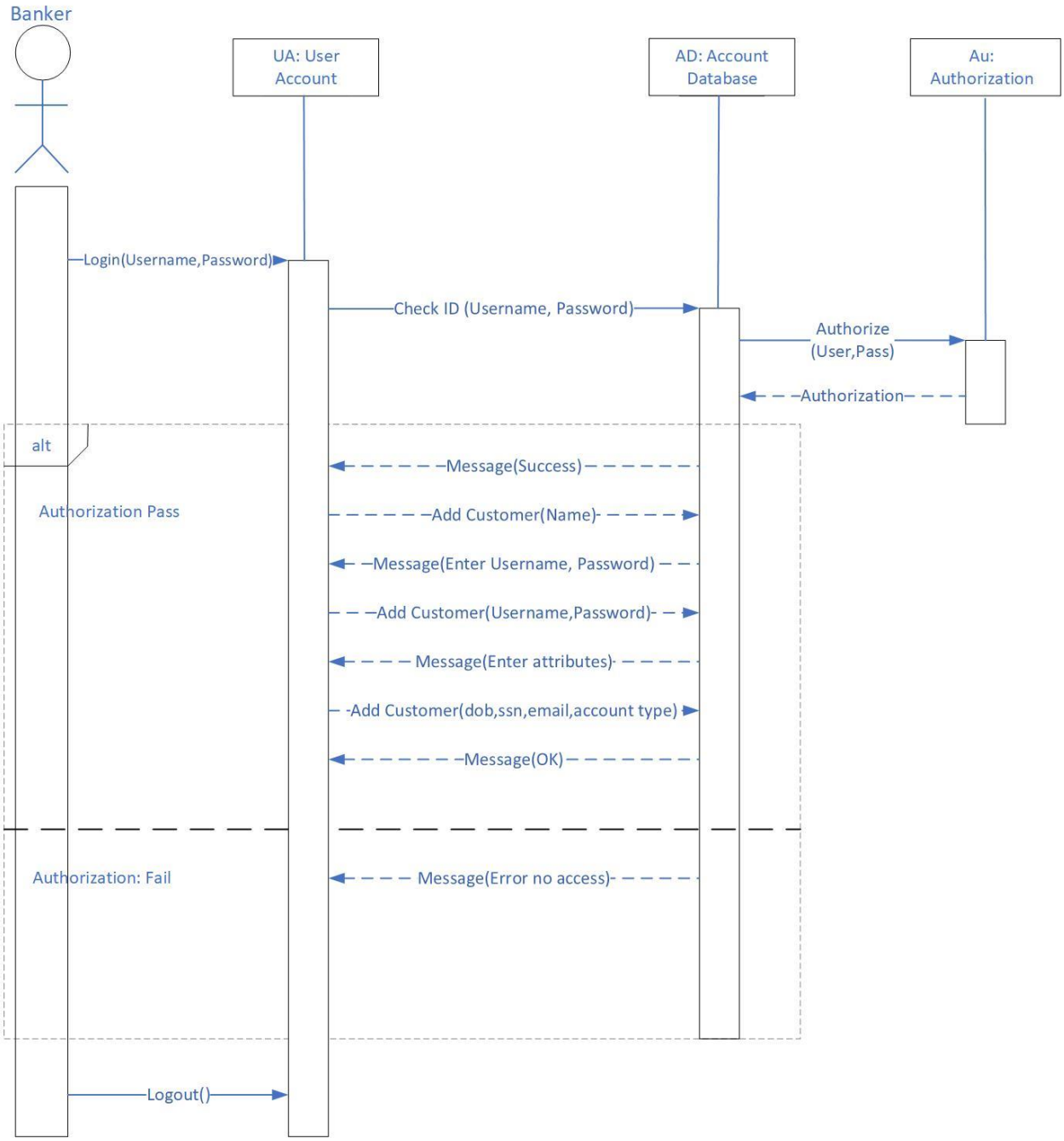

Figure 5: Behavioral Modeling diagram (UML sequence diagram)