Group 6 - Final Group Report

CSE545 Software Security

Team

Saikat Datta (Group Leader)

Miguel Bouza (Deputy Leader)

Anil Kumar

Tanner Cooper

Priyansh Hanish Mandan

Table of Contents

1 Introduction

The aim of this project was to create a skeleton Secure Banking System that required us to implement the security features learned as a part of the coursework. Implementation of the security features is a key aspect to the development of the project hence, the project has limited functionality, where the focus is on making the system as secure as possible. This final group report gives a high level overview of the Banking System developed. This document also provides detailed information about the responsibilities and tasks allocated to each group member. Security and functional vulnerabilities reported by all other groups have also been documented and justification for each valid and invalid vulnerability has been provided.

2 Responsibilities and Tasks

Name and Role	Responsibilities
Saikat Datta (Group Leader)	• Create Work Breakdown Structure (WBS) and assign deadlines on tasks needed for each phase of the project • Modify and approve weekly progress reports • Design and Build front-end for login module and customer dashboard. • Implemented Javascript to integrate front-end with server • Implemented front-end security • Tested application security features.
Miguel Bouza (Deputy Leader)	• Create detailed weekly progress reports • Organize, structure, and assign weekly meetings on Zoom with additional meetings as needed • Create template HTML pages for Tier 1 and Tier 2 employee • Create Amazon Lex bot instance and add customer intents to handle customer queries • Add and integrate Amazon Lex's UI using CloudFormation to our website
Anil Kumar	• Designed and implemented MySql database • Created backend from scratch using python flask • Created REST API's to deliver data to front-end • Tested backend integration.
Tanner Cooper	• Create HTML pages for Administrator • Update and add more functionality to Tier 1, Tier 2, and Administrator HTML pages • Research Hyperledger python libraries • Design OTP functionality using SMS • Create EB instance with AWS and handle all server deployment and version updates • Find and report functionality bugs • Fix many reported bugs
Priyansh Mandan	• Create design documentation • Create system overview and architecture

	• Create and manage MYSQL Database • Develop secure modules • Develop employee tasks • Implement Data masking • Testing UI security • Acquire domain name and apply SSL certificate to website

3 Vulnerability Report

Summary

32 unique Vulnerabilities (11 Valid, 21 Invalid)

6 Security (2 Valid, 4 Invalid)

26 Functional (9 Valid, 17 Invalid)

Security Vulnerabilities

Below is a table containing security vulnerabilities reported by other students testing our project. A description, validity (valid or invalid), and a response will be provided.

For Valid vulnerabilities: A solution will be provided

For Invalid vulnerabilities: A justification will be provided

Num	Description	Validity	Justification or Solution
1	SSN is not masked or hashed in backend	VALID	Masking was done on the frontend, however, the server still responds with full SSN. Masking will instead be done on the backend.
2	Authorization token not checked which allows for a curl request to create administrator accounts.	VALID	Although the scenario is very unlikely as an external user has to hack admin first to learn the request format, the underlying issue is still a valid one. We can mitigate this by adding a missing authorization token check to ensure that the function is being called from a logged-in user of proper authority over the function.
3	Passwords for the provided tester accounts are identical and weak.	INVALID	These were made knowing they are not "secure". They were chosen to make testing simple. Passwords can be changed or a new account can be registered to get secure passwords.
4	Client debugger left enabled	INVALID	This was done purposely as part of the code obfuscation for the frontend javascript. The debugger automatically starts and pauses the website to make using the console less convenient for attackers. It is a small deterrent. It doesn't affect any functionality or security.

5	System logs contain exceptions and unhandled errors.	INVALID	This was a design decision to show all errors to the system administrator. Part of their role is “to ensure smooth functioning of the banking system”. Thus, they should have access to all errors such that they can be fixed.
6	Changing password gives data breach error message on chrome	INVALID	Nothing from our site. Depends on the username/password combination being used by the user that is breached somewhere else and is being flagged by chrome.

Functional Vulnerabilities

Below is a table containing functional vulnerabilities reported by other students testing our project. A description, validity (valid or invalid), and a response will be provided.

For Valid vulnerabilities: A solution will be provided

For Invalid vulnerabilities: A justification will be provided

Num	Description	Validity	Justification or Solution
1	Deposit cashier check not working.	VALID	Server implementation error. Fixed by adding the missing check.
2	No validation on data types. Customers and employees can input any values to update info, create/modify employees and change password pages	VALID	Validation here was only done for the register page. The same functionality will be applied to update info pages. Server checks will also be added.
3	Unable to login into accounts. “Invalid Credentials” error is displayed and/or Error 500.	VALID	Backend was trying to fetch a python object outside try/catch statement, that was not allocated any memory. We will adopt microservices architecture using kubernetes where our web app will be hosted over multiple pods for load balancing so even if the app crashes over one pod others will be available to handle incoming requests.
4	Allowing to schedule appointments at the same date and time. Past Appointments also allowed through curl.	VALID	Server checks will be added to prevent duplicate/past/invalid appointment times.
5	Delete employee not working	VALID	Employee id check was missing on the server. Fixed by adding the check.

6	Open accounts without approval	VALID	Approval from a Tier 1 bank employee will be required to open a new account.
7	No approval from tier 2 when the sum of multiple transactions in a day exceeds the critical limit.	VALID	Additional information for the daily transaction limit will be added to the database and checked before each transaction to correctly process the approval order.
8	Decimal money transfer not possible.	VALID	Frontend blocks decimal inputs which can be easily rectified to allow this.
9	Can deposit funds to accounts that don't exist	VALID	Backend checks will be added to prevent this.
10	The "I am" validation is not working for employee logins.	INVALID	This was only done for specific server side reasons. The multiple selections for employees is purely a visual choice. It does NOT affect the tier of the logged in account. It only matters that is is a customer vs. (any) employee type
11	Unable to login into the testCustomer account. I created a testCustomer account but was unable to login into it.	INVALID	Error could not be replicated. New users registered through the register page or employee page and could be logged into.
12	Duplicate cashier cheques being displayed for customers.	INVALID	This is only a visual error. The same cheque cannot be deposited twice.
13	Pending Transaction/Approval list and approve/decline not in one page. Have to remember the request id every time. Increasing the no.of clicks for the user.	INVALID	This does not affect any functionality. This only relates to usability.
14	Invalid credentials message when choosing the wrong user type is ambiguous.	INVALID	The user role dropdown is an user input credential for logging in and hence the wrong matching is in fact an invalid input.
15	Undefined showing on appointments list if none	INVALID	When at least one appointment has been scheduled, undefined entries disappear and only the correct appointments are displayed. Does not affect any functionality or security. Minor visual error.
16	Same account transaction approve/decline not working	INVALID	Same account transfer is not possible. Having no error message is a small visual error.

17	As you start to ask the bot questions there is a vertical three dot icon that allows you to refresh the bot. However, it just appends that message to the chat conversation rather than refreshing the bot chat	INVALID	Minor visual error. The required chatbot functionality is still present and properly responds to relevant questions about the banking system.
18	Change tier 1 info and approve them from admin. Refreshing tier 1 page then shows as logged out.	INVALID	Browser tabs share the same session like (most) websites. Provided screenshots show that tier 1, and 3 dashboards were open. Logging into tier 3 page, the browser session is updated to be for tier 3, and the tier 1 account is logged out.
19	Same account can be opened in multiple tabs, even after closing all before logging out.	INVALID	Browser tabs share the same session like (most) websites. Unless the session has timed out or the user logs out, the user can continually close and return to the page.
20	Update contact number not reflecting within 5 seconds automatically.	INVALID	Not a requirement to update automatically within 5 seconds. Users may have to wait longer and MUST refresh the browser page to see the changes.
21	OTP implemented in only 1 functionality	INVALID	OTP is implemented in 3 places - transfer fund, accept fund request, and reset password.
22	Delete user closes all accounts but does not allow the userid to be used again	INVALID	Design decision to ensure that all customer data is archived in the database. Account is disabled and cannot be logged into or used again.
23	Browser performance issue when navigating between pages	INVALID	User has their devtools/console open in the screenshot and it is intended as we are forcing the debugger to stop page activity when using the console/devtools for security purposes. Minor deterrent.
24	Update information request placed without any change	INVALID	Does not affect any functionality. If approved, no info is changed.
25	Empty search employee gives undefined output	INVALID	Does not affect any functionality. Searching for a defined employee provides proper output.
26	For logging into internal employee's accounts, the response time is more than a minute	INVALID	Could not replicate as tested in different networks, different machines. All employee accounts were accessible within seconds.

Final Design Document

1. Introduction

The aim of this project is to create a skeleton Secure Banking System that would require us to implement the security features learned as a part of the coursework. Implementation of the security features is a key aspect to the development of the project hence, the project has limited functionality, where the focus is on making the system as secure as possible. This design document gives a high level overview of the Banking System to be developed. All UML diagrams, the system architecture and data description give a basic understanding of the working of the system.

1.1 Purpose

The purpose of this document is to provide the developers with a basic understanding of the different components and entities that will be used throughout the development phase. The use case diagram highlights the important tasks for each user, while the class diagram gives a high level understanding and a visualisation into the object oriented system. The attributes, their operations and the relationship with other classes are clearly denoted. The document also provides details of the database and the MYSQL tables.

1.2 Scope

The scope of this document is to provide a high level understanding of the working of different components involved in the design. This document would be used as a guide throughout to help the developer create a secure banking system. The implementation of security features has not been covered in its entirety and would be added as the development proceeds. The document is subject to change in accordance with the feedback received.

1.3 Definitions and Acronyms

Acronym	Definition
API	Application Programming Interface. Set of programs that enable the developer to implement a specific functionality.
HTML/CSS	Markup Language used for front-end. It is extensively used to design the front-end system for user interaction.
JS	JavaScript and Python would be used for back-end design.
MYSQL	Open-Source relational database management. It is based on SQL and is used to create databases and manage them(create new, add data, delete data).

2. System Overview

2.1 Design and Functionality

The goal of this project is to create a basic Secure Banking System that ensures the working of basic banking features in a very secure manner. The key security features that would be implemented are as follows :

1. Public-Key Certificates
    a. Digital Authentication and Data Integrity are two important security features that are kept in mind while implementing the Public Key Certificates. Digital Authentication ensures that the right customer would access the system while data integrity ensures the database and transactions are consistent throughout the system to avoid fraud.
2. OTP
    a. OTP security feature is added to provide a 2-Factor authentication to confirm the user identity. In this project, the OTP can be sent via an email or via a text message to verify the customer and the employee trying to sign-in.
3. Multi-User sign-in
    a. Multi-User sign in ensures that multiple customers and employees can access the system at the same time. It is a part of session management. To limit the number of users, a session token is issued which when expired logs out the customer from all sessions.
4. Login Control
    a. Login control and session management is controlled through Python Flask library. Once the user logs-in, a session token is issued. If the user is inactive for a certain period of time, the session expires and the user is logged out automatically.
5. Data Masking and Hashing Algorithms
    a. Hashing algorithms play a key role in masking the sensitive user data stored in the database. Computationally efficient hashing algorithms can make the database very secure.
6. System Logs
    a. System logs are stored to identify any integrity issues or inconsistencies in the transaction.
7. Hyperledger for transactions.
    a. Used for approved transactions as Hyperledger provides a very high degree of confidentiality. The main advantage of using Hyperledger for transactions is the flexibility it provides over consensus and access which makes it more private.

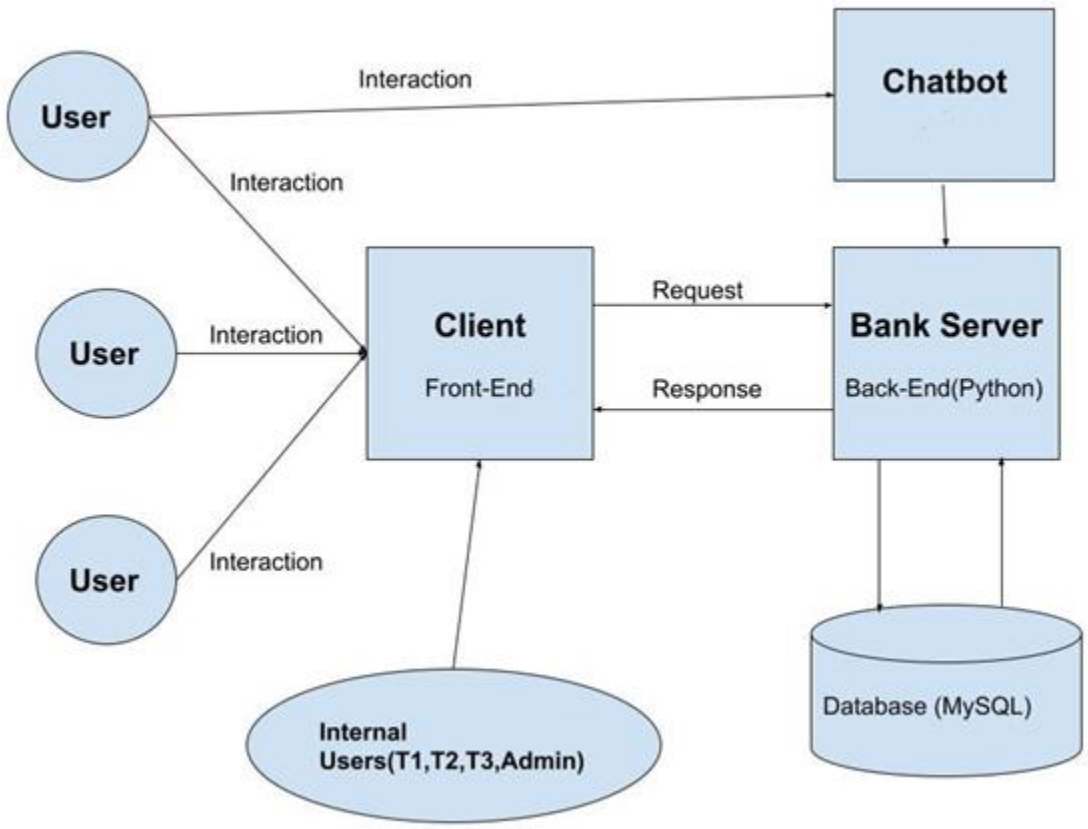
2.1.2 Users and their roles

- Internal Users
    - Tier 1 Employees - The primary role of a Tier 1 Employee is to assist customers in initiating bank transfers, adding funds to account, issue cashier’s checks and approve transactions and customer requests.
    - Tier 2 Employees - The primary role of a Tier 2 Employee is to manage a customer’s account. It involves opening, closing or modification of accounts. Another important role is to approve critical transactions which the tier 1 employee is ineligible to approve.

    - Tier 3 Employees - The tier 3 employee is the administrator whose primary role is to manage the employees' account. System logs can only be accessed by the administrators.
- External User
    - Individual Users - The primary role of an individual user is to initiate fund transfer (either debit or credit money in the bank account). The user can approve or reject a money transfer request and also request an additional account with the bank.

2.2 System Architecture

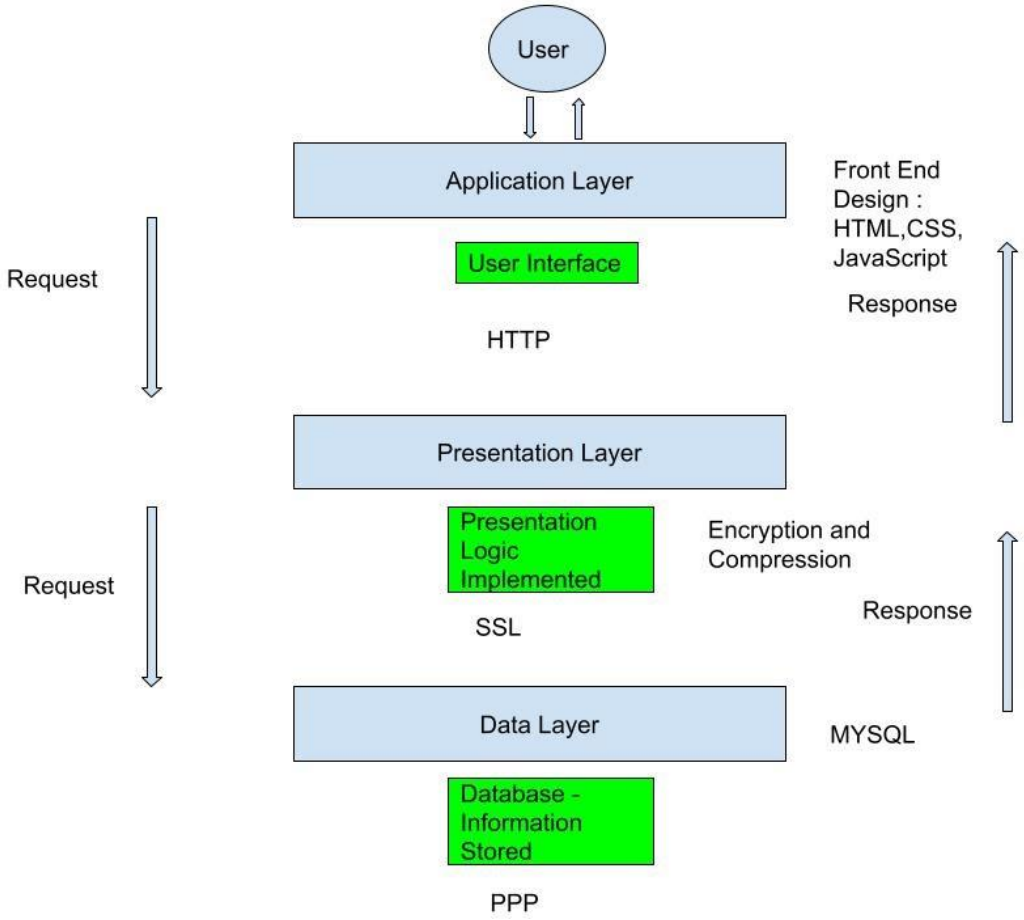
2.2.1 Architectural Design



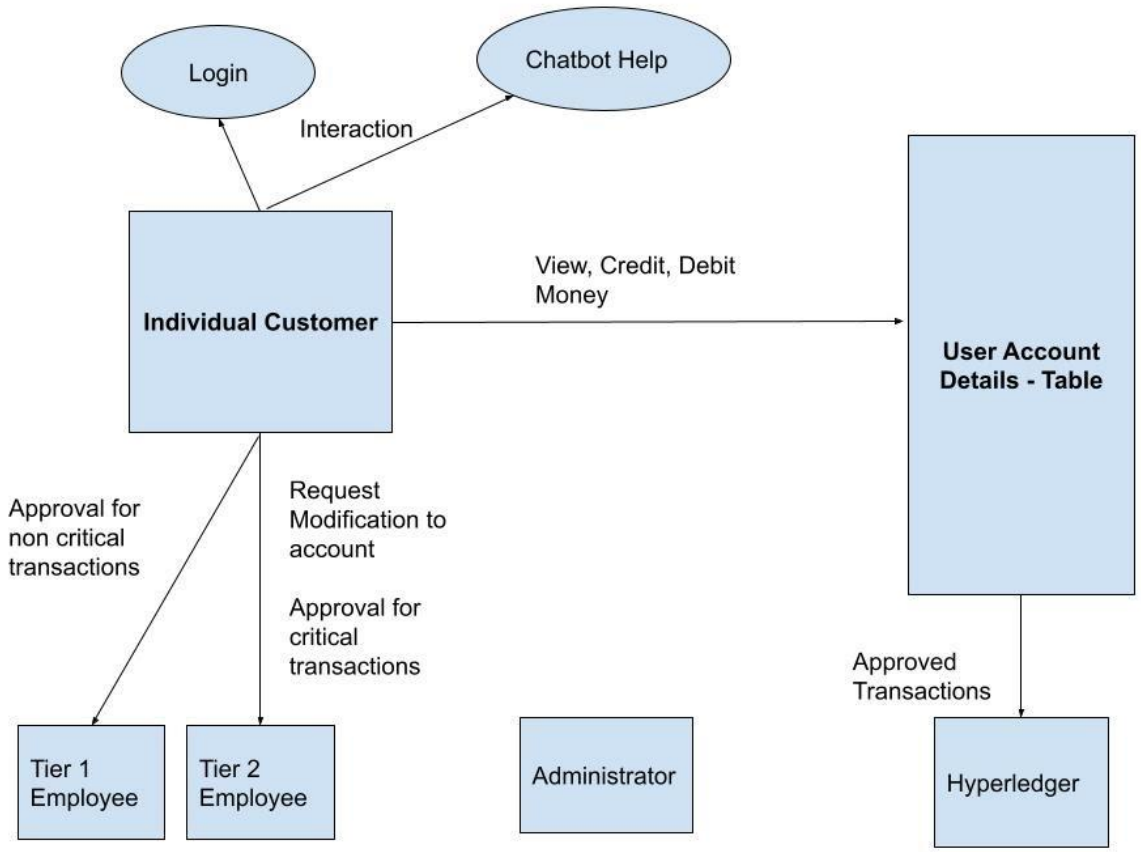
As far as architectural design is concerned, all major components of the Secure Banking System are clearly visible and the working of these components with respect to each other can be visualised. We have multiple users simultaneously accessing the system through the Front-End banking client.

2.2.2 Decomposition Description

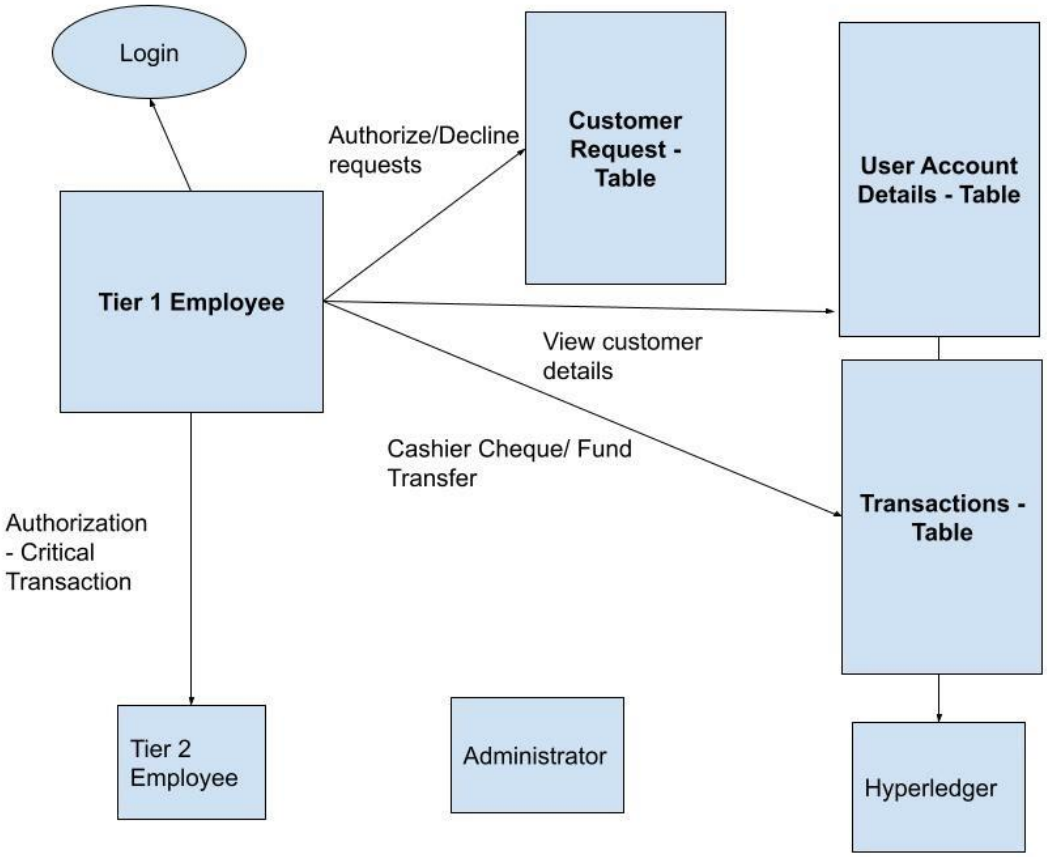
Various components of the projects are decomposed into separate entities to showcase the working. The higher level modules are broken down into simpler sub-modules outlining the purpose and type to the developer.



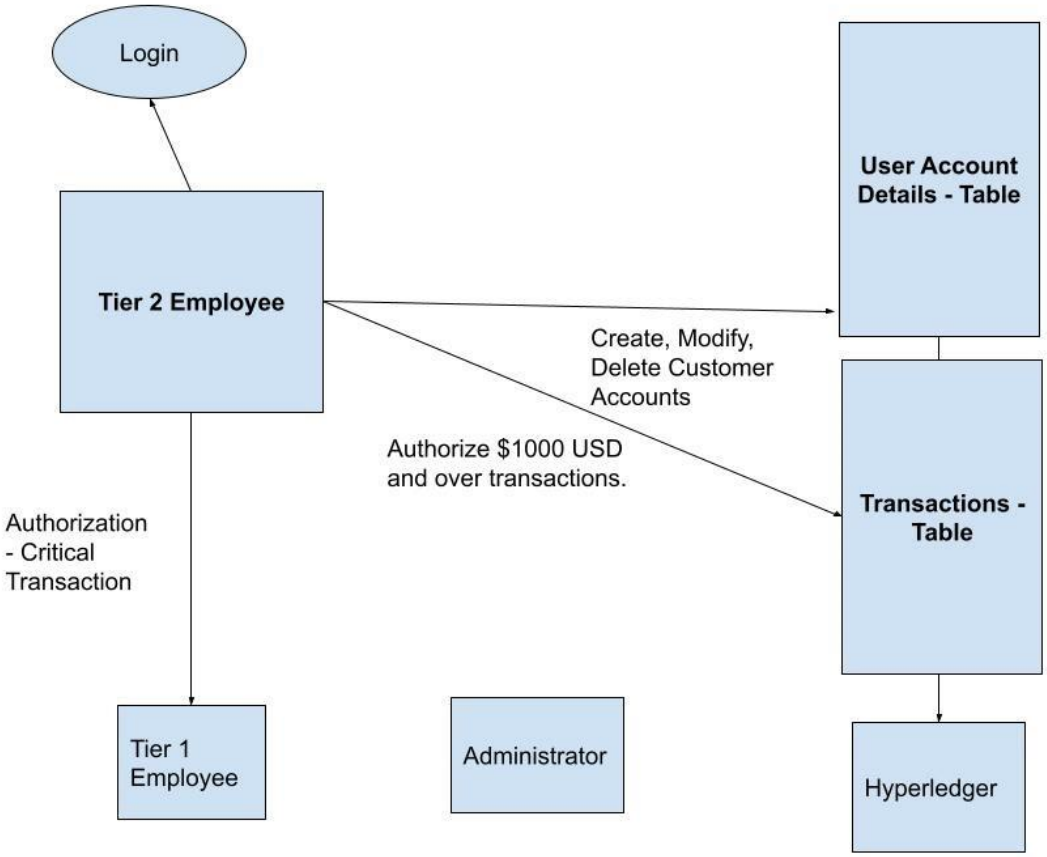
User: Covers all banking system users including tier-1/2 and admin of banking system.



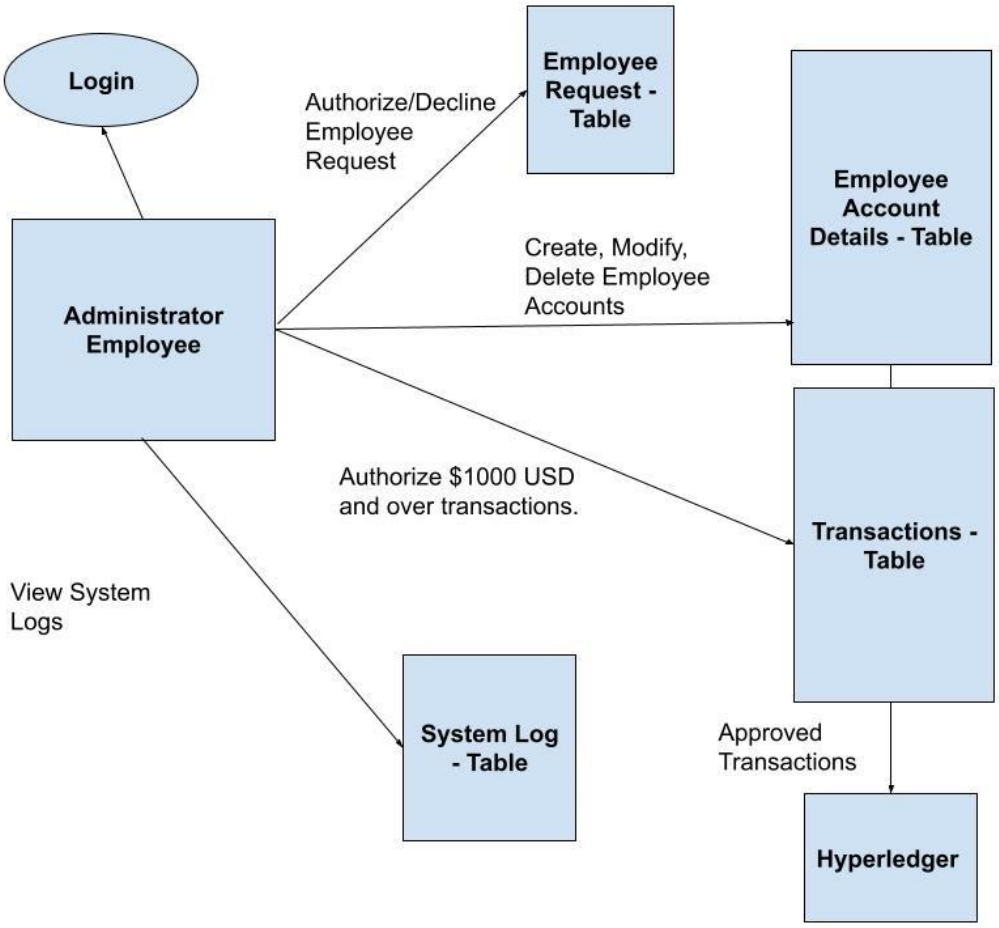
Tier-1 employee



Tier-2 employee



Administrator



3. Data Design

3.1 Data Description

Data Description gives an overview of the MYSQL tables and provides insights into the overall database design.

Based on the project requirement document, the following 5 tables will be used.

Table 1 : Sign-In System Logs(Customer)

This table logs in the details of sign-ins made by the individual customer to maintain a secure session for each customer. The table would only be accessible to the administrator. This table would be used extensively for session management, allowing multiple users to sign-in at the same time while storing the data for each user.

SIGN-IN LOG(Customer)

Entity	Description
customer_username	The customer user name would be required as a unique identifier to the record.
sign_in_time	This timestamp denotes the entry-point time for the customer in the system.
session_status	This denotes an “active” or “dormant” status for the customer activity. It will be used to create a secure session.
sign_out_time	This timestamp denotes the exit-point time for the customer in the system.

Data Dictionary - Table 1

Entity	Type	Length	Null Allowed	Unique
customer_username	String	50	No	Yes
sign_in_time	Timestamp		No	
session_status	Binary		No	
sign_out_time	Timestamp		No	

Table 2 : Sign-In System Logs(Employee)

This table logs in the details of sign-ins made by the employees to maintain a secure session for each employee. The table would only be accessible to the administrator. This table would be used

extensively for session management, allowing multiple employees to sign-in at the same time while storing the data for each employee.

SIGN-IN LOGS(Employee)

Entity	Description
employee_username	The employee user name would be required as a unique identifier to the record.
sign_in_time	This timestamp denotes the entry-point time for the customer in the system.
session_status	This denotes an “active” or “dormant” status for the customer activity. It will be used to create a secure session.
sign_out_time	This timestamp denotes the exit-point time for the customer in the system.

Data Dictionary - Table 2

Entity	Type	Length	Null Allowed	Unique
employee_username	String	50	No	Yes
sign_in_time	Timestamp		No	
session_status	Binary		No	
sign_out_time	Timestamp		No	

Table 3 : User_Account_Details(Customer)

This table stores the details of the customer and is accessible to Tier 1 and Tier 2 employees only along with the customer. It contains basic information regarding the customer and will be used extensively when transactions are made. The table would reflect all the changes once the transaction is approved.

User Account Details(Customer)

Entity	Description
customer_username	The customer user name would be required as a unique identifier to the record.
customer_account_number*	The account number is a unique identifier for the customer with respect to the bank.
customer_email	The email-id of the customer is stored to create a point of contact.
account_type_savings	Denotes the savings account type for each customer. It can or cannot be present for all customers.
account_type_checking	Denotes the checking account type for each customer. It can or cannot be present for all customers.
account_balance_savings*	Denotes the account balance for the savings account type.
account_balance_checking*	Denotes the account balance for the savings account type.
account_status	Denotes if the account is active or dormant.
customer_credit_card_number*	Stores the credit card number for each customer.
customer_credit_limit	Stores the credit limit for each customer.
customer_SSN*	Stores the Social Security Number for each customer.

Data Dictionary - Table 3

Entity	Type	Length	Null Allowed	Unique
customer_username	String	50	No	Yes
customer_account_number*	String	15	No	Yes
customer_email	String	50	No	Yes
account_type_savings	Binary		Yes	
account_type_checking	Binary		Yes	
account_balance_savings*	Double		Yes	
account_balance_checking*	Double		Yes	
account_status	Binary		No	
customer_credit_card_number*	String	16	Yes	Yes
customer_credit_limit	Double		Yes	
customer_SSN*	Integer	9	Yes	Yes

Table 4 : Employee_Account_Details(Employee)

This table stores the details of the employees (Tier 1, Tier 2 and Tier 3). It contains basic information regarding the employees and will be used extensively when transactions would be made. The table would also be used to grant special access and permissions to the employees based on the tier.

User Account Details(Employee)

Entity	Description
employee_username	The employee user name would be required as a unique identifier to the record.
employee_account_number*	The account number is a unique identifier for the employee with respect to the bank.
employee_email	The email-id of the customer is stored to create a point of contact.
employee_tier	Denotes the tier of the employee, based on which the employee would have access to customer accounts and transactions.
employee_id*	Unique identifier for each employee
employee_SSN*	Stores the SSN for employees.

Data Dictionary - Table 4

Entity	Type	Length	Null Allowed	Unique
employee_username	String	50	No	Yes
employee_account_number*	String	15	No	Yes
employee_email	String	50	No	Yes
employee_tier	Integer		No	
employee_id*	Integer		Yes	Yes
employee_SSN*	Integer	9	No	Yes

Table 5 : Transaction_Details

This table gives the details of all the transactions involved and keeps a log of it. All types of transactions including Fund transfer, Debit and Credit funds are included in the table. This table would be used to approve/decline transactions and also update the customer account details table. The bank statement used by the customer would also make use of this table.

Transaction Details(Employee)

Entity	Description
transaction_id	Unique identifier to each transaction.
transaction_amount	Denotes the total amount for the transaction.
transaction_type	Denotes the type of transaction (Fund transfer, Cheque, Credit, Debit)
to_account	Account number for the receiver.
from_account	Account number for the sender.
transaction_time	Timestamp for the transaction.
card_number*	If a card is involved, the card number is stored.

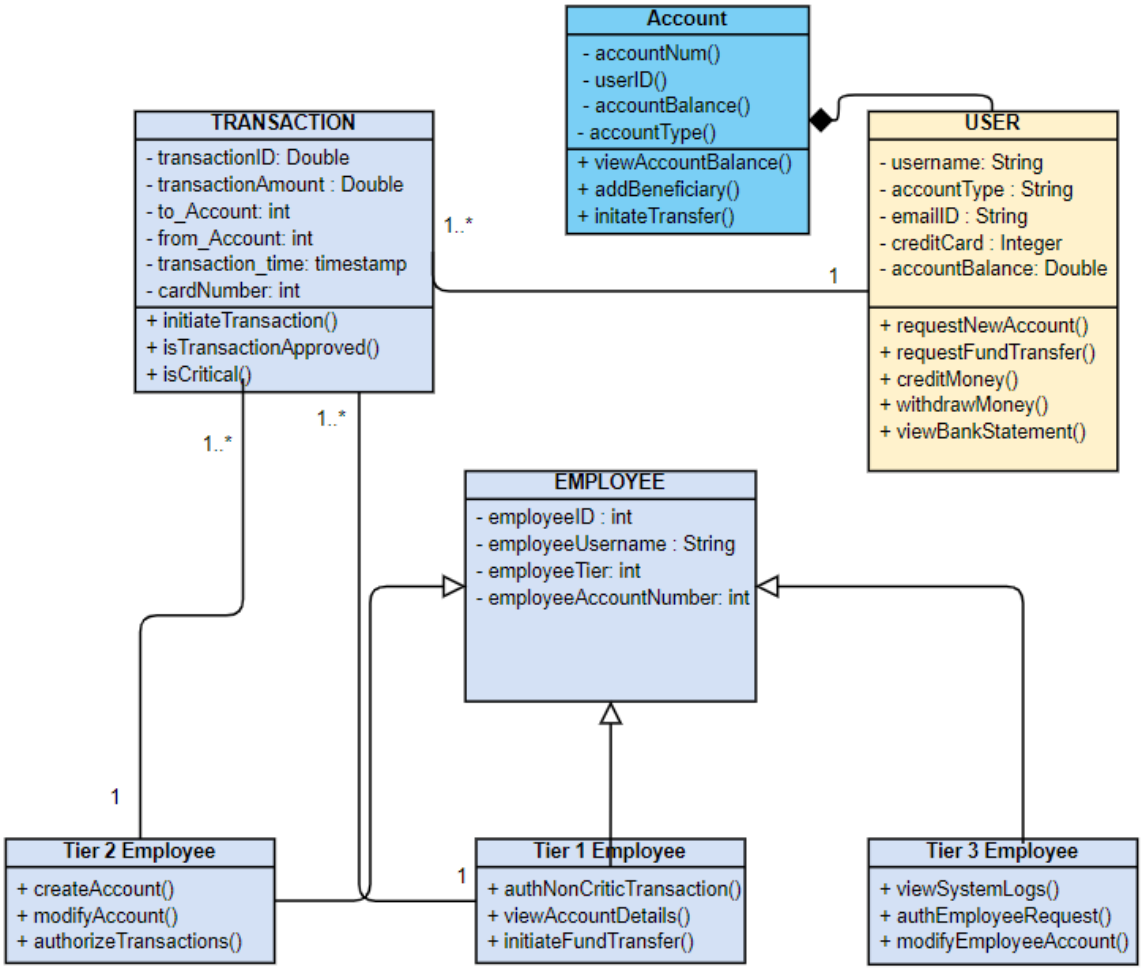
Data Dictionary - Table 5

Entity	Type	Length	Null Allowed	Unique
transaction_id	String	50	No	Yes
transaction_amount	Double		No	No
transaction_type	String	10	No	No
to_account	Integer		No	No
from_account	Integer		No	No
transaction_time	Timestamp		No	No
card_number*	Integer		Yes	Yes

4. Component Design

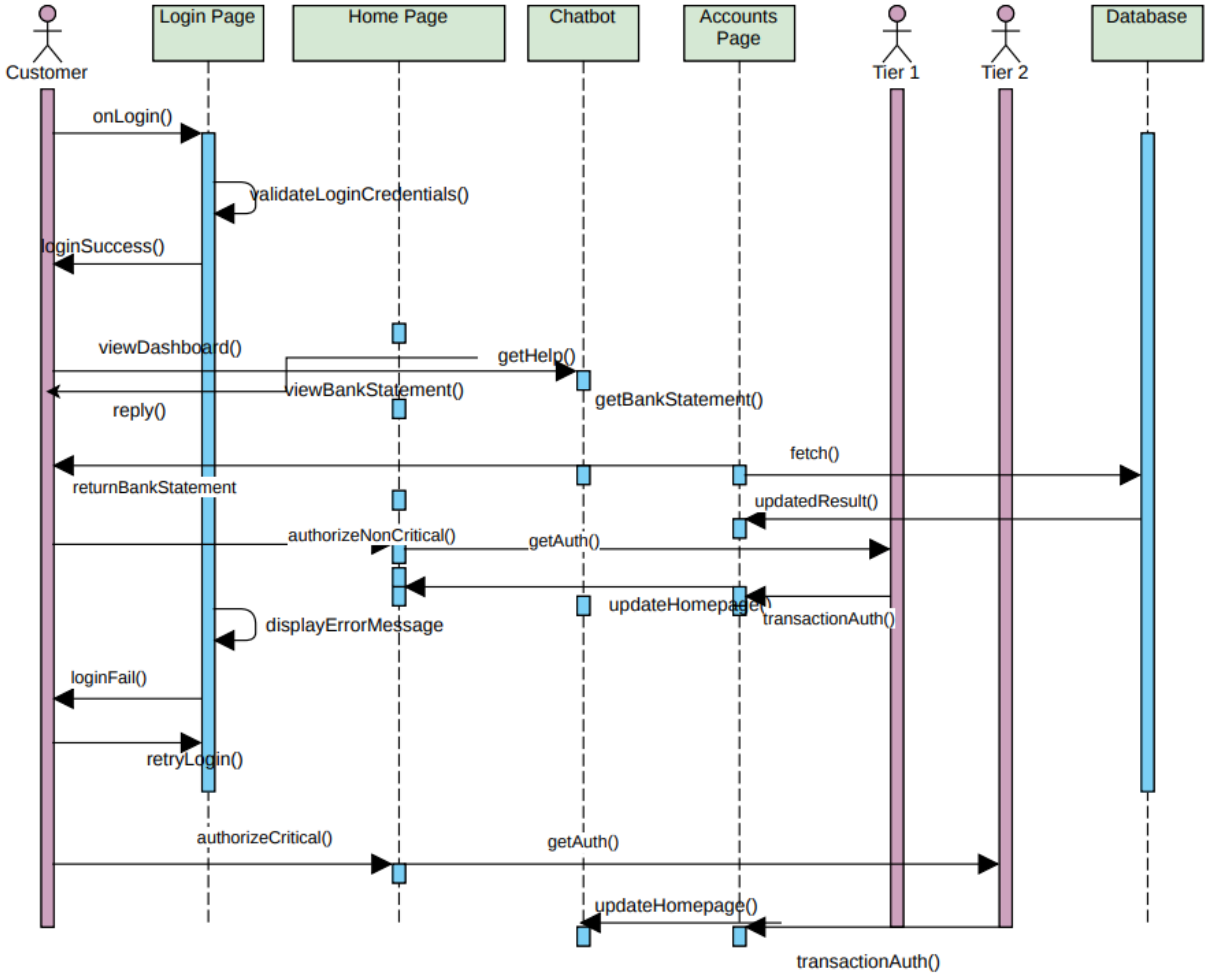
4.1 Class diagram:

This class diagram depicts the classes to be used



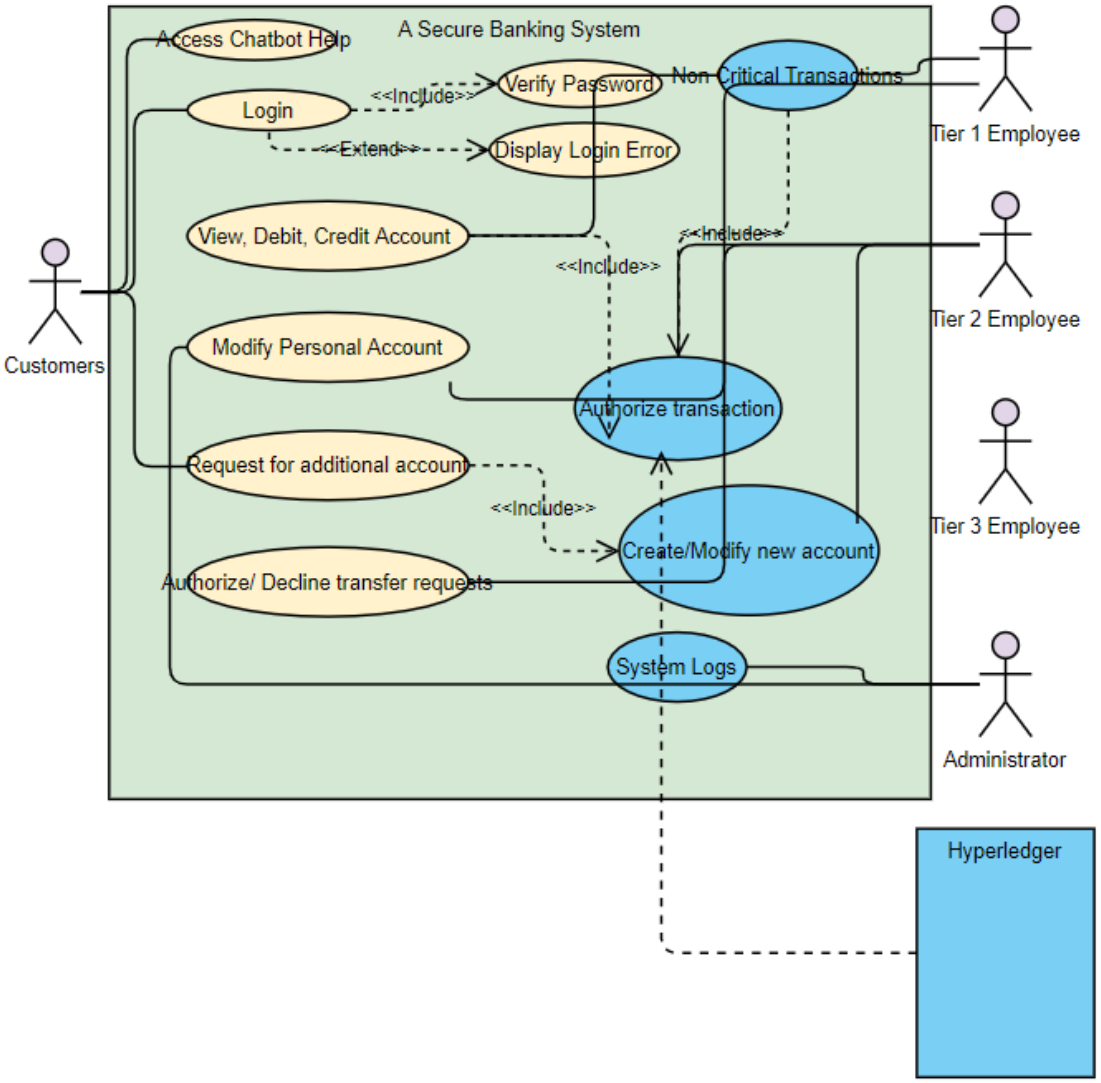
4.2 Sequence diagram:

These are detailed interaction diagrams that describe how operations in the system are carried out. They capture the interaction between different objects in the context of functional collaborations. These diagrams are time focused and show order of interactions visually by using vertical-axis.



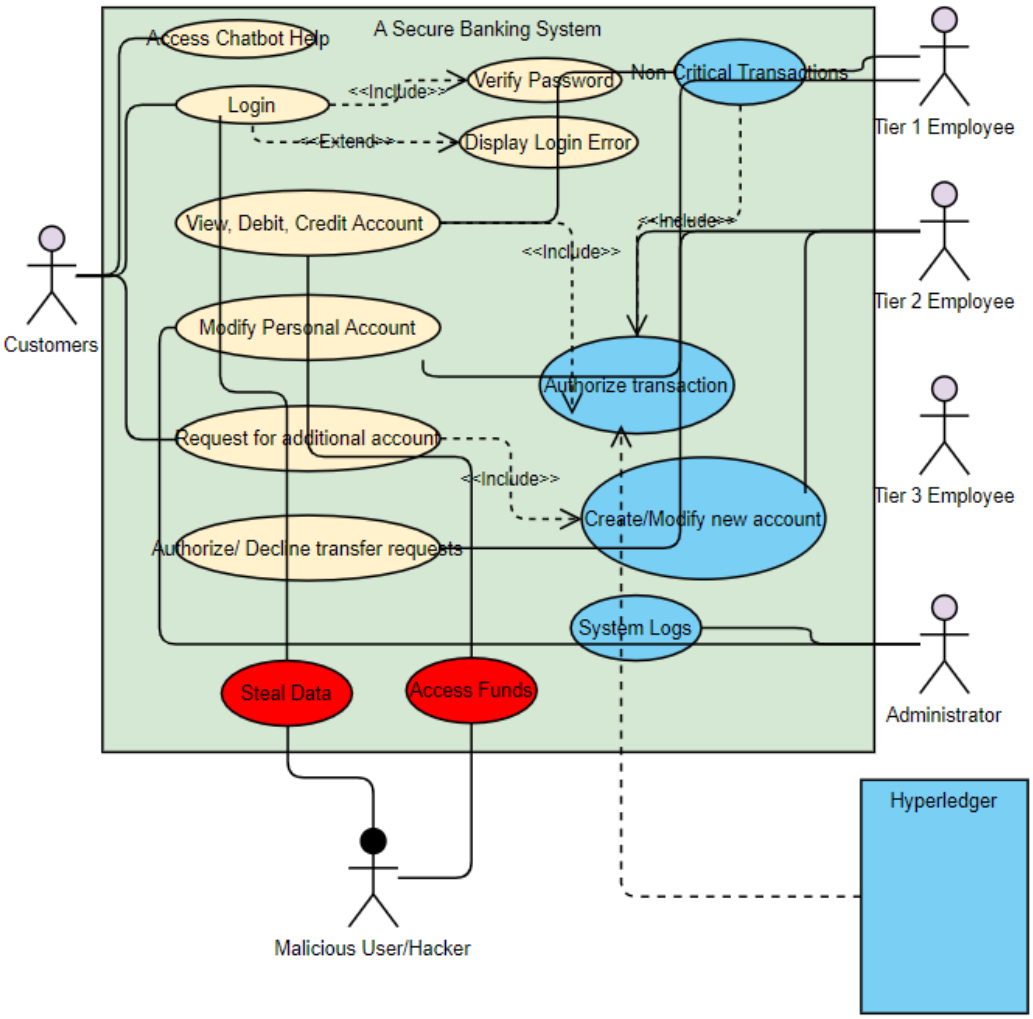
4.3 Use case diagram:

A use case diagram demonstrates the different ways that a user might interact with the system. It summarizes different types of users (also known as actors) our system has and their interactions with the system.



4.4 Misuse case diagram:

A misuse case diagram elaborates features that the system doesn't allow. They play an important role in determining non-functional requirements, like security requirements. A misuse case is always linked with a misactor.



5. Requirements Traceability Matrix

Requirements Traceability Matrix

Project Name		A Secure Banking System	Business Area		Software Security			
			Course		CSE 545			
BR#	Category/Functional Activity	Requirement Description	Priority	Test Case Reference	User Acceptance Validation	Comments		
BR#1	Login Activity	Customer Login	High	TestCase1	Accepted/Failed			
		1. Customer Login Check - Username available	High	TestCase2	Accepted/Failed			
		2. Test empty username	High	TestCase3	Accepted/Failed			
		3. Test empty password	High	TestCase4	Accepted/Failed			
		4. Test valid username and wrong password	High	TestCase5	Accepted/Failed			
		5. Test valid password and wrong username	High	TestCase6	Accepted/Failed			
		6. Test invalid both.	High	TestCase7	Accepted/Failed			
		7. Verify tasks for Employees.	High	TestCase8	Accepted/Failed			
		8. Check for logout after no activity for 5 mins.	High	TestCase9	Accepted/Failed			
		9. Check 2 Factor Authentication	High	TestCase10	Accepted/Failed			
		10. Test multiple login attemps. Block account after 3 failed tries.	High	TestCase11	Accepted/Failed			
		11. Test forgot password. Send temporary password.	Medium	TestCase12	Accepted/Failed			
BR#2	Transaction	1. Test if all approved transactions stored in hyperledger	High	TestCase13	Accepted/Failed			
		2. Test all credit and debit transactions to maintain integrity	High	TestCase14	Accepted/Failed			
		3. Test if funds credited are same as funds debited.	High	TestCase15	Accepted/Failed			
		4. Transaction ID must be consistent with amount.	High	TestCase16	Accepted/Failed			
BR#3	Customer Bank Statement	1. Test bank statement is consistent	High	TestCase17	Accepted/Failed			
		2. Test bank statement easily downloadable	High	TestCase18	Accepted/Failed			
BR#4	Chatbot	1. Test that chatbot works in all scenarios	Medium	TestCase19	Accepted/Failed			
		2. Test if chatbot is actually effective	Medium	TestCase20	Accepted/Failed			
		3. Verify chatbot replies	Medium	TestCase21	Accepted/Failed			
		4. Consistent results from chatbot.	Medium	TestCase22	Accepted/Failed			
BR #5	Hyperledger	1. All approved transactions must be present.	High	TestCase23	Accepted/Failed			
		2. Transaction ID must be consistent	High	TestCase24	Accepted/Failed			

BR#6	Customer	1. View and download bank statement	High	TestCase26	Accepted/Failed
		2. View account details. Test consistency.	High	TestCase27	Accepted/Failed
		3. Test fund transfer from account to account.	High	TestCase28	Accepted/Failed
		4. Test debit functionality.	High	TestCase29	Accepted/Failed
		5. Test credit functionality.	High	TestCase30	Accepted/Failed
		6. Test request for a new account.	High	TestCase31	Accepted/Failed
		7. Test modification for account.	High	TestCase32	Accepted/Failed
		8. Authorize request for money transfer.	High	TestCase33	Accepted/Failed
BR#7	Tier 1 Employee	1. Test customer request reply.	Medium	TestCase34	Accepted/Failed
		2. Test customer account modification.	Medium	TestCase35	Accepted/Failed
		3. Initiate fund transfer- Cashier cheque.	High	TestCase36	Accepted/Failed
		4. Test authorizing non-critical transaction	High	TestCase37	Accepted/Failed
BR#8	Tier 2 Employee	1. Test authorize transactions.	High	TestCase38	Accepted/Failed
		2. Test initiate modification of accounts.	High	TestCase39	Accepted/Failed
		3. Test modify customer account.	High	TestCase40	Accepted/Failed
BR #9	Tier 3 Employee	1. Test view system logs	High	TestCase41	Accepted/Failed
		2. Test authorize employees request	High	TestCase42	Accepted/Failed
		3. Test modify employee account.	High	TestCase43	Accepted/Failed
BR #10	Opening additional account	1. Test if previous account valid	High	TestCase44	Accepted/Failed
		2. Test account status	High	TestCase45	Accepted/Failed
		3. Test if previous account balance is positive	High	TestCase46	Accepted/Failed
		4. Intiaite new account. Add account details to customer ta	High	TestCase47	Accepted/Failed
BR#11	Performance Criteria	1. Test if multiple users log in at same time.	High	TestCase48	Accepted/Failed
		2. Test session management	High	TestCase49	Accepted/Failed
		3. Test if employees and users interact at same time.	High	TestCase50	Accepted/Failed