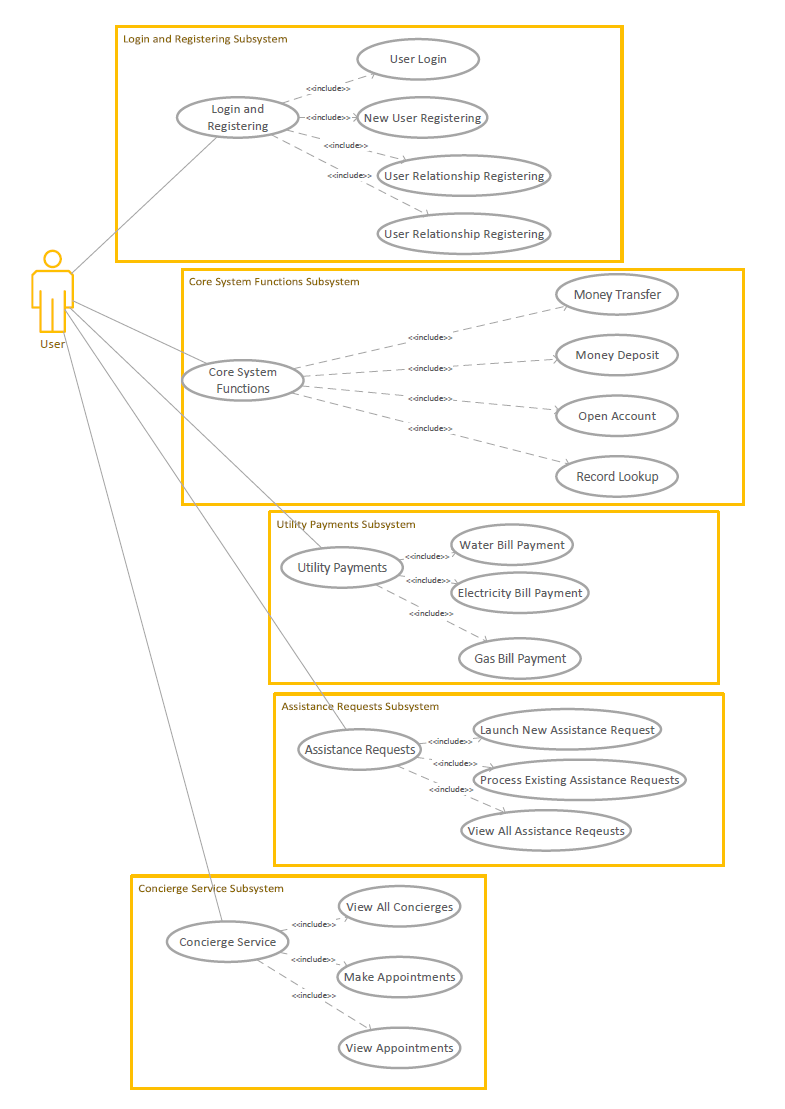
**Technical Documentations**

**I. Requirement Analysis and Illustration**

System Usecase Diagram



1. **Login and Registering**

In our online banking system, users can easily login and register their accounts on the first page.

1. **User Login**

Users input their identification number and password into the empty form, and then they get a new token stored in the localstorage which will expire in 15 days.

1. **New user registering**

Users input their identification number, password and check password into the empty form, and then they get a new token stored in the localstorage which will expire in 15 days.

1. **User relationship registering**

The son/daughter completes the relationship registering form, and his/her parent will get the information.

1. **User relationship confirming**

The parent receives the confirmation and decides whether to accept or not.

1. **Core System Functions**
2. **Money Transfer & Deposit**

Users make deposits, deposits and transfers through system simulation.

1. **Open Account**

Users can add a new bank card account through the system.

1. **Record Lookup**

Users can view all transaction records, including deposits, withdrawals, transfers, payment and other flow information.

1. **Utility Payments**

In our online banking system, users can conveniently pay their utility bills with valid bank accounts.

1. **Water Bill Payment**

Users use choose one of their existing bank accounts which has enough balance to pay for their water bills.

1. **Electricity Bill Payment**

Users can use one of their existing bank accounts which has enough balance to pay for their electricity bills.

1. **Gas Bill Payment**

Users can use one of their existing bank accounts which has enough balance to pay for their gas bills.

1. **Assistance Requests**

By launching an assistance request, a user can help their registered relative to use certain system functions i.e. Paying utility bills and Making Transfers. The assistance request launched by the actual benefiviary's relatives will then be stored by the system and will be presented to the actual beneficiary with an instantaneous message presenting on the actual benefiviary's side, pending his or her approval. Eventually, the actual beneficiary of the assistance request can approve or decline that request. If an assistance request is approved, the corresponding system function will actually be conducted by standard business logic. Otherwise, the declined or undeal with assistance requests will be stored in the database with tags.

1. **Launching an Assistance Request**

There are currently two available assistance request types in our online banking system, i.e. utility bill payment assistance and money transferring assistance.

1. **Launching a new utility bill payment assistance request**

Users can launch a new utility bill payment in our system simply by providing the ID number of the home owner. Also, valid bank account information is required.

1. **Launching a new money transferring assistance request**

Users can launch a new money transferring assistance request in our system by providing the ID number of the account owner who is transferring his or her money out of this account. Also,two valid bank accounts information is required.

1. **Process Assistance Requests Generated by Desginated Relatives**

Every time a user's registered relative has launched an assistance request for him or her, the user will receive a message about the request. The user will then see this request presenting on the assistance requests processing page. The user can then choose whether to approve this request or not. If the choice is yes, the request will actually become a valid operation in the system, all corresponding business logic will be conducted. But if the user considers this assistance request faulty or improper, he or she can simply decline this request, and the system will ignore the actions meant to be conducted in the request.

1. **View All Assistance Requests Generated by Desginated Relatives**

Users can view all of the assistance requests that have ever occurred to them, whether they are approved or not.

1. **Concierge Service**
2. View All Concierges

When you query the concierge list, you will know ample information about concierges who are at service.

1. Make an Appointment with a Concierge

By launching an concierge service, a user can make an appointment with concierge considering time, gender and language preference.

1. View the appointment

After making an appointment, this page will show your appointment information.

**II. Architecture Design**

1. **System architecture design**

The system architecture design is divided into three large levels, and the design ideas of the overall application system can be fully demonstrated through the effective division of the hierarchical structure.

**User Layer**

The user layer refers to the various types of users using the system to communicate with the service layer through a wired or wireless network to perform corresponding work. The user layer of this system includes the terminal display layer. The terminal display layer means that the system displays the corresponding data, services, applications, and protocols to the user through the PC terminal, and the data and services need to be obtained by requesting the request processing layer in the service layer. The users of this system are mainly the elderly who can simply operate the computer but cannot use the application proficiently. The front-end uses Vue as the framework, interacts with the back-end through axios, and transmits information in Json format to complete basic logic functions.

**Tomcat Server**

The system is arranged on a Tomcat Web Server, using the SpringBoot framework as a whole, divided into three levels, namely Request Processing Layer, Business Logic Layer and Data Access Layer, and uses message queue middleware.

**Request Processing Layer**

The request processing layer can also be called the Web layer, which is responsible for receiving terminal display layer requests and responding to the terminal display layer. The terminal display layer uses the HTTP protocol to send a request, and the request processing layer receives the HTTP request and completes the response. The Spring Controller is responsible for receiving requests, at the same time calling the business logic layer for business processing, and then responding to the terminal display layer with the processing results.

**Business Logic Layer**

The business logic layer can also be called the Service layer, which reflects the core value of the system architecture. This layer uses the SpringBoot framework, realizes data processing by calling the Data Access Layer interface, and at the same time provides an interface to the request processing layer, which has a link between the previous and the next. This layer is the main layer that implements business logic and provides service functions.

**Data Access Layer**

The data access layer can also be called the Dao layer, which is an abstract layer that facilitates applications for data read and write access. This layer uses the Mybatis framework, and CRUD of the database can be realized through the Mapper layer to solve the problem of accessing the database.

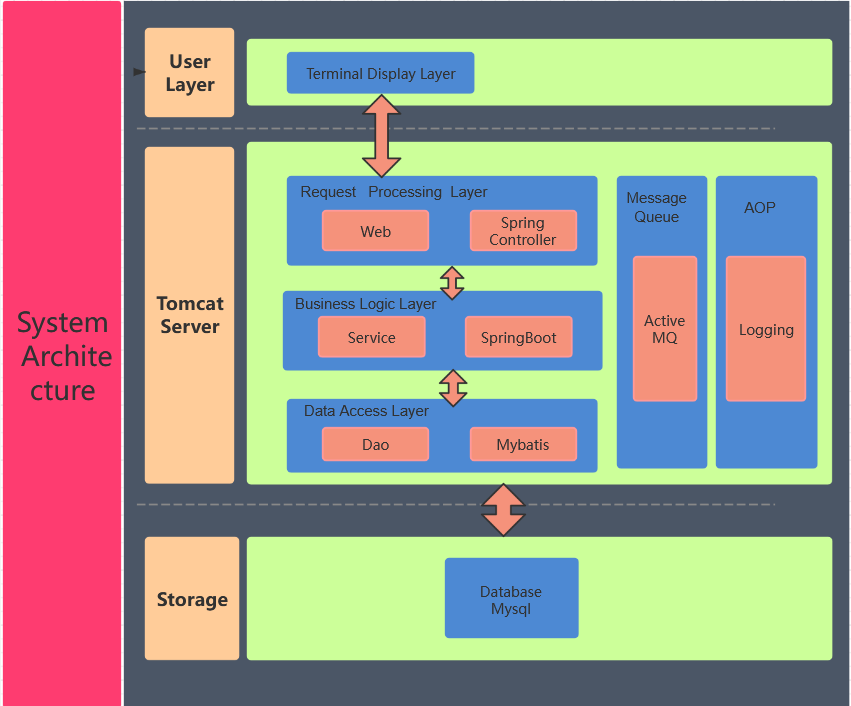
**Java Message Service**

This system uses the message queue middleware ActiveMQ, which can provide the system with efficient and flexible message synchronization and asynchronous transmission processing, storage and forwarding and reliable transportation.

**Storage**

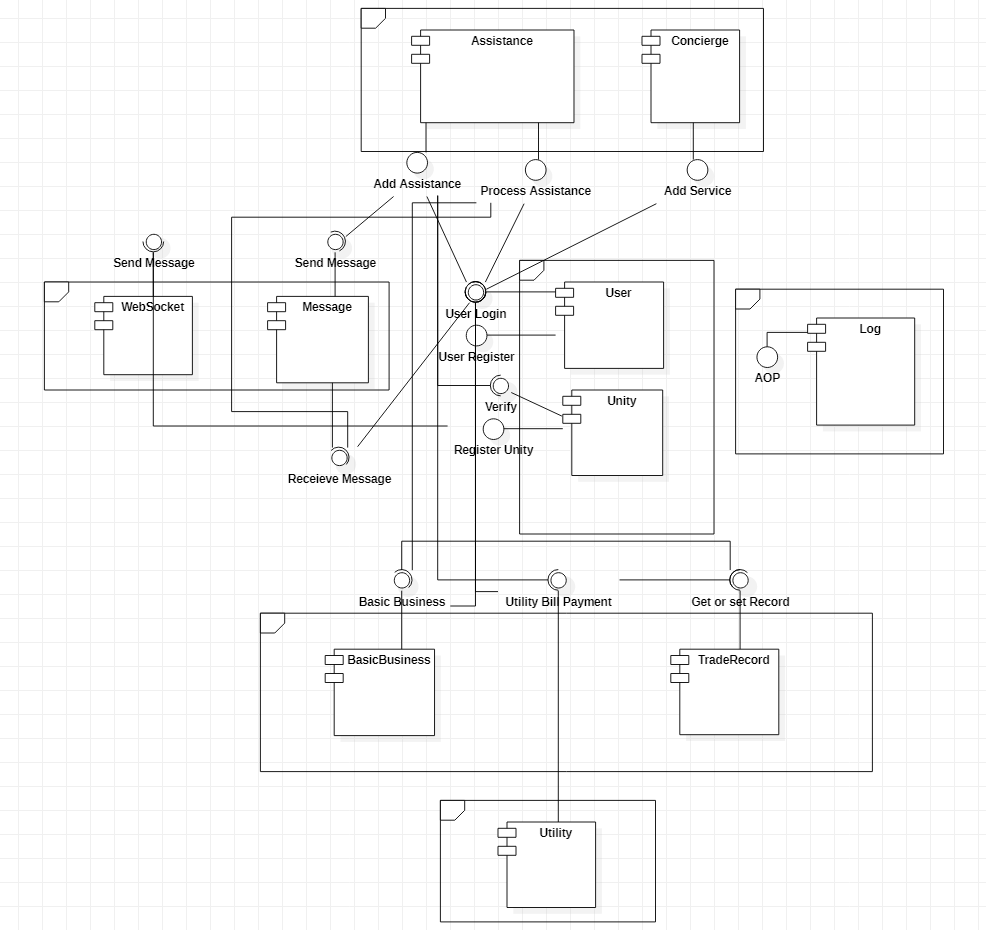
The storage of the system data uses Mysql Database, and completes the interaction with the Data Access Layer.

The system architecture design is divided into three large levels, and the design ideas of the overall application system can be fully demonstrated through the effective division of the hierarchical structure.



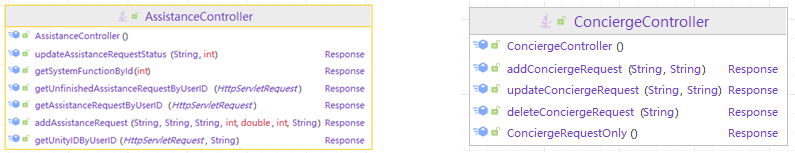
1. **Component design**

In order to organize and implement our banking system in a better way, we've done a detailed component design. We've divided our system into six major subsystems namely Assistance Subsystem, Message Subsystem, User Subsystem, Basic Business System, Utility Payment System, and Log Subsystem. Each specific subsystem contains several components to help fulfill its functions and all the components from different subsystems can cooperate with each other through a series of interfaces.



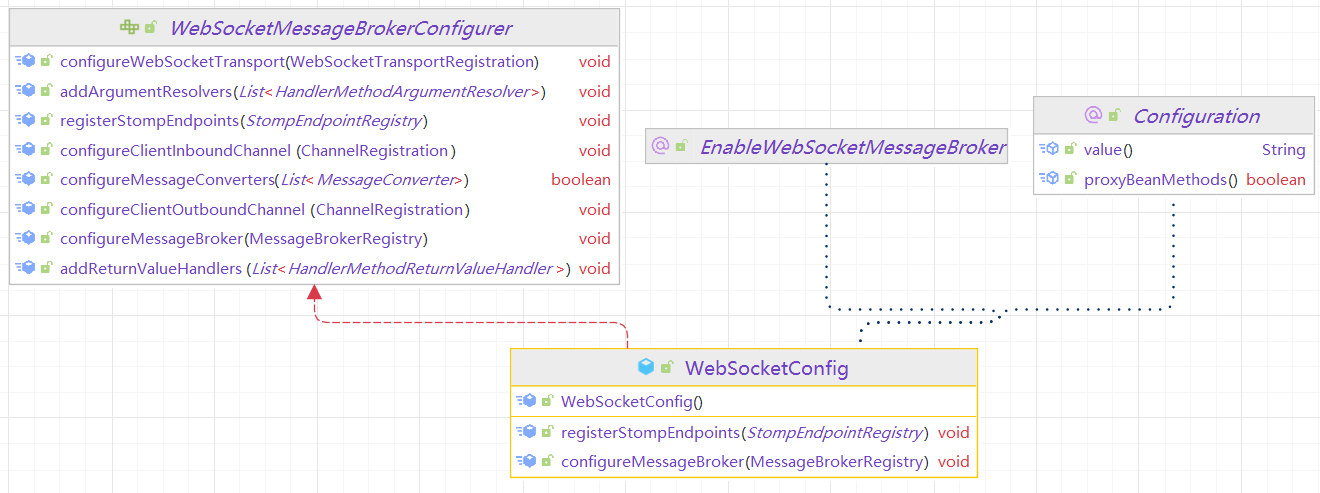
1. **Assistance Subsystem**

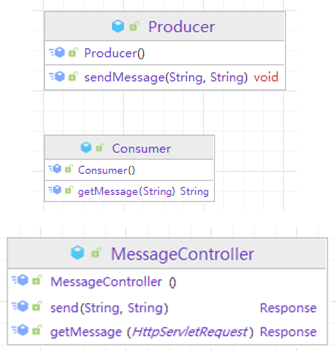
Within the assistance subsystem, there are two major components: assistance component and concierge component. The assistance component is responsible for handling users' assistance requests and helping users process them. The concierge component is mainly responsible for users' requests for booking concierge service while they're banking.

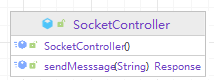


1. **Message Subsystem**

There're two components that lie within the message subsystem which is the message component and the websocket component. The message component takes the advantage of activemq to implement the JMS service. Every time a user launches an assistance request, the message component will send a message to the target user to inform him or her of the pending requests. Also, apart from ActiveMQ's message queue, we've used WebSocket as a tool to send messages to designated users while their relatives are trying to launch an account association request.

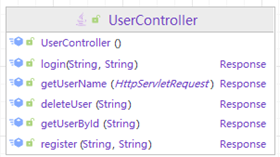


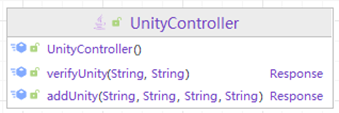




1. **User Subsystem**

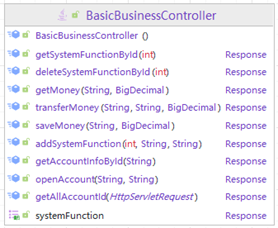
The user subsystem contains two components. The user component is mainly responsible for ordinary user login and registration, along with the management of users' token. The unity component is responsible for the registration of account associations. Only accounts in associated accounts are allowed to launch an assistance request which their registered relatives could see.





1. **Basic Business Subsystem**

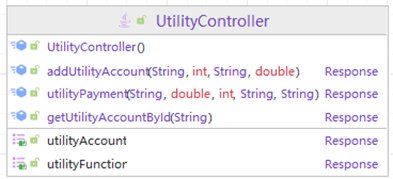
There are two components within the context of basic business subsystem. The basic business subsystem is responsible for handling usual bank services like money transferring or online account opening. The trade record component is responsible for generating users' trade records every time a user has made a payment in the system.





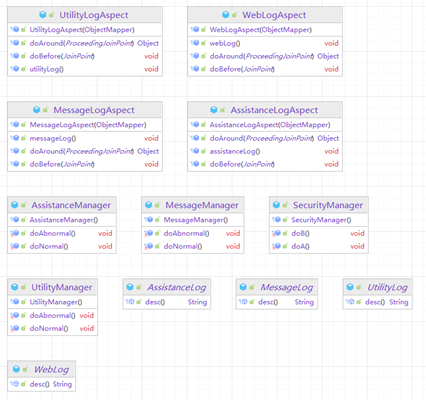
1. **Utility Payment Subsystem**

The utility payment subsystem contains a single component which is the utility component. The utility component is responsible for handling users' requests to pay utility bills like water bills, electricity bills or gas bills.



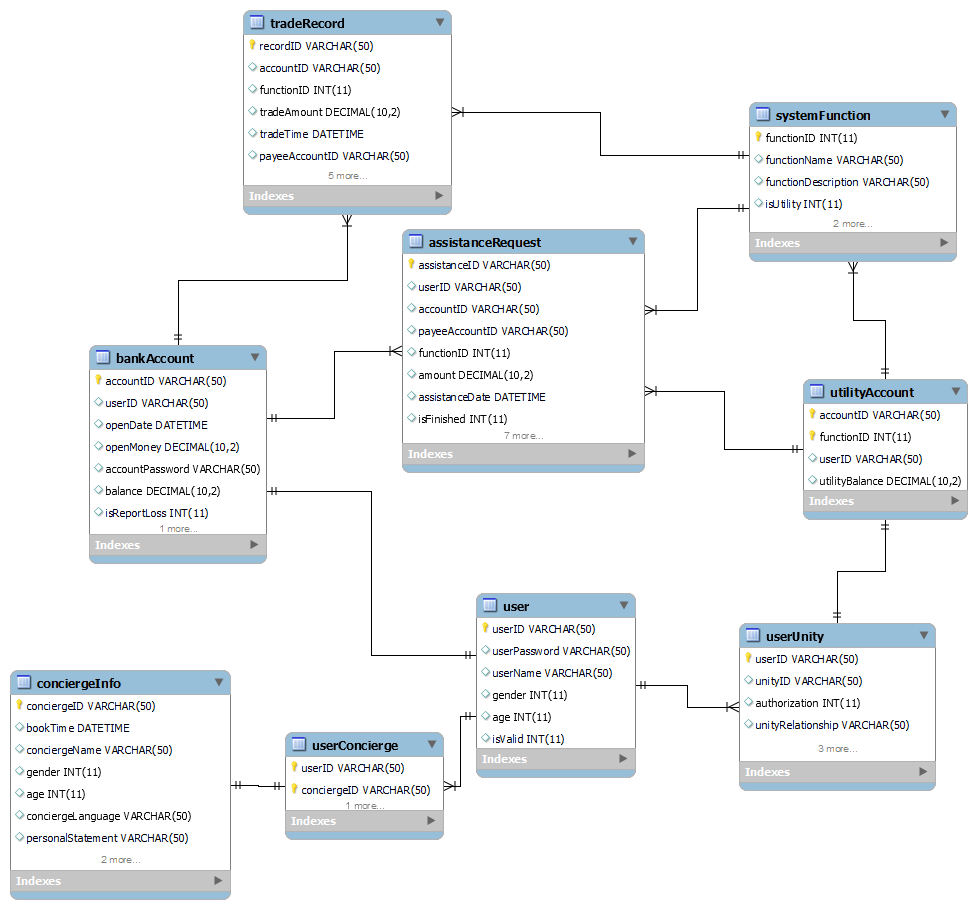
1. **Log Subsystem**

The log subsystem contains the log component which is responsible for system logging. Thanks to aspect oriented programming methods, we are able to log every request of its url, IP address, request duration and even parameters.



1. **Database Design (ER Diagram)**

The database used by this system is MySQL database, and the table design and relationship can be seen in the figure below.



**IV. Other Technical Details**

1. **System Deployment**
2. **Springboot Deployment**

Thanks to Aliyun's Elestic Computing Service, this system's back end service along with the MySql database and ActiveMQ middleware are deployed onto a server with the IP address of

106.14.25.240.

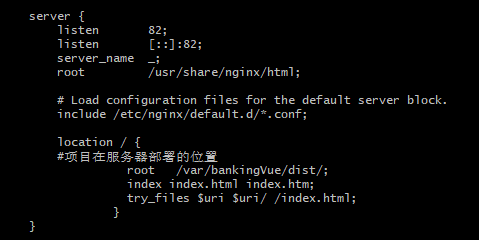


We've used Maven to pack our Springboot service and have uploaded the package onto the server. In order to keep the Springboot service running, we've used Nohup to run the package.



1. **Vue Deployment**

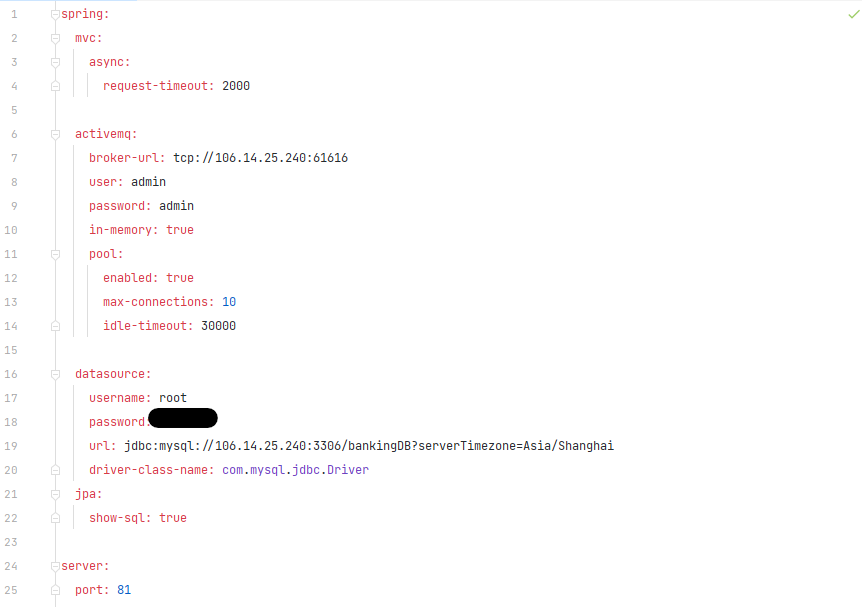
We've also deployed our website to the remote server, using Nginx. Users can visit our website with the IP address of the server via port 82.



1. **System Configuration**

Since the whole Springboot package as well as the Vue package of our system's back end and front end has been deployed, our system has also been configured accordingly.

On the server's side, port 61616 is reserved for activemq Java Message Service. Port 3306 is reserved for mySql database. And port 81 is reserved for web requests. Also, port 82 is reserved for visiting our web pages.



1. **The usage of Java Message Service to Deliver User Notifications**

As is mentioned above, every time a user's registered relative has launched a new assistance request for the user, the user who is being assisted will receive a message informing him or her that a new request is pending to be processed. This is achieved thanks to the powerful Java Message Service.



In order to implement this function, we've used activemq as middleware, creating a message queue for every individual user. Every time a new assistance request is launched for a user, a message will be delivered to the corresponding message queue. The beneficiary of this request will then automatically receive this message when he or she opens the web page to process the assistance requests.

1. **The Implementation of Aspect Oriented Programming in Logging**

Due to the special nature of the banking system, logging is of vital importance. So we've introduced aspect oriented programming methods to implement logging for every important interface. That is to say, every time a user has triggered a request from the web page, a corresponding log will present on our server, logging important information including the user's IP address, the request's URL, and the request parameters.

Generated

We've programmed the Before and Around methods to monitor the requests according to their parameters. The warnings in logging will appear if unusual empty parameters or unusual parameters should be detected.

Generated

1. **The Usage of Token**

As a measure to make our website safer, we've used token to secure user activities. A token is issued to the user upon his or her login and contains information of username and time stamp. Every time the user is making a request from the website, the information inside the token is required to ensure that the user making the request is legitimately logged in. We've used the powerful tool Hutool to generate user token from the back end, and once a user has got the token, the token will be stored in the web broswer's local storage until the user logs out or the token itself expires.



1. **The Usage of AES Encryption**

In our banking system, there are many user scenarios where passwords are needed to verify a payment. As a result, the password will always need to be transmitted to the backend to be compared with the true password stored in the database. To make sure that the password is secured during transmission and storage, we've encrypted the password with AES encryption before the password is on its way to the backend. As a result, the user password stored in the database is also encryoted. We've fully taken advantage of the powerful tool crypto to realize this function.



1. **The Usage of Transaction**

Spring supports programmatic transaction management and declarative transaction management. Declarative transaction management is based on AOP. Its essence is to intercept before and after the method, and then create or add a transaction before the target method and submit or roll back the transaction according to the execution after the target method is executed. Declarative transactions only need to make relevant transaction rule declarations in the configuration file (or through @Transactional annotations), and then the transaction rules can be applied to the business logic.

Declarative transaction management keeps the business code free from pollution. Compared with programmatic transactions, the only disadvantage of declarative transactions is that the finest granularity of the latter can only be applied to the method level, and cannot be applied like programmatic transactions to the code block level.

There are also two common ways of declarative transaction management, one is based on the xml configuration file of the tx and aop namespaces, and the other is based on the @Transactional annotation. @Transactional are used here, simple and refreshing.

1. **The Usage of Websocket**

Since every user needs to associate their accounts with their designated relatives if they want to benefit from the assistance services, we've used web socket as a tool to send messages to the user whose account is being associated. Once a user has launched a request to associate their account with another user, the user on the other side will instantly receive a message informing him or her of the upcoming request.