Criterion B: Design Document

* Some original sketch uses my native Chinese for some of the information. The English translation has been added for important information.

Table of Contents

1. Design Methodology

The overall implementation process would refer to the Waterfall model. The whole process is separated into

6 different serial stages. Each stage relies on the outcome of the last stage. This criterion would cover the first 4 stages. These stages are shown as follows:

- Requirement collection: At this stage, we would communicate with our customers and collect their requirements, thus confirming a rough picture of the system design. The outcome would be some characteristics that are necessary for our product.
- System design: This stage serves as the bridge between clients and developers. Requirements from users would be translated into the specific design of the system. We need to choose suitable technology, software structure, and modules that would satisfy the need of the system design. The outcome of this stage would be a clear design plan for software implementation.
- Implementation: Based on the design plan, we could begin writing, debugging, and testing the source code to implement the various functions and modules of the software system.
- Testing: We need to test and verify, find, and fix defects and errors in the software system. However, this stage is not the key point in this criterion, so it would provide a basic test plan that verifies only main functionalities.

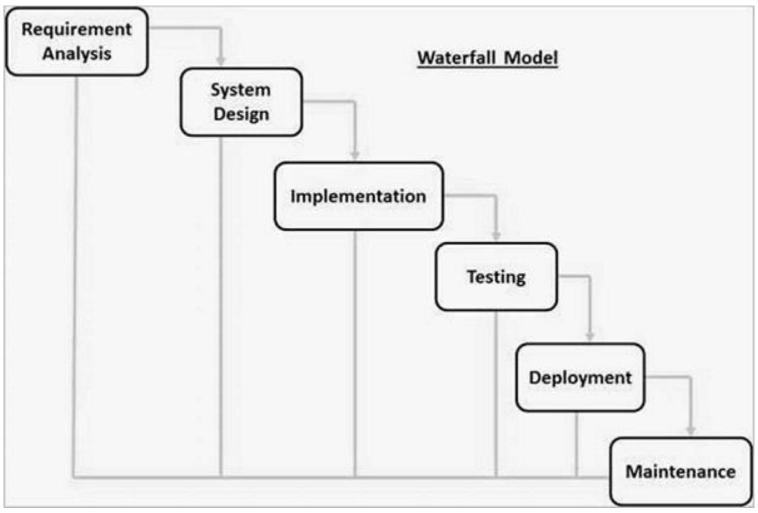


Figure 1.1 Waterfall model

1.1 Requirement Collection

1.1.1. User Story

Our customer is a corporate user. During work, employees often need to pay for business travel or procurement. In the past, reimbursement applications were processed manually, which was cumbersome and inefficient. Employees need to fill out paper form and handle it to the leader for approval.

With the expansion of business, the company now needs a convenient and efficient reimbursement platform to better manage and optimize the whole approval process. The program would serve as an essential part of digitalization, so it should contain the following characteristics:

- Easy accessibility. The platform could be accessed by employees and managers at any time and anywhere they want, without any pre-required apps or environments needing to be installed.
- Complete functionality. The platform should completely take place the old process, including form uploading, process feedback, cost checksum, and result management. Through this platform, it is easy to manage employees' reimbursement actions, finding and correcting errors in time.
- Appropriate budget. The company expected a balance of quality and cost. So, the development period of the platform should not be too long. Deployment of the platform should be designed as a one-stop, without large amounts of system rearrangement and training.
- Load & Concurrent Capacity: Considering the scale of the company, the overall data would not be a huge amount. In addition, it would be a rare condition that many users access the platform within a short period of time.

1.1.2. Requirement List

After all, we could conclude a requirement list as follows:

Table 1.1.2.1 Client's requirement list

Client's requirement	System characteristics	Importance
This platform could help company improve efficiency of the traditional reimbursement process.	A series of functions that would cover the whole process of the financial reimbursement.	High
The platform could be accessed without the limitation of time or space.	The system should be available without pre-requests and any environments.	High
The company expects a balance of quality and cost. The cost and development period should be appropriate.	The complexity and difficulties of technology and the system should be controlled.	Low
The system would be used by limited number of users, and they don't tend to access the	Not many requirements for handling heavy workload or high concurrency.	Low

service at the same time.		
Friendly for user, they could switch to the system without much training.	Well designed user interface and interaction which are user-friendly.	High

1.2 System Design

1.2.1 Goal

A comprehensive design is necessary to meet my client's requirements for functionality, performance, and security. To address this, I first considered the respective reimbursement process for managers and employees, then I expanded on my design by looking at the problem from both the frontend (client) and the backend (server) viewpoints.

1.2.2 Overall Architecture

The overall architecture for the workflow:

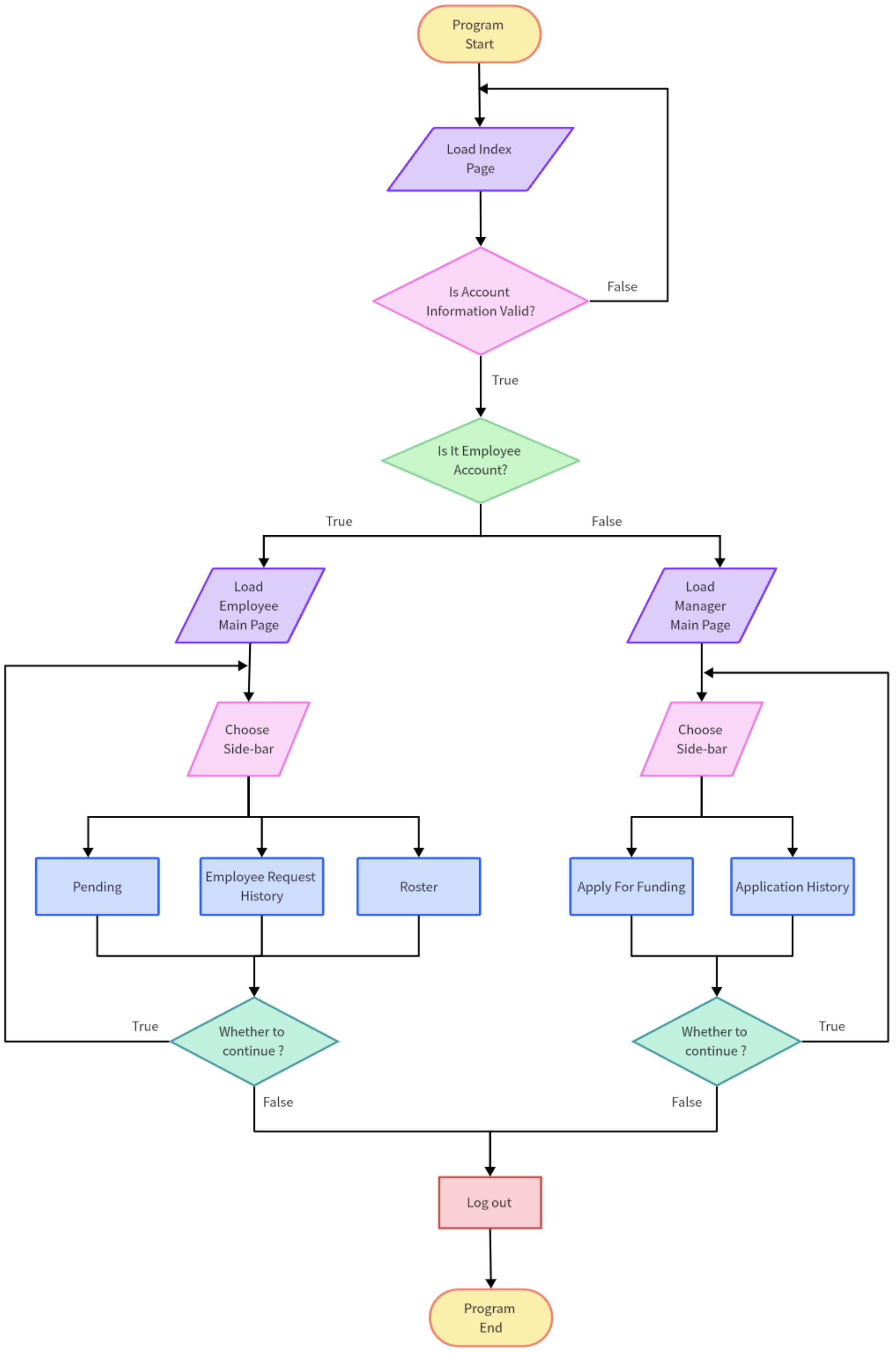


Figure 1.2.2.1 The overall architecture presented in the flowchart

1.2.3 Technology selection

Based on the requirements from customers, we decide to user Browser-Server architecture. In this architecture, the client sends a request to the server through the browser, the server receives and processes the request, and then returns the result to the client for display on the client.

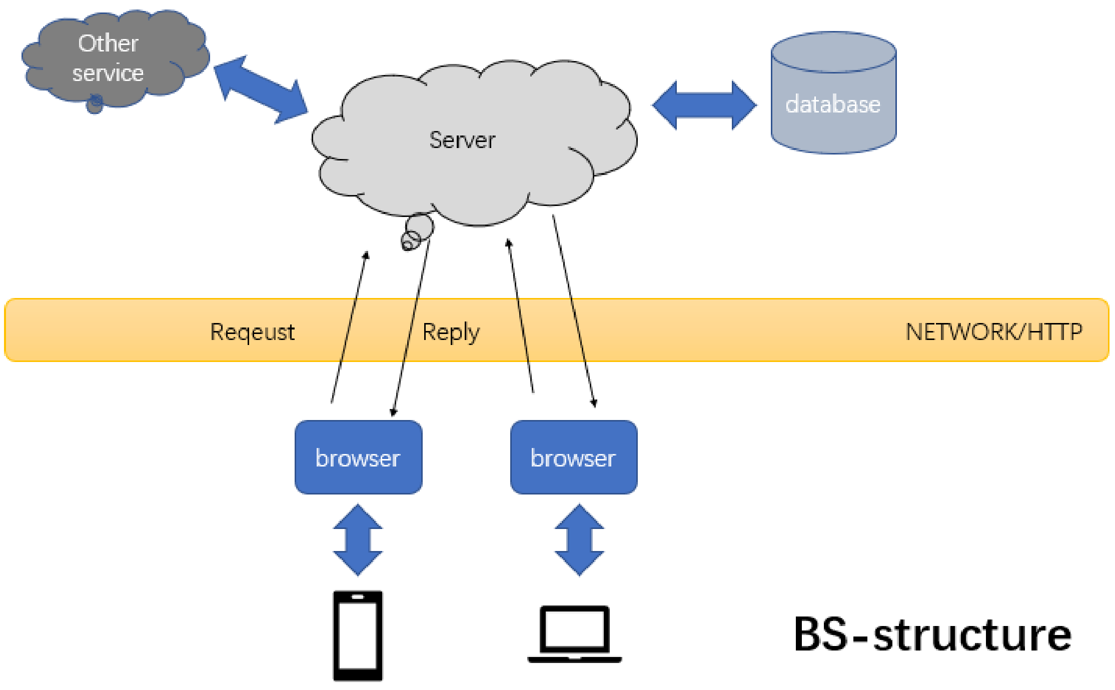


Figure 1.2.3.1 The Browser-Server structure of the system

Applying this structure could satisfy the user's needs. Firstly, it could satisfy the requirement of easy accessibility. Under the B-S structure, users could access the service by browser, which is available on most computers and mobile phones. So, users are not required to install any environment or apps. The second is budget control. B-S structure is one of the most popular architectures with many mature frameworks and infrastructures. So, we could take advantage of a good application ecology to pursue both software quality and development cost.

Go ahead to detailed techniques, I decide to user-separated frontend and backend, based on Vue + Flask + SQLITE. The front end is based on VuejS and Element UI for convenience and effectiveness. Vue is a lightweight JavaScript framework used for front-end construction. It supports reactive data binding, which is suitable for our functions like form filling and submission. And we could take the usage of the component library Element UI to achieve a consistent design on different pages. With the help of these frameworks and component libraries, I could quickly assemble functional frontend pages with minimal workload.

The web framework consists of Flask and several Flask extensions. Flask is a popular Python-based web application framework. It is designed to be simple and easy to use. It supports the most commonly used web application features, such as routing, template engine, form processing, and database integration, which makes it possible to provide stable service with high expandability. I use it for processing and routing requests from frontend, and passing data between the frontend and the backend.

I adapt SQLite as a database. It is a lightweight, embedded relational database. Compared to other traditional BS structure databases, its configuration and installation are easier, and it is very frugal in terms of memory. (Gaffney et al., 2022) It could be well integrated into my system. The access to the database is supported by Flask extensions SQLAlchemy and Migrate. We could use OOP operations instead of SQL for database access and operation, which organizes data and related operations into individual objects, making the code easier to maintain and extend.

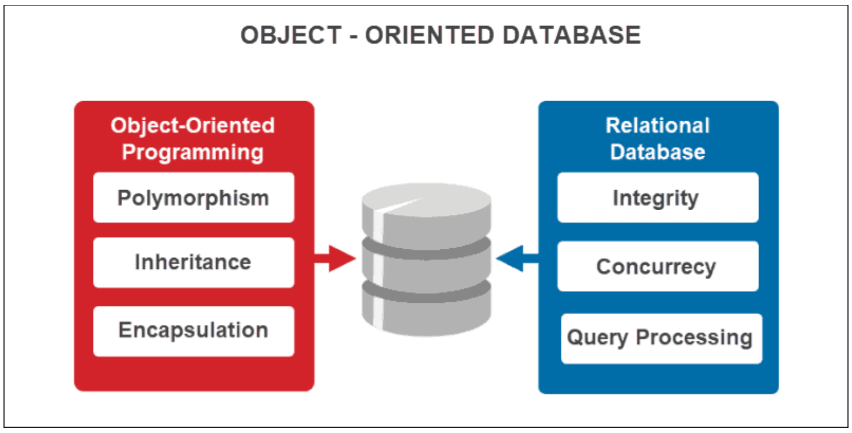


Figure 1.2.3.2 Object-oriented programming database

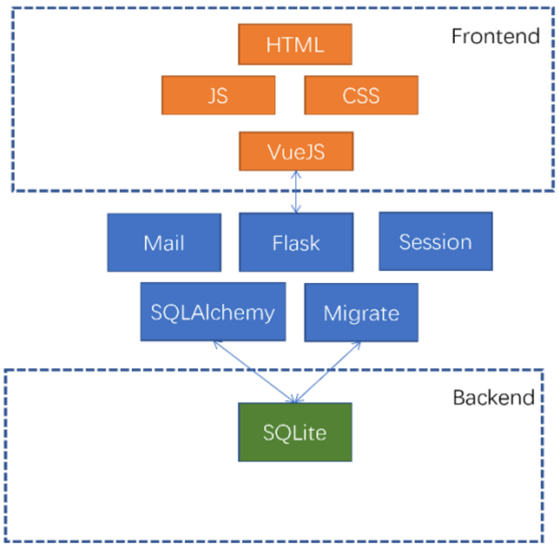


Figure 1.2.3.3 The overall architecture presented by frontend and backend

2 Approaching the Design

2.1 Frontend and User Interface Design

Users would log in with different roles. Despite different roles, the page design is almost the same to save development costs and reach unity in design pattern. Most pages are of the same pattern to achieve consistency.

2.1.1 User-login Page

The user login interface will be divided into two channels, "Employee Login" and " Manager Login". Users can choose their own login identity. (Degott, Borges and Zeller, 2019)

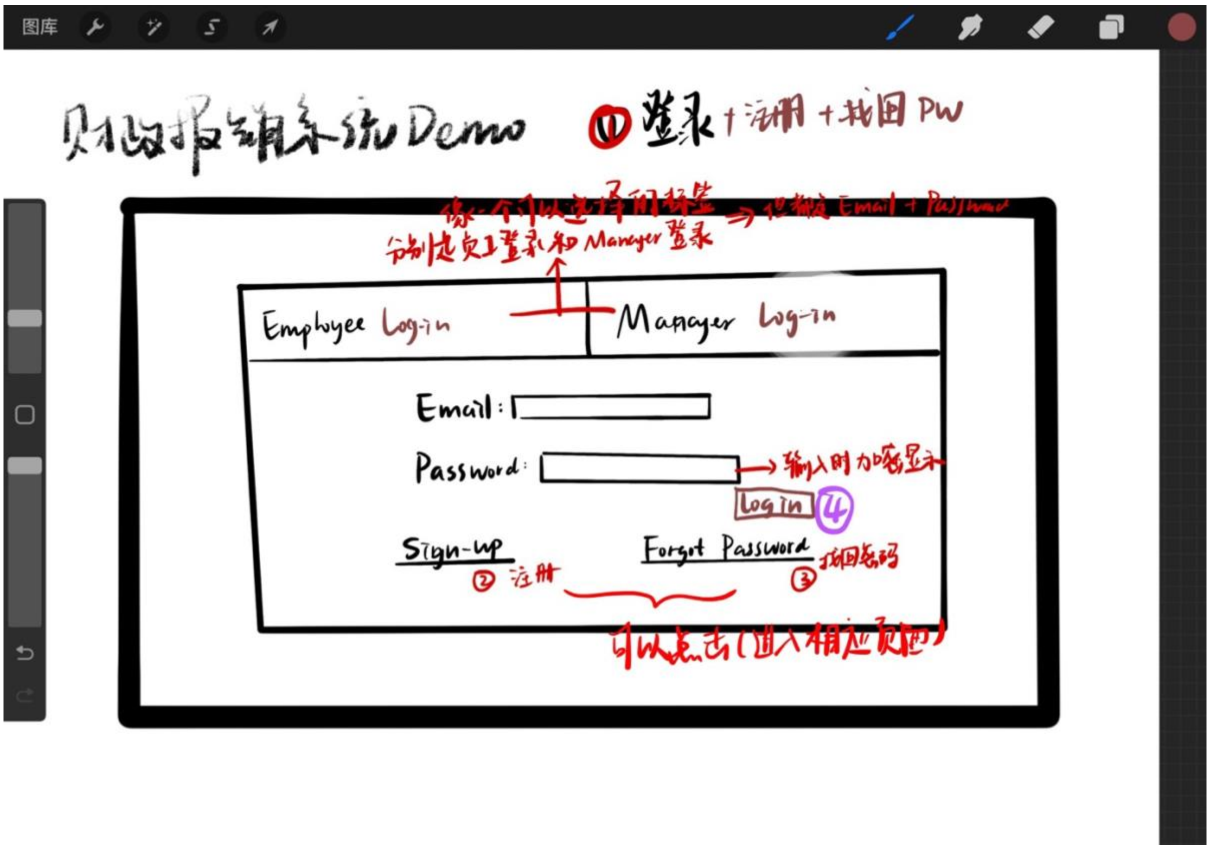
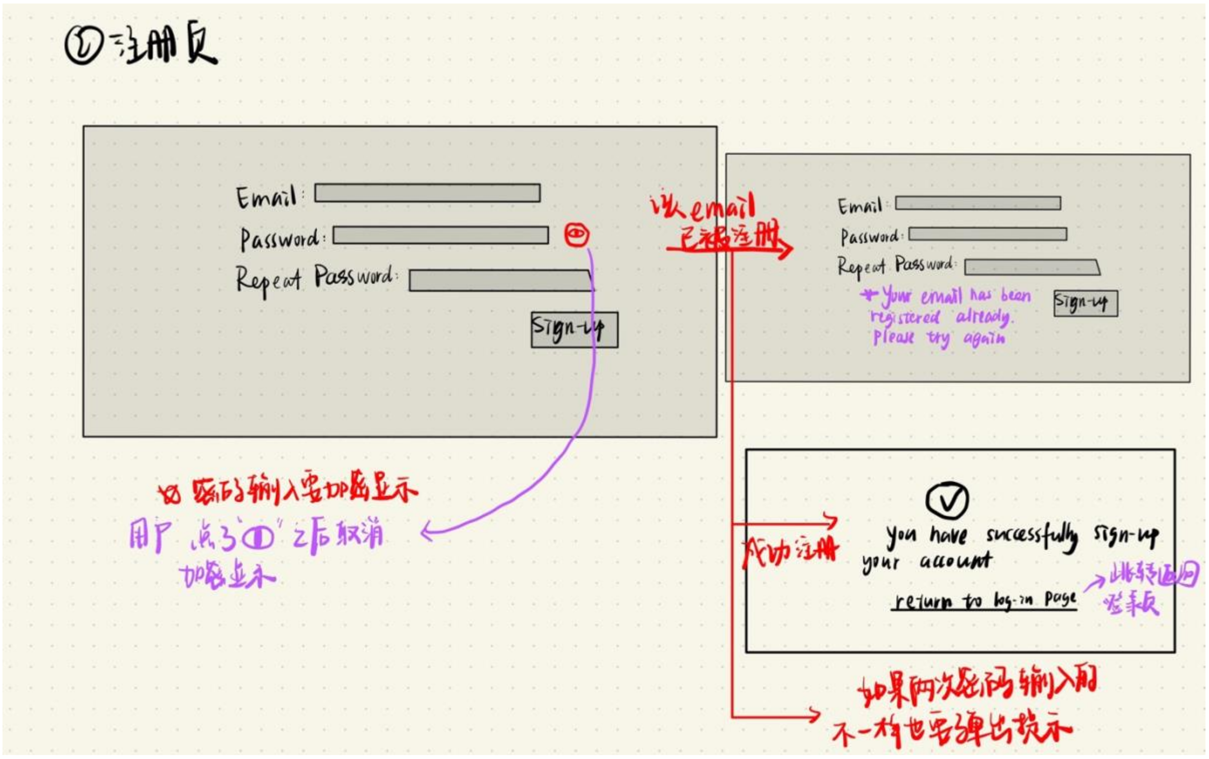


Figure 2.1.1.1 Sketch of login interface design

* The original sketch uses my native Chinese for some of the information. The English translation has been added for important information.

2.1.2 User Registration Page

If the user's information is complete and meets the requirements, the "registration successful" message will be displayed; if the user's information does not meet the requirements (e.g., the email account has been registered or the password is entered differently twice, etc.), the corresponding prompt message will pop up.



①注册页
Email:
Password:
Repeat Password:
Sign-up
该email已被注册
Email:
Password:
Repeat Password:
* Your email has been registered already. Please try again
Sign-up
密码输入要加密显示
用户点了"👁"之后取消加密显示
成功注册
You have successfully sign-up your account
return to log-in page
跳转回到登录页
如果两次密码输入的不一样也要弹出提示

Figure 2.1.2.1 Sketch of sign-up interface design

2.1.3 The Main Menu

Users would log in with different roles. Despite different roles, the page design is almost the same to save development costs and reach unity in the design pattern.

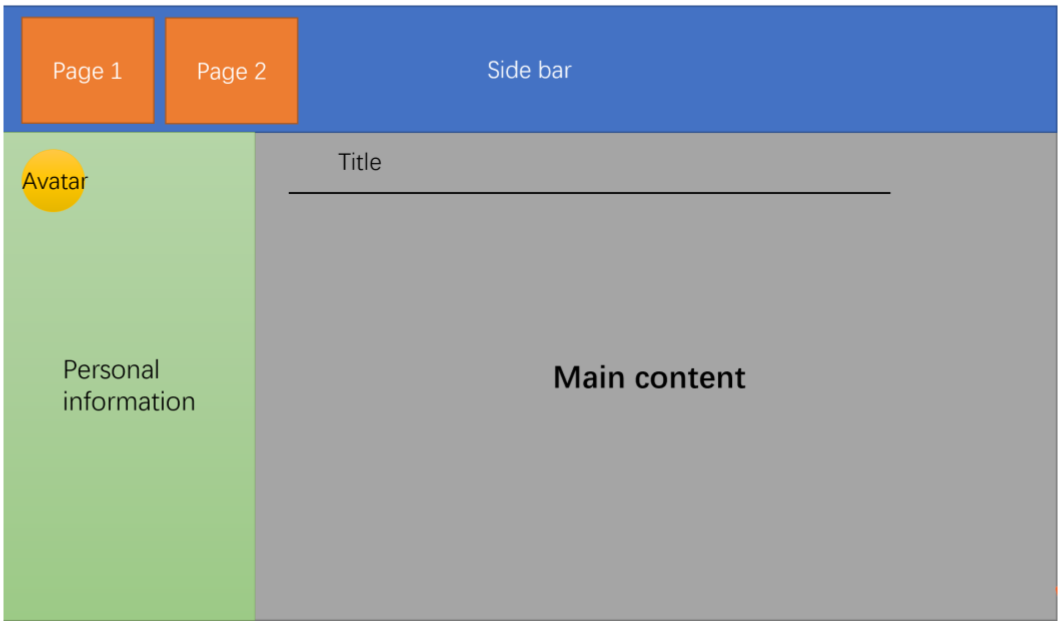


Figure 2.1.3.1 Design 1 for main page

One of the important page designs is main page design. To access different pages, I settle on the sidebar fixed on the left side, containing different page columns. Users could click column to switch to the corresponding page. In the center, its main area shows contents like application forms or history lists. On the right-up area, I applied the avatar icon, which could be replaced by the images uploaded by user themselves. When clicking the avatar, the right hidden sidebar would unfold, and user could view or change their personal information through this sidebar. Actually, there are three types of user interface design. After comparing it to other congeneric products, we choose the final version, since it provides a more tidy and visualized sense. And the hidden bar could give more room for main content.

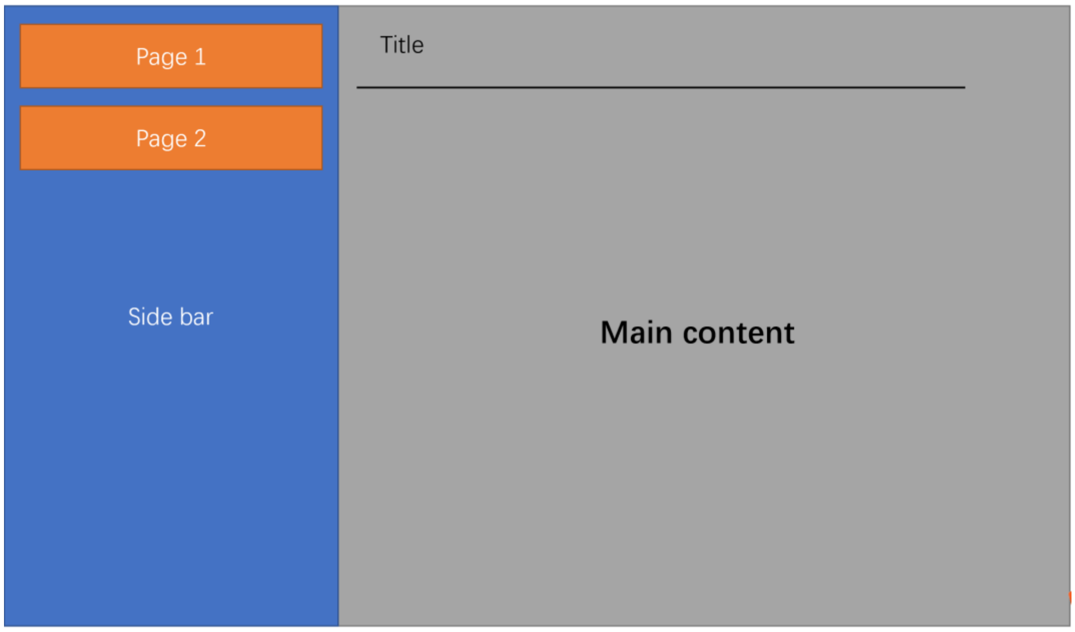


Figure 2.1.3.2 Design 2 for main page

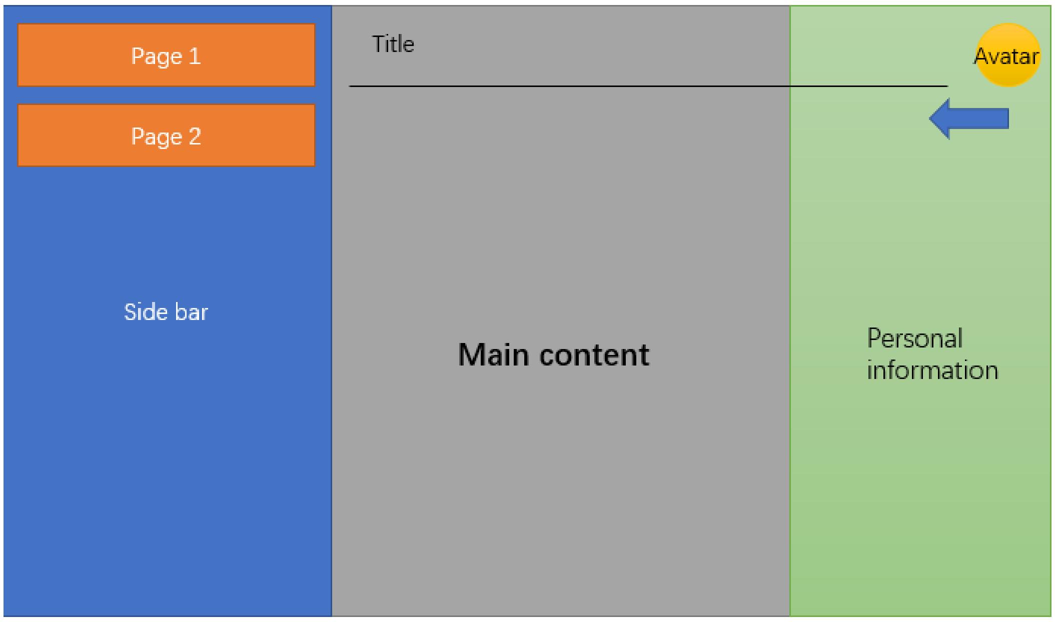


Figure 2.1.3.3 Design 3 for main page (final decision)

2.1.4 Employee Interface

There are two tabs on the left side of the screen after employees have successfully logged in, namely "Apply For Funding" and "Application History". Employees can access the different tabs by clicking the corresponding button.

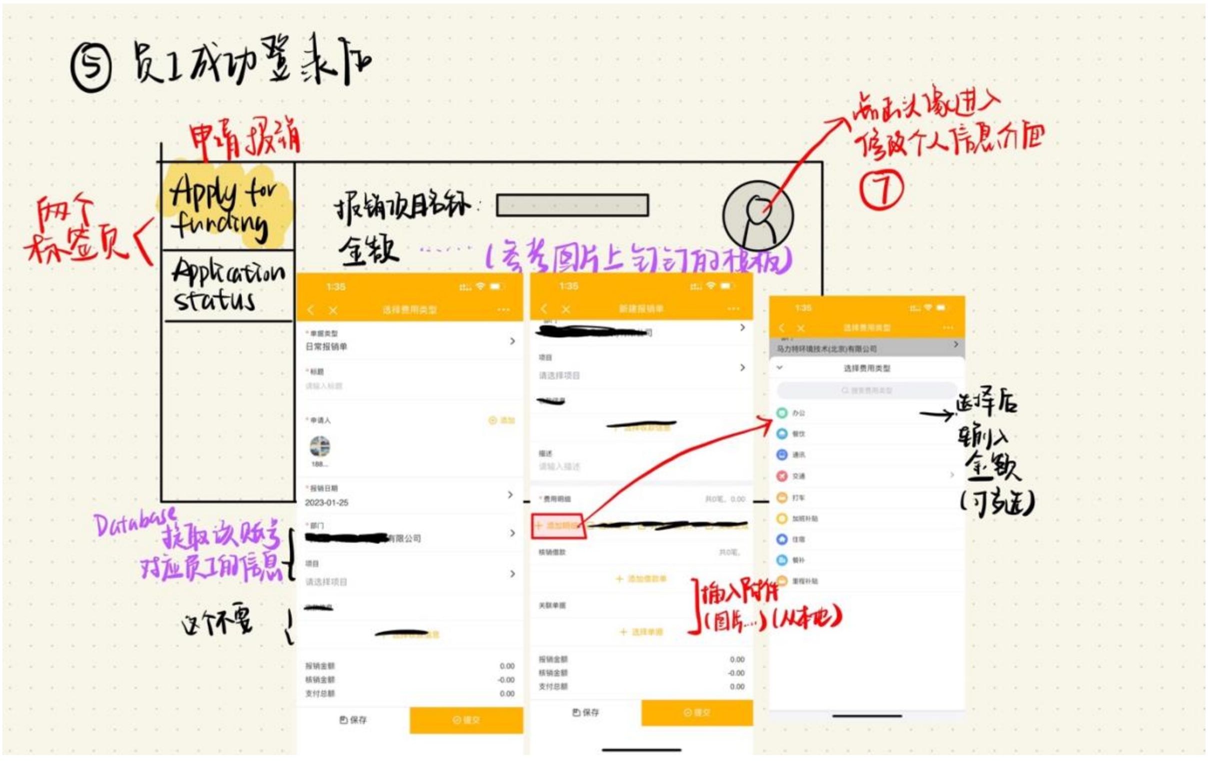


Figure 2.1.4.1 Sketch of employee main page design

When applying for reimbursement, employees need to fill in a series of information, including title, amount of reimbursement, attachments, etc. The reimbursement interface is inspired by the reimbursement application form in Alibaba's office software "Ding Ding".

2.1.5 Manager Interface

Similarly, there are two tabs on the left side of the screen after the manager has successfully logged in, which are "Pending", "Application History", and "Roster". Managers can access the different tabs by clicking the corresponding button.

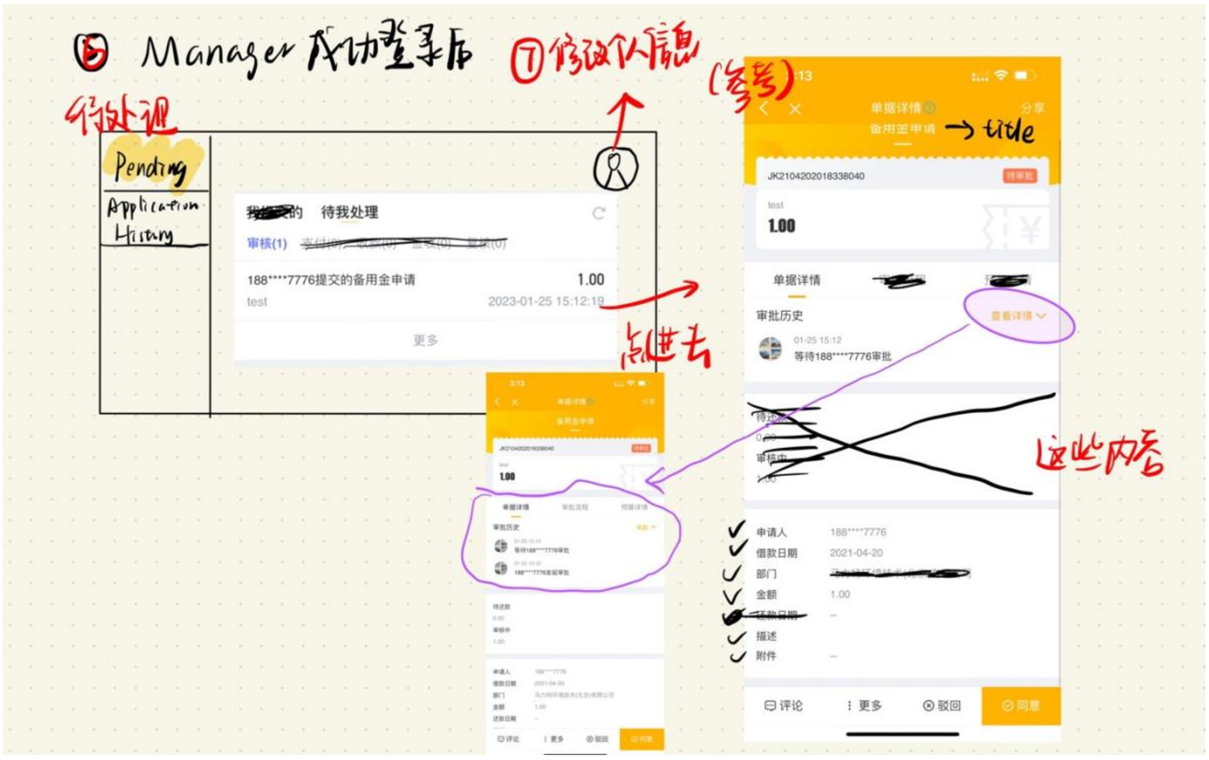


Figure 2.1.5.1 Sketch of manager main page design

The manager can click on the corresponding reimbursement request on this page to get more information and choose to approve or reject the reimbursement.

2.1.6 Personal Information Page

As mentioned above, the personal information page can be accessed by clicking the avatar button located in the upper right corner of the system interface. The Personal Information page presents the name, email address, department, etc. of the employee or manager. This information corresponds to the information in the User form in the Database (see 2.2 Database Design for more information).

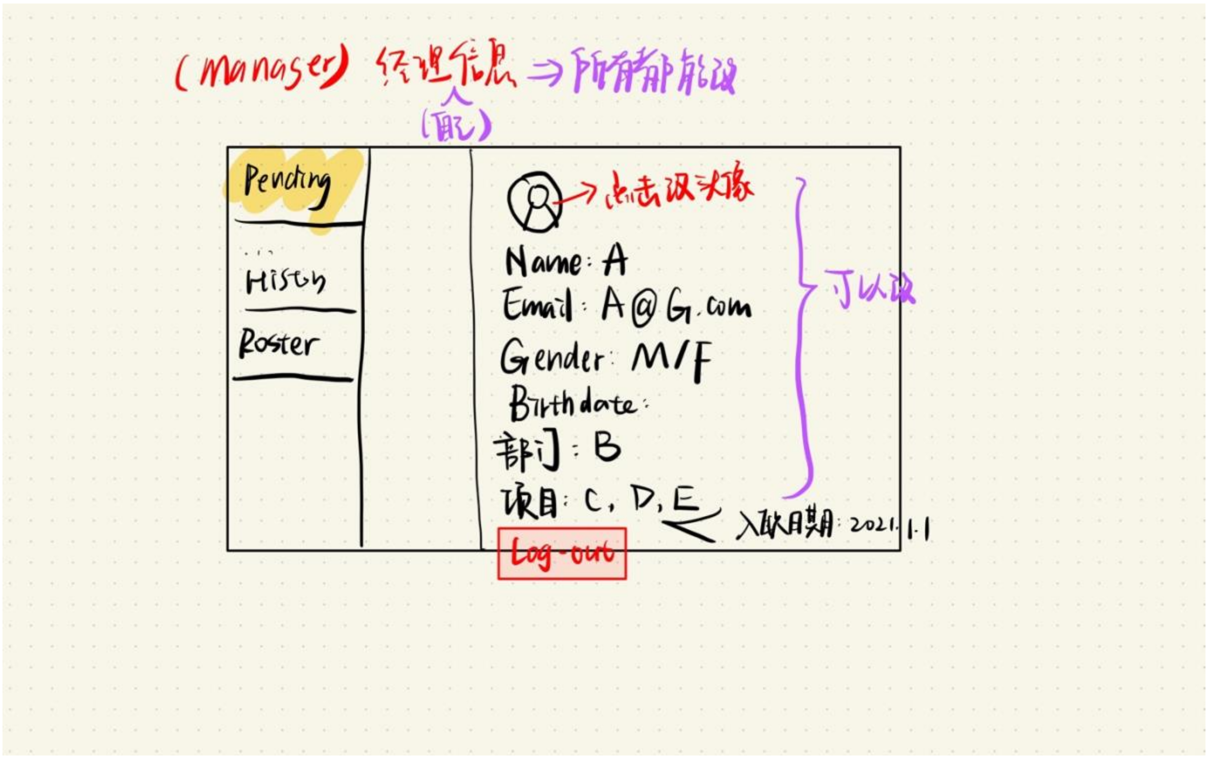


Figure 2.1.6.1 Sketch of personal information page design

2.1.7 Manager's Roster Page

The roster functionality was not presented in the initial design sketches. In a second conversation with my client, she expressed the need for roster functionality to more clearly see the current employees on board. So I added a "Roster" tab to the sidebar of the experience screen.

The roster page presents a list of all current employees and their brief information.

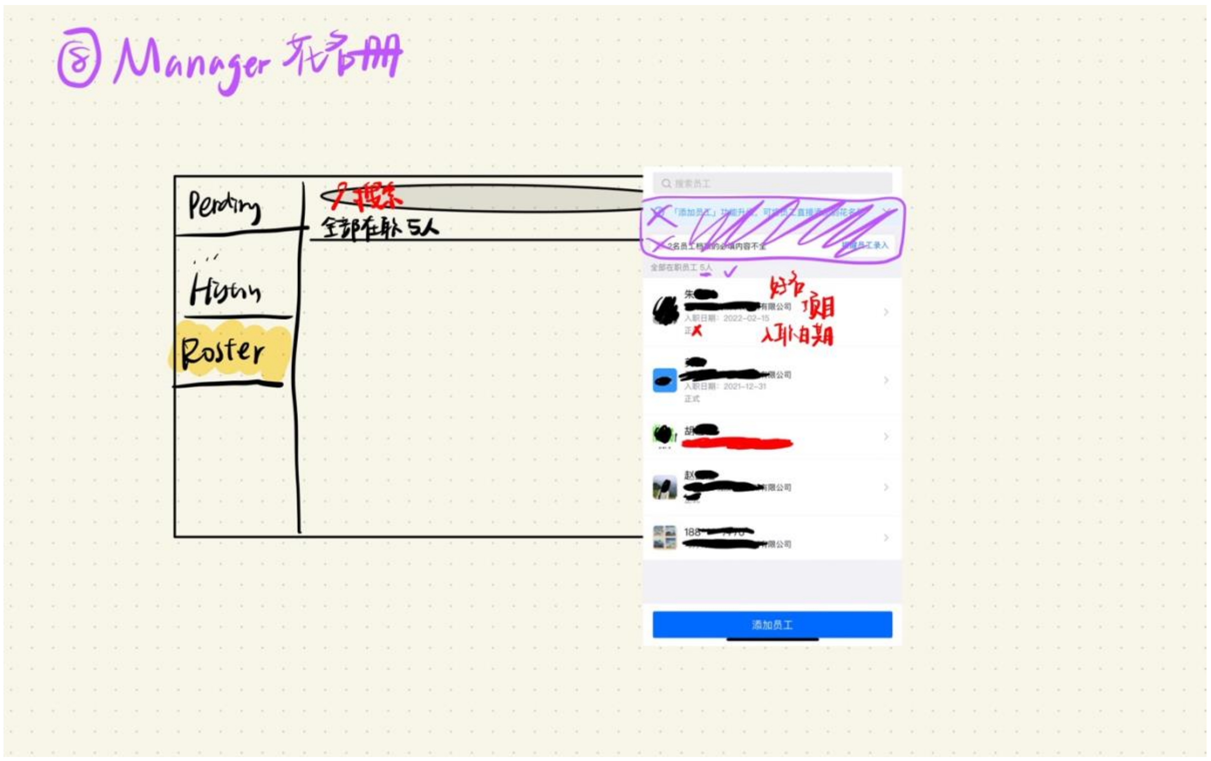


Figure 2.1.7.1 Sketch of the roster page

2.1.8 Reimbursement Form Design

Another important page design is the reimbursement form for employees. After analyzing the paper form used before, we concluded the required information:

Table 2.1.8.1 Reimbursement form required information list

Name	Type	Description
Subject	text	The subject for this application.
Total amount	number	Total amount of money needs to be funded.
Document type	Set of options	A preset options to classify different applications.
Applicant	Default is current employee	Who submits this form.
Date	Date	When is this form written.
Department	Default is current employee’s department	The department of the applicant. Only managers in this department could see the reimbursement form.
Description	text	More description added to the form.

Receipt	files	Usually images, or other digital formats which could support this form.
Attachments	files	Other necessary materials.
Detail of expense	A list of items with types and amount	Subitems for this form, users could add or delete items.

2.2 Database Design Solution

Data storage and data processing are essential parts of this platform. We need to instore the following data:

- Account information: The information about all the users of the platform. This includes basic information like name, address, email, gender, etc. In addition, we need to separate different roles apart to manage authentication.
- Reimbursement information: This kind of information includes all the received application forms from employees, which consists of id, amount, title, belonged project date, etc. It should also contain an index to the uploaded files.
- Approved information. This kind of information supports the management of reimbursement forms. When managers approved or reject reimbursement forms, relevant information including id, opinion, attachment list, and status would be stored in the database.

When initializing the table structure of the database, we use the inherence class of db. Model (See Criterion C for detailed explanation), which indicates this model could be used as an interface for direct query. Unlike the usual database schema that contains relations between different tables, this schema would focus on data classes, and the relations would be realized by OOP codes embedded in the program.

The database schema is shown as follows:

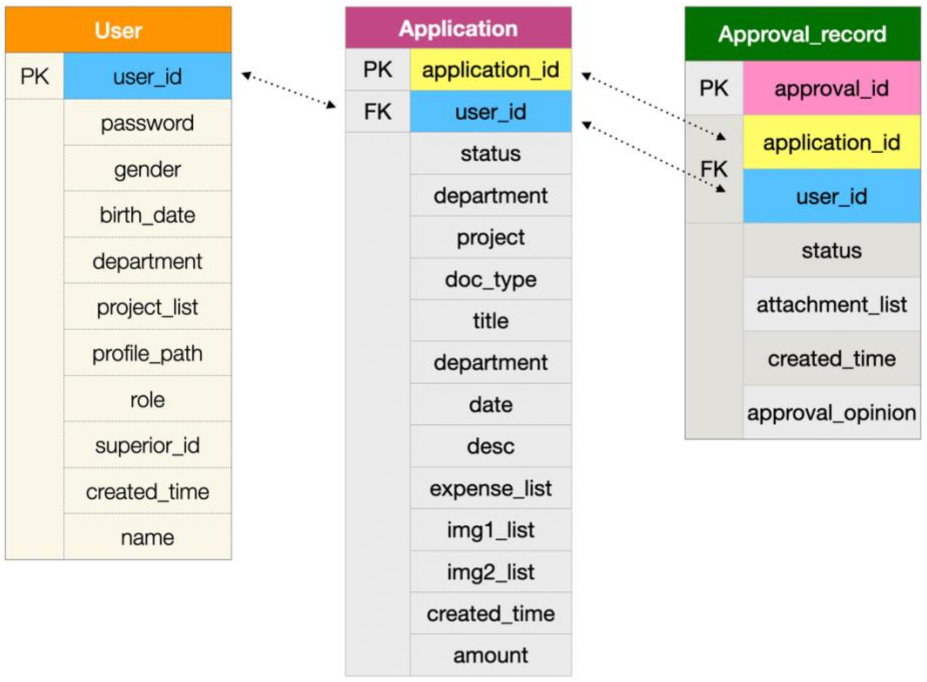


Figure 2.2.1 Database E-R model

The use of a relational database provides referential integrity which helps to ensure data integrity. A more visualized database schema is shown as follows:

```
USER (user_id, password, gender, birth_date, department, project_list,
profile_path, role, superior_id, created_time, name)

APPLICATION (application_id, user_id (fk), status, department, project,
doc_type, title, department, date, desc, expense_list, img1_list,
img2_list, created_time, amount)

APPROVAL_RECORD (approval_id, application_id(fk), user_id (fk), status,
attachment_list, created_time, approval opinion)
```

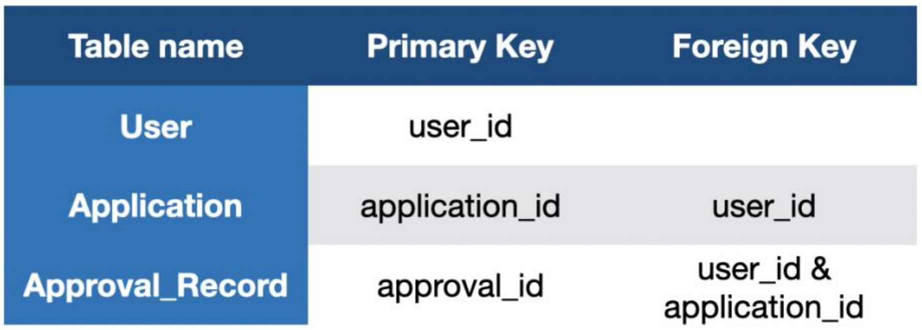


Table name	Primary Key	Foreign Key
User	user_id	
Application	application_id	user_id
Approval_Record	approval_id	user_id & application_id

Figure 2.2.2 Three tables in database and their corresponding keys

2.3 System Workflow Design

The system aims to take the place of the traditional financial reimbursement workflow, so we need to figure out the original process of financial reimbursement, then convert it into the workflow of the system.

In the original process, the financial reimbursement starts from the submission of employees. Employees would fill out a form with information about the reimbursement. Then they would hand the form to the financial department. Each month, the financial department would check up on the collected forms. This includes checking all the required information are provided, and that the sum of all subitems costs equals the result. And then the forms would be documented into report forms and handled by the manager. The manager would decide whether to approve or reject the application. Additionally, the manager could also attach comments. All these results would be marked on the report form. In the end, the report form and the application form would be sent back to the employee.

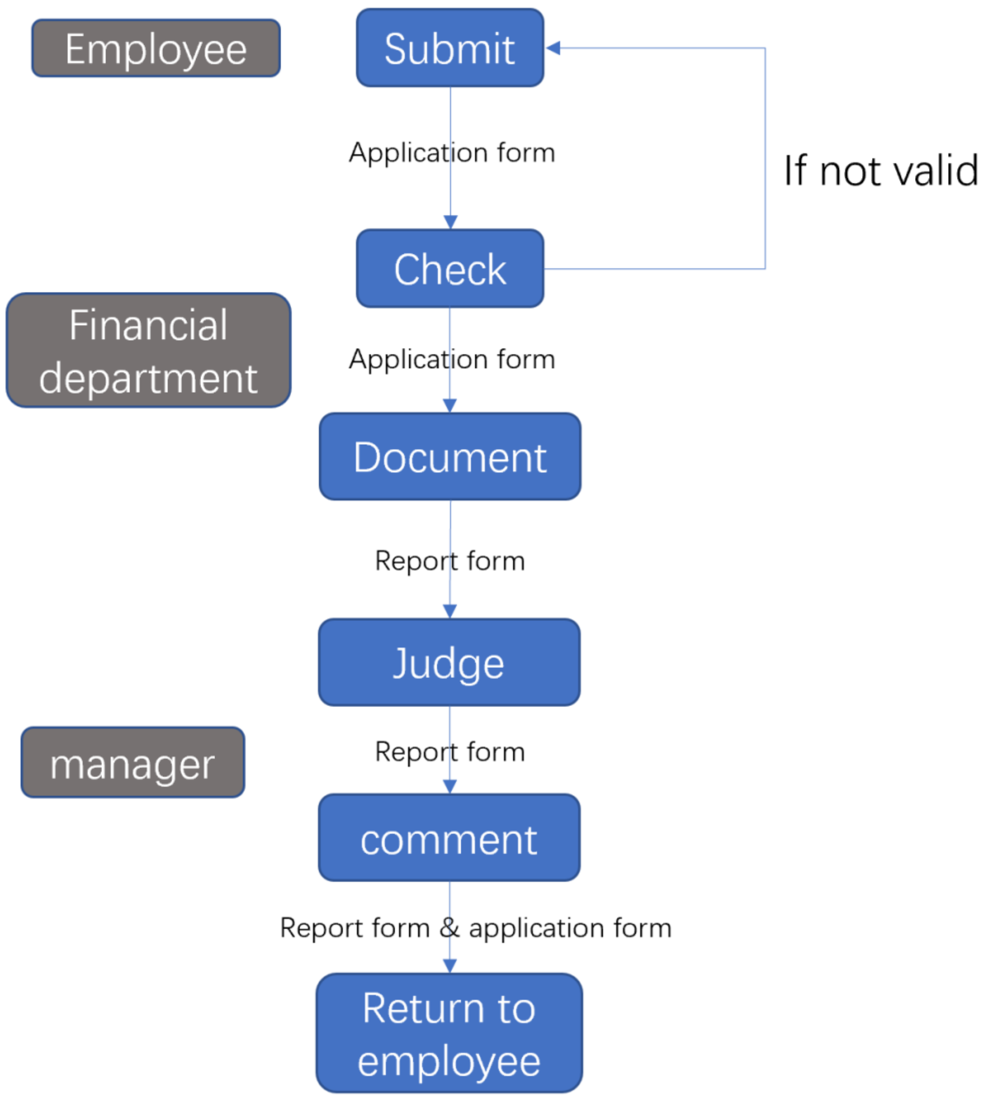


Figure 2.3.1 Workflow for original financial reimbursement

Now our target is to simplify and abstract the above workflow to design our own. We could notice that financial department is responsible for the check and documentation part of the process, which cost lots of manpower and time. This part could be optimized by a program, which could automatically check the forms before being submitted, thus invalid forms would not access the database. Moreover, users want the process of reimbursement could be done without the limitation of time and space. The original workflow is a serial one. For instance, manager needs to wait for the documented forms to start his work. We could optimize the process by in-time updating. After any forms are submitted, the system would automatically process the raw information, and represent them on the manager's end. Also, after judging and commenting, employees could immediately get the results of their application from their end. So, the workflow for our system should be as follows:

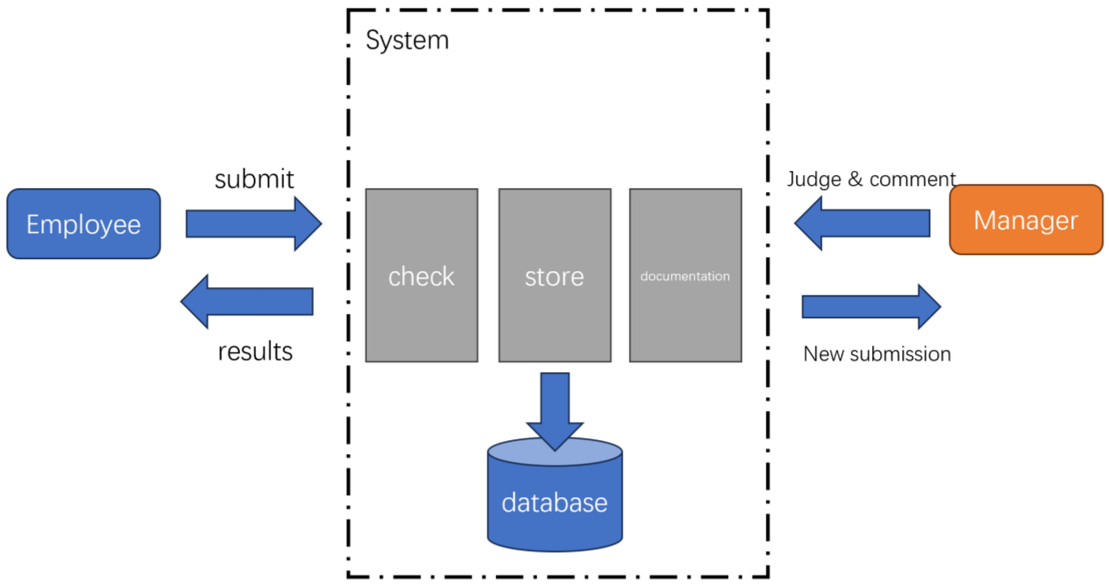


Figure 2.3.2 Workflow for our system

2.4 Functions Design

Based on the workflow design, we could carry on to functional design. We could easily notice that there are two kinds of users with different authority and responsibilities. So, the application would support employee end and manager end.

For manager end, it would have following functions:

- Representation: The documented information should be shown properly. Useless data are hidden while key information is highlighted. If needed, managers are able to view full information on each form.
- Judging and comment: This is the main operation for manager. There should be pages or areas that are easy to operate. Different colors and fonts should be used to separate different options apart.
- Roaster: Mangers could get the roasters of the employees under their command. Basic information would be shown. And managers could also do operations like inviting new users or accepting new registration.

For employee end, it would have following functions:

- Application fulfilling: This would be the main operation area for employees, which would greatly decide the user experience. The user interface should be well-designed to support smooth operation and representation of the data.

- History: Employees could view all the requests they submitted earlier. Also, each request is shown with status. If the application has been processed, they could also see whether it is approved or rejected, and the comments from the manager.

2.4.1 Authentication management

In this platform, I applied Role-Based Access Control. Users share some common pages within the flow. Different roles could access response contents. This could reduce management and development costs. To be mentioned, each user has only one belonging manager, and each manager could manage multiple users. So it's a one-to-many relationship between manager and employee. Reflected in the database, those columns with the "Employee" role would have a manager schema filled with the corresponding manager id, and those with the "manager" role would leave it empty.

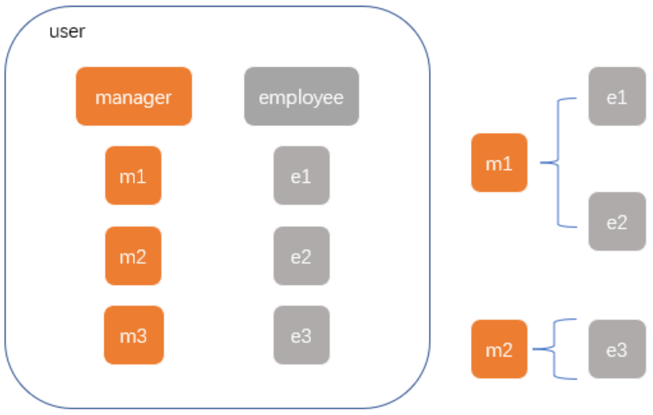


Figure 2.4.1.1 Role-Based Access Control Visualization

2.4.2 Email Verification Process

The "Forget password" function includes: the user submits a reset password request, the system sends a random verification code, and the user resets the password.

More details are presented in Criterion C. Part 4 (c). The general process of the E-mail verification process when in the "forget password" function is shown as follows:

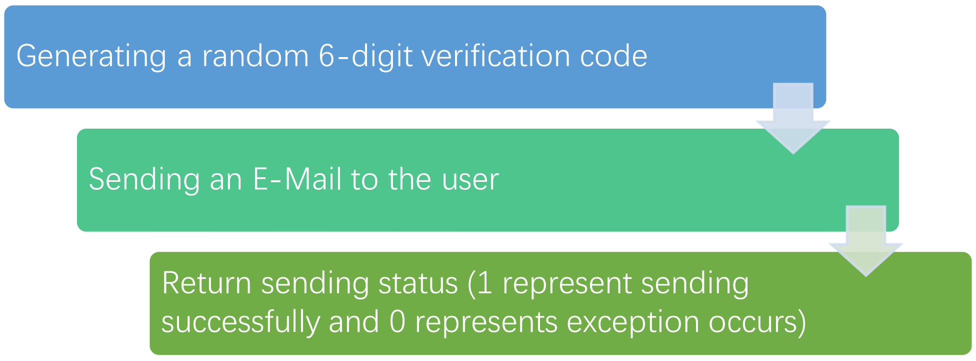

Generating a random 6-digit verification code
Sending an E-Mail to the user
Return sending status (1 represent sending successfully and 0 represents exception occurs)

Figure 2.4.2.1 Email verification processes

2.5 Information safety

Since this platform stores private data such as personal information, reimbursement information, and approval history. It's essential to ensure the information safety of this platform.

Overall, designs to approach information security includes:

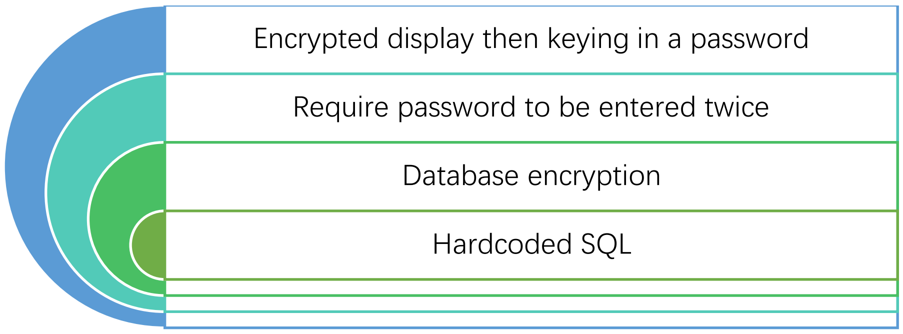

Encrypted display then keying in a password
Require password to be entered twice
Database encryption
Hardcoded SQL

Figure 2.5.1 Security approach designs presenting on a pyramidal structure

For the important login and registration process. The brief login flow is shown as follows:

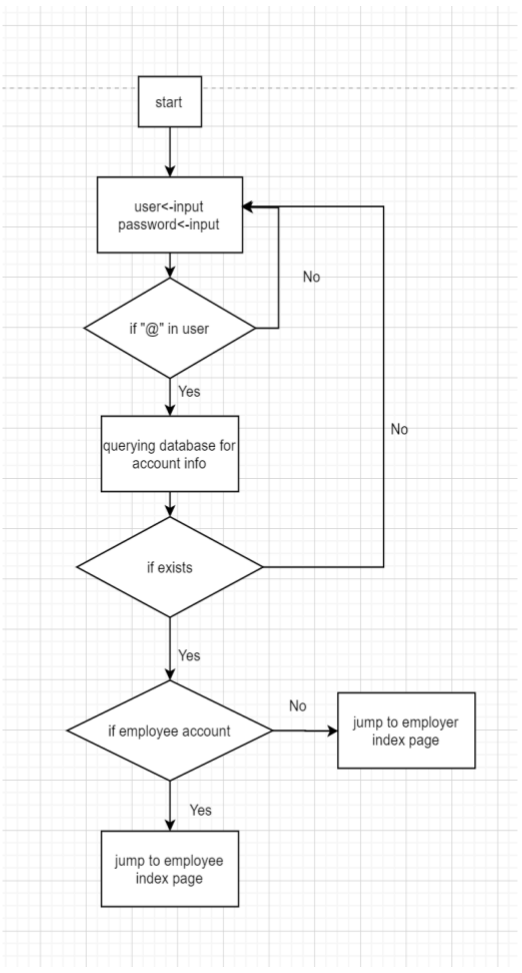

start
user<-input
password<-input
No
if "@" in user
Yes
querying database for account info
No
if exists
Yes
No
if employee account
jump to employer index page
Yes
jump to employee index page

Figure 2.5.2 User Login and registration process

When logging in, the program would first fetch the user and password as a pair and get back to the database. After querying the database with a username. The password stored in the database is encrypted. So the program would first decode the password and compare it with the user input password, thus judging the validation of this login attempt. Sensitive data are processed in the backend (on the server) and can be carefully monitored. When register, the newly added data would be encrypted by a bidirectional encryption algorithm, and the encrypted text would be stored in the database. So, even if the backend is attacked, user information could not be easily leaked.

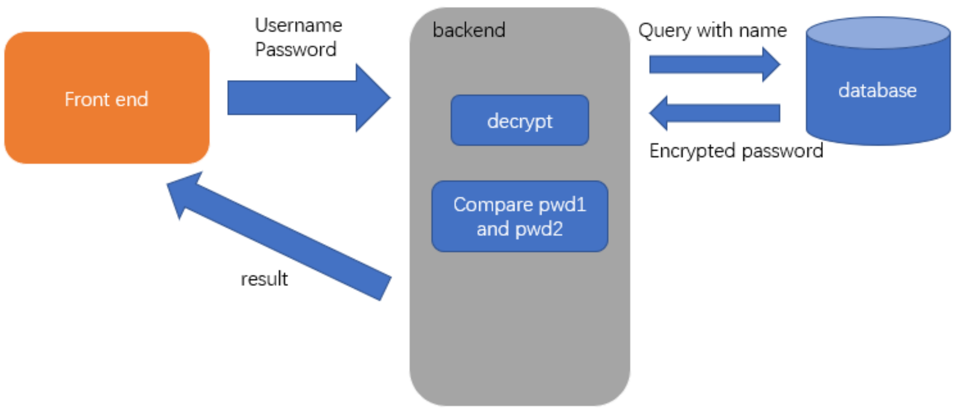


Figure 2.5.3 Program underlying logic that ensures the information security

Moreover, all the SQL is hardcoded in the code for preventing SQL injection. This is based on Flask alchemy. And the login state is maintained by sessions stored in the web browser. When the user does any operations, the program would fetch and verify data from the browser. If not verified, the user would immediately be forced offline, which could prevent invalid login.

3 Test plan

The following steps should be conducted:

Test Point (Success Criteria)	**Test Method**	**Expected Result**
Financial Reimbursement System Workflow Function		
1. For Manager		
Managers can access the "Financial Reimbursement System" module to check the employee's reimbursement requests;	Login to the manager account, click “Pending” in the sidebar	The user (manager) can click on one of the employee's requests to see more details. There will also appear a preview of the request on the "Pending" page.
By clicking on the employee's avatar, managers can view all	Login to the manager account, click the “Employee Request	All historical application records should be presented in

the employee's historical reimbursement request records.	History” in sidebar	a list.
Managers can approve the employee's reimbursement request in the "Financial Reimbursement System" module by clicking a button.	Login into the manager account, click the “Pending” in sidebar. Check a request by clicking the request preview.	After clicking the button "Agree", the manager can input approval comments or add image attachments. Then, the message "successfully approves the request" will pop up.
Managers can reject the employee's reimbursement request in the "Financial Reimbursement System" module and add comments (e.g. why the request was rejected, what reimbursement documents are lacking, etc.)	Login into the manager account, click the “Pending” in sidebar. Check a request by clicking the request preview.	After clicking the button "Reject", the manager can input approval comments. Then, the message "successfully approves the request" will pop up.
2. For Employee		
Employees can request reimbursement, which is required to fulfill the name of the reimbursement, the reimbursement amount, and the notes.	Log an employee account. User: 1234@126.com Pwd: 1234 Click “Apply for funding ” in the side bar.	After filling in all blanks and clicking the "submit" button, the message "The request is successfully submitted!" will pop up.
Employees can add attachments which is optional (e.g. reimbursement request forms, invoice evidence, etc. required by the company's reimbursement process)	Adding images (png. or jpg.) or PDF attachments in the reimbursement form.	Attachments will be presented as thumbnails.
Employees can access the "Application Status" module to check their requests status; they can also view all their historical reimbursement request records.	After login an employee account, click the “Request Status” in the side bar	All historical application records and the corresponding status will be presented in a list.
Sign-in System		
1. Sign-in		
Users can fill-in the username and the password.	Input valid user-password pair. e.g. input admin@admin.com as username	The system will jump to the "manager interface" or the "employee interface" according

	1234 as password	to the account information.
If the password or username is not correct, a "wrong" notification will pop up	Input invalid user-password pair. e.g. Input admin@admin.com as username 123444 as password	The message "Username or password is wrong!" will pop up. And the input box will be cleared for user to enter the user-password again.
2. Sign-up (register)		
Users can register by setting a username (email address) and entering a password twice.	Click “forget the password” link. Filling in all blanks and click the "Sign-up" button.	The message "Registered Successfully!" will pop up and the system will jump to the login page.
Users can reset their password through email verification	Click “forget the password” link. Entering the email address and click "Verify" button.	An email containing randomly generated verification code will be sent to a specific mail address.
Information page		
1. For managers		
Managers will be shown a list of the employees and they can read employees' information forms by clicking the particular employee's icon.	Login the manager account, click the “Roaster” in side bar.	All employee information will be presented in a list.
Managers can update their own portfolios by clicking on the avatar in the upper right corner.	Login the manager account, click the avatar in the upper right corner.	Users can change their personal information. After clicking "Confirm" button, the message "Information successfully update!" will pop up.
2. For Employees		
Employees can update their own portfolios by clicking on the avatar in the upper right corner.	Login the employee account, click the avatar in the upper right corner.	Users can change their personal information. After clicking "Confirm" button, the message "Information successfully update!" will pop up.
User Experience Optimization		
1. Security		
When users enter passwords to	Enter passwords in the login	The passwords are displayed

log in, the passwords are displayed with multiple “*” instead.	page	with multiple “*”
When a user registers for an account, the password is stored in the database in the form of a cipher text.	Go into the "User" form in the database	The password is stored in the database in the form of a cipher text
2. Notification		
When there is new information (e.g., application progress update, etc.), an alert message will appear on the home page.	Go into the main interface (Pending page)	When there is new information (e.g., application progress update, etc.), an alert message will appear on the home page.