



SYSTEM DOCUMENT

THE HIBERNIA-SINO TRAVEL INSURANCE COMPANY
MANAGEMENT SYSTEM

POLICY



GROUP MEMBERS: DONG YUEHUI, ZHANG JINGYUAN, ZHANG LEI,
GROUP MEMBERS: Dong Yuehui, Zhang Jingyuan, Zhang Lei,
Zhang Tianhui, Zhao Wenqi, Zhen Ziyang

1. Abstract

1.1 Background

With the internationalization of society, people's expenditure on travel accounts for a large part of their total expenditure. So travel Insurance seems increasingly necessary.

We accepted the Commission of the travel insurance company to understand their needs, and finally completed the development of the policy purchase system.

1.2 About our system

Our software is a cloud-based insurance service system that is developed according to the actual requirements of the Hibernia-Sino Travel Insurance Company. It includes client-side and employee-side. Users can both operate from web pages and mobile devices, and employees of the company can use the same site to manage users' operations.

Users can upload their own information through our system application form, and communicate with the employee of the company.

Through this system, the company can formulate the form policy and form type introduction. The form information uploaded by users is revised and reviewed. And timely feedback to the user's consultation. It is worth mentioning that our system has strong extensibility and flexibility. Because employee can dynamically modify the text and information of the main page of the system at any time in the employee interface, at the same time, it is also conducive to the supplement of more functions and the testing and perfection of the system in the later period.

1.3 About us

There are six people in our group. We have formulated a series of intra-group systems, rationally planned the time, and held regular meetings once a week, even if we adjust the progress of work. We adopted the Vue framework, and the back end used the mainstream technology of spring boot to develop

the system for two months. Our overall development process is on Git.

1.4 Document writing purpose

The purpose of this document is to make a summary and review of the development work and learning in the past few months by describing in detail the technology used in the development of the system, the functions completed, the assignment of developers' work, the advantages and disadvantages of the system and the space for progress. In order to achieve the purpose of further learning.

2. Introduction

2.1 Task overview

2.1.1 Objective

This is an insurance company's product introduction and online trading software. Our system should meet the basic needs of employees when purchasing insurance products. For example, user registration, employee user login, user selection of products, delivery form, employee approval form and other basic functions.

2.1.2 Assumptions and Constraints

- We assume that each user has their own mailbox, so we use the mailbox to verify the user registration.
- We assume that the baggage insurance required for the product is divided into five categories: Baggage Insurance, Baggage Insurance Plus, OnTime Insurance, Safety Insurance and Child Insurance.
- We assume that each user may apply for multiple forms, and they can view the real-time progress of those forms at the same time.
- We assume that each purchased user may purchase for a form when they purchase it and may also purchase it for others.

2.2 Completed basic functions and extended functions:

Completed basic functions:

- Visitors can browse the interface, query form types and policy policies
- User login and employee login
- User registration
- Users and employees can see the insurance product introduction and related policies of the homepage.
- User submits insurance application form
- Users can check the progress of the form they submitted even
- Users and employees can modify personal information, such as uploading avatars, filling in phone mailboxes, and more.
- Employees can see detailed personal information of all registered users and details of the forms they apply for
- Employees can annotate and review the application form
- Employees can develop their own policy policies, and only those policy applications that have agreed to the policy will be granted approval.
- This system supports bilingual

Completed advance function:

- Users can send information to employees
- Employees can receive inquiries from users and even respond.
- Employees can dynamically modify the image and text information of the system homepage through the employee
- This system has its own domain name: <http://123.207.144.103/>
- The database of this system is uploaded to the cloud, which can accommodate more user information and update the information in time.
- User password encryption

2.3 unresolved issues

User's QQ login attempt has not been implemented

Cloud database needs to be optimized

If you give us more time, I believe that these functions will be perfected.

2.4 Application technology and operating environment

2.4.1 Server Appliance

The main frequency of CPU is more than 1GHz, the memory is more than 1 GB , and the free space of hard disk is more than 1GB.

2.4.2 Software Dependency

- Database Server: MySQL 5.5.57
- Application Server: Tomcat 8.5.12
- Web Server: Nginx 1.15.10
- PHP Version: PHP-5.4
- Key-value Database: Redis 5.0.3

2.5 test plan

2.5.1 Understanding Users, Usability Testing:

Through discussion and online search for information, our team fully understands the users' needs, and then it is information security and quick trial. So we streamlined the main interface information. Put the most valuable information in the most obvious position. It simplifies the necessary information at the time of registration and hopes to bring in more registrations.

2.5.2 Functional Test:

User interface testing: Each team member clicks on the interface multiple times to check if the response is normal.

Code review: We check the open interface (API) to see if it always returns the same result and check if the code provides the protection it deserves.

2.6 Division of labor

Responsible for front-end development: Zhao Wenqi, Dong Yuehui and Zhen Ziyang

Responsible for back-end development: Zhang Jingyuan, Zhang Tianhui, Zhang Lei.

A more detailed division of labor will be described in detail below.

3. Teamwork

3.1 Team division:

3.1.1 About coding:

Reasonable team division is the guarantee of team efficiency. We divide groups into front-end and back-end groups, depending on each individual's interests and capabilities. Ensure that each team has at least one strong code-capable person who can guide the team in the direction.

Front end: Zhao Wenqi, Dong Yuehui, Zhen Ziyang. Three people divide the basic work equally, in addition, Zhao Wenqi is responsible for and considering comprehensive, mainly responsible for the overall details of the revision and functional design. Dong Yuehui code writing and learning skills, additional responsibility for difficult problem solving and hardware, environment configuration. Zhen Ziyang is more patient and calm, extra responsible for cooperating with the work. Back-end three: Zhang Jingyuan, Zhang Lei, Zhang Tianhui. Zhang Jingyuan code ability and learning ability is excellent, responsible for holding the back-end code direction. Zhang Lei is more interested in cloud server-related content, responsible for cloud affairs. Zhang Tianhui can take advantage of the limited time but proactive, so she is responsible for cooperating with the work.

3.1.2 About other thing:

In addition to dividing the front-and back-end groups according to code capabilities, we still define each individual's position in other tasks in the group, depending on individual abilities. Among them, Zhao Wenqi and Zhen Ziyang are relatively good at English, responsible for the main work of the speech.

Zhang Jingyuan and Dong Yuehui professional ability, responsible for speech, documentation and other technical aspects of writing and control. Zhang Lei is responsible for his work and the wealthiest, taking on the team's all-round assistance and most of the expenses. Zhang Tianhui with excellent designing ability and aesthetic, responsible for the group's document layout and sample production and other work.

At the end of the work, with the full implementation of the basic functions of the project, we re-planned the work of everyone in the group. Dong Yuehui, Zhao Wenqi and Zhang Jingyuan are responsible for the problem handling and overall code testing in the front-end handover process, while the back-end Zhang Lei is responsible for cloud detection and the production and improvement of bilingual functions, and Zhen Ziyang is responsible for assisting the work. Zhang Tianhui is responsible for Android software production. Everyone performed their duties and helped each other, completing a great deal of extra work at the last tense moment.

In general, our team has a clear and flexible division of labor, with different and clear responsibilities assigned to each member for different tasks. And the blurring of the work is done with each other, and members never evade each other. Therefore, we believe that our team division of labor is very successful, giving full play to everyone's specialties, doing a great deal of extra work, and helping each other, United and harmonious with each other.

3.2 Problems and Solutions:

3.2.1.first problem and solution:

First of all, because of each person's different time planning, there is a situation where individuals are a little lazy. Some team members have high requirements for grades, so they do project promotion every day; some members do not need such high grades, so they spend most of their time taking the postgraduate entrance examination or traveling. Similarly, absolute equity is difficult to achieve, given the differences in each person's abilities. At the same time, students with strong product skills can do two to three times as much work as poor ability students. As a result, it is unfair to limit the same time or the same work, so we met to discuss ways to solve the problem.

Our solution is to first assign the same tasks to everyone, students who finish ahead of schedule appropriately help students with difficult code and appropriately take on more work according to their own circumstances, so, We also add the rules for collective code writing and independent code books

within the front and back teams (method 2 and method 3) to help each other improve the team's overall productivity through weekly collaborative code time. At the same time, the difficult part is assigned to the students with stronger ability to solve the task assignment problem and to maximize the benefit of the team by delivering the troublesome and simpler work to the weak students and assigning the difficult parts to the students with stronger ability. Moreover, several students who are more professional in the communication process are willing to take the initiative to take on more work and enjoy the code implementation process. Weak students also indicated that they are willing to spend more time, saving the best of strength to play a greater role in the team.

3.2.2 Second problem and solution:

In addition to solving the small problems caused by individual differences, we have also solved the problem of uncoordinated and out-of-sync in cooperation. Once, because we first developed a working method for the separation of the front and rear ends, and then, in the process of merging the front and back ends, there was a problem that the data did not correspond to each other on the handover.

We discussed it collectively and used our voting system(method 4). The rules of weekly communication between the front and back end and the rules that the front and back end students should cooperate to solve any problems arising in the handover are formulated. But in fact, because of group culture and previous exchanges, everyone in our group is proactive in solving problems together. So this temporary front-and-back problem was solved in less than ten minutes and never happened again.

Overall, as a result of mutual understanding and positive communication, our team has hardly encountered any difficult problems. Although there are some inevitable minor contradictions, but because of the direct and sincere communication of the team members, we are more United. Whenever there is a technical or speech assignment, everyone takes an active part in the discussion.

3.3 Methods of collaboration:

3.3.1 Team leader polling system

In a six-member group, we believe that everyone will be the team leader in turn to give everyone a strong sense of collective responsibility and collective honor. Each week, we serve as team leaders, in the order of WeChat groups, to convene meetings, monitor and verify the progress of the project, and write

weekly minutes of meetings. Through this method. Each of us can understand the hard work of the team leader, from urging others to taking the initiative to finish the weekly work ahead of schedule, and also learn all the relevant knowledge about our project, rather than just staring at a small piece of code that each one are responsible for.

3.3.2 The anterior and posterior ends were separated

We initially divided the six people equally into front-end groups and back-end groups, with the exception of group collective meetings, front-end and back-end as two relatively independent groups, separate in technical and task terms. As a result, we believe that the front-end and back-end as two independent groups to discuss their respective issues, can be more efficient, save time to take charge of other groups, and more targeted problem-solving.

3.3.3 Collective code writing and stand-alone code writing

Each group, after Monday's class, assigns its own internal responsibilities to each person and then writes independently. Then, meet each other for a certain period of time each week, and solve each other's unsolvable problems by helping each other. The meeting time is agreed on the basis of the specific circumstances of the respective groups.

3.3.4 Voting system

When differences arise, a group vote is conducted and the minority obeys the majority.

3.3.5 Conflict handling rules

When encounter group contradiction, should communicate each other's thoughts sincerely. Others have an obligation to adjust. If the problem is serious, should apply for external forces, such as application for assistant teacher intervention to help solve. We know that as long as there is real cooperation, there will be more or less problems, sincere and common goals are the fundamental solution to the problem. However, specific solutions must change according to circumstances, which requires us to always put the interests of the group at the top of the list.

3.3.6 Punishment system

If there is a system, there must be people who violate it. As a kind of deterrent effect, punishment system is essential. We have decided that those who violate the Panel's agreement will be deprived of their right to vote, subject only to the Panel's decision, until they do not violate the Treaty.

3.4 Inspiration:

After months of cooperation, we learned how to work together, how to behave, how to communicate, and how to deal with problems. First of all, from a personal point of view, we are aware of the need to be sincere and understanding. Talk or work, should let others feel their sincerity, not with personal temper, affect the development of the entire team. And learn to think from the perspective of others, understand other people's difficulties, and try not to create trouble for others. From the perspective of the team as a whole, early mutual understanding and rule-making are crucial. Only by getting to know everyone's abilities, specialties and characters as early as possible, can we let everyone exert their greatest abilities and reduce the occurrence of contradictions. On the premise of understanding each other as well as possible, making as perfect team rules as possible can greatly reduce the problem. The most important thing is to have the spirit of group (unity). No matter what happens, we face each other as a group. I think this unity is the fundamental reason why our team work so smoothly.

4. Technical Implementation

In the whole project, we adopted the development method of separating the front end and the back end, in this way, we can focus on the unilateral learning and development of front-end technology or back-end technology during the development process to improve efficiency. During the test the source of the BUG can also be found more quickly and sent to the front-end or back-end engineers for modification respectively. Focusing on the future, by separating the front end and the back end, the real front end and the back end can be decoupled to reduce the server load caused by the increase of traffic in the future. If the server goes down due to excessive business requirements, there will be no significant impact on the front-end pages. Meanwhile, due to asynchronous loading, no matter how complex the front-end pages are, the response speed of the server will not be affected. In this section, we are going to introduce Front End and Back End.

4.1 Front End:

We divided the front end into two parts, customs' interface and employees' interface. Both of them were built based on VUE framework. When we coding the most different between VUE and traditional HTML + JavaScript + CSS model is in VUE frameworks we put all these three parts into one file. This method is called Single file VUE componentized development model¹. The code in a VUE file look like the [Appendix\[1\] Figure\[1\]\(login.vue\)](#).

4.1.1 Responsive programming by using VUEx:

Responsive programming is one of the features of the VUE framework, there is a folder called store, a file called store.js can be found in this folder. This file is the key part to realize the responsive programming. The code in this file can be divided into three part: view(Mapping state to the view declarative), state(Data sources that drive applications), actions(Responds to a state change caused by user input on the view), it looks like the [Appendix\[1\] Figure\[2\] \(store.js\)](#). All state changes happened on store, will be managed on store's action part, this method called Centralized State Management. Because of using Centralized State Management, we can find which kind of mutation will happen and how. When an error happened, we will also have a Log of what happened lead to BUG.

4.1.2 Route Manager plug-in Components: VUE-Router

Our web page was designed to take into account the switching of different pages and the switching of sub-pages within the page. The VUE framework provides a good solution to this problem. The VUE-Router plug-in makes it easy to build a jump between pages. A file called router.js can be found in root directory of our front end folder , we show the code in this file in [Appendix\[2\] Figure\[3\]\(router.js\)](#). This file store all the link between pages in our website. If we want use the link, we just need to import the router in router.js like the code in Figure 4.1 below.

```
toMessages() {  
  console.log("to messages page");  
  this.$router.push({ path: "/page/messages" });  
},
```

Figure 4.1 The sample code of use router

¹ We will introduce it in 4.1.3 Component Development

Vue-Router also used in Detection platform switching different screen adaptation schemes, such as PC and mobile switching, by calling the “replace” method to replace the display of the page, you can automatically display different layouts². The following Figure 4.2(login.vue 247-249) is an example of using the replace method.

```
if (this.$store.getters._isMobile) {  
  this.$router.replace("/mlogin");  
}
```

Figure 4.2 The using of replace method

4.1.3 Component Development by using iView

Another features of VUE framework is component development. Its implementation is mainly embodied in the use of iView component library. The advantages of iView component library are:

- 1.high quality, rich function, delicate and beautiful UI.
- 2.friendly API which makes components insert on web more easily.
- 3.An official document of every detail.
- 4.Single file VUE componentized development model
- 5.Developed base on npm + webpack + babel, Compatible ES2015³.

Let’s talk about the advantage No.4: Single file VUE componentized development model. Look at [Appendix\[1\] Figure\[1\]\(same as 4.1\)](#), we can find out there are three labels: template, JavaScript, style. In a single VUE file, template is responsible for templates, JavaScript is responsible for logic, and style is responsible for styles. The idea behind this is that a single file component corresponds to a functional component, and the template, style, and business logic for that component all adopt the idea of nearby maintenance. From the perspective of component reusability and late maintainability, this concept greatly improves the development efficiency of componentization. VUE single files are neither JavaScript, HTML, nor CSS files. In our development process, we think of a page as a large component composed of multiple components.

Next, we’ll show you how these components collaborate to accomplish some representative system functions⁴.

² On the mobile side of the adaptation scheme please see 4.9

³ Also called ECMAScript 6 which is a standard of JavaScript released on 2015.

⁴ Features page implementations that are not mentioned are much the same.

4.1.4 Login and Registration page

We think of the entire login page as a large component, with a smaller component in the center that provides a location where the company logo is displayed to the customer, as well as an input box for the customer to enter the user name and password. Below is the login and registration button, which triggers an event that interacts with the back end and receives the return value from the back end. In this way, the user name and password confirmation and user rights distinction. The implementation code for this section is as [Appendix\[2\] Figure\[4\]\(login.vue input part 82-107\)](#). The company logo is showed by `` label. Text is displayed in this format: `{{ $t("message.Welcome") }}`⁵, the input box is realized by `<Input/>` label, `<Button/>` label has supported the “Login” and “Register” button. All primary style was written on style part of the VUE file, only especial style was controlled by label itself.

A click event called `modal1` is used on the registration button, and by triggering the event we can provide the user with an interface to fill in the registration information. An push method will be called by `router`⁶ to find the correct registration page and show it to user. On the registration page, we also make use of the same components to provide input boxes and text compared to Login page, but the difference is that we add some function to make sure the user will enter the data that we want. As the user enters the registration information, a method named “`riRule`” is called. This method is mainly used to detect whether the necessary information is correctly filled in, for example: is the password entered? Is the password input the same two times before and after? E-mail address format is correct, etc. If the information entered does not meet the requirements, the appropriate prompt is returned, the sample implementation is shown in [Appendix\[3\] Figure\[5\]\(Login.vue 216-243\)](#).

4.1.5 Home page

In our design, the home page is divided into two components, the top navigation bar and the home page content. The top navigation bar is controlled by the “`index.vue`” file, and the home page content is controlled by the “`home.vue`” file.

Let's start with the top navigation bar, which is implemented using the `<menu>` label, and each of the navigation bar items is treated as a `<menuitem/>` and then numbered separately to identify click listening events. See [Appendix\[4\] Figure\[7\]\(index.vue 89-166\)](#) for specific implementation. We also use a

⁵ This will be introduced in 4.7 vue-i18n internationalization (different languages). Same as below

⁶ For more information, see 4.1.2 Route Manager plug-in components: VUE-Router. Same as below.

conditional rendering method called “v-if”. The feature of this method is that when the value returned by “v-if” is true, the component will be displayed, and otherwise the component will be hidden. In the implementation of the navigation bar, we set a Boolean variable named “loginFlage” with an initial value of false. By placing the classification of the components in the navigation bar in two different <div/> labels with different “v-if” return values, we have implemented the option display and hiding of the navigation bar in the unlogged-in and logged-in states, such as the login button and the log-out button. We put the login button in the <div/> labels where the “v-if” return value is not equal to the “loginFlage”, so that the login button is displayed when it is not logged in, and the log-out is reversed. The switch language button, which is used in both cases, placed under both <div/> labels. The following Figure 4.3(index.vue 158-164) shows the navigation bar component detail code that is displayed

```
<div v-if="!loginFlag" class="layout-nav">
  <i-button @click="changeLocale()">{{ $t('message.ChangeLanguage') }}</i-button>
  <MenuItem name="6">
    <Icon type="md-log-in"></Icon>
    {{ $t('message.login')}}
  </MenuItem>
</div>
```

Figure 4.3 Home page navigation bar component detail code

in the unlogged-in state. The only different between logged-in state and unlogged-in state is the “v-if” is equal to “loginFlage” or not. Considering that users need to input large text description when using the system, we introduce a “interest-quill-editor”⁷ rich text editor based on VUE framework. It supports title, paragraph, reference, superscript, font size and other commonly used article editing patterns, perfect to meet the user needs of our vision. Detailed implementation is shown in the Figure 4.4(create-article.vue 22).

```
<interest-quill-editor class="editor" v-bind:interestContent="interestContent" @editor-change="e=>{contentGet(e)}"></interest-quill-editor>
```

Figure 4.4 interest-quill-editor implementation

Next, we'll introduce some of the technique that are used only in the content section of the home page. On the front page of our design, there is a component that shows the picture, which switches the content displayed in it on time to achieve our goal of dynamically presenting our product category to our customers. This technology, called Walking Lamp, is implemented through <Carousel/> label. The “autoplay” property in the tag causes the content displayed in the component to switch automatically, while the loop property causes the stored content in the component to start over automatically after

⁷ The original version is called vue-quill-editor. We're referring to a better rewrite.

broadcast. Detailed implementation is shown in the Figure 4.5(home.vue 4-10)

```
<Carousel autoplay v-model="value2" loop>
  <CarouselItem v-for="(item,index) in bannerList" :key="index">
    <router-link :to="'/page/detail/'+item.id">
      
    </router-link>
  </CarouselItem>
</Carousel>
```

Figure 4.5 Detailed implementation of Carousel

4.1.6 Console

The supply technology used in the process of implementing the employee console is almost the same as the front page. The input box is provided by input, and the side navigation bar is designed to provide rich text editing by menu,interest-quill-editor. In order to manage the user's questions and applications, the employee side adds the form component to realize the user's question list and so on. This function is realized by calling the form tag. Considering the excessive number of entries in the table, 10 pieces of data can be displayed on a single page by setting the “pageSize” variable. At the same time, a page flipping button is added. The concrete implementation is shown in the [Appendix\[3\] Figure\[6\]\(form.vue 29-65\)](#)and [Appendix\[N\] Figure\[N\]\(form.vue 232-235\)](#).

4.2 Front-end data interaction

Because our project adopts the design idea of front and rear end separation, the technology of front and rear end data interaction is involved in many business logic implementations.

4.2.1 Main application scenarios should be

1. User information is requested from the back end when the user logs in
2. When applying for policy compensation, the form information filled by the user is sent to the back end
3. When a user views his or her own policy application list, the user application data is retrieved from the back end
4. When the employee views the list of customer application forms, it gets the application data of all users from the back end
5. The data is sent to the back end when the user consults the problem

6. When employees reply to customer questions, they get the customer inquiry list from the back end

The above is only the main application scenario, and more front-end and front-end data interaction scenarios are involved in the actual application

4.2.2 Concrete examples

Next, I will take the user's application for policy compensation as an example (main application scenario 2) to describe the implementation of front and rear end interaction in detail.

- When the user fills in the application form, click "ok", the content filled by the user will be first encapsulated in the formdata class, and the variable values in the class will be formatted and validated with the `$refs[].validate()` method. If the validation fails, the error message will be returned to the user and the user will fill in again. You can see the data structure for formdata class in the appendix[5].
- After the validation passes, the data is sent to the back end using `axios()`. Because the form contains user sensitive information, the **post** method is used here, and a **url(/form)** is specified, the formdata class is sent to the back end, and axios automatically encapsulates the data as a **json** type.

```
Enclosing axios({  
  Method: "post",  
  Url: "/form",  
  ...  
})
```

- In addition, the agent needs to be set in the configuration file of vue. Config. js to achieve cross-domain access. Thus, the front-end task is completed.

```
Proxy: {  
  "/interest": {  
    Target: "http://localhost:8080",  
    Secure: false  
  }  
}
```

- The back end receives the data sent by the front end using the `@PostMapping("/form")` annotation in the FormController class and implements the business logic processing in its annotated methods.

4.3 Back-end business logic processing

After realizing the data interaction between the front end and the back end, it is necessary to make a correct response to the data from the front end. The realization is mainly in the service layer, including the judgment and processing of business logic, and the standardized processing of business results will be returned. Then we are going to talk about the primary business logic implementation

Most of the business logic is received by the controller class data and requests, and through the implementation of service interface serviceImpl class to deal with the specific business logic, and finally through the Dao layer to achieve the interaction with the database. The following examples illustrate specific implementations of several major business logic

CONTROLLER - > SERVICE - > SERVICEIMPL > DAO - > SQL

4.3.1 User registration

- In the UserController class, first @PostMapping () receives the url for the newly created user information ("/roles/ roles ") and extracts the user information from the RoleEntity class with the @requestbody annotation, then stores the user data using the insertRole() method in the RoleService interface

```
@PostMapping("/roles/role")
    public ResponseWrapper<RoleEntity> insertRole(@requestbody RoleEntity roleEntity){
        RoleService.insertRole(roleEntity);
        log.debug("The method is ending");
        return new ResponseWrapper<>(roleEntity);
    }
```

- The RoleServiceImpl class will implement the RoleService interface and implement the insertUser() method to store the data, where we use Bcrypt to encrypt the user password to protect the user's information security

```
@Override
    public void insertUser(UserEntity userEntity) {
        UserEntity.setPassword(
            "{bcrypt}" + new BCryptPasswordEncoder().encode(userEntity.getPassword()));
        UserDao.insertUser(userEntity);
    }
```

4.3.2 Displays the list of forms requested by the logged-in user

- In the FormController class, get the number of pageSize and the current page number of the form displayed per page from the data transmitted from the front end

- Get the ID of the current user through the getId() method in the SecurityAuthenUtil class to retrieve all forms for that user's application.
- Create a new entity of the PageResult class to encapsulate the list of forms to be displayed and the number of all forms to be displayed on the current page returned from the database, and get the values of these two variables through the formList and formsSize in the FormService interface.

```
@ApiModelProperty(" the paging data ")
Private List<?> data;
@ApiModelProperty(" total paging data ")
Private Integer totalCount;
```

- The resulting PageResult entity is returned

```
return new ResponseWrapper<>(pageResult);
```

4. 4 Database layer interaction

In order to separate the business logic from the database access logic, we use the persistence layer framework MyBatis, which enables the Dao layer to realize the interaction function with the database in addition to the function of adding, deleting, modifying and checking data. You can see the ER diagram of the database in the appendix[5].

Following is the specific implementation

- First, all Dao layer classes are annotated with the @mapper annotation, which turns the original class into an interface. And @mapper generates implementation classes for these interfaces that are referenced by other classes.
- In these Dao interfaces, we add the required methods to add, delete, modify and check the data
- Later, we will correspond to a **Mapper. XML file for each Dao interface, in which only SQL statements return data requested by each method in Dao interface.

I'll pick one of several implementations in the back-end business logic processing for illustration below

Then, we are going to displays the list of forms requested by the logged-in user

- In the Regis_FormDao interface, there is a formList and a formsSize method, which pass the required data from the SQL statement to the Regis_FormMapper. XML file through @param

```
List<Regis_FormEntity>formList(
    @param("pageSize")int pageSize, @param("start")int start, @param("ID")int ID);
Integer formsSize(@param("pageSize")int pageSize, @param("start")int start);
```

- In the Regis_FormMapper. XML file, the method in the Regis_FormDao interface is corresponding to the id in the select component,resultMap is used to define the type of returned data, and SQL statement is implemented in this component.

```
<select id = "formList" resultMap = "formModelMap"></select>
```

4.5 User password encryption

To protect the user's privacy, we encrypt the user's password through Bcrypt before storing it to the database

```
UserEntity.setPassword(" {bcrypt} "+
    new BCryptPasswordEncoder().encode(userEntity.getPassword()));
```

4.6 The user authorization

- Our application is divided into client side and employee side, so we need to configure reasonable permissions for each role, and we adopt the Spring security framework. Through inheritance ResourceServerConfigurerAdapter class and overriding its methods configure () for each role assigned permissions.
- For example, clients can access all functions whose urls are public/**

```
HTTP.authorizeRequests().antMatchers("/public/**").permitAll();
```

- The employee can access not only the url that the employee can access, but also the url requests such as /admin/**

```
HTTP.authorizeRequests().antMatchers("/roles/**","/admin/**").hasRole("admin");
```

4.7 Internationalization of the project

In order to display the corresponding interface in different languages and regions without modifying the internal code, we developed an international version of the project and used the I18N plugin, which is a plugin based on Vue.

The process of nationalization is divided into several steps :

To begin with, Install I18N plugin for project. The Internationalization version we used is 8.10.0

```
"vue-i18n": "^8.10.0",
```

After importing dependencies for front-end, a specific example should be declared in project.

Vue provide a convenient way to realize international, this mechanism is called locale. We initialize locale as English

```
const i18n = new VueI18n({
  locale : "en",
})
```

Then, a lot of translated information needs to be prepared locally. Considering that the users of this project are native speakers of English and Chinese, project release version includes both English and Chinese. Consequently, two types of translation documents should be prepared.

```
const messages = {
  zh: {message: {homepage: '主页', ... }}
  en: {message: {homepage: 'Home', ... }}
}
```

Finally, use different text rendering techniques to replace all text in the project

For placeholder:	:placeholder="\$t('message. placeholder')"
For normal text:	{{ \$t("message.text") }}
For text calls:	this.\$t("message. calls ")

Buttons are placed prominently in Web servers and Android applications. Simple clicks will switch the language of the service

4.8 Cloud server

Cloud-based projects meet the needs of employees and customers to process transactions online. This project uses Tencent Cloud Server and provides stable services.

Essential information about Cloud Server:

Inside IP address	Outside IP address	Domain Name
172.21.0.9	123.207.144.103	www.hibernia-sino.cn

Cloud servers provide end-to-end services. Different protocols occupy different ports to realize data transmission and file request services together.

Order number	Port	Description
1	80	Website Default Port
2	888	PhpMyAdmin

3	8888	Visualization panel
4	20	FTP Active Mode Data Port
5	21	FTP Protocol Default Port
6	22	SSh Remote Service
7	6379	Redis
8	3306	MySQL
9	8055	Front-end and back-end interaction ports
10	8088	Development mode port

It is worth mentioning that except to input the corresponding ports, database access requires additional address and command. This insignificant operation can greatly increase the security of the project.

Due to the project adopts a separate front-end and back-end development strategy, there are different ways to publish different front-end and back-end files. All front-end files contained in the dist file need to be placed in a path named `/www/wwroot/hibernia-sino.cn`.

As for the back-end, a JAR file will be generated and located in a path named `/www/server`. In addition, an additive command is necessary for cloud server to continue running jar file.

```
nohup java -jar interest-server-2.0.0.2.jar &
```

4.9 Android

Android WebView is a WEBKIT-based View, which control for displaying web pages on the Android platform. This class can be used to display a specified web page online in the application. The internal implementation of WebView is to use the rendering engine to display the contents of the view. It provides the functionalities like: page forward and backward, page zoom in, zoom out, search... WebView is much more flexible for the web-based system. There is no need to develop or update the Android client, just to modify the web code.

4.9.1 onCreate

- finds the control, sets the loading *url* (*www.hibernia-sino.cn*) for the webView
- adds *js* listening
- sets the callback of webChromeClient and webViewClient
- obtains the WebSettings object through getSettings method, sets the allowable loading of *js*, sets the caching mode, and supports zooming.
-

4.9.2 onKeyDown

If you click the system's own **return** key, call `webView.goBack()` to return. Otherwise you will exit the app without rewriting.

4.9.3 Cache

Cache is stored in /database and /cache folder under /data/data/ directory. We save the requested *url* records in **WebViewCache.db**, and save the contents of the *url* in the **WebViewCache folder** to speed up the loading of the webView.

REFERENCE

APPENDIX[1]

```

1  <style scoped>
2  .index {...}
16 .index .ivu-row-flex {...}
19 #index_pc_bj {...}
27 /*具体内容*/
28 .wrap_conter ul {...}
35 .wrap_conter li {...}
43 .content {...}
51 .pc-high {...}
55 .wrap_conter li dl {...}
59 .name-password-error {...}
65 .ivu-form-item-content .account-list {...}
69 .account-list li {...}
73
74 .account-list .icon {...}
78 </style>
79 <template>
80 <div class="index">
145 </template>
146 <script>
147 export default {
148
149   data() {...},
249   mounted() {...},
254   methods: {
255     dateGet(e) {...},
269
270
271     okEnroll(enroll) {...},
295     cancel () {...},
298     login(formLogin) {...},
309     register() {...}
312   }
313 };
314 </script>

```

Figure [1] Example code of vue file: style, template and script label.

```

Vue.use(Vuex);
export default new Vuex.Store({
  modules: {
    users: UsersModule
  },
  state: {
    todos: [
      { id: 1, text: "...", done: true },
      { id: 2, text: "...", done: false }
    ],
    /*域名*/
    // 测试
    domainName: "http://127.0.0.1:8088",
    userUrlPre: "http://127.0.0.1:8088/page/user/",
    count: 0,
    /*登录界面判断是否显示错误提示*/
    ifSign: false
  },
  getters: {
    doneTodos: state => {
      return state;
    },
    doneTodosCount: (state, getters) => {
      return getters;
    },
    getTodoById: state => id => {
      return state.todos.find(todo => todo.id === id);
    },
    getSign: state => {
      return state.ifSign;
    },
    _isMobile() {
      let flag = navigator.userAgent.match(
        /(phone|pad|pod|iPhone|iPod|ios|iPad|Android|Mobile|BlackBerry|IEMobile|MQQBrowser|JUC|Fennec|wOSBrowser|BrowserNG|WebOS|Symbian|Windows Phone)/i
      );
      return flag;
    }
  },
  mutations: {
    add(state) {
      state.count++;
    },
    increment(state, number) {
      state.count += number;
    },
    setSignTrue(state) {
      state.ifSign = true;
    },
    setSignFalse(state) {
      state.ifSign = false;
    }
  },
  actions: {
    increment(context) {
      context.commit("add");
    }
  }
});

```

Figure [2] Example code of store file: view, state and action label.

APPENDIX[2]

```
const routers = [
  {
    path: "/image-capture",
    component: resolve => require(["./views/template/image-capture.vue"], resolve)
  },
  {
    path: "*",
    component: resolve => require(["./views/error404.vue"], resolve)
  },
  {
    path: "/",
    meta: {
      title: "Hibernia-Sino Travel Insurance"
    },
    component: resolve => require(["./views/template/index.vue"], resolve),
    children: [
      {
        path: "",
        name: "home",
        component: resolve => require(["./views/template/home.vue"], resolve),
        meta: {
          title: "home"
        }
      }
    ]
  }
],
```

Figure [3] Example code of router.js file: routers and links.

```
<Form ref="formLogin" :model="formLogin" :rules="ruleLogin">
  <div class="wrap_conter">
    <ul style="list-style: none; box-shadow: 10px 10px 20px rgba(0,0,0,0.5);">
      <li style="border-bottom: 1px solid #e9eae2;">
        <div class="content">
          
          <span style="float: right; font-size: 22px; align-items: center;">{{ $t("message.Welcome") }}</span>
        </div>
      </li>
      <li>
        <div class="name-password-error" v-if="this.$store.state.ifSign">{{ $t("message.WorryUsernameORPassword") }}</div>
        <dl>
          <FormItem prop="userName">
            <Input v-model="formLogin.userName" type="text" :placeholder="$t('message.loginuser')">
              <Icon type="ios-person-outline" slot="prepend"></Icon>
            </Input>
          </FormItem>
          <FormItem prop="password">
            <Input v-model="formLogin.password" type="password" :placeholder="$t('message.Password')">
              <Icon type="ios-lock-outline" slot="prepend"></Icon>
            </Input>
          </FormItem>
          <FormItem>
            <Button type="primary" @click="login(formLogin)" style="width: 250px">{{ $t("message.login") }}</Button>
            <Button type="primary" @click="modal1 = true" style="width: 250px">{{ $t("message.register") }}</Button>
          </FormItem>
        </dl>
      </li>
    </ul>
  </div>
</Form>
```

Figure [4] Example code of login.vue file: <input/> label used to let user enter their information.

APPENDIX[3]

```
riRule: {
  name: [
    {
      type: "string",
      required: true,
      message: this.$t("message.PleaseEnterName"),
      trigger: "blur"
    }
  ],
  phone: [
    {
      type: "string",
      required: true,
      message: this.$t("message.EnterTelNumber"),
      trigger: "blur"
    }
  ],
  email: [
    { required: true, message: this.$t("message.EnterEmailAddress"), trigger: "blur" },
    { type: "email", message: this.$t("message.PleaseEnterRightEmailFormat"), trigger: "blur" }
  ],
  userName: [
    { required: true, message: this.$t("message.PleaseEnterName"), trigger: "blur" }
  ],
  password: [{ required: true, message: this.$t("message.PleaseEnterPassword"), validator: validatePass, trigger: "blur" }],
  password2: [{ required: true, validator: validatePass2, trigger: "blur" }]
}
```

Figure [5] Example code of login.vue: riRule method used to help user enter their information correctly.

```
<Form ref="email" :rules="emailRule" :model="email" :label-width="80">
  <Form-item label="表单ID号:" prop="id">
    <strong>{{email.id}}</strong>
  </Form-item>

  <Form-item label="用户姓名:" prop="name">
    <strong>{{email.name}}</strong>
  </Form-item>
  <Form-item label="用户ID号:" prop="userid">
    <strong>{{email.userid}}</strong>
  </Form-item>
  <Form-item label="用户邮箱:" prop="email">
    <strong>{{email.email}}</strong>
  </Form-item>

  <Form-item label="表单类型:" prop="formType">
    <strong>{{email.formType}}</strong>
  </Form-item>
  <Form-item label="用户标题:" prop="title">
    <strong>{{email.title}}</strong>
  </Form-item>
  <Form-item label="内容说明:" prop="content">
    <span>{{email.content}}</span>
  </Form-item>
  <Form-item label="证明图片:" prop="image">
    <span></span>
  </Form-item>
  <Form-item label="员工备注:" prop="remark">
    <Input v-model="email.remark" type="textarea" :autosize="{minRows: 2,maxRows: 5}" placeholder="Enter remark..." />
  </Form-item>
</Form>

initPageInfo() {
  this.pageInfo.page = 0;
  this.pageInfo.pageSize = 10;
},
```

Figure [6] Example code of form.vue: <form/>label support the form in console, PageInfo control the number of data showed in form on one page

APPENDIX[4]

```

<Menu mode="horizontal" theme="light" class="menu-layout" active-name="interest"
  (@on-select="m=>{menuSelect(m)}")>
  <div style="width: 95%;margin: 0 auto">
    <div class="layout-logo">
      <a @click="backHome()">
        
      </a>
    </div>


    <div class="layout-title">
      <MenuItem name="interest">
        <span class="home-text">
          {{ $t("message.homepage") }}
        </span>
      </MenuItem>
    </div>
    <div v-if="loginFlag" class="layout-title">
      <MenuItem name="article">
        <span class="home-text">
          {{ $t("message.list") }}
        </span>
      </MenuItem>
    </div>
    <div v-if="loginFlag" class="layout-nav">
      <i-button @click="changeLocale()">{{ $t("message.ChangeLanguage") }}</i-button>
      <MenuItem name="1">
        {{user.name}}
      </MenuItem>
      <MenuItem name="2">
        <Icon type="ios-mail"></Icon>
        {{ $t("message.SubmitApplication") }}
      </MenuItem>
      <Submenu name="3">
        <template slot="title">
          <Icon type="md-list-box" />
          {{ $t("message.QuestionConsultation") }}
        </template>
        <MenuItem name="31">
          <Icon type="md-create" />
          {{ $t("message.fillForm") }}
        </MenuItem>
        <MenuItem name="32">
          <Icon type="md-list" />
          {{ $t("message.myConsultation") }}
        </MenuItem>
      </Submenu>
      <MenuItem name="4">
        <Icon type="md-log-out"></Icon>
        {{ $t("message.exit") }}
      </MenuItem>
      <MenuItem name="5" v-if="consoleFlag">
        <Icon type="md-settings"></Icon>
        {{ $t("message.employeeConsole") }}
      </MenuItem>
    </div>
    <div type="success" class="avatar-badgе-wrapper" @click="toMessages">
      
      <span v-if="unreadMsgCount > 0" class="msg-num">{{unreadMsgCount}}</span>
    </div>


    <div v-if="!loginFlag" class="layout-nav">
      <MenuItem name="6">
        <Icon type="md-log-in"></Icon>
        {{ $t("message.login") }}
      </MenuItem>
      <i-button @click="changeLocale()">{{ $t("message.ChangeLanguage") }}</i-button>
    </div>
  </div>
</Menu>


```

Figure [7] Example code of index.vue: <menu/> label realized the navigate bar of home page.



APPENDIX[5]





article	
	id: int(11)
	title: varchar(255)
	info: text
	content: longtext
	click_rate: int(11)
	comment_count: int(11)
	top: int(1)
	create_time: varchar(255)
	reply_time: varchar(255)
	userid: int(11)
	del: tinyint(1)

oauth_client_details	
	client_id: varchar(255)
	resource_ids: varchar(255)
	client_secret: varchar(255)
	scope: varchar(255)
	authorized_grant_types: varchar(255)
	web_server_redirect_uri: varchar(255)
	authorities: varchar(255)
	access_token_validity: int(11)
	refresh_token_validity: int(11)
	additional_information: varchar(255)
	autoapprove: varchar(255)

interest	
	id: int(11)
	title: varchar(50)
	info: text
	content: text
	image: varchar(255)
	sort: int(11)
	banner: int(1)

reply_card	
	id: int(11)
	content: text
	postcardid: int(11)
	createtime: varchar(255)
	userid: int(11)


regis_form	
	id: int(11)
	title: varchar(100)
	email: varchar(255)
	name: varchar(50)
	content: text
	createtime: varchar(255)
	userid: varchar(20)
	formType: varchar(50)
	image: varchar(255)
	label: varchar(50)
	remark: text


sys_user	
	id: int(11)
	name: varchar(50)
	login_name: varchar(50)
	password: varchar(255)
	email: varchar(255)
	usertype: int(1)
	heading: varchar(255)
	url: varchar(255)
	create_time: varchar(255)
	githubid: varchar(255)
	qqid: varchar(255)


article_comment	
	id: int(11)
	articleid: int(11)
	userid: int(11)
	parentid: int(11)
	comment: text
	replier_id: int(11)
	replier_name: varchar(255)
	create_time: varchar(255)

sys_user_detail	
	id: int(11)
	userid: int(11)
	article_sign: int(1)
	info: varchar(255)
	location: varchar(255)
	skill: varchar(255)

msg_records	
	id: int(11)
	ownerid: int(11)
	form: int(1)
	reply_card_id: int(10)
	comment_id: int(11)
	replytime: varchar(255)
	isread: int(1)

myusers	
	ID: varchar(255)
	username: varchar(30)
	password: varchar(100)
	email: varchar(100)
	name: varchar(30)
	phone: decimal(20, 0)
	createTime: varchar(255)

post_card	
	id: int(11)
	title: varchar(100)
	content: text
	interestid: int(11)
	createtime: varchar(255)
	replytime: varchar(255)
	userid: int(11)

sys_menu	
	id: int(11)
	name: varchar(50)
	url: varchar(255)
	parent_id: int(11)
	sort: tinyint(4)
	remark: text
	icon: varchar(30)

sys_role	
	id: int(11)
	role: varchar(50)
	name: varchar(50)
	modules: text
	describe: text

r_user_role	
	user_id: int(11)
	role_id: int(11)