Table of Contents

I. Introduction 1
I.A. Problem Statement 1
I.B. Aims of The Project 2
I.C. Project Scope 2
I.D. User Characteristics 2
I.E. Assumptions and Dependencies 2
I.F. Project Risks 2
II. Architectural Design 2
II.A. System Architecture 2
II.B. Software Modules 4
II.C. High-Level Database Design 5
III. System Design 6
III.A. High-Level Design 6
III.B. Software Support Services 9
III.C. Coding Structure and Convention 10
III.D. Software Configuration and Production Environment 11
IV. Software Testing 11
IV.A. Unit Testing 11
IV.B. Integration Testing 12
IV.C. Acceptance Testing 13
References 13

I. Introduction

A. Problem Statement

In recent years, the vehicle rental industry has experienced significant growth, driven by the increasing demand for convenient and flexible transportation options. As this industry expands, rental platforms continue to face numerous challenges in providing efficient and user-friendly services to their customers. This problem statement will discuss the current challenges faced by rental platforms and the limitations of existing solutions in the market.

Current challenges faced by platforms in the vehicle rental industry:

1. Inefficient search and selection process: Many existing platforms lack comprehensive and easily navigable vehicle catalogues. This limitation makes it difficult for users to find and compare suitable vehicles, hampering them select the best vehicle for their needs.
2. Complex payment processing: Users often encounter difficulties understanding rental charges and processing payments on existing platforms. The lack of user-friendly interfaces in payment systems creates a barrier for users to complete their transactions.
3. Inconsistent user experiences: The quality and features of existing vehicle rental platforms can vary greatly, resulting in inconsistent user experiences. Users may find some platforms easy to use, while others may be cumbersome or lack essential features.

Limitations of existing solutions in the market:

1. Insufficient focus on user experience: Many existing platforms prioritize rental service providers' needs over users, resulting in a lack of user-centric design and functionality.
2. Inadequate integration with third-party services: Many existing solutions lack seamless integration with essential third-party services,

such as payment processing systems. This lack of integration hampers the usability of the platforms.

To address these challenges and limitations, this project proposes the development of a comprehensive Vehicle Rental Service Information System that streamlines the rental process and enhances user experience.

B. *Aims of The Project*

The primary goals of this project are to:

1. Develop a web-based rental system that streamlines and simplifies the vehicle rental process for users.
2. Design a system that offers efficient vehicle cataloguing, appointment management, and payment processing.
3. Enhance user experience and satisfaction by providing a user-friendly platform for searching, renting, and returning vehicles.

C. *Project Scope*

The scope of the project includes the following:

1. A comprehensive and easily navigable vehicle catalogue, organized by factors such as vehicle brand, type, and price range.
2. User registration and profile management, allowing users to input their information, including driving license details and contact information.
3. Appointment scheduling and management for vehicle retrieval and return.
4. Payment processing and calculation of rental charges.
5. Distinct functionalities for different user roles, such as administrators and customers.

D. *User Characteristics*

The system will cater to two main user types: administrators and customers. Administrators will be responsible for managing system master files, viewing appointments and orders, and accessing statistical reports related to the business. Customers will use the system to register, edit user information, rent and return vehicles, and manage appointments.

E. *Assumptions and Dependencies*

Key assumptions related to the project include the availability of reliable internet connections for users, the adaptability of users to the new system, and the system's ability to scale with the rental market's growth.

Dependencies on third-party services and software libraries will be identified and managed during development.

- **Example 1**:
  Alibaba Cloud API (Sandbox Payment): The Vehicle Rental Service Information System will rely on Alibaba Cloud API's Sandbox Payment feature for secure payment processing during the development phase. The Sandbox Payment environment allows developers to test and validate payment transactions without affecting real customer accounts or data.
- **Example 2**:
  JaCoCo will help the development team to measure code coverage during testing, allowing them to identify areas where additional tests are needed and ensuring a more robust and reliable system.

F. *Project Risks*

Potential risks associated with this project include unforeseen technical challenges and delays in the project timeline. These risks will be closely monitored through effective project management strategies and proactive risk management.

II. Architectural Design

A. *System Architecture*

The system architecture is shown in the figure below, we divide the whole car rental system into 6 components, and each component is further divided into sub-components. Interactions between components are also marked with arrows. The six components are User Management Component, Vehicle

Management Component, Positive Feedback and User Feedback Component, Appointment Component, Payment Component, and Business Reporting and Statistics Component. The User Management Component handles user-related events, such as registering as a user of the car rental system, logging in as a user, and modifying the user's personal information. The Vehicle Management Component handles vehicle-related events like querying, adding, and changing vehicle information. The Positive Feedback and User Feedback Component handle events such as browsing and deleting the customers' comments. The Appointment Component mainly handles events related to car reservations, such as creating, cancelling, and viewing orders. The Payment Component mainly handles payment-related events, such as paying for a car rental, refunds, and viewing payment information (e.g. bank card information). The Business Reporting and Statistics Component handles events such as browsing statistical reports and charts.

A general example will be given next to show the interactions between the components. When a user enters the rental car system for the first time, he or she will register as a new user, which will add new user information in the User Management Component. The user will use the information filled in during registration to log in and access the subsequent pages of the car rental system. In the next step, the user will get the information related to the vehicle in the Vehicle Management Component, and it will display the car picture so that the user can get the vehicle information more directly. After selecting the desired vehicle, the user will enter the Appointment Component to book the vehicle. After the reservation is successful, the user enters the Payment module to pay the rental fee. After the successful execution of the payment, the order will be displayed in the Appointment Component. After completing the rental, users can evaluate the rental experience through the comment function provided by the Positive Feedback and User Feedback Component.

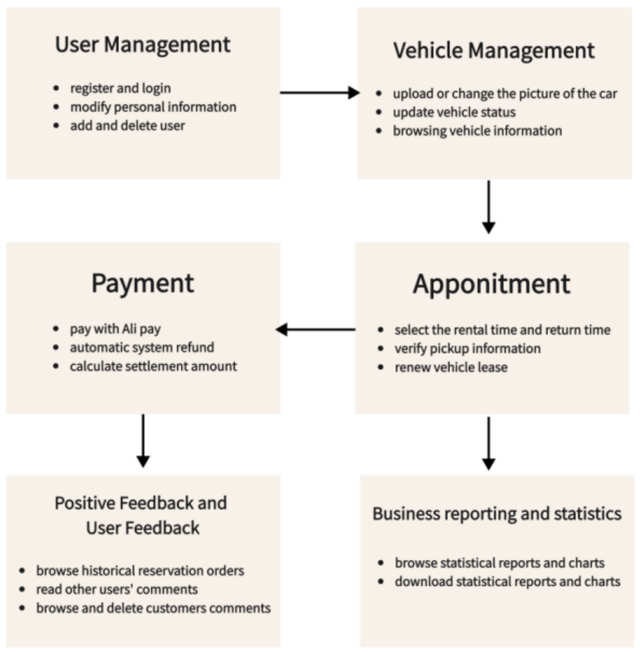

User Management
register and login
modify personal information
add and delete user
Vehicle Management
upload or change the picture of the car
update vehicle status
browsing vehicle information
Payment
pay with Ali pay
automatic system refund
calculate settlement amount
Apponitment
select the rental time and return time
verify pickup information
renew vehicle lease
Positive Feedback and User Feedback
browse historical reservation orders
read other users' comments
browse and delete customers comments
Business reporting and statistics
browse statistical reports and charts
download statistical reports and charts

Figure 1: System Architecture

In this project, we followed the MVC design pattern [1] to keep the codebase organized by separating the different building blocks into different modules.

The first is the Controller component. The user accesses the "ZRCR" car rental website through a browser and performs some actions, such as registering an account and processing order information. These requests are sent to the server. For example, when the user clicks the "add user" button on the "User" page and fills in the information to submit, a POST request is sent to the appropriate URL, each URL corresponds to an interface, and the Controller resolves this HTTP request and maps the request to a specific method.

Next is the Model component, which is divided into an Entity, a DAO layer (Data Access Object), and a Service layer; the Controller injects the data into the corresponding entity. For example, the data obtained from adding a user is injected into the User entity, while the data concerning orders is injected into the Order entity. For different request types, the Controller calls the appropriate Service to provide the service and transfer the Entity objects (e.g. User, Order, etc.) or other necessary parameters to the Service layer. The service layer also needs to interact with the DAO layer. For example, in OrderServiceImp, this involves interacting with the OrderMapper to insert the order data into the database, the Model component finishes processing the data and returns the result to the Controller.

Finally, there is the View component, which presents the data through the front end. In this car rental website, thymeleaf is used to render the data. Through thymeleaf, data is placed into a model object, through which dynamic data is displayed on the page and presented to the user. For example, when a user wants to view the order history, the OrderController passes the order data from the OrderService to the View component, which uses the data passed by the Controller to fill the page, renders the page with JS and CSS files, and presents the results to the user.

The MVC architecture allows developers to improve the user experience of the website by modifying the presentation layer without affecting the underlying business logic.

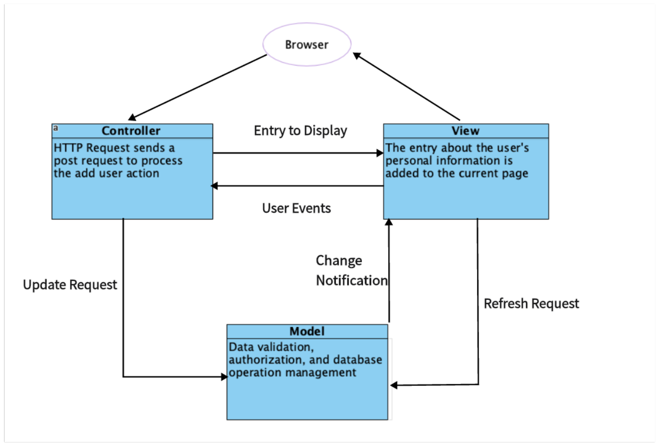

Browser
Controller
HTTP Request sends a post request to process the add user action
Entry to Display
View
The entry about the user's personal information is added to the current page
User Events
Update Request
Change Notification
Refresh Request
Model
Data validation, authorization, and database operation management

Figure 2: MVC Architecture

B. *Software Modules*

The purpose of this project is to build a comprehensive car rental system. We have broken down the entire project into seven modules by evaluating and classifying the main project functions: user management, vehicle management, reservation and rental, car return, reporting and statistics, customer feedback, and payment module. Users are given access to a simple, effective, and secure automobile rental service thanks to the cooperation of these modules. The system also provides business management and monitoring tools for managers so that they can better understand how their firm is running and plan their operations and decision-making for the future. The following provides a thorough explanation of these modules' features along with several examples of functions(The 'system module mind map' provided in the appendix illustrates more details about the system module):

- **User Management**:
  This module is responsible for user-related functions, including registration, login, and information management. Additionally, this module manages user roles, categorizing users into administrators and ordinary users, to make sure that different roles have access to the corresponding features and permissions.

- **Vehicle management**:
  This module is in charge of functions relating to vehicles. Customers can browse information and pictures of vehicles through this module. Moreover, features in this module also meet the administrators' needs for maintaining vehicles, such as entering and editing vehicle information, uploading or changing vehicle images, classifying and retrieving vehicles, updating the usage status of vehicles, changing the vehicles displayed on the homepage, and so on.

- **Appointment**:
  The appointment module is the core module of the project. The module offers several services for the common user, the system, and the administrator and is split into two main portions, booking, and return. Regular users can filter and choose vehicles, specify rental times, choose store addresses, and get verification alerts in the booking area. The system is in charge of producing the pick-up code and notifying the user. When picking up the car, the administrator can see the order details and check the pick-up code. The return section provides users with the ability to return the vehicle

in advance and make a renewal booking. The system sends reminders when a user returns a vehicle over time. When returning the car, the administrator must inspect it and record any unusual circumstances, such as damage, to complete an order.

- **Payment**:
  Processing fee payments and refunds are under the purview of the payment module. Before making a payment, users can get a detailed breakdown of their bill. Only Alipay is accepted for system fee payments.

- **Business reporting and statistics**:
  Various business-related information, including sales data by car type and rental revenue statistics, are provided via the business statistics and reporting function. Administrators may use this information to assess company performance, comprehend the popularity of various cars, and make necessary alterations and advancements.

- **Positive feedback and user feedback**:
  Customers can view the real-time status of their current orders (pending pickup, overdue, etc.) and rate the services of completed orders. They can also browse other users' reviews of the car rental experience and vehicles in this module. Additionally, administrators can have access to customers' reviews and the power to remove inappropriate material, such as horrifying remarks.

C. *High-Level Database Design*

The following ER diagram shows that our database has 8 entities, namely order, outlet, image, car, car_model, user, payment_method, and comment. The relationships in this database are as follows:

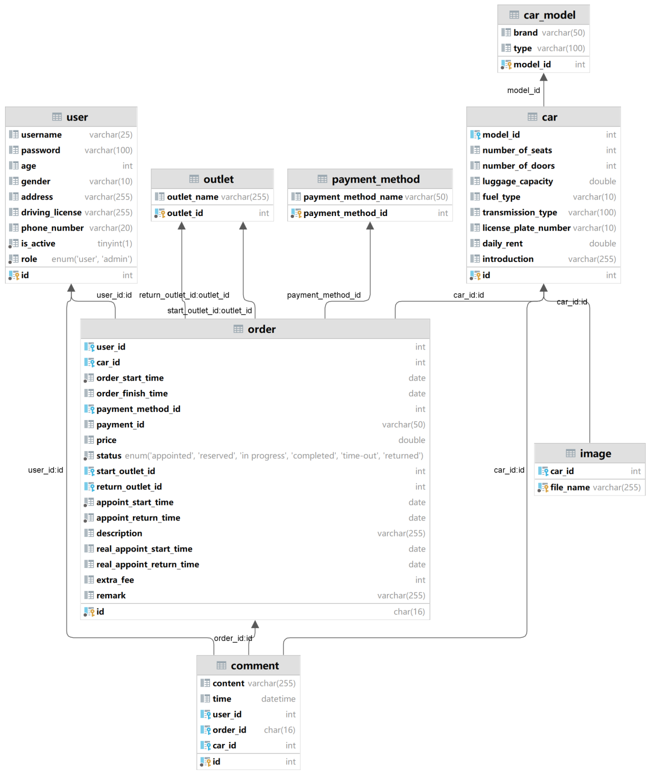


Figure 3: ER Diagram

1. User and order are one-to-many relationships, a user can have multiple orders.
2. Car and order are one-to-many relationships, one car can correspond to multiple orders.
3. Car_model and car are one-to-many relationships, a car model can correspond to more than one car.
4. Payment_method and order are one-to-many relationships, one payment method can correspond to multiple orders.
5. Outlet and order are one-to-many relationships, one outlet can correspond to multiple orders.
6. Car and image are one-to-many relationships, a car can have more than one image.
7. Car and comment are one-to-many relationships, a car can have multiple comments.

The primary key and foreign key are as follows:

1. The primary key of order is id, the primary key of outlet is outlet_id, the primary key of the im-

age is file_name, the primary key of the car is id, the primary key of car_model is model_id, the primary key of the user is id, the primary key of payment_method is id and the primary key of comment is id.

2. In the order table user_id, car_id, payment_method_id, start_outlet_id/return_outlet_id are used as foreign keys related to user(id), car(id), payment_method(id), outlet(id) in the user table, car table, payment_method table and outlet table respectively.
3. In the comment table car_id, order_id, and user_id are used as foreign keys related to car(id), order(id), and user(id) in the car table, order table and user table respectively.
4. car_id in the image table is linked to the car(id) in the car table as a foreign key.
5. model_id in the car table is linked to car_model(model_id) in the car_model table as a foreign key.

In the actual implementation, we do not use strongly constrained foreign keys. When we use foreign keys, when the referenced item changes, the referenced item changes. When the referenced item is deleted, the referenced item remains unchanged. This not only facilitates management but also prevents resource waste to a certain extent. And our database is in the 3rd Normal Form (3NF) [2], which means that there is no redundant data in each table and each table only contains data related to its topic. This design can avoid the problems of data redundancy and inconsistency, and improve the efficiency and maintainability of data querying.

In summary, the database of the "ZPCR" website was designed to manage the relevant data by analyzing the project's requirements and designing seven tables to meet the service needs and performance requirements of the car rental system.

III. System Design

A. High-Level Design

In the high-level design part, the report provides a detailed explanation of the modules of this project. The project is mainly divided into the user module, car module, reservation module, payment module, and back-end management module. Each module has many sub-models. The reason for modularization is to secure the scalability and maintainability of this project. Breaking down the system into multiple small, independent modules can make the system more scalable and maintainable. Another reason that should be mentioned is security and reliability. Validating and processing user input data can ensure the security and reliability of the system.

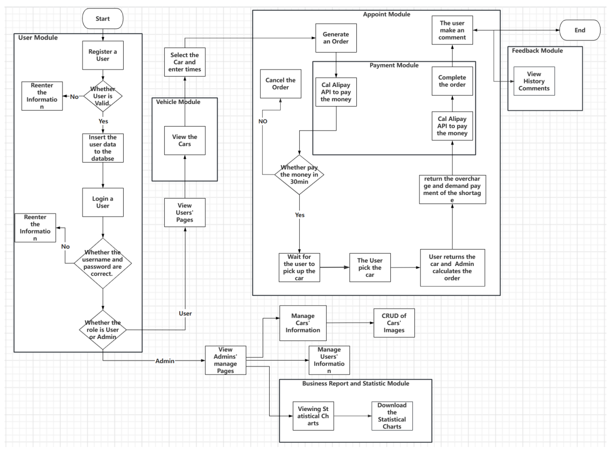

Start
User Module
Register a User
Whether User is Valid.
No
Reenter the Informatio n
Yes
Insert the user data to the databse
Login a User
Whether the username and password are correct.
No
Reenter the Informatio n
Whether the role is User or Admin
User
Admin
Select the Car and enter times
Vehicle Module
View the Cars
View Users' Pages
View Admins' manage Pages
Manage Cars' Information
CRUD of Cars' Images
Manage Users' Informatio n
Business Report and Statistic Module
Viewing St atistical Ch arts
Download the Statistical Charts
Appoint Module
Generate an Order
Payment Module
Cal Alipay API to pay the money
Cancel the Order
NO
Whether pay the money in 30min
Yes
Wait for the user to pick up the car
The User pick the car
User returns the car and Admin calculates the order
return the overchar ge and demand pay ment of the shortag e
Cal Alipay API to pay the money
Complete the order
The user make an comment
End
Feedback Module
View History Comments

Figure 4: Flow Chart of this system

The figure above represents the flow chart of the system. According to the diagram, firstly, users log in and register through the user module. Secondly, they select their desired vehicle through the car module. Finally, they make appointments and complete orders through the order and payment modules.

- **User management**:
  For an online car rental application, the user is the primary service object of this project. The main responsibility of the user module is to handle functions related to the user, such as registration, login, and modification of personal information. In order to achieve persistent user data and facilitate user service, the database stores several attributes, including username, password, role, driver's license, phone number, and address. This module can be divided into the following sub-modules:

1. User registration sub-module allows users to register a personal account on the website by inputting valid information. This function includes verifying the legality of user input, checking if the username already exists, confirming if the input password matches the confirmed password, and validating the SMS verification code. After all verification methods are completed, the user account information will be added to the database.
2. User login and recording function verifies if the user's input username and password match on the login page. Cookie technology is used to help users save their input information and facilitate automatic login for the next time.
3. User information management function helps users modify their personal information, including address, phone number, driver's license, and account password. The flowchart presented below visually demonstrates the processes of these functions.

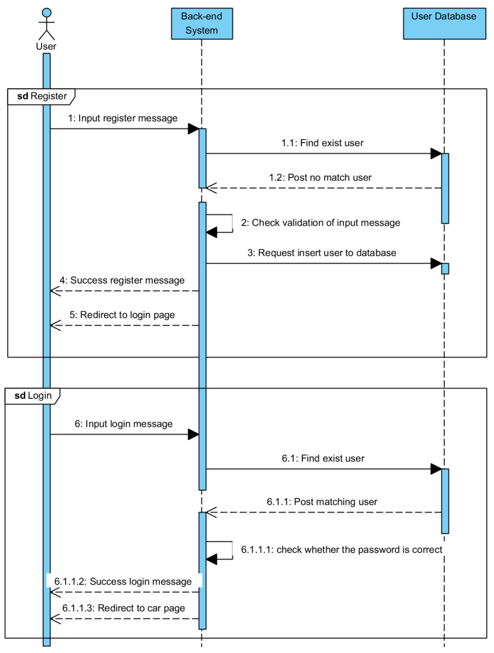

User
Back-end System
User Database
sd Register
1: Input register message
1.1: Find exist user
1.2: Post no match user
2: Check validation of input message
3: Request insert user to database
4: Success register message
5: Redirect to login page
sd Login
6: Input login message
6.1: Find exist user
6.1.1: Post matching user
6.1.1.1: check whether the password is correct
6.1.1.2: Success login message
6.1.1.3: Redirect to car page

Figure 5: Sequence Diagram of User Register and Login

- **Vehical management**:

For an online car rental application, vehicles are important assets that need to be managed by the project. The main responsibility of the vehicle module is to handle vehicle-related functionalities such as querying, adding, editing, and deleting vehicle information. To facilitate users in browsing vehicle information and to facilitate enterprise management of vehicle assets, the brand, model, specific configuration, and rental fee of the vehicles are designed in the database. This module can be further divided into the following sub-modules:

1. Vehicle query sub-module: Users can query vehicles based on different conditions.
2. Vehicle addition sub-module: Administrators can add new vehicles.
3. Vehicle editing sub-module: Administrators can edit the information on existing vehicles.
4. Vehicle deletion sub-module: Administrators can delete vehicles that are no longer needed.
5. The aforementioned sub-modules serve to enhance the efficiency of vehicle management in the car rental application.

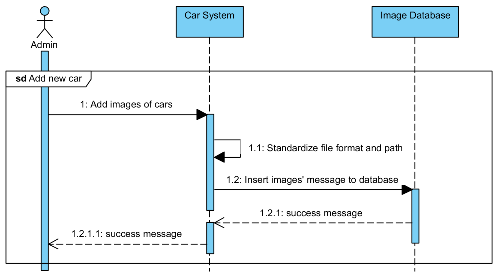


Figure 6: Example Sequence Diagram of Add Images of Cars

- **Appointment**:

The appointment module is primarily responsible for handling vehicle booking-related functions, such as creating bookings, canceling bookings, and viewing booking histories. This project stores all the relevant data including order time, order status, and payment ID in the database to manage the orders. This module can be further divided into the following sub-modules:

1. Create booking sub-module: Allows users to create a vehicle booking.
2. Cancel booking sub-module: Enables users to cancel bookings that have not started.
3. Booking history sub-module: Enables users to view their booking histories.

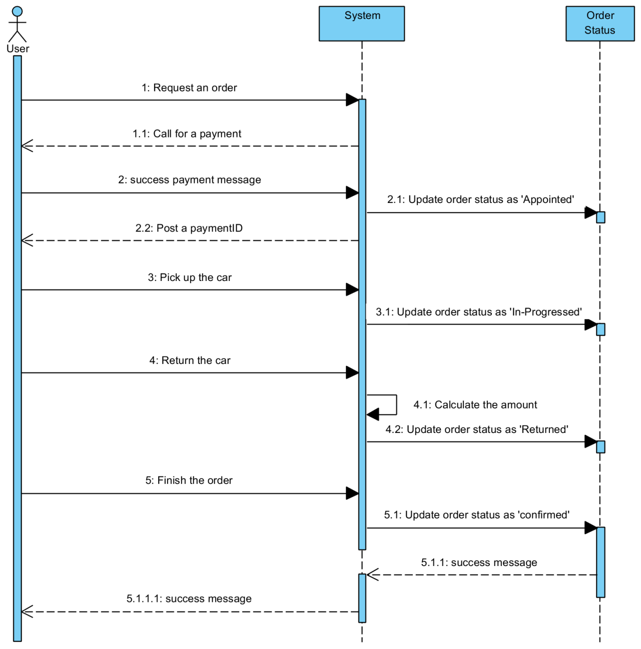

User
System
Order Status
1: Request an order
1.1: Call for a payment
2: success payment message
2.1: Update order status as 'Appointed'
2.2: Post a paymentID
3: Pick up the car
3.1: Update order status as 'In-Progressed'
4: Return the car
4.1: Calculate the amount
4.2: Update order status as 'Returned'
5: Finish the order
5.1: Update order status as 'confirmed'
5.1.1: success message
5.1.1.1: success message

Figure 7: Sequence Diagram of Appointment

- **Payment**:

  The payment module is primarily responsible for processing payment-related functionalities, such as payment of reservation fees, viewing payment history, and order settlement including refunds and adjustments. This module can be further divided into the following sub-modules:

  Reservation payment sub-module: Users can pay reservation fees. Payment history sub-module: Users can view their payment history. Order settlement sub-module: Users can calculate the amount payable based on the return time when returning the vehicle.

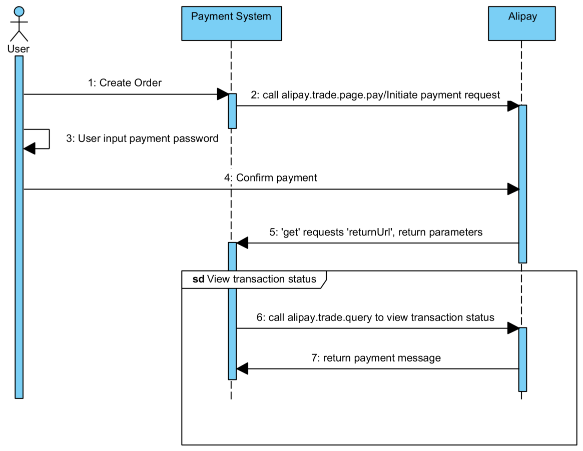

User
Payment System
Alipay
1: Create Order
2: call alipay.trade.page.pay/Initiate payment request
3: User input payment password
4: Confirm payment
5: 'get' requests 'returnUrl', return parameters
sd View transaction status
6: call alipay.trade.query to view transaction status
7: return payment message

Figure 8: Sequence Diagram of Alipay

- **Business reporting and statistics**:

  The business reporting and statistics module also called the backend management module, which is mainly responsible for handling data management-related functions, such as generating statics and reports, downloading statics, etc. This module can be further divided into the following sub-modules:

  1. Statics and report generate sub-module: The system can collect data from the database and generate reports and charts, such as order number, comments' quality, and daily flow.

  2. Statics and report download sub-module: The system can download all the reports and charts.

  3. Data management sub-module: Administrators can view and manage all the data, such as viewing reservation details, canceling reservations, viewing comments, etc.

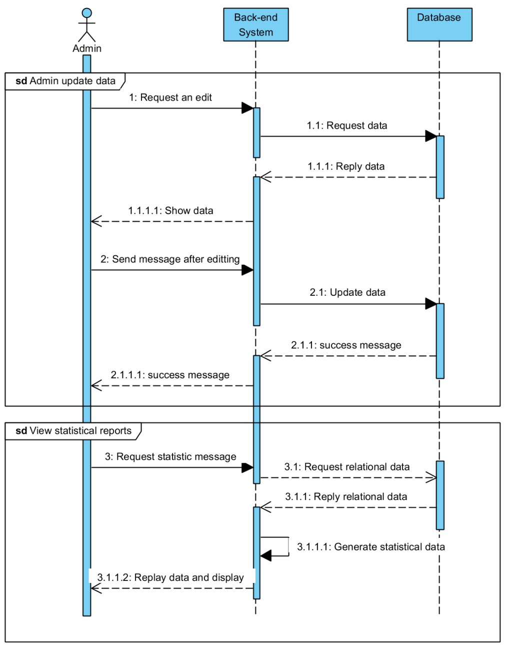

Admin
Back-end System
Database
sd Admin update data
1: Request an edit
1.1: Request data
1.1.1: Reply data
1.1.1.1: Show data
2: Send message after editting
2.1: Update data
2.1.1: success message
2.1.1.1: success message
sd View statistical reports
3: Request statistic message
3.1: Request relational data
3.1.1: Reply relational data
3.1.1.1: Generate statistical data
3.1.1.2: Replay data and display

Figure 9: Example Sequence Diagram of Change Data and View Reports

- **Feedback Module**:

  The Feedback module is primarily responsible for handling the functionality of uploading, displaying, and deleting feedback on orders.

  This module can be further divided into the following sub-modules:

1. Feeback upload sub-module: Users can upload feedback after completing the order.
2. Feeback display sub-module: Users and administrators can view feedback.
3. Feedback deletion sub-module: Users can delete their feedback if they want.

B. Software Support Services

- **Database-related services**:

  MyBatisPlus provides a service layer that helps developers to interact with the database more efficiently and flexibly. The MyBatisPlus service layer provides a set of APIs and methods that abstract the underlying database operations, making it easier to write and maintain database-related code. These services are designed to be reusable, scalable, and maintainable, allowing developers to easily incorporate them into their applications. MyBatisPlus also supports code generation, which helps to reduce boilerplate code and improve productivity. Overall, the MyBatisPlus service layer provides a convenient and powerful way to handle database interactions in modern web applications.

  Based on LambdaQueryWrapper, the implementation of pagination is a technique that enables reusable, scalable, and maintainable pagination. This approach modularizes the pagination functionality into a separate component that can be reused across multiple pages on a website. This method ensures consistency in the navigation structure, reduces code duplication, and improves maintainability. The LambdaQueryWrapper allows for complex query operations and pagination queries to be easily implemented. Using LambdaQueryWrapper simplifies the code, and improves readability, and maintainability. This technique is suitable for various application scenarios that require pagination queries, such as e-commerce websites and social networking sites.

- **System security-related services**:

  The security module of the rental system primarily focuses on user authentication and authorization. The website provides a unified login and registration page, and only logged-in users can access the main functions of the website. The system also manages user identities and permissions, distinguishing between administrators and regular users. This functionality is implemented through Spring Boot's interceptors. By setting a pre-interceptor, the system can intercept requests before they reach the backend logic and identify access paths. Only static resource access paths, login/registration paths, and successful payment callback paths can be directly accessed. Otherwise, the interceptor will check if the user's login information is in the current session scope. If it is, the request will be allowed, otherwise, it will be redirected to the login/registration page. Additionally, the system addresses common vulnerabilities such as SQL injection. Before submitting any forms, the system performs secondary validation and re-encodes any illegal expressions that may cause SQL injection. These measures effectively enhance the security of the website.

- **Webpage navigation-related services**:

  The webpage navigation-related services in the provided code are designed to be reusable, scalable, and maintainable. The navigation menu is modularized as two separate components - the header and the sidebar - and can be included on multiple pages throughout the website. This allows for consistency in the website's navigation structure, reducing the amount of code duplication and improving maintainability.

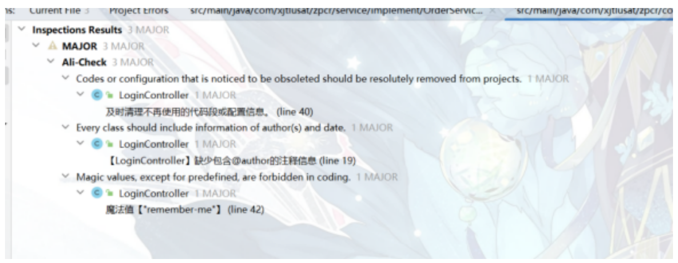
  The navigation components are designed to be responsive and can easily adapt to changes in the website's layout. The sidebar is collapsible and can be hidden on smaller screens to make more room for the main content. The header navigation includes a search bar, making it easier for users to find the content they are looking for.

  To ensure the reliability and availability of the navigation components, logging is put in place. Logging can capture any errors or exceptions that occur within the navigation components, aiding in troubleshooting and maintenance efforts.

C. *Coding Structure and Convention*

Our development team follows a set of coding conventions and structures to ensure consistency, maintainability, and readability of the code. We use the following guidelines:

1. Java methods and parameter names use camelCase, starting with a lowercase letter.
2. Database field names and object properties use snake_case.
3. We follow SOLID principles [3] to ensure the code's extensibility and maintainability.
4. We add comments to important code sections to improve readability.
5. We use automated code formatting tools to maintain a consistent coding style.
6. HTML files include CSS files in the head tag and JavaScript files at the end of the body tag.
7. We use IDE-generated code indentation.
8. We use automated tools such as Alibaba Java Coding Guidelines to maintain code quality and consistency.
9. We use liners to enforce coding standards and identify potential errors, and we use static analysis tools to detect and prevent potential security vulnerabilities.


Current File 3
Project Errors
src/main/java/com/xjtlusat/zpcr/service/implement/OrderServic...
src/main/java/com/xjtlusat/zpcr/co

```
Inspections Results 3 MAJOR
  MAJOR 3 MAJOR
    Ali-Check 3 MAJOR
      Codes or configuration that is noticed to be obsoleted should be resolutely removed from projects. 1 MAJOR
        LoginController 1 MAJOR
          及时清理不再使用的代码段或配置信息。 (line 40)
      Every class should include information of author(s) and date. 1 MAJOR
        LoginController 1 MAJOR
          【LoginController】缺少包含@author的注释信息 (line 19)
      Magic values, except for predefined, are forbidden in coding. 1 MAJOR
        LoginController 1 MAJOR
          魔法值【"remember-me"】 (line 42)
```

Figure 10: Alibaba Java Coding Guidelines Example

Our coding structure and convention contribute to efficient development, testing, and debugging. By following consistent coding standards, we can ensure that every developer can understand and maintain the code, regardless of who wrote it. This improves collaboration, reduces errors, and simplifies testing and debugging.

We also follow a code review practice to ensure code quality. Every code change goes through a peer review process, where other team members review the code for readability, functionality, and compliance with our coding standards.

Our continuous integration workflow ensures that every code change is built, tested, and deployed automatically. This ensures that any errors are caught early in the development process, reducing the time and effort required to fix them.

In summary, our coding structure and convention, automated tools, code review practices, and continuous integration workflow contribute to efficient development, testing, and debugging. By following these practices, we can produce high-quality code that is maintainable, extensible, and secure.

D. *Software Configuration and Production Environment*

- **Software configuration**:

  The rental car system is configured using the SpringBoot framework's configuration files. The common system configurations, such as the database and server port, can be easily modified by updating the configuration files, allowing for quick changes to the system. Additionally, the system has customized configurations for modules, such as the image processing module, where the storage location of images can be configured in the configuration file. The module automatically retrieves the latest configuration from the configuration file and runs accordingly.

  The use of configuration files allows for a unified and easily modifiable configuration of the rental car system. By customizing configurations for each module, the system can be tailored to meet specific requirements.

- **Production environment**:

  The rental car system is deployed on a Linux cloud server. The server has software environments such as MySQL database installed, which are necessary for the system to function correctly. The system is packaged using Maven, which is a popular build automation tool for Java-based applications. The packaged application can be directly run on the server, making the deployment process easy and efficient.

  The use of a cloud server and a packaged application allows for easy deployment and maintenance of the rental car system. The system can be easily scaled to meet the needs of a growing user base, and any issues can be quickly addressed and resolved by the development team.

  In summary, the use of configuration files and a cloud server allows for efficient and easy management of the rental car system. The system can be quickly modified to meet specific requirements, and deployment and maintenance are streamlined and hassle-free.

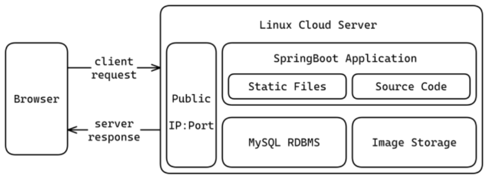


Figure 11: System Deployment Diagram

IV. Software Testing

Software testing is a crucial step in ensuring system quality and stability. Our testing efforts include unit testing, integration testing, and acceptance testing, aiming to guarantee that all modules and features of the project function correctly and meet user requirements.

A. *Unit Testing*

Unit testing involves testing the testable units in the code. In our car rental system project, we have written unit test cases for each module to ensure the correctness and robustness of the code.

For example, in a test case for a smallest testable unit: the CalculateDateDiff() method in DateUtils is as follows:

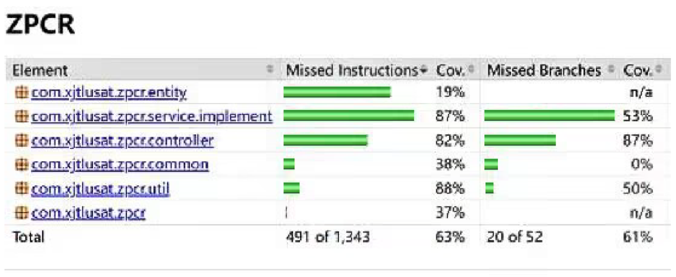
```
@Test
public void testCalculateDateDiff() {
    Calendar calendar = Calendar.getInstance();
    Date date1 = calendar.getTime();
    calendar.add(Calendar.DATE, 5);
    Date date2 = calendar.getTime();
    calendar.add(Calendar.DATE, -10);
    Date date3 = calendar.getTime();
        int diff1 = DateUtils.calculateDateDiff(-
date1, date2);
        int diff2 = DateUtils.calculateDateDiff(-
date1, date3);
        int diff3 = DateUtils.calculateDateDiff(-
date2, date3);
      assertEquals(5, diff1);
      assertEquals(-5, diff2);
      assertEquals(-10, diff3);
  }
```

A test case for a testable unit: the addUser method in UserController is as follows:

```
@Test
public void testSave() throws Exception {
    User user = new User();
    user.setUsername("testUsername");
    user.setPassword("testPassword");
        when(userInfoService.insert(user)).thenReturn("success");

    mockMvc.perform(post("/users/addUser")
                .flashAttr("user", user))
            .andExpect(status().isOk());
    verify(userInfoService, times(1)).insert(user);
}
```

JUnit and Mockito frameworks were used for unit testing [4], and code coverage reports (see image below) were generated using the JaCoCo plugin. These reports allowed us to ensure that critical functions in the project were thoroughly tested, reducing the risk of potential defects.



ZPCR

Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.xjtlusat.zpcr.entity		19%		n/a
com.xjtlusat.zpcr.service.implement		87%		53%
com.xjtlusat.zpcr.controller		82%		87%
com.xjtlusat.zpcr.common		38%		0%
com.xjtlusat.zpcr.util		88%		50%
com.xjtlusat.zpcr		37%		n/a
Total	491 of 1,343	63%	20 of 52	61%

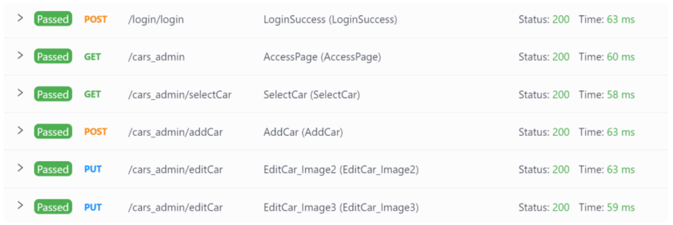
Figure 12: Coverage Report (Critical Function)

B. *Integration Testing*

Integration testing is the process of verifying the interaction and cooperation between different modules. In our car rental system project, we used Apifox to perform integration testing for functionalities involving interactions between multiple modules.

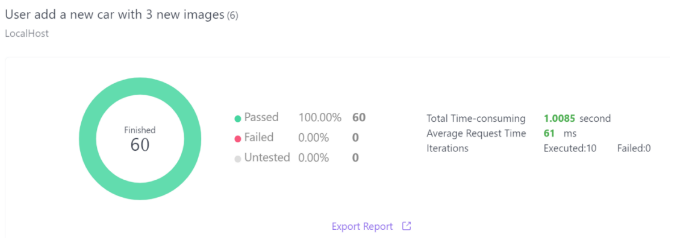
For example, the functionality of admins adding vehicles involves the interaction between the Car Module and Booking Module. Integration test cases are as follows:

- **Test Case 1**:
  Admin adds a car to the database and uploads its three-view images
- **Input**:
  Vehicle information (Brand, Type, License Plate, Seats, etc.), image type, image file
- **Expected Output**:
  The vehicle added successfully, returns to the vehicle management page, and the new vehicle entry is displayed



```
>  Passed  POST  /login/login               LoginSuccess (LoginSuccess)           Status: 200  Time: 63 ms
>  Passed  GET   /cars_admin                AccessPage (AccessPage)               Status: 200  Time: 60 ms
>  Passed  GET   /cars_admin/selectCar      SelectCar (SelectCar)                 Status: 200  Time: 58 ms
>  Passed  POST  /cars_admin/addCar         AddCar (AddCar)                       Status: 200  Time: 63 ms
>  Passed  PUT   /cars_admin/editCar        EditCar_Image2 (EditCar_Image2)       Status: 200  Time: 63 ms
>  Passed  PUT   /cars_admin/editCar        EditCar_Image3 (EditCar_Image3)       Status: 200  Time: 59 ms
```

Figure 13: Test Suites for Test Case 1



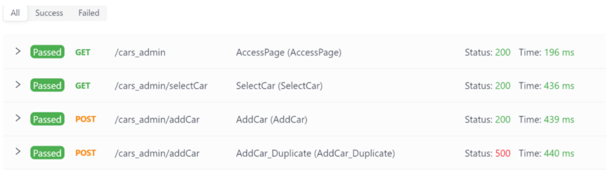
```
User add a new car with 3 new images (6)
LocalHost

Finished   ● Passed    100.00%  60    Total Time-consuming    1.0085 second
60         ● Failed    0.00%    0     Average Request Time    61   ms
           ● Untested  0.00%    0     Iterations              Executed:10   Failed:0

                         Export Report
```

Figure 14: Test Report of Test Case 1

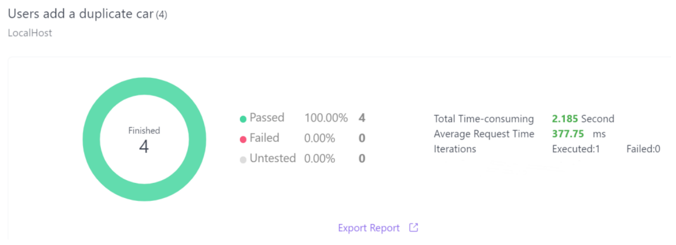
- **Test Case 2**:
  Admin adds a previously uploaded car to the database and uploads its three-view images
- **Input**:
  Vehicle information (Brand, Type, License Plate (same as an existing car in the database), Seats, etc.), image type, image file
- **Expected Output**:
  Vehicle addition failed, displays an error message, and returns to the vehicle management page.



```
All  Success  Failed

>  Passed  GET   /cars_admin                AccessPage (AccessPage)                   Status: 200  Time: 196 ms
>  Passed  GET   /cars_admin/selectCar      SelectCar (SelectCar)                     Status: 200  Time: 436 ms
>  Passed  POST  /cars_admin/addCar         AddCar (AddCar)                           Status: 200  Time: 439 ms
>  Passed  POST  /cars_admin/addCar         AddCar_Duplicate (AddCar_Duplicate)       Status: 500  Time: 440 ms
```

Figure 15: Test Suites for Test Case 2


Users add a duplicate car (4)
LocalHost

Finished 4

● Passed	100.00%	4
● Failed	0.00%	0
● Untested	0.00%	0

Total Time-consuming 2.185 Second
Average Request Time 377.75 ms
Iterations Executed:1 Failed:0

Figure 16: Test Report of Test Case 2

C. *Acceptance Testing*

Acceptance testing involves defining clear acceptance criteria, planning the testing process, designing test cases based on the criteria, executing tests, and recording results, In addition, several test cases were designed. Finally, test results were discussed and negotiated with stakeholders, and identified defects were fixed. Based on the acceptance criteria, we have derived a set of acceptance tests, which cover both functional and non-functional requirements, an acceptance test suit for PBI *LOG001* as follows:

- **Functional**:
  1. Given that I am a user, when I click the 'Login In' button with an account name and a password, then I will be redirected to the car selection page.
  2. Given that I am a user, if I logged in to my account 14 days ago and clicked on the "remember me" button, the website will automatically fill in my information saved when I visit this website again.
  3. Given that I am a user when I enter the wrong username or password, the website will remind me of login failure.
- **Non-functional**:
  Measure the response time for registration and login processes; Evaluate the user-friendliness of the registration and login interfaces.
- **Test results**:
  As shown in Figure 17 (a) and (b), all acceptance standards are fully met.

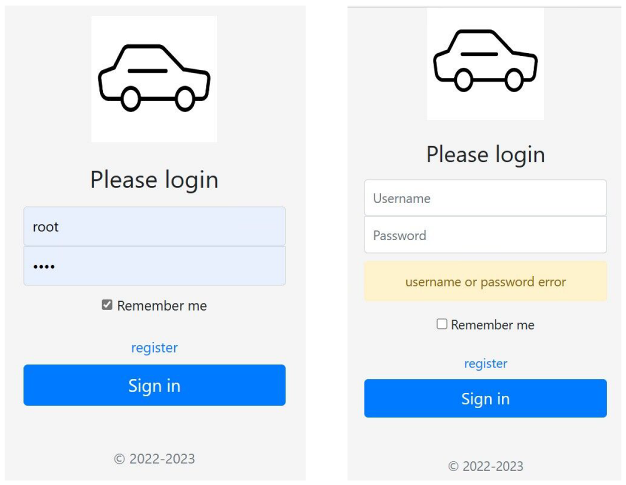


Figure 17: (a) Website automatically fills in the login information.
(b) A login failure error appears when the user filled in invalid information.

After the acceptance tests have been completed, we review the test results with the project stakeholders. Discuss any discrepancies between the expected and actual results, and negotiate an acceptable resolution. Negotiated outcomes include fixing defects (e.g., fixing a bug where no error message is displayed for invalid input during user registration or login) and updating system requirements (e.g., removing the "batch delete" feature from the cancel booking page).

A final meeting between the development team and stakeholders is held. After the meeting, it was decided that the system with the defects discovered during acceptance testing fixed would be deployed as version 1.0 and go live.

References

[1] D.-P. Pop, and A. Altar, "Designing an mvc model for rapid web application development," *Procedia Eng.*, vol. 69, pp. 1172–1179, 2014, doi: https://doi.org/10.1016/j.proeng.2014.03.106. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187770581400352X

[2] C. Zaniolo, "A new normal form for the design of relational database schemata," *ACM Trans. Database Syst.*, vol. 7, no. 3, p. 489, Sep. 1982,

doi: 10.1145/319732.319749. [Online]. Available: https://doi.org/10.1145/319732.319749

[3] V. Madasu, T. Venna, and T. Eltaeib, "Solid principles in software architecture and introduction to resm concept in oop," *J. Eng. Sci. Technol.*, vol. 2, pp. 3159–40, 2015.

[4] E. Daka, and G. Fraser, "A survey on unit testing practices and problems," in *2014 IEEE 25th Int. Symp. Softw. Rel. Eng.*, vol. 0, 2014, pp. 201–211, doi: 10.1109/ISSRE.2014.11.