CPT203
Coursework 2

2024/2025 Semester 1
<2024/12/10>

Group number: < 81 >

Student 1 Name: Runze.Zhang Student 1< ID:2252881 >

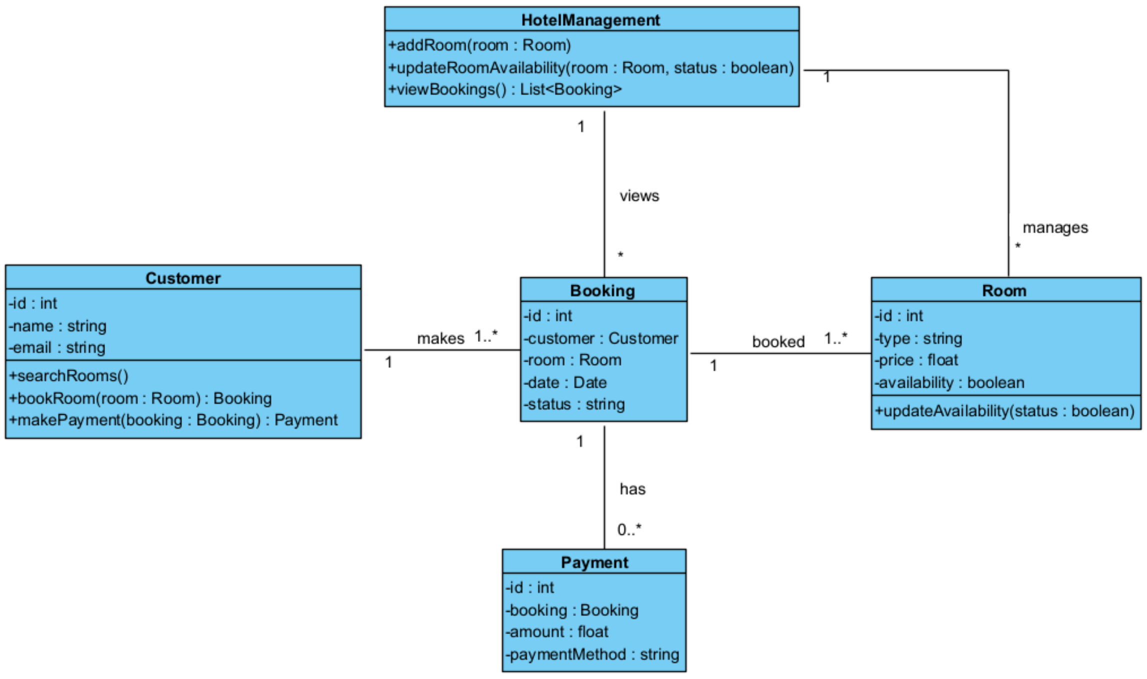
Student 2 Name: Siyu.Zhou Student 2 ID: <2257056>

Student 3 Name: Pengyu.Bao Student 3 ID: <2252520>

Student 4 Name: Enze.Zhou Student 4 ID: <2254411>

Student 5 Name:Ruoxi.Liao Student 5 ID:<2254637>

Q1. You are supposed to draw the class diagram for the case (10 marks).



Q2. Evaluate the design principles in the class diagram. (15 marks).

1. **Single Responsibility Principle**
Application: The customer class is responsible for the core functions of the customer; the room class handles the basic information of the room, such as room type, price and availability, and updates the room status; the reservation class records and manages the reservation information and logic of the customer and the room, decoupling the customer and the room; the payment class handles the payment-related logic, such as the amount and payment method; the hotel management class focuses on hotel management functions, such as adding rooms, updating room availability, and viewing reservation information.
Advantages: It reduces the complexity of the class; a class is only responsible for one responsibility. The division of responsibilities also improves the module's cohesion and increases the class's readability and maintainability.

2. **Open-Closed Principle**
Application: The room class can support new room types through inheritance and extension without modifying the existing code. The Payment class can support more payment methods through extension without modifying the original logic.
Advantages: In this case, the class's modules and functions are open to extension and closed to modification. At the same time, different classes do not need to make significant modifications to the existing code when expanding their functions, reducing the possibility of introducing errors.

3. **Interface Segregation Principle**
Application: The customer class only contains the core functions that customers need, such as searching for rooms, booking, and paying, and will not assume other responsibilities unrelated to customers. The hotel management class focuses on hotel management-related operations, such as adding rooms and viewing reservations, without involving customer functions.

Advantages: Classes should not be forced to rely on methods they do not use. Each class's functions should be consistent with its actual responsibilities. Following this principle avoids redundant dependencies.

4. **Dependency Inversion Principle**

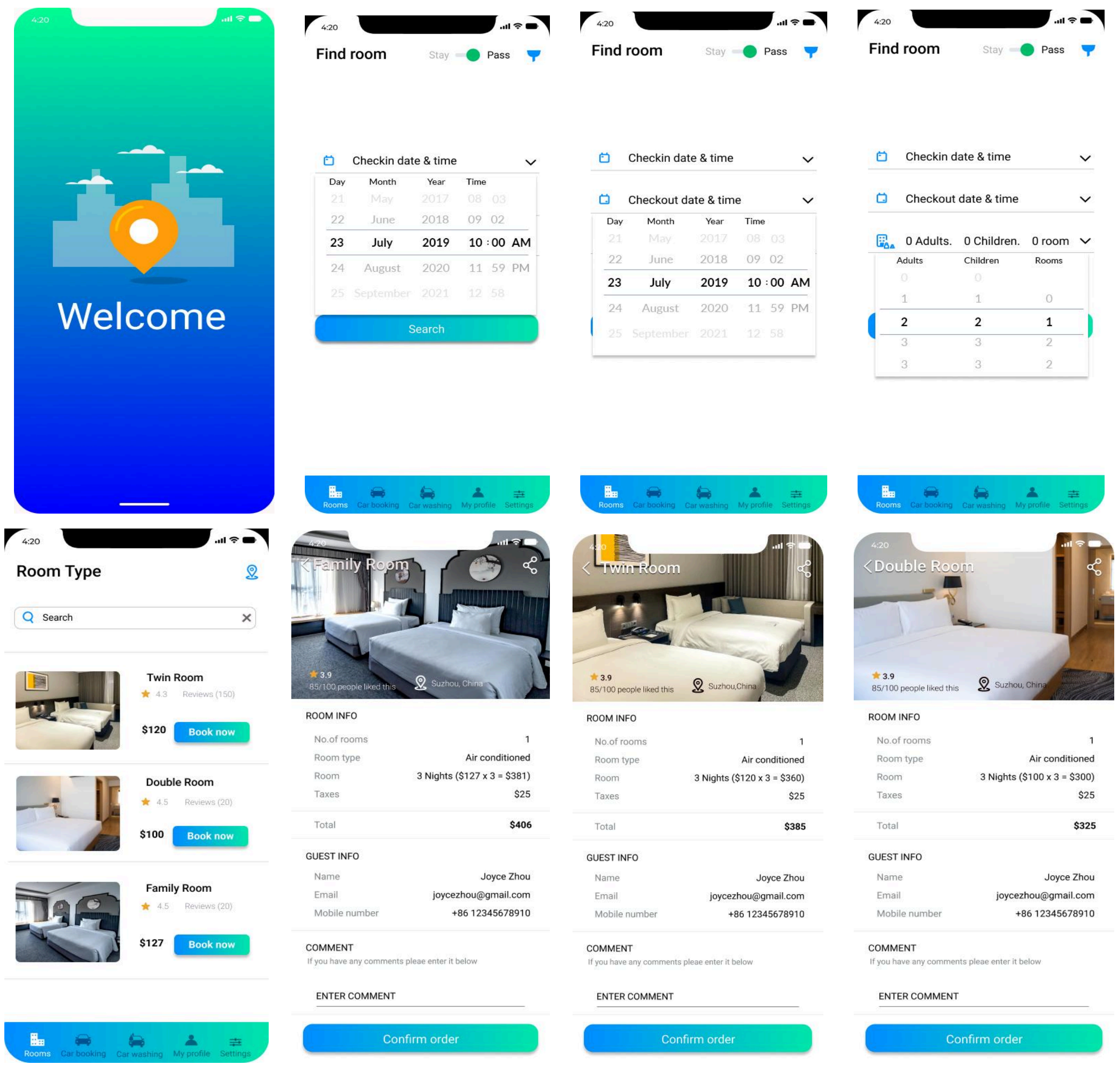
Application: The booking class does not directly rely on specific Customer or Room implementations but achieves loose coupling through object references. The smaller the association, the better; the payment class relies on the abstract Booking object instead of directly operating customers or rooms.

Advantages: Architectures built on abstraction are more stable than those built on details. By introducing abstract interfaces, new payment methods or room types can be easily added without modifying the high-level logic in the Customer or Booking class.

This can reduce direct dependencies between classes and improve the flexibility and maintainability of the system.

Q3. You are required to create the UI pages of these functions. (15 marks)

- **Customer Room Search and Booking**



Family Room

ROOM INFO

No.of rooms	1
Room type	Air conditioned
Room	3 Nights ($127 x 3 = $381)
Taxes	$25
Total	$406

GUEST INFO

Name	Joyce Zhou
Email	joycezhou@gmail.com
Mobile number	+86 12345678910

COMMENT
If you have any comments pleae enter it below
ENTER COMMENT

Twin Room

ROOM INFO

No.of rooms	1
Room type	Air conditioned
Room	3 Nights ($120 x 3 = $360)
Taxes	$25
Total	$385

GUEST INFO

Name	Joyce Zhou
Email	joycezhou@gmail.com
Mobile number	+86 12345678910

COMMENT
If you have any comments pleae enter it below
ENTER COMMENT

Double Room

ROOM INFO

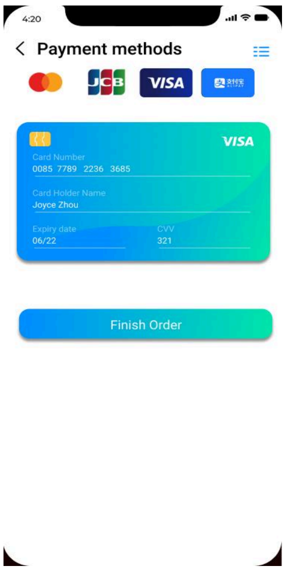
No.of rooms	1
Room type	Air conditioned
Room	3 Nights ($100 x 3 = $300)
Taxes	$25
Total	$325

GUEST INFO

Name	Joyce Zhou
Email	joycezhou@gmail.com
Mobile number	+86 12345678910

COMMENT
If you have any comments pleae enter it below
ENTER COMMENT

- **Payment Processing**

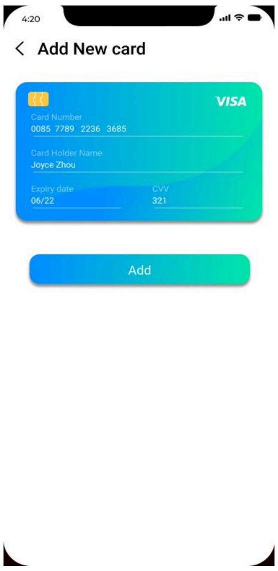

4:20

Payment methods

VISA

Card Number
0085 7789 2236 3685
Card Holder Name
Joyce Zhou
Expiry date 06/22
CVV 321

Finish Order

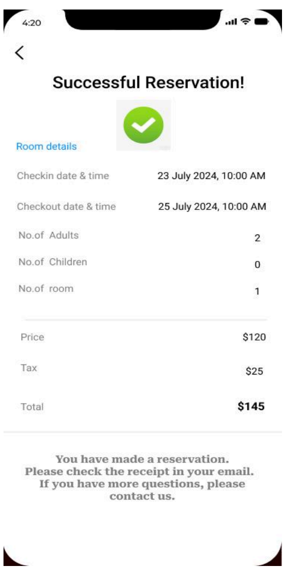

4:20

Add New card

VISA

Card Number
0085 7789 2236 3685
Card Holder Name
Joyce Zhou
Expiry date 06/22
CVV 321

Add


4:20

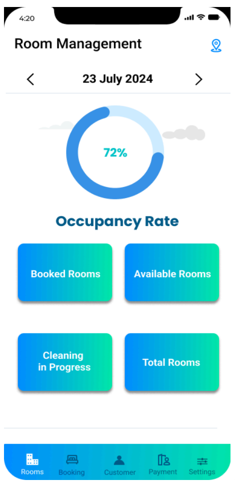
Successful Reservation!

Room details

Checkin date & time	23 July 2024, 10:00 AM
Checkout date & time	25 July 2024, 10:00 AM
No.of Adults	2
No.of Children	0
No.of room	1
Price	$120
Tax	$25
Total	**$145**

You have made a reservation.
Please check the receipt in your email.
If you have more questions, please contact us.

- **Room Booking Management**


4:20

Room Management

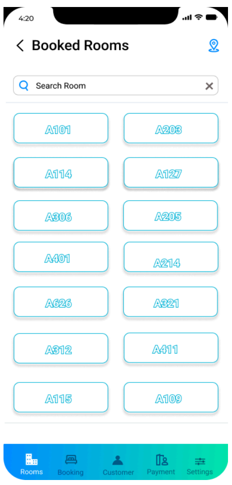
23 July 2024

72%

Occupancy Rate

Booked Rooms | Available Rooms | Cleaning in Progress | Total Rooms

Rooms Booking Customer Payment Settings

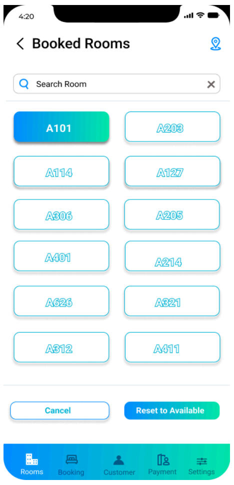

4:20

Booked Rooms

Search Room

A101 A203 A114 A127 A306 A205 A401 A214 A626 A321 A312 A411 A115 A109

Rooms Booking Customer Payment Settings


4:20

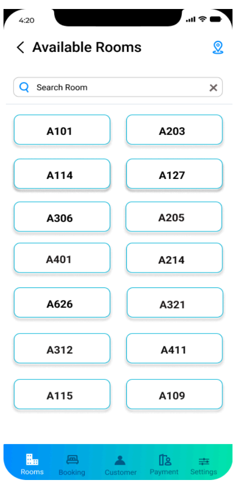
Booked Rooms

Search Room

A101 A203 A114 A127 A306 A205 A401 A214 A626 A321 A312 A411

Cancel | Reset to Available

Rooms Booking Customer Payment Settings

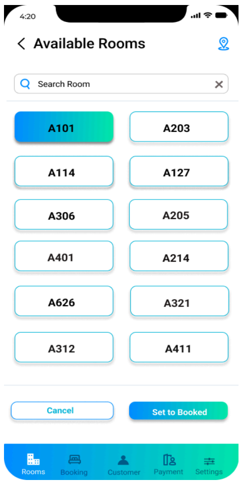

4:20

Available Rooms

Search Room

A101 A203 A114 A127 A306 A205 A401 A214 A626 A321 A312 A411 A115 A109

Rooms Booking Customer Payment Settings


4:20

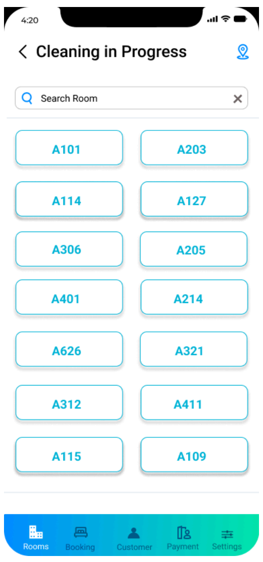
Available Rooms

Search Room

A101 A203 A114 A127 A306 A205 A401 A214 A626 A321 A312 A411

Cancel | Set to Booked

Rooms Booking Customer Payment Settings

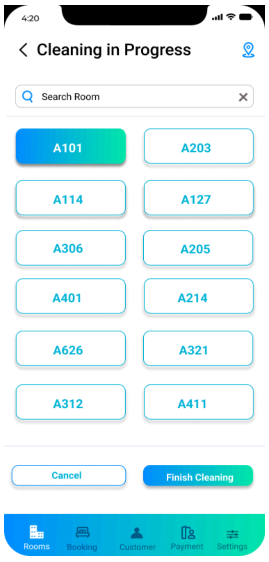

4:20

Cleaning in Progress

Search Room

A101 A203 A114 A127 A306 A205 A401 A214 A626 A321 A312 A411 A115 A109

Rooms Booking Customer Payment Settings


4:20

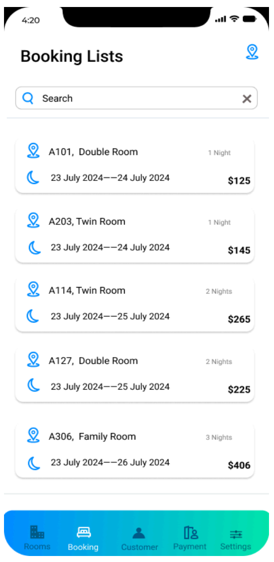
Cleaning in Progress

Search Room

A101 A203 A114 A127 A306 A205 A401 A214 A626 A321 A312 A411

Cancel | Finish Cleaning

Rooms Booking Customer Payment Settings


4:20

Booking Lists

Search

A101, Double Room — 1 Night — 23 July 2024——24 July 2024 — $125

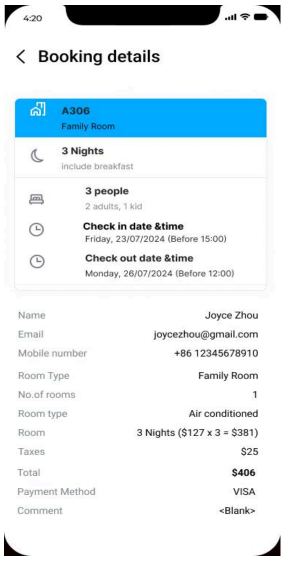
A203, Twin Room — 1 Night — 23 July 2024——24 July 2024 — $145

A114, Twin Room — 2 Nights — 23 July 2024——25 July 2024 — $265

A127, Double Room — 2 Nights — 23 July 2024——25 July 2024 — $225

A306, Family Room — 3 Nights — 23 July 2024——26 July 2024 — $406

Rooms Booking Customer Payment Settings


4:20

Booking details

A306
Family Room

3 Nights
include breakfast

3 people
2 adults, 1 kid

Check in date &time
Friday, 23/07/2024 (Before 15:00)

Check out date &time
Monday, 26/07/2024 (Before 12:00)

Name	Joyce Zhou
Email	joycezhou@gmail.com
Mobile number	+86 12345678910
Room Type	Family Room
No.of rooms	1
Room type	Air conditioned
Room	3 Nights ($127 x 3 = $381)
Taxes	$25
Total	**$406**
Payment Method	VISA
Comment	<Blank>

Q4. For each of the UI page, (1) illustrate what interface design principles are used, (2)how they are applied with specific examples, and (3)how they improve the interaction between the system and users (e.g., equality, diversity and inclusion). (20 marks)

(1)Interface Design Principles Used
Consistency: The whole system adopts a unified visual style, layout structure and interaction mode.
Clarity: Interface elements should be clear and concise, reducing barriers to understanding, so that the user can understand at a glance what he is doing and how to operate.
Simplicity: Avoid too many interface elements, highlight the most important content and avoid redundant information.
Accessibility: Design an easy-to-use interface for users with different needs (e.g., visually impaired).
Feedback: The system responds to user operations in a timely manner and provides clear status feedback.
Visual Hierarchy: Establish a clear hierarchy of information through color, size, and spacing.
Usability: Simplify the operation process and reduce the cognitive burden on users.
Error Prevention: Provide confirmation and reminders at critical operations.
(2) How They Are Applied with Specific Examples
Consistency:

1. Navigation bar design: all pages at the bottom of the unified use of “Rooms”, “Booking”, “Customer”, “Payment”, “Settings” five options
2. Button style: all operable buttons use a uniform blue-green gradient color, rounded rectangle (such as “Search”, “Add”, “Finish Order” button)
3. Room numbering: Uniform use of “letters + numbers” coding (such as A101, A203) to button presentation.

Clarity:

1. Room Status: Different colors are used to distinguish booked/available status.
2. Payment Methods: Clear universal icons to show supported payment card types.
3. Booking details: Modularized display of various information.

Simplicity:

1. Interface Layout: leave plenty of clicking area.
2. Text display: use clear fonts and appropriate size.

Visual Hierarchy:

1. Room Booking Management Page: Use a large doughnut chart showing percentages to display occupancy rates and highlight important data.
2. Successful booking page: use green checkmarks as visual focus to emphasize successful status.
3. Room Type: Use a larger portion of the page for important information such as room images to direct the user's attention, and use smaller font sizes and gray fonts for secondary information such as ratings and reviews.

Feedback:

1. Booking process: each step has clear progress prompts.
2. Payment Completion: Detailed booking information and successful prompts are displayed.
3. Search operation: real-time display of available rooms.

Usability:

1. Date Selection: Provides a calendar-style selection interface, intuitive and convenient.

2. Room Filtering: Provide a search box to quickly locate the room
3. Check-in information: adopts form-based design, clearly displaying various information.

Error Prevention:

1. Date Selection: Restricts the selection of past dates from being possible.
2. Payment Information: Provides a standardized format for entering card numbers.
3. Booking Confirmation: Displays detailed information for the user to check

(3)How they improve the interaction between the system and users (e.g., equality, diversity and inclusion).

Equality:

Equal access to information: transparent and clear display of price information, reviews, booking rules and terms of service are uniform and open.

Equality of access: Multiple payment methods (VISA, MasterCard, JCB, etc.) support multi-currency payments. Uniform customer service standards, fair room allocation mechanism, uniform service response time.

Diversity:

Not only is the room type diverse (standard room, family room, etc.), family, business, personal and other different types of users to book, and flexible price range to adapt to different budgets. In addition, the booking options are flexible (e.g., the number of occupants, the number of days of stay can be adjusted). Provide multi-language interface, taking into account the habits of users from different cultural backgrounds.

Inclusion:

Accessible design: use high contrast color matching to take care of visually impaired users, and access to screen reading to take care of the hearing impaired.

Provide a large enough clickable area, large enough fonts and clear text descriptions suitable for users of all ages to avoid comprehension barriers.

With the help of these design principles, our UI design improves the overall usability of the system, reduces the learning cost and cognitive burden on users, and results in a better user experience. At the same time, it creates a more inclusive environment that meets the needs of different groups.

Q5. Please describe how would you test the function in cases that it is working and NOT working (e.g., boundary input). For each of the test cases, please also specify what can be the solution to the problem. (10 marks)

For the payment function test of the hotel reservation system, we need to design the following types of test cases:

1. **Positive test**, including check-in and check-out within the valid date range or within a very close adjacent date range, should result in successful schedule and correct update of the order status.

2. **Negative test**, including: (1) If the check out date is earlier than the check in date, the system should throw an exception and prompt "check out date is earlier than the check in date"; (2) If you enter the wrong date format in the test (such as: 2024-99-99), the system will throw an exception and prompt "Invalid date format". These two types of problems are solved by logical processing of user input,

checking and ensuring that the check-in date is earlier than the check-out date, and using a calendar component to limit the user's selection of valid dates.

3. **Boundary test**, including :(1) The scheduled day of the test day as the check-in date, the system should allow users to choose the day date, to ensure the normal operation of the system; (2) When very distant dates are booked as check-in dates, appropriate forward dates shall be properly treated by the system but reasonably limited. If an error occurs, the possible solution is to improve the date management mechanism of the system and to add a comprehensive calendar component and set an upper limit so that it can accommodate a variety of valid and reasonable date ranges.

Finally, in order to ensure the reliability of the reservation function, it is suggested to implement the following solutions: (1) add date logic control to prevent illegal date combinations; (2) Establish a complete calendar component, limit the user to select the effective date, avoid formatting errors; (3) Implement concurrency control mechanism to ensure that the system can run normally when multiple users book at the same time.

Q6. Fill in the following code first and provide test cases verifying if the exceptions are thrown successfully. (10 marks)

```
public class Payment {
        private int paymentID;
        private int bookingID;
        private double amount;
        private String paymentMethod; // Example: "Alipay", "Wechat"
        private boolean isProcessed;

        public Payment(int paymentID, int bookingID, double amount, String paymentMethod) {
        this.paymentID = paymentID;
        this.bookingID = bookingID;
        this.amount = amount;
        this.paymentMethod = paymentMethod;
        this.isProcessed = false;
        }

        public boolean processPayment() throws IllegalArgumentException {
        // Q6.1 - Fill in the Code
      if (amount <= 0) {
         throw new IllegalArgumentException("Amount must be greater than 0");
      }

      if (!paymentMethod.equals("Alipay")&&!paymentMethod.equals("Wechat")) {
               throw new IllegalArgumentException("Payment method must be Alipay or
Wechat");
      }
```

```
        isProcessed = true;
        return true;

          }

          public boolean isProcessed() {
          return isProcessed;
          }

          public double getAmount() {
          return amount;
          }
}
```

//Q6.2 - Test cases

```
 package org.example;


import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;


public class PaymentTest {

   @Test
   public void testProcessPayment_Success() {
      Payment payment = new Payment(1, 101, 100.0, "Alipay");
      assertTrue(payment.processPayment(), "Payment should be processed successfully");
   }

   @Test
   public void testProcessPayment_AmountZeroOrNegative() {
      Exception exception = assertThrows(IllegalArgumentException.class, () -> {
         org.example.Payment payment = new org.example.Payment(2, 222, 0, "Wechat");
         payment.processPayment();
      });
      assertEquals("Amount must be greater than 0", exception.getMessage());

      exception = assertThrows(IllegalArgumentException.class, () -> {
         org.example.Payment payment = new org.example.Payment(3, 333, -100.0, "Alipay");
         payment.processPayment();
      });
      assertEquals("Amount must be greater than 0", exception.getMessage());
```

```
    }

    @Test
    public void testMaxAmount() {
        Payment payment = new Payment(4, 444, Double.MAX_VALUE, "Wechat");
         assertTrue(payment.processPayment(), "Payment should be processed successfully with
maximum amount");
    }


    @Test
    public void testProcessPayment_InvalidPaymentMethod() {
        Exception exception = assertThrows(IllegalArgumentException.class, () -> {
            Payment payment = new Payment(5, 555, 100.0, "Cash");
            payment.processPayment();
        });
        assertEquals("Payment method must be Alipay or Wechat", exception.getMessage());
    }

    @Test
    public void testIsProcessed_AfterSuccessfulPayment() {
        Payment payment = new Payment(6, 666, 100.0, "Alipay");
        payment.processPayment();
            assertTrue(payment.isProcessed(), "Payment should be marked as processed after
successful payment");
    }
}
```
Q7. Please describe what would you organize this process as a team. (20 marks)

1. Define Objectives and Scope

- **Objective: Understand what aspects of the system need validation (e.g., usability, performance, feature functionality).**
- **Scope: Identify features or components to test. Focus on high-priority or high-risk areas.**

Potential Problems:

- **Overlooking critical features.**
- **Scope creep causing delays.**

Solutions:

- **Use a priority matrix to rank features based on importance and risk.**
- **Define clear boundaries for the testing phase and revisit only if time permits.**

---

2. Identify the Target User Group

- **Engage a diverse group of users representing the actual user base.**
- **Use personas to ensure all user types are represented.**

Potential Problems:

- **Recruiting users who don’t represent real-world scenarios.**

- **Bias in feedback from a non-representative sample.**

Solutions:

- **Collaborate with stakeholders to identify the target audience and finish entire market evaluation.**
- **Diversify the user pool to include varying levels of expertise and demographics, satisfy more diverse need of users .**

---

3. Develop Test Scenarios and Tasks

- **Create real-life scenarios that users might encounter when interacting with the system.**
- **Write step-by-step tasks users will perform to achieve specific goals.**

Potential Problems:

- **Tasks that are too abstract or irrelevant to real-world usage.**
- **Users not understanding the tasks.**

Solutions:

- **Conduct a pilot test internally to refine tasks.**
- **Realistic use case simulation of user needs Simulate possible problems and modify them in advance**

---

4. Select Testing Methods

Choose methods based on objectives:

- **Usability Testing: Observe users completing tasks to identify pain points.**
- **A/B Testing: Compare variations of a feature.**
- **Surveys/Interviews: Gather qualitative feedback.**
- **Beta Testing: Allow users to explore freely over time.**

Potential Problems:

- **Data overload from too many testing methods.**
- **Users focusing on edge cases that aren’t core use cases.**

Solutions:

- **Focus on 1–2 key methods per testing phase.**
- **Make simple and clear instructions for customers to use, keep the product iterative and updated, and maintain and replace the product according to the changes in customer needs**

---

5. Conduct the Tests

- **Ensure all team members know their roles (e.g., observer, facilitator, note-taker).**
- **Use recording tools for sessions to review later.**
- **Avoid influencing user actions or responses during testing.**

Potential Problems:

- **Users feeling pressured and not acting naturally.**
- **Missing critical insights during live sessions.**

Solutions:

- **Create an open-source community platform for product development, facilitate real-time communication between users and developers, and facilitate iterative updates of products**
- **Real-time background monitoring of system operation to prevent the occurrence of vulnerabilities**

Section 4. Peer review form template

CPT203 Coursework
Peer review
Individual Contribution for Group Report

Group Number: <81>

Name	**ID Number**	**Contribution (%)** Please enter an integer, for example 15% contribution, please enter 15. The sum of this column should be 100	**Signature**
1. Runze.Zhang	2252881	20	张润泽
2.Siyu.Zhou	2257056	20	周思雨
3.Pengyu.Bao	2252520	20	鲍鹏宇
4.Enze.Zhou	2254411	20	周恩泽
5.Ruoxi.Liao	2254637	20	廖若希

END