CPT203 Software Engineering I

Coursework 2 – Project Report

Release date	12th Nov 2025
Submission date	12th Dec 2025
Coursework type	Group work
Percentage in final marks	15%
Submission method	CPT203 of Learning Mall Core
Late submission policy	5% of the awarded marks shall be deducted for each working day after the submission date, up to a maximum of five working days. Late submission for more than five working days will be not accepted.
Important notes	• Any use of Generative AI is not allowed. • Plagiarism results in award of ZERO mark; all submissions will be checked by TurnItIn. • The formal procedure for submitting coursework at XJTLU is strictly followed. Submission link on Learning Mall will be provided in due course. The submission timestamp on Learning Mall will be used to check late submission.

Section 1. Introduction

This coursework is designed to assess your ability to apply key Software Engineering principles and practices introduced in this module to a realistic system scenario: the Smart Parking System (SPS). You are expected to demonstrate competence in object-oriented analysis and design, architectural reasoning, software testing, exception handling, risk management, and user-centred evaluation.

Your answers should reflect a clear understanding of the taught concepts and an ability to justify design decisions with reference to software quality attributes such as maintainability, scalability, usability, and reliability. You are required to produce your own original work and provide well-reasoned, coherent responses supported by appropriate diagrams, examples, and explanations where relevant.

General instruction

1. Read the case study in Section 2 and solve the involved 7 questions.

2. The page limit of this coursework is 7 pages max, excluding the cover page and appendix (if any).

3. Grouping information is provided on the learning mall.

4. Use the templates at the end of this document.

Section 2. Case and Related Tasks

A university plans to launch a **Smart Parking System (SPS)** to improve on-campus parking efficiency and management transparency. The system should allow users to view available parking spaces, reserve a parking spot, record entry and exit times, calculate parking fees, and make online payments. Administrators should be able to monitor parking usage, update parking space status, handle exceptions, and generate statistical reports. The system involves five main classes:

- **User:** A person who uses the SPS to view parking availability, make reservations, and manage their parking activities.
- **ParkingSpot:** Represents individual parking spaces with basic details such as location and availability.
- **Reservation:** Handles the user's parking booking details, including the reserved time period.
- **Payment:** Processes payments for completed parking reservations.
- **AdminManagement:** Allows system administrators to update parking spot availability and view or manage parking-related records.

Q1. Class Diagram (15 marks)

In order to translate the case scenario into a software system, we need to model its structure at a high level. Based on the scenario above, draw a class diagram for the SPS. The class diagram should reflect the core logic of the system.

Hints:
1. *Include all of the above classes: User, ParkingSpot, Reservation, Payment, and AdminManagement (you may add more classes if needed).*
2. *Show the key attributes, methods, and the relationships between classes (e.g., association, aggregation, etc) together with multiplicities.*

Q2. Software Design Concepts (15 marks)

Based on the class diagram you created in Q1, evaluate your design using at least **three** of the software design concepts we learned in week 8 (e.g., Abstraction, Modularity,etc). Explain how your design reflects each selected concept and how applying these concepts improves the system's maintainability and scalability.

Hints:
You must refer to your own diagram, using specific examples to describe the design concepts involved (e.g., "In my design, X class improves cohesion because…").

Q3. Architecture Selection (15 marks)

The university is deciding which architecture is most suitable for developing the SPS. Select **one** architecture style from the options below and justify your choice based on the SPS context: Layered Architecture, MVC, Client-Server. Complete the following:

a) Draw a structural diagram of the selected architecture (hand-drawn or digital).
b) Explain the reasons for choosing this architecture, linking your justification to the SPS business requirements.
c) Discuss, critically, how this architecture supports or limits future system expansion (e.g., adding new parking areas, new payment methods, or new user-facing features).

Hints:

3. *Focus on why the chosen architecture particularly fits SPS, not just its generic pros/cons from textbook.*
4. *Consider system features that require separation of concerns, scalability, or clear data flow.*
5. *Mention at least one limitation — no architecture is perfect for everything.*

Q4. Test Case Design (10 marks)

Design test cases for the "**Reserve Parking Spot**" function. Include:

- **Two valid test cases**, checking that the function works correctly under normal and expected conditions.
- **Two invalid test cases**, checking how the system responds to incorrect or unacceptable input.
- **One boundary test case**, checking the system's behaviour at the edge of acceptable input ranges.

Example:
For a function that allows users to enter their age:
Valid: age = 25
Invalid: age = -3
Boundary: age = 0
(This example is for illustration and should not be taken as answers.)

For each test case, provide:

- Test objective (what the test aims to verify)
- Input type (e.g., DateTime; String, Integer, etc)
- Input value (the specific value you use for the test case)

- Output (describe how the system responds (e.g., confirmation message or error message) and whether any system data is created, updated, or unchanged.)
- Note (this is optional, in case you make any specific assumption for a test case)

Hint:
You may use a table like below to structure your answer for readability:

Category	***Test Objective***	***Input Type***	***Input Value***	***Output***	***Note***
Boundary test case	*To verify that the system correctly handles the minimum boundary age value*	*Integer*	*0*	*The system accepts the input and proceeds to the next step. No system data is changed.*	*Assume our system accept 0 as a valid age.*

Q5. JUnit Testing (15 marks)

As it is shown in the following code of Reservation function, the customers need our system to **throw exceptions in the following two conditions**:

1. When the reservation times are invalid (start is null, end is null, or start >= end), throw IllegalArgumentException.
2. When the parking spot is not AVAILABLE, throw IllegalStateException.

Meanwhile, **for valid inputs, the reservation should succeed and return true**.

```
public class Reservation {
  private int reservationID;
  private int userID;
   private ParkingSpot spot;
   private LocalDateTime startTime;
  private LocalDateTime endTime;
  private boolean confirmed;


    public Reservation(int reservationID, int userID, ParkingSpot spot,
LocalDateTime startTime, LocalDateTime endTime) {
        this.reservationID = reservationID;
        this.userID = userID;
        this.spot = spot;
        this.startTime = startTime;
        this.endTime = endTime;
        this.confirmed = false;
    }


    public boolean processReservation() {
        if (startTime == null || endTime == null || !startTime.isBefore(endTime)) {
            throw new IllegalArgumentException();
        }
        if (spot == null || spot.getStatus() != ParkingSpot.Status.AVAILABLE) {
            throw new IllegalStateException();
        }
        confirmed = true;
        spot.setStatus(ParkingSpot.Status.RESERVED);
        return true;
    }

    public boolean isConfirmed() { return confirmed; }
}

class ParkingSpot {
    enum Status { AVAILABLE, RESERVED, OUT_OF_SERVICE }

    private final int spotId;
    private Status status;

    public ParkingSpot(int spotId, Status status) {
        this.spotId = spotId;
        this.status = status;
    }
    public int getSpotId() { return spotId; }
    public Status getStatus() { return status; }
    public void setStatus(Status status) { this.status = status; }
}
```

To verify if the Reservation function works as expected, we need to create Junit test class to test the method processReservation() with:

- **3 test cases:**
  - Case 1: When startTime >= endTime, the method should throw IllegalArgumentException
  - Case 2: When the parking spot is not AVAILABLE, the method should throw IllegalStateException
  - Case 3: The method returns true and the reservation becomes confirmed
- **5 annotations:**
  - @BeforeEach
  - @AfterEach
  - @Test
  - @DisplayName
  - @Timeout()

Please fill in the following code to finish the test class:

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

```
class ReservationTest {

    private Reservation res;
    private ParkingSpot spot;
    private LocalDateTime now;

    Todo:Initialize a fixed LocalDateTime value (e.g., a baseline "now")

    Todo:Reset objects used in the test

    Todo: Test case 1 with proper name

    Todo: Test case 2 with proper name

    Todo: Test case 3 with proper name and finish within 100 millisecond

    }
}
```

Hints:
You may use screenshot for the solution (the filled code and results) in the submission.

Q6. Risk Management (15 marks)
Identify **four** key risks relevant to the Smart Parking System (SPS) project.

(a) For each risk:
- Classify the risk as Project, Product, or Business risk
- State the risk source (e.g., people, technology, requirement, external dependency, usability)
- Assess the likelihood and seriousness (Low / Medium / High)

(b) Based on your assessment, select the **two** highest-priority risks. Justify why these two require the most attention compared with the others.

(c) For each of the two prioritised risks, propose an appropriate risk response plan using one of the strategies taught in the module (avoidance, minimisation, contingency, acceptance). Provide a brief justification for why your chosen strategy is suitable for that risk.

Hints:
1. *Consider risks across technical, organizational, and user-related areas of the SPS.*
2. *Prioritization should be based on a combination of likelihood and seriousness, not just one factor.*
3. *Your response plan should align logically with the nature of the risk. For example, not all risks should be "avoided"; some may be better minimized or handled through a contingency plan.*

Q7. Continuous Improvement in the SPS Project (15 marks)

The Smart Parking System project is planned as an evolving system that will continue to improve after its initial release.

(a) Select **two** Continuous Improvement strategies taught in this module (e.g., Scrum retrospectives, Kanban visualisation, refactoring, CI/CD pipeline, etc.). For each selected strategy:

- Describe how it would be applied in the SPS project
- Explain how it would support improvement in system quality, team performance, or user satisfaction

(b) Propose **three** suitable metrics for evaluating the effectiveness of continuous improvement in the SPS project. For each metric, briefly justify why it is relevant in the context/application of SPS (e.g., Cycle Time, Defect Density, Deployment Frequency, Customer Satisfaction Score, Lead Time for Changes.)

Hints:
1. *Strategies should be applied to the SPS context, not described in general terms.*
2. *Metrics should be measurable and directly linked to improvement goals (e.g., speed, quality, reliability, user experience).*

Section 3 Marking Rubrics

Question	Criteria
Q1. Class Diagram (15 marks)	• Includes the required core classes (User, ParkingSpot, Reservation, Payment, AdminManagement) • Shows clear and appropriate attributes, methods, and relationships with correct multiplicities • Diagram is clear, logical, well-structured, and easy to understand
Q2. Software Design Concepts (15 marks)	• Select at least 3 relevant software design concepts (e.g., abstraction, encapsulation, cohesion/coupling, modularity, etc.) • Explain them using your own class diagram rather than textbook definitions • Show how applying these concepts improves maintainability and scalability
Q3. Architecture Selection (15 marks)	• Choose one architecture (Layered, MVC, or Client-Server) and provide a structural diagram • Justify why this architecture is suitable for the SPS — relate to system needs rather than listing

	generic advantages • Identify at least one limitation of the chosen architecture
Q4. Test Case Design (10 marks)	• Provide 2 valid, 2 invalid, and 1 boundary test • Each test clearly states: objective, input type, input value, expected output • Clear formatting, preferably in a table
Q5. JUnit Testing (15 marks)	• Use all 5 required annotations: @BeforeEach, @AfterEach, @Test, @DisplayName, @Timeout • Cover 3 test situations: invalid time, unavailable parking spot, successful reservation • Assertions correctly check both exception cases and successful outcomes
Q6. Risk Management (15 marks)	• Identify 4 realistic SPS-related risks and classify them (Project / Product / Business) • State the source of each risk + likelihood & severity • Choose the top 2 high-priority risks, justify why, and propose suitable response strategies (avoid / minimise / contingency / accept)
Q7. Continuous Improvement (15 marks)	• Select two continuous improvement strategies (e.g., Retrospective, Kanban, Refactoring, CI/CD, etc.), and explain how they would be applied in SPS • Show how they improve system quality, team performance, or user satisfaction • Provide 3 measurable metrics and explain why they are suitable for SPS

==================End of the CW2 Task Sheet=====================

=====Use the Following Template for Submission=====

XI'AN JIAOTONG-LIVERPOOL UNIVERSITY
西 交 利 物 浦 大 学
COURSEWORK SUBMISSION COVER Page

Group Number				
Students' ID Number				
Module Code				
Assignment Title				
Submission Deadline				

By uploading this coursework submission with this cover page, we certify the following:

❖ We have read and understood the definitions of collusion, copying, plagiarism, and dishonest use of data as outlined in the Academic Integrity Policy of Xi'an Jiaotong- Liverpool University.

❖ This work is **entirely** our own, original work produced specifically for this assignment. It does not misrepresent the work of another group or institution as our own.

❖ This work is not the product of unauthorized collaboration between ourselves and others.

❖ This work has not been shared wholly or in part outside of the group. Each member has performed the duty to protect the submission until the feedback is released.

❖ It is a submission that has not been previously published, or submitted to any modules.

❖ Students who would like to submit the same or similar work from previous years to the current module or other modules must receive written permission from all instructors involved in advance of the assignment due date.

❖ **All** group members are **equally** and **collectively** responsible for the **entire** submission. Violations of academic integrity including failure to monitor group member contribution originality constitutes negligence.

❖ Unreported suspicious academic misconduct by one group member will be attributed to ALL members in terms of penalties. **ALL members share the responsibilities**.

❖ Use of generative **AI is strictly prohibited** for all assessments involved in this module.

We understand collusion, plagiarism, dishonest use of data, and submission of procured work are serious academic misconducts. **All** group members are held **jointly accountable** for the integrity of the **entire** submission. By uploading or submitting this cover page, we acknowledge that we are jointly subject to penalties and disciplinary actions if we are found to have committed such acts.

~~~~~~

We acknowledge that the university late submission policy will be applied if applicable.
~~~~~~

~~~~~~

Please list the ID number of any group member NOT contributing to the submission:

Please indicate whether based on your individual contributions you meet the learning outcomes (LOs) covered by this coursework and confirm your submission meets with all above requirements by signing your name, handwritten in ink, in English (Foreign students) or PINYIN (local students):

Student ID	**✓(tick) box to confirm you contributed to all cw tasks**	**✓(tick) box to confirm you meet all LOs covered by this cw**	**Signed**
	☐	☐	
	☐	☐	
	☐	☐	
	☐	☐	

Date: ……………………………………
~~~~~~

Q1. Class Diagram (15 marks).

<Your answer with the diagram >

Q2. Software Design Concepts (15 marks).

<Your answer>

Q3. Architecture Selection (15 marks)

<Your answer>

Q4. Test Case Design. (10 marks)

<Your answers:>

Q5. JUnit Testing (15 marks)

<Your answers>

Q6. Risk Management. (15 marks)

Q7. Continuous Improvement. (15 marks)

<Your answers>

Appendix (if any)

Peer review form template

CPT203 Coursework
Peer review
Individual Contribution for Group Report

Group Number: <x>

Name	**ID Number**	**Contribution (%)** The sum of this column should be 100	**Signature**

1. <my name>	<123456>	16.67 (supposing all 6 member contributing equally, each one is ~16.67)	
2.			
3.			
4.			
5.			

END