Assessment cover

Module No:	COMP5047	Module title:	Applied Software Engineering

Assessment title:	Software Engineering of a Modern Computer Application (Resit)

Due date and time**:**	09:00 14th April 2025

Estimated total time to be spent on assignment:	84 hours per student

LEARNING OUTCOMES

On successful completion of this assignment, students will be able to achieve the module's following learning outcomes (LOs):
1. Demonstrate an understanding of the role of requirements analysis and specification in software engineering and to be able to use this knowledge to create use case models and functional models of computer applications.
2. Demonstrate an understanding of the relationship between requirements and design and to be able to apply the knowledge to create structural and behavioural models of computer applications.
3. Critically evaluate and utilise design paradigms of object-oriented analysis and design, component-based design, and service-oriented design.

4. Use software modelling language such as UML and modelling tools in the context of model-driven software engineering.
5. Work in a group to apply the knowledge and skills developed in this module

Engineering Council AHEP4 LOs assessed	
C3	Select and apply appropriate computational and analytical techniques to model complex problems, recognising the limitations of the techniques employed
C5	Design solutions for complex problems that meet a combination of societal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societal, environmental and commercial matters, codes of practice and industry standards
C6	Apply an integrated or systems approach to the solution of complex problems
C14	Discuss the role of quality management systems and continuous improvement in the context of complex problems
C16	Function effectively as an individual, and as a member or leader of a team

GROUP ID:	

STUDENT NAMES

	Student Id:	**Student Name:**	**Subsystem:**
1.	**19266562**		**Service (Resit)**
2.			

3.			
4.			

Statement of Compliance *(please tick to sign)*
I declare that the work submitted is my own and that the work I submit is fully in accordance with the University regulations regarding assessments
(www.brookes.ac.uk/uniregulations/current)

RUBRIC OR EQUIVALENT:

Marking grid and marking form are available on Moodle website of the module.

FORMATIVE FEEDBACK OPPORTUNITIES

(a) Discuss your work with your practical class tutor during practical classes;
(b) Discuss your work with lecturer and/or practical class tutor in drop-in hours.

SUMMATIVE FEEDBACK DELIVERABLES

Deliverable content and standard description and criteria
Please see attached file of *COMP6030 Coursework Marking and Feedback* for feedbacks on your coursework, which include:
(a) Breakdown of marks on each assessment criterion
(b) Comments on each aspect of the assessment against assessment criteria
(c) Annotations on your submitted work

5047 Resit Coursework

Task 1: Project Management And Uses of Software Engineering Tools (10 Marks)

Because this is a resit there is no need for group project management for task 1 however it is still important to lay out exactly what this coursework is about. In regards to the engineering tools I have used github to track and update the work as we go for better tracking and transparency.The subsystem that I will be doing is CloudTables-Service as it is the one I was doing in the original coursework. CloudTables-Service enables restaurant service employees to provide customers with better and quicker service. Staff members may access real-time information regarding reservations, tables, and orders at any time thanks to its direct cloud connection with tablets and desktop computers. This subsystem is designed to simplify the process of client check-in, table assignment, food and drink order taking and meal preparation tracking for employees.

This system is designed to be easy to use during peak hours, assisting employees in efficiently attending to consumer needs. It works with other CloudTables components to improve the overall experience for all customers. Restaurant employees can focus more on offering excellent customer service and less on handling inefficient systems or paperwork by using CloudTables-Service. Additionally, it ensures that all customer requests are handled correctly, saves time, and helps minimise errors. CloudTables-Service is essential to improving the organisation, effectiveness, and customer and employee satisfaction of restaurant operations.

Task 2: Analysis and Specify Software Quality Requirements (20 Marks)

Security and Privacy Protection

Customer and restaurant data must always be shielded from unauthorised access via the CloudTables-Service subsystem. Employees should only view the data necessary for their jobs, such as reservations for tables and food orders, and they shouldn't have access to more personal information than is necessary. To guard against data theft during transfer, all information must be sent securely using encryption (such as HTTPS and SSL/TLS). Secure passwords and timeouts must be used in user login and session management to prevent unauthorised access. Customer booking information and payment history are examples of personal data that should be securely maintained via encryption and comply with privacy regulations such as the GDPR. To ensure the system remains secure, regular security testing, including vulnerability scans, should be carried out.

Performance

CloudTables-Service needs to operate efficiently, even during peak restaurant hours. In most situations, tasks like assigning a table, checking in a customer, and submitting an order to the kitchen should take less than two seconds. Even when the restaurant is packed, the system should load all pages and menus in three seconds or less. Important functions like ordering and table distribution should still function properly even with a slow internet connection. In order to enhance customer service during peak hours, the system must be usable by the service professionals without any delays or waiting.

Reliability

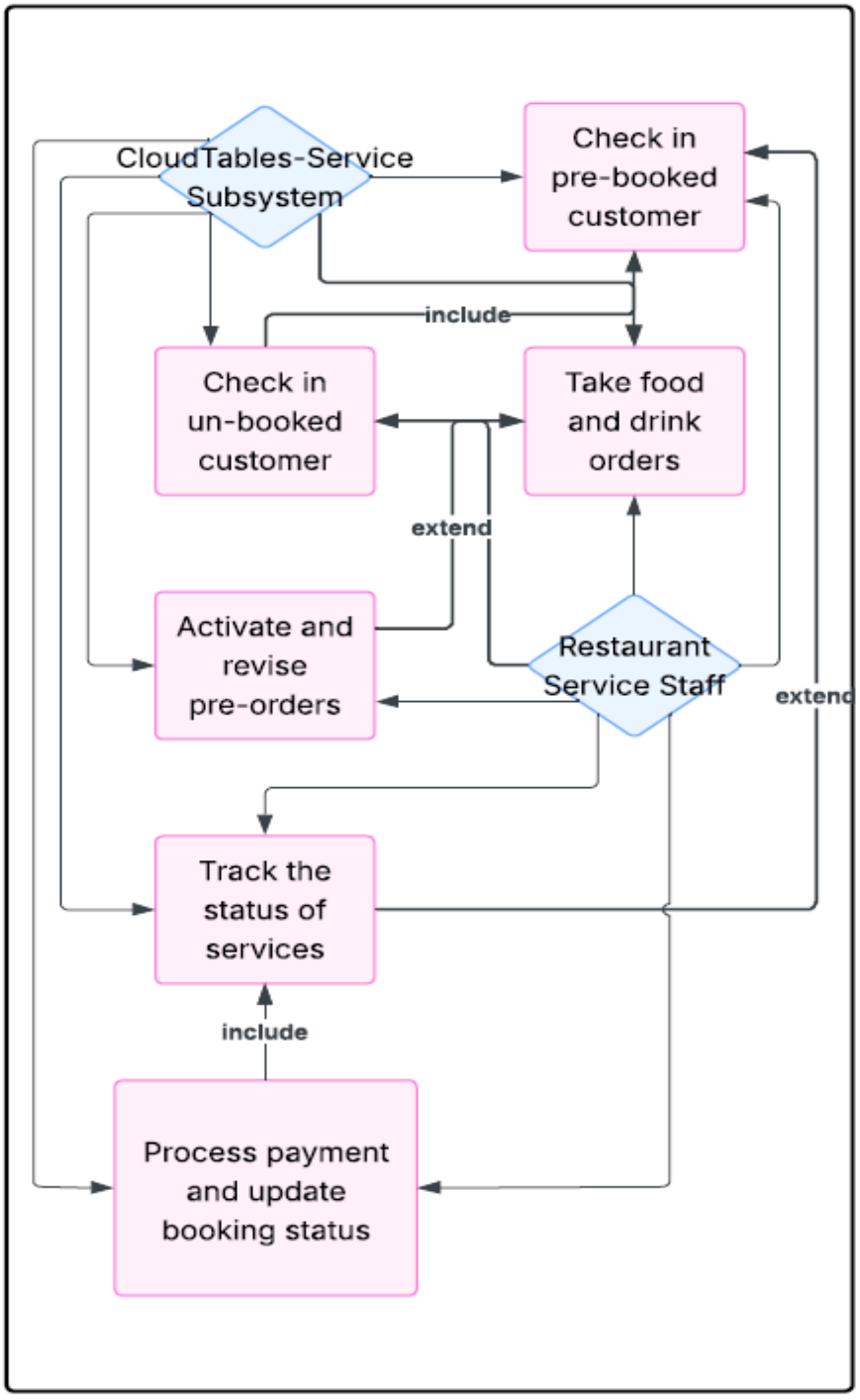
When employees use the CloudTables-Service subsystem, it must function properly and be accessible. For employees to be able to rely on it throughout all open hours, it should have a minimum uptime of 99.9%. If something goes wrong, such as a crash or network error, the system will be able to recover fast with the help of backup systems and retry choices. To avoid data loss, order and booking information should be securely stored, even in the case of an unexpected app closure or internet outage. Order tracking, table status updates, and payment processing are examples of features that must constantly display accurate data. To ensure that the system reacts appropriately to various issues, real-world scenarios will be used for testing.

Scalability

Over time, CloudTables-Service will be able to accommodate more customers without experiencing a decrease in speed or functionality. The system will continue to function just as well whether more employees begin using it or as the restaurant expands in size. Bookings and orders should be handled instantly by the system even if a restaurant adds more tables, expands its menu, or receives more patrons. it should also be able to accommodate numerous restaurants using the system simultaneously in various places. When more server resources or user traffic are required, cloud-based architecture will be deployed to facilitate easy scaling.

Task 3: Specification and Modelling Software Functional Requirements (25 Marks)

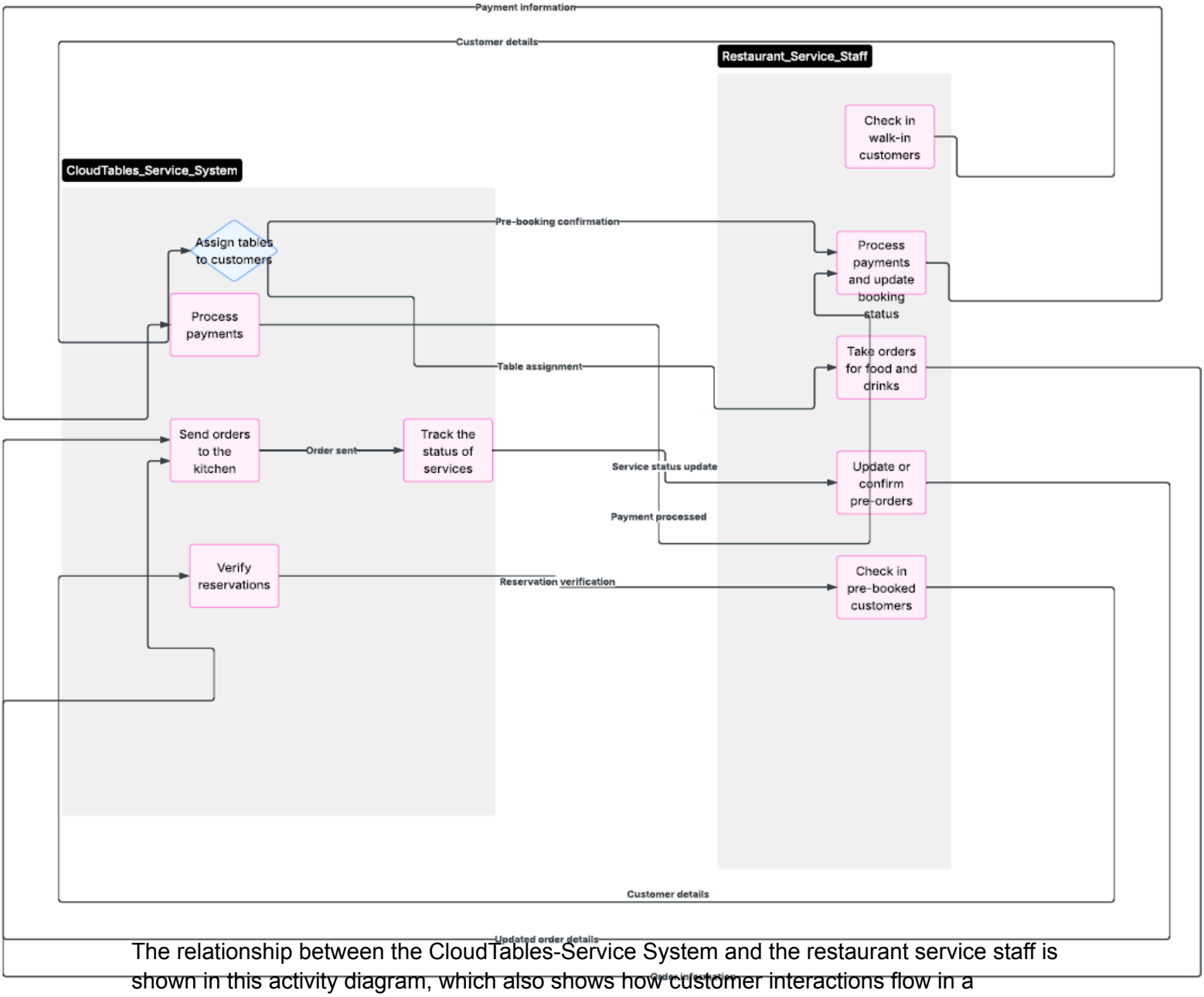
Use Case Model (10 Marks):



A key component of the restaurant table booking system, the CloudTables-Service subsystem is made to make client interactions easier for the wait staff. The restaurant employees are the main player in this use case diagram. They engage with several

subsystem functionalities to effectively handle customer interactions. Handling walk-in clients, accepting orders for food and beverages, checking in pre-booked customers, processing payments, and updating booking status are just a few of the use cases that the system offers. The use case diagram accurately defines the scope and functionality of the subsystem by using standard UML notation, which includes distinct system boundaries and relationships like "include" and "extend." Additionally, the system enables personnel to monitor the status of table reservations in real time and follow the progress of food preparation.

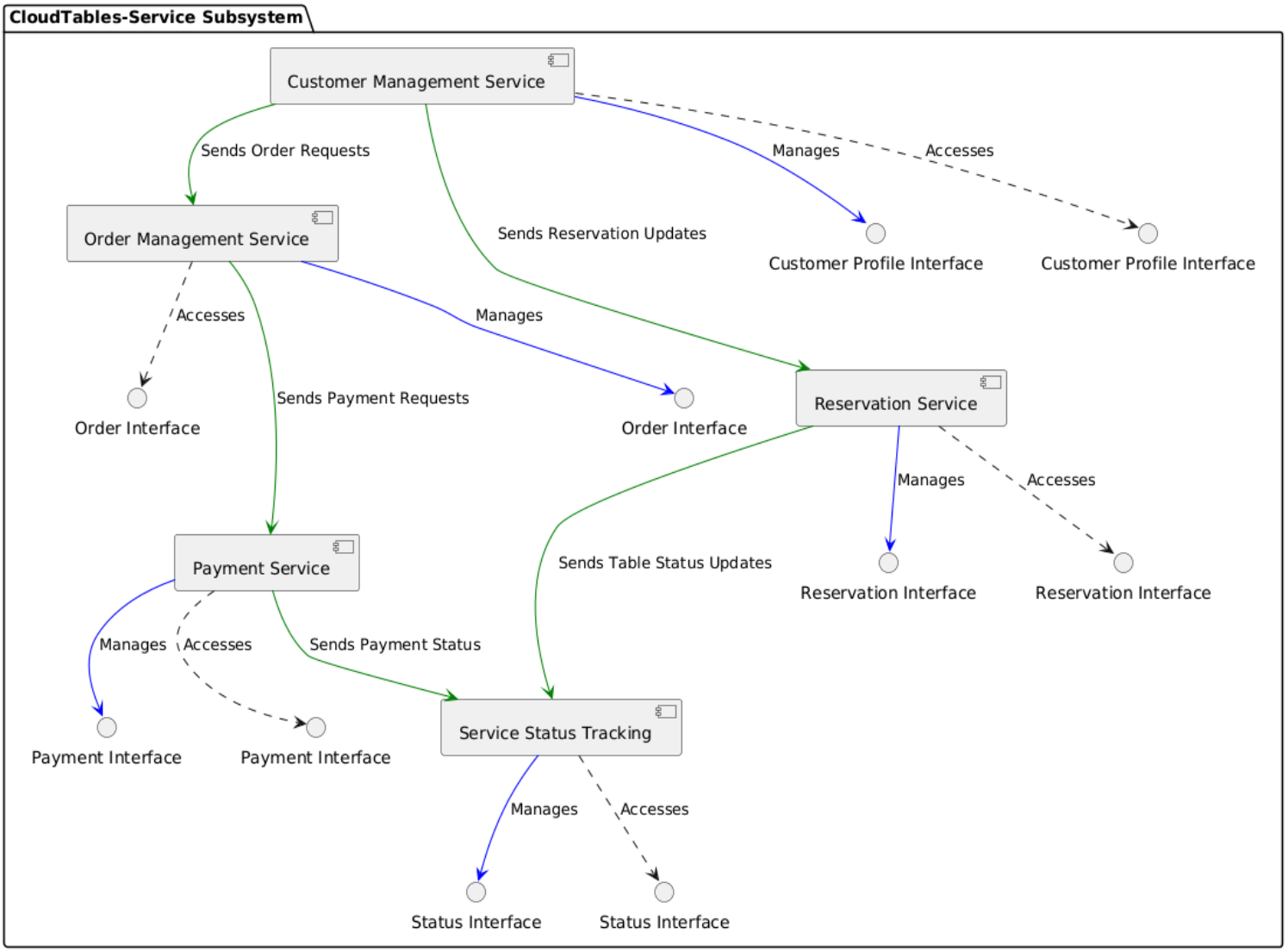
Activity Model (15 Marks):



The relationship between the CloudTables-Service System and the restaurant service staff is shown in this activity diagram, which also shows how customer interactions flow in a restaurant. It divides the duties of the system and the employees into two swimlanes.

Customer check-in, order taking, payment processing, and booking progress updates are all included in the Staff swimlane. Verifying reservations, allocating tables, submitting orders to the kitchen, monitoring service statuses, and handling payments are all included in the CloudTables-Service System swimlane. Object flow is used in the diagram to clarify how the system handles data. In accordance with the requirements description and use case model, it is intended to be understandable, and working.

Task 4: Software Architectural Design (15 Marks)

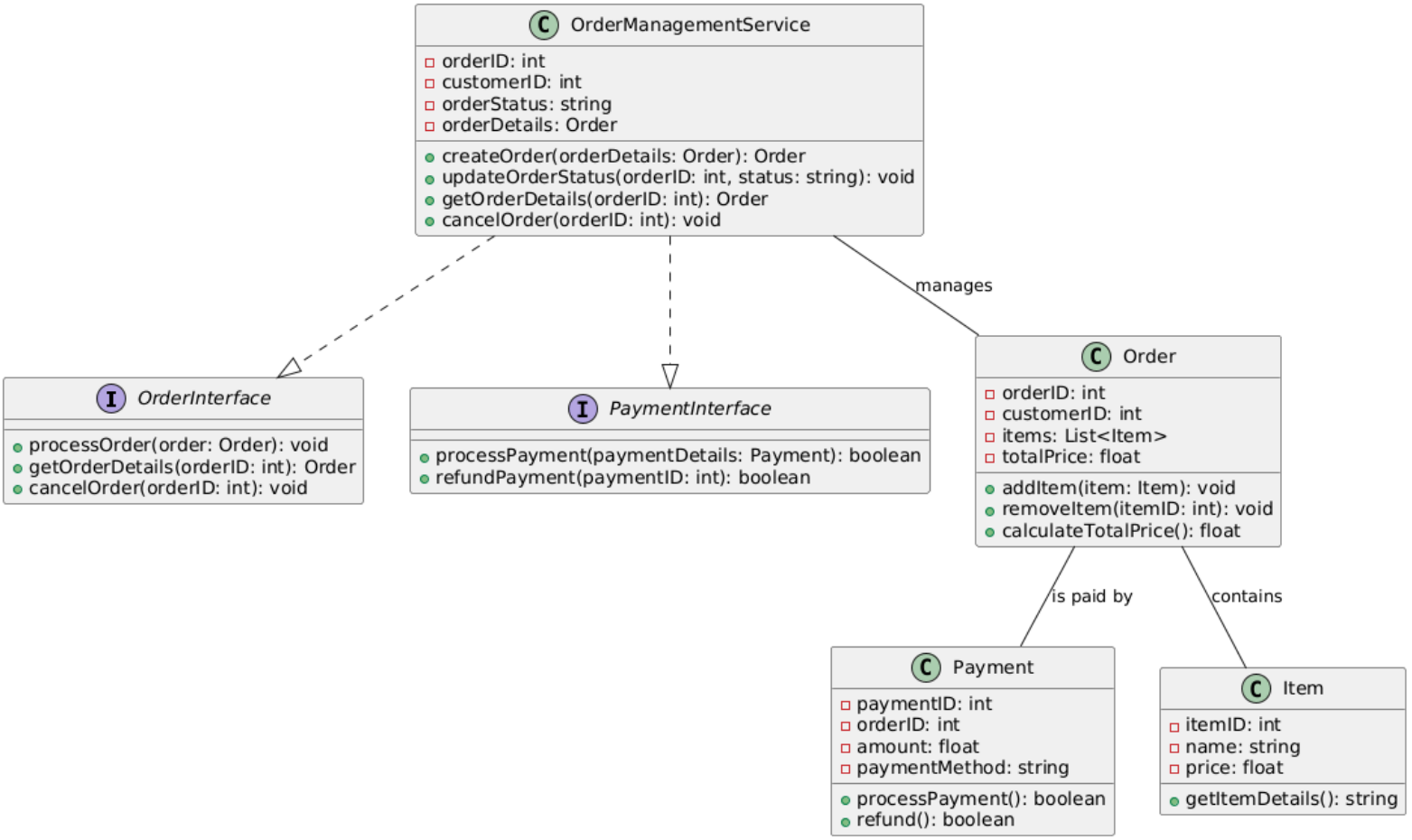


Customer Management Service , Order Management Service, Reservation Service, Payment Service, and Service Status Tracking are the five main microservices that make up the CloudTables-Service Subsystem's architectural design, as shown in the UML component

diagram. Communication between services is ensured by representing each one as a separate component with distinct interfaces. Each component in the diagram communicates with the others through straightforward interfaces. The arrows in the diagram show the relationships between various services which show interaction and data flow. It links to the coursework specification as the diagram supports visual representation, and the design complies with the functional requirements and overall system architecture. The diagram is in line with the system architecture and functional requirements since it follows the proper UML syntax .

Task 5: Software Detailed Design (30 Marks)

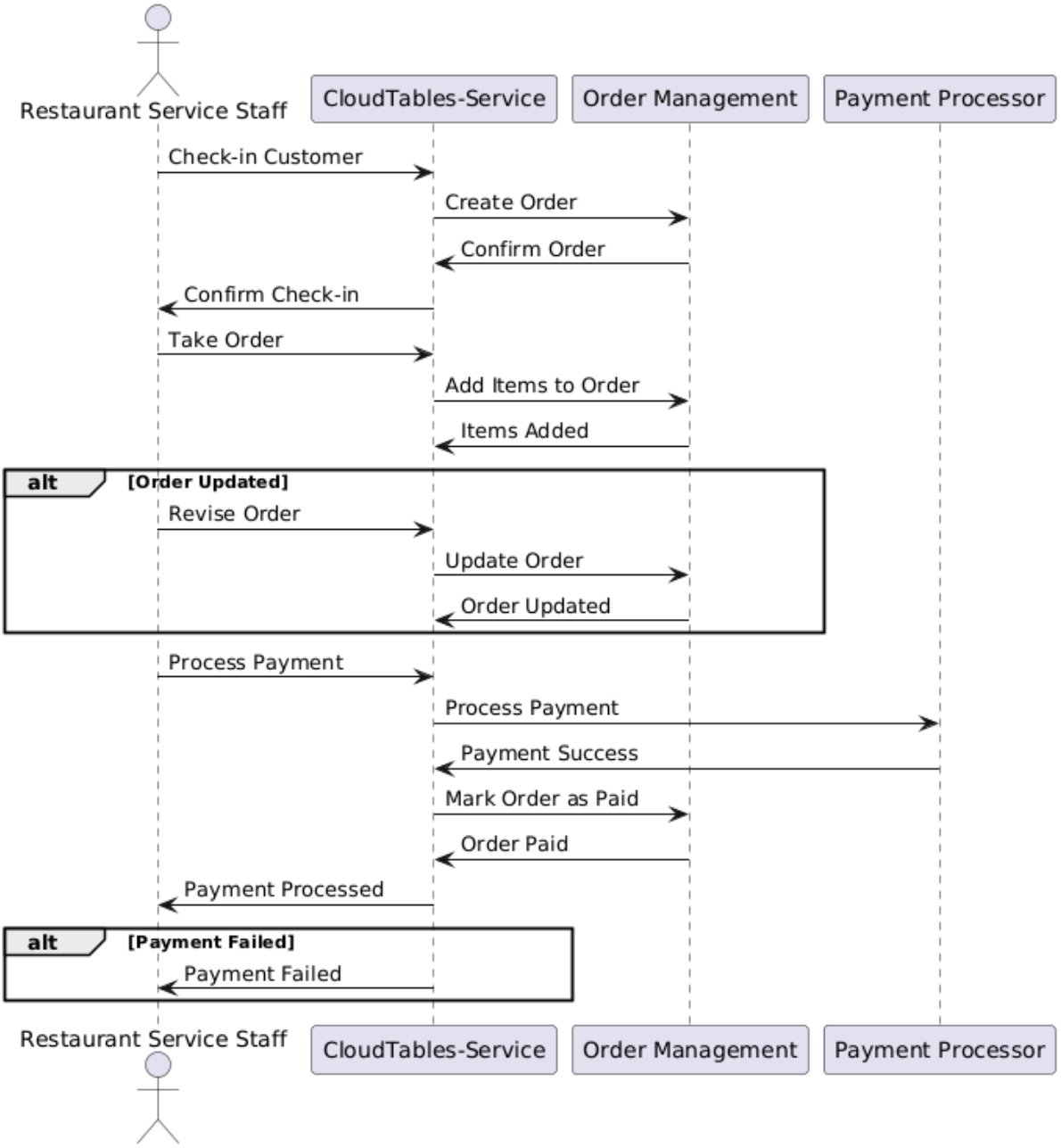
Structural Model (15 Marks):



For the "CloudTables-Service" subsystem, the UML class diagram is a comprehensive structural model created to meet requirements of the specification. Each of its three main microservices which are Order Service, Payment Service, and Table Service has a primary purpose. The required classes for every microservice are shown in the figure, and methods such as placeOrder(), processPayment(), and checkInCustomer() demonstrate the functionality of each microservice. In addition, the diagram uses proper UML notation to show the interactions between the microservices and their interfaces. Making sure that the

microservices architecture and functional specifications complement the case study's goals, the diagram conforms to the architectural design stated in Task 4. This guarantees that the subsystem's structural design complies with the functional standards and microservices architecture.

Behaviour Model (15 Marks):



The connections between the "Restaurant Service Staff," "CloudTables-Service," "Order Management," and "Payment Processor" components of a restaurant table booking and order management system are shown in the UML sequence diagram. During essential

processes like order taking, customer check-ins, modifications, and payment processing, it records the system's dynamic behaviour. According to the diagram, an order is created in the "Order Management" system when the staff checks the customer in. The staff is informed that the check-in was successful once it has been verified. After receiving the customer's order, the staff adds items to the order management system. The staff can amend the order if it is changed, and the system verifies this. The process of payment then proceeds, with the payment processor managing the transaction and informing the employees of its completion. Following proper UML sequence diagram syntax and semantics, staying consistent with architectural design, and expressing interactions with message parameters are all ways that the diagram satisfies requirements.

Link to github repository:

https://github.com/ZOAkhtar/COMP5047-Resit