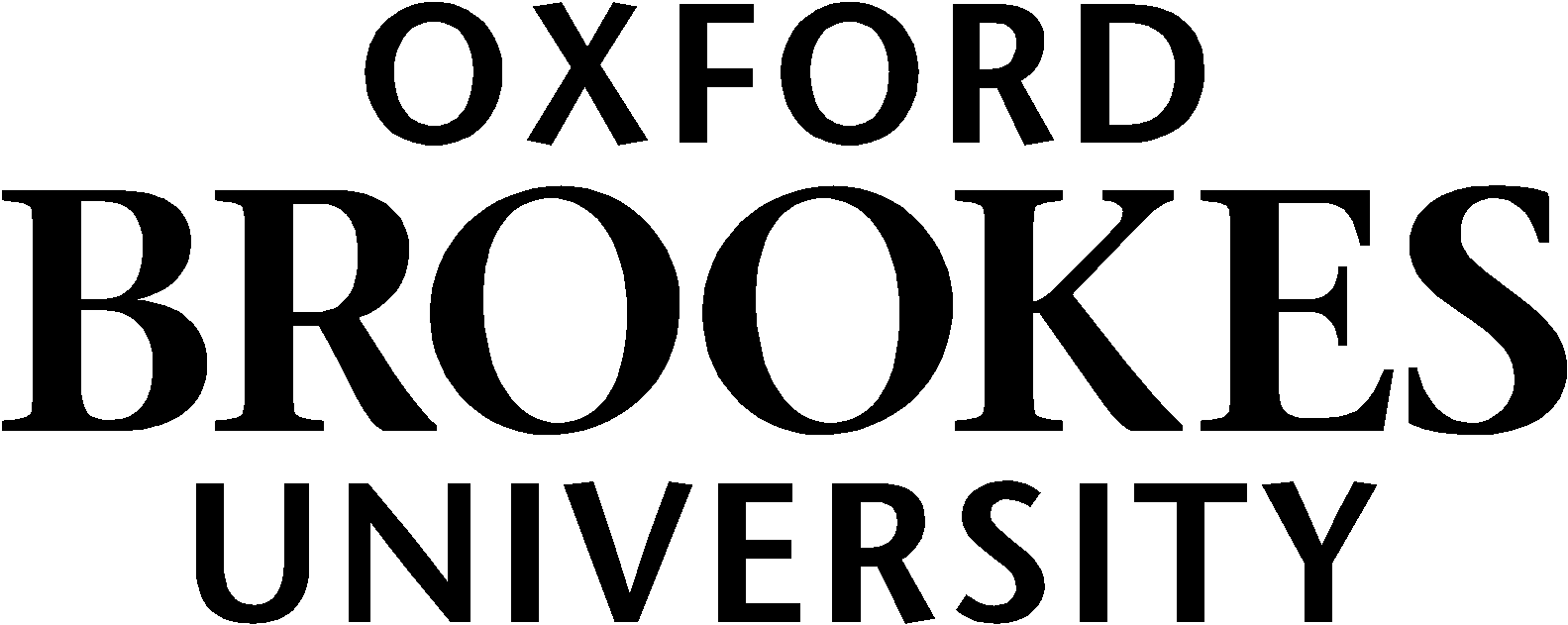
** 7**

**Assessment cover**

| Module No: | COMP5047 | Module title: | Applied Software Engineering |
| --- | --- | --- | --- |

| Assessment title: | Resit Coursework - Software Engineering of a Modern Computer Application |
| --- | --- |

| Due date and time**:** | 9:00am, 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. |
| 1. 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. |
| 1. Critically evaluate and utilise design paradigms of object-oriented analysis and design, component-based design, and service-oriented design. |
| 1. Use software modelling language such as UML and modelling tools in the context of model-driven software engineering. |
| 1. 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 |

**Student Name: Student Id: Subsystem:**

**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*](http://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**

| 1. Discuss your work with your practical class tutor during practical classes; 2. 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: |
| 1. Breakdown of marks on each assessment criterion |
| 1. Comments on each aspect of the assessment against assessment criteria |
| 1. Annotations on your submitted work |

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

### In this task, you will work as a requirements analyst to produce a document that defines the quality requirements

### on your subsystem. The definition of quality requirements should clearly specify the requirements on the

### following quality attributes for each of the functions of the subsystem listed in the Case Study Description.

Task 2: Specification of Quality Requirements

(a) Security and Privacy The CloudTables-Manager subsystem shall implement OAuth 2.0 and JWT (JSON Web Tokens) for secure and verifiable user authentication, ensuring only authorized restaurant managers can access and modify sensitive operational data. Role-Based Access Control (RBAC) will be applied to restrict functionality based on roles, ensuring that only managers can perform data modification, while staff can access information in read-only mode. All communication will be encrypted using TLS 1.2 or higher, and sensitive data at rest will be protected using AES-256 encryption. The system will comply with GDPR, supporting data minimization, consent tracking, and secure deletion protocols. To maintain security standards, the system will undergo quarterly penetration tests and security audits, and store audit logs of all user actions for traceability. These logs will be encrypted, access-controlled, and retained for at least 12 months. All security configurations and compliance will be reviewed and validated before deployment and at regular intervals.

(b) Performance The CloudTables-Manager subsystem shall respond to user actions—such as updating menus, modifying table availability, and generating reports—within 2 seconds for 95% of requests under standard operating load. The system will be stress-tested to support a minimum of 100 concurrent users without noticeable degradation in response times. To enhance performance, Redis will be used for caching frequently accessed data, reducing read load on the primary database. SQL queries will be optimized through indexing, and monthly performance diagnostics will identify and resolve latency issues. All performance-related thresholds, including concurrent user load and response time limits, will be monitored using automated tools, with alerts configured for violations. The system will be verified through load testing during pre-deployment phases and monitored post-deployment for continuous assurance.

(c) Reliability The subsystem shall maintain 99.9% system availability, equating to no more than 8.77 hours of downtime per year. Critical services shall recover from failure within a 2-minute Mean Time to Recovery (MTTR), and high-priority functions shall achieve an RTO (Recovery Time Objective) of 30 seconds. Automatic failover mechanisms will be implemented to restore service continuity in case of hardware or software failure. Data integrity will be maintained via transaction validation, real-time replication, and hourly backups stored in geographically distributed cloud locations. Quarterly disaster recovery testing will validate recovery readiness. A system monitoring service will track health metrics, and any anomaly will trigger automatic failover or recovery routines. Error logs will be stored and reviewed weekly for proactive issue resolution.

(d) Scalability The CloudTables-Manager subsystem will support horizontal scalability by dynamically provisioning additional application servers when CPU or memory usage exceeds 80%, using AWS Auto Scaling and Elastic Load Balancing. Database scalability will be handled using sharding and partitioning, ensuring that high volumes of tenant data can be distributed across multiple nodes without compromising access speed. The system will be tested under simulated loads of up to 1,000 concurrent users to verify scaling performance and stability. All scalability configurations will be monitored continuously, and scaling decisions will be managed by cloud-based orchestration tools (e.g., AWS CloudWatch). Scaling actions will be logged, and post-scaling performance will be validated through metrics like response time and throughput consistency.

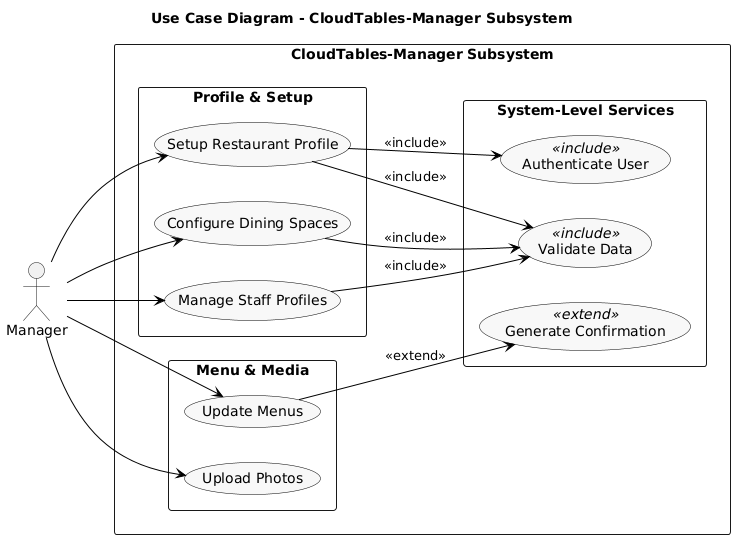
**Task 3: Specification and Modelling Software Functional Requirements (25 Marks)**

**In this task, you will work as a requirements analyst to produce a functional model of the software system to be**

**developed in UML using a software modelling tool. The UML model should contain the following types of**

**diagrams.**

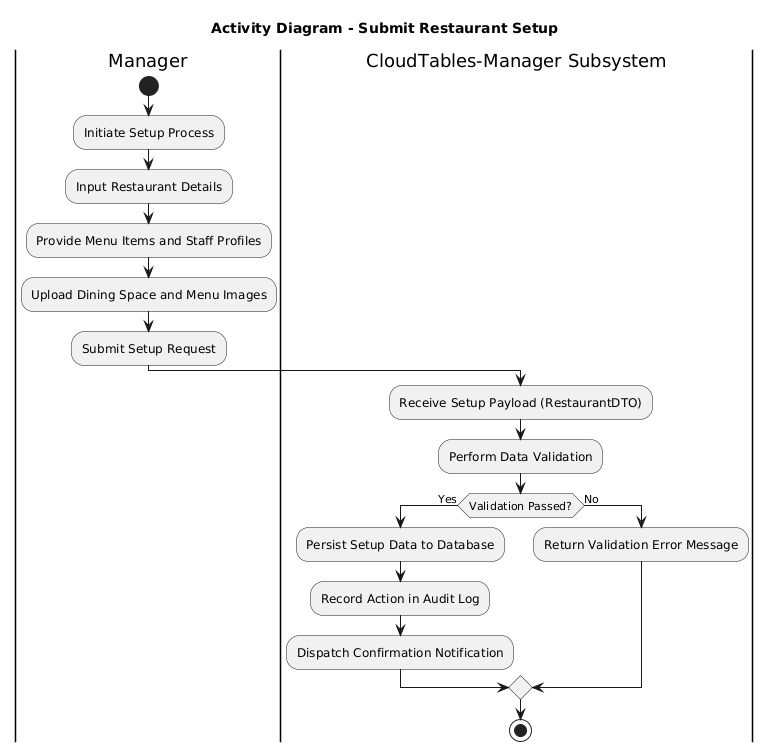
**(a) Use Case Model (10 Marks): You should develop one Use Case Diagram to define the use cases of the**

****

**§**

**(b) Activity Model (15 Marks): You should select one use case of your subsystem to produce one Activity**

**1 No coding is required in this coursework.**

****

## 

## 

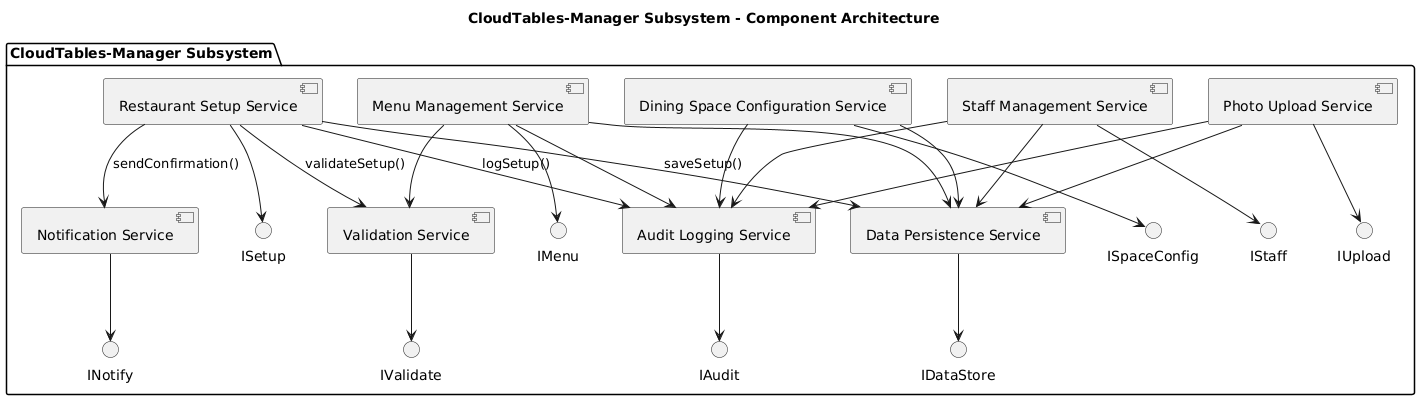
## 

## 

## 

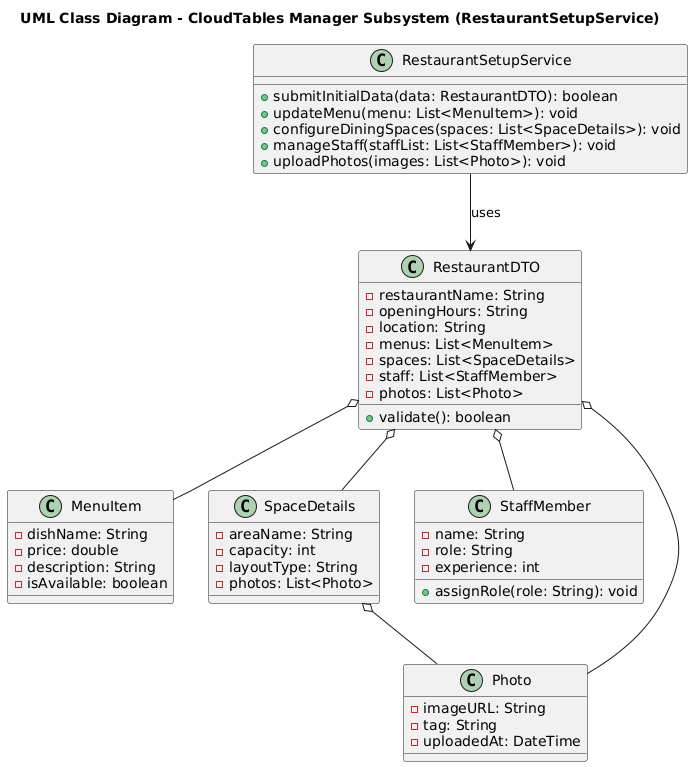
## 

## Task 4: Software Architectural Design (15 Marks)

****

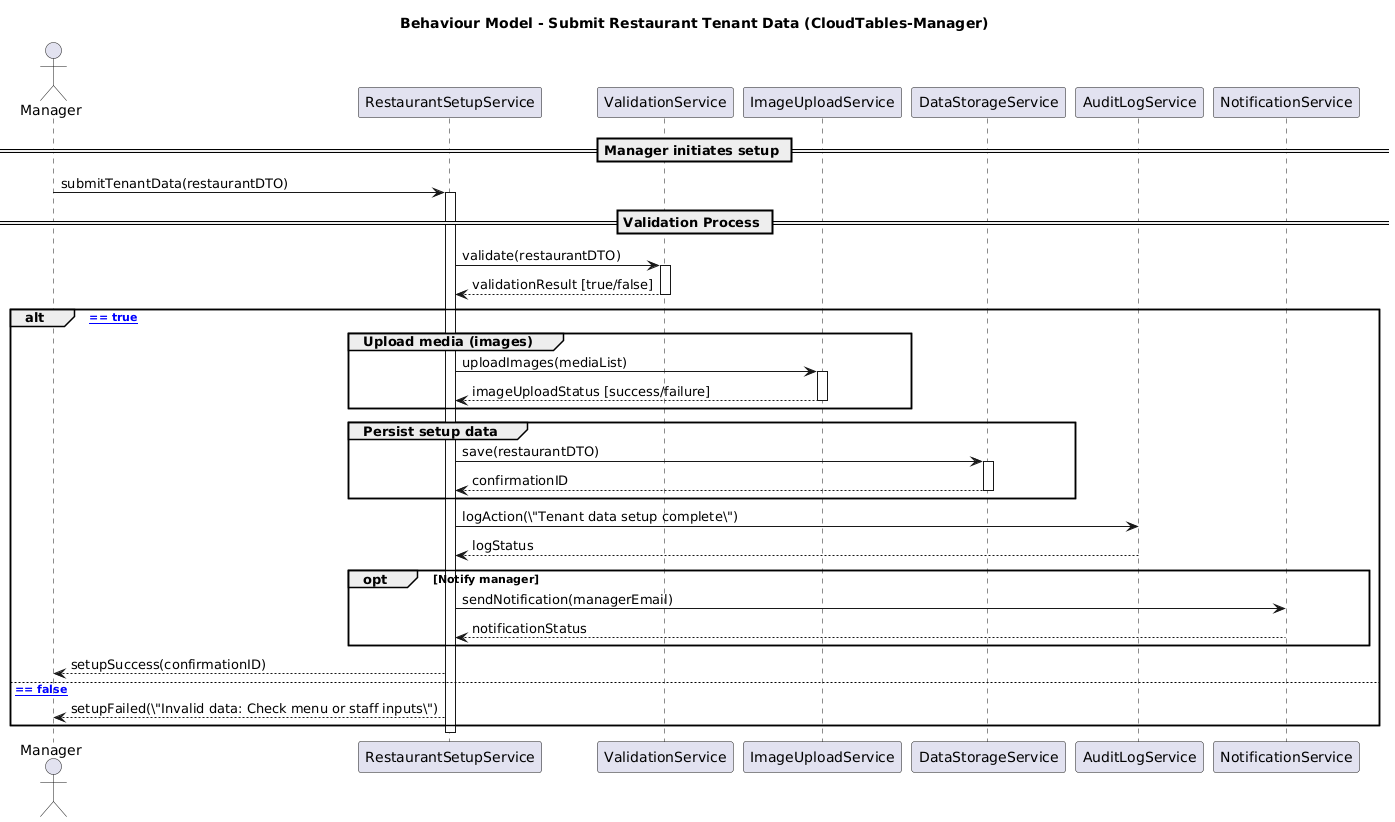
The CloudTables-Manager subsystem adopts a microservices-based architecture, with each service designed to perform a distinct and well-defined function. For example, the RestaurantSetupService is responsible for managing the initial onboarding of restaurant data, while the RestaurantUpdateService facilitates updates to existing information. The ValidationService ensures that all submitted data complies with predefined standards, and the DataStorageServicesecurely persists this information. In addition, the AuditService logs significant events for traceability purposes, and the NotificationService is responsible for delivering relevant system alerts to users when required.Each service communicates via explicitly defined interfaces, such as ISetupService and IUpdateService, which specify the operations offered and consumed by the components. This architectural approach encourages loose coupling and modularity, enabling individual services to evolve independently and reducing interdependencies. The overall workflow—from data submission by the restaurant manager, through validation, storage, and logging, to optional notification—ensures that the system is both scalable and maintainable, while remaining tightly aligned with the functional use cases specified earlier in this document.

## Task 5: Software Detailed Design (30 Marks)

**5.1**

The selected component for this structural model is the RestaurantSetupService, which is responsible for managing the initial onboarding of restaurant tenant data within the CloudTables-Manager subsystem. This microservice exposes the submitInitialData() method via the ISetupService interface, enabling restaurant managers to provide essential setup information, including menus, space configurations, and staff profiles.The class diagram illustrates a modular and object-oriented design. At the core is the RestaurantSetupService class, which collaborates with four supporting service classes: ValidationService, DataStorageService, AuditService, and NotificationService. Each of these services offers a focused method—such as validate(), save(), log(), and notify()—to perform validation, persistence, auditing, and user notification, respectively. This structure supports the principle of single responsibility and ensures service encapsulation.The RestaurantDTO (Data Transfer Object) class encapsulates all onboarding data and is composed of several related entity classes, including MenuItem, SpaceDetails, and StaffMember. These classes hold relevant attributes such as dish names, prices, layout images, staff qualifications, and experience, ensuring the completeness and clarity of the restaurant’s operational data.This structural model directly supports the “Setup Restaurant Tenant Data” use case defined in the functional specification. It clearly demonstrates how data is collected, validated, and stored, while maintaining audit trails and optional notifications for system transparency. Overall, the model is consistent with microservice principles and provides a solid foundation for reliable, maintainable, and scalable implementation.

**5.2**

****

This UML sequence diagram illustrates the dynamic behaviour of the RestaurantSetupService and is directly aligned with the structural model presented in Task 5.1. It captures the internal orchestration required to process initial onboarding data submitted by a restaurant manager. Upon receiving the request via the submitInitialData() method, the service delegates input validation to the ValidationService. If the data passes validation, it is persisted by the DataStorageService, an audit entry is created by the AuditService, and—where applicable—a confirmation notification is dispatched via the NotificationService.In the event of invalid data, the process is terminated early, with failure logging initiated and appropriate feedback returned to the manager. The diagram leverages advanced UML sequence modelling constructs, including alt for conditional branching, opt for optional messaging, and clear message signatures with parameters and return values. This ensures that the behaviour model is complete, syntactically correct, and semantically consistent with both the architectural design and the subsystem’s functional requirements.