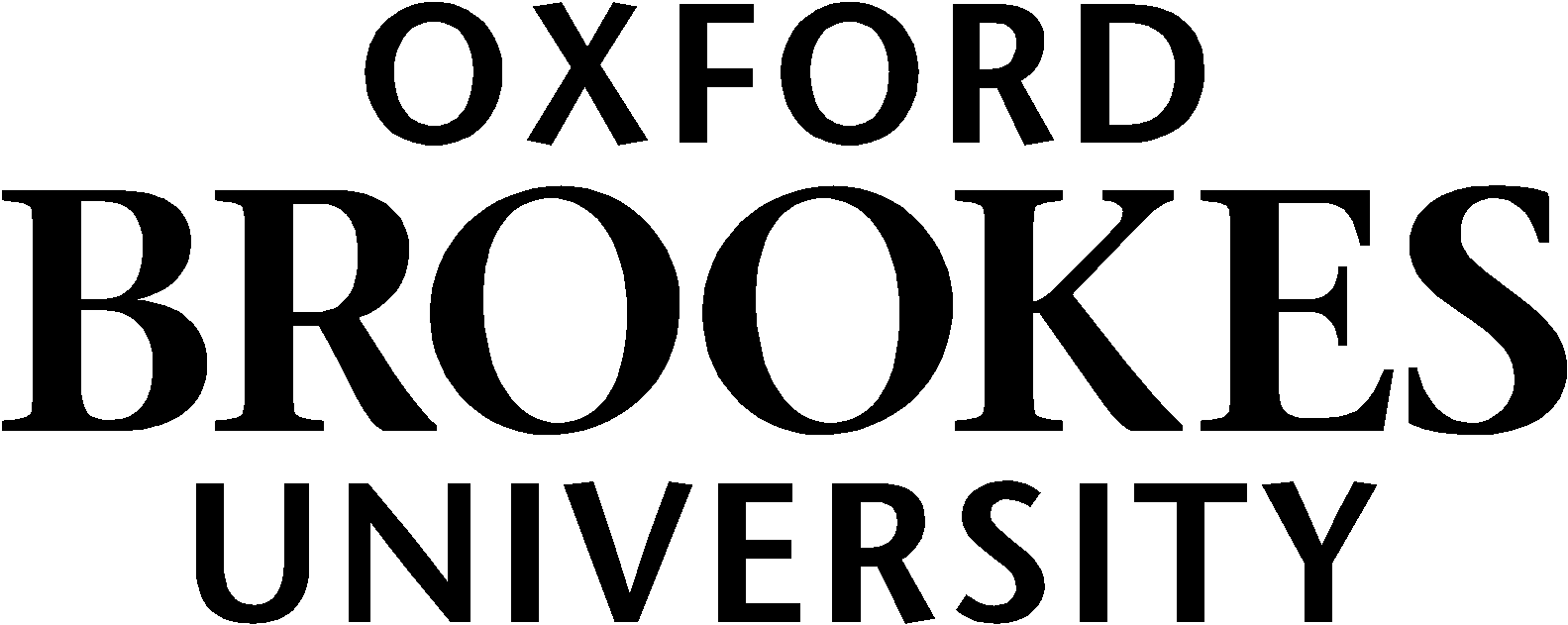
** 7**

**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.

### Security and Privacy

The CloudTables-Manager subsystem must adhere to stringent security and privacy standards to protect sensitive restaurant data. Only authenticated restaurant managers should have access to tenant setup functionalities. This is enforced through secure login protocols using OAuth 2.0 and session management with JWT tokens. Role-Based Access Control (RBAC) is implemented to restrict system access based on predefined roles, ensuring that only authorized personnel can perform critical operations such as managing menus or uploading staff information. All data must be encrypted in transit using TLS 1.3 and stored using AES-256 encryption standards. Furthermore, the system must retain audit logs for a minimum of 12 months to support data governance and comply with UK GDPR regulations. These mechanisms are testable via penetration testing, encryption verification, and access log audits.

Performance

From a managerial standpoint, system responsiveness is vital for operational efficiency during initial tenant configuration. The CloudTables-Manager subsystem must deliver fast interactions across all user inputs, including form submissions, image uploads, and data validations. Specifically, 95% of setup-related operations should complete within 2.5 seconds, even under simultaneous access by 100 active users. To meet this performance benchmark, asynchronous data handling techniques will be employed, including the use of job queues for image uploads and indexed queries for fast data retrieval. CDN caching strategies are applied to accelerate static content delivery. Performance benchmarks will be validated through stress and load testing using tools such as Apache JMeter, and monitored in real-time through infrastructure monitoring platforms like AWS CloudWatch.

### Reliability

Ensuring data continuity is critical when restaurant managers are entering essential configuration data. The subsystem must maintain a 99.9% uptime SLA and incorporate autosave functionality that triggers at regular intervals during form entry, thus safeguarding against data loss from network interruptions or session expiry. The back-end architecture is deployed on a redundant microservice infrastructure with automatic failover and replication-enabled databases. Moreover, all tenant setup data must be backed up using a combination of hourly incremental backups and daily full backups, which are routinely tested for recoverability. These measures collectively contribute to high availability and minimal disruption. Key reliability metrics such as MTTR (Mean Time To Recovery) are monitored to ensure recovery occurs within five minutes in the event of a fault.

### Scalability:

To accommodate growth in the number of restaurants adopting the platform, the subsystem must support scalable deployment. All backend services—including RestaurantSetupService, ImageUploadService, and ValidationService—are designed as independent, containerized microservices orchestrated via Kubernetes. These components will automatically scale out when resource usage exceeds thresholds (e.g., 75% CPU or 70% memory). For example, images submitted by restaurant managers will be handled via asynchronous queues, thereby decoupling media processing from the main application and enabling non-blocking scalability. Additionally, metrics are collected through Prometheus to assess system responsiveness under scaling conditions. The system must demonstrate stable behaviour and performance during simulated scenarios of at least 500 concurrent tenant setups, ensuring elasticity and reliability under growing demand.

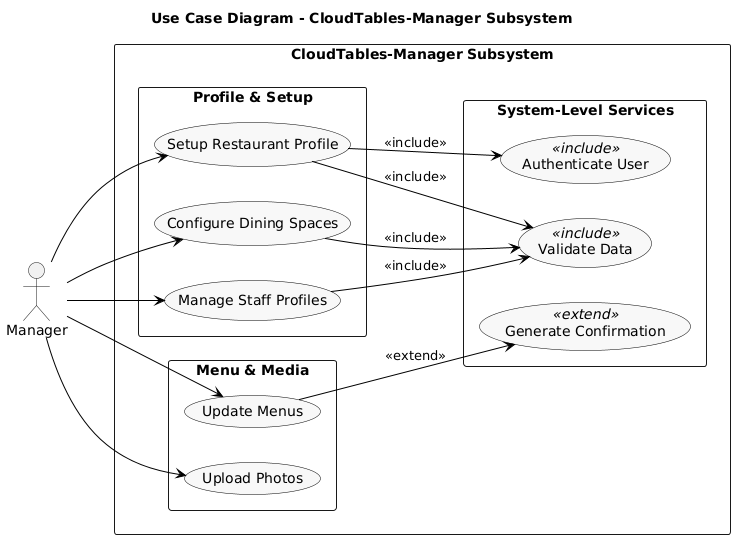
**Task 3: Specification and Modelling Software Functional Requirements (25 Marks)**

**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**

## 

## 

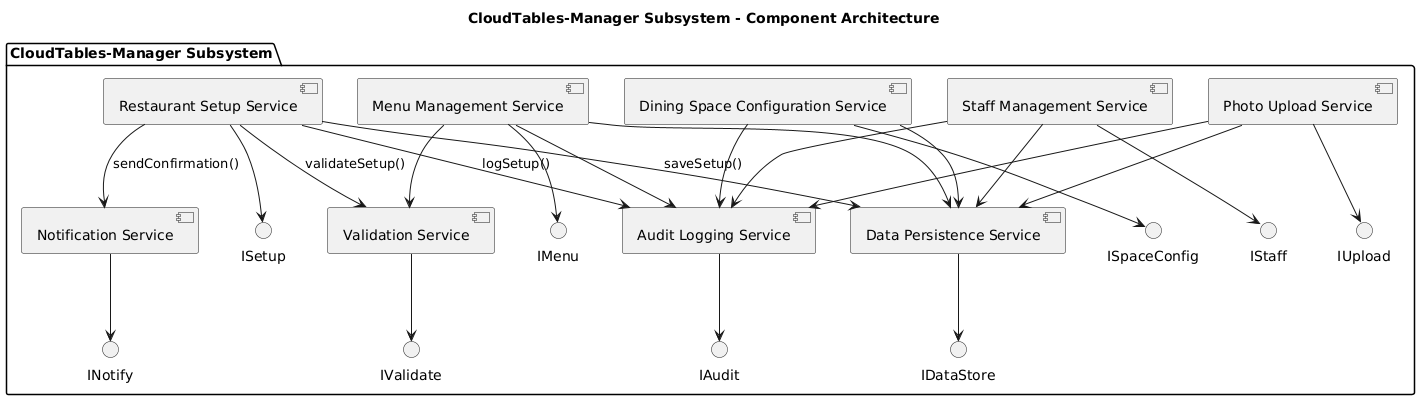
## 

## 

## 

## 

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

****

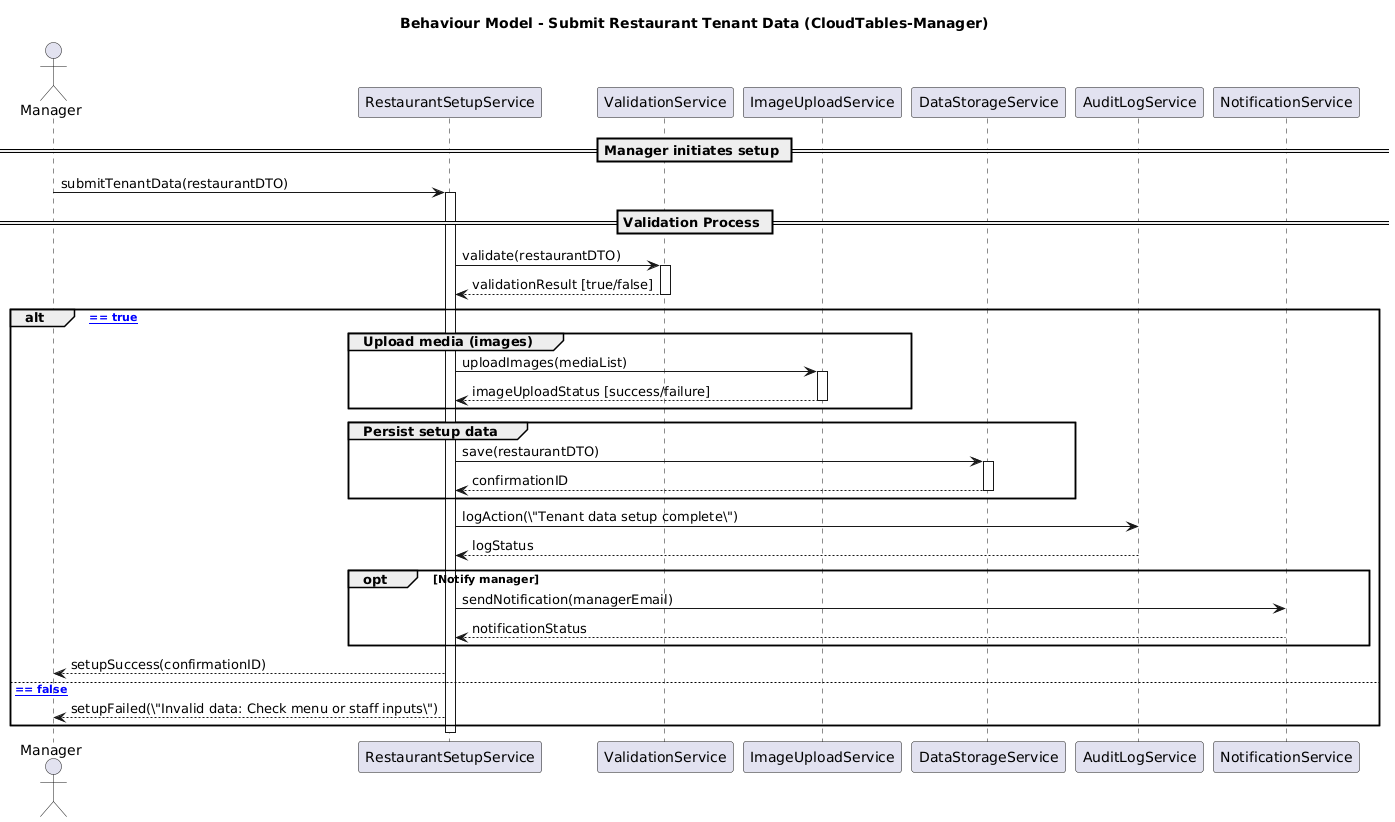
The CloudTables-Manager subsystem adopts a microservices-based architecture, with each service designed to perform a distinct and well-defined function. For example, the RestaurantSetupService is responsible for managing the initial onboarding of restaurant data, while the RestaurantUpdateService facilitates 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.

**Personal reflection:**This module has been one of the most engaging and valuable components of my academic year. I gained a strong understanding of practical software engineering concepts, particularly in system modelling, architectural design, and the application of UML. I appreciated the emphasis on connecting theory with implementation through structured tasks that mirror real-world development practices.

Although I faced unexpected personal challenges—having undergone surgery around the original deadline—I still completed the coursework and submitted it via email. However, due to a miscommunication, it was not recorded or marked. As such, this submission reflects my first formally assessed attempt. I’ve made every effort to ensure that the work demonstrates my understanding of the module content and learning outcomes. I respectfully hope this context will be taken into account in the assessment process and considered for an uncapped grade.