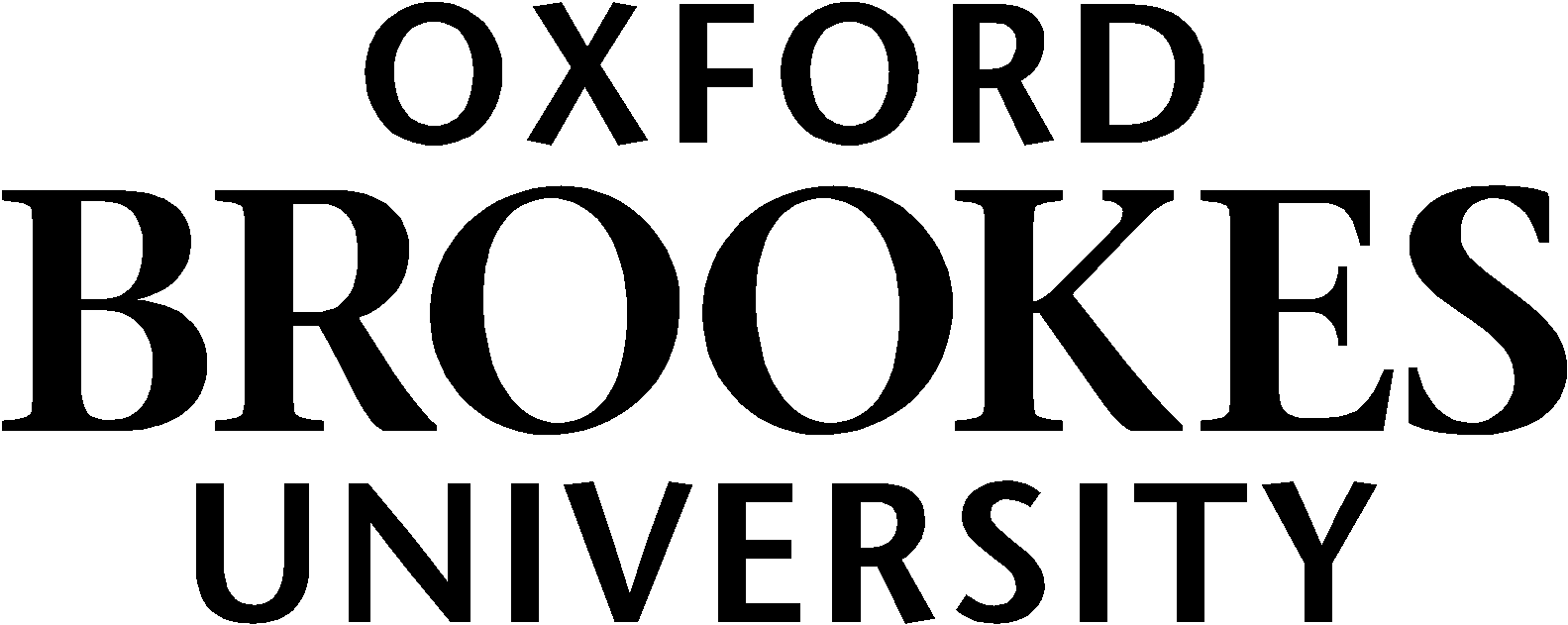
** 7**

**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 |

[**https://github.com/arjunrai111/COMP5047\_ResitCW\_19238473**](https://github.com/arjunrai111/COMP5047_ResitCW_19238473)

### 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.

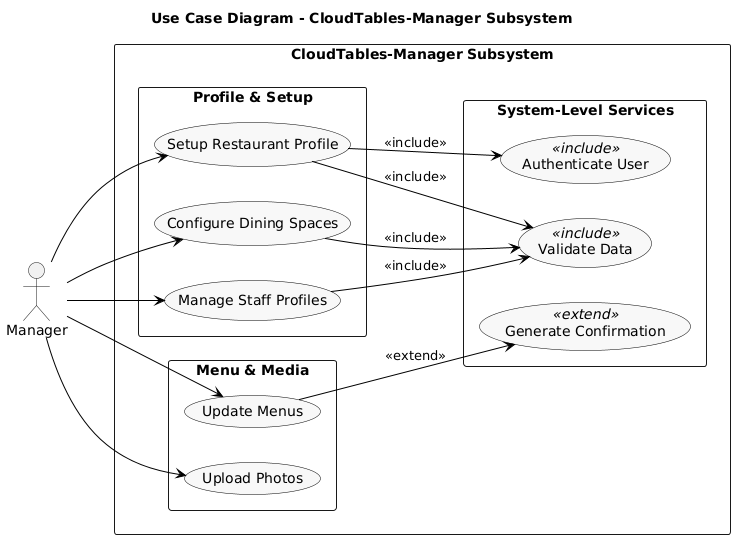
**Task 3: Specification and Modelling Software Functional Requirements (25 Marks)**

**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**

## 

## 

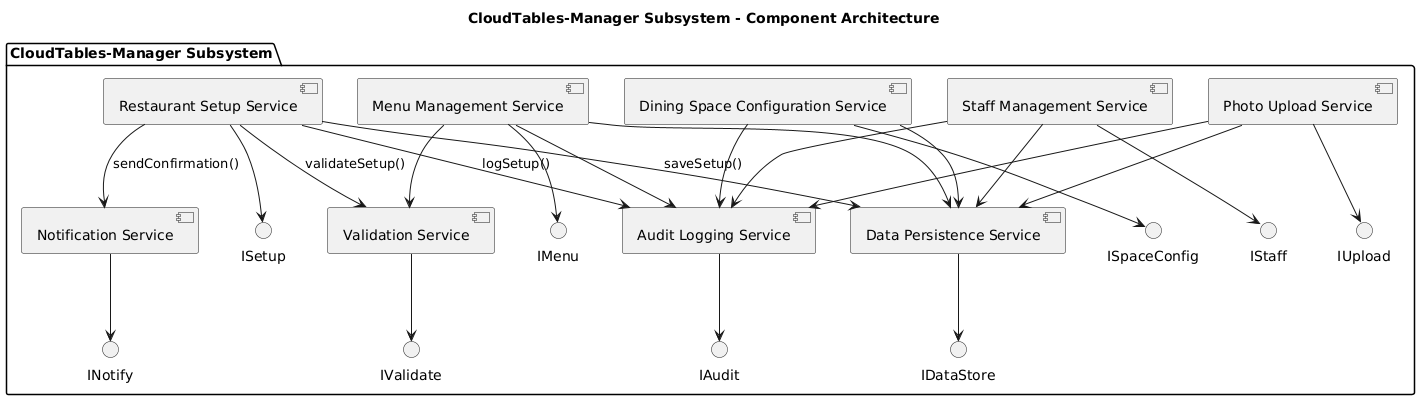
## 

## 

## 

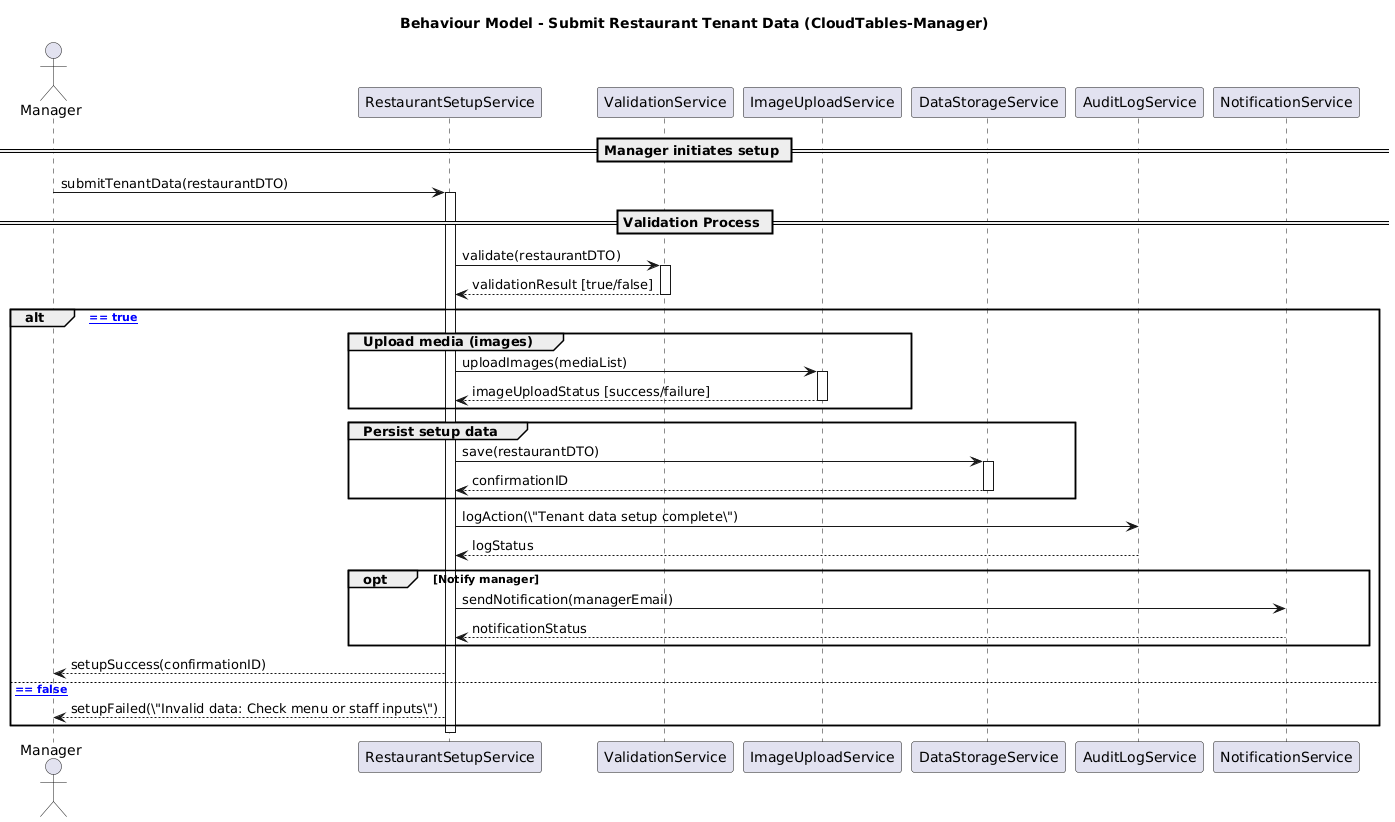
## 

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

****

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

## 5.1

****

### Summary Of Each Diagram:

The use case diagram describes and portrays the functional boundaries of the CloudTables-Manager subsystem by illustrating the primary interactions a restaurant manager can perform. Key use cases encompass setting up restaurant tenant data, updating existing information, managing menus and staff details, and receiving system notifications. Each use case is directly associated with specific functional requirements (e.g., FR-RM-1 and FR-RM-2) as outlined in the case study, ensuring clear traceability between the system's design and its intended functionalities. The "Manager" actor initiates each use case, and <<include>> relationships are employed to represent essential sub-functions such as data validation, storage, and auditing. This diagram effectively delineates the system's scope and the manager's role within it.

Activity Diagram:when i first started to create my activity diagram main focuses on the "Setup Restaurant Tenant Data" use case, modeling the sequential workflow a manager follows during the initial setup process. The process commences with the submission of core restaurant details, including menu items, layout configurations, and staff information. The diagram highlights critical decision points, such as data validation and error handling, followed by subsequent actions like saving the data, logging the activity, and optionally sending a confirmation notification. Swimlane partitions are utilized to distinguish actions performed by the manager from those handled by the system, ensuring a clear separation of responsibilities. This activity diagram comprehensively captures the operational logic of the setup process and aligns with the subsystem’s architectural design.

Component Diagram:So basically the component diagram presents the architectural layout of the CloudTables-Manager subsystem, adopting a microservices-based structure. Principal components include RestaurantSetupService, RestaurantUpdateService, ValidationService, DataStorageService, AuditService, and NotificationService. Each component interfaces through well-defined contracts such as ISetupService and IUpdateService, facilitating modular development and seamless communication. This design promotes loose coupling, scalability, and maintainability, ensuring that each service can evolve independently while collectively fulfilling the subsystem's functional requirements.

Class Diagram:The class diagram delineates the internal structure of the RestaurantSetupService component. This class implements the ISetupService interface and collaborates with four auxiliary services: ValidationService, DataStorageService, AuditService, and NotificationService. It processes a RestaurantDTO object, which aggregates related entities such as MenuItem, SpaceDetails, and StaffMember. Each of these classes encapsulates pertinent attributes, contributing to a cohesive and modular design. The diagram exemplifies adherence to object-oriented principles, ensuring clarity, reusability, and ease of maintenance.

Sequence Diagram:The sequence diagram illustrates the dynamic behavior of the RestaurantSetupService during the tenant onboarding process. It captures the interaction flow from the invocation of the submitInitialData() method to subsequent operations, including data validation, persistence, auditing, and optional notification dispatch. The diagram employs UML constructs such as alt for conditional branching and opt for optional actions, providing a clear depiction of the system's runtime behavior. This sequence diagram complements the structural models, offering insights into the temporal aspects of the subsystem's operations.

**Personal reflection:**For my COMP5047 resit project, I individually developed the CloudTables-Manager subsystem, a component of a cloud-based restaurant booking system. The project involved defining quality requirements, specifying functional use cases, and designing the system using UML. I created a Use Case Diagram to outline manager-specific operations, an Activity Diagram for the restaurant setup process, and a Component Diagram to reflect a microservices-based architecture. These services were further detailed through a Textual Interface Specification. For the detailed design phase, I produced a Class Diagram to structure the RestaurantSetupService, and a Sequence Diagram to illustrate its dynamic behaviour and interaction with supporting services like validation, storage, audit, and notification modules.

As a Computer Science student at Oxford Brookes, this coursework gave me a meaningful opportunity to apply practical software engineering skills in a real-world scenario. Designing the CloudTables-Manager subsystem helped me understand how to bridge requirements with design through structured UML modelling and component-based thinking. Although I underwent surgery around the original deadline, I completed and submitted the work via email, but it was not marked. This resit is therefore my first formally assessed attempt. I've made every effort to ensure it reflects the learning outcomes and my commitment to the module. This project has greatly improved my confidence in architectural design, version control, and system modelling—skills I’m excited to carry forward into my software development career.