Task 2:

(d) *CloudTables-Operation*: a desktop computer system for system operators to perform operations across restaurants.

Security and privacy:

System operators would have access to sensitive user information across multiple restaurants which means its important for this data to be secure. To protect this data, encryption should be used on both data at rest and data in transit, this way any data intercepted would be made unreadable. For data at rest, the AES-256 encryption algorithm can be used and for data in transit, protocols such as TLS or HTTPS can be used.

Strict access control should be used in order to protect data and operational controls. This can be done by implementing role based access control, limiting operators access to only what i deemed necessary for their job. MFA should also be used to further reduce the risk of unauthorised access especially for operators who work with sensitive data. This can be done by requiring the employee to present some sort of identification or biometric information to validate their authorization for a certain operation. All operator actions should also be recorded and stored to ensure no one is taking advantage of their privilege and to solve any security incidents.

Customer data in the operation systems must comply with the correct data regulations. Policies must be put in place to specify how long the data should be stored on the system and when it should be deleted securely. Only necessary customer information should be stored in the system, everything else should be deleted.

Performance:

System operators need access to real time information as they have to generate reports and check on different restaurant statuses which requires efficient data processing. A good way to do this is by implementing caching and database indexing so that during rush hours and weekends where the restaurant is very busy, the response time can still be quick. Another way to speed up data loading time is to use optimised SQL queries especially for the data that is most accessed.

System operators should have real time account management, this means they should be able to access and manage different restaurant accounts dynamically and have updates confirming that their actions have been processed correctly. This can be through real time notifications displaying the change the action caused such as activating or deactivating a restaurant account. Efficient and well created APIs should be used to allow system operators to access different restaurant accounts and modify them so that information is quickly available to customers.

Resource management is extremely important during peak restaurant times as operators have to perform data heavy tasks while there is already high traffic. This can be done through dynamic resource scaling, using cloud resources which automatically scale depending on the demand meaning system operators wont experience slow system processing in peak hours.

Reliability:

The operation system needs to be highly available at all times and especially during periods of high traffic, in order for system operators not to be slowed down and able to keep up at any time. This can be done by using load balancing, this way the traffic is distributed across different servers to prevent any single server from being overwhelmed. Load balancers also monitor the server health and automatically fail over to a backup server if anything goes wrong.

The data should be consistent across all systems and operator interfaces. The use of DDS(distributed data synchronisation) ensures data synchronisation across multiple systems so that any updates such as bookings or customer details are reflected immediately on any operator interface.

Effective recovery plans and regular backups would help system operators maintain data integrity. All customer, restaurant and operation data should be backed up daily to avoid any unwanted data loss.

There should also be automated consistency checks across all operator and restaurant accounts in order to detect any mistakes or synchronisation quickly. These checks would verify the integrity of the data and if possible fix them automatically or alert a system operator to prevent any operational issues.

Scalability:

The system can use elastic cloud infrastructure, this would allow the system to automatically change the amount of resources available based on the current state of traffic. This would mean during times of high traffic and peak restaurant hours, the system would increase the amount of resources available to match the need, then scale down automatically when traffic goes down. This would ensure that the operations system stays consistent and responsive at all times.

The use of microservice architecture would also benefit the scalability of the system as it splits up the system into a group of independent services that communicate with each other. This would allow any additions or modifications to be done independently and based on what the system needs. This would also help in the deployment of new services without the need of redeploying the whole system again.

Another way to increase the scalability of the operations system is through sharding, this is when a large database would be split into multiple smaller databases which are easier to manage and scale. This would be used to avoid the database getting too large which would cause the response time to be longer as all requests are made to one database. As the requests would be split over different smaller databases this would ensure the system would still have good response times even when many actions are being performed.

(c) CloudTables-Manager: a web interface for running on desktop or tablet computer for restaurant managers to manage their own restaurants.

Task 2:

Security and privacy:

- Data Encryption:
  - Encryption in Transit: Use SSL/TLS for all communication between the cloudTables-Manager interface and the cloud server. This ensures that employee records, financials, and inventory information cannot be captured and changed by a third party while being sent or received.
  - Encryption at Rest: Use AES-256 encryption for data at rest on the servers. This protects sensitive information such as payroll data or managerial notes when stored from prying eyes.

- Authentication and Authorization:
  - Multi-Factor Authentication(MFA): Use MFA for all management access logins as this will require something they have (mobile device for OTP) and something they know, which increases security beyond a password alone.
  - Role-Based Access Control (RBAC): Enforce RBAC to ensure that only authorised personnel can access or modify sensitive data. Managers should have tailored access rights, allowing them to perform their duties without exposing unnecessary information.

- Data Privacy Compliance:
  - Data Minimization: Collect only the necessary data required for managerial tasks, reducing the risk exposure in the event of a data breach.
  - Regulatory Compliance: Ensure compliance with relevant data protection regulations such as GDPR and CCPA by implementing robust data handling and storage policies. Regular audits should be conducted to verify compliance.

Performance:

- Efficient Data Retrieval:
  - Caching Frequently Accessed Data: Implement caching mechanisms for frequently accessed data such as daily sales reports and inventory levels.

This reduces the need for repeated database queries, thereby improving response times.
    - Optimised SQL Queries: Use optimised SQL queries and indexing strategies to enhance the speed of data retrieval, especially during peak business hours.

- Responsive User Interface:
    - Asynchronous Data Loading: Employ asynchronous data loading techniques to ensure the UI remains responsive even when fetching large datasets or performing complex operations.
    - Performance Monitoring: Continuously monitor application performance metrics and implement performance tuning measures as needed to maintain a high level of responsiveness.

Reliability:

- High Availability:
    - Load Balancing: Utilise load balancing to distribute incoming requests across multiple servers. This ensures that the system can handle high traffic volumes without performance degradation.
    - Failover Mechanisms: Implement failover mechanisms to switch to backup servers automatically in case of a server failure, ensuring continuous availability of the service.

- Data Integrity and Consistency:
    - ACID-Compliant Transactions: Ensure all database operations are ACID-compliant, maintaining data integrity and consistency across all transactions.
    - Regular Backups: Schedule regular backups of all critical data to enable quick recovery in case of data loss. Implement automated backup systems to ensure backups are performed consistently.

- Error Handling and Recovery:
    - Graceful Degradation: Design the system to degrade gracefully in case of partial failures, allowing managers to continue performing critical tasks.
    - Comprehensive logging: Implement comprehensive logging to capture detailed information about errors and system events. This aids in diagnosing issues and improving system reliability.

    -

Scalability:

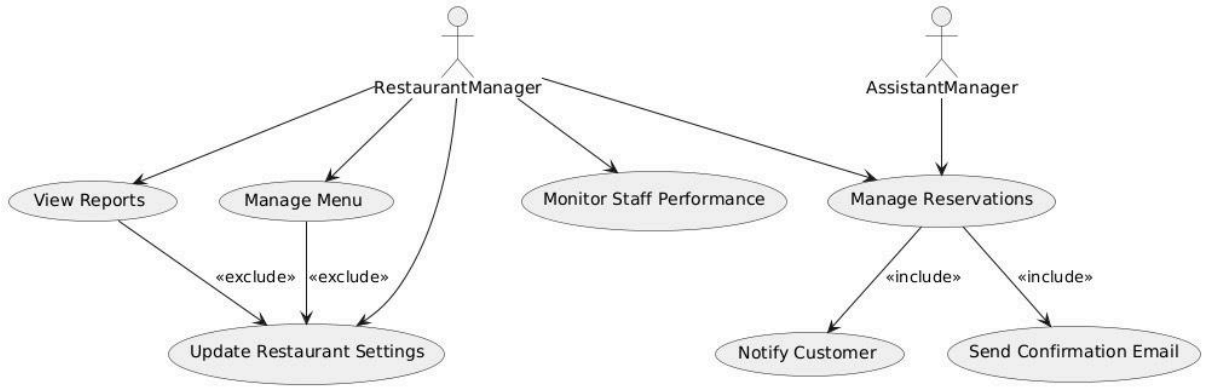
- Vertical Scalability:
    - Elastic Resource Allocation: Use cloud services that support elastic resource allocation, allowing the system to scale up resources (CPU, memory) during high-demand periods and scale down during low -demand periods.

- Horizontal scalability:

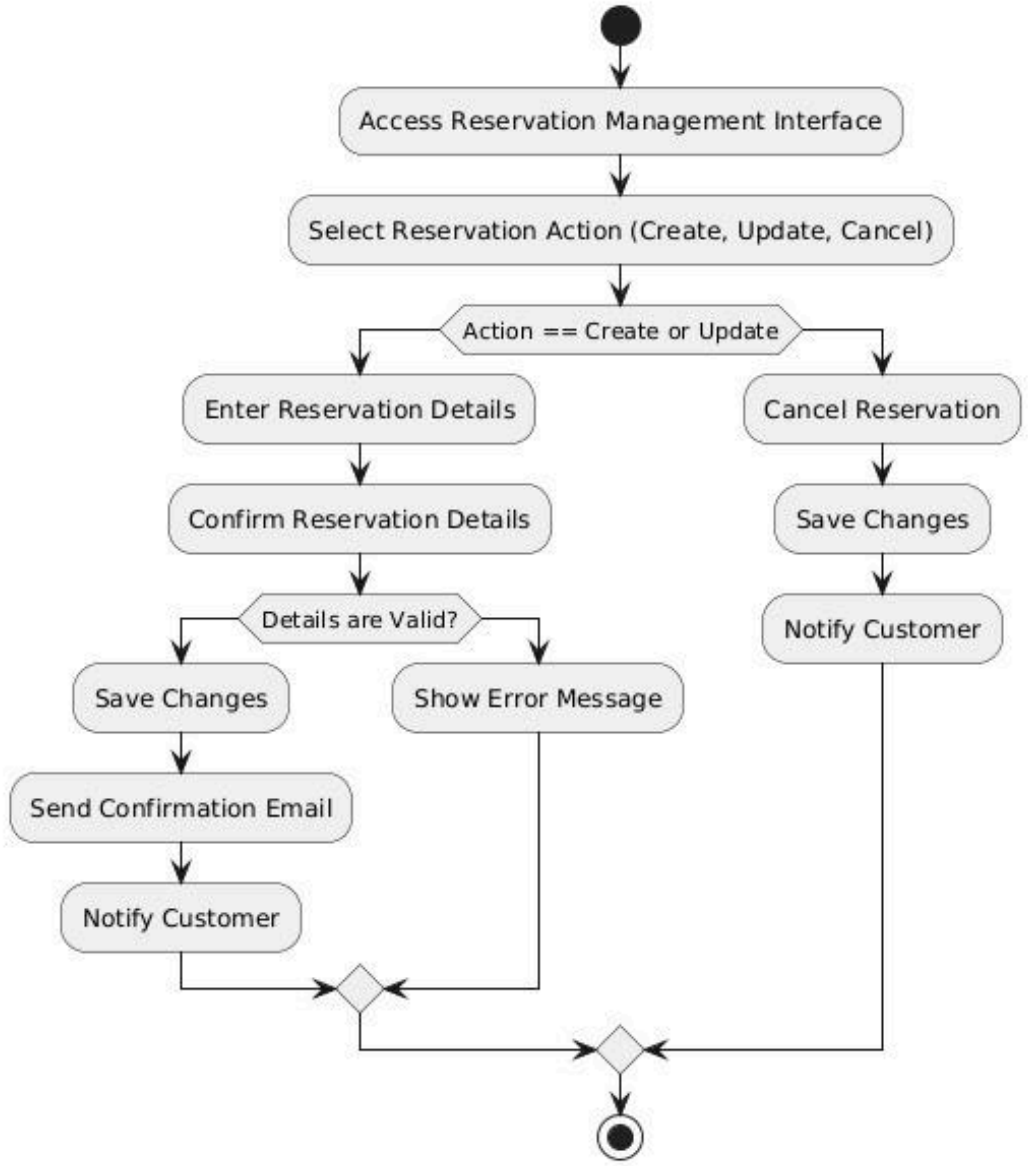
- Microservices Architecture: Adopt a microservices architecture to enable independent scaling of different system components. For instance, the reporting module can scale independently from the inventory management module.
- Containerization: Use containerization (e.g., Docker) to package and deploy microservices, facilitating easy scaling and management of services.

- Geographical Scalability:
  - Content Delivery Network (CDN): Implement a CDN to ensure fast content delivery to managers regardless of their geographical location. This reduces latency and improves user experience.
  - Multi-Region Deployment: Deploy the application across multiple geographic regions to enhance availability and performance for users in different locations. This also provides redundancy in case of regional outages.

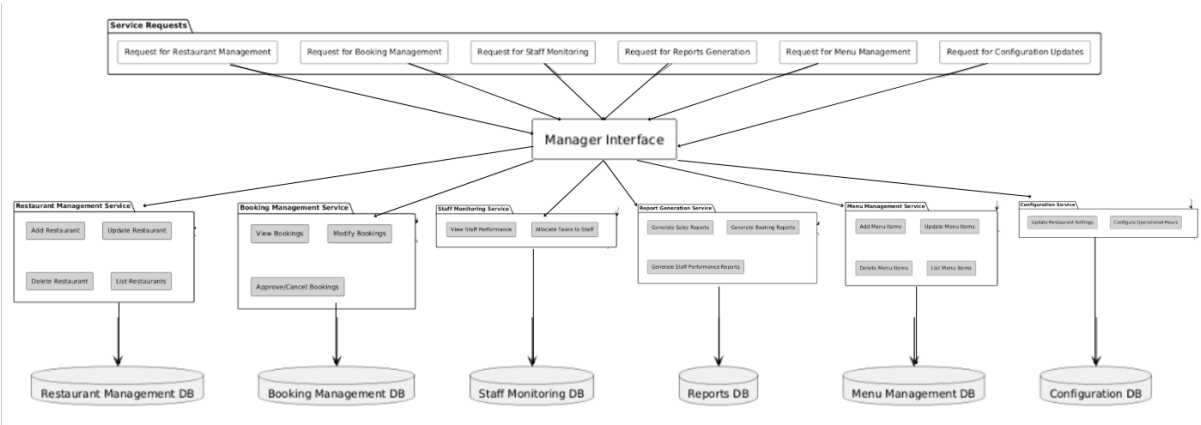
Task 3a:



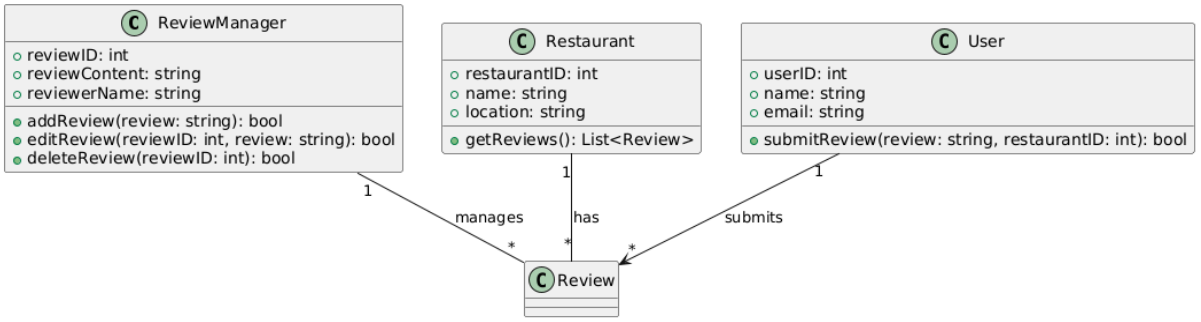
Task 3b:



Task 4a:



Task 5a:



Task 5b: