|  |  |
| --- | --- |
| **Course Title** | Software Architectures |
| **Course No(s)** | SEMTZG651 |
| **Student Name** | Banukunta Vikas |
| **Student ID** | 2021MT70057 |

**Assignment Questions (15 Marks)**

**Question #1 (5 Marks)**

**Objective:** To get familiar with the software architecture basics.

**Activity:**

1. Choose an existing system from your workplace
2. Understand the purpose (goal) of the system & its key requirements
3. Study the architecture and understand the tactics used

**Document your work in the following format in doc / pdf:**

1. Purpose of the system (Goal)
2. Key requirements of the system – functional & non-functional
3. Utility tree of Architecturally Significant Requirements (ASR)
4. Tactics used to achieve the top 5 ASRs
5. Software Architecture diagram – Context diagram, Module decomposition, Component & Connection diagram, Deployment diagram
6. Description of how the system works
7. Key learnings

**Question #2 (10 Marks)**

**Objective:** To gain experience in architecting real life applications in domains such as Retail, Transportation, Healthcare, Hospitality, etc. Example systems: Swiggy, Uber, an IoT system to monitor health of industrial air conditioners.

Document your work in the following format in doc / pdf:

**Activity**

1. Identify top 3 Architecturally Significant Requirements (ASRs) and write them in the form of a Utility tree. Why are these architecturally significant?
2. Describe in detail, the tactics you recommend for each ASR. For example, if caching is a tactic you recommend, please mention what you will cache, what tool you would use, how it will work, etc.
3. Draw 2 software architecture diagrams – component & connection view and deployment view – to understand how the system works.
4. Indicate important messages between components by labelling the connections in the C&C view. Also indicate the communication method used.
5. Draw sequence diagram for one major scenario (use case). Mention the scenario.
6. State the architecture patterns used. Explain, where in the architecture, these patterns have been used.
7. What did you learn by doing this assignment? Mention 3 key learnings.

**------------------**

1. **Software Architecture Basics Documentation**

**System**: ABC Airlines Reservation System

**1. Purpose of the system (Goal):**

The ABC Airlines Reservation System is designed to facilitate the booking and management of airline tickets for passengers. Its primary goal is to provide a seamless and efficient reservation process for customers while enabling the airline to effectively manage flight schedules, seat availability, and passenger information.

**2. Key requirements of the system - functional & non-functional:**

* **Functional Requirements:**
* Flight Booking: Ability for customers to search for flights, select seats, and book tickets.
* Seat Management: Management of seat availability, including seat assignment and upgrades.
* Payment Processing: Secure processing of payments for booked tickets using various payment methods.
* Reservation Management: Management of passenger reservations, including modifications and cancellations.
* **Non-Functional Requirements:**
* Performance: System must handle concurrent user requests efficiently and provide timely responses.
* Scalability: Ability to handle a large number of concurrent users and accommodate future growth.
* Security: Ensure secure transmission and storage of sensitive passenger information, including payment details.
* Reliability: System must be available and reliable, minimizing downtime and data loss.

**3. Utility tree of Architecturally Significant Requirements (ASR):**

* **ABC Airlines Reservation System**
* **Performance**
  + - * + Response time
        + Throughput
* **Scalability**
  + - * + Horizontal scaling
        + Vertical scaling
* **Security** 
  + - * + Data encryption
        + Authentication
        + Authorization
* **Reliability**
  + - * + Fault tolerance
        + Redundancy

**4. Tactics used to achieve the top 5 ASRS:**

* **Performance:**
  + Caching: Implement caching mechanisms to reduce response time for frequently accessed data.
  + Asynchronous Processing: Employ asynchronous processing for tasks such as payment processing to improve overall system throughput.
  + Load Balancing: Distribute incoming traffic across multiple servers to evenly distribute the workload and improve response time.
* **Scalability:**
  + Horizontal Scaling: Deploy the system in a distributed architecture to allow for the addition of more servers to handle increased load.
  + Database Sharding: Partition the database to distribute data across multiple servers, enabling horizontal scalability and improved performance.
* **Security:**
  + Data Encryption: Encrypt sensitive data such as payment information to ensure secure transmission and storage.
  + Authentication: Implement strong authentication mechanisms, such as multi-factor authentication, to verify the identity of users.
  + Authorization: Enforce role-based access control to restrict access to sensitive functionality and data based on user roles.
* **Reliability:**
  + Fault Tolerance: Implement redundant components and failover mechanisms to ensure uninterrupted service in case of component failures.
  + Automated Monitoring: Utilize automated monitoring tools to detect and address potential issues proactively, minimizing downtime.

**5. Software Architecture Diagrams:**

* **Context Diagram:**

A diagram of a company

Description automatically generated

* **Module Decomposition:**

A diagram of a flight management

Description automatically generated

* **Component & Connection Diagram:**

A diagram of a plane

Description automatically generated

* **Deployment Diagram:**

A diagram of a server

Description automatically generated

**6. Description of how the system works:**

* The ABC Airlines Reservation System operates by integrating various modules such as flight search, seat management, payment processing, and reservation management. Customers interact with the system through a user-friendly interface, where they can search for flights, select seats, and complete the booking process. Behind the scenes, the system communicates with external systems such as payment gateways and flight databases to retrieve relevant information and process transactions securely. The system ensures data integrity, security, and reliability throughout the reservation process.

**7. Key Learnings:**

* Understanding the importance of defining clear goals and requirements to guide the architectural design of a system.
* Recognizing architecturally significant requirements (ASRs) and selecting appropriate tactics to address them.
* Learning various architectural patterns and techniques to achieve desired system qualities such as performance, scalability, security, and reliability.
* Gaining practical experience in creating software architecture diagrams to visualize system components, connections, and deployment strategies.

1. **Documentation: Architectural Design for ABC Food Delivery App**

**1. Architecturally Significant Requirements (ASRs) Utility Tree:**

* **ASR: Scalability**
  + **Performance**
    - Response time
    - Throughput
  + **Reliability**
    - Fault tolerance
    - Disaster recovery
  + **Security**
    - Data encryption
    - Authentication
    - Authorization
* **Explanation:**
* **Scalability:** Due to fluctuating demand in food orders, scalability is crucial to handle varying loads during peak times. Performance, reliability, and security are directly influenced by scalability.
* **Reliability:** Ensuring continuous and reliable operation despite failures or disasters is essential for maintaining user trust and business continuity.
* **Security:** Protecting user data, ensuring authentication, and proper authorization are vital to maintaining the integrity and confidentiality of the system and its users' information.

**2. Tactics for each ASR**

* + **Scalability:** Implement horizontal scaling using containerization (e.g., Docker/Kubernetes) to handle increased loads. Utilize load balancers to evenly distribute traffic. Employ caching mechanisms (e.g., Redis) for frequently accessed data to improve response time and throughput.
  + **Reliability:** Implement redundancy at critical points using failover mechanisms. Utilize distributed databases (e.g., Cassandra) for data replication and fault tolerance. Regularly backup data and implement disaster recovery strategies (e.g., automated backups to offsite locations).
  + **Security:** Encrypt sensitive data both at rest and in transit (e.g., using HTTPS). Implement multi-factor authentication for user accounts. Employ role-based access control (RBAC) for authorization to ensure only authorized users can access specific functionalities.

**3. Software Architecture Diagrams**

* + **Component & Connection View:**

A diagram of a computer network

Description automatically generated

* + **Deployment View:**

A diagram of a software company

Description automatically generated

**4. Important Messages between Components:**

* + Labelled connections:
  + User App to Server: HTTPS for sending user requests and receiving responses.
  + Restaurant App to Server: Secure communication (HTTPS) for managing restaurant data and receiving orders.
  + Delivery Person App to Server: HTTPS for sending delivery status updates and receiving new orders.

**5. Sequence View:**

* **Scenario:** Placing an Order
* **Actors:**
  + User
  + Restaurant
  + Delivery Person
* **Steps:**
  + Step 1. User opens the User App.
  + Step 2. User browses restaurants and selects items to order.
  + Step 3. User places the order.
  + Step 4. User App sends the order request to the Server.
  + Step 5. Server receives the order request and processes it.
  + Step 6. Server forwards the order to the Restaurant App.
  + Step 7. Restaurant receives the order and prepares it.
  + Step 8. Restaurant confirms the order status to the Server.
  + Step 9. Server updates the order status and assigns a delivery person.
  + Step 10. Delivery Person App receives the order assignment from the Server.
  + Step 11. Delivery Person proceeds to the restaurant for pickup.
  + Step 12. Delivery Person picks up the order.
  + Step 13. Delivery Person delivers the order to the User's location.
  + Step 14. User receives the order and confirms delivery.
  + Step 15. Delivery Person updates the delivery status to the Server.
  + Step 16. Server updates the order status as delivered.

**6. Architecture Patterns Used:**

* **Microservices:** Utilized to break down the system into smaller, manageable services, promoting scalability, maintainability, and reusability. Each application (User App, Restaurant App, Delivery Person App) represents a microservice, enabling independent development and deployment.
* **Model-View-Controller (MVC):** Employed in each application to separate concerns between the data/model layer, presentation/view layer, and controller layer, facilitating code organization and maintainability.

**7. Key Learnings**

1. Understanding the critical importance of architecturally significant requirements (ASRs) in shaping the design and ensuring the success of real-life applications.
2. Practical application of architectural tactics and patterns to address scalability, reliability, and security concerns in complex systems like food delivery apps.
3. Gain insights into the importance of proper communication between components and the significance of using secure communication protocols to protect sensitive data.