**Day-2-Architecture Design:**

**System Architecture and Design:**

The insurance management system will be designed using a microservices architecture, where different functionalities are divided into separate, loosely coupled microservices. Spring Microservices and Spring Cloud components will be utilized to enable communication and coordination among these microservices. The system will be deployed on AWS EC2 instances, and Spring Cloud Load Balancer or AWS Elastic Load Balancer will be used for load balancing.

**The high-level architecture will consist of the following microservices:**

**Policy Management Microservice:**

Responsible for managing insurance policies.

Handles policy creation, updates, retrieval, and listing.

Communicates with the User Management Microservice for authentication and authorization.

**Claims Microservice**:

Manages insurance claims and their processing.

Handles claim creation, updates, and retrieval.

Communicates with the Policy Management Microservice for claim-policy associations.

**User Management Microservice:**

Handles user registration, authentication, and authorization.

Manages user roles and permissions.

Provides access tokens for secure API access.

**Reporting and Analytics Microservice:**

Generates various reports and provides analytics on policies and claims.

Offers insights into policy trends, claim patterns, etc.

Communicates with other microservices to fetch data.

**API Gateway (Optional):**

Acts as the single entry point for external clients to access the microservices.

Handles routing and load balancing of incoming requests.

Implements security, throttling, and other cross-cutting concerns.

**Data Model for the Insurance System:**

The data model will consist of several entities, representing different aspects of the insurance system. Here's an outline of the data model:

**Policy Entity:**

policyId (Primary Key): Unique identifier for each policy.

policyNumber: Unique policy number assigned to the policy.

policyHolderName: Name of the policy holder.

premiumAmount: Premium amount associated with the policy.

startDate: Effective start date of the policy.

endDate (Optional): Effective end date of the policy (null for ongoing policies).

**Claim Entity:**

claimId (Primary Key): Unique identifier for each claim.

claimNumber: Unique claim number assigned to the claim.

description: Description of the claim incident.

incidentDate: Date of the incident triggering the claim.

status: Status of the claim (e.g., open, in progress, closed).

policyId (Foreign Key): Reference to the associated policy.

**User Entity:**

userId (Primary Key): Unique identifier for each user.

username: Username for logging into the system.

password: Securely hashed password for user authentication.

roles: Set of roles assigned to the user (e.g., admin, manager, agent).

**Role Entity:**

roleId (Primary Key): Unique identifier for each role.

roleName: Name of the role (e.g., admin, manager, agent).

The above entities represent a simplified data model. Depending on the system's complexity and additional features, the data model may need further refinement and expansion.