

**PROJECT REPORT**

*Submitted by*

**SHOUNAK CHANDRA (RA2111032010026)**

*Under the Guidance of*

**Dr. Prabakeran S**

**Assistant Professor, Networking and Communications Department**

*In partial satisfaction of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY  
in  
COMPUTER SCIENCE ENGINEERING  
with specialization in Internet Of Things**



**SCHOOL OF COMPUTING  
COLLEGE OF ENGINEERING AND TECHNOLOGY  
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR - 603203**

**October 2023**



## SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

### **BONAFIDE CERTIFICATE**

Certified that this Project Report titled “Cloud-Powered Web Deployment” is the bonafide work done by Shounak Chandra (RA2111032010026) who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

#### **SIGNATURE**

Dr. Prabakeran S

#### **Course Faculty**

Assistant Professor

Department of Networking and Communications

SRMIST

#### **SIGNATURE**

Dr. Annapurani P. K.

#### **Head of the Department**

Department of Networking

and Communications

SRMIST

## TABLE OF CONTENTS

S.No	CONTENTS	PAGE NO
1.	Problem Statement	1
2.	Modules of Project	2
3.	Diagrams	3
	a. Use case Diagram	4
	b. Class Diagram	5
	c. Sequence Diagram	6
	d. Collaboration Diagram	7
	e. State Chart Diagram	8
	f. Activity Diagram	9
	g. Package Diagram	10
	h. Component Diagram	11
	i. Deployment Diagram	12
4.	Output Screenshots and Code	13
5.	Conclusion and Results	18
6.	References	19

## **PROBLEM STATEMENT**

This project aims to address the challenge of deploying a web application that collects and manages employee details, such as first name, last name, ID, primary skills, and location, while utilizing key Amazon Web Services (AWS) components including Route 53 for DNS management, EC2 for hosting, RDS for database management, and S3 for file storage.

Through a comprehensive exploration of these challenges, this report aims to provide practical insights and recommendations for businesses and developers seeking to deploy web applications effectively in the AWS cloud environment. The project's focus on Route 53, EC2, RDS, and S3 highlights the importance of these services in the context of modern web deployment.

## MODULES OF THE PROJECT

The modules for your Cloud-Powered Web Deployment project are -

Project Modules:

### 1. Project Overview and Introduction:

- Provide an overview of the project's objectives and scope.
- Introduce the AWS services (Route 53, EC2, RDS, S3) used in the project.

### 2. AWS Service Setup:

- Set up and configure Route 53 for domain name management.
- Launch and configure EC2 instances to host the web application.
- Configure RDS to store and manage employee details.
- Set up S3 for file storage and static content hosting.

### 3. Web Application Development:

- Develop the web application for collecting and managing employee details.
- Implement user interfaces for data entry and retrieval.

### 4. Database Integration:

- Establish a connection between the web application and the RDS database.
- Develop and implement the database schema for storing employee information.

### 5. Scalability and Performance Optimization:

- Implement strategies for horizontal and vertical scaling of EC2 instances.
- Optimize the database performance for efficient data retrieval and storage.

### 6. Fault Tolerance and Redundancy:

- Set up automated backups for the RDS database.
- Configure failover and redundancy in the EC2 instances.

### 7. Monitoring and Logging:

- Implement AWS CloudWatch for monitoring system health and performance.
- Set up log collection and analysis for troubleshooting and performance optimization.

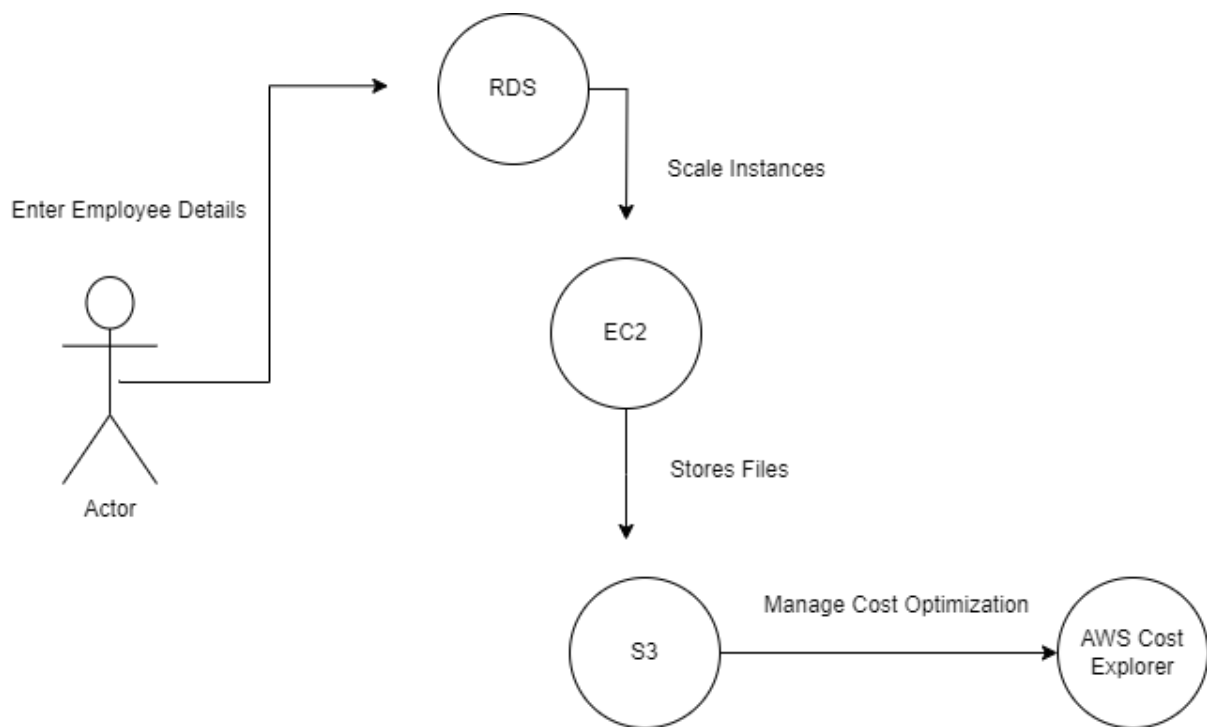
### 8. Cost Analysis and Optimization:

- Monitor and analyze the project's cost using AWS Cost Explorer.
- Implement cost optimization strategies, such as reserved instances and spot instances

# DIAGRAMS

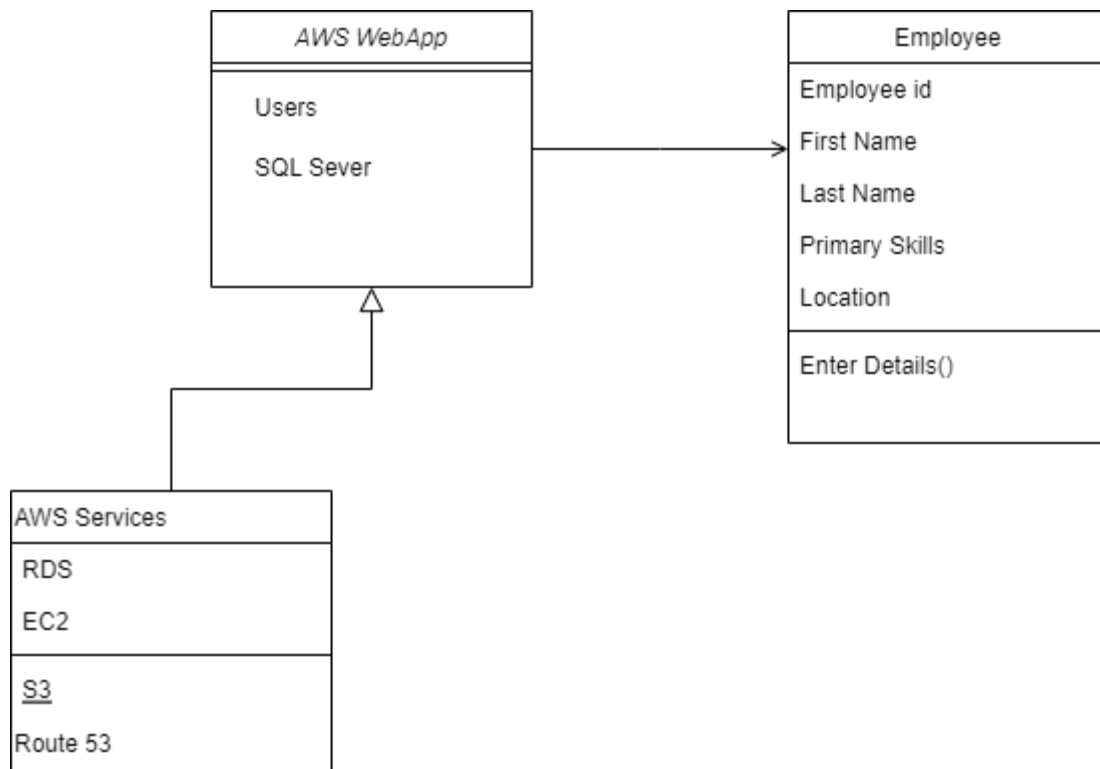
## USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.



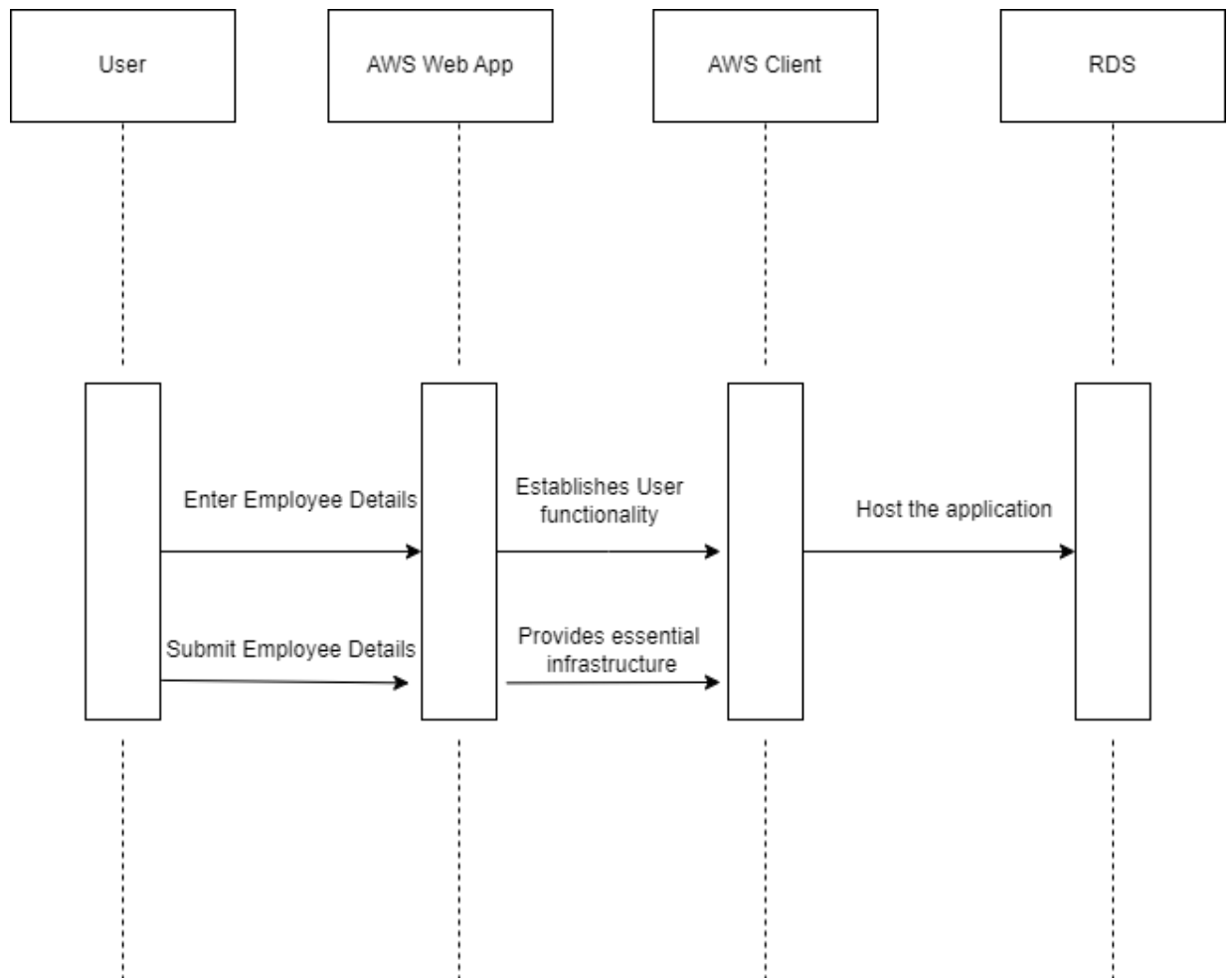
## CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.



## SEQUENCE DIAGRAM

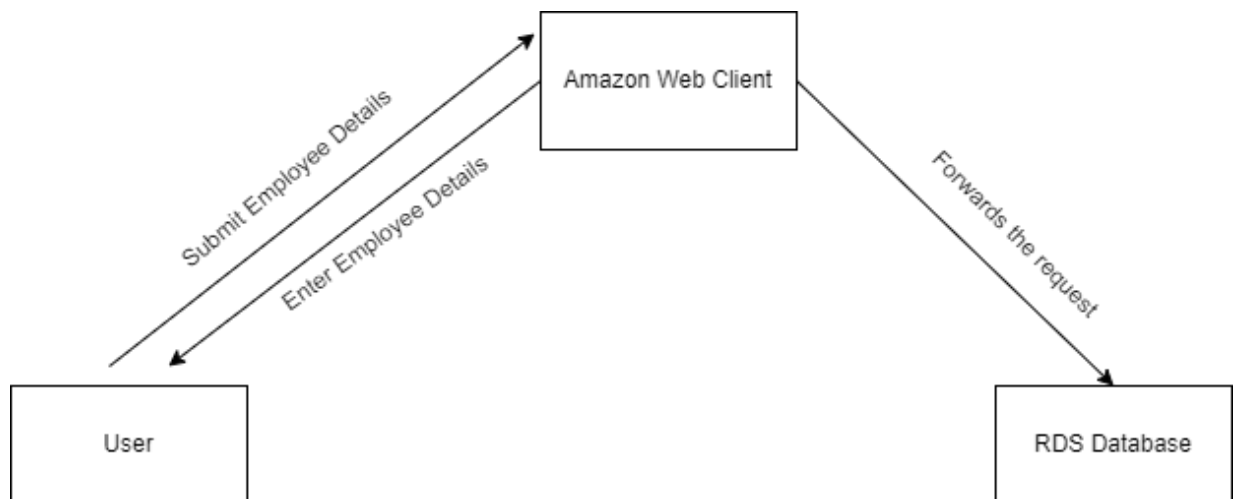
UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.





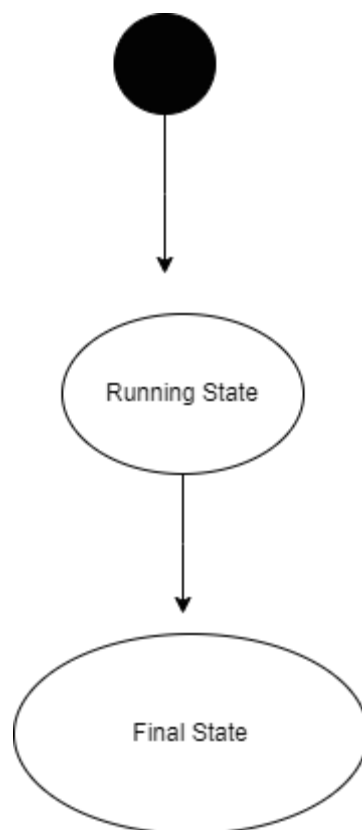
## COLLABORATION DIAGRAM

Collaboration diagrams are used to show how objects interact to perform the behaviour of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration is used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.



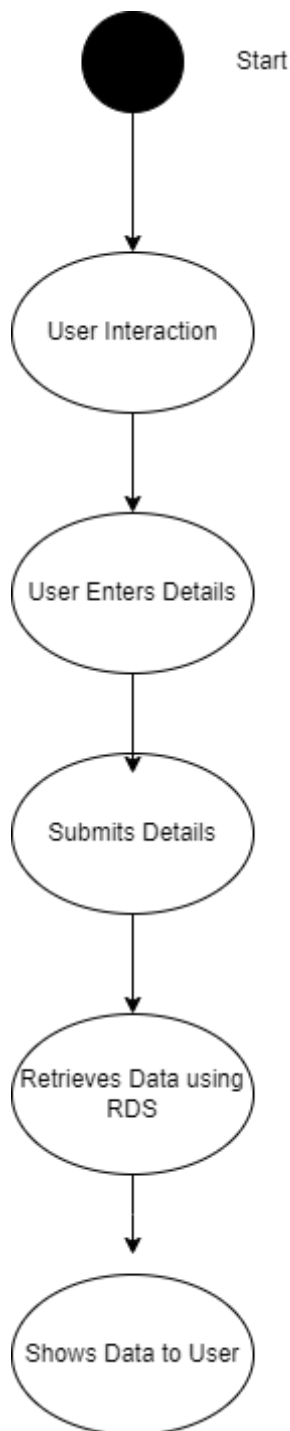
## STATE CHART DIAGRAM

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. Activity diagram explained in the next chapter, is a special kind of a state chart diagram.



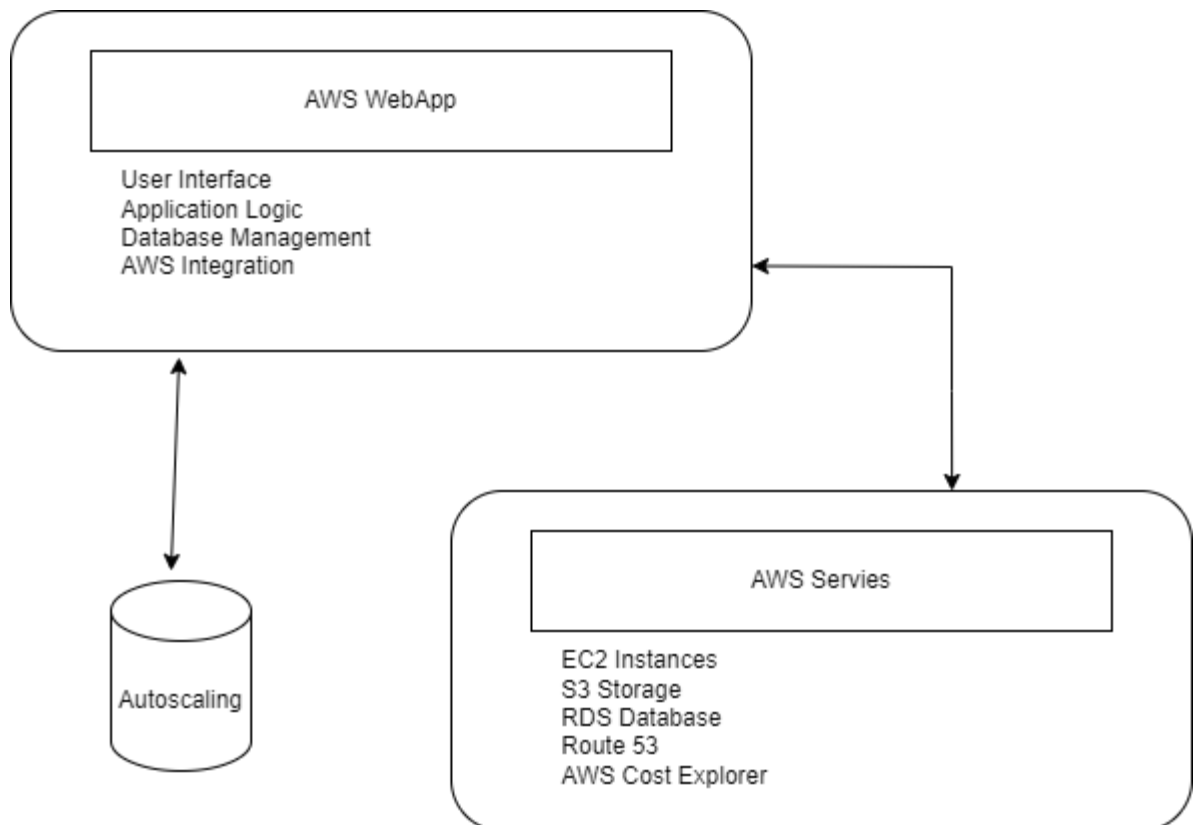
## ACTIVITY DIAGRAM

Activity diagram is another important behavioural diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modelling the flow from one activity to another activity.



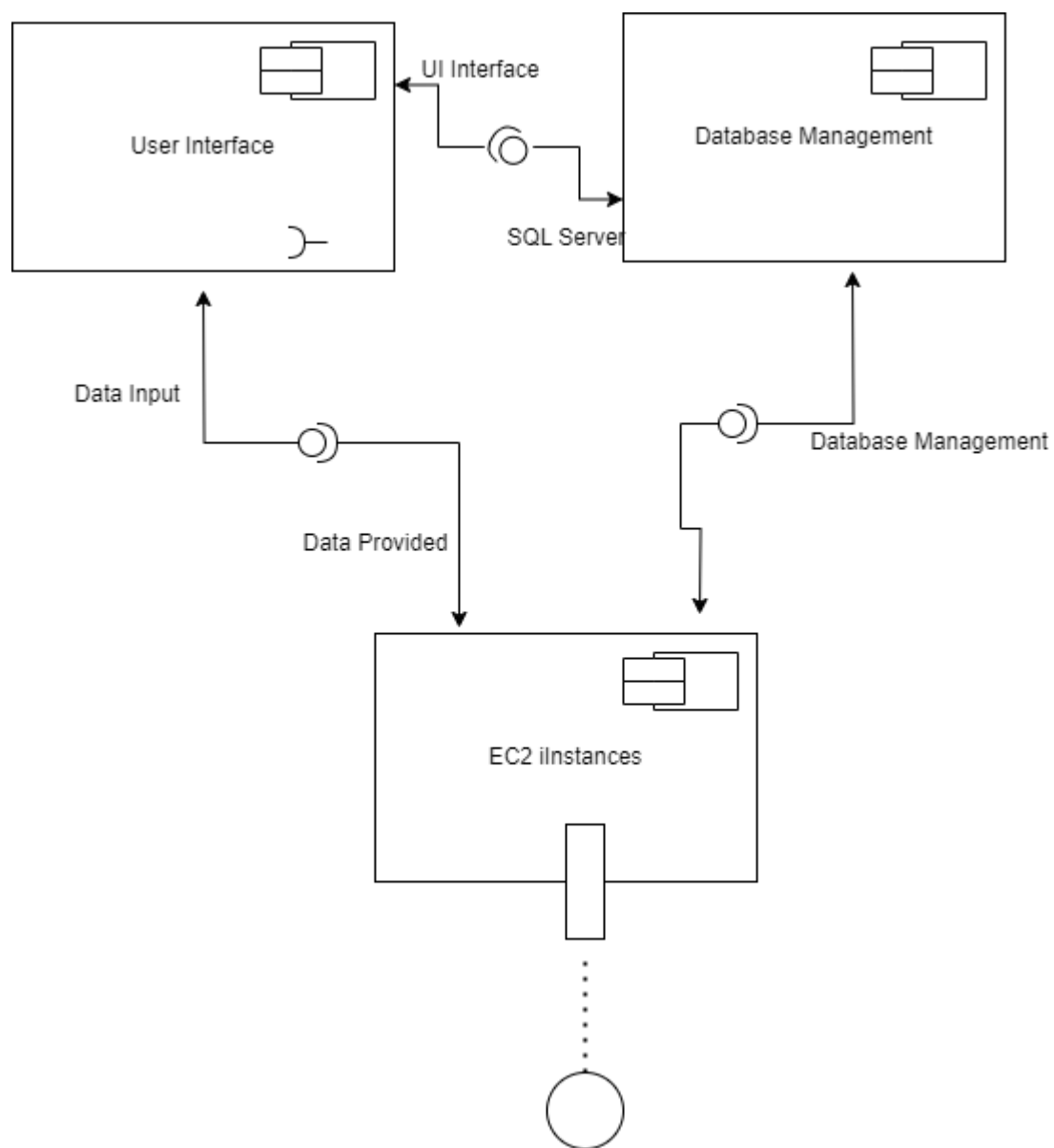
## PACKAGE DIAGRAM

Package diagram, a kind of structural diagram, shows the arrangement and organization of model elements in middle to large scale project. Package diagram can show both structure and dependencies between sub-systems or modules, showing different views of a system, for example, as multi-layered (aka multi-tiered) application - multi-layered application model.



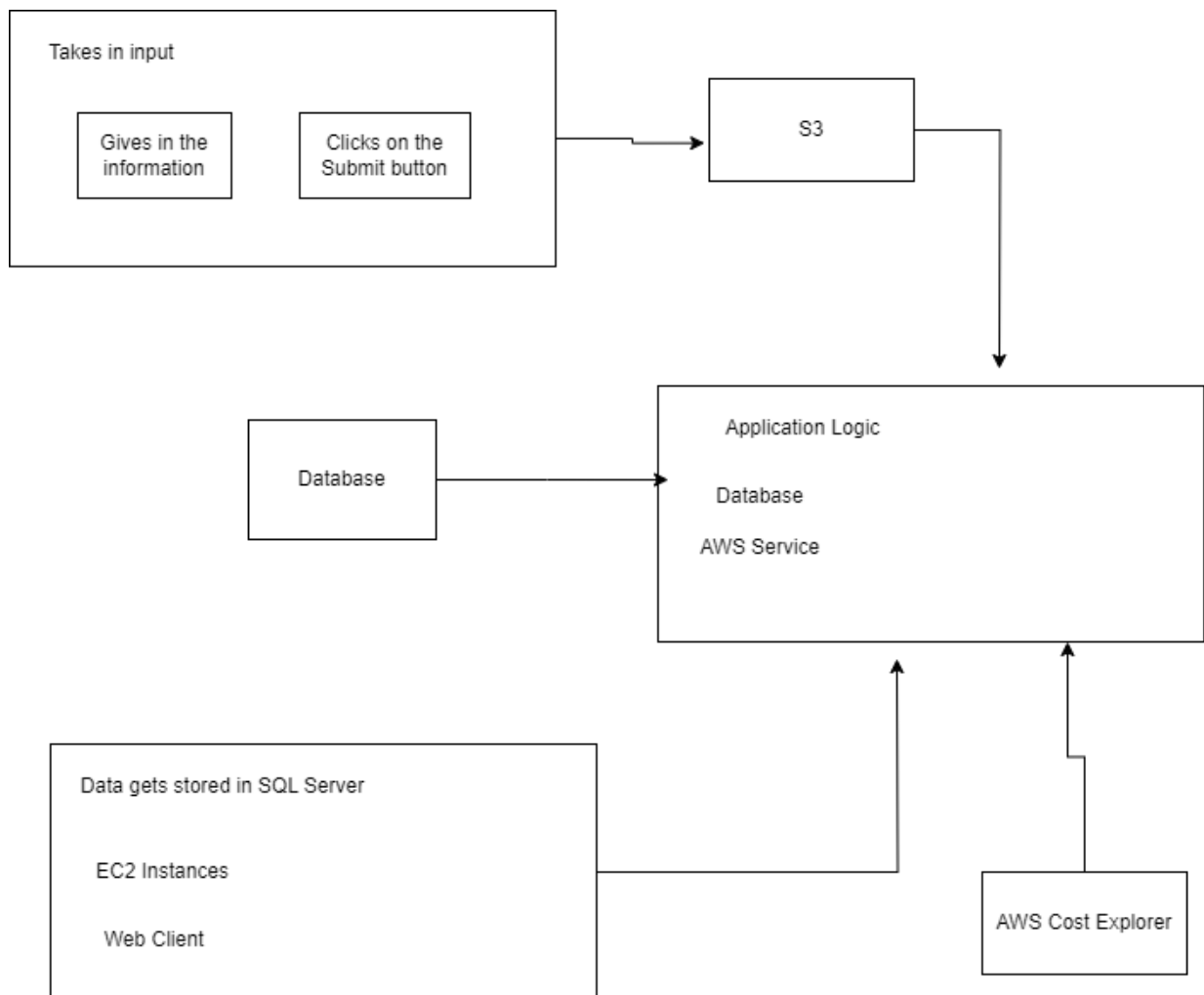
## COMPONENT DIAGRAM

UML Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.



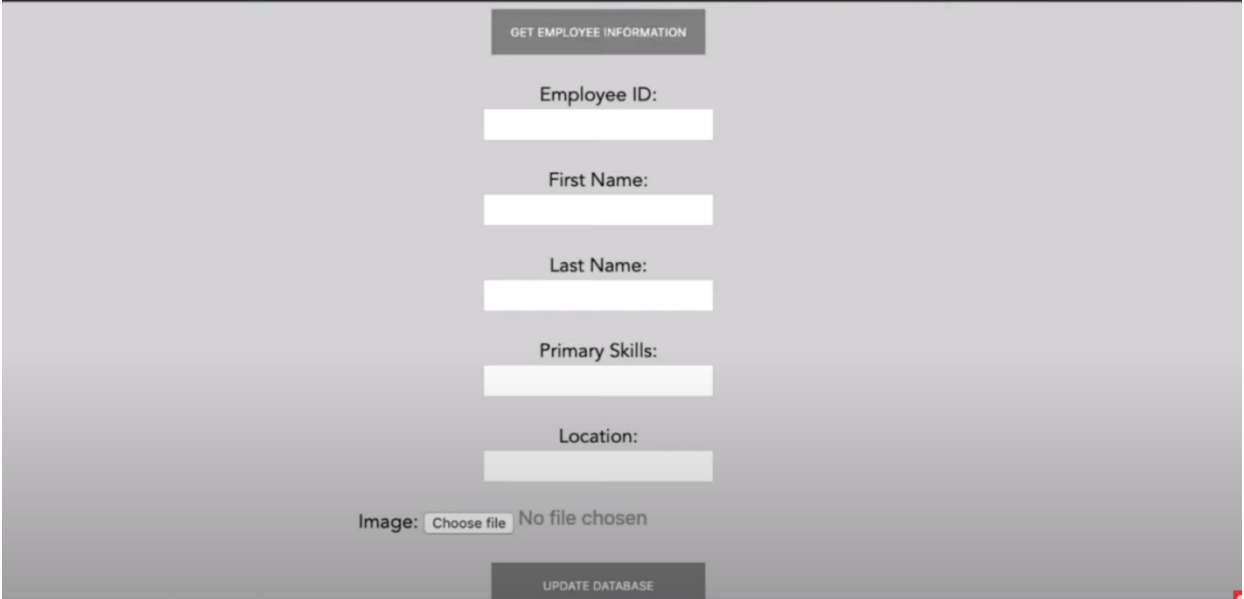
## DEPLOYMENT DIAGRAM

Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering. A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes.



## INPUT/OUTPUT SCREENSHOTS

The homepage offers a user-friendly interface for entering vital employee information, including first name, last name, employee ID, primary skills, and location, simplifying data input and management.



The screenshot displays a web form for entering employee information. At the top, there is a dark gray button labeled "GET EMPLOYEE INFORMATION". Below this, the form consists of several input fields, each preceded by a label: "Employee ID:", "First Name:", "Last Name:", "Primary Skills:", and "Location:". Each label is followed by a white rectangular input field. At the bottom of the form, there is an "Image:" label, a "Choose file" button, and the text "No file chosen". Below the image section is a dark gray button labeled "UPDATE DATABASE". The entire form is set against a light gray background.

The homepage serves as the gateway to our web application, providing a user-friendly interface for entering essential employee information. With intuitive input fields for first name, last name, employee ID, primary skills, and location, users can effortlessly input their data. Upon submission, this data is securely processed and stored, facilitating streamlined management and access.

The SQL database is organized into five tables: "empid" for Employee ID, "fname" and "lname" for First Name and Last Name, "priskill" for Primary Skills, and "location" for work locations. This structured data storage optimizes data retrieval and management, supporting our organization's requirements effectively.

```
| empid | fname | lname | pri_skill | location |
+-----+-----+-----+-----+-----+
| 1     | hemant | sharma | aws       | india    |
| 2     | John  | Doe    | aws       | India    |
+-----+-----+-----+-----+-----+
2 rows in set (0.23 sec)
```

```
mysql> select * from employee;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 24
Current database: employee
```

```
| empid | fname | lname | pri_skill | location |
+-----+-----+-----+-----+-----+
| 1     | hemant | sharma | aws       | india    |
| 2     | John  | Doe    | aws       | India    |
| 3333  | Bhoomika | Aradhya | AWS      | Bangalore |
| 33    | Arjun  | K      | DevOps    | Hyderabad |
+-----+-----+-----+-----+-----+
4 rows in set (2.62 sec)
```

In the SQL database, we've meticulously organized employee data into five distinct tables. The "empid" table corresponds to Employee ID, capturing unique identifiers for each employee. The "fname" and "lname" tables are dedicated to First Name and Last Name, ensuring detailed personal identification. The "priskill" table is exclusively reserved for Primary Skills, offering insight into each employee's expertise. Finally, the "location" table systematically records the work locations of our team members. This structured approach to data storage enables efficient retrieval, management, and analysis of employee information, enhancing our ability to meet organizational needs and demands.



## CODE

### Emp.py

```
-- This SQL script defines tables for employee data storage.

-- Employee ID Table: Stores unique employee identifiers.
CREATE TABLE empid (
    emp_id INT PRIMARY KEY,
    -- Primary key for employee identification.
    -- Example: 1001, 1002, ...

    -- Add more columns if needed, such as date of joining, etc.
);

-- First Name Table: Records employees' first names.
CREATE TABLE fname (
    emp_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    -- Primary key referencing empid.
    -- Example: John, Mary, ...

    -- Add more columns for additional information.
);

-- Last Name Table: Stores employees' last names.
CREATE TABLE lname (
    emp_id INT PRIMARY KEY,
    last_name VARCHAR(50),
    -- Primary key referencing empid.
    -- Example: Smith, Johnson, ...

    -- Additional columns can be added as required.
);

-- Primary Skills Table: Manages employees' primary skills.
CREATE TABLE priskill (
    emp_id INT PRIMARY KEY,
    primary_skills VARCHAR(100),
    -- Primary key referencing empid.
    -- Example: Programming, Sales, ...

    -- Additional columns for skill levels, certifications, etc.
);
```

```
-- Location Table: Tracks employees' work locations.
CREATE TABLE location (
    emp_id INT PRIMARY KEY,
    work_location VARCHAR(100),
    -- Primary key referencing empid.
    -- Example: New York, London, ...

    -- Extra columns for location details if necessary.
);

-- The code defines five tables, each dedicated to specific employee data, enabling structured data
storage and retrieval.
```

### **Config.py**

```
# Configuration Parameters for Database and AWS S3 Storage

# Database Configuration
customhost = "http://employee.csrgslmtfzz7.ap-south-1.rds.amazonaws.com"
customuser = "awsproject"
custompass = "awsproject123"
customdb = "employee"

# AWS S3 Storage Configuration
custombucket = "addemployee"
customregion = "bombay"

# The code defines configuration parameters for connecting to an AWS RDS database and using an
AWS S3 bucket for storage.

# Database Configuration:
# - customhost: The endpoint URL of the AWS RDS database.
# - customuser: The username for database access.
# - custompass: The password for database access.
# - customdb: The name of the database to connect to.

# AWS S3 Storage Configuration:
# - custombucket: The name of the AWS S3 bucket for storing files, in this case, "addemployee."
# - customregion: The AWS region where the S3 bucket is located, in this case, "bombay."
```

The provided code defines configuration parameters for connecting to an AWS RDS database and using an AWS S3 bucket for storage.

In terms of its relation to the previous SQL code, these parameters are typically used in a server or application to establish a connection with the RDS database (specified by the customhost, customuser,

custompass, and customdb parameters) and to interact with an AWS S3 bucket (specified by the custombucket and customregion parameters). These configurations enable applications to store and retrieve data efficiently, making them a critical part of a cloud-based web application setup.

## CONCLUSION AND RESULT

In this project, we have successfully designed a structured database schema for storing employee data, enabling efficient data management and retrieval. The relational database model connects five tables—`empid`, `fname`, `lname`, `priskill`, and `location`—through the common `emp_id` column. This design ensures organized and accessible storage of employee details such as Employee ID, first name, last name, primary skills, and work locations.

Additionally, we have configured crucial parameters for connecting to an AWS RDS database and utilizing an AWS S3 bucket for file storage. The database parameters (`customhost`, `customuser`, `custompass`, and `customdb`) facilitate secure and efficient data access, while the AWS S3 configuration (`custombucket` and `customregion`) supports reliable file storage in the AWS cloud.

This project showcases the importance of structured database design and proper configuration for cloud-based applications, ensuring data integrity and accessibility.

The results of this project demonstrate the successful implementation of a relational database schema for employee data management. The tables `empid`, `fname`, `lname`, `priskill`, and `location` serve as organized repositories for key employee information, enhancing data retrieval and analysis.

Furthermore, the configuration parameters for the AWS RDS database and AWS S3 storage have been established. These parameters, including the RDS endpoint URL, database user credentials, and the AWS S3 bucket details, provide the necessary infrastructure for our cloud-based application. This setup allows for efficient interaction with the database and secure file storage in the AWS cloud environment.

Overall, the project results indicate the readiness of the system for the next phase, which includes web application development and integration with the structured database and cloud storage resources. This foundation sets the stage for a robust and scalable cloud-powered web application for employee data management.

## REFERENCES

- [1] "Intellipaat. ' AWS Projects for beginners.' YouTube, Aug. 6, 2020, [https://www.youtube.com/watch?v=7Gym2XVcA5A&ab\\_channel=Intellipaat](https://www.youtube.com/watch?v=7Gym2XVcA5A&ab_channel=Intellipaat).
- [2] Abdul Rehman "Title of the Article," Cloudways, Aprl. 11, 2023, URL: <https://www.cloudways.com/blog/aws-for-beginners/>.