# Employee Referral System

Software Architecture

Assignment – II

(**2021MT93578**)

Ayush Shashank

## Objective

A web application that allows the employees to refer an individual for a job at their organization.

- Search jobs for referral
- Execute referrals
- Track the status of referrals
- Reward employee on successful referral

## Utility Tree for ASR

| Quality Attribute | Attribute Refinement | ASR |
|---|---|---|
| Availability | Active Redundancy | An employee tries to access the application and the datacenter fails; the application loads from “backup” datacenter, 99% uptime (M, L) |
| Performance | Response Time | An employee tries to access the application; the application renders within 15 seconds (M,M) |
| | Throughput | Multiple users try to access the application at peak load time; the application (M,L) |
| Security | Confidentiality | An individual outside of the organization tries to access the application; the access is denied (H,H) |
| | Integrity | A nefarious individual tries to manipulate the application by injecting scripts into authenticated 3rd –party services; the action is be blocked (H,M) |

## Why are these Architecturally Significant

1. Availability – For an organisation globally spread the system needs to be available at all times to let the employees be able check on the updates on the referral at all times and the hiring team to receive resumes/referrals without any interruption due to unavailability.
2. Performance – The system needs to be  fast and efficient to felicitate seamless experience to the employees. Features of the system like search for open positions, upload of resume and fetching referral status should not take time and keep the employee engaged for long time potentially wasting billable hours.

# Why are these Architecturally Significant

3. Security – Only the employees of the organisation should be able to access the application hence authentication of the employees becomes necessary. The system also deals with sensitive information of the candidate (resume) which if leaks can harms the reputation of the organisation hence prevention of attacks like session hijacking, XSS or SQL Injection to the application should be prevented.

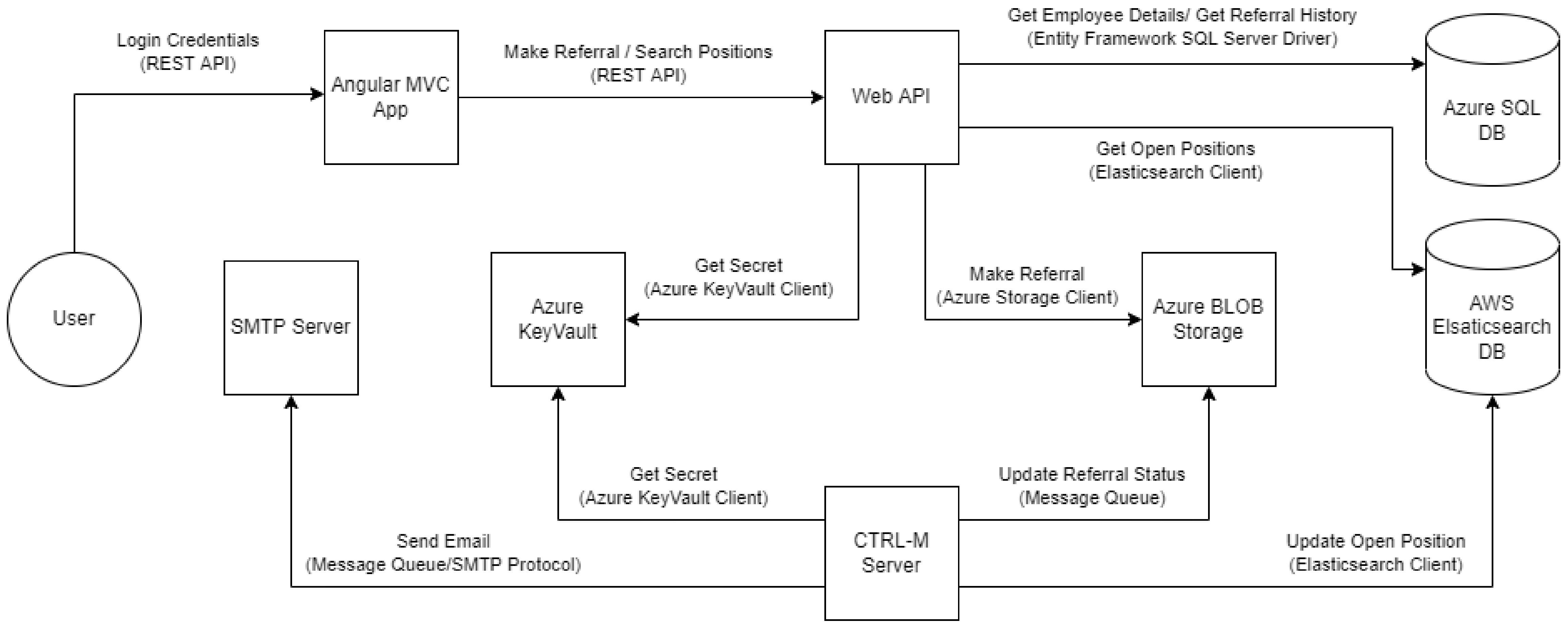
## Tactics Used To Achieve Top 5 ASR

- Security
  - Confidentiality: The system uses Azure Active Directory, a cloud-based identity and access management service, to **authenticate actors** in the scene. OAuth library is used to facilitate the login process.
    1. The user tries to access the website but is redirected by OAuth to Azure AD.
    2. The user enters the credentials in the login page to which Azure AD issues access token.
    3. OAuth redirects the access token to the web app where it is validated, and access is granted to the user.
  - Integrity: The system **limits access** to external domains and resources by applying a strict CORS policy and a strict Content Security Policy.
    1. Only specific domains are allowed to be accessed at a different origin.
    2. The Content Security Policy helps to detect mitigate certain types of attacks including XSS and data injection attacks.

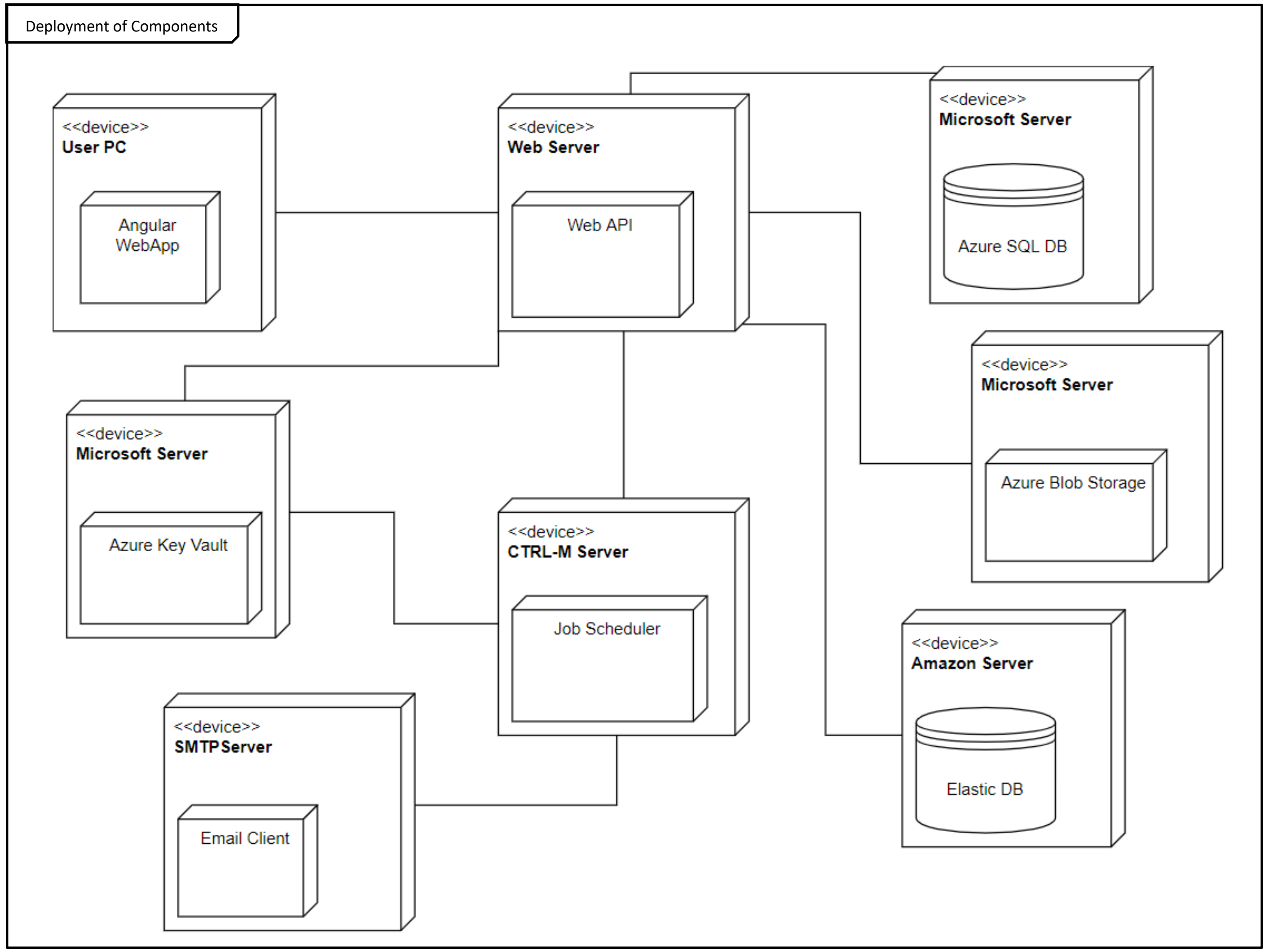
## Tactics Used To Achieve Top 5 ASR

- Performance
  - Response Time: Response images and scripts are compressed to **increase resource efficiency**. The resources are compressed in gzip format, optimising network bandwidth usage and increasing application responsiveness. To achieve this in the application Response Compression Middleware is used, with Compression Level as Optimal.
- Availability
  - Active Redundancy: The application is deployed in an Azure Availability Zone which ensures a 99.99% uptime as a Service Level Agreement and protects our data and application from datacentre failure. When the VM hosting the application goes down (fails) due to maintenance or any kind of error, traffic is redirected to the backup server hosting a copy of our application and data. This prevents a single-point-of failure.

# Component & Connection Diagram



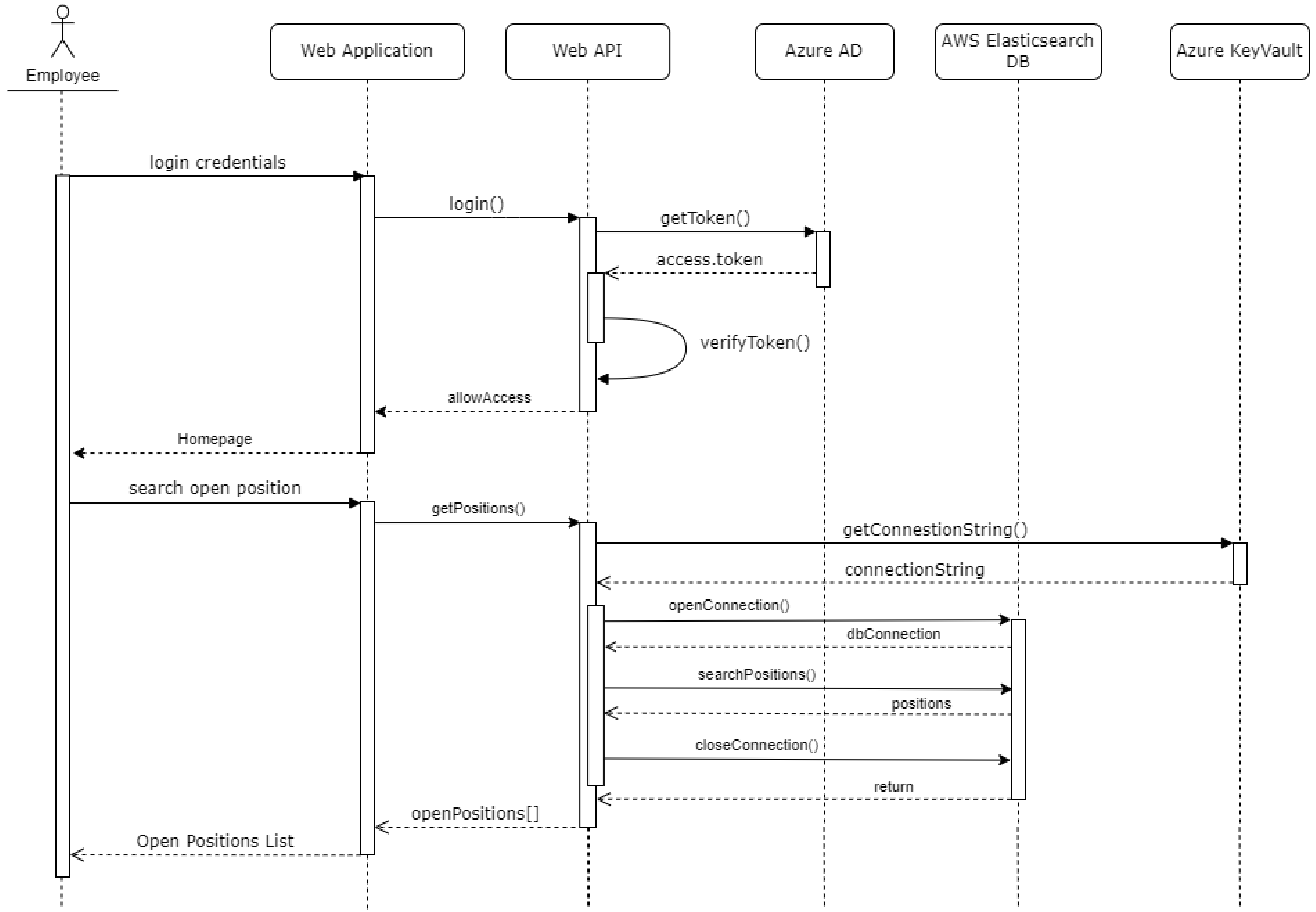
# Deployment Diagram



## Sequence Diagram

The scenario being represented in the diagram is a basic employee login and then search for open positions.

1. In login, the authentication takes place using Azure Active Directory, which returns an access token which when verified, prompts the Web Application to allow access to the Homepage.
2. The employee then searches for the open positions where the API is requested for the array list.
3. The Web API fetches the Elasticsearch DB connection string from Azure KeyVault.
4. After fetching the connection string, the Web API server creates a connection with the DB, queries the open position and then closes the connection.
5. After the connection is closed, the Web API return the open positions array to the Web Application which then displays the result in a tabular format to the user.


Employee
Web Application
Web API
Azure AD
AWS Elasticsearch DB
Azure KeyVault
login credentials
login()
getToken()
access.token
verifyToken()
allowAccess
Homepage
search open position
getPositions()
getConnestionString()
connectionString
openConnection()
dbConnection
searchPositions()
positions
closeConnection()
return
openPositions[]
Open Positions List

## Architectural Patterns Used

- Client-Server – The software is served as a web application which implements a client-server pattern, where the website is hosted on a server to which multiple clients connect. When a client tries to access the application, it connects with the server and downloads the requested resources from the server.
- Layered – The system is divided into three different layers, the presentation layer, the business layer and the data layer. The presentation layer is represented by the Angular Web Application, the business layer is represented by the Dot.NET Core Web API and the data layer is represented by Azure SQL Managed Instance and AWS Elasticsearch DB.

## Architectural Patterns Used

- Model-View-Controller – The web server implements an MVC pattern where the model represents the objects being dynamically created where data is populated from either the tables in the SQL Server database, using Entity Framework Core library, or from the from presented by the view. The view is rendered in synchronisation with the angular application which helps offload the effort of displaying all the data via the server. The controller accepts the requests coming from the view and performs interactions on the data model objects.

## Key Learnings

- Software architecture is extremely important and plays a vital role in the Software Development Life Cycle, makes it easy to understand the flow of the working of the application and is necessary for the quality of the product.
- Systems need to implement not just one but multiple architectural patterns in conjunction to achieve functional and non functional requirements in complex applications.
- Group Discussions about the architecture of the system helped a lot in understanding not just one’s own system but other systems as well. This helped in clearly understanding how common architectural decisions can be taken that could solve issues observed in multiple systems, even if the systems have entirely different application/use.