Day 1

Introduction

* In this internship, we will learn key technologies essential for modern software development, including SQL and PostgreSQL for managing and querying relational databases.
* AngularJS for creating dynamic and responsive front-end web applications.
* .NET for building robust backend services and APIs, and AWS for deploying and managing applications in the cloud.
* Through practical sessions and real-world projects, I aim to develop skills across the full stack, gaining hands-on experience in database management, web development, server-side programming, and cloud computing, which will prepare me for working on enterprise-level software solutions.

Session-1

* During the internship, I learned to use **pgAdmin**, a graphical tool for managing PostgreSQL databases.
* With pgAdmin, I can easily write and run SQL queries, create databases, tables, and select delete , insert, order etc many of them.

Day-2

* During the internship, I learned about **AngularJS**, a JavaScript framework used to build dynamic web pages by binding data and HTML together.
* As practice, I created a simple **Student Details Management** project, where I designed a dynamic page to display and manage student information. Using AngularJS directives like ng-model and ng-repeat etc. tags
* I could bind data directly to the HTML, making the content update automatically whenever the data changes.
* I did bind a student’s name and details to input fields, and display a list of students dynamically on the page.
* I also learned the difference between AngularJS and modern Angular (using TypeScript).
* AngularJS uses plain JavaScript and has a simpler, older architecture, while Angular uses TypeScript, offers better performance, a modular structure, and advanced features like components, services, and reactive programming, making it more suitable for large-scale applications.

Day-3

* Today, we learned about the .NET Framework, which is a powerful platform for building various types of applications, including web, desktop, and cloud-based solutions.
* The .NET Framework provides a large class library called the .NET Class Library (FCL) and a runtime environment known as the Common Language Runtime (CLR), which manages code execution, memory, security, and exception handling.
* We explored key components such as the Base Class Library, which includes ready-to-use classes for file handling, data access, collections, and more. We also learned about the Just-In-Time (JIT) Compiler, which converts intermediate language code into machine code during execution, improving performance.
* Another important feature we studied is the Garbage Collector (GC), which automatically manages memory by cleaning up unused objects.
* We started learning C# (C-Sharp), the main programming language used in .NET development. We covered basic syntax, data types, variables, conditionals and object-oriented programming concepts like classes and methods.
* To apply our knowledge, we built a small demo project. In this project, we created a simple C# program that allows the user to enter details like first name , last name, Gender, phone number, email, and password and then displays the information back on the console. This helped us understand how to write, compile, and execute C# code within the .NET environment.
* Through this session, we learned how .NET simplifies application development, ensures security, and offers great performance for building modern software solutions.

Day-4

* In internship training, we created a project named Mission, where we implemented user authentication and authorization to manage secure access.
* Authentication was handled using login and registration functionality, where users could sign up with their details and securely log in using their credentials.
* We used ASP.NET Identity to manage user accounts, password hashing, and login sessions.
* Authorization was implemented to control access based on user roles, such as Admin and User. For example, only Admins could access certain pages or perform actions like editing or deleting data. We used middleware and [Authorize] attributes in controllers to restrict access to specific actions.
* The project helped us understand how to build secure web applications using .NET, handle user sessions, manage roles, and protect data from unauthorized access. It was a valuable learning experience in applying real-world security concepts in a web application.

Day 5

* In our Mission project, we extended the application by adding repository and service layers for managing MissionTheme and MissionSkill entities.
* We developed MissionThemeRepository and MissionSkillRepository to handle database operations efficiently.
* To encapsulate business logic, we implemented the IMissionThemeService and IMissionSkillService interfaces, alongside services like AdminUserService, LoginService, MissionSkillService, MissionThemeService, and UserService.
* Controllers were created to manage HTTP requests for users, mission skills, and themes, providing clear API endpoints.
* We configured dependency injection in the Program.cs file to register all services and repositories, ensuring smooth application architecture.
* Additionally, we integrated Entity Framework Core with PostgreSQL to manage database interactions and set up the appsettings.json file for storing configuration details and connection strings.
* Finally, we included Swagger for generating API documentation and enabling convenient testing of our endpoints.

Day 6

* During our development work, we introduced several new features to enhance the functionality and user experience of the application.
* We implemented a Reset Password component featuring a structured HTML layout and custom CSS styling.
* This component uses a reactive form with validation for both password and confirm password fields, ensuring strong input control, and we integrated toast notifications to display success or error messages during the reset process.
* We also developed a Navbar Component to manage navigation across the application, incorporating user authentication status, displaying user details, and providing a dropdown menu for profile and logout options, along with unit tests to ensure its reliability.
* Additionally, we built a New Mission Component to allow users to create missions, featuring a modal for entering mission details, drag-and-drop support for uploading images and documents, and enhanced CSS styling for a better user interface.
* Unit tests were also included for all these components to maintain code quality and stability.

Day 7

* In our project, we developed complete CRUD functionality for managing missions. We created APIs and frontend components to create, read, update, and delete mission records in the system.
* One of the key features is fetching the mission name and displaying it dynamically in the mission list. We also implemented functionality to manage the active and inactive status of each mission, allowing admins to activate or deactivate missions based on requirements.
* The mission listing page shows all missions in a tabular format, displaying details like mission name, status, and other relevant information.
* We added filters and search options to quickly find specific missions. Users can edit mission details through a form, and changes are saved back to the database.
* We handled database operations using Entity Framework Core with PostgreSQL to ensure smooth data interactions.
* The frontend was developed using Angular for a responsive and user-friendly interface.
* Overall, building the mission CRUD helped us understand how to connect the frontend and backend, manage data efficiently, and provide a seamless user experience.

Day 8

* As part of the initial setup, we established the foundational structure of the Angular project. We configured environment files for both development and production to manage application settings efficiently.
* The main.ts file was implemented to bootstrap the Angular application, while polyfills.ts was included to ensure compatibility across different browsers.
* A basic HTML structure was defined in index.html, and we configured TypeScript settings through tsconfig.json files.
* For styling, we added a dedicated tabs.css file and placed necessary images in the assets directory for use across the application.
* Additionally, we set up the Angular testing environment to support unit testing of components and services. This setup ensures a scalable, maintainable, and ready-to-develop frontend application.

Day 9

* In this phase of the internship, we learned how to implement profile update functionality for both users and admins.
* We created editable forms where admin could update their personal details such as name, phone number, and profile image. For user, many of filds like name country name , city name, them , skill etc many this kind of details.
* we also learned about project deployment. We understood how to build and optimize the Angular application for production using ng build, and then deploy it to a server.
* For the backend, we deployed the .NET API and connected it to a PostgreSQL database.
* Configuration files like appsettings.json and environment variables were properly set up to support production.
* This session gave us hands-on experience in preparing and launching a full-stack application into a live environment.

Day 10

* During this stage of our internship, we explored AWS (Amazon Web Services) in detail, understanding its wide range of services for cloud computing.
* We learned about core AWS concepts like regions, availability zones, and the global infrastructure, which ensure high availability and scalability.
* We studied services such as EC2 (Elastic Compute Cloud) for launching virtual servers, and S3 (Simple Storage Service) for storing files and data securely in the cloud.
* We also explored RDS (Relational Database Service) for managing databases like PostgreSQL without handling server infrastructure manually.
* We learned how IAM (Identity and Access Management) helps manage user permissions and secure access to AWS resources.
* Another key concept was VPC (Virtual Private Cloud), which allows us to isolate resources and control network configurations.
* As practical work, we deployed our project using Amazon ECS ,which is used to run and manage Docker containers at scale.
* We created task definitions, services, and clusters to deploy our application smoothly. We configured load balancing and networking so that our deployed mission project was accessible over the internet.