METAGEEKS INTERNSHIP PROGRAM-2025

INTERNSHIP REPORT

INTERN NAME: AKILESH S

EMPLOYEE ID: 2629355

INTERNSHIP DURATION: APRIL – MAY

REPORT SUBMISSION DATE: 29/05/2025

SUPERVISOR: PRABHAKAR S

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who supported me throughout the course of my internship.

First and foremost, I am deeply thankful to my supervisor, **PRABHAKAR S**, for their invaluable guidance, encouragement, and constant support. Their expertise and feedback greatly enhanced my learning experience.

I also extend my thanks to the entire team at **Mphasis** for providing me with a conducive environment to apply my skills and learn new technologies. Special thanks to my colleagues and mentors who patiently helped me overcome challenges and motivated me to improve continuously.

Finally, I am grateful to **Mphasis** for facilitating this internship opportunity and for their continuous support throughout my academic journey.

AKILESH S

29/05/25

ABSTRACT

This report presents a comprehensive overview of the knowledge and skills acquired in the fields of Agentic Artificial Intelligence (AI), front-end web development technologies including HTML, CSS, JavaScript, and Angular framework, as well as cloud computing with Amazon Web Services (AWS).

The study of Agentic AI focused on understanding autonomous systems capable of decision-making and task execution with minimal human intervention. In parallel, web development competencies were enhanced through hands-on experience with foundational technologies HTML and CSS for structuring and styling web pages, JavaScript for interactive client-side scripting, and Angular for building dynamic, scalable single-page applications. Additionally, proficiency in AWS cloud services was developed, emphasizing deployment, management, and scalability of applications in cloud environments. This integrated learning approach has equipped me with the technical expertise to design, develop, and deploy intelligent, responsive web applications leveraging modern cloud infrastructure.

INTRODUCTION

In the modern era of web and cloud technologies, building scalable, maintainable, and high-performance applications requires a blend of advanced front-end frameworks, state management solutions, and cloud infrastructure. This report explores the comprehensive learning and implementation of core web development technologies such as HTML, CSS, and JavaScript, advanced front-end development using the Angular framework with NgRx for state management, and cloud deployment using Amazon Web Services (AWS).

HTML (HyperText Markup Language) and **CSS (Cascading Style Sheets)** form the foundational languages for creating and styling web pages. HTML structures the content on the web, while CSS defines the look and feel, enabling developers to create visually appealing and responsive designs. **JavaScript** complements these by adding interactivity and dynamic behavior, allowing web pages to respond to user actions in real time.

At the heart of modern front-end development, **Angular** is a powerful TypeScript-based framework designed for building dynamic single-page applications (SPAs). Angular offers a

component-based architecture that promotes modularity and reusability, two-way data binding for real-time UI updates, and dependency injection to manage services efficiently. It provides robust tooling for testing, routing, and form handling, which together accelerate development cycles and improve application quality.

A key aspect of managing complex Angular applications is handling state—especially as applications grow larger and data interactions become more intricate. This is where **NgRx**, a reactive state management library inspired by Redux, comes into play. NgRx leverages RxJS (Reactive Extensions for JavaScript) to provide a unidirectional data flow, immutable state, and powerful side-effect handling via effects. Implementing NgRx enables better predictability, easier debugging, and maintainability in applications by centralizing the app's state and allowing clear separation between UI and business logic.

In addition to front-end development, understanding and utilizing **Amazon Web Services (AWS)** is crucial for deploying and scaling applications in the cloud. AWS offers a comprehensive suite of cloud services such as EC2 for compute resources, S3 for

storage, Lambda for serverless computing, and RDS for managed databases. Using AWS, developers can ensure their applications are highly available, fault-tolerant, and capable of handling variable loads with ease.

This report emphasizes not only mastering the individual technologies but also integrating them to build robust, intelligent web applications. Combining Angular's component-based architecture with NgRx's state management allows for scalable and maintainable front-end development, while leveraging AWS cloud services ensures seamless deployment and operational scalability. This integrated approach equips developers to deliver performant, reliable, and user-friendly applications in today's competitive digital environment.

CHAPTER 1

TIMELINE

S. No	Month	Week	Topics Covered
1	April	1st	Agenti AI
2		2nd	Agentic application in transport and logistics field
3		3rd	Cloud Fundamentals
4		4th	Node.js
5	May	1st	Node.js
6		2nd	HTML,CSS,JS
7		3rd	Angular and Ngrx
8		4th	AWS services(Lambda,AppSync,DynamoDB)

CHAPTER 2

AGENTIC AI

Introduction:

Agentic Artificial Intelligence (Agentic AI) refers to autonomous systems capable of perceiving their environment, making decisions, and taking actions independently to achieve specific goals. These AI agents operate with minimal human intervention and are designed to adapt dynamically to changing environments. The study of Agentic AI encompasses principles from machine learning, cognitive science, and robotics, aiming to develop intelligent systems that can act as self-directed agents.

Key Concept of a Agentic AI:

Agentic AI is built on several foundational concepts such as autonomous decision-making, where agents use algorithms like reinforcement learning and planning to determine optimal actions. It involves the design of multi-agent systems that interact cooperatively or competitively within shared environments. Key principles include state representation, reward

maximization, policy learning, and environment modeling to allow agents to perform complex tasks and solve problems effectively.

Agentic AI Applications:

Objectives

The primary objectives of implementing Agentic AI include enhancing automation by reducing human supervision, improving adaptability in dynamic conditions, and enabling intelligent decision-making in complex systems. Applications span robotics, autonomous vehicles, smart assistants, game AI, and industrial automation where agents must act proactively and responsibly.

Tools and Technologies used

Development of Agentic AI relies on a range of tools and technologies such as Python programming, TensorFlow and PyTorch frameworks for machine learning, OpenAI Gym for reinforcement learning environments, and Robot Operating System (ROS) for robotics integration. Simulation platforms and

custom algorithms are used to train, test, and deploy agents in real or virtual environments.

Roles and Responsibilities

In Agentic AI projects, AI researchers focus on algorithm development and theoretical modeling. Data scientists manage data collection and processing for training. Software engineers integrate AI models into applications, while system architects design the overall infrastructure to ensure agent scalability and reliability. Cross-functional collaboration is crucial for developing robust autonomous systems.

Challenges Faced

Developing Agentic AI systems involves challenges such as ensuring reliable decision-making in uncertain and incomplete environments, managing the exploration-exploitation trade-off in learning, and addressing safety and ethical considerations related to autonomous actions. Real-time responsiveness, scalability, and interpretability of agent behaviors are ongoing research and engineering hurdles.

Skills Gained

Through studying Agentic AI, valuable skills have been acquired, including designing and implementing autonomous agents, applying reinforcement learning algorithms, modeling complex environments, and integrating AI agents with hardware or software platforms. Additionally, the ability to analyze agent performance, debug learning processes, and address ethical implications has been developed, equipping one to contribute effectively in advanced AI-driven projects.

CHAPTER 3

CLOUD FUNDAMENTALS

INTRODUCTION:

Cloud computing is a model for delivering computing resources such as servers, storage, databases, networking, software, and analytics over the internet ("the cloud"). It enables flexible, on-demand access to these resources without the need for owning and maintaining physical infrastructure. Cloud platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform have transformed how applications are developed, deployed, and scaled by providing high availability, scalability, and cost-efficiency.

Benefits of Using Cloud Computing:

During my internship, I gained practical experience working with cloud fundamentals, focusing primarily on AWS services. Leveraging cloud technologies improved the agility and scalability of the applications and data pipelines I worked on. Some key benefits I observed include:

- **On demand Resource Provisioning**

  Quickly set up virtual machines (EC2 instances), storage (S3 buckets), and databases without upfront hardware investments, enabling faster development and deployment cycles.

- **Scalability and Flexibility**

  Automatically scale resources up or down based on demand, ensuring applications handle varying workloads efficiently without over-provisioning or downtime.

- **Cost Efficiency:**

  Pay-as-you-go pricing models helped optimize budget usage by charging only for the resources consumed during operations, reducing unnecessary expenses.

- **Managed Services and Automation:**

  Services like AWS Lambda for serverless computing and AWS RDS for managed databases simplified infrastructure management, allowing me to focus more on application logic and less on maintenance.

Practical Applications During Internship:

- Deployed data processing pipelines using AWS EC2 and S3 for storage.
- Utilized AWS Lambda functions to automate specific workflows without managing servers.
- Configured AWS IAM roles and policies to enforce secure access controls.
- Used AWS CloudWatch for monitoring application performance and troubleshooting.

CHAPTER 4

NODE.JS

Node.js is an open-source, cross-platform runtime environment that allows developers to run JavaScript code outside a web browser. Built on Google Chrome's V8 JavaScript engine, Node.js enables server-side scripting, making it possible to build scalable and high-performance network applications using JavaScript.

Objectives:

- Gain hands-on experience with **Node.js** and **Express.js**.
- Design and implement REST APIs.
- Work with **MongoDB** and handle real-time data.
- Learn to deploy applications and manage version control.

Roles and Responsibilities:

- Developed and maintained backend logic using **Express.js**.
- Created endpoints for CRUD operations.

- Integrated external APIs (e.g., SMS, payment gateways).
- Implemented **JWT-based authentication** and middleware for security.
- Participated in code reviews and daily standups.

Tools and Technologies:

- **Languages**: JavaScript
- **Framework**: Express.js
- **Database**: MongoDB
- **Tools**: Postman, Git, VS Code

Challenges

- Handling asynchronous operations effectively using Promises and async/await.
- Structuring modular code for scalability.
- Understanding MongoDB schema design and performance tuning.

CHAPTER 5

HTML & CSS

This report presents an overview of my experience in learning HTML and CSS, focusing on the fundamental building blocks of web development and the styling techniques necessary for creating engaging and responsive web pages. I explored core HTML concepts such as semantic markup, element structuring, and form handling, alongside CSS topics including selectors, the box model, layout systems, and responsive design principles. Through practical exercises and mini-projects, I developed the ability to build well-structured, accessible, and visually appealing websites.

Goals & Objectives

- Develop a strong understanding of HTML structure and semantic elements.
- Master CSS styling techniques, including selectors, properties, and layout methods.

- Build responsive web pages by applying CSS Flexbox, Grid, and media queries.
- Implement accessible forms and interactive UI components using HTML and CSS best practices.

LEARNING JOURNEY

HTML basics

- **Document Structure:** Learned the anatomy of an HTML document including <!DOCTYPE>, <html>, <head>, and <body> tags.
- **Elements & Tags:** Explored various HTML elements such as headings, paragraphs, lists, images, and links.
- **Attributes:** Used essential attributes like id, class, href, src, and alt to enrich elements with additional functionality and accessibility.
- **Semantic HTML:** Understood the importance of semantic tags such as <header>, <nav>, <main>, <section>, <article>, and <footer> to improve SEO and accessibility.

- **Forms & Inputs:** Created interactive forms using input elements, labels, buttons, and implemented basic validation.

CSS STYLING & LAYOUT

- **Selectors & Properties:** Mastered CSS selectors (element, class, ID, attribute) and key properties to style text, colors, backgrounds, borders, and spacing.
- **The Box Model:** Gained a deep understanding of content, padding, border, and margin and how they affect element sizing and layout.
- **Typography & Colors:** Applied font families, sizes, weights, and color schemes to enhance readability and visual appeal.
- **Layout Techniques:** Practiced creating layouts using display properties such as block, inline-block, and learned modern layout methods including Flexbox and CSS Grid.
- **Responsive Design:** Utilized media queries to adapt designs across different screen sizes and devices, ensuring mobile-friendly interfaces.

- **Transitions & Effects:** Added smooth hover effects and transitions to improve user interaction and feedback.

OUTCOME

Proficiency in HTML enables developers to create well-structured and meaningful web content, laying a solid foundation for accessible and SEO-friendly web pages. It ensures that websites have a clear, logical hierarchy that browsers and assistive technologies can interpret easily.

Expertise in CSS allows developers to bring web pages to life with attractive and responsive designs. It provides the tools to control layout, typography, colors, and animations, delivering a seamless user experience across various devices. Together, HTML and CSS empower developers to build visually engaging, user-friendly, and maintainable websites that meet modern web standards.

CHAPTER 6

JAVASCRIPT

JavaScript is the foundational scripting language of the web, enabling dynamic and interactive content in browsers and server environments. Its flexibility, event-driven nature, and vast ecosystem make it essential for building modern web applications, from simple scripts to complex frameworks.

GOALS & OBJECTIVES

- Gain a strong understanding of JavaScript syntax, core language features, and programming paradigms.
- Explore fundamental concepts such as functions, objects, and asynchronous programming to write efficient code.
- Understand how to manipulate the Document Object Model (DOM) to create interactive user interfaces.
- Apply JavaScript in real-world projects to build dynamic, responsive web applications.

KEY CONCEPTS

Core JavaScript Essentials

- Dynamic Typing: Variables can hold values of any type, allowing flexible data manipulation.
- Functions & Scope: Define reusable logic blocks with function declarations, expressions, and arrow functions, managing variable visibility via scope and closures.
- Objects & Prototypes: Use objects as collections of key-value pairs and leverage prototype-based inheritance for reusable behavior.
- Events & DOM Manipulation: Handle user interactions and update web page elements dynamically.

Advanced Capabilities & Development Practices

- Asynchronous Programming: Manage delayed operations with callbacks, Promises, and async/await syntax for smoother UX.

- Modules (ES6+): Organize code with import/export syntax for maintainability and reusability.
- Closures & Higher-Order Functions: Enable powerful abstraction by capturing lexical scope and passing functions as arguments or returning them.
- Error Handling: Use try/catch blocks and custom error objects for robust and fault-tolerant code.
- Event Loop & Concurrency: Understand JavaScript's single-threaded model and how asynchronous tasks are handled via the event loop.

OUTCOME

Mastering JavaScript empowers developers to build interactive, high-performing web applications with a deep understanding of the language's quirks and strengths. It forms the backbone of front-end and back-end development, enabling versatile and creative solutions aligned with modern web standards.

CHAPTER 7

ANGULAR

This highlights my hands-on exploration of Angular, a leading front-end framework for crafting dynamic and responsive web applications. I gained practical experience with Angular's component-driven architecture, two-way data binding, service injection, routing strategies, reactive programming, and state management using NgRx, enabling me to build maintainable, scalable, and high-performance applications.

Goals & Objectives

- Develop a foundational understanding of Angular's ecosystem and architecture.
- Master core concepts including components, directives, services, modules, and forms.
- Build scalable applications using dynamic forms, route navigation, state handling, and centralized state management with NgRx.

- Execute efficient CRUD operations through Angular's HTTP client integrated with REST APIs.
- Apply coding standards and optimization techniques to improve performance and scalability.

LEARNING JOURNEY

Angular Fundamentals

- Components & Templates: Created reusable UI components with TypeScript-backed templates.
- Directives: Leveraged built-in and custom directives for dynamic rendering and styling.
- Data Binding: Implemented property, event, and two-way binding for real-time interactivity.
- Dependency Injection: Shared logic and data through service injection across modular components.
- Form Handling: Built forms using reactive and template-driven approaches with validation logic.

Routing & Navigation

- Router Configuration: Set up client-side navigation with route definitions and lazy loading.
- Route Protection: Ensured secure access using route guards tied to authentication logic.

State Management & Async Data Handling

- NgRx Store: Implemented centralized state management using NgRx to handle application state predictably and efficiently.
- Actions, Reducers & Effects: Created actions for dispatching events, reducers to update state immutably, and effects to handle side effects like API calls.
- Selectors: Used selectors to derive and expose slices of state for components.
- RxJS & Observables: Utilized observables to manage asynchronous data streams and event handling.

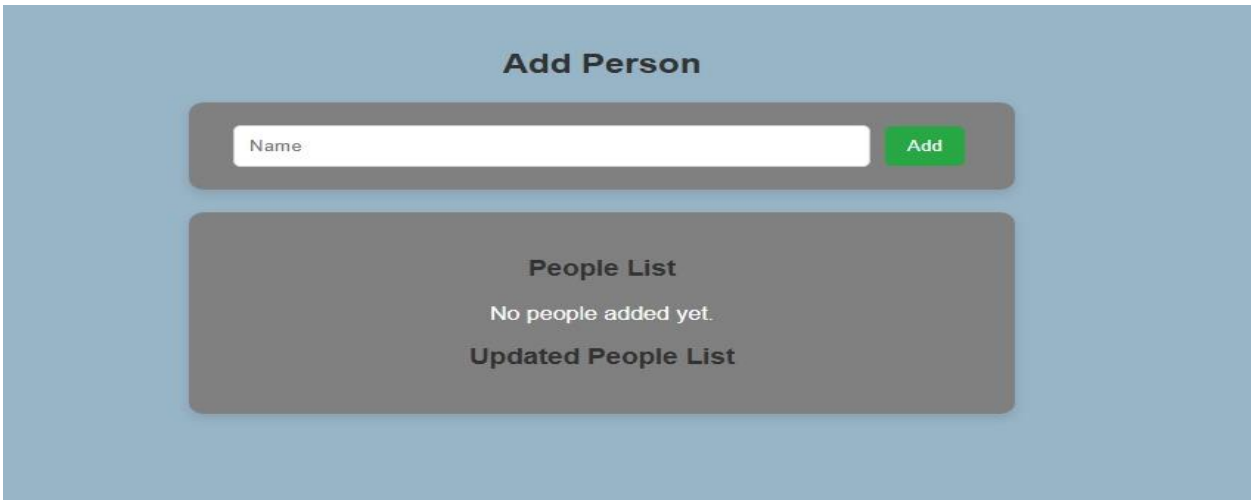
- HTTP Client Integration: Connected frontend to backend services with HTTP methods (GET, POST, PUT, DELETE).
- Change Detection: Improved UI performance through optimized change detection strategies and selective component rendering.

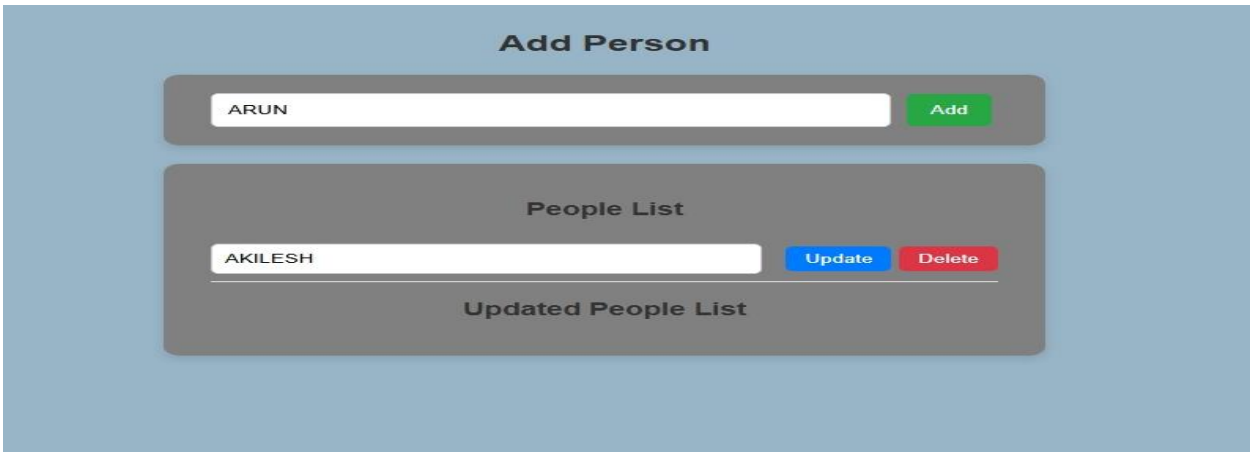
Project: Person List – CRUD with Angular & NgRx

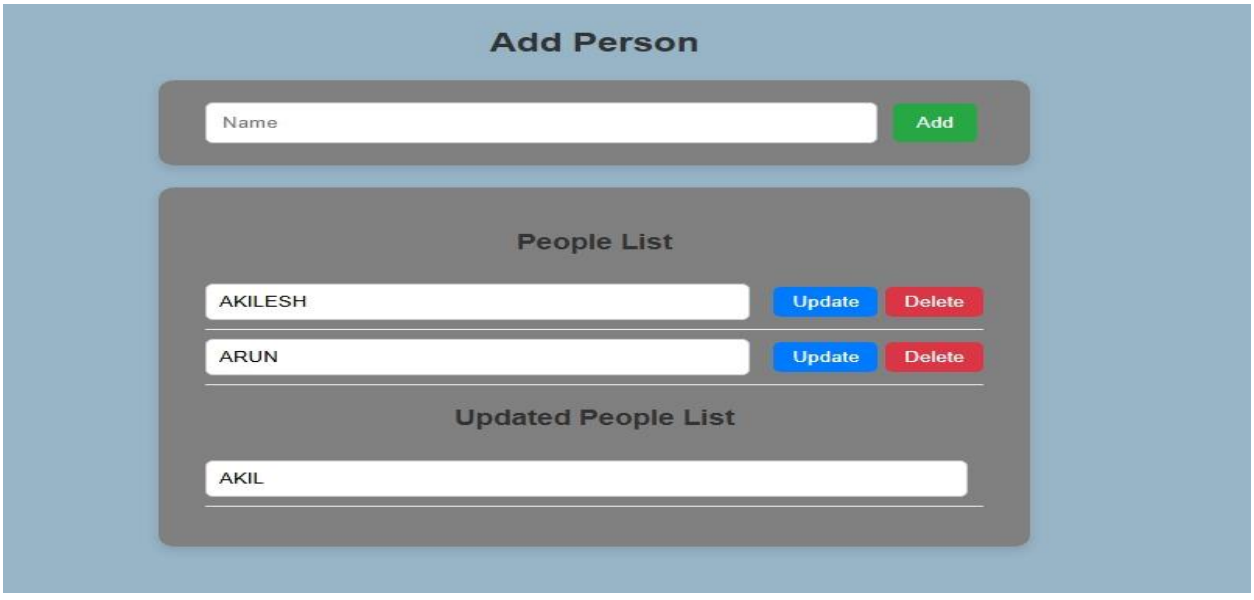
Objective:

Develop a dynamic web application using Angular that allows users to manage a list of persons. The app supports full CRUD operations (Create, Read, Update, Delete) and leverages **NgRx** for state management to maintain a consistent and predictable application state.

OUTCOME:


Add Person
Name
Add
People List
No people added yet.
Updated People List


Add Person
ARUN
Add
People List
AKILESH
Update
Delete
Updated People List


Add Person
Name
Add
People List
AKILESH
Update
Delete
ARUN
Update
Delete
Updated People List
AKIL

CHAPTER 8

AWS CLOUD SERVICES

AWS Lambda, AppSync, and DynamoDB are foundational cloud services that enable serverless, scalable, and efficient application development on AWS. Lambda provides event-driven compute capabilities without managing servers. AppSync simplifies building GraphQL APIs by connecting data sources securely and in real-time. DynamoDB offers a fast, fully managed NoSQL database service optimized for low-latency and high-throughput applications. Together, they empower developers to build modern, resilient, and highly scalable backend solutions.

Goals & Objectives

- Understand the core features and architecture of AWS Lambda, AppSync, and DynamoDB.
- Learn how to create and deploy serverless functions with Lambda.

- Explore GraphQL API development and real-time data synchronization using AppSync.
- Gain proficiency in modeling and querying data with DynamoDB's key-value and document store.
- Integrate these services to build seamless serverless applications.
- Apply best practices for performance optimization, security, and cost management.

Key Concepts

Core AWS Lambda Essentials

- Event-driven Compute: Execute code in response to events from various AWS services or HTTP requests without provisioning servers.
- Stateless Functions: Write lightweight, ephemeral functions that scale automatically based on demand.
- Triggers & Integrations: Connect Lambda to a variety of event sources such as S3, DynamoDB Streams, API Gateway, and more.

- Resource Configuration: Define memory, timeout, and concurrency limits to optimize performance and cost.

Core AWS AppSync Essentials

- Managed GraphQL APIs: Build flexible and scalable APIs with GraphQL for querying multiple data sources in a single request.
- Real-time Data & Subscriptions: Enable real-time updates with WebSockets and GraphQL subscriptions.
- Resolvers & Data Sources: Map GraphQL queries and mutations to DynamoDB, Lambda, or HTTP endpoints.
- Authorization & Security: Implement fine-grained access control using AWS IAM, Cognito, or API keys.

3. Core DynamoDB Essentials

- NoSQL Database: Store and retrieve data with single-digit millisecond latency at any scale.
- Flexible Schema: Use key-value and document data models with support for nested attributes.

- Primary Keys & Indexes: Design tables with partition keys and optional sort keys; use global and local secondary indexes for optimized queries.
- Streams & Triggers: Capture data changes and trigger Lambda functions for reactive processing.

Advanced Capabilities & Development Practices

- Lambda Layers & Versioning: Modularize shared code and manage function versions for safer deployments.
- AppSync Pipeline Resolvers: Build complex data fetching workflows with multiple resolver steps.
- DynamoDB Transactions: Ensure atomic, consistent operations across multiple items and tables.
- Monitoring & Debugging: Use CloudWatch Logs, X-Ray tracing, and AWS SDK metrics for observability.
- Security Best Practices: Implement least privilege IAM roles, encryption at rest and in transit, and API authorization strategies.

OUTCOME

Mastering AWS Lambda, AppSync, and DynamoDB enables developers to architect serverless applications that are scalable, performant, and cost-efficient. This knowledge promotes building real-time, event-driven systems with minimal operational overhead. Developers can innovate faster by focusing on business logic while leveraging AWS managed services for infrastructure, security, and reliability aligning with modern cloud-native development standards.

CONCLUSION

My internship experience has been an invaluable journey that significantly enhanced my technical skills and practical understanding of key IT concepts. Throughout the internship, I gained hands-on experience with cutting-edge technologies such as Agentic AI and its applications in the transport and logistics sector, which broadened my perspective on how AI can optimize real-world operations.

I developed a strong foundation in Cloud Fundamentals and became proficient in working with essential AWS services like Lambda, AppSync, and DynamoDB, enabling me to design scalable and efficient serverless applications. Additionally, I deepened my programming skills with Node.js, mastering backend development and API integration.

On the frontend side, I honed my expertise in HTML, CSS, and JavaScript, while exploring modern frameworks and state management using Angular and NgRx. This comprehensive exposure to both frontend and backend technologies has equipped

me with the versatility to contribute effectively across the full software development lifecycle.

Overall, this internship has not only strengthened my technical capabilities but also improved my problem-solving, collaboration, and project management skills. I am confident that the knowledge and experience I have acquired will serve as a strong foundation for my future career in IT, empowering me to take on complex challenges and contribute to innovative technology solutions.