Internship at Auracore Tech as Web Developer

by

Abdullah Jamil Sifat
20301386

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
October 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Abdullah Jamil Sifat
20301386

Approval

The internship titled "Internship at Auracore Tech as Web Developer" submitted by

1. Abdullah Jamil Sifat (20301386)

of Summer 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in October 10, 2025.

Examining Committee:

Supervisor:

(Member)

Md. Tawhid Anwar

Senior Lecturer
Computer Science And Engineering
BRAC University

Co-Supervisor:
(Member)

Intejar Meherab

Chief Technical Officer and Web Lead
Auracore Tech

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science And Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Participants may grow their skills, expertise, and communication skills by hands-on tasks in real-world settings through internships, offering individuals vital chances for personal development beyond the traditional educational environment. The objective of Aura Core Tech's internship initiative at Brac University is to offer students an actual experience of job settings, promote the incorporation of theoretical concepts to practical issues, and offer them with the abilities that they will require for their professional futures. During internships, students may familiarize themselves with a particular field of work, gain valuable work experience, develop their professional connections, and improve their probability of obtaining a permanent or temporary position with the internship hosting firm. These products include a wide range of topics, such as affiliate marketing, ad networks, digital marketing, online retail, company management, and the development of websites and mobile applications. For students, internships provide several substantial advantages.
First of all, companies lead them to a particular occupation that is illuminating, enabling students to acquire an extensive knowledge of the field they've decided. Additionally, internships provide undergraduates with valuable opportunities to acquire knowledge and enhance their expertise. Required skills, promoting personal and professional growth. Finally, internships make it less difficult to acquire practical work experience, which is crucial for job prospects in the future. Students may additionally establish professional connections in their chosen field via internships, which expands their circle of contacts. Last but not least, internships greatly increase the possibility of obtaining an internship with the organization that hosts them, thereby improving students' employment opportunities. In short, internships offer graduates with valuable practical educational opportunities which encourage intellectual growth and provide them with the knowledge and skills required to be successful in their future careers.

Keywords: Web development, MVC, ASP.Net, Python, AI Agent, FastAPI

Dedication

I would like to dedicate all my academic achievements and difficulties to my wonderful parents for their constant happiness. I also would like to express my appreciation to my excellent supervisor INTEJAR MEHERAB (CTO), for providing guidance and encouraging me during my internship, which helped me grow as an experienced individual.

Acknowledgement

Firstly, I would thank almighty Allah that the internship ended without much difficulties. Additionally, gratitude to my supervisor,Tawhid Anwar sir, for his valuable guidance and assistance throughout my internship period. Whenever I requested help, he was ready to help me. And finally, despite my parent's constant encouragement, it would not have been achievable. I am nearing completion of my degree and I am grateful for their kind support and blessings.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AI	Artificial Intelligence
API	Application programming interface
ERP	Enterprise Resource Planner
IDE	integrated development environment
IOT	Internet of things
JS	Java Script
QA	Quality Assurance
SDK	Software Developement Kit

Chapter 1

Introduction

1.1 About Internship

An internship provides students and fresh graduates alike a chance to get suited to practical work space practices and be able to implement their knowledge in real time. Students get the opportunity to gain experience that is related to their field of study. In simple terms, this helps the student's transfer from the intellectual environment into the real-world environment of employment. Also, the employers will possess a more thorough comprehension of the newly hired intern that will be prepared to allocate them to the most appropriate roles depending on what they require. In context of all of this, doing an internship additionally enhances the student's work background, which improves the resume and ultimately raises its value, opening up excellent potential for future employment.
Some students would rather do an internship than prepare a thesis paper, and internships are growing a lot more prevalent. Students at BRAC University are asked to complete a paper on their thesis or complete an internship. Although there could not be a lot of learners selecting to take part in BRAC internships, those who decide to participate received sufficient support through the business and are urged to obtain as many things as feasible from their job. As long as the topic relates to their degree, students are allowed to select any specialization for their internship. Students need to complete in excess of 72 university credits in order to become eligible for internships, which should be for six months at a reputable company.
As a BRAC University student, I decided to do an internship instead of writing a thesis. I was selected to pursue a position as a Web development intern at Aura-Core Tech.The objective of Auracore Tech aims to use technologies to encourage companies and contribute to a more intelligent, connected to one another, and environmentally friendly future. The company develops its position as a leader in the rapidly changing field of technological change by extending the limits of creativity and maintaining a desire for excellence.

1.2 About this Report

As fulfilling this internship is a prerequisite for obtaining my bachelor's degree, I have to hand in a report which will be considered a basis for my evaluation at BRAC University. I have written regarding the culture and environment of my place of work in the report I wrote. I wrote about my experience at AuraCore Tech and what I

learned there. A detailed overview of the internship's tasks and the assignment I worked on will also be presented.

1.3 Objective

The information that follows represents what this report aims to provide its readers:

1.3.1 Aim

The main objective of this report is to demonstrate my job experience and the knowledge I obtained.

1.3.2 Specific Goals

The goals for this specific report:

- Portray the company profile in details
- Describing the working environment
- To put forward all the services AuraCore Tech provides to its customers
- Explain the type of work AuraCore Tech does
- Portrays about my the things I learned in AuraCore tech during my internship
- To present an in-depth description for all the tasks I performed on the job that was provided to me

1.4 Methodology

My work experience including the lessons that I acquired throughout my internship at AuraCore Tech serve as the major topics of the paper that follow. Most of the content and data provided is based on my own personal knowledge. Additionally, a few things had been extracted from particular websites.

1.4.1 Primary Data

- My learning expiring from my work
- Conversation along with knowledge acquired from peers and team leaders
- Team meetings

1.4.2 Secondary Data

- AuraCore Tech official website
- Online Sources

Chapter 2

Company Profile

2.1 Overview

Auracore Tech is an innovative development company featuring an emphasis on personalized strategies, AI, Blockchain, ERP, IoT, and SaaS. They focus on developing reasonable, accessible digital products that benefit organizations of every kind. With an ongoing commitment to innovative technology, long-term viability and ethical standards, Auracore supports its clients in prospering in the rapidly evolving digital environment.

2.2 Vision and Mission

Through offering innovative, scalable options that complement customers' goals, Auracore Tech aims to become an expert in AI, Blockchain, ERP, and IoT. The objective of the business is to establish a more smart and connected future by prioritizing ethical behavior, sustainability, and transformation.

2.3 How Auracore Tech Limited Work

Auracore Tech Limited focuses on providing complete technological solutions in software development, AI, blockchain, ERP, and IoT. They work alongside clients to turn innovative ideas into technological alternatives that are efficient, scalable, and prepared for the future.

2.3.1 Partners and Sponsors

Details about AuraCore Tech's partners and customers are mostly kept confidential.

2.3.2 Program Management

The present growth of AuraCoreTech is administered through the "Program Manager." This individual should be engaged in all technological conversations as they are the main source for communication on an administrative level. The Program Manager has accountability for offering and developing Customer Interaction Design and Quality Assurance services that comply with the Product Owner's requirements.

2.3.3 Product Ownership

The customer selects a "Product Owner" that will gather requirements from everybody engaged daily and deliver the information in an ordered manner. AuraCore Tech's shareholders and financiers are primarily customers.

2.3.4 Quality Assurance

Software testing, commonly referred to as quality assurance, is an approach used to ensure the level of quality of the software products or services that an organization provides its clients. Making the applications manufacturing method easier and effective while conforming to the standard of excellence standards set for applications is the focus of quality management. Quality Assurance can also be referred to as QA Testing. Management teams at AuraCore Tech assign quality assurance technicians to particular projects according to their requirements.

2.3.5 User Experience Design

The objective of Auracore Tech's user experience (UX) design technique is to offer easy, intuitive interfaces that truly engage with people. Through such as seamless navigation, simple arrangements, and appealing visuals, they optimize both desktop and mobile devices using technologies like Figma. They make sure each service provides efficient, flexible, and visually appealing digital interactions by merging intensive client seminars, continuous wireframing, front-end design, and testing for usability.

2.4 What Auracore Tech Do

Through end-to-end development, smart automation, advertising, design, and cloud support, AuroraTech delivers your project from concept to launch—and beyond—with the goal of providing scalable, focused on users, and efficient digital solutions.

2.5 Software Development Process

To fulfill customer goals, Auracore Tech starts with a comprehensive review of requirements and planning of strategies. To ensure an effortless experience for users, they employ tools like Figma to create appealing UI/UX. Using modern frameworks like React, Flutter, Node.js, and Django, full-stack development comes next. To ensure efficiency and safety, extensive testing along with quality assurance is carried up. The item has been put on scalable cloud services after testing. To preserve the software's reliability and efficiency after its launch, they provide ongoing maintenance, updates, and support for technical issues.

2.5.1 Requirement Identification

Involves collaborating closely with clients at Auracore Tech to learn about their company's objectives, challenges, and customer needs. To determine an accurate scope of the project, they gather detailed technical and functional specifications

via meetings, interviews, and analysis. This phase ensures that the development team has an organized strategy to develop solutions that actually meet customer requirements and provide tangible advantages.

2.5.2 Context and Process Capturing

Auracore Tech demands an in-depth knowledge of current workflows, company procedures, and surroundings for the customer's company. They analyze the operation of the present technologies and identify any flaws or problems. For it to accurately reflect the environment in which the newly developed application will function, this involves mapping out user experiences, data flows, and interaction. Through this, they ensure that the proposed solution enhances the overall effectiveness and integrates seamlessly with the client's natural environment.

2.5.3 Proof of Concept (POC)

In the very first phase, Auracore Tech develops a compact model or concept that verifies the feasibility of the suggested approach and its critical components. In advance of complete growth and development, this helps in proving important speculation, demonstrating efficiency and getting early feedback from stakeholders. The proof of concept (POC) assures the project's technical as well as economic viability, eliminates risks, and balances requirements.

2.5.4 Eventual Releases

Auracore Tech describes the application product's phased distribution throughout multiple versions or modifications. They supply clients with functional improvements for practical use and input post initial testing and development phases. Continuous enhancement and quick versatility have been rendered feasible by this iteration procedure, and it additionally assures that the final outcome meets user requirements as well as quality standards ahead of being released.

2.6 Recruitment Process

Job listings on Auracore Tech's websites as well as additional sources act as the initial phase in the recruitment procedure. Following that, applications are reviewed to generate a short list that includes suitable candidates. Technical exams are given to chosen applicants for the purpose to evaluate their abilities. After passing, applicants proceed to one or more stages of interviewing that cover supervisory, HR, and technical issues. After obtaining invitation letters, final applicants proceed ahead with the onboarding procedure. The hiring process ensures that individuals with talent, skill, and collaboration are recruited.

2.7 Technology and Framework

The following software, tools, and frameworks are used in the development process of Shohoz Limited, restructured for an ASP.NET Core-based technology stack:

1. **System Requirements (Minimum for Development)**
   - .NET SDK: .NET 6 or .NET 7+ (LTS version preferred)
   - Microsoft SQL Server (or MongoDB v5+ if NoSQL is required)
   - Visual Studio 2022+ or Visual Studio Code with C# extensions
   - Node.js: v16+ (only if using frontend libraries like React.js or for tooling)
2. **Software Requirements**

   Frontend (Blazor / Razor Pages / React.js + ASP.NET Core API)
   - Blazor Server / Blazor WebAssembly – Frontend framework (SPA or SSR using C#)
   - Razor Pages – Server-rendered HTML templates using .cshtml
   - React.js (optional) – When using a separate SPA frontend
   - Axios / Fetch API – For HTTP requests from frontend to backend APIs
   - Tailwind CSS / Bootstrap / MudBlazor / Radzen – UI styling frameworks and components

   Backend (ASP.NET Core + EF Core)
   - ASP.NET Core – Web API or MVC backend framework
   - Entity Framework Core – ORM for SQL Server or PostgreSQL
   - ASP.NET Identity / JWT Bearer Authentication – For secure user authentication
   - IConfiguration (appsettings.json / environment variables) – For environment configuration
   - Serilog / NLog – Logging and debugging utilities
   - Dependency Injection (Built-in) – For service abstraction and testability
   - Bcrypt.Net-Next – For password hashing
   - CORS Middleware – For managing cross-origin requests

   Database
   - Microsoft SQL Server (local or cloud-hosted on Azure)
   - SQL Server Management Studio / MongoDB Compass – GUI clients for managing databases
3. **Development Tools**
   - Visual Studio 2022 / Visual Studio Code – IDE with extensions like C#, Razor, GitLens
   - Postman / REST Client – For API testing
   - Git with GitHub / GitLab / Bitbucket – Version control and collaboration
   - Docker (Optional) – For containerization and deployment environments
   - CI/CD Tools (Optional) – GitHub Actions, Azure DevOps, Jenkins for automated testing and deployment

2.7.1 Projects

Currently, AuraCore Tech takes part in several initiatives. Most of these are confidential and will not be made available until subsequently. However, the public is given access to some of the above project specifics.

Web Development

As a client-based firm, AuraCore Tech often receives proposals for web development projects which are designed to satisfy various corporate requirements. Customers often come to them for their development of scalable e-commerce platforms with secure payment interfaces and flexible, trendy websites. In addition, they are tasked with creating distinctive web applications such as customer portals, booking systems, and dashboards. To ensure long-term reliability, they deliver continuous website maintenance, performance enhancement, and technical help. Also, customers desire companies to employ SEO guidelines in order to improve web visibility and customer satisfaction, and they want CMS connectivity for simple content management.

App Development

In general, they involve developing customized iOS and Android mobile applications that correspond to specific company requirements. Clients are looking for multi-platform and native products and services including education, healthcare, logistics, and e-commerce. Furthermore, it's possible to incorporate features like GPS tracking, payment gateways, secure identification of users, and notifications in real time. For this reason, to guarantee smooth operation and user enjoyment, clients also depend on us for UI/UX design, performance optimization, and post-deployment assistance.

AI/ML

Predictive analytics, recommendation systems, sentiment analysis, and consumer behavior modeling represent some of the solutions consumers seek. They use unstructured as well as structured information to develop and apply machine learning models which are appropriate for particular company cases. They additionally take care of issues like preparing data, training models, evaluating them, and incorporating them into present systems. To guarantee enhanced intelligence and data-based decisions, clients rely on them for machine learning approaches, chatbot development, and NLP (natural language processing) applications.

UX/UI Design

They receive orders from clients to develop user-friendly, simple interfaces for internet applications, mobile apps, and websites. To ensure that the designs meet client requirements and company goals, we carry out wireframing, user testing, and user research. For the purpose of enhancing the overall user experience, our group operates on developing visually appealing, responsive, and easily accessible interfaces. Using interactive and modern design techniques, customers also look for help in updating existing designs, ensuring consistency, and improving user engagement.

Chapter 3

Internship Objectives

Being a web developer intern at Aura-Core Tech, I can anticipate a complete and challenging training program which will allow me to improve my skills and gain practical knowledge. Through the supervision of skilled instructors and leading experts in the profession, the training program seeks to give participants real-world experience and knowledge. My goals have been exceeded by the entire experience.

3.1 Technical Training and Hands-on Experience

The aim of the internship is to provide learners hands-on expertise in industrial software development. Interns will have conversations with experts in the industry, participate in real-world tasks, and acquire valuable lessons. This goal ensures that interns can apply the conceptual understanding gained academically to real-world scenarios. Additionally, they will develop a solid foundation in web development.

3.2 Skill Development

The primary objective of the training program is to enhance web development technical abilities. Interns will learn and practice databases, frameworks, programming languages, and other fundamental technologies. Aura-Core Tech aims to teach its interns how to develop robust and accessible web applications.

3.3 Collaboration and Team Work

The significance of teamwork in the workplace is highlighted throughout the internship at Aura-Core Tech. Interns will work actively with the frontend development team and interact with other departments and senior developers. The goal is to improve teamwork, effective communication, and the ability to perform well in a collaborative environment.

3.4 Mentorship and Guidance

Aura-Core Tech is dedicated to guiding and mentoring interns throughout the internship. Each intern will be paired with an instructor who will provide guidance,

advice, and evaluation. It is the supervisor's responsibility to help interns develop learning objectives, achieve success, and gain industry expertise. This ensures that interns receive individual attention and benefit from the professional knowledge and experience of their mentors.

3.5 Professional Networking

Aura-Core Tech also emphasizes assisting and mentoring interns to build professional networks. A leader will be assigned to each intern to provide ongoing direction, support, and feedback. This goal guarantees that interns receive individual attention and gain valuable professional connections that may benefit their future careers.

3.6 Exposure to Industry Practices

Interns will acquire knowledge about software development principles and business standards through the internship program. They will gain exposure to version control systems like Git/GitLab, code reviews, software development lifecycles (SDLC), and other industry standards. This ensures that interns are well-prepared for potential roles as competent developers.

3.7 Personal and Professional Growth

The objective of Aura-Core Tech's internship is to promote personal and professional growth. Interns will graduate from the program with a broad range of abilities and a sense of professional and personal achievement. In addition to technical skills, interns will develop soft skills such as communication, problem-solving, and time management.

Chapter 4

Training Experience

4.1 Overview

A significant turning point in a student's academic and professional growth is securing an internship in an established company. Real employment has grown in significance in today's highly competitive job market because it helps students transfer more easily from their studies to the workplace and improves their potential for employment. Students are able to utilize what they have learned in practical scenarios via jobs, which serves as an essential component in understanding the constantly changing nature of industrial operations.

For the purpose of fulfilling my degree requirements at BRAC University, I joined as an intern at the Web Development department at AuraCore Tech Limited. This internship provided me with an opportunity to obtain hands-on knowledge in an office setting and improve my skills with regard to quality evaluation, issue tracking, and software testing.

AuraCore Tech offers a hands-on, dynamic method when conducting internships. Rather than giving an independent training stage, the company integrates interns into ongoing tasks. Given the absence of a scheduled instruction time frame, my assigned supervisor provided me with constant assistance and guidance, helping me throughout each phase of the process while making sure I could adapt to the tasks and responsibilities.

I had the privilege of being able to take part in actual tasks throughout my internship, observe web development best practices, and gain understanding of several kinds of industry-relevant technologies and frameworks. I've included a couple of screenshots in this report to show my efforts and contributions. However, some sensitive material has been restricted or removed due to confidentiality regulations.

4.2 Training on Git and Version Control

As part of my internship at Aura-Core Tech, I underwent extensive training in Git, one of the most widely used distributed version control systems. This training was designed to help me understand how to manage code efficiently, collaborate effectively with a development team, and maintain proper version history throughout the software development lifecycle.

The training was integrated with daily project activities, ensuring that I gained practical, real-world experience in applying version control concepts. Working within a

GitLab-based environment, I learned to manage repositories, track changes, handle branching strategies, and collaborate on shared codebases following industry best practices.

4.2.1 Key Concepts and Operations Learned

- *Repository Management*: Creating new repositories and cloning existing ones to local environments. Initializing Git in local projects using `git init` and connecting them to remote repositories. Organizing project directories and ensuring consistent file structure across environments.

- *Commit and Change Tracking*: Staging changes with `git add` and committing them with meaningful messages using `git commit`. Writing descriptive commit messages to document the purpose of each change. Reviewing commit history using `git log` and understanding version tracking.

- *Branching and Workflow Management*: Creating and managing branches using `git branch` and `git checkout`. Implementing feature branching to isolate new development work from the main codebase. Merging completed features into the main branch using `git merge` and resolving conflicts. Learning branch naming conventions to maintain project organization.

- *Remote Collaboration (Push and Pull Operations)*: Connecting local repositories with remote repositories hosted on GitLab. Uploading local changes using `git push` and synchronizing team updates using `git pull`. Understanding the difference between local and remote branches and keeping codebases up to date and conflict-free through frequent pulls.

- *Forking and Merge Requests*: Forking existing repositories for experimentation or contributing to shared projects. Creating Merge Requests (MRs) in GitLab to propose changes and request peer reviews. Reviewing code from team members, discussing improvements, and maintaining code quality standards.

- *Conflict Resolution*: Identifying and resolving merge conflicts when multiple contributors modify the same files. Using GitLab's interface and local merge tools to handle conflicts systematically. Reinforcing teamwork and communication to prevent frequent conflicts.

- *Version Control Best Practices*: Following consistent branching models like Git Flow for structured development. Performing regular commits to ensure incremental progress tracking. Keeping the main branch stable and deployable at all times. Collaborating through code reviews and continuous integration workflows.

4.2.2 Tools and Platforms Used

- Git (Command Line Interface) – for version control operations

- Visual Studio Code Git Integration – to visualize branches, commits, and diffs within the IDE

4.3 Getting Familiar with Web Development Scenario

Before I began work, I was asked to study a few important terms and methods. I had to download the Visual Studio software for Windows. I followed the tips and tricks provided by my supervisor and also enrolled in a course on ASP.NET to become familiar with the software.
I was also recommended a documentation resource called *ASP.NET Core* by Microsoft. Initially, I had to gain an in-depth understanding of C#, object-oriented programming (OOP), and code reusability. When applied, these concepts proved to be straightforward and practical.

4.3.1 Getting to Know Visual Studio

On my first day of the internship, my supervisor asked me about my ideas on Object-Oriented Programming (OOP) and code reusability. As I had completed the OOP and Software Engineering course at BRAC University, it was easier for me to grasp the foundation of C#. It took about a week to get comfortable with the language, during which I practiced code on Visual Studio and gradually became familiar with ASP.NET Core.

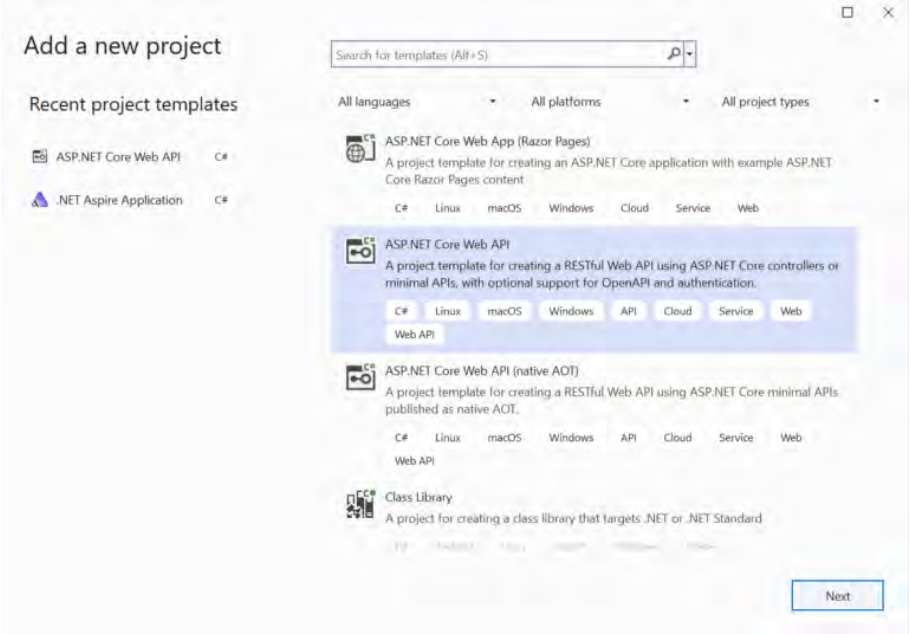

Figure 4.1: Working on an assigned web module during internship

Once ASP.NET is installed, all the necessary packages required for development are automatically included. The environment supports the full MVC (Model-View-Controller) pattern, enabling professional web application development.

4.3.2 Understanding ASP.NET Core Architecture

The ASP.NET Core programming foundation forms the primary objective of the next phase. Using `appsettings.json`, interns explore routing, dependency injection, middleware, request-response pipelines, and customization. To comprehend

the separation of concerns, we used Razor Pages or ASP.NET Core MVC to develop fundamental web applications while engaging with Views, Models, and Controllers. Furthermore, form management, data annotation validation, and basic authentication methods are introduced in this stage.

4.3.3 Frontend Integration with Blazor or Razor Pages

After gaining a thorough understanding of backend architecture, we explored Blazor Server or Blazor WebAssembly to develop dynamic frontends. At this stage, I learned fundamental lifecycle methods, components, routing, and two-way data binding in C#. For design and UI enhancement, we used frameworks such as Tailwind CSS, Bootstrap, and MudBlazor. This approach accelerated the development process by allowing full-stack app development entirely in C#, without relying on JavaScript frameworks.

Figure 4.2: Working on an assigned web module during internship

4.3.4 Data Access with Microsoft SQL Server and REST APIs

At this phase, developers focus on building robust data-driven applications by integrating Microsoft SQL Server with ASP.NET Core using Entity Framework Core (EF Core) — a modern ORM from Microsoft. To define database schema, constraints, and relationships (one-to-many or many-to-many), trainees modeled databases as C# classes using Data Annotations or Fluent API.
We set up a `DbContext`, configured the connection string in `appsettings.json`, and used Code First Migrations to generate the database structure. CRUD operations were implemented using LINQ, while EF Core managed state tracking and data persistence.
Alongside this, we learned how to develop RESTful APIs using ASP.NET Core Web API. This involved creating controller classes that expose endpoints corresponding to HTTP verbs: GET (retrieve), POST (create), PUT/PATCH (update), and DELETE (remove). DTOs (Data Transfer Objects) were used to separate domain models from client-facing data.
Security mechanisms were also introduced — using cookies or JWT for authentication, data protection, and role-based authorization. Dependency Injection (DI) was used for injecting services and repositories to promote modularity and testing. By

the end of this phase, we could build clean, secure, and scalable backends integrated with SQL Server and exposed through REST APIs.

4.3.5 Understanding the MVC Pattern in ASP.NET Core

An essential concept in ASP.NET Core is the Model–View–Controller (MVC) architectural pattern, used to build scalable, maintainable, and well-structured web applications. It divides the application into three components:

- **Model**: Represents application data and logic. It consists of C# classes that map to database tables via EF Core. These models also incorporate validation rules using Data Annotations such as `[Required]` or `[StringLength]`.

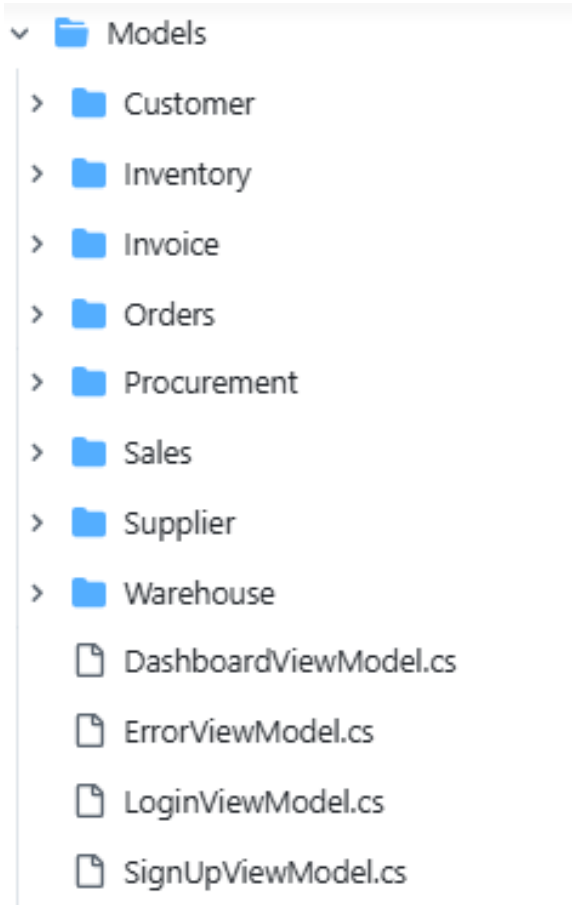

Figure 4.3: Working on an assigned web module during internship

- **View**: Represents the user interface and consists of Razor (.cshtml) pages. Razor syntax allows C# code to be embedded within HTML for dynamic content rendering. Views are lightweight and avoid business logic.

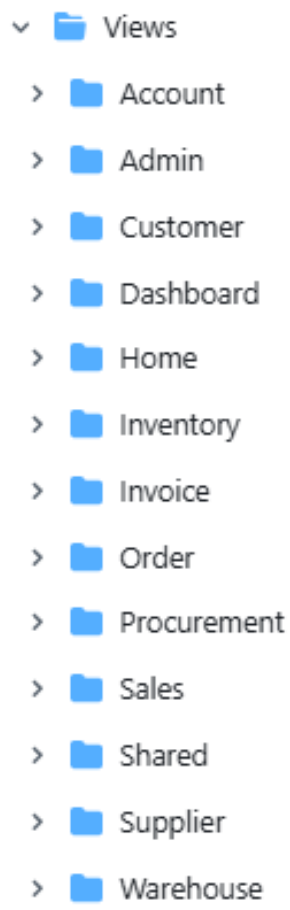

Figure 4.4: Working on an assigned web module during internship

- **Controller**: Acts as the link between Model and View. It processes incoming HTTP requests, interacts with the Model to retrieve or modify data, and returns a View with the appropriate data. Each controller class contains multiple Action Methods that handle specific routes and operations.

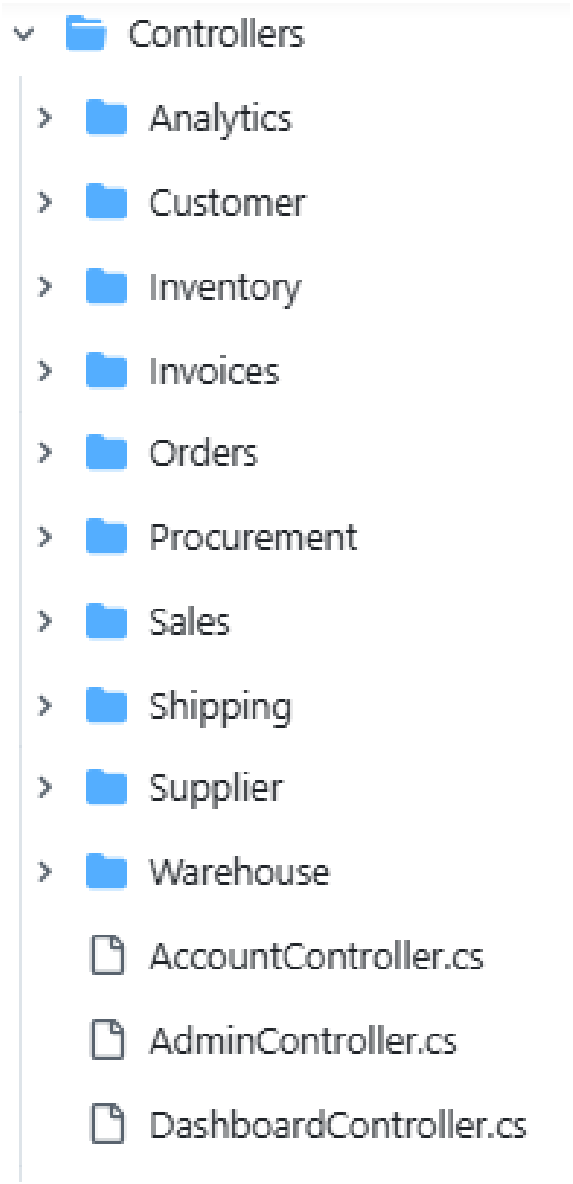

Figure 4.5: Working on an assigned web module during internship

The MVC pattern encourages separation of concerns, improves unit testing and debugging, and enhances maintainability. ASP.NET Core natively supports middleware, dependency injection, and RESTful routing, making it a powerful framework for building enterprise-grade web applications.

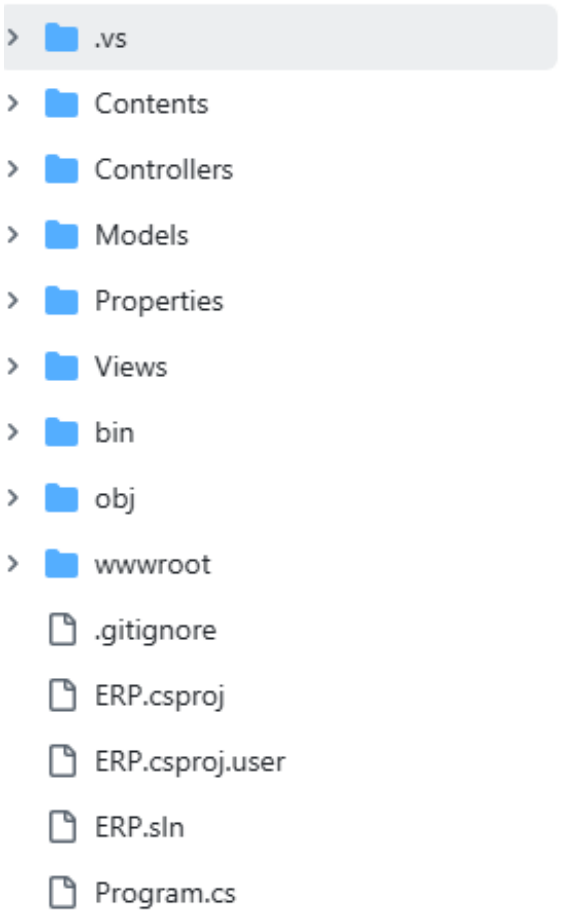

Figure 4.6: Working on an assigned web module during internship

4.3.6 Admin Panel – ERP Software (ASP.NET Web App)

The Admin Panel in this ERP software serves as the central interface for managing configuration, security, and system monitoring across the entire platform. It grants administrative users elevated access to supervise and manage all operational modules through an intuitive user interface.

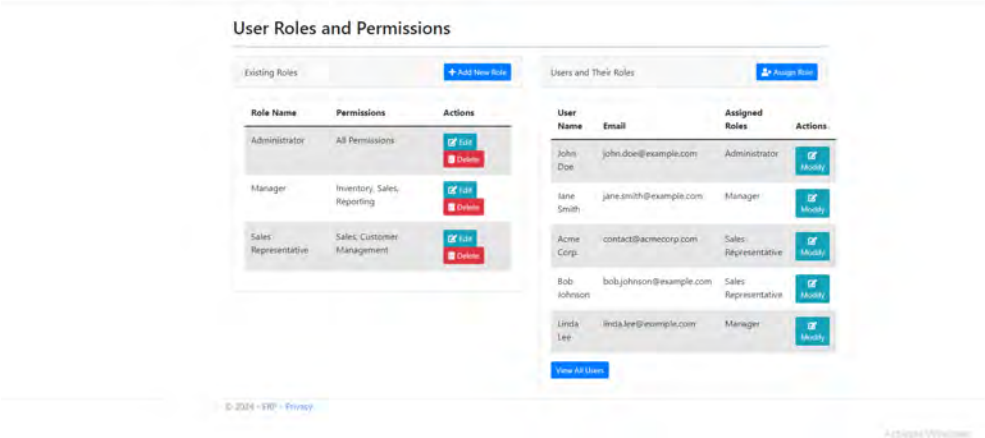

Figure 4.7: Working on an assigned web module during internship

Key Features of the Admin Panel:

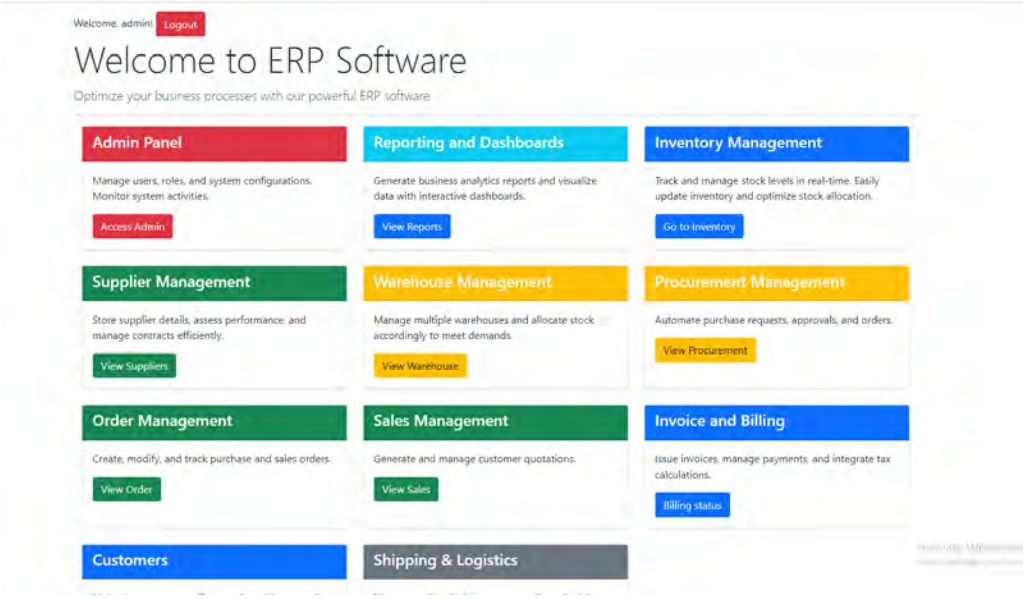

Figure 4.8: Working on an assigned web module during internship

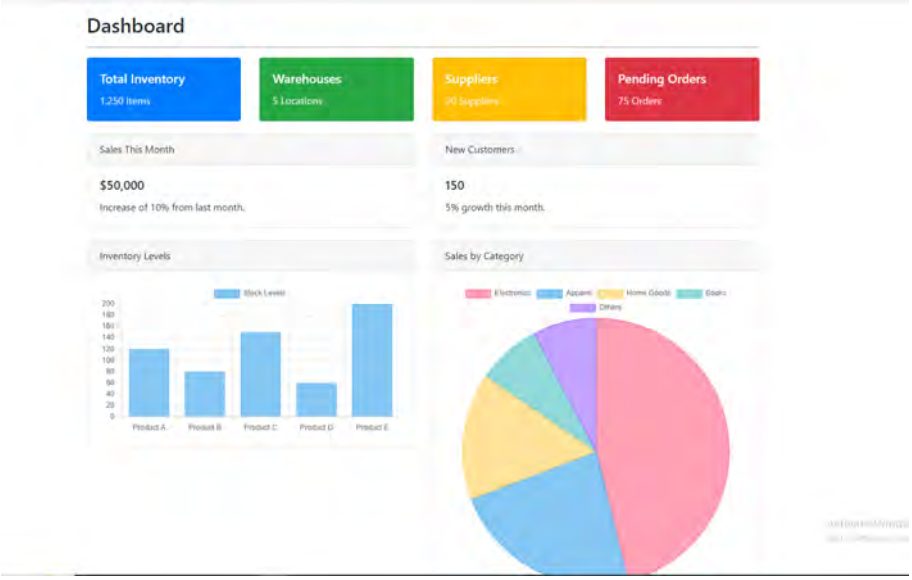

Figure 4.9: Working on an assigned web module during internship

- **User Management:** Administrators can create, modify, and delete user accounts, assign roles such as *Admin*, *Manager*, *Viewer*, and enforce access restrictions based on responsibilities. This role-based behavior ensures structured and secure system access.
- **Role & Permission Configuration:** The panel supports role-based authorization by allowing the definition and restriction of capabilities for various user roles. This includes controlling access to critical modules like Inventory, Orders, Procurement, and Reports through configurable credentials.
- **System Configuration:** Administrators can manage application-wide settings such as design preferences, notification rules, currency/localization settings, API configurations, tax rules, and more.

- **Activity Monitoring:** The system logs and monitors events such as login attempts, data updates, and system errors. Middleware tools like `Serilog` or `NLog` are integrated to assist in capturing and analyzing such activities.
- **Notification & Alerts Setup:** System-wide notifications can be configured via dashboard indicators or email alerts. Examples include low stock warnings, pending order limits, and authorization requirement notifications.
- **Backup & Maintenance Controls:** The Admin Panel allows manual or scheduled data backups, patch or update deployment monitoring, and toggling between maintenance and active states.
- **MVC/Blazor Implementation:**
  - **Model:** Entity Framework (EF) Core with SQL Server is used to represent entities such as `User`, `Role`, `AuditLog`, and `SystemSettings`.
  - **View (UI):** Razor or Blazor components are employed to display dashboard statistics, customizable cards, and buttons similar to Access Admin views.
  - **Controller/Logic:** Responsible for handling authorization checks, role assignments, user CRUD operations, and system data retrieval.

4.3.7 Tutorials and Assistance

During my internship at AuraCore Tech, I was provided with valuable tutorials and learning materials, particularly focused on `ASP.NET`, which helped me establish a solid foundation in web application development. My supervisor was consistently encouraging and responsive whenever I encountered challenges or had questions. Their support made it significantly easier to grasp complex concepts and apply them effectively in real-world scenarios.
In addition to hands-on training, I had access to best practices, well-structured documentation, and organized educational resources. These materials played a crucial role in enhancing my skills and contributed greatly to my learning throughout the internship period.

4.3.8 Communication and Meetings

Following AuraCore Tech's standard workweek, my authorized working hours as an intern were from 9:00 AM to 6:00 PM, Sunday through Thursday. Effective communication proved essential to my learning and professional development during this time.
My supervisor held weekly meetings to review ongoing tasks, assess progress, and address any issues that appeared. These sessions provided opportunities for open dialogue, constructive feedback, and guidance on improving my performance.
In addition to formal meetings, I had the freedom to reach out to my supervisor whenever I needed clarification or encountered obstacles. This ongoing interaction fosterred a collaborative environment that deepened my understanding of the project requirements and the strategic goals of the company.
In general, structured communication and regular meetings were vital in ensuring an organized workflow and enriching my internship experience.

4.3.9 Work Environment

Interning at AuraCore Tech was both informative and rewarding. From the beginning, I felt like a valued member of the team rather than just an intern. This welcoming atmosphere allowed me to quickly adapt to the company's work culture. The team members were approachable and always willing to share knowledge or provide assistance when needed. Even when working remotely, the work environment remained motivating and collaborative.
Daily morning meetings helped synchronize the team with tasks and priorities. Though optional, mid-day conference calls allowed team members to stay connected, maintain presence, and seek real-time support if needed. This method significantly supported unity and productivity among the remote workforce.
Such practices contributed to maintaining a high level of efficiency and team cohesion, making the work experience highly engaging and productive.

4.4 Getting Familiar With FastAPI

4.4.1 Getting To Know VS Code

During the initial phase of my internship, I became familiar with Visual Studio Code (VS Code) a modern, lightweight, and extensible code editor widely used for software development. It provided a powerful environment for building and testing my FastAPI-based AI backend project, offering features like integrated debugging, terminal access, version control, and extension support.

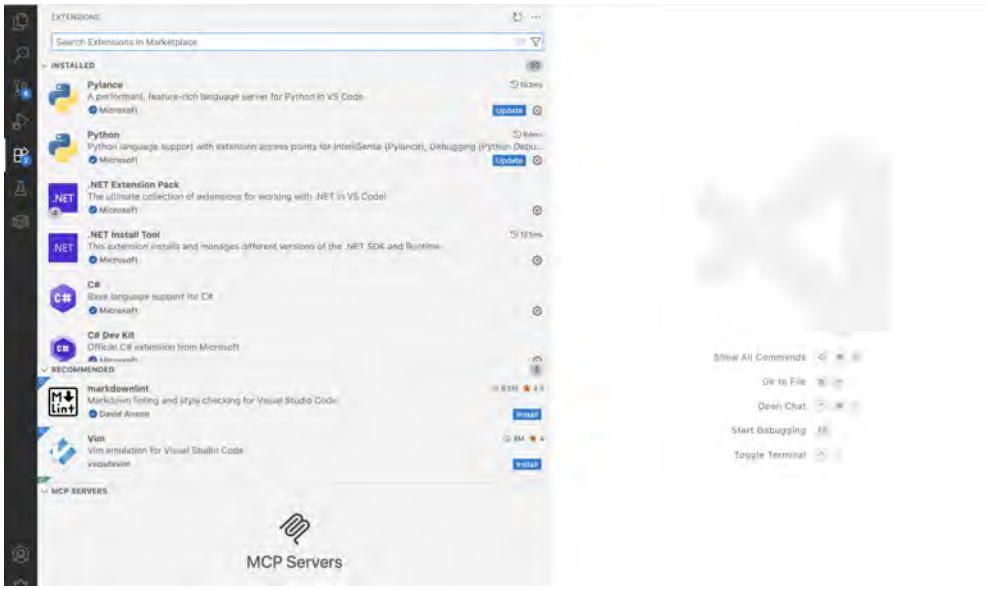

Figure 4.10: Visual Studio Code

Environment Setup

To prepare the development environment for the project, the following steps were undertaken:

1. Installed the Python extension in VS Code and configured the interpreter.

2. Created a virtual environment (`venv`) and activated it in the integrated terminal.

3. Installed the required Python libraries using:

```
pip install fastapi uvicorn openai
```

4. Set up a launch configuration in `launch.json` to run and debug the FastAPI server directly from VS Code.

5. Installed the SQLite Viewer extension to view and query local database files.

6. Added the REST Client extension to send API requests (e.g., `/ai-task` and `/login`) directly within the editor.

This setup ensured a seamless and efficient workflow for developing, testing, and debugging the backend system.

Advantages

The use of VS Code provided several significant advantages during development:

- **Simplified Project Navigation:** The sidebar explorer made it easy to browse and organize project files.
- **Integrated Terminal:** Allowed running servers, installing packages, and managing the environment without switching tools.
- **Smart IntelliSense:** Offered real-time code suggestions, autocompletions, and documentation for Python and FastAPI.
- **Built-in Debugger:** Simplified debugging of routes and functions, reducing runtime errors.
- **Version Control Integration:** Seamless Git and GitHub integration supported commits, pushes, and version tracking.
- **Extension Ecosystem:** Enhanced productivity through useful plugins for AI, API testing, and data visualization.

Outcomes

By becoming proficient with VS Code, the following outcomes were achieved:

- Efficiently developed and debugged the FastAPI application within a single environment.
- Improved productivity through integrated tools and smart features.
- Maintained better project structure and version control using built-in Git integration.

- Reduced context switching by handling coding, testing, and debugging in one platform.
- Strengthened confidence in professional development workflows using modern IDEs.

Overall, mastering VS Code played a key role in enhancing development efficiency and ensuring a smooth backend implementation process.

4.4.2 Understanding FastAPI Core Architecture

To effectively develop and extend the project, it was essential to gain a deep understanding of the FastAPI core architecture. FastAPI is a modern, high-performance web framework built on top of Starlette for web handling and Pydantic for data validation. It leverages Python's asynchronous capabilities and type hints to provide a fast, efficient, and developer-friendly environment for building APIs.

Core Components of FastAPI

The FastAPI architecture is composed of several core layers and components that work together to process client requests and generate responses:

1. **ASGI (Asynchronous Server Gateway Interface):** FastAPI is based on the ASGI specification, which allows asynchronous communication between the server and the application. It enables concurrency and high throughput, making FastAPI suitable for real-time applications.
2. **Starlette Framework:** FastAPI uses Starlette as its underlying web toolkit. Starlette handles HTTP requests, routing, middleware, background tasks, and WebSocket support.
3. **Pydantic Models:** FastAPI relies on Pydantic for data validation and serialization. It uses Python type hints to automatically validate request bodies, query parameters, and response models.
4. **Dependency Injection System:** FastAPI includes a built-in dependency injection mechanism that allows developers to define reusable components such as authentication handlers, database sessions, and configuration utilities.
5. **Routing Layer:** Routes define the available endpoints (e.g., `/ai-task`) and the associated request methods (`GET`, `POST`, etc.). FastAPI automatically maps each route to a corresponding Python function.
6. **Request and Response Cycle:** Requests are received by the ASGI server (e.g., Uvicorn), passed to FastAPI, validated through Pydantic models, processed by route handlers, and finally returned as structured JSON responses.
7. **Automatic Documentation:** FastAPI generates interactive API documentation using Swagger UI and ReDoc, enabling easy testing and exploration of endpoints.

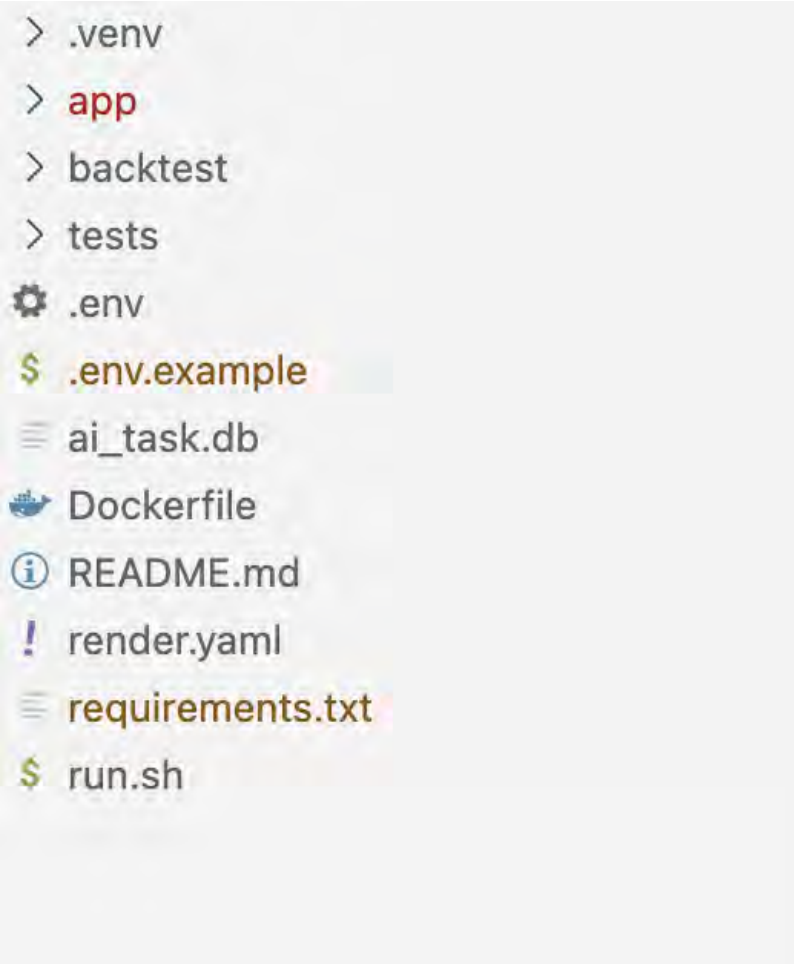

Figure 4.11: Components

Request Handling Flow

The following steps outline the internal flow when a client sends a request to a FastAPI application:

1. The client sends an HTTP request to a defined endpoint (e.g., `POST /ai-task`).
2. The ASGI server (Uvicorn) receives the request and forwards it to the FastAPI application.
3. The routing system identifies the corresponding function based on the endpoint and HTTP method.
4. Input data (query, body, path parameters) is parsed and validated using Pydantic models.
5. The business logic is executed inside the route function, potentially involving database operations, AI model calls, or tool invocations.
6. The response is serialized into JSON format and returned with appropriate HTTP status codes.

Asynchronous Architecture

FastAPI is fully compatible with Python's `async` and `await` syntax. This enables concurrent handling of multiple requests, reducing blocking operations and improving performance. Asynchronous support is particularly beneficial for I/O-bound tasks such as:

- Database queries
- API calls to external AI services
- File I/O or network communication

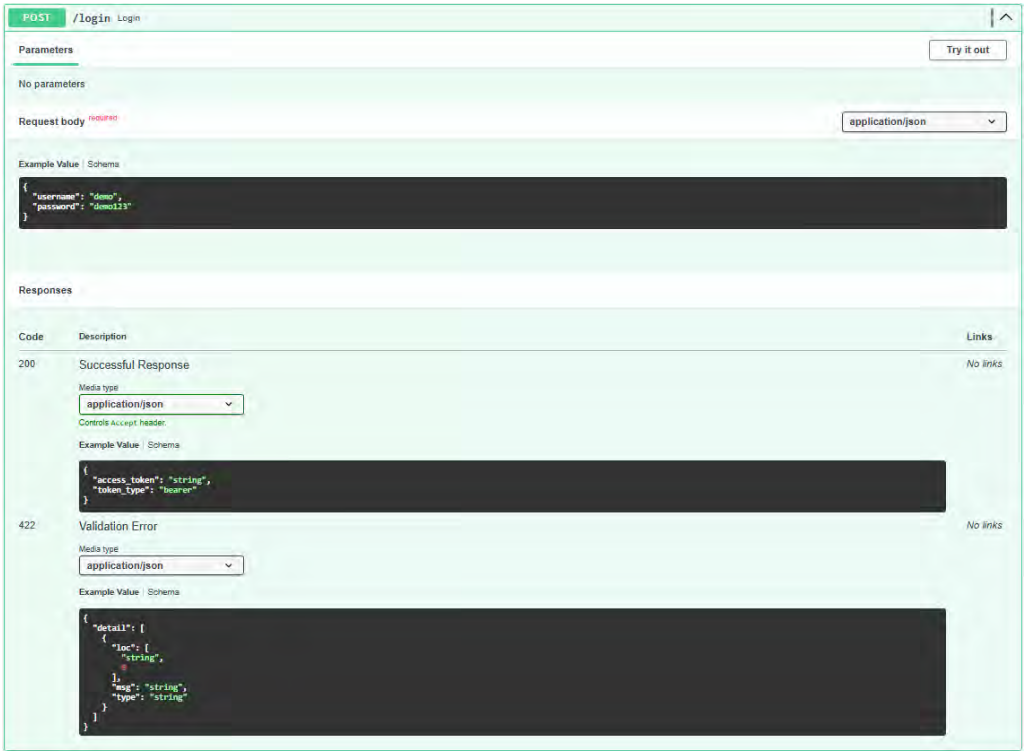

Figure 4.12: Login Endpoint

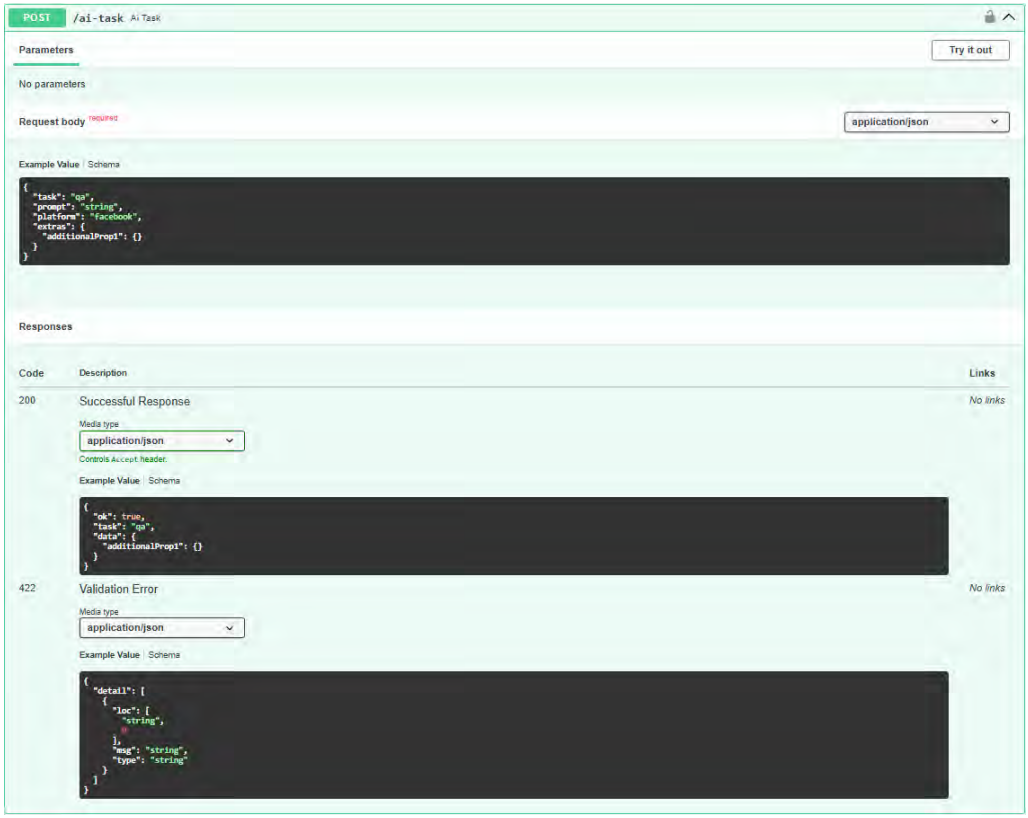

Figure 4.13: AI Task Endpoint

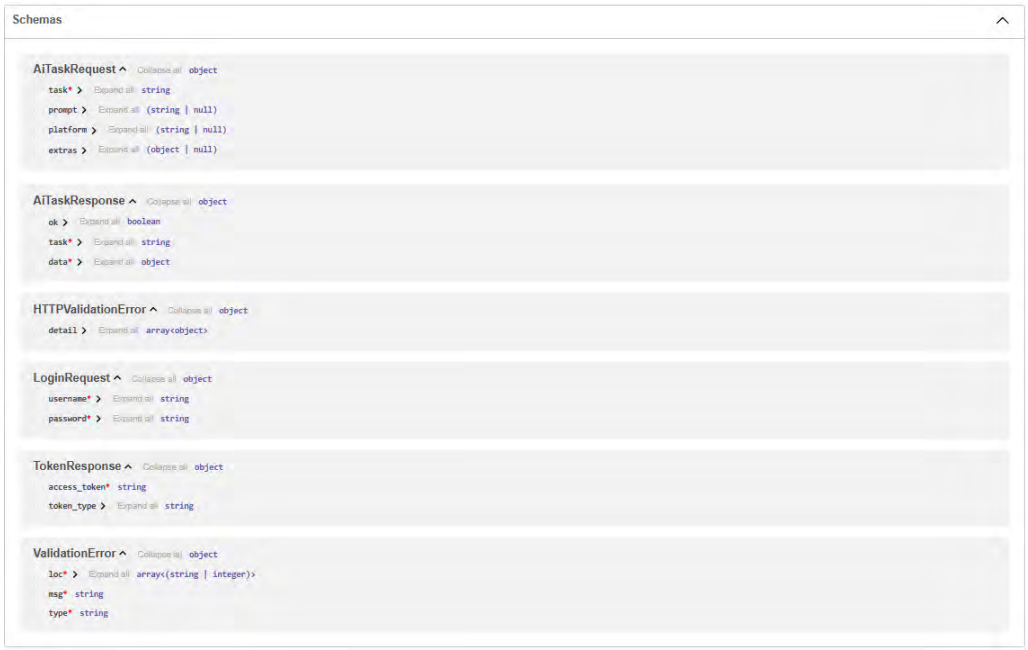

Figure 4.14: Schema

Benefits of FastAPI Architecture

- **High Performance:** Based on ASGI and async programming, it achieves performance comparable to Node.js and Go.

- **Automatic Validation:** Type hints and Pydantic ensure data integrity with minimal boilerplate.
- **Extensibility:** Dependency injection allows modular and scalable design.
- **Interactive Documentation:** Swagger UI and ReDoc are auto-generated from the code.
- **Developer Productivity:** Minimal code duplication and strong typing improve maintainability.

Application in the Project

In this project, the FastAPI architecture was leveraged to design a **single-route system** (`/ai-task`) that dynamically handled multiple AI-related tasks such as:

- Question-answering through OpenAI chat models
- Image generation and base64 encoding
- Platform-specific content creation
- Retrieval of the latest stored outputs from SQLite

The clear separation between request handling, business logic, and response serialization made it easier to extend functionality, add authentication using JWT, and integrate MCP tools and backtesting utilities.

4.4.3 Project Overview

During the internship, a robust and modular FastAPI application was developed to handle multiple AI-driven tasks through a unified interface. The project focused on designing a single-route API architecture to streamline user interactions, integrate modern AI services (LLMs, image generation), and demonstrate tool orchestration using the Model Context Protocol (MCP). The system also included an optional JWT-based authentication mechanism, and a lightweight trading backtest utility to showcase analytical capabilities.

4.4.4 Key Features

- **Single Endpoint:** All functionalities are accessed via `POST /ai-task` with a task identifier (`qa`, `latest`, `image`, or `platform_content`).
- **AI Q&A:** Integrates OpenAI chat models to answer user queries and store results in a local SQLite database.
- **Latest Query Retrieval:** Provides quick access to the most recent AI-generated response from the database.
- **Image Generation:** Utilizes OpenAI's image model to produce base64-encoded images based on text prompts.

- **Platform-Specific Content:** Generates marketing or announcement content tailored to platforms like LinkedIn, Facebook, Twitter/X, Instagram, TikTok, Reddit, and Medium.
- **JWT Authentication:** Implements secure token-based access control via `/login` endpoint.
- **MCP Integration:** Demonstrates extensibility with a local MCP server exposing a demo tool (`upper(text)`) and a client for tool calling.
- **Trading Backtest Utility:** Includes a simple SMA crossover strategy module to evaluate stock trading performance, producing CSV outputs, metrics, and charts.

```
fastapi==0.111.0
uvicorn[standard]==0.30.0
pydantic==2.8.2
python-multipart==0.0.9
python-dotenv==1.0.1
sqlalchemy==2.0.32
aiosqlite==0.20.0
passlib[bcrypt]==1.7.4
pyjwt==2.9.0
openai==1.51.0
pillow==10.4.0
pandas==2.2.2
matplotlib==3.9.0
yfinance==0.2.43
mcp==1.2.0
```

Figure 4.15: Requirements

4.4.5 Technical Stack

- **Framework:** FastAPI (Python)
- **Database:** SQLite with SQLAlchemy ORM
- **Authentication:** JWT (JSON Web Token)
- **Models:** OpenAI Chat Models for Q&A and content generation; OpenAI Image Models for visual generation
- **Tool Protocol:** Model Context Protocol (MCP) for external tool integration
- **Backtest:** Python with `yfinance`, `pandas`, `matplotlib` for financial analytics
- **Containerization:** Docker for reproducible deployment
- **IDE:** Visual Studio Code

4.4.6 System Architecture

The system was designed with separation of concerns and modularity:

- `app/main.py`: FastAPI entry point handling `/ai-task`
- `app/auth.py`: JWT creation and verification utilities
- `app/db.py`: Database initialization and model schema
- `app/mcp_server.py` and `app/mcp_client.py`: Minimal MCP server-client demo
- `backtest/sma_backtest.py`: Financial backtest logic with output metrics

```python
# Simple SMA crossover backtest to produce CSV, chart, and metrics.
import pandas as pd, yfinance as yf, matplotlib.pyplot as plt, sys, pathlib, json
from datetime import datetime, timedelta

out_dir = pathlib.Path(__file__).parent / "results"
out_dir.mkdir(parents=True, exist_ok=True)

ticker = sys.argv[1] if len(sys.argv) > 1 else "AAPL"
start = (datetime.today() - timedelta(days=365*2)).strftime("%Y-%m-%d")
df = yf.download(ticker, start=start, auto_adjust=True)
df = df.rename_axis("date").reset_index()
df["SMA50"] = df["Close"].rolling(50).mean()
df["SMA200"] = df["Close"].rolling(200).mean()
df.dropna(inplace=True)

df["signal"] = 0
df.loc[df["SMA50"] > df["SMA200"], "signal"] = 1
df["position"] = df["signal"].diff().fillna(0)

# Compute daily returns
df["ret"] = df["Close"].pct_change().fillna(0.0)
df["strategy"] = df["ret"] * df["signal"].shift(1).fillna(0.0)
equity = (1 + df["strategy"]).cumprod()

# Save CSV
csv_path = out_dir / f"{ticker}_signals.csv"
df.to_csv(csv_path, index=False)

# Metrics
cagr = equity.iloc[-1] ** (252/len(equity)) - 1
vol = df["strategy"].std() * (252 ** 0.5)
sharpe = (df["strategy"].mean() / df["strategy"].std()) * (252 ** 0.5) if df["strategy"].std() else 0.0
metrics = {"ticker": ticker, "cagr": float(cagr), "vol": float(vol), "sharpe": float(sharpe)}
(out_dir / "metrics.json").write_text(json.dumps(metrics, indent=2))

# Chart
plt.figure()
plt.plot(df["date"], equity, label="Strategy Equity")
plt.title(f"SMA Crossover Strategy – {ticker}")
plt.xlabel("Date")
plt.ylabel("Equity (Start=1.0)")
plt.legend()
png_path = out_dir / f"{ticker}_equity.png"
plt.savefig(png_path, bbox_inches="tight")
```

Figure 4.16: Backtest

```python
import os, time, jwt
from fastapi import Depends, HTTPException
from fastapi.security import HTTPBearer, HTTPAuthorizationCredentials

ALGORITHM = os.getenv("JWT_ALG","HS256")
SECRET = os.getenv("JWT_SECRET","change-me-in-prod")
security = HTTPBearer()

def create_token(sub: str) -> str:
    # 'Indefinite' per spec: no exp claim, but include iat
    payload = {"sub": sub, "iat": int(time.time())}
    return jwt.encode(payload, SECRET, algorithm=ALGORITHM)

def verify_token(creds: HTTPAuthorizationCredentials = Depends(security)) -> str:
    token = creds.credentials
    try:
        payload = jwt.decode(token, SECRET, algorithms=[ALGORITHM], options={"verify_exp": False})
        return payload.get("sub") or "unknown"
    except Exception as e:
        raise HTTPException(status_code=401, detail=f"Invalid token: {e}")
```

Figure 4.17: Auth

```python
from sqlalchemy.ext.asyncio import create_async_engine, async_sessionmaker, AsyncSession
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column
from sqlalchemy import Integer, String, Text, DateTime, func

DATABASE_URL = "sqlite+aiosqlite:///./ai_task.db"

engine = create_async_engine(DATABASE_URL, echo=False, future=True)
SessionLocal = async_sessionmaker(engine, expire_on_commit=False)

class Base(DeclarativeBase):
    pass

class Answer(Base):
    __tablename__ = "answers"
    id: Mapped[int] = mapped_column(Integer, primary_key=True, index=True)
    task: Mapped[str] = mapped_column(String(50))
    content: Mapped[str] = mapped_column(Text)
    created_at: Mapped[str] = mapped_column(DateTime, server_default=func.now())

async def init_db():
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
```

Figure 4.18: Database

4.4.7 MCP Server Integration

Model Context Protocol (MCP) was used to demonstrate AI tool orchestration and external function calls within the project.

- **Server Implementation:**
  - Located in `app/mcp_server.py`.
  - Exposes a single demo tool `upper(text: str)` which converts input text to uppercase.
  - Built using the official MCP Python SDK to handle requests over standard I/O.
- **Client Interaction:**
  - The MCP client (`app/mcp_client.py`) connects to the server and calls the exposed tools.
  - Within the FastAPI `/ai-task` route for Q&A, the server attempts a tool call before generating a final response, demonstrating AI agent augmentation.
- **Purpose and Learnings:**
  - Learned how AI agents can compose multiple tools and **enhance outputs** using auxiliary services.
  - Demonstrated a framework for external tool integration with LLMs.
  - Provided a foundation to extend the MCP server with more complex utilities in the future.

```python
import asyncio
from mcp import StdioServerParameters
from mcp.client.session import ClientSession
from mcp.client.sse import SSEClientTransport  # not used here, but available
from mcp.client.stdio import stdio_client

async def call_upper(text: str) -> str:
    params = StdioServerParameters(command=["python","-m","app.mcp_server"])
    async with stdio_client(params) as (read, write):
        async with ClientSession(read, write) as session:
            tools = await session.list_tools()
            if not any(t.name == "upper" for t in tools.tools):
                raise RuntimeError("upper tool not available")
            result = await session.call_tool("upper", arguments={"text": text})
            # result.content is a list of objects with 'type' and 'text' fields
            chunks = [c.text for c in result.content if getattr(c, "text", None)]
            return "\n".join(chunks) if chunks else ""

if __name__ == "__main__":
    print(asyncio.run(call_upper("hello mcp")))
```

Figure 4.19: MCP Client setup

```python
# Minimal MCP server exposing a simple 'upper' tool.
# You can run it with: python -m app.mcp_server
# Docs: https://github.com/modelcontextprotocol/python-sdk
from mcp.server.fastmcp import MCPServer, tool
import asyncio

server = MCPServer("FastAPI-ai-task-mcp")

@tool()
def upper(text: str) -> str:
    """Return the text in uppercase (demo tool)."""
    return text.upper()

async def main():
    await server.run_stdio()  # stdio transport

if __name__ == "__main__":
    asyncio.run(main())
```

Figure 4.20: MCP Server

4.4.8 Development Environment

- **Operating System:** Windows
- **IDE:** Visual Studio Code
- **Setup:**
  1. Clone repository and create a virtual environment
  2. Install dependencies via `requirements.txt`

3. Configure environment variables in `.env` (e.g., `OPENAI_API_KEY`)
4. Launch API using `uvicorn` (`localhost:8000/docs`)

- **Containerization:** Built Docker images and ran containers for reproducibility and cloud deployment
- **VS Code Benefits:** Integrated terminal, debugging tools, environment management, and extensions for Python, FastAPI, and Docker.

4.4.9 Approach

The development followed a unified endpoint design, allowing flexible addition of future AI tasks without changing the API structure. The project emphasized:

- Clean modular design with Pydantic-based validation
- Persistent storage of all responses for traceability
- Secure access through optional JWT
- Integration of external tools via MCP
- Reproducible financial backtesting with risk control mechanisms

4.4.10 Risk Controls and Assumptions

- Prevented look-ahead bias in trading strategy by shifting signals
- Dropped incomplete data for accurate metrics
- Simplified assumptions: zero transaction cost, full liquidity, and frictionless execution
- Tokens have no expiry for demo purposes; production systems should include expiry and rotation

4.4.11 Outcomes and Learnings

- Gained practical experience in FastAPI for building RESTful APIs and Python web development with modular architecture.
- Learned to integrate OpenAI models for different tasks:
  - **Q&A agent:** Crafting prompts and handling responses from large language models.
  - **Content generation:** Tailoring style and tone for various platforms (LinkedIn, Twitter/X, etc.).
  - **Image generation:** Producing base64 images via text prompts and understanding the image API workflow.
- Explored the concept of AI agents and tool orchestration using MCP:

  - Built a mini agent that calls an external tool (`upper(text)`) before answering.
  - Learned how AI agents can delegate tasks, augment responses, and interact with local tools or APIs.

- Implemented state management by persisting AI outputs in SQLite and retrieving the latest results.
- Developed an understanding of trading strategy analysis and **risk control** in a reproducible pipeline.
- Learned JWT authentication workflow, middleware handling, and secure API design.
- Practiced DevOps basics:
  - Containerization with **Docker**, including writing Dockerfiles and running containers locally.
  - Deployment on cloud platforms using Render.
  - Ensuring environment reproducibility and dependency management through Docker images and virtual environments.
- Strengthened skills in debugging AI outputs and handling edge cases in multimodal AI systems.
- Familiarized with VS Code, leveraging integrated debugging, terminal, extensions, and environment management for efficient development.

Chapter 5

Challenges Faced and How I Overcame Them

During my internship at Aura-Core Tech, I encountered several challenges that tested both my technical expertise and problem-solving skills. Working on projects involving ASP.NET MVC, FastAPI, and AI agent integration required continuous learning, adaptability, and effective collaboration. Overcoming these challenges helped me grow as a developer and gain practical confidence in real-world software engineering.

5.1 Understanding and Implementing New Frameworks

One of the earliest challenges was adapting to new frameworks such as ASP.NET MVC and FastAPI. Each framework had distinct patterns for routing, middleware, and database interaction. Initially, I found it difficult to manage asynchronous operations in FastAPI and to structure controllers and views in ASP.NET MVC. To overcome this, I studied official documentation, followed hands-on tutorials, and explored real project examples. I gradually learned how to design RESTful APIs using FastAPI, implement authentication and data validation with Pydantic, and connect APIs with frontend applications. Regular code reviews from senior developers helped me align with best practices.

5.2 Building and Integrating AI Agents

Implementing AI agents within existing web applications was another significant challenge. Integrating model-based decision-making and connecting it with FastAPI endpoints required both AI understanding and backend design skills. Initially, I struggled with managing API responses, handling asynchronous inference requests, and deploying lightweight models. To address this, I learned to modularize AI components, use background tasks for inference in FastAPI, and test responses using Postman. This experience enhanced my ability to bridge AI logic with production-ready backend systems.

5.3 Version Control and Git Workflow

Working with Git in a collaborative environment introduced challenges related to branching, merging, and resolving conflicts. Mistakes such as pushing code without updates or merging into the wrong branch occasionally occurred. To improve, I adopted a structured Git workflow: creating feature branches, performing frequent pulls, writing meaningful commit messages, and using merge requests through GitLab. I practiced resolving merge conflicts locally and reviewed branching strategies like Git Flow to maintain a clean and stable codebase.

5.4 Debugging and Integration Issues

Integrating backend APIs with frontend components and debugging request-response cycles was often time-consuming. Problems arose in API endpoints, schema mismatches, and incorrect data serialization. I overcame these by using tools like Postman for API testing, FastAPI's built-in interactive documentation (Swagger UI), and detailed logging for error tracking. This process strengthened my understanding of API design and debugging techniques.

5.5 Time Management and Task Prioritization

Managing multiple tasks within short sprints was a constant challenge. Some modules, especially those involving AI logic or backend deployment, required more time than estimated. To address this, I improved my planning by breaking tasks into smaller milestones, maintaining daily progress logs, and using agile tools to track deadlines. This approach enhanced my efficiency and ensured timely completion of deliverables.

5.6 Communication and Team Collaboration

Collaborating with product managers, UI/UX designers, and backend engineers required clear communication. At the beginning, I hesitated to raise issues or share ideas. Over time, I realized the importance of proactive communication. Participating in daily stand-ups, sprint reviews, and team discussions improved my ability to express technical challenges and seek timely support.

5.7 Adapting to Agile Development Practices

Working within an agile framework required flexibility and rapid adaptation. The iterative cycle of development, testing, and feedback was initially unfamiliar. Through consistent participation in sprint meetings and retrospectives, I learned to adjust to evolving requirements and deliver incremental updates effectively.

5.8 Outcome

By successfully addressing these challenges, I gained strong technical proficiency in FastAPI, ASP.NET MVC, and AI agent integration, alongside practical skills in Git, debugging, and agile teamwork. This experience not only enhanced my coding and architectural understanding but also improved my communication, problem-solving, and project management capabilities, preparing me for future professional development roles.

Chapter 6

Conclusion

However,it has only been six months since I joined the company Auracore Tech as a web development intern. The experience of working as a web developer intern for AuraCore Tech has been tremendously helpful and educational.

Focusing on real client based projects provided me with an opportunity to fill the gap between theoretical knowledge and application in real life. I've been able to enhance my web development skills while learning up resources, frameworks, and approaches that are essential to the field. The learning environment was well-organized and positive, with training on topics like ASP.NET and continuing assistance and guidance from my supervisor. Through frequent conferences, team calls, and open routes for interaction, the company kept a professional yet friendly environment .I became enabled to quickly become a part of the team and develop confidence as a novice developer due to the friendly and supportive environment. This internship proved to be an essential turning point in both my educational and professional advancement as it has improved my technical ability, interpersonal abilities, and flexibility in the workplace overall. AuraCore Tech has my deepest appreciation for this amazing experience.

On February 23, 2025, I started my internship at AuraCore Tech. The internship is scheduled to finish on August 23, 2025, as it is a six-month agreement. As completion of my university defence, I submit this report, which highlights my efforts along with my experiences through the first six months of employment as a web development intern at AuraCore Tech.