Internship Report

on

Web Development Internship at Code Jango

Course: SWE420

Submitted by

Mahir Al Shahriar
Registration No: 2019831077
Department of Software Engineering
Institute of Information and Communication Technology (IICT)

Internship Period: July 1, 2024 – March 1, 2025

Date of Submission: May 22, 2025

Institute of Information and Communication Technology
Shahjalal University of Science and Technology

Sylhet, Bangladesh

Letter of Transmittal

May 22, 2025

The Director
Institute of Information and Communication Technology
Shahjalal University of Science and Technology

Subject: Letter of Transmittal

Dear Sir,

With due respect, it is a great honor and pleasure for me to submit my internship report as part of the academic requirement of the internship program. I am sincerely grateful to the Department of Software Engineering and the Institute of Information and Communication Technology (IICT) for providing me with the opportunity to bridge academic knowledge with current software industry practices.

This report is based on my internship experience at Code Jango, a community-based coding academy, carried out from 1 July 2024 to 1 March 2025. During this period, I was actively involved as a Web Developer, contributing to live projects, UI engineering, backend development, and product integration efforts.

The report highlights the technical expertise I developed, outlines my responsibilities, and reflects on my learning outcomes. I hope it accurately represents the knowledge, skills, and growth I achieved throughout my internship period.

I sincerely look forward to your valuable feedback on this report.

Sincerely yours,

Mahir Al Shahriar
Registration No: 2019831077
Department of Software Engineering
Institute of Information and Communication Technology
Shahjalal University of Science and Technology

Letter of Endorsement

Subject: Approval of Internship Report

This letter is to formally certify that the contents of the internship report prepared by Mahir Al Shahriar (Registration No. 2019831077), a student of the Institute of Information and Communication Technology, Shahjalal University of Science and Technology, are accurate and align with the work carried out during his internship at Code Jango.

The projects, tasks, and technical activities documented in the report reflect Mahir's direct involvement in various aspects of web development at Code Jango, including frontend engineering, backend integration, and deployment practices. I have personally reviewed and approved the report prior to its final submission.

All the information provided is truthful, non-confidential, and representative of the actual contributions made during his tenure. Mahir has shown diligence, technical acumen, and a strong willingness to learn—qualities that will serve him well in his future endeavors.

I fully endorse this report and wish him continued success in his academic and professional journey.

Sincerely,

Mirazul Islam
Technical Solutions Lead
Code Jango

Signature:

Acknowledgement

First and foremost, I would like to express my heartfelt gratitude to the Institute of Information and Communication Technology (IICT), Shahjalal University of Science and Technology (SUST) for arranging this internship program, which has been a pivotal opportunity for my professional and academic development.

I am especially thankful to Prof. Mohammad Abdullah Al Mumin, PhD, Director of IICT. I would also like to extend my sincere appreciation to Mahfuzur Rahman Emon, Lecturer at IICT, for coordinating with industry personnel and providing valuable guidance, which helped make this internship possible and meaningful.

I am deeply grateful to Code Jango for recruiting me as a Web Developer intern and giving me the platform to apply and grow my technical skills in a real-world setting. My sincere thanks to the co-founders — Mirazul Islam, Technical Solutions Lead, and Mahmood Monjur, Marketing and Digital Media Lead — for their exceptional mentorship, continuous feedback, and for nurturing an environment that encouraged both personal and professional growth. Their guidance has had a significant impact on my development as an aspiring software engineer.

This internship has been an enriching experience, and I am thankful to everyone who played a role in making it successful and rewarding.

Executive Summary

This document summarizes my experiences and contributions as a Web Developer intern at Code Jango. It outlines the key projects I completed, the technologies I worked with, and the challenges I navigated during the six-month remote internship. The primary objective is to reflect on the practical software engineering skills I developed while contributing to a fast-paced, startup-like environment.

The Department of Software Engineering at SUST offers undergraduate students the opportunity to participate in an industry internship during their 8th semester. The goal of this program is to provide hands-on experience with real-world software development practices, bridging the gap between academic learning and industrial application. As a student of this department, I was selected to join Code Jango—a growing software organization—where I worked directly with the founding team on building and scaling the company's web infrastructure.

At Code Jango, I was the sole web developer, entrusted with full ownership of the frontend system, from implementation to documentation. My role involved close collaboration with the core leadership team through weekly sprint planning meetings, where we set technical goals, created Jira tickets, and aligned development tasks with business priorities. Despite working independently, I operated within a structured agile-inspired framework, reporting progress regularly and maintaining a high standard of development rigor.

One of my first tasks was to develop a responsive landing page based on a Figma design. I initially implemented it using vanilla HTML, CSS, and JavaScript to meet a tight deadline. Later, I was instructed to re-architect the frontend in React and SCSS to improve modularity and scalability. This transition marked a key learning phase, where I applied modern frontend technologies—such as JSX, functional components, React Hooks, and SCSS features like variables and mixins—to create a maintainable, component-based system.

As the product vision evolved, I played a pivotal role in developing the authentication

system for a custom Learning Management System (LMS). This included implementing secure user registration, JWT-based login, email verification workflows, and role-based access control using Node.js, Express.js, and MySQL. I also created UML diagrams and technical documentation to facilitate future onboarding and codebase understanding.

Midway through the internship, shifting priorities required me to pivot quickly. In response to a surge in student enrollment, I proposed and deployed a temporary solution using Google Forms and Google Pay for fast course registration. Once the rush subsided, I replaced this with a fully custom, multi-step form system built in React and integrated with the Stripe payment gateway—ensuring design consistency, improved validation, and backend data persistence.

This internship allowed me to apply software engineering principles in a real-world setting, strengthen my full-stack development skills, and adapt to the demands of a dynamic, product-driven environment. Working independently while remaining aligned with team goals taught me the importance of both technical initiative and collaborative planning. Ultimately, this experience has equipped me with the confidence and competence to take on professional software development roles in the future.

Contents

1 Introduction **9**

1.1 Preface 9

1.2 Objectives 10

1.3 Scope 10

1.4 Sources of Information 11

2 Company Profile **12**

2.1 Company Overview 12

2.2 Vision & Mission 12

2.3 Location & Operations 13

2.4 Key Personnel 13

2.5 Key Partners 14

3 My Working Conditions and Contract **15**

4 My Work at CodeJango as a Web Developer **17**

4.1 Initial Implementation and Responsive Design Foundation 17

4.2 Re-Architecting the Frontend in React and SCSS 18

4.3 Technical SEO and Performance Optimization: Migrating from CRA to Next.js 19

4.4 Foundational Forms For Learning Management System (LMS) 22

4.5 React Hook Form Course Registration System and Payment Integration . 25

4.5.1 Stripe Payment Integration 30

4.5.2 Addressing Security Concerns 32

4.6 Building the Foundational Learning Management System (LMS) at CodeJango 32

4.6.1 Foundational Platform Shift: Requirement Engineering and System Modeling 33

4.6.2 Software Architecture and Model-Driven Implementation 34

4.6.3 Establishing the Secure, Componentized API Core 36

5 Conclusion – Personal and Professional Growth **39**

5.1 Time Management and Self-Discipline 39

5.2 Working Under Pressure and Managing Uncertainty 40

5.3 Communication Skills 40

5.4 Interpersonal and Team Collaboration 40

5.5 Leadership and Ownership 41

5.6 Final Reflection 41

Chapter 1

Introduction

1.1 Preface

An internship is a vital bridge between academic learning and professional experience. It allows students to explore their career interests, apply theoretical knowledge in practical settings, and gain valuable skills that are essential for personal and professional growth. As part of the undergraduate program, internships help students develop a deeper understanding of industry practices and prepare them for future challenges in the workplace.

The Department of Software Engineering at the Institute of Information and Communication Technology (IICT), SUST, places strong emphasis on integrating academic study with industry exposure. Through initiatives like guest lectures from industry experts and project-based learning, the department actively encourages students to engage with real-world technology trends. Among these efforts, the six-month internship program is particularly significant in shaping students into industry-ready professionals.

I was fortunate to complete my internship at Code Jango, a reputed software company known for its innovative solutions and commitment to quality. During this period, I worked on real projects, collaborated with professionals, and gained hands-on experience with modern development tools and methodologies. This internship has not only enhanced my technical capabilities but also strengthened my communication, problem-solving, and teamwork skills.

This report outlines the work I undertook, the experiences I gathered, and the skills I developed during my internship at Code Jango.

1.2 Objectives

The main objective of my six-month internship as a Web Developer at Code Jango was to bridge the gap between academic learning and professional software engineering practice by gaining hands-on experience in real-world projects. This internship aimed to develop my technical skills, enhance my understanding of industry workflows, and prepare me for future career challenges. Specifically, the objectives of the internship included:

1. To gain practical experience in the full lifecycle of web development, from design implementation to maintaining scalable and modular frontend architectures.
2. To develop proficiency in modern frontend technologies such as React and SCSS, advancing beyond basic web design skills.
3. To deepen knowledge of backend development by implementing secure authentication, RESTful APIs, and integrating third-party services like Stripe.
4. To build skills in agile methodologies by actively participating in sprint planning, task prioritization, and technical decision-making using tools like Jira.
5. To enhance my ability to work effectively in a remote team environment, fostering responsibility, communication, and collaboration.
6. To improve adaptability to changing product requirements while producing maintainable, well-documented code and software solutions.
7. To strengthen confidence and professional readiness for future roles in software engineering and development.

1.3 Scope

This report is based on my individual observations and experiences gained during my six-month internship as a Web Developer at Code Jango. The scope of this report includes the following key areas:

1. **Company Profile and Mission**
   An overview of Code Jango, its mission, core values, and its role in the software development industry.
2. **My Work Experience as a Web Developer**
   A detailed account of my responsibilities and daily activities as the sole web de-

veloper, including remote collaboration, task management, and involvement in the full web development lifecycle.

3. **My Technical Growth and Project Involvement**
   An exploration of the technical skills I developed throughout the internship, including frontend and backend technologies, as well as the specific projects I contributed to and the technologies applied.

4. **My Professional Growth**
   Reflection on the soft skills and professional attributes I enhanced, such as communication, adaptability, teamwork in a remote environment, and confidence building through real-world challenges.

1.4 Sources of Information

To prepare this report, I have gathered information from various sources including my personal observations during the internship and official resources provided by Code Jango. Both primary and secondary data were utilized to ensure an accurate and comprehensive overview. The sources are outlined below:

Primary Data:

- Direct hands-on experience and observations throughout the internship period.
- Active participation in daily remote work activities and sprint planning sessions.
- Collaboration and communication with the co-founders and other team members.

Secondary Data:

- Official website and documentation of Code Jango.
- Internal project management tools such as Jira and communication platforms used during the internship.
- Relevant articles, tutorials, and technical resources referenced to complete assigned tasks and projects.

Chapter 2

Company Profile

2.1 Company Overview

Code Jango is a community-driven coding academy based in Halifax, Nova Scotia, dedicated to empowering youth through technology education. We offer age-appropriate, project-based courses in Python, web development (HTML, CSS, JavaScript, React), and introductory cybersecurity for students aged 8 to 18. By combining real-world problem-solving with interactive challenges, Code Jango fosters technical confidence, creativity, and critical-thinking skills in a supportive and inclusive environment.

2.2 Vision & Mission

Vision

To become Atlantic Canada's leading technology academy, where every student—regardless of background—discovers their potential through code and becomes a confident digital creator.

Mission

- **Accessible Excellence:** Deliver engaging, hands-on coding experiences that meet high educational and technical standards.
- **Community Impact:** Collaborate with schools, community centres, and local organizations to expand access to STEM education.

- **Mentorship:** Provide one-on-one support and positive role models to nurture both technical and personal growth.
- **Continuous Improvement:** Continuously evolve our curriculum and expand our instructor network to support diverse learning styles and schedules.

2.3 Location & Operations

Headquarters

Halifax, Nova Scotia

Class Delivery

- **In-Person:** Weekend and evening classes at partner sites in Halifax and Dartmouth (e.g., Ummah Masjid, Sabeel).
- **Online:** Live, interactive virtual classrooms supported by recordings and self-paced modules via our Learning Management System (LMS).

Core Operations

- **Curriculum Development:** Agile sprint cycles used to create engaging, theme-based, project-oriented learning modules.
- **Teaching:** Dynamic delivery of lessons through real-time instruction, hands-on exercises, and real-world coding challenges.
- **Instructor Training & Quality Assurance:** Weekly workshops, peer feedback sessions, and curriculum alignment meetings to ensure consistent instructional quality and student engagement.

2.4 Key Personnel

Role	**Name**
Co-Founder, Marketing & Digital Media	Mahmood Monjur
Co-Founder, Technical Solutions Lead	Mirazul Islam

2.5 Key Partners

- **Ummah Masjid** – Provides dedicated in-person teaching space.
- **Sabeel** – Supports in-person learning initiatives through venue and community engagement.

Chapter 3

My Working Conditions and Contract

I was contracted as a **Web Developer** at Code Jango for a period of six months, beginning in August 2024. The position was fully **remote**, offering flexibility in work hours while ensuring continuous collaboration with the core leadership team.

As the sole web developer, I held end-to-end responsibility for the development and maintenance of the organization's **web infrastructure**. Although the team was lean, we followed a structured, **agile**-inspired **workflow** to keep development aligned with evolving business goals.

Weekly online **sprint planning** meetings were held with the co-founders—Mirazul Islam (Technical Solutions Lead) and Mahmood Monjur (Marketing and Digital Media Manager). These sessions served as the primary venue for:

- Discussing and clarifying technical and business **requirements**.
- Reviewing progress on current tasks.
- Prioritizing upcoming features.
- Allocating and scheduling work using a **Jira** board.

Jira **tickets** were created collaboratively during these meetings, with specific **milestones**, **deadlines**, and **priority levels** assigned. I was responsible for independently completing these tasks and keeping the board updated.

In addition to implementation, I was also expected to:

- Report technical progress and blockers to Mr. Mirazul Islam on a regular basis.

- Maintain comprehensive **documentation** of the **codebase**, including **architectural** notes, development workflows, and **module specifications**.
- Produce **UML artifacts** such as **use case diagrams**, **class diagrams**, and **sequence diagrams** to model the **system architecture** and explain technical decisions—both for team alignment and for future developer **onboarding**.

Although I operated independently in terms of coding, the regular collaboration with the co-founders ensured that my work remained strategically aligned with the organization's broader objectives. Despite the absence of a large engineering team, the project maintained a strong standard of **software development rigor**, with attention paid to **scalability**, **clarity**, and **knowledge transfer**.

Chapter 4

My Work at CodeJango as a Web Developer

4.1 Initial Implementation and Responsive Design Foundation

At the outset of my internship at CodeJango, my first major task was to implement the landing page for Code Machines, based on a **Figma** design provided by the design team. This was classified as a high-priority deliverable, with a tight deadline that demanded immediate turnaround. To meet this urgency, I prioritized delivery over architectural rigor and initially built the landing page using vanilla **HTML**, **CSS**, and **JavaScript**.

Despite the lack of a full **frontend architecture** at this stage, the landing page was expected to be fully **responsive** and visually polished across a wide range of screen sizes—from large desktop monitors to mobile phones. This required me to deepen my understanding of **responsive web design** principles, especially:

- **CSS Flexbox:** I used it extensively for dynamic layouts, allowing containers to adjust their child elements based on screen dimensions and available space. Flexbox's ability to manage alignment, justification, and direction enabled fluid, stackable layouts that adapted to any device.
- **The CSS Box Model:** I studied the box model in detail, understanding how margins, borders, padding, and content interact. This was essential for precise layout control and avoiding overflow or misalignment issues, especially when dealing with nested containers and margin collapsing behavior.

- **Media Queries:** I implemented media queries to define **breakpoints** for mobile, tablet, and desktop views, enabling selective styling based on screen width.

To enhance the user experience, I also incorporated scroll-based **animations** using JavaScript and CSS **transitions**, which added subtle visual cues and movement as users navigated down the page. This helped bring the static design to life, contributing to a more engaging first impression.

4.2 Re-Architecting the Frontend in React and SCSS

Once the initial rush subsided, my Technical Solutions Lead instructed me to refactor the landing page into a **React**-based architecture to prepare for future **scalability** and **maintainability**. This marked a significant learning milestone, as I had not previously worked with React in a production setting.

I began by familiarizing myself with React fundamentals, including:

- **JSX** syntax, which offered a more expressive way to write component markup.
- **Functional components** and **React Hooks** like `useState` and `useEffect` to manage local **state** and **lifecycle** behaviors.
- **Componentization**, where each UI block (e.g., Hero section, course cards, testimonials) was modularized into its own reusable React component.

In parallel, I sought to improve the maintainability of our styling by replacing raw CSS with **SCSS (Sass)**—a **CSS preprocessor** that supports modular and scalable stylesheet architecture. Key SCSS features I adopted included:

- **Variables** for consistent color palettes, font sizes, and spacing values across the UI.
- **Nesting** for scoped styles within components.
- **Mixins** and **Functions** to abstract common patterns like responsive breakpoints or reusable layout snippets.

Together, React and SCSS allowed me to convert what was initially a tightly coupled HTML/CSS project into a modular and maintainable frontend system—laying the groundwork for future features like **user authentication**, course offerings, and form-based **interactivity**.

4.3 Technical SEO and Performance Optimization: Migrating from CRA to Next.js

As the application matured, a critical challenge emerged that shifted my focus from feature development to foundational **performance** and **visibility**. Our initial build, using **Create React App (CRA)**, relied entirely on **Client-Side Rendering (CSR)**. In this model, the browser receives a nearly empty **HTML** shell and a large **bundle** of **JavaScript**. The JavaScript must then execute to build the page's content and structure in the browser—a process that creates significant bottlenecks:

Slow Initial Load: Users faced a noticeable delay, seeing blank pages or loaders while the scripts downloaded, parsed, and executed.

Poor Search Engine Indexing: Web **crawlers** from search engines like Google, while increasingly sophisticated, still prioritize speed and complete content. A CSR site often appears "empty" to a crawler on its first pass, risking incomplete **indexing** and lower rankings.

Inefficient Caching: Caching the empty HTML page provides little benefit, as the core content is dynamically generated on the client.

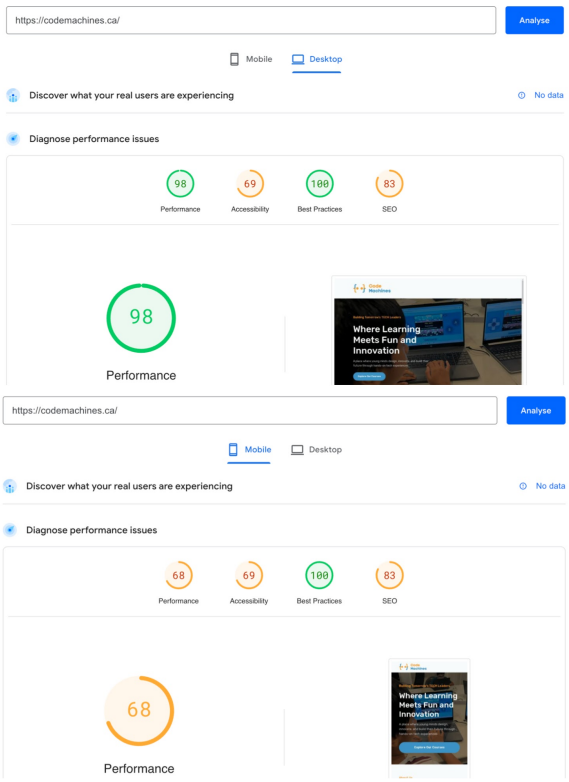

Figure 4.1: Web Vitals Before Optimization

To solve this, I needed to deeply understand the **rendering spectrum**. Through research, I evaluated two primary solutions: **Server-Side Rendering (SSR)**, where HTML is generated on-demand per request, and **Static Site Generation (SSG)**, where HTML is pre-built at **deployment** time. Both deliver fully-formed HTML to browsers and crawlers, solving our core **SEO** and performance problems.

After analyzing our content patterns—static course descriptions, stable pricing, and infrequently updated marketing pages—I determined that **SSG** was the optimal **architecture**. Unlike SSR, which requires server compute for each visit, SSG would deliver instantaneous page loads from **CDN** edge locations while maintaining perfect SEO readiness and eliminating server costs.

My learning journey involved dissecting **Next.js**'s hybrid rendering capabilities to implement the most efficient solution. I chose Next.js for its first-class support for both SSG and SSR, understanding that I could implement a fully static architecture while preserving the option for dynamic rendering if future requirements changed.

The migration was a profound exercise in modern React architecture. Here's how my architectural decision translated into implementation:

- **Implementing Static Site Generation with Next.js Export:** I configured `next.config.mjs` with `output: 'export'`, which instructs Next.js to **pre-render** all pages to static HTML/JS during **build time**. This eliminated any server-side rendering runtime, ensuring all pages could be served instantly from any CDN. The build process generated complete HTML files for every route—homepage, course pages, checkout flows—with all content embedded directly in the source code.

- **Leveraging Next.js's App Router for Build-Time Rendering:** Using the **App Router**'s default **Server Components**, I ensured that during the build process, all critical course descriptions, headings, and metadata were rendered directly into the static HTML. Unlike traditional SSR that happens per request, this build-time rendering meant our content was "baked in" and required zero server processing for visitors.

- **Strategic Use of Client-Side Hydration ("use client"):** Understanding the **hydration** model was key. For interactive components—like our animated testimonials carousel, enrollment forms with validation, and PayPal integration—I strategically marked specific files with the `"use client"` **directive**. These components rendered as static placeholders during build time, then seamlessly hydrated with interactivity on the client after the initial HTML loaded. This maintained the

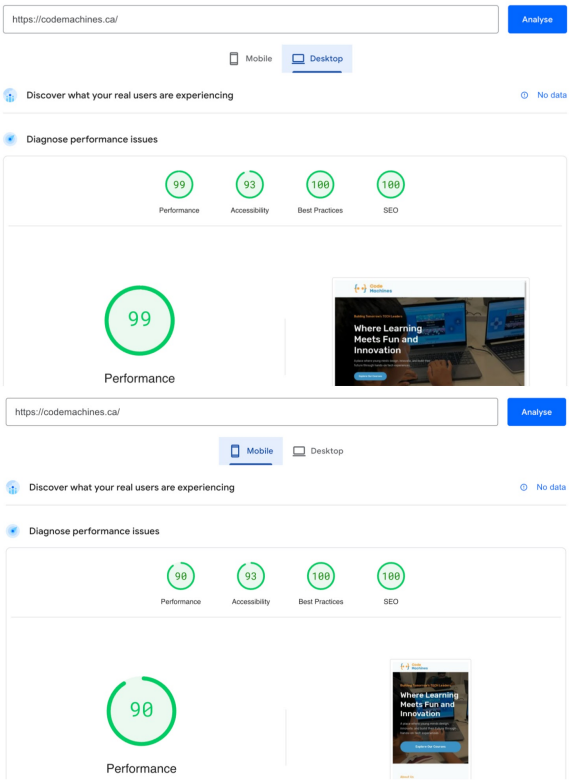

Figure 4.2: Web Vitals After Optimization

static nature of our pages while preserving all dynamic user experiences.

- **Automated Image Optimization:** I integrated Next.js's `<Image />` component, which during the build process automatically handles image **resizing**, modern **format conversion** (to **WebP**), and generates **responsive image sets**. This replaced static `<img>` tags and resulted in dramatically faster image delivery with zero **layout shift**, directly boosting **Core Web Vitals** scores.

- **Metadata API for SEO:** I implemented Next.js's **Metadata API** to generate proper `<title>`, `<meta description>`. Since all pages were statically generated, these SEO elements were permanently embedded in each page's HTML, giving us precise, crawlable control over search appearance.

The outcome was transformative. The codemachines website began appearing in search results within days of deployment, with crawlers now seeing complete, instantly loadable content. Performance metrics showed dramatic improvements: near-instantaneous **First Contentful Paint** and perfect **Lighthouse** SEO scores. Beyond technical metrics, our hosting costs reduced by eliminating server runtime entirely, while gaining infinite scalability through CDN distribution.

4.4 Foundational Forms For Learning Management System (LMS)

One of the most significant challenges during my internship at Code Jango was navigating the evolving **product vision**—particularly in the absence of a fixed **roadmap**. As the sole web developer on the team, I had to work in a lean, fast, and highly adaptable manner. Under the guidance of Mirazul Islam, the Technical Solutions Lead, I was frequently tasked with building features even when requirements were still fluid or undefined.

During one of our early sprint planning meetings, we discussed the development of a **Learning Management System (LMS)**, which was to be built incrementally. The initial scope of the LMS was limited in functionality, focusing primarily on core **user account** features. The key requirements at this phase included:

- Allowing students to **register** and **log in** to the system.
- Enabling students to **enroll** in courses.
- Displaying a list of registered courses specific to each student.

Although further discussions were scheduled to refine the complete LMS specifications, I was immediately assigned to build the foundational **authentication system**. At this point, many UI components were still under design, so I proceeded by implementing **backend** functionalities and exposing the **endpoints** with minimal front-end integration. Temporary output and test feedback were displayed via console logs to validate core flows.

On the front end, I opted to build simple yet functional forms using **controlled components** in React, managing state manually using `useState` for form inputs. Instead of leveraging external form libraries like React Hook Form at this stage, I chose to maintain full control over the input behavior to allow for easy experimentation.

For the backend, I developed a lightweight **RESTful API** using **Node.js** and **Express.js**, interfaced with a **MySQL** database to handle user data **persistence**. The backend supported:

- Secure registration and login using **JSON Web Tokens (JWT)** for **authentication** and **session management**.
- **Role-based access control (RBAC)** to distinguish between students and potential administrative users.

- An **email verification** workflow using **Nodemailer**, where newly registered users would receive a verification email.

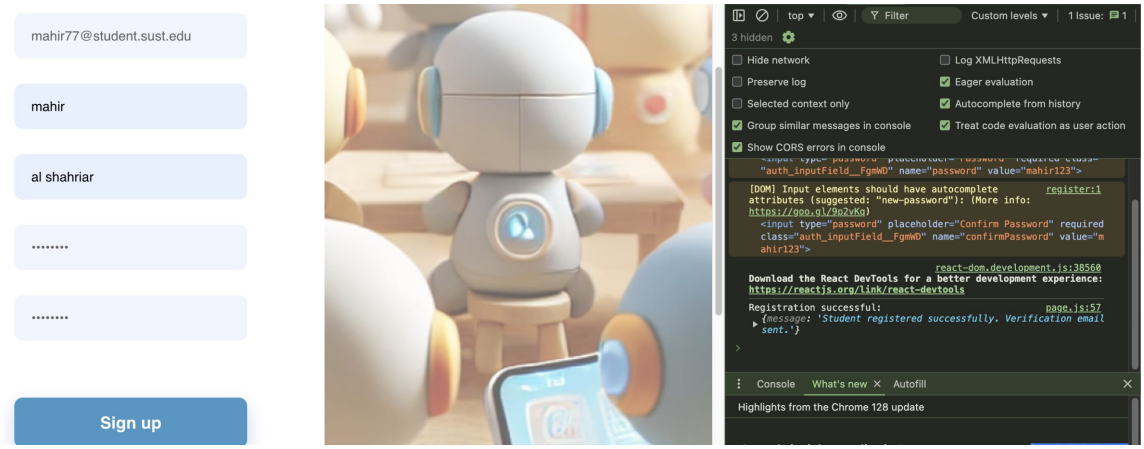

Figure 4.3: User Registration UI

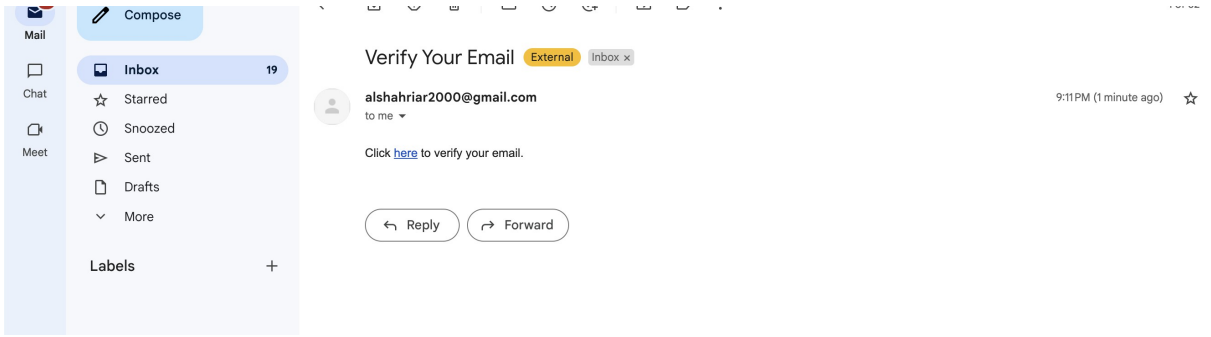

Figure 4.4: Verification Email for Account Verification

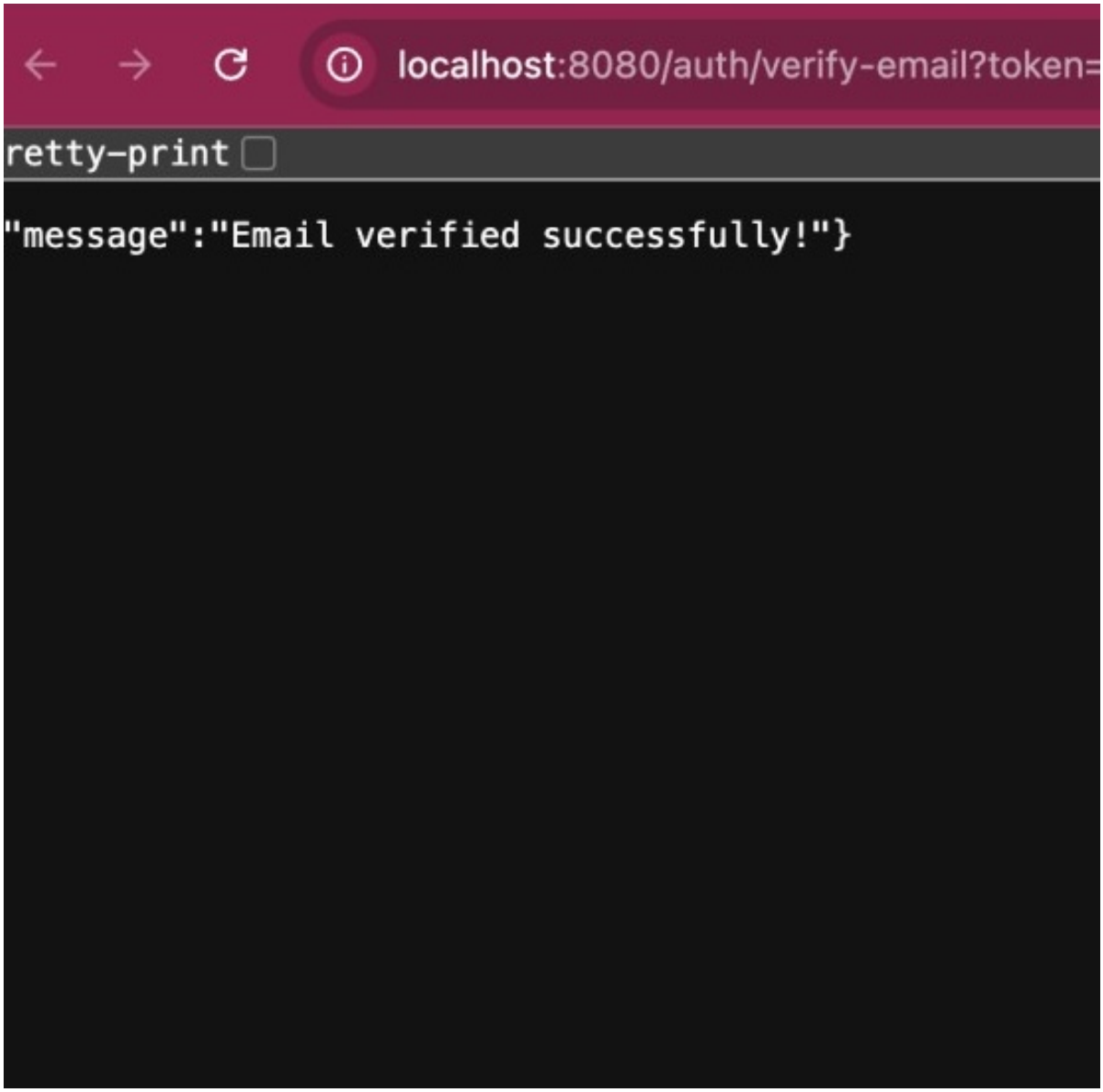

Figure 4.5: Email Verified Message Displayed

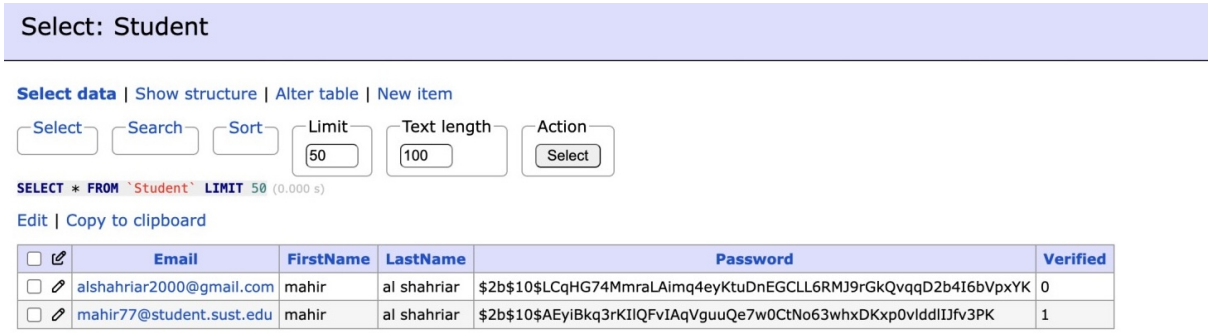

Figure 4.6: Database Table Entry Showing Verified And Unverified Account

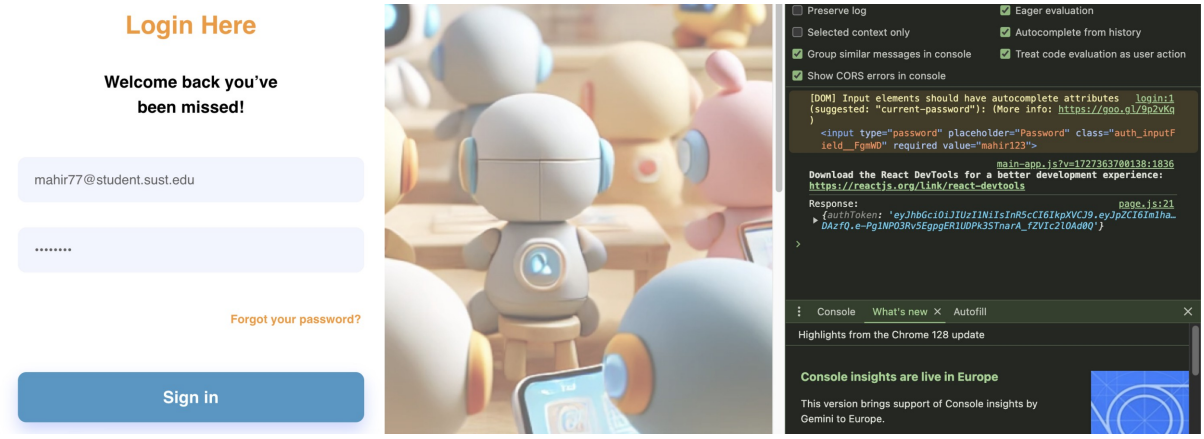

Figure 4.7: Login UI

To implement the email verification mechanism, I generated a JWT that encoded the user's ID in its **payload** and embedded it as a **query parameter** in the verification link. When a user clicked the link, the backend would **decode** the token, **validate** the user, and mark the account as verified in the database. For demonstration purposes, I used one of my personal email addresses as the sender to ensure that the system functioned end-to-end during testing. This phase laid the technical foundation for secure user management in the LMS and allowed me to iterate further on higher-level LMS functionalities with a working user authentication system already in place.

4.5 React Hook Form Course Registration System and Payment Integration

As product priorities shifted mid-development, the initial login and registration system was temporarily left unattended. A pressing need arose to focus on the course landing pages and implement a fast and scalable course registration mechanism. This urgency was triggered by a high-volume intake of students, requiring immediate solutions for course enrollment.

Given the tight timeline, I proposed using **Google Forms** as an interim solution. While this did not align with our established **design system** and **UI consistency**, it proved to be an effective and efficient stopgap. Google Forms enabled rapid **deployment**, integrated seamlessly with **Google Sheets** for real-time **data collection**, and supported **Google Pay** for processing payments—helping us meet our operational deadlines during peak registration demand.

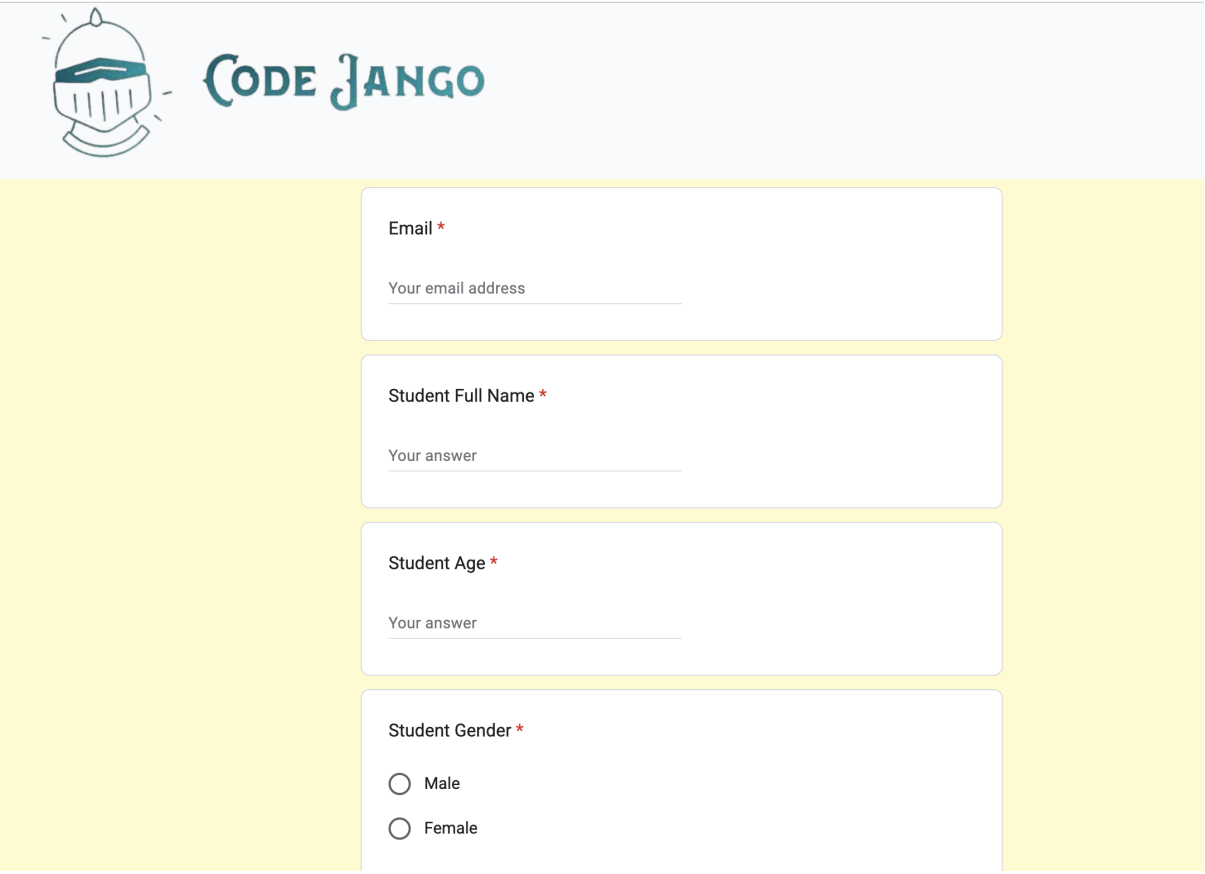

Figure 4.8: Integrated Google Form

Once the initial phase stabilized, I was tasked with replacing the Google Forms setup with a fully customized, design-consistent solution. The objectives included:

- Designing custom, visually consistent **multi-step forms** for course registration.
- **Persisting** form data in our own backend database to allow future extensibility.
- Moving away from Google Pay to integrate a more flexible and revenue-efficient **payment gateway** using the **Stripe SDK**.

To accommodate the complexity and volume of input fields, I broke the registration form into a **wizard**-style layout, where each page or "step" focused on a specific group of inputs (e.g., personal details, course preferences, payment). I created separate React components for each step to maintain **separation of concerns** and improve code readability.

A top-level `FormContainer` component orchestrated the form's flow and maintained global form state using React's `useState` hook. This allowed me to:

- Retain input values as users navigated between steps.

- Dynamically show/hide components based on the current step.
- Perform **validation** at each step before allowing progression.

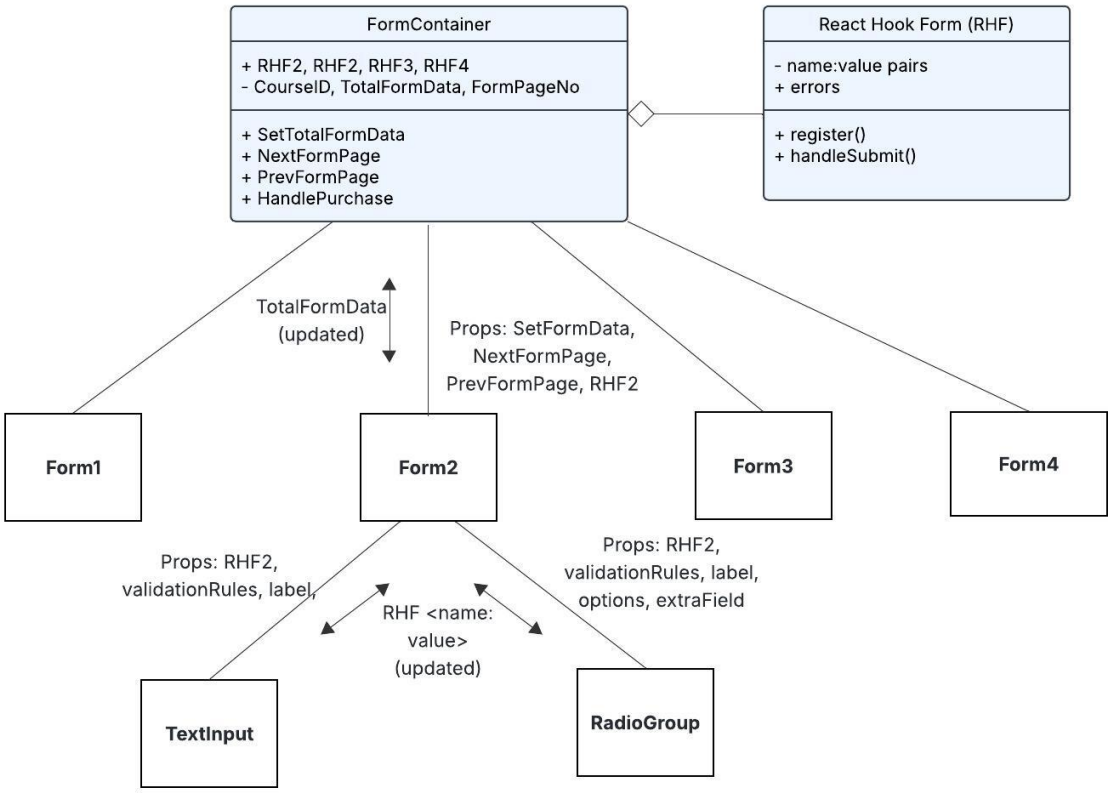

Figure 4.9: Class And Structure Diagram For Course Registration Form

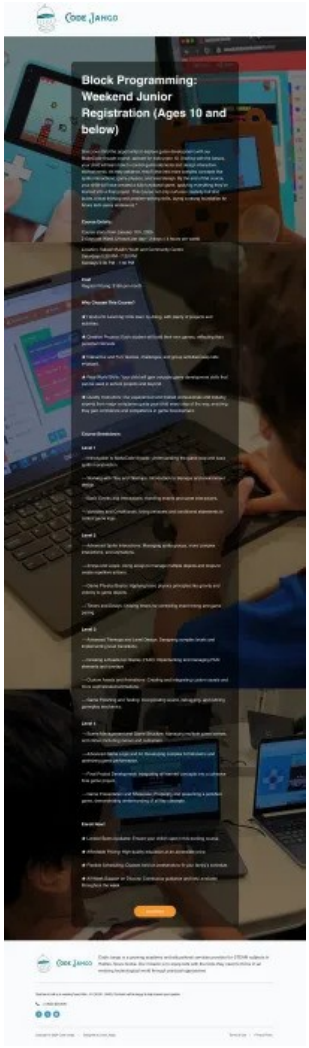

Figure 4.10: Updated Course Page

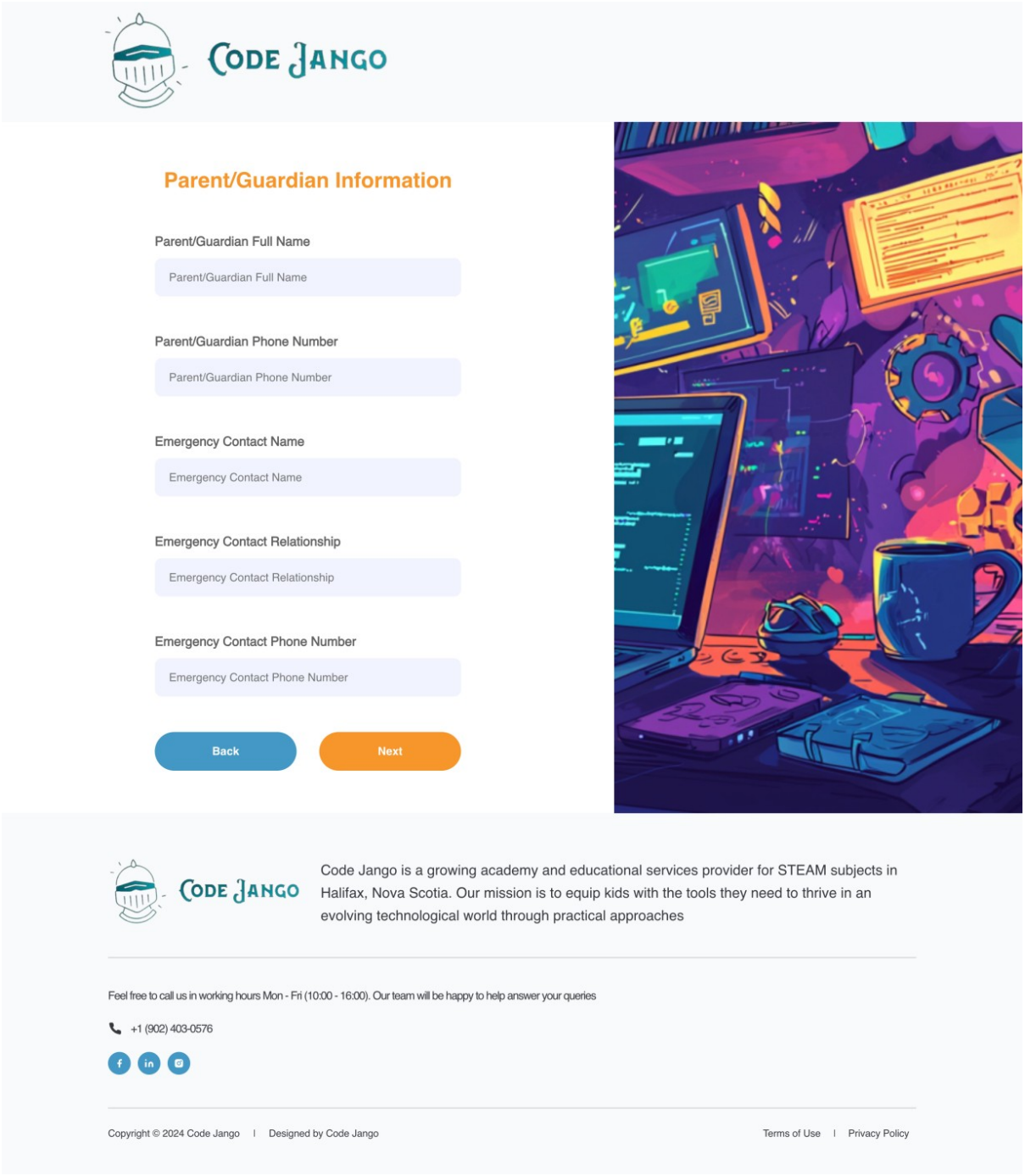

Figure 4.11: Updated Course Registration Form

Each input field was built as a reusable component with consistent styling, inline labels, and validation rules. These were wrapped in styled containers to match the application's visual language, ensuring full alignment with our design system. To handle validation and error display, I used **React Hook Form**, which provided an elegant **API** for field-level validation while still allowing manual control where necessary.

Client-side validation logic ensured that users could not move to the next step with-

out completing mandatory fields. Error states were clearly indicated below each field, and form navigation buttons were dynamically disabled or enabled based on validation results—ensuring both **usability** and **data integrity**.

4.5.1 Stripe Payment Integration

For payment handling, I integrated the **Stripe SDK** using the test (**sandbox**) environment. From the backend, I created dynamic **checkout sessions** and linked them to form submissions using secure JWT tokens. These tokens were embedded in the `success_url` and `cancel_url` parameters, allowing the server to securely identify and verify the transaction after **redirection**.

Upon successful payment, the backend would decode the token and finalize the registration by updating the user's status in the database. In contrast, if the payment was canceled, the corresponding unverified entry was flagged for deletion. This setup maintained clean records while providing a seamless user experience.

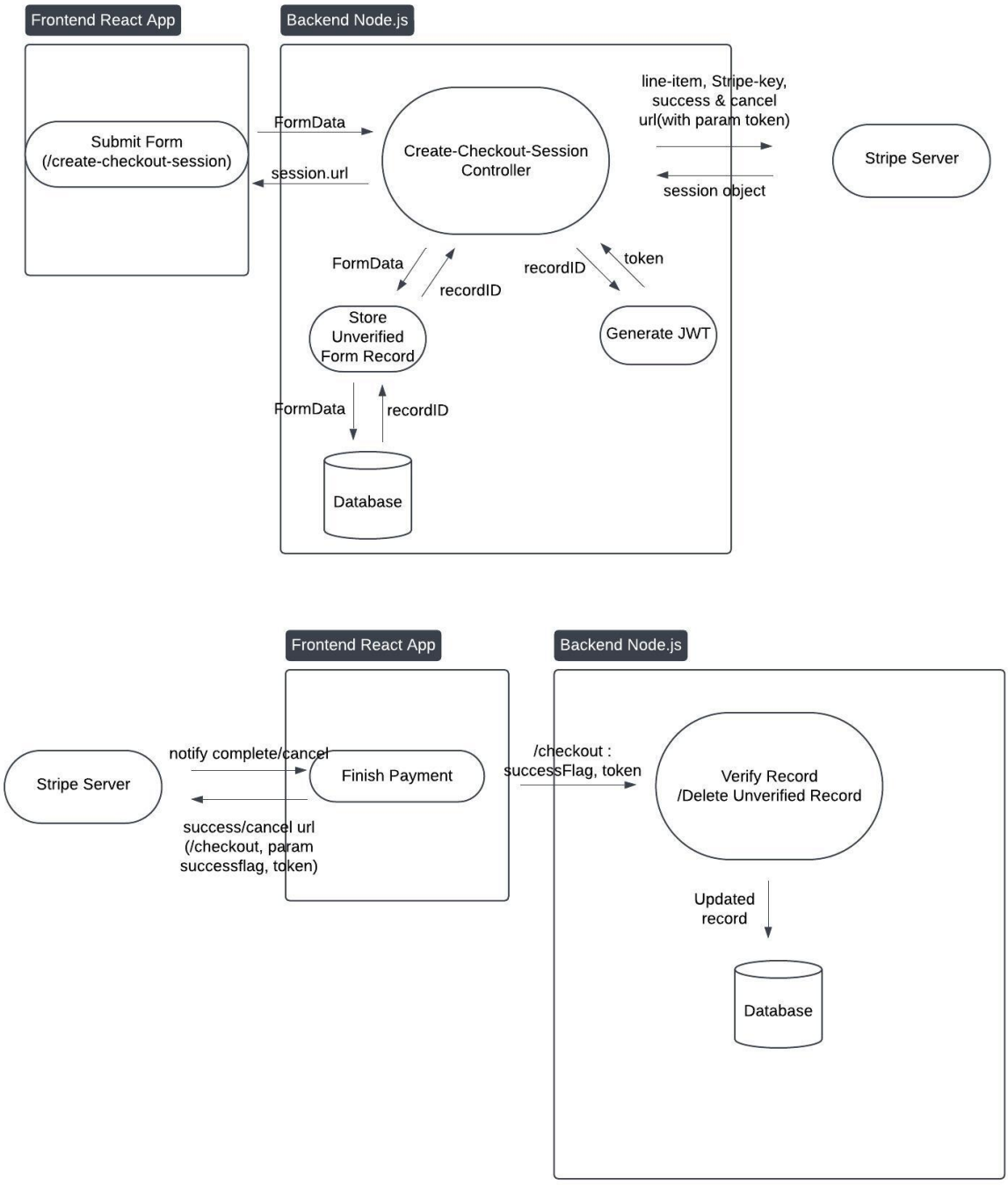

Figure 4.12: Data Flow Diagram For Integrating Stripe Payment System

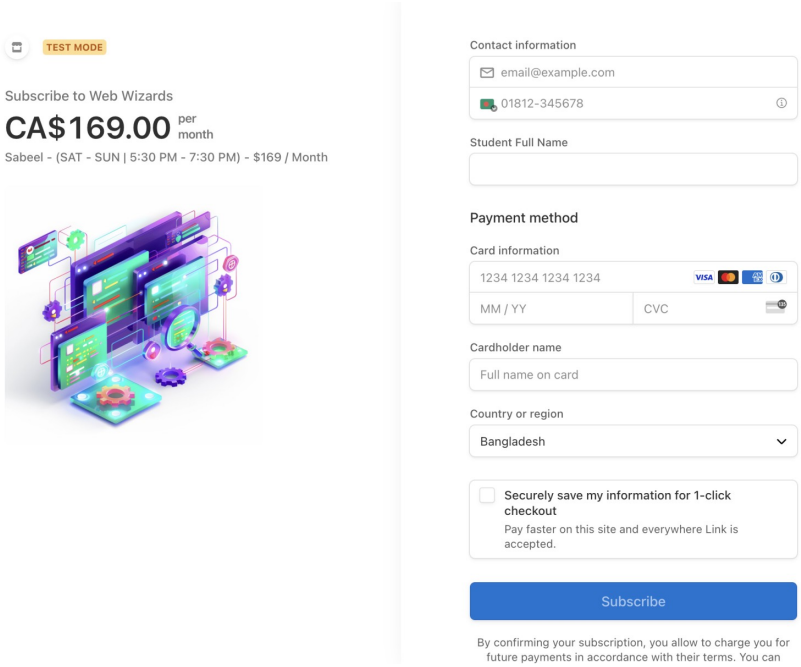

Figure 4.13: Stripe Payment Page

4.5.2 Addressing Security Concerns

To support a smooth user journey, the system temporarily allowed form submissions to be stored before payment was confirmed. However, I recognized a potential **attack surface**: a malicious actor could exploit this behavior to launch a **denial-of-service (DoS)** attack by repeatedly submitting large volumes of fake entries, thereby flooding the database.

I promptly reported this concern to the Technical Solutions Lead. As a result, we decided to postpone full deployment of the payment integration until safeguards—such as **rate limiting**, **CAPTCHA** verification, and **request throttling**—could be implemented to mitigate this risk.

4.6 Building the Foundational Learning Management System (LMS) at CodeJango

My backend engineering contributions at CodeJango were centered on the critical design and implementation of the Learning Management System (LMS), a foundational initiative initiated to fully digitize the company's existing educational and class management processes. This work constituted a significant platform shift, moving operations from physical, paper-based workflows to a fully digital, scalable, and data-driven educational environment.

4.6.1 Foundational Platform Shift: Requirement Engineering and System Modeling

The scope of this project required strict adherence to a rigorous **Software Development Life Cycle (SDLC)** methodology, beginning with deep collaborative analysis to translate complex analogue educational operations into a functional digital system.

Requirements Elicitation and Collaborative Analysis

Before commencing any coding, I prioritized robust **requirements engineering**. I facilitated multiple **requirements elicitation** sessions (analogous to **Joint Application Development (JAD)** sessions) with the Technical Solutions Lead and the instructors. These collaborative meetings were essential for capturing the critical existing administrative and academic processes that needed digitization.

We mapped out real-world educational dynamics, including how instructors manage assignments based on student progress, how courses are structured with multi-level progression, and the necessary logic for preventing students from accessing tasks beyond their current achievement level.

System Modeling and the Software Requirements Specification (SRS)

Following the comprehensive analysis, I produced a formal and comprehensive **Software Requirements Specification (SRS)**. This document detailed the to-be system, ensuring all system boundaries and interactions were clearly defined. Inspired by **structured systems analysis** methodology, I elaborated each use case with clear definitions of the **Actors** (Student, Instructor, Admin), **preconditions**, **postconditions**, and detailed main and alternative flows.

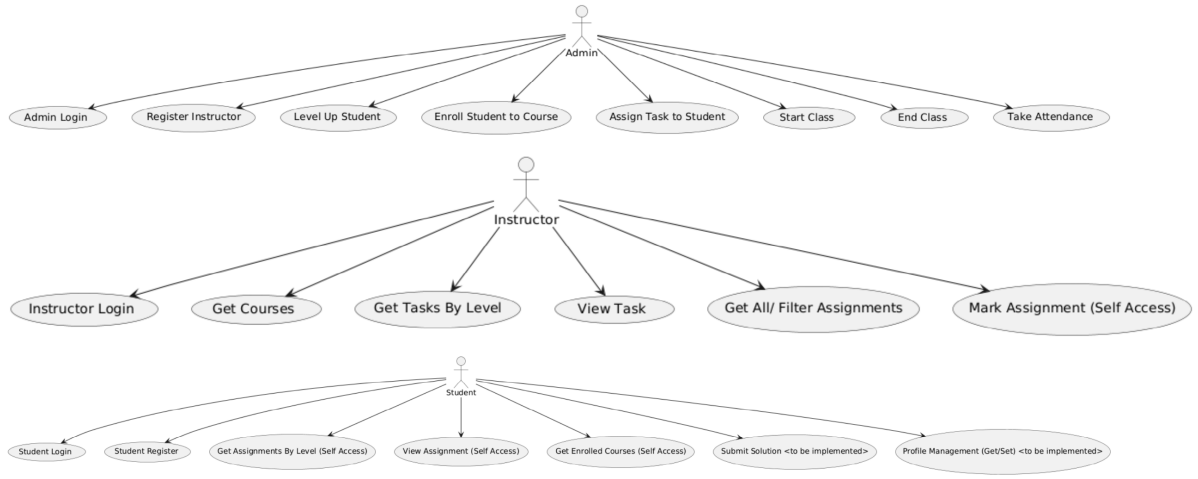

Figure 4.14: Use Case Diagram For LMS

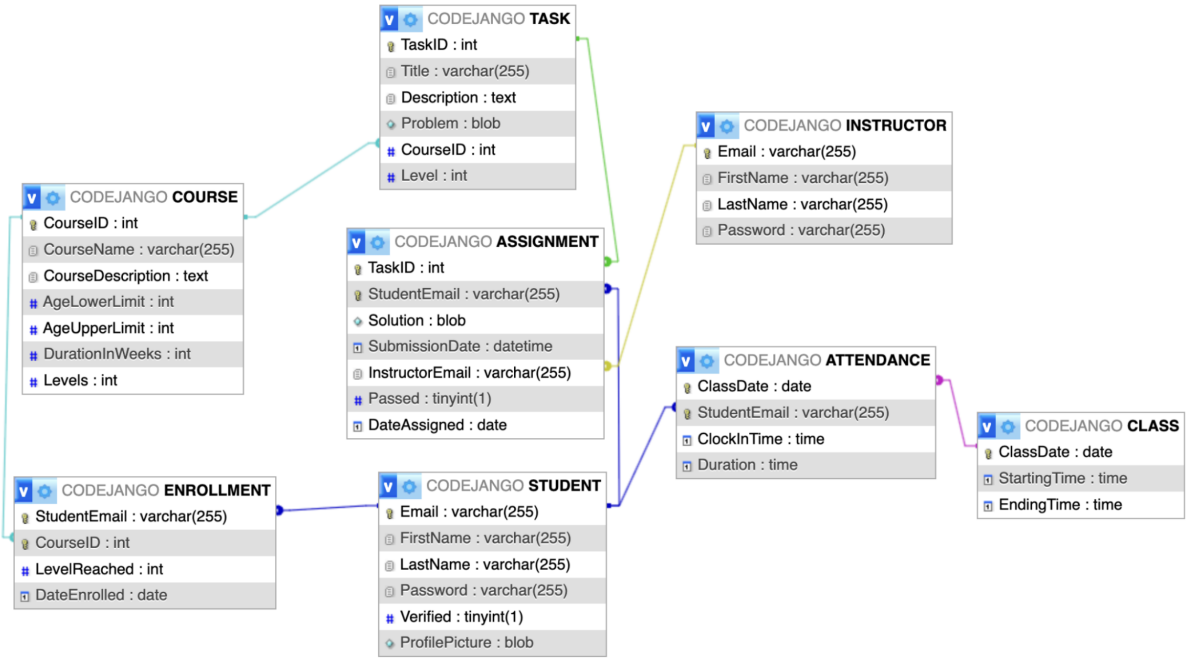

Figure 4.15: Relational Database Design For LMS

Use Case	Admin	Instructor	Student	Additional Conditions
Login	✓	✓	✓	N/A
Get Courses	N/A	✓	N/A	Requires Instructor AuthToken
Get Tasks By Level	N/A	✓	N/A	Requires Instructor AuthToken; level ≤ Levels of the Course
View Task	N/A	✓	N/A	Requires Instructor AuthToken
Get All Assignments	✓	✓	N/A	Requires Instructor or Admin AuthToken
Mark Assignment	N/A	✓(self-access)	N/A	Requires Instructor AuthToken; Only unmarked assignments or those marked by the same instructor
Get Enrolled Courses	N/A	N/A	✓(self-access)	Requires Student AuthToken; Only courses the student is enrolled in
Get Assignments By Level	N/A	N/A	✓(self-access)	Requires Student AuthToken; Only assignments for the student's enrollment and within their level limit
View Assignment	N/A	N/A	✓(self-access)	Requires Student AuthToken; Only assignments belonging to the student
Register Instructor	✓	N/A	N/A	Requires Admin AuthToken
Level Up Student	✓	N/A	N/A	Requires Admin AuthToken; Enforces max course level
Enroll Student to Course	✓	N/A	N/A	Requires Admin AuthToken; Ensures no duplicate enrollment
Assign Task to Student	✓	N/A	N/A	Requires Admin AuthToken; Ensures task level ≤ levelReached
Start Class	✓	N/A	N/A	Requires Admin AuthToken
End Class	✓	N/A	N/A	Requires Admin AuthToken; Checks if the class exists and hasn't ended
Take Attendance	✓	N/A	N/A	Requires Admin AuthToken; Ensures the class exists and attendance hasn't been logged

Figure 4.16: RBAC Matrix For LMS Functionalities

This SRS became the **single source of truth**, detailing complex actions like Mark Assignment, Level Up Student, and Take Attendance. The resulting **analysis models** derived from this phase were formally verified and signed off by the Technical Lead, enabling the seamless transition into the subsequent **design** and **implementation** phases.

4.6.2 Software Architecture and Model-Driven Implementation

With the system requirements formalized, I transitioned to designing a **layered**, **RESTful API architecture** using a robust, scalable technical stack.

Backend Stack and Component Architecture

The core technical stack chosen for the LMS backend was **MySQL**, **Node.js**, and **Express.js**. This combination provided a robust environment for building scalable RESTful APIs while ensuring a necessary separation of concerns. The backend was designed to be intentionally **frontend-agnostic**, ensuring the API was fully consumable by any client application, including the React application being developed.

The overall backend architecture was structured into six primary components to ensure clean separation of concerns and maintainability: **Database**, **Model Facades**, **Middleware**, **Controllers**, **Routers**, and the **Server**.

Model-Driven Development (MDD) Augmented by AI

I adopted a rigorous **Model-Driven Development (MDD)** methodology, grounded in the precise **analysis models**—including conceptual classes, attributes, associations, and system sequence diagrams—defined within the **Software Requirements Specification (SRS)**. MDD is a paradigm where development is centered on creating and transforming high-level **abstraction models** into executable code, rather than manual programming. The core premise is that well-specified models serve as the single source of truth from which implementation artifacts can be derived.

The integration of **Artificial Intelligence (AI)**, specifically **Large Language Models (LLMs)**, acted as the pivotal force that transformed this theoretical approach into a practical and highly efficient pipeline. Traditionally, MDD requires complex **model transformation tools** or manual translation, which can be a bottleneck. In this project, I utilized **ChatGPT** as an intelligent, dynamic **model compiler** to bridge this gap seamlessly.

The development workflow followed a clear, iterative loop:

1. **Model as Specification:** The verified SRS models provided the **formal input** for generation.
2. **AI-Powered Generation:** I presented structured prompts to the LLM, encapsulating model elements and **implementation intent** (e.g., "generate a `DataMapper` class for the Enrollment aggregate with the following attributes...").
3. **Artifact Output:** ChatGPT produced foundational code skeletons, **SQL queries** with **prepared statements**, and other **boilerplate code**.
4. **Engineer-in-the-Loop Review:** I then meticulously reviewed, tested, and refined

the AI-generated output. This involved enforcing **security best practices** (ensuring all database interactions used **parameterized queries**), validating **business logic** against post-conditions, handling **edge cases**, and optimizing for performance.

For instance, I would describe a facade method's responsibility (e.g., `increaseLevel` for the Enrollment facade) and its expected **SQL** logic, and ChatGPT would generate a draft implementation using **mysql2 prepared statements**. I then rigorously reviewed, tested, and adjusted this draft code to ensure it matched the exact flow specified in the SRS, handled **edge cases**, and adhered to **security best practices**, such as using **parameterized queries**. This process demonstrably showcased my strong understanding of the underlying technologies while integrating AI into a disciplined software engineering workflow.

4.6.3 Establishing the Secure, Componentized API Core

The implementation focused on building the core architectural layers responsible for data access, security, and business logic enforcement.

Role-Based Access Control (RBAC) via Middleware

Role-Based Access Control (RBAC) was central to the system security design, enforced at the **middleware** level to ensure permissions were enforced per use case. This was managed by the `authMiddleware`, which intercepts incoming requests.

The `authMiddleware` was responsible for **decrypting** the JSON Web Token (JWT). It extracted the user's email and role (Admin, Instructor, or Student) from the token payload. This critical user information was then augmented onto the **request object** (`req.user`), allowing the subsequent Controllers and Routers to enforce appropriate permissions based on the actor making the request. Additionally, a centralized `errorhandlingMiddleware` was implemented for standardized **error management** and **logging** across the system.

Model Facades (Data Access Layer)

The **Model Facades** were implemented to abstract **database interactions** and collaborate directly with the MySQL database, providing a clean API for the business logic layer. Key functionalities provided by these facades included:

- **Course Facade:** Handled operations such as retrieving all available courses (`getAllCourses()`) and getting course level details (`getLevelsById(courseId)`).

- **Assignment Facade:** Managed instructional functions, including retrieving all assignments (`getAllAssignments()`), filtering assignments by level and course ID (`filterByLevelAndCourseID`), and updating assignment status via `markAssignment()`.
- **Enrollment Facade:** Managed student enrollment actions, such as fetching courses a student is enrolled in (`getEnrollmentsByStudentEmail()`) and creating new enrollments (`createenrollment()`).
- **Attendance Class Facades:** Handled class scheduling (`createclass()`, `endClass()`) and logging student attendance (`createAttendance()`).

Controllers (Business Logic Layer)

Controllers abstract the system's use cases for each actor, collaborating extensively with the Model Facades to fulfill their responsibilities.

- **adminController:** Implements high-level administrative functions. Responsibilities include registering new instructors (`createInstructor()`), leveling up students (Level Up Student), enrolling students into courses (Enroll Student to Course), assigning tasks (Assign Task to Student), and managing the class lifecycle (start, end, and attendance logging). This controller collaborates with the Course, Enrollment, Task, Assignment, Instructor, Class, and Attendance facades.
- **instructorController:** Handles instructor-specific functions, primarily dealing with course management (getting all courses), retrieving tasks by level, and the critical function of marking student assignments as passed or failed (`markAssignment()`).
- **studentController:** Implements student-facing functions, including logging in, fetching their enrolled courses (Get Enrolled Courses), and retrieving their assignments filtered by course and level (Get Assignments By Level).

System Validation in a UI-Less Environment

Since the frontend UI design was not yet complete, backend functionality was tested directly through **Postman**. I constructed detailed **collections** and **test cases** that simulated real user interactions, exercising every major flow specified in the SRS use cases.

Using structured Postman requests, I systematically verified: user authentication and JWT issuance, course enrollment flows, assignment submission and grading, and class lifecycle management. Each endpoint was validated against the explicit **constraints** and

logic defined in the SRS. **MySQL Workbench** was used alongside testing to confirm **database integrity** and verify the accuracy of all inserts, updates, and relational links.

This process ensured that the backend was fully operational, logically correct, and ready for **UI integration**—delivering the secure, role-based foundation and scalable data model required for CodeJango's digital transformation.

Chapter 5

Conclusion – Personal and Professional Growth

My six-month internship at Code Jango has been a transformative experience that has not only strengthened my technical capabilities but also shaped me into a more disciplined, adaptable, and communicative professional. Beyond frontend development and software design, this internship tested and refined a range of critical soft skills that are essential for success in any professional environment.

5.1 Time Management and Self-Discipline

One of the most significant areas of growth was in time management. Working remotely and often asynchronously required me to develop a strong sense of personal accountability. Without traditional office oversight, I had to manage my workload independently—setting realistic timelines, prioritizing tasks, and maintaining consistent progress on deliverables.

To stay organized, I relied heavily on tools like Jira and personal to-do systems. I learned to break down large tasks into smaller, achievable units and align them with our weekly sprint cycles. This practice helped me avoid procrastination, track my progress, and ensure timely completion of both short-term tasks and long-term milestones.

This experience instilled in me the value of proactive planning, as well as the importance of maintaining work-life balance, especially in remote environments where boundaries can easily blur.

5.2 Working Under Pressure and Managing Uncertainty

Startups, by nature, operate in fast-paced and evolving environments—and Code Jango was no exception. I often faced ambiguity in product requirements, rapidly approaching deadlines, and limited documentation. These situations forced me to step outside my comfort zone and make informed decisions under pressure.

Rather than viewing uncertainty as a hurdle, I learned to treat it as a space for growth. I became more comfortable with iteration and adopted an agile mindset, focusing on continuous delivery and refinement. During high-pressure situations, I found that clearly documenting questions, proposing possible solutions, and proactively seeking feedback helped accelerate decision-making while reducing stress.

This strengthened my resilience and problem-solving skills, especially when I had to debug critical features or adapt to sudden changes in product scope.

5.3 Communication Skills

Remote collaboration made clear and concise communication an essential skill. I learned to write detailed technical updates, document my development decisions, and clearly express questions and blockers during sprint meetings. I regularly presented my progress to non-technical stakeholders, which helped me learn how to translate technical concepts into accessible language.

Additionally, contributing to architectural documentation—such as use-case diagrams, data models, and component structures—sharpened my ability to organize and present technical knowledge for shared understanding.

I also became more confident in asking for clarification, providing constructive feedback, and ensuring that every team interaction moved the project forward efficiently.

5.4 Interpersonal and Team Collaboration

Despite being the sole developer, I never felt isolated. The co-founders created a supportive and collaborative environment, encouraging me to share my ideas freely and engage in open discussions about product design and priorities.

I learned to listen actively, respect diverse perspectives, and work towards shared goals

even when there were conflicting opinions. This enhanced my interpersonal empathy and teamwork abilities—skills that are especially crucial in multidisciplinary teams involving both technical and non-technical contributors.

My interactions during planning and review sessions also helped me appreciate the value of feedback loops and how collaboration can elevate both the product and the people behind it.

5.5 Leadership and Ownership

Although I joined as an intern, I was given full responsibility for the development of Code Jango's web presence. This opportunity challenged me to step up as a leader, not only in technical execution but also in initiative-taking and vision alignment.

I learned how to make architectural decisions that balanced business goals with long-term scalability, how to document processes so others could build upon my work, and how to navigate competing priorities with maturity and focus.

Owning end-to-end delivery taught me the importance of leading by example, maintaining code quality, and thinking like a product owner. This experience has laid the groundwork for future leadership roles where technical expertise must be paired with accountability and strategic thinking.

5.6 Final Reflection

Overall, this internship has been a milestone in my professional development. I entered with a passion for coding and a solid foundation in software engineering. I leave with far more: a deeper understanding of real-world application development, stronger self-management habits, enhanced communication skills, and the confidence to take ownership of challenging projects.

These experiences have not only prepared me for future roles in software development but have also shaped the kind of team member, collaborator, and leader I aspire to be. I am grateful for the trust placed in me and the guidance I received, and I look forward to applying these lessons to every future challenge I take on.