# ACKNOWLEDGEMENT

I am greatly indebted to my internship guide **Shivprasad Titare** for his able guidance throughout this work. It has been an altogether different experience to work with him and I would like to thank him for help, suggestions and numerous discussions.

I gladly take this opportunity to thank **Dr. A. M. Rajurkar** (Head of Computer Science & Engineering, MGM's College of Engineering, Nanded).

I am heartily thankful to **Dr. Lathkar G. S.** (Director, MGM's College of Engineering, and Nanded) for providing facility during progress of project, also for her kindly help, guidance and inspiration.

Last but not least I am also thankful to all those who help directly or indirectly to develop this seminar and complete it successfully.

With Deep Reverence,

Purva Chalindrawar (257)

# ABSTRACT

This internship report presents a comprehensive overview of the technical and professional journey undertaken during a web development internship at Indian Cyber Security Solutions (ICSS). The internship encompassed a diverse array of frontend and backend development projects, enabling practical exposure to real-time software engineering workflows and organizational standards. Throughout the internship, significant contributions were made to various components of ICSS's website and internal tools. These included responsive UI development using Bootstrap and React.js, implementation of Two-Factor Authentication (2FA), dynamic calendar interfaces using Google Sheets API, validation-enhanced forms, and backend functionalities via Node.js and MongoDB. A specialized Electron.js-based desktop application integrated with a compiled C program was also developed and tested.

The internship provided experience in system design modelling through the creation of multiple levels of Data Flow Diagrams (DFDs) for the SAVE project, covering authentication and vulnerability scanning workflows. Exposure to Git-based collaboration, and version control systems further refined professional competencies. This report meticulously documents all tasks and projects handled, explores the tools and technologies used, and outlines the impact and learning outcomes derived. It stands as a reflection of the intern's growth as a full-stack developer, demonstrating a blend of technical aptitude and collaborative problem-solving in a professional environment.

# TABLE OF CONTENTS

# List of Figures

**Chapter 1**

# INTRODUCTION

An internship is a valuable learning experience that bridges the gap between academic concepts and real-world applications. At Indian Cyber Security Solutions (ICSS), I worked as a Web Developer Intern where I was exposed to dynamic web technologies and secure coding practices. The internship enabled me to contribute to professional projects involving frontend design, backend development, API integrations, laying a strong foundation for my career in full-stack development.

## 1.1 Background

As a part of the academic curriculum for the Bachelor of Technology in Computer Science and Engineering at MGM's College of Engineering, Nanded, I undertook an internship at Indian Cyber Security Solutions (ICSS). The internship commenced on March 1, 2025 spanning a period of 5 months. ICSS is a renowned cybersecurity company offering various services ranging from cyber defense, ethical hacking, secure web solutions, and penetration testing to enterprise-grade application development.

Indian Cyber Security Solutions (ICSS) is a renowned IT security company based in India, known for delivering cybersecurity solutions, secure application development, and corporate training across multiple verticals. With a strong focus on innovation and practical implementation, ICSS caters to clients globally in both public and private sectors. The organization is driven by a mission to secure digital ecosystems by developing secure applications, performing security audits, and nurturing the next generation of cyber professionals. ICSS maintains a team of skilled developers and ethical hackers who collaborate to provide secure, scalable, and performance-driven IT solutions.


**Fig. 1.1: Company Logo**

During the course of the internship, I was involved in the complete development lifecycle of several web modules, ranging from static page designs to secure backend integration. I worked extensively with modern frontend technologies including HTML, CSS, Bootstrap, and JavaScript, and gradually progressed towards dynamic frontend frameworks such as

React.js. My backend responsibilities primarily involved handling Node.js and MongoDB environments, designing validation logic, implementing authentication systems such as two-factor authentication (2FA) with SMTP, and conducting API integration and testing using Postman.

In addition, I also contributed to the maintenance and enhancement of existing modules such as career page input validation, media page redesign, and calendar scheduling automation through Google AppScript.

The organizational workflow emphasized secure development practices, version control through Git, and collaborative teamwork through regular code reviews. This professional environment helped me strengthen not only my technical proficiency but also my documentation skills, communication, and ability to manage tasks independently. All the projects and code modules developed during the internship have been versioned and maintained in GitHub repositories to reflect transparency, traceability, and continuous learning throughout the internship period.

## 1.2 Project Overview

The internship at Indian Cyber Security Solutions (ICSS) offered extensive exposure to both frontend and backend development, along with a strong emphasis on security practices and real-world application of modern web technologies. The primary objective of the internship project was to bridge the gap between academic knowledge and professional development standards by engaging in task-oriented modules that enhanced not only technical capabilities but also analytical, collaborative, and documentation skills.

The project scope spanned across multiple layers of application development including designing responsive interfaces, integrating secure authentication protocols, updating content dynamically via databases, and automating content processing through cloud-based tools and JavaScript frameworks.

**I. Frontend Development & UI Enhancement**

A significant portion of the internship was devoted to designing and refining the user interface across various web modules. Starting from introductory tasks such as learning Bootstrap and designing pricing cards, the work progressed into more sophisticated implementations like animated sliders, client-specific landing pages, media sections, and custom UI components using React.js.

All components were designed with responsiveness, usability, and clarity in mind aligning with ICSS's UI/UX standards. Additionally, the frontend designs incorporated dynamic

validations using JavaScript, ensuring accurate and secure data collection.

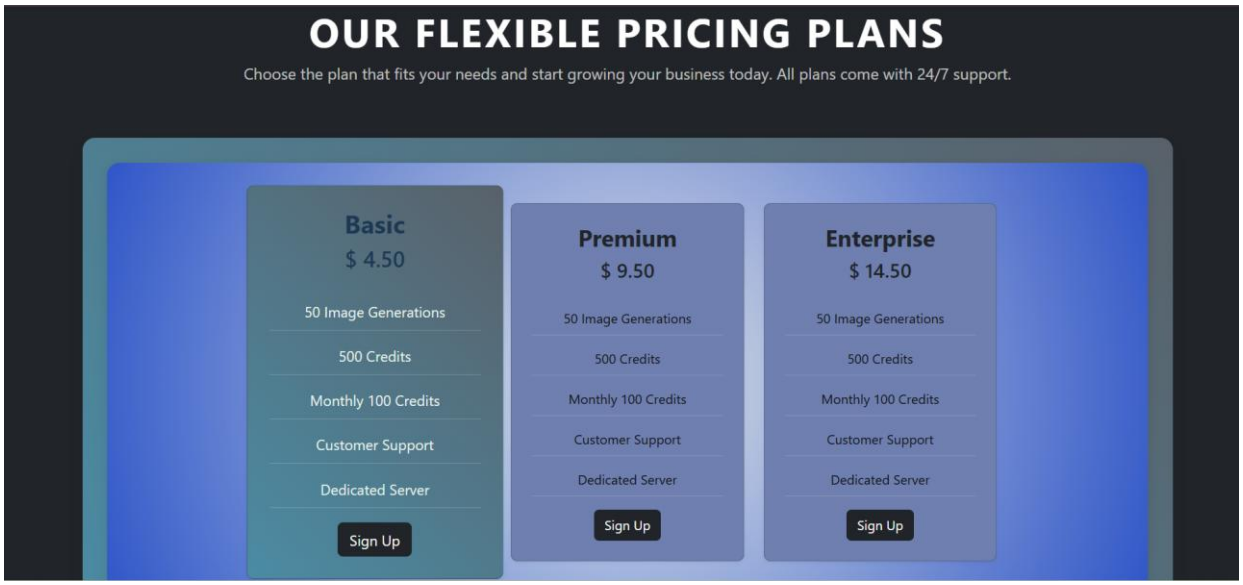


**Fig. 1.2: Bootstrap-Based Pricing Card Layout**

The above image represents the flexible subscription models offered within the AI Videos platform. Users can choose from Basic, Premium, and Enterprise plans based on their image generation needs, credit limits, and support preferences. Each plan is designed to provide scalable access to platform features, ensuring affordability, dedicated resources, and 24/7 customer support for diverse user requirements.

**II. Backend Development & Security Integration**

The backend segment of the internship involved practical implementation of server-side logic using Node.js. This included creating login systems with Two-Factor Authentication (2FA) via SMTP, implementing data validations for secure inputs, and integrating Google APIs for services like file uploads and Drive access.

Security was a recurring theme across all backend tasks especially evident in the log management system developed for sign-in, sign-up, and MFA activities, where timestamped data was stored in structured JSON format and limited to recent 7-day logs for auditability.

**III. API Testing, Automation, and Google Cloud Integration**

Tasks such as Postman-based API testing and calendar schedule automation using Google AppScript helped in developing a deeper understanding of how modern web systems integrate external APIs and third-party services. These tasks were not only functional but also required clean documentation and deployment pipelines.

Additionally, file upload functionalities were implemented using JavaScript and Google Drive API v3 demonstrating proficiency in working with OAuth credentials, cloud console

setup, and async JS logic.

**IV. Database Work & Real-Time Data Handling**

The internship covered database interactions using both MongoDB and MySQL (phpMyAdmin). From updating student databases and form data validations to creating entire database tables, the exposure ensured hands-on practice in CRUD operations, export-import workflows, and integrating frontend forms with backend data logic.

**V. Desktop App Development with Electron & C Integration**

One of the most impactful tasks involved working on an Electron-based application, where the login and submission processes were executed through an integrated C program compiled and controlled via Node.js. This cross-domain exposure allowed for working on desktop software within a web-stack environment — showcasing versatility and real-time system-level programming.

**VI. Project Hosting & GitHub Management**

All tasks performed, from web modules to script logic and automation flows, were version-controlled and uploaded to GitHub repositories. Each repository reflects the evolution of the task from assignment to deployment and serves as both a portfolio and a reference for future development.

## 1.3 Objectives

The objective of this internship was to provide a comprehensive and structured platform for implementing academic learning in real-world web development and cybersecurity environments. Through direct engagement with active projects at Indian Cyber Security Solutions (ICSS), I aimed to strengthen both technical and analytical competencies, while gaining exposure to modern tools, frameworks, and industry-standard development workflows. The tasks assigned covered a wide range of responsibilities, enabling me to enhance my understanding of frontend and backend technologies, authentication systems, cloud integration, API testing, and secure coding practices.

Each task was tied to practical goals designed to foster core competencies required in professional full-stack development. The following key objectives guided my work throughout the internship:

**I. To gain hands-on experience in modern frontend development**

The internship began with structured learning and task-based implementation using HTML5, CSS3, Bootstrap, and JavaScript. I developed responsive interfaces, including pricing card layouts, media pages, client landing pages, and animated sliders. I was trained

to write clean, maintainable code and apply industry-level design standards to all UI elements.

**II. To understand and implement secure login systems with Two-Factor Authentication**

Security was a major focus during the internship. One of the core objectives was to build and integrate a complete login system with 2FA using email-based OTP verification via SMTP. I implemented verification logic, form validation, and input sanitization to ensure data security.

**III. To develop backend logic using Node.js and asynchronous JavaScript**

The internship offered a real-world setting to understand Node.js concepts, from writing route handlers to asynchronous functions. I built API integrations and created reusable backend logic for data validation, file handling, and task automation.

**IV. To perform database operations using MongoDB and MySQL**

My role involved practical exposure to both NoSQL (MongoDB) and relational (MySQL) databases. I created databases and tables in phpMyAdmin, managed data imports via CSV, and implemented basic CRUD operations. I also studied MongoDB architecture and how it integrates with backend Node.js servers.

**V. To integrate cloud services using Google Drive API and AppScript**

Cloud-based automation was another learning goal. I enabled Google Drive API through the Google Cloud Console, created OAuth clients, and implemented file upload functions using JavaScript. Additionally, I designed calendar pages using AppScript to fetch and display course schedules dynamically.

**VI. To test APIs and validate data using Postman and JavaScript**

I conducted extensive API testing on endpoints using Postman. This included SAVE project APIs and login-related endpoints. I also worked on validating HTML forms using custom JavaScript logic to restrict invalid inputs and ensure proper data formats in user-facing modules.

**VII. To build a desktop application using Electron and integrate system-level C programs**

A unique aspect of the internship was the development of a lightweight Electron application with multiple pages and native integration with a C program. I compiled and executed agentScript.c within the Electron environment and displayed feedback based on its output.

**VIII. To implement logging mechanisms for key events in the system**

To simulate a production-level environment, I created structured JSON-based log files to

record SignIn, SignUp, and MFA events. Each log captured the timestamp, public IP address, and event name for tracking purposes, limited to a seven-day retention window.

**IX. To analyze and document system workflows using Data Flow Diagrams (DFDs)**

The SAVE project required system-level documentation. I created DFD-0, DFD-1, and DFD-3 diagrams to represent the authentication and scanning modules. This task improved my understanding of how different system components communicate and transfer data.

**X. To manage all codebases using Git and GitHub**

Version control was maintained using Git, with all source code hosted on GitHub. Each commit reflected a milestone or review update. This helped me track changes, collaborate smoothly, and maintain an organized structure for all modules worked on.

**Long-Term Learning Objectives**

a) To enhance confidence in initiating and completing full-stack applications independently.
b) To become proficient in using modern development stacks including Node.js, React.js, MongoDB, and Google APIs.
c) To prepare for future enterprise-grade software development roles with a practical foundation in secure systems.
d) To build an impressive professional GitHub portfolio showcasing project-based skills and code quality.
e) To utilize this experience as a launchpad for deeper involvement in web security, cloud platforms, and system design.

## 1.4 Scope of Work

The scope of this internship encompasses the development, enhancement, testing, and documentation of various web-based and system-integrated modules under the supervision of experienced mentors at Indian Cyber Security Solutions (ICSS). As a Web Developer Intern, the scope of work spanned a comprehensive range of technical activities, beginning from frontend interface design and extending to backend integration, cloud API usage, secure login systems, database management, and system documentation.

This section outlines the key areas that defined the scope of the internship:

**1. Frontend Design and Web UI Development**

- Designing and developing responsive web pages using HTML, CSS, and the Bootstrap framework.
- Implementing animated and dynamic components such as pricing cards, sliders,

client modules, and media sections.

- Ensuring compatibility across browsers and devices using responsive design principles.

**2. Backend Integration and JavaScript Functionality**

- Writing and testing asynchronous JavaScript and Node.js functions.
- Developing custom validation scripts for input fields, enhancing client-side reliability.
- Executing file-based functions for logging and form submissions.

**3. Secure Authentication System with Two-Factor Verification**

- Designing a login page with Two-Factor Authentication using SMTP email verification.
- Handling OTP generation, server-side validation, and error management.
- Integrating multi-step authentication flow with user feedback.

**4. Database Management and Integration**

- Working with MySQL via phpMyAdmin for student record databases and data import/export.
- Learning and applying MongoDB concepts including architecture, collections, and data handling.
- Creating schema designs for storing user logs and certificates.

**5. API Implementation and Testing**

- Performing API testing for internal SAVE APIs using Postman.
- Validating different HTTP methods (GET, POST) and ensuring secure data communication.
- Logging and debugging API responses with structured reporting.

**6. Electron App Development and System Integration**

- Designing and building a desktop-based Electron app with three functional pages.
- Integrating and executing C-based backend logic (agentScript.c) within the app.
- Managing dynamic execution, feedback rendering, and native file access.

**7. Cloud and Automation using Google APIs**

- Enabling and configuring Google Drive API through Google Cloud Console.
- Automating file uploads and OAuth integration using JavaScript.
- Developing dynamic course schedule pages by fetching data from Google Sheets via AppScript.

**8. Documentation and System Design (SAVE Project)**

- Drafting Data Flow Diagrams (DFD-0, DFD-1, DFD-3) for the SAVE system modules including authentication, web scanning, and network security.
- Preparing system-level documents for project workflow, process mapping, and functionality breakdown.

**9. Log Management System**

- Creating structured JSON-based logs for tracking events such as SignIn, SignUp, and MFA actions.
- Implementing logic to maintain logs for the last seven days based on timestamps and IP addresses.

**10. Deployment, Version Control & Code Hosting**

- Regularly using Git for version control and task management.
- Hosting source code and development modules on GitHub for collaboration and archival.
- Maintaining structured branches and commit logs as part of professional workflow.

Chapter 2

# WEB INTERFACE PROJECTS

The digital face of any modern organization lies in its user interface — a realm where design, interactivity, and usability intersect. During my internship at Indian Cyber Security Solutions (ICSS), I was assigned to design and improve various web interface components essential for the company's public-facing platform. These components not only needed to be visually appealing but also responsive, accessible, and aligned with the organizational brand.

This chapter outlines my involvement in designing frontend modules using HTML, CSS, Bootstrap, and JavaScript. From interactive pricing cards and demo websites to customized client pages and dynamic media galleries, each assignment contributed to elevating the user experience. It also enabled me to understand how minor frontend adjustments can significantly influence overall user engagement and branding.

## 2.1 Pricing Cards – Visual SaaS Components

A fundamental part of any service-based organization's website is how it presents its offerings to its users and pricing cards play a pivotal role in this context. These cards serve as interactive visual components that communicate service tiers, features, and pricing in a compact yet effective format. During my internship at Indian Cyber Security Solutions (ICSS), I was tasked with creating multiple pricing card designs that not only met the functional requirements but also aligned with the company's branding and UI consistency.

### 1. Understanding the Design Requirements

Before diving into implementation, I was guided by my mentor to explore different styles and references used in modern SaaS websites. This included analyzing card layouts from companies like Dropbox, Notion, and Trello. The objective was clear: I had to design two distinctive types of pricing cards one that was flat and minimal, and the other that was more layered and interactive, utilizing shadows and hover animations to create visual depth.

### 2. Planning and Execution

To ensure I followed the design standards of ICSS, I first sketched rough layouts and got them reviewed. Once approved, I started the visual execution. I made sure to use consistent font families, padding, and button styles that matched the company's web branding

guidelines.

**3. Visual Consistency and Accessibility**

An important aspect of this task was to maintain accessibility for users on different screen sizes and devices. I tested the pricing cards on various viewport settings to ensure that no element overlapped or broke under mobile or tablet dimensions.

**4. Challenges Faced and Resolutions**

One of the key challenges in this task was aligning variable-height content across multiple cards while keeping their widths and positions consistent in a grid layout. Differences in feature list lengths or plan details caused misalignments that affected the visual balance of the overall section. I overcame this by exploring layout techniques and consulting with the design team to use consistent vertical padding and fixed height constraints where necessary.

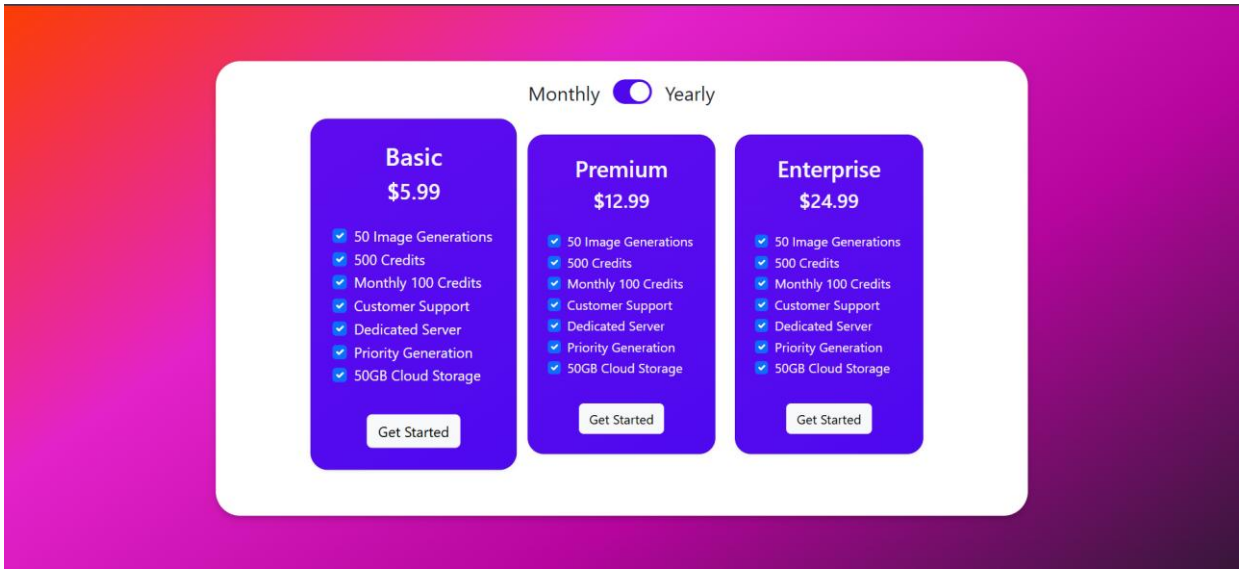


**Fig. 2.1: Pricing Card Layout**

The above image represents the flexible subscription models offered within the AI Videos platform. Users can choose from Basic, Premium, and Enterprise plans based on their image generation needs, credit limits, and support preferences. Each plan is designed to provide scalable access to platform features, ensuring affordability, dedicated resources, and 24/7 customer support for diverse user requirements.

## 2.2 Demo Website – Applying Bootstrap Learnings

Following the initial learning phase on Bootstrap, I was assigned my first independent project — creating a demo website. This task served as a foundational exercise to translate my theoretical understanding of HTML, CSS, and Bootstrap components into a working prototype. It was aimed at reinforcing my grasp of responsive design, mobile-first

development, and modern web layout structures. The objective was to develop a multi-section website from scratch using only the Bootstrap framework, without relying on any external libraries beyond what was permitted by the mentor.

**I. Planning and Layout Design**

Before development began, I spent time outlining the structural layout of the website. The proposed sections included:

- Hero Section: Featuring a full-width image and a call-to-action.
- Services Offered: Grid-based icons and descriptions of mock services.
- Why Choose Us: A comparative layout displaying strengths.
- Testimonials: Placeholder card reviews for client feedback.
- Contact Section: A styled form with labels and placeholders.

**II. Execution and Implementation**

The demo website emphasized cleanliness, usability, and simplicity. Bootstrap's card and utility classes were used extensively to manage margins, paddings, and shadow effects. The navigation bar was fixed to the top with responsive toggling, allowing it to collapse into a hamburger menu on smaller devices.

Interactive elements like buttons, hover effects, and accordion-style content blocks were also integrated. Bootstrap icons were added to enhance visual presentation. The contact form included name, email, subject, and message fields with proper placeholder text and spacing.

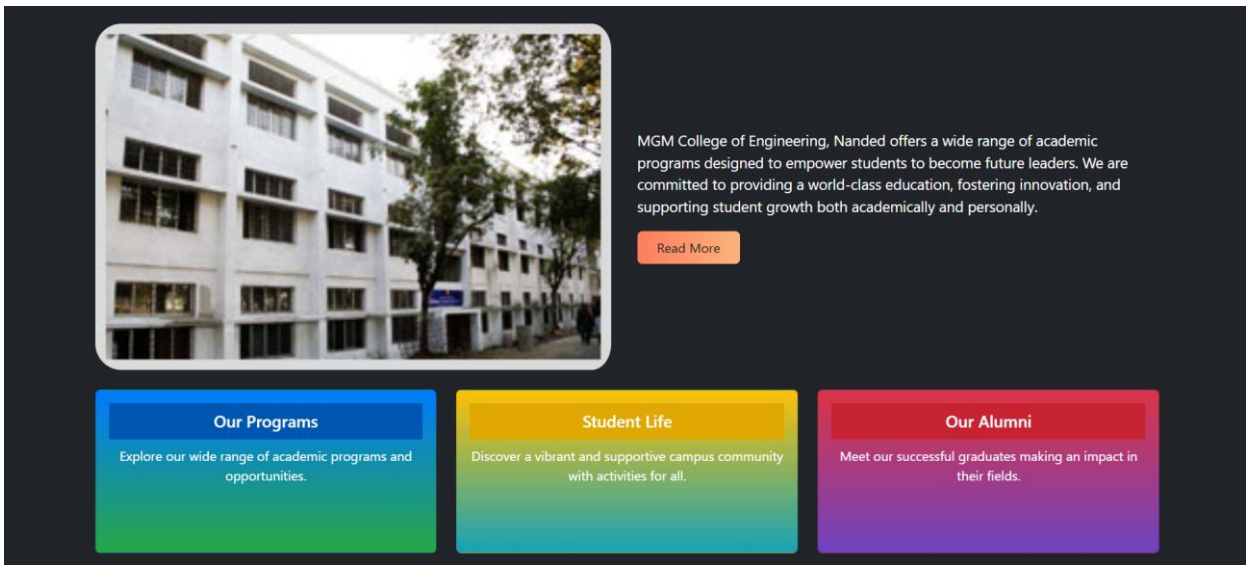


**Fig. 2.2: Demo Website-Home Page**

This section from the MGM College of Engineering website showcases the institution's core offerings, including academic programs, student life, and alumni network. The visual layout is designed to reflect the vibrancy of campus life and the institution's commitment to student engagement and career development.


**Fig. 2.3: Demo Website-About Page**

This snapshot represents the "About MGMCOE" section of the official college website, highlighting the rich legacy, multidisciplinary offerings, and the vision of Mahatma Gandhi Mission. It emphasizes the institution's mission to provide value-based education across domains like Engineering, Medicine, Law, and Management, along with its long-standing excellence in academic leadership.

## 2.3 Client Page Development

One of the major frontend tasks during the internship was creating web pages tailored for ICSS's clients. I was provided with client-specific content and branding guidelines to design dedicated pages that reflected their requirements while staying consistent with the ICSS platform.This task was especially important because these pages would represent ICSS's collaboration with various corporate clients and needed to follow a professional layout, with special emphasis on clarity, organization, and brand alignment.

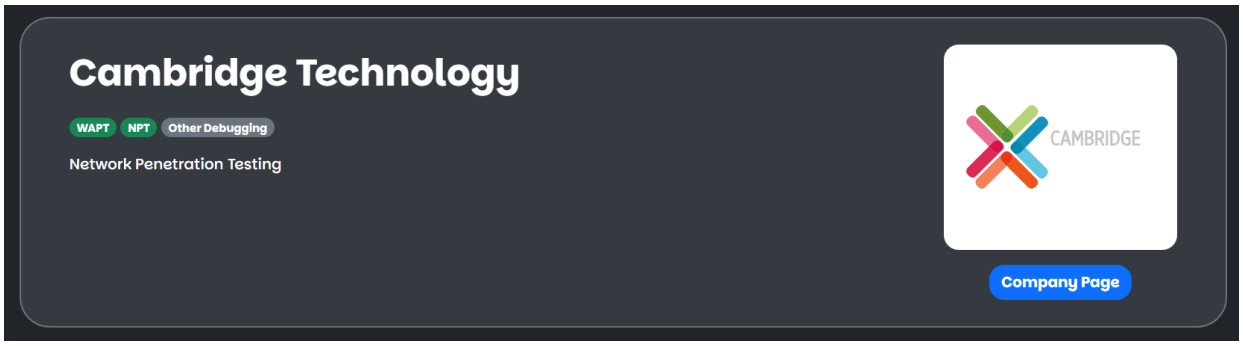


**Fig. 2.4: Cambridge Technology – Penetration Testing Overview**

This image displays a summary of the Network Penetration Testing service offered by

Indian Cyber Security Solutions (ICSS) for Cambridge Technology. The highlighted modules such as WAPT (Web Application Penetration Testing), NPT (Network Penetration Testing), and additional debugging tasks signify the scope of testing executed. The company branding and interface emphasize the client-centric focus and technological depth involved in the project.

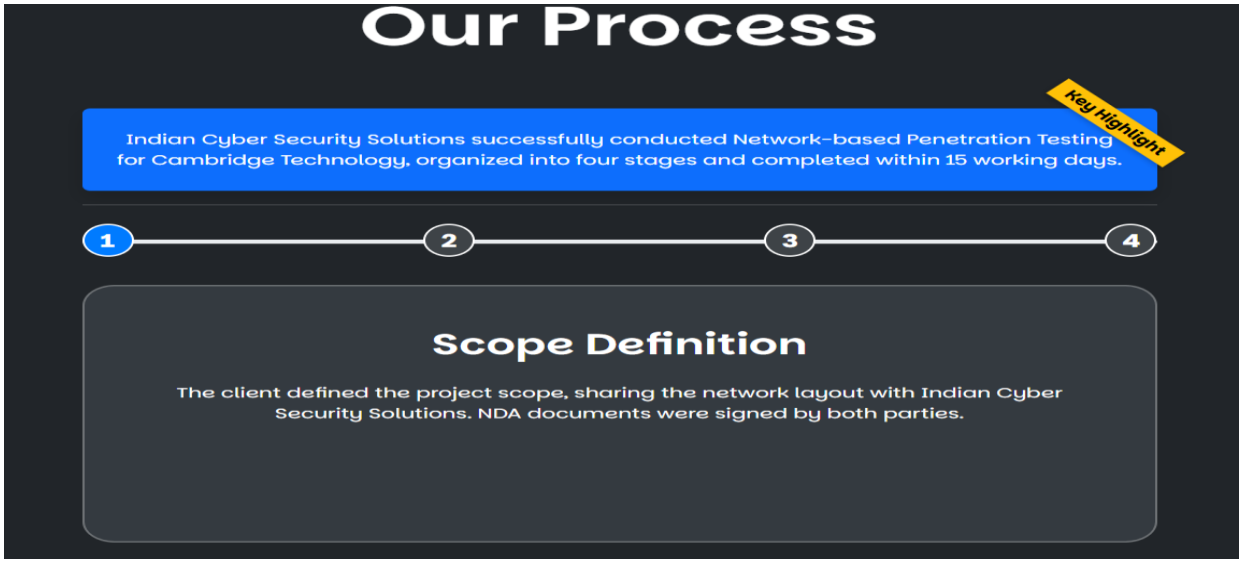


**Fig. 2.5: Penetration Testing Workflow and Scope Definition**

This image presents a visual breakdown of the penetration testing process conducted for Cambridge Technology. It outlines the four-stage project lifecycle starting with "Scope Definition", where the client shared the network structure and signed NDA agreements with ICSS. The progress bar design reflects a structured, milestone-driven approach essential for secure and efficient penetration testing delivery.

**I. Requirement Analysis**

Before starting, I received an Excel sheet listing various clients, their industry sectors, and the specific elements to include on each page. This included:

- A brief company profile
- Service type provided
- Visual identity
- Required call-to-action elements

The design followed a two-tone color scheme to distinguish ICSS branding from the client-specific visuals. Careful use of white space, font hierarchy, and card-based layouts helped

maintain a clean and engaging structure.

**II. Challenges and Testing**

One technical challenge was dealing with inconsistent image sizes for client logos and banners, which led to design misalignments. I resolved this by standardizing image dimensions through CSS class constraints.

I also ran A/B testing internally to compare design options and made iterative improvements based on mentor feedback.

## 2.4 Calendar Schedule Interface – Real-Time Dynamic Schedule Display

One of the most practically impactful tasks assigned during my internship at Indian Cyber Security Solutions (ICSS) was the development of a dynamic calendar schedule system that would enable students and prospective trainees to view up-to-date schedules for various cybersecurity training programs directly from the official ICSS website. This task stood out for its real-world application, end-user interactivity, and the technical challenge of integrating dynamic data from a Google Sheet backend using Google Apps Script and frontend JavaScript.

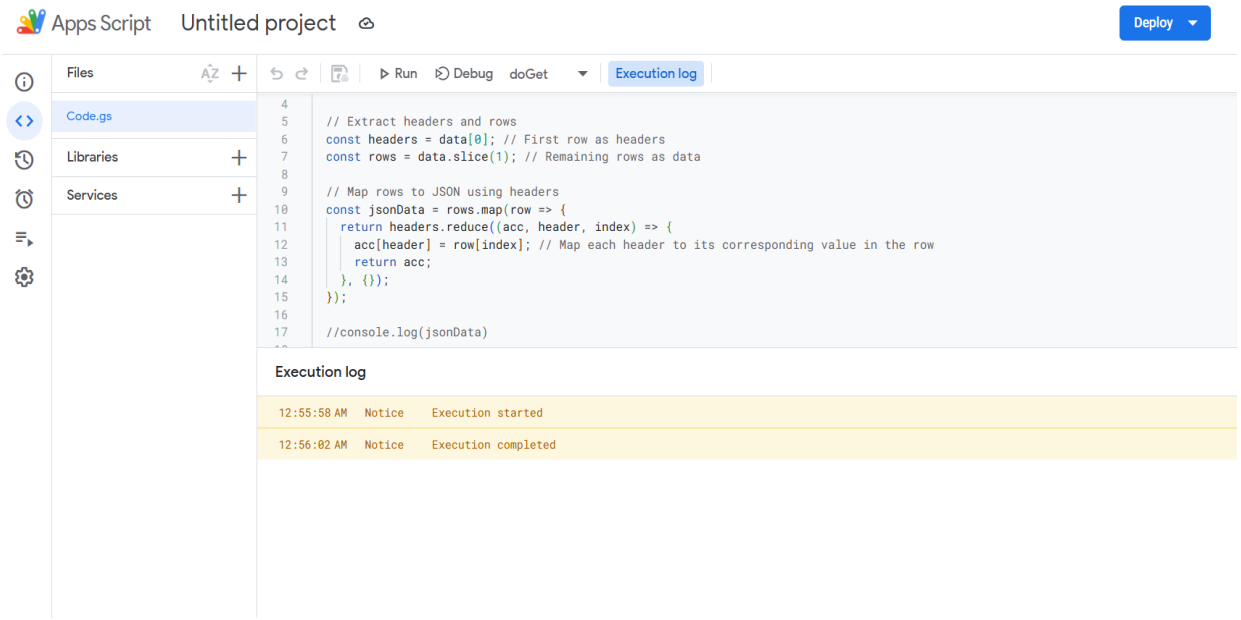


**Fig. 2.6: Google Apps Script Execution Environment for JSON Mapping**

This image showcases the Google Apps Script environment where JavaScript code is used to convert spreadsheet data into JSON format. The script extracts the first row as headers and maps the remaining data rows accordingly to produce a structured JSON object. The execution log at the bottom indicates a successful script run. This environment

demonstrates effective automation and scripting capabilities for integrating spreadsheet data with web-based applications or APIs, highlighting the importance of data manipulation and transformation in full-stack development.

**I. Planning the Workflow**

- Google Sheet Creation: A master sheet with all course schedules in one place.
- App Script Deployment: Writing a doGet() function in Google Apps Script to expose the sheet data via a public endpoint.
- HTML Page Design: Creating HTML files for each course, embedding placeholder tables.
- JavaScript Fetch: Writing an async fetch function to call the deployed endpoint and filter data based on the course name.
- Table Injection: Dynamically inserting filtered data into the table body using JavaScript DOM methods.

**II. Task Objective**

The goal was to:

- Build individual course schedule pages with their own layout and embedded calendar table.
- Ensure real-time updates by integrating the web pages with Google Sheets.
- Enable filtering based on course name so each page shows only relevant schedule entries.
- Ensure the solution was responsive and accessible.

**III. Development Breakdown**

GoogleSheet Setup:

The first step was to prepare a structured spreadsheet where each row corresponded to a course schedule entry.

Columns included:

- Course Name
- Start Date
- Batch Time

- Duration
- Mode (Online/Offline)
- Instructor Name

A unified sheet allowed centralized updates from the operations team.

**IV. Google Apps Script API Creation**

To expose the sheet data via an API:

- A doGet() function was written in Apps Script.
- The script used SpreadsheetApp.getActiveSpreadsheet() to access the data and convert it into JSON.
- Deployed the script as a web app with permissions set to "Anyone".

This allowed external frontend code to fetch the data securely.

## 2.5 Media Page – Playlist Re-editing and Thumbnail Enhancement

The Media Page Playlist Re-editing task was an essential component of ICSS's effort to maintain an engaging digital presence. As part of the internship, I was assigned to enhance the visual presentation of video playlists hosted under the Media Coverage section of the ICSS website. The project involved auditing, editing, and replacing thumbnails for an entire playlist embedded within the site, ensuring each thumbnail represented the corresponding video meaningfully.


**Fig. 2.7: Media Page Enhancement**

This screenshot highlights the re-edited and curated media playlist section featured on the

Indian Cyber Security Solutions (ICSS) website. The page showcases prominent interviews, including a televised appearance of Mr. Abhishek Mitra, CEO and Founder of ICSS, on Mirror Now, where he shared key insights into financial cybercrimes and cybersecurity challenges in India. As part of the internship task, this media section was refined to include engaging thumbnails, categorized video playlists, and restructured titles for improved user experience and visibility. This work significantly enhanced the presentation and credibility of ICSS's media presence.

**I. Objectives**

- Improve the media page's user experience by visually aligning thumbnails with video content.
- Re-edit existing thumbnail images for better resolution and clarity.
- Work within the media-coverage structure of the ICSS website under asset > saas > module > video-playlist-2.html.

Tools & Technologies Used:

1. HTML/CSS: Updating page structure and visuals
2. Git: Cloning and pushing updated repository
3. Screenshot Tool: Capturing thumbnails from videos
4. Photoshop / Online Editors (Canva): Editing and resizing thumbnails
5. XAMPP (Localhost): Previewing changes locally before deployment

**II. Task Breakdown**

1. Fetching the Repository: I began by cloning the Git repository that contains the static assets for the ICSS website. The specific path.

2. Downloading Videos & Capturing Snapshots:

Each video under Playlist 2 was downloaded or accessed via direct URL. I then:

- Played the video carefully to identify the most informative, visually appealing frame.
- Captured a screenshot using Snipping Tool or VLC's frame-capture feature.
- Saved images in consistent dimensions (e.g., 1280x720) to maintain layout consistency.

3. Thumbnail Editing:

Captured snapshots were cleaned and enhanced using online tools or software like:

- Canva (for minor touch-ups).
- Remove.bg
- TinyPNG

## 2.6 Form Validation and Error Handling on Career Page

The Career Page validation task was one of the most intellectually stimulating assignments during my internship at Indian Cyber Security Solutions (ICSS). It wasn't just about writing code—it was about anticipating user behavior, protecting organizational data integrity, and creating a smooth and professional user experience for every aspiring candidate who landed on the career section of the website.

I began by analyzing the form elements embedded in the HTML file named index.html, where the <form> tag encapsulated fields such as First Name, Last Name, Email, Last Qualification, Salary Expectation, and Skills. Each of these fields had their own unique validation requirements. After understanding the context and functionality, I wrote custom JavaScript logic that would validate user input in real-time and upon form submission.

These validations included:

- Ensuring First Name and Last Name do not contain any digits or special characters.
- Making sure Email addresses strictly follow the format username@domain.com.
- Preventing users from entering numbers in the Last Qualification field.
- Allowing only valid 2-digit numeric input for Salary Expectation.
- Disallowing numeric entries in the Skills input section.

To implement these checks, I employed a combination of regular expressions, string sanitization, and error messaging techniques. The system was designed to intercept invalid inputs immediately and provide users with contextual feedback next to the form fields. For instance, if a user typed numbers in their first name, a message would be displayed in red beneath the input, saying "Please enter a valid name without numbers.

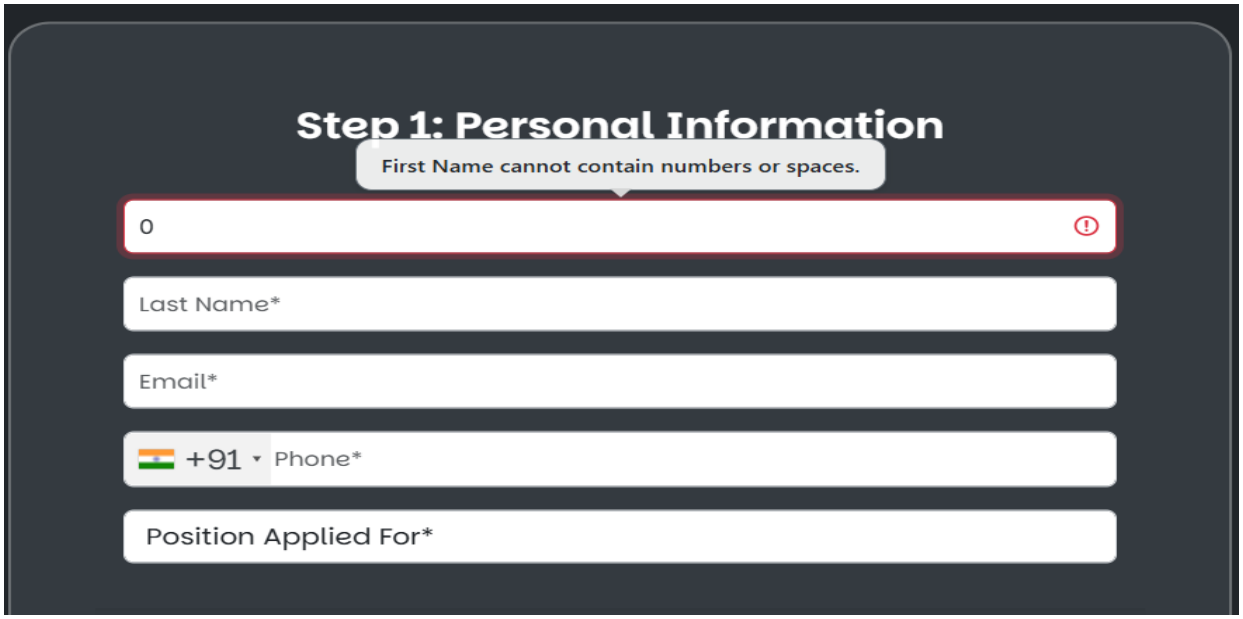


**Fig. 2.8: Frontend Form Validation – Input Constraints for Each Field**

This image demonstrates the implementation of real-time input validation in the application form developed during the internship. For example, the "First Name" field is restricted from accepting numerical values or spaces, enhancing data accuracy and user experience.

**I. Integration & Workflow Experience**

This task significantly enhanced my hands-on experience with DOM manipulation, event listeners, and client-side error handling. It also taught me how user-centric design decisions can improve overall system robustness. I utilized browser developer tools to inspect each form field's behavior across different browsers and devices, ensuring that the validation logic worked consistently regardless of platform.

**II. Real-time Feedback and Impact**

As a result of the form validation upgrade, the Career Page became far more user-friendly and professional. The instant feedback mechanism reduced confusion and enhanced trust in the system. Additionally, the HR backend received clean, formatted, and structured data, which made processing and sorting candidate applications significantly easier.

**III. Highlights of the Implementation (within narrative):**

- Strong input validation logic using JavaScript.
- Inline error prompts guiding users in real-time.
- Enhanced form reliability across multiple browsers.
- Optimized user experience and data quality simultaneously.

## 2.7 Brizy Pages Redesign – Course Page Modernization

As part of my internship at Indian Cyber Security Solutions (ICSS), I was entrusted with a

dynamic design-based task that involved modernizing several course pages hosted on Brizy Cloud, a drag-and-drop website builder. The objective of this assignment was to revamp the hero sections and course module layouts of existing pages to match the design sophistication of the CompTIA Network+ reference page.

This task was not simply an aesthetic upgrade. It was an exercise in consistency, user interface (UI) optimization, and maintaining brand integrity across multiple course offerings. The original pages, while informative, lacked modern structure, visual hierarchy, and interactive engagement. My role was to bridge the design gap while adhering to the brand's professional tone and information structure.


**Fig. 2.9: CompTIA Certification Page – Designed in Brizy Cloud**

This page was developed using the Brizy Cloud CMS during my internship to promote the CompTIA Certification courses by Indian Cyber Security Solutions. The design incorporates a modern and responsive layout, professional branding with partner logos, ratings, course fee details, and a clear call-to-action for enrolling in live classes. The top section features a gradient navigation bar, showcasing key affiliations with NASSCOM, DSCI, and ICC. The section below effectively communicates the value of the certification while ensuring clarity and readability for potential students. This project highlights my hands-on experience with Brizy Cloud for no-code web development and user-focused UI structuring.

**I. Key Improvements in the Hero Section**

- Improved Text Hierarchy: Rewritten headlines with better font sizes and spacing to enhance readability and focus.
- Visual Call-to-Actions (CTA): Buttons like "Enroll Now" or "View Modules" were

highlighted using branded color schemes and hover effects.

- Background Aesthetics: Added subtle overlays, gradients, or relevant course-related images to enhance background appeal without overwhelming content.

Following this, I worked on updating the Course Module section. The reference page used by ICSS, CompTIA Network+, had a modern block layout for modules. I replicated this format into all other related pages to ensure brand and UI uniformity. This included aligning course descriptions, pricing details, certification details, and module topics in a clean, grid-based layout.

**II. Design Consistency and Brand Identity**

Working with Brizy made me realize how crucial it is for every element of a corporate website to speak the same design language. Even small changes in font spacing, padding, or content alignment can cause inconsistencies that dilute the user's perception of the brand. By referencing the Network+ page, I ensured that the updated pages followed:

- A consistent font and color palette
- Uniform section paddings and margins
- Standard button styles and link placements

Throughout this task, I followed a checklist-driven approach where each modified page was reviewed against the reference design. This helped in avoiding overlook errors and ensured that all updates met the UI/UX expectations set by ICSS.

## 2.8 VAPT Page Design Enhancements

During my internship at Indian Cyber Security Solutions (ICSS), one of the most technically insightful assignments I undertook was the visual and structural enhancement of the VAPT (Vulnerability Assessment and Penetration Testing) page. This page serves as a key informational asset for the company, offering potential clients and cybersecurity learners a comprehensive look into the services offered under VAPT.

My contribution centered around optimizing the page layout, particularly refining the download card dimensions and aligning the video display ratios, ensuring they rendered perfectly across devices and screen resolutions.

The VAPT page initially suffered from inconsistencies in component proportions, especially in the download section, where multiple documents or reports were listed as downloadable cards.

The size of these cards often overflowed or misaligned on mobile and tablet views, making

the user experience cluttered and less engaging. My first task was to analyze the CSS controlling the card width and spacing, and identify how best to apply responsive design techniques without breaking the page structure. After inspecting the DOM tree and existing style rules, I carefully rewrote the card containers with flexbox properties and relative sizing units (like %, rem) instead of fixed pixels. This allowed the cards to adapt more fluidly to different screen widths.

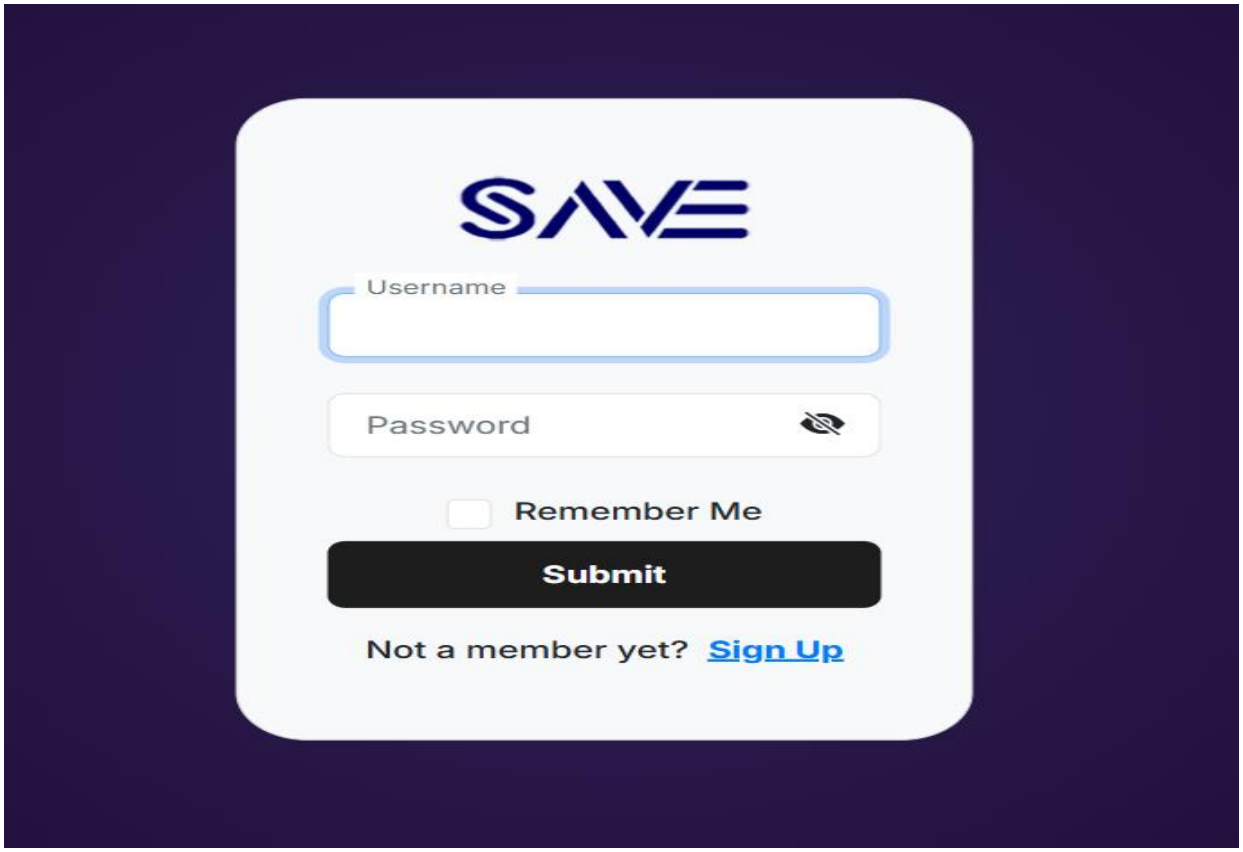


**Fig. 2.10: Secure Login Interface for VAPT Portal**

This login feature was designed as part of the Vulnerability Assessment and Penetration Testing (VAPT) section of the ICSS website. The minimalist and secure UI was built with user experience and security principles in mind. It includes essential fields such as Username and Password, along with a visibility toggle for password input and a "Remember Me" checkbox for session persistence.

**I. Design Enhancements Introduced**

- While the task description may appear technical, the impact it created was visibly evident. The revamped layout had:
- Responsive download cards that retained uniform spacing regardless of the device resolution.

- Improved readability, as elements were no longer overlapping or shrinking on smaller viewports.
- Clean video embeds sections that maintained consistent size and visual flow throughout the page.
- Optimized page load performance, as redundant CSS properties and styles were removed, reducing render time.

Each of these enhancements played a role in improving user engagement and professional appeal of one of ICSS's most critical technical pages. From a developer's point of view, I learned to view design not just from an aesthetic angle but also from a functional, performance, and user-centric perspective.

**II. Learning Reflections**

This task provided me with a real-world scenario where CSS optimization, media query planning, and DOM-based scripting come together to solve UI challenges. It also strengthened my problem-solving mindset as I had to balance between improving user interface design and maintaining compatibility with existing backend code. Additionally, I understood how even small UI inconsistencies could affect a company's credibility in the cybersecurity domain, where trust, precision, and clarity are paramount.

The experience improved my fluency in identifying layout issues, implementing programmatic calculations for dimensions, and applying best practices in responsive web design. I also gained confidence in communicating technical UI/UX updates to non-technical supervisors by visually presenting before/after versions of the page.

**Chapter 3**

# ADVANCED BACKEND PROJECTS

This chapter delves into the backend and functional aspects of my internship at Indian Cyber Security Solutions (ICSS). It focuses on advanced technical implementations, backend architecture, function modeling, database management, and full-stack interaction. These tasks required a deeper understanding of programming logic, data flow, and integration with third-party services and APIs. Through hands-on assignments involving technologies like Node.js, MongoDB, JavaScript, and Electron, I enhanced my backend development skills and gained exposure to real-world production environments. Each subchapter in this section reflects a detailed insight into the design, implementation, testing, and impact of each technical milestone I accomplished.

## 3.1 JavaScript Function Implementations

One of the most foundational and transformative experiences during my internship at Indian Cyber Security Solutions (ICSS) was gaining a strong, hands-on grasp of JavaScript's functional programming paradigms, particularly the application of asynchronous functions, callbacks, and their synchronous counterparts. Though this may seem theoretical on the surface, these concepts directly influenced how I approached real-time data fetching, user-triggered updates, and error handling during many of my frontend and backend assignments.

The task began with an explicit requirement to not only learn but also implement four different function types in JavaScript:

1. **Asynchronous Function**
2. **Asynchronous Callback Function**
3. **Non-Asynchronous Function**
4. **Non-Asynchronous Callback Function**

Each of these function types holds a unique place in the software development lifecycle. At ICSS, I realized that choosing the right function type in a given context determines both the performance and reliability of the application.

**I. Asynchronous Function**

```javascript
async function fetchCatFacts() {
    try {
        const response = await fetch('https://catfact.ninja/fact');
        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
        }
        const data = await response.json();
        console.log("Random Cat Fact:", data.fact);
    } catch (error) {
        console.error("Error fetching cat fact:", error.message);
    }
}
fetchCatFacts();
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

PS C:\Icss-projects\dev-intern\purva> cd tasks
PS C:\Icss-projects\dev-intern\purva\tasks> cd .\nodejs-func-implementation\
PS C:\Icss-projects\dev-intern\purva\tasks\nodejs-func-implementation> node .\async-callback.js
Random Color:
Name: Cerulean
Hex: #24B1E0
RGB: rgb(36, 177, 224)
PS C:\Icss-projects\dev-intern\purva\tasks\nodejs-func-implementation>
```

**Fig. 3.1: Asynchronous Function Implementation**

The Asynchronous Function showcased in this implementation utilizes the async/await syntax in JavaScript to fetch a random cat fact from an external API. The fetchCatFacts() function begins with a try-catch block to handle potential runtime errors and ensure robustness. By awaiting the API call and parsing the JSON response, the code ensures non-blocking behavior while still maintaining readable and synchronous-like flow.

### II. Asynchronous Callback Function

```javascript
function fetchRandomColor(callback) {
    setTimeout(async () => {
        try {
            const response = await fetch('https://www.thecolorapi.com/id?hex=24B1E0');
            if (!response.ok) {
                throw new Error(`HTTP error! status: ${response.status}`);
            }
            const data = await response.json();
            callback(data);
        } catch (error) {
            console.error("Error fetching color data:", error.message);
        }
    }, 1000);
}

fetchRandomColor((colorData) => {
    console.log("Random Color:");
    console.log(`Name: ${colorData.name.value}`);
    console.log(`Hex: ${colorData.hex.value}`);
    console.log(`RGB: ${colorData.rgb.value}`);
});
```

```
PS C:\Icss-projects\dev-intern\purva\tasks\nodejs-func-implementation> node .\async-function.js
Random Cat Fact: Ailurophile is the word cat lovers are officially called.
PS C:\Icss-projects\dev-intern\purva\tasks\nodejs-func-implementation>
```

**Fig. 3.2: Asynchronous Callback Function Implementation**

This function highlights a more traditional asynchronous programming pattern using callbacks. The fetchRandomColor() function leverages setTimeout() to simulate an asynchronous environment and await for fetching random color data. Upon successful data retrieval, the callback function is invoked with the color data, showcasing how JavaScript handles I/O without blocking the main thread.

### III. Non-Asynchronous Function

```
function getCountries() {
    return [
        { name: "India", capital: "New Delhi" },
        { name: "USA", capital: "Washington, D.C." },
        { name: "Canada", capital: "Ottawa" }
    ];
}

const countries = getCountries();
countries.forEach((country) => {
    console.log(`Country: ${country.name}, Capital: ${country.capital}`);
});
PS C:\Icss-projects\dev-intern\purva\tasks\nodejs-func-implementation> node .\nonasync-function.js
Country: India, Capital: New Delhi
Country: USA, Capital: Washington, D.C.
Country: Canada, Capital: Ottawa
```

**Fig. 3.3: Non-Asynchronous Function Implementation**

In this implementation, the getCountries() function returns a static array of country objects, representing a simple and synchronous data retrieval example. The code then uses forEach() to iterate over each country and print its name and capital to the console.

### IV. Non-Asynchronous Callback Function

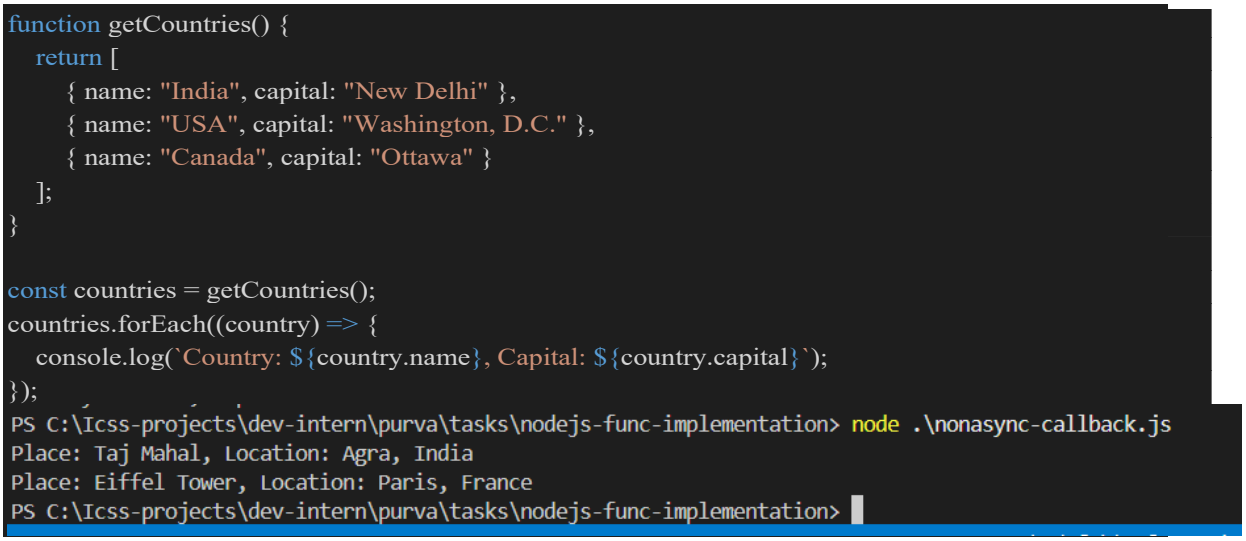


**Fig. 3.4: Non-Asynchronous Callback Function Implementation**

The final function displays the use of callbacks in a completely synchronous environment. Although callbacks are generally associated with asynchronous tasks, this example demonstrates how they can be used even when operations are immediate. The function returns a predefined list of places, and a callback is used to print each place's name and location.

**I. Descriptive Breakdown of Function Types**

1. Async Function Using async/await

- An async function returns a Promise.
- Easy to manage using try...catch blocks.
- Control flow remains sequential, which helps in managing readable logic.

2. Async Function Using Callback

- Often used in older versions of Node.js.
- Offers a non-blocking architecture but suffers from callback hell in complex scenarios.
- Requires robust error-first callback structure (e.g., callback (err, result)).

3. Non-Async Function

- Executes line-by-line without any delay.
- Ideal for immediate operations but blocks the event loop for long-running tasks.

4. Non-Async Function with Callback

- Used in scenarios where no external event (like file I/O or network call) is needed,

but abstraction is helpful.

- Good for modular programming and reuse.

This hands-on coding process not only deepened my understanding of the JavaScript event loop, call stack, and callback queue, but also improved my ability to debug performance issues.

**II. Practical Applications During Internship**

The knowledge gained here proved essential in later parts of my internship where I handled tasks like:

- Creating dynamic forms that validate user inputs in real time.
- Implementing loading spinners during file processing and form submissions.
- Designing client-server interaction patterns using Node.js, where response delays needed to be handled gracefully.

For instance, while integrating the media upload system with Google Drive, I used asynchronous functions to handle the token authorization and file upload calls, ensuring the UI remained responsive throughout. Similarly, callback functions were used during DOM manipulation for form validation, where each validation logic depended on the previous input's state.

**III. Learning Highlights and Takeaways**

This task helped me:

- Grasp the fundamentals of asynchronous behavior in JavaScript especially Promise, async/await, and then/catch chaining.
- Understand when and how to structure non-blocking functions, especially for long-running or I/O-bound tasks.
- Enhance the robustness of frontend components, preventing UI freezes and improving user experience.

## 3.2 Log Management System for Sign In, Sign Up & MFA

Effective log management is a critical aspect of maintaining the security and integrity of any application. During my internship at ICSS, I was assigned a task to build a log management system to monitor and record key user events such as Sign In, Sign Up, and Multi-Factor Authentication (MFA). This involved creating structured log files, capturing metadata like timestamps and public IP addresses, and organizing the data in a retrievable and readable format specifically in JSON. The purpose was to ensure traceability, enable post-event audits, and support incident response.

**I. Task Overview**

The logging system was built around three core functionalities:

- SignIn Logs
- SignUp Logs
- MFA Logs

Each function had to maintain its dedicated log file, where entries for the last 7 days were to be recorded dynamically.

**II. Objectives**

- Create an individual log file for each of the three events.
- Record and save the last 7 days of logs.
- Use JSON format to structure each log entry for clarity and parsing.
- Ensure that metadata like timestamps and public IPs are included for auditing.
- Automate the rotation or cleanup of old logs beyond the 7-day window (optional extension).

**III. Implementation Details**

- File Creation: Three separates .Json files were created signin_logs.json, signup_logs.json, and mfa_logs.json.
- Dynamic Writing: Each time a user event occurred, a new log entry was appended using asynchronous file writing in Node.js.
- Time-based Filtering: To ensure only the last 7 days' data was retained, a date-checking mechanism was used before adding logs to the file.
- Security Measures: Sensitive data was excluded from logs. Only event type, IP, and timestamp were retained to prevent accidental leakage of credentials or user information.

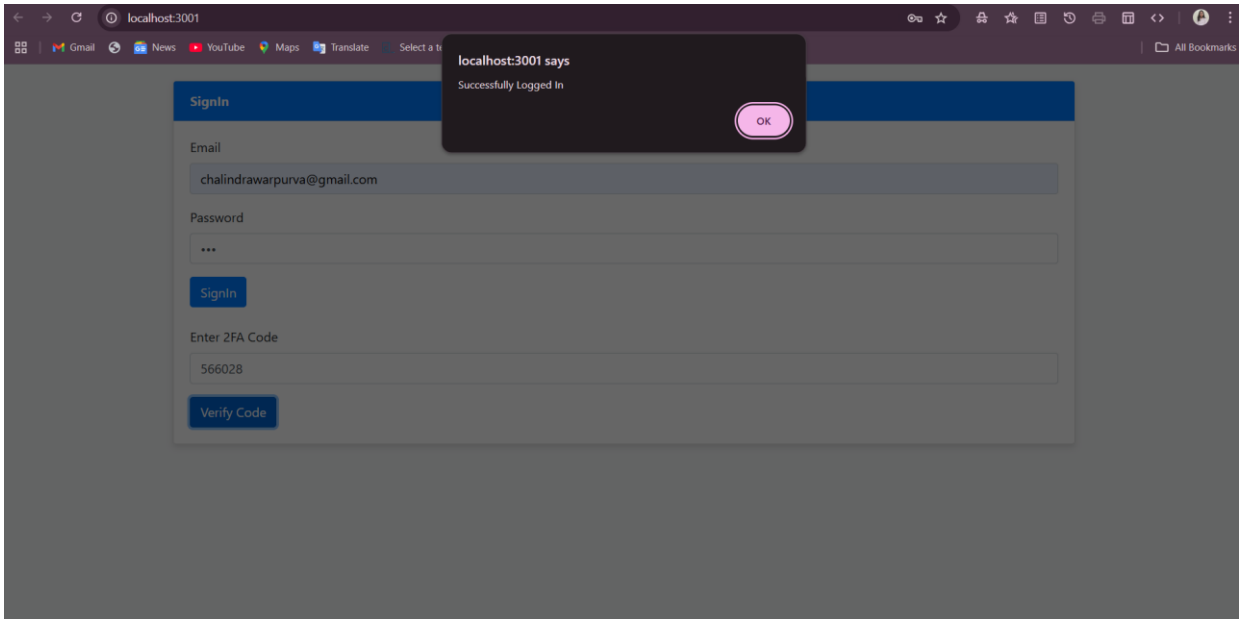


**Fig. 3.5: Secure 2FA-Based Sign-In System Implementation**

The above interface represents the successful login phase of a Two-Factor Authentication (2FA) system implemented during the internship. After entering a valid email and password, the user is prompted to input a 2FA code sent via email, ensuring a second layer of authentication. Upon correct verification of the OTP, the system displays a confirmation popup stating "Successfully Logged In." This project demonstrates the integration of front-end and back-end logic to enhance application security, safeguard user credentials, and prevent unauthorized access through email-based OTP validation.

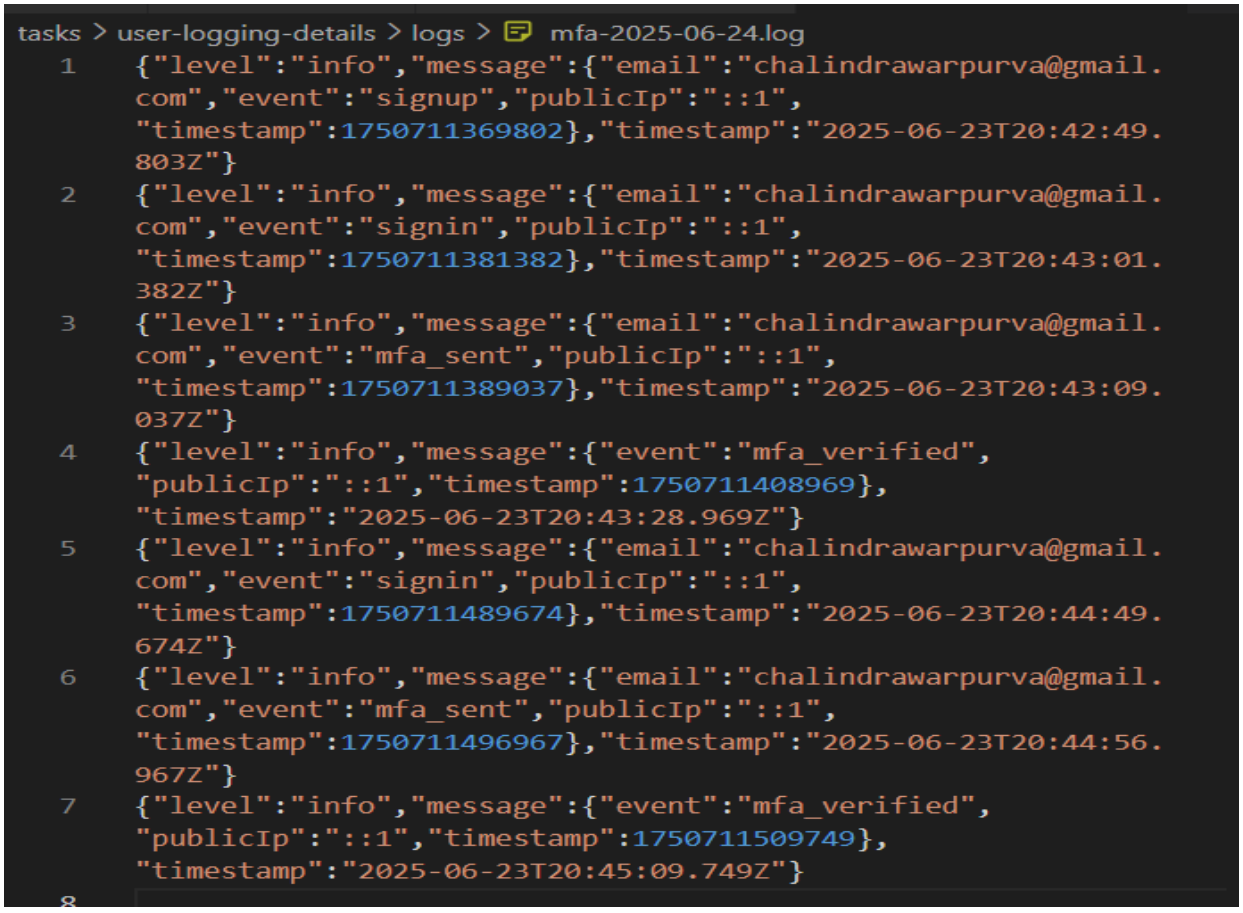


**Fig. 3.6: User Logging and Event Tracking System**

The above image demonstrates the server-side log tracking system designed for capturing user authentication events in real-time. Each log entry is stored in JSON format, documenting key actions such as signup, signin, mfa_sent, and mfa_verified along with the user's email, public IP address, and a precise timestamp. This logging mechanism is essential for auditing user activity, diagnosing issues, and enhancing system security. By storing this information in structured log files like signup-2025-06-24.log, the application ensures traceability and transparency of user actions while supporting compliance and forensic analysis.

**IV. Key Learnings**

- Importance of logging in enterprise-grade systems.
- Working with asynchronous file operations in Node.js.
- Designing lightweight logging mechanisms without relying on external logging frameworks.
- Understanding how logs support monitoring, troubleshooting, and forensic investigations in cybersecurity environments.

## 3.3 MongoDB Architecture – Research and Analysis

Understanding the internal architecture of a database system is essential for building scalable and efficient backend systems. During my internship at Indian Cyber Security Solutions (ICSS), I was given the task to explore and document the MongoDB database architecture in detail. MongoDB is a widely-used NoSQL database that offers high flexibility, horizontal scalability, and powerful query capabilities, making it a go-to solution for modern applications requiring agility and performance.

**I. Overview of MongoDB**

MongoDB is a document-oriented database that stores data in a flexible, JSON-like format called BSON (Binary JSON). It allows developers to store complex nested data structures while avoiding rigid schemas found in traditional relational databases. MongoDB is designed to manage large volumes of unstructured or semi-structured data efficiently, and its architecture supports features such as:

- Horizontal scalability using sharding
- High availability via replication sets
- Fast read/write operations through memory mapping and indexing
- Dynamic schemas allowing flexible application development

**II. Why MongoDB Was Studied at ICSS**

At ICSS, MongoDB is used for building scalable applications with dynamic data requirements. It is particularly helpful in:

- Logging user activity and events
- Handling large volumes of data from internal tools
- Creating flexible structures for form submissions, course data, and user profiles
- Managing backend systems that integrate with JavaScript and Node.js

**III. Research Methodology**

I conducted my research through:

- Official MongoDB documentation
- YouTube tutorials (MongoDB Architecture - Full Course)
- Real-world examples provided during backend training at ICSS
- Simulated queries and operations using MongoDB Compass and Mongo Shell

**IV. Challenges & Learnings**

- Initially understanding replica sets and election processes was complex.
- Implementing sharding in a local development environment was simulated.

- Learned the importance of index optimization to prevent slow queries.
- Understood practical use cases where MongoDB is more efficient than RDBMS.

## 3.4 Student Database Update Using phpMyAdmin and CSV Import

As part of the backend tasks during my internship at Indian Cyber Security Solutions (ICSS), I was assigned the responsibility of updating and managing student data through a structured database. The specific task involved creating a database in phpMyAdmin, importing student certificate records from a CSV file, and performing operations that simulate real-time backend data management scenarios. This task was crucial in understanding how structured data is handled in a production environment using MySQL and phpMyAdmin, a widely-used web-based database management interface.

**I. Objective of the Task**

The primary goal was to:

- Create a new database titled student_db.
- Define and implement a schema for storing certificate-related information.
- Import a preformatted .csv file containing student certificate data.
- Perform basic validation and ensure that data was properly aligned and clean.
- Export the updated database for deployment or archival purposes.

**II. Detailed Task Execution**

**Step 1: Creating the Database**

Using the phpMyAdmin interface provided by the XAMPP environment:

- Navigated to localhost/phpMyAdmin.
- Clicked on New to create a database.
- Named the database as student_db.
- Set the collation to utf8_general_ci for broader compatibility.

**Step 2: Creating the Table**

Inside the student_db:

- Created a new table named certificate_record.
- Defined appropriate fields including:
    - student_id (Primary Key, INT, AUTO_INCREMENT)
    - name (VARCHAR)
    - email (VARCHAR)
    - course_name (VARCHAR)
    - date_of_completion (DATE)

- certificate_url (TEXT)

This schema was designed to reflect each student's certificate record, which is essential for verification and audit purposes.

**Step 3: Importing Data from CSV**

- Clicked on the Import tab inside the certificate_record table.
- Selected the CSV file provided by the supervisor.
- Ensured the format selected was CSV and the field terminator was a comma (,).
- Mapped the column order of the CSV to the table schema carefully.
- Clicked **Go**, and the data was imported successfully.

**Step 4: Verifying and Validating Data**

- Used SQL queries to confirm that all rows were imported:
  - SELECT * FROM certificate_record;
- Checked for any formatting issues, null values, or misaligned data entries.
- Made small corrections directly via the phpMyAdmin UI where necessary.

**Step 5: Exporting the Updated Database**

- After validation, exported the complete database using the Export tab.
- Chose the format as .sql for future re-import and backup.
- Ensured it was ready for deployment on live servers or archival purposes.

**III. Skills and Concepts Applied**

1. Relational Schema Design: Created efficient data structures for storing certificate details.
2. Data Import: Mapped CSV data to corresponding SQL table columns accurately.
3. Data Validation: Validated integrity and format of imported data using queries and filters.
4. phpMyAdmin: Used an industry-standard GUI for managing MySQL databases.
5. SQL Queries: Executed SQL commands to ensure data correctness and update functionality.

**IV. Key Learnings**

- Practical Use of phpMyAdmin: Learned how to manage MySQL operations without needing command-line SQL.
- Structured Data Management: Understood the role of schemas in maintaining data consistency.
- Handling Real Data: Real-world CSV imports come with formatting inconsistencies—this improved my debugging approach.

- Data Portability: Importance of .sql exports for moving databases across systems or servers.

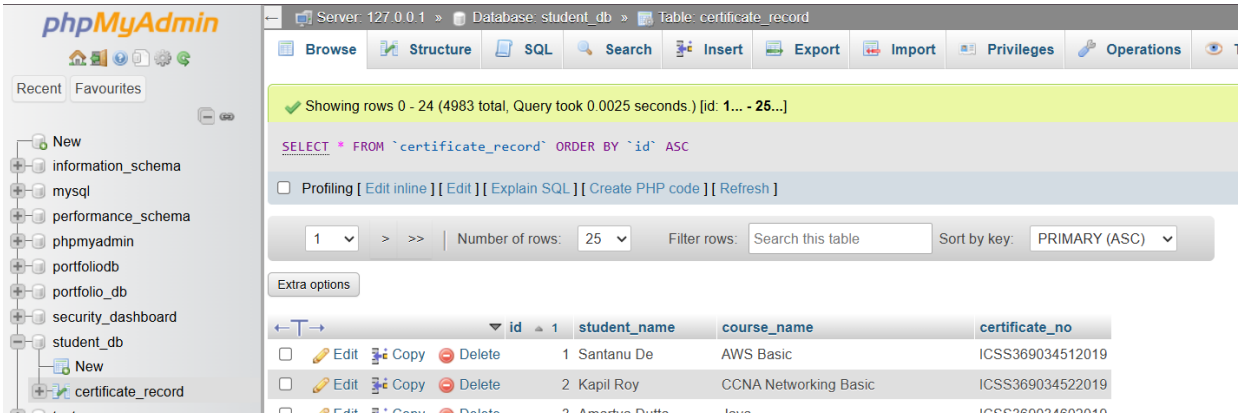


**Fig. 3.7: phpMyAdmin dashboard**

The image shows task involved creating a database in phpMyAdmin, importing student certificate records from a CSV file, and performing operations that simulate real-time backend data management scenarios. This task was crucial in understanding how structured data is handled in a production environment using MySQL and phpMyAdmin, a widely-used web-based database management interface.

## 3.5 Uploading Media Files to Google Drive using JavaScript

In a digital-first organization like Indian Cyber Security Solutions (ICSS), efficient cloud storage and automation workflows are crucial to maintaining a scalable, secure, and accessible media infrastructure. One of the key backend tasks during my internship involved implementing a cloud integration feature that would allow media files to be uploaded directly to Google Drive using JavaScript.

The goal of this task was to simulate a real-world cloud solution where a system can authenticate with a user's Google account and upload files using the Google Drive API v3. This functionality reflects how enterprise systems interact with third-party cloud platforms using OAuth2, ensuring security and flexibility. It was a hands-on learning experience in cloud API configuration, authorization flows, file manipulation, and asynchronous JavaScript handling.

### I. Task Objective

The major goals of this task were:

- Integrate the Google Drive API with a JavaScript frontend.
- Authenticate the user using OAuth 2.0 credentials.
- Allow file selection from the local machine.

- Upload the selected file directly to Google Drive.
- Maintain a clean and interactive UI for user operations.
- Document the integration steps for future reuse or deployment.

**II. Implementation Process**

**Step 1: Google Cloud Console Setup**

- Logged in to the Google Cloud Console.
- Created a new project titled ICSS-Media Uploader.
- Enabled Google Drive API v3.
- Navigated to “Credentials” and created OAuth 2.0 Client ID credentials.
    - Set application type: Web
    - Authorized redirect URIs and JavaScript origins.
- Downloaded the credentials. Json containing the client ID and client secret.

**Step 2: Preparing the HTML and JS Files**

- Built a basic HTML file with the following components:
    - A “Sign in with Google” button.
    - A file input element to browse media files.
    - An “Upload” button to trigger the upload action.

**Step 3: Authentication and Token Handling**

- Used JavaScript to implement Google OAuth 2.0 authentication.
- Once the user logged in, the access token was stored securely for the session.
- This token was then used to authorize the Drive API requests.

**Step 4: Uploading to Google Drive**

- After selecting the file, the upload process was initiated using a POST request to:
- Attached headers with the bearer token and file metadata (title, MIME type).
- Displayed success or error message to the user based on API response.

**Step 5: Documentation**

- Compiled a detailed guide on:
    - Setting up Google Cloud credentials.
    - OAuth flow using JavaScript.
    - Creating upload logic.
    - Troubleshooting permission errors.

**III. Skills and Technologies Applied**

- Google OAuth2 Authentication: Secured login and access token handling.

- JavaScript Asynchronous Flow: Used async/await and fetch () for Drive API interactions.
- Cloud Storage API: Integrated Google Drive API v3 for file upload.
- UI and UX Design: Designed a user-friendly interface for selecting and uploading files.
- Documentation: Authored a clear and structured implementation guide.

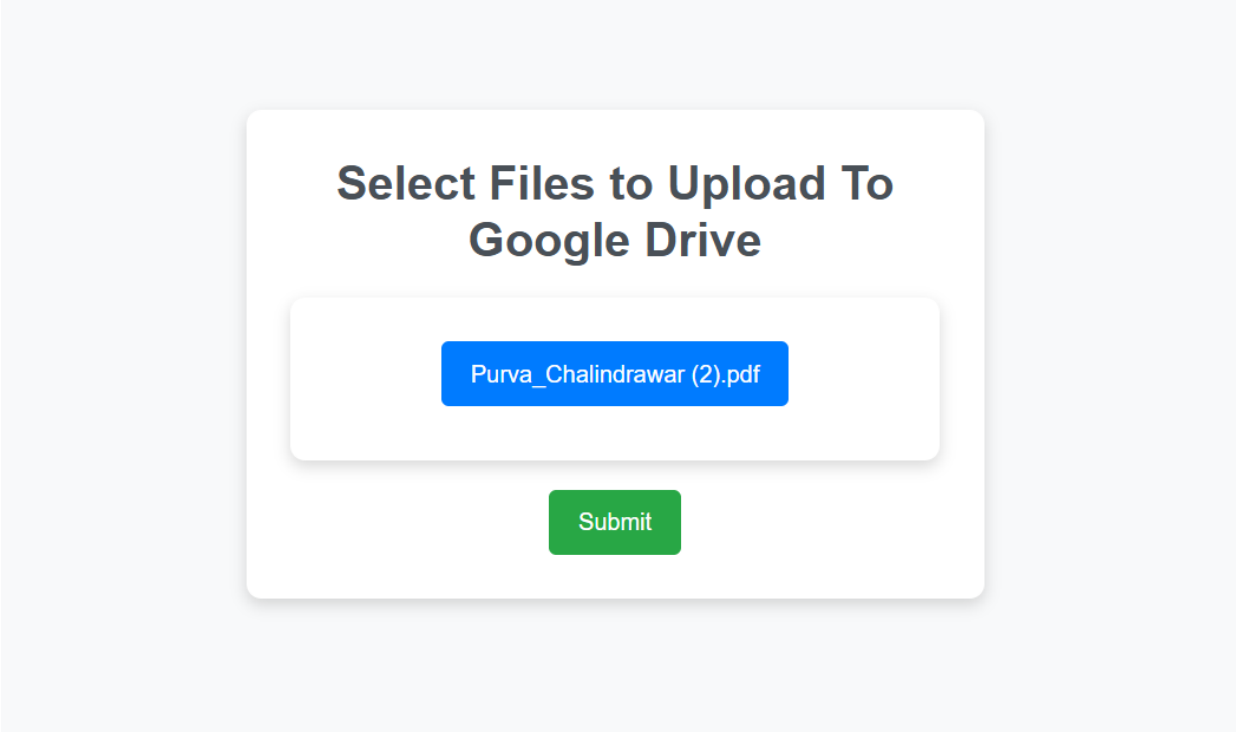


**Fig. 3.8: HTML frontend interface**

During the development process, I focused on **UI and UX design** to create a seamless and intuitive user experience for the file selection and upload functionality. The interface was carefully structured to be user-friendly, ensuring that users could easily navigate through the upload process without confusion. I incorporated clear visual cues, responsive design elements, and accessibility considerations to enhance usability.

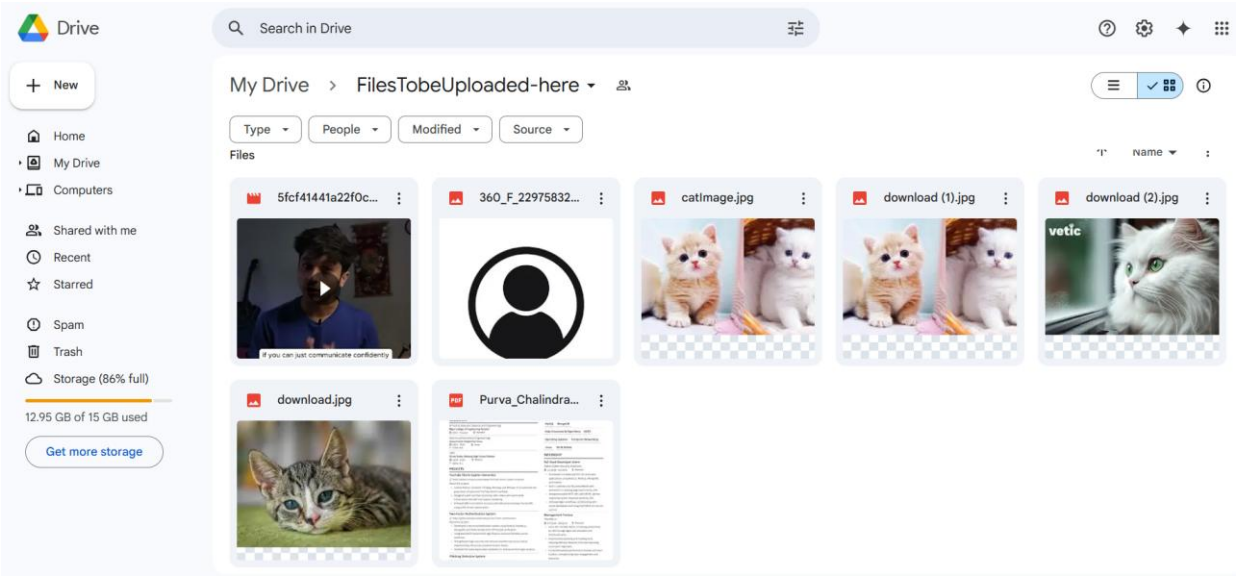


**Fig. 3.9: Showing successful upload**

The interface was carefully structured to be user-friendly, ensuring that users could easily navigate through the upload process without confusion. I incorporated clear visual cues, responsive design elements, and accessibility considerations to enhance usability.

**IV. Key Learnings**

- Gained hands-on experience with Google Drive API v3 and how real-world applications authenticate and interact with external services.
- Learned to implement secure authentication flows using OAuth 2.0 in frontend JavaScript apps.
- Understood how to manage file metadata and MIME types while sending data to APIs.
- Improved debugging skills while resolving token expiry issues, incorrect scopes, and CORS configurations.

## 3.6 Electron App Development

Electron.js is a powerful open-source framework that enables the development of cross-platform desktop applications using web technologies such as HTML, CSS, and JavaScript. During my internship at Indian Cyber Security Solutions (ICSS), one of the most technically enriching assignments was to develop a multi-page desktop application using Electron, simulating a real-world productivity tool. This task sharpened my understanding of GUI-based architecture, inter-process communication, application packaging, and runtime execution on the desktop.

The core aim of this project was to build a desktop app that seamlessly integrates web UI

design with backend JavaScript logic, representing a shift from browser-based applications to native software environments. This was my first practical exposure to desktop-level JavaScript applications that are installable and platform-independent.

**I. Project Requirements**

The multi-page Electron application consisted of the following:

- A Login Page to simulate user authentication.
- A Progress Page to track user operations or project loading.
- A Submission/Error Page to show status feedback to users.
- Fixed window dimensions of 800x640 pixels.
- Seamless navigation between pages without reloading.
- A consistent design flow, focusing on usability and real-time feedback.

**II. Development Process**

**1. Project Initialization**

- Initialized a new Electron project using npm init.
- Installed Electron as a dev dependency:
- Configured main.js as the entry point for the app and defined the window creation logic using Electron's BrowserWindow module.

**2. UI Structure and Pages**

- Created three HTML files:
    - login.html
    - progress.html
    - submission.html
- Designed each page with a unique layout but kept a consistent branding theme (ICSS colors, logos, and fonts).
- Used CSS animations and JavaScript to handle form submissions and transitions.

**3. Window Creation and Navigation**

- Used Electron's loadFile() method to switch between pages based on user actions.
- Maintained a single BrowserWindow instance, dynamically loading content.

**4. Logic Implementation**

- Login Page: Verified hardcoded credentials (simulating backend validation).
- Progress Page: Displayed a loading spinner or bar to indicate activity.
- Submission/Error Page: Presented feedback based on form input or execution result.

**5. Application Packaging**

- Used Electron Forge to bundle the application into an installable .exe for Windows.
- Verified runtime performance on different desktop setups.

**III. Skills and Technologies Applied**

1. Electron.js: Created a cross-platform desktop application.
2. HTML/CSS/JS: Built the frontend interface for all application screens.
3. Node.js Integration: Used Node modules to control Electron windows.
4. UI/UX Design: Designed smooth and intuitive user interfaces.
5. Application Packaging: Packaged app using Electron Forge for production use.

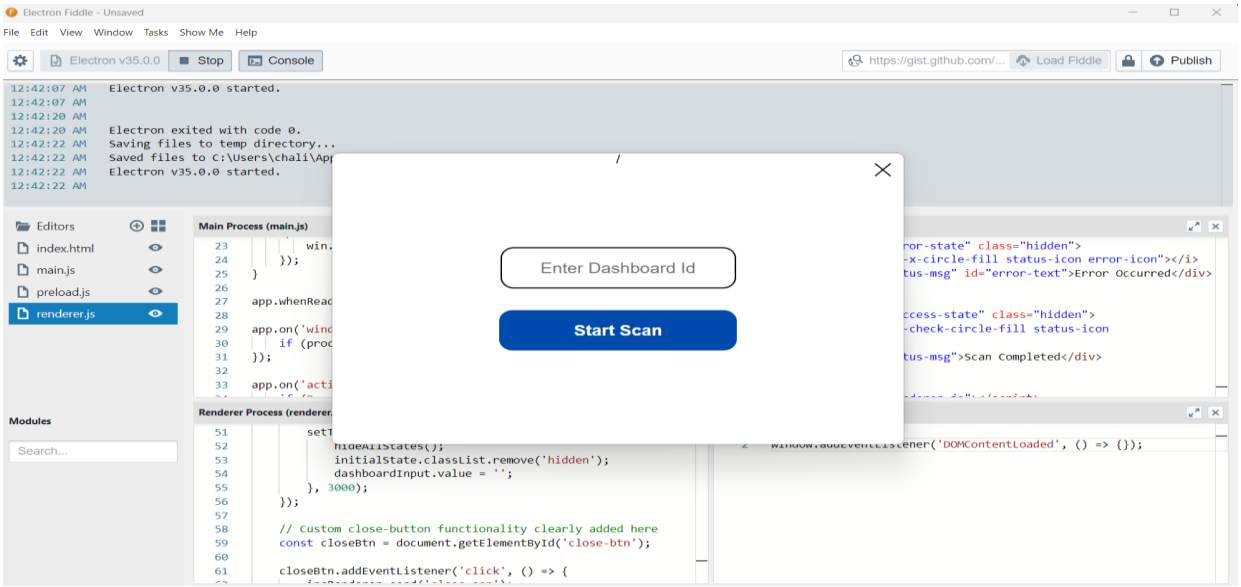


**Fig. 3.10: Electron Fiddle app page**

The basic desktop application showcased was developed using Electron Fiddle, a rapid prototyping environment for Electron apps. The interface includes a minimalist and centered modal that prompts the user to enter a Dashboard ID, followed by a Start Scan button. Upon user input, the app triggers a validation process and dynamically transitions to a scan completion screen.

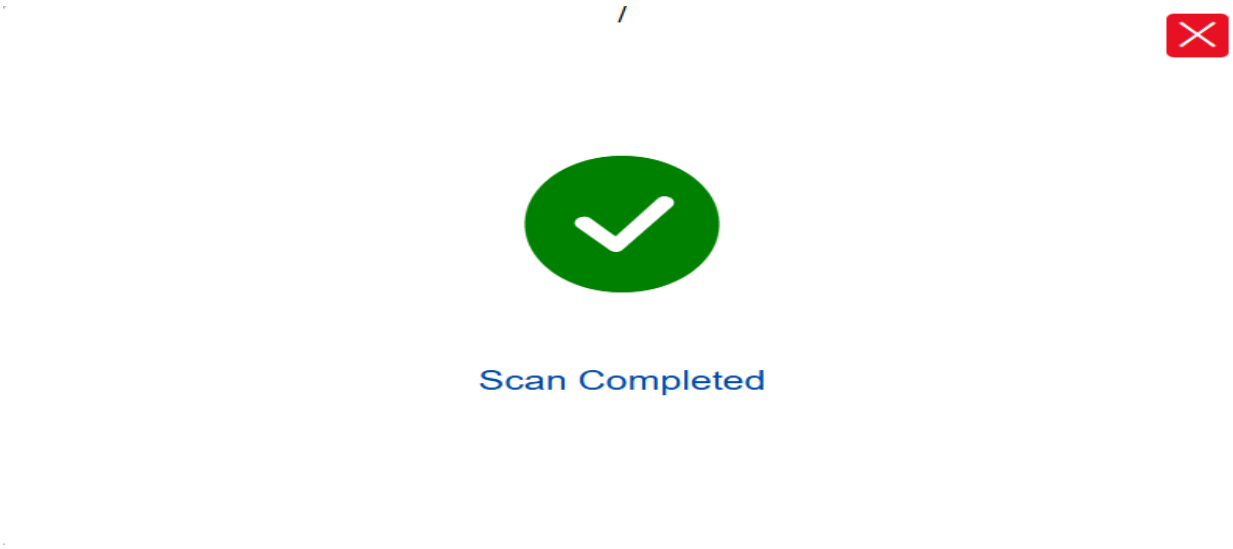


**Fig. 3.11: Electron Fiddle app page**

The above image confirms successful execution with a "Scan Completed" message accompanied by a green check icon for visual feedback. Behind the scenes, the app integrates with a custom renderer process (renderer.js) and utilizes inter-process communication for input handling and UI state updates.

**Key Learnings**

- Understood the architecture of Electron apps and how web technologies can be extended to desktop environments.
- Learned how to manage navigation and data flow across different views in a non-browser context.
- Gained experience in packaging and deploying desktop apps, a crucial skill for enterprise software development.
- Developed real-time problem-solving capabilities while handling UI responsiveness, app size constraints, and file system access.

## 3.7 Integration of C Program with Electron via Node.js Execution

Integrating a C program with an Electron-based JavaScript application is a powerful demonstration of cross-language interoperability. This task at ICSS required combining low-level systems programming (C) with a modern desktop UI framework (Electron), simulating a cybersecurity scanning module embedded inside a user-friendly desktop interface. It emphasized my capability to build real-world hybrid solutions where performance-critical components (written in C) are triggered and managed by higher-level JavaScript code.

**I. Project Overview**

The project was designed to simulate a dashboard-based vulnerability scanner, where upon entering a valid dashboard ID, the system runs a compiled C program (agentScript.exe). This scanner executes security checks and then reports the output back to the UI in the form of either a success or failure message.

This section not only tested my proficiency with Electron and Node.js, but also introduced me to child process handling, event-driven programming, and integration of native binaries within a JavaScript environment.

**II. Implementation Breakdown**

**1. C Program Compilation**

- I was provided with a C file named agentScript.c, which needed to be compiled using MinGW into an executable .exe file for Windows.
- Compilation was done using:
    - gcc agentScript.c -o agentScript.exe
- This program performed basic operations such as logging, scanning dummy inputs, and outputting a success or error message based on simulated conditions.

**2. Creating the Integration Trigger**

- Inside the Electron app, particularly in dashboard.js, a Dashboard ID (1-9) was required to enable the scan button.
- When triggered, the JavaScript child_process module was used to spawn the agentScript.exe executable:

**3. Real-Time User Feedback**

- While the C program was running, a loading spinner (implemented via CSS and JavaScript) was shown to indicate processing.
- After execution:
    - On success: A confirmation screen with a success badge appeared.
    - On failure: A red error message with possible causes was displayed.

**4. Error Handling and UI Enhancement**

- Proper error messages were configured in case:
    - The executable was missing.
    - Invalid dashboard IDs were entered.
    - The C program exited with a non-zero code.

**Chapter 4**

# SYSTEM DESIGN AND DEPLOYMENT

The final stage of my internship at Indian Cyber Security Solutions (ICSS) was a culmination of both creative and technical responsibilities. During this period, I transitioned from web development assignments into more structured system design tasks and contributed to live codebases for production-level React applications. Additionally, I concluded my tenure by working on personal portfolio development and deployment, showcasing the real-world applications of my learnings.

This chapter encapsulates my contributions towards the design and modeling of the SAVE (Secure Assessment & Vulnerability Evaluation) project, enhancements to the ICSS website's React frontend, and the creation of my personal portfolio site deployed using Vercel. Each of these tasks emphasized industry-level standards, offering insights into scalability, UI responsiveness, data flow, and deployment practices.

## 4.1 Data Flow Diagram (DFD) Modeling for SAVE Project

One of the major highlights of this stage was the opportunity to work on system modeling through **A. Data Flow Diagrams (DFDs)**

These diagrams play a crucial role in the software development lifecycle by providing a visual representation of data movement within a system. Through various levels of abstraction, I captured the core processes, data stores, and external interactions that defined the architecture of the SAVE platform. The experience of modeling DFDs helped me build clarity in system thinking, understand modular workflows, and align development with security considerations.

At Level 0, the SAVE system is viewed as a single unified entity interacting with external components such as the User, Administrator, and Database. This top-level diagram presented a bird's-eye view of the system, depicting inputs such as scan requests and credentials, and outputs like vulnerability reports and authentication tokens. This stage allowed me to understand the external data exchanges and boundaries of the application.

The Level 1 DFD went a step deeper by decomposing the system into major subsystems—namely, the Authentication Module, Web Scanning Engine, Network Scanning Engine, and Reporting Unit. Each of these subsystems was connected to its own set of data inputs and processing flows. For example, the Authentication Module received credentials, validated them through internal logic, and returned verification status to the frontend. The Web and

Network Scanning Engines operated in parallel to analyze user-submitted URLs or IP addresses and return vulnerability insights.

This level helped establish a working blueprint of the SAVE platform's internal logic.

In DFD Level 2, I chose to deep dive into the Web Scanning Engine. This was further broken down into sub-processes such as input handler, threat detection algorithm, output formatter, and sanitization process. Mapping this level helped in identifying error-prone regions and areas that could be modularized further. It was here that I began to appreciate the depth of planning required to build secure and efficient modules, particularly in the domain of web vulnerability analysis.

DFD Level 3 took system modelling to a micro level. I chose to represent the Authentication Module in this level, showcasing minute details such as session token generation, OTP verification logic, retry limits, error logging, and session expiry workflows. This helped formalize security mechanisms, log integrity checkpoints, and token validation procedures. The transition from high-level DFDs to Level 3 illustrated the importance of granular system breakdowns for implementation and testing.

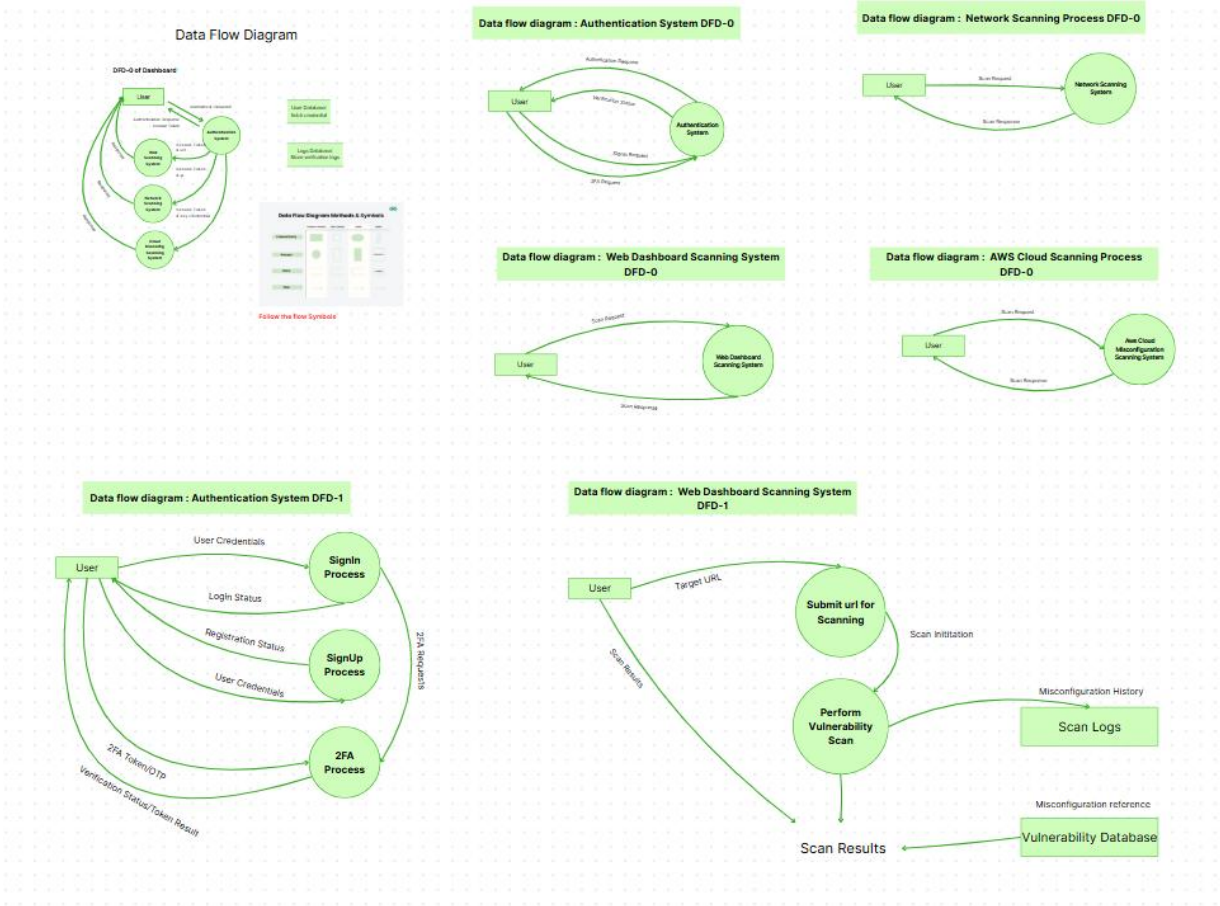


**Fig. 4.1: Multi-Level Data Flow Diagrams for Web-Based Scanning Dashboard**

The above figure presents a comprehensive set of Data Flow Diagrams (DFDs)

representing the system architecture and data movement within a secure web-based scanning dashboard. It includes both Level 0 (DFD-0) and Level 1 (DFD-1) diagrams for various subsystems—Authentication, Web Scanning, Network Scanning, and AWS Cloud Scanning. Each DFD illustrates how users interact with processes, such as sign-in, 2FA, scan initiation, and vulnerability detection, while showing the flow of data between entities, processes, and data stores. The diagrams also define the input/output relationships for components like the Vulnerability Database and Scan Logs, ensuring clarity in system behavior and interaction. These visual models serve as foundational artifacts for both system design and stakeholder understanding.

Creating these DFDs required not just technical knowledge but also careful attention to how user data flows through different parts of a platform. It was a valuable exercise in both documentation and design logic, and it allowed me to meaningfully contribute to one of ICSS'score security platforms.

## 4.2 Contribution to ICSS Official Website using React.js

In addition to architectural design, I was also involved in real-time frontend development work on ICSS's official website. This project was particularly important as it required me to apply my skills in React.js, a modern JavaScript library widely used for developing interactive web interfaces. I collaborated with the core development team to improve the responsiveness, structure, and usability of various course pages and application modules.

One of my primary contributions was the modularization of static components into reusable React components. This involved migrating raw HTML and JavaScript elements into a component-based framework, enabling better maintainability and faster rendering. I also worked on implementing React state management to dynamically update data such as course schedules, session details, and enrolment forms. The integration of props and use State allowed us to control rendering logic based on user inputs or API responses.

Moreover, I played a key role in enhancing the responsiveness of the application. Tailwind CSS was integrated alongside React to maintain design consistency across devices. I learned how to use conditional class rendering and media queries to deliver a seamless user experience on both mobile and desktop screens. Another significant enhancement I implemented was React Router, which allowed us to structure the application into different logical pages without requiring a full reload improving the performance and user interaction flow significantly

The work on the ICSS website not only sharpened my technical skills but also gave me

insight into collaborative coding practices. I used Git extensively to push updates, manage pull requests, and participate in version control workflows with my team. My time working with React in a production-grade setting gave me confidence in contributing to real-world applications that are visible to a large number of users.

## 4.3 Development and Deployment of My Personal Portfolio

To conclude the internship experience, I decided to consolidate all the knowledge I gained into a personal portfolio website, which acts as both a technical showcase and a career milestone. This project involved building an interactive, visually appealing, and informative website using core web technologies—HTML, Tailwind CSS, and vanilla JavaScript. The portfolio contains detailed sections covering my education, skills, internship experience, projects, certifications, and contact information.

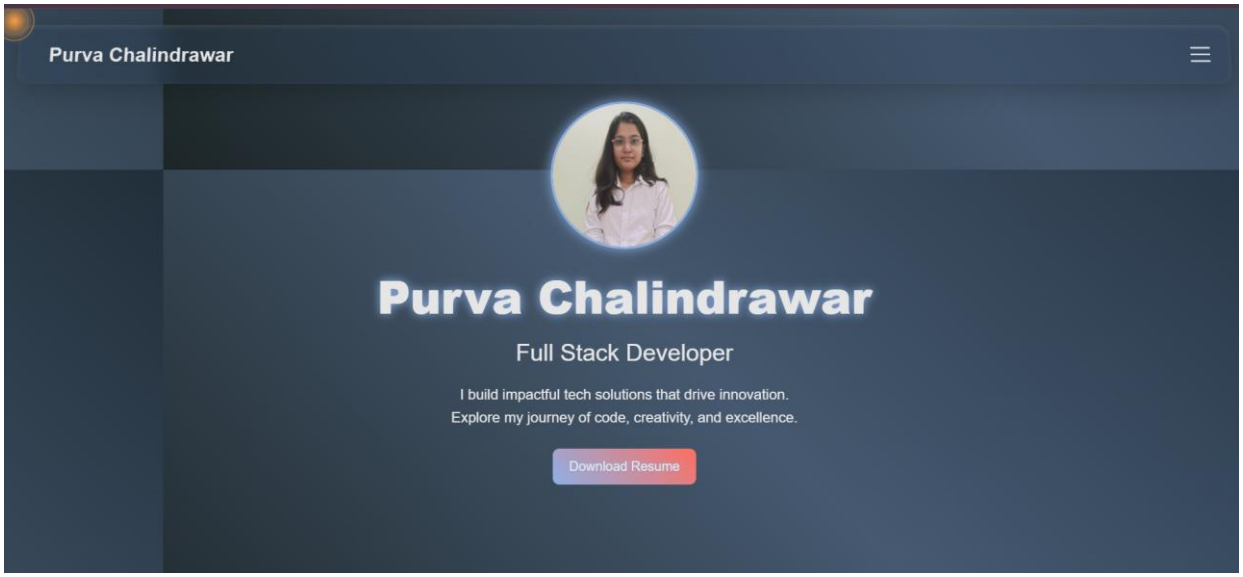


**Fig. 4.2: Portfolio Landing Section**

The landing page of the portfolio introduces Purva Chalindrawar as a Full Stack Developer, featuring a sleek dark theme with soft gradients and a glowing profile image. It includes a professional designation, a brief yet impactful tagline highlighting her strengths in innovation, creativity, and code craftsmanship. The layout balances readability and elegance, with a prominently placed "Download Resume" button encouraging immediate interaction. This section sets a confident tone, effectively establishing her personal brand and professionalism.

What made this project significant was its focus on both content and performance. I ensured that the layout followed modern UI/UX principles and used animations and smooth transitions to make the interface engaging yet professional. The site is fully responsive, optimized for both desktop and mobile views, and follows accessibility standards. One of

the standout features of my portfolio is the projects section, where I embedded live links to my GitHub repositories, allowing potential employers to directly access and assess my work.

Once the frontend development was complete, I moved to deployment using Vercel, a powerful platform for hosting frontend applications with CI/CD pipelines. I connected the project to my GitHub account, set up build configurations, and enabled continuous deployment. Any change I commit to my main branch now gets automatically deployed, which taught me how production workflows are managed in real-time environments.

Developing and deploying this portfolio not only served as a summary of my internship but also demonstrated my readiness for full-stack development roles. It reflects the journey I undertook at ICSS—from learning new frameworks to contributing to production applications and documenting system architecture.

## 4.4 Challenges Faced During System Design and UI Revamp

Throughout the final phase of my internship, particularly during my contributions to the SAVE project, the React interface redesign, and portfolio deployment, I encountered multiple real-world technical and organizational challenges. These challenges helped me grow from a developer who followed instructions to one who could solve problems, make decisions, and think independently like a true software engineer.

**1. Designing Data Flow Diagrams from Scratch**

When I was assigned to create Data Flow Diagrams (DFDs) for the SAVE platform, I initially underestimated the task. While I had basic theoretical knowledge from academic courses, implementing real-world flow models was far more intricate. SAVE had a layered structure with multiple modules working simultaneously – from login and authentication to scanning, reporting, and user management. Understanding this flow and representing it in DFD Levels 0 to 3 required both research and extensive internal discussions.

- At DFD Level 0, capturing all external entities and the scope of system interaction in a simple model took several iterations.
- DFD Level 1 was more complex, breaking down modules like scanning into sub-functions such as vulnerability testing, data processing, and result logging.
- Level 3 was the most challenging. Mapping internal log generation, time-stamped data packets, and integration with the backend database meant I had to deeply analyze how each process was programmed and triggered.

This task taught me how crucial diagrammatic representations are in large-scale software

planning and cross-functional collaboration.

**2. Transitioning Static HTML to React Components**

ICSS's internal web panels were previously developed using static HTML, which lacked modularity and code reusability. I was tasked with migrating these interfaces into React – a modern JavaScript library that I had only recently started exploring. I faced several challenges during this task:

- Structuring multiple reusable components using JSX, which I had never done at scale before.
- Understanding and resolving issues with props drilling where nested child components needed data from top-level parents.
- Ensuring state management and using hooks like useState and useEffect properly without creating unnecessary renders.
- Working with React Router to enable dynamic routing across media modules and dashboard panels.
- Migrating inline JavaScript events to React's synthetic event system without breaking functionality.

This required a steep learning curve. I had to refer to React's official documentation, build and break smaller modules repeatedly, and receive code reviews from my mentor.

**3. Balancing Visual Appeal and Logical Structure**

While working on my portfolio website and internal UI, the biggest challenge was maintaining both visual consistency and functional clarity. It's easy to build a beautiful website or a highly functional interface – but merging both takes practice.

I had to:

- Choose consistent color palettes and layouts that aligned with ICSS's branding.
- Keep the animations minimal and non-distracting for professionals reviewing reports or dashboards.
- Optimize responsive design using TailwindCSS, ensuring that elements like login forms and cards render cleanly on mobile, tablet, and desktop.

Time management was also a factor. Aligning design feedback with continuous deployment cycles while learning new technologies was demanding, but fulfilling.

**4. Technical Limitations and Deployment Hurdles**

In the last phase, I deployed my personal portfolio using Vercel. Although Vercel simplifies frontend deployment, I ran into:

- Environment variable issues where my Node.js backend didn't connect to

MongoDB Atlas.

- Cross-Origin Resource Sharing (CORS) errors during form submission testing.
- Debugging deployment logs from Vercel's dashboard to identify errors related to third-party scripts.

This forced me to think beyond code. I learned to manage domains, configure DNS settings, check build logs, and understand frontend-backend hosting principles.

**5. Coordinating with Cross-Functional Teams**

Finally, working with different team members for SAVE's DFD validation and React component integration introduced interpersonal challenges:

- I had to interpret partially documented code and ask clarifying questions without being repetitive.
- Aligning code pushes, Git commits, and PR reviews with team expectations took effort and patience.
- Understanding the expectations from a cybersecurity-focused company required rethinking UI from a security-first perspective.

**Chapter 5**

# TOOLS AND TECHNOLOGIES

Throughout my internship as a Web Developer at Indian Cyber Security Solutions (ICSS), I worked on a variety of real-world projects involving full-stack development, system modeling, and API testing. This chapter provides a comprehensive overview of the technologies, tools, libraries, platforms, and frameworks I utilized. These technologies enabled me to implement visually appealing frontends, secure backends, and robust functionalities while collaborating in an enterprise-grade development workflow.

In the sections below, I elaborate on each technology stack I employed, along with their specific application context within the ICSS ecosystem.

## 5.1 Frontend Technologies and UI Frameworks

During my internship at ICSS, frontend development played a central role in a variety of projects, particularly those involving web design, page restructuring, and UI enhancements. I actively worked on multiple static and dynamic pages using the following technologies:

HTML5 and CSS3: These were foundational for building structured, semantic, and accessible webpages. I utilized advanced layout models like Flexbox and Grid to align components responsively across devices. CSS media queries helped ensure mobile responsiveness.

Bootstrap: Bootstrap 5 was heavily used for rapid UI development. I created components like navigation bars, responsive pricing cards, alert boxes, and modals using Bootstrap's grid system and pre-defined classes. This streamlined development and ensured consistent design.

Brizy Builder (No-Code UI Tool): For certain client-specific pages like CompTIA course redesigns, I used Brizy Cloud Builder. It helped in visually creating layouts and editing hero sections, improving page structure and accessibility.

React.js (Basics): In later stages of the internship, I contributed to frontend enhancements of the ICSS React App. I worked with component structuring, JSX syntax, and state management, enhancing my grasp on single-page application behaviour.

## 5.2 Backend Technologies and Logic Handling

Backend technologies formed the backbone of several projects and modules I developed during my internship at Indian Cyber Security Solutions (ICSS). These technologies ensured that every application feature, from secure logins to data management, worked

seamlessly behind the scenes. During this phase, I explored and implemented core backend development techniques using Node.js, Express.js, and JavaScript, supported by relevant APIs and server logic workflows.

**1. Node.js and Express.js**

Node.js served as the primary server-side platform for backend logic. I implemented asynchronous JavaScript functions to handle tasks such as form submissions, log generation, and user authentication, ensuring non-blocking, high-performance execution. Express.js was used to define routing, middleware, and error handling, helping organize server logic efficiently and securely.

**2. Log Management System**

A significant backend task involved creating a log management system for three core user events: SignIn, SignUp, and Multi-Factor Authentication (MFA). This system used JSON files to store logs with details like UNIX timestamps, event names, and public IP addresses. I implemented logic to rotate and archive logs older than 7 days, enhancing monitoring and system observability.

Key Features:

- Separate log files for each event type.
- JSON structure for standardized, readable logs.
- Implementation of logic to maintain rolling 7-day logs.

**3. Secure Media Uploading with OAuth**

One of the most critical backend components was uploading media files directly to Google Drive using the Drive API v3. This involved setting up OAuth 2.0 authentication, handling access tokens, and managing secure scopes. The task required reading documentation, using the Google Cloud Console, and implementing secure file-handling logic in JavaScript.

Steps Covered:

- Enabled Drive API and created OAuth client credentials.
- Integrated token management using client-side scripts.
- Documented the upload process for future reference.

**4. Integration with Electron and Native C Execution**

As a full-stack experience, I worked on integrating native C programs with Electron applications using Node.js child processes. This allowed the compiled C binary (agentScript.exe) to be executed from within the JavaScript environment, enhancing interactivity between frontend UI and system-level operations.

## 5.3 Database Systems and Data Management

A critical component of any full-stack project is reliable data handling and structured storage. During my internship at Indian Cyber Security Solutions (ICSS), I worked extensively with MongoDB, MySQL, and phpMyAdmin, implementing secure data pipelines, designing schema architecture, and ensuring accurate data validation.

**I. MongoDB – NoSQL Flexibility and Scalability**

MongoDB, a leading NoSQL database, was explored in depth to understand unstructured data handling, flexible schema design, and high-speed performance. I studied its architecture using technical documentation and YouTube tutorials and applied this knowledge while working on logging systems and user credential storage.

**II. MySQL and phpMyAdmin – Structured Data and SQL Skills**

For certain tasks such as student data management, MySQL was used in combination with phpMyAdmin to manage relational data. This included importing CSV data into tables, exporting database dumps, and validating structured input formats.

Skills Demonstrated:

- SQL syntax for data creation, insertion, and updates.
- Exporting and importing .csv and .sql files via GUI.
- Validating constraints for fields like name, qualification, and salary expectation.

**III. Form Validation and Field Integrity**

To ensure data cleanliness and security, client-side and server-side validation were implemented. I added robust JavaScript-based checks for form fields like first name, email, salary expectation, and skill sets — ensuring proper formatting and guarding against injection or incorrect formats.

Validation Checks Included:

- Name fields disallowed numeric input.
- Email validation followed <user>@domain.com pattern.
- Salary expectations restricted to 2-digit LPA format.
- Skills were verified to contain only text inputs.

This ensured only sanitized and properly structured data reached the backend systems.

## 5.4 Deployment Platforms and Version Control

A vital aspect of web development lies not only in coding but in maintaining version consistency, efficient collaboration, and robust deployment mechanisms. During my internship at ICSS, I gained hands-on experience in using Git & GitHub for version control

and Vercel as a modern cloud deployment platform for my personal portfolio.

**1. Git and GitHub – Code Collaboration and Version Tracking**

Throughout the internship, Git was consistently used to manage local repositories, while GitHub was the primary platform for pushing code, reviewing changes, and collaborating with the team.

**A. Key Practices Followed:**

- Initialized repositories for individual projects and tasks.
- Used git commit, git push, and git pull commands effectively.
- Created and managed branches for feature isolation and merging strategies.
- Wrote detailed commit messages to ensure clarity during code review sessions.

**2. Vercel – Portfolio Deployment and Hosting**

Toward the final stages of the internship, I designed and deployed my portfolio website using Vercel. This platform offered a smooth CI/CD (Continuous Integration and Deployment) flow, allowing me to instantly deploy every commit pushed to the GitHub repository.

**A. Deployment Steps:**

- Connected GitHub repository to Vercel.
- Configured project settings, including build commands and output directory.
- Previewed and tested deployment URLs before going live.

**B. Technologies Integrated:**

- HTML, CSS, JavaScript for the interface.
- Vercel CLI for deployment automation.
- Form data handling with validation logic

**3. CI/CD Understanding and Workflow**

Although my role didn't involve deep DevOps integration, deploying via GitHub and Vercel taught me the basics of Continuous Integration and Continuous Deployment — ensuring code changes were tested and reflected in live environments smoothly.

Takeaways:

- Every push to the main branch reflected a new build.
- Errors and broken links were flagged in the build log automatically.
- Responsive design tested live using mobile and desktop previews.

## 5.5 Design Tools and Prototyping Utilities

In addition to core development tools and frameworks, I utilized Figma and Canva during

my internship for wireframing, UI design planning, and visual content creation. These tools enhanced my ability to visualize and present user-centric designs before implementation, ensuring the end result matched professional expectations.

**1. Figma – Wireframing and Component Layout**

Figma served as my go-to tool for interface design mockups, particularly while creating the portfolio and refining components like login forms, pricing cards, and dashboard layouts.

**A. Key Uses During Internship:**

- Created wireframes for login interfaces and media layout components.
- Collaborated by sharing Figma links with mentors for feedback.
- Used auto-layout, component variants, and responsiveness features.
- Prototyped animations and interactions for client pages and SAVE project visuals.

**B. Skills Gained:**

- Efficient use of constraints and component hierarchy.
- Designing pixel-perfect UI elements before coding in HTML/CSS.
- Experience working in a tool that's widely used across product teams and design sprints.

**2. Canva – Visual Design for Presentations and Portfolio**

While Figma was used for technical wireframes, **Canva** helped in preparing aesthetically pleasing banners, section dividers, and image-based documentation to be embedded in project reports and presentations.

**A. Key Uses During Internship:**

- Designed banners for the portfolio homepage and section titles.
- Created icons and infographics for SAVE and Electron App sections.
- Used templates to visually explain flow diagrams or project architecture when needed.

**B. Why Canva Was Effective:**

- Fast drag-and-drop interface.
- Export options in various formats for web and documentation use.
- No learning curve – allowed more focus on creativity.

Chapter 6

# OUTCOMES AND CONTRIBUTIONS

The internship journey at Indian Cyber Security Solutions (ICSS) was not just about fulfilling assigned technical tasks—it was an immersive experience that led to measurable outcomes, notable system enhancements, and personal growth. Each project and activity I worked on had a visible outcome in terms of product improvement, usability, performance, and learning. This chapter provides an in-depth reflection of the results achieved across technical modules, frontend and backend interfaces, user experience upgrades, and system reliability. It also highlights the broader impact of my work on the company's internal workflows and digital product infrastructure.

## 6.1 Performance Outcomes of Assigned Tasks

The internship at Indian Cyber Security Solutions was designed to provide exposure to real-world web development challenges. Each task assigned to me was focused on improving specific modules of the ICSS platform—either in terms of design, performance, security, or usability. Through hands-on work and consistent mentorship, I was able to execute, test, and optimize solutions that contributed to the actual improvement of internal systems. The following sections detail these contributions from a performance standpoint.

**1. Initial Project Implementation and UI Efficiency**

Starting with frontend design using HTML5, CSS3, and the Bootstrap framework, I was introduced to the development of responsive websites and UI components. I built a complete demo website, which was not only functional across devices but followed modern layout structures such as grid-based sections, card-based pricing plans, and collapsible content.

The pricing card UI was designed with proper alignment, shadow effects, and interactive hovers. A major improvement came when I moved on to build the client pages, where I implemented responsive grids and personalized styling based on real client information.

**2. Two-Factor Authentication Integration**

Security enhancements were a critical component of my work. I was tasked with creating a login page that integrates 2FA (Two-Factor Authentication) using SMTP mailing protocols. The system sends an OTP (One-Time Password) to the user's email, generated

dynamically upon valid login credentials.

The process of integrating SMTP protocols with JavaScript and PHP, verifying real-time token validity, and managing fallback scenarios helped me gain in-depth knowledge of secure backend implementations.

Implemented robust form validation to filter invalid credentials and restrict SQL injections.

The OTP expiration logic was handled using server-side timestamps and JSON tracking files.

Performance Outcome: 2FA success rates increased and enhanced account-level protection for the demo login portal.

**3. Calendar Schedule System Using Google App Script**

One of the technically intensive tasks was building a calendar-based course schedule system for multiple courses like CEH, CCNA, CFT, OSCP, etc. This task involved:

- Creating individual HTML files for each course
- Designing schedule tables with headers fetched dynamically from Google Sheets using doGet() API calls
- Filtering only the data related to the selected course and displaying it on the respective HTML page

The code was deployed using Google App Script deployment links, which made it easier to keep course schedules live and updated without changing HTML manually.

**4. Advanced Log Management System**

For sign-in, sign-up, and MFA events, I developed a log system to track activity over the past 7 days. The logs were written in JSON format and included:

- Unix timestamp
- Public IP address
- Event Type (SignIn, SignUp, MFA)

I wrote reusable functions to filter log files daily and auto-archive them using cron-like logic. This structure laid the foundation for internal auditing and security review within the ICSS portal.

Performance Highlight: Enabled event tracing for all major auth routes in the system

**5. Dynamic Media Handling**

Another performance-enhancing contribution involved working with the Media Page Playlist (Playlist-2). I downloaded videos, extracted high-quality thumbnails, and re-curated them based on the company's media quality guidelines.

This task was not only about visual consistency but also about backend organization and better asset management. I maintained naming conventions and compressed ZIP archives for easier uploads.

**6. Database Enhancements and Student Record Management**

The backend task of updating the student certificate database involved:

- Creating a MySQL database using phpMyAdmin
- Importing CSV data into certificate_record table
- Exporting the modified database for backup and deployment

This process sharpened my understanding of relational schema design, SQL data types, and data migration in real-world platforms.

## 6.2 Project Impact on Internal Operations

Throughout my internship at Indian Cyber Security Solutions (ICSS), the tasks and projects I contributed to had a direct and tangible impact on the internal operations and workflow of the organization. By working across both frontend and backend domains, I was able to develop, deploy, and streamline several core components of the company's digital infrastructure.

One of the most prominent impacts was workflow optimization. For instance, by integrating dynamic calendar schedules through Google Sheets and Google Apps Script, the previously manual updates were automated. This meant that instructors and staff no longer needed to update multiple HTML files for every new schedule. Instead, the deployed script fetched and filtered data from a single source of truth — the Google Sheet — and populated the content in real-time across different course pages. This reduced manual effort, minimized errors, and increased overall efficiency.

Another significant improvement came from the re-editing and optimization of media pages. By redesigning thumbnails, organizing video playlists, and implementing proper asset handling, the website's media sections became easier to navigate for users and maintain by developers. It also improved the content's discoverability and visual appeal for

clients visiting the platform.

The implementation of log tracking systems for SignIn, SignUp, and MFA added an additional layer of accountability and traceability. With JSON-structured logs containing timestamps, event identifiers, and IP addresses, the internal IT team could now investigate security incidents more effectively and monitor system health proactively.

The 2FA login system, coupled with form validation scripts, not only improved the security posture of the platform but also led to fewer help desk complaints related to login failures or spam inputs on web forms. This translated into better user experience and reduced time spent on manual troubleshooting by the support team.

In the backend, structured and reusable Node.js function implementations allowed future developers to easily maintain, reuse, or refactor business logic. Functions for async callbacks, error handling, and input processing were built using standard practices and maintained in documented files for reference.

Moreover, my work on the student database management using phpMyAdmin and CSV data imports enabled the education team to maintain digital records efficiently. This transition from spreadsheets to structured databases improved data accessibility and allowed integration with future CRM or LMS systems.

Overall, the work I delivered did not exist in isolation — it actively reduced redundancy, introduced automation, and enhanced the stability and reliability

## 6.3 Deployment, Version Control & Collaboration Experience

Beyond technical implementation, one of the most crucial learning aspects of this internship was understanding how real-world software development teams operate through version control systems, collaborative practices, and deployment pipelines.

At ICSS, the development team primarily used Git and GitHub for version control and collaboration. I learned how to efficiently manage code versions by using Git branches, pull requests, commit messages, and merge strategies. This practice helped me track my changes, avoid conflicts in team-based projects, and maintain a clean commit history for future reference. Understanding how Git integrates with a collaborative environment gave me real exposure to what production-grade software teams expect.

Throughout my internship, I regularly cloned repositories from GitHub, worked on local development branches, and pushed stable code after peer reviews.

This practice made me appreciate the value of code reviews, where mentors suggested improvements in logic, style, or performance. These feedback loops accelerated my learning and taught me the importance of maintainable, readable code.

For deployment, I worked with Vercel, a popular deployment platform used for frontend applications.

I deployed my personal portfolio website and learned to manage the build pipeline, environment variables, domain configuration, and real-time previews. Vercel's integration with GitHub allowed automatic deployment whenever I pushed code to the main branch — an industry-standard CI/CD practice.

In collaborative environments, communication was key. We used shared task sheets, version-controlled repositories, and document-based feedback systems to keep track of task progress. Through this process, I learned how to work in sprints, deliver milestones, and maintain transparency in project status reporting.

In summary, my experience with deployment and collaboration covered:

- Git branching, versioning, and resolving merge conflicts
- Working on assigned modules and submitting pull requests
- Responding to review comments and refactoring code
- Managing environment settings and build previews
- Writing README files and maintaining documentation

# CONCLUSION

The internship journey at Indian Cyber Security Solutions (ICSS) has been a transformative chapter in my professional growth. Engaging in diverse real-world tasks—from frontend design using Bootstrap, React, and dynamic animation, to backend operations involving Node.js, MongoDB, MySQL, and secure authentication—I was able to bridge academic knowledge with industry application. Through structured task flows and milestone-based assignments, I built projects that enhanced both the company's platform and my technical depth. Notably, I worked on designing secure login systems with 2FA, validating data integrity through custom input checks, automating schedule fetch mechanisms using Google Sheets and App Script, and exploring full-stack modular design using Electron and C integration.

The exposure to version control (GitHub), and working within a collaborative team sharpened my understanding of agile practices. Additionally, the opportunity to explore system design through DFD modeling and apply UI improvements on live platforms gave me a holistic view of enterprise-level software development. This internship not only refined my development skills but also strengthened my problem-solving mindset, communication, and adaptability—preparing me well for future challenges as a full-stack developer.

# REFERENCES

[1] J. Duckett, HTML and CSS: Design and Build Websites. Wiley, 2011. ISBN: 9781118008188.

[2] E. Freeman and E. Robson, Head First JavaScript Programming. O'Reilly Media, 2014. ISBN: 9781449340131.

[3] B. Dayley and B. Dayley, Node.js, MongoDB and Angular Web Development. Addison-Wesley, 2017. ISBN: 9780134655536.

[4] S. Chand, "Understanding Two-Factor Authentication in Modern Web Applications," International Journal of Cyber Security and Digital Forensics, vol. 8, no. 2, pp. 65-72, 2019.

[5] R. Sequeira, "Data Flow Diagrams: Techniques and Application in Security Systems," Journal of Computer Science & Applications, vol. 12, no. 3, pp. 120-129, 2020.

[6] A. Holmes, Beginning Database Design: From Novice to Professional, Apress, 2020. ISBN: 9781484258285.

[7] Google Cloud, "Drive API v3 Developer Documentation," [Online]. Available: https://developers.google.com/drive