Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Department of Electrical & Computer Engineering

Project Report

- **Course Code : ECE - 2100**
- **Course Title : Software Development Project- I**
- **Project Title : A Note Sharing Web Platform - ECExchange**

Submitted To:	Submitted By:
Md. Omaer Faruq Goni **Assistant Professor** **Department of ECE, RUET**	**Name : Nur Nafis Fuad** **Roll : 2210032** **Registration : 1086** **Session : 2022-2023** **Department of ECE, RUET**

List of Contents

Chapter 1: Abstract

This software project showcases ECExchange, a web based page designed to share resources within the department of Electrical and Computer Engineering (ECE) at RUET. The fundamental aim of this platform is to increase flexibility of academic and research-oriented resources among the students who thrive to learn extensively. The webpage is a centralized digital hub to upload, access and manage lecture notes, past questions, books or any kind of academic materials for a smoother studying and teaching experience. The project mainly focuses on the front-end development and also back-end development to a moderate extent. The site introduces a dynamic online community to the teachers and students that addresses the problem of finding resources of different semesters from a scattered form to an organized and well arranged manner. It has course-wise folder structure to help an individual to easily navigate to ones desired resource. The project required tools like VS Code, online icon libraries (e.g. Font Awesome), and Google's API for backend connectivity. The website architecture highlights some front-end and back-end features using HTML, CSS, and JavaScript for the interface, while PHP and Firebase are used for backend logic and real-time data management. For responsive web interaction JavaScript code has been used. The login system allows student verification using unique IDs, authenticating secure access and upload control. Visual elements like interactive cards, animated backgrounds, and auto-sliding galleries enhance user engagement. This report highlights the problem statement, design process, user interface experience, and future potential of ECExchange as a scalable academic collaboration platform.

Chapter 2: Introduction

Web development has never been my cup of tea. The zeal for computer programming and other computer science related tools started back in 2017. I recreated a short and small-scaled project available on an online platform by the end 2018 using languages like HTML, CSS and JavaScript. It was a simple cue sports (Billiards) game that was developed in the code playground of SoloLearn. Over time, I pulled myself apart from any kinds of web development program and therefore lost track of my web development journey. But I always cherished that childish yet beneficial thrill of sitting on the desk and hardcoding fun responsive designs. The Software Project-I of second year odd semester constantly rekindled those same thrilling moments of my post-childhood tech-philia.

The idea of developing a note sharing platform arised in the second semester of first year as lesser and lesser students were sharing their notes and resource availability was inadequate. Moreover, the available resources were not arranged in a suitable manner therefore, hard to manage and find notes. Specially, when class tests or other examinations were approaching near, the scarcity of a well arranged course materials was becoming more apparent. Therefore I felt the need of making a website that organized all the semester's matrials in one place.

The initial goal of this project was to arrange the notes in one place, easy upload and download features for students to spend less time on searching and more time in practising useful and relevant problems. But with time, the website had more upgrades. It was made into a hub for all ECE relevant resources like books, pdfs, etc. Another purpose of this website is to create a bridge between the teachers and students for more interactive study sessions for the betterment

of the department as a whole. Additionally, the project serves a personal objective of making a university website more lucrative. Generally, university websites are less colorful and emanates a tedious energy. Therefore, the aim was to make a website with vibrant front-end design.

The front-end design of the website incorporates various key features of HTML, CSS and JavaScript. Although most of the pages are created with PHP, because, the website is all about backend services. Intially the plan was to integrate MySQL as the data base. Even though, data base is not integrated for now, PHP files will allow to connect the pages to SQL servers easily than if they were HTML files. Here, no frameworks of CSS is used, rather basic CSS has been used as no previous experience of working on any CSS framework was available. Moreover the overall hexagonal front-end idea was inspired from a design found in a short video from Facebook. T he tools and software utilized include Visual Studio Code (VS Code) for writing and organizing the code, Photoshop for designing graphical elements, and Google Chrome for testing and debugging the website.

Chapter 3: Project Description

3.1 Fronted Design Concept

Web development has never been my primary strength, but the thrill of creative design and interactivity always lingered in my memory since I first experimented with HTML, CSS, and JavaScript in 2018. Reigniting that spark, the Software Project-I of my second-year odd semester gave me a chance to build something meaningful—a note-sharing platform tailored to our department's academic needs. The idea was born out of a genuine problem: scattered and poorly organized study materials.

As exams approached, students often found themselves scrambling for notes. This inspired me to develop a central platform where students can upload, download, and access resources effortlessly—spending less time searching and more time studying. Visually, I wanted the website to feel engaging and alive—unlike most dull and static university portals. The front-end design focuses on vibrance, clarity, and responsiveness. A dark-themed layout was chosen to give it a modern tech-inspired appearance, layered with a hexagonal background that adds structure and dynamism to the design. The hexagon motif—discovered from a Facebook design clip—inspired the site's visual identity and complements the tech-vibe I aimed to achieve. To keep the design unique and personal, I relied solely on vanilla CSS, avoiding any frameworks due to lack of prior experience. The layout adjusts fluidly across devices, thanks to relative positioning and responsive containers. Hover effects on buttons and icons enhance interactivity, making the site feel intuitive and lively.

Although most of the site is powered by PHP, reflecting the backend-centric nature of the project, the front-end holds its own with carefully crafted styling and structure. While MySQL integration is still pending, PHP ensures future compatibility for dynamic functionalities like authentication, uploads, and note management. Tools like Visual Studio Code helped manage and structure the codebase, Photoshop aided in crafting icons and banners, and Google Chrome served as the primary environment for testing and debugging. Overall, the design doesn't just aim to be functional—it strives to inspire a sense of

collaboration and efficiency among students, while standing out visually from the typical academic web portals.

3.2 User Requirements:

1. Recommended to use Google Chrome (version 110 or later), Mozilla Firefox (version 100+), Safari (version 14 or above), Opera (version 4 or above) for best user performance.
2. A Windows/Mac device is preferred, as the website may not be fully optimized for other platforms.
3. A stable internet connection is required to use login or signup features and to load maps.

3.3 Key Features:

3.3.1 HTML and CSS Features:

- **Buttons and Interactions:**
  Standard Action Buttons (e.g., Login, Sign Up):
  primary action buttons—like login or upload—feature padded rounded corners, bold font styles, and smooth hover transitions to give users interactive feedback.

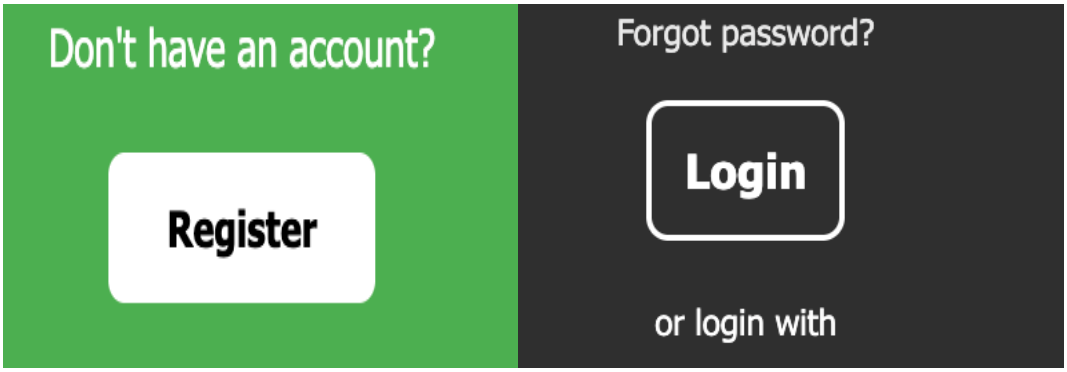

Figure 1: Login and Register Button

The standard action buttons, such as “Login” and “Sign Up”, are designed with a user-friendly aesthetic that includes rounded corners and adequate padding for better visual appeal and clickability. These buttons feature bold text and clearly defined borders to ensure prominence on the interface. Additionally, hover effects are implemented, which include a change in background color and a subtle glow, enhancing interactivity and providing immediate visual feedback to the user.

Feature Buttons/ Cards:
Special sections like the semester resource cards are styled using CSS grid and box shadows, giving a floating tile appearance. Optional courses and research/material

sections use consistent card-based UI with header underlines and padding for readability.

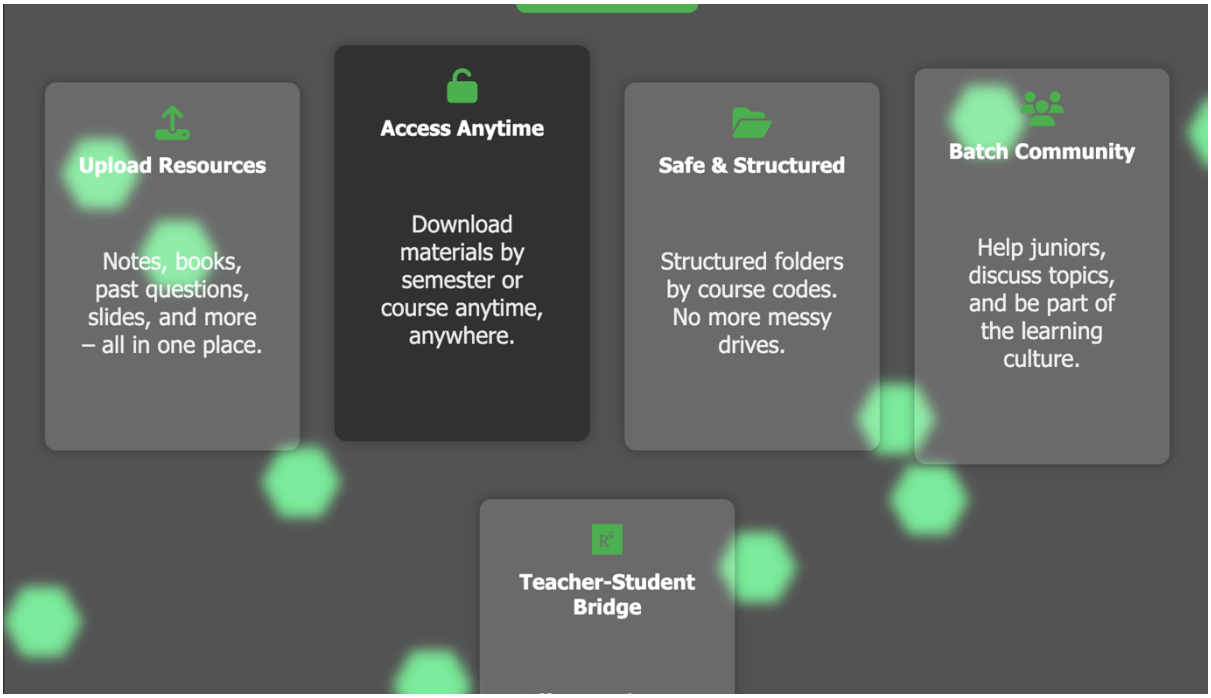

Figure 2: Feature Cards

The buttons are implemented with a *transform: scale()* effect on hover to create a subtle "pop" animation, enhancing the interactivity of the user interface. They are specifically styled for clarity and visibility on dark backgrounds, ensuring that the text and borders remain easily distinguishable and accessible across varying screen themes.

Header Ribbon Buttons:
The header section of the website includes essential navigation buttons such as "Home", "Contact", "Login" and "Sign Up". These buttons are styled with CSS to maintain consistency with the overall dark-themed interface.

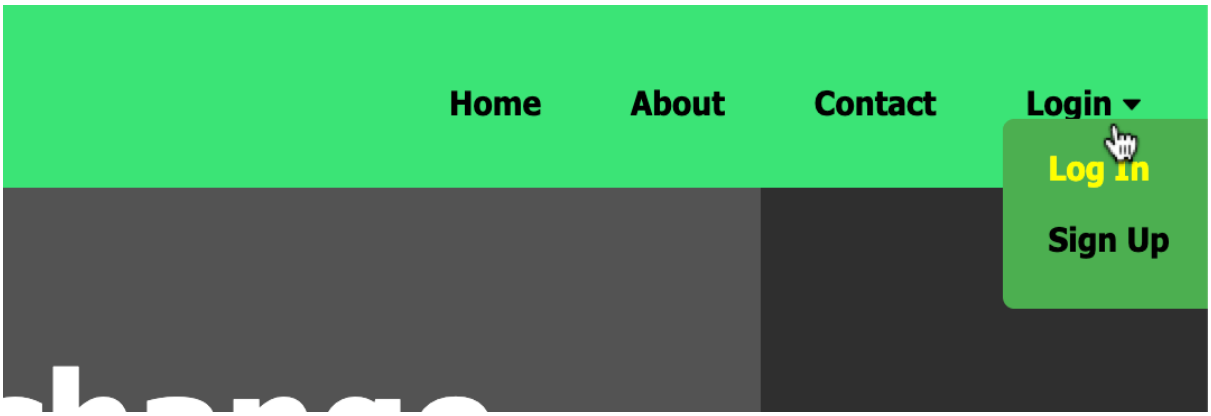

Figure 3: Home Ribbon Button Features

To improve user interactivity and feedback, a simple yet effective hover effect has been implemented — when hovered, the buttons change their background color (e.g., to

yellow), indicating they are clickable. This minimalistic approach avoids dropdown complexity while maintaining visual clarity and enhancing user experience. The hover transitions were achieved using CSS properties like :hover, background-color, and transition, ensuring responsiveness and smooth user interaction without relying on external frameworks.

Semester Resource Cards:
The Semester Resource Cards are designed using a CSS Grid layout to ensure responsive and consistent alignment across different screen sizes. Each card, such as "4th Year Odd Semester," features a semi-transparent background using rgba values to create depth, along with a box-shadow to produce a subtle floating effect. To enhance interactivity, a hover lift animation is applied using *transform: translateY(-5px)*, giving the cards a dynamic and engaging feel.

In addition, Optional Course Sections are presented in structured unordered lists (*ul-li*), organized under relevant category headers. These sections are visually separated from the main semester content through the use of distinct heading styles (*h3*) and subtle borders, ensuring clarity and a clean hierarchical structure.

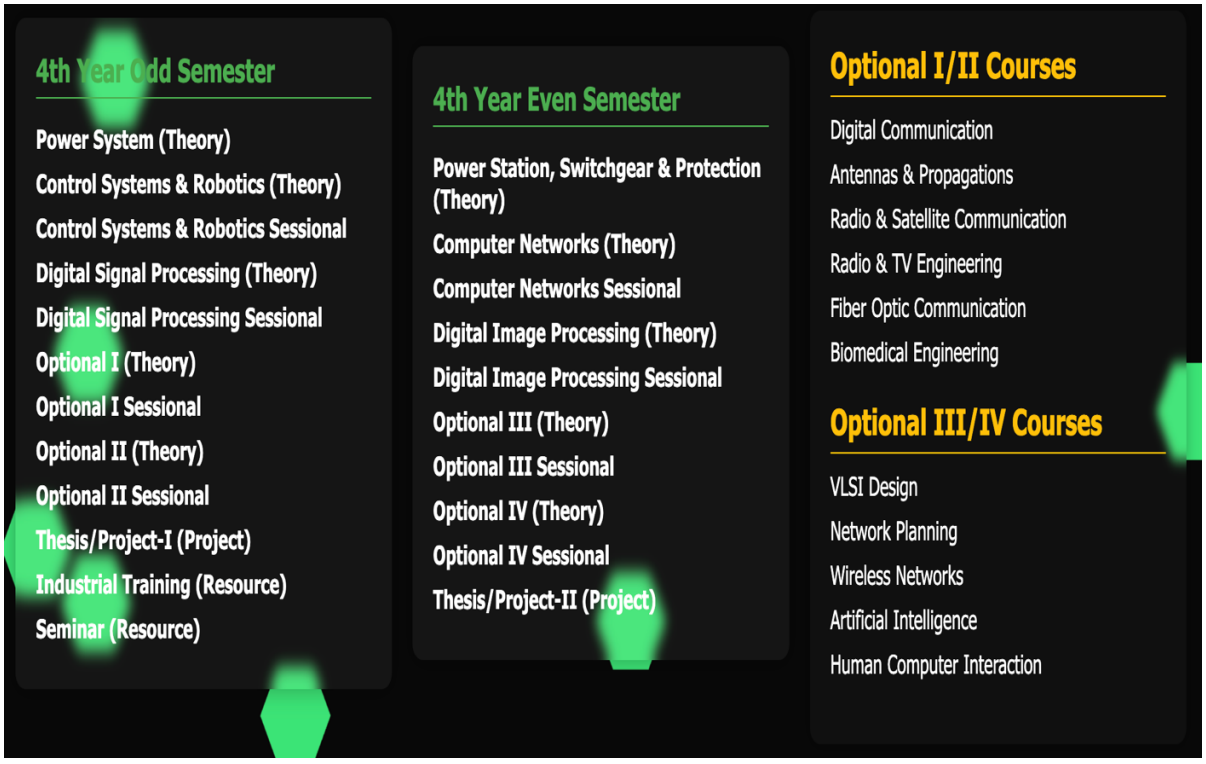

Figure 4: Semester-wise Resource Card

- **Sidebar Buttons:**
The sidebar navigation serves as a central component for organizing access to the various resources across all semesters and additional sections like research papers, software tools, and miscellaneous materials. The sidebar is positioned as a vertical panel fixed to the left of the screen and includes a set of clearly labeled buttons that guide users through the platform with ease. Each button is implemented using semantic anchor tags styled via custom CSS. On hover, these buttons highlight with a subtle

background glow and color transition to visually indicate interactivity, improving user experience.

Figure 5: Sidebar Positioning

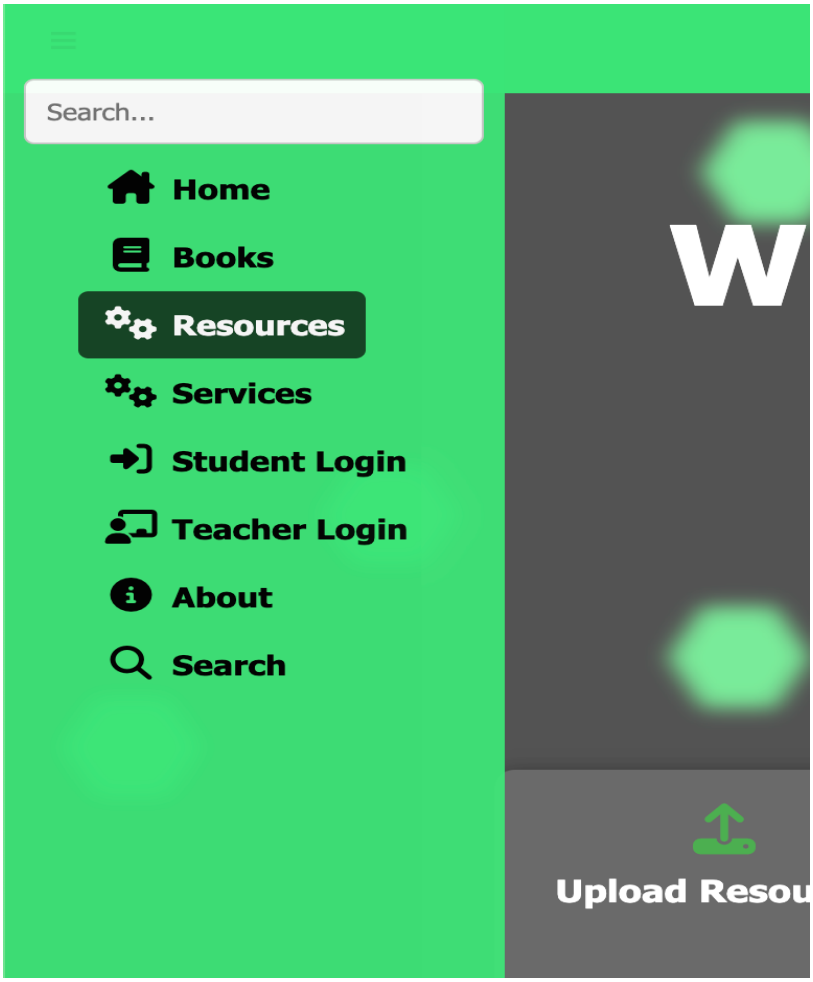

Figure 6: Sidebar, when Active

To maintain visual harmony with the dark theme of the website, the sidebar buttons feature light text, border radius, and padding, along with responsive design principles that adapt across screen sizes. The sidebar remains accessible at all times during navigation, allowing users to swiftly jump between semesters or resource sections without reloading or disrupting their workflow. This intuitive and minimalist navigation approach was chosen to keep the focus on content accessibility, especially during high-demand periods such as class tests or semester finals. The use of basic CSS (avoiding frameworks) also reflects the developer's hands-on understanding of front-end styling and layout handling.

- **Other Cards and Gallery:**
  Cards on About Page:
  The About page of ECExchange is designed using interactive flip cards that encapsulate essential information about the platform in a compact, visually appealing format. Each card uses a front-and-back face structure styled with CSS transform and perspective properties to create a smooth 3D flipping animation. This layout enhances user engagement by encouraging interaction. The cards cover topics such as About Website, *How it Started*, Why ECExchange, Features at a Glance, and Who Can Contribute. Each front face features a relevant icon using Font Awesome, and the back face presents detailed descriptions. The use of hover-triggered transitions, transparent backgrounds with blur effects, and rounded corners delivers a modern and polished feel consistent with the site's theme. This modular card-based approach also maintains responsiveness and accessibility across different screen sizes, providing both functionality and aesthetic consistency throughout the platform.

```css
.card {
    width: 400px;
    height: 280px;
    position: relative;
    transform-style: preserve-3d;
    transition: transform 6s ease-in-out;
}

.card-container:hover .card {
    transition: transform 0.3s ease-in;
    transform: rotateY(180deg);
}
```

```css
.card-face {
    position: absolute;
    width: 100%;
    height: 100%;
    background: black;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    font-weight: bold;
    text-align: center;
    padding: 15px;
    box-sizing: border-box;
    backface-visibility: hidden;
}

.front {
    background: linear-gradient(135deg, #2ecc71 0%, rgba(0, 0, 0, 0.8) 100%);
    color: white;
    font-size: xx-large;
}

.back {
    background: linear-gradient(0deg, #2ecc71 80%, rgba(0, 0, 0, .1) 100%);
    color: black;
    transform: rotateY(180deg);
    font-size: 16px;
    line-height: 1.4;
    padding: 20px;
    justify-content:space-evenly;
}
```

Figure 7: CSS for Card Design

Figure 8: About Page Cards

Login/ Register Container:

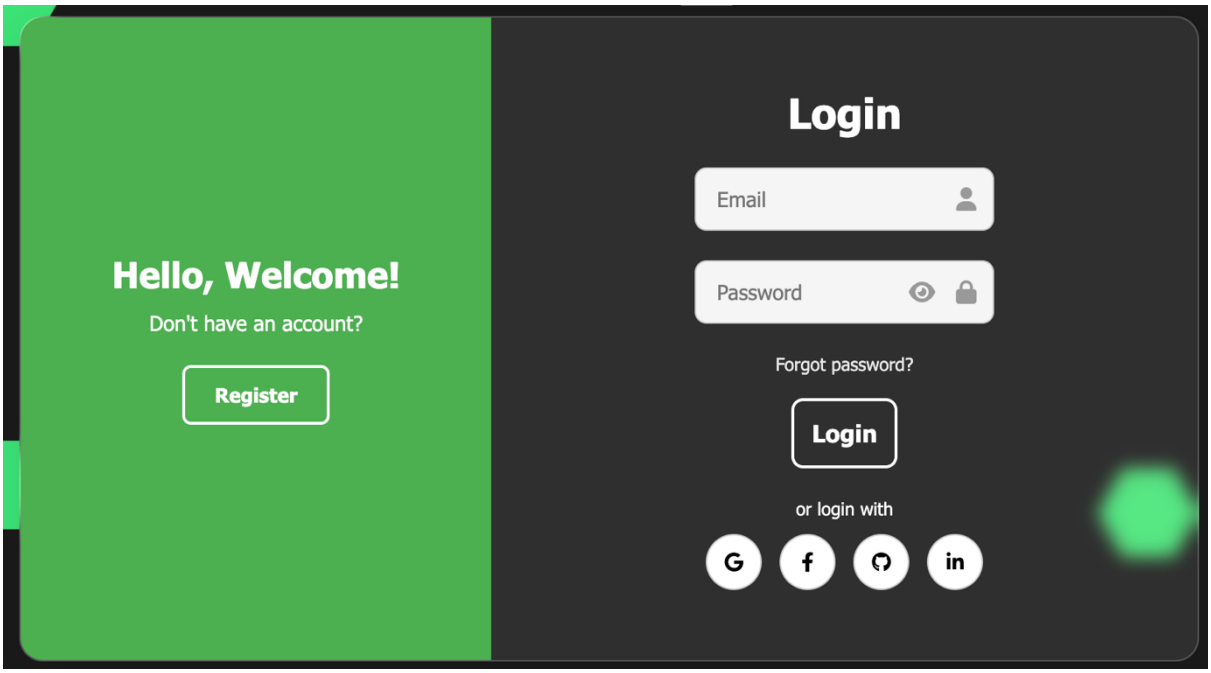

Figure 9: Login Container

The Login and Register Card in ECExchange is crafted to ensure both functionality and modern UI aesthetics. This dual-pane layout includes a welcoming left panel for new users with a Register button, and a right panel featuring a clean, interactive login form. The overall structure is styled using custom CSS without relying on any frontend framework, showcasing manual control over UI elements. The card design utilizes *glassmorphism* principles through backdrop-filter, translucent backgrounds, and subtle shadows, offering a soft, futuristic look.

Gallery/ Image Slider:

The Gallery Image Slider featured in the ECExchange landing page serves both an aesthetic and informative role. Built using pure HTML and CSS with *@keyframes*, it smoothly animates a carousel of images highlighting departmental memories, events, and significant moments. This slider is implemented using a horizontally scrolling *.gallery-track* inside a *.gallery-slider* container, ensuring full responsiveness and adaptability across various screen sizes. Images are styled using object-fit: contain to maintain aspect ratio without cropping, while a subtle *backdrop-filter: blur(12px)* adds a sleek, frosted-glass effect that blends well with the dark-themed interface.

The transition between images is controlled by the slideGallery animation, which cycles through a series of image panels using timed keyframe-based translations. This provides a continuous loop, enhancing the dynamism of the front page. For mobile responsiveness, media queries scale down the height of the slider while retaining readability. Overall, this slider is not only a modern visual enhancement but also

reinforces user connection by showcasing real, familiar glimpses of the ECE@RUET community.

Figure 10: Code Bits and Image Slider Glimpse

3.3.1 JavaScript Features (Client-side Interaction):

- **Sidebar:**
  Upon clicking the three dot (☰) we invoke the function *toggleSidebar()* which works with the onclick method of JavaScript. It is used to show or hide the sidebar on smaller screens or when the hamburger menu is clicked.

```
function toggleSidebar() {
  const sidebar = document.querySelector('.sidebar');
  sidebar.classList.toggle('active');
}
```

Figure 11: JavaScript For Click-opening the Sidebar

- **Password Visibility:**
  There is a password visibility toggle in the sign in and register page of the website. That is controlled by the *addEventListener() method increasing usability.*

```
const togglePassword = document.getElementById('togglePassword');
const passwordInput = document.getElementById('password');

togglePassword.addEventListener('click', function () {
const type = passwordInput.getAttribute('type') === 'password' ? 'text' : 'password';
passwordInput.setAttribute('type', type);


this.classList.toggle('fa-eye-slash');
});
```

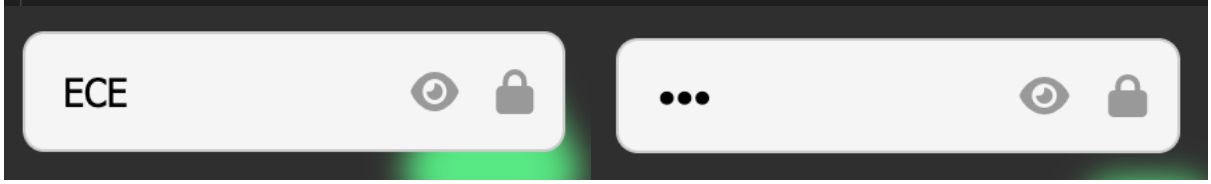

Figure 12: Password Visibility Toggle

- **Hexagon Movement and Color Changing:**
  The JavaScript function startHexagonMovement() is designed to animate hexagon elements dynamically across the screen in a floating, bouncing manner. It selects all elements with the .hexagon class and calculates their initial positions relative to the screen size, then assigns each one a randomized horizontal (xSpeed) and vertical (ySpeed) velocity. This idea is inspired from the famous *DVD logo bouncing* animation.

```
function startHexagonMovement() {

    const hexagons = document.querySelectorAll('.hexagon');

    hexagons.forEach(hexagon => {

        let xPos = parseFloat(hexagon.style.left);
        let yPos = parseFloat(hexagon.style.top);


        xPos = (xPos / 100) * window.innerWidth;
        yPos = (yPos / 100) * window.innerHeight;

        let xSpeed = 3 + Math.random() * 3;
        let ySpeed = 1 + Math.random() * 2;

        function moveHexagon() {

            xPos += xSpeed;
            yPos += ySpeed;


            if (xPos + 60 > window.innerWidth || xPos < 0) {
                xSpeed = -xSpeed;
            }
```

```
            if (xPos + 60 > window.innerWidth || xPos < 0) {
                xSpeed = -xSpeed;
            }


            if (yPos + 34.64 > window.innerHeight || yPos < 0) {
                ySpeed = -ySpeed;
            }

            hexagon.style.left = `${(xPos / window.innerWidth) * 100}%`;
            hexagon.style.top = `${(yPos / window.innerHeight) * 100}%`;

            requestAnimationFrame(moveHexagon);
        }

        moveHexagon();
    });
}

window.onload = startHexagonMovement;


const hexagons = document.querySelectorAll('.hexagon');
const body = document.body;

hexagons.forEach(hexagon => {
    hexagon.addEventListener('mouseenter', () => {
body.classList.add('theme-yellow');
    });
```

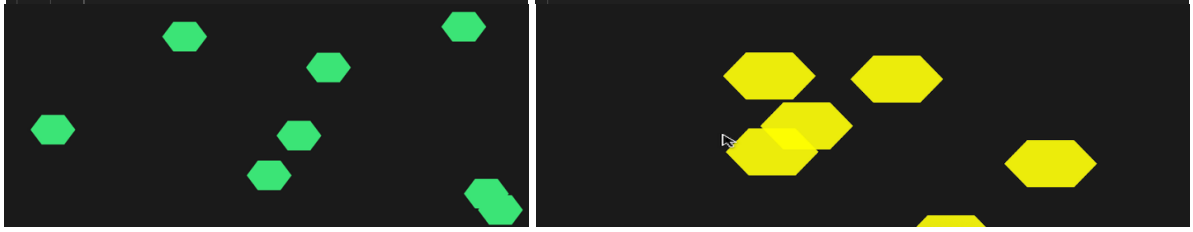

Figure 13: Hexagon Animation

Using *requestAnimationFrame*, the script updates their positions in a smooth animation loop while reversing direction upon collision with the screen boundaries—creating a continuous bouncing effect. Additionally, a separate event listener is added to change the entire page theme by toggling a theme-yellow class on the body when a user hovers over any hexagon, enabling an interactive and visually engaging user experience. This approach enriches the website's dynamic aesthetics while maintaining responsiveness and user interactivity.

3.4 Backend Design Concept

3.4.1 Include Statements using PHP:

To maintain code modularity and reuse, the project utilizes PHP's include statements to embed common components like header.php, sidebar.php, footer.php, and hexagons.php across multiple pages.

```
<body>
    <?php include 'header.php'; ?>
    <?php include 'sidebar.php'; ?>
    <?php include 'hexagons.php'; ?>
```

Figure 14: 'include' Statement in PHP

This approach follows the DRY (Don't Repeat Yourself) principle, ensuring that any update to a shared component reflects site-wide without needing repetitive code changes. It significantly simplifies the development and maintenance of the website while allowing consistent structure and layout throughout the application.

3.4.2 Searching Logic in *search.php (Server-side Filtering):*

The search.php page contains backend logic written in PHP to handle dynamic content rendering based on user queries. When a user types into the search bar, the query is sent through a GET request, and the PHP script processes it by searching through a predefined dataset (imported from *resources.php*).

The script performs case-insensitive matching of the input against available course or file names and displays relevant results. This filtering is done entirely server-side, making the search feature functional even without JavaScript, and helps offload the workload from the client browser.

```php
<?php
  if (isset($_GET['q']) && !empty(trim($_GET['q']))) {
    $query = strtolower(trim($_GET['q']));

    $searchItems = [];

    foreach ($coursesBySemester as $semester => $courses) {
      foreach ($courses as $course) {
        $searchItems[$course] = '#';
      }
    }
    //placeholder:

    foreach ($optionalCourses as $category => $courses) {
      foreach ($courses as $course) {
        $searchItems[$course] = '#';
      }
    }

    $results = [];

    foreach ($searchItems as $title => $link) {
      if (str_contains(strtolower($title), $query)) {
        $results[$title] = $link;
      }
    }
```

```php
    $results = [];

    foreach ($searchItems as $title => $link) {
      if (str_contains(strtolower($title), $query)) {
        $results[$title] = $link;
      }
    }

    if (count($results) > 0) {
      echo "<ul>";
      foreach ($results as $title => $link) {
        echo "<li><a href='$link'>$title</a></li>";
      }
      echo "</ul>";
    } else {
      echo "<p>No results found for <strong>$query</strong>.</p>";
    }
  } else {
    echo "<p>Please enter a search query.</p>";
  }
?>
```

Figure 15: Search Logic using PHP

3.4.3 Login and Signup Form Handling Using PHP:

```javascript
import { getAuth, signInWithEmailAndPassword } from "https://www.gstatic.com/firebasejs/10.11.0/firebase-auth.js";
import { initializeApp } from "https://www.gstatic.com/firebasejs/10.11.0/firebase-app.js";

const firebaseConfig = {
  apiKey: "AIzaSyDNRIJc8td0k03lKHnJ6lHAHlWXdY6yhbM",
  authDomain: "ecexchange-b6fd4.firebaseapp.com",
  projectId: "ecexchange-b6fd4",
  storageBucket: "ecexchange-b6fd4.appspot.com",
  messagingSenderId: "702935738337",
  appId: "1:702935738337:web:8419f6f6c7dd8644db54a0"
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
```

Figure 16: Firebase Connection using PHP

While Firebase Authentication is used for the actual credential verification (handled via JavaScript), the structure of the login and signup pages is built using PHP. The forms are placed inside PHP-driven layouts, which also load shared components via include. Although no session or cookie handling is currently implemented in PHP, the overall form submission process and page structure show clear preparation for a backend-based login system. The project is thus positioned to extend easily into server-side authentication and session management if a full backend with MySQL or Firebase Realtime Database is integrated later.

3.5 Database Concept

Although Firebase Authentication is successfully integrated to handle login and signup functionality for ECExchange, the project currently does not utilize any real-time database or Firestore for storing user data or uploaded resources. Initially, the plan was to implement MySQL as the backend database to allow users to store and retrieve files dynamically.

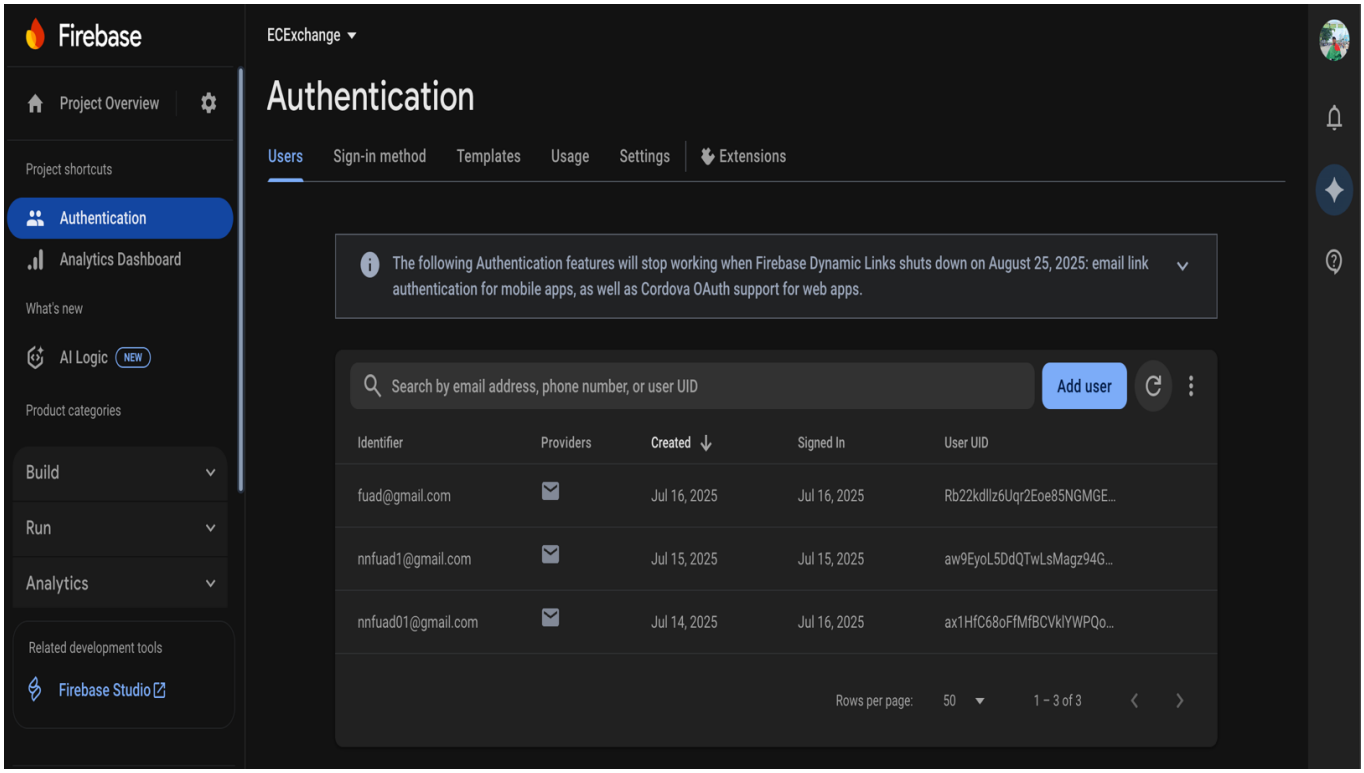

Figure 17: Firebase Dummy Database

In fact, most pages are structured using PHP rather than plain HTML to ensure easy integration with MySQL later on. However, in the current version, no actual user uploads are stored in a database — all links, resource folders, and upload logic are mock implementations or placeholders created for demonstration purposes. This allows the site to simulate full functionality while leaving scope for future upgrades like user-specific dashboards, upload history, and content moderation using a proper backend system.

Chapter 4: Development Process

4.1 Development Tools:

Figure 18: Tools Used in Development

4.1.1 HTML:

HTML is the foundational markup language used to structure and present content on the web. It defines the skeleton of any website, using tags and elements that represent headings, paragraphs, buttons, input fields, containers, links, and more. In the ECExchange project, HTML plays a crucial role in defining the layout and structure of all the pages — including the login/signup forms, dashboard layout, semester-wise resource listings, navigation menu, about section cards, landing page hero section, and even the interactive image gallery.

Elements such as <section>, , <form>, <input>, <a>, and <button> are widely used to build both content areas and interactive components. The HTML code is kept modular by separating the header, sidebar, and footer into individual include files (header.php, sidebar.php, footer.php), which promotes reusability and consistent structure across all pages. HTML in this project also provides the necessary hook points for styling (via class and ID attributes) and JavaScript-based interactivity.

4.1.2 CSS:

CSS is a stylesheet language used to describe how HTML elements should appear on screen. It allows developers to apply styles such as color, size,

spacing, layout, transitions, and animations, turning raw HTML into a visually engaging user interface. In ECExchange, custom CSS is used extensively to create a professional and modern look without the help of any external framework like Bootstrap or Tailwind.

The entire site is designed around a dark theme with glowing, responsive hexagonal backgrounds to create a tech-inspired aesthetic. CSS powers the design of the animated login and registration cards, hover effects on buttons and feature cards, the dynamic image slider on the homepage, and mobile responsiveness using media queries. Transitions and pseudo-classes (like :*hover*) are used to give feedback to user interactions. The layout system mainly uses flexbox and positioning for alignment and responsiveness, while backdrop-filter and semi-transparent elements add a glass-like blur effect across different components. CSS was written by hand for full control and understanding over each element's behavior and appearance.

4.1.3 JavaScript:

JavaScript is a client-side scripting language that enables interactivity and dynamic content on websites. It allows developers to manipulate the DOM (Document Object Model), respond to user events, validate inputs, animate elements, and fetch or send data without reloading the page. In ECExchange, JavaScript is used for both UI interactivity and logic handling. One major area is the movement of hexagon background elements, where a function using requestAnimationFrame() continuously animates hexagon positions, bouncing them off window edges while maintaining their percentages in a responsive layout.

JavaScript also enables toggling password visibility, dynamic theme highlighting on hover (mouseenter/mouseleave), and controlling sidebars with toggle functions. Moreover, JavaScript is used to handle the login form submission by interacting with Firebase Authentication modules, preventing default behavior and redirecting users upon successful login. The smooth animations, interactivity, and form validations all rely heavily on clean and efficient JavaScript code, enhancing both usability and visual polish.

4.1.4 PHP:

PHP is a server-side scripting language used to develop dynamic web content and backend logic. In ECExchange, PHP is used to modularize and organize the project efficiently. It helps keep repeated elements like the header, sidebar, and footer consistent across pages through include statements, reducing redundancy and simplifying updates. Furthermore, PHP plays a role in handling logic for pages like search.php, where it filters semester/course data based on user search input, processes $_GET queries, and returns matching results dynamically.

The login and signup pages also contain PHP logic to wrap HTML structures and support future form handling or database interactions. Though the current

authentication is managed through Firebase, the use of PHP ensures that the site is compatible with traditional back-end workflows, making it easier to transition into MySQL-based storage later. PHP acts as a bridge between frontend and backend, allowing dynamic server-side rendering of structured content in a scalable way.

4.1.5 Firebase:

Firebase [1] is a cloud-based platform developed by Google that provides tools for building and managing web and mobile applications. One of its most used features is Firebase Authentication, which enables secure user login and registration using email-password, social accounts, and more. In the ECExchange project, Firebase Authentication is integrated using ES6 JavaScript modules, providing a secure, modern, and scalable method for managing user sessions.

It replaces traditional PHP session-based login systems, reducing the need to manually handle encryption and user state management. Users can sign up and log in through a visually styled form, and upon successful authentication, are redirected to the dashboard. Although Firebase Realtime Database or Firestore is not currently connected, the code structure supports future expansion to allow users to upload and retrieve files or metadata. Firebase offers an ideal backend-as-a-service (BaaS) solution for projects like ECExchange that may scale with more active users and demand serverless data management.

4.2 Code Structure (Directory Management):

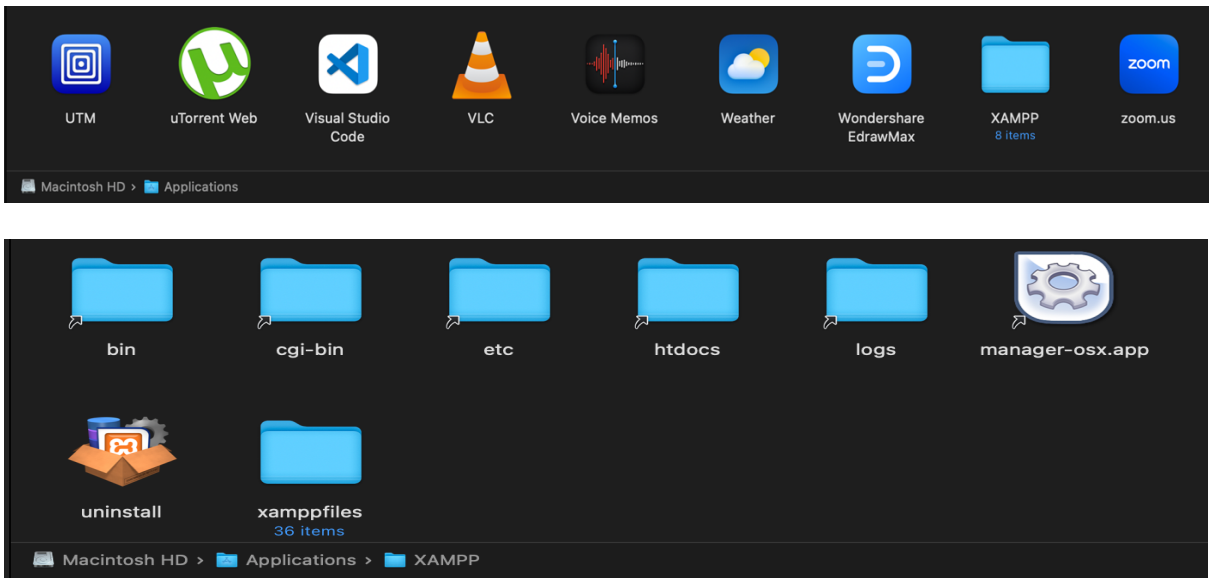

Figure 19: Folder Structure and Source

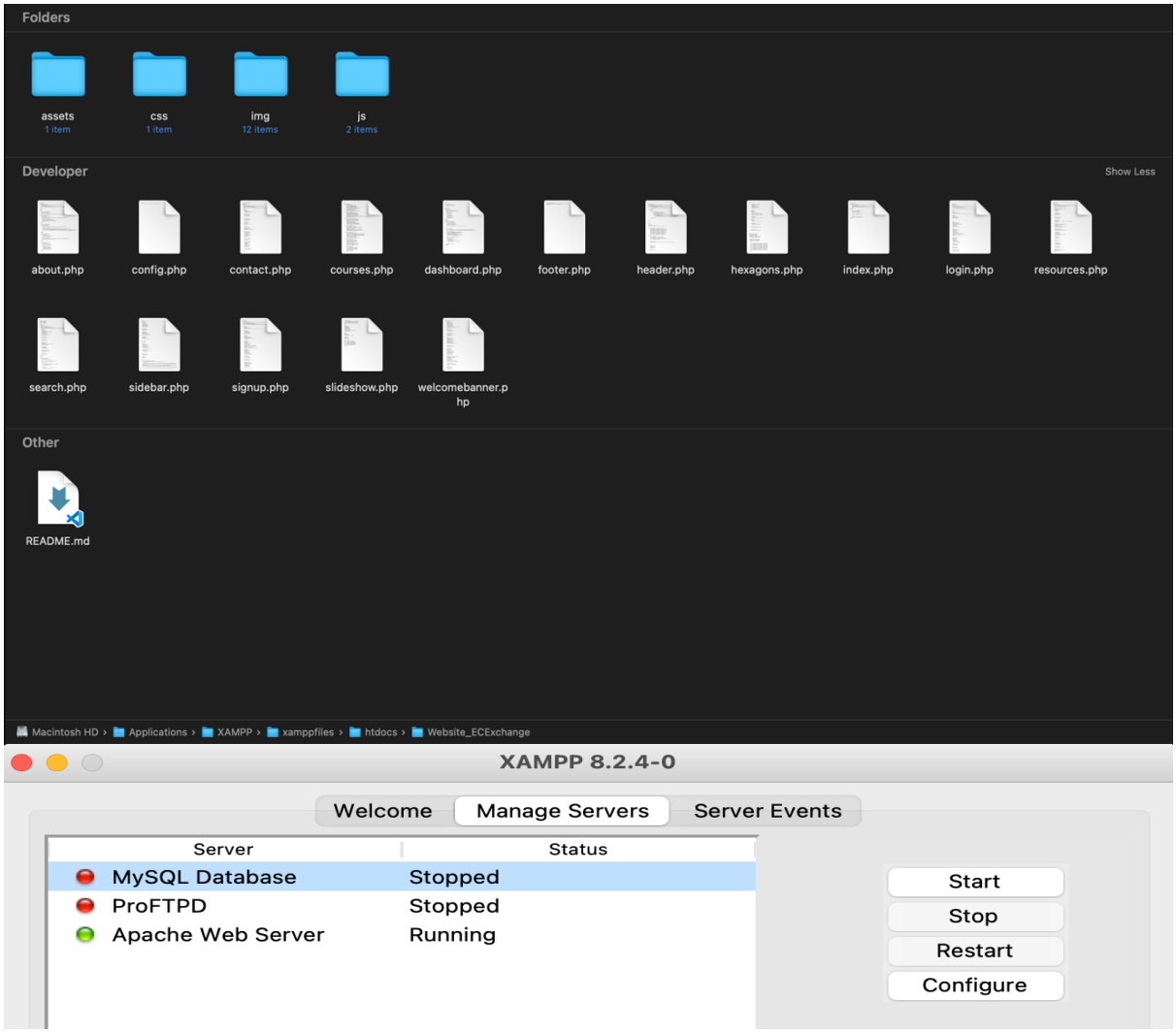

Figure 20: Source Code Destination and XAMPP Server Runner

Since the ECExchange project is developed using PHP—a server-side scripting language—it requires a local server environment to run and be tested. For this purpose, the project is hosted inside the *htdocs* directory of the XAMPP server. The *htdocs* folder acts as the root directory for all PHP-based web applications on a local server, allowing the browser to interpret .php files correctly when accessed via localhost. By placing all source code and asset folders (such as CSS, JS, and images) inside *htdocs/ecexchange*, the project can be launched in a local environment through the browser using the URL http://localhost/ecexchange. This setup ensures that PHP includes (like header.php, sidebar.php, etc.) and any backend logic (such as search.php) are properly executed by the Apache server.

In addition to the local deployment via XAMPP [2], GitHub was used throughout the development phase for version control and backup. A dedicated GitHub repository was created where all source code was regularly pushed to track changes, organize features, and prevent data loss.

4.3 Challenges:

1. **Revisiting Web Development Basics:**
Coming back to web development after a long break was a challenge in itself. I had to re-learn the basics of HTML, CSS, JavaScript, and PHP. Structuring the project files and using include statements like header.php, sidebar.php, etc., took a while to grasp properly, especially when trying to keep the layout modular and clean.

2. **Firebase Integration with PHP Frontend:**
One of the trickiest parts was integrating Firebase authentication (which is JavaScript-based) into my PHP-based website. Since PHP handles pages server-side and Firebase auth works on the client-side, syncing login logic and redirection—especially redirecting to the dashboard after login—required many rounds of debugging and adjusting the script structure.

3. **Frontend Styling without Frameworks:**
I chose not to use any CSS frameworks like Bootstrap or Tailwind, relying entirely on raw CSS as I don't have no prior experience of working with them. While this gave me creative freedom, it also meant writing and managing all styles manually. Designing features like responsive layout, hover effects, custom buttons, and the hexagon background animation took significant time and careful tuning.

4. **Database Limitations and Simulated Features:**
Since I didn't integrate a real database like MySQL, I had to simulate functionalities such as file uploads and course browsing using mock structures and placeholder links. This made some features non-functional for now but I ensured the codebase was structured so it can easily support real backend and database integration in the future.

4.4 Solutions:

To deal with the challenge of returning to web development after several years, I took a hands-on approach. I rebuilt the project step by step and referred to documentation and online resources when I needed help. I divided the UI into reusable PHP components, like header.php and sidebar.php, which made the code easier to read and maintain. For integrating Firebase with PHP, I used JavaScript's Firebase SDK for authentication on the frontend. I organized my PHP files to manage redirection and access control without relying on server-side sessions.

Although not having a database created some limits, I kept the upload and resource management logic modular and scalable. I used placeholder links for now but prepared the codebase for easy MySQL integration later. For styling the frontend, I designed the layout with pure CSS, applying responsive design techniques and interactive JavaScript features like animations and hexagon movement to create a lively user interface without using frameworks. Notably, I took help from websites like W3School [3], GeeksforGeeks [4] etc. and language models like OpenAI's gpt4-0 [5], Anthropic's Claude Sonnet 4 [6] etc.

Chapter 5: Testing

5.1 Testing Approaches:

1. Functional Testing

I manually tested each feature to ensure they behaved as expected. For instance, I tested the login and signup forms using valid and invalid credentials to verify Firebase Authentication worked correctly. I also checked button redirects, form validations, and whether dynamic PHP includes like header.php and sidebar.php loaded properly on all pages.

2. Interface Testing (UI/UX)

I tested the responsiveness and layout of the website across different screen sizes (mobile, tablet, and desktop) using the browser's developer tools. The aim was to ensure the hexagonal background, sidebar, and cards adjusted visually. I also validated hover effects, dropdowns, and animations on buttons and hexagons for consistency and usability.

3. Browser Compatibility Testing

The project was run on multiple browsers including Google Chrome, Mozilla Firefox, and Safari to ensure cross-browser compatibility. It was verified that the Firebase scripts and CSS styles rendered consistently without layout distortion or JavaScript errors.

4. Navigation and Link Testing

Each internal link (e.g., dashboard, upload, resources, login, logout) was clicked and tested to ensure they redirected correctly without breaking the layout or losing components like the sidebar or footer. The search bar was tested with different queries to confirm that results displayed accurately or gave a fallback message.

5. Firebase Auth Testing

I tested Firebase login and signup using test email addresses to ensure proper authentication, and that the correct redirect or success/failure message was shown. It was confirmed that users could not proceed to dashboard.php even after logging in.

5.2 Bug Fixes:

During the development of ECExchange, I found and fixed several bugs. One major issue occurred after successfully logging in with Firebase Authentication. The page was supposed to redirect to dashboard.php, but this redirection did not happen as expected. This was likely due to Firebase's asynchronous behavior and browser limits with file-based PHP loading. As a temporary solution, I displayed a JavaScript alert that said "Login Successful" instead of redirecting. This allowed users to confirm their login manually.

Another bug affected the search feature. The content from header.php appeared twice after a search was performed. This happened because of multiple include calls or incorrect layout rendering. I resolved it by changing the PHP structure. In the user interface, some hover

animations and hexagon movement effects conflicted with the sidebar's responsiveness. I fixed this by adjusting the CSS layering with z-index and improving position handling.

Additionally, I corrected minor interface bugs. The password visibility toggle overlapped with the input text, and there were inconsistent input resets after form submission. I made these changes through targeted JavaScript adjustments. Together, these fixes improved the usability, structure, and stability of the project across different environments.

Chapter 6: Conclusion

6.1 Summary:

ECExchange is a long-thought project idea, yet completed within small time-frame because of some schedule mismanagement. However, this website's main theme of making a university website more colorful and somewhat dynamic is achieved. This platform can be used to increase students study and research proactivity while maintaining a constant bridge between a teacher-student's academic relation. Basic programming languages have been used in the website like HTML, CSS, JavaScript, PHP and Firebase for an experimental database addition. This website integrates different basic concepts of software development scope. Besides having a dynamic design, it allows a user-friendly and smooth UI for file sharing and uploading. Thus, this project covers basic suppositions of front-end development and brief idea of backend development.

6.2 Limitations:

1. No Real-Time Database Integration:
Although Firebase Authentication is successfully used for login and signup functionalities, there is no real-time database (such as Firebase Firestore or MySQL) connected. As a result, user data, uploaded files, and session states are not stored or tracked persistently.

2. Manual File Structure:
The course materials and semester-wise folders are currently hardcoded using placeholder links. This makes it difficult to scale or update content dynamically without manually modifying the code, which could have been automated using a backend database.

3. Limited Interactivity After Login:
The login system provides authentication but lacks a proper redirection or session management flow. After login, users remain on the same page and only see a JavaScript alert, which is not ideal for user experience. The dashboard is not automatically presented, which can confuse users.

6.3 Future Improvement:

In future versions of ECExchange, I plan to integrate a MySQL database for storing user uploads, login credentials, and course materials dynamically. The project may also be hosted live so students and teachers can access it from anywhere. Additionally, I aim to add AI features like smart search or content suggestions to make resource sharing faster and more efficient.

Chapter 7: References

[1] Google, "Firebase," Google, 2014. [Online]. Available: https://firebase.google.com/. [Accessed July 2025].

[2] Apache Friends, "XAMPP Apache," Apache Friends, 2002. [Online]. Available: https://www.apachefriends.org/. [Accessed 2025].

[3] W3School, "W3School," Refsnes Data, 1998. [Online]. Available: https://www.w3schools.com/. [Accessed 2025].

[4] GeeksforGeeks, "GeeksforGeeks," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/. [Accessed 2025].

[5] OpenAI, "ChatGPT," OpenAI, [Online]. Available: https://chatgpt.com/. [Accessed 2025].

[6] Anthropic, "Claude," Anthropic, [Online]. Available: https://claude.ai . [Accessed 2025].