



CSIS 4495 - 002 Applied Research Project

Midterm Report

Finance Tracker Web Application for Budget Management

Video link- [W25_4495_S2_NidhiN_Midterm_Recording](#)

Nidhi Nidhi, 300378175

CSIS 4495-002

Instructor: Padmapriya Arasanipalai Kandhadai

Date: February 20, 2025

Douglas College

Table of Contents

1. [Introduction](#)
 - a. [Domain and Background](#)
 - b. [Problem Statement](#)
 - c. [Literature Review](#)
 - d. [Hypotheses and Benefits](#)
2. [Summary of Initially Proposed Research Project](#)
 - a. [Research Design and Methodology](#)
 - b. [Technologies](#)
 - c. [Expected Results](#)
3. [Changes to the Proposal](#)
 - a. [Frontend Framework: Flutter to Bootstrap](#)
 - b. [Authentication Mechanism](#)
 - c. [Database Structure and Access](#)
4. [Project Planning and Timeline](#)
 - a. [Updated Timeline](#)
 - b. [Project Management](#)
5. [Implemented Features](#)
 - a. [User Authentication System](#)
 - b. [Dashboard Design](#)
 - c. [Expense Visualization](#)
6. [Work Date/Hours Log](#)
7. [References](#)

Introduction

Domain and Background

Personal finance management is critical for individuals to achieve financial stability and carry out their monetary goals. Despite its importance, many individuals struggle with tracking their expenses, creating and adhering to budgets, and making informed financial decisions. This research project focuses on developing a web application that eases expense tracking, budget customization, provides AI assistance for financial decisions, and generates comprehensive PDF reports.

The application aims to bridge the gap between simple expense tracking and actionable financial insights by creating an intuitive platform that not only records financial data but also helps users understand their spending patterns and optimize their budgets.

Problem Statement

The research addresses several key questions related to personal finance management:

1. How can users efficiently track income and expenses while gaining actionable insights to improve their savings?
2. Can AI integration improve personalized budget recommendations based on individual spending patterns?
3. How can financial data visualization enhance users' understanding of their financial habits and encourage better decision-making?

These questions are particularly important in today's economic environment, where financial literacy and efficient budget management are essential skills for individuals across all income levels.

Literature Review

Current personal finance applications in the market primarily focus on basic expense tracking functionality but often lack comprehensive recommendation features. Notable gaps in existing solutions include:

1. **Limited AI Integration:** While some platforms offer basic insights, they typically lack sophisticated AI-driven recommendations tailored to individual financial behaviors.
2. **Static Visualization Tools:** Many existing applications provide basic charts and graphs, but fail to offer interactive visualization tools that can adapt to user-specific inquiries and preferences.
3. **Segregated Functionality:** Users often need multiple applications to cover expense tracking, budgeting, financial forecasting, and report generation, creating a fragmented user experience.

This research aims to address these limitations by developing an integrated solution that combines robust tracking capabilities with intelligent insights and flexible visualization options.

Hypotheses and Benefits

Primary Hypothesis: AI-driven financial recommendations will significantly improve user engagement and financial decision-making compared to traditional expense tracking methods.

Secondary Hypothesis: Interactive visualization of financial data will lead to better budget adherence and savings rates among users.

Expected Benefits:

- Enhanced user experience through intuitive interface design and reduced manual input requirements. For e.g. Input in Natural language
- Improved financial decision-making through personalized AI recommendations
- Better understanding of spending patterns through interactive visualizations
- Long-term improvement in financial planning capabilities for users
- Centralized platform for all personal finance management needs

Summary of Initially Proposed Research Project

Research Design and Methodology

The original project proposal outlined the development of a full-stack web application using Node.js for the backend, Bootstrap/EJS for the frontend, and MongoDB as the database solution. The application was designed to enable users to:

1. Track income and expenses across various categories
2. Set and manage budgets
3. Visualize financial data through interactive charts
4. Receive AI-powered insights and recommendations

5. Generate PDF reports for offline review

The project adopted an Agile development methodology, featuring iterative development cycles and regular testing. This approach was chosen specifically to accommodate the integration of AI components and allow for adjustments based on user/client feedback during the development process.

Technologies

The initially proposed technology stack included:

- **Operating System:** Windows/Linux (ensuring cross-platform compatibility)
- **Frontend:** Bootstrap 5 with EJS templating for responsive design
- **Backend:** Node.js with Express.js for API development and server-side logic
- **Database:** MongoDB (NoSQL) for flexible expense categorization and data storage
- **AI Tools:** Google's Gemini API for generating expense recommendations and insights

This technology stack was selected for its flexibility, scalability, and suitability for rapid application development.

Expected Results

The project aimed to deliver:

1. A fully functional web application enabling users to track expenses, set budgets, and generate comprehensive reports
2. AI-driven insights to help users make better financial decisions (e.g., "Fuel expenses increased by 20% this month")

3. Interactive visualization tools to help users understand their spending patterns
4. An intuitive user interface accessible across multiple devices

The primary contribution would be empowering users to make data-driven financial decisions through a combination of tracking tools, visualization capabilities, and AI-powered recommendations.

Changes to the Proposal

Since the initial proposal, several strategic changes have been made to improve the project's implementation and ensure better outcomes. These changes reflect deeper research into technology options and a clearer understanding of user requirements.

Frontend Framework: Flutter to Bootstrap

Original Plan: The first consideration included Flutter as a potential front-end framework.

Current Decision: Bootstrap with EJS has been confirmed as the front-end solution.

Justification:

- Flutter, while excellent for cross-platform mobile development, presents potential challenges for web applications including slower loading times and browser compatibility issues.
- Bootstrap offers mature community support, extensive documentation, and proven performance for responsive web design.

- EJS provides server-side templating that works seamlessly with Node.js, allowing for dynamic content generation without additional client-side frameworks.
- This combination enables faster development cycles and better performance for web-based financial applications.

Authentication Mechanism

Original Plan: Use Passport.js for user authentication.

Current Decision: Implement a custom authentication system.

Justification:

- After reviewing Passport.js documentation, it was determined that the library offers more complexity than required for the project's simple username/password authentication needs.
- A custom authentication system provides better control over the user experience and session management.
- The simplified approach reduces dependencies and potential points of failure while maintaining adequate security for the application's purposes.

Database Structure and Access

Original Plan: Use MongoDB with default security settings.

Current Decision: Configured MongoDB for broader access with plans to implement proper security measures before production.

Justification:

- Initial connection attempts to the MongoDB cluster were unsuccessful due to default security settings that block access from external IP addresses.
- For development purposes, the database has been configured to accept connections from any IP address.
- This approach facilitates easier development and testing across different environments.
- Prior to production deployment, proper security measures will be implemented, including IP whitelisting, encrypted connections, and secure authentication mechanisms.

Project Planning and Timeline

Updated Timeline

The following timeline outlines the remaining phases of the project from the current point to completion:

Phase	Dates	Key Deliverables
Midterm Submission	Feb 24, 2025	Midterm Report, Basic UI, Authentication System

Core Functionality	Feb 25 - Mar 7, 2025	Expense Tracking System, Category Management, Data Storage Implementation
Data Visualization	Mar 8 - Mar 14, 2025	Interactive Charts, Financial Summary Views, Dashboard Enhancements
AI Integration	Mar 15 - Mar 28, 2025	Google Gemini API Integration, Recommendation Engine
PDF Reporting	Mar 29 - Apr 3, 2025	Report Generation, Export Functionality
Testing & Refinement	Apr 4 - Apr 10, 2025	Testing, Bug Fixes, Performance Optimization
Final Submission	Apr 11, 2025	Complete Application, Final Report

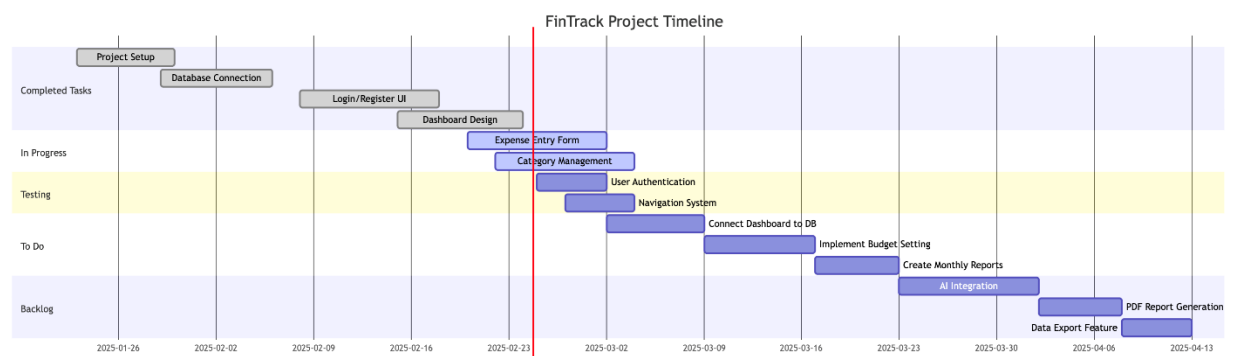
Project Management

As this is a solo project, all responsibilities are managed by a single developer (Nidhi). The development process is being tracked using a Gantt chart to visualize the project timeline and dependencies. Tasks are organized into the following categories:

- Completed

- In Progress
- Testing
- To Do
- Backlog

This approach allows for clear timeline visualization, identification of critical paths, and accurate tracking of project progress against scheduled milestones. The current Gantt chart is shown below, illustrating both completed work and upcoming deliverables:



Implemented Features

As of the midterm report deadline, several key features have been implemented, establishing the foundation for the Finance Tracking application.

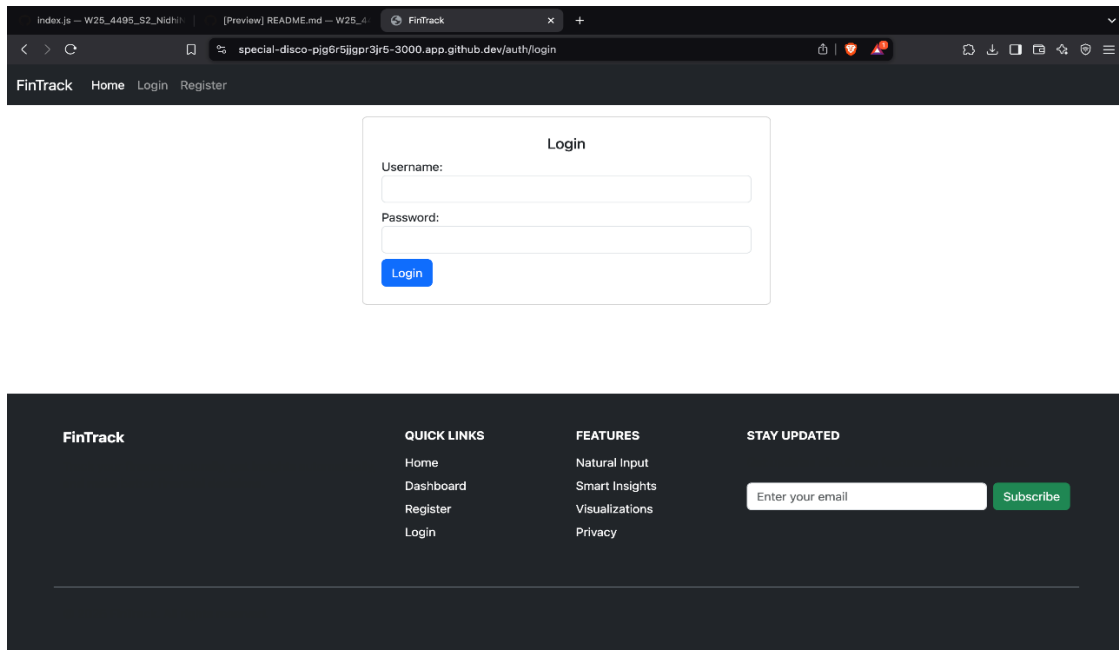
User Authentication System

The user authentication system has been implemented with custom authentication mechanisms instead of using Passport.js. This system includes:

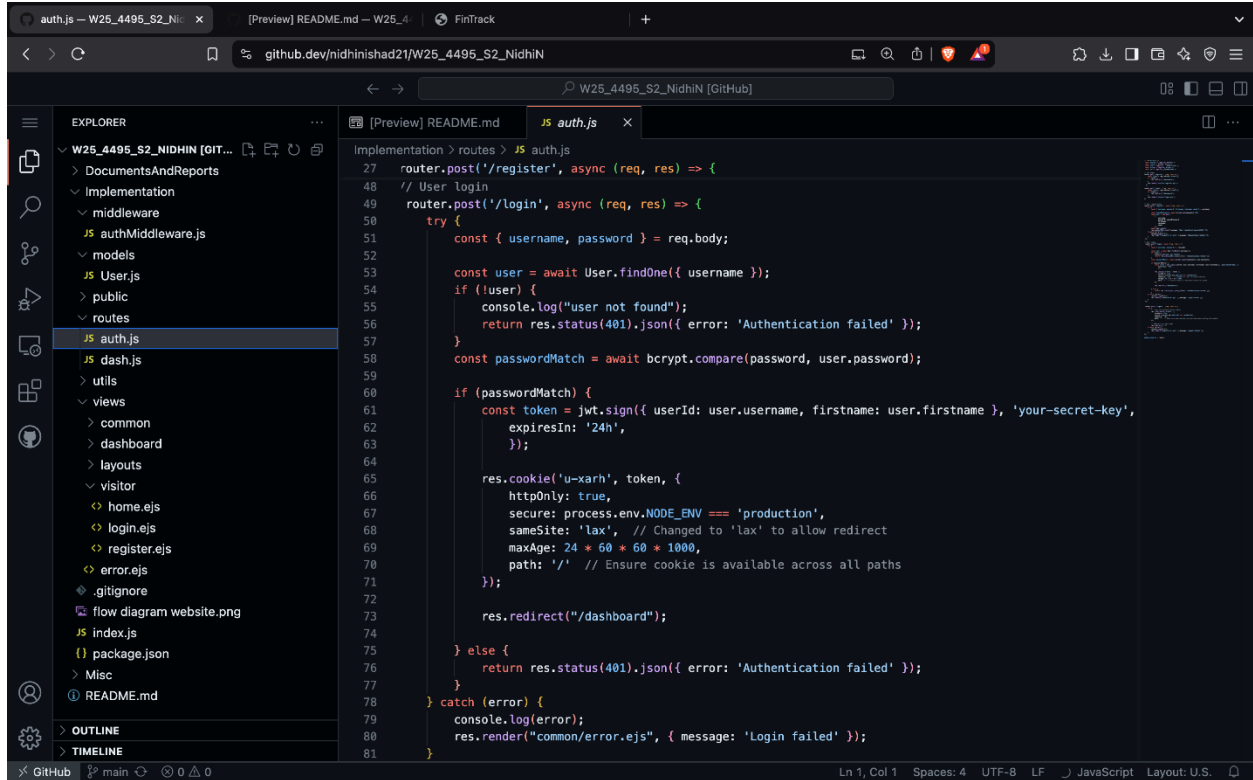
1. **User Registration Page:** Allows new users to create accounts with basic profile information.
2. **Login System:** Securely authenticates users and establishes sessions.
3. **Session Management:** Maintains user login state throughout their interaction with the application.

The authentication system differentiates between registered users and visitors, providing appropriate views and access levels based on user status.

Implementation Screenshots:



Code Sample (Login Authentication):



The screenshot shows a VS Code editor interface with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with files like `auth.js`, `dash.js`, `utils`, `views`, `common`, `dashboard`, `layouts`, `visitor`, `home.ejs`, `login.ejs`, `register.ejs`, `error.ejs`, `.gitignore`, `flow diagram website.png`, `index.js`, `package.json`, `Misc`, and `README.md`. The code editor displays the content of `auth.js`, which includes routes for `register` and `login`. The `login` route uses `bcrypt` to compare passwords and `jsonwebtoken` to generate tokens. The code is as follows:

```
27 router.post('/register', async (req, res) => {
48 // User login
49 router.post('/login', async (req, res) => {
50   try {
51     const { username, password } = req.body;
52
53     const user = await User.findOne({ username });
54     if (!user) {
55       console.log("user not found");
56       return res.status(401).json({ error: 'Authentication failed' });
57     }
58     const passwordMatch = await bcrypt.compare(password, user.password);
59
60     if (passwordMatch) {
61       const token = jwt.sign({ userId: user.username, firstname: user.firstname }, 'your-secret-key',
62         { expiresIn: '24h',
63           });
64
65       res.cookie('u-xarh', token, {
66         httpOnly: true,
67         secure: process.env.NODE_ENV === 'production',
68         sameSite: 'lax', // Changed to 'lax' to allow redirect
69         maxAge: 24 * 60 * 60 * 1000,
70         path: '/' // Ensure cookie is available across all paths
71       });
72
73       res.redirect("/dashboard");
74     } else {
75       return res.status(401).json({ error: 'Authentication failed' });
76     }
77   } catch (error) {
78     console.log(error);
79     res.render("common/error.ejs", { message: 'Login failed' });
80   }
81 }
```

Registration Page:

auth.js — W25_4495_S2_NidhiN | [Preview] README.md — W25_4 | FinTrack | Cluster0 Data | Cloud: MongoDB | +

fantastic-waffle-wr947qrr5xr9965-3000.app.github.dev/auth/register

FinTrack Home Login Register

Register

Username

Password

FirstName

LastName

Email

[Register](#)

FinTrack

QUICK LINKS

- Home
- Dashboard
- Register
- Login

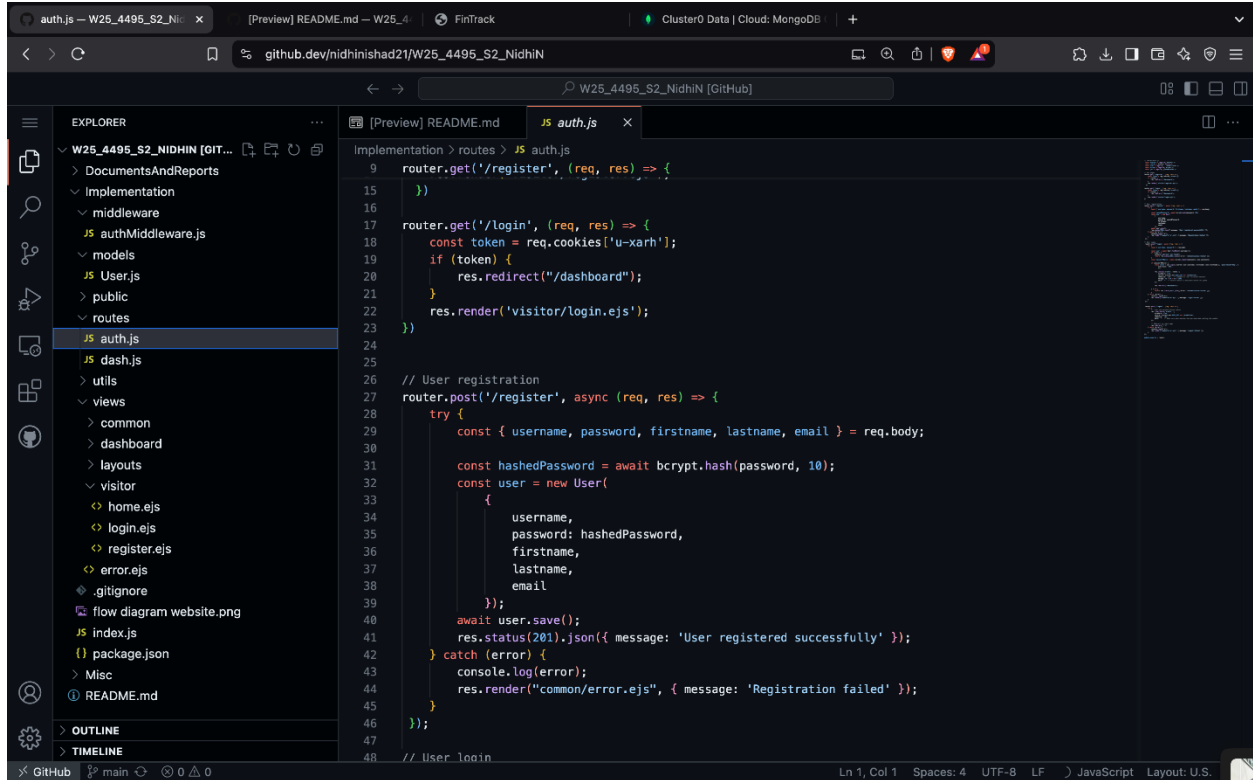
FEATURES

- Natural Input
- Smart Insights
- Visualizations
- Privacy

STAY UPDATED

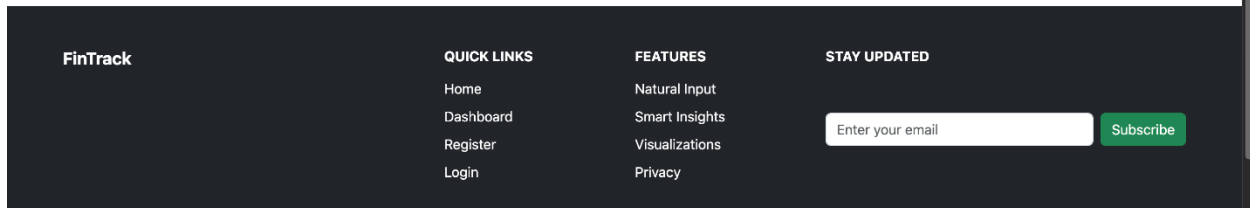
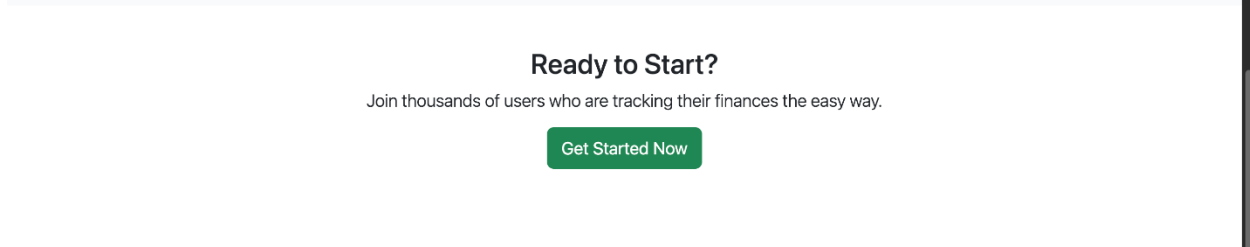
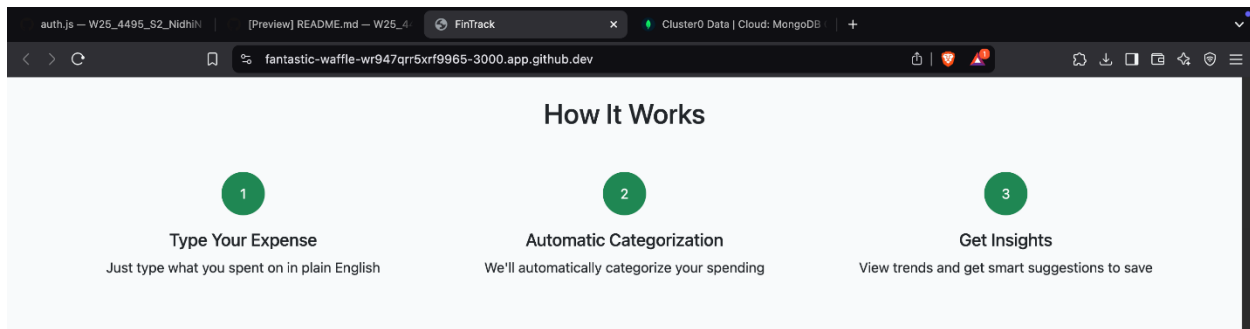
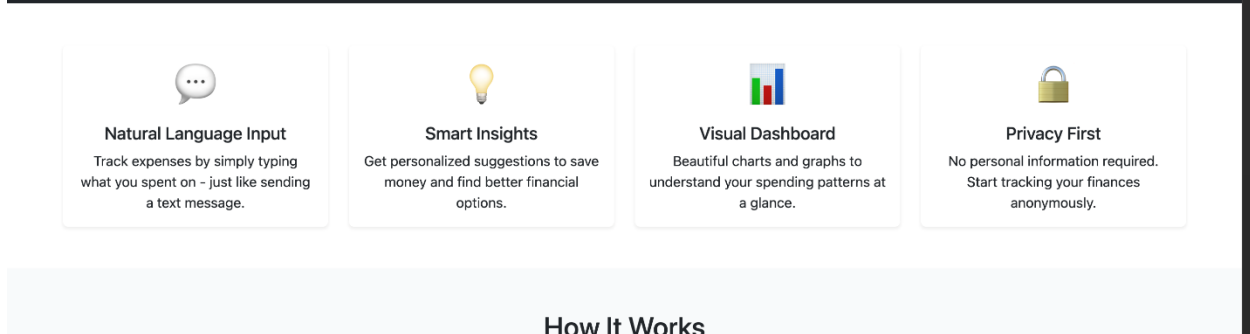
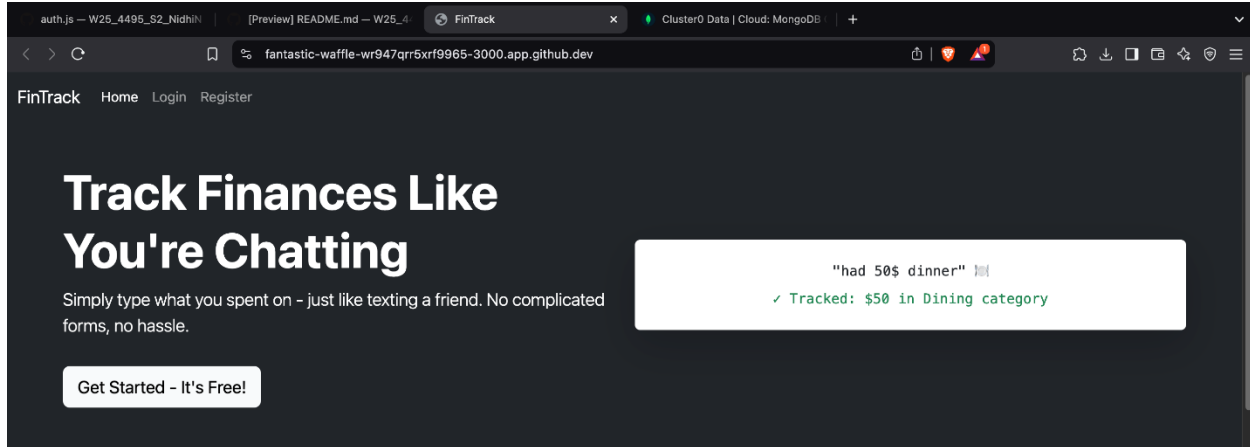
Enter your email [Subscribe](#)

Code sample:

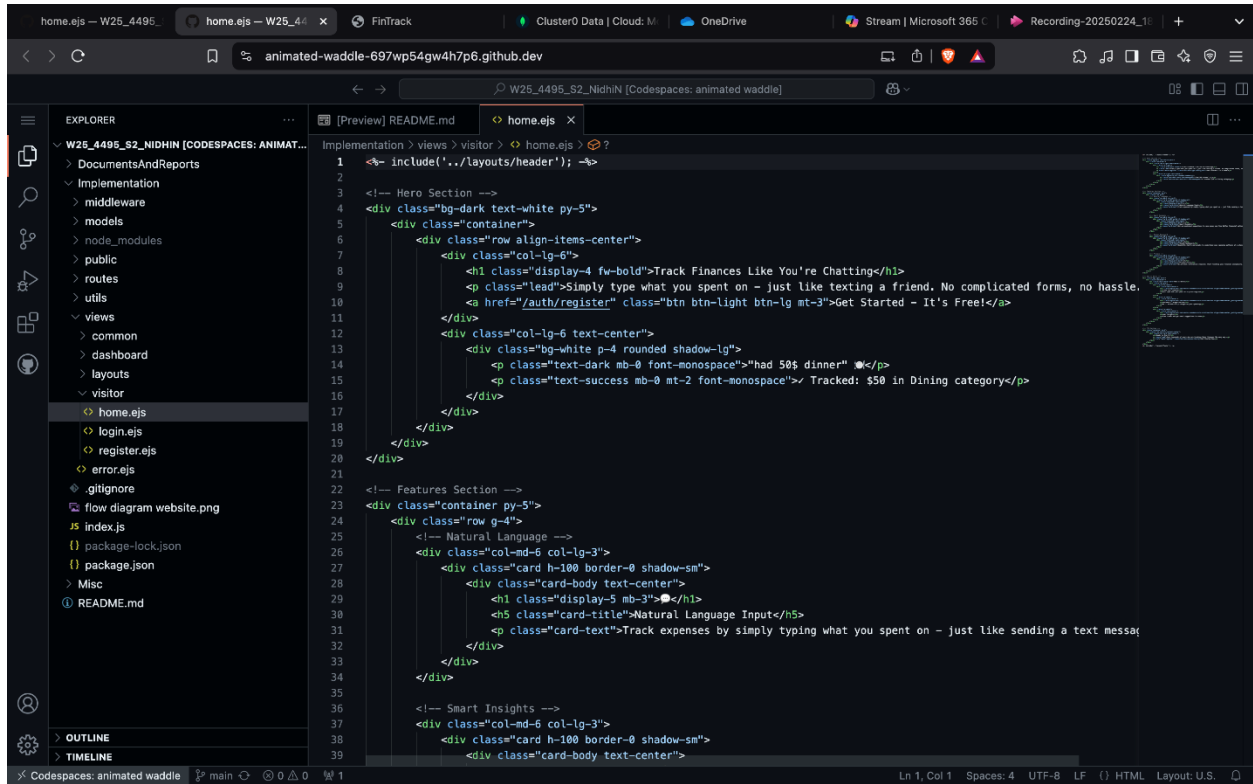


```
9 router.get('/register', (req, res) => {
15   })
16
17 router.get('/login', (req, res) => {
18   const token = req.cookies['u-xarh'];
19   if (token) {
20     res.redirect("/dashboard");
21   }
22   res.render('visitor/login.ejs');
23 })
24
25
26 // User registration
27 router.post('/register', async (req, res) => {
28   try {
29     const { username, password, firstname, lastname, email } = req.body;
30
31     const hashedPassword = await bcrypt.hash(password, 10);
32     const user = new User(
33       {
34         username,
35         password: hashedPassword,
36         firstname,
37         lastname,
38         email
39       }
40     );
41     await user.save();
42     res.status(201).json({ message: 'User registered successfully' });
43   } catch (error) {
44     console.log(error);
45     res.render("common/error.ejs", { message: 'Registration failed' });
46   }
47 });
48 // User login
```

Home Page:



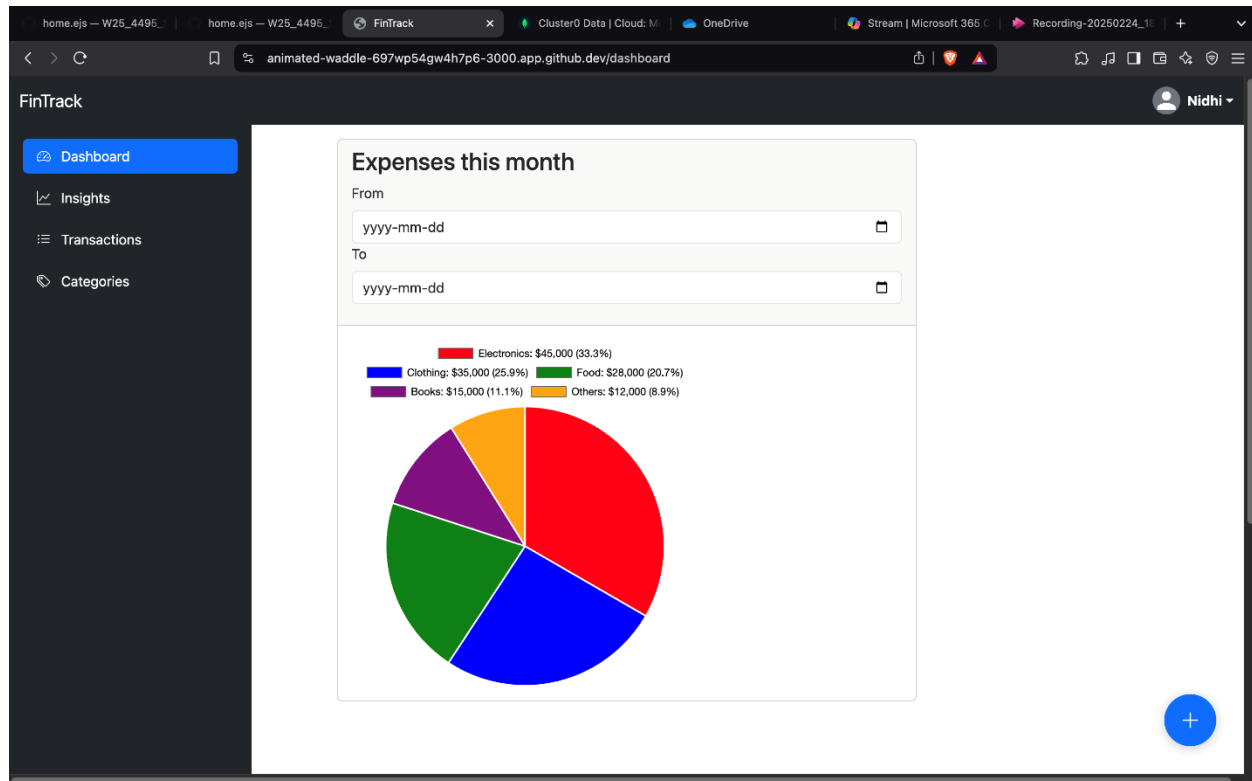
Code sample:



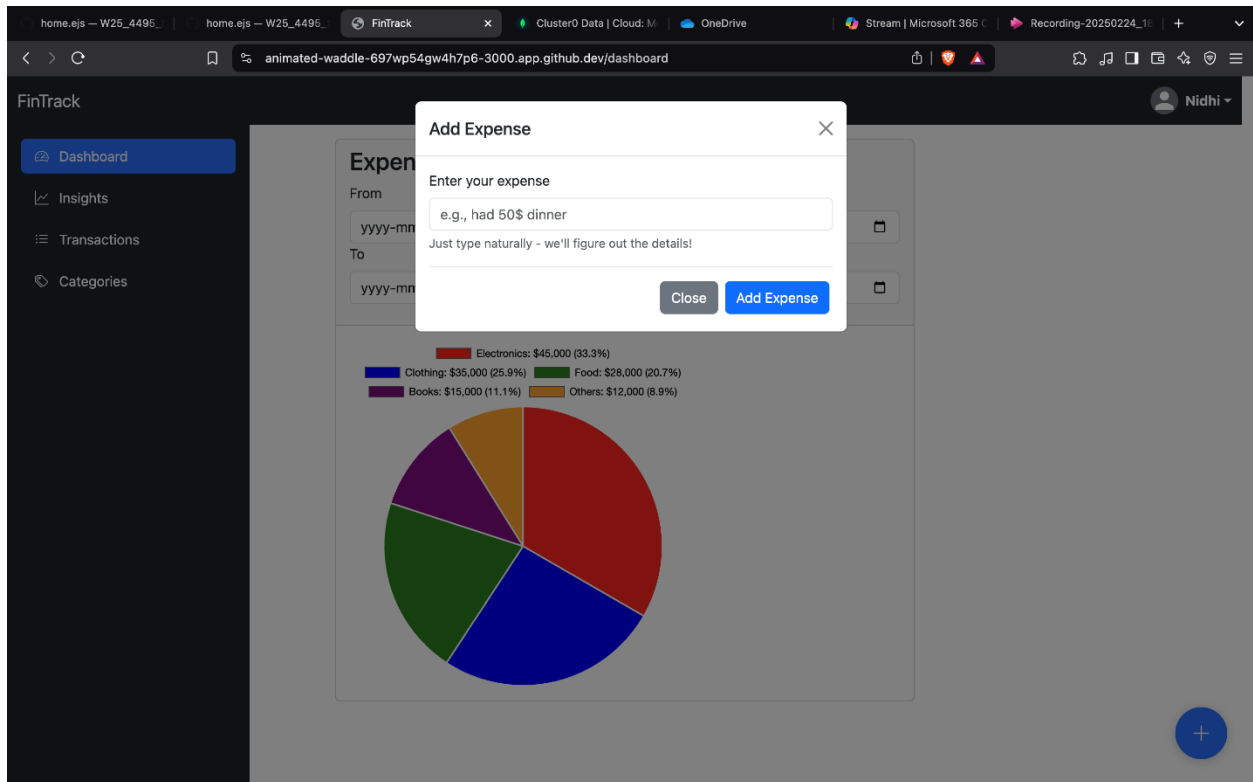
```
1 <!-- include('../layouts/header'); -->
2
3 <!-- Hero Section -->
4 <div class="bg-dark text-white py-5">
5   <div class="container">
6     <div class="row align-items-center">
7       <div class="col-lg-6">
8         <h1 class="display-4 fw-bold">Track Finances Like You're Chatting</h1>
9         <p class="lead">Simply type what you spent on - just like texting a friend. No complicated forms, no hassle.
10        <a href="/auth/register" class="btn btn-light btn-lg mt-3">Get Started - It's Free!</a>
11      </div>
12      <div class="col-lg-6 text-center">
13        <div class="bg-white p-4 rounded shadow-lg">
14          <p class="text-dark mb-0 font-monospace">"had 50$ dinner" 🍷</p>
15          <p class="text-success mb-0 mt-2 font-monospace">✓ Tracked: $50 in Dining category</p>
16        </div>
17      </div>
18    </div>
19  </div>
20
21 <!-- Features Section -->
22 <div class="container py-5">
23   <div class="row g-4">
24     <!-- Natural Language -->
25     <div class="col-md-6 col-lg-3">
26       <div class="card h-100 border-0 shadow-sm">
27         <div class="card-body text-center">
28           <h1 class="display-5 mb-3">🍷</h1>
29           <h5 class="card-title">Natural Language Input</h5>
30           <p class="card-text">Track expenses by simply typing what you spent on - just like sending a text message</p>
31         </div>
32       </div>
33     </div>
34
35     <!-- Smart Insights -->
36     <div class="col-md-6 col-lg-3">
37       <div class="card h-100 border-0 shadow-sm">
38         <div class="card-body text-center">
39
```

Dashboard design

For logged user:



Expense Visualization



Code sample:

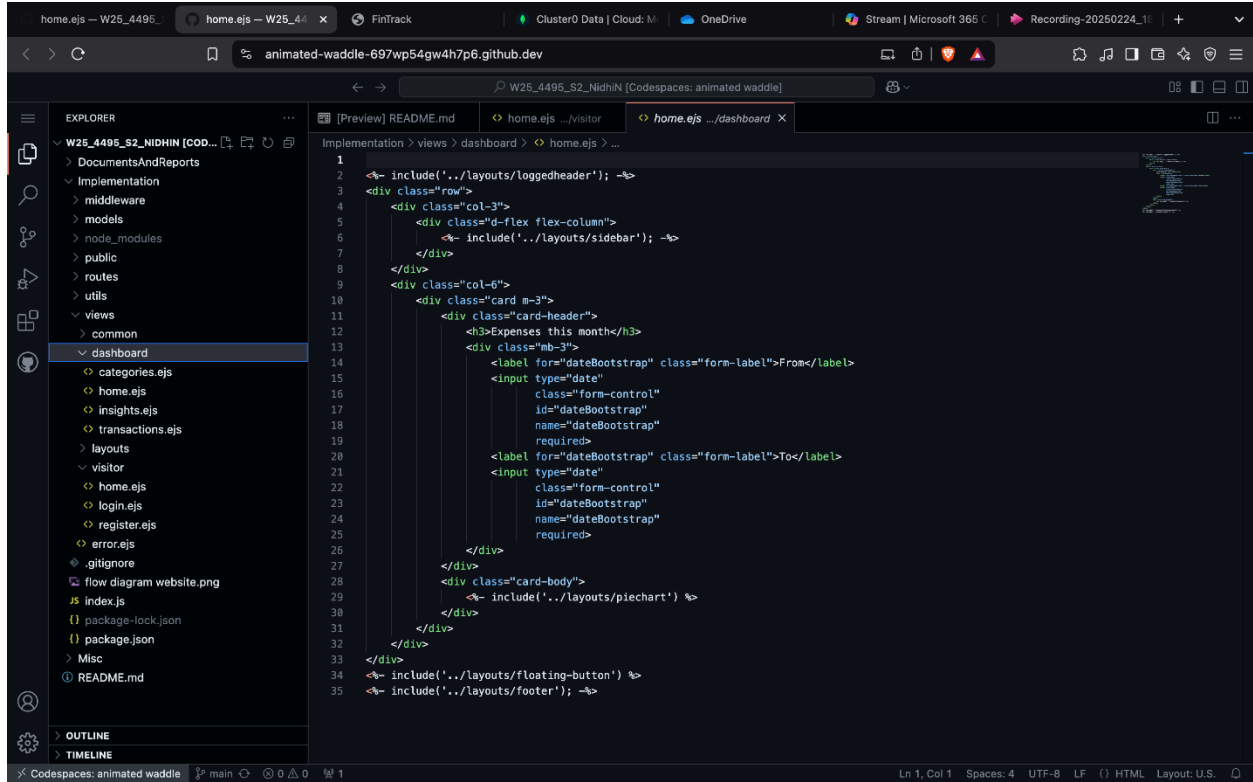
The first screenshot shows the VS Code editor with the file explorer on the left, displaying the project structure. The file explorer is expanded to show the 'floating-button.ejs' file. The main editor area shows the code for 'floating-button.ejs', which includes a button and a modal for adding an expense.

```
1 <!-- floating-button.ejs -->
2 <button type="button" class="btn btn-primary rounded-circle position-fixed d-flex align-items-center justify-content-center"
3 style="bottom: 2rem; right: 2rem; width: 60px; height: 60px; box-shadow: 0 2px 12px rgba(0,0,0,0.2);"
4 data-bs-toggle="modal"
5 data-bs-target="#addExpenseModal">
6 <i class="bi bi-plus-lg fs-4"></i>
7 </button>
8
9 <!-- Modal for Adding Expense -->
10 <div class="modal fade" id="addExpenseModal" tabindex="-1" aria-labelledby="addExpenseModalLabel" aria-hidden="true">
11 <div class="modal-dialog">
12 <div class="modal-content">
13 <div class="modal-header">
14 <h5 class="modal-title" id="addExpenseModalLabel">Add Expense</h5>
15 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
16 </div>
17 <div class="modal-body">
18 <form id="expenseForm" action="/add-expense" method="POST">
19 <div class="mb-3">
20 <label for="expenseInput" class="form-label">Enter your expense</label>
21 <input type="text" class="form-control" id="expenseInput" name="expense"
22 placeholder="e.g., had 50$ dinner" required>
23 </div>
24 <div class="form-text">Just type naturally - we'll figure out the details!</div>
25 </div>
26 <div class="modal-footer px-0 pb-0">
27 <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
28 <button type="submit" class="btn btn-primary">Add Expense</button>
29 </div>
30 </form>
31 </div>
32 </div>
33 </div>
```

The second screenshot shows the VS Code editor with the file explorer on the left, displaying the project structure. The file explorer is expanded to show the 'piechart.ejs' file. The main editor area shows the code for 'piechart.ejs', which includes a chart container and a script to render a pie chart.

```
1
2 <!-- Chart container -->
3 <div style="width: 400px; height: 400px;">
4 <canvas id="pieChart"></canvas>
5 </div>
6
7 <script>
8 let all_data = <%= JSON.stringify(c_data) %>;
9 const labels = all_data.labels;
10 const values = all_data.values;
11
12 const barColors = [
13 "red",
14 "blue",
15 "green",
16 "purple",
17 "orange",
18 "yellow",
19 "brown",
20 "pink",
21 "grey",
22 "cyan"
23 ];
24
25 new Chart("pieChart", {
26 type: "pie",
27 data: {
28 labels: labels,
29 datasets: [{
30 backgroundColor: barColors,
31 data: values
32 }]
33 },
34 options: {
35 plugins: {
36 legend: {
37 position: 'top',
38 labels: {
39 // Customize the legend labels
40 }
41 }
42 }
43 }
44 });
```

Code sample:



The screenshot shows a VS Code editor interface with a dark theme. The Explorer panel on the left displays a file tree for a project named 'W25_4495_S2_NIDHIN'. The 'dashboard' folder is selected, showing files like 'categories.ejs', 'home.ejs', 'insights.ejs', 'transactions.ejs', 'layouts', 'visitor', 'home.ejs', 'login.ejs', 'register.ejs', 'error.ejs', and 'gitignore'. The main editor area shows the content of 'home.ejs', which is an EJS template. The template includes a header, a sidebar, a main content area with a card for 'Expenses this month', and a footer. The card contains a form with two date inputs labeled 'From' and 'To'. The status bar at the bottom indicates the file is 'home.ejs' in the 'main' branch, with 0 changes and 1 line selected.

```
1  <%= include('../layouts/loggedheader'); %>
2  <div class="row">
3    <div class="col-3">
4      <div class="d-flex flex-column">
5        <%= include('../layouts/sidebar'); %>
6      </div>
7    </div>
8    <div class="col-6">
9      <div class="card m-3">
10       <div class="card-header">
11         <h3>Expenses this month</h3>
12       </div>
13       <div class="mb-3">
14         <label for="dateBootstrap" class="form-label">From</label>
15         <input type="date"
16               class="form-control"
17               id="dateBootstrap"
18               name="dateBootstrap"
19               required>
20         <label for="dateBootstrap" class="form-label">To</label>
21         <input type="date"
22               class="form-control"
23               id="dateBootstrap"
24               name="dateBootstrap"
25               required>
26       </div>
27     </div>
28     <div class="card-body">
29       <%= include('../layouts/piechart') %>
30     </div>
31   </div>
32 </div>
33 </div>
34 <%= include('../layouts/floating-button') %>
35 <%= include('../layouts/footer'); %>
```

Work Date/Hours Log

Student Name: Nidhi Nidhi

Date	Number of Hours	Description of work done
Feb 8, 2025	2	Roughly designed the sign up, login, dashboard and home pages on a paper.
Feb 12, 2025	3	Planned the design of home page and implemented it. Went through the documentation and started creating a basic home page. Later pivoted to another plan, changed the design and created different view for a registered user and a visitor.
Feb 15, 2025	3	This week, I worked on creating different pages for the app. Initially, I kept all pages in one folder but later on when pages start increasing I thought of creating multiple folders. For e.g, I kept different pages for a logged user and different pages for a visitor who just views the website.
Feb 16, 2025	2	Created Basic navbar & login page. Referred bootstrap documentation. Took lot of hit and trial while formatting CSS for navbar. Code checked in the repo -> Implementation -> views -> layouts
Feb 18, 2025	2	For authentication, I planned to use Passportjs library. However, it's a bit complicated for my use case where it's just login with username and password. Hence, decided to create my own authentication mechanism.
Feb 20, 2025	3	Tried connecting to MongoDB cluster that I created with mongodb as a service. It was failing to connect. Finally figured out that by default all ip addresses are blocked to access the database. Opened the database to receive traffic from any ip address for now.
Feb 21, 2025	3	This week, I updated completed the Login and register pages. Code checked in the repo -> Implementation -> views -> visitor. Started writing Midterm Report.
Feb 22, 2025	4	Created footer and add expense floating button. Rectified errors and fine tuned the code of sidebar. Code checked in the repo -> Implementation -> views -> layouts. Updated the Midterm Report.
Feb 23, 2025	3	Created dashboard template that breakdown your expenses into different categories and shows it in a nice pie chart. Used chart.js library for generating these charts. However, the data is hardcoded for now and will connect to database later.
Feb 24, 2025	2	Final update on MidTerm Report.

Closing and References

- MongoDB Documentation. (2023). <https://www.mongodb.com/docs/>
- Node.js Documentation. <https://nodejs.org/>
- Express.js Documentation. <https://expressjs.com/>
- Bootstrap Documentation. <https://getbootstrap.com/>
- EJS Documentation. <https://ejs.co/>
- Chart.js Documentation. <https://www.chartjs.org/docs/latest/>

- Bcrypt Documentation. <https://github.com/kelektiv/node.bcrypt.js/>
- Stack Overflow <https://stackoverflow.com/>
- Passport.js <https://www.passportjs.org/docs/>