Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – V

Subject: (CE – 520) Advanced Technologies

Project Title: Personal Project Management Tool

By:
Dhruv Dabhi
Roll No. CE021
ID: 21CEUSG042
Guided by: Sidhharth P. Shah Sir

Abstract

Second Brain, your indispensable life companion designed to simplify the lives of computer science students in India. It's not just an app; it's your ally, streamlining the intricacies of college, finance, and health into a seamless experience.

Built on the foundation of PARA (Projects, Areas, Resources, and Archives), Second Brain offers a straightforward approach to managing your life. Each 'Area' serves as a canvas where you map out your college journey, financial goals, and health priorities. 'Projects' within these 'Areas' are your essential tasks, such as saving a specific amount each month or conquering a challenging exam.

But there's more! Second Brain recognizes that every 'Project' needs resources to thrive. These resources are your tools, whether it's well-organized study material or financial planning aids. With MERN Second Brain, everything you need is just a click away.

The beauty of the system: completed 'Projects' elegantly shift to the 'Archives,' neatly organized but always accessible. Like a cherished cookbook with secret recipes, these archived 'Projects' are there when you need them.

Say goodbye to the chaos of scattered notes and never-ending to-do lists. Embrace the clarity and simplicity that Second Brain brings to your life. It's more than an app; it's your friend who understands your daily rhythm, the challenges of exams, and the importance of saving a few extra rupees.

MERN Second Brain - Your personal life management companion, simplifying your life as effortlessly as sipping chai with college buddies.

Introduction

Purpose

Second Brain is your digital life partner, designed to simplify your daily grind. It's all about making life easier for you, especially if you're a computer science student in India. This app helps you keep your college, finance, and health in order. You can break them down into 'Projects,' like saving money or acing an exam. Plus, it keeps your study materials and tools handy. When you're done with a task, it neatly stores it in 'Archives' for later. No more scattered notes and endless lists. It's your buddy, ensuring you stay on track while enjoying college life with a cup of chai.

Overall Description

Second Brain is your trusty digital companion, tailor-made for everyone who is seeking a fuss-free way to manage life. It's here to make your life easy and organized, without any complications.

Imagine your life divided into three main sections: college, finances, and health. These are the big parts, the ones that matter. Now, within each of these sections, you've got your smaller goals and tasks – we call them 'Projects.' It could be as simple as saving money every month or as challenging as preparing for an important exam.

Here's where MERN Second Brain steps in. It helps you keep these projects neat and tidy. You can even write your notes and plans using 'markdown,' a straightforward way to put your thoughts down on a digital piece of paper.

And when you've ticked off a task or completed a project, MERN Second Brain doesn't clutter your view with it. Instead, it puts them away in the 'Archives' – kind of like putting your old things in a storage box in the attic. They're out of the way but not forgotten.

Plus, it's got a nifty feature. If you've got any PDF notes, you can download them easily. No more hunting for lost notes or flipping through pages.

MERN Second Brain is more than just an app; it's your buddy, your sidekick, helping you sail through life's ups and downs while you kick back with a cup of chai and enjoy your college journey. It's all about keeping it simple and making life a tad bit easier for you.

Tools / Technologies Used

The development of “Second Brain: Personal Life Management” involved a set of tools and technologies selected to learn those technologies and build something to I get benefitted from building it and also get hands on experience building a real life solution.

Development Stack:

- **Frontend:**
    - React/NextJS: Used as Core frontend framework to build interactive UI
    - HTML/Tailwind CSS: HTML used to build skeleton and Tailwind CSS as framework for CSS to create beautiful UI
    - Javascript: Used for client-side scripting
    - DaisyUI: for beautifully created components

- **Backend:**
    - Node.JS: Used as runtime for the backend server

    - Express.JS : A web application framework for Nodejs, simplifying server-side scripting
    - MongoDB: A NoSQL Database for storing and managing data.

Version Control

- Git/Github: used for version control and managing project

Code Editor

- NeoVim: Lightweight, mouse free editor

Software Requirement Specification (SRS)

Introduction

1.1 Purpose

The purpose of developing an online personal life management website is to provide users a tool that they can use to manage their life with ease and follow a simple management idea called PARA.

1.2 Intended audience and Reading Suggestions

The intended audience for this website is primarily users who are students, working professionals and basically anyone who wants to manage their life in the most simple way possible. This website is for all the people who have the internet and are comfortable putting their ideas and daily todos online.

1.3 Product Scope

A second brain can help you with remembering ideas, thoughts, deadlines, resources. Making life more efficient and convenient for users. It can help you with creative thinking because you don’t have to remember everything.

1.4 Reference

Fundamentals of Software Engineering by Rajib Mall, PHI Learning

Build a Second Brain, Tiago Forte

Overall Description

2.1 Product Perspective

There are many similar software like this,but the way this is different from others is that this website provides PARA Method built into it with great user experience and online storing support with Markdown editor.

2.2 Product Functions

Functionality of the System

User

- Login, Signup, Logout
- Able to Add, Update, Delete
    - Projects
        - Able to update, create, delete tasks
    - Areas
    - Resources
- Able to download Markdown notes
- Tracking of the Task remaining the Project
- Able to see projects which are in some areas.
- Able to search resources
- Restore archived resources

2.3 User Classes

User: user is expected to invoke functionality mentioned above, there are not any admin users to increase the security of the website.

2.4 Operating Environment

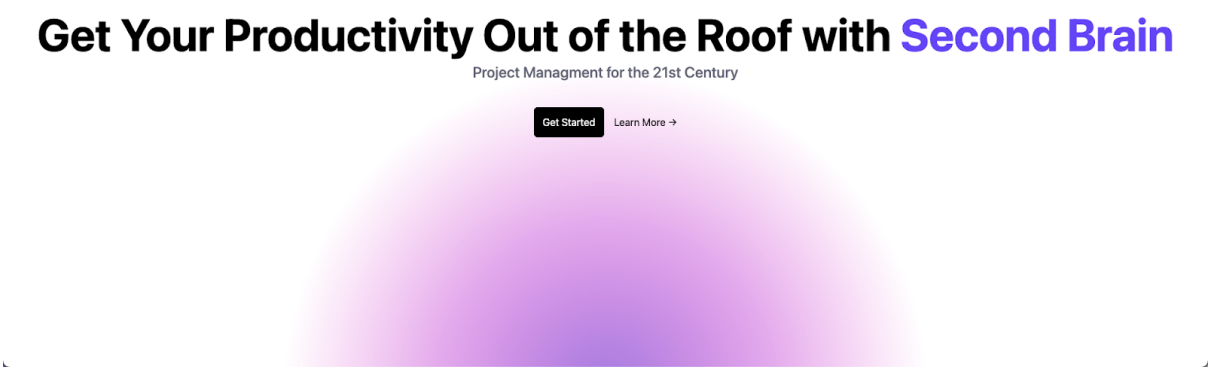
The system will be compatible with all popular web browsers, including chrome, firefox, safari, Brave and Arc. The machine should have sufficient RAM and Speed of the internet connection.

2.5 Assumptions and Dependencies

- Assumptions
    - The website will assume that users will have access to the internet and are comfortable sharing their todos, thoughts online.
    - The system will assume that users have a reliable internet connection.
    - The system will assume that users have a device that meets the minimum technical requirements, such as modern web browsers.
- Dependencies
    - The system depends on the availability of a reliable database.
    - The system depends on the availability of a stable and secure hosting environment to host the website.

External Interface Requirements

3.1 User Interface



3.2 Hardware Interfaces

The website is compatible with a wide range of hardware, including desktop computers, laptops, and mobile devices. The system will be able to run on popular operating systems such as windows, MacOs and Linux.
RAM: 4GB
Hard Drive Storage: 2GB
Other Hardware Requirements: None

3.3 Software Interface

The database connection interface is the part of the Second Brain that manages the communication and interaction between the website and the underlying database. The interface will validate and sanitize the user input before executing any database query to prevent SQL Injection attacks. For frontend we are using ReactJS and tailwind CSS.

3.4 Communication Interfaces

This uses standard network protocols, such as HTTPS. The system will comply with industry standards for data encryption and secure

communication to ensure that customer information is protected for unauthorized access or attack.

System Features

4.1 Functional Requirements

F1 User Authentication

F1.1 User Registration
Input: Enter Details in the form
Output: Home Page

F1.2 User Login
Input: Enter Details in the form
Output: Home Page

F1.3 User Logout
Input: Open the Profile and Click on logout button.
Output: Landing Page

F2 Using AREAs

F2.1 Create Area
Input: Go to Area Page and Enter Area Name.
Output : Area created on Area Page.

F2.2 Update Area
Input: Open Area Page, Click on Update, Change the properties you want to change.
Output: Area Page with Changed details

F2.3 Delete Area

Input: Click Update and Click Delete Area Button.
Output: Home Page

F3 Using PROJECTs

F3.1 Create Project
Input: Go to Project Page and Enter Project Name, Enter in which area you want to add your project.
Output : Project created on Projects Page.

F3.2 Update Project
Input: Open Project Page, Click on Update, Change the properties you want to change.
Output: Project Page with Changed details

F3.3 Delete Project
Input: Click Update and Click Delete Project Button.
Output: Home Page

F4 Using Resources

F4.1 Add Resources to the Project
Input: Open any Project Page, Click Add Resources and fill the form and submit
Output: Project Page with added resources.

F4.2 Update Resource
Input: Open Project Page, Click on Pencil icon near Resource, Change the properties you want to change.
Output: Project Page with Changed Resources

F4.3 Delete Resource
Input: Click Delete Resources Button.
Output: Project Page

F4.4 Download Notes
Input: Click on Download button.
Output: Markdown file gets downloaded

F4.5 Make Notes
Input: Click on Add Resource button, click on Add Markdown
Output: Empty textarea appears.

F5 Using Archives

F5.1 Add to Archive
Input: Click on Add to archive button to any project/area/resources
Output: Added to archives

F5.2 Remove from Archives
Input: Click on Delete Permanently in Archives page
Output: Things from the archive get deleted.

4.2 Non Functional Requirements

1. **Performance Requirements**
    - **a.** The system should have a response time of less than 2 for all user requests.
    - **b.** The system should be able to process at leasts 100 requests per minute.
2. **Safety Requirements**
    a. The system should clearly indicate to the user any potential risks associated with certain actions.
    b. The system should have a recovery process for handling errors and exceptions.
3. **Security Requirements**
    a. The system should use encryption to protect data, such as user passwords.

b. The system should have regular vulnerability assessments and penetration testing to identify and remediate any security weaknesses.

4. **Software Quality Attributes**
    a. The system should be highly available, with uptime of 99.5%.
    b. The system should be secure and protect against unauthorized access and data breaches.
    c. The system should be scalable and able to handle an increase in traffic and data storage.
5. **Design Requirements**
    a. The system should have a user-friendly interface that is easy to navigate and understand.

Database Design

UserSchema:

username: User's username (required).
email: User's email address (required).
password: User's password (required).
projects: List of user's projects.
areas: List of areas associated with the user.
resources: List of resources linked to the user.

ProjectSchema:

user: User associated with the project (required).
title: Title of the project (required).
description: Description of the project.
area: Area to which the project belongs (required).
todos: List of tasks within the project, including title, description, and completion status.
resources: Resources linked to the project.

archived: Indicates whether the project is archived (default is false).

AreaSchema:

user: User associated with the area (required).
title: Title of the area (required).
description: Description of the area.
projects: List of projects within the area.

ResourceSchema:

user: User associated with the resource (required).
title: Title of the resource (required).
description: Description of the resource.
type: Type of resource, either 'markdown' or 'link' (required).
markdownContent: Markdown text content (if type is 'markdown').
link: URL link (if type is 'link').
archived: Indicates whether the resource is archived (default is false).

Implementation Details

user.ts

createUser: This function handles the user registration process. It takes the user's provided information, like username, password, and email. It checks if a user with the same username or email already exists in the database. If not, it creates a new user with a secure password and saves the user's data. It then generates a token for the user, indicating successful registration.

signin: This function manages the user login process. It checks if the user with the provided username exists in the database. If the user is found, it verifies if the entered password matches the stored password. If everything checks out, it generates a token, signaling a successful login.

In both functions, if there are any errors during the registration or login process, it returns an error message and an appropriate status code.

Area.ts

createArea: This function handles the creation of a new area. It connects to the database, checks if the area already exists, and if not, it creates the area and links it to the user who created it. The function then returns a message indicating that the area has been successfully created.

getAreas: This function retrieves and lists all the areas created by a specific user. It connects to the database, fetches the user's areas, and sends them as a response for the user to view.

getArea: This function gets the details of a specific area by its unique identifier. It connects to the database, looks up the area, and if found, sends the area's details to the user. If the area doesn't exist, it returns a message stating that the area was not found.

updateArea:
This function allows a user to update the details of an existing area. It connects to the database, checks if the area exists, and if so, updates the title and description of the area as per the user's request. It then returns a message indicating that the area has been successfully updated.

deleteArea: This function handles the deletion of an area. It connects to the database, searches for the area, and if found, deletes it. It also removes the area from the user's list of areas. The function then returns a message confirming the deletion of the area. If the area is not found, it returns a message stating that the area was not found.

project.ts

createProject: This function handles the creation of a new project. It connects to the database, checks if the project with the same title already exists, and if not, it creates the project. It then associates this project with a specific area. The function returns a message indicating the successful creation of the project.

getProjectsByArea: This function retrieves and lists all the projects within a specific area. It connects to the database, looks up the area, and if found, it retrieves all the projects associated with that area. If the area is not found, it returns a message stating that the area was not found.

getProject: This function fetches the details of a specific project by its unique identifier. It connects to the database, looks up the project, and if found, sends the project's details to the user. If the project doesn't exist, it returns a message stating that the project was not found.
updateProject: This function allows users to update the details of an existing project. It connects to the database, checks if the project exists, and if so, updates the project's information as per the user's request. It then returns a message confirming the successful update of the project.

deleteProject: This function handles the deletion of a project. It connects to the database, searches for the project, and if found, deletes it. It also removes the project from the associated area. The function then returns a message confirming the deletion of the project. If the project is not found, it returns a message stating that the project was not found.

Resource.ts

createResource: This function manages the creation of a new resource. It connects to the database, checks if a resource with the same title already exists, and if not, it creates the resource. It also associates this resource with a specific project. The function then returns a message indicating the successful creation of the resource.

getResourcesByProject: This function retrieves and lists all the resources within a specific project. It connects to the database, looks up the project, and if found, it retrieves all the resources associated with that project. If the project is not found, it returns a message stating that the project was not found.

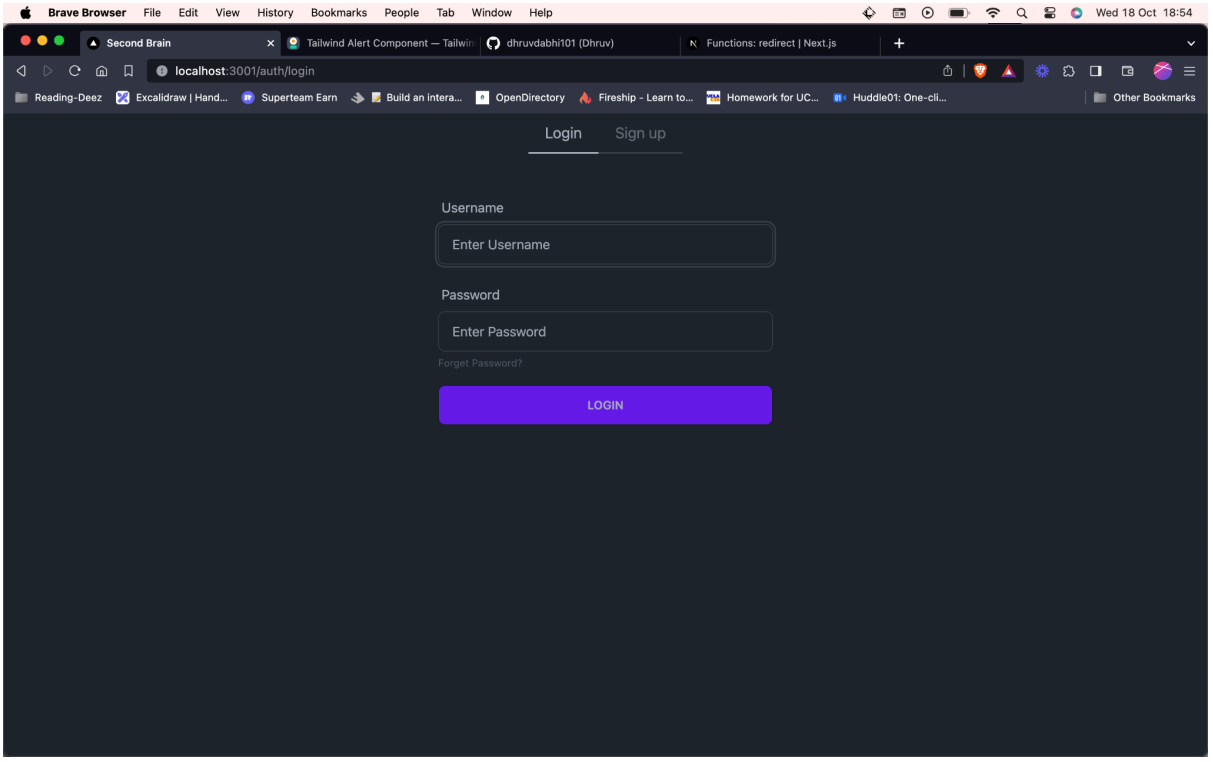
getResource: This function fetches the details of a specific resource by its unique identifier. It connects to the database, looks up the resource, and if found, sends the resource's details to the user. If the resource doesn't exist, it returns a message stating that the resource was not found.

updateResource: This function allows users to update the details of an existing resource. It connects to the database, checks if the resource exists, and if so, updates the resource's information as per the user's request. It then returns a message confirming the successful update of the resource.

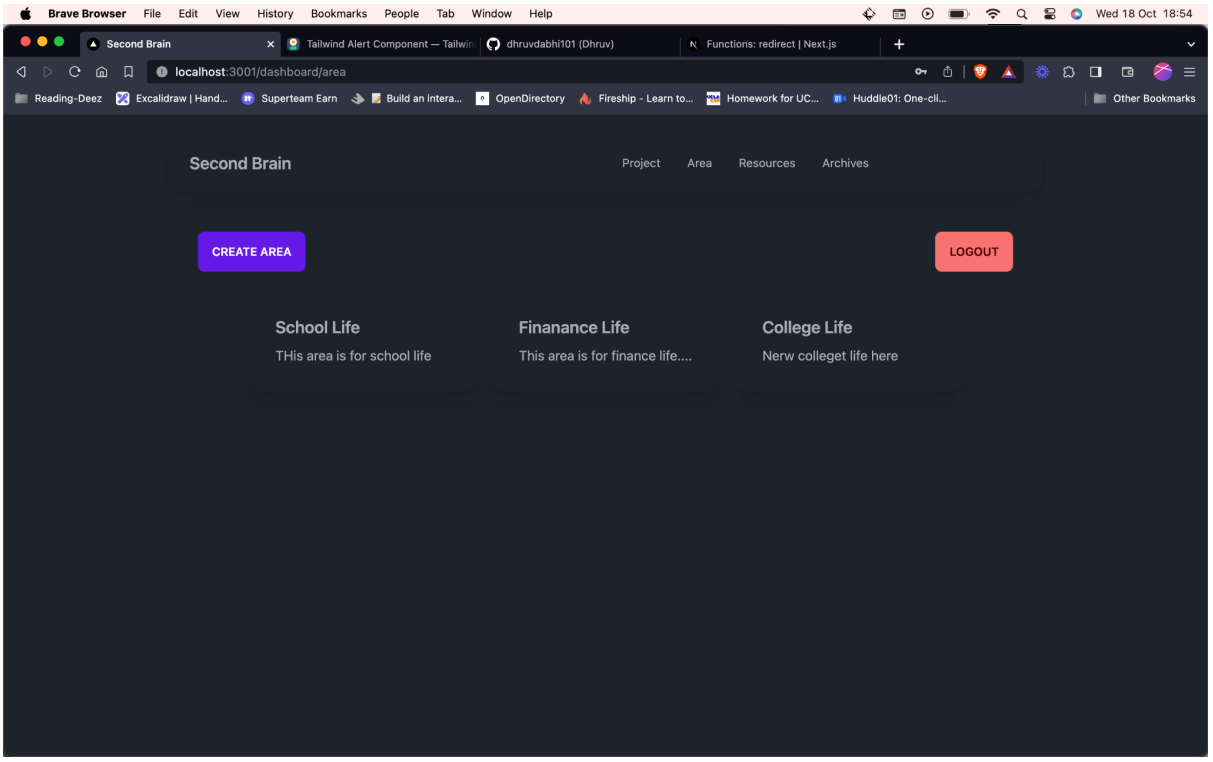
deleteResource: This function handles the deletion of a resource. It connects to the database, searches for the resource, and if found, deletes it. It also removes the resource from the associated project. The function then returns a message confirming the deletion of the resource. If the resource is not found, it returns a message stating that the resource was not found.

Screenshots

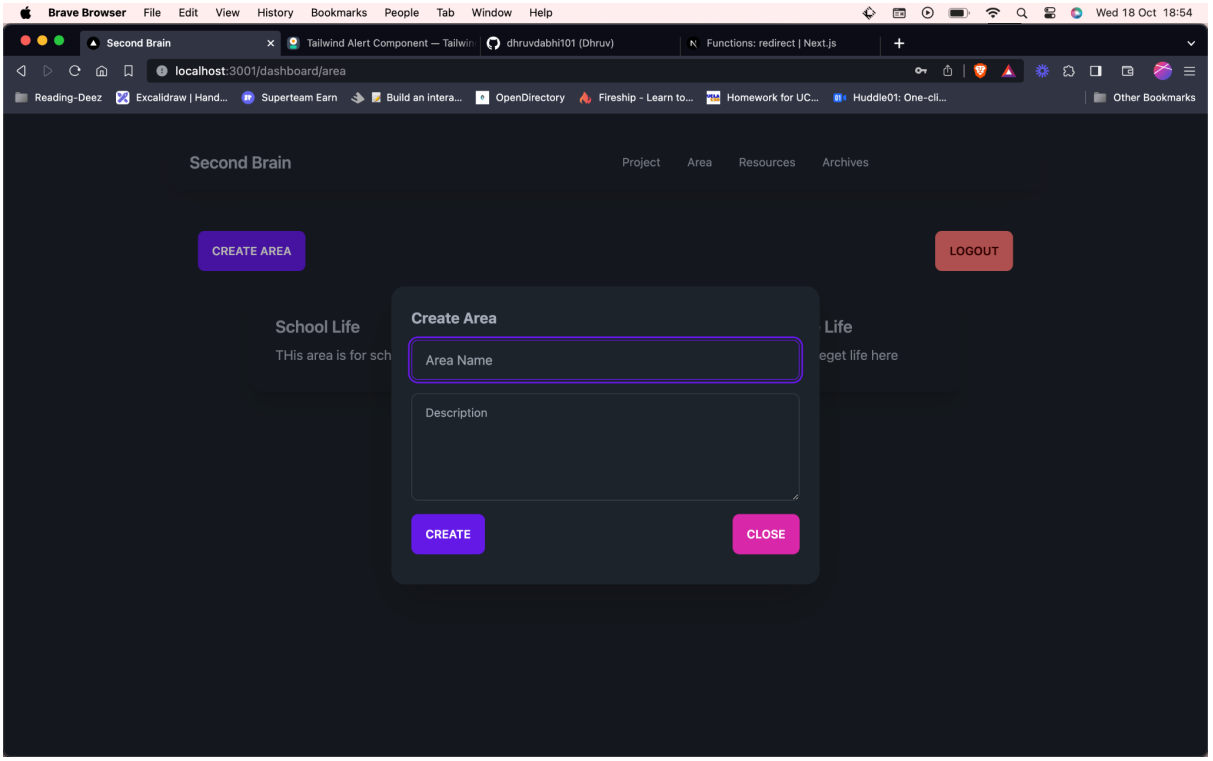
User Authentication



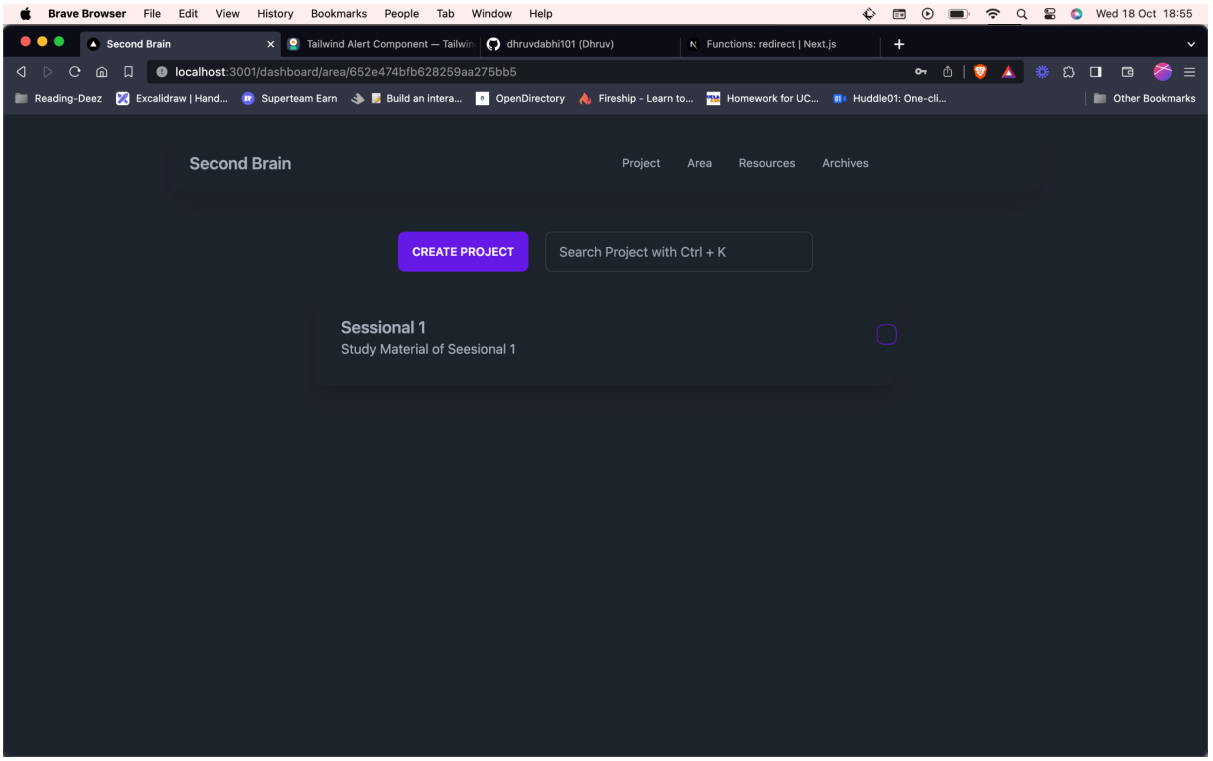
Area List



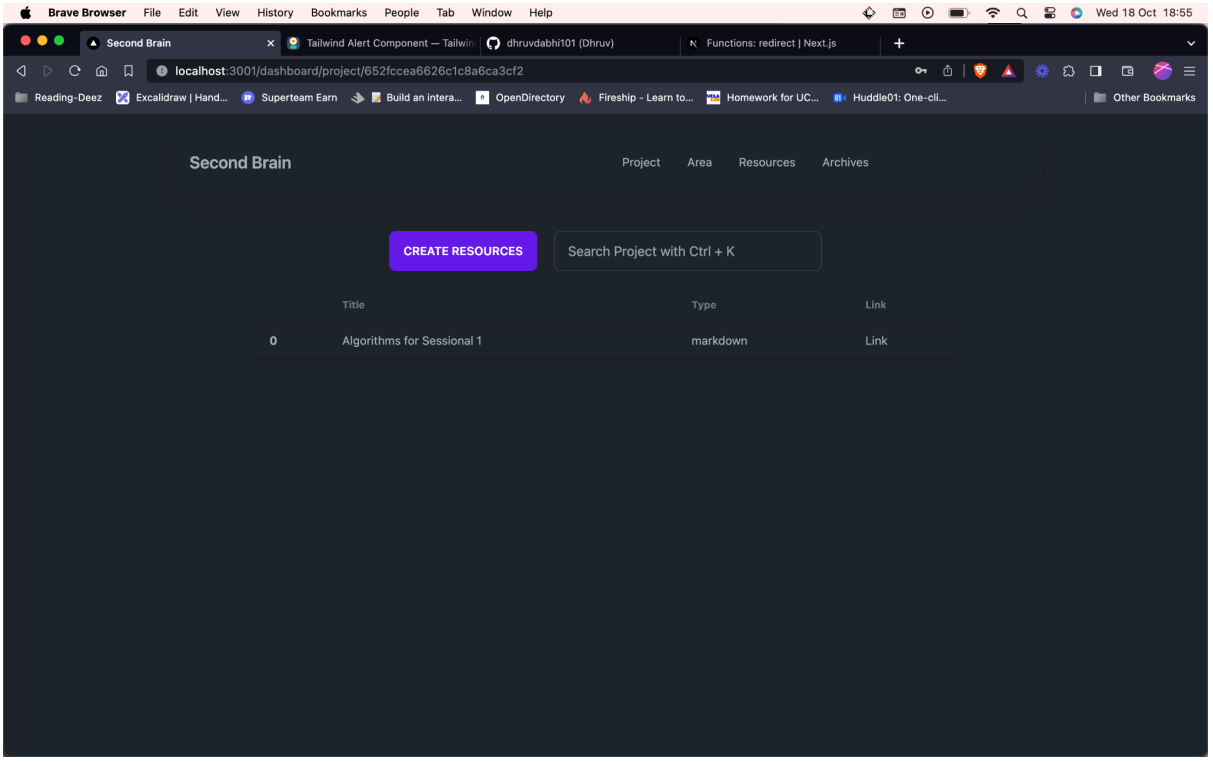
Creating Area



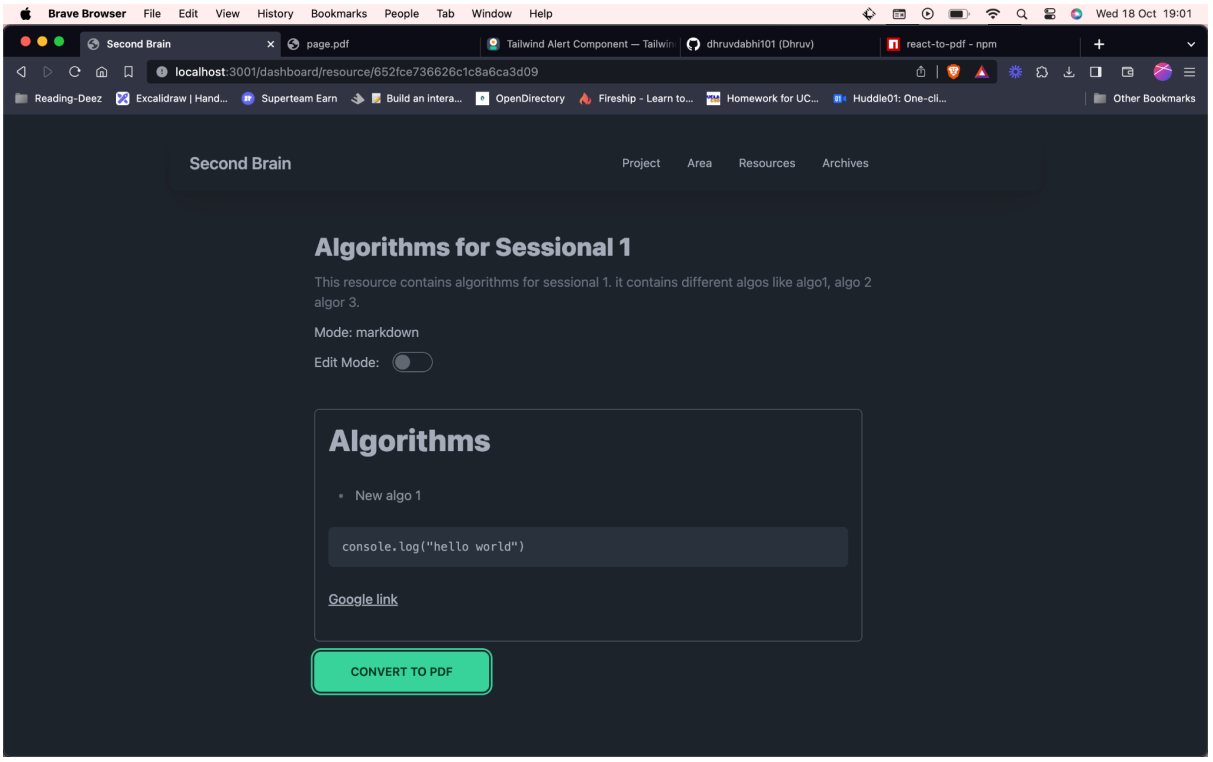
Project List



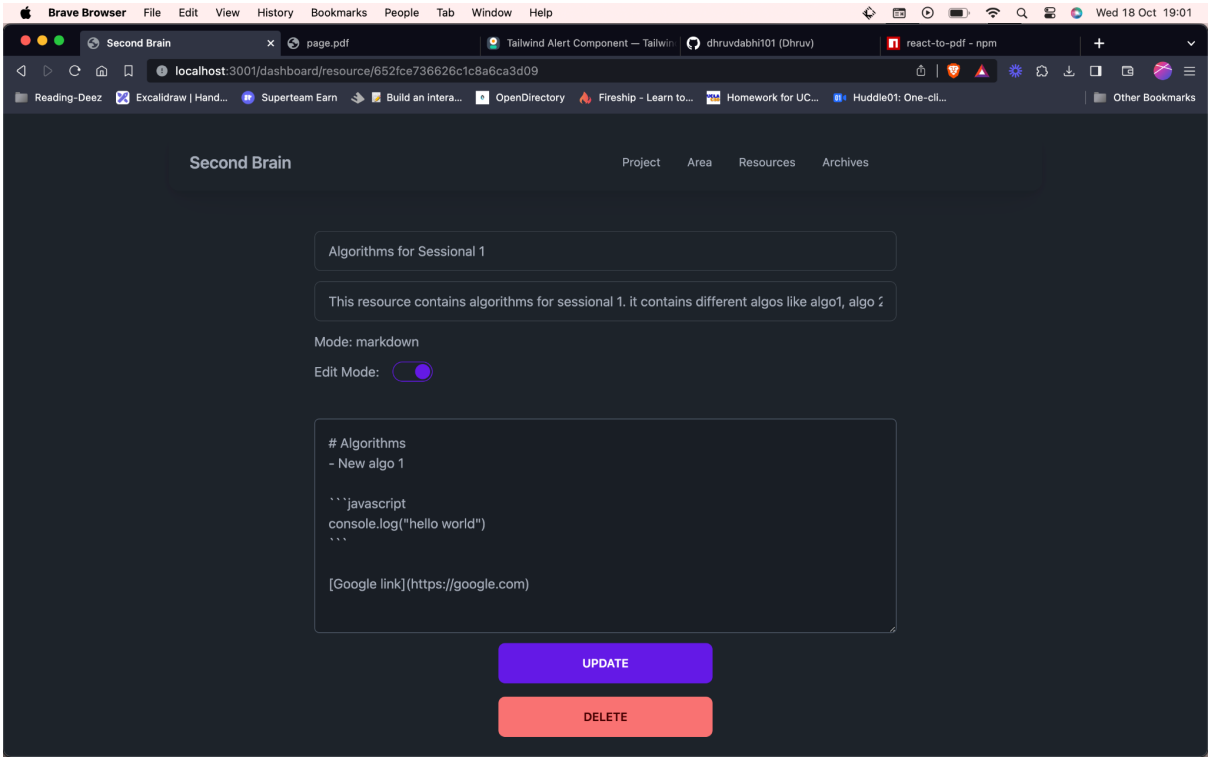
Resource List



Markdown Viewer



Markdown Editing



Conclusion

The simple conclusion we can derive from this project is to make life easy and organized. It's like a friend that helps us manage everyday concerns like college, finances, and health. It keeps everything in one place, like notes and tasks, so we can avoid unnecessary hassles. This project enables us to take a step towards happiness and makes life simpler.

Limitation and Future Extensions

Limitation

Every project has boundaries, so does Second Brain.

1. Only one markup language support: we can add multiple markup language support but keeping time constraint in mind we only integrated one.
2. Image Upload: For now, this project doesn’t let user upload their own images as resources.

Future Extensions

We should always think further. We plan to further enhance second brain in future with these changes

1. With Google Calendar Integration: with google calendar you can get reminders of your projects
2. Adding todos in projects: every project has some todos, it will be convenient to get your todos with project as well.
3. Sending your projects to someone else: We further plan to give user a link that they can share with others so others can also see the projects as web pages.

Bibliography:

- Stackoverflow used for major error solving
- Tailwindcss for making web app UI. Available at : https://tailwindui.co
- DaisyUI: used for adding already created components or HTML Tags with better UI, available at https://daisyui.com/
- MongoDB Docs : https://www.mongodb.com/docs/
- Nodejs Docs: https://nodejs.org/en/docs
- Express JS Docs: https://expressjs.com/