DV200 Progress Check

Kurt Schwimmbacher 231002

Open Window, School of Fundamentals

DV200

Tsungai Katsuro

28 October 2024

Table of Contents

Functional Requirements Checklist

- **Home Page:**
    - Show Features on the website with links to the corresponding page ✔
- **User Authentication:**
    - Log In ✔
    - Sign Up ✔
    - Log Out ✔
    - Default profile picture ✔
- **Journal Page:**
    - Create ✔

      Add a new journal entry for the current day
    - Read ✔

      Show all previous users' journal entries on the calendar
    - Update ✔

      Edit information of journal entry: including title, content and date
    - Delete ✔

      Remove journal entry
    - Require Login for CRUD on journal ✔

      Only allow users to interact with journal entries when account is created and logged in
- **Resources :**
    - Create (on account page – admin only) ✔

      Admins create resources with articles/videos/tutorials for users to read

- Read ✔

  Display all resources on the resource page, no account is needed to read
- Update (on account page – admin only) ✔

  Admins can edit resource titles, content, links, and tags if needed
- Delete (on account page – admin only) ✔

  Admins can remove resources
- Single Resource Page

  Show full resource details when a resource is clicked
- ***FRONTEND:***
  - Filter ✔

    Show resources that have tags matching users' input
  - Sort ✔

    Sort resources by criteria
  - Search ✔

    Find specific resources by title or content

- **Forum:**
  - Create (on the account page) ✔

    To create new posts, the user has to be logged in
  - Read ✔

    Display all posts on forums, no account is needed
  - Update (on account page) ✔

    Users can make changes to their post titles, content, and tags
  - Delete (on account page) ✔

    Users can remove posts they've made
  - ***FRONTEND:***
    - Filter ✔

      Show tags that have tags matching users' input

        - Sort ✔
          Sort resources by criteria
        - Search ✔
          Find specific resources by title or content

- **Professionals:**
    - Create (admin only) ✔
      Admins can create new professionals, this ensures that there is an enrollment process
    - Read ✔
      Show Professionals on the chat page with professionals' credentials and bio, with the option to start messaging a professional
    - Update (admin only) ✔
      Admins can edit the professional names, speciality, availability, and bio
    - Delete (admin only)
      Admins can remove professionals from the chat page

- **Chats:**
    - Create ✔
      Users can choose to message professionals, doing so requires a chat
    - Read ✔
      Show all messages between specific users and professional
    - Update
      Users and professionals can edit messages sent
    - Delete
      Users and professionals can delete messages sent
    - Replies from professionals
      Show replies from professionals

- **Account Page:**
  - Show user information ✔

    Show username and profile picture
  - Show posts by user ✔

    Show the number of posts made by users
  - Show resources by admin (admin only) ✔

    Show all the resources made by an admin
  - Logout Button ✔

    Button to log out of account
  - Update profile picture ✔

- **Responsiveness:**
  - Mobile ✔
  - Tablet ✔
  - Desktop ✔

System Documentation

- **Technical Architecture:**
  - MERN Stack
  - ***Components:***
    - Navigation Bar

      Does som
- **Database:**
  - Has 6 Collections:
    - Chats:
      - ID

    - Sender (user obj)
    - Receiver (user obj)
    - Message (String)
- JournalEntries:
    - ID
    - Title (String)
    - Date (Date)
    - Content (String)
    - User (user obj)
- Posts:
    - ID
    - Title (String)
    - Content (String)
    - User (User Obj)
    - createdAt (date)
    - updatedAt (date)
    - Tags ([String])
- Resources:
    - ID
    - Title (String)
    - Content (String)
    - User (User Obj)
    - createdAt (date)
    - updatedAt (date)
    - Tags ([String])
- Users:
    - Username (String)
    - Email (String)

        - Password (Hashed String)
        - Role (“Admin” or “User”)
    - Professional:
        - Name (String)
        - Email (String)
        - Speciality (String)
        - Availability (String)
        - Bio (String)
        - User (User obj)

- **API Endpoints:**
    - ***Post Routes:***
        - get(‘/) - fetches all posts
        - get(‘/user/:userID’) - fetches all posts by one user, using userID
        - post(‘/create’) - creates a new post
        - patch(‘/update/:id’) - edit a post by post ID
        - delete(‘/delete/:id’) - delete a post by post ID
- ***User Routes:***
    - post(‘/signup’) - create a new user
    - post(‘/login’) - checks if the user exists to log in user

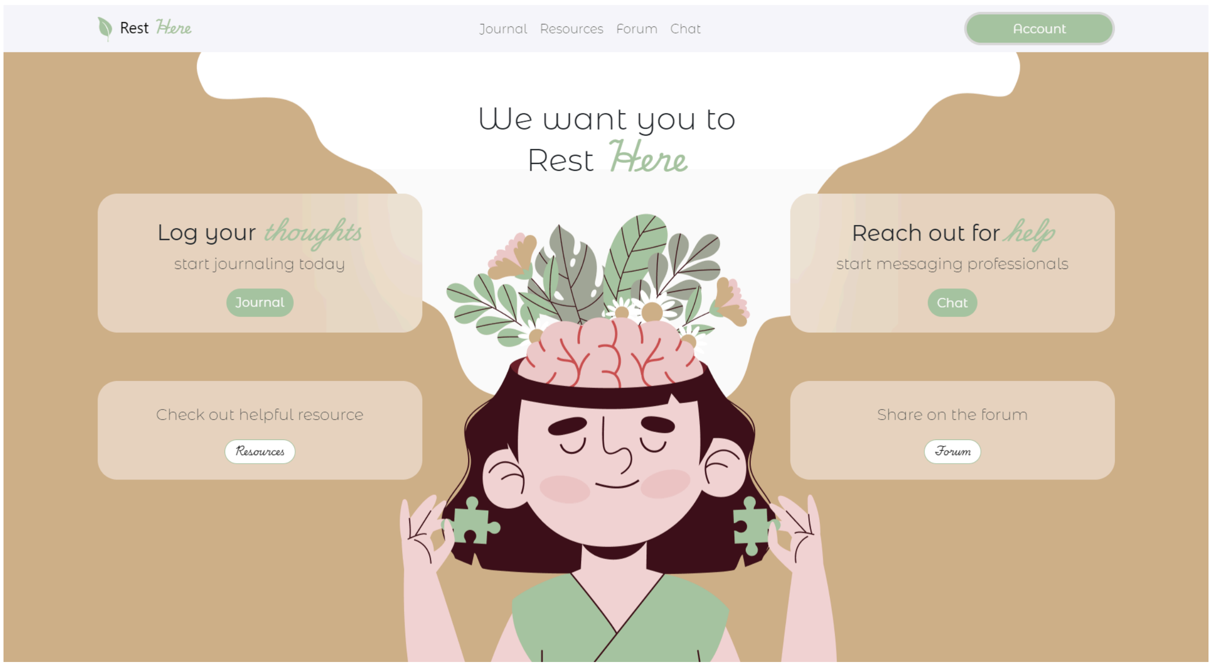
- ***Journal Routes:***
    - post(‘/create’) - add a new journal entry
    - get(‘/entry/:entryID’) - fetch one entry by entry ID
    - patch(‘/update/:entryID’) - update an existing entry by entry ID
    - delete(‘/delete/:entryID’) - delete entry using entry ID

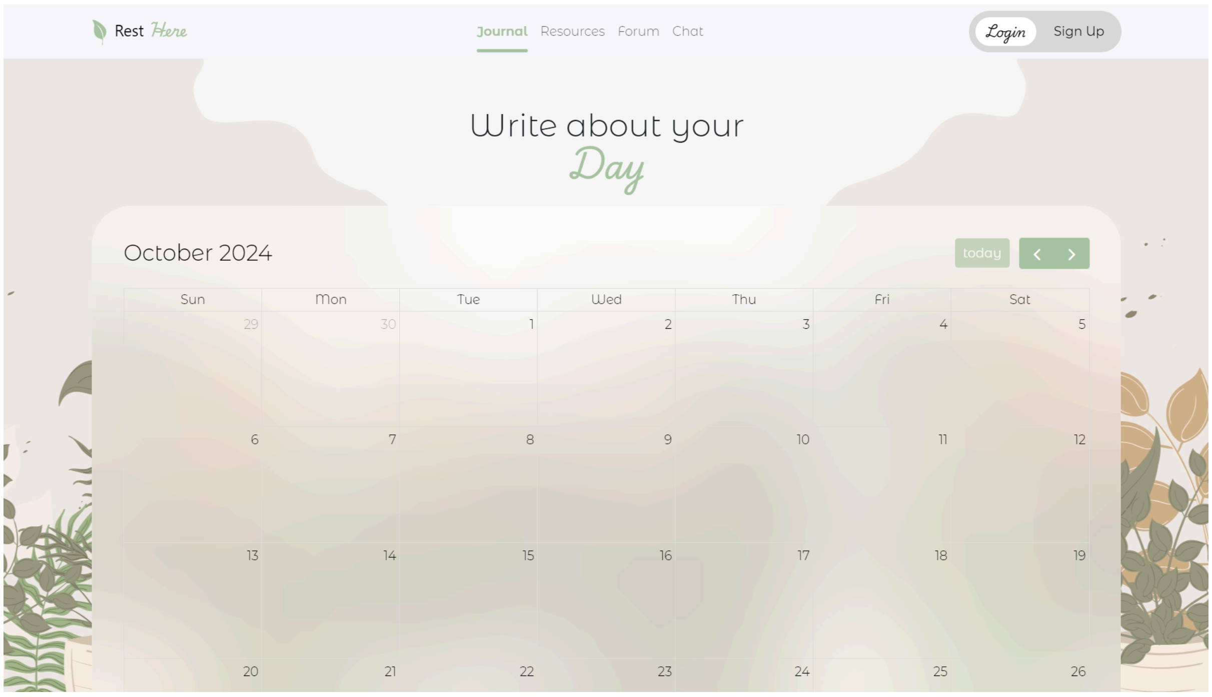
- ***Resource Routes:***
    - post('/create') - create a new resource
    - get('/') - get all resources
    - patch('/update/:id') - update an existing resource by resource ID
    - delete('/delete/:id') - delete a resource using resource ID

- ***Professional Routes:***
    - get('/') - get all professionals
    - post('/create') - create a new professional
    - patch('/update/:id') - update an existing professional using a professional ID
    - delete('/delete/:id') - delete a professional using a professional ID

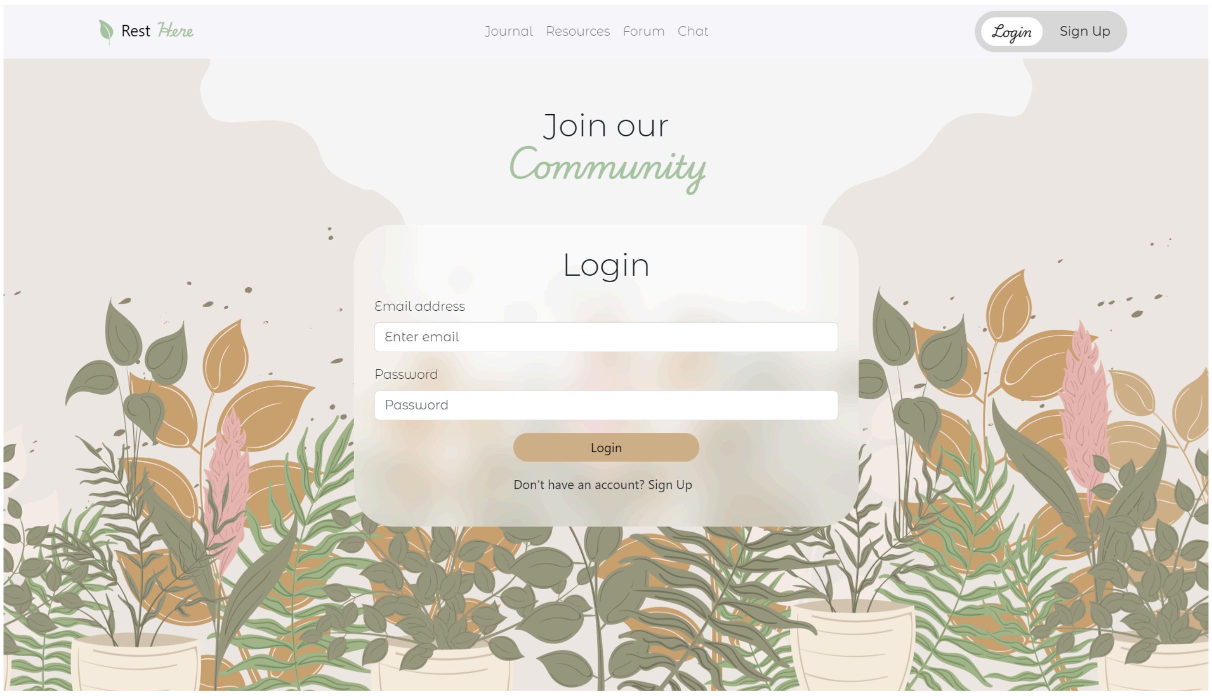
- ***Chat Routes:***
    - get('/:userId/:professionalId') - get a message between a user and a professional
    - get('/professionals/:userId') - fetch all professionals a user is messaging
    - post('/send') - send a message from a sender to a receiver => create a new message with a sender field and a receiver field
    - patch('/edit/:messageId') - allows a user / professional to edit a message's content that they have sent
    - delete('/delete/:messageId') - delete a message by message ID but only if person deleting is the sender

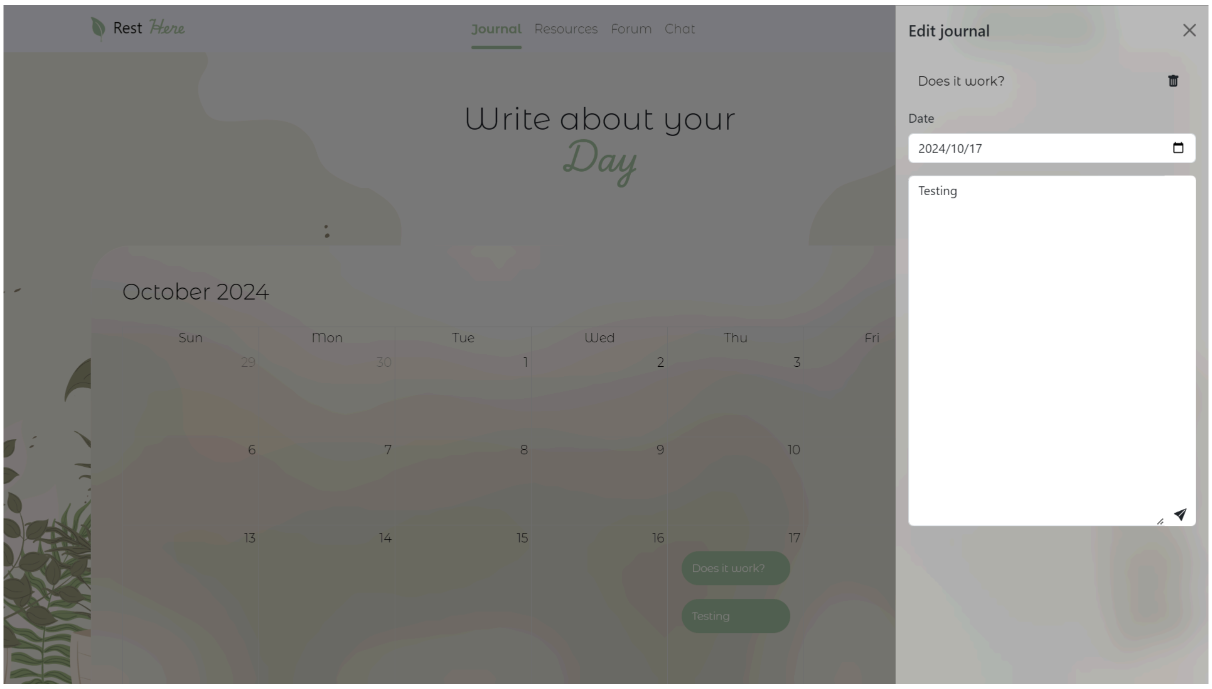
- **UI Design:**





29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26



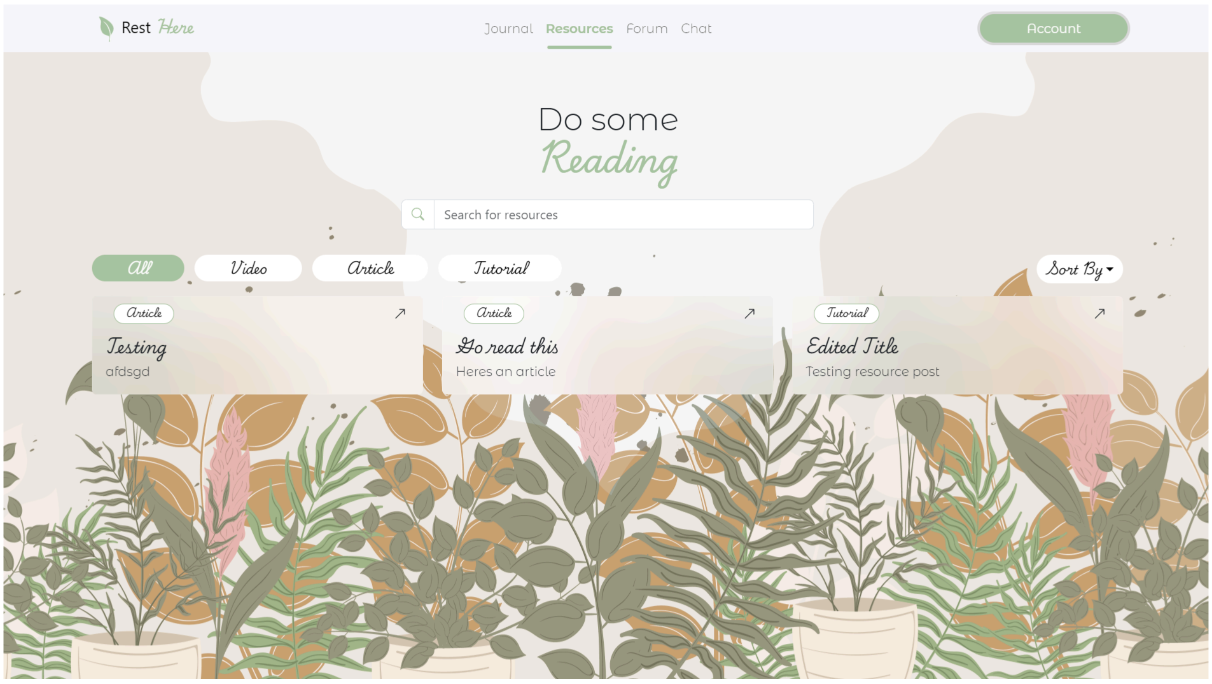


Does it work?

Date

2024/10/17

Testing


Rest Here

Journal Resources Forum Chat

Account

Do some Reading

Search for resources

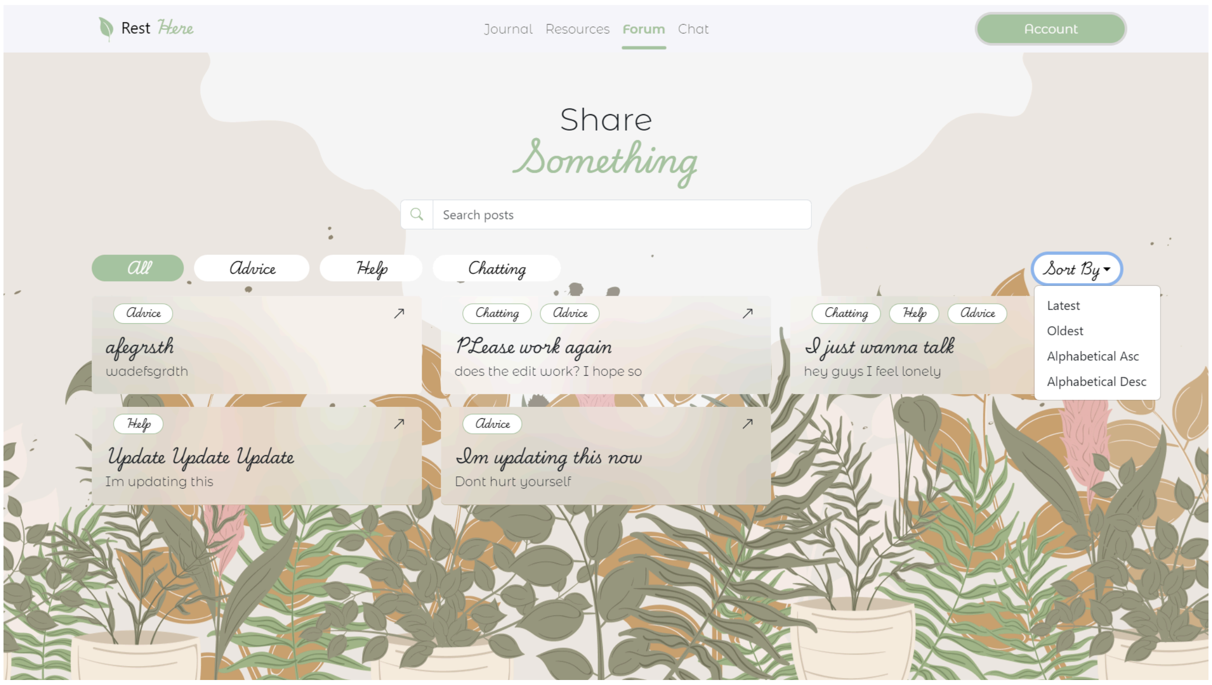
All Video Article Tutorial

Sort By

Article
Testing
afdsgd

Article
Go read this
Heres an article

Tutorial
Edited Title
Testing resource post


Rest Here

Journal Resources Forum Chat

Account

Share Something

Search posts

All Advice Help Chatting

Sort By
Latest
Oldest
Alphabetical Asc
Alphabetical Desc

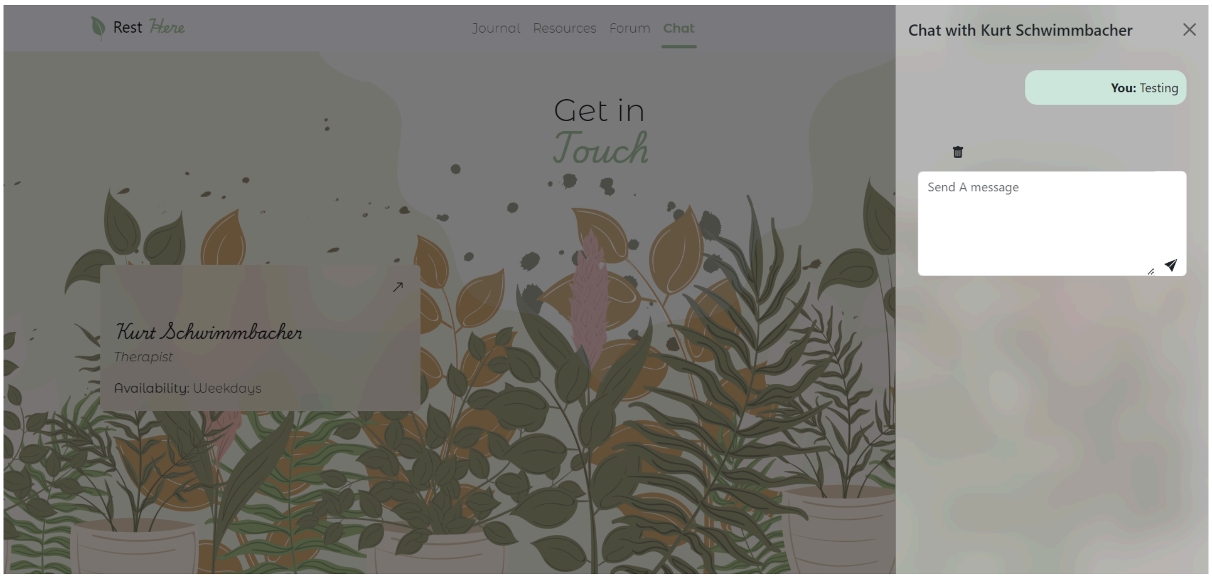
Advice
afegrsth
wadefsgrdth

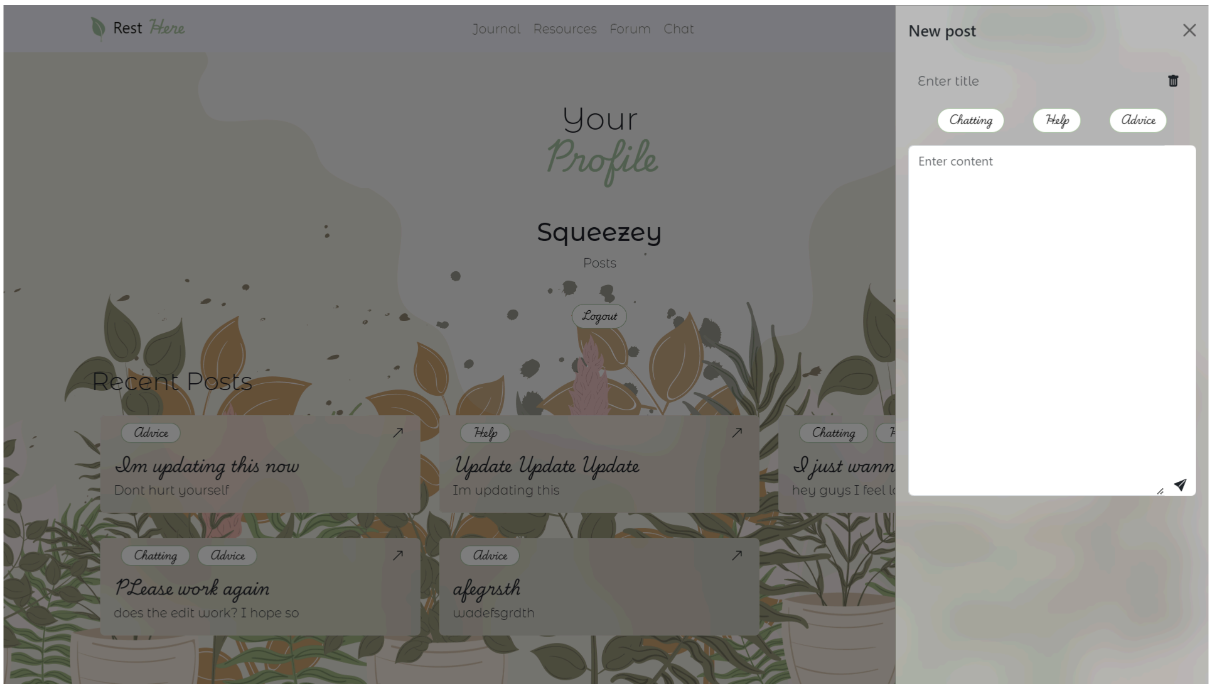
Chatting Advice
PLease work again
does the edit work? I hope so

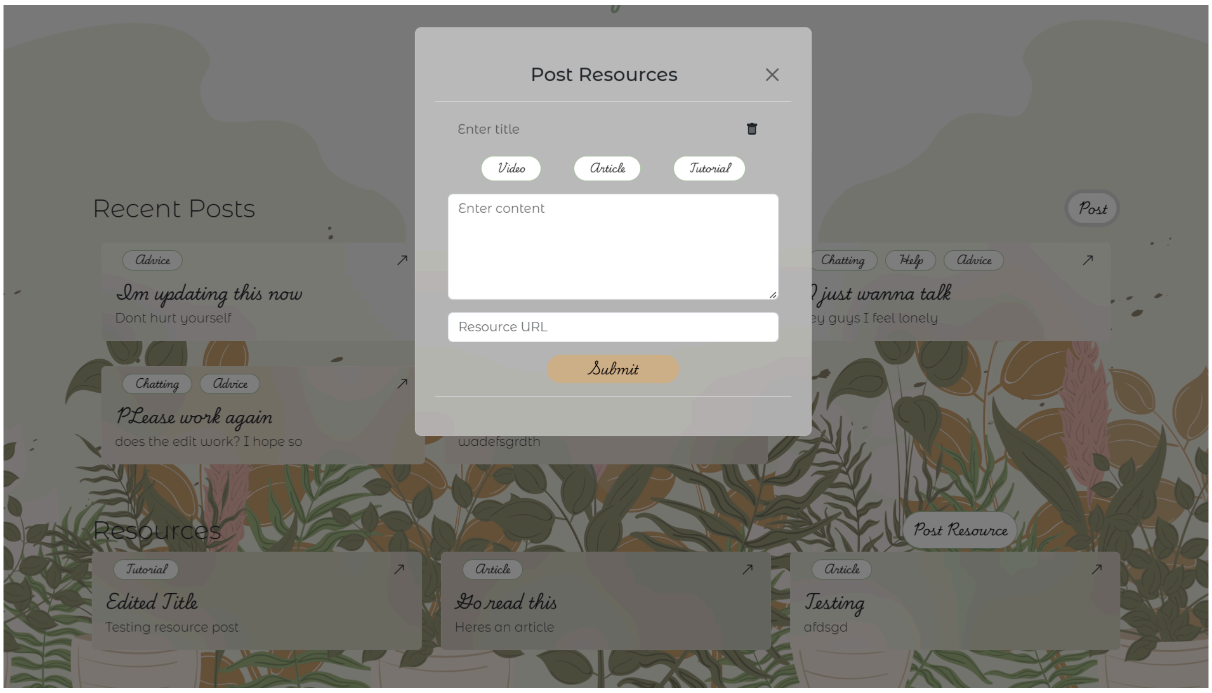
Chatting Help Advice
I just wanna talk
hey guys I feel lonely

Help
Update Update Update
Im updating this

Advice
Im updating this now
Dont hurt yourself


Rest Here
Journal
Resources
Forum
Chat
Get in Touch
Kurt Schwimmbacher
Therapist
Availability: Weekdays
Chat with Kurt Schwimmbacher
You: Testing
Send A message


Rest Here
Journal
Resources
Forum
Chat
Your Profile
Squeezey
Posts
Logout
Recent Posts
Advice
Im updating this now
Dont hurt yourself
Help
Update Update Update
Im updating this
Chatting
I just wann
hey guys I feel l
Chatting
Advice
PLease work again
does the edit work? I hope so
Advice
afegrsth
wadefsgrdth
New post
Enter title
Chatting
Help
Advice
Enter content



- **Instructions for running the project:**

Prerequisites

Ensure you have the following installed on your system:

- Node.js (version 20.18.0 or later)
- MongoDB (either locally or via a cloud provider like MongoDB Atlas)
- Git or Github

1. Clone the Repository

git clone https://github.com/KurtSchwimmbacher/DV200T4_RestHere

2. Set up environment variables

```
MONGODB_URI=your_mongodb_connection_string
PORT=5000
JWT_SECRET=your_jwt_secret_key
```

3. Install Backend Dependencies

```
cd resthere-backed
npm install (bcrypt, cors, dotenv, express,mongoose, validator)
```

4. Install frontend dependencies

```
cd ../resthere-frontend
npm install( full-calendar, axios, react-bootstrap, react-router, react-redux)
```

5. Start the backend and the frontend separately with

```
npm start
```

Problem Statement

Problem Definition:

The prevalence of mental health issues is rising. However, access to mental health resources is limited due to stigma, cost, and availability. Numerous individuals face difficulty locating a secure, confidential, and supportive environment to articulate their emotions or seek assistance.

Significance of the Problem:

Mental health is a crucial component of holistic well-being. The absence of readily available and cost-effective mental health assistance can result in grave outcomes such as anxiety, depression, and potentially, suicide. Through establishing a digital platform that promotes user engagement in sharing their perspectives, gaining access to resources, and connecting with experts confidentially, this application caters to the demand for accessible mental health aid.

Solution:

The proposed platform will offer a secure environment for users to confidentially record their thoughts, access mental health resources, and engage with professionals. The platform's emphasis on confidentiality and user-friendliness will enable individuals to initiate the initial actions towards overseeing their mental well-being.