

Progress Milestone Check

Functional Requirements Checklist:

- ✓ Sign up functionality
 - ✓ Login functionality
 - ✓ AddJuice functionality – standard and admin user
 - ✓ Juices filtering
 - ✓ Juice Pagination
 - ✓ Display Users
 - ✓ Admin – frontend setup
 - ✓ Standard user frontend setup
 - ✓ Juices Filtering
-
- User – liked juices
 - User – follow friends
 - Animations?
 - Gather personal Info for personalised suggested juices
 - Reviews – flag/unflagged
 - Editing Juices
 - Responsive Mobile
 - Responsive Tablet
 - Progress bar

System Documentation for Rejuicenate

1. Technical Architecture

Overview of the System's Components

Rejuicenate is a full-stack web application built on the MERN (MongoDB, Express, React, Node.js) technology stack. It is designed to provide users with a platform to explore and manage juice recipes, categories, and user profiles.

Technology Stack

- **MongoDB:** NoSQL database for storing user data, juice categories, and recipes.
- **Express:** Web application framework for Node.js, used to handle API requests and responses.
- **React:** Front-end JavaScript library for building user interfaces, allowing for a dynamic and responsive user experience.
- **Node.js:** JavaScript runtime that enables server-side scripting and serves as the backbone for the application.

Component Interaction

1. Client Side (React)

- Users interact with the application via a React front-end.
- The front-end makes HTTP requests to the Express API to fetch or manipulate data.

2. Server Side (Node.js + Express)

- The Node.js server processes incoming requests from the client.
- Express routes define how requests are handled, interfacing with the database through Mongoose.

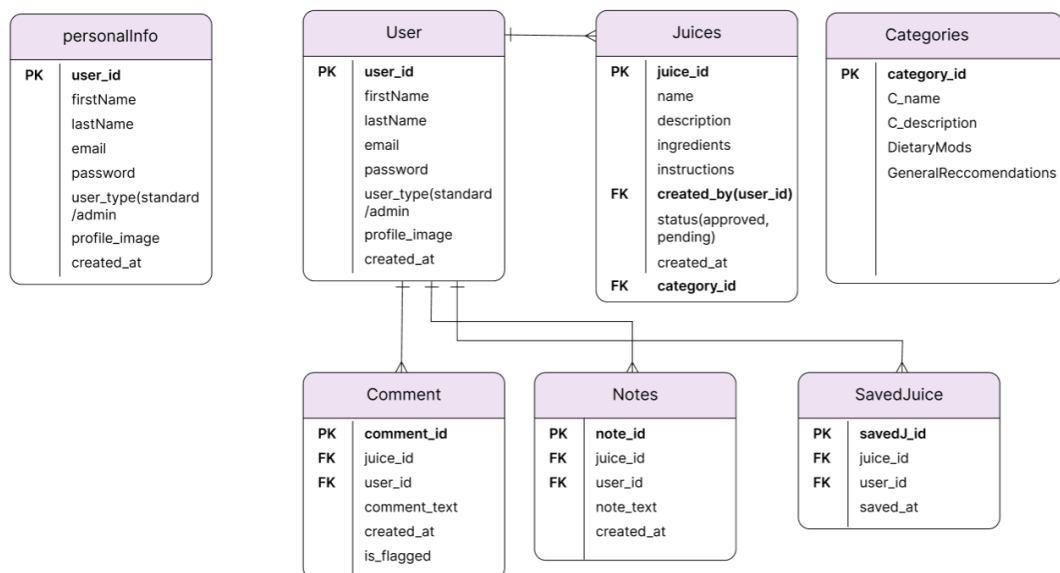
3. Database (MongoDB)

- MongoDB stores structured data in collections (e.g., users, juices, categories).
- Mongoose is used for schema definition and interaction with MongoDB.

2. Database Schema

ER Diagrams and Descriptions

Entity-Relationship Diagram (ERD)



Key Tables and Relationships

• Users

- `id`: ObjectId (Primary Key)

- name: String
- surname: String
- email: String (Unique)
- password: String
- user_type: String (Admin/Standard)
- profile_image: String
- added_at: Date
- **Juices**
 - id: ObjectId (Primary Key)
 - name: String
 - description: String
 - image: String
 - category_id: ObjectId (Foreign Key to Categories)
- **Categories**
 - id: ObjectId (Primary Key)
 - category_name: String
 - dietary_modifications: String
 - general_recommendations: String

Relationships

- **Users** can create multiple **Juices**.
- **Juices** belong to one **Category**.

3. API Endpoints

Key API Routes

Users

- **GET /api/users:** Retrieve a list of users.
- **POST /api/users:** Create a new user.
- **PUT /api/users/**

: Update user information.

- **DELETE /api/users/**

: Delete a user.

Juices

- **GET /api/juices:** Retrieve a list of juices.
- **POST /api/juices:** Add a new juice.
- **PUT /api/juices/**

: Update juice information.

- **DELETE /api/juices/**

: Delete a juice.

Categories

- **GET /api/categories:** Retrieve a list of categories.
- **POST /api/categories:** Add a new category.
- **PUT /api/categories/**

: Update category information.

- **DELETE /api/categories/**

: Delete a category.

Functionalities

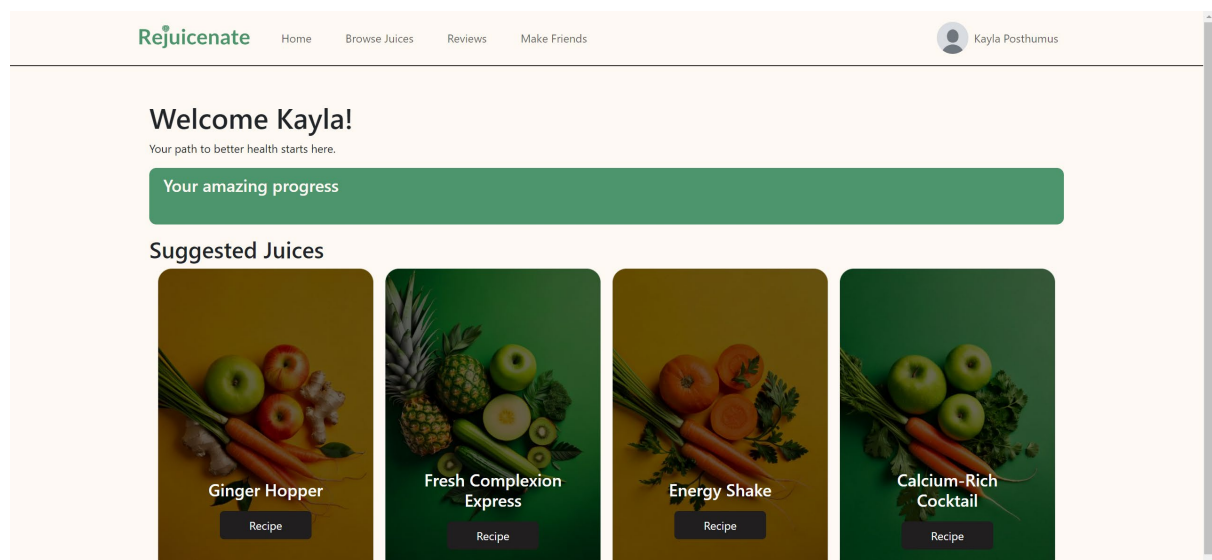
- **Authentication:** Endpoints for user registration and login.
- **CRUD Operations:** Complete Create, Read, Update, Delete functionality for users, juices, and categories.

4. User Interface (UI) Design

Screenshots and Descriptions

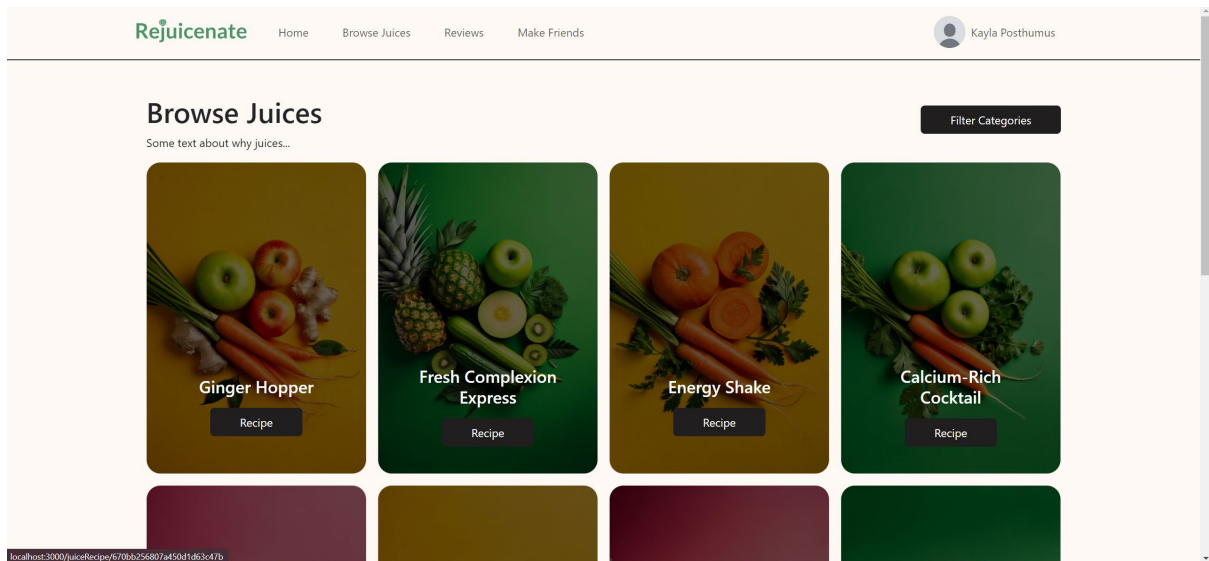
Standard User

Homepage



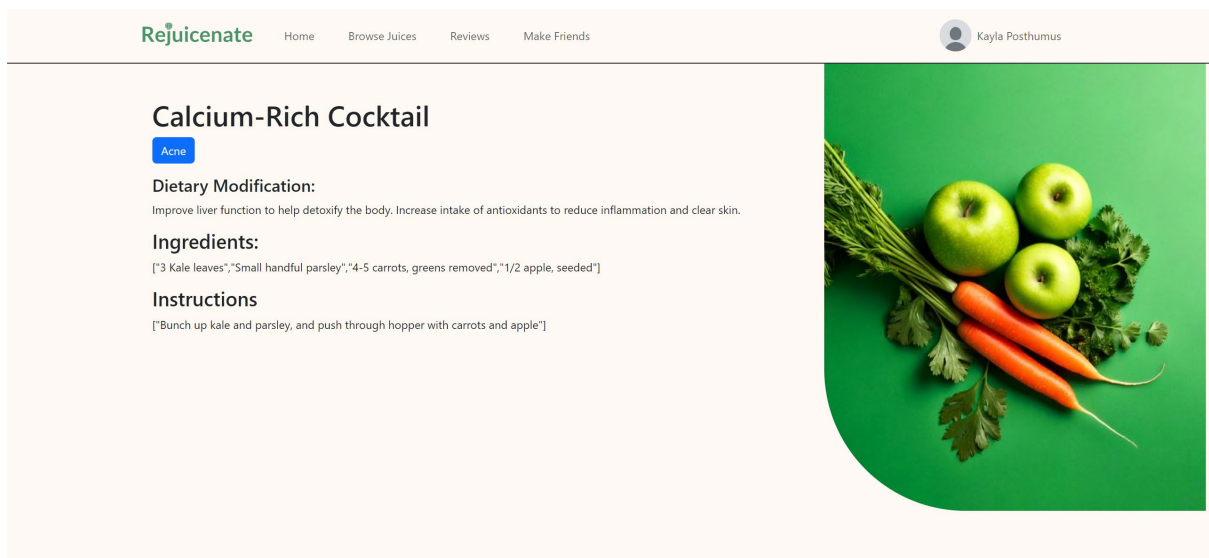
- Offers a friendly greeting to make the user feel welcome
- A progress bar for each day the user visits the website
- Suggested juices based on the user personal info

Juice List Page



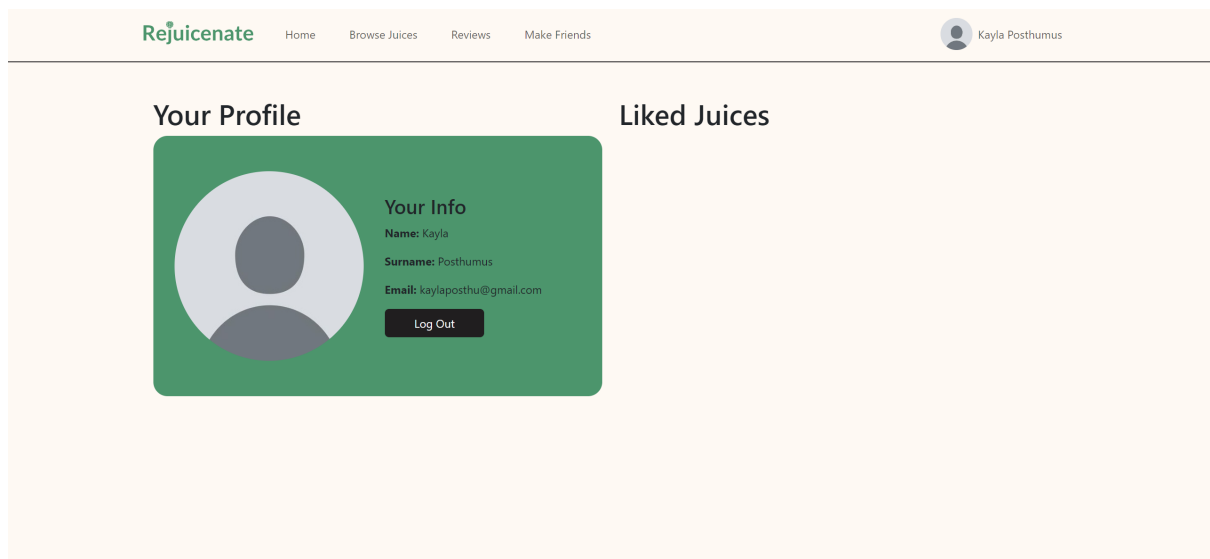
- Displays a list of juices with images and names. When clicked the user is taken to the specific juice page
- Filtering different categories

Specific Juice Page



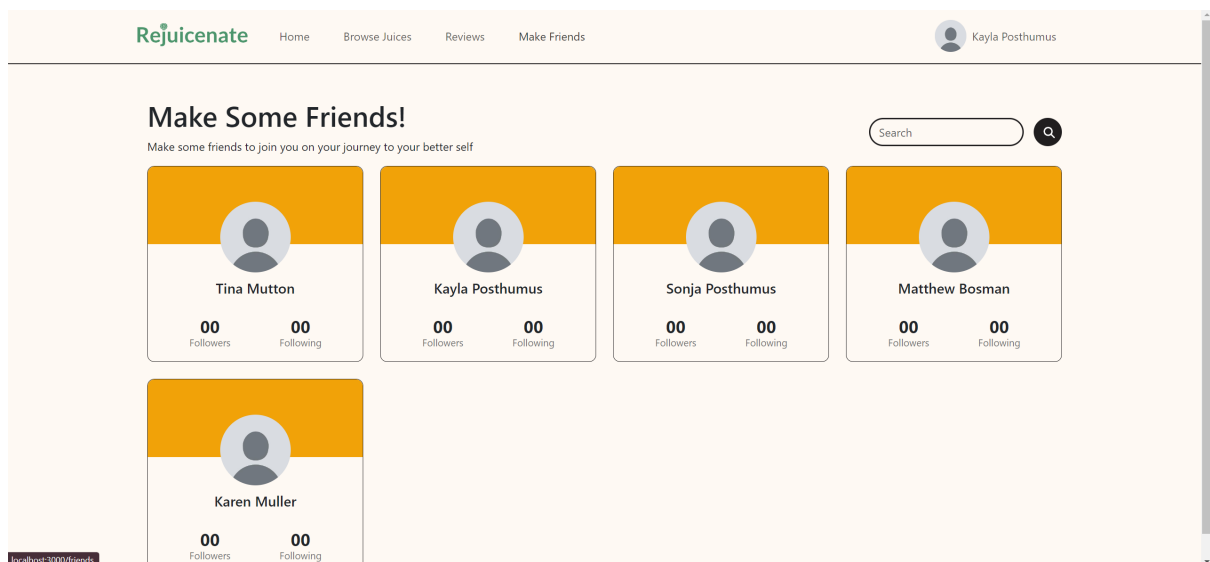
- The user is given the ingredients and instructions as well as background information about the categor.
- The user should also be able to like the juice

Profile Page



- The profile page is where the user information and liked juices is shown.
- The user can choose to add a profile image

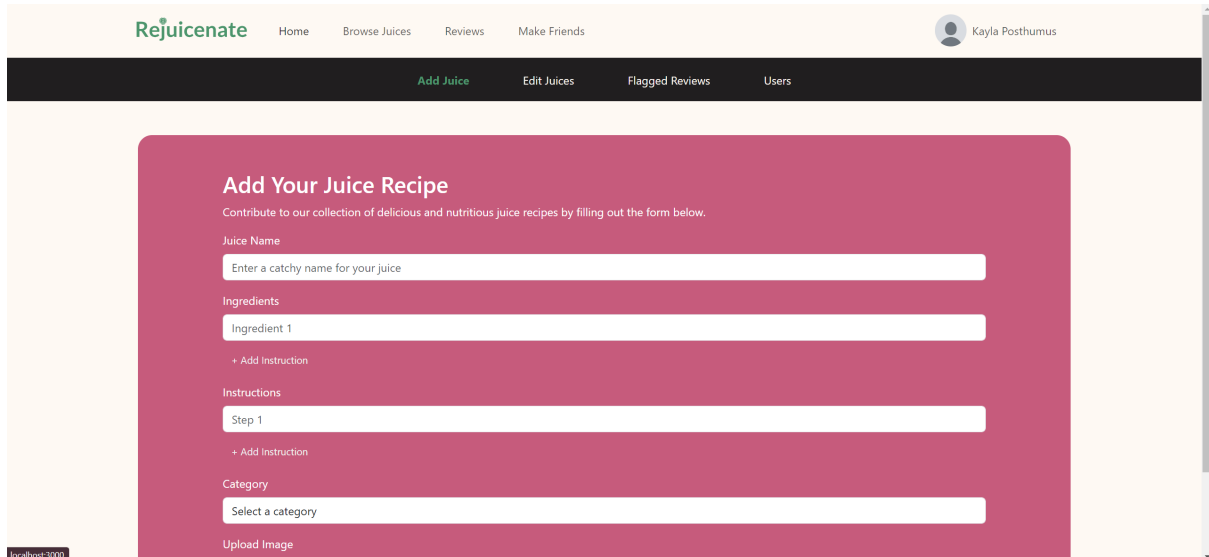
Make Friends Page



- The user is able to “follow” or make friends with other users

Admin User

Add Juice Page



The screenshot shows the 'Add Your Juice Recipe' form in the Rejuicenate application. The form is set against a pink background and includes the following fields and options:

- Juice Name:** A text input field with the placeholder 'Enter a catchy name for your juice'.
- Ingredients:** A text input field with the placeholder 'Ingredient 1' and a '+ Add Instruction' link below it.
- Instructions:** A text input field with the placeholder 'Step 1' and a '+ Add Instruction' link below it.
- Category:** A dropdown menu with the placeholder 'Select a category'.
- Upload Image:** A label for an image upload field.

The top navigation bar includes links for Home, Browse Juices, Reviews, Make Friends, Add Juice, Edit Juices, Flagged Reviews, and Users. The user 'Kayla Posthumus' is logged in.

- Admins are able to add Juices to the database

Responsiveness

None yet

5. Instructions for Running the Application

Development Environment Setup

- 1. Prerequisites:** Ensure you have the following installed on your machine:
 - **Node.js:** Download and install from nodejs.org.
 - **MongoDB:** Set up a local instance or use a cloud provider like [MongoDB Atlas](https://www.mongodb.com/atlas).
 - **Git:** For version control. Download from git-scm.com.
- 2. Clone the Repository:** Open your terminal and run the following command to clone your repository (replace the URL with your repo link):

```
bash
```

Copy code

```
git clone https://github.com/yourusername/rejuicenate.git
```

- 3. Navigate to the Project Directory:**

```
cd rejuicenate
```

- 4. Install Backend Dependencies:** If your backend is in a separate folder (e.g., backend), navigate to that folder:

cd backend

Then install the dependencies:

npm install

- 5. Set Up Environment Variables:** Create a .env file in your backend directory and configure the necessary environment variables, such as:

makefile

Copy code

PORT=5001

MONGODB_URI=your_mongodb_connection_string

JWT_SECRET=your_jwt_secret

- 6. Start the Backend Server:**

npm start

- 7. Open a New Terminal for the Frontend:** If your frontend is in a separate folder (e.g., client), open another terminal and navigate to that folder:

cd frontend

- 8. Install Frontend Dependencies:**

npm install

- 9. Start the Frontend Server:**

npm start

- 10. Access the Application:** Open your web browser and go to <http://localhost:3000> to view the application.

Deployment Instructions

- 1. Prepare for Deployment:**

- **Ensure your application is production-ready** (e.g., test all features, optimize assets).
- **Build the frontend for production:**

cd frontend

npm run build

- 2. Choose a Hosting Provider:** Select a hosting provider for your application. Common options include:

- **Heroku:** For hosting Node.js applications.

- **Vercel or Netlify:** For deploying React applications.
- **AWS, DigitalOcean, or Google Cloud:** For full-stack applications.

3. Deploy the Backend: If using Heroku:

- **Log in to Heroku:**

bash

Copy code

heroku login

- **Create a new Heroku app:**

bash

Copy code

heroku create your-app-name

- **Set environment variables:**

bash

Copy code

heroku config:set MONGODB_URI=your_mongodb_connection_string

heroku config:set JWT_SECRET=your_jwt_secret

- **Push your code to Heroku:**

bash

Copy code

git push heroku main

4. Deploy the Frontend: If using Vercel:

- **Install the Vercel CLI:**

bash

Copy code

npm install -g vercel

- **Log in to Vercel:**

bash

Copy code

vercel login

- **Deploy your frontend:**

bash

Copy code

vercel

5. **Connect Frontend and Backend:** Ensure that your frontend is correctly configured to make requests to your backend URL in production. Update any API endpoints as necessary.
6. **Monitor Your Application:** Once deployed, monitor the performance and functionality of your application through your hosting provider's dashboard.

Problem Statement:

ReJuicenate is a web application designed to tackle the growing health crisis related to poor nutrition, which affects millions of individuals and contributes to chronic conditions such as asthma and heart disease. In our fast-paced society, many struggle to maintain a balanced diet, leading to significant quality-of-life issues and increased healthcare costs. By providing personalized juice fasting plans, nutritional guidance, and progress tracking, ReJuicenate empowers users to reclaim their health. This application not only helps individuals make informed dietary choices but also fosters a healthier community overall, reducing the burden of chronic health issues.