SIT725 – Applied Software Engineering

Task 12.1P

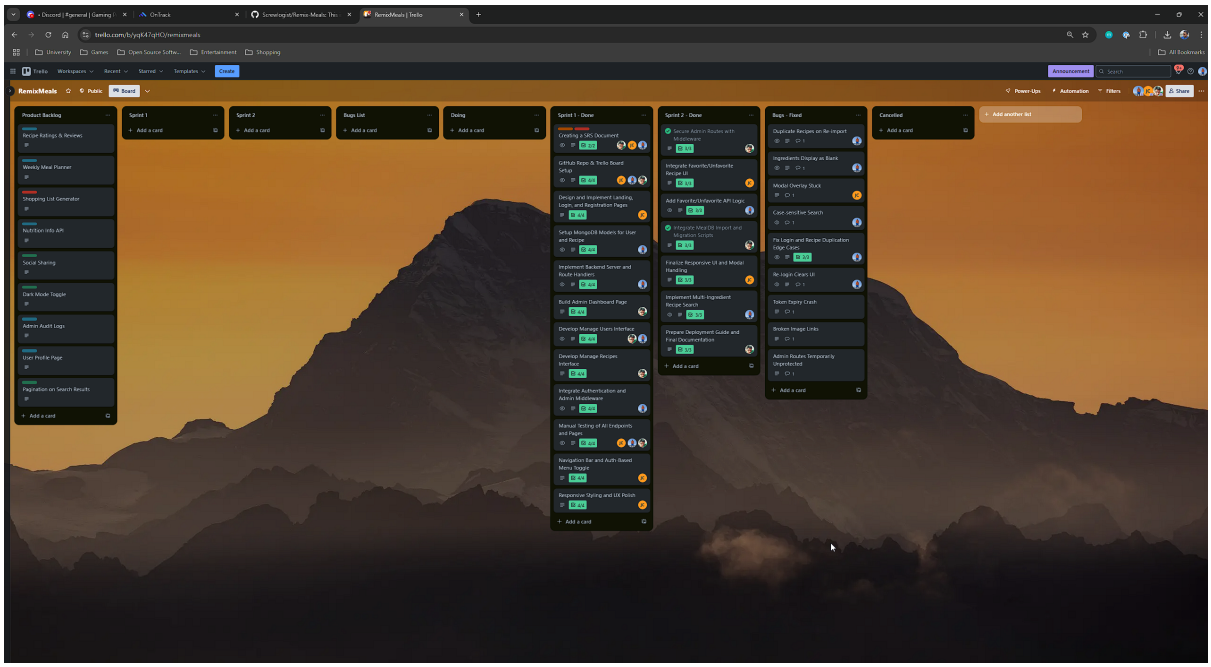
Section 1: SRS

Our SRS remains unchanged since 9.2P.

Section 2: Sprint Backlog Review

Planned Sprint 2 Tasks	Completed	Cancelled
Secure Admin Routes with Middleware	✅	
Integrate Favorite/Unfavorite Recipe UI	✅	
Add Favorite/Unfavorite API Logic	✅	
Integrate MealDB Import and Migration Scripts	✅	
Finalize Responsive UI and Modal Handling	✅	
Implement Multi-Ingredient Recipe Search	✅	
Navigation Bar and Auth-Based Menu Toggle	✅	
Responsive Styling and UX Polish	✅	
Prepare Deployment Guide and Final Documentation	✅	

Screenshot of the Trello board:



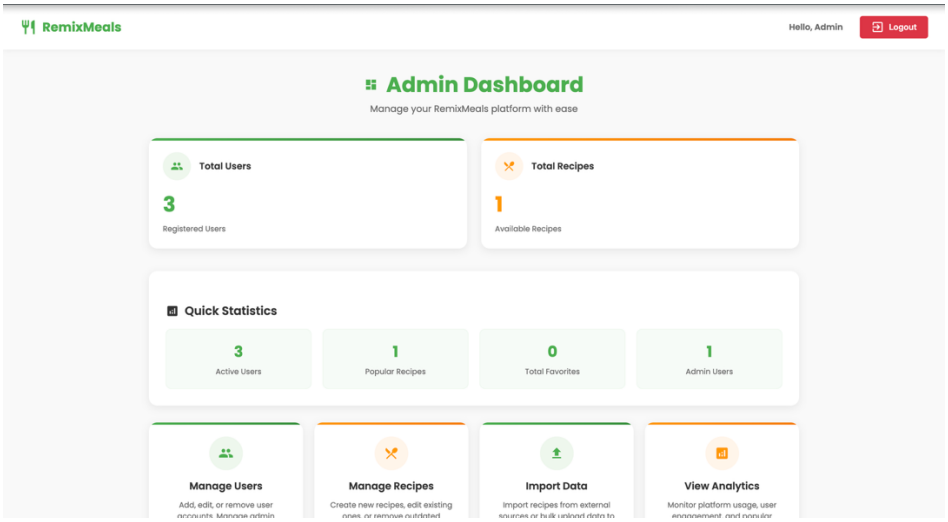
Trello board link:
https://trello.com/b/yqK47qHO/remixmeals

Section 3 Application in action:

User Story 1: Admin User Logs in and Sees Dashboard

User Story: *As an admin, I want to access a dashboard that gives me an overview of users, recipes, and system stats.*

Screenshot:



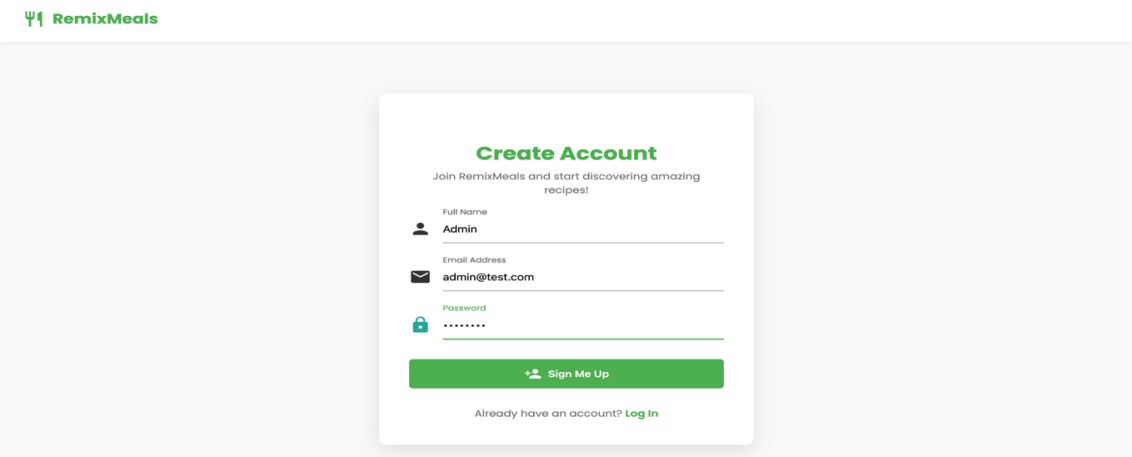
What It Shows:

- Logged-in admin view
- User count, recipe count
- Action cards to manage users, recipes, and import data

User Story 2: A New Admin Registers

User Story: *As a new user, I want to sign up with my email and password to use the platform.*

Screenshot:



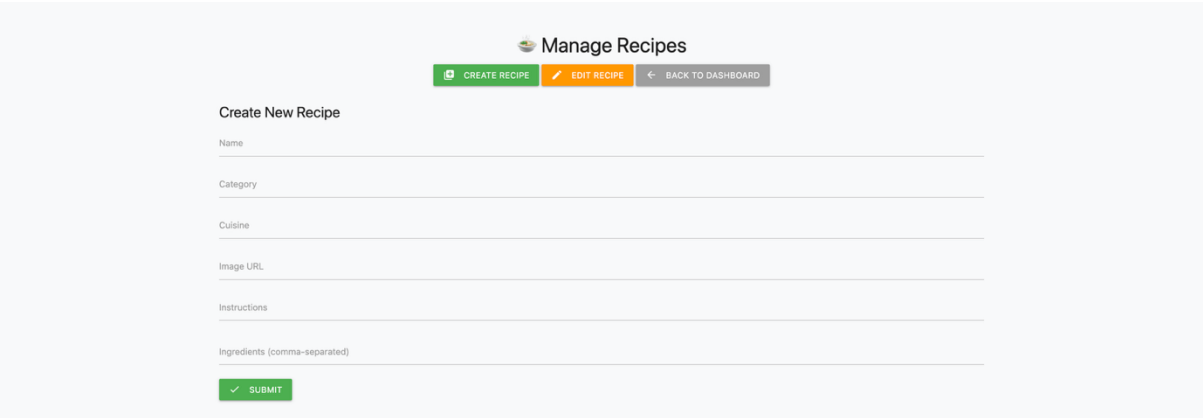
What It Shows:

- Admin registering via register.html
- Fields for full name, email, and password
- User becomes admin if they're first registered

User Story 3: Admin Creates a New Recipe

User Story: *As an admin, I want to create and manage recipes with ingredients, images, and instructions.*

Screenshot:

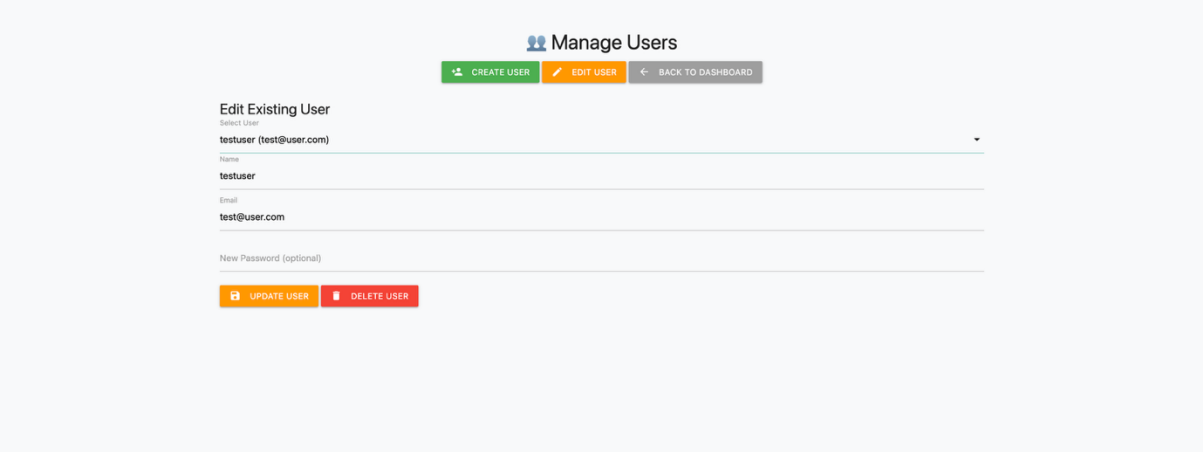


What It Shows:

- Recipe creation form
- Fields for name, category, cuisine, image, ingredients, and instructions

User Story 4: Admin Edits a User Account

User Story: *As an admin, I want to update user details and manage user accounts.*
Screenshot:



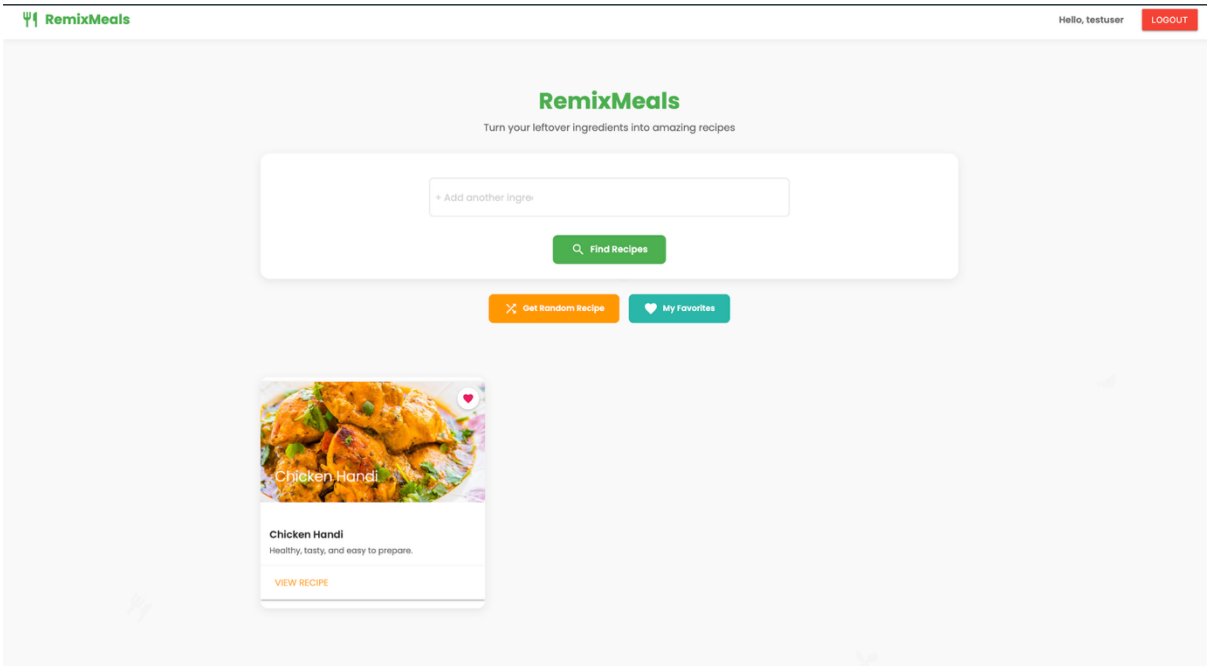
What It Shows:

- Select and edit existing users
- Update name/email/password or delete account

User Story 5: Users Add Recipes to Favorites

User Story: *As a user, I want to Favorite recipes and revisit them easily.*

Screenshot:



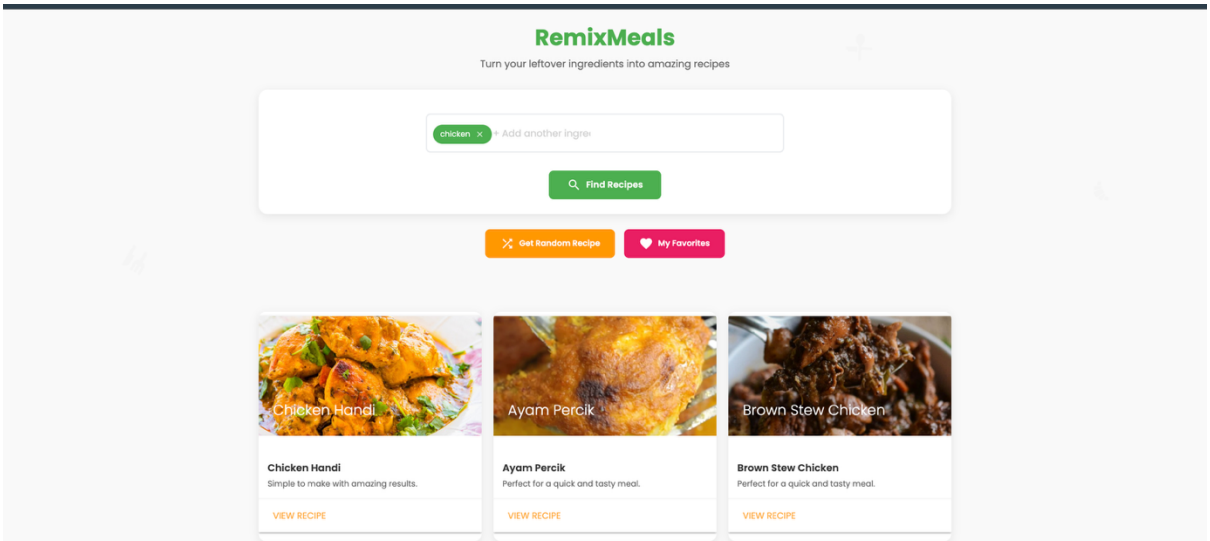
What It Shows:

- “Add to Favorites” button on recipe modal
- Heart icon toggle
- “My Favorites” button for viewing saved recipes

User Story 6: Users Search Recipes by Ingredients

User Story: *As a user, I want to input multiple ingredients and find matching recipes.*

Screenshot:

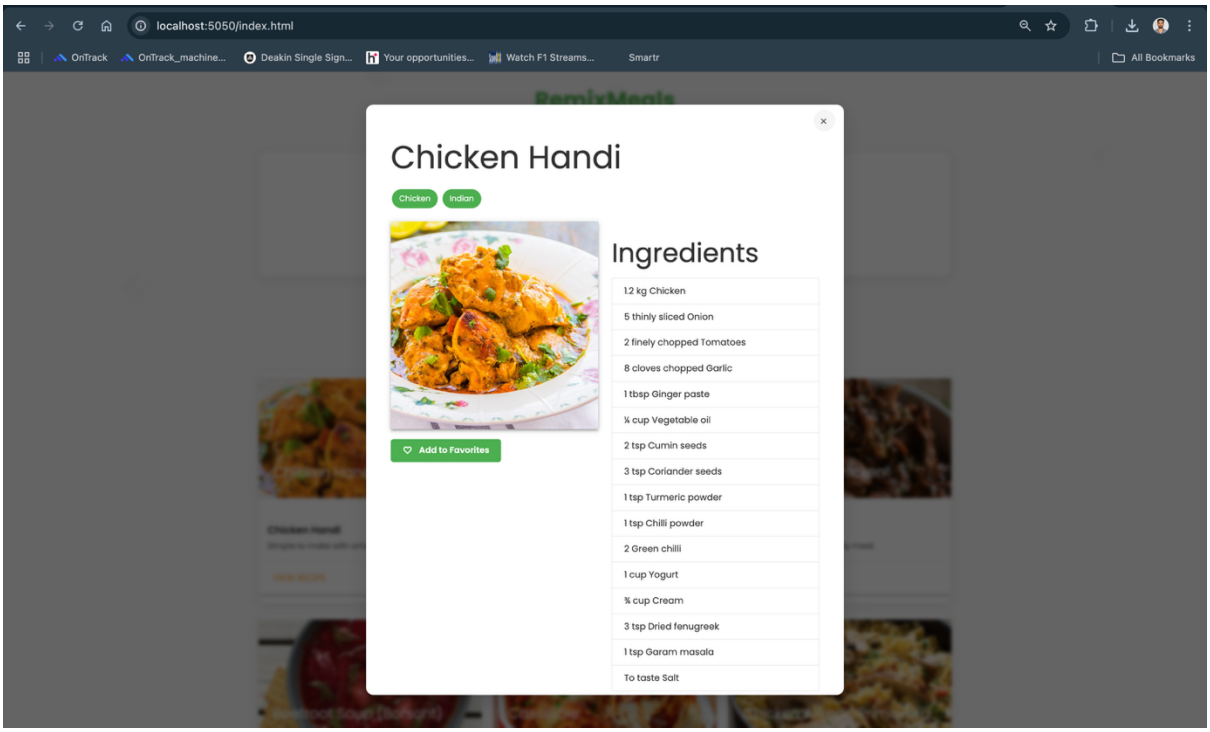


What It Shows:

- Chips-based multi-ingredient input
- Search results based on input
- “Get Random Recipe” and “My Favorites” also visible

User Story 7: Recipe Modal Displays Details

User Story: *As a user, I want to view full recipe details before deciding to cook or Favorite.***Screenshot**:



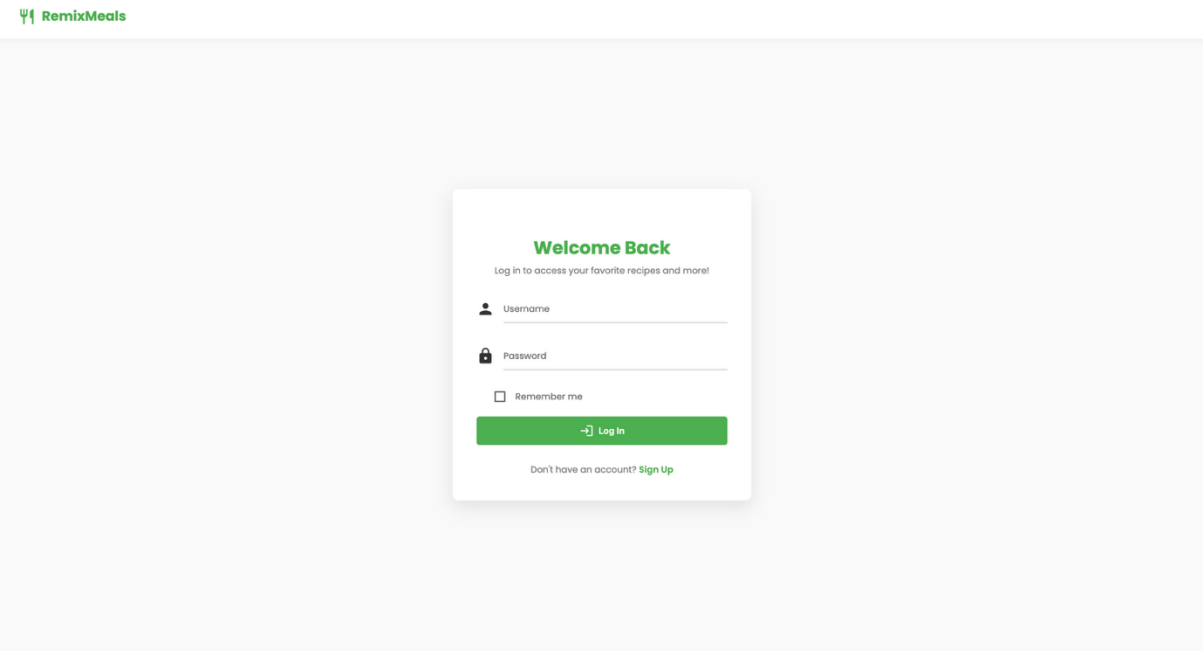
What It Shows:

- Recipe name, image, ingredient list
- “Add to Favorites” button
- Clean UI layout with proper formatting

User Story 8: Login to Access the Platform

User Story: *As a returning user, I want to log in securely to access recipes and features.*

Screenshot:



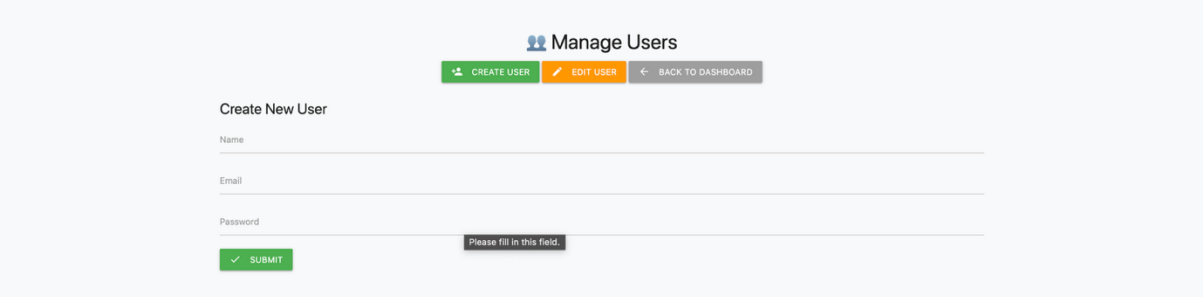
What It Shows:

- Login screen with username and password fields
- Styled form using Materialize CSS

User Story 9: Admin Creates a New User

User Story: *As an admin, I want to create users from the admin panel.*

Screenshot:



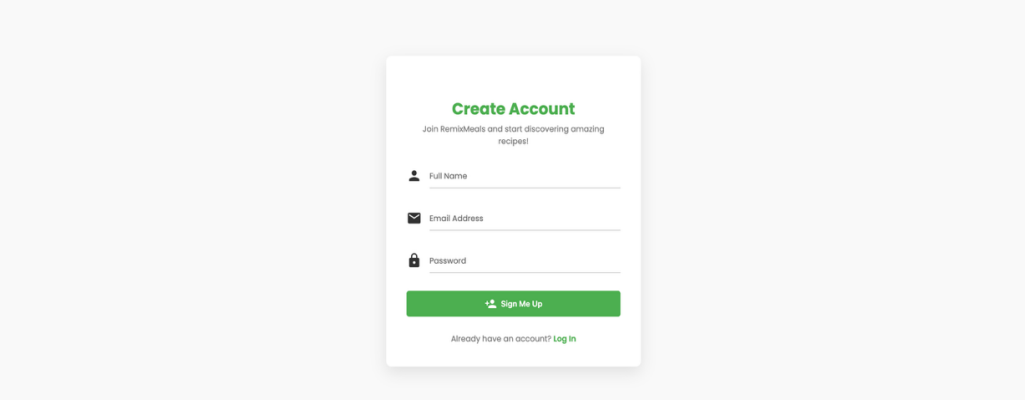
What It Shows:

- Form to input name, email, and password
- Accessible from Manage Users section

User Story 10: General User Registration

User Story: *As a new user, I want to register with RemixMeals so I can access recipe recommendations.*

Screenshot:

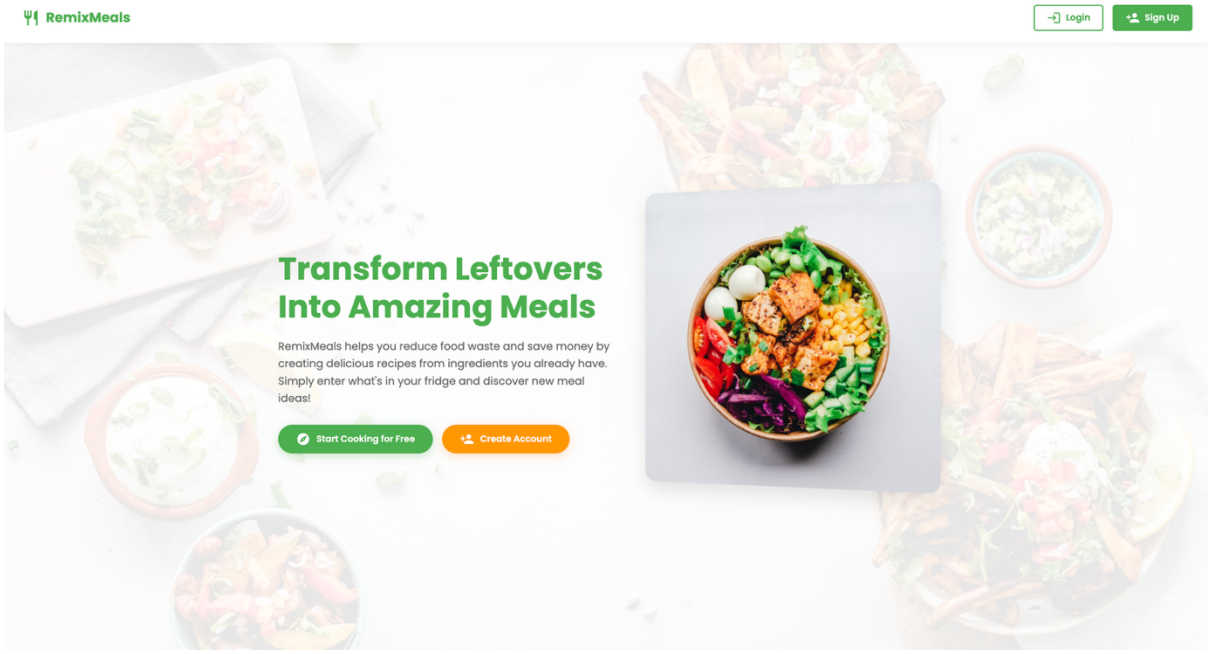


What It Shows:

- Registration form for name, email, and password
- User-friendly interface for first-time users
- Connected to backend /register API

User Story 11: Welcome Page for New or Logged-out Use
User Story: *As a user, I want a welcoming homepage that introduces the app and directs me to login or signup.*

Screenshot:



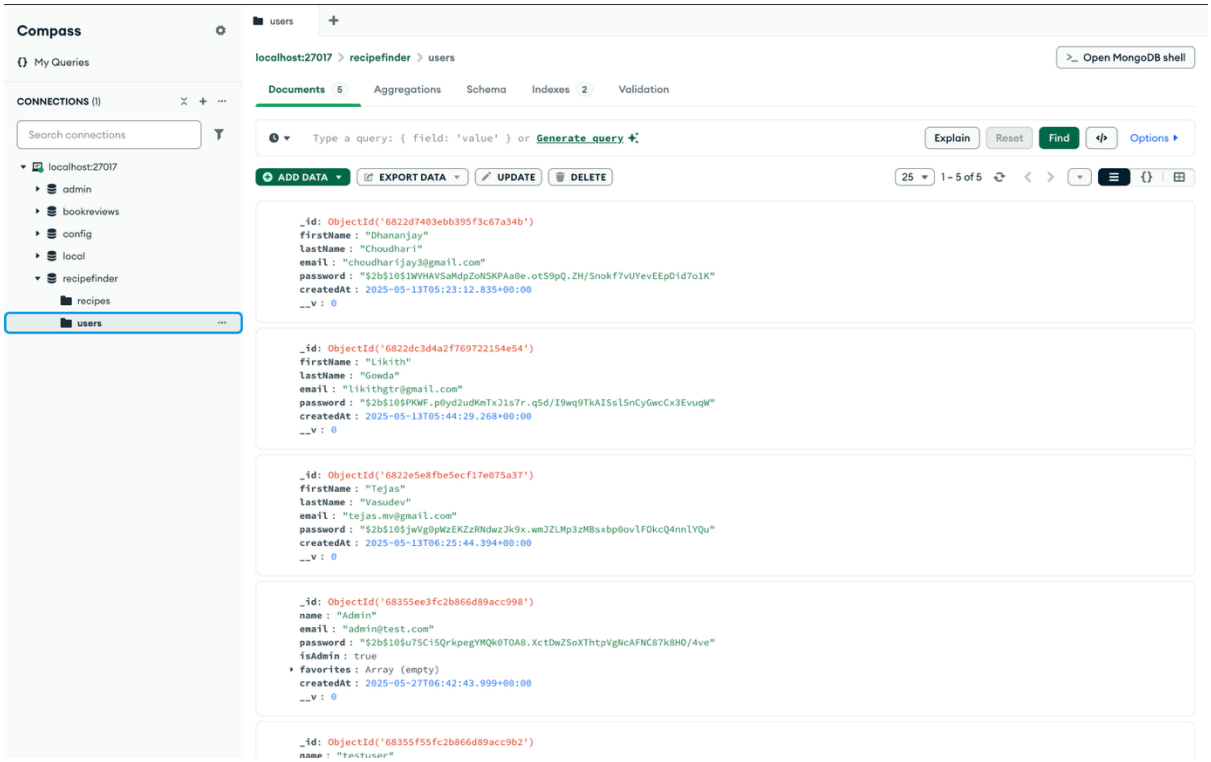
What It Shows:

- Hero section with app message: “Transform Leftovers into Amazing Meals”
- Call-to-action buttons for starting or creating an account
- Responsive, visually engaging first interaction

Backend Confirmation: User Collection in MongoDB

User Story: *As a developer/admin, I want to confirm registered users are stored correctly.*

Screenshot:

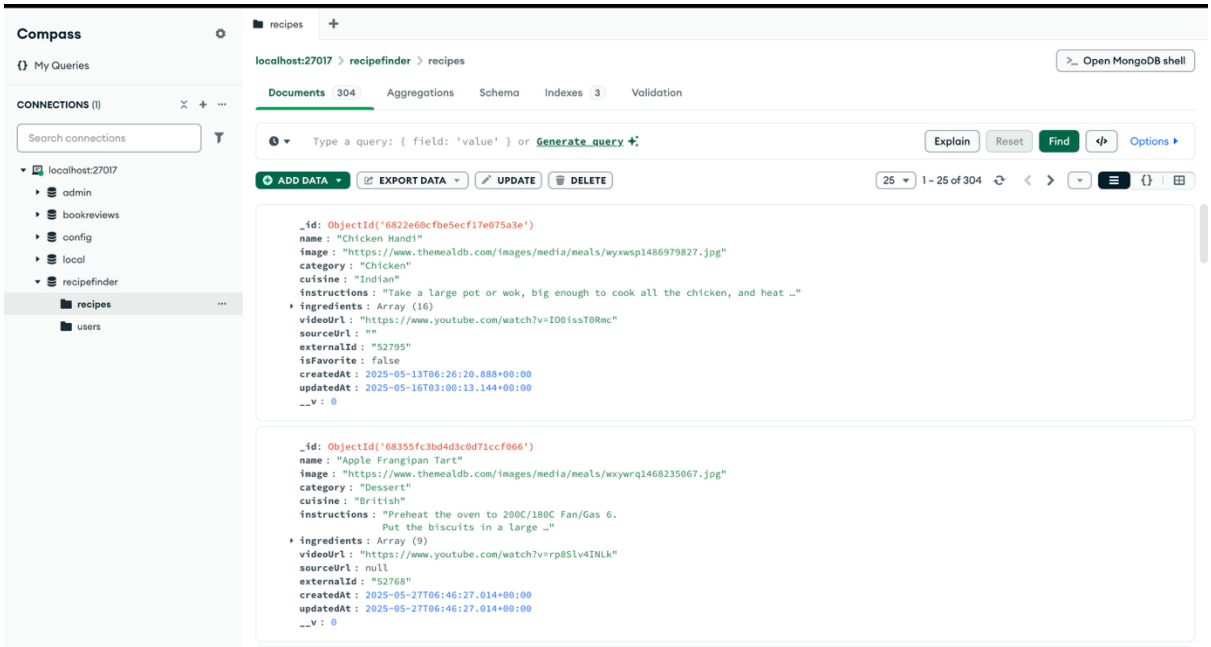


What It Shows:

- MongoDB Compass showing multiple user entries
- Fields: firstName, lastName, email, password, isAdmin, favorites[]
- Confirms backend and DB integration with registration

Backend: Recipe Database in MongoDB

User Story: *As a developer, I want to verify that recipes are stored and retrievable from the database* **Screenshot**:



What It Shows:

- MongoDB Compass view of stored recipes
- Shows _id, name, ingredients[], externalId, timestamps

Self-assessed rubric with evidence for Task 12.1P

Overview of contributions & target grade

Name	Main Role/s	Target Grade
Dhananjay	Frontend Developer, UI/UX Designer	C
Likith	Backend Developer, Database Architect	C
Utkarsh	Full Stack Developer, DevOps Lead	C

Dhananjay

Criteria	Pass -> *"can do allocated work with guidance "*	Credit -> *"performs their role in the Project"*	Distinction -> *"can see themselves in the team and assist others"*	High Distinction -> *"can demonstrate excellence"*
Core Development	]	☑ I built the **main static pages** of our app including landing.html, login.html, and register.html using HTML, CSS, and JavaScript. Initially, these were structured using the default components of Materialize CSS. I later added **session-based navigation logic** that **dynamically** updates menu options based on whether a user is logged in or not. I also ensured the interface was responsive and included smooth animations to enhance the overall experience. **Links:** https://github.com/Screwlogist/Remix-Meals/commit/9786939b2db9c5230ae160b1ba3ae3d89a0381bf		

		https://github.com/Screwlogist/Remix-Meals/commit/d0435d2fab8465c8049530040ce6e29542177806 https://github.com/Screwlogist/Remix-Meals/commit/7ad50f2a3f62dfa3b66e2d2027a08f97af53d066 https://github.com/Screwlogist/Remix-Meals/commit/11a35e720b6e8d6a8184772205a551b19275b312		
CRUD		☑ I implemented the **frontend** login and registration forms using standard HTML inputs. These forms captured user information and passed it to the backend. I added client-side validation to check for required fields and incorrect formats (e.g., invalid emails). I also built the UI for toggling favorite/unfavorite recipes using heart icons, which updated instantly on user interaction. **Links:** https://github.com/Screwlogist/Remix-Meals/commit/d0435d2fab8465c8049530040ce6e29542177806 https://github.com/Screwlogist/Remix-Meals/commit/9f3ea1889f5fbd040d89c1dd64df5262d69be815		
Design		☑ The **flow diagram** illustrates the structure of my frontend contribution. It begins with the user accessing landing.html, styled using the Materialize CSS framework for responsive design and consistency. From there, users navigate to login.html and register.html, both of which include client-side form validation, toast messages, and animations. I also implemented the Favorite/Unfavorite Recipe UI, which integrates JavaScript logic to toggle icons and provide real-time feedback. The design ensures all components are visually cohesive, responsive, and user-friendly across devices. **Link:** commit:		

		https://github.com/Screwlogist/Remix-Meals/commit/cc1b245d0a6548c509ba5cb87a7dc17edfa6e94b		
SRS		I contributed to **writing the frontend section** of our Software Requirements Specification by outlining how login and register pages would function. I made sure the implemented UI (e.g., toggle behaviour, transitions) matched what was described in the document, and kept the interface aligned with the system's requirements. **Link:** https://github.com/Screwlogist/Remix-Meals/commit/2eb708c2f60fd4eb115a9bc49497a2dce10714d2		
Testing		☑ I tested my own components during development — for example, making sure forms submitted and links worked. I **manually tested various frontend features** such as input validation, navigation behaviour, responsive layout issues, and the favourite toggle. I also checked how they behaved on different screen sizes and browsers. **Link:** https://github.com/Screwlogist/Remix-Meals/commit/dd55c46c9018cf6490dbcf67873467f1d27869da		
Leadership		☑ I was responsible for completing the UI pages I was assigned and made sure they were functional. I took initiative to ensure styling was consistent across all pages. I also supported my teammates by sharing reusable CSS components and helping resolve layout bugs. I regularly updated my other teammates with my working, so we all aligned with the project expectations.		
Tools	**Pass**	**Credit**	**Distinction**	**High Distinction**
GIT		☑		

		I regularly pushed code to GitHub while working on different UI components. I made descriptive commit messages and avoided pushing directly to main. Instead, I used feature-based commits like add-login-form-validation to keep changes organised. **Link:** https://github.com/Screwlogist/Remix-Meals/commits/main		
Trello		☑ I tracked my work using Trello by moving cards through stages like "To Do" to "Done". I created clear card titles like "Build Register Page" and updated their status. I also added checklists and wrote comments if I was blocked or had completed parts of the task. **Link:** https://trello.com/b/yqK47qHO/remixmeals		
README file		☑ I contributed **basic frontend setup instructions** to the README, like where to find the HTML pages and how to run them. I improved the README by adding clearer file structure descriptions and specifying the dependencies (like Materialize CSS). This helped ensure other team members and the tutor could easily understand the frontend setup. link : https://github.com/Screwlogist/Remix-Meals/commit/67e62302dc213f45b5382ee00a1f8513dd8f59fe		

Likith

Criteria	Pass -> *"can do allocated work with guidance"*	Credit -> *"performs their role in the project"*	Distinction -> *"can see themselves in the team and assist others"*	High Distinction -> *"can demonstrate excellence"*
Core Development		☑ During Sprint 2, I		

		successfully developed core backend functionality for the RemixMeals application. The application uses MongoDB to store user data and recipes, allowing users to search for recipes based on ingredients they have available. I implemented a complete backend system using Node.js, Express.js, and MongoDB. The backend follows RESTful conventions with clear separation of concerns through routes, models, and middleware. Key implementations include: **User Authentication System:** JWT-based authentication with secure password hashing using bcryptjs. **Recipe Management:** Full CRUD operations for recipes with ingredient-based search functionality. API Endpoints: Well-structured routes for users		

		(/api/users), recipes (/api/recipes), and admin functions (/api/admin). The backend architecture ensures scalability and maintainability through modular design patterns. https://github.com/Screwlogist/Remix-Meals/commit/d2b8c6230669a0651cd4fd377d45c30ead6a7e15		
CRUD		☑ I implemented comprehensive CRUD functionality across multiple entities: **User Management:** **Create:** User registration **Read:** User profile retrieval and authentication status **Recipe Operations:** **Read:** Multiple endpoints for recipe retrieval (all, by ID, search, random) **Update:** Recipe modification with ingredient management **Favorites System:**		

		Add/Remove favorites with user-specific tracking, Retrieve user's favorite recipes with proper authentication. All CRUD operations include proper validation, error handling, and status codes. CRUD: https://github.com/Screwlogist/Remix-Meals/commit/0c358d5bb3fec2f7e2146fef043d500b45b5d52c		
Design		☑ My design contributions covered backend architecture: **Backend Architecture:** Clean separation using MVC pattern, Modular route structure (userRoutes.js, recipeRoutes.js, adminRoutes.js). Middleware implementation for authentication and authorization. Database schema design with Mongoose models. **System Architecture:** JWT token-based		

		authentication flow. RESTful API design principles. Secure password hashing and storage. Role-based access control (RBAC) https://github.com/Screwlogist/Remix-Meals/commit/e22577f42eb174aefdd26267dbe4c75e3608261d		
SRS		☑ I contributed to the Software Requirements Specification by documenting backend functionality requirements. My contributions included: Defining functional requirements for user authentication system. Specifying API endpoint requirements and expected behaviors, Documenting non-functional requirements for security and performance, Defining data models and relationships. The SRS was maintained		

		throughout Sprint 2 to reflect new features like the favorites system and multi-ingredient search functionality. https://github.com/Screwlogist/Remix-Meals/commit/9753642139f31b8b4992b4c5b3b7feeb5043e75f		
Testing		☑ I performed comprehensive manual testing of all backend endpoints and pages: **API Testing:** Tested all authentication endpoints (register, login, profile), Verified CRUD operations for users and recipes, Tested multi-ingredient search with various combinations, Validated favorites functionality with authenticated requests **Bug Fixes Implemented:** 1. Fixed duplicate recipe imports. 2. Resolved blank ingredients display issue. 3.		

		Fixed login and recipe duplication edge cases. 4. Resolved UI clearing issue on re-login. All testing was documented to ensure reproducibility and coverage. https://github.com/Screwlogist/Remix-Meals/commit/6fa86a73446301b659336c871202cd5e0afef4d1		
Real Time				
Discuss with tutor during workshops		☑ I took the feedback received during task 7.4P and adjusted the timeframe in the Trello board, took in the advises provided during workshop and performed the same. https://trello.com/b/yqK47qHO/remixmeals		
Leadership		☑ I demonstrated consistent project involvement through: Setting up the initial project structure and database		

		models, Creating comprehensive README documentation, Managing Trello board with detailed task tracking, Implementing core authentication system that other features depend on, Resolving critical bugs that affected application functionality, Providing clear API documentation for frontend integration, My contributions established the foundation for the application's backend functionality. https://trello.com/b/yqK47qHO/remixmeals https://github.com/Screwlogist/Remix-Meals/issues?q=is%3Aissue%20state%3Aclosed https://github.com/Screwlogist/Remix-Meals/commit/9753642139f31b8b4992b4c5b3b7feeb5043e75f		

Tools	Pass	Credit	Distinction	High Distinction
GIT		☑ I maintained professional Git practices throughout Sprint 2: Clear, descriptive commit messages, Regular commits tracking progress, Proper .gitignore configuration, Organized project structure, Documentation updates alongside code changes. https://github.com/Screwlogist/Remix-Meals/commits/main/ https://github.com/Screwlogist/Remix-Meals		
Trello		☑ I actively used Trello for project management: Created detailed task cards for all backend features, Updated task progress regularly, Added descriptions and checklists for complex tasks, Tracked bug fixes and improvements, Maintained sprint backlog organization https://trello.com/b/yqK47qHO/remixmeals		
README file		☑ I created and maintained a comprehensive README that includes: 1. Project overview and features. 2. Detailed installation instructions. 3. Technology stack documentation. 4. API endpoint documentation. 5. Project structure explanation. The README serves as complete documentation for developers and users, making the project accessible and professional. https://github.com/Screwlogist/Remix-Meals/commit/29b8ebafae651a3c45c88e0f4751410295830c37		

Utkarsh Aggarwal

Criteria	Pass -> *"can do allocated work with guidance"*	Credit -> *"performs their role in the project"*	Distinction -> *"can see themselves in the team and assist others"*	High Distinction -> *"can demonstrate excellence"*

Core Development		☑ I completed my assigned work during Sprint 2 by developing and integrating admin functionality into the core backend of our application. I created secure Express.js routes for admin operations which provided user and recipe management capabilities. The routes utilized Mongoose models to interact with MongoDB data while maintaining seamless operations. The middleware system validated JSON Web Tokens (JWT) while checking user admin privileges before allowing access. The structure implemented an MVC (Model-View-Controller) architecture which distributed logic between controllers and routes and services. The modular design structure improved both code readability and scalability. **Evidence:** Added adminRoutes.js to handle admin-specific CRUD operations · Screwlogist/Remix-Meals@e1bc290 Added admin middleware to restrict access to admin-only routes · Screwlogist/Remix-Meals@60dd33b Added checkAdminStatus logic in server.js to assign first user as admin · Screwlogist/Remix-Meals@3e3d293 Added manage-users.html and manage-recipes.html for admin to manage u... · Screwlogist/Remix-Meals@5399de6 Added admin.html with dashboard layout · Screwlogist/Remix-Meals@28f63f1		
CRUD		☑ I contributed to CRUD		

		functionality primarily for the admin module, implementing all four operations securely: • **Create**: I built POST endpoints that allow the admin to create new users and recipes. When an admin fills out the relevant form and submits it, the backend receives the request, validates the input data, and then stores the new entry in the MongoDB collection. For users, this includes hashing passwords before storage. • **Read**: I implemented GET endpoints that fetch all users and recipes from the database. These endpoints return data in a paginated format to ensure efficient performance and are used to populate tables in the admin dashboard. The data is rendered using EJS templates to show user and recipe lists dynamically. • **Update**: PUT endpoints were developed for editing both users and recipes. The admin can update user roles (e.g., promote/demote) and recipe details. I included robust validation to ensure only valid fields were updated and avoided direct exposure of sensitive fields like passwords. • **Delete**: DELETE endpoints were created to allow admins to remove		

		users and recipes. I ensured that the deletion routes were protected by middleware, preventing unauthorized users from accessing them. Additionally, I added safeguards to prevent accidental deletion of the super admin (first user). To ensure the security and correctness of all CRUD operations, I implemented input validation, used Mongoose schema constraints, and tested each route extensively using Postman. These actions helped maintain database integrity and user trust. **Evidence:** Added adminRoutes.js to handle admin-specific CRUD operations · Screwlogist/Remix-Meals@e1bc290		
Design		☑ My main responsibility was backend development, yet I supported the design and structure of the admin functionality. I made sure that backend routes and logic matched frontend expectations and I helped determine how data should be provided to EJS templates for rendering admin pages. I participated in designing a flowchart which depicted the admin functionality. The flowchart demonstrates how the first registered user receives admin privileges and how JWT-based middleware authenticates all admin requests and how the admin interacts with user and recipe management modules. The visual diagram provided the team with a clear understanding of the		

		complete admin system structure and user interface. I also worked on the design and layout of the admin dashboard, ensuring it followed a clean and intuitive structure. The dashboard includes: • A navigation bar with links to manage users and recipes • Data tables displaying user and recipe information • Action buttons for edit and delete functionality • Stats display showing total number of users and recipes in the system The Materialize CSS framework served as my styling tool to maintain a uniform user interface throughout the application. The application provides real-time feedback through toast notifications which shows success messages and error notifications for admin actions. The design features responsive capabilities which enable admin features to function properly across desktop and tablet screen sizes. The design best practices I followed resulted in an admin interface which delivered both functionality and user-friendliness while achieving project usability goals and maintaining visual consistency with the application. **Evidence:** Added admin.html with dashboard layout · Screwlogist/Remix-Meals@28f63f1 Added manage-users.html and manage-recipes.html for admin to manage u... · Screwlogist/Remix-Meals@5399de6		

		Added admin flow diagram illustrating dashboard logic and access control · Screwlogist/Remix-Meals@3c46cda		
SRS		☑ I maintained the SRS documentation by incorporating the new admin functionalities. I created a new functional requirement section which details the CRUD operations that the admin can perform for user and recipe management. I created a new use case which explains the admin login process and dashboard viewing and user management capabilities. I updated the "User Characteristics" section to show the existence of an admin user, and I modified the architecture section to include role-based access control logic. The updates maintained the documentation consistent with the system implementation and provided clear information for developers, testers and stakeholders. **Evidence:** Updating SRS document to include use cases, admin features · Screwlogist/Remix-Meals@7caabf7		
Testing		☑ In Sprint 2, I completed basic manual testing to ensure the functionality of the features I implemented, particularly within the admin module and user registration process. I tested the backend endpoints by simulating form submissions and observing the results in both the frontend interface and the database. For **user registration**, I verified:		

		• Duplicate email handling: the system correctly prevents users from registering with an already used email address. • Duplicate username check: the application does not allow two users with the same username. • Email format validation: the system only accepts valid email addresses and displays appropriate error messages for incorrect formats. • Password confirmation: mismatched passwords during registration are correctly rejected. For the **admin features**, I manually tested: • Creating a new user with valid input data • Deleting users and recipes from the admin dashboard These tests were performed using the application UI as well as tools like Postman, and results were recorded to ensure correct backend behavior. The testing ensured that key workflows such as registration, login, and admin operations performed reliably and met expected input constraints. **Evidence:** Added adminRoutes.js to handle admin-specific CRUD operations · Screwlogist/Remix-Meals@e1bc290 testing screenshots for admin side · Screwlogist/Remix-Meals@58afde0 Performed negative testing on admin UI for missing fields, invalid em... · Screwlogist/Remix-Meals@077cbc0		

		Completed Postman testing for admin CRUD operations and route validat... · Screwlogist/Remix-Meals@b3bafb8		
Real Time				
Discuss with Tutor during workshop	☑ I did not directly interact with the tutor during workshop sessions, but we carefully reviewed the feedback provided on our submitted tasks and made the necessary changes to align our admin implementation with the project requirements. This included refining the logic for assigning the first registered user as admin and securing admin-only routes using middleware.			
Leadership		☑ I took full responsibility for the admin functionality development from planning to implementation even though I did not lead the entire project. I independently built all backend features for admin functionality which included user and recipe management through CRUD operations and role-based access control for admin routes. I performed all manual testing for the admin module as part of my responsibilities. I tested different system scenarios including unauthorized access attempts and input validation and response handling to verify proper system behavior and security. I tested each CRUD operation through both the user interface and Postman to verify their correct functionality.		

		I documented the admin flow through a system flowchart and confirmed that the frontend displayed admin features properly by using data from the backend. My concentrated work resulted in the completion of this essential application component which met both testing requirements and sprint deadlines. **Evidence:** https://trello.com/b/yqK47qHO/remixmeals		

Tools	Pass	Credit	Distinction	High Distinction
GIT		☑ During this sprint, I used Git to manage my contributions effectively. I committed code regularly with clear and descriptive messages and ensured that changes were pushed in an organized manner. I also merged updates responsibly and helped resolve minor conflicts to maintain a clean and stable main branch. My use of Git supported team collaboration and ensured that all admin-related code was version-controlled and traceable. **Evidence:** Screwlogist/Remix-Meals: This is the group project of SIT725		
Trello		☑ I used Trello to track and manage my assigned tasks throughout Sprint 2. I updated the status of my cards as I progressed through development, from "To Do" to "In Progress" and "Completed." I included short descriptions of the tasks, estimates, and occasional comments for context. This helped my teammates and tutor stay informed about my progress, especially regarding admin features and integration. While I didn't manage the whole board, I actively used it to stay aligned with the sprint workflow. **Evidence:** https://trello.com/b/yqK47qHO/remixmeals		
README file		☑ I contributed to improving the README file by reviewing the setup instructions and confirming they matched the updated backend and admin functionality. I added basic notes about the admin access flow (e.g., first user becomes admin) and ensured that any new routes or instructions I		

		worked on were documented properly. I helped keep the README clear and accurate for anyone setting up or testing the application, reflecting the current state of the project. **Evidence:** Updated README with admin access details and usage instructions · Screwlogist/Remix-Meals@60d4699		