PrepPal Release Summary

Team members

Name	Net ID	GitHub username	Role
Kenny Li	lik5	lik5	Developer, Test Manager
Ryan Loeppky	loeppk10	Rloeppky	Developer
Henry Wong	umwongh	hwbit	DevOps, Back-end developer
Frieda Bi	bif	bfrieda	Developer
Izan Cuetara Diez	cuetarai	algorizan	Front-end Developer

Project summary

Vision statement: PrepPal is a web-based tool that allows anyone to save their favorite recipes and easily plan their meals and shopping trips.

Users will be able to create their own recipes following a simple template. Users will also be able to explore and save recipes that were shared by other users. PrepPal can help users organize their recipes in a single location and incorporate recipes into their schedules.

Cooking for yourself can be a daunting challenge if you have no prior experience or knowledge. PrepPal will make it easy for people to prepare meals on a long-term basis by combining the work needed for shopping, prep work, and cooking in one place. With PrepPal, home cooks can search for recipes with specific ingredients, decide on what they want to eat in advance, and follow the instructions given to buy ingredients and prepare meals.

GitHub repository link

https://github.com/hwbit/PrepPal

DockerHub repository link

1) Frontend: https://hub.docker.com/repository/docker/hwbit3/preppal-fe/general
   Backend: https://hub.docker.com/repository/docker/hwbit3/preppal-be/general
2) Instructions: https://github.com/hwbit/PrepPal/blob/main/documentation/instructions.md

List of user stories for each sprint

Sprint 2

- US #1: Profile Editing [Status: Pushed]
- US #2: Discovering New Recipes [Status: Pushed]
- US #3: Custom Recipe Creation [Status: Pushed]
- US #4: Private/Public Recipes [Status: Pushed]

Sprint 3

- US #5: Personalized Meal Planning [Status: Pushed]
- US #6: Follow Manager [Status: Pushed]
- US #7: Saving Recipes [Status: Pushed]
- US #8: Recipe Ratings [Status: Pushed]
- US #9: Recipe Comments [Status: Pushed]

Sprint 4

- US #10: Upload Profile Image [Status: Pushed]
- US #11: Sharing Meal Plan/Calendar [Status: Removed]
- US #12: Sharing Shopping List [Status: Removed]
- US #13: Push Notifications [Status: Removed]

US #14: Customization For Notification Preferences [Status: Removed]
US #15: Manually Add Ingredients to Shopping List [Status: Removed]
US #16: Automated Shopping List Generation [Status: Removed]
Release

User manual

1. Account Management
    a. Login
    b. Registration
    c. Edit your profile
    d. View all recipes of another user
    e. Follow another user
2. Recipes
    a. View a recipe
    b. Create a recipe
3. Collections
    a. Favorite a recipe
    b. View your favorited and created recipes
4. Search
    a. Quick search for a recipe
    b. Search for a recipe by title/author/description/ingredients/cook time
5. Calendar/Meal Planner
    a. Save a recipe on the calendar
6. Reviews and Comments
    a. Leave a rating and comment on a recipe
    b. Where to read comments

Overall Arch and Design

https://github.com/hwbit/PrepPal/blob/main/documentation/architecture.png

Infrastructure

Languages:

- Typescript

Technologies:

- React
- Node.js
- Express.js
- CORS
- MongoDB
- Azure

Other Tools:

- Figma
- ESLint
- GitHub
- GitActions
- Docker

Testing Tools:

- Automation and Regression: GitHub Actions
- Backend test: Jest, Supertest
- Frontend test: Jest, React-Testing
- Issue/Bug-tracking: GitHub Issues

- Load testing: Jmeter
- Manually Api testing: Postman

Name and link

Front end server was built with React framework, the backend built with Node and Express, and Database was hosted on MongoDB Atlas. We chose React because of its popularity and because we could carry the knowledge over to future projects. We chose to implement a noSQL database since a lot of data was text-based and each data object is unrelated to one another. None of us had full stack development experience, so we didn't really have a preference as to what our tech stack should look like. We agreed to this tech stack as it is well-supported and documented.

Naming Conventions

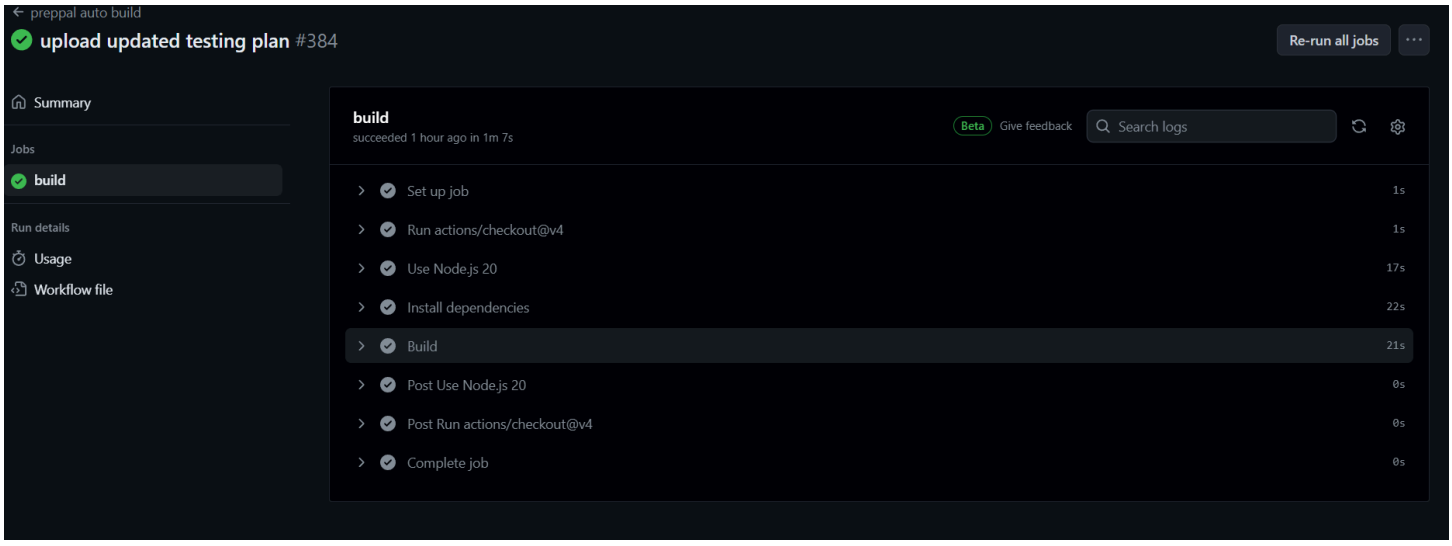
CamelCase
PascalCase

Code

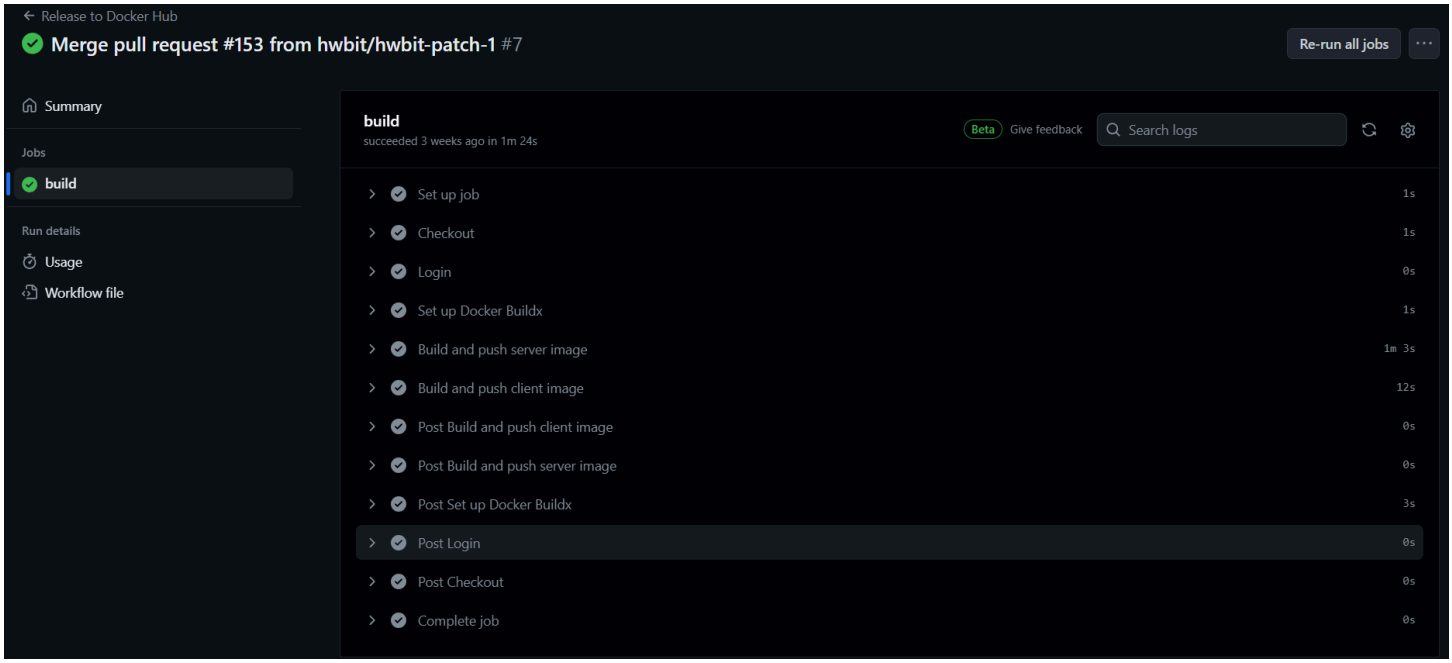
File path with a clickable GitHub link	Purpose (1 line description)
https://github.com/hwbit/PrepPal/blob/main/preppal-be/routes/userApi.ts	Does all user account related activities
https://github.com/hwbit/PrepPal/blob/main/preppal-be/routes/recipeApi.ts	Does all recipe related activities
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/components/recipe-card/recipe-card.tsx	Recipe card - give a short description of the recipe that is reused in many places
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/components/nav-bar/nav-bar.tsx	Navigation bar
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/pages/recipe.tsx	Displays the recipe page

Continuous Integration and deployment (CI/CD)

1) The CI/CD used GitHub Actions. https://github.com/hwbit/PrepPal/actions
2) CI - Build:



CD - Deploy to DockerHub:



Testing

Testing plan: https://github.com/hwbit/PrepPal/blob/main/documentation/PrepPal%20Testing%20Plan.pdf
Note: There are many tests in a single file

Unit Tests

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/authApi.test.ts (test name: "correct test - login successful”)	Test successful login
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/authApi.test.ts (test name: "incorrect test - incorrect password")	Test unsuccessful login
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/userApi.test.ts (test name: "correct test - lookup user")	Test to see if user exists (lookup)
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/userApi.test.ts (test name: "correct test - creating a new user")	Test create account
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/userApi.test.ts (test name: "correct updateUser test - updating an existing user")	Test update user/account
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/userApi.test.ts (test name: "correct saveRecipe test - save recipeId")	Test user saving/favouriting a recipe
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/recipeApi.test.ts (test name: "correct test - lookup a valid recipe by name")	Test looking up a recipe
https://github.com/hwbit/PrepPal/blob/main/preppal-be/	Test creating a recipe

tests/recipeApi.test.ts (test name: "correct test - create a new recipe")	
https://github.com/hwbit/PrepPal/blob/main/preppal-be/tests/recipeApi.test.ts (test name: "correct test - post review with review already present")	Test adding a review
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/__tests__/pages/create-recipe.test.tsx (test name: "Render standard component --> Input title fields")	Test to ensure all the form fields exists to create a recipe

Integration Tests

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/__tests__/pages/signup.test.tsx (test name: "Successful signup updates the navigation bar display")	User signing up
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/__tests__/pages/login.test.tsx (test name: "Successful login in updates the navigation bar display")	User successfully logging in and seeing the navigation bar change
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/__tests__/pages/home.test.tsx (test name: “Failed login in does not update the navigation bar display")	Unsuccessful login does not give you additional features
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/__tests__/pages/collections.test.tsx (test name: "Should populate the tabs with recipes")	Populate collection pages
https://github.com/hwbit/PrepPal/blob/main/preppal-fe/src/__tests__/pages/profile.test.tsx (test name: "Clicking on other's profile should display recipes")	Other user’s profile page displaying recipes

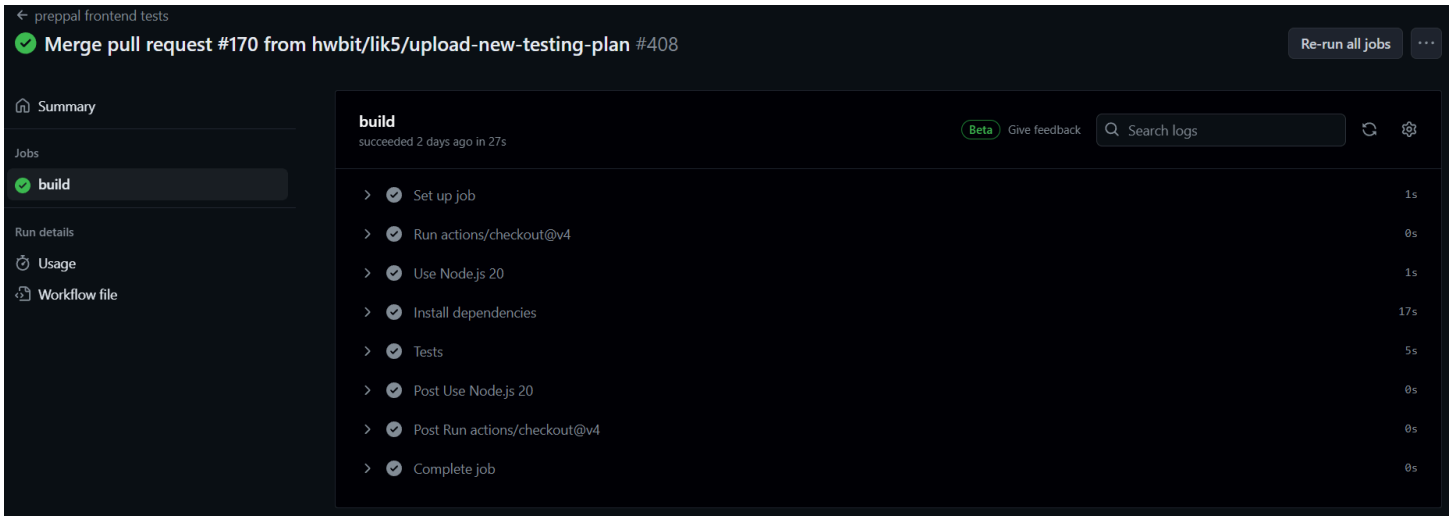
Acceptance Tests

Test File path (if you automated the test) or as comments in Github issues (if it is with clickable GitHub link)	Which user story is it testing
https://github.com/hwbit/PrepPal/blob/main/cypress/e2e/login.cy.ts (test name: “Successful login in as user”)	User successfully logging in
https://github.com/hwbit/PrepPal/blob/main/cypress/e2e/signup.cy.ts (test name: "Successful signup as user")	User sign up
https://github.com/hwbit/PrepPal/blob/main/cypress/e2e/collections.cy.ts (test name: "View own recipes as user")	View own recipes as a user
https://github.com/hwbit/PrepPal/blob/main/cypress/e2e/collections.cy.ts	View saved recipes as a user

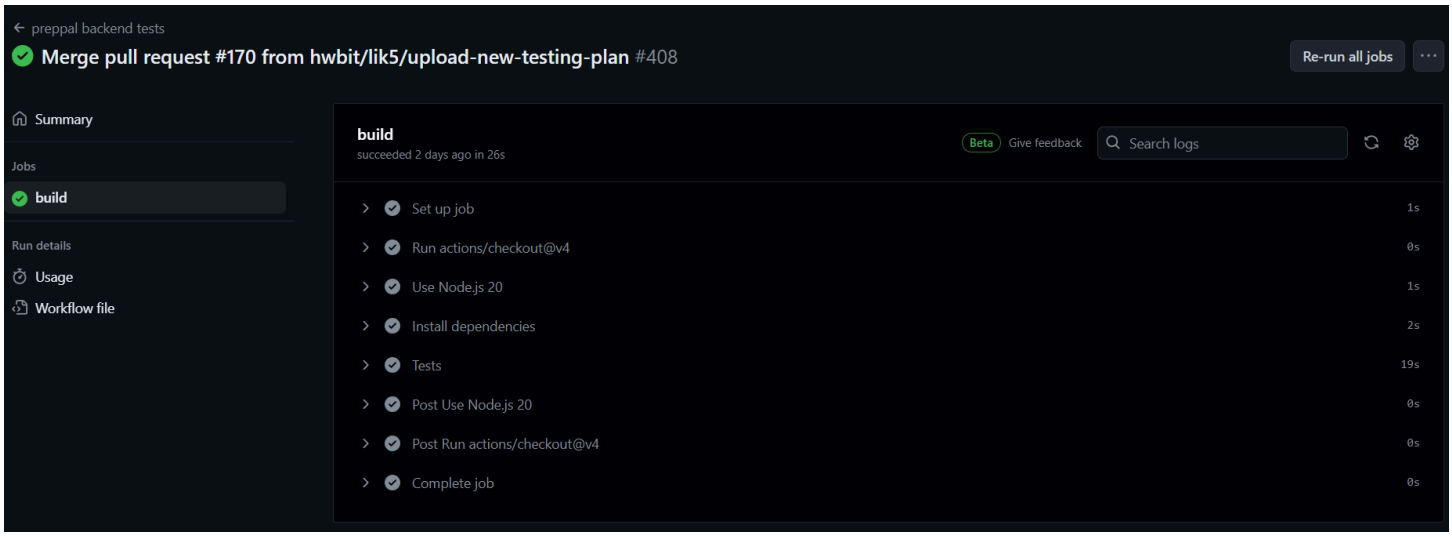
(test name: "View saved recipes as user")	
https://github.com/hwbit/PrepPal/blob/main/cypress/e2e/collections.cy.ts (test name: "Add recipe to my collection as user")	Create my own recipe

Regression testing

1) Regression (unit) tests are run every time a new commit is pushed to the main and dev branches via GitHub Actions. The tests use Jest, supertest, and React Testing Library.
2) Frontend regression tests: https://github.com/hwbit/PrepPal/tree/dev/preppal-fe/src/__tests__/pages
Frontend workflow script: https://github.com/hwbit/PrepPal/blob/dev/.github/workflows/preppal-be-tests.yml
Backend regression tests: https://github.com/hwbit/PrepPal/tree/dev/preppal-be/tests
Backend workflow script: https://github.com/hwbit/PrepPal/blob/dev/.github/workflows/preppal-fe-tests.yml



https://github.com/hwbit/PrepPal/actions/runs/8533190634/job/23375518933



https://github.com/hwbit/PrepPal/actions/runs/8533190633/job/23375518820

Load testing

1) Testing script: https://github.com/hwbit/PrepPal/blob/main/jmeter/testplan.jmx
Environment: Used JMeter for load testing. 20 concurrent users each making 10 requests for each request given (200 requests in total per request given).
There are 10 requests being tested:

1. Get recipes for home page
2. Get a recipe by id
3. Search recipes
4. Get reviews by recipe id
5. Login attempt with valid user
6. Get calendar
7. View own recipes
8. View own profile
9. View saved recipes
10. View other users' profiles

2) Test report for load testing.
Testing results: https://github.com/hwbit/PrepPal/blob/main/jmeter/testing_results.csv
Summarized testing results: https://github.com/hwbit/PrepPal/blob/main/jmeter/testing_summary.csv
3) Bottleneck found: The request to get own/others recipes

- Due to using a non-relational database structure (JSON) but making it relational. This caused additional server-side processing and ultimately slowing down the app

Security analysis

1) Security analysis tool and usages:

- There were several options that we found for Typescript, which was our most used language. The first one we tried was **Snyk**, due to GitHub containing a supported third-party workflow template to automatically run the scan.
- To run the workflow, we needed to update the template slightly as it was out-of-date to include the required secret key in all jobs run, and set up the environment variable in GitHub environment secrets.
- This workflow was set to run on any pull request into dev, main, or deploy branches, as well as whenever we push any of those three branches.
- Any and all security issues are then recorded and displayed under the Security tab on GitHub.

2) Security link: https://github.com/hwbit/PrepPal/security/code-scanning
Security issues as PDF saved as "Code scanning alerts 1-3.pdf" under documentation/Code scanning alerts on our repo. https://github.com/hwbit/PrepPal/tree/main/documentation/Code%20scanning%20alerts

Security Issue	Description
Regular Expression Denial of Service (ReDoS) https://github.com/hwbit/PrepPal/security/code-scanning/23	The regular expression is vulnerable to DoS attacks since we are not properly sanitizing user inputs.
Hardcoded Secret https://github.com/hwbit/PrepPal/security/code-scanning/1	We are showing our secret JWT key to the public which could be reverse engineered and exploited to gain information or access to things the attacker is not authorized to see.
Code Injection https://github.com/hwbit/PrepPal/security/code-scanning/6	Recipe card is vulnerable to code injection since we are just grabbing the description text from our database without any sanitization.
Information Exposure https://github.com/hwbit/PrepPal/security/code-scanning/24	We are showing what framework we are using and it could be exploited if future vulnerabilities are found.

Use of Hardcoded Credentials https://github.com/hwbit/PrepPal/security/code-scanning/5	Even if it is a test account, we shouldn't code a password or remove it before it reaches production.