Tastely Progress Report 3

Team Members: Yug Patel, Kush Solanki, Arjun Kirubakaran, Abraham Yirga

Progress This Week

This week marked the final development sprint for Tastely, where we focused on completing core functionality, adding test coverage, and refining the user experience. One of our key accomplishments was completing the recipe CRUD feature set. Users can now create, view, update, and delete their recipes through a simple interface, and all recipe modifications are stored reliably in the backend database. We also finalized the cookbook system, allowing users to save and manage a personalized list of their favorite recipes.

We successfully integrated the Spoonacular API to import a diverse set of starter recipes tagged by cuisine and dietary restrictions. These imported recipes appear in the search feed and provide users with content to engage with from the start. Alongside this, we expanded the frontend to support better user interaction, including improved validation for forms (login, signup, and recipe creation) and responsive layout tweaks using Tailwind CSS.

One major addition this week was the introduction of automated testing. We implemented backend test cases using pytest, covering key areas such as user authentication, recipe creation, cookbook management, and error responses for invalid requests. This has helped us catch and resolve edge cases and regressions, and we now have a baseline testing framework for future development. Tests are run manually for now, but we've written them in a modular fashion so they can be integrated into a CI/CD workflow post-MVP.

What Went Well

We are pleased to report that most of the core MVP features outlined in our original plan have been completed. Backend and frontend integration has matured significantly, and the API contract between the two layers is consistent. The database schema has remained stable with only minor adjustments, and the modular design allowed us to add features like third-party recipe importing with minimal disruption.

Team communication and division of labor were major strengths throughout this sprint. Tasks were managed via GitHub Projects, and all code changes were tracked and reviewed through pull requests on feature branches. This disciplined workflow allowed us to maintain clean, functional code with minimal bugs and redundancy. The inclusion of pytest this week added technical depth and increased our confidence in backend reliability.

Additionally, documentation was finalized for our API endpoints, database schema, and local deployment steps. UML diagrams were also updated to reflect changes made during this sprint. These additions will help future maintainers or collaborators understand the structure and logic of our codebase more quickly.

What Could Be Improved

Despite solid technical progress, we faced some limitations in polish and testing scope. While pytest helped us establish backend reliability, we were not able to implement frontend test automation tools like Cypress due to time constraints. Input validation on the frontend, though functional, could be further enhanced with clearer error prompts and client-side feedback. The styling of some UI components is also basic and could be made more engaging.

Another improvement area is deployment. Although we were able to test the application locally and simulate production behavior, we did not reach the stage of deploying Tastely to a live environment like AWS or Render. This would be a natural next step for showcasing the platform beyond our local machines. Lastly, while we documented major workflows, a more detailed user onboarding tutorial or in-app guidance could make the experience smoother for first-time users.

Sprint Retrospective – Start, Stop, Continue

Start
Start writing tests alongside feature development instead of waiting until the final sprint. Begin implementing frontend testing frameworks like Cypress early in the sprint to avoid falling short on UI-level test coverage. Start integrating CI tools such as GitHub Actions to automate tests and catch failures quickly. Also, begin using prototyping tools like Figma for early UI iterations to clarify user flows before implementation.

Stop
Stop pushing visual polish and UI tweaks to the very end of the sprint. These often consume more time than expected and could have been addressed incrementally. Stop assuming third-party APIs will integrate smoothly without data transformation—future integrations should be prototyped earlier. Finally, stop manually testing endpoints when automated tests could be used—this slowed down validation.

Continue
Continue using GitHub feature branching and PR reviews as a development workflow. Continue dividing tasks across backend, frontend, and documentation work to avoid bottlenecks. Continue maintaining clean, modular, and well-documented code. Most importantly, continue our focus on delivering a usable, functional product first, then refining it iteratively.

This concludes our final progress report. Tastely is now a functioning MVP that allows users to register, log in, create and manage recipes, maintain a cookbook, and interact with curated external content. Although deployment and frontend automation remain as next steps, the core system is stable, tested, and ready for demonstration. The team has grown significantly in full-stack development skills, testing practices, and collaborative software engineering throughout this project.