Capstone Progress Report

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

Tastely is a social cookbook platform designed to help users share recipes, discover global cuisines, and personalize their culinary experience through profiles, dietary preferences, and interactive features. Users can create accounts, upload or save recipes, and build a digital cookbook. Admins will have the ability to manage the recipe library, ensuring quality and moderation. The main goal of Tastely is to foster culinary exploration and community by combining recipe logging with social functionality.

The original vision has stayed intact, but we've streamlined our initial goals to focus on a solid MVP that supports user registration, profile creation, recipe posting, and cookbook management. We also plan to integrate a third-party recipe API to preload the platform with a general selection of diverse recipes, so users have content to explore right away, even before a large user base develops.

We've completed backend setup with Flask and started building REST API endpoints for user authentication, recipe upload, and cookbook management. Frontend components are being developed using plain JavaScript with Tailwind CSS for styling. The project is currently in its foundational phase with core functionality underway, and a functional database schema already in place.

Software Design and Architecture

Tastely follows a modular two-tier architecture:

- Frontend: Built with HTML, JavaScript, and styled using Tailwind CSS. The UI interacts with the backend through REST API calls.
- Backend: Flask handles routing, logic, and all API endpoints. Services are separated into modules for user auth, recipe management, and admin operations.
- Database: A relational database (SQLite in development, with plans to shift to PostgreSQL in deployment) stores users, recipes, ratings, and cookbook information.

Design patterns implemented or planned:

- Factory Pattern: Used for generating user types (admin, regular user) and recipe objects.
- Singleton Pattern: Applied to manage the database connection object in Flask.

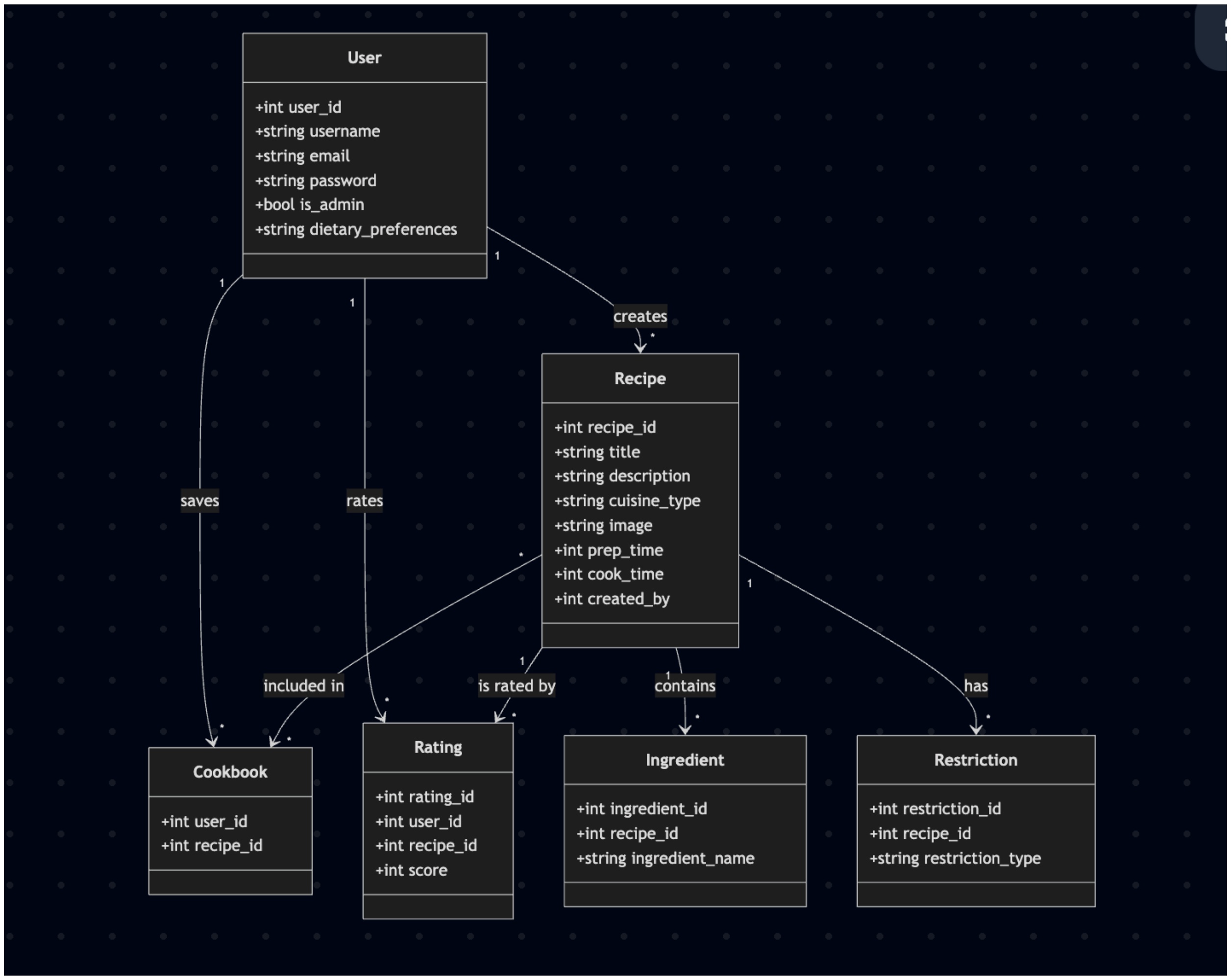
- MVC Pattern: Flask routes act as controllers, HTML templates serve as views, and database models handle logic.

Database schema includes:

- users: user_id, username, email, password, is_admin, dietary_preferences
- recipes: recipe_id, title, description, created_by, cuisine_type, image, prep_time, cook_time
- ingredients: ingredient_id, recipe_id, ingredient_name
- restrictions: restriction_id, recipe_id, restriction_type
- cookbook: user_id, recipe_id
- ratings: rating_id, user_id, recipe_id, score

We are also developing a script to fetch a set of default recipes using an external API (Spoonacular). These will populate the database with globally diverse dishes tagged with cuisine and dietary info. These system-generated recipes will not be editable by regular users.

UML Diagram;



Code Management
Version control is managed using GitHub. The repository is organized into folders for /static, /templates, and the main root directory contains database and backend files.
Branching strategy:

- main: stable production-ready code
- dev: current working version
- feature/*: branches for each new feature or bugfix

All code changes are made through pull requests. Peer review and testing are done before merging to dev. A GitHub project board is used to manage tasks and track development progress.

GitHub Repository:
https://github.com/pateyu/Tastely

Timeline Assessment

Milestone	Planned	Actual	Status
Setup dev environment	Week 1	Week 1	Complete
Database schema design	Week 1	Week 1	Complete
Flask backend setup	Week 2	Week 2	Complete
Auth & user profile APIs	Week 3	Week 3	In Progress
Frontend structure	Week 3	Week 3	Complete
Recipe CRUD API	Week 4	TBD	Not Started
Third-party API ingest	Week 5	TBD	Not Started

So far, we are on schedule. Backend and frontend foundations are complete, and user authentication is being finalized. We may push certain advanced features (like comment threads) to a later phase to ensure a stable and functional MVP.

Next Phase Implementation Plan
In the next sprint, we will:

- Finish user login and profile setup
- Complete recipe upload, edit, delete features
- Build cookbook saving functionality
- Integrate third-party recipe API to preload database
- Build user dashboard and frontend interaction

Design patterns planned for next phase:

- Observer Pattern: To notify users of new recipe suggestions or updates

- Strategy Pattern: For applying dynamic filters to the recipe feed (e.g., sort by cuisine, time, rating)

Testing strategy:

- Backend unit tests using unittest in Python
- Postman for API testing
- Manual UI walkthroughs to validate user flows
- Planned future integration with Cypress for automated frontend tests

Technical resources required:

- Spoonacular or similar API access for recipe seeding
- Deployment environment (AWS)
- PostgreSQL migration for production
- GitHub Actions for future CI/CD setup

This reflects the current status of Tastely and our direction for the coming development phases. The core architecture is in place, and we're actively building toward a fully functional MVP that supports both user-generated and API-integrated recipe content.