Stove & Oven

Tanvi Hanamshet

Sadie Stokes

Abhijit Admin

Robert Chin

Yinghui Cai

GitHub Url: https://github.com/rasodu/Prep_Your_Meal

INTRODUCTION

Stove & Oven is a recipe planner application where user will be able to add required ingredients and can get different recipe suggestions with respect to the ingredient list. This is the basic functionality.

Problems: As the previous interface wasn't that knowledgeable to the user, so we will be working on modifying the interface. Also we will be adding local authentication.

User story: As a user I will be able to navigate to the next steps required to achieve the desired task.

Architecture Review Documents:

The system architecture / documents (existing) will depict the overall system architecture of the end-to-end application (i.e. Heroku (ReactJS, ExpressJS) and MongoDB Atlas, at this time).

The project's future security architecture documentation will depict the admin interfaces (i.e. developers, testers, SAs) user access control to the application, source code repo/GitHub, the DB, and the Heroku "server" administration, user/customer interfaces, and will depict the separation of development/test/staging from production.

The GitHub / code repo will have role based access supporting developers and SAs. And testers will be able to report issues (or future feature requests via GitHub Issues). Heroku "server" administration or the DB will be controlled/managed by SAs or DBAs. Testers/Customers will have access to Internet facing Staging/Production environments.

It will depict the end customer user interface to view/plan recipes and meal planner (and controls using authentication mechanisms), but also the possible integration or fallback to local authentication (note, depending on the scoping of local authentication and its integration, would drive additional security architectural considerations… i.e. more complexity on passwords, password change intervals, inactivity of accounts… etc).

Physical separation of user data from application recipe/ingredient data, and possible scaling to support large customer/user base or large recipe/ingredient repository for production scaling purposes

Physical separation of development/test/staging data from production data

The architecture (while considering the data or what it'll be classified as) will provide guidance/direction of whether encryption of data in transit across interfaces between Heroku <-> MongoDB Atlas (at this time), or whether encryption of data at rest in storage of data inside MongoDB Atlas (at this time).

Further security architectural considerations would be to take in account of risks of Internet facing interfaces and potential safeguards/controls, but ultimately - would not rely on "free-based" services such as Heroku and/or MongoDB Atlas.

ROLES AND RESPONSIBILITIES

Build Master/ Server Admin – Abhijit, Ben

Developers - Ben, Sadie

DBA/ DB Server - Tanvi

Project Manager - Tanvi

Testers - All

Test Lead - Abhijit

Documentation - All

Designer - Ben, Sadie

Architect - Rob

Security Architecture - Rob

METHOD

Software:

- MERN (Mongo, Express, React, Node)
- Linux based Docker containers
- Node.js libraries
- ESLint, Prettier

Frequent Database Backup

Review Process:

Design Review

- Security Review
- Code Review
- Architects will meet to review ideas
- We will perform informal reviews
- The team will be responsible for reviews and their respective items/topics

Build Plan:

- GitHub
- We will merge all dev branches at the end of each sprint o 2 Weeks
- Staging and Production
- Regression test process – see test plan

Modification Request Process:

- GitHub Pull Requests, GitHub Issues
- Trello

Virtual and Real Workspace

Entire team is virtual, we will use Google Drive/GitHub for documentation

COMMUNICATION PLAN

"Heartbeat" meetings

Once a week meeting for status/help needed. Team will keep in communication during the week using Whatsapp

Status meetings

Once a week virtual status meeting.

Issues meetings

Once a week virtual status/issues meeting.

TIMELINE AND MILESTONES

They are:

- **Week of Feb 10th – description of first demo**
- **Week of Feb 24th first demo, description of second demo**
- **Week of March 23th second demo, description of third demo**
- **Week of April 6rd third demo, description of fourth demo**
- **Week of April 20th fourth demo, description of final product**
- **Week of May 4th final product**

TESTING POLICY/PLAN

Testing will be performed every sprint, or when the feature is ready. We are planning to implement testing automatically.

RISKS

- **We will estimate the time required for building any feature and keep everyone updated using Trello.**
- **We will plan the next steps in every sprint meeting.**
- **We will consider listing every possible worst scenario heaviest-use scenario.**

- **We will follow a strict schedule, such that we will complete all the features on time.**
- **As initially the specifications are not clear, we will start building concrete specifications and requirements for the project.**

ASSUMPTIONS

- **We are assuming that the entire team is somewhat familiar with the programming language and database that is being used**
- **We are also assuming that the roles each person has selected is according to their expertise**
- **The existing system architecture (defined in 690) will not drastically change**
- **Additional (possible) fallback local login functionality will not cause large delays or conflicts with the existing Google Authentication**