MealWize

Group 4

Your Recipes, Your Goals, Your Community

A web application that redefines how you explore recipes and manage your health

- Integrated Solution
- Personalized Experience
- Cultural Inclusivity
- Holistic Diet Tracking
- Streamlined Meal Planning

Background

- Fragmented Experience
- Limited Control
- Complex/Intrusive Sign-up processes
- Insufficient Integration

Goals

- Combined recipe/meal planning and health goal tracking application
- Help users keep their health goals in mind when they decide to cook meals
- User friendly
- Maximum functionality without requiring users to make an account
- Allow users to share their own recipes with others
- Let users choose their own meals
- Personalized Recommendations

Users

- Anyone who wants to cook
- Looking for new recipes
- Health-focused
- Want to share recipes with others
- Not health focused, but trying to be
- Careful about counting calories and nutrients

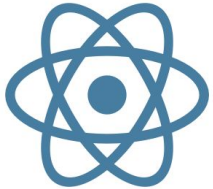
Features

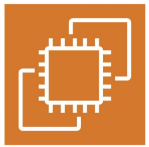
- Recipes:
  - Searching
  - Uploading
  - Favoriting
  - Personalized recommendations
- Shopping cart:
  - Get ingredients for different recipes in different quantities
- Diet tracking:
  - Track recipes from MealWize
  - Track meals outside the app
  - View metrics

Process

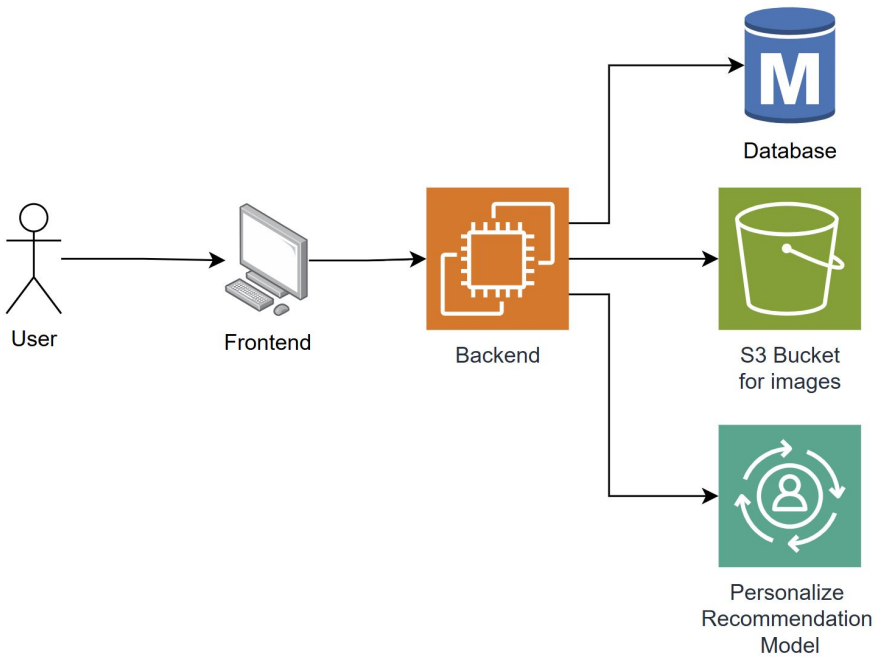
Tools

- **IDEs:**
  - IntelliJ
  - VS Code
- **Languages:**
  - Typescript/React (Frontend)
  - Java/Spring Boot (Backend)
  - MySQL (Database)
- **AWS Services:**
  - EC2
  - RDS
  - S3
  - Personalize





Design



- Traditional frontend/backend/database website architecture
- Frontend
  - Component reuse and data encapsulation in contexts
  - Protected routes and authentication context ensure correct access authorization
- Backend
  - 36 API endpoints
  - Modular code for abstraction and reuse
- Database
  - 12 tables including recipes, users, meals, etc.
- Followed industry standard design patterns
- Choice of tools and architecture motivated primarily by team members' prior experience

Difficulties

- AWS
  - Took us longer than expected to coordinate with the TAs to get the AWS access we needed
  - Differences in app setup between our local machines and AWS
  - We weren't careful at first with our AWS Personalize model, which caused us to go over budget during testing. This resulted in more strict monitoring of our AWS usage going forward
- Matching team members' skills with different parts of the project
- Agreeing on how each feature is expected to work in detail
- Getting sample data for both testing and training
- Went back and forth agreeing on the specific API for each endpoint.

Demo

Final Demo Link

Conclusion

Reflection

Some good:

- Our capacity planning was fairly accurate. We planned and distributed features across sprints well. We didn't have to rush due to overloading any sprints.
- Everyone was accommodating and helpful.
- Following industry standards was helpful in the long run.

Some not-so-good:

- On the flip side, we probably could have added one or two more small features
- Planning with group members working full-time jobs can be difficult

Future Work

Model adjustments based on —

- Community features – comments and blogging
- Rating recipes
- Mobile application
- OAuth2 (sign in with Google/Facebook/etc.)
- Improve performance
- Integrate with other sources of recipes
- Moderation of user-uploaded content (either AI or manual moderation)
- Real-world testing from users
- Improve error handling and notifying users of errors
- Formally document APIs using Swagger or a similar tool

Thank you!