Project Plan Presentation

ASE 485 Capstone Project

NomNom Safe

Focus: Business-Facing MVP & Learning with AI

Problem Domain

- Food-service business data management
    - Primarily affecting small businesses
        - Locally- and independently-owned
        - Limited skills, time, & resources
    - Inconsistent, inefficient, or non-existent
    - No common, public location to manage or access data

Problem Importance & Impact

- Reputation
  - Unsafe, unaccommodating, uncaring, untrustworthy
- Limited customer reach
  - 53% of households report having dietary restrictions (source)
- Legal liability
- Wasted time (staff & customers)
- Dangerous miscommunications

Solution

NomNom Safe provides:

- Centralized platform for managing, viewing, & sharing menu & dietary data
- Intuitive, streamlined onboarding & data management tools

Solution Importance & Impact

- Strengthens business reputation
  - Demonstrates transparency, awareness, & care
- Expands customer reach
- Eliminates inconsistency & inefficiency
- Reduced risk
  - Minimized chance of miscommunications
  - Lowered risk of outdated data

Sprint 1

Focus: Business-Facing Side

> Context: A working version was developed in ASE 285

Goal:

- Functional, maintainable, & extensible **business-side MVP**

Features

1. Switch backend to Firebase
2. Refactor using SW Design Patterns & Principles
3. Improve navigation
4. Implement responsive design

Feature 1 Purpose

1. Switch backend to Firebase

Have a single source of data for user- and business-facing sides.

- Current backend uses MongoDB
- Convert backend to access the Firebase DB currently being used by the user-facing side of the app

Feature 2 Purpose

2. Refactor using SW Design Patterns & Principles

Refactor `server` & `client/src/components/auth` folders to enforce maintainability & extensibility by applying:

- SOLID principles
- DRY principle
- Separation of concerns (separate business logic from UI)
- SW Design Patterns

Feature 3 Purpose

3. Improve Navigation

Enhance user experience by implementing intuitive navigation consistently across the app.

- Develop & implement consistent locations for navigation & navigation methods
- Add navigation to pages lacking it
- Add/clarify navigation indicators

Feature 4 Purpose

4. Implement Responsive Design

Enhance user experience across different devices by implementing a clean, flexible design that adjusts properly with different screen sizes.

- Standardize styles across different components
- Standardize layouts across different pages
- Ensure each component & layout adjusts to prevent horizontal overflow

Sprint 1 Requirements Overview

Comprehensive requirements can be found on the GitHub repo

Total # of requirements: 41

Feature 1 Requirements

Feature 1: Switch backend to Firebase
# of Requirements: 8

Requirements:

- Migrate the backend fully to Firebase by replacing all MongoDB/Mongoose logic with Firestore equivalents and updating API routes to use a clean service-layer architecture.
- Maintain system quality by validating data with Zod, securing Firebase credentials through environment configuration, and ensuring performance and behavior remain consistent or improved after the migration.

Feature 2 Requirements

Feature 2: Refactor using SW Design Patterns & Principles
# of Requirements: 11

Requirements:

- Strengthen system reliability and maintainability by enforcing consistent behavior, stable updates, and standardized patterns such as unified error/loading handling and centralized API logic.
- Refactor using clean architecture and SOLID principles—removing duplication, separating concerns, breaking down large components, and improving naming and documentation to support long-term scalability and contributor onboarding.

Feature 3 Requirements

Feature 3: Improve Navigation
# of Requirements: 11

Requirements:

- Strengthen navigation clarity and usability by creating intuitive entry points, consistent patterns, and responsive, accessible layouts supported by contextual aids like breadcrumbs.
- Ensure smooth, uninterrupted movement between pages by preserving user state, preventing broken routes, and providing helpful feedback such as loading indicators during transitions.

Feature 4 Requirements

Feature 4: Implement Responsive Design
# of Requirements: 11

Requirements:

- Deliver a fully responsive interface that adapts fluidly across mobile, tablet, and desktop through flexible layouts, scalable typography, and properly resizing visual elements.
- Maintain a professional, accessible user experience by ensuring consistent visuals, eliminating horizontal scrolling, and supporting responsive navigation patterns that work cleanly on all screen sizes.

Sprint 1 Milestones

> 4-week sprint (2/2/2026 - 3/1/2026) with weekly goals

Week 1: Convert Backend to Firebase

- **Dates:** 2/2/2026 – 2/8/2026
- **Focus:** Convert the entire backend to accommodate a Firebase DB
- **Requirements:** 1.1-1.8

Week 2: Refactor Foundation

- **Dates:** 2/9/2026 – 2/15/2026
- **Focus:** Lay the foundation for refactoring the `server` & `auth` directories using Software Design Principles
- **Requirements:** 2.1-2.11

Week 3: Complete Refactor + Improve Navigation

- **Dates:** 2/16/2026 – 2/22/2026
- **Focus:** Complete the refactor and improve navigation across the app for consistency & simplicity
- **Requirements:** 3.1-3.11

Week 4: Implement Responsive Design

- **Dates:** 2/23/2026 – 3/1/2026
- **Focus:** Implement a consistent design across the app that responds appropriately to different screen sizes
- **Requirements:** 4.1-4.11

Directory Structure

Directory Structure Overview

```
Business-Side/
├── client/                 # Client side
│   ├── __tests__/
│   │   ├── integration/
│   │   └── setup/
│   ├── public/
│   ├── src/
│   │   ├── assets/
│   │   ├── components/
│   │   ├── styles/
│   │   ├── utils/
│   │   ├── App.jsx
│   │   └── index.jsx
│   └── package.json
├── docs/                   # Project & individual documentation
│   ├── individual/
│   │   ├── anna-dinius/
│   │   └── jeff-perdue/
│   └── modules/
├── scripts/                # Python scripts that aid in documentation
├── server/                 # Server side
│   ├── __tests__/
│   │   ├── integration/
│   │   └── setup/
│   ├── src/
│   │   ├── config/
│   │   ├── models/
│   │   ├── routes/
│   │   └── utils/
│   ├── package.json
│   ├── README.md
│   └── server.js
├── package.json            # Top-level package.json for installing all dependencies with a single command
└── README.md
```

Client-Side Directory Structure

```
client/
├── __tests__/                            # Multi-component tests (e.g. integration, e2e)
│   ├── integration/
│   └── setup/
├── public/
│   └── index.html
├── src/
│   ├── assets/
│   │   ├── icons/
│   │   └── images/
│   ├── components/
│   │   ├── setup/                        # Components grouped by category
│   │   │   ├── ChooseBusiness/           # Individual component folders
│   │   │   │   ├── ChooseBusiness.jsx
│   │   │   │   └── ChooseBusiness.scss   # Colocated stylesheets
│   │   │   └── ...
│   │   └── ...
│   ├── styles/                           # Global/shared styles
│   │   └── global.scss
│   ├── utils/
│   ├── App.jsx
│   ├── App.test.js                       # Colocated unit test files
│   └── index.jsx
└── package.json                          # Separate package.json for client side dependencies
```

Server-Side Directory Structure

```
server/
├── __tests__/                  # Multi-component tests (e.g. integration, e2e)
│   ├── integration/
│   └── setup/
├── src/
│   ├── config/
│   │   ├── db.js
│   │   └── db.test.js          # Colocated unit test files
│   ├── models/
│   │   ├── Business.js
│   │   ├── Business.test.js    # Colocated unit test files
│   │   └── ...
│   ├── routes/
│   │   ├── adminRoutes.js
│   │   ├── adminRoutes.test.js   # Colocated unit test files
│   │   └── ...
│   └── utils/
├── package.json                # Separate package.json for server side dependencies
├── README.md
└── server.js
```

Docs Directory Structure

```
docs/
├── individual/
│   ├── anna-dinius/             # Anna's individual progress reports & presentations
│   │   └── ppp-dinius.md
│   └── jeff-perdue/             # Jeff's individual progress reports & presentations
│       └── ppp.md
├── modules/                     # Docs for each module
│   ├── AdminModule.md
│   ├── AuthenticationModule.md
│   └── ...
├── file-structure.txt           # Comprehensive file structure of entire repo
├── Requirements.md
└── UserManual.md
```

Learning with AI

Topics

1. Food allergen ontology
2. Market validation & user-focused research

Topic 1 Goals

Food Allergen Ontology

Understand:

- Basic ontology concepts
- Ontology design through the lens of food allergens
- Importance of ontology

> More information can be found on the GitHub repo README

Topic 2 Goals

Market Validation & User-Focused Research

Explore:

- User outreach methods & effectiveness
- User testing methods & applications
- Data collection strategies
- Data analysis

> More information can be found on the GitHub repo README

Links

GitHub Repos

- NomNom Safe Business-Side repo
- Learning with AI repo

Canvas Pages

- Individual Project - Anna Dinius Canvas page
- Individual Progress - Anna Dinius Canvas page

Questions?