Salvar: A nutrition app with the apple watch that encourages better health habits

Final Proposal for ECE49595O

Dhruv Singh
Akash Lavu
Purdue University, Elmore Family School of Electrical and Computer Engineering
singh651@purdue.edu
alavu@purdue.edu

***Index Terms*—Organization, Source code repository, maintainability**

I. Introduction

This document outlines the repository architecture for the Salvar project, as well as the review & merge policies. This document is entirely intended for developers of the project.

II. Repository Architecture

The repository is structured to facilitate organization and collaborative development. It includes a clear file and directory layout, as well as a branching strategy that supports the project's workflow.

A. File & Directory Structure

The project adheres to common open-source practices for directory structures, with adjustments for the specific technologies and frameworks used. Key directories include "src" for source code, "docs" for documentation, "tests" for test scripts, and "assets" for non-code assets

The structure may evolve as the project grows to ensure maintainability and ease of navigation.

```
/salvar
|-- /app                          # React
|   |-- /src
|       |-- /components           # UI
|       |-- /navigation           # Nav
|       |-- /state                # State
|       |-- /services             # API calls
|       |-- /styles               # Styling
|       |-- /firebase             # Firebase
|           |-- /auth             # Authentication
|           |-- /database         # Firestore
|           |-- /storage          # Storage
|           |-- /functions        # Cloud
|
|-- /watch                        # Apple Watch
|   |-- /Interface                # UI
|   |-- /DataCommunication        # Data sharing
|   |-- /Performance              # Performance
|   |-- /Notifications            # Notifications
|
|-- /shared
|   |-- /data                     # Data sync
|   |-- /offline
|
|-- /tests                        # Testing
|   |-- /unit
|   |-- /e2e
|   |-- /performance
|
|-- /docs                         # Documentation
|   |-- README.md
|   |-- API_DOCS.md               # API usage
|   |-- CODING_GUIDELINES.md
|
|-- .gitignore
|-- package.json
```

B. Branching strategy

The project uses Git for version control with the following branching strategy:

Main Branches: main for stable releases and develop for ongoing development. Feature Branches: Prefixed with feature/ for new features Pull Requests: Feature branches are merged into develop via pull requests for review.

We will be creating multiple branches to develop different features. The application UI and apple watch features will be merged at the end. Contributors may also make separate branches to work on new features or to develop.

a) main branches: The two branches are release branches, named `main` and `release`. The former will contain all changes merged from feature branches, and the latter will contain the latest changes staged for the next release.

b) feature branch: We will make different branches based on the features being developed, as well as features that may have to be done separately due to testing with ios testing platforms such as Expo.

III. Code Review & Merge Policy

The code review policy will be the following:

- **Approval Count**: 1, as we are partners we can review each others code before pushing.
- **Merge Window**: Anytime, just discuss with group
- **Merge Target**:Feature branches merge into develop; develop merges into main for releases.
- **Proposal Structure**:
  - **Description**: changes proposed and motivation
  - **Relevant Issues**: a list of issues in our repo that relate to this
  - **Methods**: How did were changes implimented or fixes made.

IV. Conclusions

This document presented the final proposed design of the Ceiba application.

References