Team Working Agreements

Logistics

Work Room and Location

- **Remote Collaboration**: Our team will primarily collaborate remotely using Discord
- **In-Person Meetings**: We usually meet in room BE 302 for necessary face-to-face meetings.

Meeting Times

- **Weekly Stand-ups**:
    - Monday at 2:30 pm
    - Saturday at 12:00 pm
- **General Meetings:** Thursday at 1:45-3:00 pm
- **TA Meetings:** Tuesday 1:45-2:45 pm
- **Attendance**: All members are expected to attend all meetings. Exceptions are notified to the team in advance

Communication Channels

Discord will be our primary form of communication.

- **Channels**:
    - #general: for overall team communication.
    - #tasks: for technical discussions and task brainstorming
    - #times: for meeting-related announcements, updates, and notices.
- **Response Time**: Team members should check messages regularly within 24 hours on weekdays.

Project Repository

- **Version Control**: Our codebase is hosted on **GitHub** in a private repository.
- **Branching Strategy**:
    - Use feature branches for developing new features.
    - Merge changes into the main branch after code review.
    - The product branch contains stable releases only.
- **Commit Messages**: Must be clear and descriptive.

Organization

- **Project Management**: We use **Jira** for tracking tasks, bugs, and the project's progress.
- **Task Assignment**: Tasks are assigned during meetings or via Jira. Team members can pick tasks based on interest and expertise.

- **Sprint Planning**: This is conducted during general meetings to plan the upcoming week's work.

Development Environment

Platforms

- Development is done on personal computers running Windows, macOS, or Linux.

IDE

- The preferred IDE is **Visual Studio Code**, which all team members are using.

Other Tools

- **Node.js** and **npm** for server-side development.
- **Firebase** for backend services and database.
- **React** for frontend development.
- **Gemini** Model for meal generation.

Coding Styles/Standards

- Follow the Google JavaScript Style Guide.
- Use ESLint and Prettier for code linting and formatting.
- Write clear, maintainable, and well-documented code.

Work (Process) Patterns

Definitions of Done

A task is considered done when:

- The code is written, tested, and integrated without breaking existing code.
- The code has been peer-reviewed and approved by the team.
- Relevant documentation comments are updated.
- The feature is merged into the main branch.
- The task status is updated in Jira.

Team Collaboration

- **Knowledge Sharing**: Regularly share insights, challenges, and solutions during meetings and on Discord. Especially teammates working on adjacent tasks.
- **Respect and Support**: Foster a respectful environment where team members support each other's growth.
- **Pair Programming:** team members are to work on tasks that others have worked on to learn full-stack development and to have a comprehensive understanding of the project's inner workings.

Collaboration with Experts (SMEs)

- **TA Meetings**: Utilize scheduled TA meetings for guidance and feedback.
- **Documentation**: Record insights from SMEs in #general and share them with the team.

Areas of Responsibility

- Team members are expected to complete all work assigned to them on the group Jira.
- **Frontend Developers:**
    - Responsible for React components, UI/UX design, and client-side logic
- **Backend Developers:**
    - Manage server-side code, API integrations, and database interactions.
- **AI Integration:**
    - Handle OpenAI API interactions and AI feature development.
- **Quality Assurance:**
    - Oversee testing, bug tracking, and ensuring product quality.

Work Hand-off/Integration

- Team members must write sufficient documentation for another team member to take over the work being done.
- Team members must communicate the functionality within comments.
- Ensure the task is integrated into the project repository without breaking other features.
- Run all tests to confirm compatibility with the current codebase.
- Users must use pull requests for merging code. The reviewer will be the scrum master.
- Ensure all new code passes unit tests and does not break existing functionality.
- Notify team members when work is ready for hand-off or requires integration.

Product Design Patterns

UX/UI Look and Feel

- **Consistency**: All pages and parts should be consistent in design.

- **Accessibility**: Make the application accessible to all users.
- **Responsive Design**: The application should be compatible with different screen sizes and devices.
- **Design Framework**: Utilize established UI frameworks or libraries to speed up development. (DaisyUI + tailwindcss)

Product Architecture

- **Modularity**: Code should be modular for maintainability and reusability.
- **Separation of Concerns**: Keep the code for frontend, backend, and database concerns separate.
- **Scalability**: The architecture should be designed in a way that it can meet the demands of growth and other additional features.

Common Approach to Common Problems

- **Error Handling:**
    - Global error handlers in the client and server
    - Error messages to the user should not reveal sensitive information.
- **Data Validation:**
    - Client and server-side input validation prevent invalid data entry.
- **Security Practices:**
    - API keys or sensitive information shall be stored within environment variables.
    - Do not commit your credentials to the repository.
    - Keep dependencies up to date; most security vulnerabilities are fixed in newer versions.

Error Handling

- **Logging**: Provide logging features that capture errors and significant events (maybe use Winston or Morgan)
- **User Feedback**: Provide friendly error messages and lead users to solve issues.
- **Monitoring**: Use any monitoring tool to trace the performance and errors of applications in real-time.