Tender - A Swipe-Based Recipe Discovery & Meal Planning App

CS 4398/5394: Software Engineering Project - Assignment 1

Project Title: Tender - Swipe-Based Recipe Discovery & Meal Planning Application

Group Number: [TODO: Fill in your group number]

Group Members:

Name	TXST Email
Johann Steinhoff	[TODO: your @txst.edu email]
[TODO: Member 2 name]	[TODO: @txst.edu email]
[TODO: Member 3 name]	[TODO: @txst.edu email]
[TODO: Member 4 name]	[TODO: @txst.edu email]

---

1. Summary of the Project

1.1 Title

Tender - A swipe-based recipe discovery and meal planning web application.

1.2 Project Description

Tender is a full-stack web application that brings the familiar swipe-based interaction pattern (similar to dating apps like Tinder) to the world of cooking and meal planning. Users are presented with recipe cards one at a time and can swipe right to "like" a recipe or swipe left to "dislike" it. Liked recipes are saved to the user's personal collection and can then be organized into weekly meal plans, with ingredients automatically populated into a smart grocery list.

The application addresses a common everyday problem: deciding what to cook. By turning recipe discovery into a fun, gamified swiping experience and coupling it with practical meal planning and grocery list tools, Tender makes the entire "what's for dinner?" workflow effortless.

GitHub Repository: https://github.com/JohannSteinhoff/3498_SoftwareEngineeringProject2026

1.3 Objectives / Goals

1. **Recipe Discovery via Swiping** - Provide a Tinder-style card swiping interface where users can quickly browse through recipes and like or dislike them. The system remembers preferences and only shows unseen recipes.

2. **Personalized User Profiles** - Allow users to create accounts with detailed food preferences including cooking skill level, dietary restrictions (vegetarian, vegan, gluten-free, keto, halal, kosher, dairy-free), favorite cuisines, household size, weekly food budget, and meals cooked per week.

3. **Meal Planning** - Enable users to organize their liked recipes into a weekly meal plan calendar, assigning recipes to specific days and meal types (breakfast, lunch, dinner).

4. **Smart Grocery List** - Provide an interactive grocery list that lets users add items manually, check off purchased items, and manage quantities. Items are categorized for easy shopping.
5. **Recipe Management** - Allow users to create their own custom recipes (with name, description, cook time, servings, difficulty, cuisine type, emoji icon, ingredients list, and step-by-step instructions) and share them with the community.
6. **User Authentication & Security** - Implement secure user registration and login with password hashing (bcrypt), session-based authentication, and protected API endpoints.
7. **Admin System** - Provide an admin role that can manage recipes and moderate user content.
8. **Responsive Design** - Build a mobile-first responsive interface that works seamlessly on phones, tablets, and desktops, with a dedicated mobile bottom navigation bar.

1.4 Final Outcomes

By the end of the semester, Tender will be a fully functional web application with the following completed features:

- **Landing Page** - An attractive marketing page introducing the app with a call-to-action to sign up.
- **User Registration Flow** - A multi-step signup wizard (3 steps: account info, food preferences, cuisine selection) that personalizes the experience from the start.
- **Login System** - Secure email/password authentication with session management.
- **Dashboard Home** - A personalized home screen showing user stats (liked recipes count, grocery items, meal plan entries, days as member), a quick preview of liked recipes, and a grocery list widget.
- **Swipe Page** - A card-based swiping interface for discovering new recipes, showing recipe details (name, emoji, description, cook time, servings, difficulty, cuisine, ingredients, instructions) with like/dislike buttons.
- **Discover Page** - A searchable, filterable grid/list view of all available recipes for browsing without the swipe mechanic.
- **Meal Plan Page** - A weekly calendar view where users can assign liked recipes to specific days and meal types.
- **Grocery List Page** - A full-featured grocery list with add, check/uncheck, delete, and clear-all functionality.
- **Profile Management** - View and edit user profile, preferences, dietary restrictions, and cuisine preferences.
- **Recipe CRUD** - Create, read, update, and delete user-created recipes.
- **Admin Panel** - Promote/demote admin users, manage all recipes.
- **15+ Seed Recipes** - A starter database of recipes spanning Italian, Mexican, American, Chinese, Japanese, Indian, Greek, Thai, French, Korean, British, and Vietnamese cuisines.

1.5 Benefits and Limitations

Benefits:

- **Solves a Real Problem** - "What should I cook?" is a universal daily decision. Tender makes it fun and fast.
- **Engaging UX** - The swipe mechanic is intuitive and addictive, encouraging users to explore more recipes than they would with a traditional recipe website.
- **All-in-One Workflow** - Unlike most recipe apps that stop at discovery, Tender connects the full pipeline: discover recipes -> plan your week -> generate a grocery list.
- **Personalization** - User preferences (dietary restrictions, cuisine preferences, skill level, budget) allow for tailored recipe recommendations.
- **Accessibility** - Web-based, no installation required. Works on any device with a modern browser.
- **Open Source / Educational** - The codebase serves as a learning platform for full-stack web development concepts.

Limitations:

- **No Real-Time Collaboration** - Currently, users cannot share meal plans or grocery lists with family members in real time (this is a potential future feature).
- **No Image Upload** - Recipes use emoji icons rather than actual food photography. Adding image upload would require file storage infrastructure.
- **No Nutritional Data** - The app does not currently calculate calories, macros, or other nutritional information for recipes.
- **No External API Integration** - Recipes are community-sourced or seeded; there is no integration with external recipe databases (e.g., Spoonacular, Edamam).
- **Session-Based Auth** - Sessions are stored in-memory on the server, meaning they are lost on server restart. A production deployment would need persistent session storage (e.g., Redis).
- **Single Server** - The app runs on a single Node.js process. Scaling would require load balancing and database migration to a production-grade database (e.g., PostgreSQL).

---

2. Implementation Outline

2.1 Software Development Techniques

- **Agile / Iterative Development** - The project is being developed iteratively, with a working prototype already in place that is continuously refined with new features each sprint.
- **Version Control with Git** - All code is managed in a shared GitHub repository with branching and collaborative workflows.
- **Client-Server Architecture** - The application follows a clear separation between the frontend (HTML/CSS/JS) and the backend (Node.js/Express REST API).
- **RESTful API Design** - The backend exposes a well-structured REST API with endpoints organized by resource (auth, users, recipes, grocery, mealplan, admin, stats).
- **MVC-Inspired Pattern** - The codebase separates concerns into routing (server.js), data access (database.js), and presentation (HTML files).
- **Responsive / Mobile-First Design** - CSS media queries and flexible layouts ensure the app works across screen sizes, with a dedicated mobile bottom navigation bar appearing at the 768px breakpoint.
- **Progressive Enhancement** - The app starts with core HTML structure and enhances with JavaScript for interactivity.
- **Database Migrations** - The database schema uses "ALTER TABLE ... ADD COLUMN" migration patterns to evolve the schema without breaking existing data.

2.2 IDE, Framework, and Tools

Category	Tool	Purpose
IDE	[TODO: What IDE does your team use? e.g., VS Code, WebStorm, etc.]	Code editing, debugging, integrated terminal
Runtime	Node.js (v18+)	Server-side JavaScript execution
Web Framework	Express.js (v4.18)	HTTP server, routing, middleware, static file serving
Database	SQLite via sql.js (v1.10)	Lightweight relational database (pure JS, no native dependencies)

Password Hashing	bcryptjs (v2.4)	Secure password hashing and verification
CORS	cors (v2.8)	Cross-origin resource sharing middleware
Version Control	Git + GitHub	Source code management and team collaboration
Package Manager	npm	Dependency management
Browser DevTools	Chrome / Firefox DevTools	Frontend debugging, network inspection, responsive testing
Deployment	[TODO: Where are you planning to host? e.g., GitHub Pages for frontend, Render/Railway/Heroku for backend, or just local demo?]	Application hosting

2.3 Programming Languages

Language	Usage
JavaScript (ES6+)	Primary language for both frontend and backend. Used for all application logic, API client (api.js), server routes (server.js), and database operations (database.js).
HTML5	Page structure and semantic markup for all frontend views (index.html, login.html, signup.html, dashboard.html).
CSS3	Styling, layout (Flexbox, CSS Grid), animations, transitions, CSS custom properties (variables), and responsive design via media queries.
SQL	Database schema definition, queries, and data manipulation within the SQLite database (via sql.js).

---

3. Current Project Status (Prototype / Skeleton)

The following features are already implemented and functional as a working prototype:

Feature	Status	Notes
Landing page (index.html)	Exists (redirect only)	Currently just redirects to tender.html; needs a proper landing page
User registration (3-step wizard)	Complete	Collects name, email, password, cooking skill, household size, dietary restrictions, budget, cuisines, meals/week
User login	Complete	Email/password auth with form validation
Express.js backend server	Complete	Full REST API with 20+ endpoints

SQLite database	Complete	8 tables: users, user_dietary, user_cuisines, recipes, liked_recipes, disliked_recipes, grocery_items, meal_plans
Password hashing (bcrypt)	Complete	Secure password storage
Session-based auth	Complete	Bearer token authentication middleware
Dashboard home page	Complete	Stats cards, liked recipe preview, grocery widget
Swipe page	Complete	Card-based recipe swiping with like/dislike
Discover page	Complete	Browse all recipes in grid view
Meal plan page	Complete	Weekly calendar with meal type slots
Grocery list page	Complete	Add, check, delete, clear items
Recipe CRUD	Complete	Create, view, edit, delete user recipes
Admin system	Complete	Promote/demote admin, admin-only recipe management
15 seed recipes	Complete	Multicultural recipe database
Responsive design	Complete	Mobile bottom nav, responsive grid layouts
Profile view/edit	Complete	View and update all user preferences

---

4. How to Set Up and Run the Project

Prerequisites

- **Node.js** v18 or higher - Download from https://nodejs.org
- **npm** (included with Node.js)
- **Git** - Download from https://git-scm.com

Step-by-Step Setup

```
# 1. Clone the repository
git clone https://github.com/JohannSteinhoff/3498_SoftwareEngineeringProject2026.git

# 2. Navigate into the project directory
cd 3498_SoftwareEngineeringProject2026

# 3. Install dependencies
npm install

# 4. Start the server
npm start
# or: node server.js

# You should see:
```

```
#   Tender Server Running!
#   Local: http://localhost:3000
```

5. Open in your browser

Navigate to **http://localhost:3000** in any modern web browser.

Quick Test

1. Click through to the **Sign Up** page
2. Create an account (fill in all 3 steps)
3. You'll be redirected to the **Dashboard**
4. Click **Start Swiping** to try the recipe discovery feature
5. Like some recipes, then check your **Meal Plan** and **Grocery List** pages

Project File Structure

```
3498_SoftwareEngineeringProject2026/
├── server.js          # Express.js backend server (routes + middleware)
├── database.js        # SQLite database setup, schema, and data access layer
├── api.js             # Frontend API client (makes HTTP requests to backend)
├── auth.js            # Authentication helpers
├── index.html         # Landing/redirect page
├── login.html         # Login page with form validation
├── signup.html        # Multi-step registration wizard
├── dashboard.html     # Main app (SPA with 5 views: Dashboard, Swipe, Discover, Meal Plan,
Grocery)
├── tender.db          # SQLite database file (auto-created on first run)
├── package.json       # Node.js dependencies and scripts
├── package-lock.json  # Dependency lock file
├── start.bat          # Windows quick-start script
└── README.md          # Setup instructions
```

API Endpoints Reference

Method	Endpoint	Auth Required	Description
POST	/api/auth/register	No	Create new user account
POST	/api/auth/login	No	Log in and receive session token
POST	/api/auth/logout	No	End session
GET	/api/auth/me	Yes	Get current user profile
PUT	/api/users/profile	Yes	Update user profile
DELETE	/api/users/account	Yes	Delete user account
GET	/api/recipes	No	Get all recipes
GET	/api/recipes/discover	Yes	Get unswiped recipes for user
GET	/api/recipes/user/created	Yes	Get user's created recipes

GET	/api/recipes/user/liked	Yes	Get user's liked recipes
GET	/api/recipes/:id	No	Get single recipe by ID
POST	/api/recipes	Yes	Create a new recipe
PUT	/api/recipes/:id	Yes	Update a recipe (owner/admin)
DELETE	/api/recipes/:id	Yes	Delete a recipe (owner/admin)
POST	/api/recipes/:id/like	Yes	Like a recipe
POST	/api/recipes/:id/dislike	Yes	Dislike a recipe
DELETE	/api/recipes/:id/like	Yes	Unlike a recipe
GET	/api/grocery	Yes	Get grocery list
POST	/api/grocery	Yes	Add grocery item
PUT	/api/grocery/:id	Yes	Update grocery item
POST	/api/grocery/:id/toggle	Yes	Toggle checked status
DELETE	/api/grocery/:id	Yes	Delete grocery item
DELETE	/api/grocery	Yes	Clear entire grocery list
GET	/api/mealplan	Yes	Get meal plan
POST	/api/mealplan	Yes	Add recipe to meal plan
DELETE	/api/mealplan/:date/:mealType	Yes	Remove from meal plan
POST	/api/admin/promote	Yes	Promote user to admin
POST	/api/admin/demote	Yes	Demote admin user
GET	/api/stats	Yes	Get user statistics

---

5. Semester Roadmap / Future Enhancements

[TODO: Add any planned features or improvements your team wants to tackle this semester. Some ideas based on the current codebase:]

- [ ] Build a proper landing page (index.html) with app screenshots and feature highlights
- [ ] Add recipe image upload support (replace emoji with real photos)
- [ ] Integrate a third-party recipe API for a larger recipe database
- [ ] Add nutritional information (calories, protein, carbs, fat) to recipes
- [ ] Implement social features (share recipes, follow other users)
- [ ] Add recipe search and filtering (by cuisine, difficulty, cook time, dietary tags)
- [ ] Implement "smart" grocery list that auto-generates from meal plan recipes
- [ ] Add push notifications for meal plan reminders
- [ ] Migrate to a production database (PostgreSQL)

- [ ] Deploy to a cloud hosting platform
- [ ] Add unit and integration tests
- [ ] Implement OAuth social login (Google, Apple, Facebook) - currently buttons are placeholders

---

[TODO NOTES FOR JOHANN - Items to fill in before submitting:]

1. **Group Number** - Fill in at the top of the document
2. **All group member names and TXST emails** - Fill in the table on the first page
3. **IDE** - What IDE does your team use? (VS Code, WebStorm, etc.)
4. **Deployment plan** - Where will you host the app? (local demo only, or a cloud service?)
5. **Semester Roadmap** - Review the suggested future enhancements and pick which ones your team actually plans to implement. Remove or add items as needed.
6. **Convert to PDF** - The assignment requires .pdf format for submission via Canvas