IT 110 PROJECT REPORT

Group 5 members:

Jennie Rose E. Auditor - Documentation & Styling Lead & Frontend Assistant

Ronald S. Aguillas - Project Lead/Quality Assurance Lead & Technical Lead (Frontend)

Paul Timothy G. Albaciete - Technical Lead (Backend)

James Roland M. Salar - Technical Lead (Backend)

Github Repository with README file link:

https://github.com/Ron368/IT-110-Final-project.git

I. Technical Implementation

- System Architecture
  - Project Title: **Capy & Co**

    A dynamic Single Page Application (SPA) that transforms raw public API data into a curated, visual narrative.

  - Project Narrative:

    In this project you will meet Chef Capy, a world-renowned capybara gourmand with a passion for travel and taste. Tired of staying in one swamp, he founded Capy & Co., a traveling kitchen that never stays in one place.

    Chef Capy runs across the globe, from the busy streets of Tokyo to the romantic avenues of Paris, collecting recipes and delivering the world's most famous cuisines directly to your screen. He doesn't just cook; he brings the culture of every nation he visits along with him.

    Built on RILT stack (React, Inertia, Laravel, Tailwind), the platform offers a single-page application experience that feels less like browsing a database and more like exploring a world. From the moment Chef Capy runs and passing through country flags in the loading screen, users are invited to discover curated recipes, artisanal goods, and cozy finds, all "foraged" from around the globe by the world’s most relaxed connoisseur.

  - Project Objectives:

    To cure "doom-scrolling" by creating a "bloom-scrolling" culinary experience. We aim to provide a digital space where utility meets tranquility. Whether a user is searching for a rare authentic dish from across the ocean or just a moment of digital peace, Capy&Co delivers it with efficiency and charm. It is a global kitchen where users can discover flavors from every continent, and curate their personal collection of favorites.

    To demonstrate that modern web performance (Laravel/Inertia) and playful, high-fidelity design (Three.js/React) can coexist. Capy&Co proves that a utility app doesn't have to be boring, and a fun app doesn't have to be useless.

  - Architecture Pattern:

    We utilized Inertia.js as the glue between our server-side framework (Laravel) and client-side library (React). This allowed us to build a modern, reactive frontend while keeping the routing and controller logic within the classic Laravel structure.

- Database Design:
  A relational MySQL database was designed to store user-generated content. Key schemas include the *users* table for authentication, *recipes* for storing imported meal data, and auxiliary tables like *reviews* and f*avorites* to handle many-to-one and many-to-many relationships.

- Core Modules
  - Authentication & Security:
    We implemented Laravel Breeze to handle secure user registration, login, and session management. This ensures that features like "Add to Favorites" and "Write a Review" are protected and accessible only to verified users.
  - External API Integration:
    The application consumes data from the MealDB API. We implemented a service layer (seen in *RecipeController.php* via *importFromMealDB*) that fetches raw JSON data (ingredients, instructions, thumbnails) and transforms it into a standardized format compatible with our local database.
  - Frontend Interactivity:
    The UI is built with React and styled using Tailwind CSS. We utilized Framer Motion to add fluid animations, enhancing the visual storytelling objective of the project.

- Project's Core Features:
  - ✓ Visual Storytelling
  - ✓ External API Integration
  - ✓ User Personalization (CRUD)
  - ✓ Dynamic SPA Experience
  - ✓ Secure Authentication

II. Challenges Faced and Solutions

During the development lifecycle, the team encountered significant hurdles regarding environment configuration and data persistence. Below are the primary challenges and the solutions we did to overcome them.

- Development Environment & Stack Configuration (Set up phase)

  The initial phase of the project was stalled by difficulties in manually configuring the development environment. Specifically, integrating React into a standard Laravel application proved complex; we faced issues with bundling assets, configuring Webpack/Vite, and establishing the communication channel between the PHP backend and the JavaScript frontend.

  This effectively halted our progress for one and a half days. The initial setup became so corrupted/damaged that we were forced to abandon the original repository and restart the setup process from scratch.

  Solution:

  - ➢ In the new repository, we pivoted from a manual setup to using Laravel Breeze with the React/Inertia stack. Breeze provided a pre-scaffolded authentication system and automatically configured the build tools (Vite) and frontend routing. This standardized our development environment, allowing every team member to run *npm run dev* and *php artisan serve* (as noted in our README) without dependency conflicts.

- Connecting Backend Logic to Frontend Interfaces (CRUD)
  A major technical blocker occurred when attempting to implement CRUD (Create, Read, Update, Delete) features. While our Frontend forms (React components) were sending data, and our Controllers (*RecipeController*) were receiving requests, the data was not persisting to the database. We faced "Table Not Found" errors and data mismatches, breaking the flow of the application.

  Solution:
  - ➢ We identified that our local database schemas were out of sync with our application logic. The Model definitions existed in code, but the actual tables were missing in MySQL. We utilized Laravel's migration system, specifically running *php artisan migrate*. This command executed our schema files (such as *2025_12_03_..._create_recipes_table.php*), creating the necessary tables with the correct columns and data types. This ensured that when a user submitted a review or saved a recipe, the backend had a valid destination to store that record.