Payment Management System - Project Documentation

GitHub Repository URL:
https://github.com/hussain-2004/Payment-Management-System-Frontend-UI

Project Overview

The Payment Management System is a web application built using Vue, designed for tracking payments and managing users. This document outlines its core logic, architecture, and implementation approach.

Technical Architecture & Solution Approach

Framework Selection

Vue: Chosen for its reactive data handling and component-based structure.

Vite: Used for a fast development server and efficient building process.

Vue Router: Implements client-side navigation for a smooth user experience.

Vuex: Manages application data centrally, ensuring consistency.

Tailwind CSS: Provides a utility-first approach to styling for a consistent look.

Project Structure Logic

The project is organized logically to ensure maintainability:

src/views/: Contains page-level components that users navigate to.

src/components/: Houses reusable UI elements.

src/store/: Holds the central state management logic.

src/router/: Configures the application's navigation paths.

src/data/: Includes sample data and mock API responses.

src/test/: Contains the suite for unit testing.

Core Logic Implementation

1. Authentication System

This system features simple sign-in and sign-up pages designed to mimic real-world applications, including email validation. It provides a basic user experience without requiring a backend. User data is managed via local storage.

2. State Management Strategy

An approach using Vuex for centralized state management, combined with Local Storage to save data across browser sessions. This ensures data persists even after closing and reopening the browser, enabling fast client-side operations and automatic ID/timestamp generation when new data is added.

3. Component Architecture

UI components are designed using principles of atomic design, communicating through props and state. This includes reusable components like statistic cards, sortable tables, and a navigation bar, alongside page-specific components for distinct sections.

4. Data Flow Logic

A centralized Vuex store is utilized, with components accessing and updating data through it. Computed properties are used to derive data for display, such as active user counts or total completed payment amounts.

5. Routing & Navigation Logic

Vue Router manages different pages, and the navigation bar's visibility is controlled dynamically based on the current route, such as being hidden on login or registration pages.

Technical Implementation Details

Data Persistence Strategy

Local Storage is used to save data. Helper functions ensure data is parsed correctly and provides default values if none exist or if an error occurs.

Form Handling Logic

Create Operations: A unique ID is generated, often using the current timestamp.

Edit Operations: Existing data is loaded into the form for modification and saved upon confirmation.

Validation: Standard form validation is used, highlighting required fields.

Navigation: Users are automatically redirected after successful form submissions.

Search & Filtering

The application includes functionality to search and filter data, such as payments, based on multiple criteria like keywords in descriptions and payment status.

Testing Strategy

Framework: Jest is used with Vue Test Utils for writing unit tests. The focus is on testing core components and logic to ensure reliability.

Setup Process

Development Workflow

Common commands are provided for starting the development server, running tests, and checking code quality.

GitHub Repository

Repository URL:
https://github.com/hussain-2004/Payment-Management-System-Frontend-UI

Repository Structure: The main branch contains the stable code, with a README providing setup instructions and project details.

Instructions for Evaluators

System Requirements

Node.js (version 16 or higher)

A package manager (npm or yarn)

A modern web browser

Setup Instructions

Clone Repository: First, use the command

git clone
https://github.com/hussain-2004/Payment-Management-System-Frontend-UI.git

to download the project files onto your local machine.

Then, navigate into the newly created project directory using

cd Payment-Management-System-Frontend-UI.

Install Dependencies: Run the command

npm install to download and install all the necessary libraries and packages that the project relies on.

Start Development Server: Execute npm run dev to launch a local development server. This command compiles the project and makes it accessible via your web browser.

Run Tests: To verify the application's components and logic, run npm test. This executes the suite of unit tests.