HackStack – Project Report

Submitted By:
Name: Akash Wadode
Email: ajwadode25@gmail.com
Program: MCA AI/ML
Institute: Lovely Professional University, Punjab
GitHub Repository: https://github.com/akashwadode/HackStack

Project Overview

HackStack is a web platform that enables users—especially hackathon participants and developers—to create, manage, and showcase their personal project portfolios. Users can authenticate, add new projects, and share a public portfolio page, making it easy to present their work to others.

Tech Stack

- **React:** Main frontend framework for building user interfaces.
- **Vite:** Fast development server and build tool for React.
- **React Router:** Handles client-side routing and navigation.
- **Supabase:** Provides authentication, database, and file storage services.
- **CSS Modules:** Modular and component-scoped styling for maintainable UI.

Key Features

- **User Authentication:** Secure sign-up, login, and session management.
- **Dashboard:** Personalized dashboard for managing projects and profile.
- **Project Management:** Add, edit, and display hackathon or personal projects.
- **Public Portfolio:** Each user has a shareable public portfolio page.
- **Responsive UI:** Modern, mobile-friendly layouts and reusable components.

Supabase Integration

Supabase is used as the backend for:

- **Authentication:** Handles user sign-up, login, and session management.
- **Database:** Stores user profiles, project data, and other app content.
- **File Storage:** (If implemented) Allows users to upload and manage project images or files.

Integration is managed via a dedicated `supabaseClient.js` file, which initializes and exports the Supabase client for use throughout the app.

Public Portfolio

Each user has a unique public portfolio page accessible via a URL (e.g., /u/id/:userId). This page displays the user's profile information, tech stack, and a list of their projects, making it easy to share their work with others.

Challenges & Learnings

- **Routing Logic:** Managing conditional UI (like hiding the navbar on auth pages) required careful use of React Router and layout components.
- **Supabase Integration:** Learning Supabase's API for authentication and data management was essential for a smooth user experience.
- **Componentization:** Breaking the UI into reusable, maintainable components improved scalability and code organization.
- **Responsive Design:** Ensuring the platform looks good on all devices involved mastering CSS Flexbox and modular styles.

These challenges led to a deeper understanding of modern React patterns, third-party integrations, and best practices for scalable web app development.