MERN Stack Cross Platform Application Launcher

Project Overview

This project is a Cross Platform Application Launcher built using the MERN (MongoDB, Express.js, React, Node.js) stack. The application allows users to manage and launch various applications through a user-friendly interface.

Backend Implementation

File Structure

- index.js : The entry point for the Node.js server, where Express is configured and routes are set up.
- connectDB.js : Establishes a connection to the MongoDB database using Mongoose.
- routes/appRoutes.js : Contains the API routes for creating, reading, updating, and deleting applications.
- models/appsModels.js : Defines the Mongoose schema for the application data.
- controllers/appsControllers.js : Implements the CRUD operations for managing applications.

Dependencies

- Express : Handles the web server and routing.
- Cors : Enables Cross-Origin Resource Sharing.
- Dotenv : Loads environment variables from a .env file.
- Mongoose : An ODM (Object Data Modeling) library for MongoDB and Node.js.
- Multer : A middleware for handling file uploads.
- Nodemon : Nodemon will restarts the server when changes are detected.

API Endpoints

- POST /apps/create : Creates a new application entry with an icon, name, and configuration.
- GET /apps/get : Retrieves a list of all applications with their icons.
- PUT /apps/update/:id : Updates an existing application by ID with new information and/or icon.
- DELETE /apps/delete/:id : Deletes an application by ID.
- GET /apps/get/:id : Retrieves details of a specific application by ID.

Uploading Images

- Icons for applications are uploaded using Multer middleware and stored in the 'uploads/' folder.
- The file path is saved in the database.

Documentation

Setting Up the Backend

- npm init -y
- Run npm install to install dependencies.
- Create a .env file with the MongoDB connection URL.

Running the Backend

- Execute npm start in the terminal.
- The server will run on the specified port (default is 4000).

API Usage

- Use API endpoints mentioned above for CRUD operations.
- When creating or updating an application, provide icon, name, and configuration data.

Time Spent

- Setting up backend structure: 30 min
- Implementing CRUD operations: 4 hours
- Testing and debugging: 1 hour

Frontend Implementation

File Structure

- App.js : The entry point for the React application, where routes are defined using react-router-dom.
- HomePage.jsx : The component for the home screen, displaying a list of applications.
- Settings.jsx : The component for the settings page, allowing users to add new applications.
- Configuration.jsx : The component for the application configuration page, providing details and options for updating and deleting applications.

Dependencies

- React : A JavaScript library for building user interfaces.
- Axios : A promise-based HTTP client for making requests to the backend.
- react-router-dom : A library for declarative routing in React.
- @fortawesome/react-fontawesome: Provides React components for Font Awesome icons.
- Tailwindcss : A utility-first CSS framework used for styling.

Routing

- The application uses the Routes and Route components from react-router-dom to handle navigation between different pages.

Styling

- Tailwind CSS: A utility-first CSS framework used for styling, providing a flexible and responsive design.

Components

HomePage.jsx:

- Displays a list of applications with their icons.
- Allows users to click on an application to navigate to its configuration page.

Settings.jsx:

- Provides a form for users to add new applications.
- Supports input fields for app name, configuration, and image upload.
- Handles form submission and sends data to the backend for storage.

Configuration.jsx:

- Retrieves and displays details of a specific application based on the ID provided in the URL parameters.
- Allows users to update the app name, configuration, and icon.
- Provides options to update and delete the application.

API Interaction

- Uses the axios library to make HTTP requests to the backend API for retrieving, adding, updating, and deleting applications.

Documentation

Setting Up the Frontend

- Npm init vite
- Run npm install to install dependencies.

Running the Frontend

- Execute npm start in the terminal.
- The React application will run on the specified port (default is 4000).

Pages

Home Page (/):

- Displays a list of applications.
- Clicking on an application navigates to its configuration page.

Settings (/settings):

- Allows users to add new applications.
- Includes input fields for app name, configuration, and image upload.

Configuration (/configuration/:id):

- Displays details of a specific application based on the provided ID.
- Allows users to update app name, configuration, and icon.
- Provides options to update and delete the application.

Styling

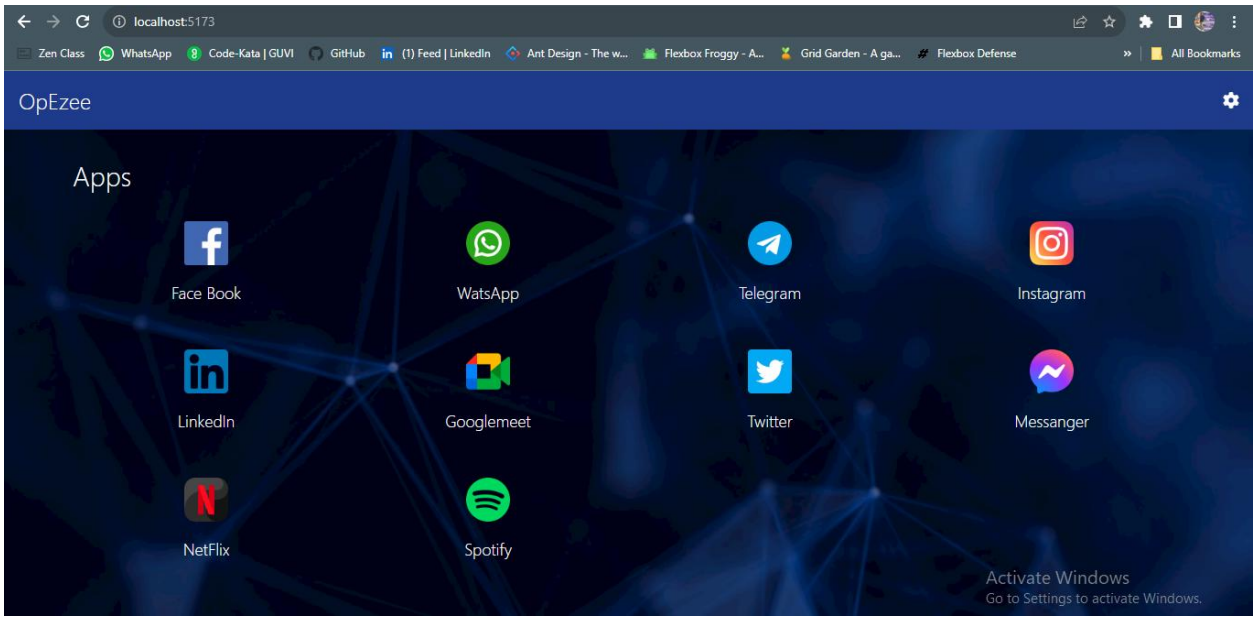
- The project uses the Tailwind CSS framework for styling. Refer to the official Tailwind CSS documentation for customization and additional styling options.

Time Spent

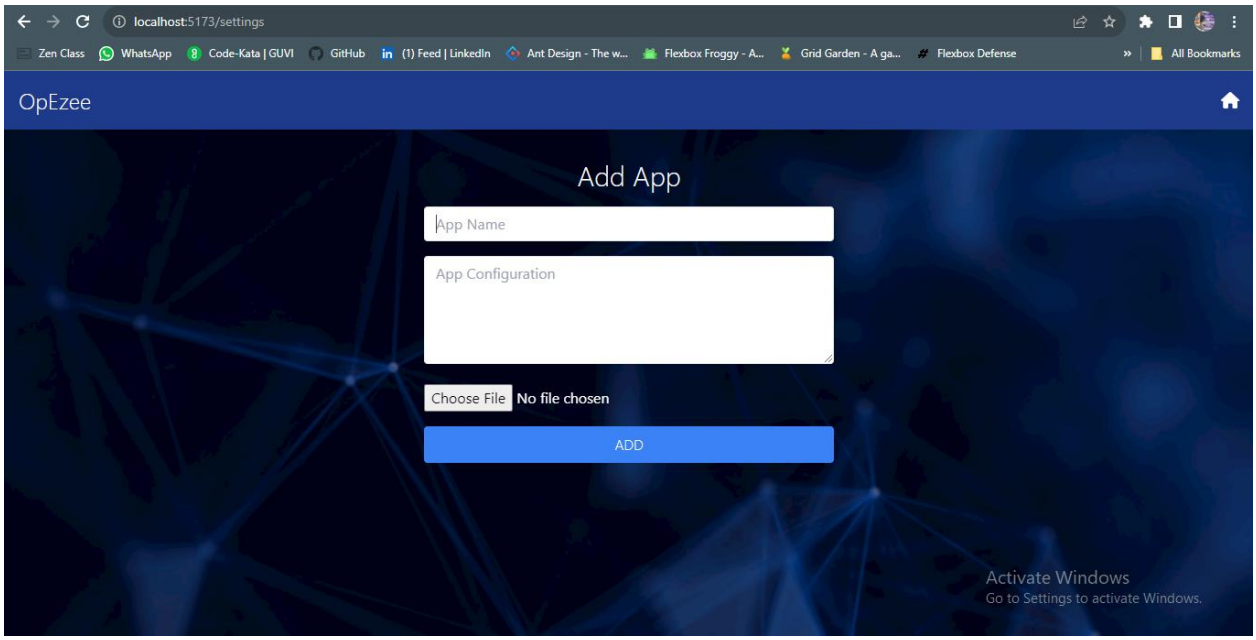
- Setting up the frontend structure: 30 min
- Implementing home page functionality: 2 hours
- Creating settings and configuration pages: 3 hours
- Testing and debugging: 1 hours

Screenshots

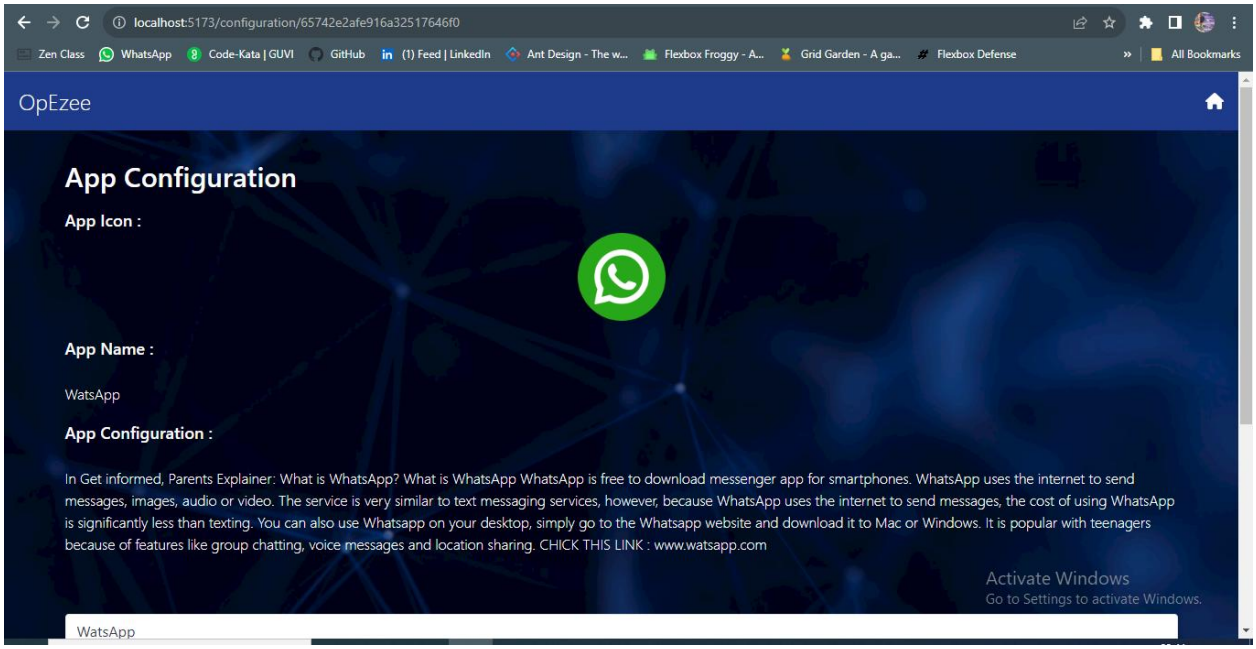
HomePage

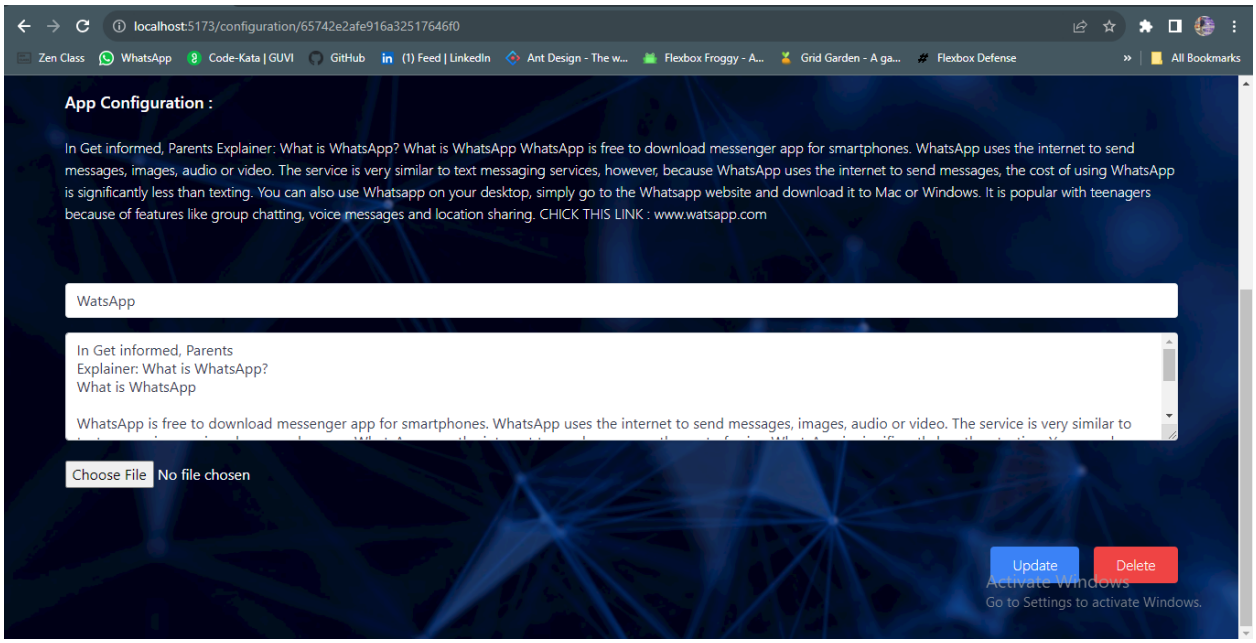


SettingsPage



ConfigurationPage





Check This Link For Project Video : https://drive.google.com/file/d/1Fxsk-WcOLVDVXhnIAEA8ZL3nEHj4zpSP/view?usp=sharing

Back-End GitHub URL : https://github.com/sanjaikannang/OpEzee-Backend.git

Front-End GitHub URL : https://github.com/sanjaikannang/OpEzee-Frontend.git