APK MANAGEMENT SYSTEM DOCUMENTATION

Table of Contents

1. OVERVIEW:

This documentation details the APK Management System, including both the backend and frontend components. The system is designed for managing APK data, providing functionalities such as data scraping, editing, deletion, and validation of APK versions.

2. BACKEND:

2.1 TECHNOLOGIES:

1. **Node.js**: The runtime environment for executing JavaScript on the server side.
2. **Express**: A web application framework for Node.js.
3. **Mongoose**: An ODM (Object Data Modeling) library for MongoDB and Node.js.
4. **MongoDB**: The database used for storing APK data.

2.2 SETUP AND INSTALLATION:

1. **Prerequisites**: Node.js, MongoDB, and npm.
2. **Installation**: Clone the repository and run npm install to install dependencies.
3. **Running the Server**: Execute npm start to launch the server on http://localhost:3000.

2.3 KEY COMPONENTS:

1. App.js:

Description: Entry point of the application. It sets up the Express server, MongoDB connection, middleware, and routes.

Key Libraries:

1. Express: Framework for handling HTTP requests and routing.
2. Mongoose: ODM for mongodb interaction.
3. Helmet: Security middleware.
4. Morgan: Logging middleware.
5. Cors: Middleware to enable CORS (Cross-Origin Resource Sharing).

2. **Scraper.js:**

Purpose: Handles the scraping of APK data from APK Mirror.

Functions:

1. Loadwebpage(url): Fetches HTML content from a given URL.
2. Sleep(ms): Utility to delay execution.
3. Parsetitle(fulltitle): Extracts version and type from the title.
4. Extractinstagramapkdata(html): Processes HTML to extract APK data.
5. Fetchvariantdetails(apkentry): Fetches additional details for each APK variant.
6. Fetchapkdata(): Orchestrates the scraping process and returns APK entries.

3. **Controllers/ApkController.js:**

Functions:

1. Scrapeandsavedata(): Triggers scraping and saves data to mongodb.
2. Getversions(): Retrieves a list of versions from the database.
3. Getversiondetail(): Fetches details of a specific version.
4. Deleteversion(): Deletes a specified version.
5. Updateversion(): Updates details of a specific version.
6. Validaterequest(): Validates a request against stored data.
7. Getversioninfo(): Retrieves detailed information for a specified version.

4. **Models/ApkModel.js:**

Schema: Defines the structure for storing APK data in MongoDB.

5. **Routes/Router.js:**

Routes: Defines endpoints for the API.

1. /api/scrape: Triggers the scraping process.
2. /api/versions: Retrieves version list.
3. /api/versions/:versionId: Fetches/deletes/updates details of a specific version.
4. /api/validate: Validates incoming requests.
5. /api/versions/info/:versionId: Retrieves detailed info of a specific version.

6. **Error Handling:**
   1. Error handling is implemented in each controller function.
   2. Provides meaningful error messages and appropriate HTTP status codes.

3. FRONTEND:

3.1 TECHNOLOGIES:

1. **React**: A javascript library for building user interfaces.
2. **Axios**: Used for making HTTP requests to the backend.
3. **React Router**: For handling routing in the application.

3.2 SETUP AND INSTALLATION:

1. **Prerequisites**: Node.js and npm.
2. **Installation**: After cloning the repository, run npm install in the project directory to install dependencies.
3. **Running the Application**: Use npm start to run the app in development mode. Open http://localhost:3000 to view it in the browser.

3.3 KEY COMPONENTS:

1. **App.js:**
    a. **Description**: The main component that sets up routing for the application using React Router.
    b. **Features**: Includes navigation links to various functionalities like scraping, editing, deleting, and validating APK data.

2. **VersionList.js:**
    a. **Description:** Displays a list of all APK versions with details fetched from the backend.
    b. **Features**: Automatically fetches and displays a list of APK versions, with each version showing its ID, release date, and total number of variants.
3. **VersionDetails.js:**
    a. **Description:** Allows users to retrieve and view detailed information about a specific APK version.
    b. **Features**: Provides a form for users to enter a version ID and retrieve its detailed information, including variant-specific data like architecture and Android version compatibility.
4. **ValidateRequest.js:**
    a. **Description:** Provides an interface for validating APK version compatibility based on user input.
    b. **Features**: Offers a form for users to validate compatibility of APK versions based on inputs like version ID, variant ID, Android version, and DPI, along with displaying the validation results.
5. **StartScraping.js:**
    a. **Description:** Contains a button to initiate the scraping process of APK data from the backend.
    b. **Features**: Features a button to initiate the scraping process with real-time updates on the status of scraping, including success or error messages.
6. **EditVersion.js:**
    a. **Description:** Offers functionality to edit and update details of a specific APK version.

b. **Features**: Facilitates editing and updating APK version details through a form, including functionalities to load existing data for a specific version and submit updated information.

7. **DeleteVersion.js:**
    a. **Description:** Allows users to delete a specific version of an APK from the database.
    b. **Features**: Allows deletion of a specific APK version from the database through an input field for version ID, including feedback messages post-deletion attempts.