**Development Plan**

**1. Setting Up the Environment**

* **Tools**: React, TypeScript, Tailwind CSS, Vite (for faster development).
* **Packages Installed**:
  + axios for API calls.
  + react-router-dom for navigation.
  + lucide-react for icons.

**2. Architectural Decisions**

**Front – End:**

* **Component-Based Design**:
  + All UI components are reusable and modular.
  + Components such as Header, ShowGrid, ShowCard, and Filter are created for scalability.
* **State Management**:
  + **Local State**: Handled with useState and useEffect.
  + **Local Storage**: Used for persisting favorites and search history.
* **Routing**:
  + **Pages**: HomePage, ShowPage, FavoritesPage, SearchHistoryPage.
  + Used react-router-dom to define routes.

**Back-End:**

**Server File**:

* Entry point of the application.
* Configures middleware (e.g., body parsers, CORS) and starts the server.

**Routes**:

* Contains the routing logic to handle API endpoints.
* Maps client requests to appropriate controllers.

**Controllers**:

* Handles request processing and calls services for business logic.
* Returns the processed response to the client.

**Services**:

* Contains the core business logic.
* Interacts with external APIs (e.g., TVMaze) or any future database layer.
* Ensures separation of concerns by abstracting data processing from the controllers.

**Logger Utility**:

* Provides consistent logging across the application.
* Logs important events such as API calls, errors, and system-level warnings.
* Improves supportability and troubleshooting.

**Key Features**

**1. Home Page**

* **Search Functionality**:
  + Users can search for shows by entering a query in the search bar.
  + Filters (rating, genres, language) are applied dynamically without additional API calls if shows are already loaded.
* **API Interaction**:
  + Endpoint: GET /search/shows?q={query}.
  + Filtering logic applied client-side after fetching shows.
* **UI Components**:
  + Header: Includes the search bar.
  + Filter: Dropdowns for genres, rating, and languages.
  + ShowGrid: Displays a grid of ShowCard components.

**2. Show Details Page**

* Displays detailed information about a selected show.
* Lists episodes, grouped by season.
* **API Interaction**:
  + GET /shows/{id} for show details.
  + GET /shows/{id}/seasons for season details.
  + GET /shows/{id}/episodes for all episodes.
* **Architectural Decisions**:
  + Episodes are filtered client-side to minimize API calls.
  + Default images provided for episodes without thumbnails.

**3. Favorites page**

* Allows users to mark shows as favorites using a star button.
* Favorites persist via localStorage.
* **State Handling**:
  + addFavoriteShow and removeFavoriteShow manage favorite shows.
  + isFavoriteShow checks if a show is already marked as a favorite.
* **UI Updates**:
  + ShowCard reflects favorite state dynamically.
  + Favorites page lists all favorited shows.

**4. Search History page**

* Logs user searches (query and filters) in localStorage.
* **State Handling**:
  + Search history is grouped by date for better organization.
  + Includes functionality to clear individual searches or all history.
* **UI Components**:
  + Search history items displayed with a timestamp and result count.

**Styling and UX Decisions**

* **Tailwind CSS**:
  + Used for rapid UI design.
  + Dark mode support implemented via dark classes.
  + Pagination
* **Hover Effects**:
  + Interactive hover effects for buttons and cards.
  + Genres are displayed as styled chips on hover in ShowCard.
* **Responsiveness**:
  + Grid layout adjusts for different screen sizes.