Epic: Poke-Store Mobile Application Development

As a Pokémon enthusiast, I want to use a mobile application where I can explore, select, and purchase Pokémon in an engaging and educational way, so that I can enjoy my passion for Pokémon with the convenience of technology.

Use these Frameworks and Libraries:

- **Typescript and Android App:** write in typescript and must run in Android
- **React Native**: Used as the core framework to develop the cross-platform mobile application, ensuring a native-like user experience.
- **Redux**: Redux for state management
- Optional:
    - **Redux Toolkit**: Redux Toolkit to simplify the configuration and enabling efficient management of the app's state, actions, and reducers.
    - **Redux Thunk**: Utilized for handling asynchronous logic to interact with the PokeAPI, fetching data, and performing side effects.

Story 1: Setup and Configuration

As a developer, I need to set up the React Native project and install necessary dependencies like Redux and Redux Toolkit, so that I can start developing the application with the required frameworks and libraries.

Acceptance Criteria:

- Initialize the React Native project.
- Install Redux, (Redux Toolkit), and any other necessary dependencies.
- Create a READ.me file and write initial instructions.

Story 2: Explore Pokémon

As a user, I want to view a list of Pokémon with their names and icons displayed in a grid layout, so that I can easily explore and choose my favorite Pokémon to learn more about or add to my cart.

Acceptance Criteria:

- Fetch Pokémon data from the PokeAPI. Here's the link PokéAPI
- Display Pokémon in a grid layout with names and icons.
- Ensure the menu is scrollable to handle a large number of Pokémon.

Story 3: Manage Cart

As a user, I want to add Pokémon to my cart, adjust quantities, and remove items, so that I can manage my selections before making a purchase.

Acceptance Criteria:

- Implement add-to-cart functionality for selected Pokémon.
- Allow quantity adjustments and item removal within the cart.
- Provide a summary view of the cart and selected Pokémon details.

Bonus Features: Showcase Your Talent (Optional)

Complete as much bonus stories as you want.

Bonus Story: Dynamic Pricing Based on Weight

As a user, I want the cost of Pokémon in my cart to be calculated based on their weight, so that I can see how the unique attributes of each Pokémon affect their purchase price.

Acceptance Criteria:

- Fetch and utilize the "weight" attribute of each Pokémon from the PokeAPI.
- Calculate the cost of each Pokémon in the cart based on weight.
- Update the cart's total cost dynamically as items are added, removed, or quantities are changed.

Bonus Story: Optimize Performance and User Experience

As a developer, I want to ensure that the application runs smoothly and provides an excellent user experience, especially considering potential challenges like API rate limiting and the dynamic content nature of the app.

Acceptance Criteria:

- Implement efficient data fetching and caching strategies to mitigate excessive API calls.
- Utilize memoization and optimize re-renders for better performance.
- Design a user-friendly and responsive interface, considering mobile usability best practices.

Bonus Story: Call Native Features in Kotlin/Java

As a developer, I want to integrate native Android features using Kotlin or Java, so that I can enhance the application with capabilities not directly available through React Native.

Acceptance Criteria:

- Implement a native module in Kotlin or Java that interacts with Android SDK features.
- Integrate the native module with the React Native codebase using the Native Modules interface.
- Demonstrate the use of at least one native feature (e.g., accessing the device's camera, sending SMS, Map services or using location services) that enhances the app's functionality.

Bonus Story: Write Unit Tests

As a developer, I want to write unit tests for the application's components and logic, so that I can ensure the reliability and quality of the app through automated testing.

Acceptance Criteria:

- Set up a testing framework (e.g., Jest) for the React Native application.
- Write unit tests for Redux actions, reducers, and React components.
- Achieve a significant percentage of code coverage to ensure most functionalities are tested.

Bonus Story: Implement Turbo Native Module

As a developer, I want to implement a Turbo Native Module to improve the performance of the application, especially in areas requiring high computational power or smoother animations.

Acceptance Criteria:

- Implement a Turbo Native Module for a specific functionality that benefits from native performance optimization.
- Integrate the Turbo Native Module with the React Native application, ensuring seamless operation.

Bonus Story: Implement Lazy Loaded Redux Slice

As a developer, I want to implement lazy-loaded Redux slices, so that I can optimize the application's startup time and overall performance by loading state management logic on demand.

Acceptance Criteria:

- Configure Redux Toolkit to allow for lazy loading of slices.
- Implement at least one feature of the application using a lazy-loaded Redux slice.
- Verify that the Redux slice is only loaded and initialized when it is required by the application, improving initial load times.

Bonus Story: Show the Ability to Log

As a developer, I want to implement comprehensive logging throughout the application, so that I can monitor, debug, and improve the application based on real usage patterns and errors.

Acceptance Criteria:

- Integrate a logging library that supports both development and production environments.
- Implement logging across critical application flows, including API calls, user actions, and error handling.
- Ensure logs are structured in a way that supports easy analysis and monitoring.

Bonus Story: Write E2E Tests

As a developer, I want to write end-to-end (E2E) tests for the application, so that I can ensure the app works as expected from a user's perspective, covering critical user journeys.

Acceptance Criteria:

- Set up an E2E testing framework (e.g., Appium or Maestro) for the React Native application.
- Write E2E tests covering key user journeys, such as exploring Pokémon, adding items to the cart, and making a purchase.
- Ensure E2E tests can be run automatically and are integrated into the CI/CD pipeline for regular validation.

Bonus Story: Create an API using RTK Query and Entity Adapter

As a developer, I want to create a scalable API layer using RTK Query and the Entity Adapter, so that I can manage and optimize API calls and data normalization efficiently within the application.

Acceptance Criteria:

- Utilize RTK Query to set up an API service layer for the application, handling all interactions with the PokeAPI.
- Implement the Entity Adapter to manage the normalization of fetched data, enabling efficient storage and retrieval from the Redux store.
- Ensure the application's data layer automatically updates based on CRUD operations performed through RTK Query endpoints.

Bonus Story: Create a Cache System Using Android's Internal Room Database

As a developer, I want to create a cache system that utilizes Android's internal Room database, so that I can provide a seamless and efficient user experience by minimizing network requests and loading times.

Acceptance Criteria:

- Integrate Android's Room database with the React Native application to store and retrieve cached data.
- Implement caching logic to store fetched Pokémon data from the PokeAPI into the Room database after an initial load.
- Ensure that the application checks the Room database for existing data before making network requests, falling back to the network if the data is not present or is outdated.