Специфікація Вимог до Програмного Забезпечення (SRS)

1. Вступ

1.1 Призначення

Legend-FSM — це легка, декларативна бібліотека кінцевих автоматів, створена на основі Legend-State для управління переходами станів і логікою валідації у застосунках React Native. Її мета — покращити логіку форм, спростити обробку подій та підвищити продуктивність за рахунок реактивного управління станом.

1.2 Область застосування

Legend-FSM використовується для управління логікою інтерфейсів користувача, таких як форма створення облікового запису, де є багато полів, переходів і асинхронних процесів (валідація, вхід через соцмережі). Вона замінює імперативну логіку useState / useReducer на прозорий декларативний підхід.

1.3 Визначення та скорочення

• FSM: Кінцевий автомат (Finite State Machine)

• Legend-State: Бібліотека реактивного управління станом

• RN: React Native

1.4 Цільова аудиторія

• Розробники React Native

• Мобільні інженерні команди

• Учасники open-source

• Академічні рецензенти

2. Загальний опис

2.1 Контекст продукту

Legend-FSM розширює можливості Legend-State, додаючи декларативну модель кінцевих автоматів. Бібліотека напряму інтегрується з інтерфейсом через реактивні зв’язки і вже використовується у реальних сценаріях, таких як «Create Account».

2.2 Основні функції

• Декларативна конфігурація FSM

• Побудова на observable з Legend-State

• Переходи з валідацією та побічними ефектами

• Обробники входу/виходу зі станів

• Інтеграція з UI React Native

• Авторизація через Google, LinkedIn, Apple

2.3 Характеристики користувача

• Досвід роботи з React Native

• Розуміння state management і кінцевих автоматів

• Знання TypeScript і асинхронних процесів

2.4 Обмеження та залежності

• React Native >= 0.76

• Legend-State >= 3.0.0-beta

• Платформи: iOS 13+, Android 8+

• Node >= 18.x

3. Функціональні вимоги

FR-1: Ініціалізація автомата

Система повинна дозволяти створення автомата станів через декларативну конфігурацію.

FR-2: Обробка введення користувача

Поля форми (firstName, email, password тощо) мають бути реактивними і зв’язаними з UI.

FR-3: Переходи між станами

Перехід повинен здійснюватися за подіями (SUBMIT, VALIDATE) з оновленням активного стану.

FR-4: Валідація

При події VALIDATE система має заповнювати поле errors згідно з виявленими помилками.

FR-5: Асинхронні побічні ефекти

Подія SUBMIT має запускати імітацію запиту до сервера з переходом до стану SUCCESS або ERROR.

FR-6: Інтеграція з React Native

Бібліотека має працювати з елементами типу <TextInput />, <Switch />, <Button /> через @legendapp/state/react-native.

4. Нефункціональні вимоги

Продуктивність

• Переходи мають виконуватись менш ніж за 1 мс

• Споживання памʼяті має бути мінімальним (до 2x у порівнянні з useState)

Сумісність

• Повна підтримка iOS та Android

• Сумісність із сучасними бібліотеками навігації та стилізації

Супроводжуваність

• Код повинен бути повністю типізованим (TypeScript)

• Логіка FSM має бути покрита юніт-тестами

Розширюваність

• У майбутньому — підтримка вкладених машин та інвокацій

5. Інтерфейси зовнішніх компонентів

UI Інтерфейс

• Компоненти React Native повинні відображати значення та помилки в реальному часі

Репозиторій

• Legend-State-Machine на GitHub: <https://github.com/VladyslavMartynov10/Legend-State-Machine>

Інтеграції зі сторонніми сервісами

• Авторизація через Google, Apple, LinkedIn (імітація логіки)

6. Майбутні покращення

• Панель налагодження FSM у UI

• Плагін для DevTools для перегляду стану

• Схема конфігурацій через Zod або JSON Schema

• Автогенерація графу переходів зі структури автомата

7. Додатки

• Автор: Влад Мартинов

• Ліцензія: MIT

• Репозиторій: Legend-FSM на GitHub: <https://github.com/VladyslavMartynov10/Legend-State-Machine>