**Proof of Concept (POC) of Sprint-1:**

## **Project: AI-Powered Language Learning Platform (Fluentify)**

## **Sprint Goal**: Design and implement the initial framework by developing the user interface, database schema, and authentication workflow (with validation and role-based access), along with a prototype dashboard to support learner onboarding and preferences.

## **Sprint Objective:**

## To implement the core authentication and onboarding workflow for the platform covering both frontend and backend components:

## Responsive Sign-Up and Login UI for learners and admins.

## Secure backend APIs for authentication and preferences.

## Password encryption with bcrypt and JWT-based session management.

## Database schema setup with learners, admins, and preferences tables.

## Prototype dashboard accessible upon successful login.

## **Features Completed:**

## **1.Frontend Development**

## **Sign-Up Page:**

## Email validation to block fake entries.

## Strong password policy (minimum length of 8).

## **Login Page:**

## Credential validation.

## Redirection to dashboard upon successful authentication.

## **Dashboard (Prototype):**

## Provides entry point for learners/admins post-login and includes placeholders for various modules.

## **Routing & State Handling:**

## React Router DOM configured for protected routes.

## Session management with JWT decode.

## **Styling & Standards:**

## TailwindCSS for responsive UI

## ESLint for code quality

## **2.Backend Development**

## **Database Setup**

## **Tables Created:**

## learners (unique email constraint, timestamps).

## admins (secure storage with hashed passwords).

## learner\_preferences (language + expected duration).

## **Migration Scripts:**

## Create\_tables.sql

## Create\_learn\_preferences.sql

## **Authentication APIs (/api/auth)**

## Learner Signup (POST /signup/learner): stores learner details with encrypted password.

## Returns JWT with role = learner.

## Admin Signup (POST /signup/admin): stores admin details securely.

## Returns JWT with role = admin.

## Learner Login (POST /login/learner): validates credentials, checks preferences.

## Returns JWT + hasPreferences flag.

## Admin Login (POST /login/admin): validates credentials.

## Returns JWT with admin role.

## Profile (Protected) (GET /profile): validates JWT and returns user details.

## **Preferences APIs (/api/preferences)**

## Set Preferences (POST /learner): learners store language + duration.

## Get Preferences (GET /learner): fetch learner preferences.

## Role-based access enforced (learners only).

## **Middleware & Helpers**

## authMiddleware.js: JWT verification and role extraction.

## db.js: centralized query helper for Postgres.

## **Server Setup**

## server.js: Express with CORS + JSON parsing.

## Health Check Routes:

## /: backend status.

## /db-check: verifies DB connection.

## **3.Security Implementations**

## Passwords hashed with bcrypt before storage.

## JWT signed with process.env.JWT\_SECRET, expiring in 2 hours.

## Role-based access control (learner vs admin).

## Protected routes require Authorization: Bearer header.

## **4.Testing & Verification**

## **Frontend:**

## Form validation tested (invalid email, weak passwords rejected).

## Login → Dashboard flow verified.

## **Backend:**

## /db-check confirms DB connection.

## Postman/Thunder Client tested all endpoints.

## Unauthorized access attempts returned correct 401/403 errors.

## **UI Research:** Conducted analysis on existing language learning platforms to identify strengths and weaknesses, and decided on UI structure and navigation flow aiming for a scalable and user-oriented interface.

## **Database Creation:** Created initial database comprising of admin, users (user credentials, authentication data), and learner preferences (proficiency levels, learning objectives, language preference) using DBMS principles to ensure data integrity and security.

## **User Authentication & Validation:**

## Sign-up process: Features strict validation including strong password policy and email verification to prevent fake registrations.

## Login process: Authenticates users against hashed credentials and establishes a secure, persistent session to prevent unauthorized access to the user dashboard.

## **Dashboard Implementation:** Upon successful login, users are directed to the dashboard which currently functions as a prototype interface, incorporating placeholders for forthcoming features like AI conversational tutor, progress analytics, and cultural-context modules.

## **Time Duration:** Two weeks

## Week 1 → Research + UI wireframe + Database design

## Week 2 → Authentication module + Dashboard prototype + Documentation

## **Sprint-1 Deliverables:**

## UI wireframe

## Database schema

## Authentication system (sign-up, login, validation)

## Dashboard Prototype

## Documentation

## **Sprint 1 Outcome:**

## Objectives were successfully implemented.

## The platform now supports:

## Secure sign-up and login for learners and admins.

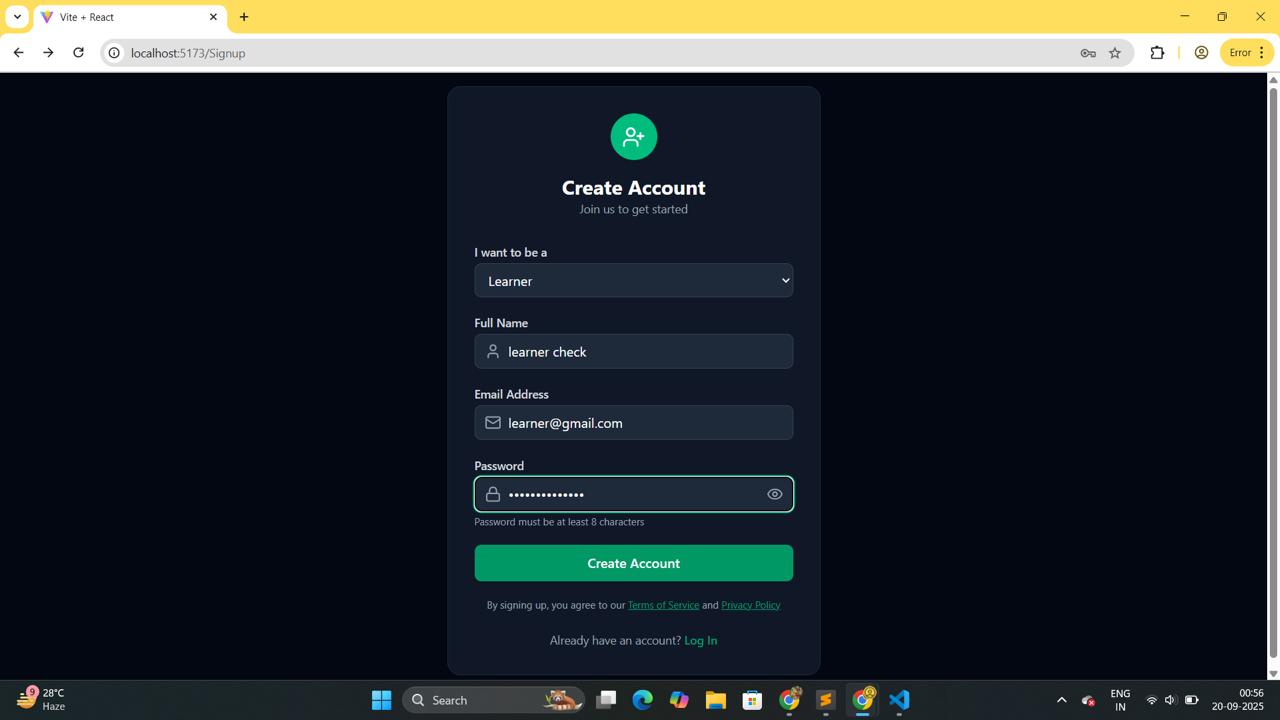
## Database-backed authentication with encrypted credentials.

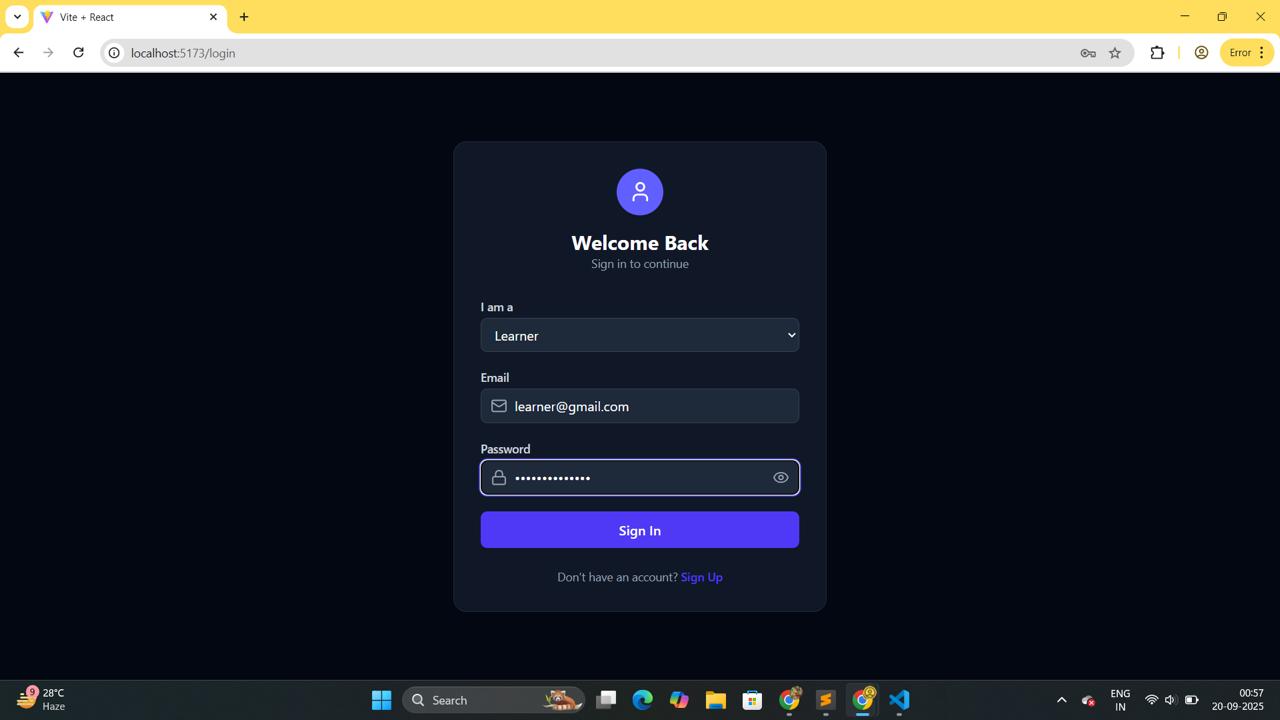
## Preferences storage and retrieval.

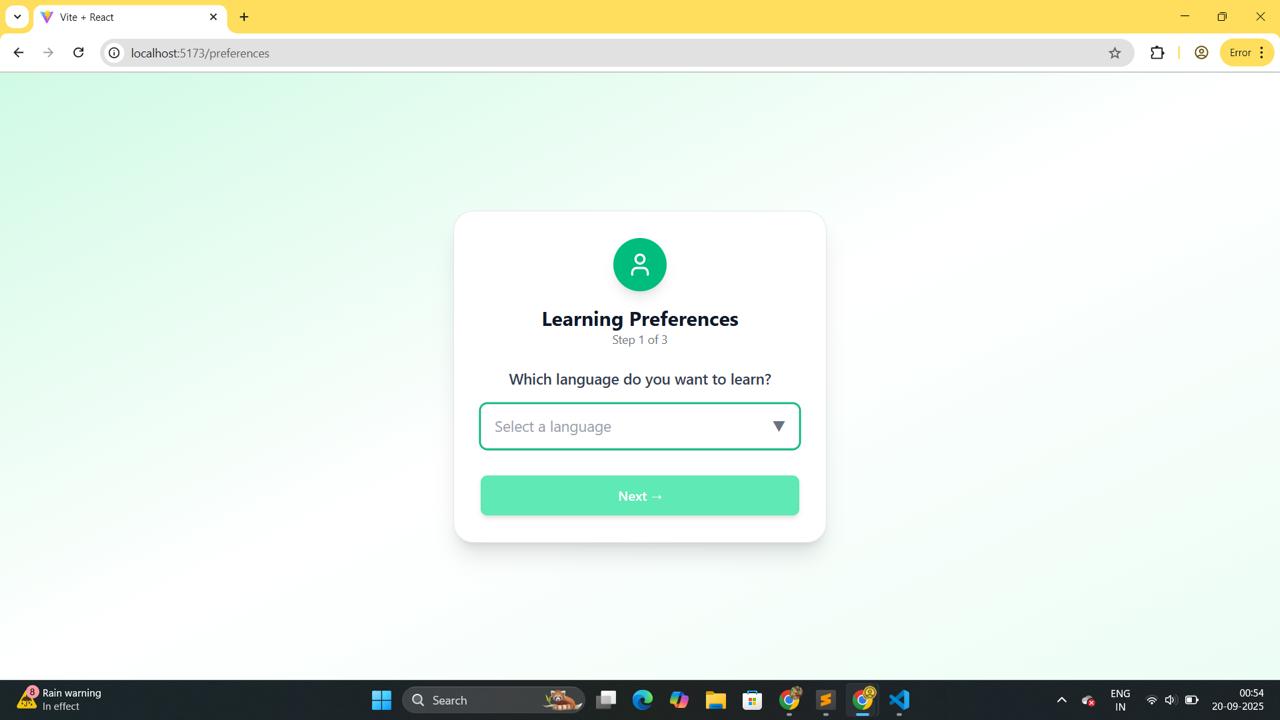
## A prototype dashboard with navigation flow.

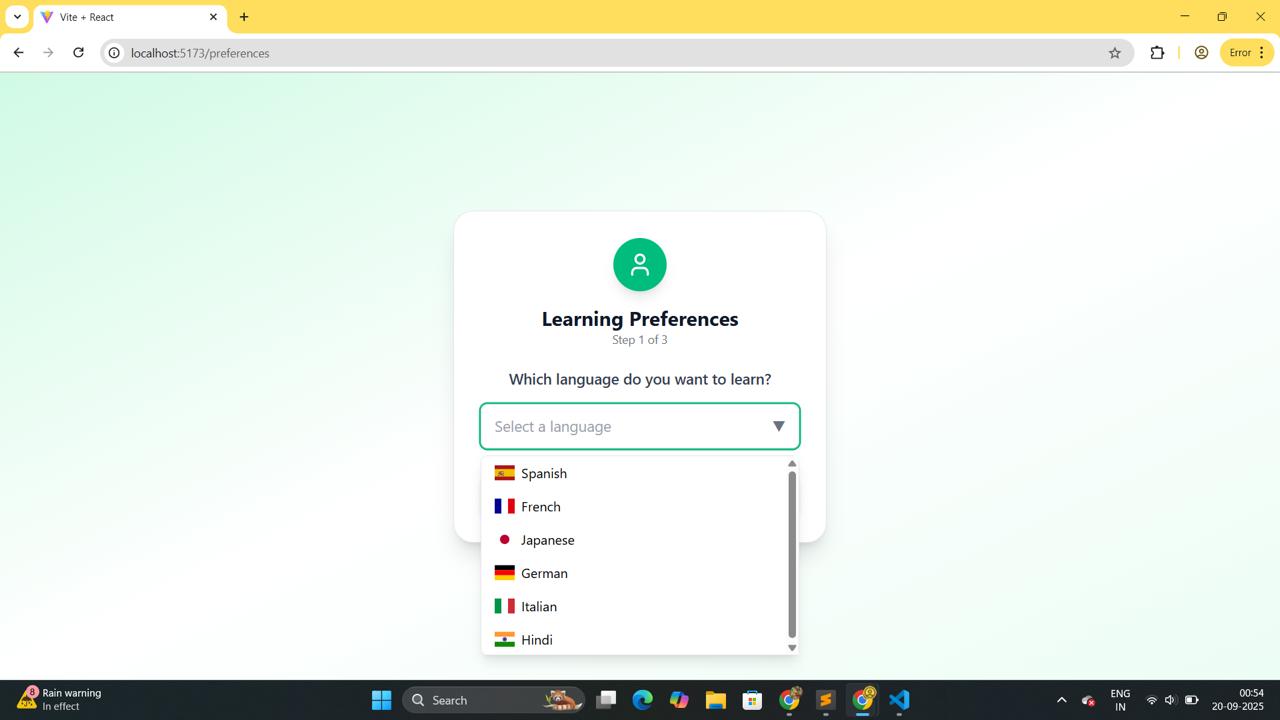
## This Sprint establishes the baseline system architecture for upcoming sprints, including AI tutor integration, admin dashboards, and advanced learner progress tracking.

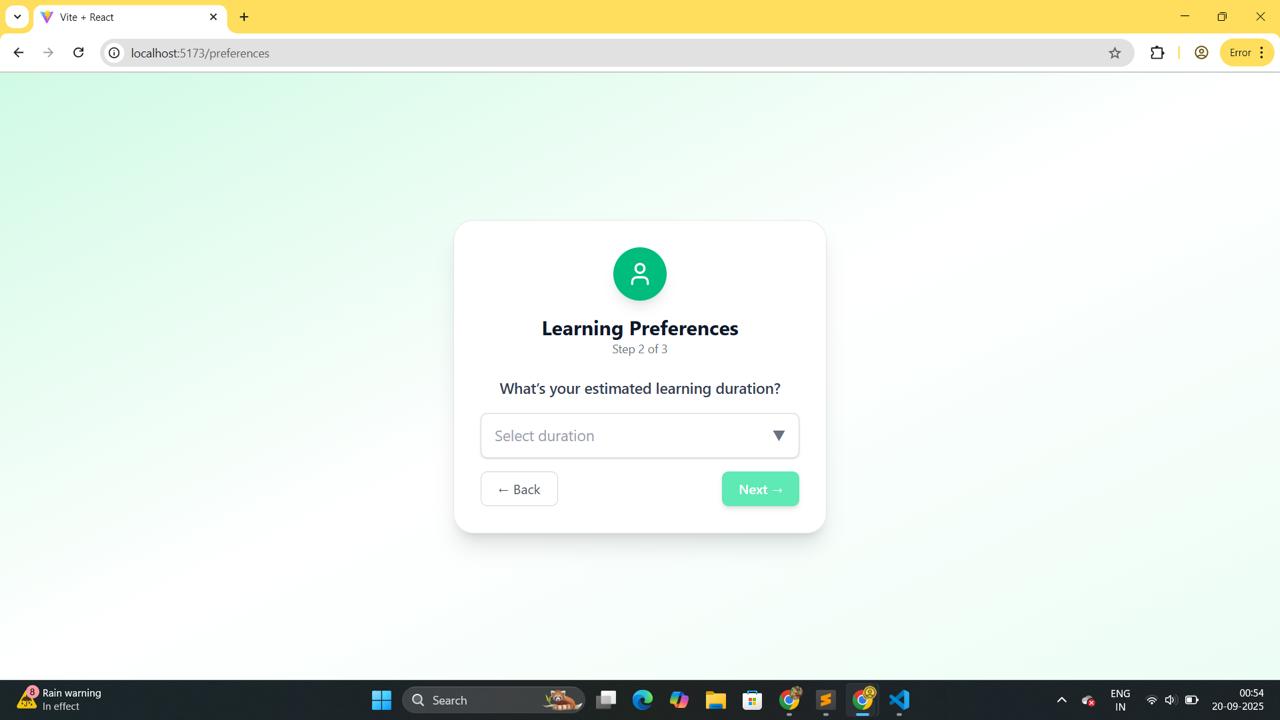
## **Sprint 1 Status**: Completed

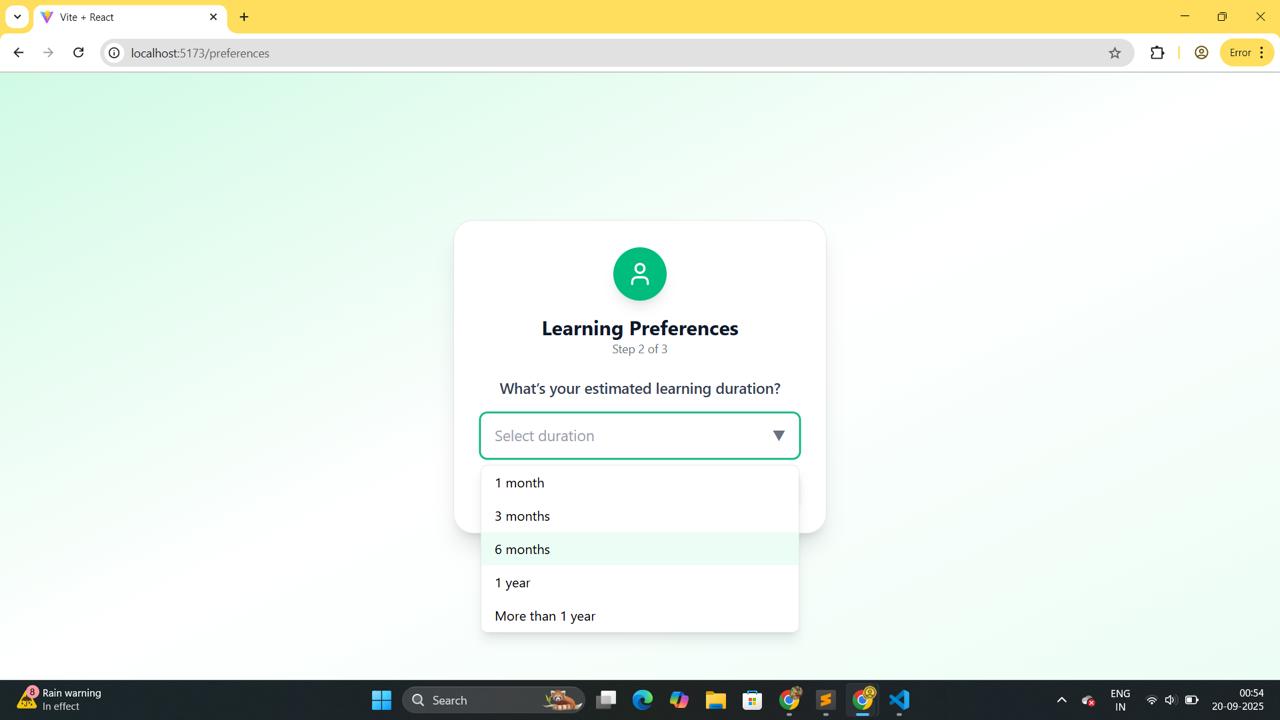


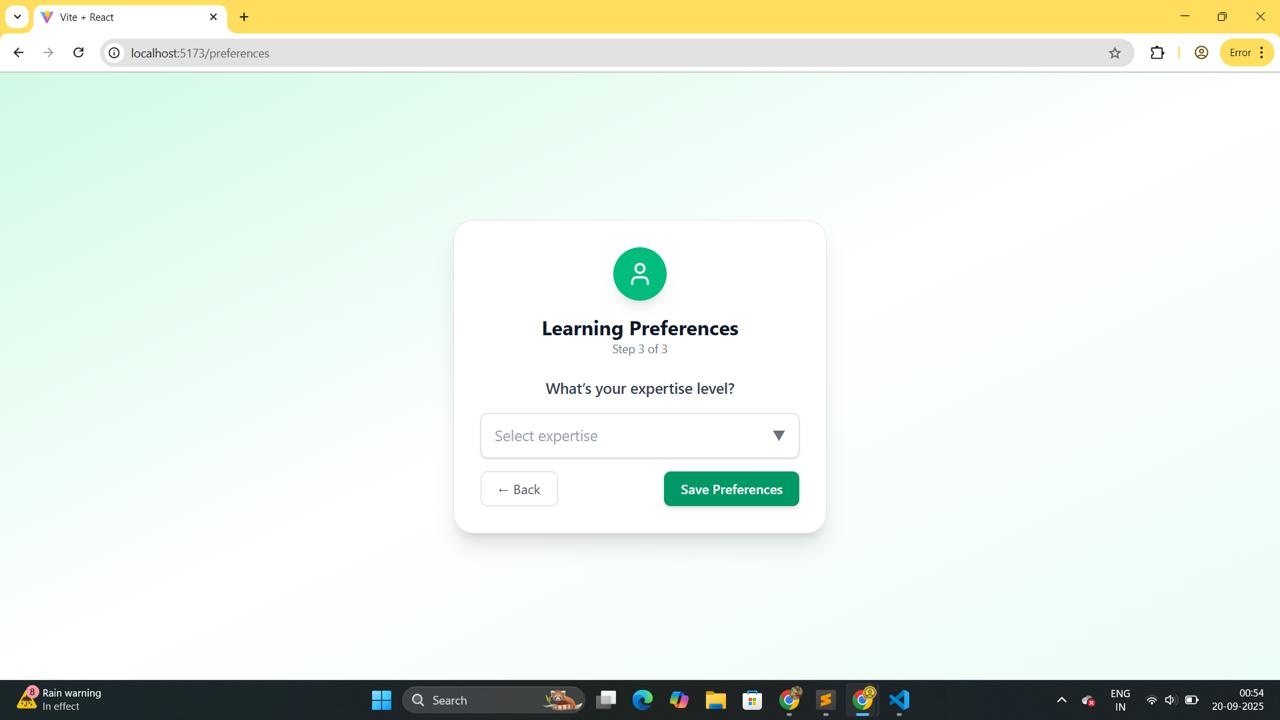


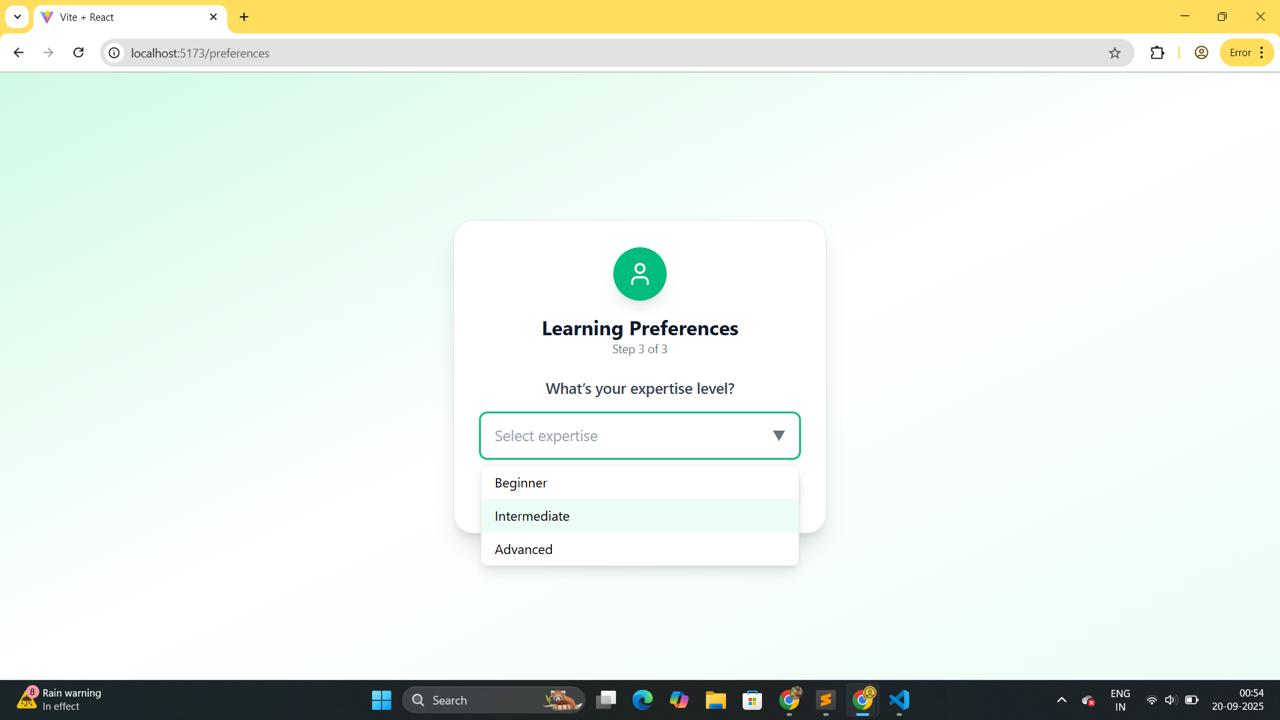


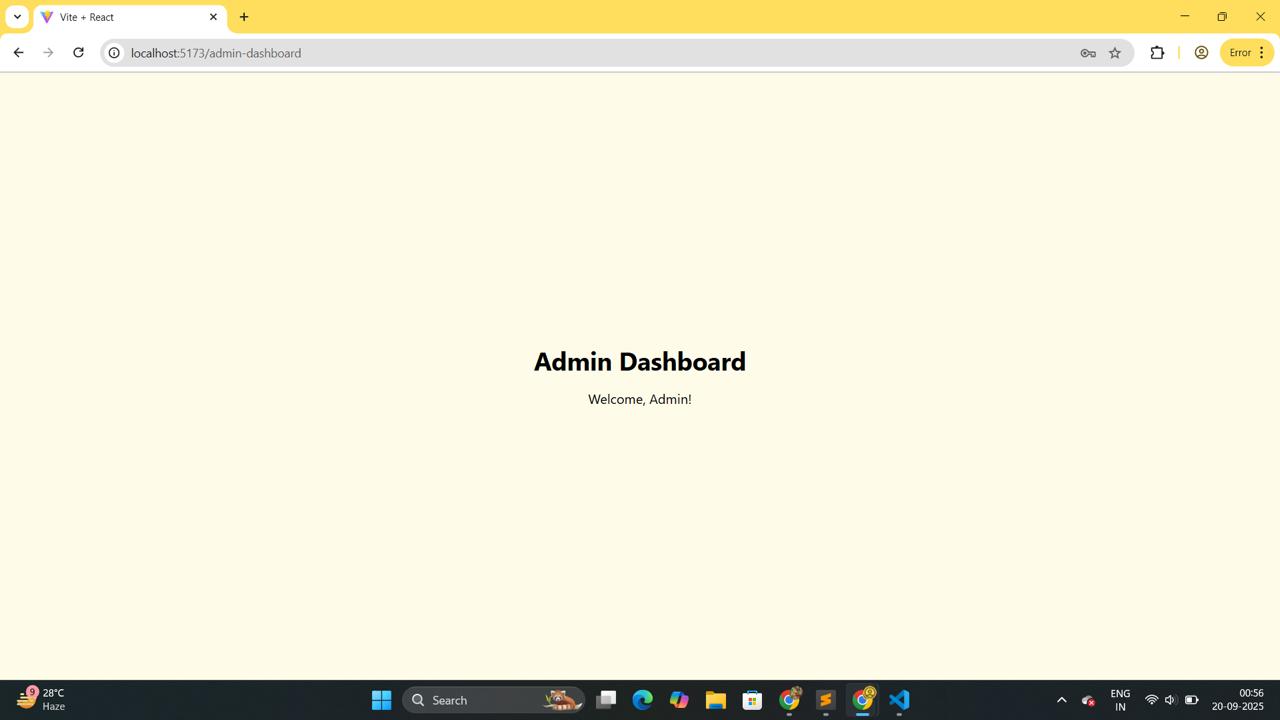


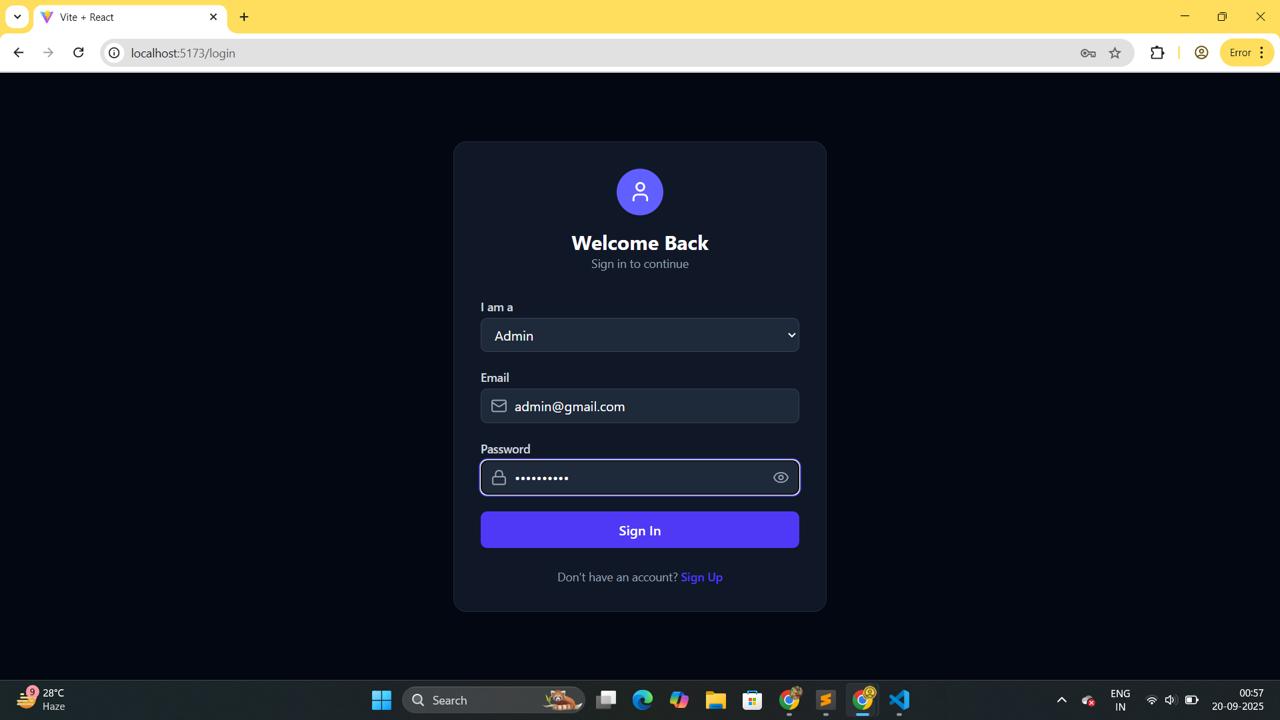


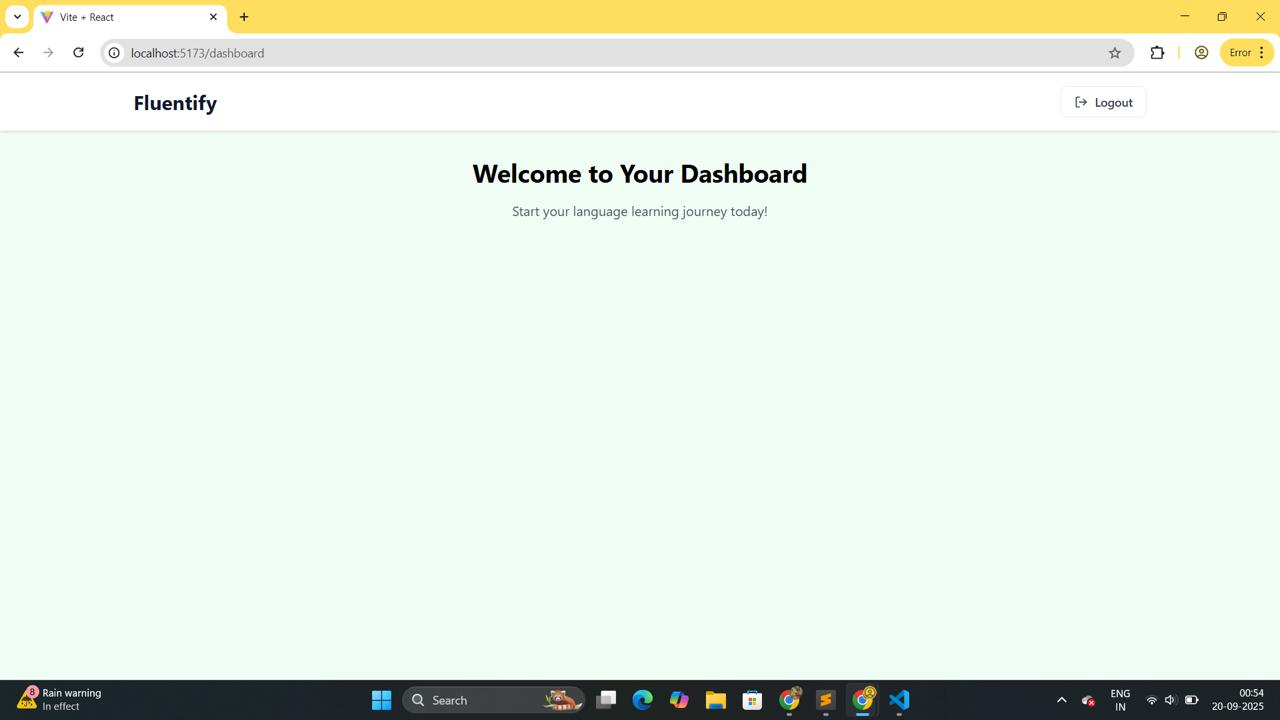












These outputs collectively confirm that Sprint-1 deliverables - **authentication system, and prototype dashboard** were successfully implemented and tested.