Sprint Backlog Plan - SaaS Exam Preparation Application

This document provides a detailed Sprint Backlog Plan for the SaaS Exam Preparation Application. The project will follow the Scrum methodology, using Jira as the tool to manage sprints. The goal is to complete ~80% of the application within 4 sprints over 1 month, with each sprint lasting 1 week.

# Sprint 1: Authentication & Profile

**Epic 1: Project Setup & Infrastructure**

* **Story 1.1:** Setup project repositories (frontend + backend)
  + Sub-task 1.1.1: Create GitHub repo for frontend (React + Next.js).
  + Sub-task 1.1.2: Create GitHub repo for backend (FastAPI + Python).
  + Sub-task 1.1.3: Setup PostgreSQL locally & on cloud (RDS).
  + Sub-task 1.1.4: Setup Docker for containerization.
  + Sub-task 1.1.5: Setup CI/CD pipeline with GitHub Actions.

**Epic 2: Authentication System**

* **Story 2.1:** User Registration & Login
  + Sub-task 2.1.1: Create users table in PostgreSQL.
  + Sub-task 2.1.2: Implement signup API in FastAPI.
  + Sub-task 2.1.3: Implement login API with JWT auth.
  + Sub-task 2.1.4: Create Signup/Login UI in Next.js with TailwindCSS.
  + Sub-task 2.1.5: Connect frontend with backend APIs.

**Epic 3: Profile Management**

* **Story 3.1:** User Profile Page
  + Sub-task 3.1.1: Add profile fields in DB (name, email, mobile, district).
  + Sub-task 3.1.2: Build Profile API (view/edit profile).
  + Sub-task 3.1.3: Create Profile UI (React form).
  + Sub-task 3.1.4: Integrate with backend.

# Sprint 2: Mock Tests & Full Mock Tests

## ****Epic 1: Database & Metadata Setup****

**Story 1.1: Setup metadata storage for test files**

* Sub-task 1.1.1: Create table mock\_test\_files
* Sub-task 1.1.2: Create table user\_results
* Sub-task 1.1.3: Create table user\_answers

## ****Epic 2: Excel Question Handling****

**Story 2.1: Define Excel template for questions**

* Sub-task 2.1.1: Define columns (ID, Questions, Options, Correct Answer, Explanation, Subject, Difficulty)
* Sub-task 2.1.2: Create partial test Excel (Aptitude, Reasoning, English)
* Sub-task 2.1.3: Create full mock test Excel

**Story 2.2: Backend support for reading Excel**

* Sub-task 2.2.1: Implement Excel → JSON using pandas
* Sub-task 2.2.2: API GET /tests/:id/questions (pagination 10/page)
* Sub-task 2.2.3: Hide correct answers until submission

## ****Epic 3: Test Selection & Instructions****

**Story 3.1: Test selection page**

* Sub-task 3.1.1: Frontend page listing Partial Mock Tests
* Sub-task 3.1.2: Frontend page listing Full Mock Tests
* Sub-task 3.1.3: Connect with backend GET /tests

**Story 3.2: Instructions page**

* Sub-task 3.2.1: Display test details (duration, number of Qs, rules)
* Sub-task 3.2.2: Add **Start Test** button

## ****Epic 4: Test Attempt Experience****

**Story 4.1: Question display & navigation**

* Sub-task 4.1.1: Display 10 questions per page
* Sub-task 4.1.2: Add Next/Previous navigation
* Sub-task 4.1.3: Add question grid (jump to specific Q)
* Sub-task 4.1.4: Add timer countdown (auto-submit)

**Story 4.2: Answer submission**

* Sub-task 4.2.1: API POST /attempts/:id/save
* Sub-task 4.2.2: Auto-save answers on selection
* Sub-task 4.2.3: API POST /attempts/:id/submit

## ****Epic 5: Results & Explanations****

**Story 5.1: Backend evaluation**

* Sub-task 5.1.1: Compare answers with Excel correct answer
* Sub-task 5.1.2: Calculate score & accuracy
* Sub-task 5.1.3: Store results in DB

**Story 5.2: Results UI**

* Sub-task 5.2.1: Display score summary
* Sub-task 5.2.2: Show correct vs selected answers
* Sub-task 5.2.3: Display explanations from Excel

**Story 5.3: Email integration**

* Sub-task 5.3.1: Format email with score, answers, explanations
* Sub-task 5.3.2: Send email via SMTP
* Sub-task 5.3.3: Add email delivery logging

## ****Epic 6: Integration & Testing****

**Story 6.1: Frontend ↔ Backend integration**

* Sub-task 6.1.1: Connect test list page to GET /tests
* Sub-task 6.1.2: Connect attempt UI to GET /tests/:id/questions
* Sub-task 6.1.3: Connect submission → POST /attempts/:id/submit
* Sub-task 6.1.4: Connect results page to GET /attempts/:id/result

**Story 6.2: Testing & QA**

* Sub-task 6.2.1: Write backend unit tests (pytest)
* Sub-task 6.2.2: Manual test Partial Mock Test flow
* Sub-task 6.2.3: Manual test Full Mock Test flow
* Sub-task 6.2.4: Verify email delivery in results

# Sprint 3: Mock Tests & AI Assistance

## Epics

* - Mock Tests
* - AI Assistance

## User Stories

* - As a student, I want to attempt a full mock test so that I can simulate a real exam.
* - As a student, I want to see my score immediately after submission.
* - As a student, I want to ask doubts and get instant AI-powered answers.
* - As a student, I want daily AI feedback on my performance.

## Tasks

* - Implement Mock Test Generator (random questions from DB).
* - Build Mock Test UI (timer, navigation, submit button).
* - Integrate scoring API in FastAPI.
* - Integrate AI Doubt Solver (Hugging Face/LangChain).
* - Generate Daily Performance Feedback using AI.

# Sprint 4: Payments, Analytics & Monitoring

## Epics

* - Payments
* - Analytics
* - Monitoring

## User Stories

* - As a student, I want to purchase a subscription so that I can unlock mock tests & premium features.
* - As a student, I want to view analytics of my performance so that I can improve weak areas.
* - As a dev team, we want monitoring alerts so that the system is stable during peak exam hours.

## Tasks

* - Integrate Razorpay/Stripe for subscription payments.
* - Store transaction details in DB.
* - Build Performance Dashboard (Chart.js + Next.js).
* - Setup Monitoring: Prometheus + Grafana (metrics), ELK (logs), Sentry (error alerts).
* - Add Gamification (coins/rewards for students).