**Project Overview**

**Core Requirements**

**1. User Types**

1. **Administrator**:
   * Register/login via an institutional account.
   * Create and manage tests.
   * Upload question banks or manually add questions.
   * Choose question types (MCQ, True/False, Text-based).
   * Configure test settings:
     + Randomization of questions.
     + Number of questions per test.
   * Share test links with students.
   * View test results.
2. **Student**:
   * Register/login to the platform.
   * Access the dashboard to view active tests.
   * Join tests via provided links.
   * Submit responses.

**2. Platform Features**

1. **Landing Page**:
   * A simple landing page providing an overview of the platform.
2. **Administrator Dashboard**:
   * Test creation and management interface.
   * Upload and organize question banks.
   * Configure test settings.
   * Share links to tests.
3. **Student Dashboard**:
   * Display active tests.
   * Enable test-taking functionality.
4. **Randomized Question Distribution**:
   * Generate unique sets of questions for each test-taker.
5. **Data Security**:
   * Store test data and question banks securely using decentralized storage (IPFS).
   * Ensure immutable logs for exams and results using blockchain (Solidity smart contracts).
6. **Basic Authentication**:
   * Login/registration for both user types.
   * Basic role-based differentiation between administrators and students.

**Technology Stack**

1. **Frontend**:
   * React.js (for dynamic and responsive UI).
   * CSS (for styling).
2. **Backend**:
   * Spring Boot (to handle APIs, business logic, and database communication).
3. **Database**:
   * PostgreSQL (for secure storage of user data, exam details, and results).
4. **Blockchain**:
   * Solidity (for creating smart contracts).
   * Ganache (local blockchain development environment).
5. **File Storage**:
   * IPFS (for decentralized and secure file storage).
6. **Deployment**:
   * Localhost for initial development and testing.

**Development Phases**

**Phase 1: Core Functionality**

1. Build the frontend with React.js and CSS.
2. Develop backend APIs with Spring Boot:
   * User authentication (basic login and registration).
   * Test creation and retrieval.
   * Question bank management.
3. Integrate PostgreSQL for data persistence.

**Phase 2: Blockchain Integration**

1. Create and deploy Solidity smart contracts to record test metadata.
2. Use Ganache for local blockchain testing.

**Phase 3: File Storage**

1. Implement IPFS for decentralized storage of question banks.
2. Integrate IPFS with the backend.