Web application name: Intern Ease

**Functional Requirements:**

**User Registration**

1. Enable role-based user registration.
2. Create a registration form with essential fields.
3. Enable user account creation.
4. Ensure strong passwords, reset options, and secure authentication.
5. Enable privacy settings (Public /Private)
6. Validate account data and verify emails.
7. Implement account deactivation.
8. Enable students to add location and industry preferences to their account profile.

* **User Account Profile Authentication and Access:**
  + All users must authenticate before accessing their respective functionalities.
  + Users can only view and update their own user profiles.
  + **User Account Management:**
  + Students, employers, and university staff can update their account profiles.
  + Only the administrator can manage user accounts globally.

**1. User Registration:**

* **Email Verification:**
  + Users register with their email. (for student & staff should use their university emails )
  + A verification email is sent to the registered email address.
  + Users click on a verification link to confirm their email.
* **Password Requirements:**
  + Enforce strong password policies (e.g., minimum length, special characters).

**2. User Login:**

* **Secure Password Storage:**
  + Hash and salt passwords before storing them in the database using a strong hashing algorithm (e.g., bcrypt).

**3. User Account Management:**

* **Password Reset:**
  + Allow users to reset their password through a secure process.
  + Send a password reset link to the registered email.

**4. Security Best Practices:**

* **HTTPS:**
  + Ensure the entire application uses HTTPS to encrypt data in transit.
* **User Permissions:**
  + Implement role-based access control (RBAC) to restrict access based on user roles.
  + Roles: Student, Employer, University Staff, Administrator.

**Internship Listings**

1. Manage an internship database.
2. Enable keyword search, filters, and sorting.
3. Maintain real-time updates
4. Let partner employers post internships.
5. Send notifications to students about new internships that match their location and industry preferences.
6. Send notifications to university staff about all new internships.

* **Internship Listings Authentication and Access:**
  + Students can only view and search internships.
  + Employers can manage only their own provided internships.
  + University staff can manage all internship listings.
  + The administrator can manage all internship listings.
* **Internship Approval:**
  + Only university staff and the administrator can approve or reject internships.

**Approving Internships:**

1. **Internship Submission:**

1.1 Employers or university staff log in to their respective accounts.

1.2 They navigate to the "Add Internship" section, fill in the required details, and submit the internship opportunity.

2. **Initial Review:**

2.1 The submitted internships are directed to the "Pending Approval" section.

2.2 The system administrator logs in and accesses the "Internship Approval" dashboard.

2.3 The administrator reviews the submitted internship details for completeness, accuracy, and adherence to system standards.

3. **Approval Decision:**

3.1 If the internship passes the detailed review, the administrator approves it for publication.

3.2 Approved internships are moved to the "Published" section, making them visible to students.

3.3 If the internship requires revisions, it is moved back to the "Pending Approval" section, and the employer or university staff is notified.

4. **Notification to Users:**

4.1 Once an internship is approved, students who match the specified preferences receive notifications about the new opportunity.

4.2 Employers and university staff are notified of the approval status of their submitted internships.

**Search and Filtering:**

Here are common criteria for searching and filtering internships:

1. **Location:**
   * **City:** Students can filter internships based on the city where they prefer to work.
2. **Industry:**
   * **Industry Category:** Students can select specific industry categories such as technology, finance, healthcare, etc.
   * **Sub-Industry:** Further refinement within industry categories, allowing students to narrow down their search.
3. **Duration:**
   * **Internship Length:** Students can filter internships based on the preferred duration, such as summer internships, semester-long internships, or short-term projects.
4. **Paid/Unpaid:**
   * **Compensation Type:** Students can filter internships based on whether they are paid or unpaid
5. **Remote/On-site**

* Students can filter internships based on it is remote or on-site.

1. **Application Deadline:**
   * **Deadline Date:** Students can filter internships based on application deadlines

**User (Students) Ratings and Reviews**

1. Enable students to submit reviews for their internship experiences.

2. Enable students to provide ratings for their internship experiences(implement 5-star rating)

* **Ratings and Reviews Authentication and Access:**
  + Students can rate and review internships.
  + Employers can view ratings and reviews for their provided internships.

**Rating and Review System:**

**1. Rating and Review Process**

1.1 **Internship Completion:**

* After completing an internship, students navigate to access the 'Internships listings' section.

1.2 **Review Submission:**

* Students can choose to submit a review for the completed internship.
* The review form includes fields such as work environment, tasks, mentorship, and overall satisfaction.

1.3 **Rating Scale:**

* A rating scale (e.g., 1 to 5 stars) is provided for quantitative feedback on different aspects of the internship.

**2. Review Visibility**

2.1 **Peer Access:**

* Submitted reviews are made accessible to other students through the internship listing.
* Peers can view overall ratings and read detailed reviews for each internship opportunity.

**Notifications**

1. Send email alerts with verification.
2. Display in-app alerts.
3. Management and viewing of notifications are centralized.

* **Notifications Authentication and Access:**
  + Users receive notifications relevant to their role and activities.
  + The administrator will receive system-wide notifications about critical activities.

**Notification System**

1. **New Internship Notification:**
   * **Requirement:**
     + Students will receive notifications about new internships that match their location and industry preferences.
   * **Conditions Triggering:**
     + Whenever a new internship listing is approved that aligns with a student's preferences.
   * **Delivery:**
     + Real-time notification through the application.
     + Optionally, an email notification can be sent.
2. **System-Wide Notifications:**
   * **Requirement:**
     + The administrator will receive system-wide notifications about critical activities.
   * **Conditions Triggering:**
     + Account management actions (e.g., user activation, deactivation).
     + Approval or rejection of internships.
   * **Delivery:**
     + Real-time notification through the application.
     + Optionally, an email notification can be sent.
3. **Notification Center:**
   * **Requirement:**
     + Users should have access to a notification center displaying a history of their notifications.
   * **Conditions Triggering:**
     + Anytime a new notification is generated for the user.
   * **Delivery:**
     + Notifications are displayed in a centralized notification center within the application.
4. **Notification Read Status:**
   * **Requirement:**
     + Users should have the ability to mark notifications as read.
   * **Conditions Triggering:**
     + Whenever a user accesses the notification or notification center.
   * **Delivery:**
     + Changes in read status are reflected in real-time.

**Admin Panel**

1. Establish admin roles, secure login, and multi-factor authentication.
2. Manage user roles (student, employer, and university staff) and accounts.
3. Administer internship listings.
4. Moderate user reviews**.**

**DB Design:**

1**. User Table**

- UserID (Primary Key)

- UserType (Student, Employer, University Staff, Administrator)

- FirstName

- LastName

- Email

- PasswordHash

- Privacy

2. **Student Table**

- StudentID (Foreign Key referencing User.UserID)

- LocationPreference

- IndustryPreference

3. **Employer Table**

- EmployerID (Foreign Key referencing User.UserID)

- CompanyName

- ContactPersonName

- ContactPersonMail

- ContactPersonMobile

- Address

4. **UniversityStaff Table**

- StaffID (Foreign Key referencing User.UserID)

5. **Administrator Table**

- AdminID (Foreign Key referencing User.UserID)

6**. Internship Table**

- InternshipID (Primary Key)

- Title

- Description

- Location

- Industry

- Requirement

- EmployerID (Foreign Key referencing Employer.EmployerID)

- UniversityStaffID (Foreign Key referencing UniversityStaff.StaffID)

- IsApproved (Boolean indicating whether the internship is approved)

- ApplicationDeadLine

- PaidOrUnpaid

- Duration

- SummerOrYear (internship during summer or not)

- RemoteOrOnsite

7. **Rating Table**

- RatingID (Primary Key)

- Score

- Comment

- StudentID (Foreign Key referencing Student.StudentID)

- InternshipID (Foreign Key referencing Internship.InternshipID)

8. **Review Table**

- ReviewID (Primary Key)

- Content

- StudentID (Foreign Key referencing Student.StudentID)

- InternshipID (Foreign Key referencing Internship.InternshipID)

9. **Notification Table**

- NotificationID (Primary Key)

- Content

- UserID (Foreign Key referencing User.UserID)

- InternshipID (Foreign Key referencing Internship.InternshipID, nullable)

- ReadStatus

**Notes:**

* The **User** table holds common user information, and the **UserType** field distinguishes between different types of users.
* Each user type has its own associated table (**Student**, **Employer**, **UniversityStaff**, **Administrator**) to store specific information related to that user type.
* The **Internship** table represents internship opportunities, and it includes references to the **Employer** and **UniversityStaff** who are involved.
* The **Rating** table is used to store ratings given by students for internships.
* The **Review** table is used to store written reviews by students for their internship experiences.
* The **Notification** table is designed to handle notifications for new internships.

Add constraints and indices as required

**Classes**

1. **User Class:**
   * *Attributes:*
     + **UserID** (Primary Key)
     + **FirstName**
     + **LastName**
     + **Email**
     + **PasswordHash**
     + **UserType** (Student, Employer, UniversityStaff, Administrator)
     + **IsActive** (Boolean indicating whether the user account is active)
     + **Privacy** (Private/Public)
   * *Methods:*
     + **registerAccount(userInfo)**: Registers a new user account.
     + **login(email, password)**: Authenticates the user.
     + **updateProfile(userID, newInfo)**: Updates user profile information.
     + **deactivateAccount(userID)**: Deactivates a user account.
2. **Student Class: (extends User)**
   * *Attributes:*
     + **StudentID** (Primary Key, Foreign Key referencing User.UserID)
     + **LocationPreference**
     + **IndustryPreference**
   * *Methods:*
     + **createStudentProfile(userID, preferences)**: Creates a new student profile.
     + **searchInternships(criteria)**: Searches and filters internship opportunities based on specified criteria.
     + **rateInternship(studentID, internshipID, score, comment)**: Allows a student to rate and review an internship.
     + **getStudentProfile(userID)**: Retrieves the student's profile information.
3. **Employer Class: (extends User)**
   * *Attributes:*
     + **EmployerID** (Primary Key, Foreign Key referencing User.UserID)
     + **CompanyName**
   * *Methods:*
     + **createEmployerProfile(userID, companyInfo)**: Creates a new employer profile.
     + **provideInternship(employerID, title, description, location, industry)**: Allows an employer to provide a new internship opportunity.
     + **getEmployerProfile(userID)**: Retrieves the employer's profile information.
4. **UniversityStaff Class: (extends User)**
   * *Attributes:*
     + **StaffID** (Primary Key, Foreign Key referencing User.UserID)
   * *Methods:*
     + **createStaffProfile(userID, staffInfo)**: Creates a new university staff profile.
     + **manageInternship(internshipID, approvalStatus)**: Manages the approval status of internships.
     + **getStaffProfile(userID)**: Retrieves the university staff's profile information.
5. **Administrator Class: (extends User)**
   * *Attributes:*
     + **AdminID** (Primary Key, Foreign Key referencing User.UserID)
   * *Methods:*
     + **createAdminProfile(userID, adminInfo)**: Creates a new administrator profile.
     + **manageUserAccounts(adminID, userID, action)**: Manages user accounts (e.g., activate, deactivate).
     + **manageInternships(adminID, internshipID, action)**: Manages internships (e.g., approve, reject).
     + **getAdminProfile(userID)**: Retrieves the administrator's profile information.
6. **Internship Class:**
   * *Attributes:*
     + **InternshipID** (Primary Key)
     + **Title**
     + **Description**
     + **Location**
     + **Industry**
     + **EmployerID** (Foreign Key referencing Employer.EmployerID)
     + **UniversityStaffID** (Foreign Key referencing UniversityStaff.StaffID)
     + **IsApproved** (Boolean indicating whether the internship is approved)
   * *Methods:*
     + **notifyStudents(interestCriteria)**: Notifies students about new internships that match their preferences.
     + **getInternshipDetails(internshipID)**: Retrieves detailed information about a specific internship.
     + **updateInternshipStatus(internshipID, newStatus)**: Updates the status of an internship.
7. **Rating Class:**
   * *Attributes:*
     + **RatingID** (Primary Key)
     + **Score**
     + **Comment**
     + **StudentID** (Foreign Key referencing Student.StudentID)
     + **InternshipID** (Foreign Key referencing Internship.InternshipID)
   * *Methods:*
     + **getAverageRating(internshipID)**: Retrieves the average rating for a given internship.
8. **Review Class:**
   * *Attributes:*
     + **ReviewID** (Primary Key)
     + **Content**
     + **StudentID** (Foreign Key referencing Student.StudentID)
     + **InternshipID** (Foreign Key referencing Internship.InternshipID)
   * *Methods:*
     + **getReviews(internshipID)**: Retrieves reviews for a given internship.
9. **Notification Class:**
   * *Attributes:*
     + **NotificationID** (Primary Key)
     + **Content**
     + **UserID** (Foreign Key referencing User.UserID)
     + **InternshipID** (Foreign Key referencing Internship.InternshipID, nullable)
     + **IsRead** (Boolean indicating whether the notification has been read)
   * *Methods:*
     + **sendNotification(userID, content)**: Sends a notification to a user.
     + **markAsRead(notificationID)**: Marks a notification as read.

**Technology Stack:**

**1.Programming Language:**

* **Backend:** 
  + **Recommendation: Python**
    - **libraries (Django, Flask)**
* **Frontend:** 
  + **Recommendation: JavaScript (ES6+)**

**2. Web Frameworks:**

* **Backend Framework:**
  + **Recommendation: Django (Python)**
* **Frontend Framework:**
  + **Recommendation: React.js**

**3. Database Management System:**

* **Choose a database management system for storing and retrieving data.**
  + **Recommendation: MySQL**

**4. Authentication and Authorization:**

* **Implement secure user authentication and authorization mechanisms.**
  + **Authentication Library:**
    - **Recommendation: Django Allauth (Python)**
  + **Authorization Library:**
    - **Recommendation: Django Guardian (Python)**

**5. API Development:**

* **Develop a robust API for communication between frontend and backend components.**
  + **Recommendation: Django REST Framework (Python)**

**6. Real-Time Communication:**

* **Implement real-time features, such as notifications.**
  + **Recommendation: WebSocket with Django Channels (Python)**
    - **Extends Django to handle WebSockets, enabling real-time communication.**

**7. Testing:**

* **Ensure code reliability and quality through testing.**
  + **Testing Framework:**
    - **Recommendation: Jest (for React.js), Django TestCase (for Django)**

**Testing Plan:**

**1. Unit Testing:**

* **Objective:** Ensure individual units of code (functions, methods) perform as expected.
* **Tools:**
  + Django TestCase for backend Python code.
  + Jest for React.js frontend code.
* **Testing Focus:**
  + Core functionalities such as user authentication, internship submission, and data retrieval.

**2. Integration Testing:**

* **Objective:** Validate the interaction between different components (backend, frontend, database).
* **Tools:**
  + Django TestCase for backend integration.
  + React Testing Library for frontend integration.
* **Testing Focus:**
  + Data flow between frontend and backend.
  + API integrations.

**3. End-to-End (E2E) Testing:**

* **Objective:** Verify the entire application workflow from user interaction to database storage.
* **Tools:**
  + Selenium or Cypress for E2E testing.
* **Testing Focus:**
  + User registration and login.
  + Internship submission and approval process.

**Deliverables arrangements:**

* The application source code with clear comments
* The download link of the platform which will be used to test the code locally on my computer.
* The DB files ( MySQL ) containing sample data