Hackathon Project Requirement: Crime Reporting and Community Verification Platform

Project Overview:

The goal of this project is to create a web application that allows users to report crimes in their area, attach evidence (images/videos), and enable the community to verify the authenticity of the reports through upvotes, downvotes, and comments with mandatory proof attachments. The platform will also include AI-generated descriptions for uploaded images, user authentication, and a robust system for filtering, sorting, and searching crime reports.

Functional Requirements:

1. User Authentication and Authorization

- **User Registration**: Users can register with their email, password, and phone number. By default, users are marked as "unverified."
- **User Login**: Users can log in using their email and password.
- **Password Management**:
    - Users can change their password.
    - Users can recover their password via email or phone number using OTP.
- **Refresh Token**: Implement a refresh token mechanism to generate new access tokens.
- **Phone Number Verification**: Unverified users can verify their identity by entering an OTP sent to their phone number. No admin verification is required.
- **Admin Ban**: Admins can ban any user at any time, preventing them from posting, commenting, or interacting with the platform.

2. Crime Reporting

- **Report a Crime**:
    - Only verified users (those who have completed OTP verification) can report crimes.
    - Users must upload **at least one image** of the crime scene (video is optional).

    - Users must select the **division** and **district** of the crime (use a free API or a predefined list of Bangladesh divisions and districts).
    - The system will auto-generate a description of the crime scene using an AI tool (e.g., OpenAI's GPT or similar) based on the uploaded image. Users can modify the description before posting.
    - If a video is uploaded, no AI-generated description will be provided. The user must manually add a description.
- **Crime Post Details**:
    - Title (user-defined).
    - Description (AI-generated for images and user-editable; manually added for videos).
    - Division and district.
    - Image(s) and optional video.
    - **Post Time**: Timestamp of when the post is submitted.
    - **Crime Time**: Timestamp of when the crime occurred (user-defined).

3. Community Interaction

- **Upvote/Downvote**: Users can upvote or downvote crime posts based on their perceived authenticity or importance.
- **Comments with Proof**:
    - Users can comment on crime posts to add additional context or proof.
    - Proof attachment (image/video) is mandatory for commenting.
- **Post Verification Score**: Each post will have a score based on upvotes, downvotes, and verified comments.

4. Crime Feed

- **Pagination**: Display crime posts in a paginated manner.
- **Filtering**: Users can filter posts by division, district, or verification score.
- **Sorting**: Users can sort posts by date, upvotes, or verification score.
- **Searching**: Users can search for posts by keywords in the title or description.

5. User Roles

- **Unverified User**:
    - Can view crime posts.
    - Cannot post crimes, comment, upvote, or downvote.
- **Verified User**:
    - Can post crimes, comment, upvote, and downvote.
- **Admin**:
    - Can view all posts, users, and comments.

    - Can remove inappropriate posts or comments.
    - Can ban any user at any time.

6. User Profile

- Users will have a profile page showcasing their details.
- **Profile Image**: Users must upload a profile picture, which will be displayed on their profile and in their interactions (posts, comments).
- **Crime Reports**: Displays a list of crime reports filed by the user, including details like report title, location, and date.
- **Other Information**: Additional fields such as bio and contact information (optional).
- **Edit Profile**: Users can update their profile picture, bio, and other details through an edit option.

Non-Functional Requirements:

1. **Security**:
    - Hash passwords using a secure algorithm (e.g., bcrypt).
    - Use JWT for authentication and authorization.
    - Implement security best practices to protect against common vulnerabilities (e.g., SQL injection, XSS, CSRF).
2. **Responsive Design**: The application should be mobile-friendly and responsive.

Technical Requirements (Technology-Agnostic):

1. Frontend:

- **Framework**: Use any modern frontend framework or library (e.g., React, Angular, Vue.js, Svelte, or plain HTML/CSS/JavaScript).
- **Styling**: Use any CSS framework or custom styling (e.g., Tailwind CSS, Bootstrap, Material-UI, or SCSS).
- **State Management**: Optional, depending on the framework (e.g., Redux for React, Vuex for Vue.js, or built-in state management tools).

2. Backend:

- **Framework**: Use any backend framework (e.g., Node.js with Express, Django, Flask, Spring Boot, Ruby on Rails, or Laravel).

- **Database**: Use any database system (e.g., MongoDB, PostgreSQL, MySQL, Firebase, or SQLite).
- **File Storage**: Use any cloud storage service (e.g., AWS S3, Firebase Storage, min.io).

3. AI Integration:

- Use any AI tool or API for auto-generating descriptions from images (e.g., OpenAI's GPT, Google Vision API, or custom machine learning models).

4. APIs:

- Use any free or paid API for fetching divisions and districts of Bangladesh (or create a static list if no API is available).
- Use any OTP service for phone number verification (e.g., Firebase, Nexmo, or a local provider).

5. Hosting (This is a bonus feature. Not mendatory):

- **Frontend**: Deploy on any platform (e.g., Netlify, Vercel, GitHub Pages, or Firebase Hosting).
- **Backend**: Deploy on any platform (e.g., Heroku, AWS, DigitalOcean, or Render).
- **Database**: Use any cloud database service or self-hosted database.

Key instructions:

- **Flexibility**: Teams are free to choose any technology stack they are comfortable with.
- **Focus on Core Features**: Prioritize implementing the core functionality (crime reporting, community interaction, and user authentication) over advanced features.
- **Scalability**: Ensure the application is designed in a way that it can be scaled or extended in the future.

Additional Features:

1. **Heatmap**: Display a heatmap of crime reports based on location.
2. **Leader board**: Show top contributors (users with the most posts or helpful comments).

Please note, some features of this problem/project will be disclosed at Hackathon event day morning at 12 February.

—--------

New features for Hackathon Day (12 February):

More features for the day of hackathon

1. Real-Time Notifications

- **Description: Implement a real-time notification system to alert users about updates on their crime reports, comments, or upvotes/downvotes.**
- **What you need to do:**
  - **Use WebSockets or a real-time database (e.g., Firebase) to push notifications.**
  - **Notifications should include:**
    - **New comments on their posts.**
    - **Upvotes/downvotes on their posts or comments.**
    - **Admin actions (e.g., post removal or user ban).**
- **Bonus: Allow users to customize which notifications they want to receive.**

2. Crime Report Verification Badge

- **Description: Introduce a system where highly upvoted and verified crime reports receive a "Verified" badge.**
- **What you need to do:**
  - **Define criteria for verification (e.g., minimum upvotes, verified comments with proof, etc.).**
  - **Implement a backend process to automatically assign the badge to qualifying posts.**
  - **Display the badge prominently on the crime feed and post details.**
- **Bonus: Allow admins to manually verify posts and override the automated system.**

3. Anonymous Reporting

- **Description: Allow users to report crimes anonymously.**
- **What you need to do:**
  - **Ensure anonymity while maintaining accountability (e.g., prevent spam or fake reports).**
  - **Anonymous posts should still require image/video uploads and follow the same verification process.**
  - **Anonymous users cannot comment or upvote/downvote.**
- **Bonus: Allow anonymous users to claim ownership of their posts later by verifying their identity.**

4. AI-Powered Fake Report Detection

- **Description: Use AI to detect potentially fake or misleading crime reports.**
- **What you need to do:**
  - **Integrate an AI model (e.g., sentiment analysis, image recognition) to analyze the content of posts and comments.**

        - **Flag suspicious posts for admin review.**
        - **Provide a confidence score for the AI's assessment.**
- **Bonus: Allow users to see the AI's confidence score on flagged posts.**

5. Emergency Contact Integration

- **Description: Integrate emergency contact information (e.g., local police, hospitals) based on the division/district of the crime report.**
- **What you need to do:**
    - **Fetch and display relevant emergency contacts for the location of the crime.**
    - **Allow users to quickly call or message the contacts directly from the app.**
    - **Ensure the contact information is accurate and up-to-date.**
- **Bonus: Add a feature to report incorrect or outdated contact information.**

6. Image Compression for Optimized Storage and Performance

- **Description: Implement image compression to reduce the file size of uploaded images without significantly compromising quality. This will optimize storage usage and improve application performance (e.g., faster loading times).**
- **What you need to do:**
    - **Integrate an image compression library or tool (e.g., Sharp for Node.js, Pillow for Python, or ImageMagick).**
    - **Compress images during the upload process while maintaining a balance between quality and file size.**
    - **Ensure compressed images are stored in the database or cloud storage.**
    - **Handle edge cases (e.g., very large images, unsupported formats).**
- **Bonus:**
    - **Allow users to choose the compression level (e.g., low, medium, high).**
    - **Display the original and compressed file sizes for transparency.**

7. Watermarking for Content Protection

- **Description: Add watermarks to uploaded images to protect them from unauthorized use or redistribution.**
- **What you need to do:**
    - **Integrate a watermarking library or tool (e.g., Sharp for Node.js, Pillow for Python, or ImageMagick).**
    - **Automatically add a watermark (e.g., the platform's logo or a text-based watermark) to images during the upload process.**
    - **Ensure the watermark is subtle but effective (e.g., semi-transparent and placed diagonally across the image).**
    - **Handle edge cases (e.g., images with dark backgrounds where the watermark might not be visible).**
- **Bonus:**
    - **Allow users to customize the watermark (e.g., add their username or a custom text).**
    - **Provide an option to download the original, non-watermarked image for verified users or admins.**