User Authentication and Authorization

1. User Registration
    - `POST /api/register`
        - Request Body: `{ email, password, phone_number }`
        - Response: `{ user_id, status, message }`
2. User Login
    - `POST /api/login`
        - Request Body: `{ email, password }`
        - Response: `{ access_token, refresh_token, user_id, status, message }`
3. Change Password
    - `PUT /api/change-password`
        - Headers: `Authorization: Bearer <access_token>`
        - Request Body: `{ current_password, new_password }`
        - Response: `{ status, message }`
4. Password Recovery
    - `POST /api/forgot-password`
        - Request Body: `{ email_or_phone }`
        - Response: `{ status, message }`
5. Verify OTP
    - `POST /api/verify-otp`
        - Request Body: `{ otp, phone_number }`
        - Response: `{ status, message }`
6. Generate New Access Token
    - `POST /api/refresh-token`
        - Request Body: `{ refresh_token }`
        - Response: `{ access_token, status, message }`
7. Admin Ban User
    - `POST /api/admin/ban-user`
        - Headers: `Authorization: Bearer <admin_access_token>`
        - Request Body: `{ user_id }`
        - Response: `{ status, message }`

Crime Reporting

8. Report a Crime
    - `POST /api/report-crime`
        - Headers: `Authorization: Bearer <access_token>`

- Request Body: `{ title, description, division, district, crime_time, images, video }`
- Response: `{ crime_id, status, message }`

Flow for Generating AI Description

1. User Uploads Image : The user uploads an image as part of the crime report.
2. Image Upload to Storage : The image is stored in a cloud storage service (e.g., AWS S3, Firebase Storage).
3. Call AI Service : The backend makes an API call to the external AI service to generate a description of the image.
4. Store and Return Description : The generated description is stored in the database and returned to the user for review and modification.

Updated API Endpoints

1. Upload Image and Generate Description

```
Endpoint : POST /api/upload-image
Headers : Authorization: Bearer <access_token>
Request Body : { image_file }
Response : {
  "image_url": "https://example.com/path/to/image.jpg",
  "description": "A person standing near a broken window."
}
```

2. Report a Crime with Generated Description

```
Endpoint : POST /api/report-crime
Headers : Authorization: Bearer <access_token>
Request Body :{
  "title": "Broken Window Incident",
  "description": "A person standing near a broken window.", // This can be
modified by the user
  "division": "Dhaka",
  "district": "Dhaka South",
  "crime_time": "2023-10-01T12:00:00Z",
  "images": ["https://example.com/path/to/image.jpg"],
  "video": "https://example.com/path/to/video.mp4" // Optional
}
```

```
Response:{
  "crime_id": "12345",
  "status": "success",
  "message": "Crime report submitted successfully."
}
```

9. Get Crime Report Details
    - `GET /api/crime/:id`
        - Response: `{ crime_id, title, description, division, district, images, video, post_time, crime_time, user_id, upvotes, downvotes, verification_score, comments }`

Community Interaction

10. Upvote/Downvote Crime Post
    - `POST /api/crime/:id/upvote`
        - Headers: `Authorization: Bearer <access_token>`
        - Response: `{ status, message }`
    - `POST /api/crime/:id/downvote`
        - Headers: `Authorization: Bearer <access_token>`
        - Response: `{ status, message }`
11. Comment on Crime Post
    - `POST /api/crime/:id/comment`
        - Headers: `Authorization: Bearer <access_token>`
        - Request Body: `{ comment, proof_image, proof_video }`
        - Response: `{ comment_id, status, message }`

Crime Feed

12. Get Paginated Crime Posts
    - `GET /api/crimes?page=:page&limit=:limit`
        - Query Parameters: `division, district, sort_by, search_query`
        - Response: `{ crimes: [ { crime_id, title, description, division, district, images, video, post_time, crime_time, user_id, upvotes, downvotes, verification_score } ], total_pages, current_page }`

User Roles

13. Get All Users (Admin Only)
    - `GET /api/admin/users`
        - Headers: `Authorization: Bearer <admin_access_token>`
        - Response: `{ users: [ { user_id, email, phone_number, verified, banned } ] }`
14. Get All Comments (Admin Only)

- `GET /api/admin/comments`
    - Headers: `Authorization: Bearer <admin_access_token>`
    - Response: `{ comments: [ { comment_id, user_id, crime_id, comment, proof_image, proof_video } ] }`

User Profile

15. Get User Profile
    - `GET /api/profile/:user_id`
        - Response: `{ user_id, email, phone_number, verified, banned, profile_image, bio, crime_reports }`
16. Update User Profile
    - `PUT /api/profile`
        - Headers: `Authorization: Bearer <access_token>`
        - Request Body: `{ profile_image, bio, other_details }`
        - Response: `{ status, message }`

Additional Features

17. Get Heatmap Data
    - `GET /api/heatmap`
        - Query Parameters: `division, district, date_range`
        - Response: `{ heatmap_data: [ { location, count } ] }`
18. Get Leaderboard
    - `GET /api/leaderboard`
        - Query Parameters: `type (top_contributors, most_helpful_comments)`
        - Response: `{ leaderboard: [ { user_id, username, score } ] }`

Utility APIs

19. Get Divisions and Districts
    - `GET /api/divisions-districts`
        - Response: `{ divisions: [ { division_name, districts: [ { district_name } ] } ] }`
20. Send OTP
    - `POST /api/send-otp`
        - Request Body: `{ phone_number }`
        - Response: `{ status, message }`
21. Generate AI Description
    - `POST /api/generate-description`

- Request Body: `{ image_url }`
- Response: `{ description }`

Security and Miscellaneous

22. Health Check
    - `GET /api/health-check`
        - Response: `{ status: 'OK' }`
23. Logout
    - `POST /api/logout`
        - Headers: `Authorization: Bearer <access_token>`
        - Response: `{ status, message }`