Complete API List - Codeforces Investigation Project

1. External APIs (Codeforces)

User Information API

- **Endpoint**: `https://codeforces.com/api/user.info?handles=${handle}`
- **Method**: GET
- **Function**: `fetchUserInfo(handle)`
- **Purpose**: Retrieves basic user profile information including rating, max rating, and rank
- **Response**: User profile data with rating information

User Contest History API

- **Endpoint**: `https://codeforces.com/api/user.rating?handle=${handle}`
- **Method**: GET
- **Function**: `fetchUserContests(handle)`
- **Purpose**: Retrieves all contests a user has participated in and their performance history
- **Response**: Array of contest participation records with ratings and ranks

User Submissions API

- **Endpoint**: `https://codeforces.com/api/user.status?handle=${handle}&from=1&count=1000`
- **Method**: GET
- **Function**: `fetchUserSubmissions(handle)`
- **Purpose**: Retrieves up to 1000 most recent submissions made by the user
- **Response**: Array of submission records with problem details and verdicts

2. Internal REST APIs

Student Management APIs

Basic CRUD Operations

- **GET** `/api/students`
  - **Purpose**: Get all students with their Codeforces data
  - **Response**: Array of all student records
- **POST** `/api/students`

- **Purpose**: Create a new student
- **Request Body**: Student data including Codeforces handle
- **Response**: Created student record

- **GET** `/api/students/:id`
  - **Purpose**: Get a specific student by ID
  - **Parameters**: `id` - Student ID
  - **Response**: Single student record
- **PUT** `/api/students/:id`
  - **Purpose**: Update a student's information
  - **Parameters**: `id` - Student ID
  - **Request Body**: Updated student data
  - **Response**: Updated student record
- **DELETE** `/api/students/:id`
  - **Purpose**: Delete a student
  - **Parameters**: `id` - Student ID
  - **Response**: Deletion confirmation
- **PUT** `/api/students/:id/toggle-emails`
  - **Purpose**: Toggle email notifications for a student
  - **Parameters**: `id` - Student ID
  - **Response**: Updated notification settings

Codeforces Data APIs

Student Performance Data

- **GET** `/api/students/:id/codeforces`
  - **Purpose**: Get complete Codeforces profile data for a student
  - **Parameters**: `id` - Student ID
  - **Response**: Full CF profile including contests, submissions, and analytics
- **GET** `/api/students/:id/contests`
  - **Purpose**: Get contest history with optional date filtering
  - **Parameters**:
    - `id` - Student ID

    - `days` (optional) - Filter contests from last N days
  - **Response**: Array of contest participation records
- **GET** `/api/students/:id/problems`
  - **Purpose**: Get problem-solving statistics with optional date filtering
  - **Parameters**:
    - `id` - Student ID
    - `days` (optional) - Filter problems from last N days
  - **Response**: Problem-solving analytics and statistics

3. System Configuration APIs

Sync Configuration

- **Purpose**: Configure automatic data synchronization settings
- **Features**:
  - Set sync frequency (default: daily at 2 AM UTC)
  - Configure rate limiting delays
  - Set data refresh intervals (default: 6 hours)

4. API Implementation Details

Rate Limiting Strategy

- **Contest Data Delay**: 500ms between API calls
- **Student Processing Delay**: 2000ms (2 seconds) between students
- **Data Refresh Logic**: Only refresh if data is older than 6 hours

Error Handling

- Try/catch blocks for all external API calls
- Graceful degradation when Codeforces API fails
- Detailed error logging for debugging
- Continues processing other students if one fails

Caching Strategy

- MongoDB storage for persistent caching
- 6-hour cache validity period

- Forced refresh option available
- Reduces load on Codeforces API

Scheduled Operations

- **Cron Schedule**: `0 2 * * *` (Daily at 2 AM UTC)
- **Function**: `syncCodeforcesData()`
- **Purpose**: Automatic data synchronization for all students

5. Data Processing APIs

Analytics Generation

After fetching raw data, the system processes:

- Contest performance metrics
- Problem-solving statistics
- Rating progression analysis
- Submission heatmaps
- Difficulty distribution analysis

Special Algorithms

- **Unsolved Problems Calculator**: Custom algorithm to estimate unsolved problems (workaround for API limitations)
- **Unique Problems Counter**: Deduplicates solved problems across submissions
- **Rating Trend Analysis**: Calculates performance trends over time