1. Social Media API

- **Description**: Build a backend API for a social media platform where users can post updates, follow others, and like or comment on posts.
- **API Specifications**:
    - **POST /api/users/register**: Register a new user.
    - **POST /api/users/login**: Authenticate user with username and password.
    - **GET /api/users/**
      : Get user profile by ID (requires login).
    - **POST /api/posts**: Create a new post (requires login).
    - **GET /api/posts/**
      : Get a post by ID.
    - **PUT /api/posts/**
      : Update a post by ID (requires login).
    - **DELETE /api/posts/**
      : Delete a post by ID (requires login).
    - **POST /api/posts/**
      /like: Like a post (requires login).
    - **POST /api/posts/**
      /comment: Comment on a post (requires login).
    - **GET /api/feed**: Get the feed of posts from followed users (requires login).
    - **POST /api/users/**
      /follow: Follow a user (requires login).
- **Database Structure**:
    - Users collection: `{ username, password, email, followers, following }`
    - Posts collection: `{ userId, content, likes, comments }`
    - Comments collection: `{ postId, userId, content }`

2. E-commerce API

- **Description**: Develop a backend API for an e-commerce platform to handle products, shopping carts, orders, and payments.
- **API Specifications**:
    - **POST /api/users/register**: Register a new customer.
    - **POST /api/users/login**: Authenticate customer with username and password.
    - **GET /api/products**: List all products.
    - **GET /api/products/**
      : Get product details by ID.
    - **POST /api/cart**: Add a product to the cart (requires login).
    - **GET /api/cart**: View items in the cart (requires login).
    - **DELETE /api/cart/**
      : Remove a product from the cart (requires login).

    - **POST /api/orders**: Create a new order (requires login).
    - **GET /api/orders/**
      : View order details (requires login).
    - **POST /api/payments**: Process payment for an order (requires login).
- **Database Structure**:
    - Users collection: `{ username, password, email, address, orders }`
    - Products collection: `{ name, description, price, stock, category }`
    - Orders collection: `{ userId, items, totalAmount, paymentStatus }`
    - Cart collection: `{ userId, products }`

3. Library Management API

- **Description**: Create a backend API for a library system to manage books, members, and borrowing records.
- **API Specifications**:
    - **POST /api/members/register**: Register a new library member.
    - **POST /api/members/login**: Authenticate member with username and password.
    - **GET /api/books**: List all books.
    - **GET /api/books/**
      : Get details of a book by ID.
    - **POST /api/books**: Add a new book (Admin only, requires login).
    - **PUT /api/books/**
      : Update book details (Admin only, requires login).
    - **DELETE /api/books/**
      : Delete a book (Admin only, requires login).
    - **POST /api/borrow**: Borrow a book (requires login).
    - **POST /api/return**: Return a borrowed book (requires login).
    - **GET /api/borrow/records**: Get borrowing history for a member (requires login).
- **Database Structure**:
    - Members collection: `{ username, password, email, borrowedBooks }`
    - Books collection: `{ title, author, genre, availability }`
    - Borrowing records collection: `{ memberId, bookId, borrowDate, returnDate }`

4. Real-Time Chat API

- **Description**: Develop a real-time chat application backend API where users can send and receive messages.
- **API Specifications**:
    - **POST /api/users/register**: Register a new user.
    - **POST /api/users/login**: Authenticate user with username and password.
    - **GET /api/users/**
      : Get user profile by ID (requires login).

    - **POST /api/conversations**: Start a new conversation (requires login).
    - **GET /api/conversations/**
      : Get messages in a conversation by ID (requires login).
    - **POST /api/conversations/**
      /message: Send a message in a conversation (requires login).
    - **GET /api/conversations**: List all conversations for a user (requires login).
- **Database Structure**:
    - Users collection: `{ username, password, email }`
    - Conversations collection: `{ participants, messages }`
    - Messages collection: `{ conversationId, senderId, content, timestamp }`

5. Fitness Tracker API

- **Description**: Build a backend API for a fitness tracking application that allows users to log workouts, track progress, and set goals.
- **API Specifications**:
    - **POST /api/users/register**: Register a new user.
    - **POST /api/users/login**: Authenticate user with username and password.
    - **GET /api/workouts**: Get all workouts for a user (requires login).
    - **POST /api/workouts**: Log a new workout (requires login).
    - **GET /api/workouts/**
      : Get details of a workout by ID (requires login).
    - **PUT /api/workouts/**
      : Update a workout (requires login).
    - **DELETE /api/workouts/**
      : Delete a workout (requires login).
    - **GET /api/goals**: Get all fitness goals (requires login).
    - **POST /api/goals**: Set a new fitness goal (requires login).
    - **PUT /api/goals/**
      : Update a fitness goal (requires login).
    - **GET /api/progress**: Get progress towards goals (requires login).
- **Database Structure**:
    - Users collection: `{ username, password, email, goals }`
    - Workouts collection: `{ userId, type, duration, caloriesBurned, date }`
    - Goals collection: `{ userId, goalType, target, progress }`