Frontend Endpoints (GET Requests)

- Implement **3 separate GET endpoints** that serve pages with content:
  i. **Landing Page**: The main entry point of the application (e.g., `/` or `/home`).
  ii. **Form Page**: A page containing a form for data submission (e.g., `/form` or `/submit`).
  iii. **About Page**: A page with information about the application or organization (e.g., `/about`).

Backend Endpoints (CRUD Operations)

- Implement **at least 2 endpoints** for each of the following HTTP methods:
  - **POST**: Create new data entries.
  - **PUT**: Update existing data entries.
  - **DELETE**: Delete existing data entries.

Data Storage

- Use **at least 2 data files** storing JSON objects.
  - Each JSON file should contain objects with **at least one nested array and one nested object**.
    - Example: A user object that contains an array of messages or transactions.

Technical Implementation

- **Express.js**: Use Express.js for server-side routing and handling HTTP methods.
- **Fetch API with `async/await`**: Utilize `async/await` syntax on the frontend for asynchronous requests.
- **Form Validation**: Perform client-side form validation using **JavaScript regex** for inputs like emails, passwords, dates, etc.

Examples of How Requirements Might Be Met

Project 1: Airline Management System

Frontend Endpoints (GET Requests):

1. **Landing Page (`/` or `/home`):**
   - Displays a welcome message, login/register options, and a search form for flights.

2. **Flight Booking Form (`/book-flight`):**
    - Contains a form where users can select flights and enter passenger details.
3. **About Page (`/about`):**
    - Provides information about the airline, services offered, and contact details.

Backend Endpoints (CRUD Operations):

- **POST Endpoints:**
    i. **`/register`** : Allows new users to create an account.
        - **Data Received**:
          ```
          { "name": "John Doe", "email": "john@example.com", "password": "Pass123!" }
          ```
        - **Action**: Adds a new user to `users.json` .
    ii. **`/flights/book`** : Users can book a flight.
        - **Data Received**: `{ "userId": 1, "flightId": 101 }`
        - **Action**: Adds the flight to the user's bookedFlights array in `users.json` .
- **PUT Endpoints:**
    i. **`/user/update`** : Users can update their profile information.
        - **Data Received**: `{ "userId": 1, "newEmail": "newjohn@example.com" }`
        - **Action**: Updates the user's email in `users.json` .
    ii. **`/flights/update`** : Admins can update flight details.
        - **Data Received**: `{ "flightId": 101, "newTime": "15:00" }`
        - **Action**: Updates the flight's time in `flights.json` .
- **DELETE Endpoints:**
    i. **`/user/delete`** : Users can delete their account.
        - **Data Received**: `{ "userId": 1 }`
        - **Action**: Removes the user from `users.json` .
    ii. **`/flights/cancel-booking`** : Users can cancel a flight booking.
        - **Data Received**: `{ "userId": 1, "flightId": 101 }`
        - **Action**: Removes the flight from the user's bookedFlights array.

Data Storage:

- **`users.json`** :

```
{
  "users": [
    {
      "userId": 1,
      "name": "John Doe",
      "email": "john@example.com",
      "password": "hashed_password",
      "bookedFlights": [101, 102]
    }
  ]
}
```

- **Nested Array**: `bookedFlights` contains flight IDs the user has booked.

- **`flights.json`**:

```
{
  "flights": [
    {
      "flightId": 101,
      "origin": "New York",
      "destination": "London",
      "departureTime": "10:00",
      "passengers": [1, 2]
    }
  ]
}
```

- **Nested Array**: `passengers` contains user IDs who have booked the flight.

Project 2: Banking Website

Frontend Endpoints (GET Requests):

1. **Landing Page (`/`):**
   - Welcomes users and offers options to log in or register.
2. **Transaction Form (`/transaction`):**
   - Contains forms for depositing or withdrawing funds.
3. **About Page (`/about`):**
   - Provides information about the bank's history and services.

Backend Endpoints (CRUD Operations):

- **POST Endpoints:**
  i. **`/register`** : Users create a new bank account.
    - **Data Received**:
      ```
      { "name": "Alice", "email": "alice@example.com", "password": "SecurePass!" }
      ```
    - **Action**: Adds a new user to `users.json` .
  ii. **`/transactions/deposit`** : Users deposit money.
    - **Data Received**: `{ "userId": 1, "amount": 500 }`
    - **Action**: Increases the user's balance and adds a transaction record.
- **PUT Endpoints:**
  i. **`/user/update`** : Users update their personal details.
    - **Data Received**: `{ "userId": 1, "newAddress": "123 Main St" }`
    - **Action**: Updates user information in `users.json` .
  ii. **`/transactions/update`** : Update a scheduled transaction.
    - **Data Received**: `{ "transactionId": "tx1001", "newAmount": 300 }`
    - **Action**: Updates the transaction in `transactions.json` .
- **DELETE Endpoints:**
  i. **`/user/delete`** : Users close their account.
    - **Data Received**: `{ "userId": 1 }`
    - **Action**: Removes the user from `users.json` .
  ii. **`/transactions/delete`** : Users cancel a pending transaction.
    - **Data Received**: `{ "transactionId": "tx1001" }`
    - **Action**: Removes the transaction from `transactions.json` .

Data Storage:

- **`users.json`**:

```
{
  "users": [
    {
      "userId": 1,
      "name": "Alice",
      "email": "alice@example.com",
      "password": "hashed_password",
      "balance": 1000,
      "transactions": ["tx1001", "tx1002"]
    }
  ]
}
```

  - **Nested Array**: `transactions` contains transaction IDs associated with the user.

- **`transactions.json`:**

```
{
  "transactions": [
    {
      "transactionId": "tx1001",
      "userId": 1,
      "type": "deposit",
      "amount": 500,
      "date": "2023-10-05"
    }
  ]
}
```

  - **Nested Object**: Each transaction contains detailed information.

Project 3: Recipe Sharing Platform

Frontend Endpoints (GET Requests):

1. **Landing Page (`/`):**
   - Displays featured recipes and navigation options.
2. **Recipe Submission Form (`/submit-recipe`):**
   - Allows users to submit new recipes.
3. **About Page (`/about`):**

- Shares the platform's mission and contact info.

Backend Endpoints (CRUD Operations):

- **POST Endpoints:**
  i. **`/register`** : New users sign up.
    - **Data Received**:
      ```
      { "username": "chef101", "email": "chef@example.com", "password": "CookMaster!" }
      ```
    - **Action**: Adds user to `users.json` .
  ii. **`/recipes`** : Users submit recipes.
    - **Data Received**:
      ```
      { "userId": 1, "title": "Pancakes", "ingredients": [...], "instructions": [...] }
      ```
    - **Action**: Adds recipe to `recipes.json` .
- **PUT Endpoints:**
  i. **`/recipes/:id`** : Users update their recipes.
    - **Data Received**: `{ "recipeId": 101, "newTitle": "Fluffy Pancakes" }`
    - **Action**: Updates the recipe in `recipes.json` .
  ii. **`/user/update`** : Update user profile.
    - **Data Received**: `{ "userId": 1, "newEmail": "newchef@example.com" }`
    - **Action**: Updates user info in `users.json` .
- **DELETE Endpoints:**
  i. **`/recipes/:id`** : Users delete their recipes.
    - **Data Received**: `{ "recipeId": 101 }`
    - **Action**: Removes the recipe from `recipes.json` .
  ii. **`/user/delete`** : Users delete their account.
    - **Data Received**: `{ "userId": 1 }`
    - **Action**: Removes user from `users.json` .

Data Storage:

- **`users.json`** :

```
{
  "users": [
    {
      "userId": 1,
      "username": "chef101",
      "email": "chef@example.com",
      "password": "hashed_password",
      "myRecipes": [101, 102]
    }
  ]
}
```

- **Nested Array**: `myRecipes` holds recipe IDs created by the user.

- **`recipes.json`**:

```
{
  "recipes": [
    {
      "recipeId": 101,
      "authorId": 1,
      "title": "Pancakes",
      "ingredients": ["Flour", "Milk", "Eggs"],
      "instructions": ["Mix ingredients", "Cook on griddle"]
    }
  ]
}
```

- **Nested Arrays**: `ingredients` and `instructions` provide detailed recipe steps.

Project 4: Health & Mental Wellness Tracking App

Frontend Endpoints (GET Requests):

1. **Landing Page (`/`):**
   - Introduces the app and encourages users to sign up.
2. **Wellness Entry Form (`/log-entry`):**
   - Users can log exercises, meditation, and mood entries.
3. **About Page (`/about`):**
   - Explains the app's purpose and provides support info.

Backend Endpoints (CRUD Operations):

- **POST Endpoints:**
  i. **`/register`** : Users create accounts.
    - **Data Received**:
      `{ "name": "Mike", "email": "mike@example.com", "password": "Wellness123" }`
    - **Action**: Adds user to `users.json` .
  ii. **`/entries`** : Users submit wellness entries.
    - **Data Received**: `{ "userId": 1, "date": "2023-10-05", "activities": {...} }`
    - **Action**: Adds entry to `entries.json` .
- **PUT Endpoints:**
  i. **`/entries/:id`** : Users update their entries.
    - **Data Received**: `{ "entryId": 201, "newMood": "Relaxed" }`
    - **Action**: Updates the entry in `entries.json` .
  ii. **`/user/update`** : Update user profile.
    - **Data Received**: `{ "userId": 1, "newName": "Michael" }`
    - **Action**: Updates user info in `users.json` .
- **DELETE Endpoints:**
  i. **`/entries/:id`** : Users delete their entries.
    - **Data Received**: `{ "entryId": 201 }`
    - **Action**: Removes entry from `entries.json` .
  ii. **`/user/delete`** : Users delete their account.
    - **Data Received**: `{ "userId": 1 }`
    - **Action**: Removes user from `users.json` .

Data Storage:

- **`users.json`**:

```
{
  "users": [
    {
      "userId": 1,
      "name": "Mike",
      "email": "mike@example.com",
      "password": "hashed_password",
      "entries": [201, 202]
    }
  ]
}
```

- **Nested Array**: `entries` holds IDs of the user's wellness entries.

- **`entries.json`:**

```
{
  "entries": [
    {
      "entryId": 201,
      "userId": 1,
      "date": "2023-10-05",
      "activities": {
        "exercise": { "type": "Yoga", "duration": 30 },
        "meditation": { "duration": 15 },
        "mood": "Calm"
      }
    }
  ]
}
```

  - **Nested Object**: `activities` contains detailed information about the day's activities.

Implementing Technical Requirements

Form Validation with JavaScript Regex

- **Email Validation:**
  - Use regex to ensure the email input matches a standard email format.
  - Example:

```
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
if (!emailRegex.test(emailInput.value)) {
  // Show error message
}
```

- **Password Validation:**
  - Ensure passwords meet complexity requirements (e.g., at least 8 characters, one uppercase letter, one number).
- **Date Validation:**
  - Validate date inputs to ensure they are in the correct format (e.g., YYYY-MM-DD).