Basic API Application

This is a simple Spring Boot 3 REST API for managing users with data stored in an embedded H2 database. The application is built with Jakarta EE specifications and organized into feature-based packages for easy navigation and maintenance.

Objective
The primary objective of this application is to provide REST endpoints to manage users. Users have personal information and an address, and can be searched by their accountId or name. This API can:

- Create new user records
- Retrieve all user records
- Search for users by name or account ID

The design aims for simplicity and modularity by following a feature-based package structure.

Prerequisites

- Java 17
- Maven

How to Run

1. **Extract ZIP File **AFCU Take Home Test** **
   Part1 Backend
   basic-api
   cd basic-api

2. Build the Project: mvn clean install
3. Start the Application: mvn spring-boot:run
4. Access the API: The API will be available at `http://localhost:8080/api/user`

Accessing the H2 Database Console

To view the in-memory database, go to `http://localhost:8080/h2-console` and use:
- **JDBC URL**: `jdbc:h2:mem:testdb`
- **Username**: `sa`
- **Password**: *(leave this blank)*

API Endpoints

1. **Create a User**
- URL: `POST /api/user/`
- Request Body
(example):

```
{
 "accountId": "A123456789",
```

```
  "firstName": "John",
  "lastName": "Doe",
  "line1": "123 Main St",
  "city": "Anytown",
  "state": "CA",
  "country": "USA",
  "zip": "90210"
}
```

Responses:

- 201 Created: User created successfully.
- 400 Bad Request: If validation fails (e.g., `accountId` is already taken).

2. Get All Users

- URL: `GET /api/user/`
- Response: Returns a list of all users.

3. Search Users

- URL: `GET /api/user/search`
- Query Parameters:
  - `query`: The search term (e.g., name or account ID).
  - `searchBy`: Either `"accountId"` or `"name"`.
- **Responses:**
  - 200 OKList of users that match the search.
  - 400 Bad Request: If `query` is too short or invalid.

Error Handling

- Validation Errors: Detailed messages are returned if required fields are missing or have invalid values.
- Custom Exceptions: The API gracefully handles specific errors like duplicate `accountId`s.

Running Tests: mvn test

Tests cover creating users, validation checks, and search functionality.

Design Choices

- Account ID Uniqueness: Each user has a unique `accountId` (e.g., `A123456789`).
- Feature-Based Structure: The project is organized by features (entities, user management, utilities).
- In-Memory Database: Uses H2 database, which resets on each run, making it ideal for quick testing.