2025

CAMPUS API

Education

API for Managing Courses & Faculty using FastAPI

Name :Sanjai B

Date: 19.07.2025

CAMPUS API EDUCTION

Table Of Contents

01 part

Overview & Features

02 part

Project Structure

03 part

Data Models

04 part

Core API Endpoints (Courses & Faculty)

05 part

Advanced Analytics

06 part

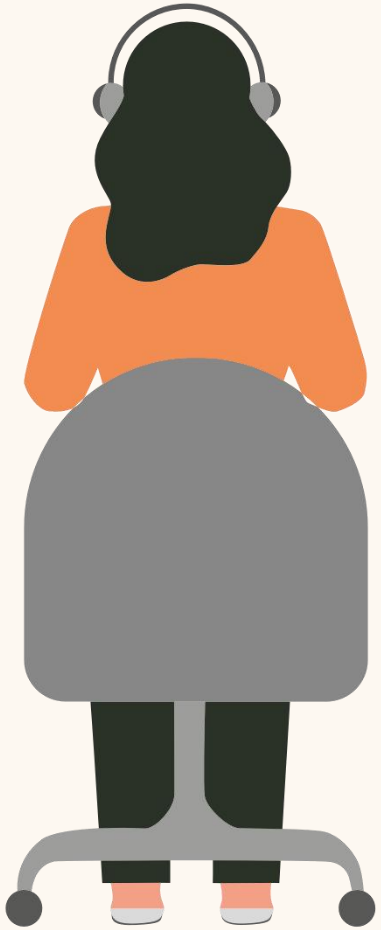
Setup Instructions

07 part

Live Demo Use Cases

08 part

Key Takeaways & Future Scope



01

What is the Campus API?

What is Campus API

1. A RESTful API built with FastAPI and Pydantic
2. Manages data for:
   - -Courses
   - -Faculty
3. CRUD operations, advanced queries, fuzzy search
4. Stores data in JSON files for testing/mock storage and Auto-generated Swagger docs at /docs

02

Project Files & Structure

main.py:
Initializes the FastAPI app and registers all route modules (courses, faculty, advanced). It also serves a simple homepage with links to Swagger/docs.

database.py:
Handles loading and saving of mock JSON data (courses.json, faculty.json) and provides utility functions to fetch individual records from memory.

models.py:
Defines the Pydantic data models (Course, Faculty) used for request validation and response serialization across the entire API.

routers/courses.py:
Implements all course-related API operations including CRUD, search, filtering, recommendations, specialization matching, and advanced analytics (e.g. most prerequisites).

routers/faculty.py:
Implements faculty-related API operations including CRUD, fuzzy search, availability, contact directory, section-wise filtering, and advanced faculty analytics.

routers/advanced.py:
Contains cross-domain analytic features and advanced filters such as department course counts, global search, and credit-based filtering.

mock_data/courses.json:
Stores mock course records including course details, prerequisites, and lead faculty IDs; used to simulate a working database.

mock_data/faculty.json:
Stores mock faculty information including departments, specialization, availability, and linked courses; used for testing and response data.

mock_data_generator.py:
Automatically generates realistic mock data for both courses.json and faculty.json using the faker package; saves time when testing the full API.

03

Data Models

Course & Faculty Models

COURSE AND FACULTY MODELS

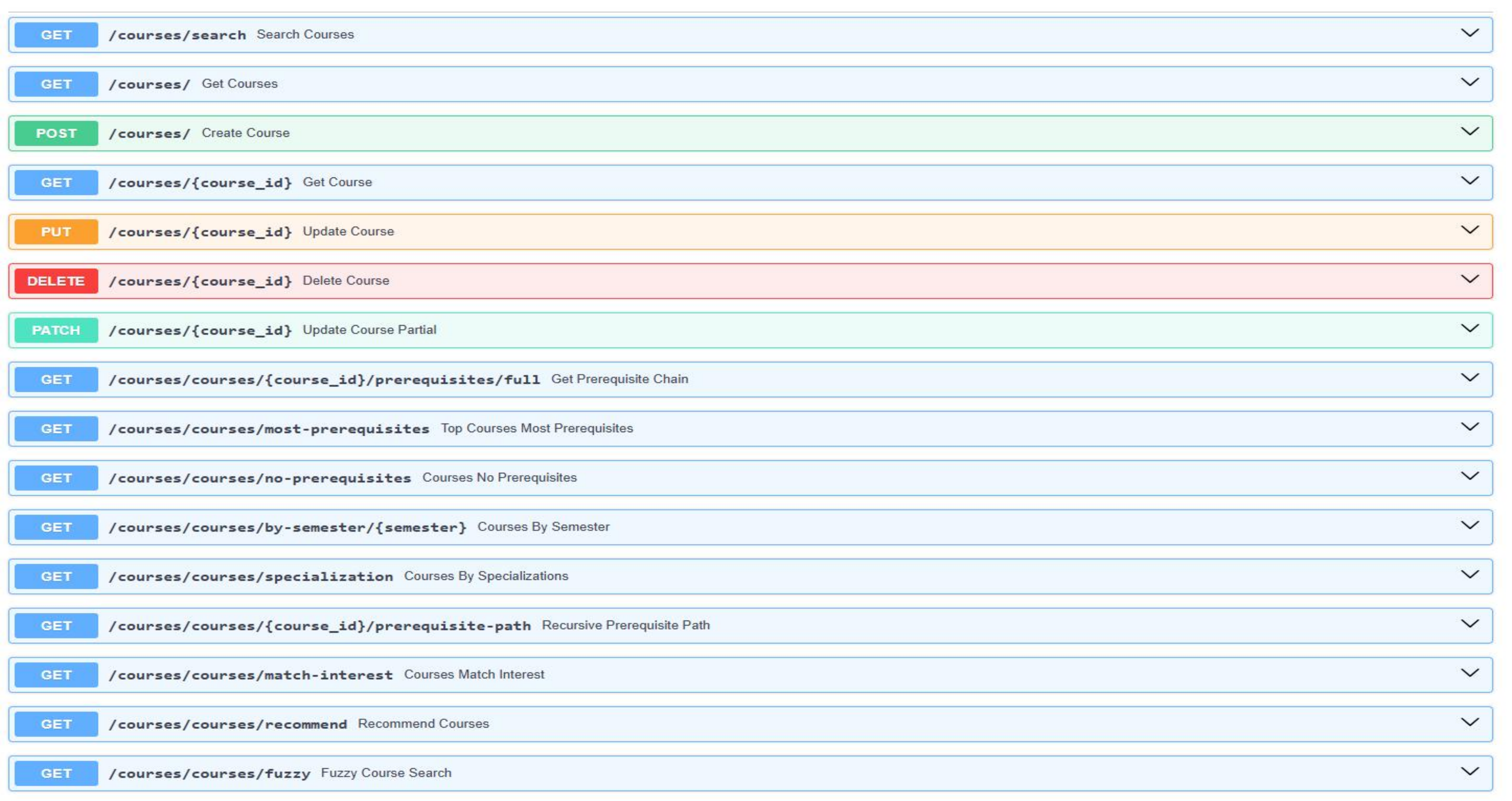
```
class Course(BaseModel):
    id: int
    name: str
    code: str
    description: str
    credits: int
    department: str
    prerequisites: List[int]
    faculty_id: int
```

```
class Faculty(BaseModel):
    id: int
    first_name: str
    last_name: str
    email: str
    department: str
    specialization: List[str]
    office_location: str
    office_hours: str
```

04

Core API Endpoints (Courses & Faculty)

COURSES



GET `/courses/search` Search Courses

GET `/courses/` Get Courses

POST `/courses/` Create Course

GET `/courses/{course_id}` Get Course

PUT `/courses/{course_id}` Update Course

DELETE `/courses/{course_id}` Delete Course

PATCH `/courses/{course_id}` Update Course Partial

GET `/courses/courses/{course_id}/prerequisites/full` Get Prerequisite Chain

GET `/courses/courses/most-prerequisites` Top Courses Most Prerequisites

GET `/courses/courses/no-prerequisites` Courses No Prerequisites

GET `/courses/courses/by-semester/{semester}` Courses By Semester

GET `/courses/courses/specialization` Courses By Specializations

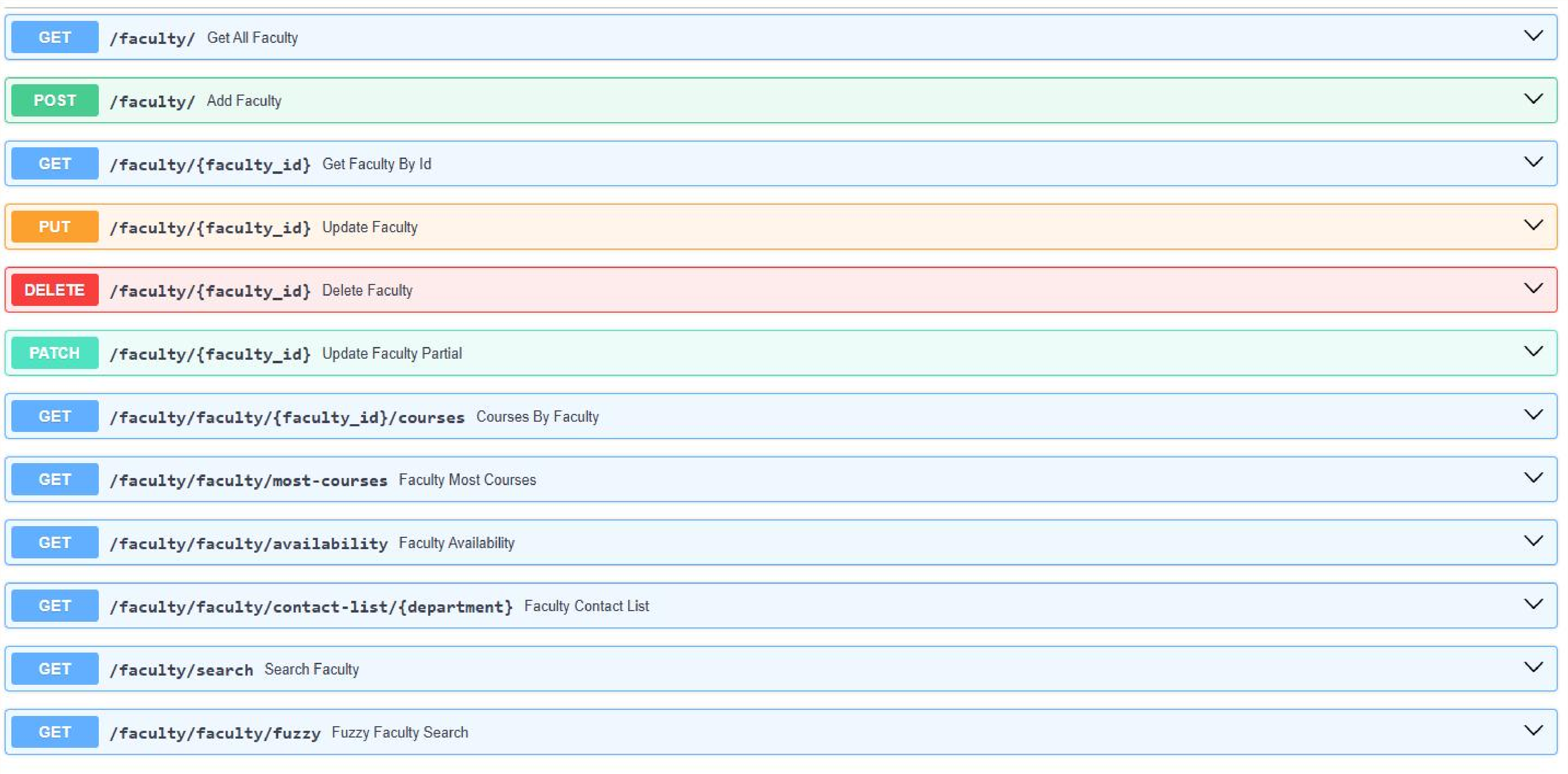
GET `/courses/courses/{course_id}/prerequisite-path` Recursive Prerequisite Path

GET `/courses/courses/match-interest` Courses Match Interest

GET `/courses/courses/recommend` Recommend Courses

GET `/courses/courses/fuzzy` Fuzzy Course Search

FACULTY



GET `/faculty/` Get All Faculty

POST `/faculty/` Add Faculty

GET `/faculty/{faculty_id}` Get Faculty By Id

PUT `/faculty/{faculty_id}` Update Faculty

DELETE `/faculty/{faculty_id}` Delete Faculty

PATCH `/faculty/{faculty_id}` Update Faculty Partial

GET `/faculty/faculty/{faculty_id}/courses` Courses By Faculty

GET `/faculty/faculty/most-courses` Faculty Most Courses

GET `/faculty/faculty/availability` Faculty Availability

GET `/faculty/faculty/contact-list/{department}` Faculty Contact List

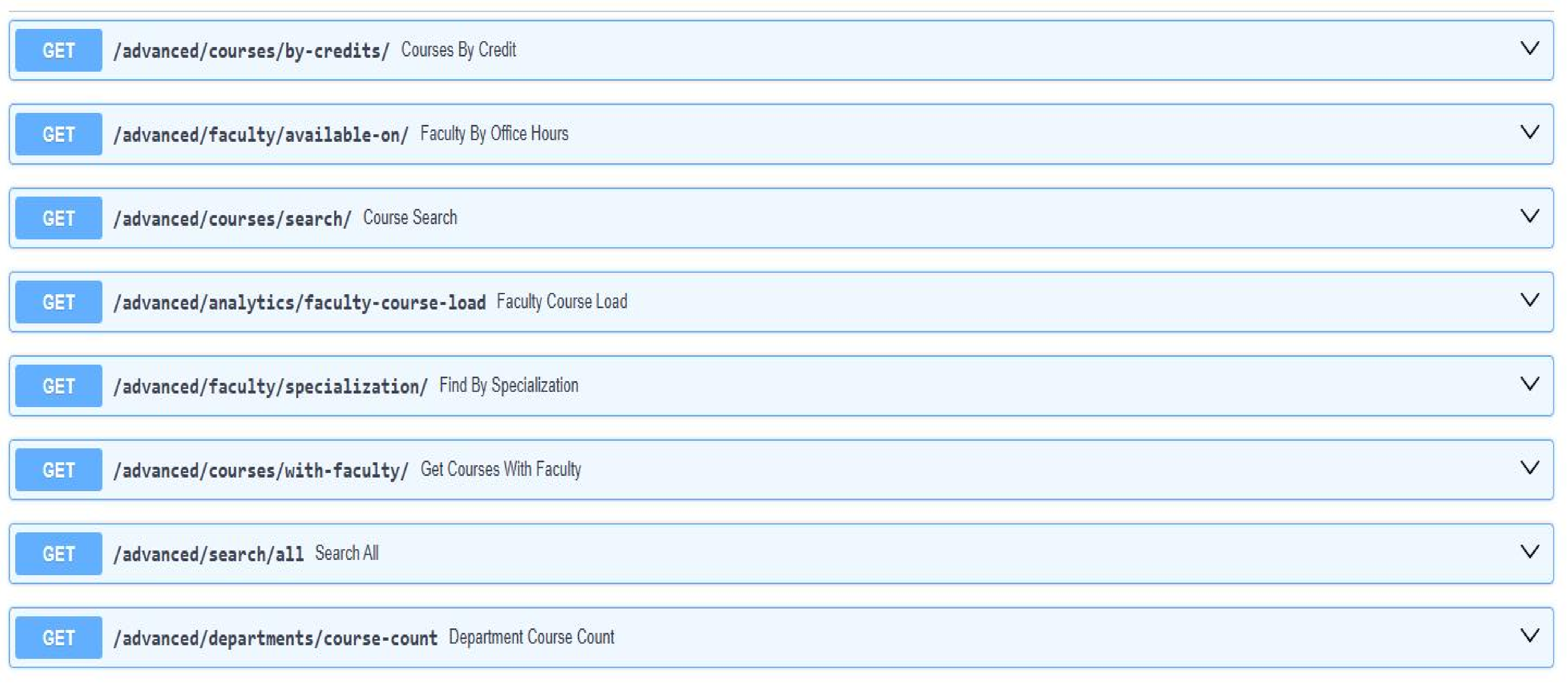
GET `/faculty/search` Search Faculty

GET `/faculty/faculty/fuzzy` Fuzzy Faculty Search

05

Advanced Analytics Endpoints

ADVANCED STATISTICS & FILTERS



GET /advanced/courses/by-credits/ Courses By Credit

GET /advanced/faculty/available-on/ Faculty By Office Hours

GET /advanced/courses/search/ Course Search

GET /advanced/analytics/faculty-course-load Faculty Course Load

GET /advanced/faculty/specialization/ Find By Specialization

GET /advanced/courses/with-faculty/ Get Courses With Faculty

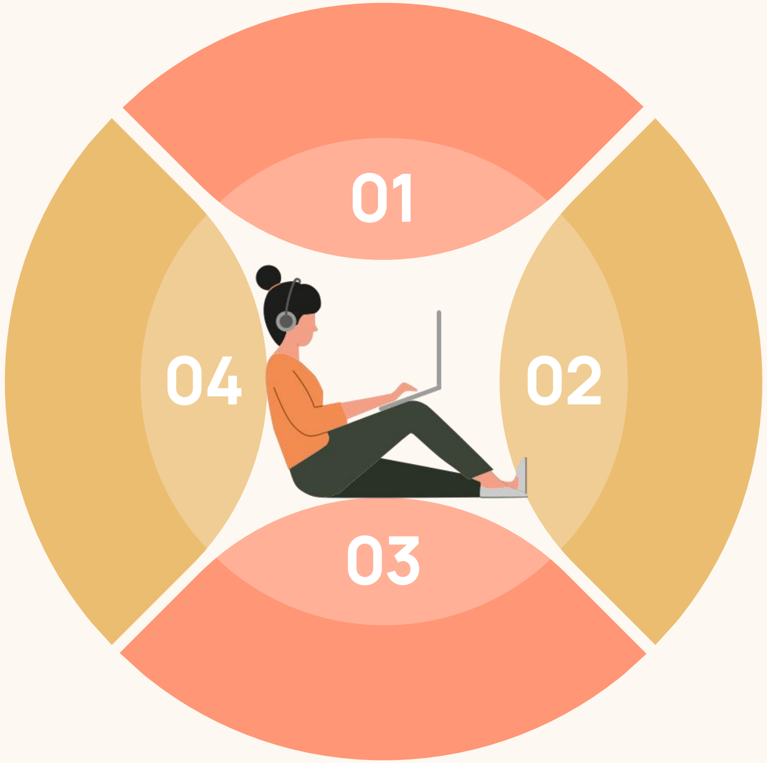
GET /advanced/search/all Search All

GET /advanced/departments/course-count Department Course Count

06

Setup instructions

Steps



Install dependencies:

```
pip install -r requirements.txt
```

Go to folder with main.py

```
cd app
```

Run Server

```
uvicorn main:app --reload
```

Explore Docs or Redoc:

http://localhost:8000/docs

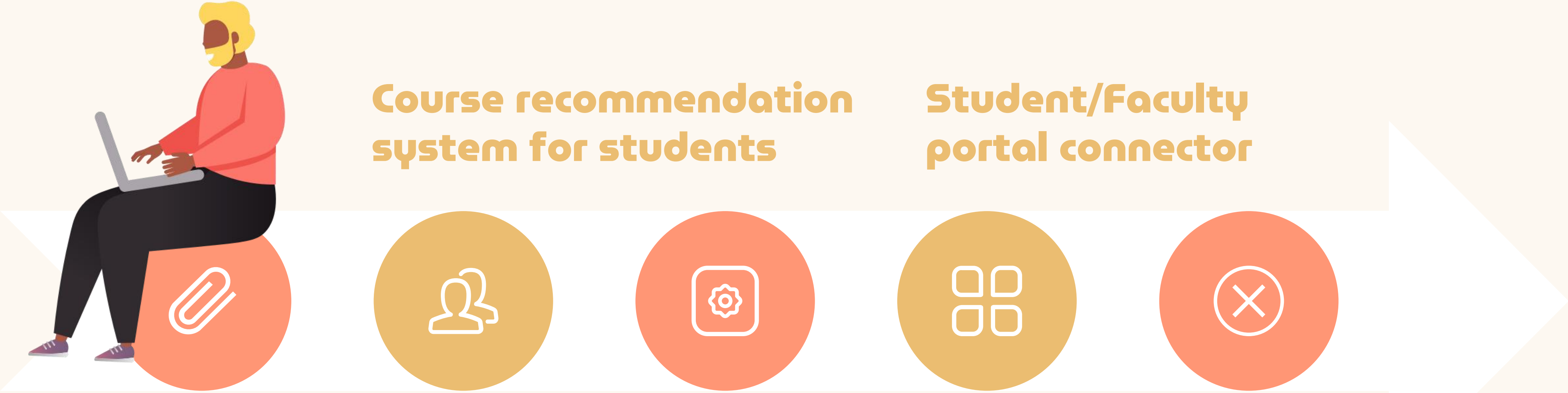
or

http://localhost:8000/redoc

07

Use Cases

Use Cases



Course recommendation system for students

Student/Faculty portal connector

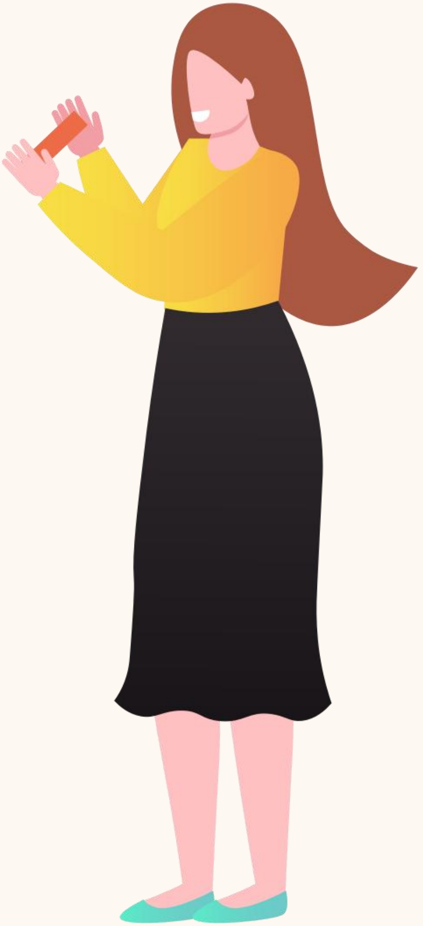
Real-time search assistant for courses & faculty

Integrated into a College Content Management System

08

Key Takeaways and Future scope

Validation & 422 Errors



1 **Type Enforcement:**

FastAPI automatically checks if inputs match the expected data types using Pydantic and Python type hints.

2 **422 Error Response:**

Invalid or missing input returns a 422 Unprocessable Entity with details explaining the validation error.

3 **Applies to All Inputs:**

Validation applies to query parameters, path variables, and request bodies to ensure clean and correct data.

Key Takeaways

01

FastAPI enables rapid development of robust, type-safe REST APIs with built-in documentation.

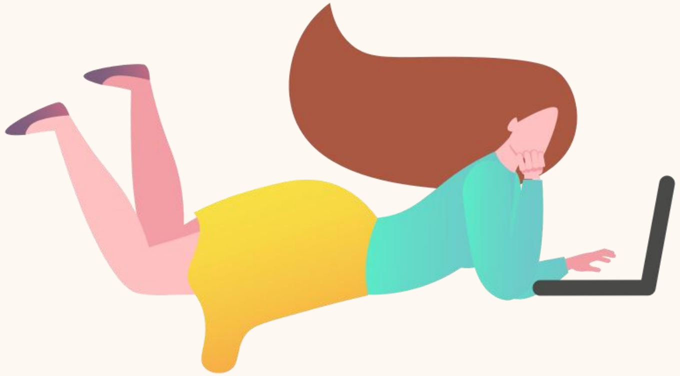
02

The College Campus API supports full CRUD operations with advanced search and analytics.

03

Title here

REST APIs provide a standardized way for different applications to communicate and exchange data reliably over the web, enabling integration between diverse systems and technologies.



2025

THANK YOU Research

Education

It serves a variety of purposes, making presentations powerful tools for convincing and teaching.

Speaker name :Sanjai B

19.07.25

CAMPUS API EDUCATION