**Python**

**Task 1**

* User Authentication with JWT:
  + Develop a user authentication system using JWT (JSON Web Tokens) in Python.
  + Create endpoints for user registration, login, and token refresh using frameworks like Flask or Django.
  + Use libraries like PyJWT for token generation and verification.
  + Secure API endpoints using JWT-based authentication middleware.
* Simple CRUD API:
  + Implement a simple CRUD (Create, Read, Update, Delete) API for managing a resource (e.g., users, products, tasks) in Python.
  + Utilize frameworks like Flask or Django for building RESTful APIs.
  + Define routes, views, and models for handling CRUD operations.
  + Integrate with a database (e.g., SQLite, PostgreSQL) for data storage.
* File Upload and Download API:
  + Create an API for uploading and downloading files in Python.
  + Implement endpoints for uploading files and generating download links using frameworks like Flask or Django.
  + Use libraries like Flask-Uploads or Django file handling for handling file uploads.
  + Ensure security by validating file types, limiting file size, and securing download links.

**Task 2**

Create a Python program that acts as a simple arithmetic operation API protected by authentication. The program should allow clients to perform basic arithmetic operations (addition, subtraction, multiplication, division) on two numbers after authenticating themselves. The API should use JWT (JSON Web Tokens) for authentication.

* Implement JWT-based authentication
* Implement API endpoints for arithmetic operations
* Each API endpoint should accept two numbers as input parameters and return the result of the corresponding arithmetic operation.
* Use the Flask or Djnago

**Node.js**

**Task 1**

Secure Arithmetic Operation API with Authentication

Create a Node.js application that acts as a simple arithmetic operation API protected by authentication. The program should allow clients to perform basic arithmetic operations (addition, subtraction, multiplication, division) on two numbers after authenticating themselves. The API should use JWT (JSON Web Tokens) for authentication.

* Implement JWT-based authentication
  + Generate a JWT token upon successful authentication.
  + Protect the API endpoints using the JWT token.
* Implement API endpoints for arithmetic operations:
  + Addition: /add
  + Subtraction: /subtract
  + Multiplication: /multiply
  + Division: /divide
* Each API endpoint should accept two numbers as input parameters and return the result of the corresponding arithmetic operation.
* Use Express.js framework for creating the API and jsonwebtoken (jwt) module for JWT authentication.

### 

**Task 2**

* User Authentication with JWT:
  + Develop a user authentication system using JWT (JSON Web Tokens) in Node.js.
  + Create endpoints for user registration, login, and token refresh using frameworks like Express.js.
  + Use libraries like jsonwebtoken for token generation and verification.
  + Secure API endpoints using JWT-based authentication middleware.
* Simple CRUD API:
  + Implement a simple CRUD (Create, Read, Update, Delete) API for managing a resource (e.g., users, products, tasks) in Node.js.
  + Utilize frameworks like Express.js for building RESTful APIs.
  + Define routes, controllers, and models for handling CRUD operations.
  + Integrate with a database
* File Upload and Download API:
  + Create an API for uploading and downloading files in Node.js.
  + Implement endpoints for uploading files and generating download links using frameworks like Express.js or Koa.js.
  + Use libraries for handling file uploads.
  + Ensure security by validating file types, limiting file size, and securing download links.