**Stage Two Task**You are to build a simple REST API capable of CRUD operations on a "person" resource, interfacing with any database of your choice. Your API should dynamically handle parameters, such as adding or retrieving a person by name. Accompany the development with UML diagrams to represent your system's design and database structure. Create an automated testing script that verifies each of your API's operations. Host your entire project on GitHub, and provide a well-structured documentation in the repository that outlines request/response formats, setup instructions, and sample API usage.**Task Breakdown: Develop a REST API with Basic CRUD Operation**  
**Objective:** Build a simple REST API capable of CRUD operations on a resource, say, a "person". The chosen programming language should interface with any chosen database of your choice.**1. REST API Development:**  
Develop an API with endpoints for:

* **CREATE**: Adding a new person.
* **READ**: Fetching details of a person.
* **UPDATE**: Modifying details of an existing person
* **DELETE**: Removing a person.
* Ensure all interactions with the database are secure and free from common vulnerabilities (e.g., SQL injections).

**2. Database Modelling: (Bonus)**

* UML Diagram: Design and present a UML (Unified Modeling Language) diagram that represents the structure and relationships of your API's classes and models.

**3. Testing:**

* Develop a script (e.g., using tools like Postman or scripts written in Python using the requests library) that automates the testing of each CRUD operation in your API.
* This  should:
* Add a new person (e.g., "Mark Essien").
* Fetch details of a person
* Modify the details of an existing person.
* Remove a person

**4. Dynamic Parameter Handling:**

* Your API should be flexible enough to handle dynamic input. If we provide a name (or other details), your backend should be able to process operations using that name.
* Example: If we pass "Mark Essien", we should be able to perform all CRUD operations on "Mark Essien".
* **Add validation – field** should only be strings; integers or any other data type should not be allowed.

**5. GitHub Repository:**

* Create a GitHub repository for this project.
* Ensure the repository contains:
* A detailed README.md file explaining how to set up, run, and use the API.
* The source code for the API.
* UML diagrams (or links to view them).

**6. Documentation:**

* Provide a documentation file (e.g., DOCUMENTATION.md in your GitHub repo) that outlines:
* Standard formats for requests and responses for each endpoint.
* Sample usage of the API, including example requests and expected responses.
* Any known limitations or assumptions made during development.
* Instructions for setting up and deploying the API locally or on a server.

7. **Hosting**

* Using the same Server used in the Stage One task (or another server, if possible), modify it accordingly to  host your endpoint with a URL like this <https://theirdomain.com/api>
* Test extensively with various testing tools to make sure it is accessible before submitting

**Acceptance Criteria**

* **Functional REST API:**The API should successfully perform all CRUD operations.
* **Modelling Diagrams:**Clear and accurate UML and E-R diagrams should be provided.
* **Effective Testing Script:**The script should successfully test all the CRUD operations without manual intervention.
* **Dynamic Parameter Handling:** The API should correctly handle and respond to different parameters provided.
* **GitHub Repository:**Repository should be well-organized, contain all necessary files, and be publicly accessible.
* **Detailed Documentation:** Documentation should provide clear guidance on how to use the API, including setup, request/response formats, and sample usage.

**Submission Mode**  
To submit your completed task, please fill out the provided [Google Form](https://forms.gle/GvbUaQCyMxhP68jo9). In the form, include the GitHub repository link containing your source code, UML and E-R diagrams, testing script,  endpoint URL and detailed documentation. Ensure your repository is publicly accessible and that your README.md or DOCUMENTATION.md provides clear instructions for setting up and using your API. Submissions that do not adhere to these guidelines or lack any of the required components will not be considered complete. We recommend double-checking all your deliverables and testing your repository's accessibility before submitting.  
**Submission Deadline:**

* The deadline for submissions is 15th September 2023, 11:59 PM GMT. Late submissions will not be entertained.
* If you encounter any issues or have questions regarding the submission process, please contact the Backend Mentors

**Backend Stage Two Submission**  
<https://forms.gle/KKUV3pf5GTd2xMXe6> (edited)