

Take home question:

Build a RESTful API that manages users and posts. Below are some tasks that may guide your solution.

Possible Tasks:

- Using any programming language and framework of your choice, create a simple RESTful API that exposes two endpoints: /users and /posts.
- The /users endpoint should support the following operations:
 - GET /users: Return a list of all users in JSON format, with each user having an id, a name, and an email attribute.
 - POST /users: Create a new user with the given name and email in the request body, and return the created user in JSON format, with a unique id assigned by the server.
 - GET /users/{id}: Return the user with the given id in JSON format, or a 404 error if the user does not exist.
 - PUT /users/{id}: Update the user with the given id with the new name and email in the request body, and return the updated user in JSON format, or a 404 error if the user does not exist.
 - DELETE /users/{id}: Delete the user with the given id, and return a 204 status code, or a 404 error if the user does not exist.
- The /posts endpoint should support the following operations:
 - GET /posts: Return a list of all posts in JSON format, with each post having an id, a title, a content, and a user_id attribute, which references the id of the user who created the post.
 - POST /posts: Create a new post with the given title, content, and user_id in the request body, and return the created post in JSON format, with a unique id assigned by the server. If the user_id does not match any existing user, return a 400 error.
 - GET /posts/{id}: Return the post with the given id in JSON format, or a 404 error if the post does not exist.
 - PUT /posts/{id}: Update the post with the given id with the new title, content, and user_id in the request body, and return the updated post in JSON format, or a 404 error if the post does not exist. If the user_id does not match any existing user, return a 400 error.
 - DELETE /posts/{id}: Delete the post with the given id, and return a 204 status code, or a 404 error if the post does not exist.
- You can use any data storage method of your choice, such as an in-memory database, a file system, or a relational or non-relational database.
- You should provide clear and concise documentation for your API, including the expected request and response formats, the possible error codes and messages, and any assumptions or limitations you made.

- You should also write unit tests and integration tests for your API, using any testing framework of your choice, and provide instructions on how to run them.

Bonus (optional):

- Choose a cloud platform of your choice, such as AWS, Azure, Google Cloud, or Heroku, and deploy your API to it, using any deployment method of your choice, such as Docker, Kubernetes, or serverless.
- You should provide a public URL for your API, and ensure that it is accessible and functional.
- You should also provide a brief explanation of your deployment process, including the steps you took, the tools you used, and the challenges you faced or solved.
- Include a README file explaining how to set up and run the API, and test everything