Summary

For the Giftogram Developer Test, you will be required to develop a simple API Backend that is designed for use with a mobile chat application. You will be writing backend scripts that interact with a MySQL database, as well as architecting the MySQL database's structure based on the App's requirements.

Requirements

Imagine that you have been asked to create a RESTful backend API for a simple mobile chat application. In this chat application, users can create an account, login, and send individual messages to other users in the app. Below you will see a list of required API Endpoints that you must implement. Besides these API Endpoints, you are free to structure the backend scripts and MySQL database any way you like. All API endpoints will return valid JSON responses.

You can use any Javascript/PHP framework you would like. You can also not use any framework and hand-code everything.

Error Response

Some API Endpoints will require you to return an error message or response back to the requester. The only requirements for an error response is that it must have an error_code, error_title, and error_message. You may decide the details of exactly what data goes inside each error response.

```
{
  "error_code": 101,
  "error_title": "Login Failure",
  "error_message": "Email or Password was Invalid!"
}
```

API Endpoints

Endpoint Name	Request Type	Description	Example Request Body	Example Response
/register	POST	Handles registering a brand new user.	**{ “email”:” info@giftogram.com ”, “password”:”Test123”, “first_name”:”John”, “last_name”:”Doe” }**	**{ "user_id": 1, "email": "info@giftogram.com", "first_name": "John", "last_name": "Doe" }**
/login	POST	Handles a login request from the user. If the login was unsuccessful, you must send an appropriate error response.	**{ “email”:”info@giftogram.com”, “password”:”Test123” }**	**{ "user_id": 1, "email": "info@giftogram.com", "first_name": "John", "last_name": "Doe" }**

/view_messages	GET	Returns all messages that these two users have sent to each other in date order.	**{
			"user_id_a": "1",
			"user_id_b": "2"
			}**
				**{
				"messages": [
				{
				"message_id": 1,
				"sender_user_id": 1,
				"message": "Hey what is up?",
				"epoch": 1429220026
				},
				{
				"message_id": 2,
				"sender_user_id": 2,
				"message": "Not much, how are you"
				}
				]
				}**

/send_message	POST	Sends a message from one user to another. Returns a success code if the message was sent successfully.	**{ "sender_user_id": 1, "receiver_user_id": 2, "message": "Example text" }**	**{ "success_code": "200", "success_title": "Message Sent", "success_message": "Message was sent successfully" }**
/list_all_users	GET	Displays all of the users that have registered to use the	**{ "requester_user_id": 3 }**	**{ "users": [{**

		app excluding the requester.		**"user_id": 1,
				"email": "ppeck@giftogram.com",
				"first_name": "Preston",
				"last_name": "Peck"
				},
				{
				"user_id": 2,
				"email": "jgreen@giftogram.com",
				"first_name": "Jake ",
				"last_name": "Green"
				}
				]
				}**

Deliverables

- Preferably, a link to a public github.com (or other) repository containing all the code. If you are using a framework, do an initial commit with the base framework installation followed by your code changes in subsequent commits. If this is not convenient then you can send us a ZIP file with all the code instead.

- A SQL dump of your MySQL database so we can see how you structured your tables. (Or, you can include the sql dump file in your git repository. This is often not a best practice but it's fine for this demo.)

- The repository (or zip file) should contain a README.md file that contains:
    - Your Name
    - How long it took you to complete the test
    - A list of general steps you took from start to finish

- Let us know of any possible issues that you see with how these endpoints are structured. What changes would you make regarding security and/or general usability?