Below is an example of a README.txt file for your assignment, providing a comprehensive overview of the project and instructions for setting up and testing the database models and CRUD operations.

====================================================

README.txt - IELTS Speaking Test Platform Assignment

====================================================

Project Overview:

-----------------

This project involves developing SQLAlchemy models for the User, SpeakingTest, and ListeningTest entities in the backend, creating a database schema that supports basic CRUD functionality for the IELTS Speaking Test platform.

The project includes:

1. Setting up SQLAlchemy for database interaction.

2. Defining models for User, SpeakingTest, and ListeningTest with appropriate relationships.

3. Implementing CRUD operations for these models.

4. Testing the CRUD functionality using Flask routes and providing evidence such as query results and API responses.

Files Included:

---------------

1. config.py - Configuration file for database connection.

2. models.py - Contains SQLAlchemy model definitions and CRUD functions.

3. app.py - Flask application with routes for testing CRUD operations.

4. README.txt - This readme file with setup and usage instructions.

5. test\_data.sql - SQL script to seed sample data in the database.

6. Documentation - Folder containing documentation of test scenarios, query results, and API responses.

Setup Instructions:

-------------------

1. Install necessary packages:

Ensure you have the required packages installed. You can install them via pip:

pip install SQLAlchemy Flask

2. Configure the database connection:

Edit the `config.py` file and set the DATABASE\_URI to match your database setup. For example, to use SQLite:

DATABASE\_URI = 'sqlite:///test.db'

3. Set up the database schema:

Run the `models.py` script to create the necessary tables in the database:

python models.py

4. (Optional) Seed the database with sample data:

If you have provided a `test\_data.sql` file, run it to insert sample data into your test database:

sqlite3 test.db < test\_data.sql

Running the Flask Application:

------------------------------

1. Start the Flask application to test the CRUD operations via API:

python app.py

2. Use Postman, curl, or any other tool to interact with the API endpoints.

API Endpoints:

--------------

1. User Endpoints:

- Create User: POST /users

- Get User by ID: GET /users/<user\_id>

- Update User: PUT /users/<user\_id>

- Delete User: DELETE /users/<user\_id>

2. SpeakingTest Endpoints:

- Create Speaking Test: POST /speaking\_tests

- Get Speaking Test by ID: GET /speaking\_tests/<test\_id>

- Update Speaking Test: PUT /speaking\_tests/<test\_id>

- Delete Speaking Test: DELETE /speaking\_tests/<test\_id>

3. ListeningTest Endpoints:

- Create Listening Test: POST /listening\_tests

- Get Listening Test by ID: GET /listening\_tests/<test\_id>

- Update Listening Test: PUT /listening\_tests/<test\_id>

- Delete Listening Test: DELETE /listening\_tests/<test\_id>

Example Usage:

--------------

To add a new user:

curl -X POST <http://127.0.0.1:5000/users> -H "Content-Type: application/json" -d '{"name": "John Doe", "email": "[john@example.com](mailto:john@example.com)", "password": "securepassword123"}'

To fetch a user by ID:

curl <http://127.0.0.1:5000/users/1>

Documentation:

--------------

The `Documentation` folder contains the following:

1. \*\*Test Scenarios\*\*: Detailed scenarios outlining the CRUD operations tested, including both successful and edge cases.

2. \*\*Query Results\*\*: Screenshots or saved outputs of manual SQL queries demonstrating the results of CRUD operations.

3. \*\*API Responses\*\*: Logs and screenshots of API responses from Postman/curl demonstrating CRUD operations for all entities.

Submission:

-----------

Include the following in your submission:

1. `models.py` file with model definitions and CRUD functions.

2. Evidence of database testing in the `Documentation` folder.

3. Ensure all configuration and dependency details are clear and easy to follow.

Evaluation Criteria:

--------------------

1. Model Design (40%):

- Proper fields, types, and relationships defined for each model.

2. CRUD Functionality (30%):

- Basic CRUD operations implemented and tested successfully.

3. Code Quality (20%):

- Clean, modular, and well-documented code.

4. Submission Completeness (10%):

- All required files and testing evidence included.

Thank you for reviewing this assignment. Feel free to reach out for any further clarifications.

====================================================

This README.txt file provides a clear and detailed guide for setting up, running, and testing the project, ensuring that anyone who follows it can easily understand and verify the functionality of the database models and CRUD operations.