JS311 Checkpoint 2:

Line Please! A Memorization App

For a visual reference supporting this document, please refer to the following links:

- **Tables (Google Sheet): Line Please! A Memorization App**
- **Wireframe Explanation: Line Please! A Memorization App**

Layout Explanation

Overview:

This app is a memorization app in which the database holds the following information:

- User first names & last names
- Script Titles
- Names of playwrights and authors
- Acts and Scenes within a script (to help divide large bodies of text for actors to work on)
- The text of the dialogue that is to be memorized

Table Relationship & Data Held:

- actors Table: user's id, first_name, last_name
- scripts Table: script's id, user_id, script_title
- playwrights Table: playwright id, user_id, playwright_lastName, playwright_firstName
- actorsLibrary Table: actorsLibrary id, user_id, playwright_id, script_id
  Note: This table connects the actors Table, scripts Table, and the playwrights Table.
- scriptScenes Table: user_id, script_id, sceneNumber, sceneName, dialogue, punctuation response, addFirstLetter response, addEveryOtherLetter response

  Note: This table holds the actual dialogue that the actor is memorizing. It also stores the script by using scene numbers and names for easier reference as more data is stored.

  This is the heart of the application. It should do the following:
  - It should be able to take in a full body of text given by the user.
  - It should return just the punctuation of the dialogue entered by the user.
  - It should then add the first letter of every word, in addition to the punctuation.
  - It should then add every other letter in a word, in addition to the first letter of a word, and the punctuation which was returned by the first two functions.

SQL Statements for Table Creation:

actors Table:

```
CREATE TABLE actors(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        first_name VARCHAR (100) NOT NULL,
        last_name VARCHAR (100) NOT NULL
);
```

scripts Table:

```
CREATE TABLE scripts(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        FOREIGN KEY (actor_id) REFERENCES actors (id),
        script_title VARCHAR (100) NOT NULL
);
```

playwrights Table:

```
CREATE TABLE scripts(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        FOREIGN KEY (actor_id) REFERENCES actors (id),
        playwright_lastName VARCHAR (100) NOT NULL,
        playwright_firstName VARCHAR (100) NOT NULL
);
```

actorsLibrary Table:

```
CREATE TABLE actorsLibrary(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        FOREIGN KEY (actor_id) REFERENCES actors (id),
        FOREIGN KEY (playwright_id) REFERENCES  playwrights (id),
        FOREIGN KEY (script_id) REFERENCES scripts (id)
);
```

scriptScenes Table:

```
CREATE TABLE acriptScenes(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        FOREIGN KEY (actor_id) REFERENCES actors (id),
        FOREIGN KEY (script_id) REFERENCES scripts (id),
        sceneNumber INT NOT NULL,
        sceneName VARCHAR (100) NOT NULL,
        dialogue VARCHAR (10000) NOT NULL,
        punctuation VARCHAR (1000) NOT NULL,
        addFirstLetter VARCHAR (10000) NOT NULL,
        addEveryOtherLetter VARCHAR (10000) NOT NULL
);
```

List of Routes:

GET ROUTES

GET /actors - This route returns an array of actors, their ids, lastName, and firstName. It does not need any input.

GET /actors:id - This route returns a single actor by their id, which includes all of the information tied to that id including first and last name, and script titles that they are working on.

GET /scripts - This route returns an array of scripts, their ids, and titles. It does not need any input.

GET /scripts/:id - This route returns a single script by their id, which includes all of the information tied to that id including the actor_id who is using the script and the script title.

GET /playwrights - This returns an array of playwrights, the actor_ids, and the playwright's last and first names.

GET /playwright/:id - This route returns a single playwright by their id, which includes all of the information tied to that id including the actor_id who is using the script, the script title, and the last and first names of the playwrights belonging to that script.

GET /actorsLibrary - This route returns an array of ids, the actor_ids, the playwright_ids, and the script_ids.

GET /actorsLibrary/:id - This route returns a single actorsLibrary based on its id, which includes all of the information tied to that id including the actor_id, the playwright_id, and the script_id.

GET /scriptScenes - This route returns an array of scriptScenes, their ids, the actor_ids, the script_ids, sceneNumbers, sceneNames, dialogue, punctuation return, addFirstLetter return, and addEveryOtherLetter return.

GET /scriptScenes/:id - This route returns a single scriptScene by its id, which includes all of the information tied to that id including the actor_id, the script_id, the sceneNumber, the sceneName, the dialogue, the punctuation return, the addFirstLetter return, and the addEveryOtherLetter return.

POST ROUTES

POST /actor - This route adds a new user to the database. It takes in the first and last names of the user as input in the request body. The body of the actor includes the id, first, and last name of the user.

POST /script - This route adds a new script to the database for a given user. It takes in the script title as input in the request body. The body of the script includes id and the title of the script.

POST /playwright - This route adds a new playwright to the database for a given user. It takes in the playwright's last name and first name. The body of the playwright includes the id, the first name, and the last name of the playwright.

POST /scriptScene - This route adds a new scene to the database for a given user. It takes in the sceneNumber, sceneName, and the dialogue for the scene. The body of the scriptScene includes the id, the actor_id, the script_id, the sceneNumber, the sceneName, and the dialogue for the scene.

PUT ROUTES

PUT /actor/:id - This route updates an existing actor. It takes in the id of the actor to update as a path parameter, and the new actor first_name and/or last_name in the request body.

PUT /script/:id - This route updates an existing script. It takes in the id of the script to update as a path parameter, and the new script_title in the request body.

PUT /playwright/:id - This route updates an existing playwright. It takes in the id of the playwright as a path parameter, and the new playwright_lastName and playwright_firstName in the request body.

PUT /scriptScene/:id - This route updates an existing scriptScene. It takes in the id of the scriptScene as a path parameter, and the new sceneNumber, sceneName, and dialogue in the request body.

DELETE ROUTES

DELETE /actor/:id - This route deletes an existing actor. It takes in the id of the actor to delete as a path parameter.

DELETE /script/:id - This route deletes an existing script. It takes in the id of the script to delete as a path parameter.

DELETE /playwright/:id - This route deletes an existing playwright. It takes in the id of the script to delete as a path parameter.

DELETE /scriptScene/:id - This route deletes an existing scriptScene. It takes in the id of the scriptScene to delete as a path parameter.