README:

We used node.js and npm to install all dependencies for our project:
https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

Dependency:
For convenience, run the following script in both directories to install dependencies:

```
npm run install-all
```

Front-end dependencies: dbms-project

```
npm install react
npm install bootstrap
npm install redux
npm install @reduxjs/toolkit react-redux
npm install react-router-dom
npm install axios
npm install bootswatch
npm install bootstrap-icons
```

Back-end dependencies: dbms-project-node

```
npm install express
npm install nodemon
npm install cors
npm install axios
npm install express-session
npm install mysql2
npm install dotenv
```

Environment variables: the password must be filled in for the node server to access the database.

dbms-project-node/.env:

```
PORT=4000
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=
```

dbms-project/.env:

```
REACT_APP_API_URL=http://localhost:4000/api
```

Running the application:
/dbms-project-node:

```
nodemon app.js
```

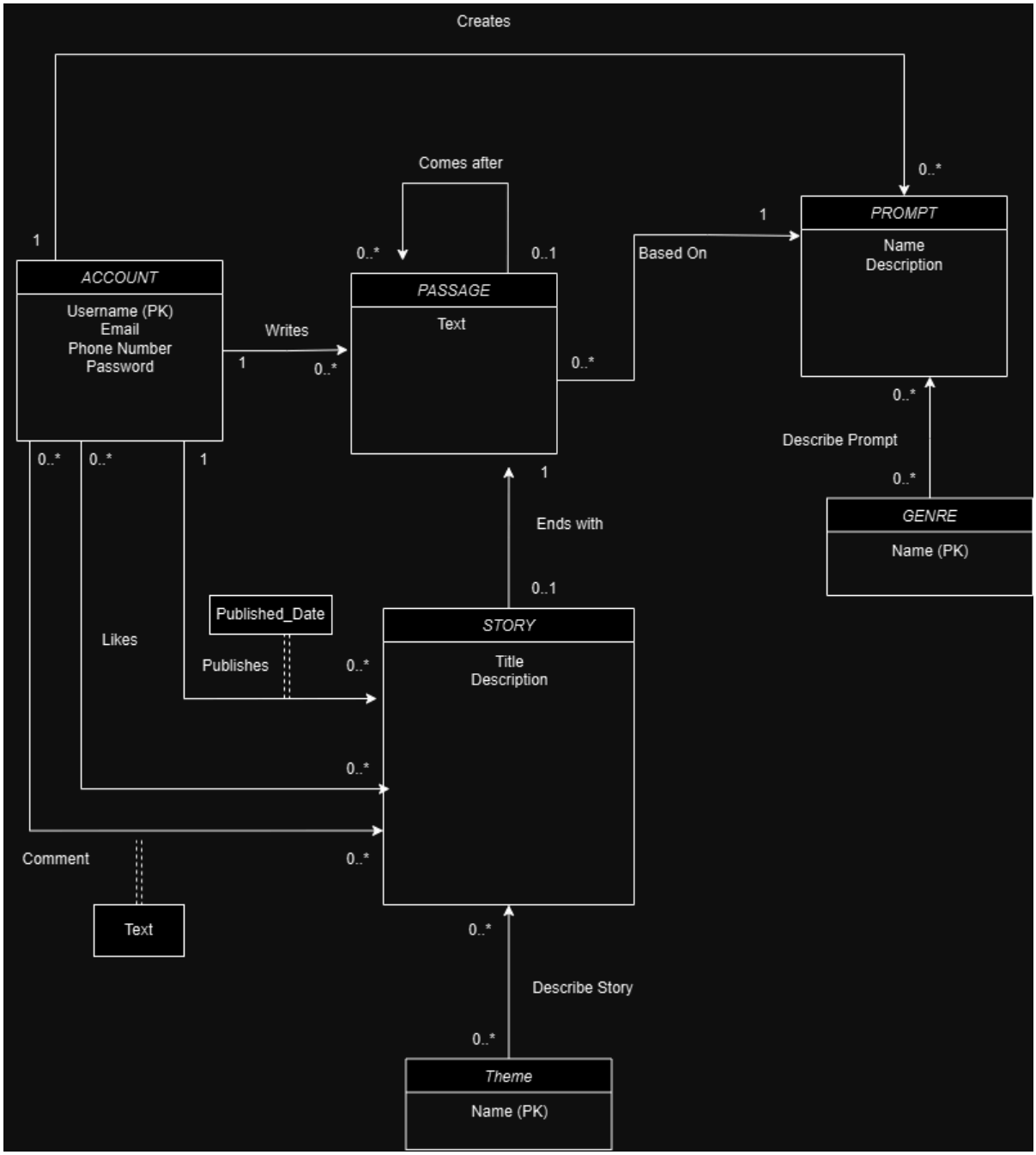
/dbms-project:

```
npm start
```

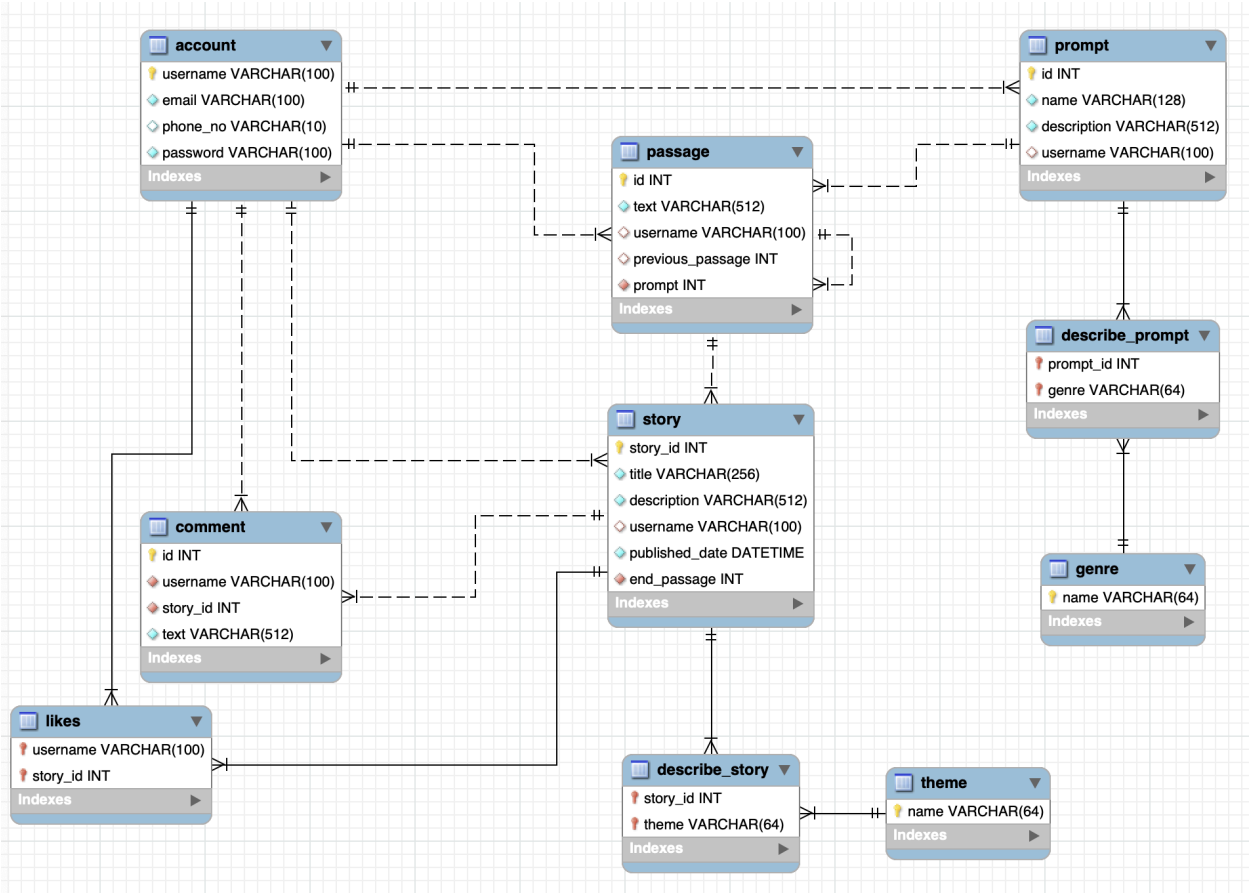
Tech Spec:

Language: Javascript
Front-End Framework: React
Back-End Framework: Node.js
Database: MySQL

UML Diagram:


Creates
Comes after
0..*
1
PROMPT
Name
Description
1
ACCOUNT
Username (PK)
Email
Phone Number
Password
0..*
0..1
Based On
PASSAGE
Text
Writes
1
0..*
0..*
0..*
Describe Prompt
0..*
0..*
0..*
1
1
GENRE
Name (PK)
Ends with
0..1
Published_Date
STORY
Title
Description
Likes
Publishes
0..*
0..*
Comment
0..*
Text
0..*
Describe Story
0..*
Theme
Name (PK)

Logical Design:



User Pages:

Below are different pages in our website and their respective workings:

- Navigation Bar:
    - Always there:
        - Stories: list of stories
        - Prompts: list of prompts
    - Signed in:
        - Profile: see your profile
        - Logout: sign out the user
    - Anonymous User:
        - Login: where a user can log in
        - Register: where a user can register
- /
    - Home page that displays all stories
    - READ: all stories from database, pulling info about their themes, number of likes, and prompts as well
    - NAV:
        - Click on the title of a story to get to /details/:storyId

- /prompts
    - Displays all of the prompts made by users
    - READ: all prompts from database, pulling info about their title, description, genres and creator
    - NAV:
        - Click on prompt card to get to /prompts/:promptId
        - Click on create new prompt button to get to /prompts/create (must be signed in)
- /prompts/create
    - Displays the information needed to create a new prompt. Includes a text box for the prompt name, a text box for the prompt description, and checkboxes for the associated genres. At the bottom is a button to create the prompt.
    - READ: genres that can be associated
    - CREATE: prompt
    - CREATE: describe_prompt (table that connects prompts and genres)
    - NAV: (must be signed in to be on page)
        - Creating a prompt will take you to /prompts
- /prompts/:promptId
    - Displays the prompt name associated with the prompt id at the top. If logged in, there is a button to create a new story that will be based off of the prompt with the prompt id. At the bottom is a list of the stories that have already been created based on the prompt with prompt id.
    - READ: prompt title
    - READ: stories based on the prompt
    - NAV:
        - Clicking create new story will take you to /make/begin/:promptId (must be logged in)
        - Clicking a story will take you to /details/:storyId
- /make/:passageId AND make/begin/:promptId
    - Displays the prompt, previous passage, and next passages of a user's position in a story. Here, a user can choose to move up a spot by selecting the previous passage, down a spot by choosing one of the already created next passages, or create their own new passage that will follow the previous passage. The user types their desired passage into the text box in the middle of the screen, then they have two options. Firstly, they can click "finish passage", and the passage will be added to the database if it is unique (a passage is unique if no others match its text and position/previous passage). The other option allows a user to publish a story. Publishing a story will bring the user to a new display that asks for a title, description, and theme. Once published, the passage is created, the story is created (being linked to the passage just created as its ending), the themes are connected to the story, and the user is directed to the details page of the newly created story.
    - READ: previous passage (if not at the top)
    - READ: next passages

    - READ: prompt
    - CREATE: passage
    - CREATE: story
    - CREATE describe_store (table that connects stories and themes)
    - NAV: (must be signed in to be on page)
        - Publishing a story will take you to /details/:storyId
        - Click on a passage to be taken to /make/:passageId
- /details/:storyId
    - Displays information related to a story: title, passages, authors, likes, and comments. Also allows a user to like or comment on the story if they are signed in.
    - READ: builds the story's content by traversing through the passages tree
    - READ: story metadata
    - READ: comments about the story
    - CREATE: a user's new comment on the story
    - UPDATE: a user's comment (user can only update their comments)
    - DELETE: a user's comment (user can only delete their comments)
    - CREATE: a user likes a story
    - DELETE: a user dislikes a story
    - NAV:
        - Click any username to be taken to /profile/:username
        - Click a passage's text to be taken to /make/:passageId (must be signed in)
    -
- /login
    - Logs in a user and initiate a session with the server
    - READ: account
    - NAV: directed to home page when logged in
- /register
    - Registers a new user
    - CREATE: new account
    - NAV: Directed to home page when registered
- /profile
    - Displays the current user's profile
    - READ: current user's account
    - READ: current user's passages
    - READ: current user's story
    - READ: current user's prompts
    - NAV: (must be signed in to be on page)
        - Click edit to be taken to profile/edit
        - Click on a story to be taken to details/:storyId
        - Click on a passage to be taken to make:passageId
        - Click on a prompt to be taken to /prompts/:promptId
- /profile/edit

    - Allows a user to change their email and phone number
    - UPDATE: current user's account
    - NAV:
        - Click save to be taken to /profile
- /profile/:username
    - Views another user's profile and work
    - READ: other user's account
    - READ: other user's passages
    - READ: other user's story
    - READ: other user's prompts
    - NAV:
        - Click on a story to be taken to details/:storyId
        - Click on a passage to be taken to make:passageId (must be signed in)
        - Click on a prompt to be taken to /prompts/:promptId

Lessons Learned:

Technical: The project was a great opportunity to get our hands dirty in web development processes. Neither one of us were very experienced working within the React and Node.js framework, so getting oriented and learning the different measures to ensure a well-functioning site helped us grow as programmers. We learned a lot about RESTful API calls, using Postman to explore our backend endpoints. We learned how to dynamically react to data changes on the front-end and service requests on the back-end. We worked with reducers, components, query pools, and controllers. Our database also had us figure out how to represent a tree structure in mySQL for our passages, building the stories by traversing up the tree.

Planning: It became clear to us throughout the project how important it is to come up with exact definitions of schemas, especially when transporting data between the server and the client. A simple misnaming or improper structure could break the pipeline. It was important to openly communicate how the data was going to be structured and the uses of the data thereafter. This allowed us to design our website with intention and save time debugging network logs.

Design: We were initially a little worried about how the tree structure of passages would be used, so we were considering making every passage linked, then when a story is published, linking the story to each passage with a sequence number. In the end, we did not have to do this as we figured out how to build the stories through a recursive mySQL statement. We found our final design represented our data exactly how we wanted it to.

Error Handling: We realized during development how important it is to properly handle errors throughout our design. A solid principle we came to was: if the backend requests fail to deliver, our website should not crash and should be capable of recovery. We enacted this by turning off our server and navigating through the different pages, adding checks for null pointers and try-catches to aid in the site's durability.

Roles: We were not able to fully implement user roles as a result of time. If we had, an admin role would allow a user to create themes and genres and delete any work they deem offensive.

Future Work:

Functionality: In the future, we would like users to be able to filter stories by theme, genre, and name. We currently do not support any search features, so once the website scales, it will be hard to find the stories our users are searching for. Our display pages are also not modularized yet. The home page displays all stories, which is not feasible for a considerable amount of data. Ideally, we would create a sort of paged display that limits the amount of stories queried for on each request. Then, a user could move to the next page to access more stories. We do not support the ability to unpublish stories. Nor do we support the ability to delete passages, but passages would become more complex as they rely on each other, so only a passage without any branches off it would be eligible for deletion. We also think it would be a cool idea to allow users to follow one another, allowing them to easily see what passages and stories their friends are publishing on the main page (becoming a sort of timeline feed) instead of navigating to a user's profile page.

Security: At the moment, our server is not secure at all to SQL injections and other malicious behaviours. We authenticate a user's login with session cookies, but this does not prevent users from injecting sql code that could do some serious damage to our database. In the future, we would look to guard against this. We use some procedures which help, but we would further fortify our defenses by using prepared statements and/or validating and sanitizing calls to our back-end.