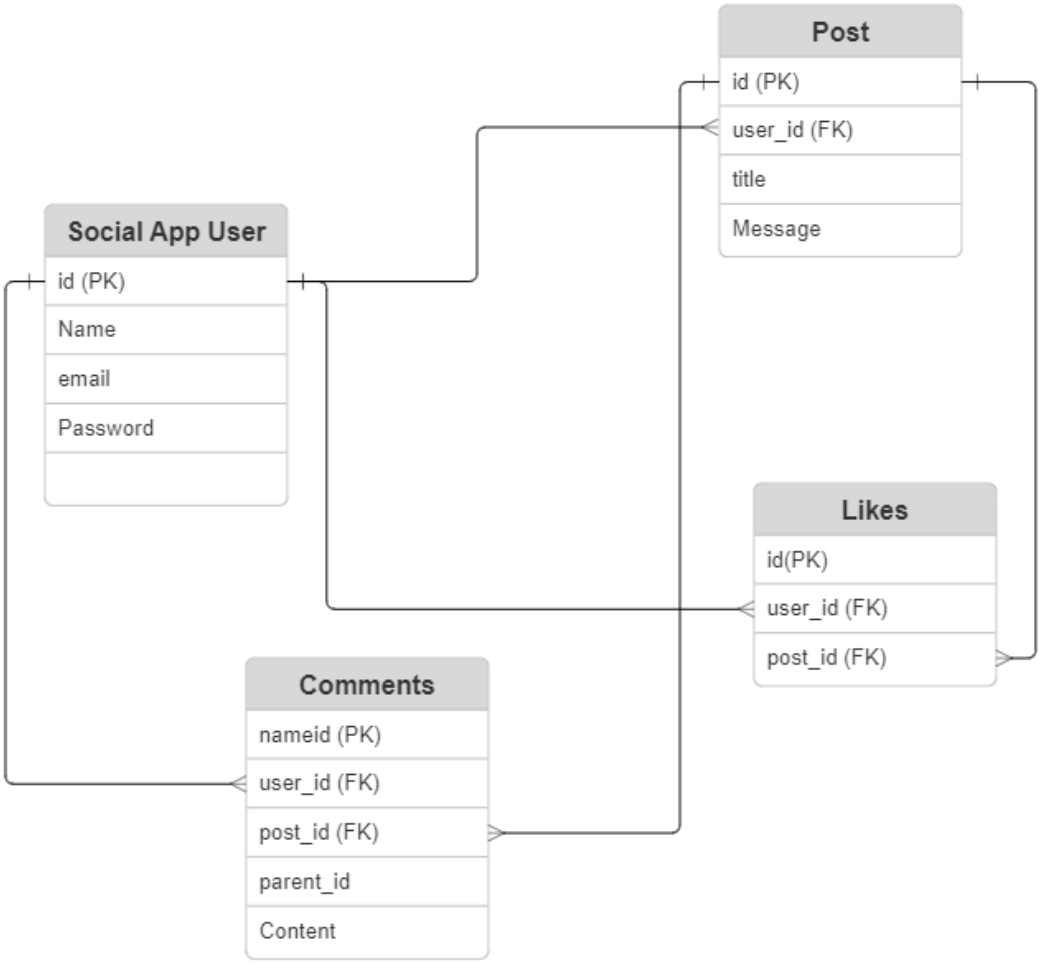
Documentation:

1) ER DIAGRAM:



2) I tried to my best to use blade components and Laravel livewire components to structure my social network site. I tried to organize every front-end structure into reasonable blade components but could not completely do it because of the time factor. I applied a good enough of styling (using Tailwind CS) to the HTML elements but left a little at the end part. I have included input and SQL sanitization to the point applicable in our case. Specifically I used parameterized queries most of the time with values bound with parameters placeholder instead of directly injected to the query statement(except one time because I could not get a proper workaround). I had done the input validation (client side) of adding comments as per the requirement of the questions and also the server-side validation for the addition of posts. I have used proper JS visuals or session flash messages to communicate the errors/success to the users.
I tried to be very consistent with the naming conventions and tried to pick variable names that had a connection with the value it is going to store. I have mostly used camelCase and kebab

case as per the documentation of Laravel. Almost 70% of my code is well indented but couldn't fully indent my code because of the time factor. I have used transaction confirmation and validation feedback when the users submit forms. I also used factories and seeded data into my database. I took my time to make my front end as reactive and interactive as possible with some basic Alpine js(I used it just for navigation bar) and normal JavaScript.
I was not able to exhibit the nesting of comments and replies, though I got the idea of doing so (by recursively calling the same view component for the child comments). This was because I had already implemented another template inheritance which made it difficult to readjust to call the template recursively to achieve this nesting comments layout.

3) I underestimated the task and tried to start the project without proper planning (apart from the tables and models that were going to be used). Thus, I faced lot of challenges as I tried to incorporate every requirement of the questions in the assignment. I had difficult time to change my code to fit the requirements of the assignments which made me submit the assignment late as I had errors propping out when trying to adjust the code (as in the case of trying to change my previous code to implement the comment nesting). Since I lacked proper planning, I was sometimes in a situation where I couldn't think of changing my code because I had done so much even if the requirement required me to do so. But it was a good learning experience even if I faced lot of difficulty as It allowed me to explore a bit about the TALL (Tailwind Alpine Livewire Laravel) components that work very well with each other. I would like to refactor my code and make my logic as short as possible in the next part of the assignment.