Jacob Rangel Sean Eppling Rebecca Santos Anthony Delgado

Assignment 1

In this assignment we will come up with an initial design for a software application that you will build in this semester. We will not be writing any code in this assignment, but only looking at some initial design ideas and high level architecture.

Description:

A partner of your company has requested to build a software application that will predict the rate of the fuel based on the following criteria:
- Client Location (in-state or out-of-state)
- Client history (existing customer with previous purchase or new)
- Gallons requested
- Company profit margin (%)

Software must include following components:
- Login (Allow Client to register if not a client yet)
- Client Registration (Initially only username and Password)
- Client Profile Management (after client registers they should login first to complete profile)
- Fuel Quote Form with Pricing module (Once user enters all required information pricing module calculates the rate provides total cost)
- Fuel Quote History

Answer these questions:

1. Discuss your initial thoughts in detail on how you will design this application? (2 points)

Initial Thoughts

So when it comes to creating this application at a high level, there are a lot of components to start thinking about.

- ***Database Design:*** Since we are making a dedicated application where we need to extract information from users and store it, a database would be pivotal to make our application vital to the parent company. Some things to think about is storing client information like their login information, company credentials and their fuel quote histories.
- ***User Authentication:*** It is also important that we create an intuitive way of verifying users with a secure login system. Ways we could do this is with password encryption, 2-Step Verification or a Captcha system.
- ***Profile Management System:*** We need a way for client users to also be able to modify any of their companies information **AFTER** their account has been created whether it has to do with scaling of the company, revenue changes or simple operations like a name rebranding.
- ***Fuel Quote Module:*** We need to implement a system where we are able to access client information correctly from the database and utilize that information to create an

accurate fuel quote for the client company. It is important that we are also able to verify that the information is being inputted correctly from the user so we don't get any formatting issues when calculating the fuel quote.
- ***Fuel Quote Histories:*** A significant factor for our application is letting the client user be able to see all of their previous and current fuel quotes and us as software designers, we want to make sure that this information is detailed in a structured manner so that they are able to review these factors that calculated their quote. It's highly important that we make this design look appealing for the client consumers, *since this is the major reason clients come to our application*.

Interface Design

Start with a landing page where new users can create an account or old users can sign in.

- New users are taken to another page where they are able to input their new login information and their company details. They are then encouraged to make a new quote based on their new data.
- Old users are taken to a page with current and past quotes and capability to make new quotes, etc.

Users can also look at all their previous quotes and view all the information they inputted to receive that corresponding quote. Some important information, ***not limited to***, that we will use to calculate this quote are:

Client Data
- Client Location
- Client History
- Gallons Requested
- Company Profit Margin
- Company Size
- Fuel Type

Application Data
- Supplier Relationship
- Market Conditions
- Economic Conditions

Once a quote is placed, users can also go into their profile settings to update any company information that needs to be modified at a later time.

2. Discuss what development methodology you will use and why? (2 points)

Thoughts

There are a lot of development methodologies but the one that would suit our project better would be **Agile,** most notably the **Test-Driven Development Model.** Considering our parent company wants us to create this application for our clients, it would most likely need to evolve through its lifespan which is why we choose this model. Here are some factors that led us to this resolution:

Agile Development

Since Agile is well-suited for projects with changing requirements and frequent client interactions, this would let us prioritize features based on client feedback throughout the development process. Albeit once we are able to fulfill all the client feedback, maintenance and improvement to the application is all that will need to be casually worked on.

Test Driven Development Model

What makes the **TDD Model** unique is that this model will allow us to envision the end goal and strategize a way to write a working code to reach that end goal. This allows us as a team to quickly prioritize features and actively work to find solutions in quick bursts.

Once we are able to find a working solution, we clean up the code and begin the process of the next feature to be implemented. This promotes a more systematic and goal-oriented development process and allows us to facilitate working code rather than the best approach in the beginning, even though the working code will later be managed with cleanliness later on.

3. Provide high level design / architecture of your solution that you are proposing? (6 points)

Technologies

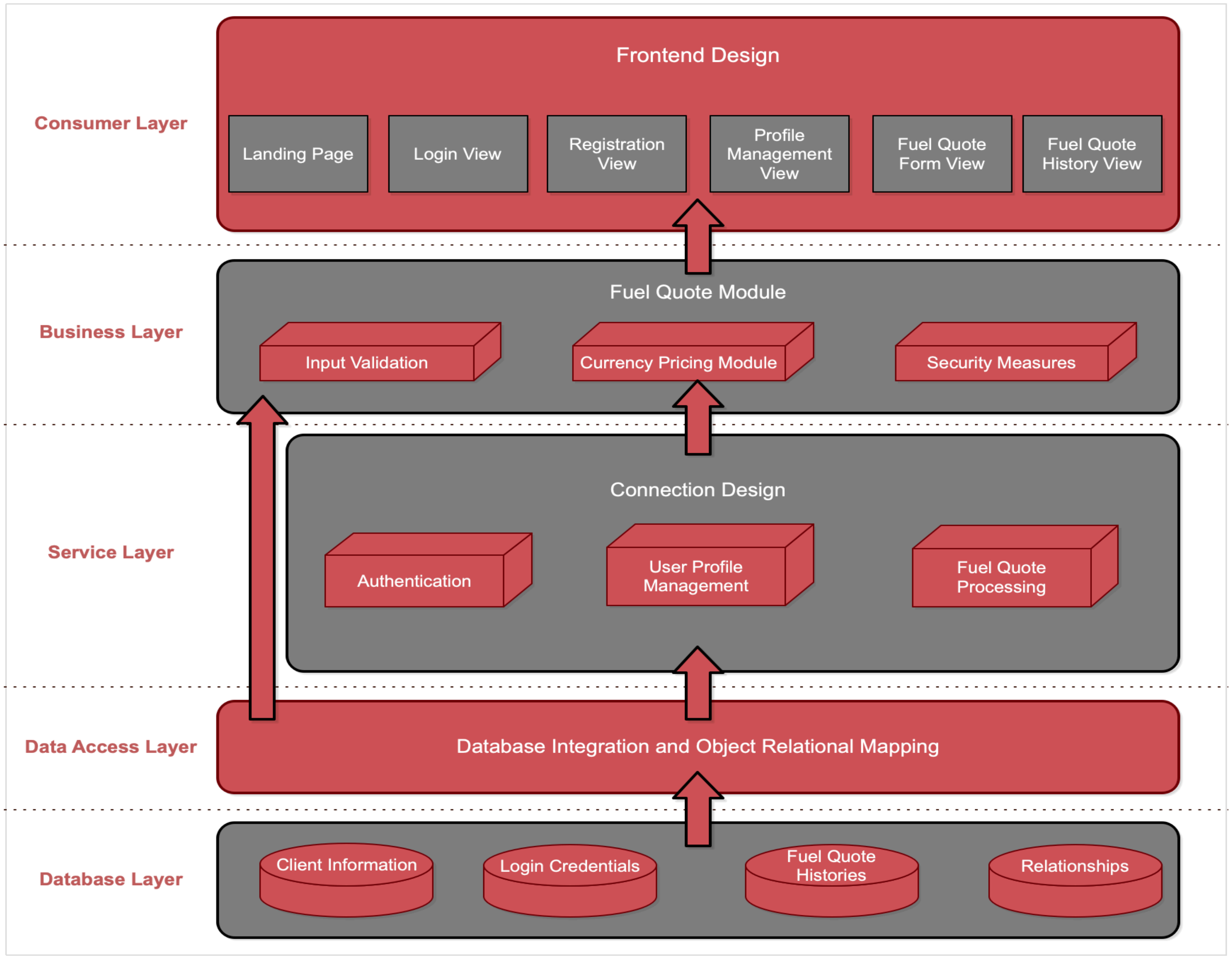
We will be using the ***PERN Stack***:

- **PostgreSQL** for database creation, maintaining, and storing data needed at a larger scale for a company that deals with multiple clients.
- **Express** for assisting the Node.js server and facilitating the backend of our application.
- **React** for our front-end development, allows us to create components more efficiently.
- **Node.js** for translating javascript code into helping the HTML structure of our application.

Application Architecture

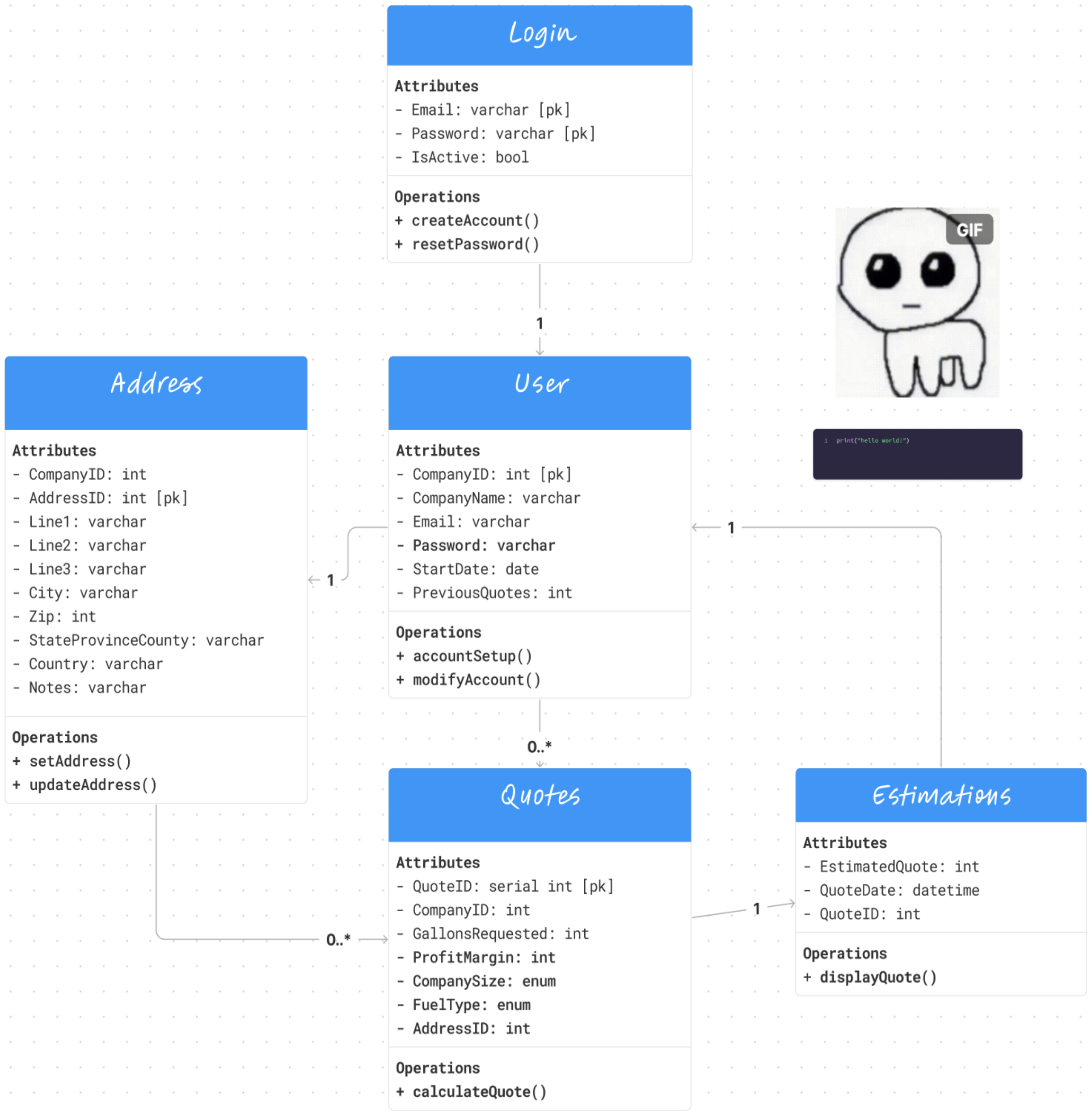
Here is a visualization of our fuel quote application with all the layers portraying which levels are depending on another level. The layers we included are:

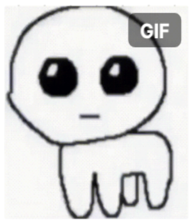
- **Consume/Presentation:** Represents the user interface of our application, implemented using the React framework with various components.
- **Business:** Encapsulates the core mechanics and rules of the application.
- **Service:** Implemented with Node.js and Express, handles the communication between the frontend and the backend.
- **Data Access:** Focuses on integrating with the PostgreSQL database and includes an ORM system for translating between the application and the database.
- **Database:** Serves as the storage and retrieval system for the application.



UML Class Diagram

This is our UML Class Diagram visualizing all the connections within our application. It also has all of the association relationships between the classes. You can check out our design at Figma with the link attached here.

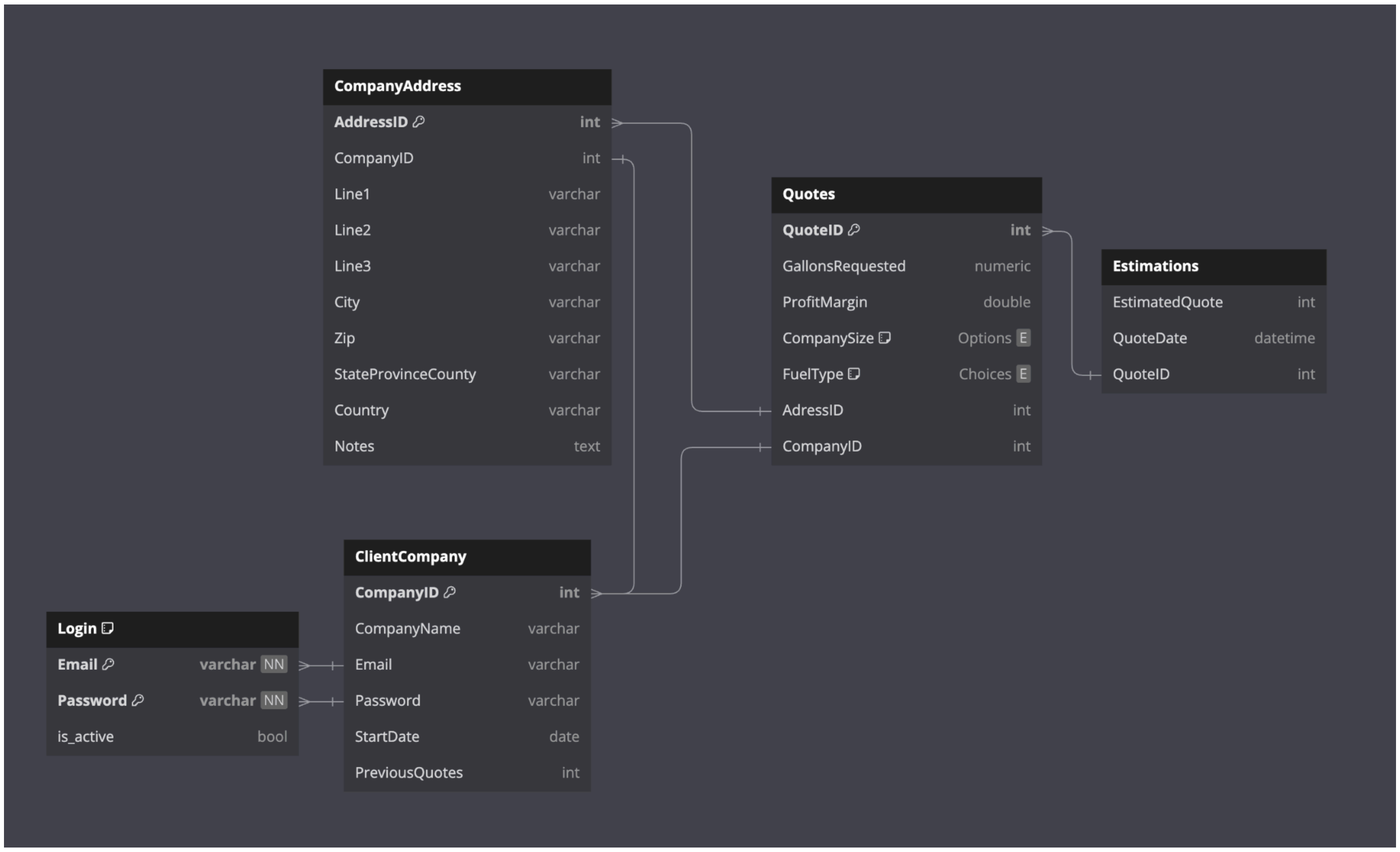




```
print("hello world!")
```

Database Diagram

Here is a Database Diagram visualizing the entities and attributes of important information we would need to acquire in order to create the backend of our fuel quote application. You can also check out the working DBML code on dbdiagram at the link here.



CompanyAddress	
AddressID	**int**
CompanyID	int
Line1	varchar
Line2	varchar
Line3	varchar
City	varchar
Zip	varchar
StateProvinceCounty	varchar
Country	varchar
Notes	text

Quotes	
QuoteID	**int**
GallonsRequested	numeric
ProfitMargin	double
CompanySize	Options E
FuelType	Choices E
AdressID	int
CompanyID	int

Estimations	
EstimatedQuote	int
QuoteDate	datetime
QuoteID	int

ClientCompany	
CompanyID	**int**
CompanyName	varchar
Email	varchar
Password	varchar
StartDate	date
PreviousQuotes	int

Login	
Email	**varchar** NN
Password	**varchar** NN
is_active	bool

Group Member Name:	Contributions	Discussion Notes
Jacob Rangel	Worked on sections of the Google Doc, worked on the UML class diagram, worked on the database diagram, helped enforce quality and cohesiveness of diagram designs	- Our team is hardworking. 😎 - When developing the idea of our application, we weren't sure how intricate we needed to be for the assignment, so we answered each question with ample information and made each high-level design in accordance with the others. Hope it seems well executed.
Sean Eppling	Worked on UML class diagram, designed entities, Helped visualize the relationships between entities and classes for the high-level designs	Love my team so much <4 - Important that we make sure to follow through with what we plan in this document throughout the development cycle of our application. It's the whole reason that big companies follow their development methodology to keep complex systems manageable for teams.
Anthony Delgado	Worked on UML class diagram, helped establish relationships between classes, shared thoughts on structuring fuel quote histories, advocated for chosen development methodology	- We need to take scalability into account, are we already dealing with a lot of clients? Or will we need to just stay within the context of the class project, this is an important question in regards to system security and so on - There might be additional components we can include that can improve user experience, such as real-time notifications for users where a quote has been approved, or even data visualization for fuel pricing trends - Happy with what our team has been able to get down for our initial design, looking forward with moving on with our project
Rebecca Santos	Worked on UML class diagram, added operations, drove the idea of a "user-first" experience, helped finalize the tech stack that will be used for the project	- It's really important that we carefully design the user-interface so that clients can seamlessly have quotes be created and retrieve past ones if needed - Team is great at communicating and working together!

Thank you!

Team 1