COSC 4353 - Group 1 Assignment 1

1. The application has four criteria/variables to determine the fuel quote for a customer; client location, client history, amount requested, and profit margin. These four criteria can have values set and shipped with a desktop software application or be dynamically available through a web application or call to a web server. As client account information has to be stored and retrieved as well, likely through a database, a web application would make sense for this project. The four values can be seen as weights on a total rate of the fuel. For example, a possible algorithm may be Fuel Rate = 1.0 (In-state) x 0.9 (Returning Customer) x 1.15 (15% Profit Margin) x 1.1 (Gallons request < 300) x 4.2 (Base price of fuel).

   This application can then be implemented as a website frontend where a user is confronted with a homepage that asks them to login or register if they do not have an account. The login credentials are checked against a backend database which contains all of the profile information for each user. Once logged in, the user can view their profile where they can change their username, password, or location. They can also view their own fuel quote history or request a new quote. The new quote will be a form that only asks for location, which can be auto filled with the user's profile data if not null, and number of gallons requested. In the backend the rate will be calculated based on the above mentioned four variables, and then returned to the user. The user can then accept the quote, after which it will be added to their fuel quote history, or reject it.

   In the backend, the application will require a database with two entities: user and fuelQuote. The user entity will have a one-to-many relationship with the fuelQuote entity, and have attributes that include username, password, location, name, email, and phone number. The fuelQuote entity will have attributes quoteNumber, fuelAmount, fuelRate, and username where quoteNumber is an artificial key and username a foreign key referencing the user entity.

   Outside of the database, the application will require four classes, the login, profile manager, fuel quote, and fuel quote history. The login class will have getters and setters for the user's username and password variables as well as methods to register and login the user. The profile manager will allow the user to edit their location, profile name, and other common profile attributes like email address or phone number. It will also track the user fuel quote history, and have getters and setters for those attributes. The fuel quote class will have methods to calculate the fuel rate and total cost of the quote using the requested fuel amount and user info, as well as getters and setters for those attributes. Finally the fuel quote history will have functions to save a quote to the user's history or retrieve all of the quotes they have made.

2. We will be using the Agile methodology and Scrum framework for this project. We chose this methodology because of its adaptive and light-weighted nature in contrast to a thorough step-by-step Prescriptive approach. Prescriptive Models could work as well since the requirements are well defined and the project is not a very large/complicated project. Due to how rigid and not as time efficient it is, Agile would be a better approach since its a more collaborative-based approach and the availability of feedback from the professor and TA's. We will be using a Web based application with relatively straightforward functions like storing quotes/login information and calculating fuel price therefore it would be useful to minimize time towards design and documentation. Also project requirements may evolve over time so having the flexibility to go back and tweak things without making major changes will be helpful.

3. See attached.

4.

Group Member Name	What is your contribution?	Discussion Notes
1. Bhuvana Chandra Adhugiri	Built the use case and class diagrams with Jeremy. Built the sequence diagram.	Worked together to determine the attributes for each class as well as determining the relationships between those classes.
2. Jeremy Crain	Wrote up application design and built the use case and class diagrams with Bhuvana.	Wrote up the application design based on assignment requirements. Worked together to determine the attributes for each class as well as determining the relationships between those classes.
3. Sean Nguyen	Explained what methodology will be followed and why	Group decided on following Agile methology.