Assignment 1

In this assignment we will come up with 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 (%)

Requirements

- 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

Questions

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

Suryansh Sharma

I think we should design this based on easist/ fastest. Meaning we do 1st: Set up base page to build modules off of aka index.js 2nd: Login Page 3rd: Registration Page 4th: Profile Management Page 5th: The last two should be further split up into smaller parts when we get to them so they can be tackled properly - Fuel Quote Form with Pricing module (Once user enters all required information pricing module calculates the rate provides total cost) - Fuel Quote History I also think we should use HTML,JS,CSS,React for front end, something for backend?, something as database? im familar with postgreSQL for backend if thats needed and im sure we can figure out something for the backend portion

Ryan Ball

The first step will be deciding on langauges team members are familiar with and their experience with frontend and backend development. The frontend and backend technologies will then need to be decided to best fit the group as a whole to maximize the efficiency of the group. Foundational knowledge in REST would be benefitial, along with SQL (assuming not using NoSQL type databases). As mentioned by Suryansh, there should be a focus on getting the server up and running initially with an index.{html, js} then working getting communication between the frontend and backend working.

After we have a functional testing platform for implementing features both on the front and backends, we start branching out into modularizing the application. Good candidates for modules would be:

- Homepage
- Login
- User registration
- User
- Input Form
- Output / History

Benjamin Mogeni

My initial strategy would be to approach the problem from a user centered viewpoint and build from there. Break down the application into core components for the front end and backend and based upon the teams technical experience we divide tasks as neeeded. For the architecture I believe a React based frontend with Node.js backend and sql fdor the database. This will be open to evolution of course as the project develops and more precise tools are neeeded. I agree with Suryansh on the order of development as going from easiest to hardest will allow for the team to get used to the tech stack and learn as we build.

Abhinav Krothapalli

One of my initial thoughts is to first sort out what tasks should be developed in the front end and back end. For example, tasks like the client login and/or registration, client profile management, and fuel quote form will be made in the front end because of how the client will need to interact with these. Once the client inputs the information for those tasks, the information can then be stored in a database. The pricing module will be doing all of the calculations in the back end and send the fuel quote history back through the front end for the client to see. In terms of what languages to use for this, I agree with Ryan in that we should decide on something that we are all comfortable with.

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

Suryansh Sharma

I think we should use the Agile development methodology because its a lot more common in the industry, its also meant for projects like this because we can easily split up step in the software component requirements list into 1 or more stories based on their difficulty and tackle issues in each component as they arise. We could also make a story on issues in compoenents interactin with each other if it comes to that.

Ryan Ball

I agree Agile will be a great approach. Many hands will be woring simulataneously on the same project and splitting it into multiple smaller modules would be beneficial to have speedier turnaround on the project.

Benjamin Mogeni

I also agree on Agile as the development and flexibilty needed for this project would greatly ebnefit. I believe we can also take from some DevOps methodolgies as well in our workflows to keep up our efficicency in regards to deployment. A couple key features to take would be focus on continious integration, automated

testing, and rapid delivery. A harmonious combination can positively impact the rate of development and quality as well.

Abhinav Krothapalli

I think having a mix of the Waterfall method and the Agile method would be best. The reason for the Waterfall method is because we will be working on this project in a very linear manner as we are starting off with the structure/planning then will be moving on to building the front end and back end. Waterfall method is good in these cases where all of the steps build off each other and the requirements of the projects stay static. The agile method is good for splitting the parts up as already stated.

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

Ryan Ball

REST Requests

- `GET /` (index.{html, js}), will render the Homepage
- `GET /user` presents the user information
- `GET /user/edit` presents the form to edit the user profile.
    - Clicking submit performs **POST** request that changes user information at `/user`.
    - Clicking cancel redirects to `/user`
- `GET /login` presents the login form.
    - Clicking submit performs **POST** request to create the user. Redirects to `/user`.
    - Clicking cancel redirects to `/`
- `GET /register` presents a registration form.
    - Clicking submit performs **POST** request and redirects to `/user`.
    - Clicking cancel redirects to `/`
- `GET /quote` presents a quote form to create a new quote.
    - Clicking submit performs **POST** request and redirects to `/quote/history`.
    - Clicking cancel redirects to `/user`
- `GET /quote/history` presents a history of all quotes that can be seen.

Models

- User, contains basic user information that
    - Created in `/register`
    - Can be edited in `/user/edit`
- Quote, the required information to create a new quote.
    - Will be displayed in `/quote/history`

Additional notes

These models are relatively simple to be represented in any SQL database. CRUD can also be easily performed due to their simplicity

Suryansh Sharma

I like this architecture / high level design so i'll include a UML drawing in the folder based off it.

4. IMPORTANT: list who did what within the group. TAs should be able to validate in GitHub, otherwise team members who didn't contribute will receive a ZERO.

1. Suryansh Sharma: Answered Questions
2. Ryan Ball: Answered Questions
3. Benjamin Mogeni: Answered Questions
4. Abhinav Krothapalli: Answered Questions