This document explains the goal of our website, the progress made over the span of Sprint 1, and our plan for Sprint 2. Finally, and a general summary of our task breakdown can be found in the "issues" and "projects" tab on our git page.

Project Goal

The team met frequently during Sprint 1, and close contact was held in order to move the process along smoothly. During our weekly meetings, we discussed at length the desired product our website will represent upon completion. We have used many forum-based websites as a reference to create a basic template – one that will be improved in the later sprints – with the common goal that our website will be a question and answer based-forum where engineers from all over the world can come together to collaborate on problem solving. The website will be tailored to Engineers, but nothing will stop those from other backgrounds from asking questions. The goal is to solve those engineering questions that have been asked throughout the academic life-cycle of every engineer with the purpose of allowing engineers to fast-track their studies by clearing up ambiguous concepts once-and-for-all.

Sprint 1

Progress

The template design started as a template .html and .css file that was uploaded to git with which all team members branched off from and created their respective page(s). This helped tremendously in bringing all the pages together once everyone had finished their part as the initial template maintained common spacing and coloring across all pages. All pages necessary to the functioning of the website are currently implemented, albeit more on the prototype-level in terms of visuals. The visuals will be upgraded in later sprints as the front-end and back-end will branch off into separate tasks as the entire product ascends to completion. The next section will describe each page, and who was assigned to them

Task & Website Breakdown

The currently implemented website contains seven basic pages – please note that the back-end is not yet implemented – with full front-end functionality and are as follows;

1. The home page (landing page) contains dynamically loaded information such as the newest questions, the most popular questions, and featured questions along with the point-metrics of the question. This task (as most were) was self-assigned to our teammate ***Jason.***
2. The questions page contains what will be the future question and answer page for any given unique question (accessed through a search for said question). As it stands, this page is accessible through the navigation-bar in order to allow the client to visualize the page without implementing any back-end logic, however as soon as our node.js server is up and running this page will be removed from the navigation-bar and loaded dynamically when a user searches for questions. At the bottom of the page, the user has the ability to post an answer to the forum. All questions and answers have user metrics displayed beside them for convenience. This task was assigned to ***Anthony.***
3. The ask page contains a very bare-bones question-input section. When this is filled out and submitted, the server will store the question for later searches. This task was assigned to ***Ruochen.***
4. The about page contains a small user-input area where a short message can be filled and sent to our email address (this sending aspect has not yet been implemented). This task was assigned to ***Nizar.***
5. The user profile page contains generic user-info on the left hand side, along with the ability to display all questions asked and answered to-date (as can be seen on the right-hand-side of the page). This task was assigned to ***Anthony.***
6. The login page is where the user will be able to input his previously registered credentials in order to use the website's features (such as asking and answering questions). This task was assigned to ***Saleh.***
7. The sign up page allows the user to input their desired information in order to register an account with our server. On completion, and after login, the user will be able to ask and answer questions. This task was assigned to ***Abishand*.**

A list of all completed tasks can be found in the "done" column throughout our "projects" on github.

Planning for Sprint 2

User Stories

Our team had initially gone under the assumption that we could implement the website as we desired in terms of when to focus on front-end, when to focus on back-end, however the client made it clear that the front-end was to be completed first, and that is what we did in Sprint 1. Although we had assumed incorrectly, we had begun planning the back-end before knowledge of the front-end requirement, leaving us with an advantage over Sprint 2.

For Sprint 2, our team is planning on splitting into smaller groups that will focus on improving the front-end visuals in parallel with implementing the back-end server response and database system. For example, our team-member Saleh will be in-charge of prepping the databases, and writing abstracted query functions in order for us to pull data quickly and without re-writing query code to populate our pages with dynamic information. Another handful of us will be in charge of the back-end: converting all current .html pages into .ejs to allow the dynamic populating of queried data, to implement the get/post responses, and allowing each user to maintain their own login-sessions ala php's sessions, only this time using Node.js. The remaining team members will be in-charge of the front end. As mentioned previously, they will improve visuals, but also add any front-end embedded javascript to the .ejs files as required by the back-end team.

The following are some user stories our team wishes to complete in Sprint 2:

User Story: Login/Sign-Up

"As a user, I would like to sign-up to this website, and login when desired with my email and password"

The user requires a login and sign-up system. For the initial phase, this login and sign-up will be very basic in order to follow the paradigm of incremental development. The login/sign-up forms will be on the same page, where the left-hand side includes the login system of email and password, and the right-hand side will include the sign-up system of email and password. The databases will store these values, along with other user properties that will be added to the sign-up in a future user story.

During sign-up and login, the email should be validated via simple *html based validation,* no JS at this point as it is unnecessary.

In the sign-up system, once the email is validated, the password should be validated via simple character limit requirements. If both email and password are valid, the email is checked against the user-login database to see if the email already exists. If it doesn't, then a new account is added to the database, and the user is notified of this with an appropriate message. If the email is already taken, then the user is notified of this and nothing occurs.

For the login system, the email is checked for validity, and if it is valid, the email is used to query the database. If the email exists in the database, it's password is checked against one provided by the user. If the email and password provided by the user match an equivalent pair in the database, the user is logged in and notified of this; else the user is notified of the incorrectly provided data.

User Story: Post a Question

"As a user, I want to be able to post a question once logged in to my own personal account"

The question page should have a *title* text box with a 100-character limit, and a question *body* with a 1000-character limit. No submissions will go through unless both boxes contain content. These requirements will be refined as the project moves ahead.

If the user is not logged in, the page should display the option for the user to login/sign-up before posting. If the user is logged in, he/she will be able to send a question. Once a valid question is submitted by a logged-in user, the title and body will be sent to the server side via a *post*. Along with the title and body of the question, several other auto-generated metrics (date,time,user_id) will be appended to the entry and stored in an appropriate database. A message is returned telling the user the question was successfully submitted.

User Story: Search for Question

"As a user, I want to be able to search for a question in hopes of having it already answered"

Almost every website has a search bar for the users to look up things which interest them. In the navigation bar – which will be present in every single page – there will be a search bar.

When the user enters a search query, it is sent to the server where it is matched via a custom-made algorithm with question tiles pulled from the appropriate database. All matching results will be sent back to the client via the server and listed on a single page (many in the future) where the user may click on any entry to visit its Question Forum (where question, answers, and ability to answer resides). The initial search method will be fairly simple, but will be tweaked in the future based on user requirements.

User Story: Read the Question Forum

"As a user, I want to be able to view a question and all of its answers"

The user will be able to read the entire question (including metrics such as date and time of post), along with all answers provided by other users. There are 3 different divisions on the page. From top-to-bottom, it starts with the question (title and body), followed by the answers, and finally a text box for new answers including a button to submit. The submit button is only available for logged-in users. If the user is not logged-in, there will be a message which askes the user to sign-up or sign-up via a button which redirects the user to the log-in/sign-up page

The server needs to pull all data related to the question at hand including: the title, the body, the user id, the date and time. It also needs to pull data of all answers linked to the question as objects including the following fields: the answer, the user id of the person who answered the question, the date, time and score of the answer.

There will be a text area for the user to write new answers and a submit button. The actual submission of the answer will be covered in the next user story below.

User Story: Answer a Question

"As a user, once I find an appropriate question through a search, I want to be able to answer the question"

Each question will have its own dedicated page that lists the question title and body, and all previous answers to that question below it, and at the bottom of it all, a text area of a logged-in user to submit his/her own answer. The answer should be a maximum of 1000-characters, and an empty answer will not be submitted. If the user is not logged on, there should be an appropriate redirect button to allow the user to quickly login/signup otherwise they will not be able to submit an answer. Once a valid answer is submitted, it will be stored in an appropriate server-side database with a few auto-generated fields (date, time, user_id, score…). When the data is finally stored on the server, the server returns the same page updated with the newly submitted answer.