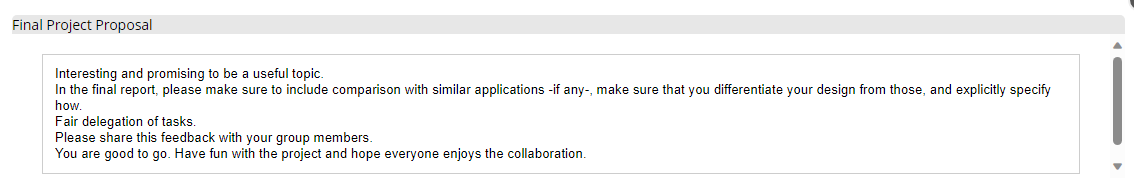
CS3354 Software Engineering

Final Project Deliverable 1

Where2Next

Group Members: Oluwadara Morakinyo, Charlie Alpert, Lennon Crow, Ariel Ong, Nuria Habib, John Kasperbauer, Joseph Farghal, Manh Ngo

1. [5 POINTS] **(Charlie)** Please attach here the Final Project draft description (that contains the instructor feedback). It is ok to include a picture of the original document. Address the feedback provided for your proposal by listing what you did / plan to do to comply with those proposed changes and or requests for additions to your Project



1. [10 POINTS] Setting up a Github repository. Please use your utdallas email accounts only for each group member.
   1. Each team member should create a GitHub account if you don’t already have one.
   2. Create a GitHub repository named 3354-teamName. (whatever your team name will be).
   3. **(Joseph - 1.3)** Add all team members, and the TA as collaborators. Our TA will post his GitHub info on EL:
      1. TA GitHub id:
      2. TA email:
   4. **(Dara - 1.4)** Make the first commit to the repository (i.e., a README file with [team name] as its content).
   5. **(Charlie - 1.5)** Make another commit including a pdf/txt/doc file named “project\_scope”. If you choose a predefined topic (one of the 4 topics described in the “Project Topic Ideas” section of this document), the contents of the file should be identical to the corresponding project in this section. If you choose other topics, the contents should follow a similar structure.
   6. Keep all your project related files in your repository as we will check them. Include the URL of your team project repository into your project deliverable 1 report.
   7. Important Note:
      1. Tasks 1.3 - 1.5 should be performed by different team members. We will check the commit history for these activities.
      2. Do not include credentials (e.g., UTD ID) in the repository.
      3. Only commits performed before the deadline will be considered. Do not forget to push your changes after you have done the work
2. [5 POINTS] **(Ariel)** Delegation of tasks: Who is doing what. If no contribution, please specify as it will help us grade each group member fairly.
3. [5 POINTS] **(Ariel & Lennon)** Which software process model is employed in the project and why. (Ch 2)
4. [15 POINTS] **(Dara)** Software Requirements including
   1. [5 POINTS] Functional requirements. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)
   2. [10 POINTS] Non-functional requirements (use all non-functional requirement types listed in Figure 4.3 - Ch 4. This means provide one non-functional requirement for each of the leaves of Figure 4.3. You can certainly make assumptions, even make up government/country based rules, requirements to be able to provide one for each. Please explicitly specify if you are considering such assumptions.)
5. [15 POINTS] **(John)** Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note than there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)
6. [15 POINTS] **(Manh)** Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)
7. [15 POINTS] **(Nuria)** Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).
8. [15 POINTS] **(Joseph)** Architectural design – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3)
   1. Model-View-Controller (MVC) pattern (similar to Figure 6.6)
   2. Layered architecture pattern (similar to Figure 6.9)
   3. Repository architecture pattern (similar to Figure 6.11)
   4. Client-server architecture pattern (similar to Figure 6.13)
   5. Pipe and filter architecture pattern (similar to Figure 6.15)