Bus project Design document

**1. Overall Design Decisions**

**1.1 System Design**

From the Start of the project I had a clear design in my head about how I wanted the project to work from a top down wide vision perspective I had a clear picture of a front end UI interacting with a singular POC backend that then queried the rest of the data. I followed the following class diagram with regards to overall system design and how I wanted the program to function.

Chart, diagram

Description automatically generated

**1.2 Priorities**

My priority for this project was to create a super clean and efficient program that handled error checking in a competent manner. The goal wasn’t to create the best looking or most efficient program but one that handled the data well and gave the user a complete and usable program that made sense. I wished to complete all 4 tasks and designed my project with this in mind.

**1.3 UX**

I decided I want to give my application the feel of a 1970-80s computer mixed with some text based games such as Rogue or dnd. I wanted the application to feel like those games I did this through simple loading screens, welcome and leaving messages and number based menu screens.

**1.4 Reading in Data**

I decided that rather then loading in all the data at the start of the application I would only load in what was necessary as it was necessary. For example, I would only load in the graph object and its associated data when the user wanted to find the shortest path between two stops. I loaded my data in through the input class which would take in the object that the data would be loaded into and then process the text files as needed. My reasoning for approaching loading the data in this way was to reduce latency between when the user starts the program and when they get to find the piece of information which is most pressing to them and their individual query. There is no need to load in the graph if the user only wishes to find out the information about a specific stop.

**2. Task 1 (Finding the Shortest Path)**

For this task I decided to tweak and re-implement my DiGraph class from assignment 2 in this module. I loaded in the vertices and their incident edges via reading input from the text files. In terms of inside the graph object I decided to store these edges and vertices in a hashmap containing a hashmap which pointed to the weight of the edges.

Diagram, engineering drawing

Description automatically generated

Storing the edges in this way allowed me a constant O(1) time to store each edge and an O(1) time to look it up when searching the graph. For the actual shortest path search I decided to implement Dijkstras Path finding algorithm with a worst-case Time Complexity of O(V +E log V) I decided to use Dijkstra as I found that Floyd Warshall was too slow initially and most of the time a user would only be searching for at most 2 or 3 paths and thus it was a waste to load in all possible routes. I also did this since there were no negative weights and there was the possibility of being cycles.

Along side task 1 I also implemented a function that would print all reachable stops from an initial stop this also uses Dijkstra for the same reasons and was only implemented as an add on so a user can check where they can go to and then can find the shortest path.

**3. Task 2 (Bus Stop Search)**

For the bus stop search I decide to store the stops as a Ternary Search Tree as it seemed to me the most logical data structure for words that store information inside of the nodes that were stop names, I stored an object called StopInfo which contained all interesting data regarding the stop and when called would return it in an aesthetic and understandable format.

Shape

Description automatically generated

This gave me a worst case O(n) time complexity for searching and inserting the data which I was happy enough with conidering the size of the data only being aprox ~ 8000 stops.

I looked at storing the Names in a table and searching for input terms but using TST made more sense as it easily alowed me to search via a prefix as well which was a neccisary task.

**4. Task 3 (Search By Time)**

For this task I decieded to store the time data in a hashmap I would store all trips in a hashmap where the time pointed to an arraylist of string arrays containing the information for all trips matching the original time. When loading them into the array I inserted them in order since for most times there were only a few entries I just looped through and found its correct possition in the array list and put it there.Then I displayed them to the user. This made the most sense as It reduced the amount of space I would need i.e not storing duplicate times for each trip and then gave me a look up complexity of O(1) for the array of stop info then an O(N) to display them all (where n is the number of trips mathicng that time). Meaning that while it took a bit longer to load in all the information it gave me a very quick look up time which was better in my opnion as the slowish load time added to the retro feel.

Besides the above approach I also looked at storing the Times in a list/table sorting them and then using binary search. The sort would have taking O(N^2) in worst case and O(NlogN) in average case and the search would take O(logN) so in total the first search would take O(N^2logN) time in the worst case which was beaten out by the above solution as well as the above solution using less active memory.

**5. Task 4 (UI)**

For the UI I made a dedicated UI class that was what interacted with the user. It is a basic text menu based UI where you input which screen you want to move to. This was the simplest way to deal with user input and it gives the application a retro/70s feel which I like. I had to add in a few waits so that the application doesn’t run to fast so that it doesn’t break the immersive feel.

**6. Timeline/ Version Control**

I used github for version control and I decided I would commit after any major change or whenever I had finished work for the day. The git hub is <https://github.com/MaxCunningham19/bus_project> and I startined coding on the 21/02 and worked on it fairly regularly til the close of the project. Below is the git history as of me writing this.Todays date (05/04/2022)Text

Description automatically generated