HSC Software Major Project

**Project Overview**

This project will be a messaging app, being able to send and receive messages between multiple clients, over the internet. This will be using a server (being run at my own house on a computer), as well as the individual clients who connect to that server, and can then interact with it to send messages. These messages will be held in permanent storage externally (on the server), but also locally cached. Users can sign in to access this data.

This project is technically heavy in order to setup all of the networking features (as well as error handling and other features to maximise compatibility and minimize issues). Outside of the technical challenges, it will support multiple ‘chats’ (with individual users), as well as group chats with multiple people.

NodeJS and Express will all be used on the server side to create the API response & data holding. A web browser (using JavaScript and HTML) will be used client-side in order to interact with the server. Firebase will be used to hold permanent data.

**Defining & Understanding the Problem**

* This solution meets the **needs** of those needing a chat interface to use to communicate with other people through the internet.
* This solution is **feasible** to achieve, through following some tutorials (in order to help get up-to-scratch with unfamiliar tools)
* The **constraints** are with time (as there are a lot of components for me to implement), as well as networking behaviours (as IP addresses may change and need to by dynamically updated & accounted for). Nothing else has major constraints on it.
* **Resources** can be found all over the internet in terms of tutorials & guides in order to help learn new features that I’m unfamiliar with when it comes to networking & databases (Firebase).
* I’m confident in JavaScript & HTML, and have found multiple courses to use to understand and implement Networking features & use Firebase. I will be proficient in using my skills to successfully bring this project together
* Yes, **many alternative solutions** do exist but are heavily cluttered and overly feature-rich; Mine will bring a more simplistic and fast way to provide a chat interface.

**Planning & Designing the Solution**

* There are many **modules** for the project; to break them down:
  + Firebase to stores conversation data & handle user-authentication (external database).
  + Client computers will use HTML, JavaScript & CSS to send & receive messages through an API.
  + A server will facilitate the connection to send messages using NodeJS & Express (for writing an API for interaction).
* All of these modules can work together to complete the product.
* A handful of **data types** will be used; including
  + Booleans
  + Numbers
  + Strings
* The main **data structures** that will be used are:
  + JSON Objects (JSON Files)
  + JS Arrays
  + JS Objects
* To solve **each piece of the puzzle**:
  + Communication between Clients & Server
    - This will be done by using API’s, with a custom backend to host all of the connections. Both clients connect to the web server, and periodically send a request to check for any new messages. If there are, then they are sent back to the client.
  + Storing Data & User Information
    - This will be done by using Firebase, a multi-purpose service that can both store message data (for later retrieval), as well as process user-authentication to retrieve relevant data.
  + Page Formatting
    - The page will need dynamic elements to adjust for user-message content, and will be set up using HTML, JavaScript & CSS where needed to make sure that it all remains under control & readable.
* **Tasks:**
  + System Designing
  + Server Code (Node (JS) /Express)
  + Client Code (HTML/JS/CSS)
  + Firebase Setup (Storage/Auth)
  + Testing (development)
  + Testing (production/ ‘real life’)
  + Final Deployment
  + Documentation

**Implementing the Solution**

* Discuss the **interfaces** needed.
  + Web GUI
    - As the web is the front-end interface for all users, this has the most important when it comes to design & readability. This will be done appropriately using CSS to ensure elements are given space as needed, and arranged logically.
    - The backend API will be using a CLI interface for any use cases, although is only intended to be used within development reason and should not concern the user.
* What **error detection** will be used?
  + To detect errors throughout the implementation process, lots of safeguards will be used to minimize & correct potential errors. This includes:
    - Automated Spelling/Syntax Checking
    - Breakpoints for debugging
    - Watching variables whilst running
    - Single-line stepping
* What **backup/version control** will be used?
  + GitHub will be primarily used, as it is the most flexible and feature-rich solution for version control of code. It allows for pushing & pulling changes, viewing changes between versions, all in the cloud.
  + OneDrive will be used as a fallback, holding copies of files in case of corruption or other errors with my local version as well as GitHub’s version.

**Testing & Evaluating the Solution**

* How will you **test** your software to ensure it’s reliability and quality?
  + Heavy testing will be done using test data, to control and evaluate how it works within controlled conditions, using Visual Studio Code’s ‘debugging’ process, as well as with Insomnia ([insomnia.rest](https://docs.insomnia.rest/insomnia/get-started#:~:text=Insomnia%20is%20an%20open%20source,code%20generation%2C%20and%20environment%20variables.)) to help develop & test the API.
  + In standard use, error handling will be implemented and used where possible in order to catch and deal with potential errors, expected almost exclusively with networking features, in order to avoid logic errors.
* How will you **evaluate** your design?
  + Design will be evaluated on a functionality basis. If it works successfully with minimal error, that is how we define a functional product. Any parts of the product that are available must be fully functional, as that was one of the main goals of the project; a simplistic messenger that simply works.

[gantt chart.png](https://newington-my.sharepoint.com/:i:/g/personal/59436_newington_nsw_edu_au/EcywC5eoIU9Cn8l3xSRlEhcB4e8UdMJkUGgZZnQ4UCjFFg?e=q0KShh) (download)

