2. Functional design for the Mars Rover(brief) and for each module ()

The command module is responsible for the web application that displays rover status and provides a manual control interface. Using a cloud AWS server written in Express.js based on Node.js framework as the backend, the command system retrieves status information about other modules and sends back optional manual commands through http requests with the control module, including the distance travelled by the rover, positions of the rover, aliens, towers, and underground infrastructures discovered. The coordinates information is saved by the server in a database through SQLite. The rover and alien information are displayed on a frontend webpage, which is written in React.js, in the form of a ‘status metre’ and a live map. The website also includes a control panel with buttons for manual control.

Command

1. Technical details of subsystems (related theories)

Diagram

Description automatically generated

* + Communication with HTTP

The Command subsystem is concerned with two main communications –between server and ESP32, server and webpage client. HTTP is used for both communications through the methods of GET and POST, where the clients send requests to retrieve information or upload information from and to the server respectively, while the server responds to the requests. Although there are other commonly used protocols such as TCP, web socket, socket.io etc, (more detailed comparison between HTTP and web socket seen in the Control section) HTTP is chosen concerning its reliability due to it is based on connection-oriented TCP, also allowing the specific endpoints to construct independent connections to different components in the frontend, and to keep the inter-system and intra-system communication protocol consistent.

* + Backend
    - Express and Node

Node.js is a runtime environment for server-side JavaScript and it is used for its high performance and scalability. Being a framework based on Node, Express.js is used to achieve more functionality and easier implementation of HTTP requests through built-in http, provided routing and more powerful methods of ‘res’ and ‘req’ than Node such as ‘res.send’.

* + - AWS EC2 server

The server is run on an EC2 instance as a cloud service as it allows for a remote host that does not restrict the physical location of clients. It also enables data backups of the rover exploration information in a database which is accessible to all users with the ppk key.

* + - SQLite

SQLite is a file based DBSM library that creates database files. It sits alongside with the server in EC2, and the server stores the coordinates data posted from the ESP32 in 3 tables and reads the tables to send data to the website client. It is chosen over another database since it does not require configuration, and it is lightweight and simple to implement using a Node.js module of sqlite3. These properties serve for the purpose of basic data storage for an IoT system of the rover as it handles large quantity of data better than a server-side buffer variable and keeps a retraceable record when the program is ended.

* + Frontend

(code tree)

* + - React and CSS

The frontend website is built with React.js using hooks and styled with CSS, reusable objects as an imported file. Hooks are functions that utilise React State and lifecycle features through ‘useState’ and ‘useEffect’. Hooks are used instead of React classes as the multiple lifecycle functions in classes are replaced and simplified with the Effect Hook to perform side effects of state changes, serving for event handler, re-rendering and executing HTTP requests. The main event handler is the control panel which includes a toggle mode button, 4 direction buttons and a stop button, which all have keyboard input alternatives. HTTP requests to and from the server is implemented with ‘fetch (URL, {configuration})’ followed by methods of Promise object of ‘.then’ and ‘.catch’ to handle asynchronous code in a chain order when the Promises are resolved and rejected respectively.

* + - Libraries

React Konva is a library based on the Konva framework to draw canvas on the webpage. It is used in rendering a fixed size scaled live map where given coordinates; basic shapes are drawn with Konva on the screen to represent aliens and its reliability range with coloured points surrounded by a faded circle; towers as black circles; infrastructure as purple squares; rover path as a line with a pointer for current rover position.

Grid is a layout library used to divide the screen into sections for a clean, panel-like look.

1. Practical examination of individual modules

Diagram

Description automatically generated

1. Evaluation

Overall, the command module is crucially well-finished since the functionality correctly fits its purpose of reliable data transmission to and from the ESP32, displaying a map with correct positions and well-defined labels given inputs, and allows user to input manual control with self-explanatory arrow buttons.

The design abides with the REST architecture for web server. The server is platform independent as both the ESP and React.js website can access this API; the URI has resource-oriented endpoints for different clients and purposes; the server side contains a central database that reduces the size of variables in the server program and provides backup; the information exchange with clients is through JSON objects.

Limitations:

The server has no security system where can cause confusion to the server if several clients attempting to connect to the same URI endpoints. HTTP sends a request for building connection for every communication round which has a header of 700-800 bytes on average which makes transmission slow and expensive.

Improvement:

* + Use secure data transmission protocol of SSH instead of all-TCP.
  + Login System (Andy加这儿)
  + Use faster and less expensive communication, requires more research and experimentation due to the amount of information collected on mars and the cost of transmission
  + Page design can be improved to be more engaging by adding animation and user interactive features given more time.

[1] GeeksforGeeks. 2022. *What is web socket and how it is different from the HTTP? - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/> [Accessed 11 June 2022].

[2] GeeksforGeeks. 2022. *Node.js vs Express.js - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/node-js-vs-express-js/> [Accessed 13 June 2022].

[3] Tutorialspoint.com. 2022. *SQLite - Overview*. [online] Available at: <https://www.tutorialspoint.com/sqlite/sqlite\_overview.htm> [Accessed 14 June 2022].

[4] "How to create a React frontend and a Node/Express backend and connect them", *freeCodeCamp.org*, 2022. [Online]. Available: https://www.freecodecamp.org/news/create-a-react-frontend-a-node-express-backend-and-connect-them-together-c5798926047c/. [Accessed: 11- Jun- 2022].

[5] "react-konva", *npm*, 2022. [Online]. Available: https://www.npmjs.com/package/react-konva. [Accessed: 20- Jun- 2022].

[6] "React Grid component - Material UI", *Mui.com*, 2022. [Online]. Available: https://mui.com/material-ui/react-grid/. [Accessed: 20- Jun- 2022].

[7] "Web API design best practices - Azure Architecture Center", *Docs.microsoft.com*, 2022. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design. [Accessed: 19- Jun- 2022].