PROJECT REPORT

Indian Space Research Organization

Telemetry, Tracking and Command Network

Plot No. 12 & 13, 3rd Main, 2nd Phase, Peenya Industrial Area, Bengaluru - 560058

NEXT GENERATION MONITORING AND CONTROL SYSTEM WEB APPLICATION

Submitted in partial fulfilment for the award of the degree

Bachelor of Technology

Computer Science Engineering

Dated: 28.04.2023

Project Guide:
Mrs. M. Parvathi Devi
Manager,
Software Development Division, ISTRAC

Submitted By:
Yoghesh Kumar R
Shakthi Aravind B

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project.

We express our profound thanks to **Mr. B. Sankar Madaswamy, Manager HRD, ISTRAC** for giving us an opportunity to do the project work and allot us a project of our interest at ISTRAC.

We respect and thank our project guide **Mrs. M. Parvathi Devi, Manager, Software Development Division, ISTRAC** who took keen interest in our project work and guided us all along. Also, we would like to extend our sincere esteems to **Mrs. S. Santhalakshmi, Mrs. T. Haripriya, Miss Sonal Shekhawat, Mr. Divyang Arora, Mr. Sajith Menon** and all staff in the laboratory for their timely support.

We are immensely grateful to our internal project guide, **Dr. Suriya Praba T.**, **Assistant Professor-II, School of Computing, SASTRA Deemed University** for her constant support, erudite feedback throughout the project.

Lastly, we would like to extend our heartfelt appreciation to our family for their continuous support and encouragement on whatever path and decision we took and for always being with us in different phases of our life.

ABSTRACT

The complex system of monitoring satellites involves the use of ground station terminals, which are strategically located in close proximity to antennas. These terminals are responsible for managing and controlling various equipment related to the antenna and the facilities responsible for encoding and decoding antenna signals. To facilitate this process, a software system known as the Monitoring and Control System is utilized. However, this software is an outdated, software-based application that is still being used in ground stations. Recognizing the need for a more modern solution, our project aims to develop a web-based application with the same functionality and feel as the current system. This web-based application is being designed with a focus on enhancing the existing architecture by moving away from the one-way cluster-oriented model and introducing a mesh architecture that connects every cluster of terminals to every other cluster.

One of the primary advantages of this new architecture is that it enables a more flexible and scalable system, which is better suited to managing the ever-increasing volume of data being transmitted by satellites. The new system will also allow for better communication between the different clusters, thereby enabling a more coordinated and efficient approach to managing the equipment.

To achieve our goals, this full-stack project will leverage a range of cutting-edge technologies, including React.js for the front-end, Node.js for middleware, Redis for session storage, and MySQL for database management. Additionally, the Real-Time Control software will be employed to provide real-time monitoring and control data from the equipment.

This project is an exciting opportunity to revolutionize the way in which we monitor and control satellites, improving the efficiency and reliability of this vital technology. With the increasing importance of satellites in our everyday lives, a more modern and efficient approach to monitoring and control is essential to ensure that we can continue to rely on this technology for years to come.

TABLE OF CONTENTS

Abbreviation

API	Application Programming Interface
CDT	Countdown time
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
ISRO	Indian Space Research Organisation
ISTRAC	ISRO Telemetry, Tracking and Command Network
JSON	JavaScript Object Notation
OWASP	Open Worldwide Application Security Project
QLD	Quick look display
RTC	Real time control
TCP/IP	Transmission Control Protocol/Internet Protocol
TTC	Telemetry, Tracking and Command
URL	Uniform Resource Locator
WSTG	Web Security Testing Guide
XML	Extensible Markup Language

ISTRAC

The Indian Space Research Organisation (ISRO), over the years, has established a comprehensive global network of ground stations to provide Telemetry, Tracking and Command (TTC) support to satellite and launch vehicle missions. These facilities are grouped under ISRO Telemetry, Tracking and Command Network (ISTRAC). ISTRAC is entrusted with major responsibility to provide tracking support for all the satellite and launch vehicle missions of ISRO.

Objective:

- Carrying out mission operations of all operational remote sensing and scientific satellites, providing TTC services from launch vehicle lift-off till injection of satellite into orbit and to estimate its preliminary orbit in space and hardware and software development activities that enhance the capabilities of ISTRAC for providing flawless TTC and Mission Operation services.
- Telemetry data acquisition support and ISRO launch vehicle missions and lift-off until satellite acquisition and down range tracking support for monitoring and determining the satellite injection parameters.
- Coordinating between spacecraft and launch vehicle teams supporting ground stations right from planning till the completion of missions for national and international satellite missions.

Towards these objectives, ISTRAC has established a network of ground stations at Bengaluru, Lucknow, Mauritius, Sriharikota, Port Blair, Thiruvananthapuram, Brunei, Biak (Indonesia) and the Deep Space Network Stations.

In keeping with its long-established TTC support responsibility, ISTRAC has also been mandated to provide space operations support for Deep Space Missions of ISRO, undertake development of radar systems for launch vehicle tracking and meteorological applications, establish and operationalize the ground segment for Indian Regional Navigational Satellite System, provide Search & Rescue and Disaster Management Services and support space based service like telemedicine, Village Resource Centre and tele-education.

1. Purpose

To build a dynamic and robust web application that enables its user to remotely monitor and control the various ground station equipment.

2. Functional Requirements

Remote access

Allow authorised users to interact with ground station equipment from anywhere within the campus and perform TTC operations using a web browser thus reducing the need for human intervention and improving operational efficiency.

Authentication

Appropriate authentication mechanisms must be in place to validate user's identity and determine their permissions based on their role and allow access to resources.

Access control

Proper access control policy must be defined which gives details about the permissions for a given user including the clusters to which the user has access and the operations that can be performed by the user i.e only monitoring or both monitoring and control.

Monitoring and Control

Check the permissions for the authorised user and allow them to perform actions accordingly.
Monitoring allows the user to look at the real time values of the various parameters of any equipment present in the ground stations.
Control allows the user to change the values of the editable parameters of an equipment and also makes sure whether the user has entered an appropriate value for the particular parameter.

Track sessions

Whenever a session is scheduled the application must automatically load the session details 3 minutes prior to the start of the session and continue updating the details throughout the duration of the session.

Add/edit schedules

Allow authorised users to add upcoming session details or edit an existing session from the list of sessions read from the schedule file. Check whether the details entered for the new session or edited fields of the existing session have appropriate values.

3. Non-functional Requirements

Performance
The application must be responsive and be able to handle concurrent users efficiently.

Usability
The application must be user friendly with a GUI similar to the GUI of the actual equipment at the ground station and allow the user to navigate across the various pages and menus seamlessly.

Reliability
The application must be robust and function correctly and consistently under normal and abnormal conditions. Should be able to detect and recover from failures.

Scalability
The application must be scalable and be able to accommodate new equipment, terminals and clusters without the need for any major changes in the code.

Data integrity
The application must ensure the integrity of data in rest and transit by enforcing access control policies and proper validation mechanisms.

Maintainability
The code should be documented properly and all the static values should be declared in separate XML files and should be imported into the main code to make it easier for future maintenance and updates. The application must be modular and allow for easy updates and changes to specific components.

Portability
The application must be built in a way that it can be easily transferred across different hardware and software environments.

Security
Since the application deals with sensitive data related to various ground station equipment it is a must to follow strict security guidelines and protect data from unauthorised access. The system must be able to detect and recover from security breaches.

4. Tools/software Used For Development

React

React is used to develop the GUI for the web application.

React is an open-source front-end JavaScript library for building user interfaces. It follows a component-based architecture, i.e. each part of the UI is encapsulated in a separate component. This makes it easy to reuse code and build complex UIs with a modular approach. React uses a virtual DOM, which is a lightweight representation of the actual DOM. This allows for efficient updating of the UI by only updating the parts that have changed, without having to re-render the entire page. Because of its efficient use of the virtual DOM and its optimised rendering techniques, react is known for its fast performance, even for large and complex applications.

Node.js

Node.js is an open-source server-side runtime environment that allows to run JavaScript on the server-side, outside of the web browser. Node is designed to be event-driven, non-blocking and it provides a rich set of built-in modules, such as HTTP, HTTPS, File System, and Stream which makes it well-suited for building scalable, high-performance web applications and APIs.

Express.js

Express.js is used to build the RESTful API middleware along with Node.js.

Express.js provides a minimalist and flexible approach for building web applications, allowing developers to create a customized architecture that suits their specific needs. It provides a robust set of features and middleware for handling HTTP requests, routing, error handling, and more.

MySQL

MySQL database is used to store cluster, terminal, user login and access control details.

MySQL is an open-source relational database management system (RDBMS) that is widely used for building web applications. It uses Structured Query Language (SQL) to manage and manipulate data. It stores data in tables with a predefined structure. MySQL supports a wide range of features, such as transactions, foreign key constraints, stored procedures, triggers, and views. It is also highly scalable, allowing developers to manage large datasets with ease.

Redis

Redis is used as the session store for Express.js.

Redis (REmote DIctionary Server) is an open-source, in-memory data structure store that can be used as a database, cache, and message broker. Redis stores data in memory, which makes it very fast for read and write operations. It supports a wide range of data structures such as strings, hashes, lists, sets, sorted sets, and provides atomic operations for manipulating the data structures. Redis can be used as a session storage to store user session data, such as login credentials and user preferences. This allows web applications to scale horizontally by distributing user sessions across multiple servers.

HTML

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages and consists of a set of tags and attributes that define the structure and content of a web page. HTML is designed to be used in conjunction with other web technologies, such as CSS and JavaScript.

CSS

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of HTML and other markup languages and controls the visual appearance of a web page. CSS provides a wide range of features for styling and formatting content, including color and background properties, typography properties, layout properties, and animation and transition properties thus making web pages visually appealing.

Javascript

JavaScript is a high-level programming language used to create dynamic and interactive web content. JavaScript is used to manipulate the Document Object Model (DOM) of a web page, which represents the HTML elements and other content on a web page. It can be used to dynamically update the content of a web page, respond to user events, validate form data, and interact with web APIs.

Tool/software	**Version**
React	17.0.2
Node.js	14.10.1
npm (Node Package Manager)	6.14.8
Express.js	4.17.1
MySQL	8.0.26
Redis	5.0.3
mkcert	1.4.4

5. Web Modules Developed

This web application comprises several independent components, each serving a specific purpose. These components are referred to as "modules" within our framework and include middleware and backend servers. To provide a comprehensive understanding of our system, the detailed descriptions of each module has been mentioned below, outlining their individual methodologies and functionalities.

Webpages:

- Login
- Cluster
- Terminals
- **Main Page**
  - Home
  - Session
  - Schedule
  - Config
  - Report
  - Logs

5.1 LOGIN:

The purpose of the page in question is to serve as a login page, as its name suggests. It is designed to authenticate the user's session by validating the details entered by the user with the backend system. Once the entered details are validated, the page fetches the necessary information to establish a user session.

Functionality:

- Captcha fetching
- Password hashing
- Cookie setting

5.1.1 Captcha Fetching:

When a user attempts to access a website, it is essential to ensure that they are a human rather than a machine or a malicious bot. This is where captchas come in. Captchas are used to distinguish human users from automated scripts or bots. A captcha is a type of challenge-response test used to determine whether the user is a human or not. One common way to implement a captcha is to display an image with a series of distorted characters that the user must type correctly into a form.

To achieve this, the website sends a request to the middleware, which generates a unique captcha ID that is associated with the user's computer. This captcha ID is a randomly generated token that helps the middleware identify which captcha belongs to which computer. For example, when a single computer requests a captcha, there is no issue because the middleware knows which computer replied since there is only one. However, when multiple clients are involved, it becomes more challenging to determine the mapping from user to captcha.

One way to solve this problem is to issue a cookie for each client and validate it, but this may lead to other issues such as maintaining a session unnecessarily without a login. Instead, the middleware generates a unique captcha ID for each client, which is then used to request the captcha image from the node. The node stores the captcha image in its memory for future authentication.

Once the captcha ID has been created and retrieved from the node, the next step is to fetch the captcha image. To achieve this, the captcha ID is used to request the node, which returns the corresponding captcha image. Finally, the received image is displayed using the image tag, located just above the login button.

This process helps to ensure that only human users can access the website, providing an additional layer of security against automated attacks and malicious bots. By generating a unique captcha ID for each client, the website can verify that the user is a human without relying on cookies or other less secure methods.

Overall, the use of captchas is an essential component of modern web security and helps to protect against a wide range of online threats.

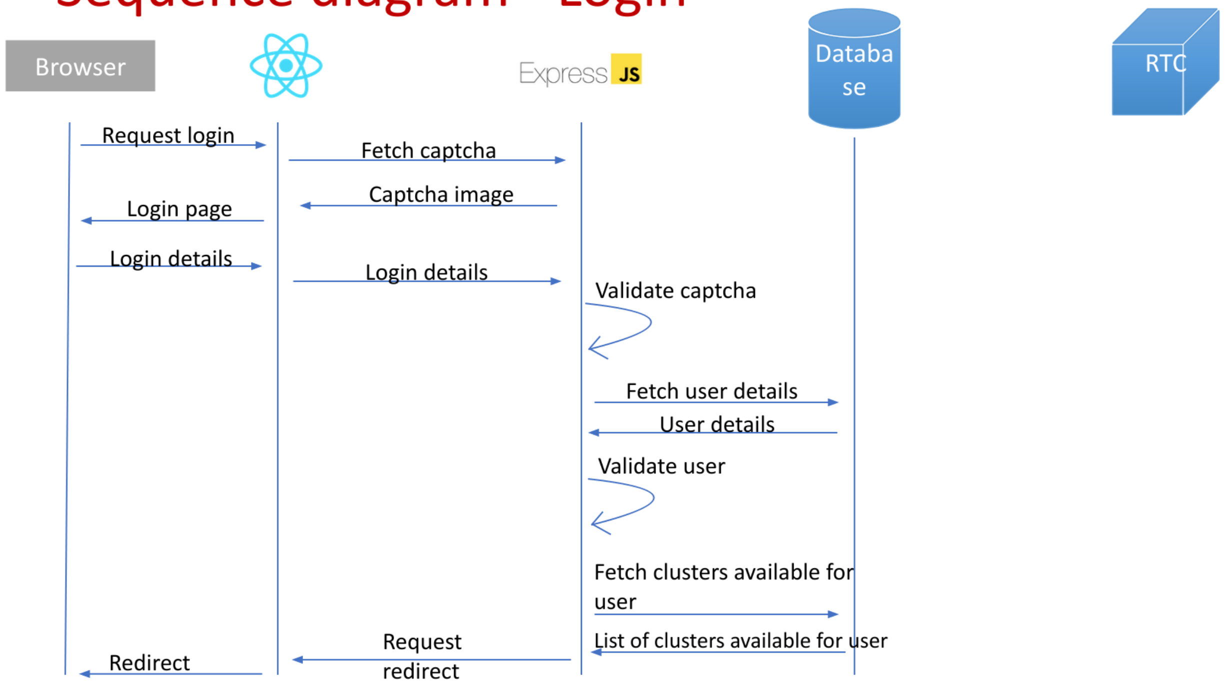
5.1.2 Password Hashing:

While transmitting passwords over a secure HTTPS network a good level of protection is assured but it is still considered a best practice to hash passwords. Additional measures are taken to ensure that the hashed password is unique to each individual. In particular, the passwords stored in the database are further secured using salting techniques to protect against unauthorized access. In the event of a data breach, this added layer of security ensures that all sensitive information remains safeguarded.

5.1.3 Cookie Setting:

This page is responsible for setting the cookie which validates our entire session.The cookie is set by the first response received after the validation is complete which cookie is now sent with all the requests to the middleware

Sequence diagram - Login



5.2 CLUSTER:

The functionality of this page is relatively straightforward, as it serves as a selection page for choosing clusters. Once the user selects the appropriate cluster, various operations may occur based on the user's location and the selected cluster. Further details regarding the specific operations are elaborated upon in the upcoming passages.

Functionalities:

- Display the list of authorized clusters in a map fashioned manner
- Make the session cluster appropriate

5.2.1 Display:

The cluster selection page is responsible for displaying a list of authorized clusters that users can select to initiate various operations within our web application. To achieve this, the page receives a cluster list from the middleware, which obtains it from the backend. The list is transmitted to the front-end as a set of distinct buttons, each corresponding to a particular cluster. These buttons are positioned on a world map background that highlights the ISRO ground station area, with each button located in the appropriate location determined by a local dictionary object. Through this intuitive and visually appealing interface, users can easily navigate through the various functionalities of the application.

5.2.2 Make the session cluster appropriate:

Once a cluster is selected on the cluster selection page, it is used to establish a connection with the corresponding backend server. Prior to this selection, all pages are connected to the default local cluster. However, this decision is subject to change based on the cluster selected on this page. The selected cluster will remain active for all subsequent pages, and the user will only need to return to this page to make changes to their cluster choice. By enabling users to select their desired cluster, we aim to provide a seamless and customized experience, tailored to their specific needs and preferences.

5.3 TERMINALS:

Upon selecting the desired cluster on the previous page, users are redirected to a page that displays all the available terminals within that cluster. The page enables users to easily navigate to any of the listed terminals, and connects to the appropriate middleware for the selected cluster. This middleware may not necessarily be the local one, depending on the cluster selected on the previous page. By providing users with a straightforward and convenient interface for selecting terminals, we aim to enhance their overall experience with our application.

Functionalities:

- Retrieve Terminal data
- Display the Terminals QLD and provide hyperlink

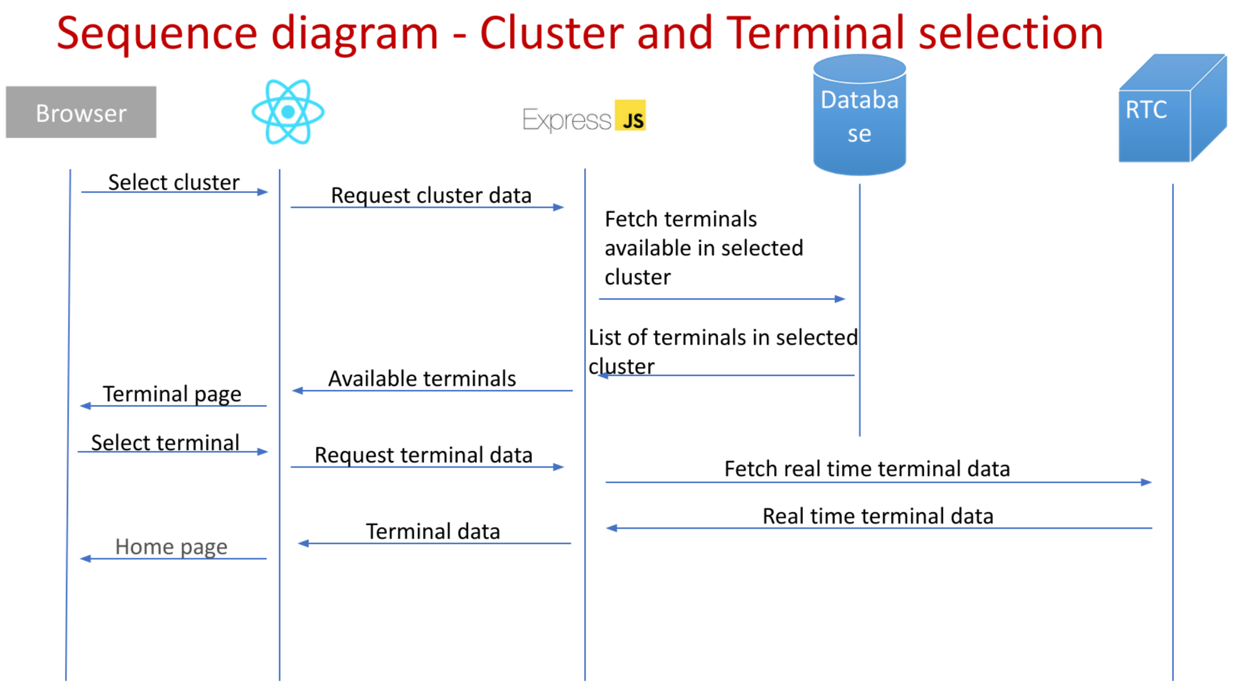
5.3.1 Retrieve Terminal Data:

The main purpose of this page is to retrieve Quick Look information related to the selected terminal from its corresponding RTC over the network. Currently, this is accomplished through an emulation process using a single RTC, but we plan to expand the general scope in the future. The page retrieves the necessary IP addresses of the terminal RTCs from the backend, which enables it to establish the appropriate connections for retrieving the relevant information. By providing users with up-to-date and accurate information on the selected terminal, we aim to enhance their overall experience and ensure that they can access the functionalities they need with ease.

5.3.2 Display the terminals QLD and provide hyperlink:

Quick Look Display (QLD) consists of information that is considered most relevant by the users. This data is mapped to each terminal as cards and displayed in an orderly fashion. It should be noted that this data may not be uniform across all the terminals, even if they belong to the same cluster. Therefore, the mapping of data is dynamic, based on this assumption.

The terminal selection made on this page is crucial, as it determines the terminal-specific data displayed on future pages, instead of being cluster-specific. The selection is sent as a URL query, and subsequent pages utilize the selected terminal from the URL as their current terminal. By providing users with relevant and accurate information specific to their chosen terminal, we aim to enhance their experience and ensure that they can efficiently access the functionalities they need.



5.4 MAIN PAGE:

The Main Page is not a singular page, but rather an umbrella term used to describe multiple pages that fall under the same category of a selected terminal. These pages are where all the important information and sensitive operations, both in monitoring and control categories, take place. The pages that fall under this category are:

- Home
- Session
- Schedule
- Config
- Report
- Logs

5.4.1 Menubar (component) :

The Menu Bar is a commonly used React component that is included in all the Main page modules. It serves as a navigation bar, containing hyperlinks to switch between the different Main pages. Additionally, it provides a logout button that allows the user to log out of the site.

5.4.2 Home:

The Main Page is the default and most important page during an idle time, which is referred to when the antenna is currently inactive and does not expect a pass. This page consists of a current and upcoming session, the server statuses, upcoming session countdown time (CDT), an information panel for priority messages, and most importantly, the equipment that is currently populated in the terminal, which is updated from the backend. Additionally, there is a common menu of functionalities for all the equipment.

The equipment is populated as buttons on the home page, and the functionality of each button may vary depending on the number of components within that button and the user-friendly visual. For TTCP equipment, a panel containing all the components is opened when the button is clicked. Each component can individually open different dialog boxes, which are dynamically populated by input from the backend. For TTCP-style dialog boxes, which we refer to as the LARGE category, an extra level of dialog box is always opened before the actual parameters.

We have the option to enable XML tabs, which helps reduce clutter on the screen by a lot, and closely resembles the actual hardware panel for ease of use. For STC and ACU-style dialog boxes, we try to replicate the actual software that has been the norm for many years. This is a challenging task, as we need to code in a lot of special features that are approximated as closely as possible. Since we are not mapping the XML features directly, we have less use of the XML skeletons here, and they can be ignored if needed. Instead, we use a middleware XML or a frontend JSON, which are used to map the data we receive from the RTC to the GUI.

5.4.3 Session:

This interface selectively displays only those components that are relevant to the current session and showcases the data flow from the antenna to the components, which is determined by the data supplied by the Real-Time Control (RTC). The primary objective is to ensure that the session specific components are aptly displayed to the user and main flow seamlessly switches to the Redundant chain in the event of any issues with the Prime chain.

Functionalities:

- Automatic Switch to Active Session
- Real-time Monitoring of Active Session
- Equipment Control

5.4.3.1 Automatic Switch To Active Session:

The interface automatically transitions from the Home page to the Active Session page when an active session is detected. But the user is also given the option to navigate back to the Home page using Menubar.

5.4.3.2 Real-time Monitoring Of Active Session:

The interface enables real-time monitoring of the active session by displaying only the pertinent components, allowing the user to monitor the data flow and identify any issues that may arise during the session.

5.4.3.3 Equipment Control:

The page includes equipment with a small QLD, which is useful during sessions. Clicking on these icons initiates dialog boxes that grant access to the components' information and control.

Overall, the interface is designed to provide the user with an intuitive and streamlined experience during an active satellite session, with an emphasis on real-time monitoring and easy access to equipment control.

5.4.4 Schedule:

This page focuses solely on schedules and utilizes authorized data from the middleware in the form of JSON to facilitate visually appealing functions and execute appropriate operations based on the schedule.

Functionalities:

- Fetch Authorized Data
- Display the Data Optimally
- Allow Authorized Edits

5.4.4.1 Fetch Authorized Data:

Upon loading, the page initiates a request to retrieve data from the middleware. The middleware then verifies the user's permissions and fulfills the request by sending the data in JSON format if the user is authorized to access it.

5.4.4.2 Display The Data Optimally:

The retrieved data is presented in the form of tables, which can be modified by the user using the control buttons located at the bottom of the screen to adjust the table's length.

By using the control buttons the user can:

- Alter number of rows
- Navigate to different pages of the table

To display the received data as a table, we first take the array from the received data and map it into the table tag in HTML. The data includes a **"S.No."**

column based on the order of the array, which is used to identify the data row while performing operations on it.

We offer functions for users to sort the data based on their preferences. This is achieved by utilizing the headers as keys upon which we sort the data.

Filters are also available to enable users to filter the table data. Currently, we have filters for Satellites and Terminals. These filters can accept multiple parameters by adding a comma "," between the keywords.

5.4.4.3 Allow Authorized Edits:

Authorized users have the capability to modify the schedule data, which is then reflected in the schedule file on the middleware once we verify the relevant parameters to be edited. However, additional verifications are required for such edits beyond just authorization. For instance, even authorized users are not allowed to make edits to schedules within three minutes of a pass.

After the data is verified and user authorization is completed, the middleware updates the local JSON data and the file from which the data was fetched. This ensures that the changes made to the schedule are saved and can be viewed by all authorized users.

6. Node Modules:

We classify the specific functionalities that were developed in NodeJS for this project in this chapter, they can be broadly classified into:

- Captcha handling
- Backend Communication
- Session Handling
- RTC Fetch
- Node Switching
- Logout

6.1. Captcha Handling:

The use of captchas is necessary to prevent brute force attacks by bots. In this project, we have implemented our own captcha server which deploys a simple text-to-image based captcha. This captcha system helps to ensure that only human users are able to access the system and prevent any malicious attempts to gain unauthorized access.

Working:

In preparation for implementing this functionality, it is important to recognize and familiarize ourselves with two terms that are primarily utilized to address concurrent multi-user scenarios on our server. These terms are:

CaptchaID (capID):

It is a token corresponding to a captcha Text.

Captcha IMG:

It is an image generated using corresponding captcha Text and stored with capID as the key.

The operation of our captcha server can be summarized in the following steps:

1. The Login page fetches the captcha ID from the middleware upon loading. The captcha ID is generated by a local function within the program and is a random string token.

2. Once the captcha ID is generated, we locally authenticate the captcha ID to prevent anyone from overloading the server by repeatedly requesting captcha images. This is achieved by requiring users to visit the root URL first to get the captcha ID, and then visit another URL **'/loginimg'** to retrieve the captcha image. This two-step process helps reduce the risk of users having the same captcha ID, and prevents ID tampering.

3. The middleware first checks if the user is authorized to access the captcha image by verifying the captcha ID. If authorized, the middleware checks if a captcha image already exists for the given captcha ID. If it exists, the middleware retrieves the text from the existing captcha and sends an image created with that text to the user. If it doesn't exist, the middleware generates a new random string of length 5, creates a new captcha image with that string, and sends the image to the user.

4. Verifying the captcha is an important step to ensure that the user is not a bot and is a human attempting to log in. Once the user submits their credentials, the server first checks if the captcha ID and the captcha text entered by the user match with the ones stored in the server's database. If they don't match, the server rejects the login attempt and prompts the user to enter the correct captcha.

5. If the captcha is successfully verified, the server then proceeds to check the user's credentials, i.e., username and password. If the credentials are correct, the server generates an authentication token for the user, which can be used to access authorized resources on the server. The authentication token is typically a long string of characters that is unique to each user and is used to verify their identity in subsequent requests.

6. It's important to clear the captcha immediately after a successful login attempt to prevent someone from reusing the same captcha for multiple login attempts. This helps to ensure the security of the login process. If the captcha does not match, the local cache will eventually timeout and delete the captcha, freeing up that ID for future use.

In summary, the captcha server's design is relatively simple and straightforward, and it involves only two GET requests to complete the entire process.

6.2. Backend Communication:

For our project we use MySQL server for our backend, this server is hosted locally and the credentials of this server's user are kept in the middleware and hidden from the user.

We use the backend for three primary requests:

- Username and Password validation
- User Cluster Authorization
- Node IP Fetching

6.2.1. Username And Password Validation:

This action happens during the process of a user Login and we perform this after we verify our captcha. Once the captcha has been verified we query the database looking for the provided username in the request and we fetch the password of the first result we get. This way we can prevent the user from performing any SQL injection attacks. We then verify the user by checking the hashed password he provided with the salted and hashed password we stored in the database. To perform this task we utilize the bcryptjs module's compareSync function which salts and compares two strings in a synchronized manner. The reason we use salting is to prevent user passwords from leaking even during a data breach, more on this regard in later passages.

6.2.2. User Cluster Authorization:

This action takes place once the user moves on from the **Login** page and into the **Cluster** page. For the response in this page, given that the user is verified, we provide the user with the list of authorized clusters from the back end.

As an additional part during the user moves from **Cluster** to the **Terminals** page we also authorize and provide the user with the list of terminals and proposedly we provide each terminal's QLD display IP address from which we fetch their QLD data. But at the moment we fetch the QLDs from the Node itself.

This action is relevant because different users may have access to different clusters and it is up to the middleware to verify each user and their corresponding clusters.

Once we authorize the clusters we store this information in the User's session and everytime the user requests a cluster or terminal specific request we respond only after verifying this parameter.

6.2.3. Node IP Fetching:

This is a special feature that is used to transfer the user session from one Node to another. It is done during the time when the user who originates from the local Node wishes to view a different cluster rather than his local one. To achieve this we must not only send the user the IP address of the remote cluster but also send his current session details and authorize it with the remote Node, we elaborate on this later.

Once we transfer the session we retrieve the remote Node IPs from the database and send it to the user for future requests.

Note that this IP fetching only occurs when the user is authorized for it and this is a proposed feature yet to be implemented.

6.3. Session Handling:

Session here refers to the user session. For our middleware the session is handled by a module called "express-session". This module is simple to use and very resilient. Whenever we mention that we store the data into the session we mention the session created by this module.

Upon configuring the module we are asked to provide some parameters for the session, they are:

- **resave:false ->** *Forces the session to be saved back to the session store, even if the session was never modified during the request.*
- **saveUninitialized: false ->** *Forces a session that is "uninitialized" to be saved to the store.*
- **secret: 'this is my secret' ->** *This is the secret used to sign the session ID cookie.*
- **cookie**
  - **maxAge: 1000 * 60 * 15 ->** Specifies the number (in milliseconds) to use when calculating the **'Expires Set-Cookie'** attribute.
  - **sameSite: 'strict' ->** Specifies the boolean or string to be the value for the SameSite Set-Cookie attribute.
  - **httpOnly:true ->** Cookies will only be sent on http requests.
  - **secure:true ->** Cookie will only be sent on secure requests.

For the storing of information within the session we simply need to use the req.session object which will bypass and act as a local variable. By default express-session uses MemoryStorage which is volatile so for our project we use the Redis persistent storage which we assign to Express by giving **Redis-Client** into the **Store** property.

6.4. RTC Fetch:

This occurs after reaching the **Main** pages. RTC fetch is a TCP/IP socket based action that repeatedly fetches the data from the RTC server. This data is then parsed into JSON format, then formatted and later used by different Node middlewares.

This function upon performing locally does not need any other external help in parsing the file, but once we run this module over LAN we see that the receiving packets are not whole and we cannot parse them as they arrive.

In order to address this issue, we incorporate headers and footers to group the data into packets, and we also include the packet size within each packet. Our TCP/IP client receives the data in whole or in small chunks, and writes it to a local file called **"output.bin"**.

As soon as the footer is written to the file, indicating the end of the packet, the file is read and the data is parsed. For each new packet, the file is erased and the process starts anew. Once the footer is read, the parsed data is stored in a local variable for use by the middleware modules.

6.5. Node Switching:

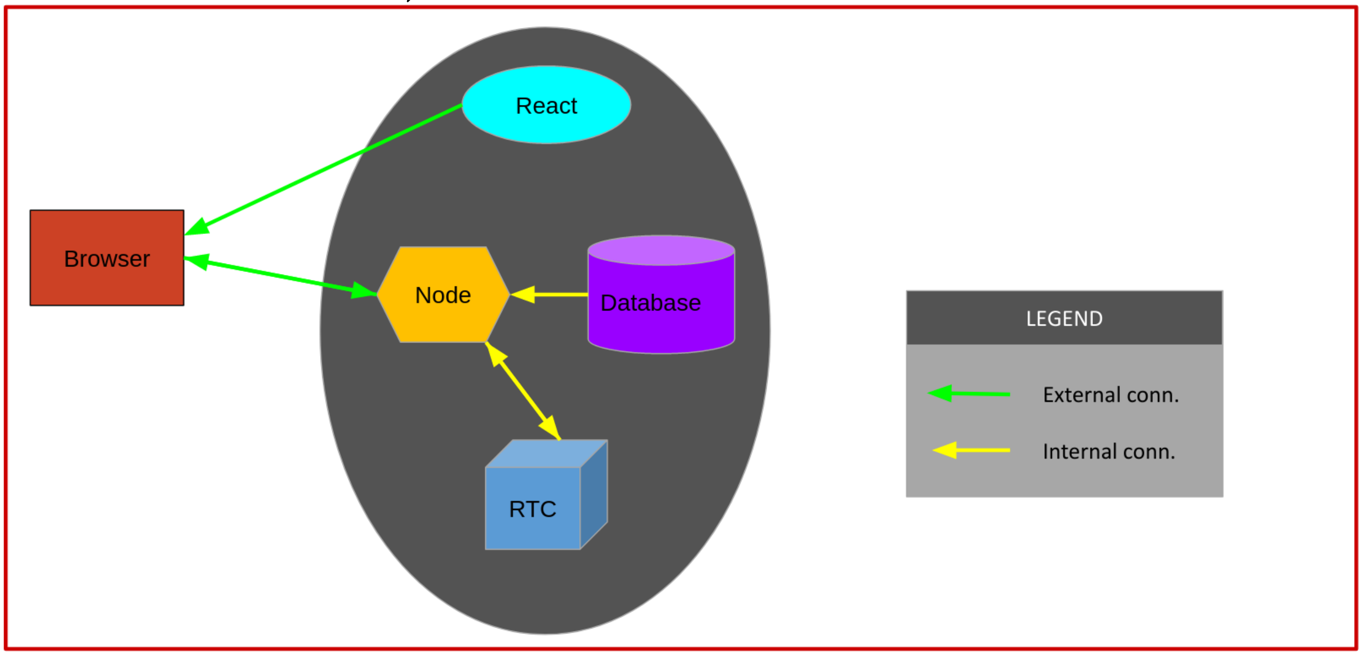
As we mentioned earlier if the user attempts to use a remote Cluster rather than a local one we need to transfer his details to the remote node. Otherwise our user will have to login again in the remote node. This process of transferring sessions is what we mention as Node switching.

Inorder to achieve this we send an https request with appropriate keys, certificates and certificate authority and a special hash along with the user session data to a special path on the remote node. The IP address for the remote Node is stored locally or retrieved from the backend.

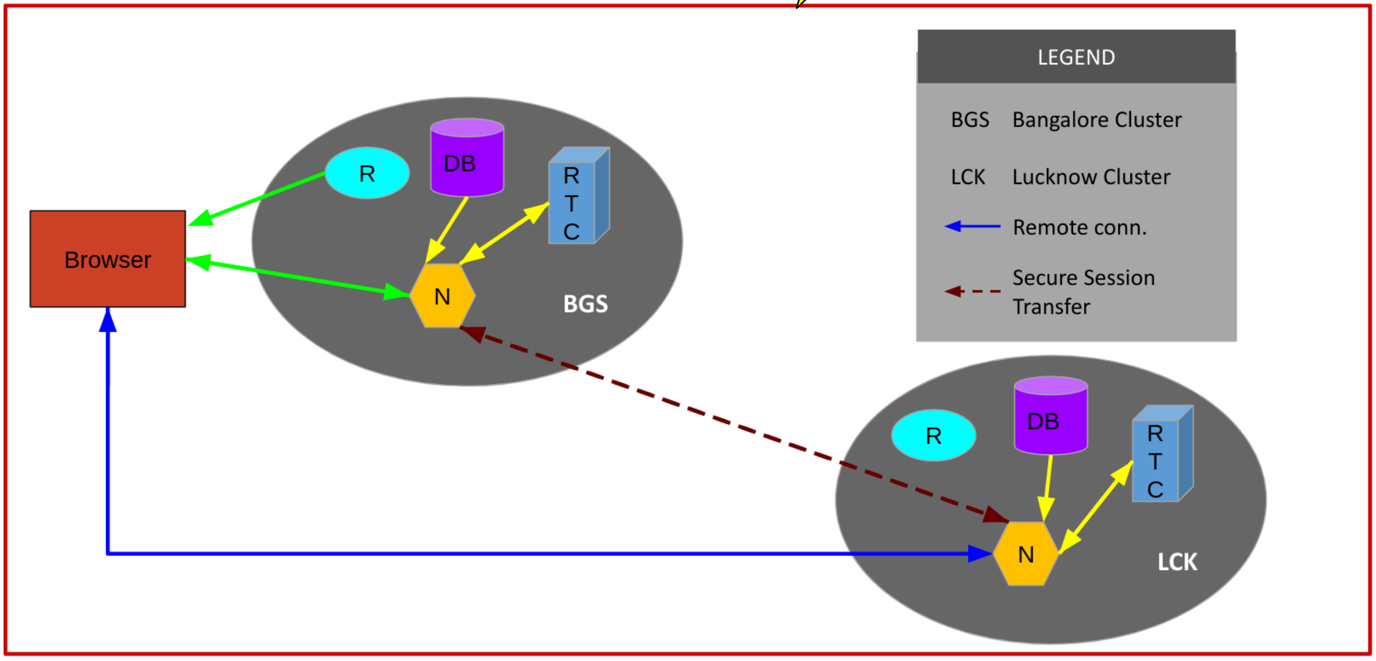
Once we perform this transfer successfully we send the user their current cookie with their local node along with the IP address of the remote Node. This current cookie will be the token to let the remote Node accept our user.

Now the user performs a post request sending the previous cookie to the received IP of the remote Node. The remote Node recognises this cookie as a token and forms the session for this user with a new cookie specific to this Node.

Singular Cluster



Node to Node Communication



6.6. Logout:

Logout is a node function accessed through the menu bar for the Main pages. When the user initiates Logout their corresponding session is deleted on the server side and their cookie is deleted on the Client side. Once the user logs out he is automatically sent back to the Login page

7. Security Testing

Security testing is a type of software testing that focuses on identifying vulnerabilities and weaknesses in a software application's security features. The goal of security testing is to ensure that the application is secure from potential threats and attacks that could compromise its confidentiality, integrity, and availability.

Security testing involves various techniques and methodologies, including:

- Vulnerability scanning: This involves using automated tools to scan the application for known vulnerabilities, such as unsecured ports or weak encryption.
- Penetration testing: This involves simulating real-world attacks on the application to identify vulnerabilities and weaknesses in its security defences.
- Risk assessment: This involves analysing the potential risks and threats to the application and prioritising the security testing efforts accordingly.
- Authorization and authentication testing: This involves testing the application's authorization and authentication features to ensure that they are functioning correctly and securely.
- Input validation testing: This involves testing the application's ability to validate and sanitise user input to prevent malicious inputs from compromising the application's security.
- Security configuration testing: This involves testing the application's security configurations, such as firewalls and encryption settings, to ensure that they are configured correctly.

Overall, security testing is a critical aspect of software testing to ensure that the application is secure from potential threats and attacks.

7.1 OWASP

OWASP stands for the Open Worldwide Application Security Project, which is an international non-profit organisation dedicated to improving the security of software. It provides a community and a range of resources including tools, frameworks and guides to help software developers, security professionals, and organisations to develop and maintain secure web applications.

7.2 OWASP Top Ten

OWASP is known for its list of the top 10 web application security risks, which is a widely recognized standard for identifying and addressing common vulnerabilities in web applications. The list is updated regularly to reflect changes in the threat landscape and new types of vulnerabilities.

8. Tools/software Used For Testing

8.1 WSTG

OWASP WSTG stands for the "OWASP Web Security Testing Guide," which is a comprehensive guide for testing the security of web applications.
The OWASP WSTG is designed to provide a standardized methodology for testing the security of web applications, covering all aspects of the testing process, from planning and preparation to reporting and follow-up. The guide is divided into several sections, each of which covers a different aspect of web application security testing, including:

- Information Gathering
- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input Validation Testing
- Error Handling and Logging Testing
- Cryptography Testing
- Business Logic Testing

The OWASP WSTG provides a set of guidelines and best practices for each of these testing areas, as well as detailed test cases and examples to help testers understand how to apply the guidelines in practice. It is a valuable resource for security professionals, developers, and organisations that want to improve the security of their web applications.

8.2 Burp Suite

Burp Suite is a software application used for web application security testing, developed by PortSwigger Web Security. It is a popular tool used by security professionals to identify security vulnerabilities in web applications.

Burp Suite consists of several modules that work together to perform a variety of security testing tasks. The main modules are:

- Proxy - Allows the user to intercept and modify web traffic between the browser and the server.

- Scanner - An automated tool that scans web applications for vulnerabilities, such as SQL injection and cross-site scripting (XSS).
- Intruder - A tool used for automated testing of web application parameters, such as usernames and passwords.
- Repeater - A tool used for manually manipulating and re-sending individual requests to the server.
- Sequencer - A tool used to test the randomness of tokens and session cookies.
- Decoder - A tool used to decode and encode various data formats, such as base64 and URL-encoded data.

Burp Suite is highly customizable and extensible, allowing users to create their own plugins and extensions to enhance its functionality. It is widely used in the web application security industry, and its features and capabilities make it a powerful tool for identifying and testing security vulnerabilities in web applications.

Tool/software	**Use**	**Version**
WSTG	A guide for web security testing maintained and regularly updated by OWASP	4.2
Burp Suite	Proxy - capturing request and response Sequencer - Analyse randomness of tokens	Community Edition v2023.3.3
Firefox developer tools	Check response headers	-
Cookie-Editor	A browser extension that allows to edit cookie values	1.11.0
Python	Writing simple scripts for testing	3.6

9. Tests And Observation

9.1 Authentication Testing

9.1.1 Testing For Bypassing Authentication Schema

Test Objectives:

Ensure that authentication is applied across all services that require it.

How To Test:

a. Direct Page Request

Request protected page via force browsing. Directly try to access different pages through the address bar in the browser.

Observation

Force browsing to directory	Result
https://localhost:3001/clusters	Blank page
https://localhost:3001/clusters/terminals	
https://localhost:3001/home?id=BL1	
https://localhost:3001/session?id=BL1	
https://localhost:3001/schedule?id=BL1	

Note

Redirect to the login page instead of displaying a blank page.

9.1.2 Testing For Browser Cache Weakness

Test Objectives

Review if the application stores sensitive information on the client-side.

How To Test

a. Browser History

Enter sensitive information into application and logout. Click the back button of the browser and check if previously displayed sensitive data can be accessed whilst unauthenticated.

Observation

Able to access terminal and cluster pages for sometime even after logging out using the back button of the browser.

Note
Application did not forbid the browser from storing sensitive data
Back button can be stopped from showing sensitive by
Delivering page over HTTPS
Setting Cache-Control: must revalidate

b. Browser Cache
Check if the browser does not leak any sensitive data into the browser cache. Search server responses and check if the server instructed the browser not to cache any data.

Observation
Missing necessary headers related to cache in server responses

Note
Cache-Control: no-cache, no-store
Expires: 0
Pragma: no-cache
Additional flags can be included for Cache-Control
Cache-Control: must-revalidate, max-age=0, s-maxage=0

9.2 Session Management Testing

9.2.1 Testing For Session Management Schema

Test Objectives
Gather session tokens, for the same user and for different users where possible.
Analyse and ensure that enough randomness exists to stop session forging attacks.

How To Test
a. Cookie Collection
Make a note of the number of cookies used by the application, pages that set them, domains for which they are valid and if cookies remain constant

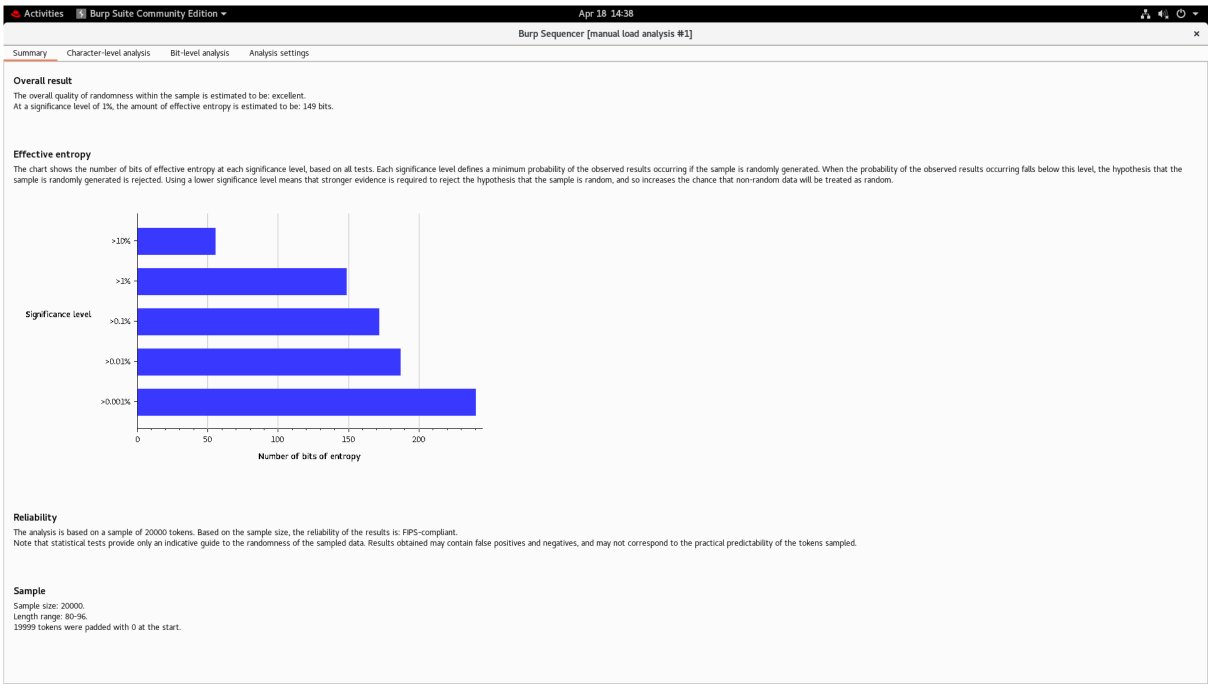
Observation

User	No. of Cookie	Value	Page that set the cookie	Domains for which cookie is valid	Constant
test	1	s%3AZqG-cDwcUHH67SwxaVzByZDaxZalIcdR.PJdOQeWdxTi8X8J19D%2BSIlXEHPKdhltYIvFBt2BKjUQ	login	/clusters; /clusters/terminals; /home; /session; /schedule;	yes
test	1	s%3AsOeKL-GOSH89uN6dJj_Qq4V8XCMIplVe.Qc4NLw4dtHLI9CwdXxG4MsorgLcrimw6wKdGOIvVabQ	login	/clusters; /clusters/terminals; /home; /session; /schedule;	yes

b. Session ID predictability and randomness

Analysis of the variable areas (if any) of the Session ID should be undertaken to establish the existence of any recognizable or predictable patterns.

Summary of tests on randomness of cookie generated by Burp Suite sequencer
Overall result: Excellent
Reliability: FIPS-compliant
Sample size: 20,000



Overall result

The overall quality of randomness within the sample is estimated to be: excellent.
At a significance level of 1%, the amount of effective entropy is estimated to be: 149 bits.

Effective entropy

The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

Reliability

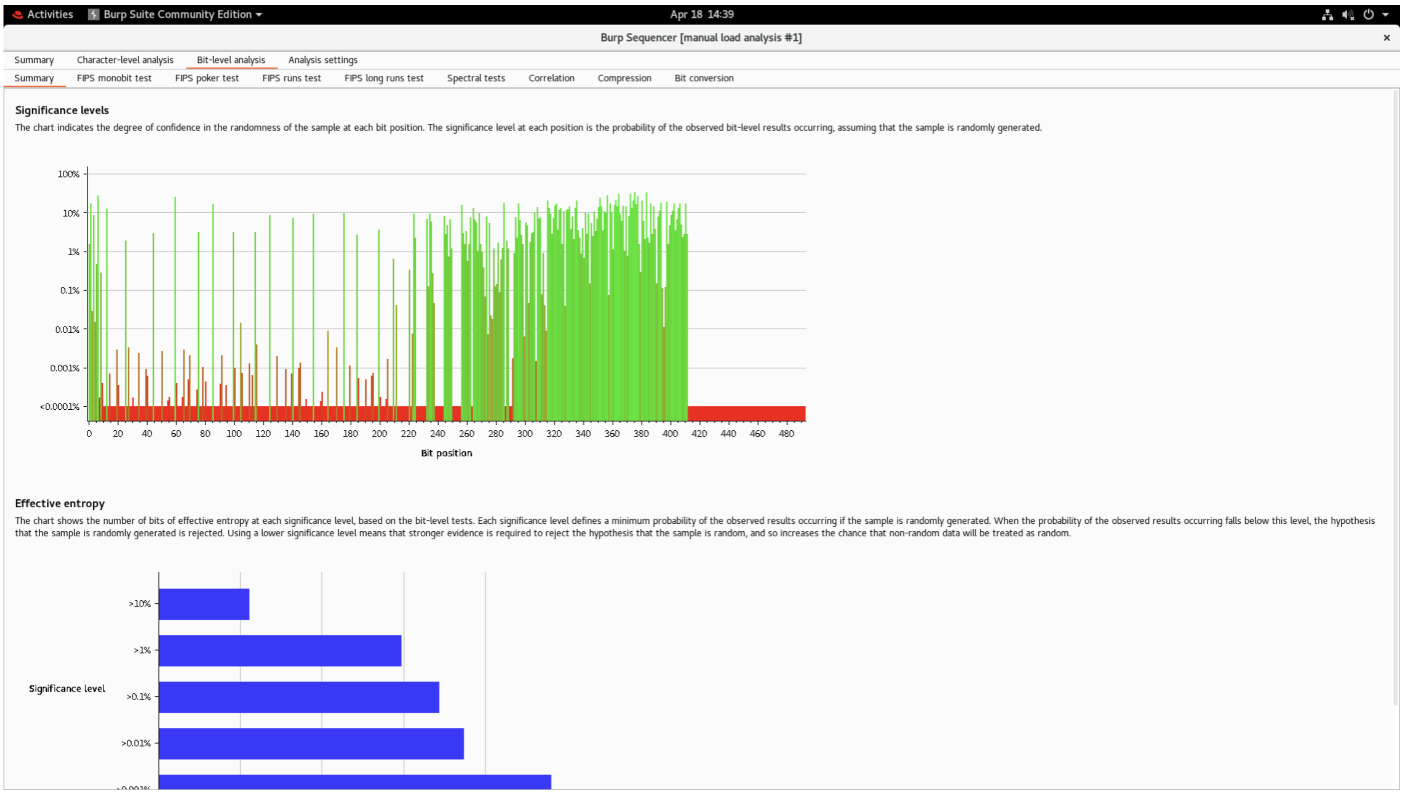
The analysis is based on a sample of 20000 tokens. Based on the sample size, the reliability of the results is: FIPS-compliant.
Note that statistical tests provide only an indicative guide to the randomness of the sampled data. Results obtained may contain false positives and negatives, and may not correspond to the practical predictability of the tokens sampled.

Sample

Sample size: 20000.
Length range: 80-96.
19999 tokens were padded with 0 at the start.

The **Effective entropy chart** shows the number of bits that pass the randomness tests at various significance levels, ranging from > 0.001% to 10%.

Significance levels chart shows the lowest overall probability of the observed results occurring for each character or bit position, if the results are randomly generated. It enables you to quickly gauge the randomness of the sample at each position against various significance levels.



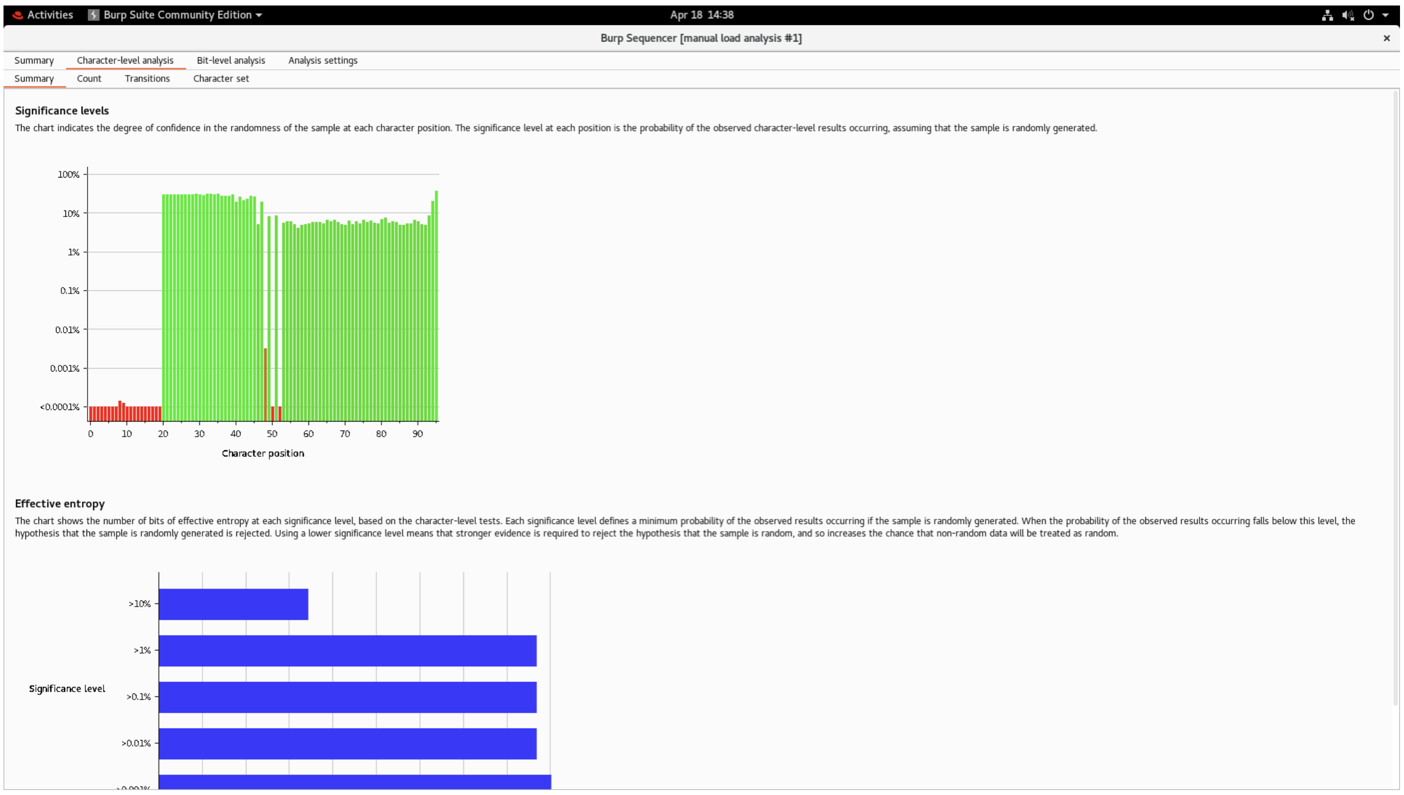
Significance levels

The chart indicates the degree of confidence in the randomness of the sample at each bit position. The significance level at each position is the probability of the observed bit-level results occurring, assuming that the sample is randomly generated.

Effective entropy

The chart shows the number of bits of effective entropy at each significance level, based on the bit-level tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

Summary of Bit-level analysis



Significance levels

The chart indicates the degree of confidence in the randomness of the sample at each character position. The significance level at each position is the probability of the observed character-level results occurring, assuming that the sample is randomly generated.

Effective entropy

The chart shows the number of bits of effective entropy at each significance level, based on the character-level tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

Summary of Character-level analysis

9.2.2 Testing For Cookie Attributes

Test Objectives

Ensure that the proper security configuration is set for cookies.

Observation

Cookie attribute	Use	Expected value	Observed value
Secure	Tells the browser to send the cookie only if the request is sent over secure channel	true	false
HttpOnly	Prevents session leakage by not allowing cookies to be accessed by client side scripts. LImits attack surface of XSS	true	true
Domain	Compares the	Domain that issued	localhost

	domain of the cookie against the domain of the server for which HTTP request is made	the cookie	
Path	Used in conjunction with Domain attribute and plays major role in setting the scope of the cookie	/	/
Expires/Max-Age	Set lifespan of the cookie	Depends on the application	Session
SameSite	Assert that cookies should not be sent along with cross-site requests	Strict	None

Note
Most secure cookie attribute configuration:
Set-Cookie: __Host-SID=<session token>; path=/; Secure; HttpOnly; SameSite=Strict

9.2.3 Testing For Logout Functionality

Test Objectives
Assess the logout UI.
Analyse the session timeout and if the session is properly killed after logout.

How To Test

a. Testing For Logout User Interface
Log out button must be present on all the pages and should be visible without scrolling.

Observation
Log out button missing in terminal page.

b. Testing For Server Side Session Termination
Store the value of the cookie used to identify a session

Invoke logout functionality.
Try navigating to pages that are only visible in authenticated sessions.

Observation
Able to access terminal and cluster pages for sometime after logout due to browser cache.
Cookie is not reset.

9.2.4 Testing Session Timeout

Test Objectives
Validate that a hard session timeout exists.

How To Test
Check if timeout exists by logging in and waiting for logout to be triggered. Test if timeout is enforced by client or server or both. If a cookie contains time related data it is possible that the client is involved in timeout. If the cookie is not cryptographically protected try modifying the time value.

Observation
Timeout takes place after sometime and the session is terminated but the cookie value is not destroyed.
Able to modify the expiry time of the cookie and prolong the session.

Note
Used Cookie-Editor for modifying the expiration field.

10. Conclusion

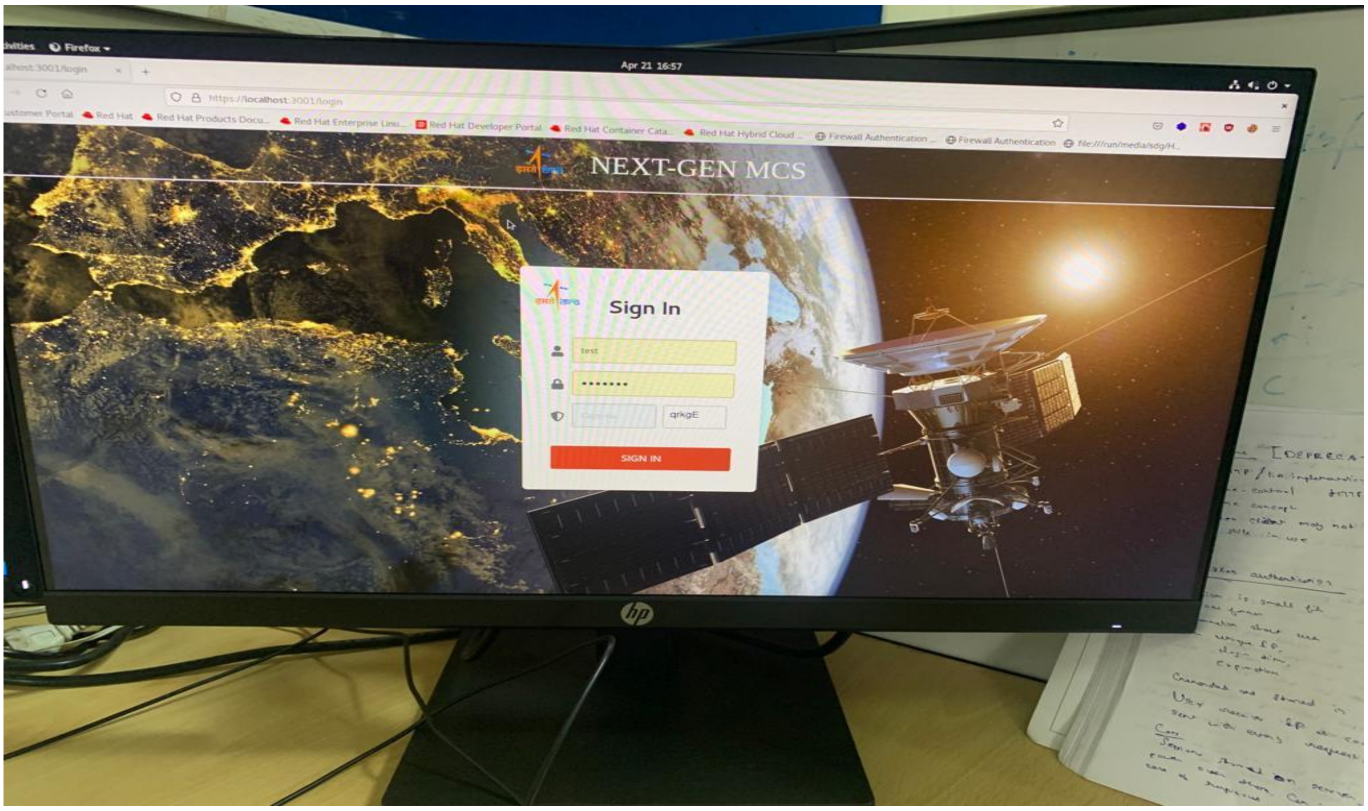
Overall, this project is a simple yet effective web application that is visually appealing and aptly enacts the existing interface for user friendliness. Proper programming measures and software development cycles were practised where feedback from the users are prioritised. The system load for this application was clearly studied and made sure to be in a safe limit for memory usage and processor usage. Program cache storage was made optimal and minimal to reduce storage strains. Application was built with a broad sense of dynamism to accommodate various characteristics and to prove versatility. Once the project was nearing completion the production build was made, which is optimised for running complexity and security.

As for the future development of the project, large scale deployment and industry grade servers are to be implemented. Large scale database handling which includes synchronicity, secure storage and proper authentication mechanics. For the deployment version of the Redis session storage proper separate local only redundant storage fetch server is optimal.

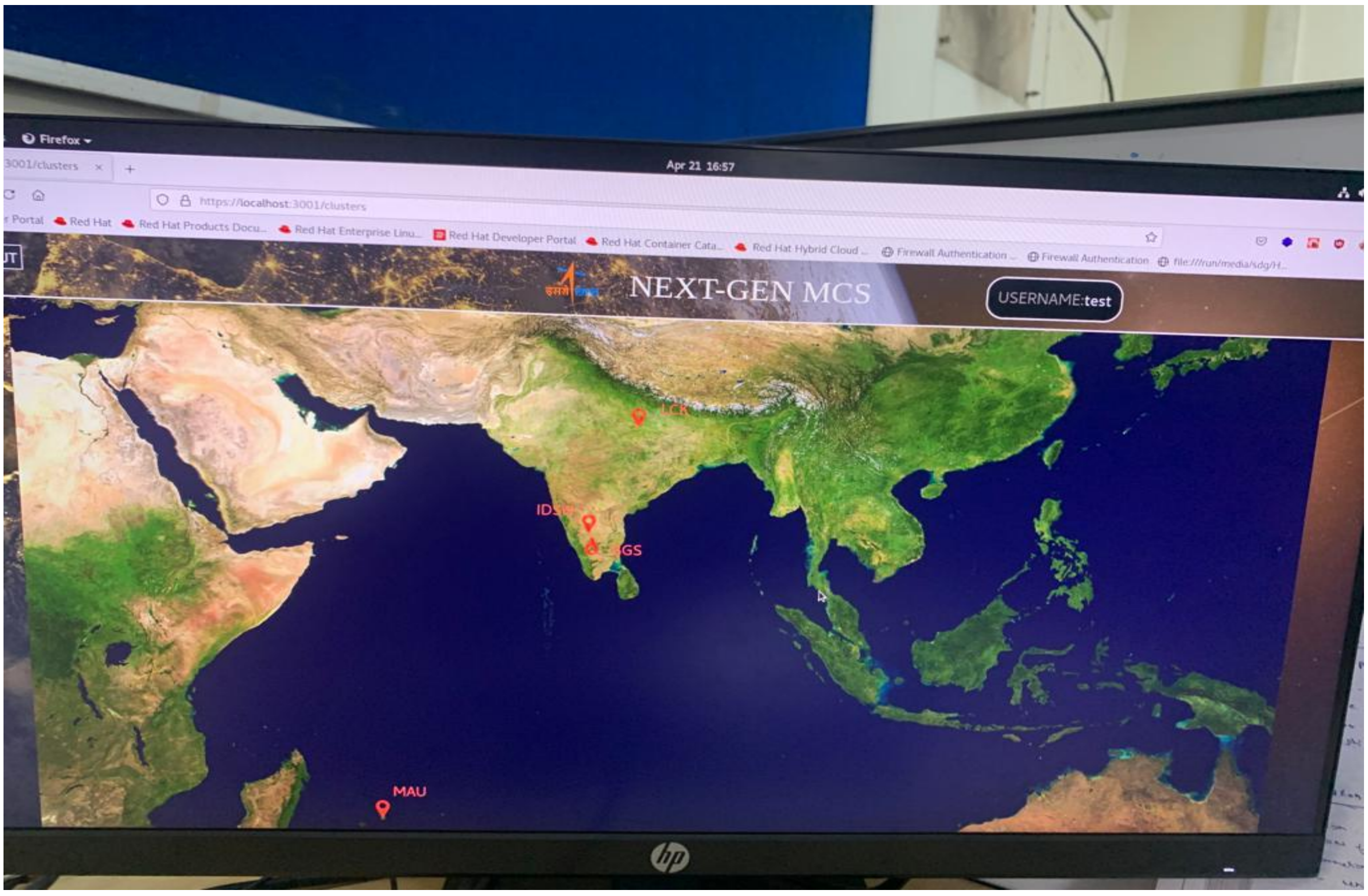
Implementation of roles for the users which could prove to be crucial security measures as well as being a reflection of real world scenario is desired. Roles would indicate how higher up the user is and how much information they are privy to. Proper implementation of this functionality is sketched but yet to be decided upon.

11. Screenshots

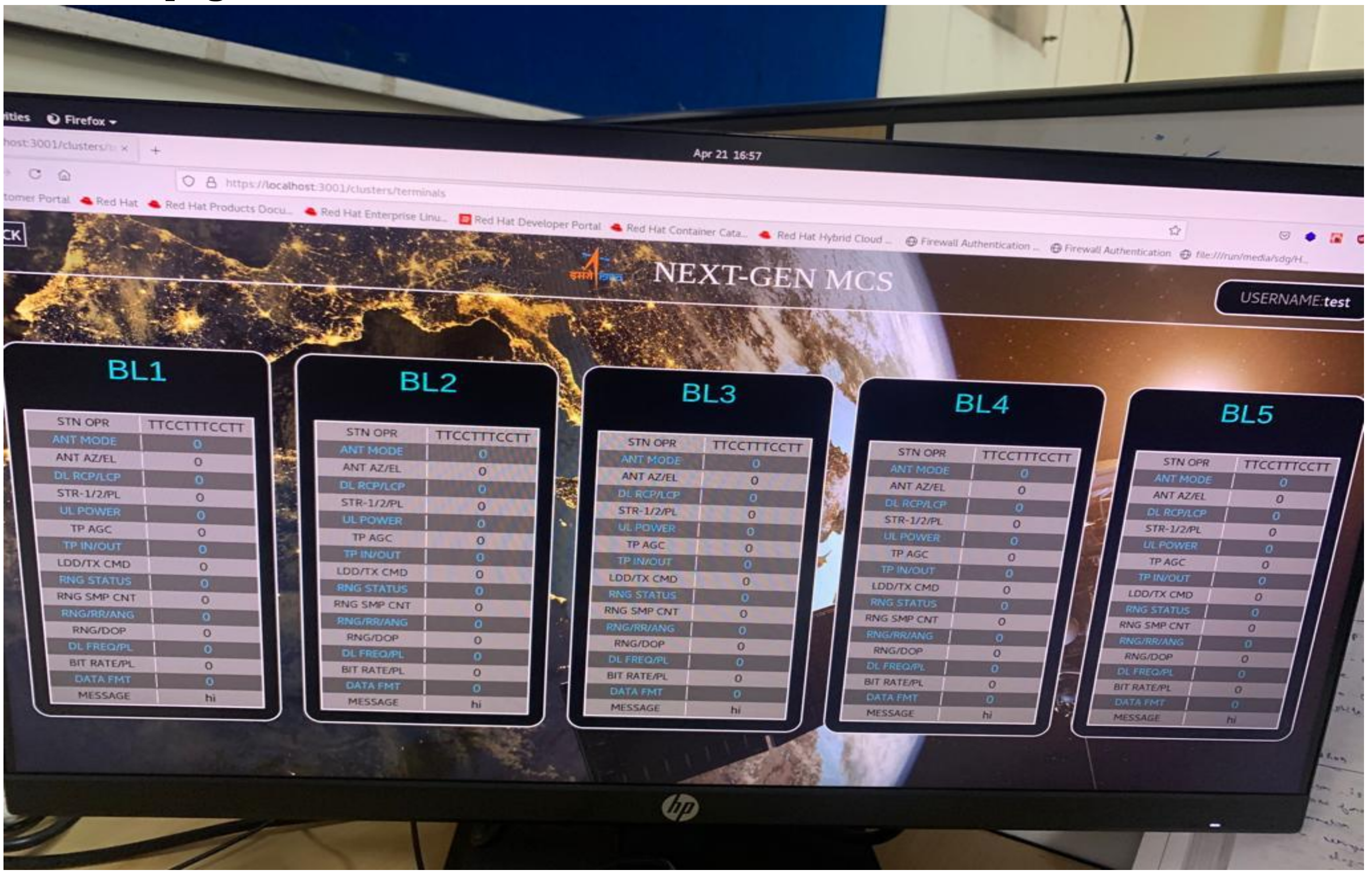
Login page



Cluster page

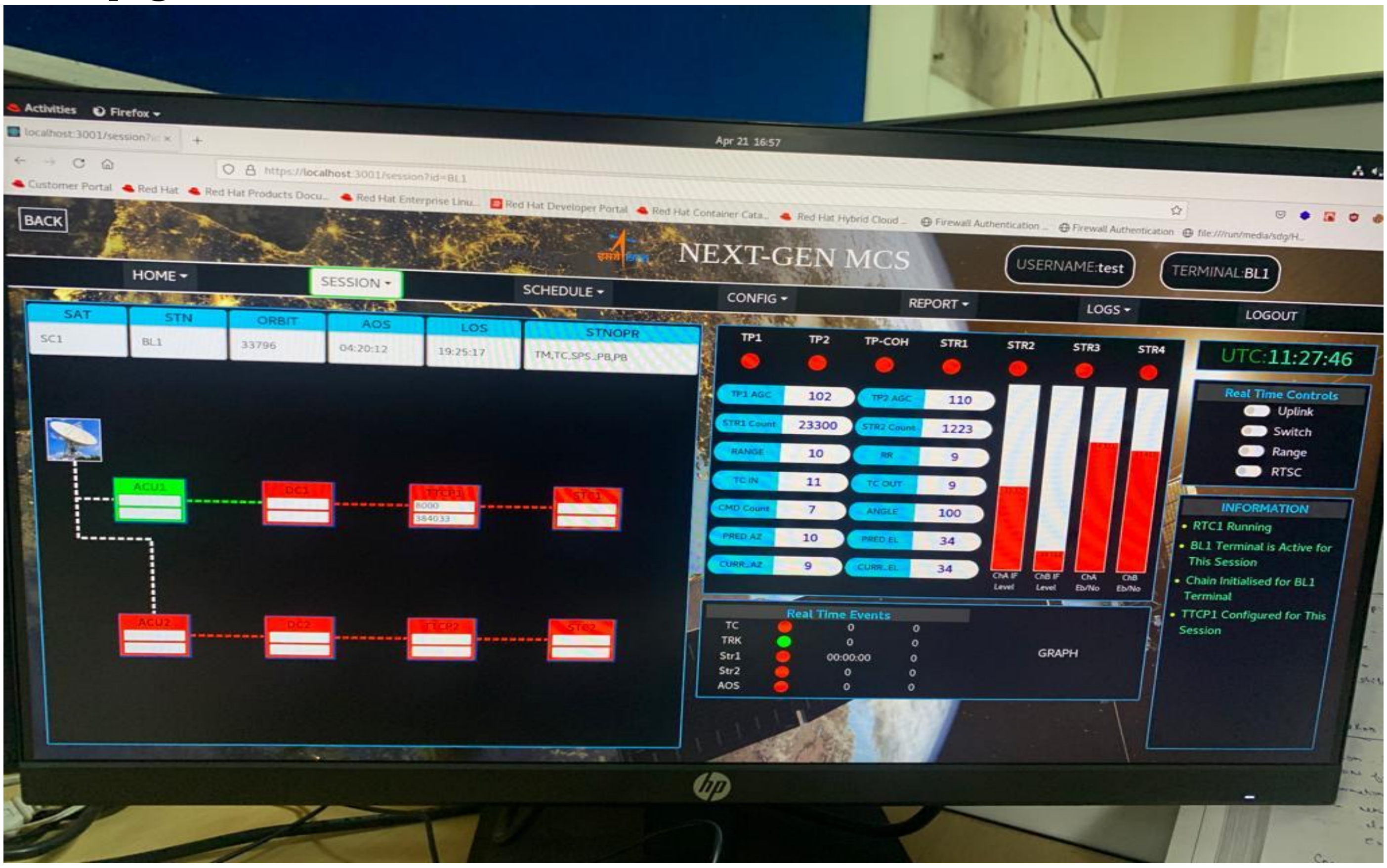


Terminal page



	BL1	BL2	BL3	BL4	BL5
STN OPR	TTCCTTTCCTT	TTCCTTTCCTT	TTCCTTTCCTT	TTCCTTTCCTT	TTCCTTTCCTT
ANT MODE	0	0	0	0	0
ANT AZ/EL	0	0	0	0	0
DL RCP/LCP	0	0	0	0	0
STR-1/2/PL	0	0	0	0	0
UL POWER	0	0	0	0	0
TP AGC	0	0	0	0	0
TP IN/OUT	0	0	0	0	0
LDD/TX CMD	0	0	0	0	0
RNG STATUS	0	0	0	0	0
RNG SMP CNT	0	0	0	0	0
RNG/RR/ANG	0	0	0	0	0
RNG/DOP	0	0	0	0	0
DL FREQ/PL	0	0	0	0	0
BIT RATE/PL	0	0	0	0	0
DATA FMT	0	0	0	0	0
MESSAGE	hi	hi	hi	hi	hi

Home page with active session



SAT	STN	ORBIT	AOS	LOS	STNOPR
SC1	BL1	33796	04:20:12	19:25:17	TM,TC,SPS,,PB,PB

TTCP1: 8000, 384033

TP1 AGC	102	TP2 AGC	110
STR1 Count	23300	STR2 Count	1223
RANGE	10	RR	9
TC IN	11	TC OUT	9
CMD Count	7	ANGLE	100
PRED AZ	10	PRED EL	34
CURR_AZ	9	CURR_EL	34

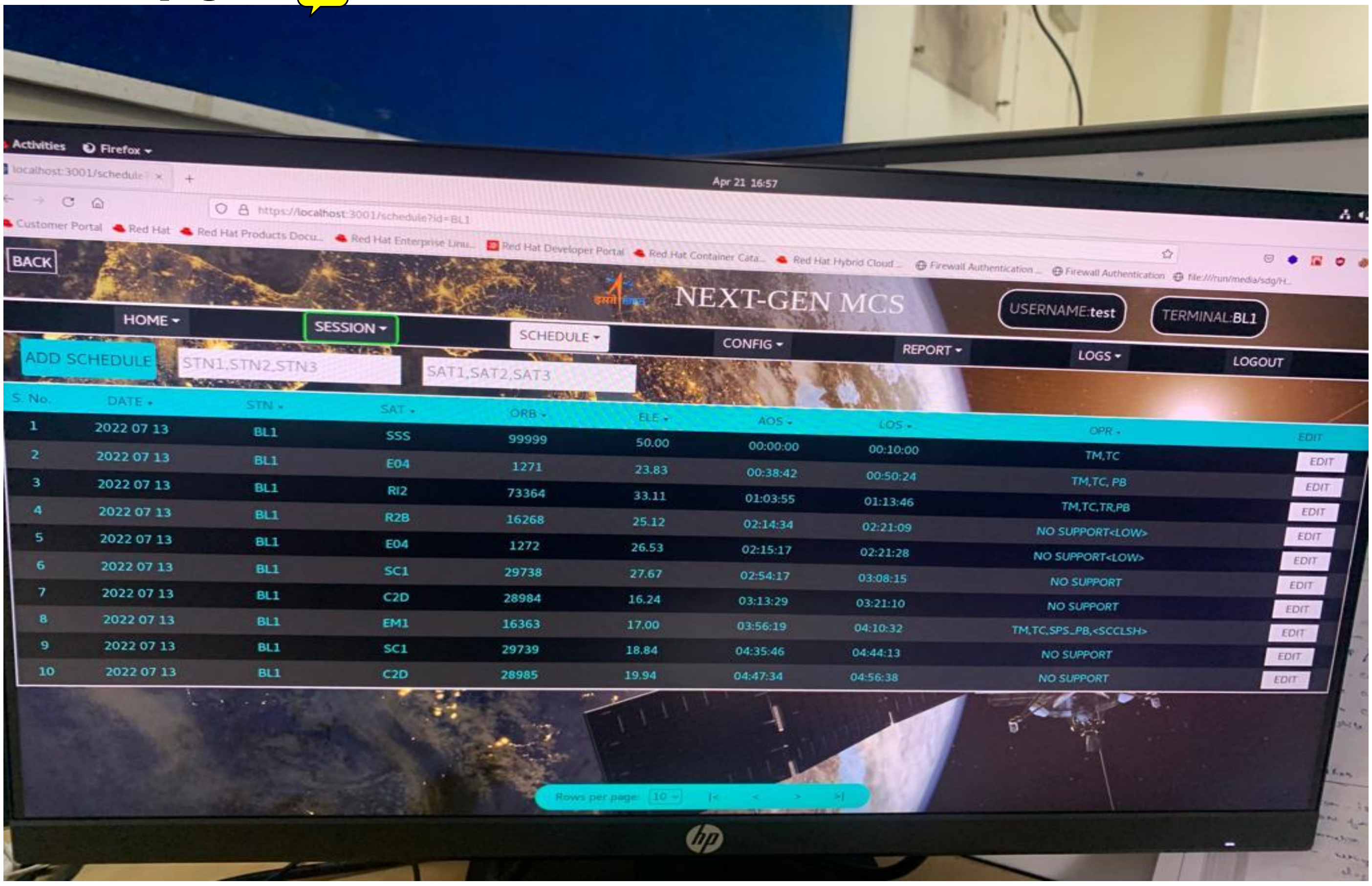
UTC:11:27:46

Real Time Events

TC	🔴	0	0
TRK	🟢	0	0
Str1	🔴	00:00:00	0
Str2	🔴	0	0
AOS	🔴	0	0

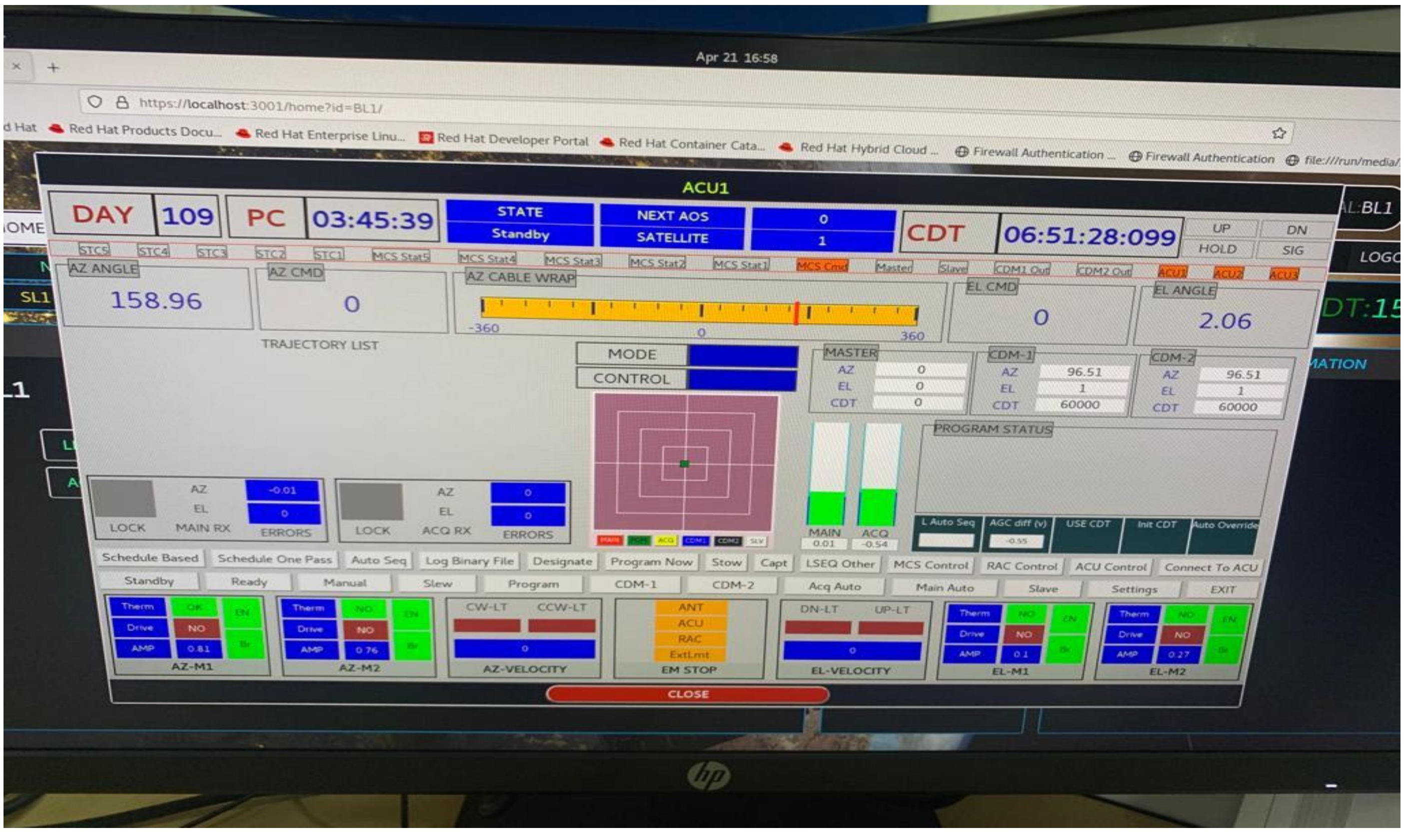
INFORMATION
- RTC1 Running
- BL1 Terminal is Active for This Session
- Chain Initialised for BL1 Terminal
- TTCP1 Configured for This Session

Schedule page

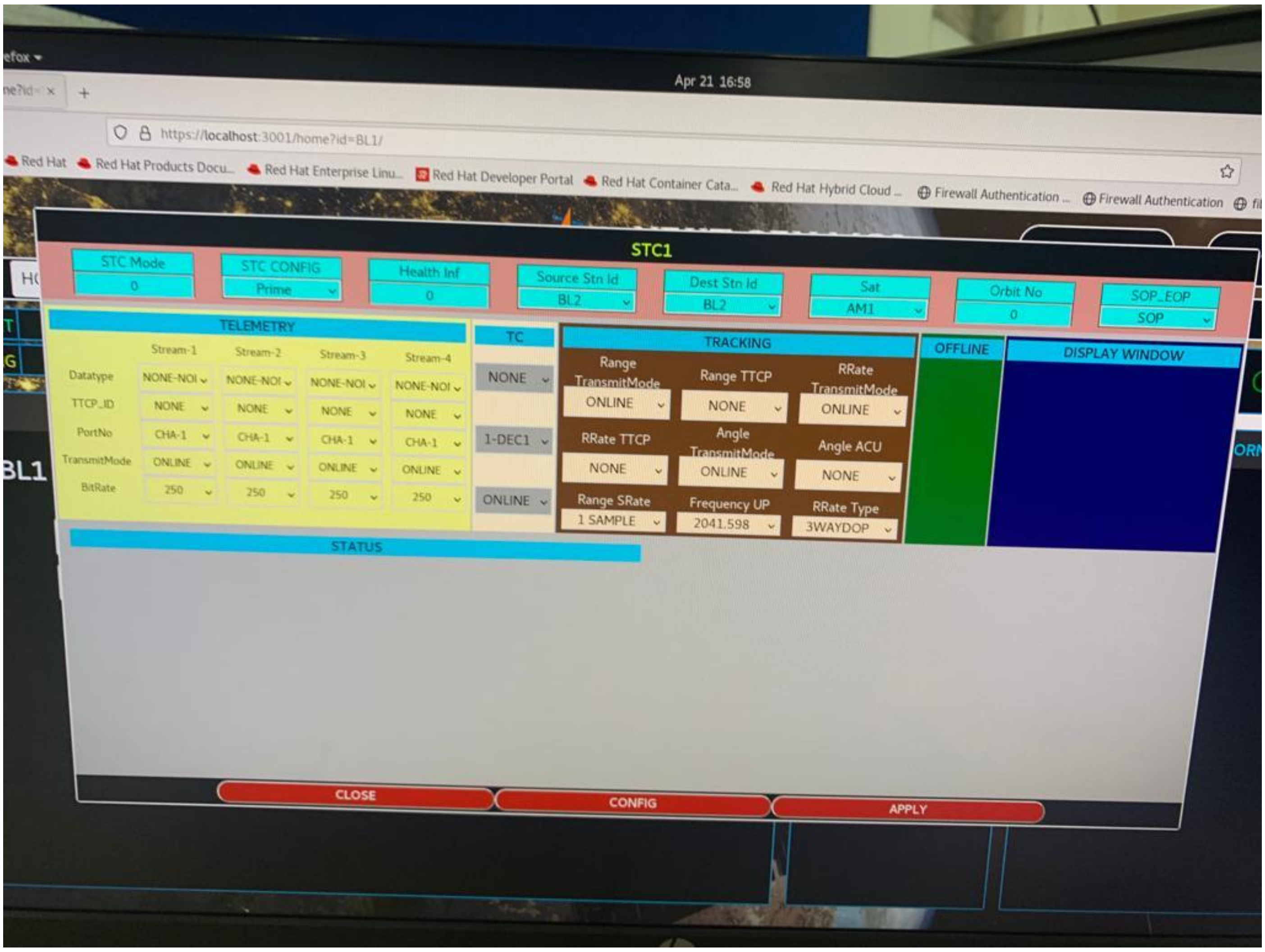


S. No.	DATE	STN	SAT	ORB	ELE	AOS	LOS	OPR	EDIT
1	2022 07 13	BL1	SSS	99999	50.00	00:00:00	00:10:00	TM,TC	EDIT
2	2022 07 13	BL1	E04	1271	23.83	00:38:42	00:50:24	TM,TC, PB	EDIT
3	2022 07 13	BL1	RI2	73364	33.11	01:03:55	01:13:46	TM,TC,TR,PB	EDIT
4	2022 07 13	BL1	R2B	16268	25.12	02:14:34	02:21:09	NO SUPPORT<LOW>	EDIT
5	2022 07 13	BL1	E04	1272	26.53	02:15:17	02:21:28	NO SUPPORT<LOW>	EDIT
6	2022 07 13	BL1	SC1	29738	27.67	02:54:17	03:08:15	NO SUPPORT	EDIT
7	2022 07 13	BL1	C2D	28984	16.24	03:13:29	03:21:10	NO SUPPORT	EDIT
8	2022 07 13	BL1	EM1	16363	17.00	03:56:19	04:10:32	TM,TC,SPS_PB,<SCCLSH>	EDIT
9	2022 07 13	BL1	SC1	29739	18.84	04:35:46	04:44:13	NO SUPPORT	EDIT
10	2022 07 13	BL1	C2D	28985	19.94	04:47:34	04:56:38	NO SUPPORT	EDIT

ACU



STC



STC1

STC Mode	STC CONFIG	Health Inf	Source Stn Id	Dest Stn Id	Sat	Orbit No	SOP_EOP
0	Prime	0	BL2	BL2	AM1	0	SOP

TELEMETRY

	Stream-1	Stream-2	Stream-3	Stream-4
Datatype	NONE-NOI	NONE-NOI	NONE-NOI	NONE-NOI
TTCP_ID	NONE	NONE	NONE	NONE
PortNo	CHA-1	CHA-1	CHA-1	CHA-1
TransmitMode	ONLINE	ONLINE	ONLINE	ONLINE
BitRate	250	250	250	250

TC

NONE

1-DEC1

ONLINE

TRACKING

Range TransmitMode: ONLINE	Range TTCP: NONE	RRate TransmitMode: ONLINE
RRate TTCP: NONE	Angle TransmitMode: ONLINE	Angle ACU: NONE
Range SRate: 1 SAMPLE	Frequency UP: 2041.598	RRate Type: 3WAYDOP