

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018



A PROJECT REPORT (15CSP85) ON

“UI FOR LIVE MONITORING OF ONBOARD STORAGE SYSTEM”

Submitted in Partial fulfillment of the Requirements for the Degree of
Bachelor of Engineering in Computer Science & Engineering

By

CHANDNI V SHARMA (1CR16CS041)

ELISHA PATNAIK (1CR16CS047)

PRATHAM AGARWAL (1CR16CS118)

Under the Guidance of,

Mrs Danthuluri Sudha

Associate Professor, Dept. of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work entitled “**UI FOR LIVE MONITORING OF ONBOARD STORAGE SYSTEM**” carried out by Ms. **CHANDNI V SHARMA**, USN **1CR16CS041**, Ms. **ELISHA PATNAIK**, USN **1CR16CS047**, Mr. **PRATHAM AGARWAL**, USN **1CR16CS118** bonafide students of CMR Institute of Technology, in partial fulfillment for the award of Bachelor of **Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Mrs. Danthuluri Sudha
Associate Professor
Dept. of CSE, CMRIT

Dr. Prem Kumar Ramesh
Professor & Head
Dept. of CSE, CMRIT

Dr. Sanjay Jain
Principal
CMRIT

External Viva

Name of the examiners

- 1.
- 2.

Signature with date

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**UI FOR LIVE MONITORING OF ONBOARD STORAGE SYSTEM**" has been successfully completed under the guidance of Prof. Mrs. Danthuluri Sudha, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Indian Space Research Organization (ISRO), Bangalore

Date: 31st May 2020

Team members:

CHANDNI V SHARMA (1CR16CS041)

ELISHA PATNAIK (1CR16CS047)

PRATHAM AGARWAL (1CR16CS118)

ABSTRACT

The project aims to obtain the data/health parameters of an onboard storage system and present it to the user as an interactive UI and a plotted graph. The health parameters of the onboard system and the transmission of data is taken care by the testbed and a ZeroMQ control which is mainly the hardware aspect of the project which is taken care by the ISRO team. The web application project obtains live data and continuously plots a graph which is the Telemetry section. The polling time can be customized from the UI and the parameters plotted in the graph can also be customized. The UI also has the Telecommand section which takes the input from the user and stores it in a CSV file after validation.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project.

I have a great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain**, Principal, CMRIT, Bangalore, for his constant encouragement.

I would like to thank **Dr. Prem Kumar Ramesh**, Professor and Head, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide **Mrs. Danthuluri Sudha, Associate Professor**, Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

We express our deep gratitude to **Director, URSC** and the **HRD Dept.** for giving us the wonderful opportunity to work in an elated organization.

I would like to thank **Mr. P. S. Sura, Group Director, Payload Data Management Group, URSC** and **Mr. Jyothi Soman, Head, Data Storage Division, URSC** for permitting us to execute our project in this esteemed group.

I would like to thank **Mr. Srinidhi M S, Scientist Engg. SE, URSC** for the constant guidance and mentoring us in every phase of the project.

I would like to extend my thanks to all the members of the **SSR team, URSC** for their encouragement and support during the entire project.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

	Page No.
Certificate	ii
Declaration	iii
Abstract	iv
Acknowledgement	v
Table of contents	vi
List of Figures	viii
List of Tables	ix
List of Abbreviations	x
1 INTRODUCTION	1
1.1 Relevance of the Project	2
1.2 Problem Statement	2
1.3 Objective	3
1.4 Scope of the project	4
1.5 Methodology	5
2 LITERATURE SURVEY	9
3 SYSTEM REQUIREMENTS SPECIFICATION	14
3.1 Hardware Requirements	14
3.2 Software Requirements	14
3.3 Testing Requirements	14
4 SYSTEM ANALYSIS AND DESIGN	15
4.1 System Architecture	15
4.2 Data Flow Diagram	16
5 IMPLEMENTATION	17
5.1 Traditional Web Based Approach	18
5.2 Standalone JAVA Application	24
5.3 Play Framework Architecture	26

5	RESULTS AND DISCUSSION	34
6	TESTING	41
8	CONCLUSION AND FUTURE SCOPE	42
	8.1 Conclusion	42
	8.2 Future Scope	43
	References	44

LIST OF FIGURES

	Page No.
Fig 2.1 CFDP Architecture	10
Fig 2.2 Various subsystems associated with bus management unit	12
Fig 4.1 Proposed System	15
Fig 4.2 Dataflow Diagram	16
Fig 5.1 Project Insert Snippet	18
Fig 5.2 Project Navigation Snippet	18
Fig 5.3 Static Table Population	19
Fig 5.4 Play FrameWork Layout	27
Fig 5.5 Project Structure	27
Fig 5.6 SBT Console	28
Fig 5.7 Localhost 9000	28
Fig 5.8 Code for Homepage – Play Framework	30
Fig 5.9 Code for Login Page – Play Framework	30
Fig 5.10 Routing File	31
Fig 5.11 Controller code	32
Fig 5.12 Model code	33
Fig 6.1 Login and registration page	34
Fig 6.2 Spline graph and data table	35
Fig 6.3 Spline and Pie graph	35
Fig 6.4 Telecommand Section	36
Fig 6.5 Telecommand file	36
Fig 6.6 JAVA Program	37
Fig 6.7 CSV File - Result	38
Fig 6.8 Login page – Play framework	39
Fig 6.9 Home page – Play framework	39
Fig 6.10 Spline graph – Play Framework	40
Fig 6.11 Chart – Play Framework	40

CHAPTER 1

INTRODUCTION

In an imaging spacecraft scenario payload generates raw image data; this data is compressed and formatted.

The satellite has a tele-command & telemetry package. The tele-command is responsible for reception of commands from the ground and transferring it to different subsystems. The telemetry is responsible for transferring the health parameters of all the subsystems to ground.

Added to this, the high speed data handling systems transfer the video data from satellite to ground. Formatted payload data is then transferred to a ground station for further processing over the communication link.

When the desired ground station is not visible to the spacecraft, data is stored on an on-board solid state recorder (SSR). This is referred to as the record operation. When the spacecraft passes over a ground station, the desired data is transmitted from spacecraft to ground station. This is termed as a playback operation.

A telemeter is a device used to remotely measure any quantity. It consists of a sensor, a transmission path, and a display, recording, or control device.

Data Analytics has an important role in Telemetry display that favours data visualization to communicate insight.

1.1 Relevance of The Project

Data dissemination and user interface always help the user to understand the behaviour of the system in a better way, identify problems faster, speedup debugging. Satellites are being launched more frequently these days. Hence automation of data transfer instead of manual extraction will decrease the turnaround time and thereby reduce the analysis time for the designer as well as the user. An enhanced visual representation of data makes it easier and faster to analyse.

1.2 Problem Statement

The goal of this project is to establish a server client software for data transfer of the health parameters of On-board storage system, analyse the various parameters and present them as useful information either as charts or graph for easy user understanding.

The existing system have a flat file structure in which the parameters are stored. There is no visual display of the parameters. Moreover, the data dissemination is also done as a single file in the system standalone itself (There is no server client data transfer). Files are created for the respective subsystems after manual extraction.

The solution provided is to have an elegant server client data transfer protocol thereby automatically data can be disseminated. Visual representation of the data is also provided thereby reducing the analysis time of the user.

1.3 Objective

The objective of this project is to make data display and analysis easier.

The data is obtained from the onboard storage system via the test bed and flight hardware.

Two architectures using different code bases have been developed to give the user a choice based on their requirements. The two solutions developed are - the traditional web based approach and the second being the Play Framework. The UI streams the live data as a Spline and Pie graph along with additional tables which might help in the analysis of the graph. The application also has provision to send Telecommands from the UI and store it in a common place.

A Java standalone application also has been developed which will continuously be populating a local file from the central database.

Overall the objective of the project is to simply and provide a small solution to a significantly huge day to day problem.

1.4 Scope of the Project

The project is divided into 3 significant software components –

The first component of the project is the JAVA program which is hosted on the server and it continuously populates the CSV file with the data from the database in regular intervals which acts as a mechanism to implement the live data streaming.

The second aspect of the project is the UI which is designed using HTML and PHP along with Highcharts integration to plot the live streaming of the data. The UI also has the provision to update commands through the Telecommand section.

The project also implements a Play Framework based platform to provide an another solution to the problem statement.

Constraints:

- The constraints are that since it is meant for the satellite, our software shall have to follow the ISRO Software Process Qualification procedure.
- The software has to be developed in the platform which is specified by the ISRO team.
- Secrecy shall be maintained by the project team about the sensitivity of the mission.
- The software shall not be distributable across various agencies.

1.5 Methodology

The data transmission from the flight hardware to the Test bed happens via the Bus Controller – 1553 and UART interface.

The next data transmission occurs from the Test bed to the server using the Zero Message Queue(ZMQ) protocol.

The application will be hosted on the server which should ideally be accessed even remotely.

The application is designed using the traditional Web Application based architecture.

The other solution designed is JAVA Play Framework with protocols administering at different levels of the architecture.

1.5.1 Traditional Web Application Based Architecture

The entry point to the main UI is the login page.

To gain access to the main UI, the user has to provide the Project ID (given by the organization to each of their Projects) and a Project Key.

Suppose a new project has been registered and the user needs to analyse the charts, the user has to first contact the admin to setup the necessary files from which data flow can be configured. The source of data and the page changes based on project.

The Web Application has the following key features –

- A data table which is populated from a local file which will help in the analysis of the graph by providing additional parameter's data.
- The live data plot which has the data stored in the cloud and also has the following sub features –
 - Full screen view
 - Data table of the plotted graph
 - Exporting the data plotted as a CSV, PDF document and XLS

- Exporting the graph as an image
- The Telecommand section is to send the command, RT Address , RT Sub Address and the Word Count which is then validated and sent to the local file.
- The source of the CSV file can also be given as the input by the user through the UI.

1.5.2 JAVA Program to populate data in CVS file

We use JDBC to read data from database and use File I/O to write CSV file and MySQL connector for making the connections with the underlying database.

We will create a database first. Here, we have used the MySQL server to do so. We have used Xampp in order to start the Apache and MySQL servers. We have added the MySQL connector jar file to the class path of the program in order to connect to the database.

Then, we add import statements to your Java program to import required classes in your Java code. We have added the java.io packages here.

To establish a connection, we have used `DriverManager.getConnection()` method. Here, we have used the method `DriverManager.getConnection (String url, String user, String password)`. In the try block, we created a connection object, connection.

Now, we gave our sql query as a string. Here, we just need the last few rows of the database as a CSV file. Interaction with the database was established using the JDBC Statement. A ResultSet object, result was created.

A CSV file was created using file I/O. Now, we insert the SQL query results into the CSV file. The JDBC Statement was closed. The CSV file was closed. We have a catch block for database error. We also have a second catch block for file I/O error. Once the code is run, the CSV file with the required data is present in Eclipse workspace, in the directory with the same name as that of the package, which contains our code.

This CSV file then gets updated, every time the code is run, giving us the required result.

1.5.3 Play Framework architecture

Play Framework is an open source web application framework which follows the model-view-controller (MVC) architectural pattern. It is written in Scala and usable from other programming languages that are compiled to JVM Bytecode, e.g. Java. It aims to optimize developer productivity by using convention over configuration, hot code reloading and display of errors in the browser.

Play web applications can be written in Scala or Java, in an environment that may be less Java Enterprise Edition-centric. Play uses no Java EE constraints. This can make Play simpler to develop compared to other Java-centric platforms.

What is MVC framework?

The **Model-View-Controller (MVC)** framework is an architectural pattern that separates an application into three main logical components Model, View, and Controller. Hence the abbreviation MVC. Each architecture component is built to handle specific development aspect of an application. MVC separates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs).

Features of MVC -

- Easy and frictionless testability. Highly testable, extensible and pluggable framework. Offers full control over your HTML as well as your URLs
- Leverage existing features provided by ASP.NET, JSP, Django, etc.
- Clear separation of logic: Model, View, Controller. Separation of application tasks viz. business logic, UI logic, and input logic
- URL Routing for SEO Friendly URLs. Powerful URL- mapping for comprehensible and searchable URLs
- Supports for Test Driven Development (TDD)

CHAPTER 2

LITERATURE SURVEY

2.1 Flow control for onboard solid state recorder by Siddhartha B. Rai, Srinidhi M.S., Kuruvilla Varghese

Published in: [2017 4th International Conference on Electronics and Communication Systems \(ICECS\)](#)

This work [1] deals with the hardware implementation of the Consultative Committee for Space Data Systems File Delivery Protocol, using Consultative Committee for Space Data Systems Packet Service and Telemetry/Telecommand framing schemes, which is less software intensive, to replace highly CPU intensive architectures for reliable space communication. A low latency, low area, intelligent programmable hardware has been presented here which can accomplish reliable data transfer up to ~380 Mbps.

In an imaging spacecraft scenario payload generates raw image data; this data is compressed and formatted. Formatted data is then transferred to a ground station for further processing over the communication link. When the desired ground station is not visible to the spacecraft, data is stored on an on-board solid state recorder (SSR). This is referred to as the record operation. When the spacecraft passes over a ground station, the desired data is transmitted from spacecraft to ground station. This is termed as a playback operation. Commands required to carry out these operations are generated and uploaded from time to time.

Space links differ greatly from terrestrial links. Terrestrial systems assume semi-permanent links, low latency, large resources etc. However, space links have larger latencies, dissimilar upstream and downstream link capacities and intermittent connectivity. The space link also attenuates the transmitted signals and there is a possibility of errors creeping into data. As the characteristics between Terrestrial Links and Space Links are very different from one another, protocols used in terrestrial applications cannot be applied directly, but require modification or tuning for space applications. The Consultative Committee for Space Data Systems (CCSDS) has come

up with an international standard to support the operation of spacecraft called the CCSDS File Delivery Protocol (CFDP).

This paper presents the design and implementation of a FCM, which uses CFDP and works alongside Solid State Recorder to attain reliable data transfer.

Terrestrial systems use file transfer protocol (FTP), which is based on transmission control protocol (TCP) or Trivial FTP. Efforts have been made to extend terrestrial protocols to space links. Space Communications Protocol Specifications Transport Protocol (SCPS-TP) and Space Communications Protocol Specifications File Protocol (SCPS-FP), derived from FTP, are two examples. CFDP Protocol has evolved from the shortcomings of SCPS-TP and SCPS-FP. In addition to data delivery over multiple packet delivery services, CFDP also provides mission management capability which makes it more powerful.

In its simplest conception it operates over a single link. However, it can be extended to complex scenarios where it provides store-and-forward functionality across an arbitrary network, containing multiple links with disparate availability, as well as subnetworks with heterogeneous protocols. It also provides file management services to give the user control over the storage medium. This protocol is agnostic to the technology being used and there are no restrictions to the type of data which can be transferred over the link. It can be utilized for a wide range of applications involving the loading, dumping, and control of spacecraft storage.

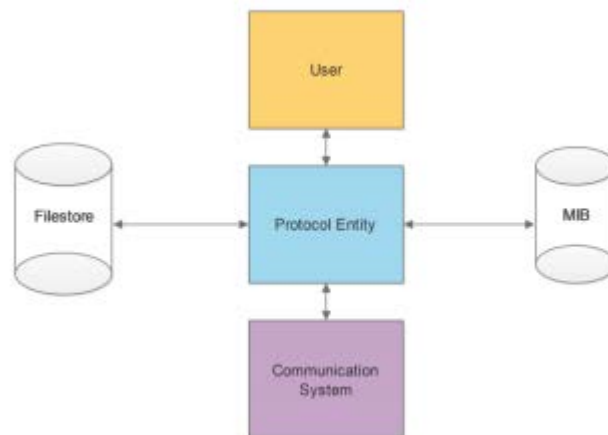


Figure 2.1: CFDP architecture

The auto sequencing of PDU formation is carried out in the Flow Control Module. Software initiates the protocol and handles higher level interaction. More intelligence is built into hardware which requires less intervention. Frame generation is carried out with very little latency courtesy a zero copy scheme. The hardware is optimized for low area and is scalable. The Flow Control hardware can handle a wide range of input and output data rates.

2.2 Automated verification of spacecraft telemetry data by A. Savitha, M. Ravindra, Prasanna Kumar N, Rajiv R. Chetwani, K.M. Baradwaj (ISRO)

Published in: [2014 IEEE International Conference on Computational Intelligence and Computing Research](#)

Telemetry package is one of the inevitable subsystems in a complex spacecraft. Telemetry (TM) data of the mainframe and payload section of the spacecraft is required to be monitored continuously throughout its life span to check the subsystems health and take appropriate decision from ground or autonomously from on-board. In various types of spacecraft like geostationary, remote sensing and interplanetary machines, the telemetry data collected for the health monitoring of hardware, software and its functions will be huge. This huge data of collection is aided by Telemetry database code which has to be verified against the TM requirements. These telemetry requirements data are maintained in flat word and excel worksheet. This paper delineates the methodology and implementation of automation of entire telemetry database verification. This paper [2] also describes the comprehensive process for verifying spacecraft telemetry related information of all types of space crafts to complete the traceability analysis very fast and display result in GUI form. This software reduces a lot of valuable time during project schedule and ease the process of analysis.

An On-Board Computing system (OBC)/ Bus Management Unit (BMU) is designed to check the health of spacecraft and to control the operations of a satellite. The BMU/OBC software modules are capable of performing functions such as Telecommand (TC) decoding, telemetry (TM) monitoring, attitude control, thermal control, data storage and playback and also interface with various payloads.

Figure 2 Shows the BMU with various subsystems. Bus Management unit receive data from various subsystems like sensors, actuators, temperature controller etc., and formats them into a suitable form for transmitting to the ground. It creates frame to be transmitted by adding the frame sync code, spacecraft ID, frame header, sub-frame id etc., for Normal Telemetry.

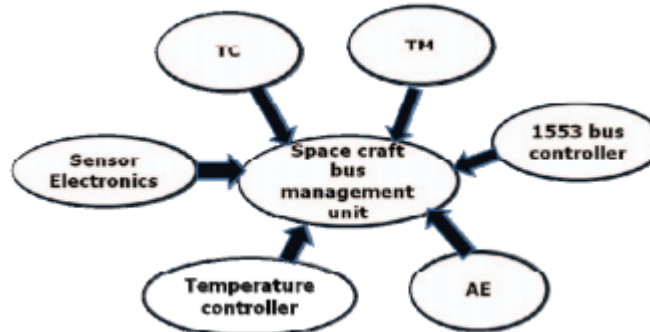


Figure 2.2: Various subsystems associated with bus management unit

The telemetry database contains the package id and word number information of parameters to be transmitted in each cycle. The database is generated by taking the normal index table from the code based on the channel/frame allocated to each parameter. This module decodes the Package ID and word number which is transmitted in each 32ms and requests for the data from the respective package. Telemetry data has to be verified from all the packages.

The TM document and TM database (taken from onboard software) is an input for the telemetry database verification. The channel, frame, dwell address is read from the excel sheet column by column and stored in text file also in the list box of software. The package_id and word number is also extracted from the available data read from excel sheet for comparing with the onboard TM database. The telemetry parameter that is extracted from code is also put in text file for verification. Finally, both are TM data generated from the code and excel sheet are compared and result file is generated which says each telemetry parameter is correct or not.

Advantages of using automated analysis of spacecraft telemetry data:

The Telemetry data verification software can be loaded on any commercial system. The test and evaluation engineer who work for different projects can use this software without installing any interfaces.

This facility can be used for testing the telemetry data generated during Independent verification and validation of the system which is much earlier to initial bench level testing.

The package_id and the word number is auto generated from the dwell address.

The package_id and the word number extracted are compared with the actual on board code and result is generated saying 'OK or 'Not OK' for all the telemetry. Case study done for GSAT-4 have 2048 normal telemetry parameters.

This software can be used for any other project without any modification even if the package_id and word number changes from project to project.

Manual method of analysing the data has considerably come down with this automation.

It was successfully tested and verified for gsat-4 spacecraft.

It also verifies the telemetry data available after initial bench level testing.

Automated telemetry database verification had been used for many projects. Labour-intensive method of evaluating the data has significantly reduced with this automation. The automated telemetry database verification report file illustrates the telemetry database verified. This has resulted in considerable reduction of labour-intensive effort by verification and validation engineers. The time required for analysis has drastically come down by eighty percent.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Tools:

- 1553 Simulator
- Mainframe server
- UART interface

3.2 Software Tools:

- IntelliJ IDE
- Play Framework
- Xampp
- Eclipse IDE
- Highcharts

3.3 Testing Tools:

- System under test is the Solid State Recorder (SSR).

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 SYSTEM ARCHITECTURE

The data transmission from the flight hardware to the Test bed happens via the Bus Controller – 1553 and UART interface. A universal asynchronous receiver/transmitter (UART) is a block of circuitry responsible for implementing serial communication. Essentially, the UART acts as an intermediary between parallel and serial interfaces. On one end of the UART is a bus of eight-or-so data lines (plus some control pins), on the other is the two serial wires - RX and TX.

A software development kit (SDK) is a collection of software development tools in one installable package. They ease creation of applications by having compiler, debugger and perhaps a software framework. They are normally specific to a hardware platform and operating system combination.

An application programming interface (API) is a computing interface which defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees.

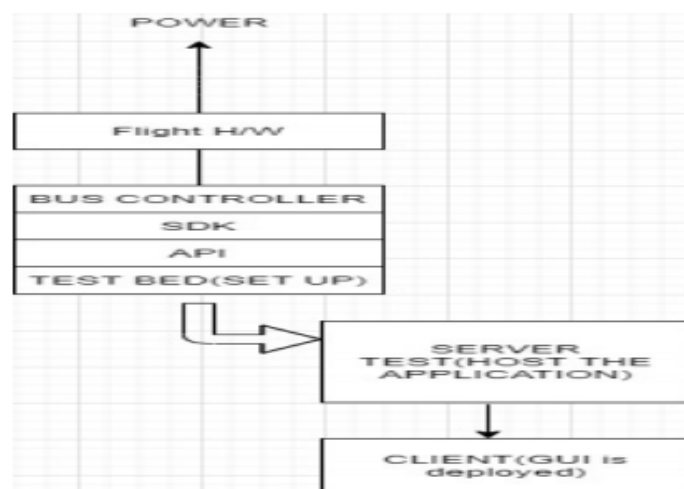


Figure 4.1: Proposed system

4.2 DATA FLOW DIAGRAM

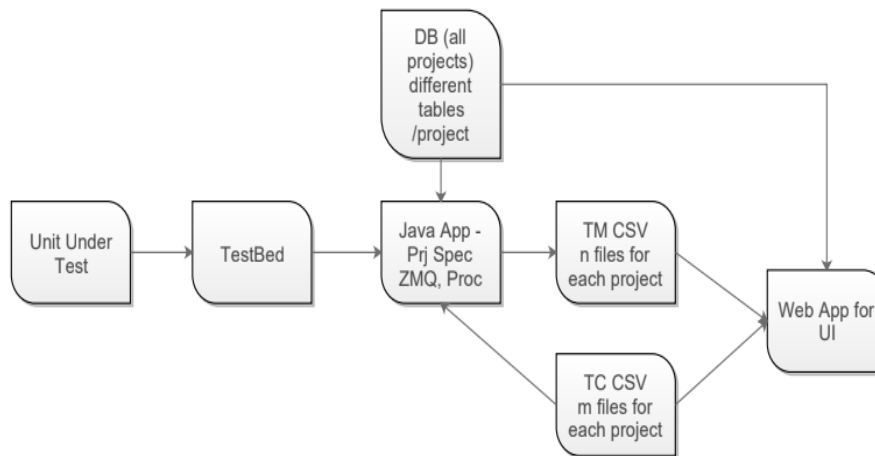


Figure 4.2 Dataflow

The data is first obtained from the Flight Hardware and transmitted to the Test Bed via the 1553 Bus Controller.

The next data transmission occurs from the Test bed to the server using the Zero Message Queue(ZMQ) protocol.

ZeroMQ is a high performance asynchronous messaging library, aimed at use in distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, a ZeroMQ system can run without a dedicated message broker.

The application will be hosted on the server which should ideally be accessed even remotely.

This project is concerned with the Java app to convert data from the database into CSV files, telemetry (plotting the data from the CSV files into graphs and Telecommand (user input from UI). All the Telecommand and telemetry operations will take place on the UI itself. The application is designed using the traditional Web Application based architecture. The other solution designed is JAVA Play Framework with protocols administering at different levels of the architecture.

CHAPTER 5

IMPLEMENTATION

The project can be disintegrated into 3 main parts, each performing a different function. The project also provides 2 solutions to the problem statement.

The first one is using the traditional web application based approach and the second is using the Play Framework Architecture.

The java program is the intermediate part which will act as a bridge between the 2 solutions by populating the necessary data for plotting from the database.

The database is populated by the ZMQ protocol which receives data as packets from the hardware. ZeroMQ is a high performance asynchronous messaging library, aimed at use in distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, a ZeroMQ system can run without a dedicated message broker. The ZeroMQ API provides sockets (a kind of generalization over the traditional IP and Unix domain sockets), each of which can represent a many-to-many connection between endpoints. Operating with a message-wise granularity, they require that a messaging pattern be used, and are particularly optimized for that kind of pattern.

The basic ZeroMQ patterns are:

- Request–reply - Connects a set of clients to a set of services. This is a remote procedure call and task distribution pattern.
- Publish–subscribe - Connects a set of publishers to a set of subscribers. This is a data distribution pattern.
- Push–pull (pipeline) - Connects nodes in a fan-out / fan-in pattern that can have multiple steps, and loops. This is a parallel task distribution and collection pattern.
- Exclusive pair - Connects two sockets in an exclusive pair.

5.1 Traditional Web Application based UI –

The entry point to the main UI is the login page.

To gain access to the main UI, the user has to provide the Project ID (given by the organization to each of their Projects) and a Project Key.

Suppose a new project has been registered and the user needs to analyse the charts, the user has to first contact the admin to setup the necessary files from which data flow can be configured. The source of data and the page changes based on project.

5.1.1 Algorithm for validation and implementation of the login page

The login page takes the Project ID and the Project Key as the input, validates them and accordingly navigates the user to the corresponding graph page based on the project.

The following is the basic algorithm used for the login page –

1. The user enters the credentials.
2. The data is first received from the database by passing the credentials entered by the user.
3. By doing this we get to know if the Project already exists or not.
4. The project ID is then checked and the corresponding page is rendered.

```

15
16 $result = mysqli_query($con, $s);
17
18 $num = mysqli_num_rows($result);
19
20 if($num==1){
21     echo"<script>alert('Already registered!');window.location='login.php'</script>";
22 }
23 else{
24     $reg= "insert into usertable(name, password) values ('$name', '$pass')";
25     mysqli_query($con, $reg);
26 }
27
28 $result = mysqli_query($con, $s);
29
30 $num = mysqli_num_rows($result);
31
32 if($num == 1){
33     $_SESSION['username'] = $name;
34     if($name == 'Admin' || $name == 'RISAT-1'){header('location:index.php');}
35     elseif($name == 'gsat'){header('location:gsat.php');}
36     else {header('location:home.php');}
37 }
38 else{

```

Figure 5.2 project based navigation

5.1.2 Algorithm for populating static data table

A data table which is populated from a local file which will help in the analysis of the graph by providing additional parameter's data.

1. The required local file is opened in read mode.
2. Each row of the csv file is read.
3. The read data is echoed to the UI.

```

10 while (($data = fgetcsv($handle, 1000, ",")) != FALSE) {
11     $num = count($data);
12     if ($row == 1) {
13         echo '<thead><tr>';
14     }else{
15         echo '<tr>';
16     }
17
18     for ($c=0; $c < $num; $c++) {
19         //echo $data[$c] . "<br />\n";
20         if(empty($data[$c])) {
21             $value = "&nbsp;";
22         }else{
23             $value = $data[$c];
24         }
25         if ($row == 1) {
26             echo '<th>'.$value.'</th>';
27         }else{
28             echo '<td>'.$value.'</td>';
29         }
30     }
31
32     if ($row == 1) {
33         echo '</tr></thead><tbody>';
34     }else{
35         echo '</tr>';
36     }
37     $row++;
38 }

```

Figure 5.3 static table population

5.1.3 Algorithm for plotting the graph

The live data plot which has the data stored in the cloud and also has the following sub features –

- Full screen view
- Data table of the plotted graph
- Exporting the data plotted as a CSV, PDF document and XLS
- Exporting the graph as an image

Highcharts is a modern SVG-based, multi-platform charting library. It makes it easy to add interactive charts to web and mobile projects. It has been in active development since 2009, and remains a developer favourite due to its robust feature set, ease of use and thorough documentation.

Keeping this in mind, this project also integrates Highcharts to plot the graph due to the ocean of features it offers.

The code for creating a chart is as follows -

```
function createChart() {  
  
    Highcharts.chart('spline_container', {  
  
        chart: {  
  
            type: 'spline'  
  
        },  
  
        title: {  
  
            text: 'Live Data'  
  
        },  
  
        accessibility: {  
  
            announceNewData: {  
  
                enabled: true,  
  
                minAnnounceInterval: 15000,  
  
                announcementFormatter: function (allSeries, newSeries, newPoint) {  
  
                    if (newPoint) {  
  
                        return 'New point added. Value: ' + newPoint.y;  
  
                    }  
  
                    return false;  
  
                }  
  
            }  
  
        }  
  
    }  
  
}
```

```
    }  
  },  
  data: {  
    csvURL: urlInput.value,  
    enablePolling: pollingCheckbox.checked === true,  
    dataRefreshRate: parseInt(pollingInput.value, 10)  
  }  
}),
```

```
Highcharts.chart('pie_container', {  
  chart: {  
    type: 'pie'  
  },  
  title: {  
    text: 'Live Data'  
  },  
  accessibility: {  
    announceNewData: {  
      enabled: true,  
      minAnnounceInterval: 15000,  
      announcementFormatter: function (allSeries, newSeries, newPoint) {
```

```
        if (newPoint) {  
            return 'New point added. Value: ' + newPoint.y;  
        }  
        return false;  
    }  
}  
},  
data: {  
    csvURL: urlInput.value,  
    enablePolling: pollingCheckbox.checked === true,  
    dataRefreshRate: parseInt(pollingInput.value, 10)  
}  
});  
  
if (pollingInput.value < 1 || !pollingInput.value) {  
    pollingInput.value = 1;  
}  
}  
  
urlInput.value = defaultData;
```

```
// We recreate instead of using chart update to make sure the loaded CSV  
  
// and such is completely gone.  
  
pollingCheckbox.onchange = urlInput.onchange = pollingInput.onchange =  
createChart;  
  
  
  
// Create the chart  
  
createChart();
```

5.1.4 Telecommand Section –

This section is used to send the commands along with some other parameters to be stored in a CSV file which will be used for further processing of data based on the command passed.

The steps to do this is as follows –

1. The user first enters the data in the text boxes. This data is received and a null check is performed.
2. The entered data is stripped off all the spaces and special characters.
3. Once the data is cleaned, it can be populated in the file
4. The data is first stored in an array.
5. The array is then simply appended to the file.

5.2 JAVA app to convert database into CSV file -

The data is obtained from the database and the part of the database to be plotted in a graph is inserted as data into a CSV file.

Live data to be projected is obtained from the CSV files hosted on the server.

Data flows through from the hardware to the database in real time via the ZMQ protocol.

The CSV file, obtained from the database is used for plotting data in real time.

We use JDBC to read data from database and use File I/O to write CSV file. And a JDBC driver library for the underlying database is necessary.

Step 1: Create database. Here, we have used the MySQL server to do so.

Step 2: Add the MySql connector jar file to the classpath of the program.

Step 3: Add import statements to your Java program to import required classes in your Java code.

Step 4: Establish a connection using `DriverManager.getConnection()` method. Here, we have used the method `DriverManager.getConnection (String url, String user, String password)`.

Step 5: In the try block, create a connection object.

Step 6: Now, give your sql query as a string. Here, we just need the last few rows of the database as a CSV file.

Step 7: Interact with the database using the JDBC Statement.

Step 8: Create the ResultSet object, result.

Step 9: Create a CSV file using file I/O.

Step 10: Insert the SQL query results into the CSV file.

Step 11: Close Statement.

Step 12: Close the CSV file.

Step 13: Write a catch block for database error.

Step 14: Write a second catch block for file I/O error.

Once the code is run, the CSV file with the required data is present in Eclipse workspace.

5.3 Play Framework –

5.3.1 Components of MVC -

Model

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

View

Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

Controller

Accepts input and converts it to commands for the model or view.

In addition to dividing the application into these components, the model–view–controller design defines the interactions between them.

The model is responsible for managing the data of the application. It receives user input from the controller.

The view means presentation of the model in a particular format.

The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.

UI FOR LIVE MONITORING OF ONBOARD STORAGE SYSTEM

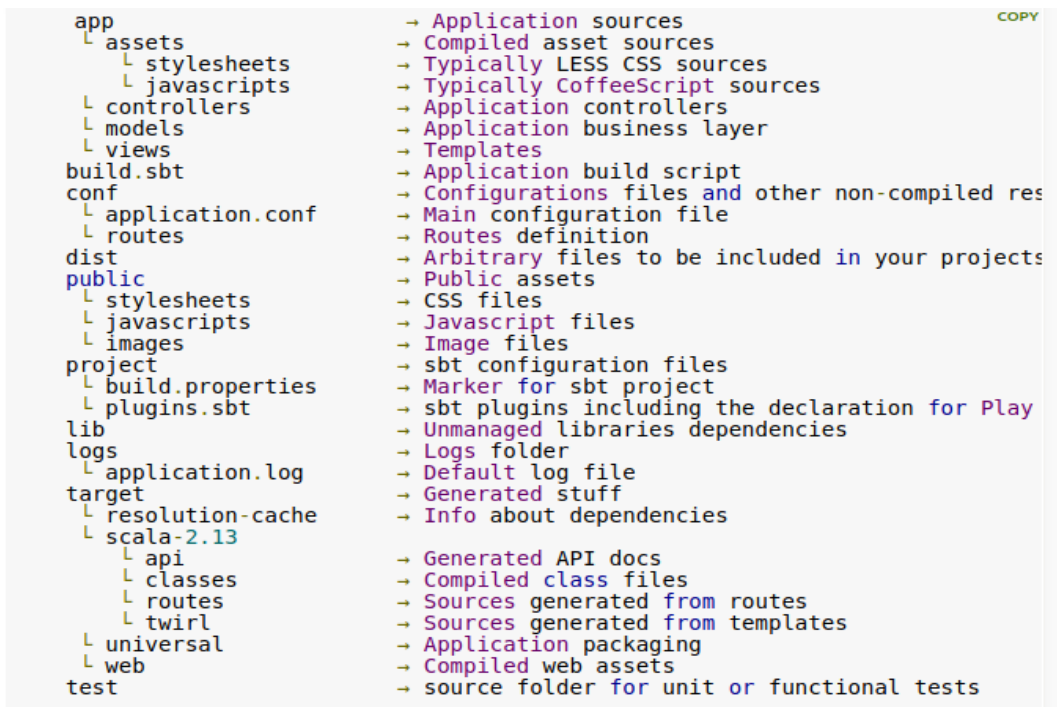


Figure 5.4 Play FrameWork Layout

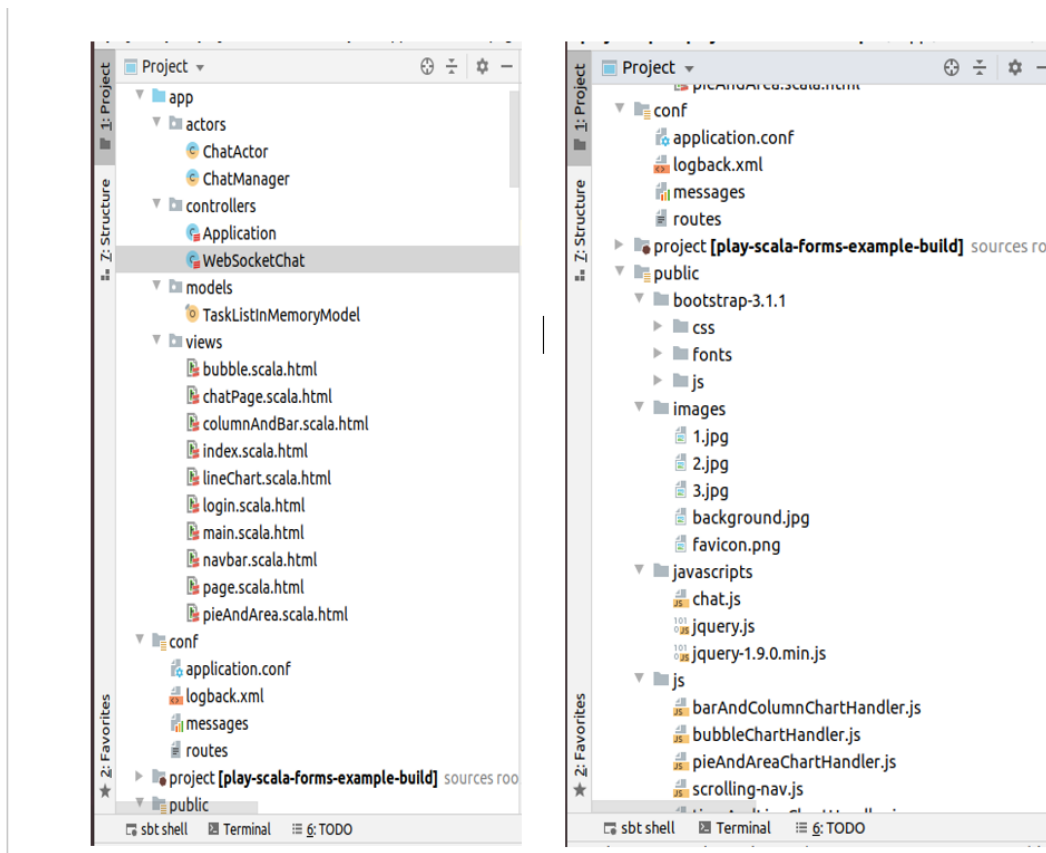



Figure 3.5 Project Structure

5.3.2 Using the SBT console –

You can manage the complete development cycle of a Play application with SBT. SBT has an interactive mode (shell), or you can enter commands one at a time. The interactive mode can be faster over time because SBT only needs to start once. When you enter commands one at a time, SBT restarts each time you run it.



```

pratham@pratham-HP-245-G5-Notebook-PC:~/Desktop/ISRO/ISRO trainee project/great work/completion of play/play-samples-play-scala-forms-example$ sbt run
[info] Loading global plugins from /home/pratham/.sbt/1.0/plugins
[info] Loading settings for project play-samples-play-scala-forms-example-build from plugins.sbt ...
[info] Loading project definition from /home/pratham/Desktop/ISRO/ISRO trainee project/great work/completion of play/play-samples-play-scala-forms-example/project
[info] Loading settings for project root from build.sbt ...
[info] Set current project to play-scala-forms-example (in build file:/home/pratham/Desktop/ISRO/ISRO%20trainee%20project/great%20work/completion%20of%20play/play-samples-play-scala-forms-example/)

--- (Running the application, auto-reloading is enabled) ---

[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:9000
(Server started, use Enter to stop and go back to the console...)
  
```

Figure 5.6 SBT Console

On running the SBT and building the project, the localhost opens on port 9000.

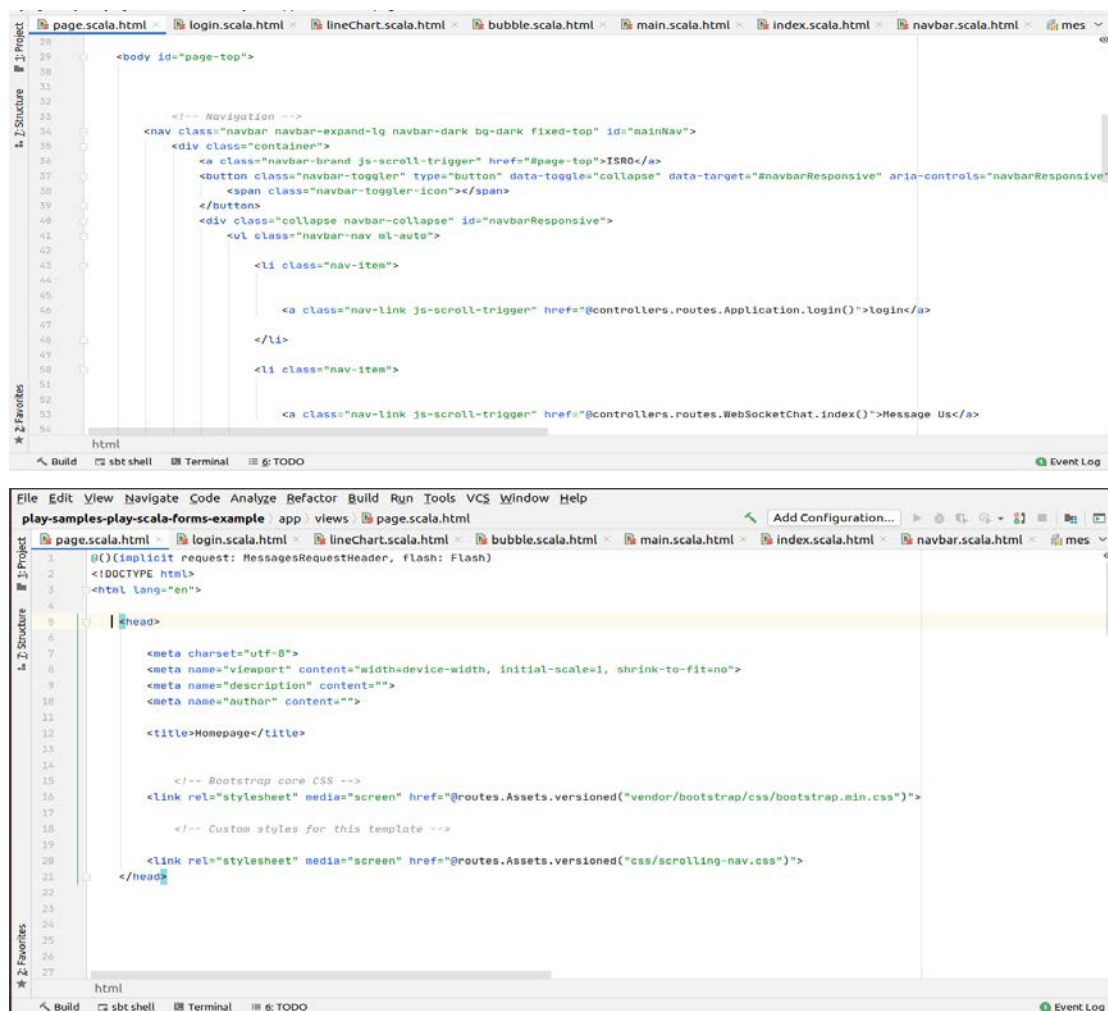


Figure 5.7 Localhost 9000

5.3.2 Homepage –

The entry point to the main UI is the Homepage. To gain access to the main UI, the user has to provide the login credentials.

On login verification of the client, the client is redirected to the charts page where a client has different options to view the data like line chart, bar chart, etc. Whenever we enter the url the application takes us to the homepage where a user is informed about the website and asked for a login.



The image shows two screenshots of an IDE (likely IntelliJ IDEA) displaying HTML code for a web application. The top screenshot shows the navigation bar code, and the bottom screenshot shows the page header code.

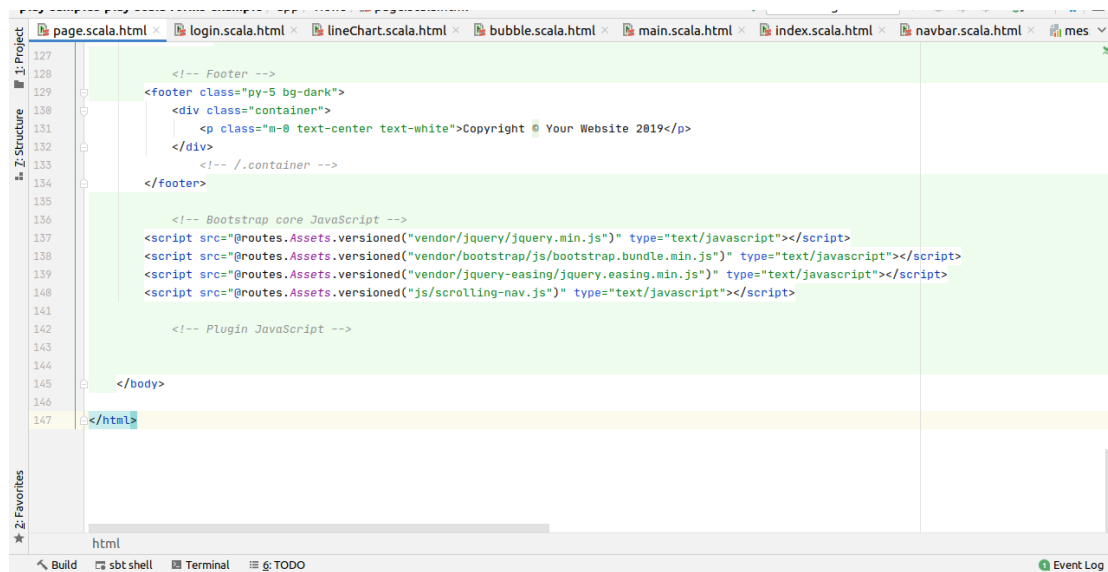
```

<body id="page-top">

  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top" id="mainNav">
    <div class="container">
      <a class="navbar-brand js-scroll-trigger" href="#page-top">ISR0</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive"
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
            <a class="nav-link js-scroll-trigger" href="@controllers.routes.Application.login()">login</a>
          </li>
          <li class="nav-item">
            <a class="nav-link js-scroll-trigger" href="@controllers.routes.WebSocketChat.index()">Message Us</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
  </body>
  
```

```

@()(Implicit request: MessagesRequestHeader, flash: Flash)
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Homepage</title>
    <!-- Bootstrap core CSS -->
    <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("vendor/bootstrap/css/bootstrap.min.css")">
    <!-- Custom styles for this template -->
    <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("css/scrolling-nav.css")">
  </head>
  
```



```

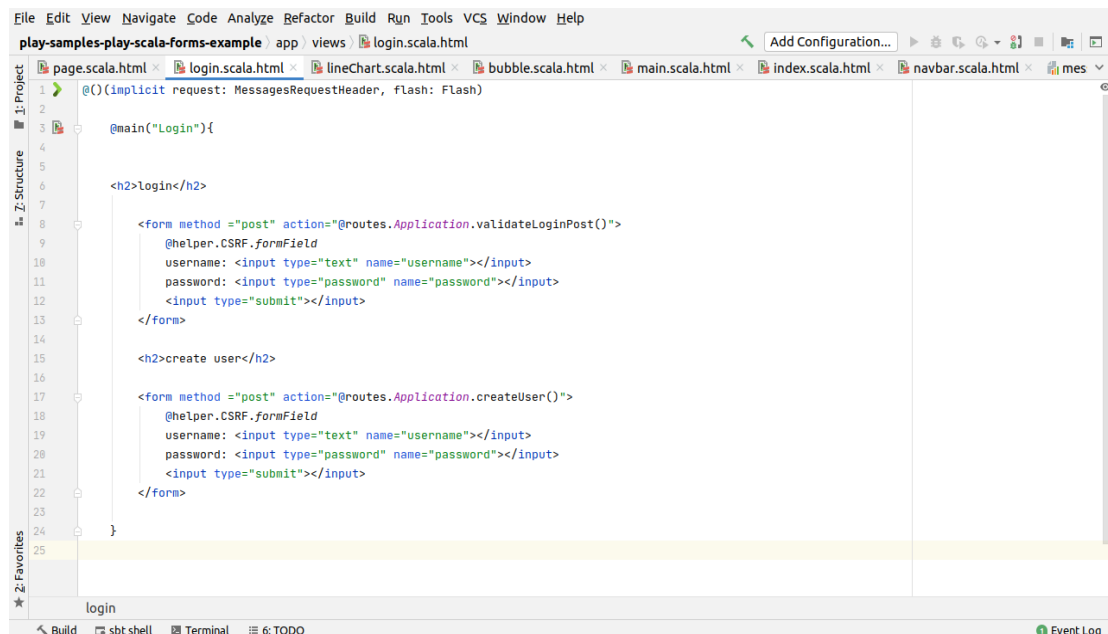
127
128
129 <!-- Footer -->
130 <footer class="py-5 bg-dark">
131   <div class="container">
132     <p class="m-0 text-center text-white">Copyright © Your Website 2019</p>
133   </div>
134 </footer>
135
136 <!-- Bootstrap core JavaScript -->
137 <script src="@routes.Assets.versioned("vendor/jquery/jquery.min.js")" type="text/javascript"></script>
138 <script src="@routes.Assets.versioned("vendor/bootstrap/js/bootstrap.bundle.min.js")" type="text/javascript"></script>
139 <script src="@routes.Assets.versioned("vendor/jquery-easing/jquery.easing.min.js")" type="text/javascript"></script>
140
141 <!-- Plugin JavaScript -->
142
143
144
145 </body>
146
147 </html>

```

Figure 5.8 Code for Homepage – Play Framework

5.3.3 Login page –

In the above code we can see that to make the application secure we have used a inbuilt feature of play framework i.e. CSRF form field which protect from unauthorised login.



```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
play-samples-play-scala-forms-example app | views | login.scala.html
@()(implicit request: MessagesRequestHeader, flash: Flash)
1
2
3 @main("Login"){
4
5
6
7 <h2>login</h2>
8
9 <form method="post" action="@routes.Application.validateLoginPost()">
10   @helper.CSRF.formField
11   username: <input type="text" name="username"></input>
12   password: <input type="password" name="password"></input>
13   <input type="submit"></input>
14 </form>
15
16 <h2>create user</h2>
17
18 <form method="post" action="@routes.Application.createUser()">
19   @helper.CSRF.formField
20   username: <input type="text" name="username"></input>
21   password: <input type="password" name="password"></input>
22   <input type="submit"></input>
23 </form>
24
25 }

```

Figure 5.9 Login page – Play Framework

5.3.4 Algorithm for Routing –

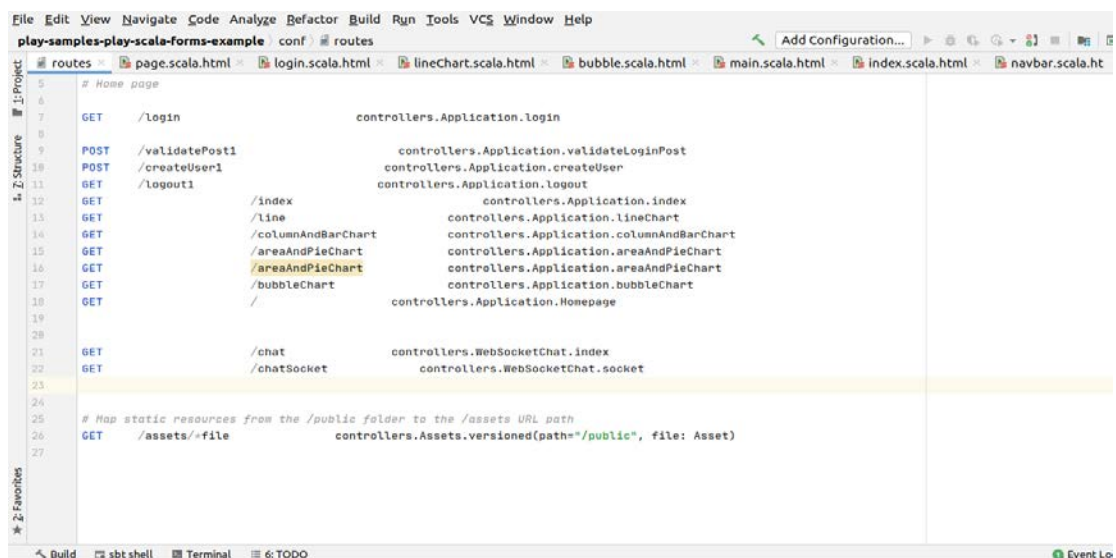
The router is the component in charge of translating each incoming HTTP request to an Action.

An HTTP request is seen as an event by the MVC framework. This event contains two major pieces of information

- the request path (e.g. /clients/1542, /photos/list), including the query string .
- the HTTP method (e.g. GET, POST, ...).

Steps -

1. create the routes file in config folder i.e. config/routes.
2. specify the type of request required to call the function i.e. for login function POST method is used to hide the user credentials in the URL
3. call for the respective function required for the specific method using reverse routing.



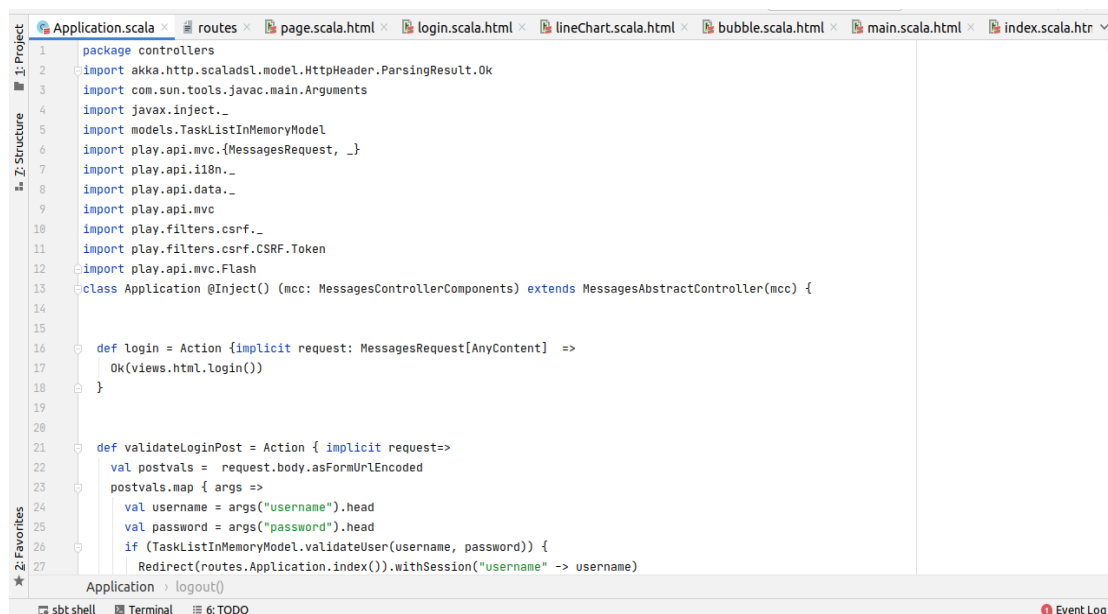
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
play-samples-play-scala-forms-example conf # routes
# routes
# Home page
5
6
7 GET /login controllers.Application.login
8
9 POST /validatePost1 controllers.Application.validateLoginPost
10 POST /createUser1 controllers.Application.createUser
11 GET /logout1 controllers.Application.logout
12 GET /index controllers.Application.index
13 GET /line controllers.Application.lineChart
14 GET /columnAndBarChart controllers.Application.columnAndBarChart
15 GET /areaAndPieChart controllers.Application.areaAndPieChart
16 GET /areaAndPieChart controllers.Application.areaAndPieChart
17 GET /bubbleChart controllers.Application.bubbleChart
18 GET / controllers.Application.Homepage
19
20
21 GET /chat controllers.WebSocketChat.index
22 GET /chatSocket controllers.WebSocketChat.socket
23
24
25 # Map static resources from the /public folder to the /assets URL path
26 GET /assets/*file controllers.Assets.versioned(path="*/public", file: Asset)
27
  
```

Figure 5.10 Routing file

5.3.5 Algorithm for Controller –

1. create an Application controller in the app/controller folder.
2. Create a method /function that calls the respective views or model as per the requirement.
3. To create a new function add “def” before the function name where def stands for defining the new function.
4. Pass the respective Messages as described in the messages folder in the config file i.e. config/messages in the Return argument.



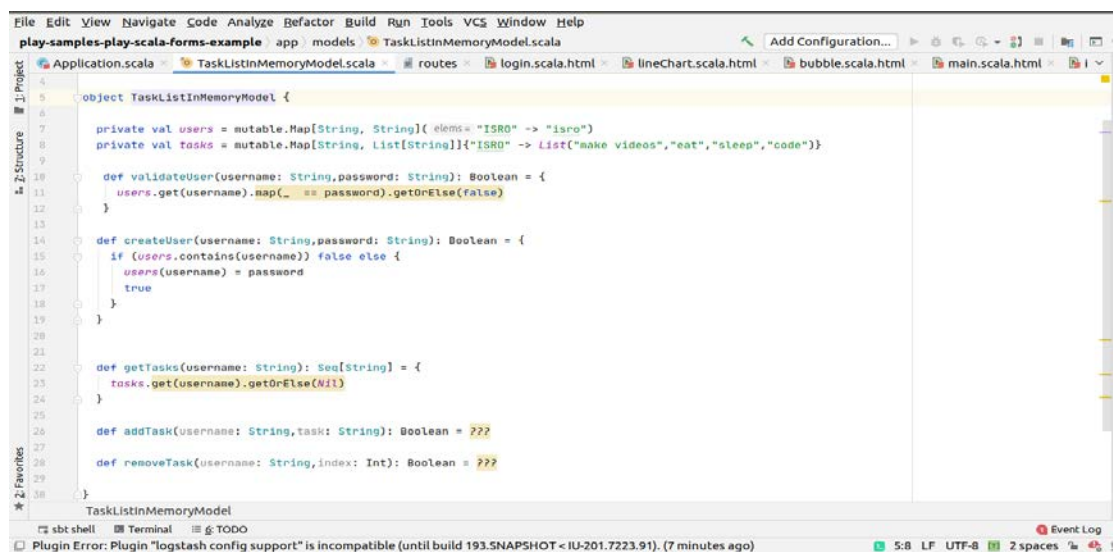
```

1 package controllers
2 import akka.http.scaladsl.model.HttpHeader.ParsingResult.Ok
3 import com.sun.tools.javac.main.Arguments
4 import javax.inject._
5 import models.TaskListInMemoryModel
6 import play.api.mvc.{MessagesRequest, _}
7 import play.api.i18n._
8 import play.api.data._
9 import play.api.mvc
10 import play.filters.csrf._
11 import play.filters.csrf.CSRF.Token
12 import play.api.mvc.Flash
13 class Application @Inject() (mcc: MessagesControllerComponents) extends MessagesAbstractController(mcc) {
14
15
16   def login = Action {implicit request: MessagesRequest[AnyContent] =>
17     Ok(views.html.login())
18   }
19
20
21   def validateLoginPost = Action { implicit request=>
22     val postvals = request.body.asFormUrlEncoded
23     postvals.map { args =>
24       val username = args("username").head
25       val password = args("password").head
26       if (TaskListInMemoryModel.validateUser(username, password)) {
27         Redirect(routes.Application.index()).withSession("username" -> username)
28       }
29     }
30   }
31 }
  
```

Figure 5.11 Controller code

5.3.6 Algorithm for Model –

1. create a package models in the application folder.
2. create a new file as the TaskInlistMemory where we execute the logic part as per the requirements.
3. Create a method for Validate User where we provide the logic for successful Authentication.



```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
play-samples-play-scala-forms-example app models TaskListInMemoryModel.scala Add Configuration...
Application.scala TaskListInMemoryModel.scala routes login.scala.html lineChart.scala.html bubble.scala.html main.scala.html
4
5 object TaskListInMemoryModel {
6
7   private val users = mutable.Map[String, String](elems = "ISRO" -> "isro")
8   private val tasks = mutable.Map[String, List[String]]{"ISRO" -> List("make videos","eat","sleep","code")}
9
10  def validateUser(username: String,password: String): Boolean = {
11    users.get(username).map(_ == password).getOrElse(false)
12  }
13
14  def createUser(username: String,password: String): Boolean = {
15    if (users.contains(username)) false else {
16      users(username) = password
17      true
18    }
19  }
20
21
22  def getTasks(username: String): Seq[String] = {
23    tasks.get(username).getOrElse( Nil)
24  }
25
26  def addTask(username: String,task: String): Boolean = ???
27
28  def removeTask(username: String,index: Int): Boolean = ???
29
30 }
  
```

Figure 5.12 Model code

CHAPTER 6

RESULTS AND DISCUSSION

The project aimed to achieve a base dummy model so that the ISRO team can improvise the code to suit their requirements. Here are the attached screenshots of the various solutions provided.

6.1 Traditional Web Application based UI –

6.1.1 Login and registration page – This is the first page that the user sees on launching the application. The user can view the main pages and graphs if the existing project is already configured. If the user enters a new project ID and key, the admin should be contacted so that the necessary routes can be set up.

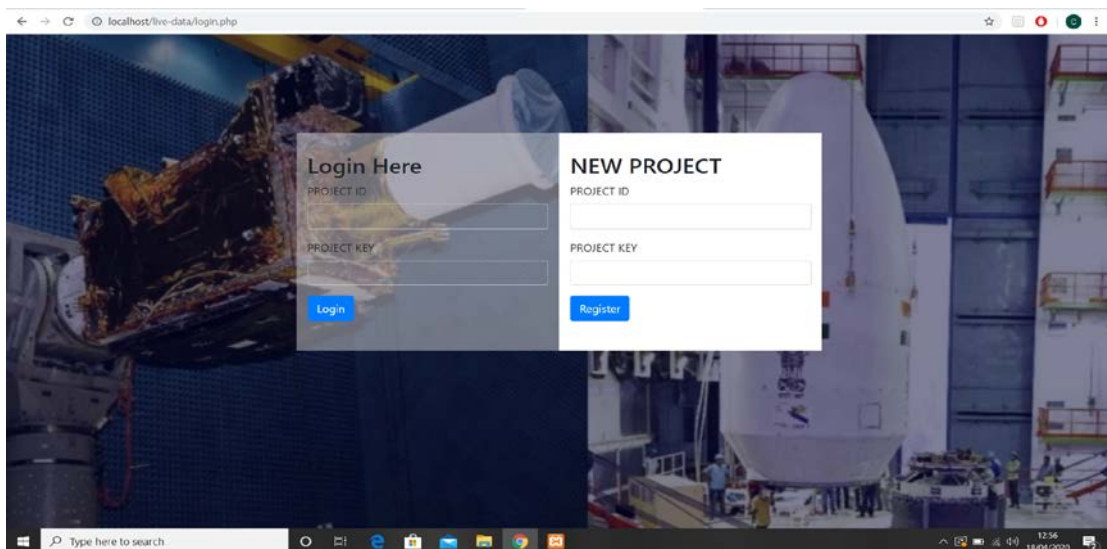


Figure 6.1 Login and registration page

6.1.2 The main UI which has the static data table which has data populated from local file -



Figure 6.2 Spline graph and data table

6.1.3 Pie graph and spline graph which get their data from a common data source

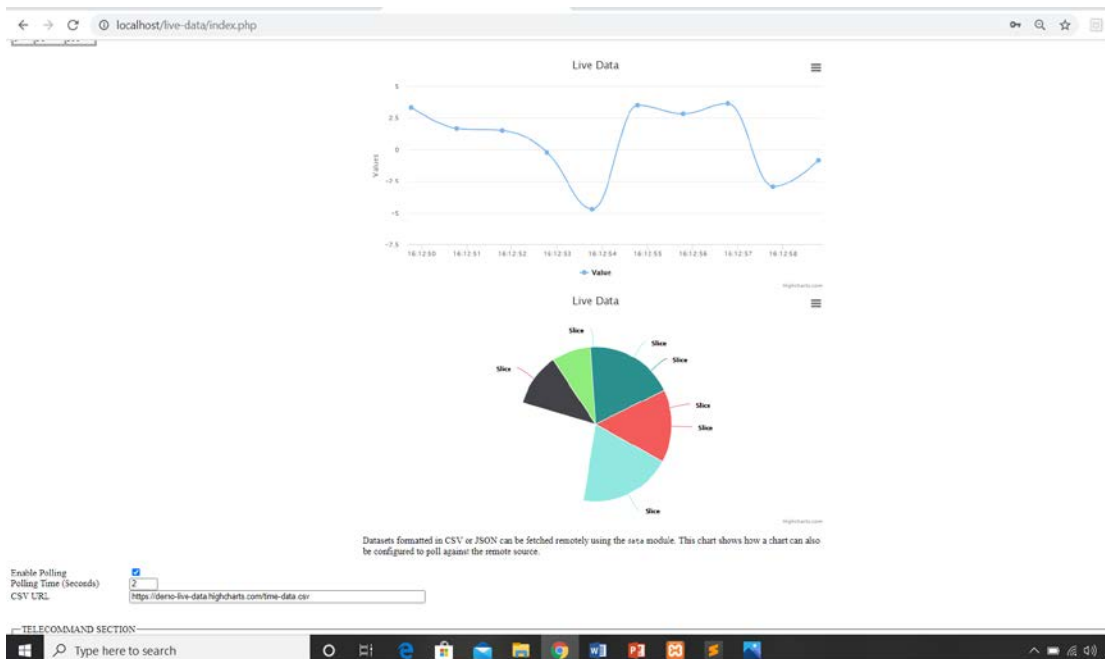


Figure 6.3 Spline and Pie graph

6.1.4 Telecommand section – This section is used to take the input from the user and update the contents in a file after validations.

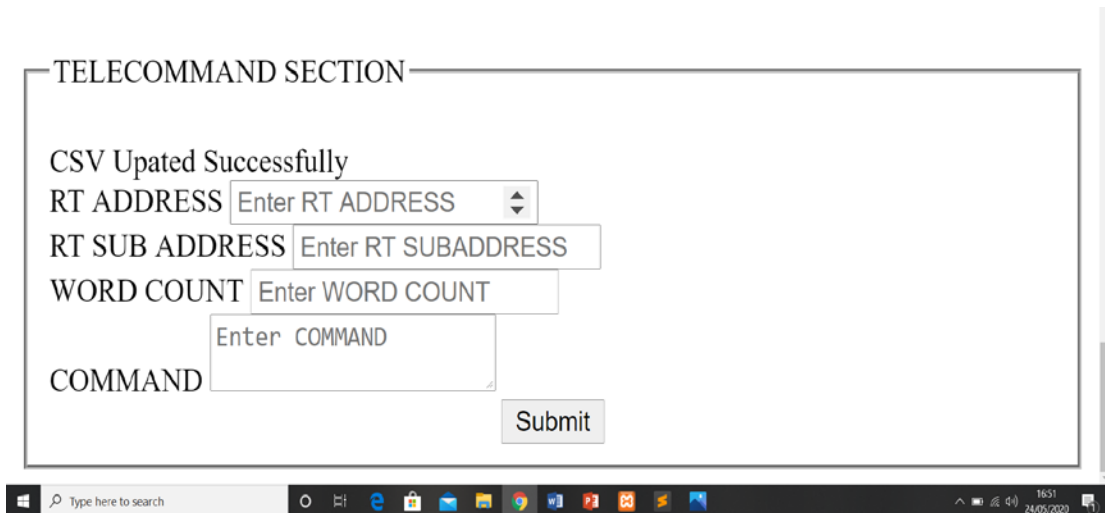


Figure 6.4 Telecommand Section

6.1.5 CSV file containing the entered Telecommand along with the RT Address and RT SubAddress – This is the resultant file which we obtain after submitting the Telecommand form.

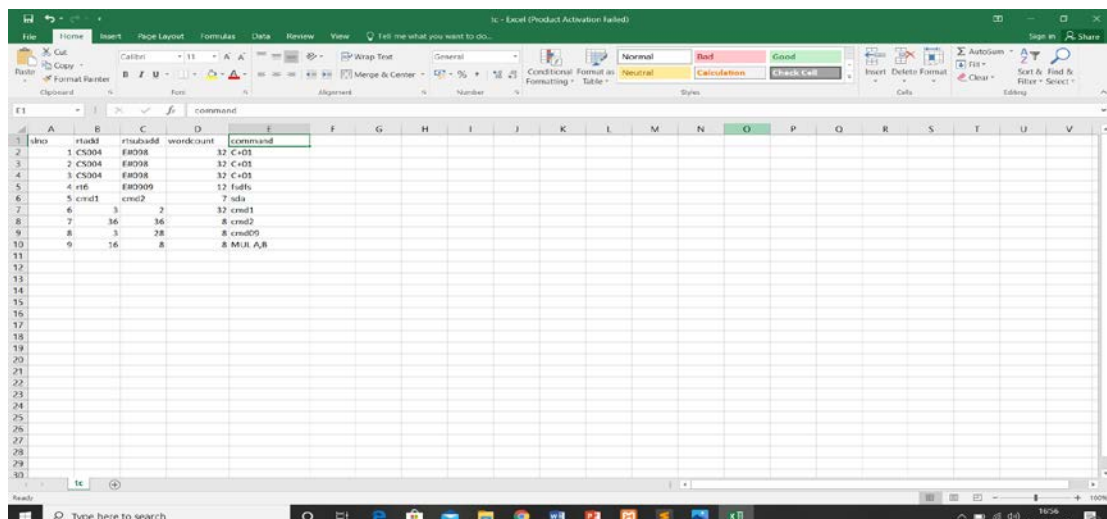


Figure 6.5 Telecommand file

6.2 JAVA program screenshots –

```

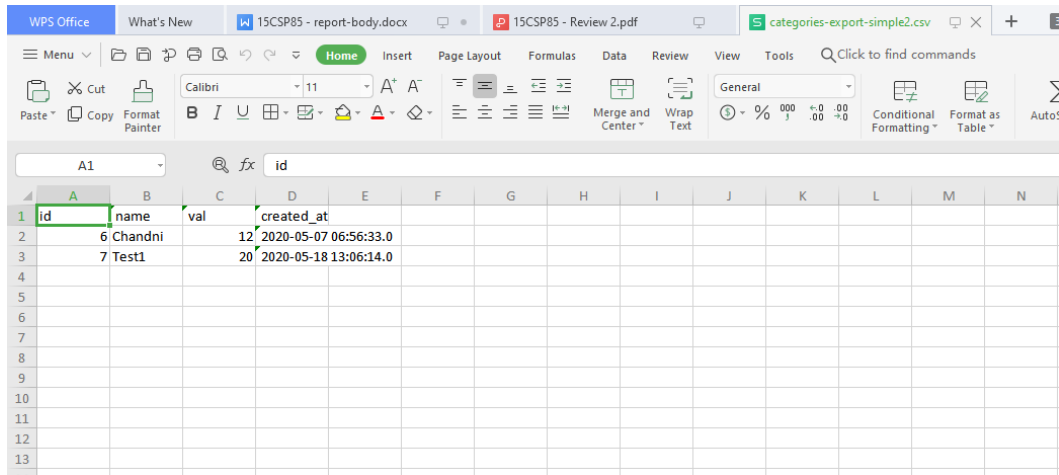
1 package net.code.java;|
2
3 import java.io.*;|
4
5
6 public class SimpleDb2CsvExporter {
7
8     public static void main(String[] args) {
9         String jdbcURL = "jdbc:mysql://localhost:3306/javascv";
10        String username = "root";
11        String password = "";
12
13        String csvFilePath = "categories-export-simple.csv";
14
15        try (Connection connection = DriverManager.getConnection(jdbcURL, username, password)) {
16            String sql = "SELECT * FROM (SELECT * FROM categories ORDER BY id DESC LIMIT 2) sub";
17
18            Statement statement = connection.createStatement();
19
20            ResultSet result = statement.executeQuery(sql);
21
22            BufferedWriter fileWriter = new BufferedWriter(new FileWriter(csvFilePath));
23
24            // write header line containing column names
25            fileWriter.write("id, name, val, created_at");
26
27            while (result.next()) {
28                int id = result.getInt("id");
29                String name = result.getString("name");
30                int val = result.getInt("val");
31                Timestamp created_at = result.getTimestamp("created_at");
32
33
34                String line = String.format("%d,%s,%d,%s",
35                    id, name, val, created_at);
36
37                fileWriter.newLine();
38                fileWriter.write(line);
39            }
40
41            statement.close();
42            fileWriter.close();
43
44        } catch (SQLException e) {
45            System.out.println("Database error:");
46            e.printStackTrace();
47        } catch (IOException e) {
48            System.out.println("File IO error:");
49            e.printStackTrace();
50        }
51    }
52 }
53
54 }
55

```

Figure 6.6 JAVA Program

UI FOR LIVE MONITORING OF ONBOARD STORAGE SYSTEM

The resultant CSV file which gets updated periodically with the database contents is shown below.



id	name	val	created_at
6	Chandni	12	2020-05-07 06:56:33.0
7	Test1	20	2020-05-18 13:06:14.0

Figure 6.7 CSV file - result

6.3 Play FrameWork based Solution –

6.3.1 Login frame – This is a basic login frame. It uses CSRF to incorporate security and validations which is a feature of Play framework.

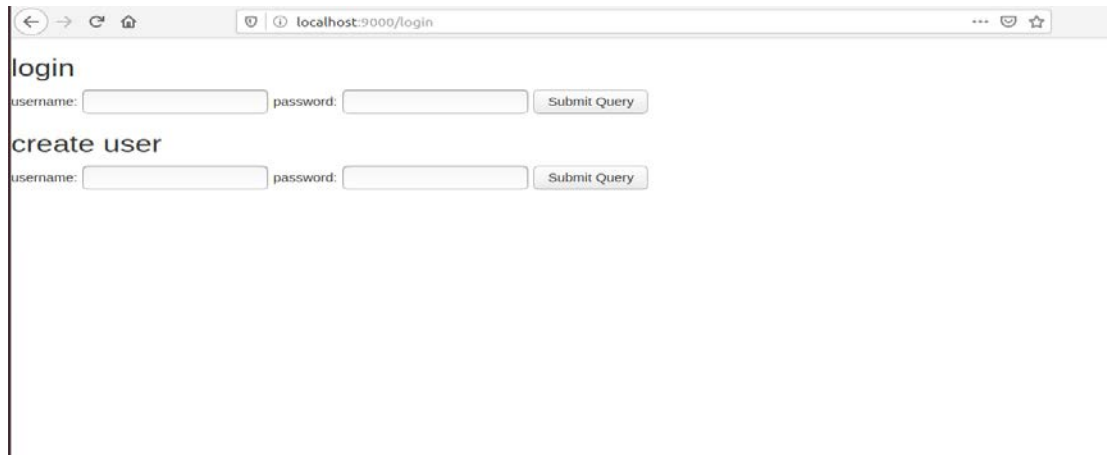


Figure 6.8 Login page – Play framework

6.3.2 Home Page – The first page that the user sees where the basic information can be found.

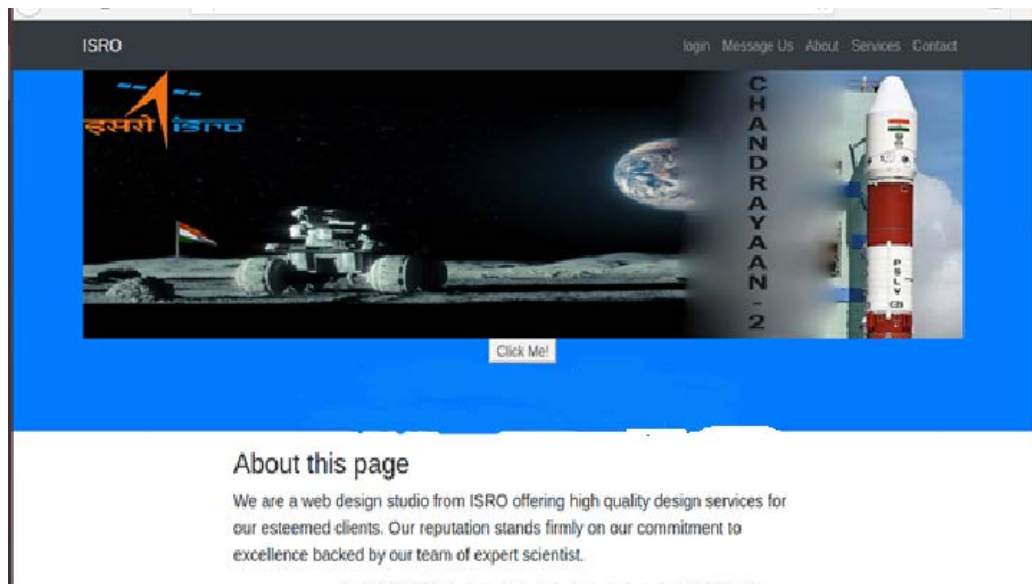


Figure 6.9 Home Page – Play Framework

6.3.3 Graphs and main UI – This is the main page where different graphs are plotted.

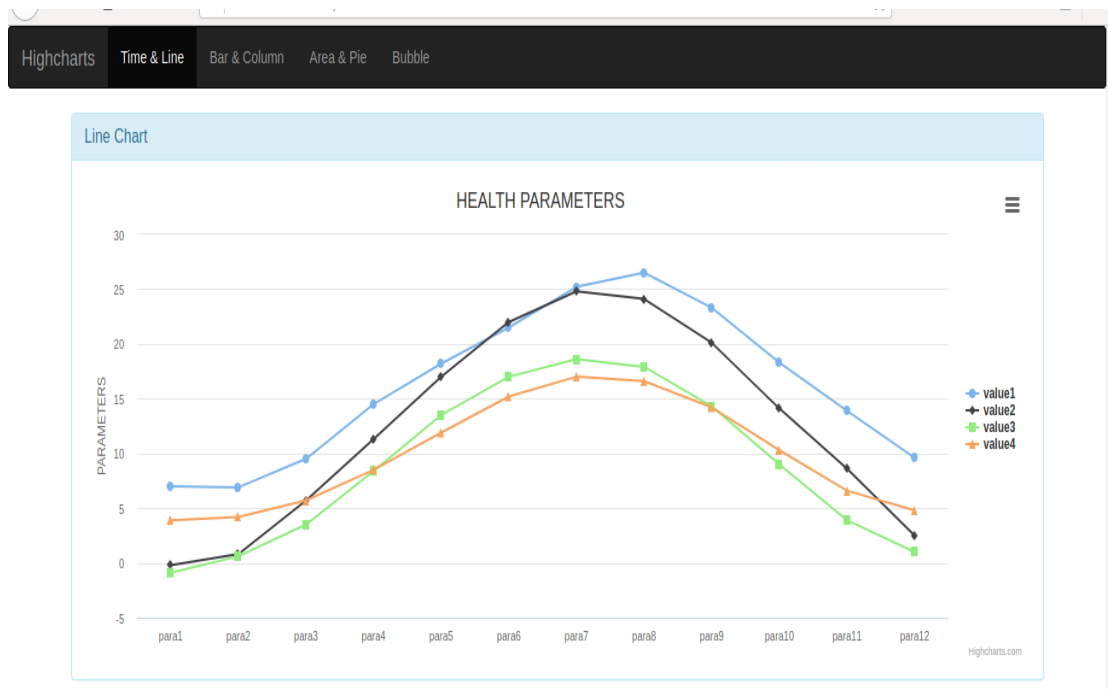


Figure 6.10 – Spline graph – Play Framework



Figure 6.11 – Chart - Play Framework

CHAPTER 7

TESTING

The project can be disintegrated into 3 main parts, each performing a different function.

The following tests were performed –

1. Unit Testing – Each of these were tested locally on the system ensuring that corner cases also are handled.
2. Integrated Testing – The three parts were tested to ensure that they work coherently and that there are no compatibility issues with respect to versions and other dependencies.
3. Multiple logins – This test is just to ensure that the content that is being viewed in one user's system is independent of the others though multiple users might be using the same project's login.
4. Database Server – Tests were done to see if this situation is handled where the server is down or there is some connection failure.
5. CSV Permission – Different scenarios for which the permission of the CSV file was checked and whether the CSV files already exists or a new one is created every time.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

From the theoretical survey done in the previous semester, we understood the model being used for the analysis of data and how the data transmission happens in the industry. Over the course of the project we have also gained a complete understanding of the feasible solutions. A lot of R&D had been done to come up with architectures suitable for our project but also making sure it meets the requirements and constraints specified.

The main objective of the project was to provide a base model which is generic and can be used for further projects. The first two months of the project was utilized to understand the technical constraints and to decide the code base that can be used.

Post which we came up with the plan of providing 2 solutions – the traditional Web based approach and the Play FrameWork architecture. The streaming of live data as a graph was a challenge as we had to find a suitable platform which is feature rich, so we went ahead with Highcharts. The java program is the intermediate part which will act as a bridge between the 2 solutions by populating the necessary data for plotting from the database. The database is populated by the ZMQ protocol which receives data as packets from the hardware.

The project gave us an insight into the industry expectations and how to work together in a team with the deadlines and to suit the project demo to the meet the requirements.

8.2 Future Scope

The solutions developed use basic concepts and tools. The project can definitely be developed using more complex tools and concepts and probably be coded in more complex languages and frameworks.

The focus of the project could definitely be shifted towards data manipulation and data analytics.

Concepts such as Machine Learning can be implemented to provide a predictive solution from the plotted data.

A better user interface can also be designed in the near future which will enhance the user experience along with introducing more features to make the application more feature-rich.

Along with the two solutions provided, more architectures can also be ventured into to develop the application to provide a broader spectrum of solutions.

REFERENCES

- [1] Flow control for onboard solid state recorder (IEEE paper) by Siddhartha B. Rai, Srinidhi M.S., Kuruvilla Varghese, Srividhya R
<https://ieeexplore.ieee.org/document/8067838>
- [2] Automated verification of spacecraft telemetry data (IEEE paper) by A. Savitha, M. Ravindra, Prasanna Kumar N, Rajiv R. Chetwani, K.M. Baradwaj (ISRO)
<https://ieeexplore.ieee.org/document/7238391>
- [3] Highcharts website for installation and documentation –
<https://www.highcharts.com/>
- [4] Documentation and setup for Play framework –
<https://www.playframework.com/>
- [5] Basics about Maven projects –
https://www.tutorialspoint.com/maven/maven_overview.htm
- [6] Programming examples and tutorials -
<https://www.tutorialspoint.com/scala/index.htm>
<https://plotly.com/scala/>

