A black background with blue text

Description automatically generated

WIX1002 FUNDAMENTALS OF PROGRAMMING

**SEMESTER II, SESSION 2023/24**

**GROUP PROJECT : TECHNICAL REPORT**

GROUP NAME : CODE4ERROR, (OCC 2)

|  |  |
| --- | --- |
| **NAME** | **MATRIC NUMBER** |
| 1. SARAH ANIS RAGEH AL-MALEKI | 23061281 |
| 1. MUHAMMAD AZKA HASYIM | 22118852 |
| 1. SYAHID AKBAR PASYA | 22106687 |
| 1. MD. MUINUL ISLAM | 22111481 |
| 1. SIDIQ MUSLEH BIN IQBAL FITHRI | 22006596 |

**COURSE INSTRUCTOR**

PROFESSOR MADYA DR. LIEW CHEE SUN

TABLE OF CONTENTS

[PROJECT TIMELINE 3](#_Toc169217301)

[REQUIREMENT ANALYSIS 4](#_Toc169217302)

[PROBLEMS / ISSUES FACED 5](#_Toc169217303)

[SOURCE CODE WITH EXPLANATION 6](#_Toc169217304)

[GRAPHS AND CHARTS 23](#_Toc169217305)

# **PROJECT TIMELINE**

|  |  |
| --- | --- |
| **WEEK** | **PROGRESS** |
| **07** | * Formation of assignment team of 5 members * Analyzed the extracted\_log file to gain a comprehensive understanding and statistical insights. |
| **09** | * Held online meeting to divide tasks via Google Meet * Defined deadlines for each task. |
| **11** | * Finalized decision on the additional feature to incorporate. * Conducted debugging sessions on the codebase. |
| **12** | * Organized meeting to compile the code. * Continued debugging and refinement of the code. |
| **13** | * Conducted meeting to generate graphs using JavaFX. * Worked on drafting the project reports. |
| **14** | * Prepared for Project Presentation. |

# **REQUIREMENT ANALYSIS**

KEY METRICS

* Number of jobs created/ended within a given time range.
* Number of jobs by partitions i.e. EPYC, Opteron and GPU.
* Number of jobs causing error and the corresponding user.
* Average execution time of the jobs submitted to UMHPC.
* Write a program to extract the useful information and show in table and graph.

ADDITIONAL FEATURES

* Number of jobs completed with error and without errors.
* Number of jobs killed.
* Jobs By Partition, Node
* Total Execution Time Per Partition

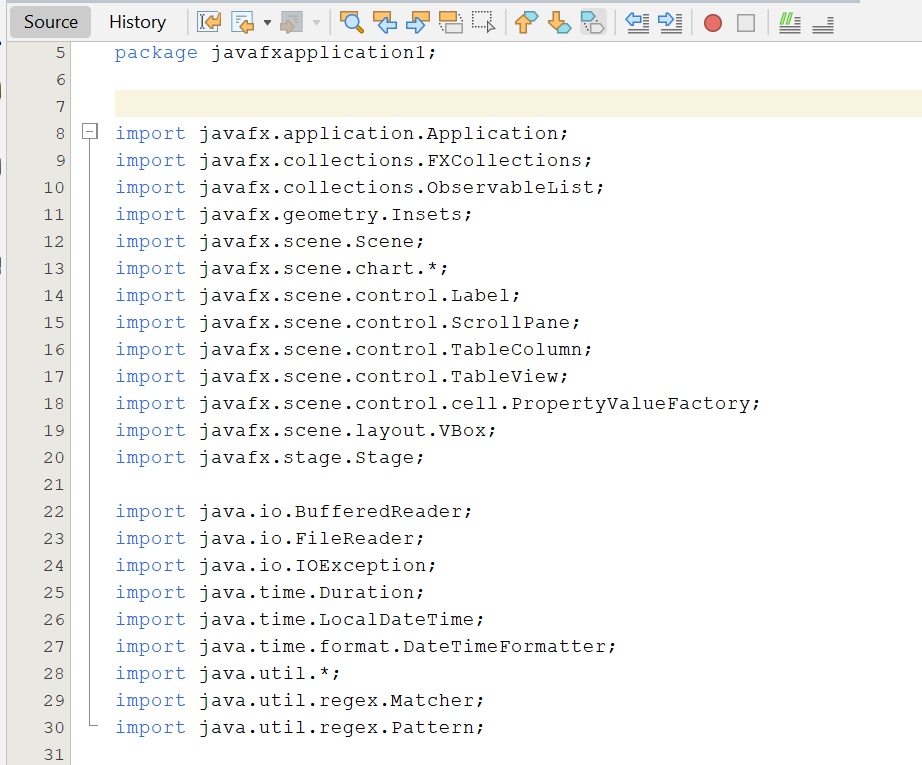
**Extract Performance Metrics**: By Utilizing extracted\_log file to capture and document key insights derived from system monitoring. Effective monitoring and log analysis are critical for maintaining the quality and stability of HPC services at DICC.

# **PROBLEMS / ISSUES FACED**

|  |  |  |
| --- | --- | --- |
| **NO.** | **CHALLENGE** | **SOLUTION** |
| **1** | Encountered unfamiliar keywords and functions. | Dedicated time to research each new concept online for thorough understanding |
| **2** | Difficulty in installing JavaFX application. | Watched instructional videos to properly learn installation procedures and usage techniques |
| **3** | Needed to display output in charts using Java after installing JavaFX. | Conducted extensive online research to learn how to effectively implement chart display functionalities. |
| **4** | Difficulty in accurately matching statistics due to the large size of log files. | Utilized Java for parsing log files, implementing efficient parsing algorithms to extract and process relevant statistics. Employed methods such as data filtering and aggregation to manage and analyze large datasets effectively. |
| **5** | Identifying and analyzing statistics that are crucial for the project's objectives. | Leveraged GRAPE (Graph Analysis and Exploration) tools to visualize and analyze the extracted statistics. Utilized GRAPE's capabilities to create meaningful visual representations of data trends and patterns, aiding in the identification of key statistics relevant to the project goals. |

# **SOURCE CODE WITH EXPLANATION**

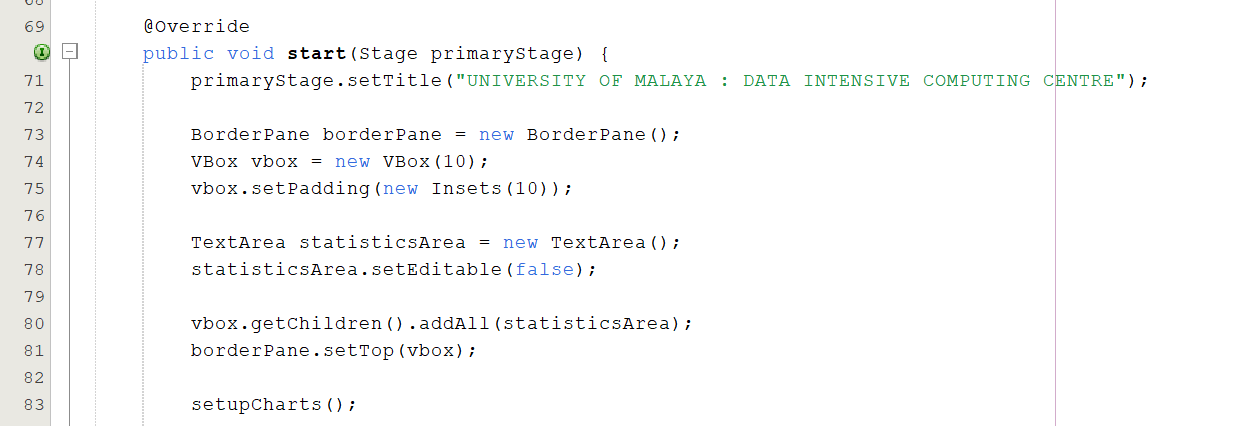
**Code Snippet & Explanation of MainFeatures.java**





* A screenshot of a computer code

  Description automatically generatedlogFilePath: Path to the log file to be processed.
* partitions, partitionCounts: Lists to store partition names and their corresponding job counts.
* startTime, endTime: Variables to store the start and end times of job logs.
* createdCount, endedCount, errorJobs, completedWithZero completedWithPositive, killedJobs: Counters for various job statuses.
* userErrorCount: Map to store error counts per user.
* totalExecutionTimeSeconds, jobCount: Variables to track total execution time and number of jobs.
* Calls launch(args) to start the JavaFX application.





* setupCharts(): Initializes the four charts (executionTimeBarChart, jobsPerPartitionBarChart, jobsPerNodeBarChart, jobDurationLineChart) using the createBarChart() method.
* CategoryAxis and NumberAxis: Sets up the x-axis and y-axis for the line chart with appropriate labels.
* createBarChart(): A helper method to create and return a BarChart with specified x-axis and y-axis labels and a title. This method is reused to create multiple bar charts with different labels and titles.

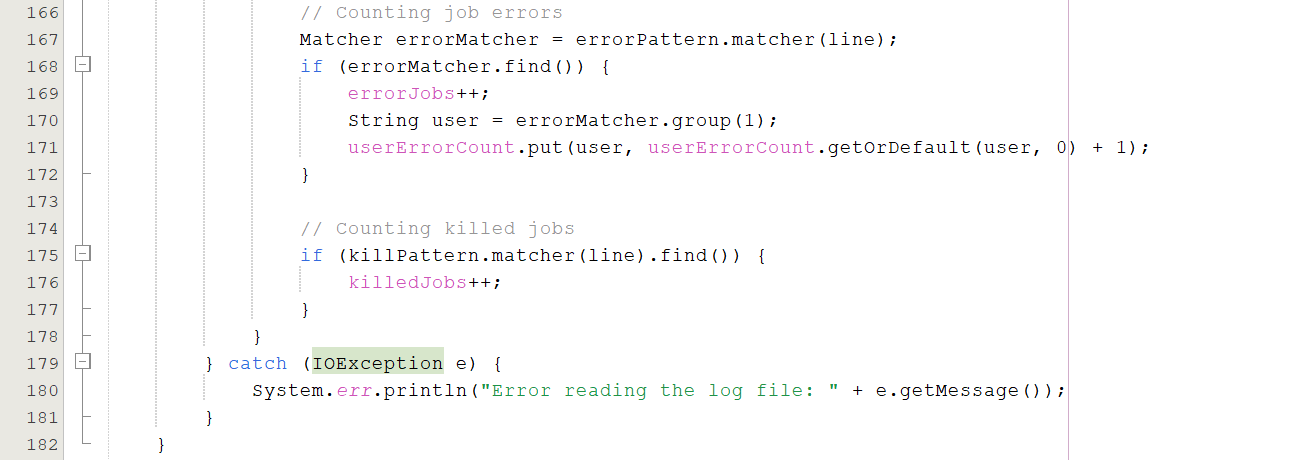


* updateStatistics(): Updates a TextArea with various job statistics.
* Clearing Previous Data: Clears the TextArea to remove any previous content.
* Average and Maximum Execution Time: Appends average and maximum execution times calculated using methods from the Job class.
* Jobs by Partition: Appends information about the number of jobs in each partition.
* Jobs by Node: Appends information about the number of jobs per node.
* Jobs by Exit Code: Appends information about the number of jobs per exit code.
* Total Execution Time per Partition: Appends the total execution time for each partition.

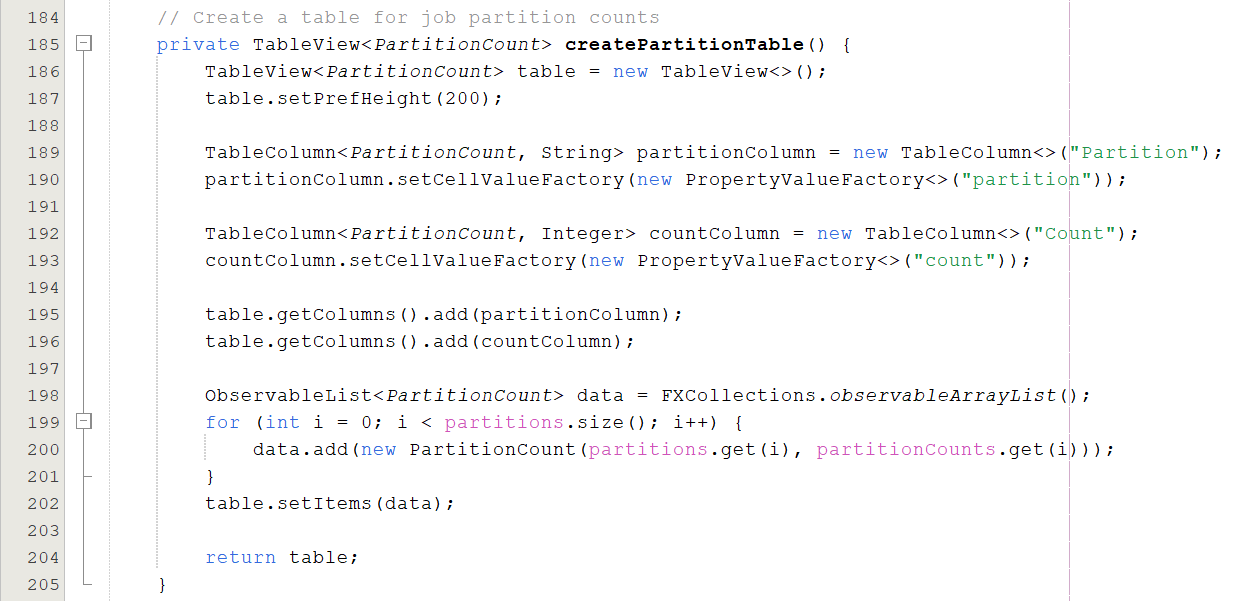
A screen shot of a computer

Description automatically generated

* Parses timestamps (timestamp) from the log line.
* Updates startTime and endTime to track overall job execution timeframe.
* Counts job submissions (submitPattern) and completions (completePattern).
* Parses exit status using exitStatusPattern and categorizes jobs based on their completion status (completedWithZero, completedWithPositive).
* Calculates job execution time using calculateExecutionTime method.
* Updates totalExecutionTimeSeconds with the duration in seconds and increments jobCount.



* Uses errorPattern to match lines indicating job errors.
* Increments errorJobs for each error found.
* Retrieves user information from the error line (userErrorCount map) and increments error count for that user.
* Counts jobs that were killed using killPattern.



* Sets up a TableView to display job partition counts.
* Creates columns (partitionColumn and countColumn) for partition and count.
* Populates data into the table using partitions and partitionCounts.

A screenshot of a computer program

Description automatically generated

* Sets up a TableView to display user error counts.
* Creates columns (userColumn and errorCountColumn) for user and errorCount.
* Populates data into the table using userErrorCount.

A screenshot of a computer program

Description automatically generated

* Sets up a BarChart to visualize job partition counts
* Creates xAxis (CategoryAxis) and yAxis (NumberAxis) for partition and count.
* Sets chart title, axis labels, and populates data series (dataSeries) with partitions and partitionCounts.
* Change the color of the bar chart.

A screenshot of a computer program

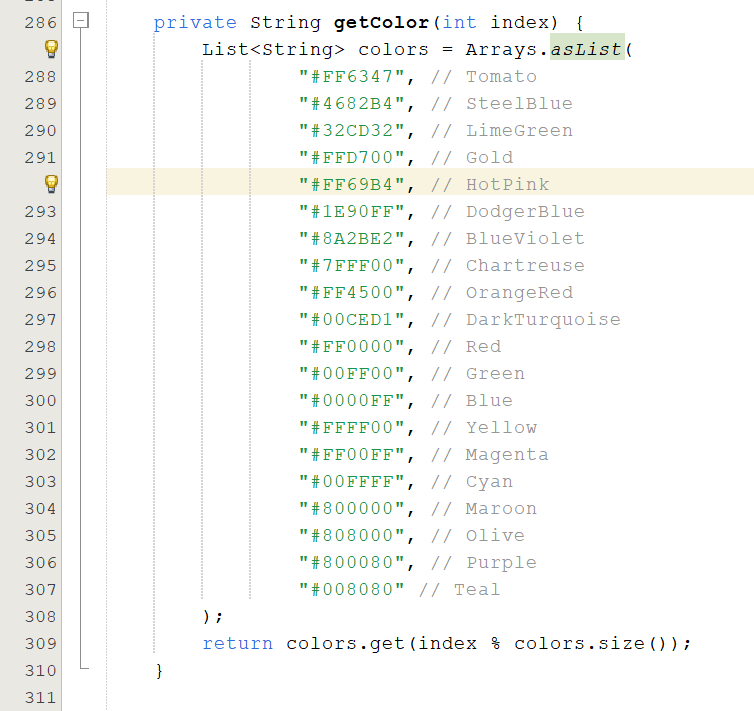
Description automatically generated

* Sets up a BarChart to visualize user error counts.
* Creates xAxis (CategoryAxis) and yAxis (NumberAxis) for user and errorCount.
* Sets chart title, axis labels, and populates data series (dataSeries) with userErrorCount.
* Change the color of the bar chart.

A screen shot of a computer code

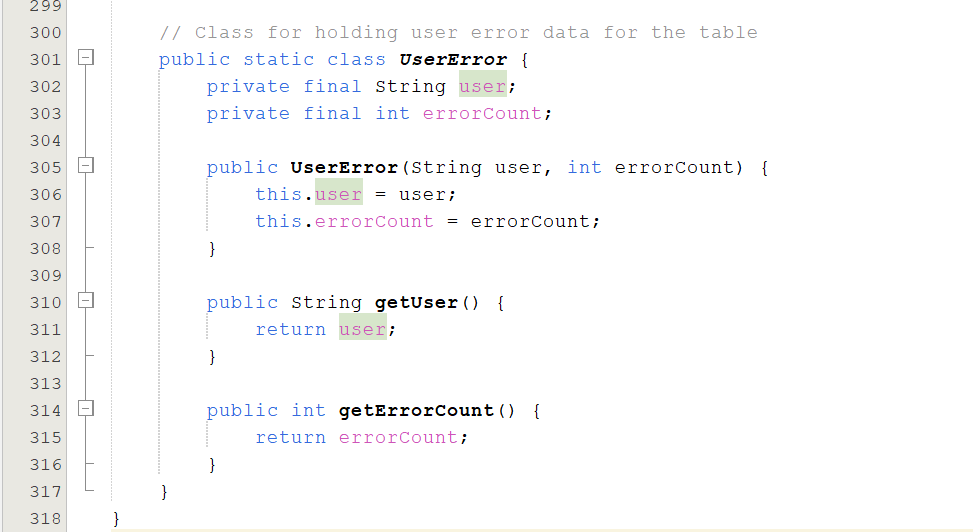
Description automatically generated

* Provides utility methods for calculating execution time between timestamps and formatting durations.

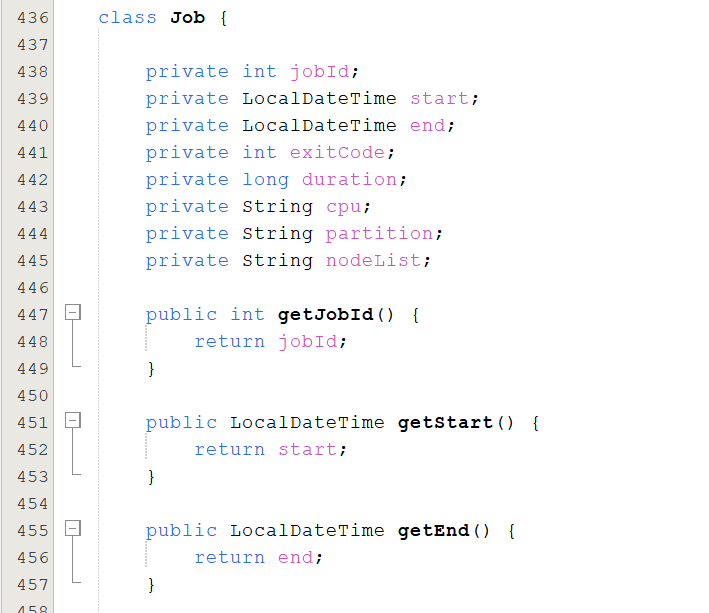
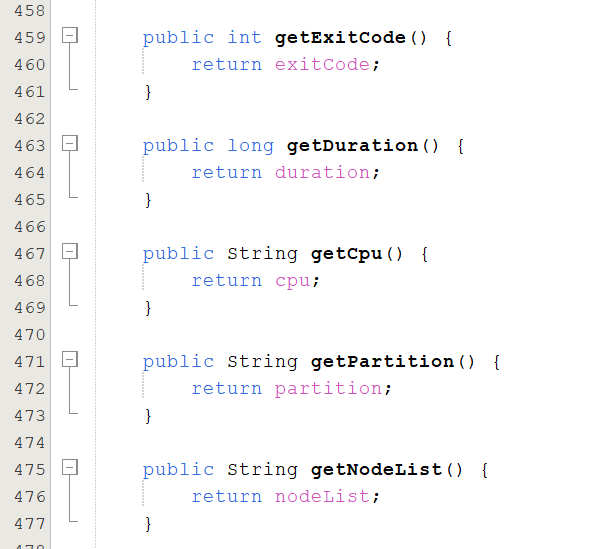


A screen shot of a computer code

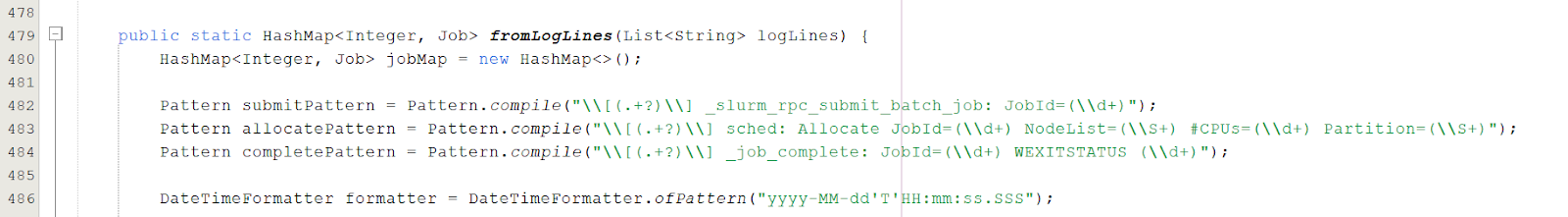
Description automatically generated

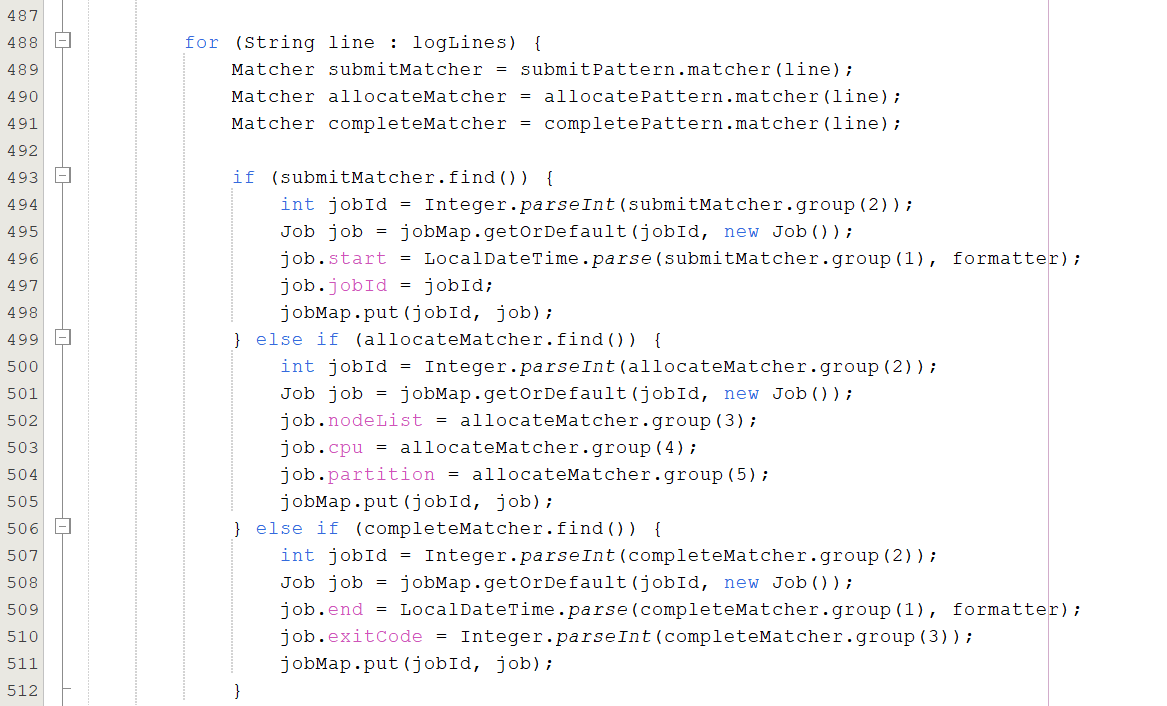


* The getColor(int index) method returns a hexadecimal color code from a predefined list based on the provided index, ensuring cyclic selection with modulo operation.
* Defines inner classes (PartitionCount and UserError) to encapsulate data for TableView visualization.

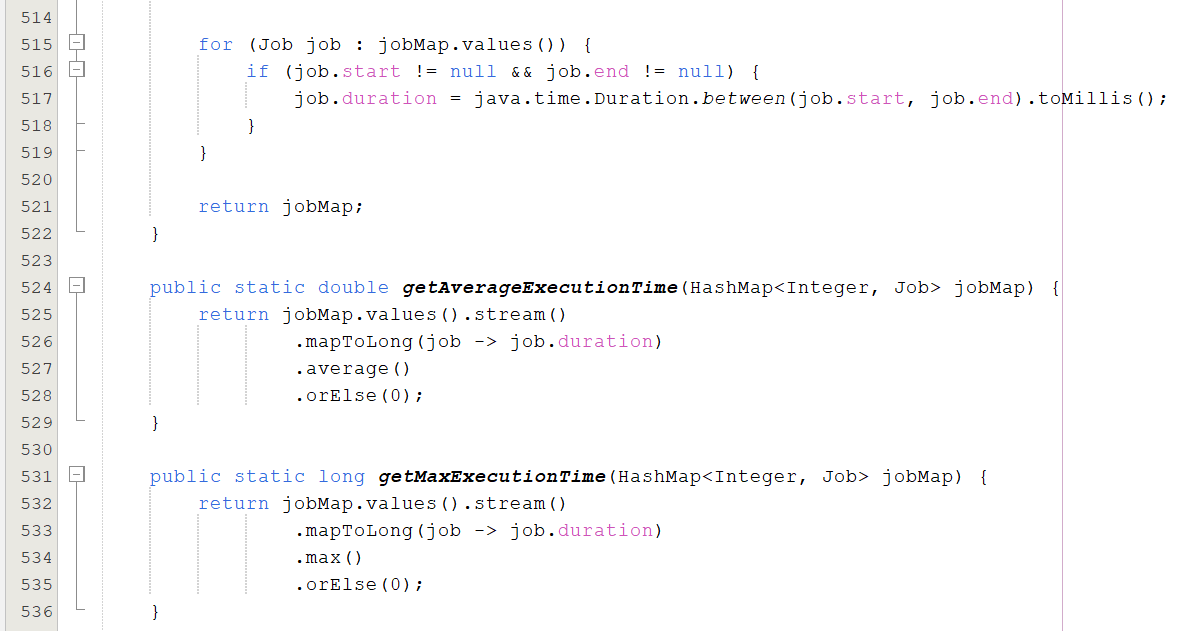


* The Job class has several member variables representing the properties of a job
* Getter Methods: Provide access to the private member variables. These methods are standard in Java for accessing private fields from outside the class.





* fromLogLines(): Parses a list of log lines and creates a map of Job objects indexed by job ID.
* Patterns: Defines three regex patterns to match different types of log entries (submission, allocation, and completion).
* DateTimeFormatter: Defines a formatter to parse date-time strings in the log.
* Log Parsing Loop: Iterates through each log line, matches it against the patterns, and updates the Job objects accordingly:
* Submission: Sets the start time and job ID.
* Allocation: Sets the node list, CPU count, and partition.
* Completion: Sets the end time and exit code.
* Duration Calculation: After parsing, calculates the duration for each job that has both a start and an end time.



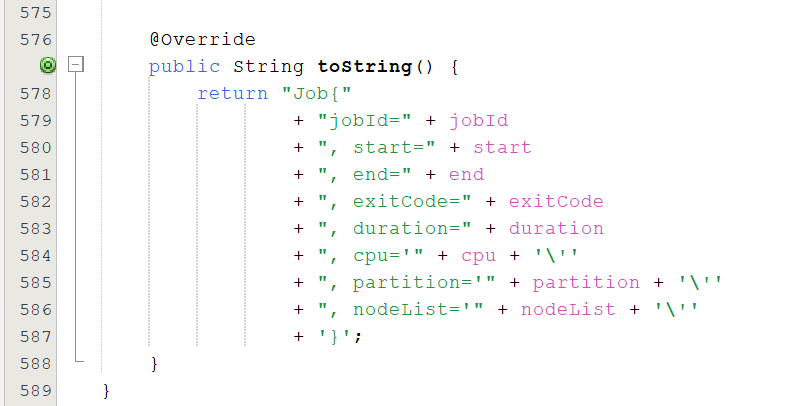
* getAverageExecutionTime(): Computes the average execution time of all jobs.
* getMaxExecutionTime(): Finds the maximum execution time among all jobs.
* getMinExecutionTime(): Finds the minimum execution time among all jobs.

A screenshot of a computer code

Description automatically generated



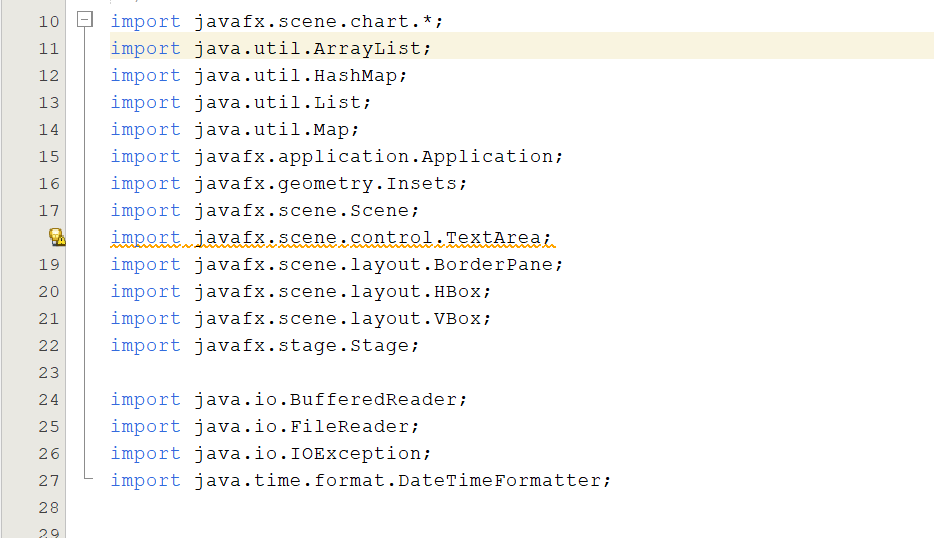
* getJobsByPartitionCount(): Groups jobs by partition and counts the number of jobs in each partition.
* getJobsByNodeCount(): Groups jobs by node list and counts the number of jobs on each node.
* getJobsByPartition(): Groups jobs by partition and returns a list of jobs for each partition.
* getJobsByNode(): Groups jobs by node list and returns a list of jobs for each node.
* getJobsByExitCode(): Groups jobs by exit code and returns a list of jobs for each exit code.
* getTotalExecutionTimePerPartition(): Groups jobs by partition and sums the total execution time for each partition.

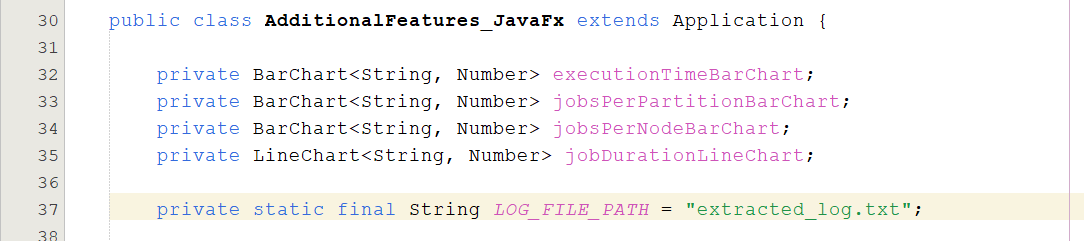


* toString(): Provides a string representation of the Job object, useful for debugging and logging.

s

**Code Snippet & Explanation of AddtionalFeatures\_JavaFX.java**

****

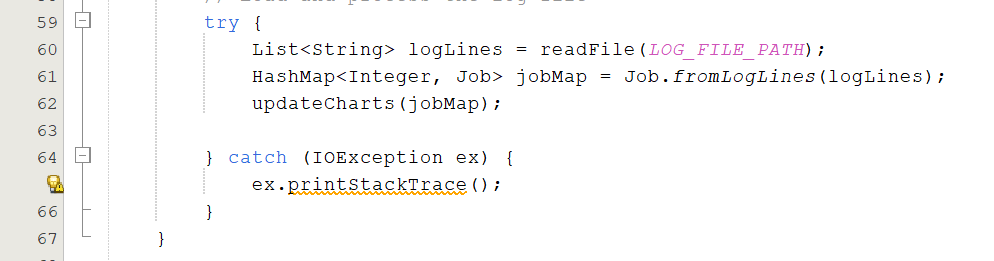
****

Declares four charts (three BarChart and one LineChart) for visualization. The LOG\_FILE\_PATH is a constant storing the path to the log file.

A screen shot of a computer

Description automatically generated

* Start Method: This is the entry point for JavaFX applications. It sets the title of the main window (Stage).
* Layout: Uses a BorderPane as the main layout. Adds a VBox (vertical box) with padding to the top and an HBox (horizontal box) with padding to the center of the BorderPane.
* Setup Charts: Calls setupCharts() to initialize the charts and adds them to chartsBox.
* Scene and Stage: Creates a Scene with the BorderPane as the root node, sets the scene on the primary stage, and shows the stage.



* Read Log File: Reads the log file using readFile(LOG\_FILE\_PATH) and processes it into a jobMap using Job.fromLogLines(logLines).
* Update Charts: Calls updateCharts(jobMap) to update the charts with the job data.
* Exception Handling: Prints the stack trace if an IOException occurs.

A computer screen shot of a computer code

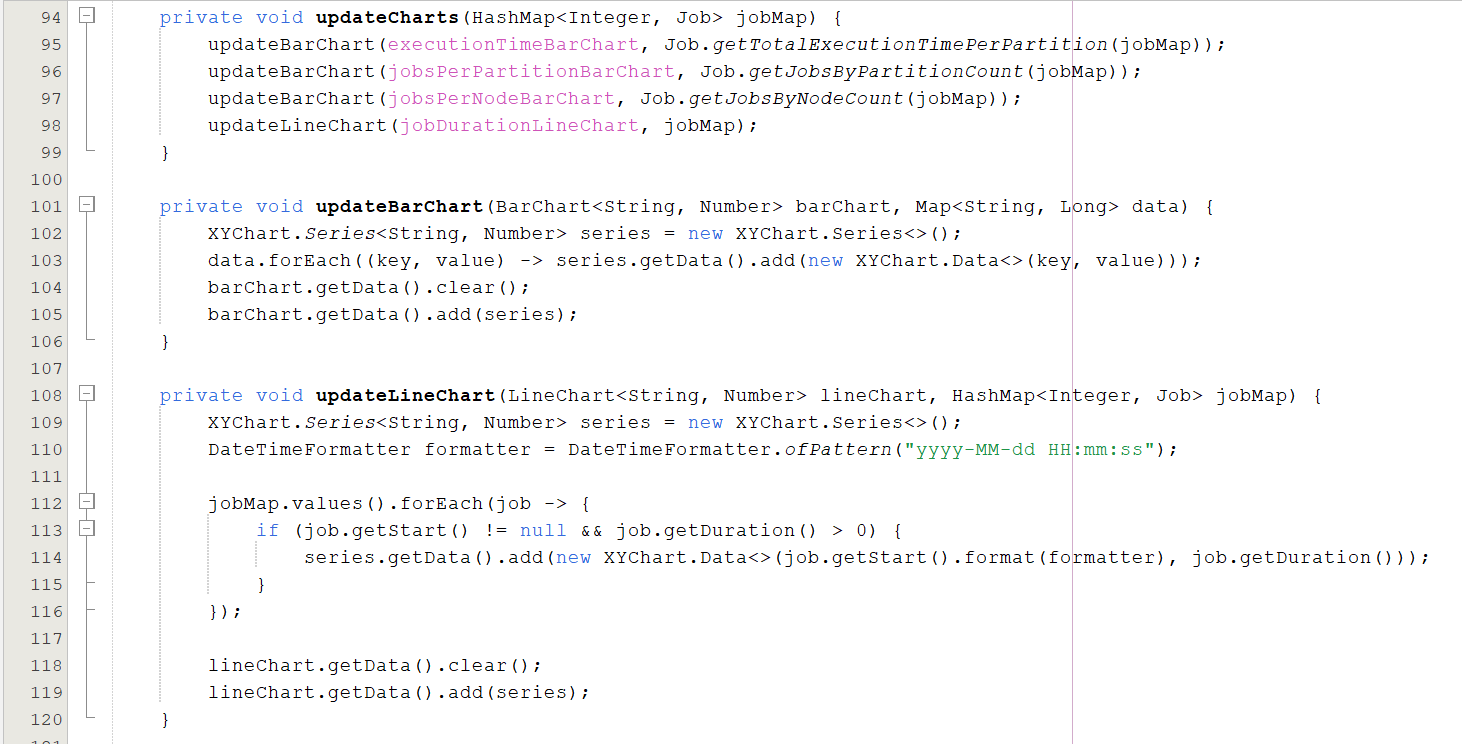
Description automatically generated

* Setup Charts: Initializes the charts by calling createBarChart for each BarChart and manually creating the LineChart with appropriate axes.

A screenshot of a computer code

Description automatically generated

* Create Bar Chart: Helper method to create and return a BarChart with specified axis labels and title.



* Calls methods to update each chart with data derived from jobMap.
* Clears current data and adds new data to a BarChart. It creates a new XYChart.Series and populates it with the provided data.
* Clears current data and adds new data to a LineChart. It creates a new XYChart.Series and populates it with job start times and durations formatted appropriately.



* Reads a file line-by-line and returns a list of lines. Uses a BufferedReader for efficient reading and a try-with-resources block to ensure the reader is closed properly

# **GRAPHS AND CHARTS**

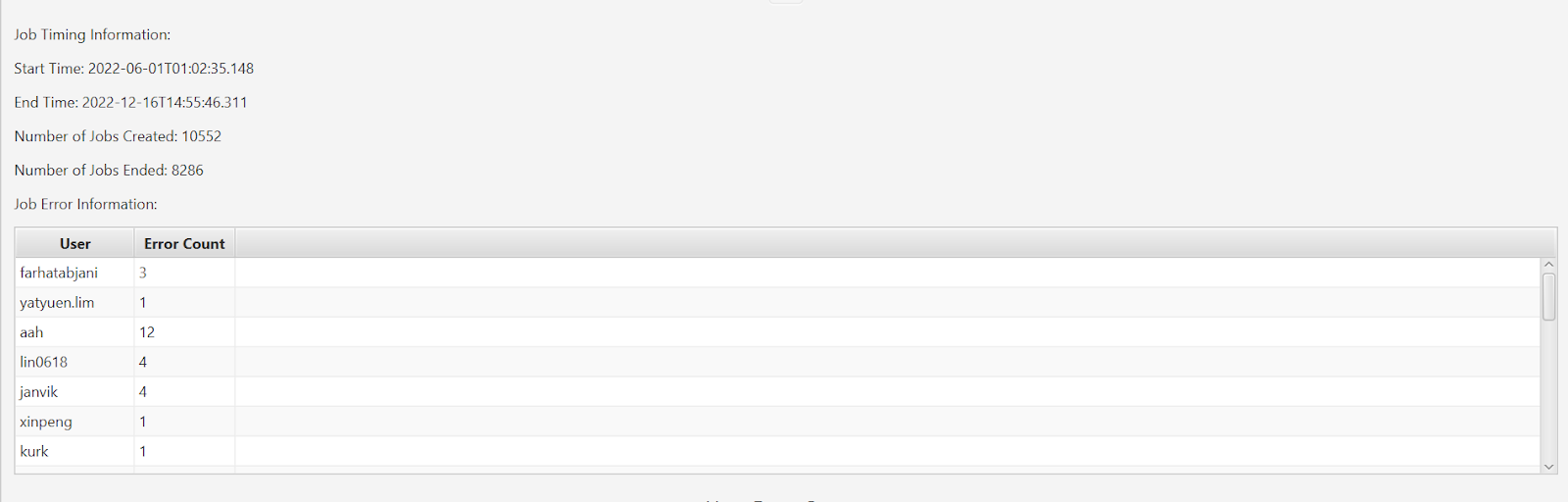
**CHART OF KEY METRICES**

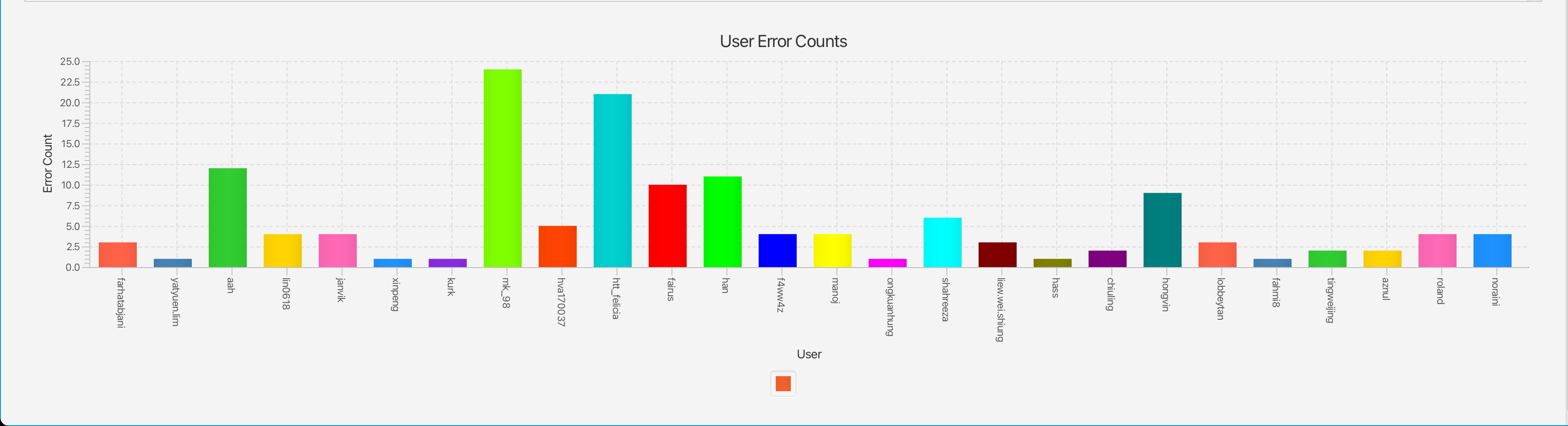
* Number of jobs by partitions i.e. EPYC, Opteron and GPU.

A screenshot of a graph

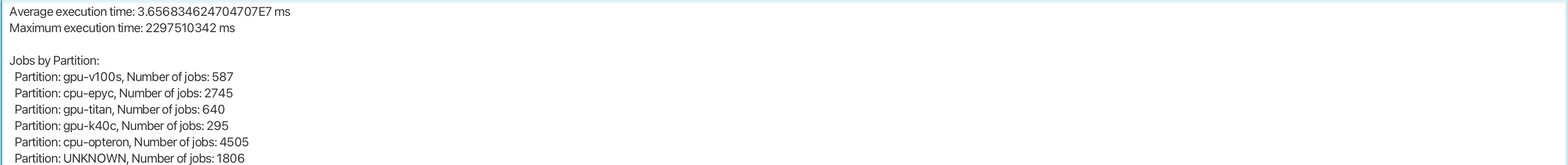
Description automatically generated

* Number of jobs created/ended within a given time range.
* Number of jobs causing error and the corresponding user.

****



* Average execution time of the jobs submitted to UMHPC.



GRAPHICAL VIEW OF ADDITIONAL FEATURE

A screenshot of a graph

Description automatically generated