

Software Engineering Department  
ORT Braude College

Capstone Project Phase A – 61998

Analytics software for intel’s CPU’s test results

24-2-D-34

Supervisors:

Prof. Miri Weiss-Cohen

Alexei Raskin (on Intel’s behalf)

Submitted by: Roi Shamayev - Roi.shamayev@e.braude.ac.il

Table of Contents

[Abstract 3](#_Toc177750130)

[1.Introduction 3](#_Toc177750131)

[2.Background and Related Work 4](#_Toc177750132)

[2.1 Correlation between hardware failures and software anomalies 4](#_Toc177750133)

[2.2 Dashboards in Semiconductor environments 5](#_Toc177750134)

[2.3 Log Analysis 5](#_Toc177750135)

[2.4 interactive analytical dashboards 5](#_Toc177750136)

[2.5 Visualize Text-Based Event Logs 5](#_Toc177750137)

[3.Expected Achievements 6](#_Toc177750138)

[3.1 Significant reduction in response time 6](#_Toc177750139)

[3.2. Near-instant recovery 6](#_Toc177750140)

[3.3 Improved planning efficiency 6](#_Toc177750141)

[3.4.Allowing different data consumers 6](#_Toc177750142)

[3.5 Reduction in costs 6](#_Toc177750143)

[4.Engineering Process 6](#_Toc177750144)

[4.1 Process 6](#_Toc177750145)

[4.1.1 Requirements 6](#_Toc177750146)

[4.1.2 Process research 7](#_Toc177750147)

[4.1.3 Tech stack 8](#_Toc177750148)

[4.1.4 System Architecture 9](#_Toc177750149)

[4.1.5 Worker service 9](#_Toc177750150)

[5.Diagrams 11](#_Toc177750151)

[5.1 Class diagram 11](#_Toc177750152)

[5.2 State diagram 12](#_Toc177750153)

[5.3 Sequence diagram 13](#_Toc177750154)

[6.Definitions 13](#_Toc177750155)

[7. Prototypes 14](#_Toc177750156)

[7.1 Early prototype 14](#_Toc177750157)

[7.2 Web app prototype 15](#_Toc177750158)

[8. Evaluation methods 16](#_Toc177750159)

[9.Limitation and Scope 17](#_Toc177750160)

[10.Early feedback 17](#_Toc177750161)

[References 18](#_Toc177750162)

# Abstract

In semiconductor manufacturing, it is crucial to ensure the reliability of the processors. Intel, a leading chip producer, employs stress-testing to validate processor performance and durability. However, current testing methodologies in some laboratories face challenges in real-time monitoring, causing delays in responding to failures and planning resources. Additionally, substantial time is lost to tester hardware issues, requiring manual, error-prone, and time-consuming data extraction for key metrics. To address these inefficiencies, This paper presents a full-stack web application with real-time log analysis and dynamic visualization of testing data. This solution leverages technologies for data processing, web development, and databases. At its core, the application extracts critical metrics from vast amounts of test data in real time. An intuitive user interface, enables engineers to monitor test progress, identify anomalies, and make data-driven decisions promptly. Initial results showed a significant reduction in the response time to test failures, near-instant recovery after tester malfunction, and improvement in planning efficiency. The proposed solution can accelerate product development cycles, and contribute to high-quality processor production.

**Keywords:**

Full Stack Web Application, Quality Assurance, Processor Testing, Log Analysis, Data Visualization

# 1.Introduction

In the evolving sector of processor manufacturing, ensuring processor quality and reliability is a key goal. Since the early days of processor development, stress testing was employed as a quality assurance measure. Intel, at the forefront of semiconductor manufacturing, has used rigorous testing methodologies. One such method is Burn-In (BI), a process where devices are subjected to accelerated stress conditions, such as high temperatures and voltages, to identify early-life failures and ensure reliability before they are delivered to customers (Suhir, 2019). Today, stress testing and other quality assurance measures are more crucial than ever, given the increasing complexity of modern processors.

At Intel's Haifa Development Center, dedicated testing labs conduct Burn-In tests on processors. As mentioned,Burn-In include long exposure to high temperatures and intensive workloads. However, the current testing process faces several significant challenges that hinder efficiency and accuracy:

1. **Time Loss Due to Tester Malfunctions:** productive time is lost when testers malfunction or human errors occur. When large volume of processors are running ,Determining the total runtime and key metrics of all of them from vast amount of log files that spread over multiple days is a process that is both labor-intensive and prone to inaccuracies.
2. **Lack of Real-Time Monitoring:** Engineers have to wait until the completion of lengthy stress testing in order to receive automated summaries of the results. While those summaries are serving their purpose. They come too late and in the event of tester hardware issues they are not generated at all.
3. **Inefficient Planning:** Without a clear interface and real-time monitoring, the planning team, responsible for coordinating lab activities, faces major issues in coordinating with stakeholders.

To address these challenges, this project proposes a comprehensive solution with two main components:

1. **Real-Time Analysis and Data Extraction**: A worker service written in C#, Running on an external PC will extract key test results and runtimes from vast amount of log files at regular intervals. Then, it will parse the results and upload them to a SQL database, this approach will ensure up to date result and will allow for quick retrieval. Additionally , an API endpoint will be established to allow request for data to be served.
2. **Web Application for Visualization**: A web application built with the Angular framework will be deployed to display the fetched data through an intuitive dashboard with real-time data. This interface will provide engineers with a convenient and accurate way to monitor test progress. this approach will allow to view results, including any anomalies, facilitating faster recovery from failures and more efficient pre-planning.

All of the solutions mentioned will be implemented using a dashboard that will display real-time results of different CPU’s , including valuable key data points such as serial id of the CPU , status of the test , test result and other key metrics that will be discussed later in the paper.

The dashboard will be highly valuable for engineers but is also served to lab workers to ensure the hardware in adequate condition. The proposed solution offers quick recovery from errors, a user-friendly interface for real-time test monitoring, and a clear and efficient planning process. This approach will ensure all stockholders are informed and are able to make data driven actions in a timely manner.

# 2.Background and Related Work

This work focuses on labs at Intel Development Center in Haifa, Israel. However, Its planned to distribute the implementation in other sites around the world ,where much  higher volumes of CPU’s are tested. Those sites will highly benefit from this automated logging analysis and dashboard approach as they face the same challenges at a production scale.

This section will cover existing efforts done in similar fields and will lay the foundation and gaps the proposed solution aim to improve upon.

## 2.1 Correlation between hardware failures and software anomalies

Traditionally, stress testing has focused primarily on hardware performance, an area that is well understood. However, the correlation between software failures triggered by hardware issues is often overlooked or tested in less extensive ways, leading to degraded performance and unexpected failures . While Wu’s work focuses on the military field, its implications are transferable to the semiconductor industry, where hardware faults can cause software anomalies, especially during development.(Wu, 2012)

The dashboard solution described aims to address this gap by including both hardware metrics and testing results, providing stakeholders a full view and allow them to make informed decisions ,and identify anomalies during the testing phase hence preventing failures from reaching market or production environments early on.

## 2.2 Dashboards in Semiconductor environments

Dashboards in Semiconductor environments are not a novelty,Many approaches have been taken to ensure visibility and improve decision-making. current efforts in creating a dashboard have offered tailored information for different data consumers from maintenance technicians up to the management level (Collier, 2012). However, most of those dashboards focus on manufacturing processes and equipment health monitoring. The dashboard proposed in this paper follows a similar logic but focuses on the testing phase, which is largely overlooked.

## 2.3 Log Analysis

Analysing vast amounts of log data to extract actionable insights has become increasingly important in various industries. Logs are often the primary source of information about a system's behaviour challenge in stress testing environments is the sheer volume of log data that needs to be processed and analysed to gain meaningful insights. As Shilin He (2021) explores in his survey of automated log analysis, logs are invaluable for ensuring system reliability, but their growing scale makes manual analysis impractical. He’s work outlines advancements in log parsing and event detection that aim to automate and streamline log analysis, making it more feasible to handle large-scale datasets in real time .The proposed solution backend will utilize log analysis techniques to extract key metrics in real time, ensuring up tp date data for the dashboard.(He et al., 2021)

## 2.4 interactive analytical dashboards

Further supporting the need for interactive, real-time dashboards, (Nadj et al., 2020) investigate the cognitive effects of dashboards in decision-making processes. Their findings suggest that while interactive analytical dashboards improve task performance, there are potential downsides related to reduced situational awareness. This is a critical consideration for the design of dashboards in high-stakes environments, such as semiconductor manufacturing, where engineers rely on live data to make decisions during stress testing. Incorporating these findings into the design of real-time monitoring systems can help mitigate potential cognitive overload and ensure that engineers remain actively engaged in the decision-making process.

## 2.5 Visualize Text-Based Event Logs

Visualizing text-based event logs effectively is also crucial. transforming large, text-based logs into a manageable database format can enhance problem identification and troubleshooting efficiency. This methodology supports the core functionality of the proposed worker service in parsing and visualizing critical metrics from extensive log files, promoting timely and accurate data analysis.(Lee & Jeong, 2021)

# 3.Expected Achievements

The expected achievements for the proposed dashboard is to address the inefficiencies outlined previously. Specifically, the key expected outcomes include:

## 3.1 Significant reduction in response time

Significant reduction in response time to test failures by providing real-time visibility into test status and metrics

## 3.2. Near-instant recovery

Near-instant recovery from tester hardware issues by automating data extraction, allowing for engineers to quickly continue testing with most recent data

## 3.3 Improved planning efficiency

Improved planning efficiency ,by providing accurate estimates of test progress and key metrics to all relevant stakeholders, enabling better resource allocation.

## 3.4.Allowing different data consumers

Allowing different data consumers (engineers, managers, etc.) to access relevant information through a tailored UI

## 3.5 Reduction in costs

Reduction in costs, while not a primary objective, Quicker identification of failures will minimize downtime, and more accurate data collection will reduce the need for re-testing, lowering overall testing costs.

The approach and implementation details of the solution are described in length in the following sections.

# 4.Engineering Process

This section will outline the methodology and technical details of the proposed web application, including technical details and specific approaches ,tech stack and design considerations.

## 4.1 Process

With the problem definition and expected outcomes established, the engineering process will follow agile principles with iterative design and development, incorporating feedback from key stakeholders.

### 4.1.1 Requirements

The requirements were gathered from key stakeholders ,including managers ,engineers, lab technicians.

The main functional requirements of the system are:

* The system must extract key test results and runtimes from log files in real-time.
* The system must parse extracted results.
* The system must upload parsed data to a SQL database.
* The system must provide an API endpoint to serve data requests.
* The system must display fetched data through an intuitive dashboard.
* The system must enable real-time monitoring of test progress.
* The system must visualize key metrics including CPU serial ID, test status, and test result.
* The system must support different user roles (engineers, lab workers, managers).
* The system must identify and display anomalies in test results.
* The system must generate automated summaries of test results.
* The system must allow users to view historical test data.

The main non-functional requirements are:

* Performance: The system should process and display up to date data, which dates less than 15 min back.
* Scalability: The system should be able to handle increasing volumes of log data without significant degradation in performance.
* Reliability: The system should maintain consistent operation, even during tester hardware issues.
* Usability: The user interface should be intuitive and easy to use for different stakeholders.
* Security: The system should implement proper access controls for different user roles.
* Maintainability: The system should be designed for easy updates and maintenance.
* Compatibility: The system should be compatible with different Intel testing sites globally.
* Efficiency: The system should contribute to reducing response time to test failures.
* Accuracy: The system should ensure precise extraction and representation of test data.
* Availability: The system should maintain high uptime to ensure continuous monitoring capability.

### 4.1.2 Process research

In order to better understand the implementation process and meet the requirements, the following internal research was conducted:

* Researching and understanding the different equipment and its supporting software, this include the log formats, hardware specifications, relevant paths of where logs are stored.
* Analysing large amount of folder and files that might contain relevant data. This required scheduling meetings with known subject matter experts and going through example logs to understand the semantics and structure.
* Listing the relevant keywords that will be used to parse the logs and extract the necessary metrics such as CPU serial, test status, runtime etc.
* Assessing how reliable the logs are and identifying edge cases.
* Researching how to combine partial data points into a complete status of the overall test.
* Learn existing infrastructure and evaluating compatibility with the new solution.

\*Note this section is general and didn’t go into specifics due to confidentiality constrains.

### 4.1.3 Tech stack

When choosing the tech-stack for any project , there are always trade-offs to consider in terms of performance, scalability, cost, maintainability, developer familiarity, and ecosystem maturity. Often it is beneficial to use technologies the rest of the team already has experience with, and already built great software on. With that in mind, The solution will be developed as a full-stack web application, leveraging the following technologies:

-Backend: The backend will be written in C#, specifically using the Worker service in .NET . The worker service serves as a useful framework in windows environments to continuously parse logs, extract metrics, and upload to an SQL DB.

-Databases: The metrics extracted from logs will be stored in Maria DB, a relational database, providing fast querying and integration with the web application.

-API: A RESTful API will be built with C#, allowing the front end to fetch data from the backend in a decoupled manner.

-Front-end: As discussed , the UI will be built using the Angular framework ,which comes with a large set of components out of the box, and integrates well with the underlying backend using data-binding and HTTP requests. The visualization will incorporate the charting library ngx-charts.

### 4.1.4 System Architecture

The system architecture follows a traditional 3 layers web application architecture with the addition of a worker service responsible for parsing and storing data in The database. (See figure 1)

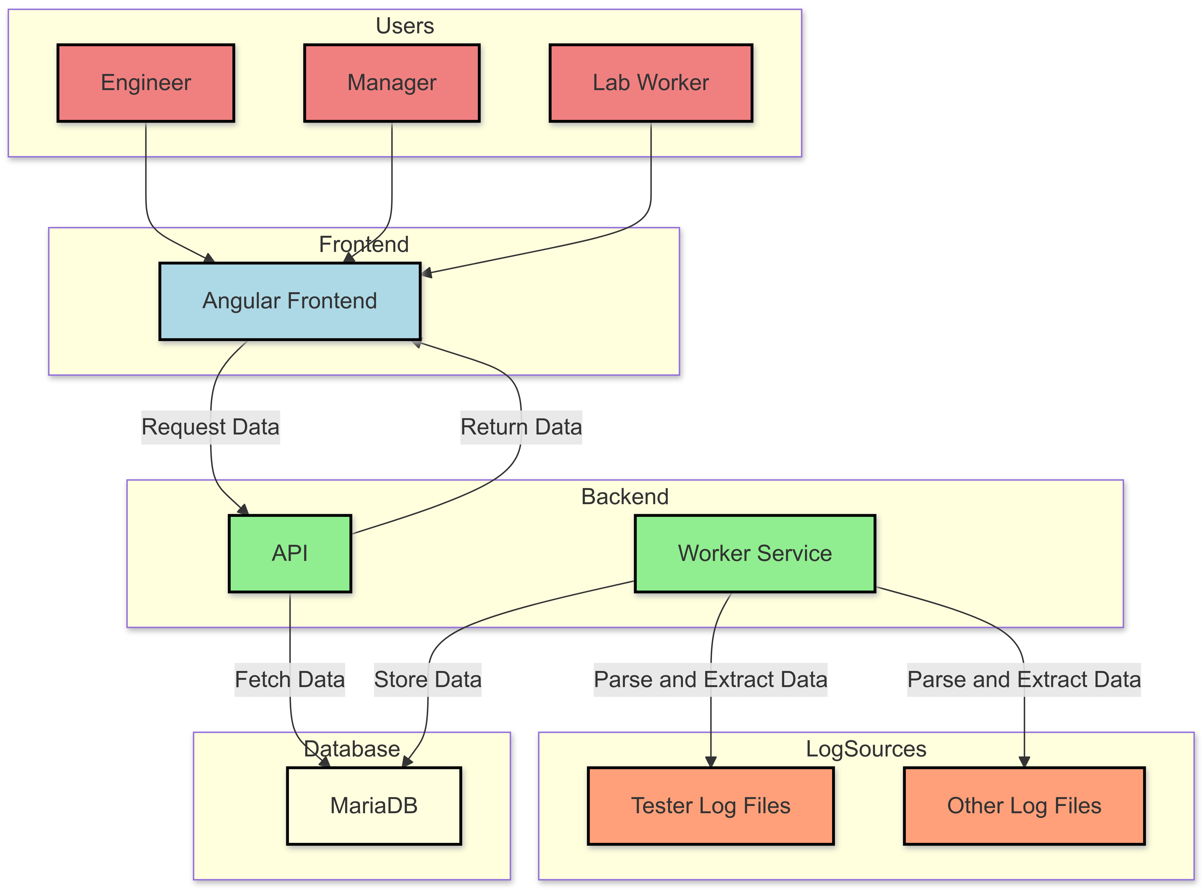


Figure 1 -System Architecture

### 4.1.5 Worker service

The high level responsibility of the worker service in the context of the project is already well defined. However , The worker service serves as the root data source for the entire application , and that comes with its own set of design considerations. The following Flowchart steps performed by the worker service(see figure 2) A diagram of a process

Description automatically generated

Figure 2 -Worker service

**Breakdown:**

1. Initialize Worker Service: When the worker service starts, it initializes the log copying process, including Paths, DB credentials , User credentials etc.
2. Copy Logs to Local Folder: Copies relevant logs to a local folder to prevent overwriting the original machine logs.
   * If the copy fails, the system waits 5 minutes before retrying.
3. Combine Logs into a Single File: Once the logs are successfully copied, the service combines them into one file, ensuring the data is in chronological order.
4. Filter Irrelevant Data: The service removes any unnecessary or irrelevant data to reduce the log file size.
5. Start Parsing Process: Begins the detailed parsing of relevant data.
6. Test Program Settings Parsing: Parses the main test program (TP) file, looking for referenced files.
   * If files are missing, default values are attached, and a flag is raised to notify the user and The service will retry after 5 minutes.
7. Parse Live Test Data Logs: Extracts key metrics from the reduced log file using regex and other extraction methods.
8. Mapping to Class Properties: Maps the extracted metrics, like Serial ID, test result, and hardware metrics, to a class with these properties.
9. Data Interpolation and Anomalies: Applies data interpolation techniques to detect any anomalies in the parsed data and introduces additional metrics like the total pass/fail count and estimated time of completion (ETA).
10. Database Interaction:
    * Checks if the data for the specific CPUs already exists in the database.
    * If it exists, the service updates the data; otherwise, it inserts new records.
11. End: Once the data is successfully added to the database, the process logs the success and loops back to continue monitoring for new logs.

# 5.Diagrams

## 5.1 Class diagram

The class diagram describes the high-level components of the system and their interactions.(See figure 3)

While the methods of some components are encapsulated into a single main metho, the general flow is understood.

A diagram of a software application

Description automatically generated

Figure 3 – Class diagram

## 5.2 State diagram

The state diagram shows the main states the system can be in and the transitions between them. (See figure 4) when “System failure” occurs the application will retry after 5 min

A diagram of a system failure

Description automatically generated

Figure 4- State diagram

## 5.3 Sequence diagram

The sequence diagram illustrates the flow of data and control among the different components of the system.(See figure 5) A screenshot of a diagram

Description automatically generated

Figure 5 -Sequence diagram

# 6.Definitions

To better understand the following sections definition of key terms is needed.

* **Serial ID** - A unique identifier for a specific CPU
* **Lot** - a unique identifier for a group of CPU’s
* **Location code(L.C.)** - Identifier used In combination with Lot to define which test program will run on the CPU’s
* **BIN** - A number, Refers to binning ,process of sorting and categorizing CPUs based on their performance . Usually Bin 1 or 100 indicate a good CPU , and other Bins suggest a problem in a specific submodule assigned to that bin.
* **Cell** - The identifier of the Tester equipment
* **Socket** - As different cells have multiple sockets, each socket is identified by its number
* **Retest** - Defines the number of times a test was performed on a processor
* **Stress Time** - The runtime of the CPU under stress test conditions.

# 7. Prototypes

In this section, We discuss the user interface prototypes and their key contributions to the development of this project.

## 7.1 Early prototype

In modern software development, early prototyping can help validate requirements, gather feedback, and identify potential issues before going into full-scale implementation(Gordon & Bieman, 1993)(Petrucci & Luqi, 2002).

The validity of the different results presented to the engineers are paramount, as they will be the foundation of critical decisions. With that in mind, as soon as the Worker service and its interactions with the Database was ready a mini app was developed for data validation purely and distributed only to lab technicians (See figure 6). The app is a Windows Presentation Foundation (WPF), a framework that relies on XAML defined views and C# for the logic. While it does steer away from the project goal it was created with minimal development and time efforts, Thanks to Visual Studio offering “Click and Drop” designer for simple and quick UI prototypes.

This approach allowed for a quick distribution of the app , lab technicians were giving feedback about protentional errors and feature requests over real data, allowing for iterative improvements before the full web solution is developed.

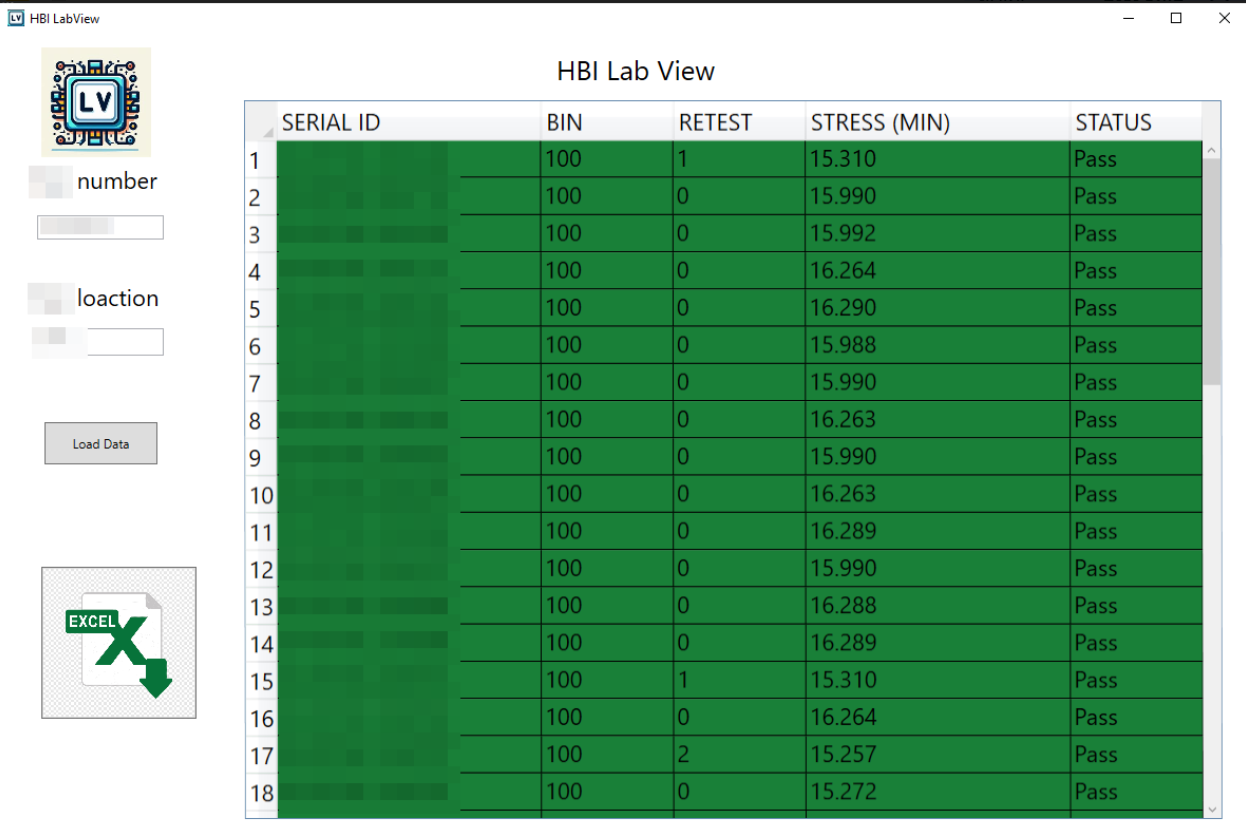


Figure 6- Early prototype

## 7.2 Web app prototype

While the early prototype provided a good starting point for data validation, the final proposed solution is a full-stack web application which provides a more comprehensive set of features and a better user experience.

For testing and demonstration purposes a web app prototype was developed using Angular, with 2 components:

1. **Card component** - This component hold the currently running lots leveraging built in components(See figure 7).
2. **Result component** - This component display the processors testing results and uses the ngx-charts library for visualization (See figure 8).

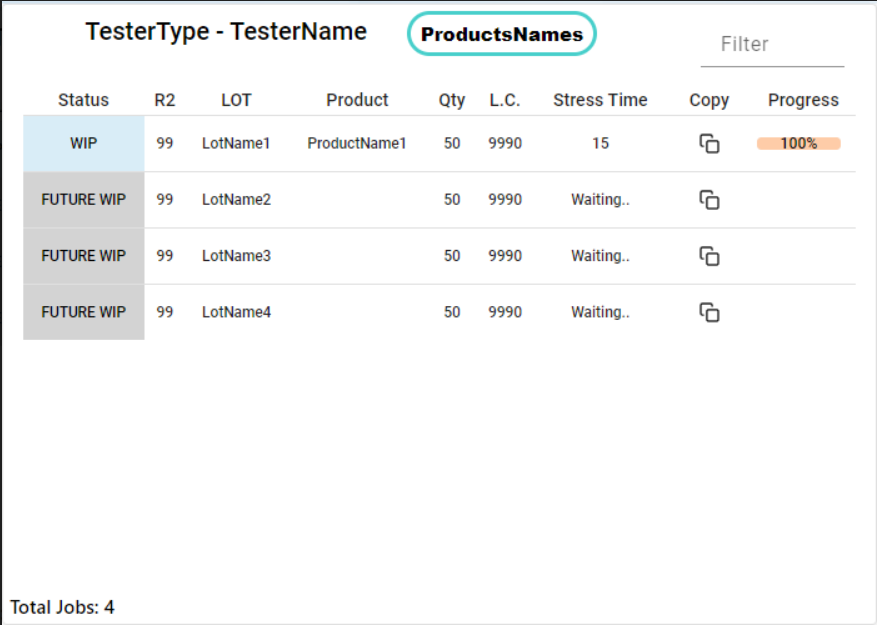


Figure 7 -Card component

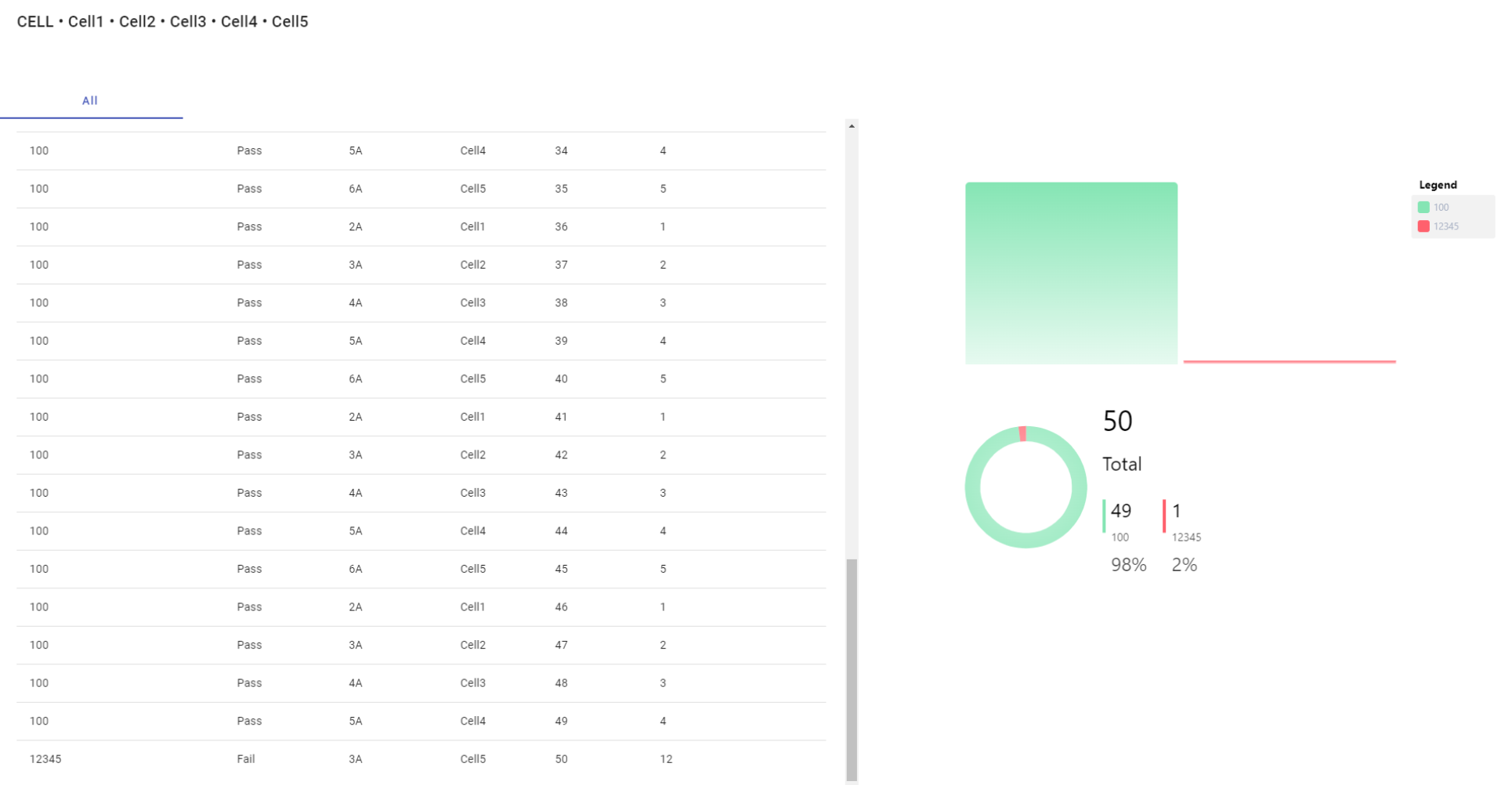


Figure 8 – Result Component

Note: In figures 6,7,8 the real data was either blurred or replaced with fake data To comply with confidentially constraints.

# 8. Evaluation methods

The following table outlines the key metrics and methods used to evaluate the success of the proposed dashboard(see table 1):

| **Metric** | **Description** | **Evaluation Method** | **Target** |
| --- | --- | --- | --- |
| Response Time to Test Failures | Time taken to identify and respond to test failures | Survey Engineers | 50% reduction in response time |
| Recovery Time from Tester Hardware Issues | Time required to resume testing after hardware malfunctions | Measure time from issue detection to test resumption | 75% reduction in recovery time |
| Planning Efficiency | Improvement in resource allocation and scheduling | Survey stakeholders | 30% improvement in planning efficiency |
| User Satisfaction | Satisfaction level of different user groups (engineers, managers, lab technicians) | Conduct user surveys and interviews | 80% user satisfaction rate |
| Data Accuracy | Precision of extracted and displayed test data | Compare dashboard data with manual log analysis | 99.9% accuracy rate |
| System Uptime | Availability of the dashboard and backend services | Monitor system uptime and track downtimes | 99.5% uptime |
| Data Refresh Rate | Frequency of data updates on the dashboard | Measure time between log updates and dashboard refreshes | Updates every 15 minutes or less |
| Cost Savings | Reduction in overall testing costs due to improved efficiency | Calculate cost savings from reduced downtime and retesting | Optional- any reduction |
| Anomaly Detection Rate | Accuracy in identifying test anomalies | Compare system-detected anomalies with manually identified ones | 95% detection rate |

Table 1 - Evaluation methods

# 9.Limitation and Scope

While the proposed project offer great improvements in real time monitoring in testing processes and environments, Its very important to understand the limitation and scope of the system :

1. **Complementary tool** : The dashboard provides comprehensive overview of the testing data , But it doesn’t replace the engineers responsibility to perform deep analysis to identify root cause of failures and anomalies. It serves as complementary tool to enable faster responses to failures and quicker recovery from hardware issues.
2. **Limited level details:** The system presents summarized data for quick insights and might not display all fine-grained details that exists in original log files. For edge cases, engineers might still need to refer to raw log files or use other utilities.
3. **Network strain** : While the dashboard strives to presents data as close to real time as possible , its not realistic to process and update data too often as frequent and large files transfers pose a considerable strain on network drives. A meeting was scheduled with IT and other stakeholders to identify the balance - which was agreed to 10-15 minute intervals.

# 10.Early feedback

While this project is still in the development phase, some initial feedback was received and is worth mentioning. This section will highlight real cases where the dashboard was able to provide critical insights to engineers.

**Overstressing CPU’s** - As mentioned before the stress time of each CPU is determined by the engineer. Lab technicians using the prototype to monitor testing reported higher than expected stress time. The test was scheduled for 60 minutes stress and ended up with 140 minutes stress, That’s 233% of the intended stress time. This information was shared with engineers and validated to be true. A fix to the test program was released.

**Recovery from human error** - In one case , An employee accidentally restarted the main server of the testers. This caused failures and confusion as it was not clear how to continue. A key metric in such cases is the stress time remaining for each CPU , a quick summary report was generated detailing the stress time remaining and we were able to resume testing in less than an hour. To compare , A similar accident happened before without the existence such system - and caused a delay of days.

# References

1. Collier, S A. (2012, December 1). Semiconductor manufacturing dashboard. https://repositories.lib.utexas.edu/items/9727eb34-e14c-4e23-a975-c215c03b3bcf
2. Gordon, V S., & Bieman, J M. (1993, June 1). Reported effects of rapid prototyping on industrial software quality. Springer Science+Business Media, 2(2), 93-108. https://doi.org/10.1007/bf00590438
3. He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M R. (2021, July 13). A Survey on Automated Log Analysis for Reliability Engineering. Association for Computing Machinery, 54(6), 1-37. https://doi.org/10.1145/3460345
4. Lee, G., & Jeong, J. (2021, June 26). An Efficient Analytical Approach to Visualize Text-Based Event Logs for Semiconductor Equipment. Multidisciplinary Digital Publishing Institute, 11(13), 5944-5944. https://doi.org/10.3390/app11135944
5. Nadj, M., Maedche, A., & Schieder, C. (2020, August 1). The effect of interactive analytical dashboard features on situation awareness and task performance. Elsevier BV, 135, 113322-113322. https://doi.org/10.1016/j.dss.2020.113322
6. Petrucci, L., & Luqi, L. (2002, September 1). An introduction to rapid system prototyping. IEEE Computer Society, 28(9), 817-821. https://doi.org/10.1109/tse.2002.1033222
7. Suhir, E. (2019, March 6). To Burn-In, or Not to Burn-In: That’s the Question. Multidisciplinary Digital Publishing Institute, 6(3), 29-29. https://doi.org/10.3390/aerospace6030029
8. Wu, C J. (2012, September 1). Stress testing software to determine fault tolerance for hardware failure and anomalies. https://doi.org/10.1109/autest.2012.6334582