Analysing and Visualising Benchmarks (Basic Modal Logic)

Dharani Shanmugam 201687803

July 5, 2024

1 Statement of ethical compliance

1.1 Data category

Data category: A

No use of data derived from humans or animals. The data was provided by the supervisor.

1.2 Human participant category

Human participant category: 2

Human participants will be used during software evaluation. To get feedback about the website via any of the permitted activities under ethical guidance.

2 Project description

This project aims to develop a web-based application for benchmarking theorem provers in basic modal logic. Benchmarking involves evaluating software systems based on various criteria such as execution time, memory usage, correctness of results, and other performance metrics. For this project, about 3,000 inputs will be used to benchmark around 8 different theorem provers, some with multiple configurations, resulting in approximately 150,000 data points.

The web application will allow users to upload performance data in CSV or log file formats, store these measurements in a database, and perform detailed analysis through SQL queries. Users will be able to compute aggregate statistics, compare different systems, and visualize the results using various chart types like bar charts, scatter plots, and heat maps.

A significant aspect of the project is to identify interesting inputs from the 3,000 available inputs, reducing the dataset to a manageable subset of about 100 inputs that exhibit significant performance variability. This will streamline future benchmarking efforts by focusing on critical scenarios, making the benchmarking process more efficient.

3 Aims and Requirements

3.1 Aims

1. The first aim of this project is to develop a web-based application that supports these activities:

- Look at specific measurements for specific systems, filtered by SQL query.
- Obtain aggregate measurements, for example, average execution time over all benchmarks or some subset of the benchmarks.
- Compare different systems.
- Visualisation of the measurements: bar charts, scatter plots, heat maps of execution time over size/difficulty of the benchmark.

2. The second aim of the project is to find a way to identify best inputs using Principal Component Analysis among the 3,000 available inputs, say, 100 inputs. These should allow to perform benchmarking in a less time consuming way.

3.2 Requirements

- **Data Upload:** The system must allow users to upload measurements in CSV or log file formats.Uploaded data will be stored in a database for efficient querying and analysis.
- **Data Query and Filtering:** Users should be able to filter and query the data using SQL commands to look at specific measurements for specific systems or configurations.
- **Aggregate Measurements:** The application will provide functionalities to calculate aggregate measurements, such as the average execution time over all benchmarks or specific subsets of benchmarks.
- **Comparison Tools:** Users will be able to easily compare the performance of different theorem provers and their configurations. The comparison will include various performance metrics such as execution time, memory usage, and correctness of results.
- **Data Visualization:** The system will offer visualizations like bar charts, scatter plots and heat maps to represent measurements. Inputs that show significant variability in performance across different theorem provers or configurations.
- **Identification of Key Inputs:** Using machine learning method Principal Component Analysis to identify inputs with significant performance differences among theorem provers.

4 Literature review

Benchmarking is a standardized process utilized to evaluate the performance of software systems. It typically involves running a series of tests and measurements on different software configurations to compare performance metrics. While execution time is often a primary criterion, additional factors such as memory usage, correctness of results, and other abstract measures are also considered.

In the context of theorem proving for basic modal logic, benchmarking involves evaluating various theorem provers across numerous input cases to determine their effectiveness. The goal is to identify the best-performing configurations under different scenarios and to gain insights into their strengths and weaknesses.

Benchmarks in the context of modal logic systems refer to a set of standardized problems or formulae used to evaluate the performance of different theorem provers. These benchmarks help in comparing the efficiency, effectiveness, and robustness of various proving systems under controlled conditions.(Nalon et al. 2018)

Provers are automated reasoning tools designed to determine the suitability of logical formulae. In modal logic systems, provers use various algorithms and techniques to handle the complexity and unique characteristics of modal logic.(Nalon et al. 2018)

The evaluation of these provers involves running them on benchmark formulae and comparing their performance metrics, such as runtime and the number of solved instances. For instance, KSP has shown superior performance on the MQBF collection, particularly on formulae with high modal depth, due to its efficient handling of layered normal forms. By comparing these provers across standardized benchmarks, we can identify strengths and weaknesses, guiding the development of more advanced and efficient automated reasoning systems for modal logic.(Nalon et al. 2018)

In the study by Nalon, Hustadt, and Dixon, benchmarking is applied to compare the performance of different theorem provers for basic modal logic. The benchmark involves around 3,000 inputs and multiple theorem prover configurations, resulting in a substantial amount of data points. This process helps in identifying which provers perform best under specific conditions and how they compare to one another across various metrics.(Nalon et al. 2018)

The large volume of data generated from benchmarking presents challenges in terms of analysis and visualization. With around 150,000 data points, finding significant patterns or "interesting" inputs becomes a daunting task. The project aims to develop tools and methodologies to facilitate this analysis, making it easier to draw meaningful conclusions from the benchmarking data.(Nalon et al. 2018)

Visualization plays a crucial role in interpreting benchmarking results. By presenting data through visual means such as bar charts and scatter plots, researchers can more easily identify trends and outliers. This is especially important when dealing with large datasets, where raw numerical data alone can be overwhelming. Effective visualization helps in comparing different systems and understanding their performance characteristics across various benchmarks.(Nalon et al. 2018)

In the context of Linear Temporal Logic (LTL) satisfiability solvers, Schuppan and Darmawan have conducted similar benchmarking studies. They emphasize the importance of systematic evaluation and comparison of different solver configurations. Their work highlights how benchmarking can reveal the strengths and limitations of each solver, providing valuable insights for further development and optimization.(Biere et al. 2009)

Principal Component Analysis (PCA) is a technique used to simplify complex datasets by reducing their dimensions. By transforming data into principal components, researchers can identify key patterns and correlations that might not be immediately apparent. PCA is particularly useful in benchmarking studies where multiple performance metrics need to be analyzed simultaneously. It helps in highlighting the most significant factors influencing performance, thereby aiding in the identification of "interesting" inputs for further analysis.(Abdi & Williams 2010)

The integration of visualization techniques and advanced statistical methods like PCA further enhances the ability to interpret large datasets, making the benchmarking process more efficient and informative.(Abdi & Williams 2010)

5 Development and Implementation Summary

5.1 Development Environment

- **Integrated Development Environment (IDE):** Visual Studio Code offers robust support for web technologies, extensions for additional functionalities, and a user-friendly interface.
- **Version Control:** GitHub for source code management. It provides reliable tools for code collaboration and version tracking.

  https://github.com/dharaaani/COMP702
- **Project Management:** Trello for task tracking and workflow management. Trello is effective for tracking progress, managing tasks, and ensuring that the project stays on schedule.

  https://trello.com/invite/b/CWtd2M6C/ATTI63139e8c52bcb80132b647d7f098bd00C93407A9/comp702

5.2 Languages and Tools

- **Front-end Languages: HTML5**: For structuring the web application.
- **CSS3**: For styling the application, ensuring a responsive website.
- **JavaScript**: For interaction between the web pages.
- **React.js**: A JavaScript library for user interfaces, used for its component-based architecture and efficient rendering. Enables full-stack development using a single language, making it easier to manage and maintain the code base.
- **Back-end Languages: Node.js**: A JavaScript runtime built on Chrome's engine, chosen for its performance and scalability.
- **Database: SQL**: A relational database system known for its reliability. Robust support for handling large datasets and performing queries efficiently.
- **Visualization: Chart.js**: A flexible JavaScript library for creating various types of charts and graphs to visualize benchmarking data. Provides an easy way to integrate dynamic and interactive charts into the web application.
- **Machine Learning and Data Analysis: Python**: For implementing Principal Component Analysis (PCA) and other data analysis tasks. Well-suited for data analysis and machine learning tasks, with extensive libraries.
- **Pandas**: For data manipulation and analysis.
- **Scikit-learn**: For implementing PCA and other machine learning algorithms.

5.3 Agile Methodology

- **Sprints:** Divide the project into sprints, each focusing on a specific set of tasks.
- **Sprint Reviews:** At the end of each sprint, review completed tasks and plan for the next sprint.
- **Trello Boards:** To track progress and set deadlines.
- **Milestones:** Define key milestones to ensure the project stays on track and meets deadlines.

5.4 Implementation Summary

Phase 1: Planning and Design

- Requirement Analysis: Finalize the list of requirements
- Architecture Design: Create the system architecture, including database schema and front-end components.
- Wireframing: Design UI wireframes for the application screens.

Phase 2: Set Up Development Environment

- Install Tools: Set up the development environment with the necessary software and tools.
- Initialize Repository: Create GitHub repositories for source code and documentation.

Phase 3: Front-End Development

- Build Core UI Components: Develop the main interface components using React.js.
- Implement Data Visualization: Integrate Chart.js to create interactive charts and graphs.

Phase 4: Back-End Development

- Set Up Server: Configure the Node.js server.
- Integrate Database: Connect the server to the database and implement data models.

Phase 5: Data Analysis and Machine Learning

- Data Preprocessing: Implement data preprocessing scripts in Python using Pandas.
- Principal Component Analysis: Develop PCA algorithms using Scikit-learn to identify interesting inputs.

Phase 6: Testing and Evaluation

- Unit Testing: Write unit tests for front-end components.
- User Evaluation: Collect feedback from users and make necessary improvements.

Phase 7: Deployment and Documentation

- Deploy Application: Host the application on a web server.
- Documentation: Create comprehensive documentation.

6 Data Sources

The primary data source will be the CSV or log files uploaded by users, containing performance measurements of theorem provers. The dataset has provided by the supervisor .All data is used with permission, and no personal or sensitive information is involved.

7 Testing and Evaluation

7.1 Testing

- **Unit Testing:** Individual components will be tested to ensure they work as expected.

7.2 Evaluation

- **User Feedback:** Users will use the application and provide feedback.
- **Requirement Satisfaction:** Ensure all essential and desirable requirements are met.

8 Project ethics and human participants

This project involve human participants during software evaluation. To get feedback about the website via any of the permitted activities. This project does not involve sensitive data. All activities adhere to ethical guidelines.

9 BCS Project Criteria

Application of Practical and Analytical Skills: The project involves developing a web-based application for benchmarking theorem provers. This requires practical skills in software development, database management, and data analysis, all of which are core competencies gained during the modules like Web Programming, Database and Information Systems, Machine Learning, Efficient Algorithms. The application will utilize these skills to handle large datasets, implement complex queries, and generate meaningful visualizations.

Innovation and/or Creativity: Website involves creative problem-solving to address challenges such as handling diverse inputs, optimizing performance comparisons, and presenting data in intuitive formats. The choice of technology stack and the approach to data visualization will showcase innovative thinking.

Synthesis of Information, Ideas, and Practices: The project synthesizes information by integrating theories of benchmarking with practical implementation in software development. It combines best practices in database design, web application architecture, and user interface design to deliver a quality solution. The evaluation component involves assessing how well the application meets benchmarking criteria and user needs.

Ability to Self-Manage a Significant Piece of Work: Developing a comprehensive web application involves managing a significant project from inception to completion. This includes planning, design, implementation, testing, and documentation. Project management skills are essential to ensure milestones are met, issues are resolved promptly, and resources are allocated effectively.

Critical Self-Evaluation of the Process: Throughout the project, critical self-evaluation is essential. This involves reflecting on design decisions, development methodologies, and outcomes achieved against initial goals. Regular reviews, feedback loops, and iterative improvements are part of the development process to ensure the final product meets quality standards and user expectations.

10 UI/UX Mockup

Screen 1: A homepage with upload CSV option to upload files. SQL Query text box section to filter the data according to the query. Aggregate Measurements section to view the output results. Data Visualization section to view the charts.

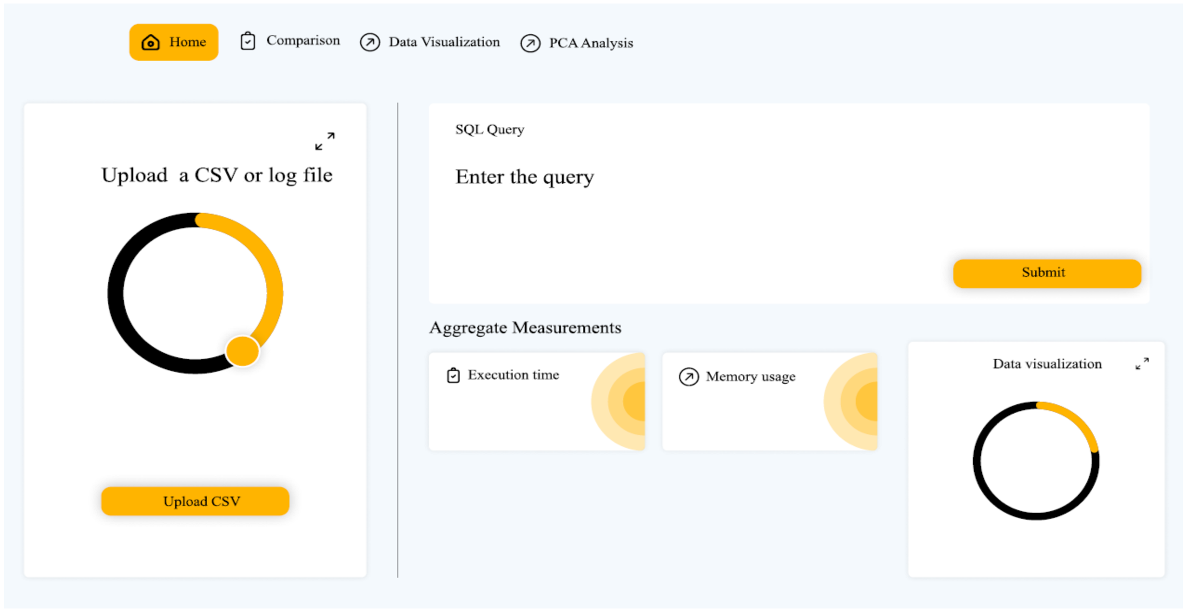

Figure 1: Wireframes

Screen 2: A comparison tab to upload CSV files and compare between two different provers. Aggregate Measurements section to view the aggregate and comparison results. Comparison chart section to view the comparison between two different provers.

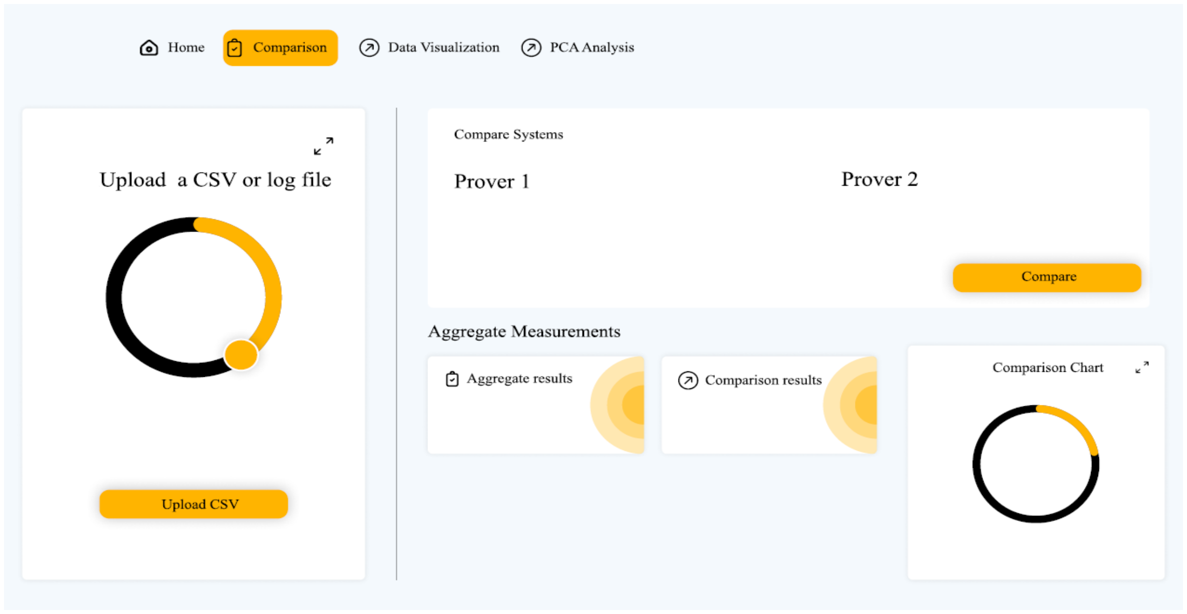

Figure 2: Wireframes

Screen 3: A data visualization tab to visualize data in different formats like bar chart, scatter plots and heat maps. The data visualization has three options like visualization drop down menu to select the required type of view, Prover configurations to mention the prover details text box and data details text box to mention the data. Aggregate Measurements section to view the aggregate and comparison results. Visualization chart section to view the charts.

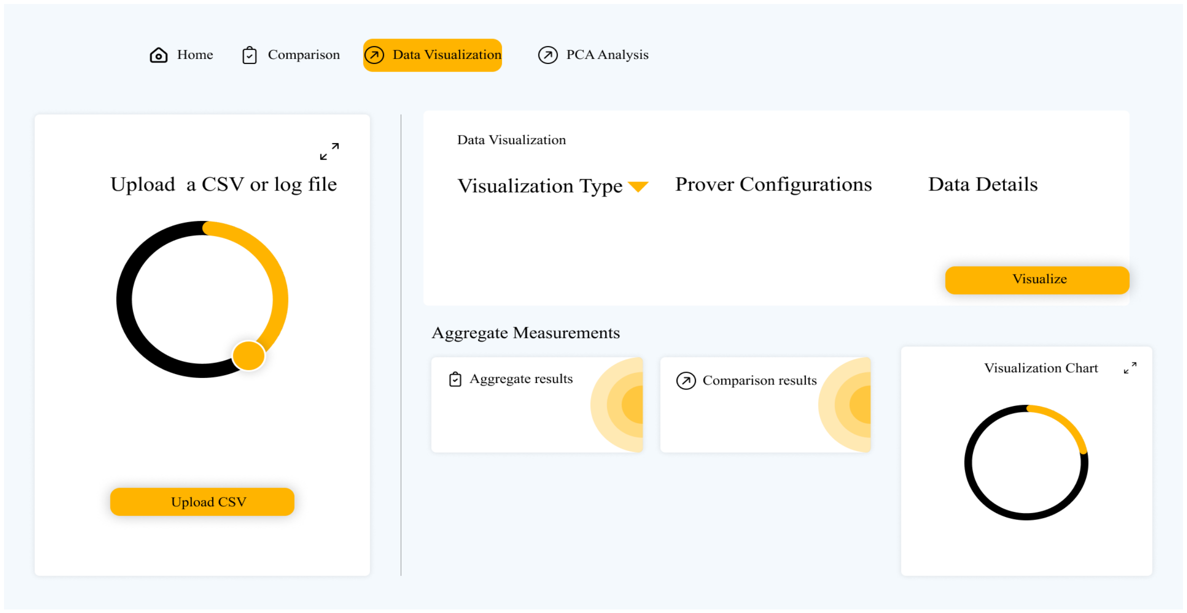

Figure 3: Wireframes

Screen 4: A PCA analysis tab to identify the key 100 inputs. PCA bi-plot to visualize the bi-plots and component ladings to view the coefficients, Visualization chart section to view the charts.

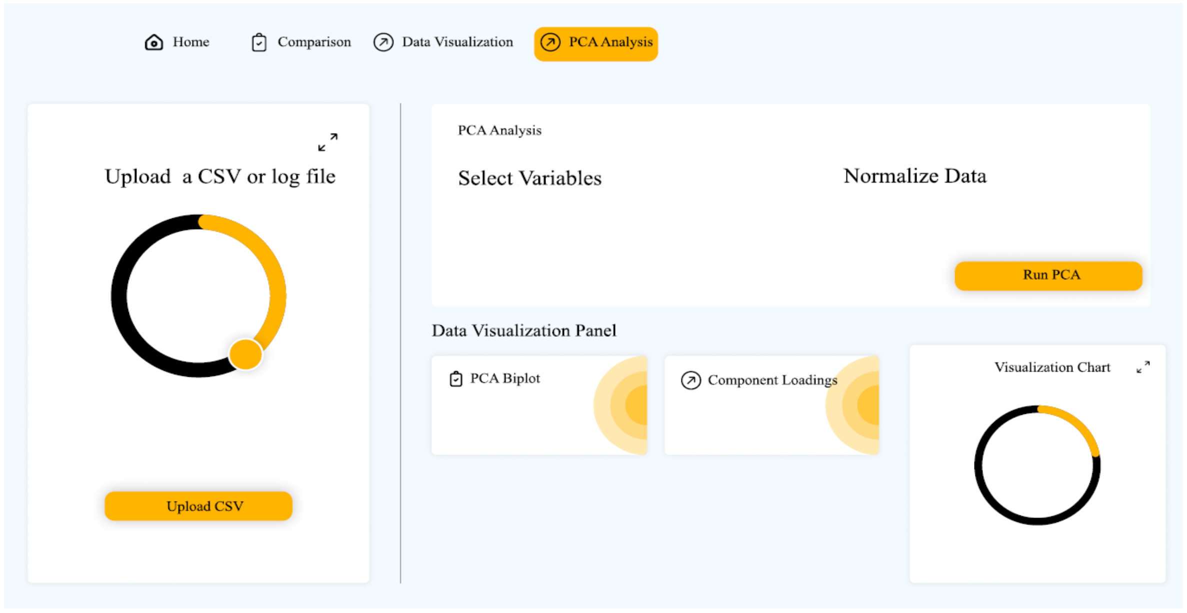

Figure 4: Wireframes

11 Project plan

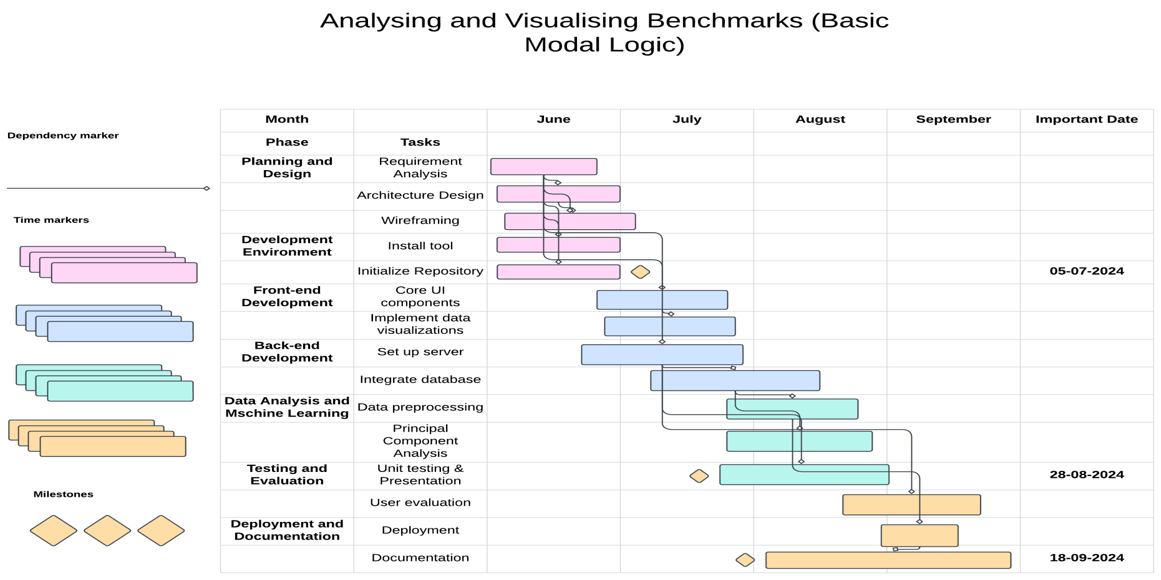

Figure 5: Gantt Chart

12 Risks and Contingency Plans

Risks	Contingencies	Likelihood	Impact
Hardware Issues	Backup of the files and code	Low	High
Software Issues	Version Control - Git	Medium	Medium
Time Overrun	Strict scheduling, priority management	Medium	High
Security vulnerabilities	Conduct regular security assesments	Low	Medium
Third-party APIs	Dependence on third-party APIs for data visualization	Low	Medium

12.1 Risk Mitigation Strategies

- **Compatibility issues:**
  - **Mitigation:** Conduct cross-browser and cross-platform testing during development.
  - **Contingency:** Develop fallback options for critical functionalities.

- **Performance bottlenecks**:
  - **Mitigation:** Implement performance testing early in the development cycle.
  - **Contingency:** Optimize database queries and consider scaling resources if needed.

- **Security vulnerabilities**:
  - **Mitigation:** Follow secure coding practices and conduct regular security audits.
  - **Contingency:** Have a response plan for immediate patching and data recovery.

12.2 Contingency Planning

- **Response Plan:** Define escalation procedures for critical issues like security breaches or severe performance issues.
- **Timeline:** Develop contingency timelines for critical milestones to minimize project delays.
- **Monitoring and Review:** Conduct risk assessment sessions periodically to identify new risks or changes in existing ones.
- **Documentation:** Maintain a Risk document with all identified risks, their assessments, mitigation strategies, and contingency plans.

References

Abdi, H. & Williams, L. J. (2010), 'Principal component analysis', *Wiley Interdisciplinary Reviews: Computational Statistics* **2**(4), 433–459.
URL: *http://dx.doi.org/10.1002/wics.101*

Biere, A., Heule, M., van Maaren, H. & Walsh, T. (2009), *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*, IOS Press, NLD.

Nalon, C., Hustadt, U. & Dixon, C. (2018), 'A resolution-based theorem prover for k_n: Architecture, refinements, strategies and experiments', *Journal of Automated Reasoning* **64**.