1. Premise and Introduction

In the aeronautics sector, a successful launch and mission holds the potential to significantly bolster the reputation and innovation credentials of the organization, affording it a distinct competitive advantage, particularly when the spacecraft in question embodies state-of-the-art technology. Conversely, unsuccessful project launches can inflict substantial harm, like damage to their reputation. Over a sixteen-year span, small satellite initiatives have experienced a notable 41.3% failure rate, comprising a 24.2% total failure rate, 11% partial failure rate, and a 6.1% launch failure rate (Jacklin, 2019). Given the gravity of these statistics, it is essential that collective efforts are directed toward reducing this failure rate to a more reasonable and desirable level, approaching zero. However, a fundamental question makes us all wonder: How can we achieve a near-zero percent failure rate? To answer, we must scrutinize a key factor contributing to project failures.

The propulsion system, for instance, plays a pivotal role in the incidence of project failures, with the feeding system accounting for approximately 65% of all propulsion-related failures (Fernández et. al, 2022). To ensure the integrity of the propulsion system, referencing Section 6 of the NASA-STD-5012 B technical standard is necessary, as it talks about various pressurized components within the system, such as hydraulic pumps, bellows, and combustion chambers. While this section is generally informative, it is not without its shortcomings. Terms like "MDC loads" and "FOS" can be rather ambiguous, owing to the lack of information and/or lack of clarity in the document's exposition. Because of this, project teams may falter, incurring both time and financial losses. This shows how vague technical documentation can harm a mission, which is further exemplified by the fact that an alarming 45 to 60% of all incidents reported in the Aviation Safety Report Systems (ASRS) are attributed to issues arising from them (Avers et al., 2012). So, how exactly can a group of six software engineers devise a plan to develop a program that can alleviate these challenges faced by technicians?

This question was inspired by the concept of Human-Centered Computing, with a specific focus on human-system modeling. The way human-system modeling works, lies in a model that can emulate human decisions and performance (Shafto and Hoffman, 2002). Recognizing that the process of processing documentation can be time-consuming and sluggish, we strived to create a model capable of simulating proofreading while delivering the utmost accuracy and performance so that technicians can focus on what they do best. As this hackathon concludes, the Intergalactic Pirates are excited to introduce a groundbreaking solution harnessing the power of artificial intelligence (AI) and large language models (LLMs) to streamline various technical standards in aeronautics: Aero Copilot.

Aero Copilot is a state-of-the-art document processing system backed by AI that streamlines and corrects technical documents so that technicians can save time and money, as well as reducing the high rate of spacecraft failure because of technical documentation, so that

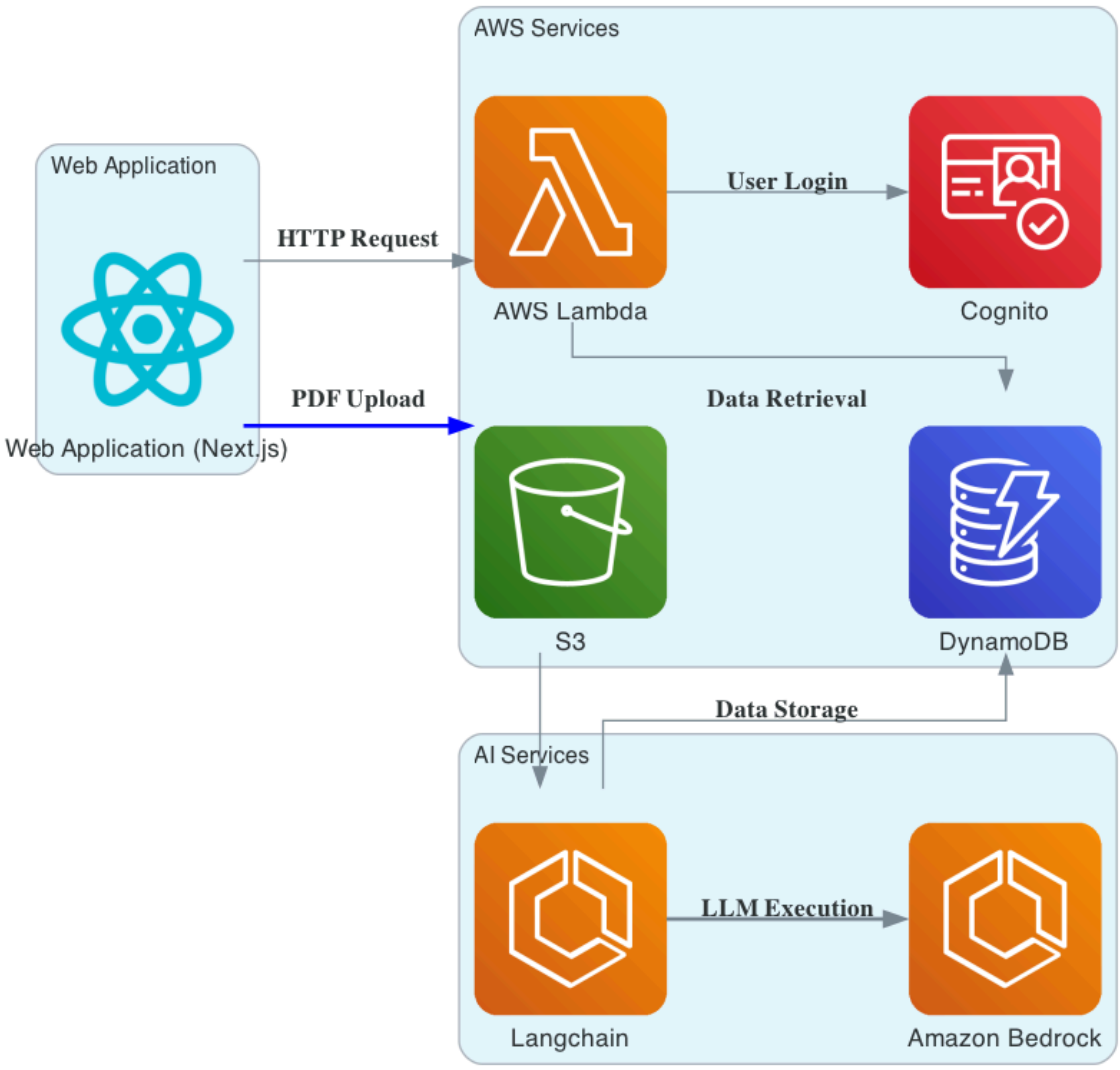
they can focus on launching a successful mission. This technical document will serve as a comprehensive guide on how to use the tool. There will be a brief overview of the product and the impact that it will cause, how to set it up, a look at how the system functions with insights into its functionality, explanations on things like security, deployment, testing methodologies and performance, and a roadmap for further enhancements.

2. The Impact of Using Aero Copilot

The usage of Aero Copilot benefits everyone involved, from consumers to space agencies. With Aero Copilot ensuring that technical documents can be consistent and relevant, there would be a global standard for spacecraft development, enabling collaboration with other space agencies to harmonize data. Additionally, its processing speed and reliability by the trained LLM, makes producing safer standards more efficient and an increase in satisfaction, due to the lack of time reading through standards. These enhanced safety standards can also allow passengers and consumers to have greater confidence in public space transportation services, increasing usage and trust in this revolutionary transportation. Finally, data analysis capabilities that Aero Copilot provides will allow organizations to generate insights in their document processes, allowing data-driven decision making and continuous improvements in the technical document. Based on these applications and the benefits they provide, Aero Copilot's wide-ranging benefits extend to everyone, creating a new era of standardized, efficient, and data-driven spacecraft development that enhances safety, satisfaction, and trust in space transportation services while fostering global collaboration and continuous improvement.

3. System Overview

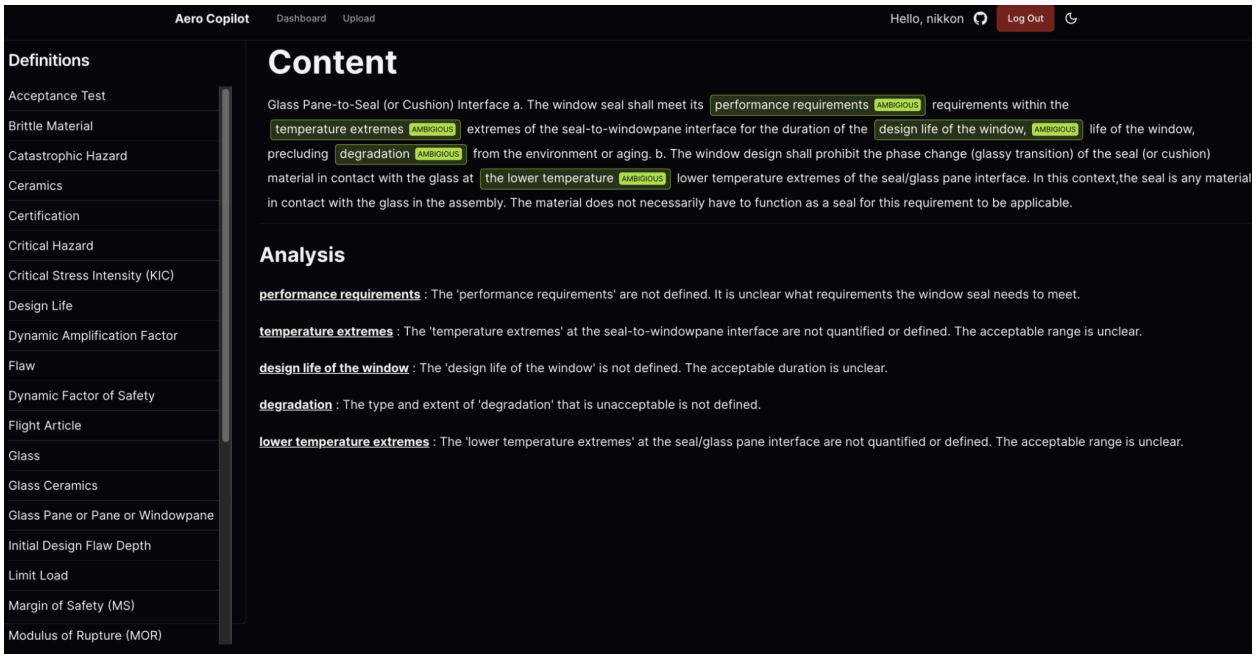
Aero Copilot's system architecture is built using a microservices architecture consisting of Amazon Web Services (AWS), LangChain and Claude 2 via Amazon Bedrock. The technical architecture that we used is as follows:



From the Flutter application, a request is thrown to AWS Lambda (a service that automatically runs code), where technician sign-up and verification is provided by AWS Cognito. The profile that was created by the technician is then stored using a database provided by DynamoDB. Once a technician uploads a PDF in the website, it gets transferred to an AWS S3 bucket for aggregating and parsing the text.

The text will be compared to other technical standards, lessons, bulletins and other documents before being stored in a raw text database in a knowledge store. This database also gets referenced in the backend during the searching phase of the document. Based on semantics, the raw database becomes a vectorized database which is fetched by the backend as snippets of the document that are related to the PDF. The backend also needs similar document snippets, which are the knowledge graphs from the vectorized database being linked to relevant documents. The raw and vectorized databases, as well as the knowledge graphs are stored remotely on the cloud for future uses.

With similar and relevant content fetched from the knowledge store, an automated prompt is created to suggest improvements based on the information that was fetched. The document then goes through the LLM for each section. The custom prompt scrapes all technical keywords, phrases and concepts and checks if a term is ambiguous and should be clarified, by comparing with the vectorized database and knowledge graphs by similarity. Finally, it will show the text with any ambiguous terms flagged, with additional details being provided to the technician at the bottom half of the website. This is what the expected output should be like in an earlier version of this application:



4. A Step-by-Step Guide on Getting Started

1. To start, Make sure you have an AWS account, Python and Next.js installed on your computer.
2. Clone the repository on GitHub.
3. Set up your AWS Credentials.
4. Install packages that are going to be used in the project.

5. Configure AWS services and environmental variables.

5. User's Guide

Here is a process on how it works:

1. Register an account on Aero Copilot and login.
2. Upload your PDF to the website through the user-friendly web interface.
3. Refresh the page and click on the new file. The text will be flagged if the LLM has deemed it ambiguous.
4. The technician should review and accept suggestions for document improvements, which will be tracked and managed into an automated database.

6. System Functionality

Here is a list on the functionality of the system:

- PDF Ingestion: PDF documents are ingested into AWS S3 bucket and processed using a custom script
- AI Document Analysis: Textract analyzes technical PDFs, extracting content and identifying areas for improvement.
- Suggestion Generation: Custom algorithms analyze parsed results to generate document improvement suggestions.
- Workflow Automation: JIRA implementation allows .

7.1. Details about Security

Since we are building a web application, security needs to be prioritized to prevent any leaks from happening. To provide security for the end user, we ensured authentication with the integration of AWS Cognito, encrypted all data from the parsing of the data to the final output and restricted user actions by access control.

7.2. Details about Application Deployment

For scalability throughout the project, AWS, along with Lambda and the API Gateway was implemented. We also used strategies that can only be done on a scalable platform, like load balancing and auto-scaling to boost performance. CloudWatch was also implemented to provide logging in real time.

7.3. Details about Application Test Cases

We used unit tests to cover each part of Aero Copilot, integration cases for the interaction of multiple systems and user acceptance tests as an example for how it showcases real world usability.

7.4. Details about Application Performance

Currently, this application is scientifically valid and can work in real life, but we need more time to optimize the performance of our model. As of right now, implementations of the scalable load balancing and auto-scaling boosts the performance and the reliability of the application. Additionally, we will improve the performance for years to come as we further optimize our Amazon Bedrock model.

7.5. Details about Documentation

Along with this guide, API documentation is available for developers and code documentation is provided by Python docstrings and JavaScript comments.

7.6. Details about Accessibility

Next.js is used for this project, instead of other frameworks because of its variety of accessibility features. So far, a light and a dark mode are implemented, which allows for improved text readability. There will be more updates that can improve accessibility in the near future.

8. Application Roadmap

Our roadmap consists of trying to get the front-end and Amazon Bedrock to work together, so that the application can be used for all technical PDFs; for the technical example, it was hardcoded. Then, we will enhance our AI and fine-tune the model, so we can optimize the performance to the best of our abilities. Collaboration with other technicians will also be implemented in the future, so that others can use the suggestions for a group project. Finally, to make it more usable for everybody, polishing the user experience and implementing accessibility features like text to speech and text resizing will be our top priority going forward.

If technicians want more features, they can submit feature requests through GitHub.

9. Conclusion

While working on this project, we have learned a bit about how important AI can be in processing documents. We have witnessed the benefits that AI can bring, like improving the efficiency of checking all the PDF files, compared to checking them by hand. We have also witnessed AI can also make us more collaborative with fellow group members with the reduction of time needed to check all files. If it were not for AI, we would not be writing an entire project at the end of this hackathon, but we would still be checking a PDF.

To conclude, our team has learned a lot from building this project. We are a vibrant blend of cultures, backgrounds, and knowledge levels. We have a Google Developer Student Club Lead from Waterloo, an enthusiastic high school student and one hails from as far as King City, a

distant oasis from the bustling heart of Toronto. The challenge we embarked upon was undeniably tough, but we rose to the occasion, working in harmony as a team and producing something truly extraordinary. We are elated that we chose to tackle the STAR (Standards Technical Assistance Resource) challenge. Throughout this journey, we had an absolute blast, delving into the world of machine learning, seamlessly managing both software and hardware aspects, and learning how to bridge these technologies with the realm of space. Our optimism is boundless as we look forward to presenting our final project, a testament to our collective effort, innovation, and unwavering teamwork. We sincerely hope that all the love and care put into Aero Copilot can make this the new technical standard for aerospace.

Reference List

Avers, K., Johnson, B., Banks, J., & Wenzel, B. (2012). *Technical documentation challenges in aviation maintenance: A proceedings report*. Federal Aviation Administration, Office of Aerospace Medicine.

Fernández, L. A., Wiedemann, C., & Braun, V. (2022). Analysis of Space Launch Vehicle Failures and post-mission disposal statistics. *Aerotecnica Missili & Spazio*, *101*(3), 243–256. https://doi.org/10.1007/s42496-022-00118-5

Jacklin, S. A. (2019, March). *Small-satellite mission failure rates - NASA technical reports server ...* NASA. https://ntrs.nasa.gov/api/citations/20190002705/downloads/20190002705.pdf

Shafto, M. G., & Hoffman, R. R. (2002). Human-centered computing at NASA. *IEEE Intelligent Systems*, *17*(5), 10–14. https://doi.org/10.1109/mis.2002.1039827