sUAS Safety Case Tool
Project Design

TEAM: GRP_3_AERIAL
AUTHORS: Brady Bargren, Blake Bryan, Benjamin Kelly,
Gabriel Perez, and Gautham Suresh

December 16, 2024

Executive Summary

Development Standards & Practices Used

- XML Parsing: Handling complex XML models for generating questionnaires.
- Frontend Development Standards: React, TypeScript, Bootstrap for responsive and accessible user interfaces.
- Backend Practices: Node.js with Express for API development, MongoDB for data persistence.
- YAML Safety Case Standards: Goal Structuring Notation (GSN) for visual safety diagrams.
- Risk Assessment Integration: Algorithms for dynamic risk evaluation.

Summary of Requirements

- Generate safety cases dynamically based on user input and flight configuration.
- Provide a user-friendly frontend interface for pilots.
- Store user responses securely in a MongoDB database for analysis and retrieval.
- Incorporate a risk assessment algorithm to evaluate flight safety.
- Generate visual safety diagrams with color-coded risk levels.

Applicable Courses from Iowa State University Curriculum

- COM S 3090: Software Development Practices
- COM S 3110: Algorithms
- CPR E 2880: Embedded Systems
- SE 3190: User Interface Design
- COM S 3630: Databases
- SE 3290: Software Project Management

New Skills/Knowledge Acquired

- Parsing and processing XML models dynamically.
- Creating Goal Structuring Notation (GSN) diagrams programmatically.
- Developing algorithms for automated risk assessment.
- Frontend-backend integration for real-time data processing.
- Visualizing safety data in SVG format using custom scripts.

Contents

1 Introduction

1.1 Acknowledgement

The team extends gratitude to our client, Dr. Myra Cohen, for her guidance and for providing the project scope. We also acknowledge Iowa State University for offering a robust curriculum that prepared us for this project.

1.2 Problem and Project Statement

General Problem Statement: Small Uncrewed Aerial Systems (sUAS) are increasingly used in shared airspace, posing risks due to hardware failures, software bugs, human errors, and adverse environmental conditions. Currently, no formalized safety case generation exists for sUAS, leaving pilots ill-equipped to assess flight safety.

General Solution Approach: Our project addresses this gap by developing a web-based tool that programmatically generates safety cases for sUAS. By analyzing user input and flight conditions, the tool generates a visual safety case and evaluates associated risks.

1.3 Operational Environment

The tool is expected to operate on modern web browsers and is accessible from any device with internet connectivity. The backend server runs on cloud-hosted or local environments, and the MongoDB database handles concurrent user queries efficiently.

1.4 Requirements

Functional Requirements

- Parse XML models to generate questionnaires dynamically.
- Store and retrieve user responses securely.
- Generate safety cases in YAML and visualize them as GSN diagrams.
- Assess and display flight risk levels dynamically.

Non-Functional Requirements

- The tool should have low latency and handle multiple users.
- Ensure data security for stored user responses.
- Provide an intuitive and accessible user interface.

1.5 Design Assumptions and Limitations

Assumptions

- Users have basic knowledge of sUAS operations and risk factors.
- The system will primarily be used in the United States.

Limitations

- The system is limited to configurations described in the XML model.
- Real-time weather data and NOTAMs integration depend on external APIs.

2 Architectural Diagram (Figure 1):

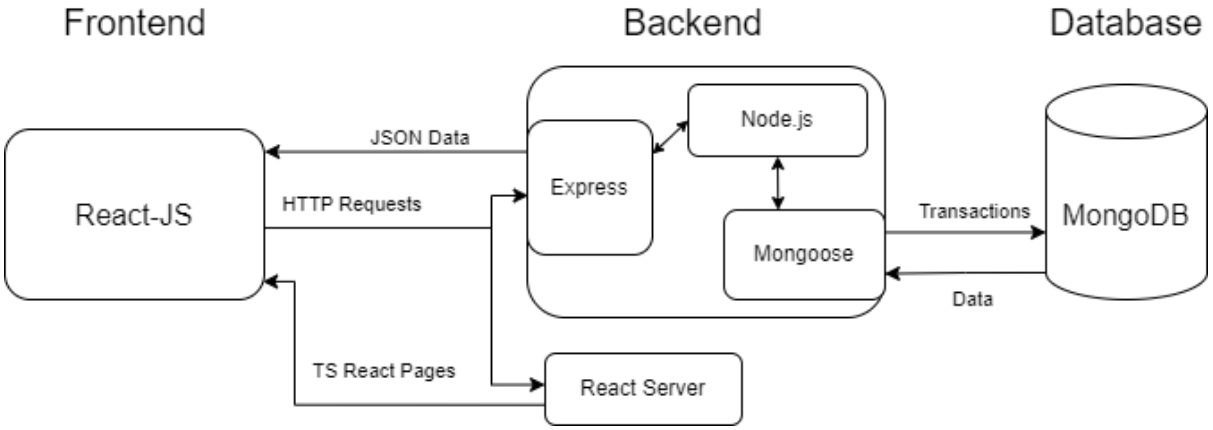

Figure 1:

3 Components

3.1 Component-1: Frontend Application

- Built using React and TypeScript.
- Dynamically generates questionnaires based on XML model input.
- Handles user responses and displays safety case diagrams.

3.1.1 Module 1: Dynamic Questionnaire Generator

- Parses XML and generates flight-specific questions.

3.1.2 Module 2: User Interface

- Designed with Bootstrap for consistent styling.
- Includes features like cookie-based state management.

3.2 Component-2: Backend Server

- Developed using Node.js and Express.
- Interacts with MongoDB for storing and retrieving data.

3.2.1 Module 1: Risk Assessment Algorithm

- Evaluates user responses and calculates overall flight risk.

3.2.2 Module 2: Safety Case Generator

- Converts user responses into YAML and visualizes them as GSN diagrams.

4 Data Decomposition

- **Types of Data:**
  - User responses gathered through dynamic questionnaires.
  - XML models representing flight configurations and system dependencies.
  - Safety case configurations stored as JSON, YAML, and argument files.
  - Risk assessment data and generated visual artifacts (e.g., GSN diagrams).
- **Persistent Storage:**
  - All data is stored in a MongoDB database, with collections for user responses, XML models, and generated safety case files.
  - Unique identifiers (e.g., timestamps) are assigned to each session, enabling data versioning and retrieval.
  - Safety case data is stored in structured formats (JSON/YAML) to support visualization and regulatory analysis.
- **Data Relationships:**
  - User responses are mapped to XML models to derive safety case nodes.
  - Each node in the safety case tree is associated with a specific risk level based on user responses.
  - Safety case diagrams (GSN) are generated from the processed YAML files, showing dependencies and risk assessments.
- **Data Flow:**
  - Data flows from the frontend (user input) to the backend, where it is processed and stored in MongoDB.
  - Stored data is transformed into YAML safety case files, which are used to generate GSN diagrams.
  - Pruned safety case trees are generated to highlight nodes that fail risk checks, aiding in visualization.
- **Data Integrity and Security:**
  - MongoDB employs access controls and authentication to secure data.
  - Safety case data is validated before storage to ensure compliance with the XML model schema.
  - Sensitive data is anonymized where applicable, especially when generating diagrams for public use.

- **Example Data Structures:**
  - **User Responses:** Stored as JSON with key-value pairs linking questions to answers.
  - **Safety Case Tree:** Represented as a hierarchical JSON/YAML structure with nodes, attributes, and risk classifications.
  - **Generated Artifacts:** Stored as SVG files for GSN diagrams, alongside YAML/JSON metadata.

5 Detailed Design

Documentation: https://git.las.iastate.edu/SeniorDesignComS/2024fall/402c/grp_3_aerial/-/tree/main/docs?ref_type=heads

6 Design Rationale

6.1 Design Issues

Issue 1: XML Parsing Complexity

- **Description**: XML models have nested structures with AND/OR logic.
- **Factors Affecting the Issue**:
  - XML models' complex, hierarchical structure.
  - Need for recursive parsing to handle depth and logical dependencies.
- **Alternatives and Their Pros and Cons**:
  - **Flat Parsing Approach**: Simplifies parsing but loses logical relationships.
  - **Custom XML Parser**: Maintains structure but requires more development effort.
- **Resolution**: Implemented recursive tree traversal, maintaining logical dependencies and handling depth effectively.

Issue 2: Dynamic Risk Calculation

- **Description**: Risk levels depend on multiple user inputs.
- **Factors Affecting the Issue**:
  - Complexity of input combinations affecting scoring accuracy.
  - Lack of a previously established risk scoring algorithm.
- **Alternatives and Their Pros and Cons**:
  - **Static Scoring**: Simpler but not adaptive to varying inputs.
  - **Dynamic Algorithm**: Adaptive but required iterative testing and refinement.
- **Resolution**: Developed a scoring algorithm to assess risk dynamically based on user inputs. The current algorithm needs further refinement for accuracy.

Issue 3: Submit Button Losing Functionality

- **Description**: The submit button occasionally failed to navigate correctly after submission.
- **Factors Affecting the Issue**:
  - Delays in backend response during submission processing.
  - Asynchronous issues in handling navigation after form submission.
- **Alternatives and Their Pros and Cons**:
  - **Adding Timeout for Navigation**: Unreliable for slower connections.
  - **Force Navigation on Submission**: Ensures user progression but risks incomplete data transfer.
- **Resolution**: Forced the submit button to navigate to the next page regardless of submission completion status, ensuring continuity in user experience.

Issue 4: YAML vs. Alternative Goal State Notation

- **Description**: The GSN2X generator was incompatible with dynamic generation due to horizontal indexing issues, necessitating a switch to alternative goal state notation.
- **Factors Affecting the Issue**:
  - YAML safety cases preferred by stakeholders like NASA.
  - Visual and dynamic generation challenges in YAML using GSN2X.
- **Alternatives and Their Pros and Cons**:
  - **Use of Alternate Notation**: Easier to dynamically generate but lacks stakeholder familiarity.
  - **Refinement of GSN2X for YAML**: Aligns with stakeholder needs but requires extensive debugging.
- **Resolution**: Temporarily switched to an alternate goal state notation. Future efforts must focus on resolving GSN2X issues for YAML compliance.