Software Requirements Specification

for

AviationQuiz

Prepared by Agape Ioan  
Macovei Răzvan  
Chiriac Mihai  
Caraulan Egor

May 26, 2025

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 2

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 5

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 6

3.3 Software Interfaces 6

3.4 Communications Interfaces 6

4. System Features 6

4.1 Discipline Selection 7

4.2 Quiz Execution 7

4.3 Help Access 7

4.4 Error Handling 8

5. Other Nonfunctional Requirements 8

5.1 Performance Requirements 8

5.2 Safety Requirements 8

5.3 Security Requirements 9

5.4 Software Quality Attributes 9

5.5 Business Rules 9

6. Other Requirements 9

Appendix A: Glossary 9

Appendix B: Analysis Models 9

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

AviatieQuiz is an interactive desktop quiz application designed to help users learn and test their knowledge in the field of aviation. The system is structured using a Model-View-Presenter (MVP) architecture, promoting separation of concerns and maintainability. Users can select a discipline from a list, start a quiz, view questions with multiple-choice answers, and receive a final score upon completion. The system supports loading questions from a JSON file (quiz.json), displaying a help file (ajutor.chm), and managing user interactions via a graphical interface developed in Windows Forms. The quiz logic ensures that the user selects an answer for each question and progresses through the quiz sequentially, with the option to finalize the quiz at the end. The project is built using C# and is designed to be a simple, standalone Windows application without requiring external dependencies. The details and specifications of the application’s functionality are defined in sections 3 and 4 of this document. An overview of the system is presented in section 2, and a complete list of requirements is detailed in section 5. This documentation represents the first complete version of the AviatieQuiz project.

## Document Conventions

This document follows the IEEE standard formatting for software development. The structure adheres to a logical and consistent style, using Times New Roman, 12-point font, with single line spacing. All content is written in third-person, passive voice, and uses clear, concise, and grammatically correct English. Each requirement is uniquely identified with a tag (e.g., REQ-1) for easy reference.

## Intended Audience and Reading Suggestions

This document is not intended for the end user because it provides a detailed specification of how the software is to be implemented. Since a user would need information on how to use the quiz application, rather than how to build it, this document is primarily intended for the developers, testers, and professors responsible for validating the functionality of the application. The document begins with an overview of the software system and its major components in section 2, followed by detailed descriptions of external interfaces in section 3. Section 4 covers the system features in depth, outlining the core functionality of the application, while section 5 includes the nonfunctional requirements that ensure quality, security, and performance.

Readers should start with section 2 for an overall understanding of the application. Testers are encouraged to review sections 5.1 through 5.4 for quality-related aspects, then section 3.1 for user interfaces, and section 4 for functionality details. Developers should read the document in full, as it is designed to support the development and testing of the AviatieQuiz project.

## Product Scope

AviatieQuiz is a desktop quiz application developed using C# and the Windows Forms framework. The application provides an interactive platform where users can select a discipline from a list and answer multiple-choice questions related to aviation topics. The goal of the software is to enhance the learning experience of students by offering a structured and engaging way to test their knowledge, receive instant feedback, and track performance through final scores.

The quiz system includes key features such as discipline selection, question display, answer tracking, and scoring. Users interact with the application through a graphical interface, where they can navigate questions, select answers, and complete quizzes. A help file (ajutor.chm) is provided for additional guidance. The application is designed to run on Windows machines with minimal system requirements and does not rely on external services or network connectivity.

AviatieQuiz supports the educational objectives of the institution by offering a practical tool for reinforcing theoretical knowledge in aviation studies. The software aligns with academic goals by providing a reliable, user-friendly platform that enhances student engagement and knowledge retention.

## References

N/A

# Overall Description

## Product Perspective

AviatieQuiz is an interactive quiz application developed using C# and Windows Forms. It is a new, self-contained software product created specifically to support students in learning aviation theory through an engaging and interactive question-and-answer format. AviatieQuiz is not part of a product family, nor does it replace an existing system. It is a standalone tool designed to be lightweight and easy to use, without relying on external services or network connectivity.

The application structure is inspired by educational quiz systems and learning management tools, combining features commonly seen in e-learning platforms and exam simulators. For instance, applications like Kahoot and Socrative allow users to interactively answer questions and receive immediate feedback, but AviatieQuiz focuses on a more streamlined, local desktop experience. Users can select a specific discipline (such as Meteorology, Navigation, or Aircraft Systems), start a quiz session, answer multiple-choice questions, and receive a final score upon completion. The system reads all quiz data, including questions and answers, from a local JSON file (quiz.json) and allows users to access a help file (ajutor.chm) for additional guidance.

AviatieQuiz employs a Model-View-Presenter (MVP) design pattern, with distinct responsibilities for the interface (Windows Forms), presentation logic (Presenters), and data handling (Models). The user interface is composed of forms such as the MainForm and QuizForm, which provide visual elements like combo boxes, buttons, and radio buttons for user interaction. The Presenter classes manage the flow of data and user input, ensuring a clear separation between the user interface and the underlying logic.

The software is designed to run on any Windows computer with the .NET Framework installed, requiring only minimal system resources. By offering a local, offline solution, AviatieQuiz ensures accessibility and ease of use, making it an ideal tool for educational settings, particularly for aviation students who need a focused environment to reinforce their theoretical knowledge. The overall system is modular and extendable, with the possibility of adding new disciplines, features like quiz history, or enhancements to the user interface in future versions.

## Product Functions

Major functions that AviatieQuiz must perform for its end users are as follows:

* Discipline selection – Users must be able to select a quiz discipline (such as Meteorology, Navigation, or Aircraft Systems) from a list.
* Quiz generation – The system must load questions and answers for the selected discipline from a local JSON file (quiz.json).
* Interactive question display – The quiz must present questions one at a time, along with multiple-choice answer options, allowing the user to select their response.
* Quiz progression – Users must be able to move through quiz questions sequentially, with navigation controlled by a “Next” button.
* Final score calculation – The system must calculate and display the user’s final score upon quiz completion.
* Help access – Users must be able to open a help file (ajutor.chm) that provides additional information about the application’s use.
* Error handling – The system must display error messages when critical resources (like the JSON or CHM files) are missing or when invalid actions are attempted (e.g., starting a quiz without selecting a discipline).
* User interface – The application must provide a graphical user interface (GUI) using Windows Forms, including forms for discipline selection, quiz interaction, and result display.

## User Classes and Characteristics

AviatieQuiz is designed for ease of use, ensuring that any user with basic computer literacy can operate the application effectively. The primary user class consists of students and aviation enthusiasts who want to test and reinforce their knowledge in aviation theory. These users are expected to have a general understanding of the subject matter but do not require advanced technical skills to navigate the application.

The system is intended to provide a straightforward and user-friendly experience, with clear labels, simple navigation, and intuitive question flow. While most users will be students, instructors or professors may also use the application to review its content or assist students in their learning process.

Advanced users, such as educators with more technical experience, may also explore customizing or extending the quiz content by modifying the quiz.json file, although no in-app editor is provided. All user classes are supported by the same interface and feature set, ensuring consistency and ease of access for all.

## Operating Environment

AviatieQuiz is a Windows desktop application developed in C# using the Windows Forms framework. The software is designed to run on computers with the Windows operating system (Windows 10 or later) and requires the .NET 9 Runtime.

AviatieQuiz does not depend on external services or network connectivity. All necessary resources, including the question database (quiz.json) and help file (ajutor.chm), are stored locally within the application’s directory. The application is lightweight and optimized for typical educational hardware, requiring only a basic PC configuration.

AviatieQuiz is intended to coexist peacefully with other desktop applications and does not interfere with system processes or other software components.

## Design and Implementation Constraints

AviatieQuiz is being developed as a standalone desktop application using the .NET 9 framework and the Windows Forms library. The project is created as part of an academic assignment, which means time constraints and academic deadlines are significant factors influencing the scope of development.

The application is designed to run only on Windows operating systems, which limits cross-platform compatibility. Additionally, it relies on a local JSON file (quiz.json) for loading quiz data, which means users must ensure that the file is present in the correct location for the application to function properly.

Due to the project's scope, there is no integrated feature for dynamic content management, such as adding or editing quiz questions within the application; this must be done manually by modifying the JSON file. No external database systems, networked features, or web-based components are used, simplifying the system but also constraining future scalability and extensibility.

Additionally, since the application is developed in a limited timeframe, advanced features such as user authentication, quiz history tracking, or advanced analytics are considered out of scope for the current version.

## User Documentation

AviatieQuiz will include a digital user manual in the form of a compiled help file (ajutor.chm), which will be distributed alongside the application. This file will provide instructions on how to use the application, including selecting disciplines, starting a quiz, navigating questions, and viewing scores.

## Assumptions and Dependencies

We assume that the .NET 9 framework and Windows Forms will provide the necessary stability and performance for the development and execution of AviatieQuiz. The application depends on the correct installation of the .NET 9 runtime on the target system.

The system also assumes the presence of required files, including quiz.json for quiz data and ajutor.chm for user help. If these files are missing, corrupted, or improperly formatted, the application’s functionality may be limited or fail entirely.

It is assumed that users will operate the application on a Windows 10 or newer system, with sufficient hardware resources to support standard desktop applications. No third-party libraries or external services are currently used, but future versions of AviatieQuiz may introduce additional dependencies as the feature set expands.

These assumptions form the basis for the current system design and may need to be revisited if new requirements or constraints arise during future development phases.

# External Interface Requirements

## User Interfaces

The AviatieQuiz user interface is a traditional Windows Forms desktop app with a clear, intuitive layout. At startup, users must log in as either an admin or a student. There are 2 input forms for entering the login and password, as well as the “Submit” button. The main screen lets users select a discipline from a dropdown and start a quiz by clicking “Incepe Testul”. A “Ajutor” button opens the help file (ajutor.chm).

During a quiz, questions with multiple-choice answers are shown using buttons. The current question number and total are displayed. A “Următoarea Întrebare” button moves forward; it changes to “Finalizează Testul” on the last question.

Errors and feedback use standard MessageBox dialogs. The design keeps the interface simple and focused, with no keyboard shortcuts or chat features. Each form serves a single purpose: login, discipline selection, quiz, or results.

## Hardware Interfaces

AviatieQuiz requires a standard Windows PC with a mouse, keyboard, and display monitor. The mouse is used to select disciplines, answer questions, and navigate the interface. The keyboard is not directly used within the application but is necessary for launching and operating the system.

No special hardware or peripherals are required. The application runs on standard desktops or laptops with a minimum resolution of 1280x720 pixels.

## Software Interfaces

AviatieQuiz requires the .NET 9 runtime, which must be installed on the user’s computer. The application is developed in C# using Windows Forms and runs only on Windows 10 or later. The .NET 9 runtime can be downloaded from the official Microsoft website:  
<https://dotnet.microsoft.com/en-us/download/dotnet/9.0>

Quiz data is loaded from a local quiz.json file, and the help file (ajutor.chm) is accessed through the Windows help viewer. No external databases, APIs, or network services are used. All data interactions are local.

## Communications Interfaces

AviatieQuiz is a standalone desktop application and does not require any network or communication interfaces. The application runs entirely on the user’s local machine, without the need for an internet connection, web browser, or external communication protocols such as HTTP, FTP, or email.

All data is read from and written to local files (quiz.json, ajutor.chm), and no external messaging, synchronization, or security protocols are used.

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## Discipline Selection

4.1.1 Description and Priority

Allows users to select a quiz discipline from a predefined list. High priority, as it is essential for starting a quiz.

4.1.2 Stimulus/Response Sequences

* + User opens the main form.
  + User selects a discipline from a dropdown menu.
  + The application displays the selected discipline and enables the “Start Quiz” button.

4.1.3 Functional Requirements

* + REQ-1: The application shall load a list of available disciplines from quiz.json.
  + REQ-2: The application shall display the disciplines in a dropdown menu on the main form.
  + REQ-3: The user must select a discipline before starting a quiz.
  + REQ-4: If no disciplines are available, an error message must be shown.

## Quiz Execution

4.2.1 Description and Priority

Manages the display of questions and answers, tracks user selections, and calculates the final score. High priority.

4.2.2 Stimulus/Response Sequences

* + User clicks “Start Quiz.”
  + Application displays the first question and possible answers.
  + User selects an answer and clicks “Next.”
  + Application moves to the next question or shows the final score when complete.

4.2.3 Functional Requirements

* + REQ-5: The application shall load quiz questions for the selected discipline.
  + REQ-6: The application shall present one question at a time with multiple-choice answers.
  + REQ-7: The application shall track user selections and calculate the score.
  + REQ-8: The application shall display the final score upon quiz completion.

## Help Access

4.3.1 Description and Priority

Allows users to open the help file (ajutor.chm). Medium priority.

4.3.2 Stimulus/Response Sequences

* + User clicks the “Help” button.
  + The system opens the help file in a separate window.

4.3.3 Functional Requirements

* + REQ-9: The application shall provide a “Help” button on the main form.
  + REQ-10: The application shall open ajutor.chm when the user requests help.
  + REQ-11: If the file is missing, an error message must be displayed.

## Error Handling

4.4.1 Description and Priority

Ensures the application handles missing files and invalid actions gracefully. High priority.

4.4.2 Stimulus/Response Sequences

* + User starts the quiz without selecting a discipline: error message shown.
  + quiz.json file missing: error message shown.
  + ajutor.chm file missing: error message shown.

4.4.3 Functional Requirements

* + REQ-12: The application shall validate required files (quiz.json, ajutor.chm) on startup and display error messages if missing.
  + REQ-13: The application shall prevent the quiz from starting if no discipline is selected.

# Other Nonfunctional Requirements

## Performance Requirements

AviatieQuiz must run smoothly on computers with Windows 10 or later, equipped with the .NET 9 runtime. The application is designed for minimal system resource usage and should perform reliably on systems with at least 4 GB of RAM and a standard integrated graphics card.

Quiz questions should load in under 1 second after the user clicks the “Start Quiz” button, ensuring a responsive user experience. Navigation between questions must also occur instantly, with no perceptible delay. Error messages and help files should display within 2 seconds of user request.

No high-performance hardware or specialized configurations are required. The application must maintain stable performance without freezing or crashing during normal use.

## Safety Requirements

AviatieQuiz is an educational quiz application and does not include flashing lights, rapid image transitions, or other visual elements known to trigger photosensitive epilepsy. However, users with known sensitivities are advised to use the application with caution.

There are no additional safety certifications required for AviatieQuiz, as it is intended for use in academic environments on standard Windows PCs.

## Security Requirements

AviatieQuiz is a local desktop application and does not collect or store any personal user information. No user authentication or identity management features are implemented, as the application operates entirely offline.

The application uses local files (quiz.json and ajutor.chm) for all data and does not transmit data over a network. There are no network connections, reducing the risk of external attacks or data breaches.

## Software Quality Attributes

AviatieQuiz must be reliable, easy to use, and free of critical bugs. The system should have a simple, intuitive interface with minimal learning curve.

The code must be maintainable, allowing future updates and improvements. Quiz content can be updated by editing the quiz.json file, ensuring flexibility.

The application must run on any Windows system with .NET 9, prioritizing usability and stability over advanced features.

## Business Rules

The development and use of AviatieQuiz must comply with all university policies, academic integrity guidelines, and applicable codes of conduct. The system must not be modified, distributed, or used for commercial gain without proper authorization.

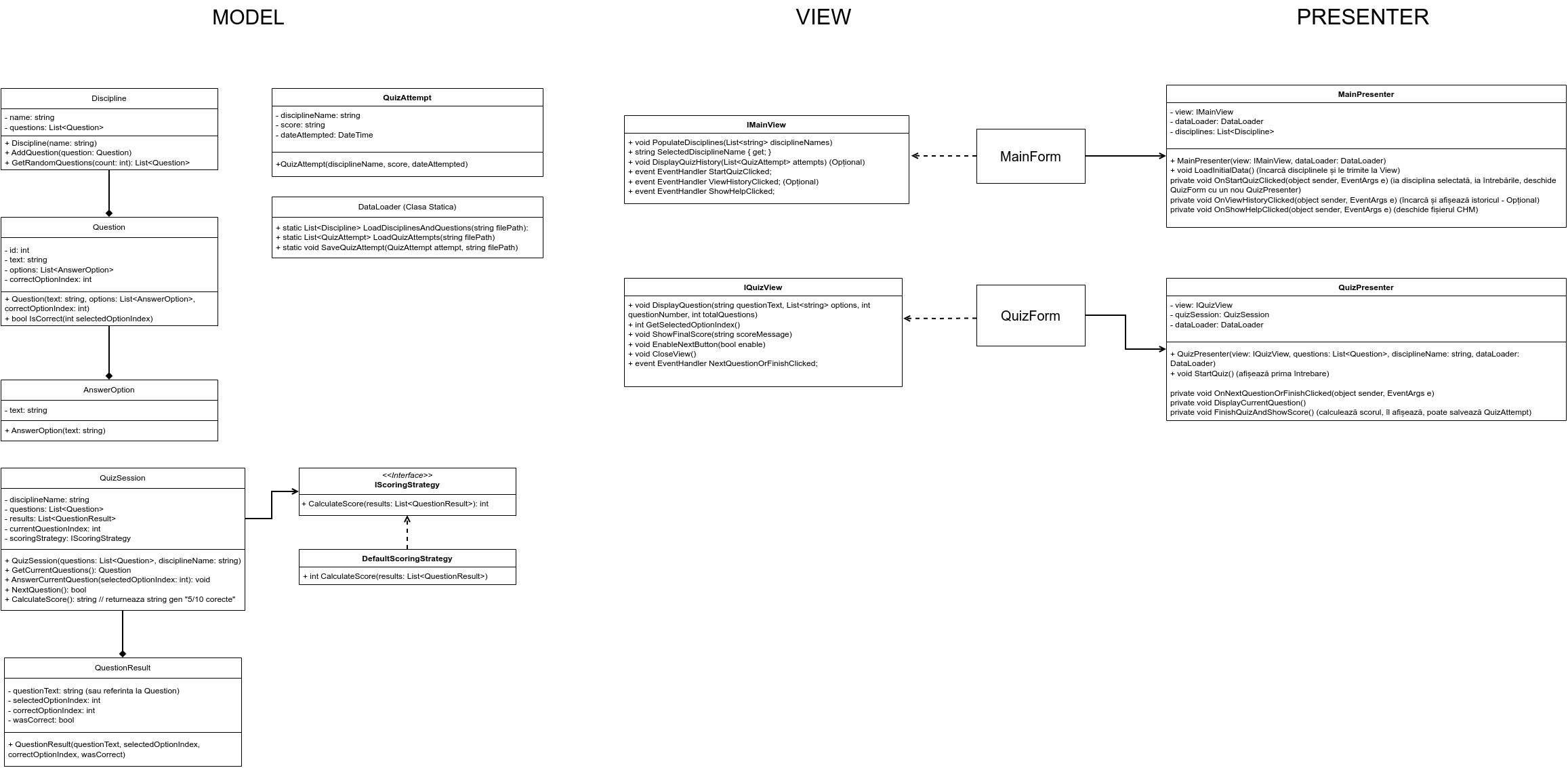
# Other Requirements

All code and resources are developed for educational purposes and are intended for internal use within academic settings.

Appendix A: Glossary

N/A

Appendix B: Analysis Models



The class diagram is also available online: <https://drive.google.com/drive/folders/112Xq2AUf161bLMv4Oyrxkupq7QfaFgTO?usp=sharing>