Airline Management system

Executive Summary:

Airline Management System is a dedicated and highly configurable system for all airlines, which canbe easily accessed by all users. It helps the users to book flights without visiting offline bookingcounters. This system can be accessible by any user from any location at any time. In such a system, apassenger should be able to view the availability of flights’ details, as per their requirement. They canbook the flights online and can also cancel the reservation. The administrator manages the passengerbooking system and updates the reservation status.

Table of Contents

[1. Introduction 5](#_Toc154151278)

[1.1 Purpose (Nikolas Filippou) 5](#_Toc154151279)

[1.2 Product Overview (Angelos Psillis) 7](#_Toc154151280)

[1.3 Terms, Acronyms, and Abbreviations (Angelos Psillis) 8](#_Toc154151281)

[2. PROJECT MANAGEMENT PLAN 9](#_Toc154151282)

[2.1 Project Organization (Giorgos Pittis) 9](#_Toc154151283)

[2.2Lifecycle Model Used (Giorgos Pittis) 11](#_Toc154151284)

[2.3 Risk Analysis (Angelos Psillis) 13](#_Toc154151285)

[2.4 Hardware and Software Resource Requirements (Angelos Psillis) 18](#_Toc154151286)

[2.5 Deliverables and Schedule (Angelos Psillis) 20](#_Toc154151287)

[2.6 Monitoring, Reporting, and Controlling Mechanisms (Nikolas Filippou) 21](#_Toc154151288)

[2.7. Professional Standards (Nikolas Filippou) 22](#_Toc154151289)

[2.8 Evidence all the artifacts have been placed under configuration management. 23](#_Toc154151290)

[3. Requirement Specifications 24](#_Toc154151291)

[3.1 Stakeholders for the system (Giorgos Pittis) 24](#_Toc154151292)

[3.2 Use Case Model and Textual Description (Giorgos Pittis) 26](#_Toc154151293)

[3.2.1 Graphic Use Case Model (Giorgos Pittis) 35](#_Toc154151294)

[3.3 Rationale for Use Case (Giorgos Pittis) 36](#_Toc154151295)

[3.4 Non-Functional Requirements(Giorgos Pittis) 37](#_Toc154151296)

[4. Architecture 40](#_Toc154151297)

[4.1) Architectural style(s)/model used: (Nikolas Filippou) 40](#_Toc154151298)

[4.2)Technology, software, and hardware used (Nikolas Filippou) 42](#_Toc154151299)

[4.3) Rationale for Choosing Client-Server Architecture: (Nikolas Filippou) 43](#_Toc154151300)

[5. DESIGN 45](#_Toc154151301)

[5.1.GUI (Giorgos Pittis) 45](#_Toc154151302)

[5.2. Static model – class diagram (Giorgos Pittis) 46](#_Toc154151303)

[5.3.Dynamic model – sequence diagram (Aggelos Psillis) 47](#_Toc154151304)

[5.4.Rationale of the detailed designs above (Giorgos Pittis and Angelos Psillis) 48](#_Toc154151305)

[5.5. Traceability from requirements to detailed design model (Giorgos Pittis) 49](#_Toc154151306)

[6. Test Plan 51](#_Toc154151307)

[6.1 Requirements/specifications-based system level test cases 51](#_Toc154151308)

[6.2 Traceability of test cases to use cases. 51](#_Toc154151309)

[6.3. Techniques used for test generation. 51](#_Toc154151310)

[6.4. Assessment of the goodness of your test suite 51](#_Toc154151311)

[7. Documentation 51](#_Toc154151312)

[7.1 Generate the user manual of your system. 51](#_Toc154151313)

[7.2 Generate the installation and deployment manual of your system. 51](#_Toc154151314)

List of Figures:

[Figure 1 The Interface of the user on how to search flights. 5](#_Toc154151315)

[Figure 2 Risk Analysis 13](#_Toc154151316)

[Figure 3 Gantt chart – Deliverables and Schedule 20](#_Toc154151317)

[Figure 4 Passenger Use Case Diagram on how to search flights. 26](#_Toc154151318)

[Figure 5 Passenger Use Case Diagram on how to book flights. 27](#_Toc154151319)

[Figure 6 Passenger Use Case Diagram for cancelling flights. 28](#_Toc154151320)

[Figure 7 Use Case Diagram on how to login/sign up as a passenger. 29](#_Toc154151321)

[Figure 8 Use Case Diagram on how to edit your profile as a passenger. 30](#_Toc154151322)

[Figure 9 Use Case Diagram on how to login as an administrator. 31](#_Toc154151323)

[Figure 10 Administrator Use Case Diagram for managing flights. 32](#_Toc154151324)

[Figure 11 Administrator Use Case Diagram on how to update flights. 33](#_Toc154151325)

[Figure 12 Administrator Use Case Diagram on how to manage passenger account. 34](#_Toc154151326)

[Figure 13 Generic Use Case Diagram of the Administrator. 35](#_Toc154151327)

[Figure 14 Generic Use Case Diagram of the Passenger. 35](#_Toc154151328)

[Figure 15 Client-Server 41](#_Toc154151329)

[Figure 16 Static model - class diagram of the airline management system 46](#_Toc154151330)

[Figure 17 Sequence Diagram of the administrator. 47](#_Toc154151331)

[Figure 18 Sequence Diagram of the user. 47](#_Toc154151332)

List of Tables:

[Table 1 Risk Analysis 9](#_Toc154150231)

# 1. Introduction

## 1.1 Purpose (Nikolas Filippou)

The purpose of the Airline Management System is to provide an advanced and user-friendly software solution that improves the management of airline operations and passenger services. This system is designed to enable and automate key functions, making it convenient for passengers to interact with the airline and for administrators to efficiently manage airline operations. The primary purposes of the system include:

**Efficient Booking:** The software authorizes passengers to book flights easily without the need for physical visits to airline counters, plus improving the booking process.

**Information Accessibility:** Passengers can readily access flight details, including schedules, destinations, and ticket pricing, which empowers them to make better decisions when planning their travels.

**Reservation Management:** Passengers can reserve seats and associated services through the system with less effort, providing a simplified and organized booking experience. The system also supports reservation modifications and cancellations for greater flexibility.

**Administrative Control:** Airline administrators can manage passenger bookings and maintain reservation statuses.

**Accessibility and Availability:** The system is accessible to users 24/7 from any location with an internet connection, enabling passengers to interact with airline services.

**Scope**:

The Airline Management System encompasses a broad scope, covering the following aspects:

A screenshot of a computer screen

Description automatically generated

Figure 1 The Interface of the user on how to search flights.

**User Management:** The system provides user accounts for both passengers and administrators. Passengers can create accounts, log in, and manage their reservations, while administrators have access to additional functionalities for control of the system.

**Flight Information:** The system maintains a complete database of flight information, including schedules, destinations, available seats, and ticket prices for both domestic and international flights.

**Booking and Reservations:** Passengers can search for flights, make reservations, and purchase tickets online. They can also cancel or modify reservations as needed.

**Payment Processing:** The system securely handles payment processing for ticket purchases, integrating with various payment gateways to ensure the safety of financial transactions.

**Administrative Controls:** Administrators can manage flight schedules, update reservation statuses, and monitor the performance of airline services.

**Security:** The system implements robust security measures to safeguard passenger information, payment details, and system integrity.

## 1.2 Product Overview (Angelos Psillis)

Capabilities:

Accessible anytime, anywhere: This system, at its core, provides ease to users from any point in the world at any time of day. Passengers can use a user-friendly interface to access flight details, create reservations, and manage their bookings, reducing their reliance on physical booking methods.

Effortless Flight Booking: The Airline Management System is a complete platform developed to provide travellers with seamless flight booking experiences. It allows customers to search, view, and book domestic and international flights without the need for offline booking counters.

Passenger-Centric Features: Passengers can access full flight information such as flight type, schedules, in-between locations, class categories, meal options, and cost. They can create, change, or cancel bookings based on their preferences, giving them greater control over their trip plans.

Admin Management: Administrators have powerful capabilities for managing passenger reservations and updating reservation statuses. This control ensures that reservations are handled efficiently and that proper records are kept.

Scenarios for Using the Product:

Booking and Reservation: A traveller plans to purchase a flight online. They log in, choose a flight based on availability and then complete the reservation. The booking is confirmed by the system, and the passenger receives a digital ticket.

Reservation Modification or Cancellation: Situations change, and a passenger's reservation must be modified or cancelled. They log in to their profile, examine their booking, and make any necessary adjustments, while receiving real-time updates on the reservation's status.

Efficient Flight Search: A traveller wants to arrange a vacation and is looking for flights that meet specified parameters such as date, time, source, and destination. The system allows them to rapidly identify available flights that fulfil these criteria, making the booking process easier.

Administrator Oversight: The technology supports administrators in efficiently managing passenger bookings. They can handle passenger information, update reservation statuses, and assure correct records, all of which contribute to an organised booking system.

## 1.3 Terms, Acronyms, and Abbreviations (Angelos Psillis)

**Terms:**

1. **Airline Management System:** A dedicated and customisable system for airlines that facilitates online flight booking and management.
2. **Passenger:** A person who uses the Airline Management System to book, view, or cancel flights.
3. **Administrator:** An authorised user in charge of managing passenger reservations and system updates.
4. **Flight Details**: Flight information such as origin, destination, date, time, class type, and so on.
5. **Reservation**: A passenger's confirmed reservation for a certain flight.
6. **Cancellation**: The procedure for cancelling a reserved seat on an aircraft.

**Acronyms and Abbreviations:**

1. **ID**: Identification.
2. **Class**: Flight class (e.g., Economy, Business, First).
3. **Dest**: Destination.
4. **Admin**: Administrator.
5. **Source**: Departure location
6. **DB**: Database.

# 2. PROJECT MANAGEMENT PLAN

## 2.1 Project Organization (Giorgos Pittis)

Our group has chosen to work together in an organized manner in order to effectively manage the Airline Management System development. For a well-coordinated effort, each team member has been given distinct duties and responsibilities. The project is organized around important stages, and we have followed a schedule in order to achieve our goals.

* Lead Developer(Coding and Implementation): (Martin)

---Responsible for coding the core functionalities of the Airline Management System.

---Ensures the system is robust, scalable, and meets the specified requirements.

---Collaborating with other team members to ensure seamless integration with the documentation and testing phases.

* Documentation Team:

---Documentation Lead(Giorgos) :

* Oversees the entire documentation process.
* Coordinates with team members to gather information for each section.
* Ensures the documentation aligns with the project's goals and requirements.

---Introduction(Aggelos, Nikolas):

* Outline the purpose and scope of the airline management system.
* Provide a comprehensive product overview highlighting key features and functionalities.
* Define terms, acronyms, and abbreviations for better understanding.

---Project Management Plan(Aggelos ,Nikolas ,Giorgos):

* Define the chosen lifecycle model and justify its selection.
* Conduct risk analysis and propose risk mitigation strategies.
* Specify hardware and software resource requirements.
* Develop a detailed schedule of deliverables and project milestones.
* Establish mechanisms for monitoring, reporting, and controlling project progress.
* Ensure adherence to professional standards.
* Implement configuration management for all project artifacts.

---Requirements Specification(Giorgos):

* Identify stakeholders for the airline management system.
* Develop the use case model and provide a graphical representation.
* Draft textual descriptions for each use case.
* Provide rationale for the selected use case model.
* Outline non-functional requirements.

---Architecture(Nikolas):

* Collaborate on architectural style/model selection.
* Document the technology, software, and hardware used in the system.
* Provide rationale for the chosen architectural style and model.

---Design(Aggelos, Giorgos):

* Contribute to GUI (Graphical User Interface) design documentation.
* Collaborate on static model creation, including class diagrams.
* Collaborate on dynamic model creation, such as sequence diagrams.
* Establish traceability from requirements to the detailed design model.

---Test Plan(Martin, Giorgos):

* Collaborate on the creation of requirements/specifications-based system level test cases.
* Ensure traceability of test cases to use cases.
* Contribute to the selection of techniques for test generation.
* Participate in the assessment of the quality of the test suite.

---Documentation(Giorgos, Nikolas , Aggelos, Martin):

* Generate the user manual for the airline management system.
* Develop the installation and deployment manual for the system.

**MAIN APPROACHES:**

* Agile Methodology:

Adopting an agile approach to ensure regular communication and flexibility in responding to changes.

* Daily Stand-up Meetings:

Conducting brief daily meetings to discuss progress, challenges, and plan the day's tasks.

* Collaborative Tools:

Utilizing collaboration tools (e.g., Git for version control, Slack for communication) to facilitate seamless coordination among team members.

* Parallel Development and Documentation:

Simultaneously working on coding and documentation to maintain a cohesive timeline and prevent delays.

* Iterative Development:

Implementing features iteratively, allowing for continuous testing and feedback.

**TIMELINE:**

Project Initiation (Week 1-2):

* Team familiarization with project requirements.
* Division of roles and responsibilities.
* Initial planning and discussions.

Coding and Implementation (Week 3-8):

* Lead Developer focuses on core functionalities.
* Regular updates and feedback loops with the documentation team to align with evolving features.

Documentation Phases (Week 3-10):

* Each documentation team member works on their designated phase.
* Collaboration sessions to ensure consistency across all sections.

Integration and Testing (Week 9-12):

* Integration of coded modules.
* Testing and debugging phases.

Finalization and Submission (Week 13):

* Final documentation review and edits.
* System demonstration and preparation for submission.

## 2.2Lifecycle Model Used (Giorgos Pittis)

* Lifecycle model used🡪 Agile Methodology  
  The reasons we used Agile methodology for the Airline Management system:

1.Flexibility and Adaptability:  
🡪 Dynamic requirements are part of the Airline Management System project, and they could change as it develops. Agile, with its iterative and incremental nature, allows us to easily adapt to changing customer needs and industry demands.

2.Continuous User Involvement:

🡪 Agile places a strong emphasis on regular stakeholder collaboration, including with end users. Because our system serves both administrators and passengers, it is essential to have ongoing user feedback in order to improve features and guarantee user satisfaction.

3.Iterative Development:

🡪 The iterative development approach in Agile enables us to break down the project into smaller, manageable increments. Each iteration results in a potentially shippable product increment, allowing for early delivery of valuable features.

4. Collaboration and Communication:

🡪 Agile promotes a collaborative environment where team members actively communicate and work together. This aligns with our team structure, ensuring efficient coordination between the coding and documentation teams for seamless integration.

5. Quick Response to Changes:

🡪 Agile allows requirements to be changed at any point during the development process. For a project like ours, where user preferences may change or the airline industry may experience regulatory changes, this is especially beneficial.

6. Transparent Progress Tracking:

🡪 Agile approaches, with their frequent sprint reviews and daily stand-up meetings, offer transparent insight into the status of projects. By being transparent, the team makes sure that everyone is aware of the status of the project and that any problems/errors can be quickly resolved.

7. Risk Mitigation:

🡪Agile allows for early identification and mitigation of risks through continuous testing and validation during each iteration. This proactive approach reduces the likelihood of late-stage project setbacks.

Why Agile Methodology has been selected:

🡪The Agile (Scrum) methodology has been selected for the Airline Management System project due to its inherent flexibility, iterative development approach, and emphasis on continuous collaboration. By adopting Agile, we aim to respond to evolving requirements, involve users throughout the development process, and deliver a high-quality system that meets both customer expectations and industry standards. This approach ensures a streamlined and adaptive development process, aligning with the project's dynamic nature and the need for consistent communication and collaboration within the team.

## A screenshot of a computer Description automatically generated2.3 Risk Analysis (Angelos Psillis)

Table 1 Risk Analysis

Figure 2 Risk Analysis

1. Risk type: Cybersecurity Risk

Risk Description: Passenger data is compromised as a result of a hacking attempt on the airline management system.

Affects: The breach exposes personal information, such as passport numbers and payment information, resulting in a loss of trust among passengers and significant legal ramifications.

Probability levels: Very high

Impact levels: Very high

Existing control measures: Implementing advanced encryption, conducting regular security audits, and teaching employees on cybersecurity protocols are all examples of cybersecurity best practices.

Prevention measures: Regularly updating security systems, implementing two-factor authentication, and educating people on safe online practices.

1. Risk type: Reputation Risk

Risk Description: Service interruptions caused by system flaws result in unfavourable social media reviews from travellers.

Affects: Public discontent develops, resulting in a drop in bookings and ruining the airline's reputation.

Probability levels: High

Impact levels: High

Existing control measures: Implementing proactive communication techniques, providing compensation or alternative options, and responding quickly to client complaints.

Prevention measures: Maintaining the system on a regular basis, responding quickly to difficulties, and regularly monitoring and connecting with customers on social media sites.

1. Risk type: Technology Obsolescence Risk

Risk Description: The airline management system is based on obsolete software that is incompatible with contemporary security standards.

Affects: The system becomes subject to security risks, and its functionality falls behind that of competitors.

Probability levels: Moderate

Impact levels: High

Existing control measures: Planning system improvements, developing scalable technology, and completing future-proofing evaluations are all important steps.

Prevention measures: Updating software components on a regular basis, maintaining updated about industry changes, and providing tools to facilitate timely upgrades.

1. Risk type: Scope Creep Risk

Risk Description: Additional features are constantly sought by stakeholders during the system development process, with no clear evaluation or oversight.

Affects: Project schedules are extended, resources are stretched, and the system's basic objectives are spaced, impacting the project's overall efficiency.

Probability levels: Low

Impact levels: Moderate

Existing control measures: Creating a change control board to analyse and authorise scope changes, conducting regular scope evaluations, and keeping clear scope documentation are all important steps.

Prevention measures: Defining a detailed project scope from the beginning, receiving stakeholder approval, and enforcing strict adherence to the established scope throughout the project lifetime.

1. Risk type: Financial Risk

Risk Description: Unexpected inflation raises operating expenses for maintaining servers and system infrastructure significantly.

Affects: Budget overruns put financial resources under strain, limiting funding for system improvements or upgrades.

Probability levels: Low

Impact levels: High

Existing control measures: Thorough financial planning, the establishment of a contingency fund, and the implementation of frequent financial audits.

Prevention measures: Analysing cost trends, negotiating long-term service contracts to offset cost volatility, and doing proactive cost-benefit assessments prior to infrastructure improvements are all examples of proactive cost-benefit studies.

1. Risk type: Operational Risk

Risk Description: Because of insufficient redundancy in the system infrastructure, a catastrophic server failure occurs during peak booking hours.

Affects: The system faces downtime, causing flight bookings to be disrupted and passengers to be inconvenienced.

Probability levels: Moderate

Impact levels: High

Existing control measures: Putting in redundant systems, doing regular system health checks, and putting in failover procedures.

Prevention measures: Ensure a strong system design with suitable backup and failover options, and schedule maintenance during low-traffic periods to avoid disruptions.

1. Risk type: Natural Disaster Risk

Risk Description: A major weather event, such as a tropical storm or an earthquake, causes damage to the data centre that houses the airline management system.

Affects: Downtime of the system, loss of crucial data, delay in flight bookings, and potential financial losses.

Probability levels: Very low

Impact levels: High

Existing control measures: Putting in place geographically dispersed backup centres, disaster recovery strategies, and regular data backups.

Prevention measures: Investing in resilient infrastructure, placing data centres in less vulnerable places, and establishing robust disaster recovery methods are all priorities.

1. Risk type: Economic Downturn Risk

Risk Description: A substantial economic slump reduces travel demand, which has an impact on the airline's revenue and operational budgets.

Affects: Budget cuts for system upgrades, fewer resources for system improvements, and probable delays in technical developments are all possibilities.

Probability levels: Moderate

Impact levels: High

Existing control measures: Conducting financial stress tests, developing contingency budgets, and prioritising key system enhancements are all part of the job.

Prevention measures: Diversifying revenue streams, introducing cost-cutting initiatives, and anticipating potential economic situations for proactive planning are all examples of proactive planning.

1. Risk type: Competitive Disruption Risk

Risk Description: A competitor launches a more sophisticated and user-friendly booking system, luring customers away from the airline's system.

Affects: Reduced market share, bookings, and potential loss of client loyalty.

Probability levels: High

Impact levels: High

Existing control measures: Conducting market research, investing in continual system enhancements, and improving user experience are all examples of activities that fall under this category.

Prevention measures: To remain competitive, keep up with market trends, upgrade the system on a regular basis based on user input, and introduce novel features.

## 2.4 Hardware and Software Resource Requirements (Angelos Psillis)

Hardware Requirements:

1.Server Infrastructure:

* Servers: High-performance servers for system hosting, database management, and handling concurrent user requests.
* Storage: Storage capacity sufficient to manage passenger data, flight information, and system backups.
* Redundancy: Redundant systems improve continuity and reduce downtime in the event of hardware breakdown.

2.Network Infrastructure:

* Networking Equipment: Reliable networking hardware (routers, switches) to provide continuous communication between servers and users.
* Bandwidth: High-speed internet connectivity to support concurrent user access and data transfers.

3. Client Devices:

* Desktops/Laptops: Standard computing devices for administrative personnel who need to access the system interface.
* Mobile Devices: Compatibility with a wide range of mobile devices for passengers using the system for booking and information.

4. Scalability

* Assuring that the server architecture can scale, allowing for greater user traffic at peak times without impacting system performance.

5. Redundancy and Reliability

* Adding redundancy to the server architecture to enhance service dependability and continuity while minimising downtime in the event of hardware breakdown.

Software Requirements:

1.Operating Systems:

* Server OS: For hosting the system, compatible server operating systems (e.g., Windows Server, Linux) must be used.
* Client OS: Compatibility with popular operating systems (Windows, macOS, iOS, and Android) for user convenience.

2. Database Management:

* Database Software: Database management systems (e.g., MySQL, PostgreSQL) that are capable of storing and managing information about passengers, flight information, and reservations.

3. System Software:

* Security Software: To protect against cyber dangers, use antivirus, firewalls, and intrusion detection systems.
* Backup Systems: Backup systems that automatically secure and restore system data on a regular basis.

4. Web/Application Servers:

* Web Server Software: Appropriate web server applications (e.g., Apache, Nginx) for serving web pages and managing online reservations.
* Application Framework: Application frameworks are used to create and manage the system interface.

5. Development Tools:

* Programming Languages: Programming languages (such as Java, Python, or PHP) are required for system development and maintenance.
* Integrated Development Environment (IDE): Development environments for system coding, testing, and debugging.

6. Performance Optimization (Hardware and Software):

* Optimising software components to reduce resource utilisation and improve system performance, resulting in faster response times for consumers.

## 2.5 Deliverables and Schedule (Angelos Psillis)

A screenshot of a project

Description automatically generated

Figure 3 Gantt chart – Deliverables and Schedule

## 2.6 Monitoring, Reporting, and Controlling Mechanisms (Nikolas Filippou)

In our software development effort for the Airline Management System, we employ robust tools and methodologies for effective monitoring, reporting, and control. The use of GitHub and GitKraken serves as the foundation for streamlined collaboration and issue management throughout the project lifecycle.

**1. GitHub:**

Version Control and Collaboration:GitHub is utilized for version control, enabling us to track changes, manage branches, and facilitate collaborative development. It ensures that our codebase remains organized and that team members can seamlessly work on different aspects of the project simultaneously.

**Issue Tracking:**

GitHub's issue tracking system allows us to log, prioritize, and assign tasks or bugs. This feature enables a transparent and structured approach to issue resolution, providing a centralized platform for communication and updates related to project tasks.

Pull Requests and Code Review:Pull requests on GitHub are instrumental in reviewing and merging code changes. This process ensures code quality, adherence to coding standards, and knowledge sharing among team members.

**2. GitKraken:**

**Visual Version Control:**

GitKraken provides a visual representation of our Git repositories, making it easier for developers to understand and manage version control. The visual interface simplifies complex branching and merging operations, enhancing the overall efficiency of our development process.

**Graphical Interface for Git Commands:** GitKraken's graphical interface allows developers to execute Git commands intuitively. This minimizes the learning curve for team members and reduces the likelihood of errors in version control operations.

Monitoring, Reporting, and Controlling Methodologies:

1**. Regular Code Reviews**:Regular code reviews are conducted using GitHub pull requests. This practice ensures that code changes align with coding standards, are well-documented, and have undergone scrutiny from team members. Code reviews contribute to code quality and knowledge sharing within the team.

2**. Continuous Integration (CI) with GitHub Actions:**GitHub Actions are employed for continuous integration, automating the process of building, testing, and validating code changes. This ensures that the codebase remains stable, reducing the likelihood of integration issues.

3. **Agile Methodology:**We follow the Agile methodology, incorporating sprints and regular sprint reviews. This allows for adaptive planning, continuous improvement, and frequent deliveries of functional increments. Agile practices enhance collaboration, responsiveness to changes, and early issue identification.

4**. Real-time Communication:**Utilizing GitHub issues and GitKraken's collaboration features, our team engages in real-time communication. This ensures prompt issue resolution, facilitates discussions, and keeps all team members informed about the project's progress.

**Mitigation of Issues:1. Proactive Issue Management:**GitHub issues serve as a proactive mechanism for identifying and managing problems. The team collaboratively addresses issues, assigns tasks, and monitors progress to ensure timely resolution.

**2. Continuous Feedback Loop:**Regular code reviews, sprint reviews, and continuous integration practices create a continuous feedback loop. This loop allows for the prompt identification of issues and facilitates immediate corrective actions, preventing the escalation of problems.

**3. Iterative Development:** The iterative development approach of Agile allows for incremental enhancements and adjustments. This flexibility enables the team to adapt to changing requirements and address issues in subsequent iterations.

Our choice of GitHub and GitKraken, combined with Agile practices and continuous integration, establishes a robust framework for monitoring, reporting, and controlling our software development effort. These tools and methodologies not only enhance collaboration but also provide effective mechanisms for issue mitigation and the overall success of the project.

## 2.7. Professional Standards (Nikolas Filippou)

During the development of the Airline Management System, we distinguished set of professional standards, with a primary focus on IEEE Software Engineering Standards that are recognized for their comprehensive guidance, these standards serve as the cornerstone of our software development processes, ensuring exceptional quality in all facets of our work.

While acknowledging the significance of other existing professional standards such as:

**Code of Ethics and Professional Conduct:** Guiding our team with ethical principles, emphasizing integrity, honesty, and user welfare in all software-related endeavours.

**ISO/IEC 12207: Software Life Cycle Processes:** An international standard meticulously defining processes across the software development life cycle, fostering a systematic and structured approach to our work.

**OWASP Guidelines:** Followed to fortify the security and robustness of our web applications based on guidelines provided by the Open Web Application Security Project.

**Accessibility Standards (e.g., WCAG):** Ensuring inclusivity and an enhanced user experience by aligning with accessibility standards such as WCAG.

**Continuous Learning and Professional Development:** Encouraging continuous learning within our work culture, empowering our team to enhance their skills and stay at the forefront of industry advancements.

**Why IEEE Software Engineering Standards?**

Our choice to primarily embrace IEEE Software Engineering Standards is rooted in specific advantages:

1. **Emphasis on Quality Management**: IEEE standards ensure an unwavering focus on delivering an Airline Management System of superior quality, aligned seamlessly with industry expectations.
2. **Standardization of Processes:** IEEE standards provide a structured framework, fostering consistency and predictability throughout our software development activities.
3. **Customer-Centric Approach:** These standards reinforce our dedication to understanding and meeting customer requirements, placing their needs at the core of our decision-making processes.
4. **Continuous Improvement:** Rooted in the principle of continual enhancement, IEEE standards empower us to adapt, evolve, and consistently deliver successful projects.
5. **Systematic Documentation and Record Keeping:** Adherence to IEEE standards ensures rigorous documentation, promoting transparency, traceability, and accountability – critical elements in our commitment to delivering reliable solutions.
6. **Proactive Risk Management:** IEEE standards empower our team to identify and address potential risks proactively, enhancing our project's resilience.

## 2.8 Evidence all the artifacts have been placed under configuration management.

# 3. Requirement Specifications

## 3.1 Stakeholders for the system (Giorgos Pittis)

Passengers: These are the primary users of the system. Passengers use the system to search for flights, book tickets, check flight details, and manage their reservations.

Airline Staff: This includes the airline's employees who use the system to manage flight schedules, ticketing, and passenger information. They may include reservation agents, ticketing agents, and flight attendants.

Administrators: System administrators are responsible for the maintenance, security, and overall management of the system. They handle user accounts, database management, and ensure the system is functioning correctly.

Airline Management: This group includes higher-level executives and managers who use the system for decision-making, such as analyzing booking trends, monitoring revenue, and managing flight schedules.

IT Department: The IT department is responsible for the technical aspects of the system, including software development, server maintenance, and system updates.

Regulatory Authorities: Airlines are subject to various regulations and may need to provide information to regulatory authorities. These authorities may need access to the system for oversight and compliance.

Third-Party Vendors: If the airline management system integrates with third-party services (e.g., payment processors, booking engines, or data providers), these vendors are stakeholders as their services affect the system's functionality.

Marketing and Sales Teams: These teams use the system to promote flight offers, manage discounts, and analyze customer behavior to tailor marketing strategies.

Travel Agencies: If the airline collaborates with travel agencies, they might have access to the system to book flights on behalf of their clients.

Passenger Support and Customer Service: These teams use the system to assist passengers with booking, cancellations, and addressing issues or complaints.

Security Auditors: In the context of data security and compliance, security auditors may periodically assess the system to ensure it meets industry standards and is protected against security breaches.

Financial Departments: The finance team uses the system to track revenue, manage billing, and ensure financial transactions are accurate.

Competitors: Competing airlines may monitor the system to stay informed about pricing, flight schedules, and other competitive factors.

Investors and Shareholders: Individuals or organizations who have invested in the airline may be interested in how the system impacts the company's financial performance.

Government and Law Enforcement: In cases of security or legal issues, government agencies and law enforcement may need access to the system's data and records.

## 3.2 Use Case Model and Textual Description (Giorgos Pittis)

For the Airline Management System there are primarily two Users (Actors), the Passenger, which can search for flights etc. and the Administrator which manages the flights and Bookings etc. For each one we have different Use Cases :

**Passenger Use Cases:**

* Search Flights:

--As a passenger, i can search for available flights based on criteria such as departure city, destination, date, and other preferences, so I can book a flight.

--Textual Description: The passenger is provided with the search criteria (textbox, calendar etc.) and after inserting his/her preferences , the system shows a list of available flights based on the criteria and the passenger can continue to the next stage.

--Breakdown:

* As a passenger I can enter my desired departure and arrival destinations by typing in the corresponding box
* As a passenger I can enter the date that I want to fly , via clicking on the desired date in the calendar picker.
* A diagram of a flight schedule

  Description automatically generated with medium confidenceAs a passenger I will be able to see a list of all the available flights based on the search terms provided.

Figure 4 Passenger Use Case Diagram on how to search flights.

* Book Flights:

--As a passenger, I can book a flight by selecting a flight from the list, after providing my personal information and payment details, so the system can confirm the booking and issue a ticket.

--Textual Description: The passenger selects a flight from the list, insert its personal information and payment details, the system retrieves the information and books the flight , issue a ticket and send feedback later on.

--Breakdown:

* As a passenger I will retrieve a list of flights based on the previous information that I provided.
* As a passenger I will have to select a flight based on my personal preferences and click on it.
* As a passenger I have to insert any personal information and payment details that the system requires to process.
* As a passenger I can click on the button saying “Book Flight” and the system will retrieve the information issue a ticket and send feedback later on.

A diagram of a flight

Description automatically generated

Figure 5 Passenger Use Case Diagram on how to book flights.

* Cancel Reservation/Flight:

--As a passenger ,I can cancel an existing flight reservation by logging into my account, selecting the menu and then flights to cancel , so that the system will cancel the reservation and process any refunds.

--Textual Description: The passenger will log in via its account ,select the menu tab/button , click on flights to cancel , find the flight he/she wants to cancel click on it and proceed.

--Breakdown:

* As a passenger I will be able to find the menu tab/button on the top of the screen and click on it.
* As a passenger I will be able to see the button for the cancellation of the flights and click on it to proceed.
* As a passenger I will be provided with a list of my booked flights.
* As a passenger I will have to click on the flight I want and cancel it by clicking on the button cancel flight.
* The system will get notified and proceed with the cancellation , with returning refunds and feedback information.

A diagram of a flight

Description automatically generated

Figure 6 Passenger Use Case Diagram for cancelling flights.

* Login/Signup:

--As a passenger, I can log in or sign up to the system , so that I can continue to search/book/cancel flights in the system.

--Textual Description: The passenger will select the menu button on the top of the page, he/she will have choices of Log in/Sign up, depends on the current situation of the passenger he/she will choose one of the two, then the passenger will have to provide any personal information required for the system to process.

--Breakdown:

* As a passenger when I am at the home screen I will be able to see the menu tab/button on the top of the screen and click on it.
* As a passenger after clicking on the menu button I will be able to see the Log In/Sign up button ,click on it.
* As a passenger I will be provided with an interface with multiple textboxes that I must fill up.
* As a passenger before I fill up the textboxes , based on my current situation I have to press on the top , either “Log in” if I already have an account, but if not I will have to press “Sign up” to create an account.
* As a passenger then I fill up any personal information that the system requires to proceed and then press Log in/Sign up.A diagram of a login system

  Description automatically generated

Figure 7 Use Case Diagram on how to login/sign up as a passenger.

* Edit Profile:

--As a passenger , I can edit my profile changing passwords/emails etc., so I can improve the security safety of my account( Password, Email ,Username/Surname etc.)

--Textual Description: The passenger can change the password for security reasons, the Email, Username/Surname for personal reasons and generally change personal information for its safety.

--Breakdown:

* As a passenger I will have to press on the menu button on the top of the screen.
* As a passenger I will be able to see the button to change any personal information and click on it .
* As a passenger I will be provided with a list of my personal information in textboxes.
* As a passenger by pressing on the textboxes I can edit any personal information I want(Passwords, Emails, Username, Surname etc.)
* As a passenger after any changes I made I will have to press on the button save changes so don’t lose any change I made.

A diagram with text and images

Description automatically generated with medium confidence

Figure 8 Use Case Diagram on how to edit your profile as a passenger.

**Administrator Use Cases:**

* Log in as an Administrator:

--As an Administrator, I can login into the system with a different objective than the passenger, so I can manage bookings , manage passengers status, cancel flights , change flights etc.

--Textual Description: The Administrator , when he/she is on the home screen he will have to press on the menu button on the top of the screen , then press Log in/Sign up , then there will be an option available to log in as an administrator and after clicking on it two textboxes will appear , one for the Username and one for the password and log in, this information only the administrator will be able to know.(Username, Password).

--Breakdown:

* As an administrator I will have to press on the menu tab/button on the top of the screen.
* As an administrator I will have to press on the Log in/Sign Up button to proceed.
* As an administrator later on I will be able to press on the button saying log in as an administrator and click on it.
* As an administrator I will be provided with two textboxes, one for the Username and one for the Password.
* As an administrator I will have to insert he corresponding details that have given to me and press log in.

A diagram of a log in

Description automatically generated

Figure 9 Use Case Diagram on how to login as an administrator.

* Manage Bookings:

--As an administrator, I can manage passenger bookings and reservation statuses by logging into the system, so I can later on delete , add or modify any flight.

--Textual Description: The administrator will press the menu button on the top of the screen , then press the button manage bookings and he/she will be provided with the list of all flights, so if any flight has been cancelled or its hour has changed , or a new one is announced , he can proceed to delete , add or modify later on by clicking on the buttons, delete a flight, add a flight, update a flight etc.

--Breakdown:

* As an administrator I will press the menu button , and then the manage flights button.
* As an administrator I will be provided with the list of all flights available.
* As an administrator I will have to press on the button delete a flight , if a flight has been cancelled
* As an administrator I will have to press on the button add a flight, if a new flight is announced
* As an administrator I will have to press on the button update flight, if I want to change any flight information such as destinations, timelines, stops etc.
* As an administrator after modifying any flight I have to press save changes so that any process I made will be saved.

A diagram of a button

Description automatically generated

Figure 10 Administrator Use Case Diagram for managing flights.

* Update Flight Information:

-- As an administrator, I can choose to update flight details, such as schedules, timelines, add/remove stops, availability, and pricing for passengers, so I can prevent any mistakes.

--Textual Description: The administrator will press on the menu button on the top of the screen then press on the manage flights button and he/she will be provided with all the available flights. A button will be available to press that is saying update flight on each flight pressing the one he/she wants to modify, from there the administrator can see timelines to change, add/remove stops, change pricing, change destinations etc.

--Breakdown:

* As an administrator I will press on the menu button and then the manage flights button.
* As an administrator I will see all the flights available.
* As an administrator I will be able to see the button for the update of flights, on each flight.
* As an administrator I will choose the flight I want to modify
* As an administrator I will see all the requirements I can change for the flight (Prices, Timelines, Add/remove stops, change destinations etc.) and proceed to modify.

A diagram of a flight

Description automatically generated

Figure 11 Administrator Use Case Diagram on how to update flights.

* Manage User-Passenger Accounts:

-- As an administrator, I can manage passenger user accounts, so I can change its personal information for security reasons or block a passenger if necessary (e.g., reset passwords, block/delete/add accounts etc.).

--Textual Description: The administrator will press on the menu button, then find the button saying manage passenger accounts and click on it and all the accounts registered will be shown in a list, and from there he/she can select a profile by clicking on it and delete or change any information.

--Breakdown:

* As an administrator I will press the menu button and then the manage passenger accounts button.
* As an administrator I will see a list with all the registered members of our system.
* As an administrator I will select a passenger that I want to delete, and press delete passenger.
* As an administrator I will select a passenger that I want to reset its password, and press reset password.

.

A diagram of a flowchart

Description automatically generated

Figure 12 Administrator Use Case Diagram on how to manage passenger account.

### A screenshot of a computer Description automatically generated3.2.1 Graphic Use Case Model (Giorgos Pittis)

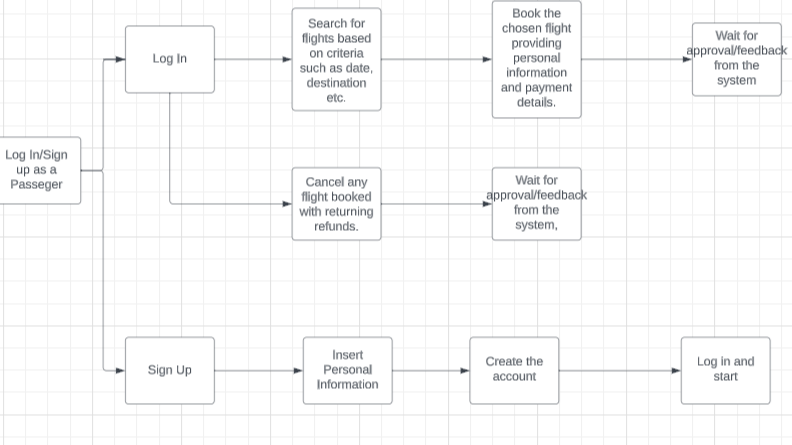
****Figure 13 Generic Use Case Diagram of the Administrator.

Figure 14 Generic Use Case Diagram of the Passenger.

## 3.3 Rationale for Use Case (Giorgos Pittis)

* User-Centered Design:

The use case model is designed with a focus on the primary system users: passengers and administrators. Passengers can efficiently book flights, view flight availability, and cancel reservations, which enhances the user experience and reduces the need for offline booking counters.

* Efficient Booking Process:

Passengers can book flights online, reducing the need for physical visits to booking counters. This is not only convenient for passengers but also cost-effective for airlines, as it streamlines the booking process.

* Flexibility and Accessibility:

The system is designed to be accessible from any location at any time, which aligns with modern passenger expectations. Passengers can view flight availability and make bookings whenever they want, enhancing the flexibility of the system.

* Administrator Control:

The "Manage Booking " use case provides administrators with control over the system's core functionalities, such as flight availability and reservation management. This enables airlines to adapt to changing circumstances and efficiently allocate resources.

* Feedback Loop:

While not explicitly shown in the model, the system can incorporate a feedback loop where passengers can provide feedback on their experiences. This information can be used to continuously improve the system and the overall quality of service.

* Cost Savings and Operational Efficiency:

By reducing the need for offline booking counters, the system contributes to cost savings and enhances operational efficiency for the airline.

* Scalability:

The use case model is adaptable and scalable. As the airline business evolves and grows, additional use cases can be integrated into the system to meet new requirements without significantly altering the existing structure.

## 3.4 Non-Functional Requirements(Giorgos Pittis)

Performance:

* The system must support a minimum of 1000 concurrent users during peak hours.
* Response time for flight availability queries should be under 2 seconds.
* The system should handle at least 99% of transactions without errors.

Availability:

* The system should be available 24/7, with scheduled maintenance periods minimized.
* Availability should be at least 99.9%.

Reliability:

* The system should have a mean time between failures (MTBF) of at least 10,000 hours.
* In case of system failure, data integrity must be maintained, and transactions in progress should not be lost.

Security:

* User data and financial information must be encrypted and stored securely.
* Authentication and authorization mechanisms should be in place to prevent unauthorized access.
* The system should comply with industry security standards and regulations (e.g., GDPR, PCI DSS).

Scalability:

* The system must be scalable to accommodate increasing numbers of users and flights.
* Scalability should be achieved both horizontally and vertically.

Usability:

* The user interface should be intuitive and user-friendly to cater to users with various levels of technical expertise.
* Accessibility standards should be followed to ensure that the system can be used by individuals with disabilities.

Compatibility:

* The system should be compatible with a wide range of web browsers and devices.
* It should work on different operating systems.

Maintainability:

* The system should be easily maintainable, with the ability to update and patch software without significant downtime.
* Documentation for administrators and developers should be comprehensive.

Compliance:

* The system must adhere to aviation industry regulations and standards.
* It should also comply with data protection and privacy laws, such as GDPR.

Auditability:

* There should be a robust logging and audit trail system to track all user and administrator activities.
* The logs should be securely stored and tamper-evident.

Interoperability:

* The system should integrate with other airline systems, such as reservation systems, payment gateways, and flight tracking systems.

Data Backup and Recovery:

* Regular data backups must be performed, and there should be a disaster recovery plan in place to restore the system in case of data loss or catastrophic events.

Load Testing:

* Regular load testing should be conducted to ensure the system can handle the expected volume of transactions and users.

Geographic and Time Zone Considerations:

* The system should support different time zones, especially for international flights.
* It should provide localized content and language options for users in different regions.

Cost-effectiveness:

* The system should be cost-effective to develop, maintain, and operate, without incurring unnecessary expenses.

# 4. Architecture

## 4.1) Architectural style(s)/model used: (Nikolas Filippou)

The architectural framework serves as more than just the operational structure of our software. It acts as a representation that empowers software engineers to achieve several key objectives:

**Analysis of Design Effectiveness:** It enables an evaluation of how the design corresponds to its specified requirements. This assessment is vital to ensure that the intended objectives are achieved efficiently.

**Consideration of Alternatives:** By providing a clear architectural representation, it enables engineers to explore alternative design options at a stage where implementing changes is still feasible. This flexibility contributes to making informed decisions in the design process.

**Risk Reduction:** The architecture plays a pivotal role in mitigating risks associated with software construction. Identifying potential challenges and issues early in the design phase allows for proactive measures to minimize risks during implementation.

**Why Architecture Matters a lot in our project:**

**Facilitates Communication**: Architectural representations serve as a means of communication among all stakeholders involved in the development of a computer-based system. This ensures a shared understanding among team members, fostering effective collaboration.

**Highlights Early Design Decisions**: The architectural framework brings attention to critical design decisions made early in the process. These decisions significantly influence subsequent software engineering work and ultimately impact the success of the system as an operational entity.

**Intellectually Graspable Structure:** Architecture provides a concise and comprehensible mode of understanding how the system is structured and how its components collaborate. This clarity aids in maintaining a unified vision among the development team.

Architecture plays a role, in our software beyond being a blueprint. It serves as a way to communicate guide decision making and manage risks. Architecture provides a model that shapes the structure and collaboration of system components.

In software engineering, various architectural styles/models exist, each suited for specific applications and requirements. Common architectural styles include:

* Client-Server Architecture
* Microservices Architecture
* Monolithic Architecture
* Service-Oriented Architecture (SOA)
* Event-Driven Architecture
* Layered Architecture
* Component-Based Architecture
* Peer-to-Peer Architecture

For the Airline Management System, we have opted for the Client-Server Architecture. This choice is rooted in the unique characteristics and requirements of the airline industry, as well as the specific needs of our software system.

**Why We Chose Client-Server Architecture:**

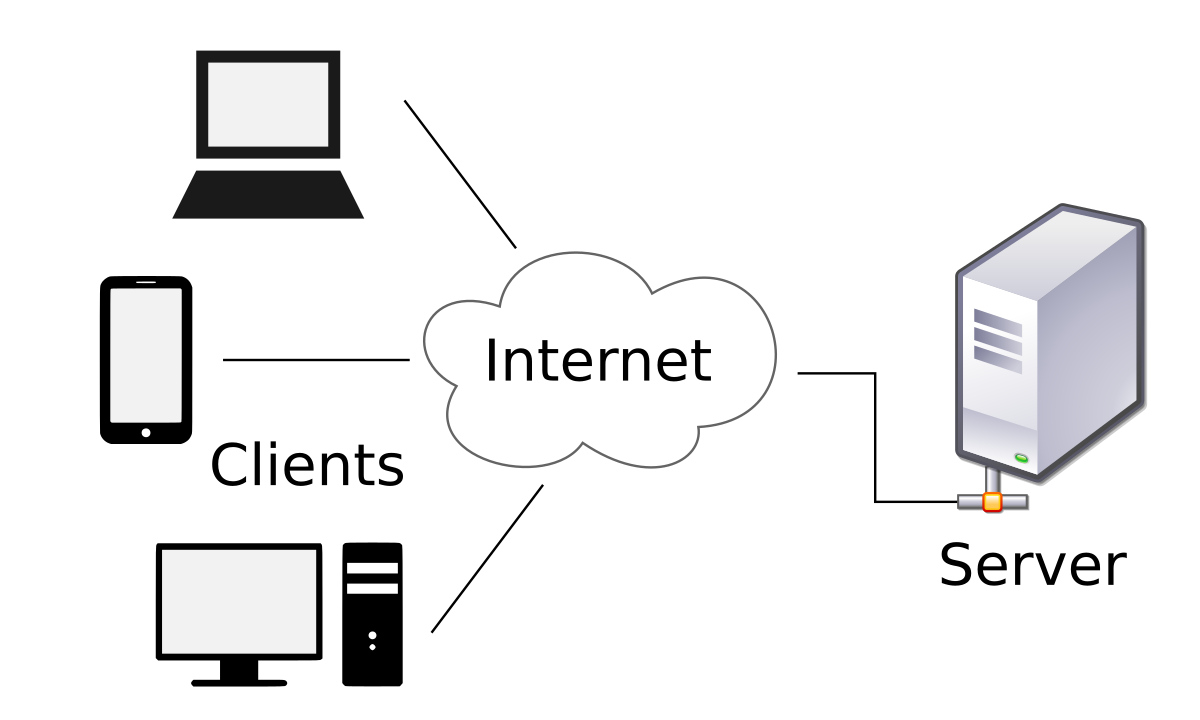


Figure Client-Server

1. **Scalability**: Client-Server Architecture is inherently scalable, making it well-suited for systems with potentially large numbers of users, such as an airline management system. The division of functionality between clients and servers allows for easier scaling of components based on demand.
2. **Centralized Control and Management:** In the airline industry, centralized control and management are critical for overseeing reservations, flight schedules, and overall system functionality. The server component acts as the central point of authority, ensuring consistency and coherence in data management.
3. **Efficient Resource Utilization:** With a dedicated server managing core functionalities, clients (passenger interfaces) can be relatively lightweight. This ensures efficient resource utilization, especially for end-users accessing the system from various devices and locations.
4. **Enhanced Security:** Client-Server Architecture facilitates centralized security measures, allowing for robust protection of sensitive data. Security protocols and access controls can be implemented at the server level, providing a secure environment for both passenger information and administrative controls.
5. **Separation of Concerns**: The architectural style enables a clear separation of concerns between the client-side (user interface and user interactions) and the server-side (business logic, data management). This modularity enhances maintainability and allows for easier updates or modifications to specific components without affecting the entire system.
6. **Flexibility and Interoperability:** Client-Server Architecture promotes flexibility and interoperability. Clients with different interfaces (web, mobile applications) can interact with the server using standardized protocols, providing adaptability to various user needs and preferences.
7. **Improved Performance:** The division of tasks between clients and servers contributes to improved performance. Client devices can focus on rendering user interfaces and handling user input, while the server manages the processing of complex business logic and database interactions.
8. **Adaptability to Distributed Environments:** Client-Server Architecture is well-suited for distributed environments, accommodating scenarios where clients and servers may be located in different geographical locations. This is particularly relevant for an airline management system accessed by passengers and administrators worldwide.

In conclusion, the decision to employ Client-Server Architecture for the Airline Management System is rooted in its scalability, centralized control, security benefits, separation of concerns, flexibility, and adaptability to the distributed nature of the airline industry. This architectural style aligns with the specific requirements of managing flight reservations, passenger interactions, and administrative controls within the context of a modern and efficient airline management system.

## 4.2)Technology, software, and hardware used (Nikolas Filippou)

**Backend:**

**Database - SQLite:**SQLite is employed as the backend database for its lightweight nature and efficiency, making it suitable for our Airline Management System. It's well-suited for projects with moderate data storage needs, providing simplicity without compromising on functionality. The relational structure of SQLite aligns with our data requirements, facilitating seamless data management.

API - Python + Flask:

Python, coupled with Flask, serves as the backend API framework. Python's readability and versatility make it an ideal choice for rapid development, while Flask provides a lightweight and flexible framework for building APIs. The combination offers an efficient and straightforward way to handle backend logic and communicate with the frontend.

**Frontend:**

**HTML:**HTML forms the backbone of our frontend, providing the essential structure for web page development. Its simplicity and compatibility make it a fundamental building block for creating user interfaces in a standardized manner.

**Bootstrap:**Bootstrap is utilized for frontend styling and layout. Its responsive design and pre-built components expedite the development process, ensuring a visually appealing and consistent user experience across various devices. Bootstrap's grid system and components enhance the overall aesthetics of the Airline Management System.

**General Technologies:**

**Version Control - Git:**

Git is employed for version control, enabling collaborative development and easy tracking of changes. Its branching and merging capabilities support efficient collaboration among team members working on different aspects of the project.

**Web Development Framework - Flask (Backend):**

Flask, a lightweight web framework for Python, facilitates rapid development of the backend API. Its simplicity, extensibility, and compatibility with Python align with the project's requirements for creating scalable and efficient backend services.

Integrated Development Environment (IDE) - VS Code:

Visual Studio Code is chosen as the primary IDE for its lightweight nature, extensive plugin support, and user-friendly interface. It enhances the development workflow for both frontend and backend tasks.

## 4.3) Rationale for Choosing Client-Server Architecture: (Nikolas Filippou)

The decision to use Client Server Architecture for our Airline Management System is based on the requirements and characteristics of our project. This architectural option is strategically aligned with the nature of the airline industry. Fulfills the needs of our software solution.

1. **Scalability for Dynamic Demand**:

Rationale: The airline industry experiences dynamic fluctuations in demand, especially during peak travel times. The Client-Server Architecture is inherently scalable, allowing our system to efficiently handle varying levels of user interactions. This adaptability is crucial for ensuring optimal performance during periods of high booking activity.

**2. Centralized Control and Management:**

Rationale: Airline operations demand centralized control and management of critical information, such as flight schedules, reservations, and passenger details. The server component in Client-Server Architecture serves as the central authority, facilitating streamlined data management and consistent decision-making. This centralized control is essential for the smooth functioning of our Airline Management System.

**3. Security for Sensitive Data:**

Rationale: Security is paramount in the airline industry, given the sensitive nature of passenger information and financial transactions. The Client-Server model allows for the implementation of robust security measures at the server level, safeguarding passenger details and ensuring secure financial transactions. This approach aligns with the high-security standards required in the aviation sector.

**4. Separation of Concerns for Modularity:**

Rationale: The separation of concerns in Client-Server Architecture enhances modularity and maintainability. By delineating user interfaces (client-side) from business logic and data management (server-side), we create a modular system that facilitates easier updates, modifications, and maintenance. This modularity is crucial for long-term sustainability and adaptability to evolving requirements.

**5. Adaptability to Distributed Environments:**

Rationale: The airline industry involves a geographically dispersed user base, including passengers and administrators located worldwide. Client-Server Architecture accommodates this distributed nature, allowing clients to interact with the server seamlessly irrespective of their geographical location. Standardized communication protocols ensure interoperability and adaptability to diverse environments, aligning with the global reach of our airline services.

**6. Efficient Resource Utilization:**

Rationale: Client-Server Architecture optimizes resource utilization by assigning specific roles to clients and servers. Clients focus on rendering user interfaces and handling user input, while the server manages resource-intensive tasks such as database interactions and business logic. This allocation ensures efficient use of resources, contributing to the overall performance and responsiveness of our Airline Management System.

To summarize we have chosen the Client Server Architecture for our Airline Management System based on factors. These include its ability to scale effectively provide control offer security features promote modularity adapt well to distributed environments and optimize resource usage. This architectural decision has been specifically designed to meet the needs and demands of the airline industry. By selecting this approach we can ensure that our system is reliable, secure and able to deliver an experience, for both passengers and administrators.

# 5. DESIGN

## 5.1.GUI (Giorgos Pittis)

## A screenshot of a computer Description automatically generated5.2. Static model – class diagram (Giorgos Pittis)

Figure 16 Static model - class diagram of the airline management system

## 5.3.Dynamic model – sequence diagram (Aggelos Psillis)

A diagram with blue squares and black text

Description automatically generated

Figure 17 Sequence Diagram of the administrator.

A diagram of a company

Description automatically generated

Figure 18 Sequence Diagram of the user.

## 5.4.Rationale of the detailed designs above (Giorgos Pittis and Angelos Psillis)

Static Class Diagram Rationale: (Giorgos Pittis)

-Passenger Class:

* Represents passengers in the system.
* Contains attributes such as passengerID, name, and contactInfo to store passenger information.
* Has a one-to-many relationship with reservations, allowing passengers to have multiple reservations.

-Administrator Class:

* Represents system administrators.
* Contains attributes such as adminID, username, and password to manage administrator accounts.
* Includes operations for managing passengers and flights, as administrators need to have control over these aspects.

-Flight Class:

* Represents flight information.
* Contains attributes such as flightID, airline, origin, destination, and schedule details.
* Has a one-to-many relationship with reservations, allowing multiple reservations for a single flight.

-Reservation Class:

* Represents passenger reservations.
* Contains attributes like reservationID, passenger, flight, status, and bookingTime.
* Relates passengers to their reservations and flight bookings.

Sequence diagram Rationale: (Aggelos Psillis)

User:

* Users provide their information to log in.
* The login page communicates user-entered information to the verification page.
* The verification page verifies the information from database page.
* Verification page communicates the validation status to the login page.
* User interaction with the authenticated system.
* User can edit profile and the database page verifies it.
* User can request to view available flights as per his requirements.
* Database shows the requirements, and it displays them to the user.
* User can choose and request for reservation from database.
* After confirmation from database passenger can see the status
* Then he/her might cancel the ticket.

Admin:

* Admins provide their information to log in.
* Admins can change/edit flights, passengers, tickets, and reservations.
* Admins can update the passenger status.

## 5.5. Traceability from requirements to detailed design model (Giorgos Pittis)

1.Requirement Identification:

* Identify and list the key requirements outlined in the Airline Management System description provided.
* Clearly specify the user and system inputs, operations, and expected outputs.

2. Administrator Module Design:

* Based on the administrator operations outlined in the requirements (Login, Manage passengers, Update passenger's status), design the module structure.
* Specify how the administrator login functionality is implemented in the detailed design.

3. Passenger Module Design:

* For passenger operations (Login, Edit Profile, View available flights, Reservation, Ticket cancellation), design the module structure.
* Provide details on how the passenger login and profile editing are implemented in the detailed design.

4. Flight Information Module Design:

* Design the module for handling Flight Information based on the provided requirements.
* Clearly specify how both Domestic and International flights are represented and managed in the detailed design.

5. Input/Output Handling:

* Detail how user inputs (Passenger and Administrator information, Flight details) are processed in the system.
* Specify the mechanisms for displaying available flights, confirming reservations, and printing tickets.

6. Error Handling:

* Describe how errors or exceptional scenarios, as mentioned in the requirements, are handled in the detailed design.
* Ensure that error messages are appropriately generated and displayed to users.

7. Documentation:

* Include appropriate comments, annotations, or documentation within the design model to explain the rationale behind design decisions and how they fulfill the specified requirements.

# 6. Test Plan

## 6.1 Requirements/specifications-based system level test cases

## 6.2 Traceability of test cases to use cases.

## 6.3. Techniques used for test generation.

## 6.4. Assessment of the goodness of your test suite

# 7. Documentation

## 7.1 Generate the user manual of your system.

## 7.2 Generate the installation and deployment manual of your system.

List of Tables:

[Table 1 Risk Analysis 9](#_Toc154150231)