**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)

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.

A screenshot of a project management system

Description automatically generated2.5 Deliverables and Schedule (Angelos Psillis)

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.