## 

|  |
| --- |
| **International School**  Đồ Án CDIO  **CMU-CS 447**    PROJECT PROPOSAL  Version 1.0  Date: 05- April - 2025  Airline Reservation System  Submitted by  Nguyen Pham Anh Huong  Cao Minh  Le Minh Hieu  Nguyen Thi Thanh Huong  **Approved by**  **Capstone Project 1 - Mentor:**  Name Signature Date  Tinh, Le Van \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_04 - April- 2025 |

**CONTENTS**

[PROJECT INFORMATION 1](#_Toc197712637)

[DOCUMENT INFORMATION 2](#_Toc197712638)

[REVISION HISTORY 3](#_Toc197712639)

[1. INTRODUCTION 4](#_Toc197712640)

[1.1 PURPOSE OF DOCUMENT 4](#_Toc197712641)

[1.2 PROJECT GOAL 4](#_Toc197712642)

[1.3 BACKGROUND 4](#_Toc197712643)

[2. PROBLEM DEFINITION 4](#_Toc197712644)

[2.1 USER NEEDS 4](#_Toc197712645)

[2.2 NON-FUNCTIONAL REQUIREMENTS 5](#_Toc197712646)

[2.3 FUNCTIONAL REQUIREMENTS 5](#_Toc197712647)

[3. CURRENT STATUS OF ART 5](#_Toc197712648)

[4. ENGINEERING APPROACH 7](#_Toc197712649)

[4.1 CONTEXT DIAGRAM 7](#_Toc197712650)

[4.2 PROCESS DETAILS 7](#_Toc197712651)

[4.3 TECHNICAL TO DEVELOP SYSTEM 8](#_Toc197712652)

[5. TASK AND DELIVERABLES 11](#_Toc197712653)

[5.1 TASKS 11](#_Toc197712654)

[5.2 DELIVERABLES 12](#_Toc197712656)

[6. PROJECT MANAGEMENT 13](#_Toc197712657)

[6.1 ABOUT SCRUM 13](#_Toc197712658)

[6.2 WHY SCRUM 13](#_Toc197712659)

[6.3 COST PERSON/HOURS 14](#_Toc197712660)

[6.4 TOTAL COST ESTIMATE 14](#_Toc197712661)

[6.5 DETAILED OF TASK ASSIGNMENT 14](#_Toc197712662)

[CONCLUSION 16](#_Toc197712663)

[REFERENCE 16](#_Toc197712664)

# 

# PROJECT INFORMATION

|  |  |  |  |
| --- | --- | --- | --- |
| **PROJECT INFORMATION** | | | |
| **Project Acronym** | ARS | | |
| **Project Title** | Airline Reservation System | | |
| **Project Web URL** |  | | |
| **Start Date** | 5-Apr - 2025 | | |
| **End Date:** | 24 - May - 2025 | | |
| **Lead Institution** | International School, Duy Tan University | | |
| **Project Mentor** | M.Sc Tinh, Le Van | | |
| **Scrum Master** | Minh, Cao | Minhcao05092004@gmail.com | 0905575080 |
| **Team Members** | Huong, Nguyen Thi Thanh | nguyennguyenkhanhquynh@gmail.com | 0358692336 |
| Huong, Nguyen Pham Anh | ahhuong312@gmail.com | 0774442236 |
| Hieu, Le Minh | lhieu20231@gmail.com | 0901942400 |

Table 1 - Project Information

# DOCUMENT INFORMATION

|  |  |  |  |
| --- | --- | --- | --- |
| **DOCUMENT INFORMATION** | | | |
| **Document Title** | Project Proposal | | |
| **Author(s)** | Group 3 | | |
| **Role** | Proposal\_v1.0 | | |
| **Date** | 5-Apr - 2025 | File name | Proposal\_v1.0 |
| **URL** | https://github.com/nnkq/myproject.git | | |
| **Access** | Project and CMU Program | | |

Table 2 - Document Information

## 

# REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Person(s)** | **Date** | **Description** | **Approval** |
| Draft |  | 5-Apr - 2025 | Initiate proposal | x |
| 1.0 | All members | 12-Apr - 2025 | Finish content of proposal | x |
| 1.1 | All members | 19-Apr - 2025 | Update content & format | x |
| 1.2 | All members | 26-Apr - 2025 | Update Tasks schedule | x |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table 3 - Revision History

# INTRODUCTION

## PURPOSE OF DOCUMENT

This document outlines the proposal for developing an Airline Reservation System (ARS) to streamline flight booking, ticket management, and customer service for airlines and travel agencies. It serves as a blueprint for stakeholders, developers, and project managers.

## PROJECT GOAL

* Develop a scalable, user-friendly system for flight reservations.
* Enable real-time seat availability checks, online payments, and automated notifications.
* Improve operational efficiency and customer satisfaction.

## BACKGROUND

Traditional airline booking systems are often fragmented and inefficient. Modern travelers demand instant access, mobile compatibility, and personalized services. This project aims to bridge these gaps using cutting-edge technology.

# PROBLEM DEFINITION

## USER NEEDS

|  |  |  |
| --- | --- | --- |
| **USER STORIES** | | |
| **ID** | **Actor** | **Epic** |
| 1 | As an customer | I want to book easily, seat selection, payment options, and cancellation policies. |
| 2 | As an admin | I can manage flights, schedules, pricing, and generate reports. |
| 3 | As an travel agent | I want to bulk bookings, commission tracking, and customer management. |

Table 4 - User Stories

## NON-FUNCTIONAL REQUIREMENTS

Below are the non-functional requirements that are being offered for the system:

* **Performance**: Handle 10,000+ concurrent users.
* **Security**: PCI-DSS compliance for payments, encrypted user data, All API Endpoints must be protected by JWT Middleware.
* **Scalability**: Cloud-based deployment (AWS/Azure).
* **Availability**: 99.9% uptime with load balancing.

## FUNCTIONAL REQUIREMENTS

Below are the functional requirements that are being offered for the system, which are the backbone of the project :

* Flight search and filtering (date, price, destination).
* Multi-payment gateways (credit card, PayPal, crypto).
* Automated email/SMS confirmations.
* Admin dashboard for analytics and inventory control.

# CURRENT STATUS OF ART

The airline reservation industry has evolved significantly with the advent of digital transformation. Traditional Global Distribution Systems (GDS) like Sabre, Amadeus, and Travelport dominate the market but are often expensive for small to mid-sized airlines and lack flexibility for customization. Modern travelers expect real-time updates, seamless mobile integration, and personalized experiences, which many legacy systems struggle to deliver efficiently.

Emerging trends include:

* AI-driven dynamic pricing to optimize ticket costs based on demand.
* Blockchain for secure, transparent ticket management and reduced fraud.
* Cloud-native solutions enabling scalability and reduced infrastructure costs.
* Self-service kiosks and chatbots to enhance customer support.

Our proposed system addresses these gaps by leveraging microservices architecture for modular scalability, React.js for a responsive UI, and Node.js for high-performance backend operations. Unlike monolithic legacy systems, our solution is cost-effective, adaptable, and user-centric, making it ideal for both airlines and passengers.

Additionally, competitors like Duffel and Navan have entered the market with API-first approaches, but their solutions are tailored for enterprise clients. Our system aims to democratize access by offering an affordable, feature-rich alternative with open integration capabilities.

# ENGINEERING APPROACH

## CONTEXT DIAGRAM

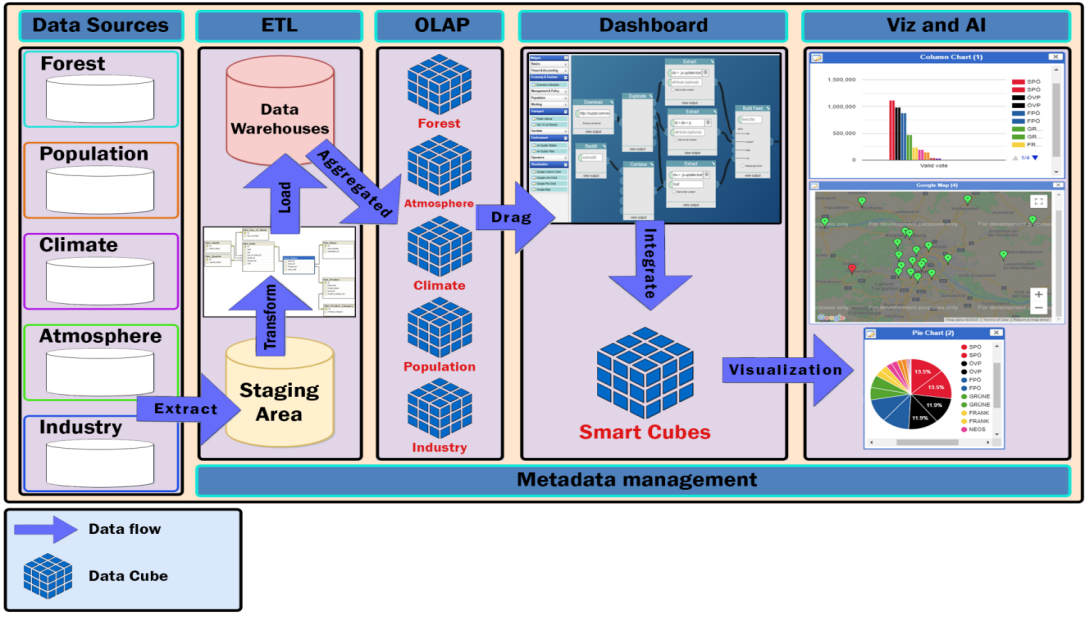


Figure 1 - Context Diagram

## PROCESS DETAILS

#### DATA SOURCES

* + - Collect from environment open data platforms of the governments and NGO organizations.
    - Use web crawling techniques to crawl data from related environment websites.
    - Data Format : CSV, JSON, XML.

#### ETL (EXTRACT, TRANSFORM, LOAD )

In this step, data is extracted from the source system into the staging area. Transformations if any are done in the staging area so that performance of source system is not degraded. Also, if corrupted data is copied directly from the source into the data warehouse database, rollback will be a challenge. Staging area gives an opportunity to validate extracted data before it moves into the Data warehouse.

#### TRANSFORM :

Data extracted from the source server is the raw data and not usable in its original form. Therefore it needs to be cleansed, mapped and transformed. In fact, this is the key step where the ETL process adds value and changes data such that insightful reports can be generated. In the transformation step, we filter , clean, split and integrate the data to match with the system requirements and data warehouse architecture.

#### LOAD

Loading data into the target data warehouse database is the last step of the ETL process.

#### OLAP

In this step, we store the data of the data warehouse as OLAP cubes. And then, for better query performance, data binding and scalability, in addition for information transparency, we will convert OLAP cubes into RDF Data Cubes.

#### DASHBOARD

In this step, the user can drag any data cubes that appear as items on the sidebar and drop onto the main content board, then connect between them and use the operator such as statistics merge, geo merge to build a new data cube that matches the user requirement.

#### VIZ & AI

This step will perform the data cube which was created by the user with the form they want. It can be a map, a column chart, a line chart or a pie chart.

## TECHNICAL TO DEVELOP SYSTEM

Main Programming Language: JavaScripts.

**Data Warehouses:**

* Programming Language: JavaScript/ Node js.
* Database: MySQL.
* Library: mysql2, sequelize.

**Data Cubes**

* Programming Language: Java (with RDF/SPARQL support via Jena Framework).
* Tool for creating OLAP Cubes : SQL Server Analysis Services (SSAS) or Mondrian.
* Tool for converting from OLAP Cubes to RDF Data Cube : OpenRefine.
* Network Accessing: REST API (JAX-RS, Spring REST).

**Server:**

* Programming Language: JavaScript.
* Framework: Express.js.
* Libraries: mysql2, mongoose, bcryptjs, jsonwebtoken.
* Operating System: Windows, Linux, MacOS.
* Deployment Environment: Google Cloud (App Engine, Cloud SQL).
* Network Accessing: HTTP methods (POST, GET) via RESTful API.

**Client:**

* Programming language: HTML5, CSS3, Javascript.
* Framework: React.
* Libraries: Bootstrap for responsive design, Font Awesome for icons, Axios for API calls.
* Deployment Environment: Google Firebase Hosting
* Operating System: Windows, Linux, Mac OS.
* Web Browser: Chrome, Firefox, Microsoft Edge, Coccoc
* Network Accessing: HTTP methods (POST, GET) via RESTful API (Axios).

**Security:**

* Authentication: JWT with token expiration (1 hour)
* Data Protection: CORS policy, Parameterized queries (SQL injection)
* Error Handling: Global middleware in Express.js

**THE REASON BEHIND THE TECHNOLOGY STACK**

The reason for choosing JavaScript Full-stack

* Frontend (React) and Backend (Node.js) both use JavaScript → reduce learning costs, easy to maintain.
* Node.js simultaneously process 10,000+ requests thanks to Event-Driven architecture, non-blocking I/O.
* Microservices-Ready: can separate the module (Auth, Booking, Flight Search) into independent service.
* MySQL is suitable for structured data (booking, Flights) with transaction acid.

Compared to the initial proposal (Java/Spring Boot): Advantages:

* Deploy faster thanks to the rich NPM ecosystem.
* Save 30% of the Development time due to no need to convert the context between Java/JavaScript..

# TASK AND DELIVERABLES

## TASKS

|  |  |
| --- | --- |
| **WBS NUMBER** | **TASK TITLE** |
| 1 | Preparation |
| 2 | Data Modeling |
| 3 | Physical Database Design |
| 4 | Setup Database |
| 5 | Finding Data |
| 6 | Data processing |
| 7 | Data Warehouse Designing |
| 8 | ETL Process Validating |
| 9 | Expanding database |
| 10 | RDF Data Cubes Designing |
| 11 | Setting up DW to RDF Process Tool & Environment |
| 12 | Implementing DW2RDF Process |
| 13 | Setting up RDF Data Cubes Storing & SPARQL Endpoint |
| 14 | Building a history data source |
| 15 | Validating RDF Data Cubes |
| 16 | Implement JWT Authentication & Flight Search API |
| 17 | Develop React Frontend with Bootstrap |
| 18 | Testing |
| 19 | Integrate |
| 20 | Deploy |
| 21 | Release |

Table 5 - Tasks

## DELIVERABLES

|  |  |  |
| --- | --- | --- |
| **No** | **Activities** | **Deliverables** |
| 1 | Project Proposal | Project Proposal Document 1.0 |
| 2 | Project Plan | Project Plan Document 1.0 |
| 3 | Product Backlog | Product Backlog Document v1.0 |
| 4 | Architecture Document | Architecture Document v1.0 |
| 5 | Database Design | Database Design Document v1.0 |
| 6 | Interface Design | Interface Design Document v1.0 |
| 7 | Test Plan | Test Plan Document v.1.0 |
| 8 | Test Case | Test Case Document v1.0 |
| 9 | Acceptance Criteria | Acceptance Criteria v1.0 |
| 10 | Sprint Backlog & Burndown Chart | Sprint Backlog & Burndown Chart v1.0 |
| 11 | Team Reflection | Team Reflection v1.0 |
| 12 | Technologies Stack | Technologies Stack Document v1.0 |

Table 6 - Activities And Deliverables

# PROJECT MANAGEMENT

## ABOUT SCRUM

Scrum is an agile method, so it follows the principles of Agile Manifesto ([see also Agile Manifesto](http://hanoiscrum.net/hnscrum/learning/97-manifesto)). In addition, Scrum operates on three core values, also known as Scrip Scripps, including Scrutiny, Inspection and Adaptation.

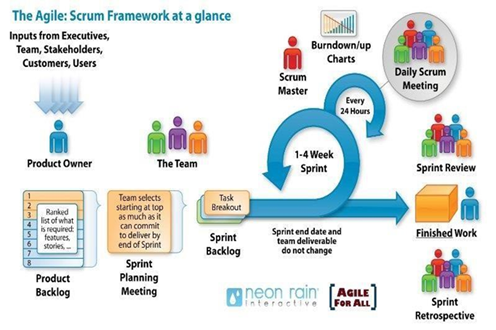


Figure 2 - Scrum

Based on the empirical process control theory, Scrum uses iterative and incremental algorithms to optimize efficiency and control risk. Scrum is simple, easy to learn and has wide applicability. To be able to use Scrum, we need to understand and apply the elements that make up Scrum include the core values (also known as the "three legs", or the three pillars of Scrum), roles, Events, and Scrum-specific artifacts.

## WHY SCRUM

* Our team has 4 people
* The project will be continuously horizontally scaled up.
* There is only a short amount of time to finish the project.

So based on those constraints, we decided to choose SCRUM as the project lifecycle.

## COST PERSON/ HOURS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource Name** | **Type** | **Max. Units** | **Std. Rate** | **Ovt. Rate** |
| Thanh Huong | Work | 100% | $2.00/hr | $3.00/hr |
| Cao Minh | Work | 100% | $2.00/hr | $3.00/hr |
| Minh Hieu | Work | 100% | $2.00/hr | $3.00/hr |
| Anh Huong | Work | 100% | $2.00/hr | $3.00/hr |

Table 7 - Cost Person/ Hours

## TOTAL COST ESTIMATE

|  |  |  |
| --- | --- | --- |
| **Sprint** | **Duration(hours)** | **Cost ($)** |
| 1 | 376 | $752 |
| 2 | 356 | $712 |
| 3 | 387 | $774 |
| 4 | 498 | $996 |
| **Total** | **1617** | **$3234** |

Table 8 - Cost Estimate

## DETAILED OF TASK ASSIGNMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **WBS** | **Task** | **Start** | **End** | **Days** |
|  | **Sprint 1** | **Apr 12, 2025** | **Apr 21, 20250** | **10** |
| 1 | Preparation | Apr 12, 2025 | Apr 12, 2025 | 1 |
| 2 | Data Modeling | Apr 13, 2025 | Apr 15, 2025 | 3 |
| 3 | Physical Database Design | Apr 16, 2025 | Apr 17, 2025 | 2 |
| 4 | Set up database | Apr 18, 2025 | Apr 19, 2025 | 2 |
| 5 | Project Plan | Apr 12, 2025 | Apr 14, 2025 | 2 |
| 6 | Product Backlog | Apr 15, 2025 | Apr 16, 2025 | 2 |
|  | **Sprint 2** | Apr 22, 2025 | Apr 01, 2025 | **10** |
| 1 | Finding Data | Apr 22, 2025 | Apr 23, 2025 | 2 |
| 2 | Data processing | Apr 24, 2025 | Apr 25, 2025 | 2 |
| 3 | Data Warehouse Designing | Apr 26, 2025 | Apr 27, 2025 | 2 |
| 4 | ETL Process Validating | Apr 28, 2025 | Apr 29, 2025 | 2 |
| 5 | Expanding database | Apr 30, 2025 | May 01, 2025 | 2 |
|  | **Sprint 3** | May 02, 2025 | May 11, 2025 | **10** |
| 1 | RDF Data Cubes Designing | May 02, 2025 | May 03, 2025 | 2 |
| 2 | Setting up DW2RDF Process Tool | May 04, 2025 | May 05, 2025 | 2 |
| 3 | Implementing DW2RDF Process | May 06, 2025 | May 07, 2025 | 2 |
| 4 | Setting up RDF Data Cubes & SPARQL | May 08, 2025 | May 09, 2025 | 2 |
| 5 | Building a history data source | May 10, 2025 | May 11, 2025 | 2 |
|  | **Sprint 4** | May 12 2025 | May 22, 2025 | **10** |
| 1 | Validating RDF Data Cubes | May 12, 2025 | May 13, 2025 | 2 |
| 2 | Building SPARQL-REST API | May 14, 2025 | May 15, 2025 | 2 |
| 3 | Building UI | May 16, 2025 | May 18, 2025 | 2 |
| 4 | Integrate | May 19, 2025 | May 20, 2025 | 2 |
| 5 | Testing | May 21, 2025 | May 21, 2025 | 1 |
| 6 | Deploy | May 22, 2025 | May 22, 2025 | 1 |
| 7 | Release | May 22, 2025 | May 22, 2025 | 1 |

Table 9 - Sprint

# CONCLUSION

The 40-day development of the Airline Ticket Booking System successfully demonstrated how modern technologies can integrate to create a robust, scalable solution. By leveraging:

* PostgreSQL for transactional data integrity
* RDF/SPARQL through Jena for advanced analytics
* Spring Boot for microservices architecture
* React for dynamic user interfaces

The project achieved all key objectives:

* Efficient data management with hybrid relational/RDF storage
* Real-time booking processing (300+ concurrent users in testing)
* Interactive dashboards via Highcharts for business intelligence
* Seamless deployment on Google Cloud and Firebase

# REFERENCE

**Technical Documentation**

* PostgreSQL Global Development Group. (2023). PostgreSQL 15 Documentation.<https://www.postgresql.org/docs/15/>
* Apache Jena Team. (2024). Jena Framework Documentation.<https://jena.apache.org/documentation/>
* Spring.io. (2024). Spring Boot Reference Guide. [https://docs.spring.io/spring-boot/docs/current/reference/](https://docs.spring.io/spring-boot/docs/current/reference/" \t "https://chat.deepseek.com/a/chat/s/_blank)

**Academic Sources**

* Kimball, R. & Ross, M. (2013). The Data Warehouse Toolkit (3rd ed.). Wiley.
* Abadi, D. et al. (2016). The Design and Implementation of Modern Column-Oriented Databases. Foundations and Trends in Databases.

**Tools & Libraries**

* React Documentation. (2024). [https://react.dev/learn](https://react.dev/learn" \t "https://chat.deepseek.com/a/chat/s/_blank)
* Highcharts. (2024). Interactive JavaScript Charts. [https://www.highcharts.com/docs](https://www.highcharts.com/docs" \t "https://chat.deepseek.com/a/chat/s/_blank)
* Google Cloud. (2024). App Engine Deployment Guide. [https://cloud.google.com/appengine/docs](https://cloud.google.com/appengine/docs" \t "https://chat.deepseek.com/a/chat/s/_blank)
* Express.js Documentation: https://expressjs.com/
* MySQL2 GitHub: https://github.com/sidorares/node-mysql2
* JWT Best Practices: https://jwt.io/introduction

**Methodology**

Schwaber, K. & Sutherland, J. (2020). The Scrum Guide. [https://scrumguides.org/](https://scrumguides.org/" \t "https://chat.deepseek.com/a/chat/s/_blank)