Logo

Description automatically generated  
Deliverable 3 : Design Specification (DS)

Andrew Daur, Joshua Snider, Matthew Lavin

CS420 Software Engineering Project  
Professor: Elvis Foster  
Spring 2022

Design specification

Lambert Air Management System

Table of Contents

[1. System Overview 2](#_Toc100055587)

[1.1 Problem Definition 2](#_Toc100055588)

[1.2 Proposed Solution 4](#_Toc100055589)

[1.3 System Architecture 5](#_Toc100055590)

[1.3.1 Object Flow Diagram 6](#_Toc100055591)

[1.3.2 Information Topology Chart 7](#_Toc100055592)

[2. Design Conventions 8](#_Toc100055593)

[3. Database Specification 10](#_Toc100055594)

[3.1 Introduction 10](#_Toc100055595)

[3.2 Database Specification 10](#_Toc100055596)

[3.3 Logical Views 23](#_Toc100055597)

[4. User Interface Design 24](#_Toc100055598)

[5. Operations Design 28](#_Toc100055599)

[5.1 Port Information Subsystem 29](#_Toc100055600)

[5.2 Airline & Aircraft Subsystem **Error! Bookmark not defined.**](#_Toc100055601)

[5.3 Arrival/Departure Subsystem 41](#_Toc100055602)

[6.Other Design Considerations 44](#_Toc100055603)

[6.1 Message and Help Specification 56](#_Toc100055604)

[7.Summary & Concluding Remarks 57](#_Toc100055605)

[7.1 PERT and Project Schedule 58](#_Toc100055606)

# 1. System Overview

LAM is a flight management system designed to combat the fierce need for efficiency in handling everyday operations at airlines. Ever increasing customer demand means it is harder to organize flights, especially in bulk, with the current mechanisms in place (faculty, departments, expanding physical space, more queues with faster processing) and demands a new solution. This system organizes information fed into it into easy to read tables and each faculty member can have separate permissions for what they can access according to their job description. This way all detail info is updated in real time and is synced across the entirety of the LAM system, cutting space requirements and faculty stress heavily while increasing the amount of processing power heavily. This solves the demand problem and is scalable to any size airline and is customizable to meet the needs of any particular airline.

## 1.1 Problem Definition

The airline industry has gone through significant upheaval in recent years; challenges in staffing and consumer demand have placed considerable strain on even well-established airlines and airports. Daily operation of an airport involves the management of traffic and staff from multiple airlines operating numerous flights, some detailed procedures that are required to maintain efficiency include:

1. Runway access for landing without long holding patterns
2. Prompt access to an open gate upon landing, without having to hold on the tarmac
3. Open gate when flight is ready to be boarded/disembark (short wait times ideally)
4. Flight crew arrangement and attendance for scheduled flight, staff emergency scheduling
5. Maximizing the flight schedule for more flights in a day, increase in revenue daily
6. Control tower staff scheduling and flight directing for multiple runways to avoid collision
7. Oversee a flight and all related information about that flight to check if it is on schedule
8. Resource management for landside operations, airside operations, billing and invoicing, and information management departments will be based off flight information for all airports.

Poor airline service due to delays reflects poorly on both us and the firms we work with. The air travel industry continues to grow steadily as more and more demand hits the markets, airport staffing, and scheduling gets increasingly difficult as airports try to expand to accept more customers and keep all related departments in the green. Below is a figure from World Bank that shows very clearly how year after year the air travel industry must react to increasing demand with exception of events that ban air travel en masse. If there is no solution proposed beyond simply more manpower and more processing stations for various tasks the air travel industry will very quickly run out of resources and manpower to accomplish the tasks needed to sustain the demand professionally enough that it is deemed acceptable.

|  |
| --- |
| Figure 1.1: World Bank statistics on increase in international tourism (1995 to 2019) |

|  |
| --- |
| *Figure 1.2: International Tourism, Number of Arrivals – World Bank* <https://data.worldbank.org/indicator/ST.INT.ARVL>  Chart, line chart  Description automatically generated |

Customers that are waiting around costs money for all involved airports and the air travel industry at large, a streamlined solution is required. Take for example a flight ‘x’ that is to be boarded in an hour as it has been scheduled, but for whatever reason a flight crew member ends up missing their check in time. We now need another crew member to fill in for flight ‘x’, but first you have to figure out who exactly the flight crew is by identifying and labeling the flight appropriately, then cross reference the crew information and scheduling to find a replacement, you also may want to go through delayed or canceled flights that may have crew still available to fill your need. You could do all of that with people power, but you may find that task quite daunting especially if the airports grow in size, but as discussed previously it simply has it’s limitation in terms of speed and efficiency, so it is not sustainable long term.

## 1.2 Proposed Solution

We propose an automated solution to this problem with the development of a Flight Management System (FMS) **can be configured for any port in any country**. This solution will reduce manpower requirements for all departments involved in flight logistics, streamlining the decision-making process for management by providing concise and accurate information about flights, crew and boarding/disembarking controls to a variety of airports internationally. This will standardize the workflow of all impacted airlines and participating ports around the world. This process is fast enough to communicate all the way around the world in a matter of seconds about a specific crew member, a flight, a port, an aircraft type, etc. It is a time saving, cost saving, space saving, and highly efficient system if done properly. Air traffic control becomes more fluid, and they are notified about flights of concern far faster and more consistently than by traditional means, thereby avoiding more potential expenses for accidents of varying kinds. This system will store empirical, quantifiable, and identifiable data on the following systems:

1. Countries of origin with associated unique code
2. Participating ports using FMS
3. Airlines operating using FMS
4. All aircraft that have flown with FMS even once, associated code and characteristics
5. All runways at associated ports using FMS
6. All terminals at associated ports using FMS
7. All gates at associated ports using FMS
8. Simple Flight Faculty (Flight Officials for this project) scheduling
9. All flights flown from an FMS port with identifiers, origin and destination
10. Flight scheduling with crew and departure date

This system will be on windows devices but strive for platform independence and as such will be able to communicate with any other windows system with this software solution. This will utilize a MySQL database to serve and maintain information and form necessary relationships between our entities of note (the systems we’re keeping track of).

## 1.3 System Architecture

The system will comprise four main components:

* Port Information Subsystem
* Aircraft & Airline Subsystem
* Arrival/Departure Subsystem
* Relation Database

### 1.3.1 Object Flow Diagram

The flow of data within the FMS is simple: the system is divided into three subsystems, each of which are connected to the central relational database. Desynchronization of data is avoided by having all subsystems’ queries being done on a shared database, eliminating the need to synchronize between distributed systems.

|  |
| --- |
| Figure 1.3: Object Flow Diagram (OFD) |

[Same for Information Topology Chart]

The entities of the FMS are divided into three subsystems: the Port Information Subsystem (PIS), Arrival/Departure Subsystem (ADS), and Aircraft & Airline Subsystem (AAS).

* Airport infrastructure, including airports themselves, terminals, gates, and runways are defined within the PIS.
* The ADS contains information on arrivals, departures, recurring scheduled flights, and their associated flight crew.
* The AAS stores information on airlines, their respective fleets of aircraft, and their flight crews, as well as general-purpose information on models of aircraft.

### 1.3.2 Information Topology Chart

Following is the information topology chart. It illustrates the separation of entities within the database into the three subsystems. Figure 3.1 and 3.2 will cover the relationships between these and how they all interact.

|  |
| --- |
| Figure 1.4: Information Topology Chart (ITC) |

# 2. Design Conventions

Describe the conventions used to name objects and the notation used in describing the database.

The database specification of the system will be covered in this section. The methodology employed for database specification is based on the entity specification grid (ESG) seen in section 3.2. Following is a summary of the conventions used:

1. Each information entity referenced is identified by a reference code, a descriptive name, and an implementation name indicated in square brackets. ([…])
2. For each entity there are a set number of attributes (data elements) to be stored and are identified by a descriptive name and implementation name also indicated in square brackets. ([…])
3. Each of the entities are presented in a specific way to make it easier to transition into a set of normalized relations in a normalized relational database.
4. Data elements that will be implemented as foreign keys in the normalized relational database will all be identified by curly braces ({…}), and will specify what entity they reference.
5. For each attribute, the physical characteristics will be given. And the attribute implementation names will be indicated in square brackets. ([…]).
6. Indexes (including primary keys or candidate keys) to be defined on the entity are indicated.
7. For each entity a comment describing the data to be stored is provided.
8. Each operation defined on an entity will be given an implementation name indicated in square brackets ([…])

Naming of objects will be very important for the following reasons:

1. The software system will host several objects. Without a proper naming convention, it will be extremely difficult to keep track of them.
2. The naming convention will enable us to easily categorize objects on sight.

|  |
| --- |
| *Figure 2.1:* Object Naming Convention. This figure provides the object naming convention for the project, excerpter from [Foster 2014] with appropriate adjustments for the current project. |

# 3. Database Specification

## 3.1 Introduction

The Lambert Air Management System consists of 11 normalized entities all implemented as relational tables. The Entity Relationship Diagram (ERD) offers a summary of how connections work, and more detail is provided in the Entity Specification Grid (ESG), detailed below.

## 3.2 Database Specification

Following is the Entity Relationship Diagram (ERD). It illustrates the relationships between all entities including data tables, logical views, and other interacting objects within the system that act on it in some way.

|  |
| --- |
| *Figure 3.1:* Entity Relationship Diagram (ERD) |

**Entity Specification Grid**

Below is an Entity Specification Grid (ESG) that we use to outline in detail, all entities (usually tables of data), how they are ordered, and how we will use the data in the entities for our solution. We have followed the naming scheme from Figure 2.1, to further detail each entity’s purpose.

*Figure 3.2*: Entity Specification Grid (ESG)

|  |
| --- |
| **E01 – Countries [LAM\_countries\_BR]** |
| **Attributes:**   1. Country Code [ CnCode ] [A3] 2. Country Name [ CnName ] [ VarChar(100) ] 3. Country Abbreviation [ CnAbbr ] [ VarChar(10) ] |
| **Comments:**  Entity to keep track of all participating countries for LAM system, referenced by many other entities for location information. Country Code and Country Abbreviation are not necessarily the same and could be confused with each other. Consider Country Code as alphanumeric representation of country, whereas country abbreviation is just alphabetic. Country Name is NOT NULL. CnCode can either be straight numeric or alphanumeric, recommend alphanumeric for easier interpretation.  Example: [ CnCode=’NA2’ ; CnAbbr=’USA’ ] representing United States. |
| **Indexes:**  1. [PK] Primary Key: LAM\_countries\_BR #1;  Constraint [ LAM\_countries\_BR\_PK ]  2. LAM\_countries\_BR #2 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_countries\_MO ]  1.1 ) Add Countries [ LAM\_countries\_AO ]  1.2 ) Update Countries [ LAM\_countries\_UO ]  1.3 ) Delete Countries [ LAM\_countries\_ZO ]  2 ) Inquire on Countries [ LAM\_countries\_IO ]  2.1 ) Report Countries [ LAM\_countries\_RO ] |

|  |
| --- |
| **E02 – Participating Ports [LAM\_participatingports\_BR]** |
| **Attributes:**   1. Port Code [ PortCode ] [A6] 2. Port Full Name [ PortFullName ] [ VarChar(100) ] 3. Port Name [ PortName ] [ VarChar(50) ] 4. Port Country Code [ PortCnCode ] [A3] {**Refers E01.CnCode**} |
| **Comments:**  Entity to keep track of all participating ports across various countries, shorthand name ‘PortName’ added for convenience of query. PortFullName is NOT NULL. |
| **Indexes:**  1. [PK] Primary Key: LAM\_participatingports\_BR #1;   Constraint [ LAM\_participatingports\_BR\_PK ]  2. LAM\_participatingports\_BR #2  3. LAM\_participatingports\_BR #3 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_participatingports\_MO ]  1.1 ) Add Ports [ LAM\_participatingports\_AO ]  1.2 ) Update Ports [ LAM\_participatingports\_UO ]  1.3 ) Delete Ports [ LAM\_participatingports\_ZO ]  2 ) Inquire on Ports [ LAM\_participatingports\_IO ]  2.1 ) Report Ports [ LAM\_participatingports\_RO ] |

|  |
| --- |
| **E03 – Airlines [LAM\_airlines\_BR]** |
| **Attributes:**   1. Airline ID Code [ AL\_IDCode] [ Char(9) ] 2. Airline Name [ AL\_Name ] [ VarChar(100) ] 3. Airline Callsign [ AL\_Callsign] [VARCHAR(32)] 4. Airline Host Country [ AL\_CnCode ] [ Char(3) ] {**Refers E01.CnCode**} 5. Airline Host Port [ AL\_PortCode ] [ Char(6) ] {**Refers E02.PortCode**} 6. Airline Primary Address [ AL\_HQAddress1 ] [ VarChar(100) ] 7. Airline Host City [ AL\_HQCity ] [ VarChar(50) ] 8. Airline Contact Phone [ AL\_HQPhone ] [ Number/Decimal(11,0) ] 9. Airline Contact Name [ AL\_Contact ] [ VarChar(100) ] |
| **Comments:**  Entity to keep track of all participating airlines, their hosting country/port, and contact information. Primary key is defined as ‘PPPPPPXXX’ where ‘P’ is the host port code, and ‘X’ is alphanumeric. AL\_Name and AL\_HQAddress1 and AL\_HQPhone are NOT NULL. |
| **Indexes:**  1. [PK] Primary Key: LAM\_airlines\_BR #1; Constraint [ LAM\_airlines\_BR\_PK ]  2. LAM\_airlines\_BR #2  3. LAM\_airlines\_BR #7 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_airlines\_MO ]  1.1 ) Add Airlines [ LAM\_airlines\_AO ]  1.2 ) Update Airlines [ LAM\_airlines\_UO ]  1.3 ) Delete Airlines [ LAM\_airlines\_ZO ]  2 ) Inquire on Airlines [ LAM\_airlines\_IO ]  2.1 ) Report Airlines[ LAM\_airlines\_RO ] |

|  |
| --- |
| **E04 – Aircrafts [LAM\_aircrafts\_BR]** |
| **Attributes:**   1. Aircraft Code [ AC\_Code ] [ Char(10) ] 2. Aircraft Name [ AC\_Name ] [ VarChar(100) ] 3. Aircraft Description [ AC\_Desc ] [ VarChar(500) ] 4. Aircraft Capacity [ AC\_NumSeats ] [ Number/Decimal(3,0) ] 5. Aircraft FeatureList [ AC\_FeatureList ] [ VarChar(MAX)/Text ] 6. Aircraft Type ID [ AC\_ATypeCode ] [ Char(3) ] {**Refers E11.ATypeCode**} 7. Aircraft Host Airline [ AC\_AL\_IDCode ] [ Char(9) ] {**Refers E03.AL\_IDCode**} |
| **Comments:**  Entity to keep track of all aircraft as children of participating airlines with a corresponding recognizable aircraft type. Feature list could be a number of different data types including: File,VarChar,Text,Blob depending on preference of storage medium for this information; For maximum compatibility across all airlines, countries, and ports, as well as minimal digital storage requirements for database, recommend either VarChar with max length or Text type. Aircraft Name and capacity NOT NULL. |
| **Indexes:**  1. [PK] Primary Key: LAM\_aircrafts\_BR #1; Constraint [ LAM\_aircrafts\_BR\_PK ]  2. LAM\_aircrafts\_BR #2 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_aircrafts\_MO ]  1.1 ) Add Aircrafts [ LAM\_aircrafts\_AO ]  1.2 ) Update Aircrafts [ LAM\_aircrafts\_UO ]  1.3 ) Delete Aircrafts [ LAM\_aircrafts\_ZO ]  2 ) Inquire on Aircrafts [ LAM\_aircrafts\_IO ]  2.1 ) Report Aircrafts [ LAM\_aircrafts\_RO ] |

|  |
| --- |
| **E05 – Runways [LAM\_runways\_BR]** |
| **Attributes:**   * Runway Code[ RwCode ] [ Char(8) ] * Runway Name [ RwName ] [ VarChar(100) ] * Runway Length [ RwLength ] [ Number/Decimal(4,0) ] * Runway Width [ RwWidth ] [Number/Decimal(4,0) ] * Runway Host Port [ RwPortCode ] [ Char(6) ] {**Refers E02.PortCode**} |
| **Comments:**  Entity to keep track of all runways as child of hosting port, with runway dimensions. Runway Code defined as ‘AAAAAANN’ where ‘A’ is a alphabet letter and ‘N’ is a number. Runway Name NOT NULL. |
| **Indexes:**  1. [PK] Primary Key: LAM\_runways\_BR #1; Constraint [ LAM\_runways\_BR\_PK ]  2. LAM\_runways\_BR #2 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_runways\_MO ]  1.1 ) Add Runways [ LAM\_runways\_AO ]  1.2 ) Update Runways [ LAM\_runways\_UO ]  1.3 ) Delete Runways [ LAM\_runways\_ZO ]  2 ) Inquire on Runways [ LAM\_runways\_IO ]  2.1 ) Report Runways [ LAM\_runways\_RO ] |

|  |
| --- |
| **E06 – Terminals [LAM\_terminals\_BR]** |
| **Attributes:**   * Terminal Code [ TermCode] [ Char(8) ] * Terminal Name [ TermName ] [ VarChar(100) ] * Terminal Status [ TermStatus ] [ Char(1) ] * Terminal Host Port [ TermPortCode ] [ Char(6) ] {**Refers E02.PortCode**} |
| **Comments:**  Entity to keep track of all terminals and associated port, complete with terminal status. TermCode defined as ‘AAAAAANN’ where ‘A’ is an alphabet letter and ‘N’ is a number. TermStatus should be open(O), closed(C), delayed(D), or unknown(?). Terminal Name NOT NULL. |
| **Indexes:**  1. [PK] Primary Key: LAM\_terminals\_BR #1;   Constraint [ LAM\_terminals\_BR\_PK ]  2. LAM\_terminals\_BR #2 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_terminals\_MO ]  1.1 ) Add Terminals [ LAM\_terminals\_AO ]  1.2 ) Update Terminals [ LAM\_terminals\_UO ]  1.3 ) Delete Terminals [ LAM\_terminals\_ZO ]  2 ) Inquire on Terminals [ LAM\_terminals\_IO ]  2.1 ) Report Terminals [ LAM\_terminals\_RO ] |

|  |
| --- |
| **E07 – Gates [LAM\_gates\_BR]** |
| **Attributes:**   * Gate Code [ GCode ] [ Char(8) ] * Gate Name [ GName ] [ VarChar(100) ] * Gate Status [ GStatus] [ Char(1) ] * Gate Host Port [ GPortCode ] [ Char(6) ] {**Refers E02.PortCode**} |
| **Comments:**  Entity to keep track of all gates including associated port and status. GCode is defined as ‘AAAAAANN’ where ‘A’ is an alphabet letter and ‘N’ is a number. GStatus should be open(O), closed(C), delayed(D), or unknown(?). Gate name NOT NULL. |
| **Indexes:**  1. [PK] Primary Key: LAM\_gates\_BR #1; Constraint [ LAM\_gates\_BR\_PK ]  2. LAM\_gates\_BR #2 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_gates\_MO ]  1.1 ) Add Gates [ LAM\_gates\_AO ]  1.2 ) Update Gates [ LAM\_gates\_UO ]  1.3 ) Delete Gates [ LAM\_gates\_ZO ]  2 ) Inquire on Gates [ LAM\_gates\_IO ]  2.1 ) Report Gates [ LAM\_gates\_RO ] |

|  |
| --- |
| **E08 – Flight Officials [LAM\_flightofficials\_BR]** |
| **Attributes:**   1. Flight Official ID Code [ FOffCode ] [ Char(6) ] 2. Flight Official First Name [ FOffFName ] [ VarChar(50) ] 3. Flight Official Middle Initial [ FOffMInitial ] [ VarChar(3) ] 4. Flight Official Family Surname [ FOffLName ] [ VarChar(50) ] 5. Flight Official Host Airline [ FOff\_ALCode ] [ Char(9) ] {**Refers E03.AL\_IDCode**} |
| **Comments:**   1. Entity to keep track of all airline staff related to flights and which airline that employs them. 2. FOffCode defined as ‘XXXXXX’ where ‘X’ is alphanumeric. All employees must have at least a first name and family surname to be valid. |
| **Indexes:**  1. [PK] Primary Key: LAM\_flightofficials\_BR #1;   Constraint [ LAM\_flightofficials\_BR\_PK ]  2. LAM\_flightofficials\_BR #4  3. LAM\_flightofficials\_BR #2, #4 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_flightofficials\_MO ]  1.1 ) Add Flight Officials [ LAM\_flightofficials\_AO ]  1.2 ) Update Flight Officials [ LAM\_flightofficials\_UO ]  1.3 ) Delete Flight Officials [ LAM\_flightofficials\_ZO ]  2 ) Inquire on Flight Officials [ LAM\_flightofficials\_IO ]  2.1 ) Report Flight Officials [ LAM\_flightofficials\_RO ] |

|  |
| --- |
| **E09 – Flights [LAM\_flights\_BR]** |
| **Attributes:**   1. Flight ID Number [ FlyCode ] [ Char(6) ] 2. Flight Origin Port [ FlySrcPortCode ] [ Char(6) ] {**Refers E02.PortCode**} 3. Flight Destination Port [ FlyDestPortCode ] [ Char(6) ] {**Refers E02.PortCode**} 4. Flight Intermediate Port [ FlyInterPortCode ] [ Char(6) ] {**Refers E02.PortCode**} 5. Flight Host Airline [ FlyALCode ] [ Char(9) ] {**Refers E03.AL\_IDCode**} 6. Flight Aircraft ID [ FlyAC\_Code ] [ Char(10) ] {**Refers E04.AC\_Code**} |
| **Comments:**   1. Entity to keep track of all flights, and key stopping locations of the flight path. 2. All flights will require flight officials and have at minimum two interacting ports of note. FlyCode defined as ‘XXXXXX’ where ‘X’ is alphanumeric. |
| **Indexes:**  1. [PK] Primary Key: LAM\_flights\_BR #1; Constraint [ LAM\_flights\_BR\_PK ]  2. LAM\_flights\_BR #2 and #3  3. LAM\_flights\_BR #6 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_flights\_MO ]  1.1 ) Add Flights [ LAM\_flights\_AO ]  1.2 ) Update Flights [ LAM\_flights\_UO ]  1.3 ) Delete Flights [ LAM\_flights\_ZO ]  2 ) Inquire on Flights [ LAM\_flights\_IO ]  2.1 ) Report Flights [ LAM\_flights\_RO ] |

|  |
| --- |
| **E10 – Scheduling [LAM\_scheduling\_BR]** |
| **Attributes:**   1. Flight ID Number [ SchFlyCode] [ Char(6) ] {**Refers E09.FlyCode**} 2. Flight Date [ SchDepartDate ] [ Number/Decimal(8,0) ] 3. Flight Primary Pilot 1 [ SchFOff\_P1 ] [ Char(6) ] {**Refers E08.FOffCode**} 4. Flight Primary Pilot 2 [ SchFOff\_P2 ] [ Char(6) ] {**Refers E08.FOffCode**} 5. Flight Primary Crew Member 1 [ SchFOff\_A1 ] [ Char(6) ] {**Refers E08.FOffCode**} 6. Flight Primary Crew Member 2 [ SchFOff\_A2 ] [ Char(6) ] {**Refers E08.FOffCode**} 7. Flight Primary Crew Member 3 [ SchFOff\_A3 ] [ Char(6) ] {**Refers E08.FOffCode**} 8. Flight Secondary Crew Member 1 [ SchFOff\_A4 ] [ Char(6) ] {**Refers E08.FOffCode**} 9. Flight Secondary Crew Member 2 [ SchFOff\_A5 ] [ Char(6) ] {**Refers E08.FOffCode**} 10. Flight Secondary Crew Member 3 [ SchFOff\_A6 ] [ Char(6) ] {**Refers E08.FOffCode**} 11. Is Flight Departed [ SchFlightOut ] [ BOOL ] 12. Is Flight On Schedule [ SchFlightOnTime ] [ BOOL ] |
| **Comments:**   1. Entity to keep track of flight schedule and associated crew members for flights. 2. All flights MUST HAVE 2 pilots and at minimum 3 assisting crew members to be considered launch-ready, all primary attributes must be filled. Primary Key is composite key of #1 and #2, the flight in question and the date of departure in that order. Date should be in the format yyyymmdd (year,month,day). |
| **Indexes:**  1. [C-PK] Primary Key: LAM\_scheduling\_BR #1 and #2;   Constraint [ LAM\_scheduling\_BR\_PK ]  2. LAM\_scheduling\_BR #2, #11, and #12 |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_scheduling\_MO ]  1.1 ) Add Scheduled Flights [ LAM\_scheduling\_AO ]  1.2 ) Update Scheduled Flights [ LAM\_scheduling\_UO ]  1.3 ) Delete/Cancel Scheduled Flights [ LAM\_scheduling\_ZO ]  2 ) Inquire on Scheduled Flights[ LAM\_scheduling\_IO ]  2.1 ) Report Scheduled Flights[ LAM\_scheduling\_RO ] |

|  |
| --- |
| **E11 – Aircraft Types [LAM\_aircrafttypes\_BR]** |
| **Attributes:**   1. Aircraft Type Code [ ATypeCode ] [ CHAR(3) ] 2. Aircraft Type Description [ ATypeDesc] [ VarChar(MAX)/Text ] 3. Aircraft Seating Capacity [ATypeSeating] [NUMBER(4)] |
| **Comments:**   1. Entity to keep record of recognized aircraft types, from which E04-Aircrafts is derived. 2. Various vehicle types are catalogued here to officially stamp them as approved aircrafts for international traveling and identify features of an aircraft type that may be beneficial for specific flight routes or use-cases. Example: Smaller aircraft types/classes may be better on fuel and hence better suited for longer distance travel to destination ports that don’t have as much demand from consumers as other ports. |
| **Indexes:**  1. [PK] Primary Key: LAM\_aircrafttypes\_BR #1;   Constraint [ LAM\_aircrafttypes\_BR\_PK ] |
| **Valid Operations:**  1 ) Maintenance Operations [ LAM\_aircrafttypes\_MO ]  1.1 ) Add Aircraft Types [ LAM\_aircrafttypes\_AO ]  1.2 ) Update Aircraft Types [ LAM\_aircrafttypes\_UO ]  1.3 ) Delete Aircraft Types [ LAM\_aircrafttypes\_ZO ]  2 ) Inquire on Aircraft Types [ LAM\_aircrafttypes\_IO ]  2.1 ) Report Aircraft Types [ LAM\_aircrafttypes\_RO ] |

## 3.3 Logical Views

The following logical views will be included within the database. They will offer various pre-processed views of the data, as will be requested by various system users, without the queries involved needing to be replicated each time.

*Figure 3.3:* Relationship Joins

|  |  |
| --- | --- |
| Descriptive Name | Summarized Specification |
| **Port Connections** | Joins Port Definition (E02) to Country Definition (E01) |
| **Airline Connections** | Joins Airline (E03) to Port (E02) and Country (E01) |
| **Crew of Specific Flight** | Joins Flight Crew (E08) with Flights (E09) and Scheduling (E10) |
| **Crew Schedule** | Joins Flight Crew (E08) with Scheduling (E10) |
| **Airframe Schedule** | Joins Aircraft (E04) with Aircraft Type (E11), Flights (E09), and Scheduling (E10) |
| **Arrival/Departure Board** | Joins Flights (E09), Scheduling (E10), Airline (E03), Port (E02), Terminal (E06), and Gate (E07) |
| **Per-Gate Schedule** | Joins Flights (E09) with Scheduling (E10), Airline (E03), and Gate (E07) |
| **Delays & Cancellations** | Arrival/Departure board with added selection of only cancelled or delayed flights. |
| **Airline Fleet** | Joins Airline (E03) with Aircraft (E04) and Aircraft Type (E11) |
| **Airport Infrastructure Report** | Joins Port (E02) with Country (E01), Runway (E05), Terminal (E06), Gate (E07) |

# 4. User Interface Design

The information entities outlined in the previous section will be implemented as a relational database using MySQL, VS Code and a mixture of Python and Java. Superimposed on this database will be an object-oriented graphical user interface consisting of various operations that provide functionality and flexibility to the end users.

The user functionality and flexibility will be provided based on the following approaches:

* For each of the information entities, there will be a defined object type with the same name.
* For each object type, define a set of basic operations that will facilitate the addition, modification, deletion, inquiry, or reporting of the data related to that object type. User should note that each of the object types do no necessarily require all five of the listed operations. For example, you cannot add records to a logical view, because it is not a data table, it is simply a virtual read only report. You would instead alter the search the view is performing or make a new view.
* For selected object types, introduce flexibility and sophistication. These will be expressed as, more complex inquiry abilities, or more in-depth reporting functionality. This will give the end users the ability to specify specific criteria of their choosing. These features will be provided in a clear and straightforward manner through use of dropboxes, checkboxes, buttons, and user prompts without cluttering the screen.
* This will be deployed as a web application sticking with the cross-platform and flexibility approach so that clients can access from any device powerful enough to run a web browser, that is to say anything that is a smart phone or more powerful.
* The utility operations will be defined when the user interface is further explored. Specifically with inquiry methods, we will approach starting with a button that pops up a prompt for the user to choose what objects they want to search, and then based on that input, provide with input boxes for filtering or refining the search before submitting it. Other methods may be overloaded to provide optional refining parameters, which the UI will allow the user to access through input boxes next to the matching button to apply the method. This provides a better user experience as long as the User Interface doesn’t become overwhelming or distracting.

***Figure 4.1:* User Interface Topology Chart (UITC) for LAMS**

|  |
| --- |
| **LAMS Administrative Menu** |
| 1. Port Information System (PIS)  2. Arrival/Departure System (ADS)  3. Airline/Aircraft System (AAS) |

|  |
| --- |
| **1. Port Information System** |
| * 1. **Country Definitions**      1. Add Country Definitions      2. Modify Country Definitions      3. Delete Country Definitions      4. Inquire Country Definition   2. **Airport Definitions**      1. Add Airport Definitions      2. Modify Airport Definitions      3. Delete Airport Definitions      4. Inquire/Report Airport Definitions   3. **Terminal**      1. Add Terminal Definitions      2. Modify Terminal Definitions      3. Delete Terminal Definitions      4. Inquire/Report Terminal Definitions   4. **Gate**      1. Add Gate Definitions      2. Modify Gate Definitions      3. Delete Gate Definitions      4. Inquire/Report Gate Definitions   5. **Runway**   1.5.1 Add Runway Definitions   * + 1. Modify Runway Definitions     2. Delete Runway Definitions     3. Inquire/Report Runway Definitions |

|  |
| --- |
| **2. Arrival/Departure System** |
| **2.1 Flights Definitions**  2.1.1 Add Flights Definitions  2.1.2 Modify Flights Definitions  2.1.3 Delete Flights Definitions  2.1.4 Inquire Flights Definition  **2.2 Flight Schedule Definitions**  2.2.1. Add Flight Schedule Definitions  2.2.2 Modify Airport Definitions  2.2.3 Delete Airport Definitions  2.2.4 Inquire/Report Airport Definitions  **2.3 Flight Crew Definitions**  2.3.1 Add Flight Crew Definitions  2.3.2 Modify Flight Crew Definitions  2.3.3 Delete Flight Crew Definitions  2.3.4 Inquire/Report Flight Crew Definitions |

|  |
| --- |
| **3. Arrival/Departure System** |
| **3.1 Airline Definitions**  3.1.1 Add Airline Definitions  3.1.2 Modify Airline Definitions  3.1.3 Delete Airline Definitions  3.1.4 Inquire Airline Definition  **3.2 Aircraft Definitions**  3.2.1 Add Aircraft Definitions  3.2.2 Modify Aircraft Definitions  3.2.3 Delete Aircraft Definitions  3.2.4 Inquire/Report Aircraft Definitions  **3.3 Aircraft Category Definitions**  3.3.1 Add Aircraft Category Definitions  3.3.2 Modify Aircraft Category Definitions  3.3.4 Delete Aircraft Category Definitions  3.3.5 Inquire/Report Aircraft Category Definitions  **3.4 Flight Crew Definitions**  3.4.1 Add Flight Crew Definitions  3.4.2 Modify Flight Crew Definitions  3.4.3 Delete Flight Crew Definitions  3.4.4 Inquire/Report Flight Crew Definitions |

# 5. Operations Design

Order of Items should be as follows: System Rules, Generic, Actual Specs.

In this chapter, operation design for the LAMS project operations are outlined below. Majority of the generic algorithms needed for these operations are fairly similar. Therefore, in interest of being brief, some pseudo-code is not unnecessarily repeated.

Software systems will always have operations, whether the functional approach or the object-oriented approach is employed. The operation specifications for the operations of the system are vital aspects of preparing for the Operations Design.

MOVE PSEUDO-CODE TO AFTER FOR GENERIC

TEMPLATE FOR O-SPEC:

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_\*ENTITY\_NAME\*  Operation Description: \*Entity Description\*  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**  Input Form  **Outputs:**  **Validations Rules:**      **Special Notes:**  None.  **Operation Outline:**  See generic ADD pseudo-code  See generic MODIFY pseudo-code  See generic DELETE pseudo-code |

## 5.1 Port Information Subsystem

The operation specification for the Port Information Subsystem follows.

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_Country\_AO  Operation Description: Facilitates addition of Countries.  Operation Category: Mandatory  Complexity Rank: 3 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**   1. Country code 2. Country name 3. Abbreviation   **Outputs:**  **Validations Rules:**   * Country key must conform to ISO-3166-1 standards, e.g. United States is "USA", Great Britain is "GBR", Germany is "DEU" * Country name may not be null.   **Special Notes:**  It will be possible to view information based on the following access paths:  By country code  By country name  **Operation Outline:**  See generic ADD pseudocode  See Figure 5.1: ADD Method [Generic] Pseudo-code. |

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Port Information

Operation Name: LAM\_Country\_AO/ZO

Operation Description: Facilitates addition/deletion of Countries.

Operation Category: Mandatory

Complexity Rank: 3 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on the countries which have connected airports

**LAM\_Country\_BR** – Country definitions (E01)

**Outputs:**

**LAM\_Country\_BR** – Country definitions (E01)

**Validation rules (AO):**

* Country key must conform to ISO-3166-1 standards.Country key must conform to ISO-3166-1 standards, e.g. United States is "USA", Great Britain is "GBR", Germany is "DEU"
* Country name may not be null.

**Validation rules (ZO):**

* Country key must exist in database

**Special Notes:**

It will be possible to view information based on the following access paths:

By country code

By country name

**Operation Outline:**

See generic ADD/DELETE pseudocode

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Port Information

Operation Name: LAM\_Port\_AO/MO/ZO

Operation Description: Facilitates addition/modification/deletion of Ports.

Operation Category: Mandatory

Complexity Rank: 5 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

1. Information on participating ports
2. LAM\_Port\_BR

**Outputs:**

**LAM\_Port\_BR**

**Validation rules (AO):**

* Port key must refer to real port
* Port name, abbreviation, may not be null.
* Related country key must refer to country record

**Validation rules (MO):**

All addition rules apply, and port key must refer to existing record.

**Special Notes:**

It will be possible to view information based on the following access paths:

By port code

By port name

By country

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Port Information

Operation Name: LAM\_Terminal\_AO/MO/ZO

Operation Description: Facilitates addition/modification/deletion of airport terminals.

Operation Category: Mandatory

Complexity Rank: 6 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on airport terminals

LAM\_Terminal\_BR

**Outputs:**

LAM\_Terminal\_BR

**Validation rules (AO):**

* Terminal code must be unique and non-null
* Terminal name may not be empty
* Terminal abbreviation may not be empty
* Port key must refer to a valid airport

**Validation rules (MO):**

* All previous rules apply
* Terminal key must refer to existing record

**Special Notes:**

It will be possible to view information based on the following access paths:

* By terminal code
* Terminal name
* Associated port

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Port Information

Operation Name: LAM\_Gate\_AO / LAM\_Gate\_MO / LAM\_Gate\_ZO

Operation Description: Facilitates addition/modification/deletion of airport gates.

Operation Category: Mandatory

Complexity Rank: 3 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information regarding airport gates

LAM\_Gate\_BR

**Outputs:**

LAM\_Gate\_BR

**Validations Rules:**

* Gate code must be unique and non-null
* Gate name may not be empty
* Gate abbreviation may not be empty
* Associated port key must be valid

**Special Notes:**

It will be possible to view information based on the following access paths:

* By gate name
* By associated port

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Port Information  Operation Name: LAM\_Runway\_AO / \_MO / \_ZO  Operation Description: Generic ADD/MODIFY/DELETE operation for entity to add data record, required parameters change depending on what entity you wish to add to, and will be set by the entity definition, but the structure will follow the outline here.  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**  Input Form for Runways:   1. Runway Code 2. Runway Name 3. Runway Length 4. Runway Width 5. Host Port Code   **Outputs:**  Completed Input Form showing that adding of object was successful with confirmation. This can be done through UI through dialog box or by presenting/ writing some sort of message to the display.  **Validations Rules:**  Varies by entity, but all parameters on the input form must conform to the limits set by the entity, and meet Entity Specification Grid requirements for input to ensure a successful record addition to a new table.   1. Conform to the underlying database, data type must match 2. Conform to ESG, special requirements for Primary Keys or other indexes 3. Must not be NULL 4. Runway Length (measured in meters) must be non-negative 5. Runway Width (measured in meters) must be non-negative 6. Valid associated Port Key   **Special Notes:**  It will be possible to order information using the following options:  -- Terminal Code  -- Terminal Name  -- Associated Port  **Operation Outline:**  See Figure 5.1 - 5.3: ADD/MODIFY/DELETE Method [Generic] Pseudo-code. |

## 5.2 Airline and Aircraft Subsystem

The operation specification for the Airline & Aircraft Subsystem follows.

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Aircraft/Airline

Operation Name: LAM\_Airline\_AO/MO/ZO

Operation Description: Facilitates addition/modification/deletion of airlines.

Operation Category: Mandatory

Complexity Rank: 3 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on participating airlines

LAM\_Airline\_BR

**Outputs:**

LAM\_Airline\_BR

**Validation rules (AO):**

* Airline code must refer to real port (IATA Resolution 763)Airline code must refer to real port (IATA Resolution 763), e.g. LaGuardia Airport is "LGA"
* Airline name may not be null or empty
* Host country and airport keys must be valid.

**Validation rules (MO):**

All previous rules apply, and key must refer to existing record.

**Special Notes:**

It will be possible to view information based on the following access paths:

By airline code

By airline name

By host country

By host airport

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Aircraft/Airline

Operation Name: LAM\_Aircraft\_AO/MO/ZO

Operation Description: Facilitates addition/modification/deletion of individual aircraft.

Operation Category: Mandatory

Complexity Rank: 3 of 10Complexity Rank: 9 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on individual aircraft

LAM\_Aircraft\_BR

**Outputs:**

LAM\_Aircraft\_BR

**Validation rules (AO):**

* Aircraft code must be unique and non-null
* Aircraft registration code may not be empty
* Seating capacity must be non-negative
* Type code must refer to valid aircraft type
* Airline code must refer to real airline

**Validation rules (MO):**

All previous rules apply, and key must refer to existing record.

**Special Notes:**

It will be possible to view information based on the following access paths:

* By aircraft key
* Registration code
* By type code
* By host airline code

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Aircraft/Airline

Name: LAM\_AircraftType\_AO/MO/ZO

Operation Description: Facilitates addition/modification/deletion of aircraft types.

Operation Category: Mandatory

Complexity Rank: 4 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on aircraft type

LAM\_AircraftType\_BR

**Outputs:**

LAM\_AircraftType\_BR

**Validation rules (AO):**

* Aircraft type descriptor must be validAircraft type code must be valid, e.g. Boeing 737-800 is "738"
* Aircraft type name may not be empty
* Aircraft seating must be non-negative

**Validation rules (MO):**

All previous rules apply, and key must refer to existing record.

**Special Notes:**

It will be possible to view information based on the following access paths:

* By aircraft key
* Registration code
* By type code
* By host airline code

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

## 5.3 Arrival/Departure Subsystem

The operation specification for the Arrival/Departure Subsystem follows.

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Arrival/Departure

Operation Name: LAM\_Flight\_AO/MO/ZO

Operation Description: Facilitates addition/modification/deletion of flights.

Operation Category: Mandatory

Complexity Rank: 3 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on recurring flight

LAM\_Flight\_BR

**Outputs:**

LAM\_Flight\_BR

**Validation rules (AO):**

**Validations Rules:**

* Airline code must refer to real port (IATA Resolution 763)
* Airline name may not be null or empty
* Host country and airport keys must be valid.

**Validation rules (MO):**

All previous rules apply, and key must refer to existing record

**Special Notes:**

It will be possible to view information based on the following access paths:

By airline code

By airline name

By host country

By host airport

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

**Operation Biography:**

System: Lambert Air Management System (LAMS)

Subsystem: Arrival/Departure

Operation Name: LAM\_Schedule\_AO/MO/ZO

Operation Description: Facilitates addition of scheduled flights.Operation Description: Facilitates addition/modification/deletion of scheduled flights.

Operation Category: Mandatory

Complexity Rank: 8 of 10

Spec. Author: M. Lavin, A. Daur, J. Snider

Date: 3-22-2022

**Inputs:**

Information on scheduled flight

LAM\_Schedule\_BR

**Outputs:**

LAM\_Schedule\_BR

**Validation rules (AO):**

1. Flight number may not be empty, and refer to valid flight
2. Flight date must be valid date
3. First & second pilots, as well as any attendant keys, must not be duplicates.
4. Flight crew may not be booked for two flights occurring at the same time

**Validation rules (MO):**

All previous rules apply, and key must refer to existing record.

**Special Notes:**

It will be possible to view information based on the following access paths:

By flight number

By date

By flight crew

By on-time flag

**Operation Outline:**

See generic ADD/MODIFY/DELETE pseudocode

## 5.4 Generic Methods

**Generic ADD method Operations Specification.**

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_\*ENTITY\_NAME\*\_AO  Operation Description: Generic ADD operation for entity to add data record, required parameters change depending on what entity you wish to add to, and will be set by the entity definition, but the structure will follow the outline here.  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**  Input Form containing entry parameters that vary according to the entity you want to add to. Adding to Countries would involve ‘Country Name’ and ‘Country Abbreviated’ parameters, but if adding a Gate or Terminal would require different parameters suited to that entity. Indexing handled automatically.  **Outputs:**  Completed Input Form showing that adding of object was successful with confirmation. This can be done through UI through dialog box or by presenting/ writing some sort of message to the display.  **Validations Rules:**  Varies by entity, but all parameters on the input form must conform to the limits set by the entity, and meet Entity Specification Grid requirements for input to ensure a successful record addition to a new table.   1. Conform to the underlying database, data type must match 2. Conform to ESG, special requirements for Primary Keys or other indexes 3. Must not be NULL   **Special Notes:**  None.  **Operation Outline:**  See Figure 5.1: ADD Method [Generic] Pseudo-code. |

**Generic UPDATE method Operations Specification.**

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_\*ENTITY\_NAME\*\_UO  Operation Description: Generic UPDATE operation for entity, will edit any record in the entity with the user’s new entry, capable of updating Primary Key.  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**  Input Form – same as generic add method  PK\_Index – Primary Key of record to update, so we know which record to fill with new information, ESG may require other restraints for this, unique to an entity.  Input Form containing entry parameters that vary according to the entity you want to add to. Adding to Countries would involve ‘Country Name’ and ‘Country Abbreviated’ parameters, but if adding a Gate or Terminal would require different parameters suited to that entity.  **Outputs:**  Completed Input Form showing that adding of object was successful with confirmation. This can be done through UI through dialog box or by presenting/ writing some sort of message to the display.  **Validations Rules:**   1. PK\_Index must be within bounds, if outside of bounds, use ADD method 2. Conform to the underlying database, data type must match 3. Conform to ESG, special requirements for Primary Keys or other indexes 4. Must not be NULL   **Special Notes:**  None.  **Operation Outline:**  See Figure 5.2: UPDATE Method [Generic] Pseudo-code. |

**Generic DELETE method Operations Specification.**

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_\*ENTITY\_NAME\*\_ZO  Operation Description: Generic DELETE method for any record in the entity, removes data and does not recycle Primary Key if the item removed is not at the top of the stack.  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**  PK\_Index – Primary Key of record to update, so we know which record to fill with new information, ESG may require other restraints for this, unique to an entity.  **Outputs:**  Confirmation of an object deletion or removal via Dialog Box or printout to display. Show object information similarly to Generic ADD method for the record that was removed.  **Validations Rules:**   1. Must be a value to conforms to Database and ESG diagram 2. Must be a value that is written onto already, cannot delete non-existent record 3. Must not be NULL   **Special Notes:**  Recommend running generic inquire first for locating the record you want to delete, that way you have maximum searching capability without directly inserting Primary Key index.  **Operation Outline:**  See Figure 5.3: DELETE (rm or zap) Method [Generic] Pseudo-code. |

**Generic INQUIRE method Operations Specification.**

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_\*ENTITY\_NAME\*\_IO / \_RO  Operation Description: Generic INQUIRE method for entity, allow common filtering parameters, and an optional String where user may enter an SQL ‘SELECT’ statement directly. Retrieves data from system.  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**  sub\_routine option – Number to indicate which of the sub-routine methods should be run to give user the appropriate input prompt.  Input Form – Consists of columns or properties of the entity you want to search for, how you will search for them (sub-routine option), and what filtering should be applied. How you search, can consist of Primary Key Index (PK\_Index) range or a combination of other indexes registered in the ESG for the desired entity that could be used to locate a record in a more user friendly way. Example in special notes.  \*\*filters – optional to refine the search (give me records like “property LIKE some\_value”)  **Outputs:**  Returns Array of results (or some object that can hold multiple data types filled by database)  **Validations Rules:**   1. Input Form Must not be NULL (opt 1 or 2 must be chosen) 2. If filters are applied they must be verified as valid SQL statement snippet (remember filters go at the end)   **Special Notes:**  It will be possible to query information with two possible main methods after detailing what columns or properties in the entity you would like to search for:  1. Directly give Primary Key index starting number and ending number (range)  2. Provide other indexes to look for, use combination of indexes for better results  Example: I have a record in some entity with PK\_ID=10 FNAME=”Ben” LNAME=”Steward” that I want to INQUIRE on, here are two different ways to call it.  LAM\_\*ENTITY\_NAME\*\_IO(int mode) where mode can be any integer that matches a valid sub-routine. Then a menu will be pulled up for user to input the information for the variables mentioned below, that then get passed to the sub routine.  INQUIRE\_1(selectedColumns[],startPK,endPK) or INQUIRE\_2(selectedColumns[],“COLUMN\_TO\_MATCH”[],”VALUE\_OF\_COLUMN”[])  I could also pass SQL fliters using this calling  INQUIRE(selectedColumns[],startPK,endPK, true, \*\*filters[])  INQUIRE(selectedColumns[],“COLUMN\_TO\_MATCH”[],”VALUE\_OF\_COLUMN”[], \*\*filters[])  **Operation Outline:**  See Figure 5.4: INQUIRE (Query) Method [Generic] Pseudo-code |

**Scheduling Report Operation Specification**

|  |
| --- |
| **Operation Biography:**  System: Lambert Air Management System (LAMS)  Subsystem: Flight Management  Operation Name: LAM\_SCHEDULING\_RO  Operation Description: Scheduling report operation that can run some specific searches/ retrieval of data that would not be covered in the generic methods. Search flights by arrival / departure datetime, filter for on time departures, filter for flights that are already departed, or search flights with specific crew member(s).  Operation Category: Mandatory  Complexity Rank: 8 of 10  Spec. Author: M. Lavin, A. Daur, J. Snider  Date: 3-22-2022  **Inputs:**   1. Flight Date   Or   1. Crew Member ID 2. Crew Member ID -- optional -- 3. […] -- optional --   Optional additional parameters   1. Is Flight On Schedule ( T/F/null ) 2. Is Flight Departed ( T/F/null )   **Outputs:**  Organized report of flights that match the user search criteria depending on their search method choice.    **Validations Rules:**  IF USING FLIGHT DATE SEARCH   1. Flight Date must conform to the database and Entity Specification Grid (ESG) 2. Flight Date must not be null   IF USING CREW MEMBER SEARCH   1. Crew Member ID must conform to the database and ESG 2. Crew Member ID must not be null 3. Number of inserted Crew Member IDs must be between 1 and maximum flight crew capacity ( eight members )   Optional parameters if left blank/null will return results list without checking these columns. Only use additional parameters if you are trying to narrow your search further.  **Special Notes:**  Operation has two searching modes, one for flight dates, and one for crew member searching, implemented via overloaded functions.  **Operation Outline:**  **START**  Initialize dynamic array ‘query\_results’;  Initialize boolean ‘continue\_’ = True;  **WHILE** (User wishes to continue)  Prompt user to select search mode : Date or Crew;  Accept user field, store in variable ‘response’;  **IF** (response == 0)  Call sub-routine LAM\_SCHEDULING\_RO\_01 & store in variable ‘query\_obj’;  **ELSE**  Call sub-routine LAM\_SCHEDULING\_RO\_02 & store in variable ‘query\_obj’;  **END-IF**  Append ‘query\_obj’ from sub-routine to ‘query\_results’;  Prompt user to check if they want to continue;  **END-WHILE**  Display query\_results to screen and return a copy;  **STOP**  **Outline For LAM\_SCHEDULING\_RO\_01 : Date Mode**  **START**  Prompt User for Flight Date;  Accept user input;  Prompt for optional parameters;  Accept possible user input;  Connect to Database;  Run Database query using user input to build SQL SELECT statement;  Execute and store built query into an array/list called ‘query\_container’;  Return ‘query\_container’;  **STOP**  **Outline For LAM\_SCHEDULING\_RO\_02 : Crew Mode**  **START**  Initialize boolean ‘continue\_’ = True;  Initialize integer ‘MAX’ = 8; // Because 8 crew members max for one flight  Initialize integer ‘count’ = 0;  **WHILE** (User wishes to continue && ‘count’ not equal ‘MAX’)  Prompt User for Flight Crew Member ID;  Accept user input;  Prompt User to enter another Crew Member ID;  **IF** (User does not wish to continue)  Set ‘continue\_’ = False;  **END-IF**  ‘count’ = ‘count’ + 1;  **END-WHILE**  Prompt for optional parameters;  Accept possible user input;  Connect to Database;  Run Database query using user input to build SQL SELECT statement;  Execute and store built query into an array/list called ‘query\_container’;  Return ‘query\_container’;  **STOP** |

Figure 5.1: ADD Method [Generic] Pseudo-code.

**START**

**WHILE** [User wishes to add record] **DO**

Accept key field(s);

Check existence of record in database;

**IF** (Record is absent) **THEN**

Accept non-key fields;

Validate non-key fields according to validation rules;

**WHILE** (Error exists) **DO**

Display non-key fields for update;

Display error messages;

Validate non-key fields according to validation rules;

**END-WHILE**

Display full record for confirmation

**IF** (User confirms) **THEN**

Write record to database;

**ELSE**

Inform user that record was not written;

**END-IF**

**ELSE**

Inform user that record already exists;

**END-IF**

Prompt user to quit, setting exit flag if yes;

**END-WHILE**

**STOP**

Figure 5.2: UPDATE Method [Generic] Pseudo-code.

**START**

**WHILE** (User wishes to continue) **DO**

Accept key field(s)

Check record existence in database

**IF** (Record exists) **THEN**

Retrieve record

Display non-key fields for possible update

Validate non-key fields based on validation rules

**WHILE** (Any error exists) **DO**

Re-display non-key fields for possible update

Display appropriate error message(s)

Validate non-key fields based on validation rules

**END-WHILE**

Display full record for confirmation

**IF** (Confirmation obtained) **THEN**

Update record in database

**ELSE**

Inform user that record was not updated

**END-IF**

**END-IF**

Check if user wants to quit, setting exit flag if necessary

**END-WHILE**

**STOP**

Figure 5.3: DELETE (rm or zap) Method [Generic] Pseudo-code.

**START**

**WHILE** (User wishes to continue) **DO**

Accept Key Field(s);

Check Record Absence or Existence in the primary file;

**IF** (Record Present) **THEN**

Retrieve Record;

Display full Record for confirmation;

**IF** (Deletion Confirmation Obtained) **THEN**

Update Audit Log Fields (with *current-values*)

Write New Record to file

Delete Record from the primary file;

**ENDIF;**

**ELSE** Inform the User that nothing was saved; **END-ELSE;**

**ENDIF;**

**ELSE** Display Message (‘Record does not exist’); **END-ELSE;**

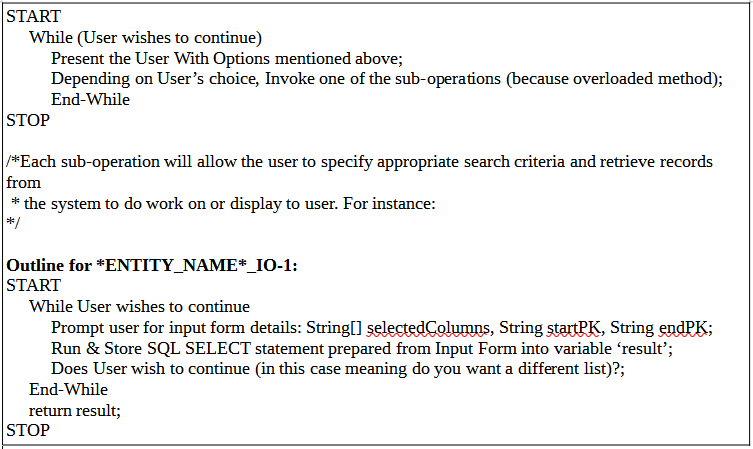
Check if User wishes to quit and set an exit flag if necessary;

**END-WHILE;**

Generate Edit-List;

**STOP**

Figure 5.4: INQUIRE (Query) Method [Generic] Pseudo-code



# 6.Other Design Considerations

## 6.1 Message and Help Specification

System documentation will be provided as a digital .PDF file, physical USB and paper copy will be provided after inquiry about the system as a new customer. We do not replace these but the .PDF file is always up at our website. Benefits of this are low cost universal media for distribution to prospective clients, both physical and digital for a one time fee. Because we have to know the client’s address, full name, etc. for physical delivery this discourages pranking or spoofing for free USB drives. Users can replace or download additional/updated copies for free at our website that is already paid for and hosted, so no additional costs for .PDF document. We will also have the ability to email it at any time for any reason to any client.

# 7.Summary & Concluding Remarks

This design specification has outlined the blueprint for a Flight Management System that will be robust and adaptable for any organization. It includes the following:

1. An overview of the system, including the problem definition, proposed solution, and some basic system architecture diagrams. The overview includes an Information Topology Chart (ITC) which helps us identify the key information entities in three comprehensive subsystems – the Port Information Subsystem (PIS), Arrival/Departure Subsystem (ADS), and the Airline/Aircraft Subsystem (AAD).
2. A database specification which includes both design and naming conventions, as well as an Entity Specification Grid (ESG) which covers all entities across each subsystem.
3. An operations specification which gives an entity operation specification (EOS) for each operation supported by the system.
4. A user interface specification which describes the interface through a User Interface Topology Chart (UITC).
5. A message specification which describes how system messages will be handled, as well as a help specification which describes the system of help messages which will be provided.

Future Updates/Add-ons:

1. Inclusion of an Infrastructure Subsystem would bring the FMS closer towards being a fully integrated airport management system. Management of resources such as fuel supplies, terminal amenities, and fleet vehicles would offer productivity increases and streamline identification of problems.

## 7.1 PERT and Project Schedule

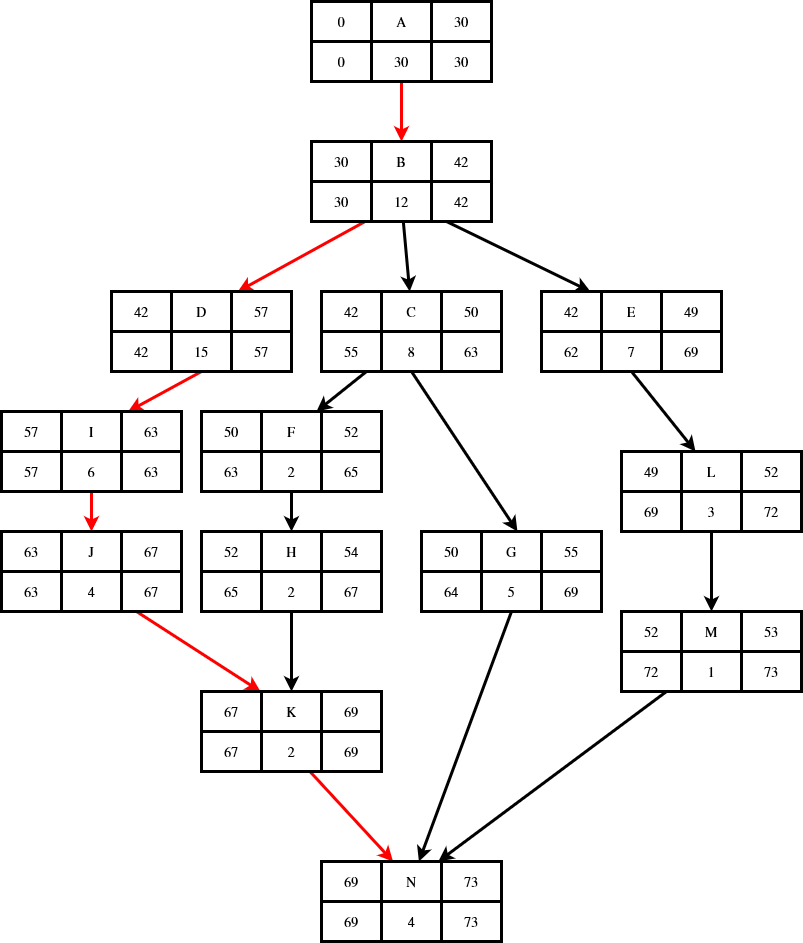
PERT is an acronym for Program Evaluation and Review Technique. We will be using this PERT diagram to manage our project, representing it graphically, showing important events that will occur in the project.

The table below shows the schedule for Lambert Air Management software project. Notice that except for the starting activity, A, every one of the activities has a predecessor. Every activity also includes an estimated time (here it is expressed in days).

**Figure 7.1 Schedule of Activities for Lambert Air Management System**

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Predecessor** | **Description** | **Estimated Time (days)** |
| A | -- | Design System Architecture | 30 |
| B | A | Design Operation Specifications | 12 |
| C | B | Design Control Operations | 8 |
| D | B | Design Modification Operations | 15 |
| E | B | Design Inquiry/Report Operations | 7 |
| F | C | Code Control Operations | 2 |
| G | C | Prepare System User Guide | 5 |
| H | F | Test Control Operations | 2 |
| I | D | Code Modification Operations | 6 |
| J | I | Test Modifications Operations | 4 |
| K | H, J | Test Control/Modification Operations | 2 |
| L | E | Code Report Operations | 3 |
| M | L | Test Inquiry/Report Operations | 1 |
| N | K, M, G | Integrations Test | 4 |

**Figure 7.2 PERT Diagram for Lambert Air Management System**



Diagram

Description automatically generated

Example PERT Diagram from Software Engineering: A Methodical Approach p.184