SMART FARM MONITORING SYSTEM

PROJECT REPORT

18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

HARDIKA (RA2111003010242)

SANDESH RAJBHAR (RA2111003010261)

KEERTHANA K (RA2111003010250)

JIIYAA JAISWAL (RA2111003010244)

Under the guidance of

Assistant Professor: Dr. G. Padmapriya

Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**Airport Check-In"** is the bonafide work of **HARDIKA (RA2111003010242)**, **SANDESH RAJBHAR (RA2111003010261)**, **KEERTHANA K (RA2111003010250)**, **JIIYAA JAISWAL (RA2111003010244)**

who undertook the task of completing the project within the allotted time.

Signature of the Guide

Dr. G. Padmapriya

Assistant Professor

Department of CINTEL,

SRM Institute of Science and Technology

Technology

Signature of the II Year Academic Advisor

Professor and Head

Department of CINTEL

SRM Institute of Science and Technology

About the course:-

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram

6.Create programs using object oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises

(Internal Mark Splitup:- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10

15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagram.
4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.
5. Draw relevant state charts and activity diagrams.
6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

ABSTRACT

This is an Airport security check-in in C++20. This project is a simple idea with a sophisticated execution.

Users that hop onto the platform may log in using existing credentials or create a new account with minimal effort. All credentials are stored and maintained on the platform's database for authentication.

Once logged in, the user is greeted by the main profile window from where they may

- Check existing bookings, made using the platform.
- Initiate a new booking using searching for flights given fixed criteria such as arrival and departure times, flight class, meal preferences, etc.
- Logout, and thereby exiting the platform

Issue Boarding pass:

So after security screening

Searching for flights:

Criteria filled in by the user are processed and cross-referenced with the airlines' flight databases to yield accurate results for onward and return trip flights. The user may choose from the list of flights according to their needs and proceed to checkout. If the user has opted for online payment, the bank transaction window shows up holding an array of payment options such as credit, debit, upi or wallet.

MODULE DESCRIPTION

It includes the whole area where passengers and boarding passes are controlled under certain security parameters before getting accepted to boarding. All passengers have to follow the instructions for airport safety and to have a safe flight.

After passing the airport security checkpoints, you need to deliver your baggage. If you have none you are free to head to the boarding area. Please prepare your boarding pass and identity documents before the flight staff control your documents to save time.

Use case diagram with explanation

Use case diagrams are used during requirements elicitation and analysis as a graphical means of representing the functional requirements of the system.

A use case is a sequence of transactions performed by a system that yields an outwardly visible, measurable result of value for a particular actor.

A use case diagram is a visual representation of the relationships between actors and use cases together that documents the system's intended behavior.

Actors: An actor represents whoever or whatever (person, machine, or other) interacts with the system.

In this system actors are: Employee, Manager, Admin.

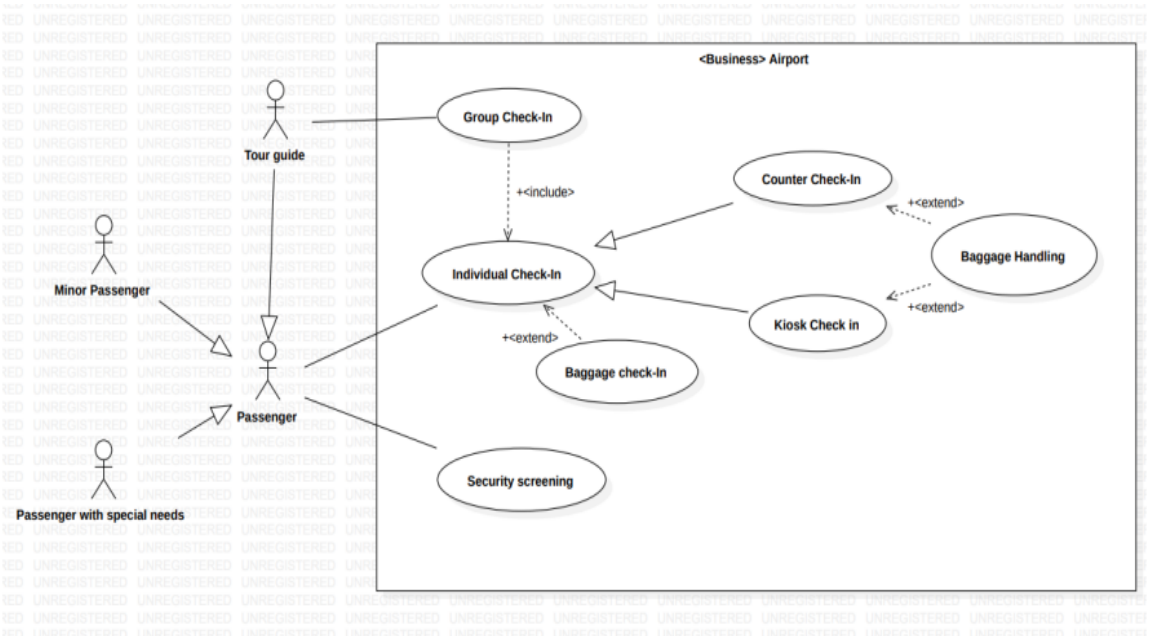
Use case: Use cases is the interaction between users and system. In this system:

Business actors are Passenger, Tour Guide, Minor (Child), Passenger with Special Needs (e.g. with disabilities), all playing **external roles** in relation to airport business.

Business use cases are Individual Check-In, Group Check-In (for groups of tourists), Security Screening, etc. - representing business functions or processes taking place in airport and serving the needs of passengers.

Business use cases Baggage Check-in and Baggage Handling extend Check-In use cases, because

passenger might have no luggage, so baggage check-in and handling are optional



Class diagram with explanation

Class diagrams are used in both the analysis and the design phases. During the analysis phase, a very high-level conceptual design is created. At this time, a class diagram might be created with only the class names shown or possibly some pseudo code-like phrases may be added to describe the responsibilities of the class.

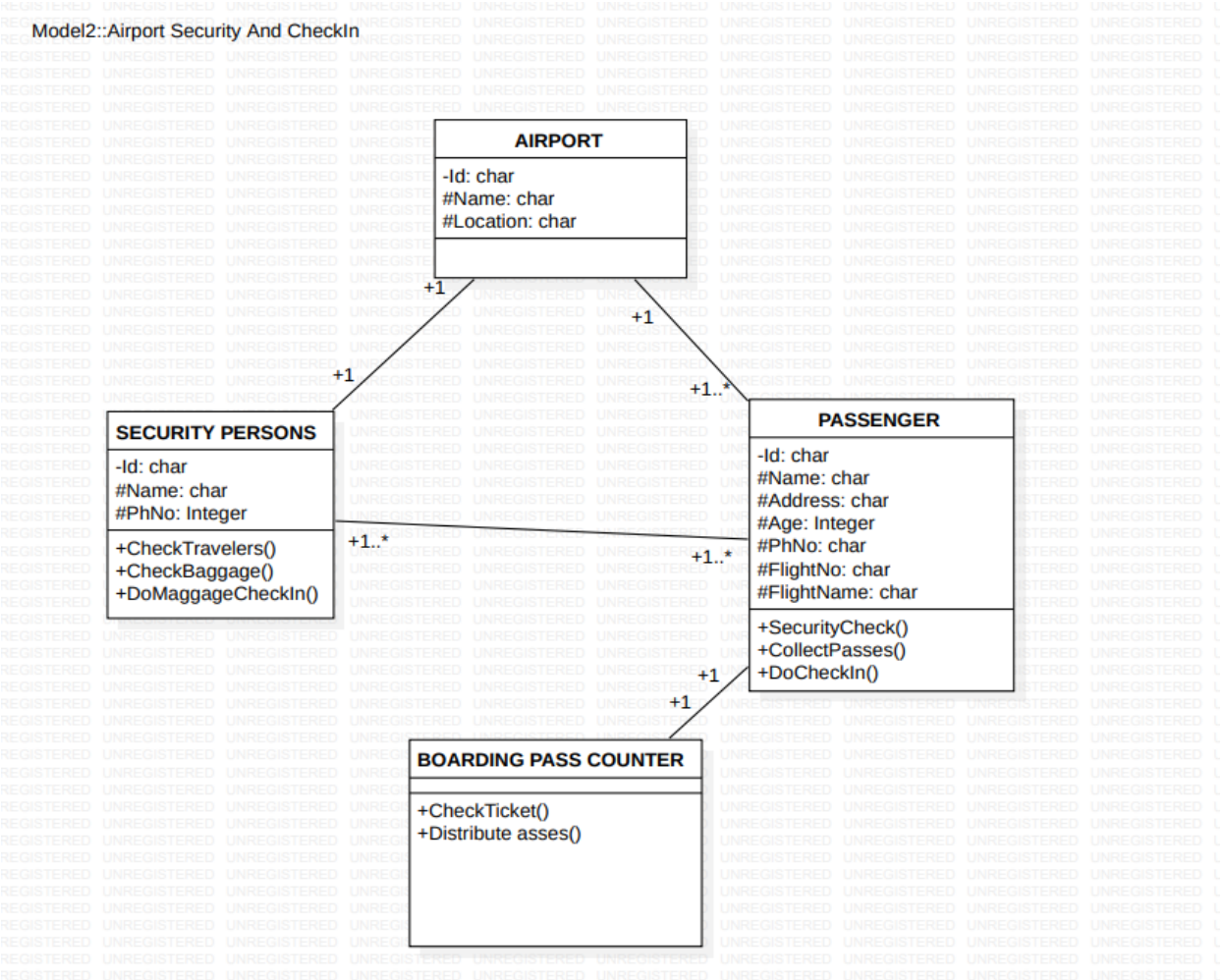
The class diagram created during the analysis phase is used to describe the classes and relationships in the problem domain, but it does not suggest how the system is implemented. By the end of the design phase, class diagrams that describe how the system to be implemented should be developed.

The class diagram describes the types of objects in a system and the various kinds of static relationships that exist among them.

A class is drawn as a rectangle with three components by horizontal lines. The top compartment holds the class name. The middle compartment has the attributes of the class and the bottom compartment holds a list of operations.

A static object diagram is an instance of a class diagram. It shows the detailed state of a system at a particular point of time.

Class interface notation is used to describe the externally visible behavior of a class; an operation with public visibility. The UML notation for an interface is a small circle with the name of the interface connected to the class. A class that require the operation in the interface may be attached to the circle by a dashed arrow.

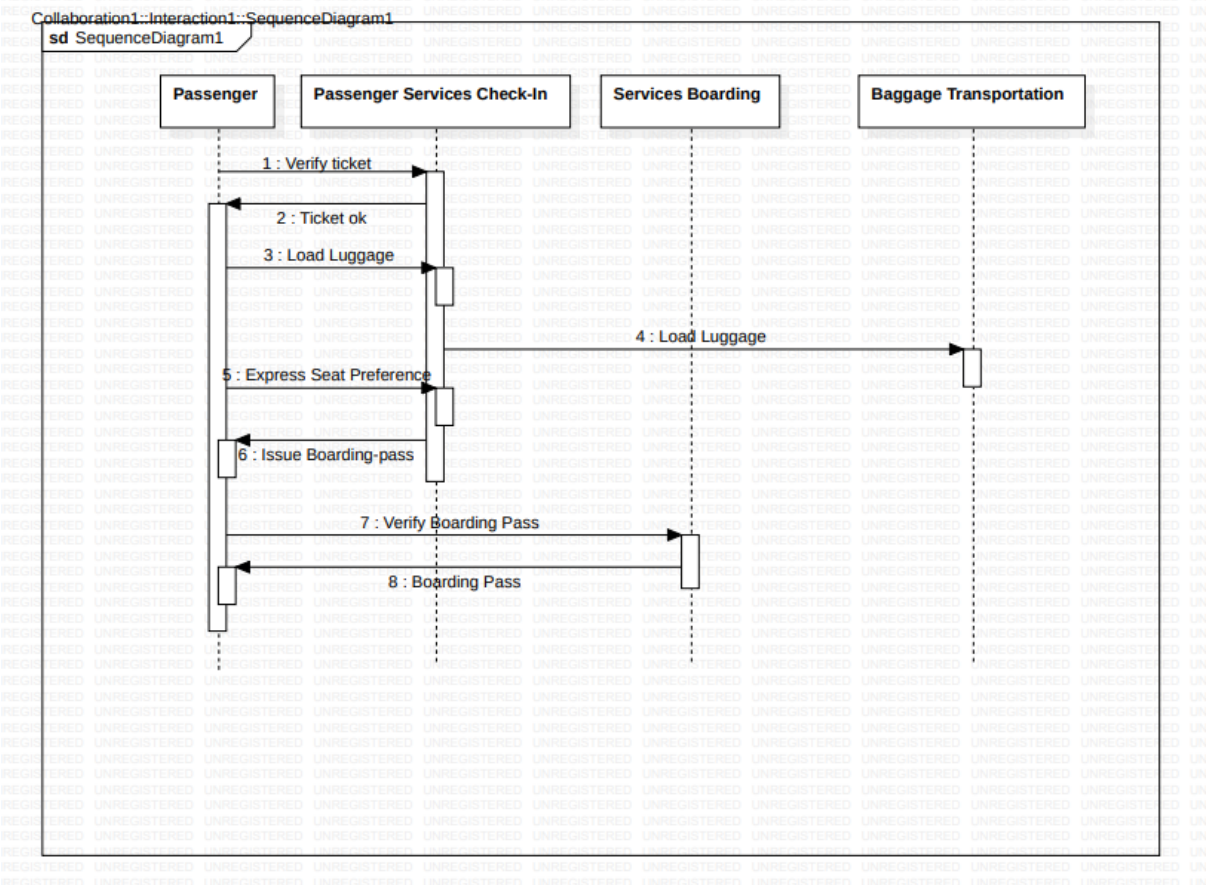


Sequence diagram with explanation

A sequence diagram shows an interaction between the system and its environment arranged in a time sequence. It shows the objects participating in the interaction by their lifelines and the messages they exchange, arranged in a time sequence. A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. The vertical line is called the object's lifeline. The lifeline represents the object's existence during the interaction. An object is shown as a box at the top of a dashed vertical line.

Each message is represented by an arrow between the lifelines of two objects. Each message is labeled with the message name. The label can also show self-delegation, a message that an object sends to itself, by sending the message arrow back to the same lifeline.

In a sequence diagram, objects are shown in columns, with their object symbol on the top of the line. Similar to the class diagram, the object name appears in a rectangle. If a class name is specified, it appears before the colon. The object name always appears after a colon (even if no class name is specified). If an external actor initiates any interaction, the stick figure can be used rather than a rectangle.



Collaboration diagram with explanation

A collaboration diagram, also known as communication diagram represents a collaboration, which is set of objects and interaction, which is a set of messages exchanged among the objects to achieve a desired outcome. In a collaboration diagram the sequence of message is indicated by numbering the message. Objects are show as rectangle with naming labels inside. The relationship between the object are show as line connecting with rectangles. The message between objects is show as arrows connecting the relevant rectangles along with labels that define the message sequencing.

A collaboration diagram is similar to sequence diagram but the message are in number format. In a collaboration diagram sequence diagram is indicated by numbering the message.

A collaboration diagram is also known as communication diagram or interaction diagram. A sophistication modeling tool can easily convert a

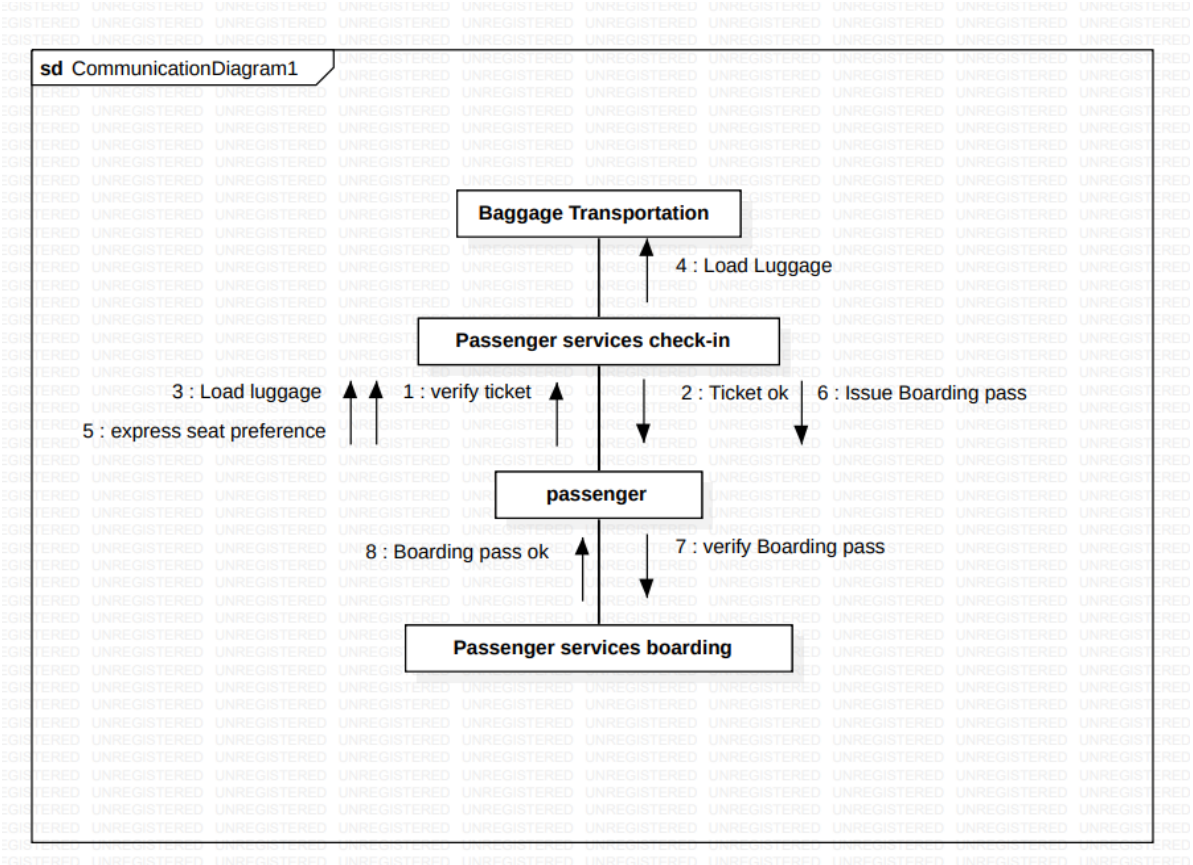
collaboration diagram into a sequence diagram and the vice versa. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time.

The main advantage of interaction diagram (both collaboration and sequence) is simplicity. We easily can see the message by looking at the diagram. The disadvantage of interaction diagram is that they are great only for representing a single sequence process; they begin to break down when you want to represent condition looping behaviour.

- Collaboration diagrams describe how objects interact.

- Collaboration diagrams focus on space. • This means that the relationships (links) between the objects (in space) are of particular interest.

- A collaboration diagram is isomorphic (to a degree) with a sequence diagram.



State chart diagram with explanation

A State chart Diagram (also called a state diagram) shows the sequence of states that an object goes through during its life in response to outside stimuli and

messages/events. The state is the set of values that an describe an objects at a specific point in time and is represented by state symbols and the transitions are represented by arrows connecting the state symbols.

A state is represented as a rounded box, which may contain one or more compartments. The name compartment holds the optional name of the state. States without names are “anonymous” and all are distinct.

The internal transition compartment holds a list of internal actions or activities performed in response to events received while the object is in the state, without changing states.

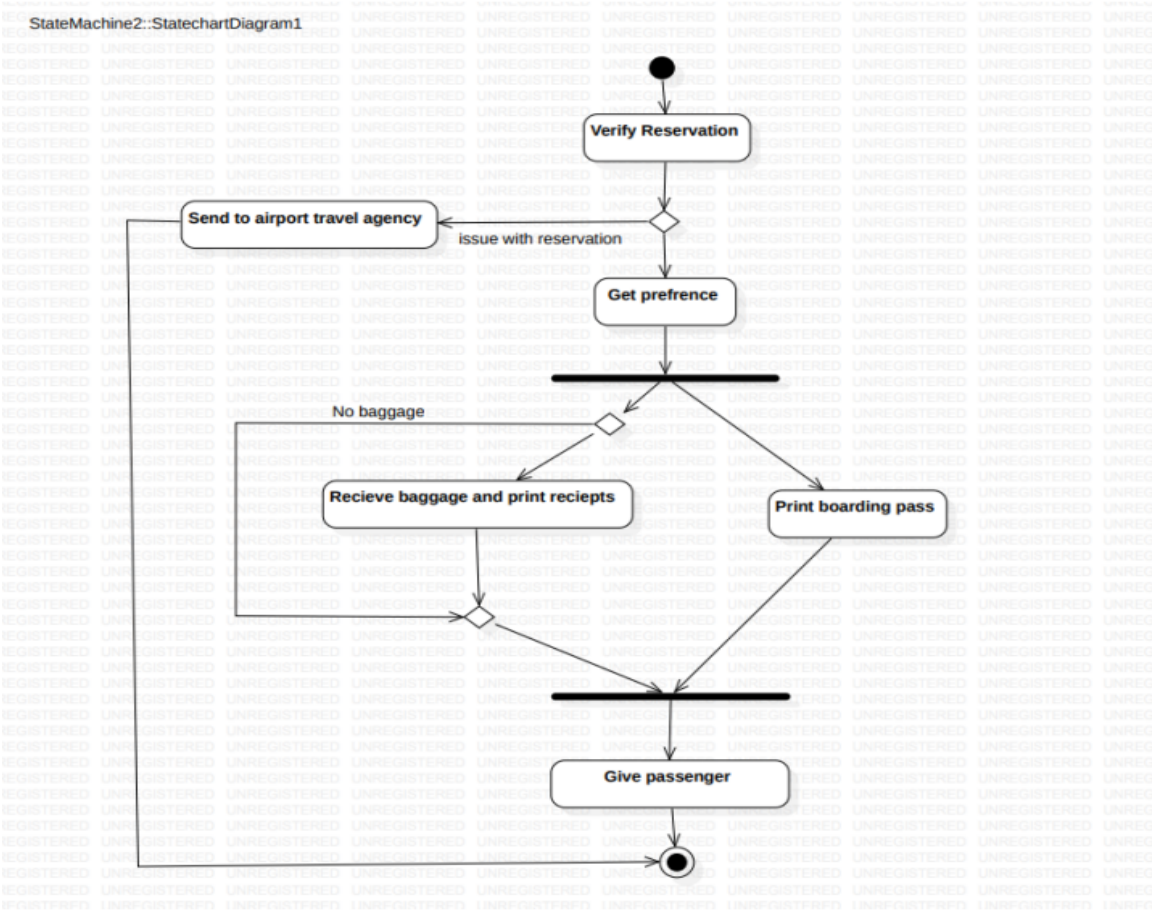
The transition can be simple or complex. A simple transition is a relationship between two states indicating that an object in the first state will the second state and perform certain action when a specific events occurs. A complex transition may have multiple source and target states.

The states chart diagram models the dynamic behavior of individual classes or any other kind of object. It show the sequences of states, events, and actions.

States represents a condition or situation during the life of an object during which it satisfies some condition or waits for some events. Start states shows the beginning of workflow. End states represent the final or terminal states.

The state chart diagram contains the states in the rectangle boxes and starts in indicated by the dot and finish is indicated by dot encircled. The purpose of state chart diagram is to understand the algorithm in the performing method.

State chart Diagram



Activity diagram with explanation

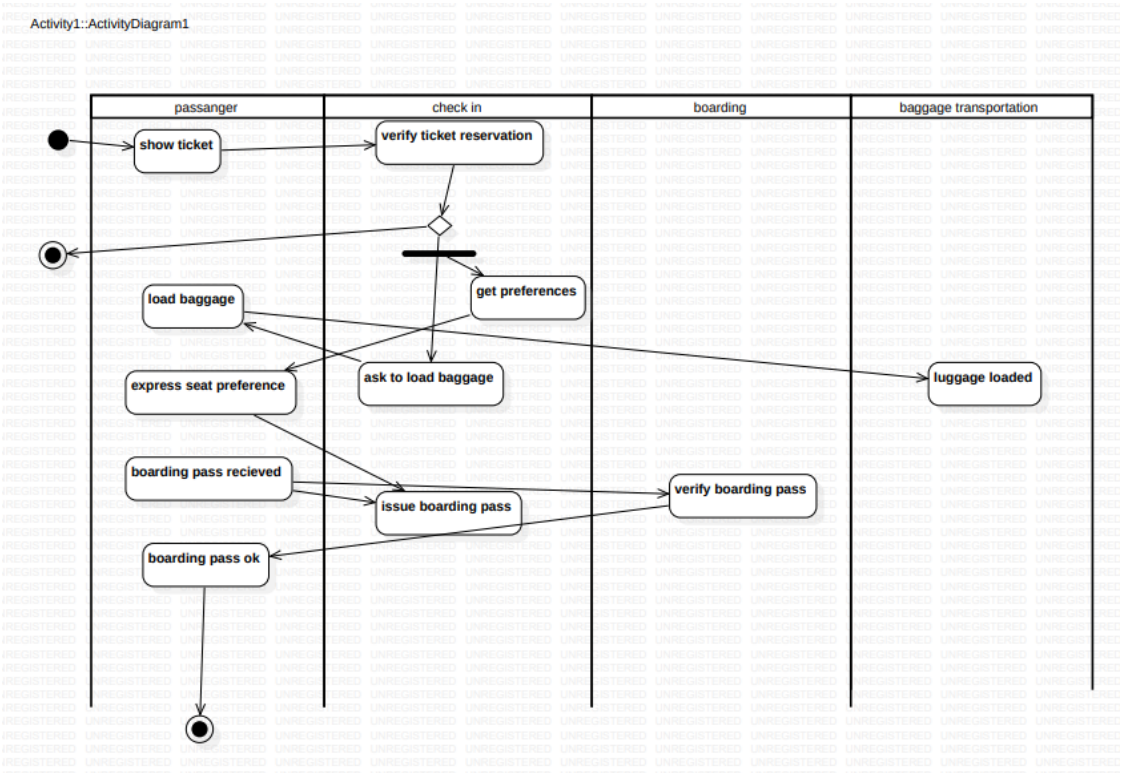
An activity diagram is used for modeling the sequence of activities in a process. It is a variation or special case of a state machine, in which the states are activities representing the performance of operations and transitions are triggered by the completions of operation. The purpose of an activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. An activity represent the performance of task or duty in a work flow. It is show as a rounded box containing the same of operation.

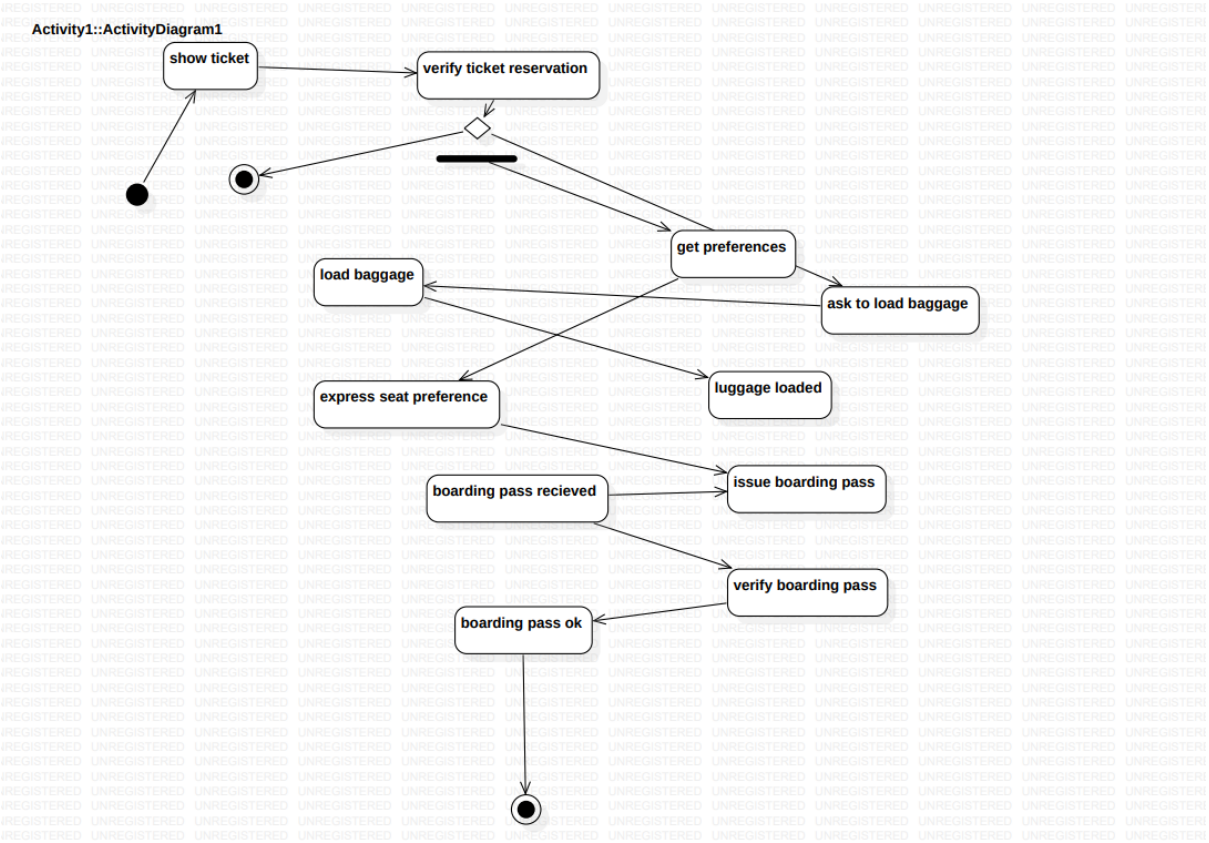
Several transition with different conditions imply a branching off of control. The concurrent control is represented by multiple arrows leaving a synchronization bar, which is represented by a short thick bar with incoming and outgoing arrows.

Joining concurrent control is expressed by multiple arrows entering the synchronization bar.

Activity and state diagram express a decision when condition are used to indicate different possible transition. The representation used for a decision is the diamond shape, with one or more incoming arrows and two or more outgoing arrows, each labeled by a distinct guard condition.

Action may be organized into swim lanes, each separated from neighbouring swim lanes by vertical solid lines on both sides. Each swim lane represents responsibility for [part of the overall activity.



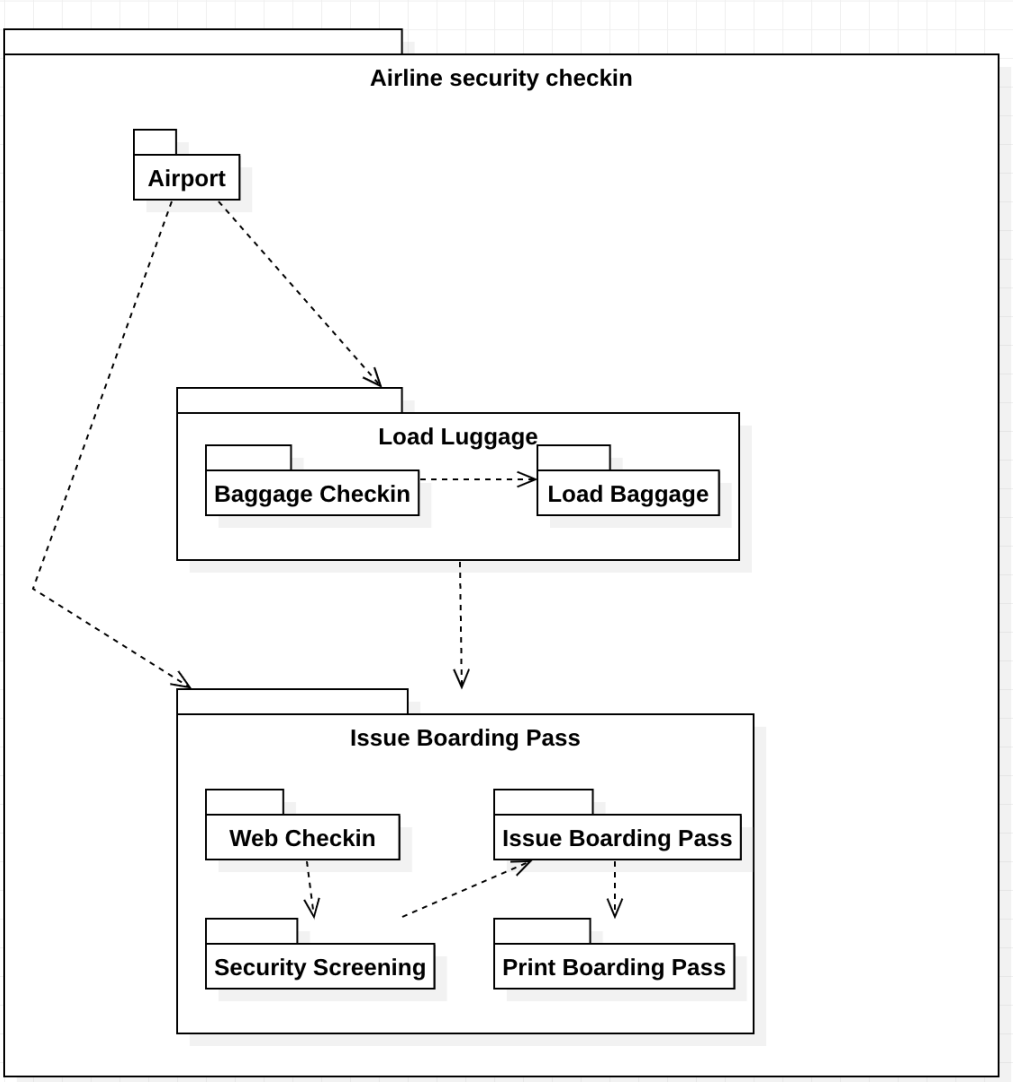


Package diagram with explanation

A package is a grouping of model elements. Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. Packages themselves may contain other packages. A package may contain both subordinate packages and ordinary model elements. All UML model elements and diagrams can be organized into packages.

A package is represented as a folder, shown as a large rectangle with a tab attached to its upper left corner. If contents of the package are not shown, then the name of the package is placed within the large rectangle. If the contents of the package are shown, then the name of the package may be placed on the tab. The contents of the package are shown within the large rectangle.

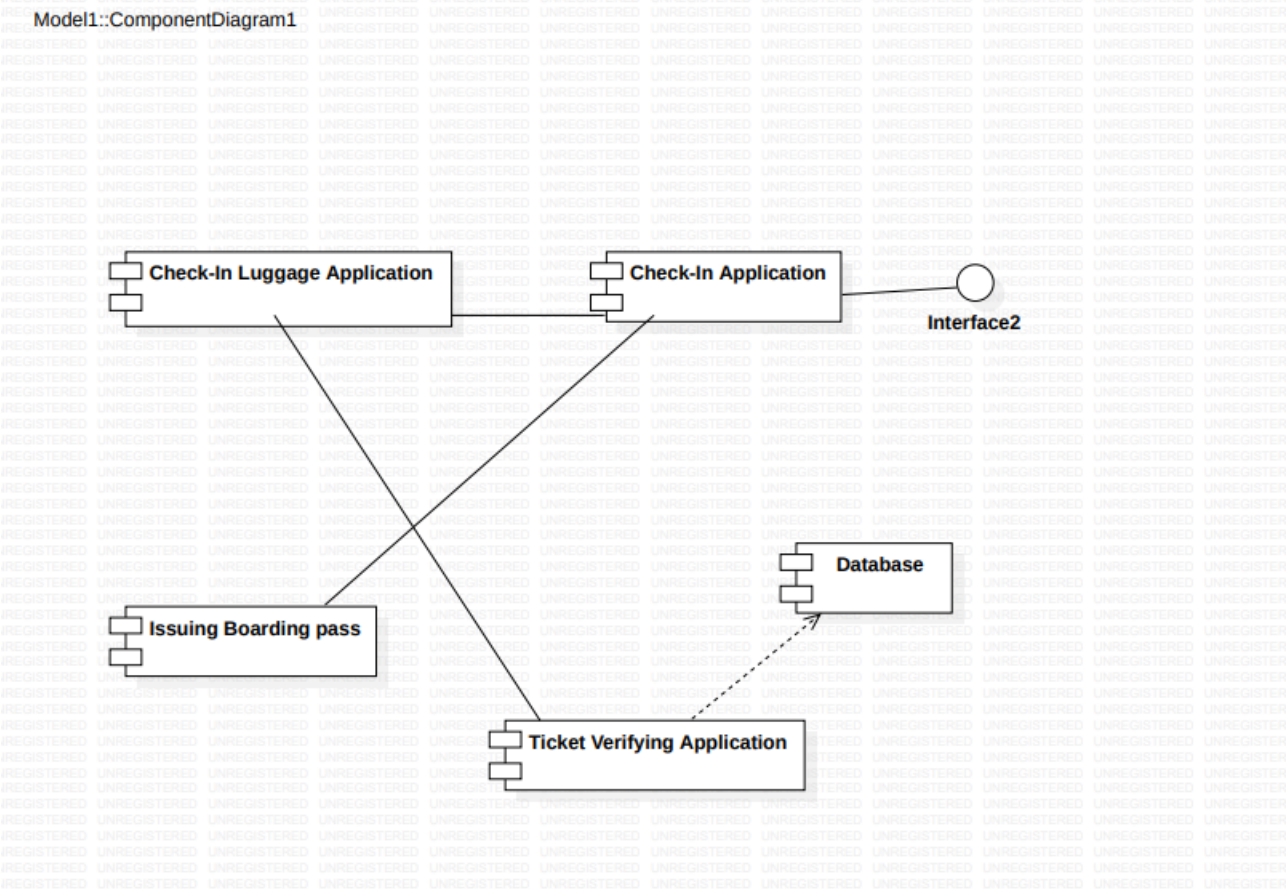
Packages can be used to designate not only logical and physical groupings but also use-case groups. A use-case group, as the name suggests, is a package of use cases.



Component diagram with explanation

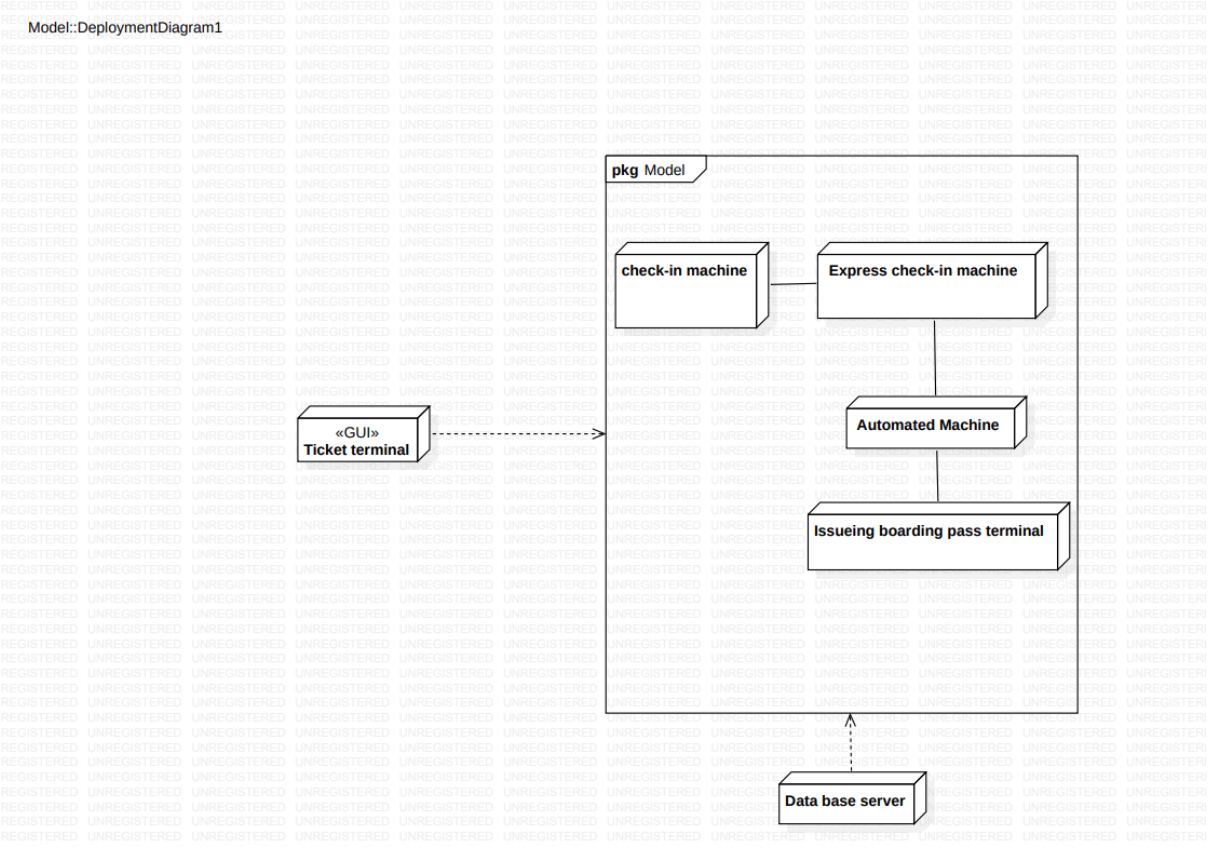
Component diagram is a special kind of diagram in UML. It does not describe the functionality of the system but it describes the components used to make those functionalism. So, the component diagrams are used to visualize the physical components (such as source code, executable program, user interface) in the system. A component diagram is a graph of the design's components connected by the dependency relationships.

A component is represented using a rectangle box, with two rectangles protruding (sticking out) from the left side. A dependency is used to model the relationship between two components. The notation for a dependency relationship is a dotted arrow, pointing from a component to the component it depends on.



Deployment diagram with explanation

A Deployment diagram is a graph of nodes connected by communication association. Nodes may contain component instances, which means that the component lives or runes at that node. A deployment diagram in the UML serves to model the physical deployment of artifacts on deployment targets. It is represented by three-dimensional box. Dependencies are represented by communication association. The basic element of a deployment is a node. It is of two types:



Conclusion

Hence the above are the representation of Airport security Check-In system using UML Diagram. This gives clear idea about working of Airport security Check-In system. in all processes and the things that can be done by the system.

References

https://en.wikipedia.org/wiki/Airport_check-in

https://staruml.io/