# - Airport Management Software System - (AMSS)

SUBMITTED BY:

David Urban
*david.urban@student.univaq.it*

Kacper Pudelko
*kacperhenryk.pudelko@student.univaq.it*

## Introduction

### Purpose of the Document

This document serves to outline the assignment in detail, highlighting its objectives and specifications. It also includes explanations of the UML diagrams developed for the assignment, offering additional insight and promoting a deeper understanding.

## Project Overview

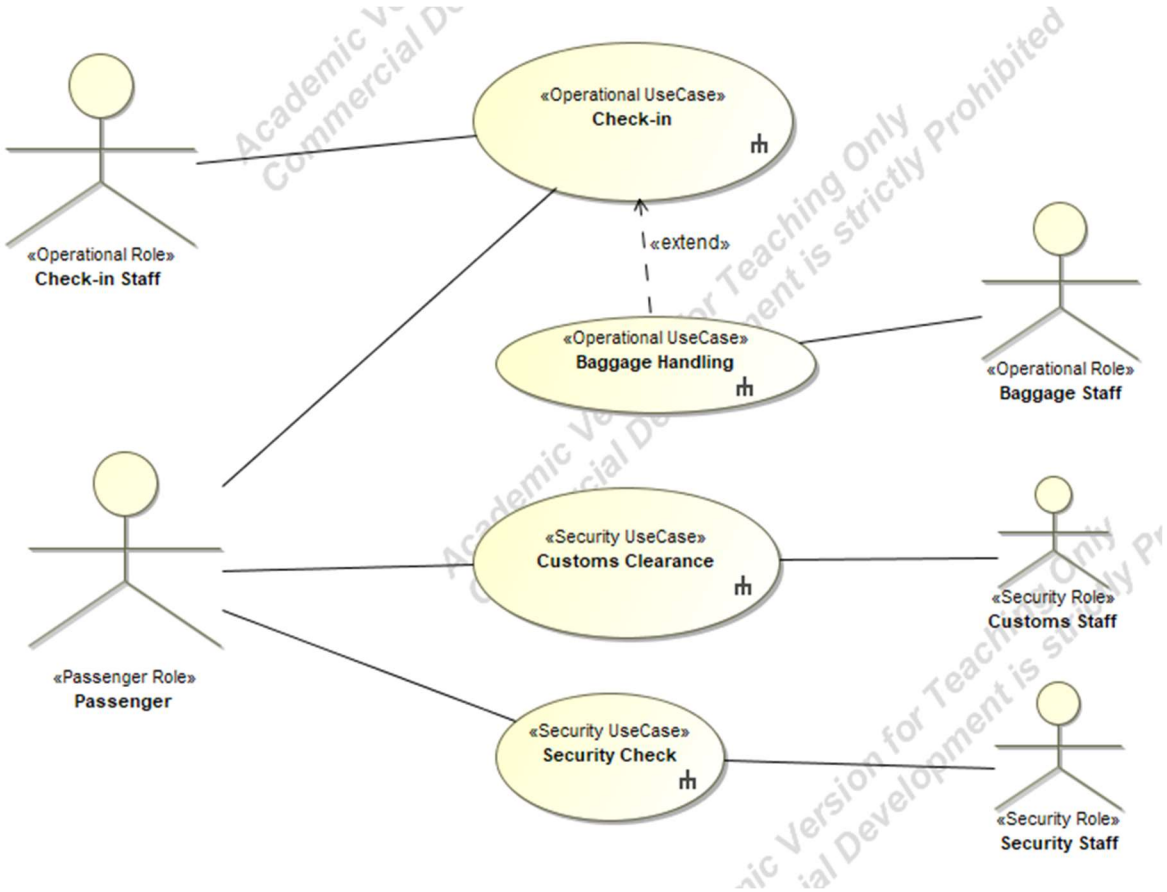
The primary functions of our Airport Management System (AMS) are centered exclusively on landside operations. These operations include:

- Check-in System
- Baggage Handling
- Customs Clearance
- Security Check

Our system is designed specifically to manage and streamline these landside processes, ensuring efficient operation within the airport environment.

## Use Case Diagram

The use case diagram highlights the key interactions between five distinct actors and the Airport Management System (AMS). Each actor has been assigned a stereotype that specifies their role within the system, along with the type of tasks they perform. These roles have varying access levels, dictating the kind of data they can access and manage, which is further elaborated in the Profile Diagram section of this document.




**The identified actors include:**

Passenger Role – Represents individuals utilizing the airport services.

Check-in Staff – Responsible for operational tasks associated with passenger check-in processes.

Baggage Staff – Manages the baggage handling systems.

Customs Staff – Oversees customs clearance tasks, ensuring compliance with regulations.

Security Staff – Handles security checks to ensure the safety of passengers and airport operations.

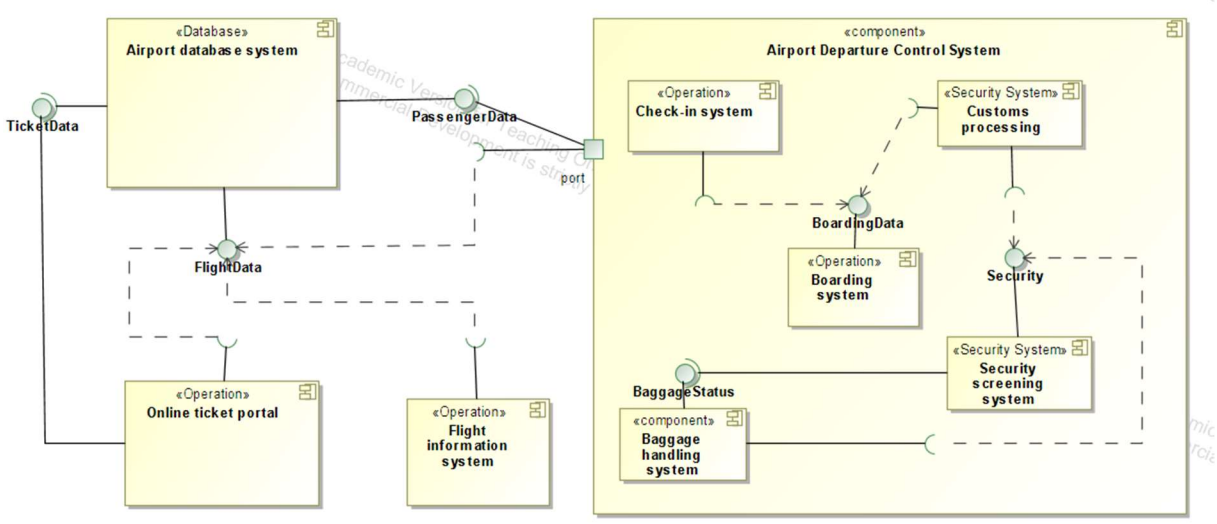
**The use cases in this diagram are divided into two main categories:**

Operational Use Cases – Such as "Check-in" and "Baggage Handling," which are directly related to passenger and luggage management.

Security Use Cases – Such as "Customs Clearance" and "Security Check," focusing on security and regulatory compliance.

## Component Diagram

This component diagram identifies and organizes the key systems involved in the Airport Management System (AMS). The system is composed of multiple interconnected components that collectively handle landside operations. The interaction between components is facilitated through interfaces, represented using lollipop notation, and certain interfaces are exposed for external actor interactions.




Key components include:

1. Airport Database System:

- Centralized repository for storing data such as Passenger Data and Flight Data.
- Provides access to critical information required by other components such as the check-in system and online ticket portal.

2. Online Ticket Portal:

- Facilitates ticket booking and validation.
- Interacts with the Airport Database System to retrieve and store ticket-related data.

3. Flight Information System:

- Manages and provides real-time updates on flight schedules.
- Shares Flight Data with other systems to ensure accurate and synchronized operations.

4. Check-in System:

- Handles passenger check-in operations, ensuring smooth processing of Passenger Data and linking it to Boarding Data.

5. Baggage Handling System:

- Manages the routing, tracking, and status updates of baggage throughout the airport.
- Interfaces with the Check-in System to ensure baggage is linked to the correct passenger.

6. Boarding System:

- Manages the boarding process by utilizing Boarding Data.

- Ensures passengers are directed to the correct gate and updates the system with real-time boarding status.

7. Security Screening System:

- Responsible for screening passengers and luggage to maintain security standards.
- Works in tandem with other components to verify passenger and baggage data.

8. Customs Processing System:

- Manages customs clearance for passengers.
- Utilizes Passenger Data and integrates with other systems to ensure a smooth customs process.

System Interactions:

The components are interconnected to ensure seamless information flow across the AMS. For instance:

- The Airport Database System provides a backbone for data integration and sharing between components.
- The Check-in System and Baggage Handling System work closely to reconcile passenger and baggage data.
- Security and customs systems utilize data from both the check-in and database systems to perform their respective roles effectively.

## Profile Diagram

In the profile diagram, we have defined specific stereotypes, tagged values, and constraints that are applied to elements in other diagrams (e.g., components, actors, and use cases) to meet unique requirements. The profile introduces tailored stereotypes that reflect the roles, data access levels, and types of interactions in the system.

Roles and Access Levels :

Below is a table describing the roles and their access to various types of data :

- Passenger Role - Access to passenger data.
- Operational Role – Access to operational data.
- Security Role – Access to security data.

Use Cases :

The system includes four types of use cases, each associated with specific actors and interactions :

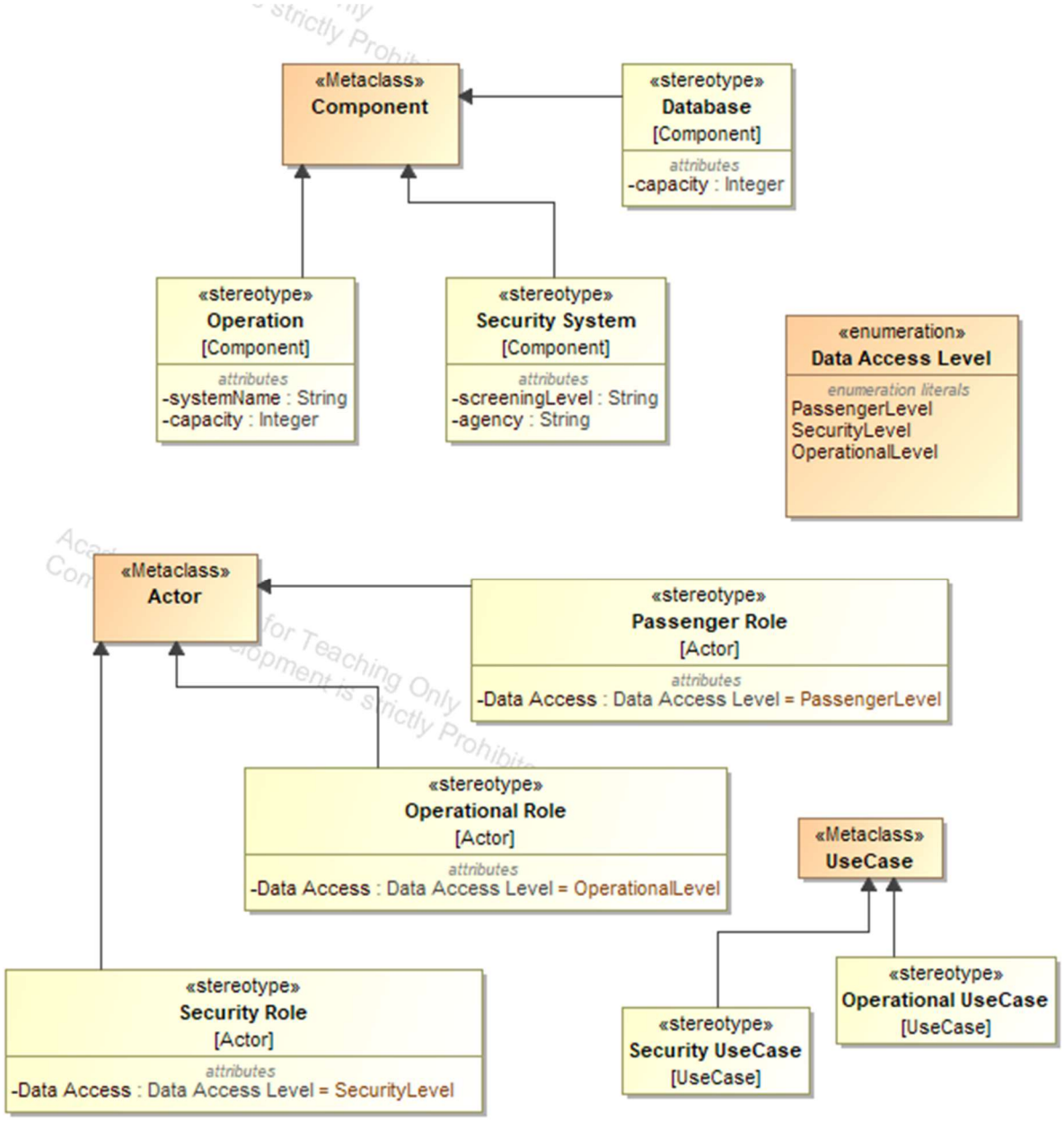
- Standard Use Case- Covers basic operations like Check-in and Passenger Boarding.
- Operational Use Case- Includes tasks for real-time operations, such as Flight Status Updates.
- Security Use Case- High-security operations, such as Passenger Screening and Baggage Screening

Enumeration Literals :

The Data Access Level enumeration includes the following values :

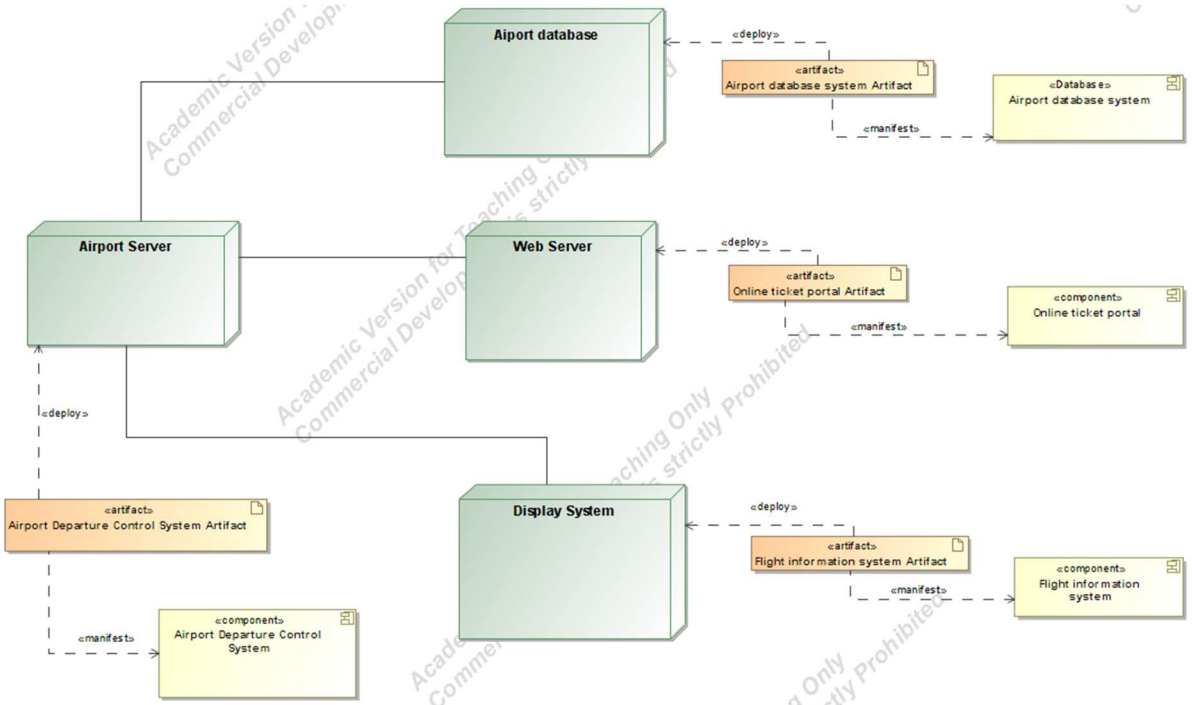
- PassengerLevel
- OperationalLevel
- SecurityLevel

These literals define the specific levels of access available to actors in the system.




## Deployment Diagram

The system is deployed across multiple infrastructure nodes to ensure seamless and efficient airport operations. The deployment includes server nodes for handling various backend processes and client nodes for user-facing operations.




Nodes and Artifacts :

Server Nodes:

- Airport Database:

  Hosts the Airport Database System Artifact, which stores critical data such as flight schedules, passenger information, and ticket details.

- Airport Server:

  Facilitates the core operations of the airport, deploying the Airport Departure Control System Artifact that manages processes like check-in, baggage handling, and security operations.

- Web Server:

  Hosts the Online Ticket Portal Artifact, which handles online bookings and ticket management. This artifact manifests the Online Ticket Portal Component for external user interaction.

- Display System:

  Deployed with the Flight Information System Artifact, which manages the Flight Information System Component to provide real-time flight updates via displays.

- Artifacts:

  Each artifact corresponds to a component within the system and is deployed on specific nodes for optimal performance and separation of responsibilities.

The Airport Departure Control System Artifact interacts with other artifacts and components, ensuring synchronization of operations across the system.

System Interactions:

- The Airport Database System acts as the central data source, ensuring other nodes like the Check-in System, Security System, and Boarding System have access to accurate and real-time information.
- The Web Server supports external user access for online ticketing while remaining integrated with the central Airport Database.
- The Display System ensures passengers receive timely updates about their flights through its integration with the Flight Information System Component.

## Sequence Diagram: Airport Check-In Process

This sequence diagram outlines the interactions between various actors and systems involved in the airport check-in process. It focuses on the flow of data and operations to ensure smooth passenger handling and boarding pass generation.

Actors and Components

- Passenger Role: The individual providing the ticket for check-in.
- Operational Role: Check-In Staff: Responsible for initiating and managing the check-in process.
- Check-In System: The system that processes passenger and flight data.
- Airport Database System: Stores passenger, flight, and baggage-related data.
- Boarding System: Generates and updates the boarding pass for passengers.

Flow of Interactions

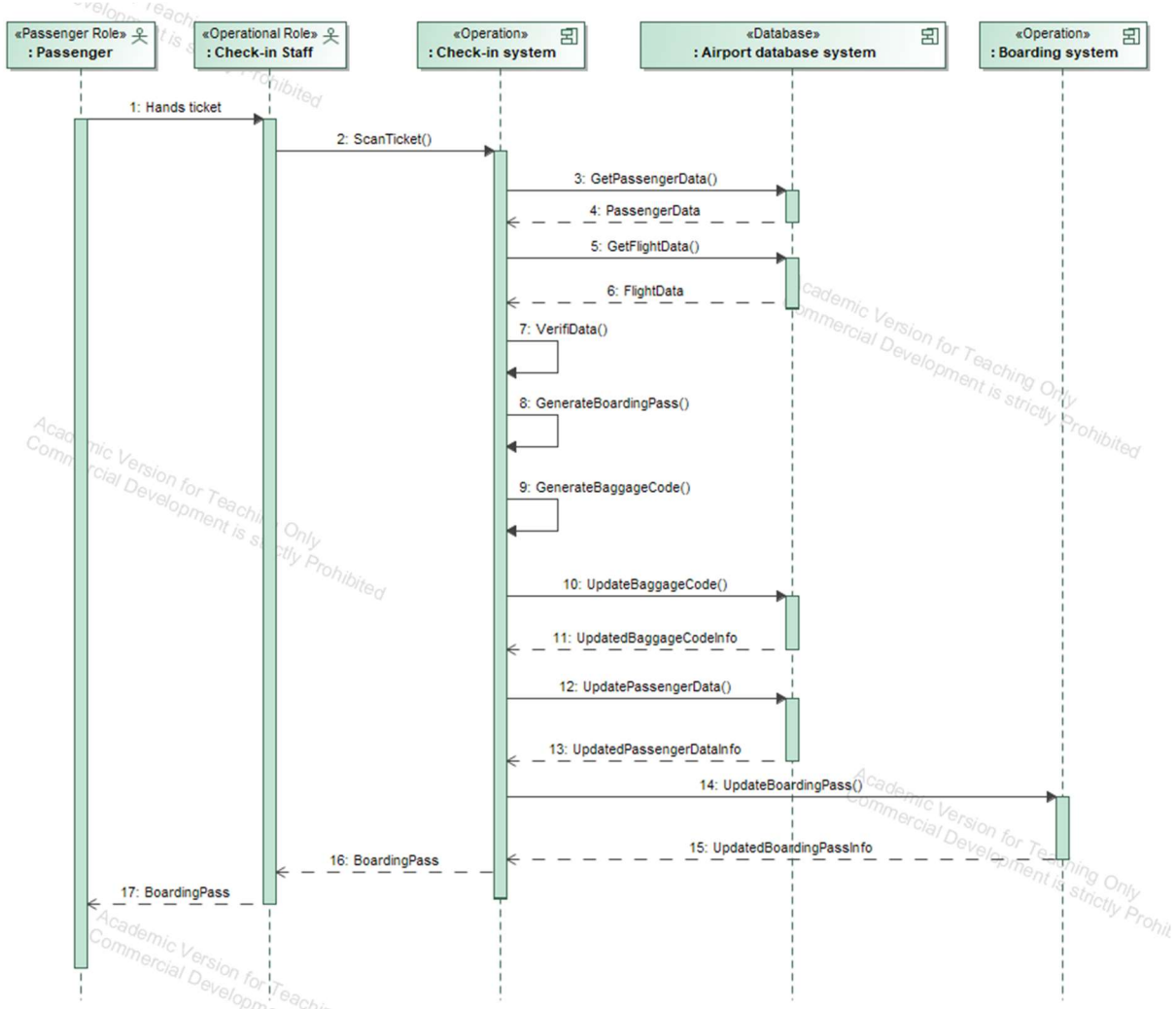
1. Passenger Hands Ticket:
   - The passenger hands over their ticket to the check-in staff.
2. Ticket Scanning:
   - The check-in staff scans the ticket in the Check-In System.
3. Data Retrieval:
   - The Check-In System retrieves passenger data (GetPassengerData()) and flight information (GetFlightData()) from the Airport Database System.
4. Data Verification:
   - The system verifies the validity of passenger and flight data (VerifyData()).
5. Boarding Pass Generation:
   - The Check-In System generates the boarding pass (GenerateBoardingPass()).
6. Baggage Code Generation:
   - The system generates a unique baggage code (GenerateBaggageCode()).
7. Database Updates:
   - The Check-In System updates the baggage code (UpdateBaggageCode()) and passenger data (UpdatePassengerData()) in the Airport Database System.
8. Boarding Pass Update:
   - The updated boarding pass information (UpdateBoardingPass()) is shared with the Boarding System.

9. Boarding Pass Issuance:
    - The Check-In System provides the final boarding pass to the passenger.

Key Operations

- GetPassengerData() and GetFlightData(): Retrieve essential data for processing.
- GenerateBoardingPass() and GenerateBaggageCode(): Ensure seamless passenger and baggage tracking.
- UpdateBaggageCode() and UpdatePassengerData(): Keep the system data synchronized.

At the end of the process, the passenger receives a boarding pass, ready for further steps in the travel process.




## Sequence Diagram: Baggage Handling

This sequence diagram illustrates the interactions between the Baggage Staff, systems, and databases involved in the baggage handling and security screening process. It highlights the process of validating baggage, scanning for security purposes, and updating baggage status.

Actors and Components

- Baggage Staff : Responsible for initiating baggage scanning and ensuring secure handling.
- Baggage Handling System : Manages baggage validation and scanning processes.

- Airport Database System : Stores baggage-related data, including unique baggage codes.
- Security Screening System : Receives alerts for potential security threats during baggage scans.

Flow of Interactions

1. Baggage Staff Scans Baggage Code:
   - The baggage staff initiates the process by scanning the baggage code into the Baggage Handling System (ScanBaggageCode()).
2. Retrieve Baggage Code:
   - The Baggage Handling System queries the Airport Database System to fetch the relevant baggage code (GetBaggageCode()).
3. Baggage Code Validation:
   - The Baggage Handling System validates the retrieved baggage code against the scanned input (ValidateCode()).
4. Baggage Scanning:
   - The Baggage Handling System scans the baggage for security purposes (ScanBaggage()).
5. Optional Security Alert:
   - If the scan results indicate a potential threat (ScanResults == alert), and SecurityAlert() is triggered by the Security Screening System to flag the issue.
6. Update Baggage Status:
   - The Baggage Handling System updates the baggage status in the Airport Database System (UpdateBaggageStatus()) to reflect the completed security scan.
7. Provide Final Scan Result:
   - The updated scan result (ScanResult) is provided to the Baggage Staff for further processing.

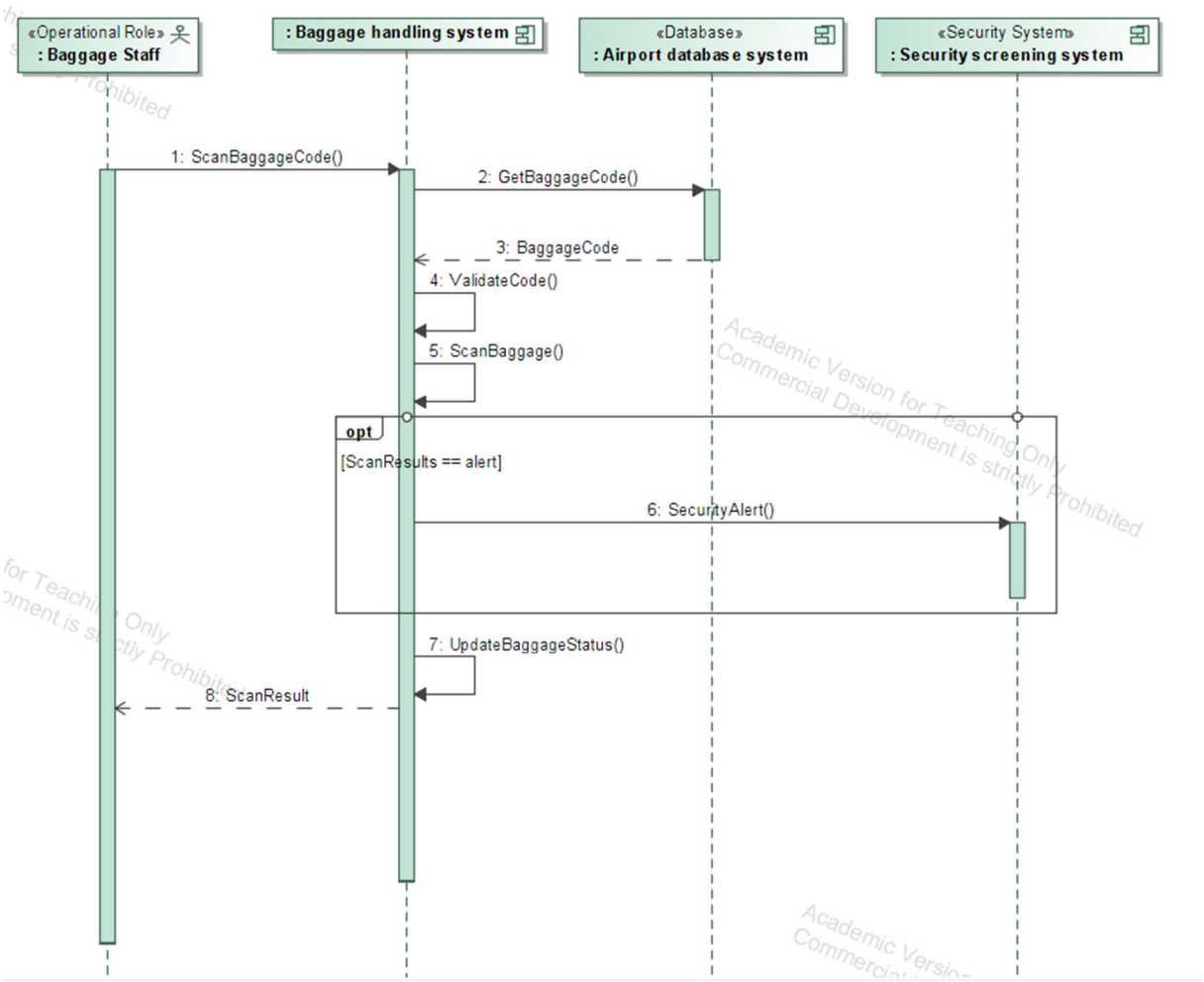
Key Operations

- ScanBaggageCode(): Ensures the system captures the unique baggage identifier accurately.
- GetBaggageCode(): Retrieves the associated baggage data from the database.
- ValidateCode(): Confirms that the baggage code is valid and matches system records.
- ScanBaggage(): Conducts a comprehensive security scan of the baggage.
- SecurityAlert(): Triggers an alert if the scan results indicate any security concerns.
- UpdateBaggageStatus(): Synchronizes the updated status of the baggage in the database.

Optional Flow

- Security Alert Scenario: If a security threat is detected during the baggage scan, the system issues an alert. This flow is conditional and does not occur for all baggage scans.

At the end of the process, the baggage status is updated in the system, ensuring it is secure and ready for further handling. Any potential security concerns are flagged promptly for investigation.




## Sequence Diagram: Security Screening Process

This sequence diagram illustrates the interactions between various actors and systems involved in screening passengers during airport security checks. It ensures that passengers and their baggage are verified before being allowed to proceed to their boarding gate.

Actors and Components

- Passenger: Represents the individual undergoing security screening.
- Security Staff: Conducts the passenger screening and manages the interaction between systems for verification.
- Security Screening System: Handles the security verification process for passengers.
- Database: Airport Database System: Stores and retrieves data about passengers, boarding passes and baggage statuses.
- Baggage Handling System: Ensures baggage-related details are consistent with the passenger's boarding pass.

Flow of Interactions

1. Passenger Provides Boarding Pass:
   - The Passenger hands their boarding pass to the Security Staff, initiating the verification process (ScanBoardingPass()).
2. Retrieve Passenger Data:

- The Security Screening System communicates with the Airport Database System to fetch the passenger's data (GetPassengerData()).

3. Verify Boarding Pass:
   - The Security Screening System verifies the retrieved passenger data against the boarding pass details (VerifyBoardingPass()).
4. Retrieve Baggage Status:
   - The Security Screening System requests the baggage status from the Airport Database System (GetBaggageStatus()) to ensure that all baggage is accounted for and properly screened.
5. Direct Passenger to Screening:
   - Upon successful verification, the Passenger is directed to undergo physical screening (DirectToScreening).
6. Undergo Screening:
   - The Passenger proceeds with the screening process managed by the Security Staff (UndergoScreening).
7. Scan Passenger:
   - The Security Screening System performs the physical scanning (ScanPassenger()) and generates the scan results.
8. Generate Scan Results:
   - The results of the scan (ScanResults) are processed by the Security Screening System.
     - If the scan results indicate no issues (ScanResults == OK), the Passenger receives a notification indicating they are allowed to proceed (PassAllowedNotification).
     - If the scan results indicate a problem, the Passenger is flagged with a notification (PassNotAllowedNotification).

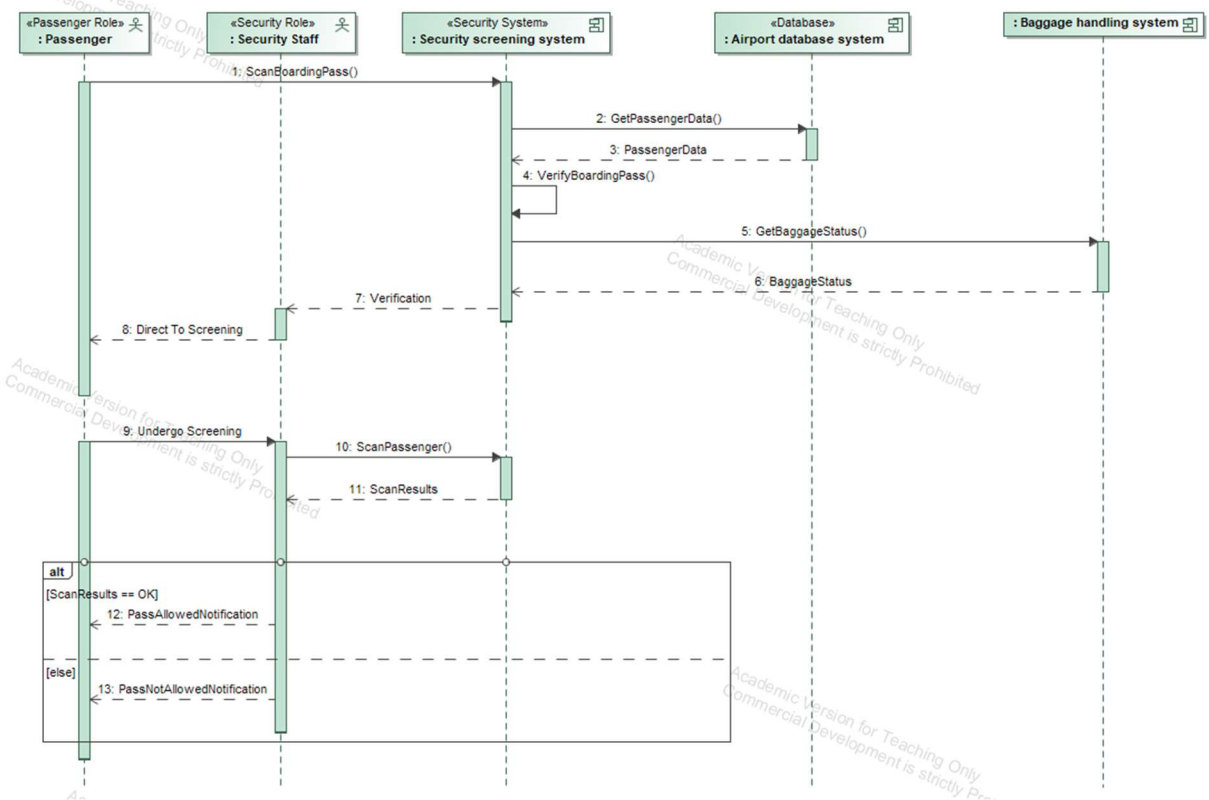
Key Operations

- ScanBoardingPass(): Validates the passenger's boarding pass.
- GetPassengerData(): Retrieves associated passenger information.
- VerifyBoardingPass(): Ensures the boarding pass is valid and matches the system's data.
- GetBaggageStatus(): Confirms that all checked baggage is appropriately screened.
- ScanPassenger(): Conducts physical scanning of the passenger for security purposes.
- PassAllowedNotification / PassNotAllowedNotification: Communicates the outcome of the security screening to the passenger.

Optional Flow

- Alternate Path (ScanResults != OK): If the screening identifies an issue, the passenger is prevented from proceeding and flagged for further action.

At the end of the process, the system ensures both the passenger and their baggage are cleared for boarding. Any security issues are flagged and addressed promptly.




## Sequence Diagram:  Customs clearance Process

This sequence diagram outlines the interaction between a passenger, customs staff, and various airport systems during the customs and security clearance process. The goal is to verify the passenger's status and ensure they are cleared or flagged for further action before proceeding.

Actors and Components

- Passenger: A person who presents their passport for customs clearance.
- Customs Staff: Personnel responsible for interacting with the customs system to process passenger information.
- Customs Processing System: The system responsible for verifying passenger information and generating their clearance status.
- Airport Database System: Stores and retrieves passenger data for verification.
- Boarding System: Updates the passenger's status for further steps in the boarding process.
- Security Screening System: Receives security alerts when passenger documents do not match.

Interaction Flow

1. Passenger Presents Passport:
    - The passenger presents their passport to customs, initiating the process (PresentPassport).
2. Passport Scanning:
    - Customs officers scan the passport into the Customs Processing System (ScanPassport()).
3. Retrieving Passenger Data:

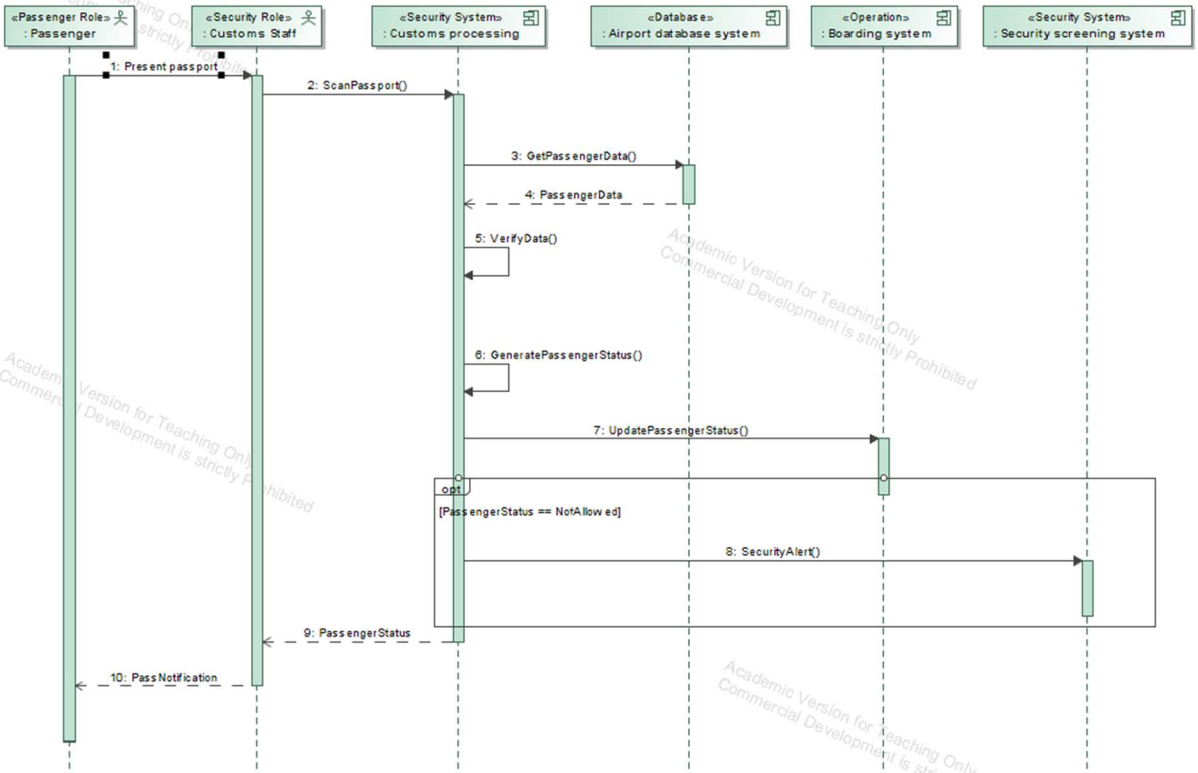
- The Customs Processing System requests passenger information from the Airport Database System (GetPassengerData()).

4. Returning Passenger Data:
   - The Airport Database System sends the retrieved passenger data back to the Customs Processing System (PassengerData).
5. Verifying Passenger Data:
   - The Customs Processing System verifies the data to ensure it is correct (VerifyData()).
6. Generating Passenger Status:
   - Based on verification, the system generates a check-in status for the passenger (GeneratePassengerStatus()).
7. Updating Passenger Status:
   - The passenger status is updated in the Boarding System (UpdatePassengerStatus()).
8. Conditional Flow: Security Alert:
   - If the passenger status is marked as “NotAllowed”, a security alert (SecurityAlert()) is triggered and sent to the security control system.
9. Passenger Status Notification:
   - The customs processing system notifies customs personnel and the passenger about the outcome (PassengerStatus).
   - If the passenger is checked in, he/she proceeds to the boarding process.
   - If he/she is not checked in, he/she is flagged for additional security checks.

Key Operations

- PresentPassport(): The passenger starts the process by presenting his/her passport.
- ScanPassport(): Customs personnel enters the passport into the system for processing.
- GetPassengerData(): Gets the recorded passenger data.
- VerifyData(): Checks the validity and consistency of the passenger information.
- GeneratePassengerStatus(): Determines whether the passenger is checked in or flagged for additional checks.
- UpdatePassengerStatus(): Communicates the passenger’s check-in status to downstream systems.
- SecurityAlert(): Sends alerts for passengers who have not been checked in.
    - Passenger Not Allowed: If the passenger status is "NotAllowed", the system triggers a security alert to the Security Control System for further action.

Once the process is complete, the system ensures that the passengers have been cleared for boarding or flagged for additional checks, maintaining the airport security protocols.


«Passenger Role»
: Passenger
«Security Role»
: Customs Staff
«Security System»
: Customs processing
«Database»
: Airport database system
«Operation»
: Boarding system
«Security System»
: Security screening system
1: Present passport
2: ScanPassport()
3: GetPassengerData()
4: PassengerData
5: VerifyData()
6: GeneratePassengerStatus()
7: UpdatePassengerStatus()
opt
[PassengerStatus == NotAllowed]
8: SecurityAlert()
9: PassengerStatus
10: PassNotification