Team 2 Submission 1

Benjamin Frost, Nasra Rage, Nimet Ozakca, Zhiheng Jiang

February 20, 2020

Contents

[1 Introduction 2](#_Toc33358326)

[2 Requirements Specification 2](#_Toc33358327)

[2.1 Description of the Existing System 2](#_Toc33358328)

[2.2 Use Case Diagram 3](#_Toc33358329)

[2.3 Use Case Specifications 4](#_Toc33358330)

[2.4 Indexed List of Use Cases 18](#_Toc33358331)

[3 System Design 19](#_Toc33358332)

[3.1 Design Class Diagrams 19](#_Toc33358333)

[3.2 ER Diagram 20](#_Toc33358334)

[3.3 Database Schema 21](#_Toc33358335)

[3.4 SQL Statements 22](#_Toc33358336)

[3.4.1 Create Database 22](#_Toc33358337)

[3.4.2 Operations 25](#_Toc33358338)

[3.4.3 Sales Reports 26](#_Toc33358339)

[3.5 GUI Designs 29](#_Toc33358340)

[3.5.1 Login Page 29](#_Toc33358341)

[3.5.2 System Administrator Home Page 30](#_Toc33358342)

[3.5.3 System Administrator Database 31](#_Toc33358343)

[3.5.4 System Administrator Stock 32](#_Toc33358344)

[3.5.5 System Administrator Add Blank 33](#_Toc33358345)

[3.5.6 System Administrator Create Staff Account 34](#_Toc33358346)

[3.5.7 Travel Advisor Home Page 35](#_Toc33358347)

[3.5.8 Access Reports 36](#_Toc33358348)

[3.5.9 Cancel Ticket 37](#_Toc33358349)

[3.5.10 Ticket Selection 38](#_Toc33358350)

[3.5.11 Card Payment 39](#_Toc33358351)

[3.5.12 Cash Payment 40](#_Toc33358352)

[3.5.13 Create Account 41](#_Toc33358353)

[3.5.14 Exchange Rate 42](#_Toc33358354)

[3.5.15 Office Manager Home Page 43](#_Toc33358355)

[3.5.16 Office Manager Tickets Menu 44](#_Toc33358356)

[3.5.17 Settings 45](#_Toc33358357)

[3.5.18 Maintain Customer Account 46](#_Toc33358358)

[3.6 Access 47](#_Toc33358359)

# Introduction

In this document we will present our design for the Air Ticket Sales system. Some diagrams are large, and therefore have provided additionally in a separate folder for optional viewing. All resources and submitted files are presented in this document.

# Requirements Specification

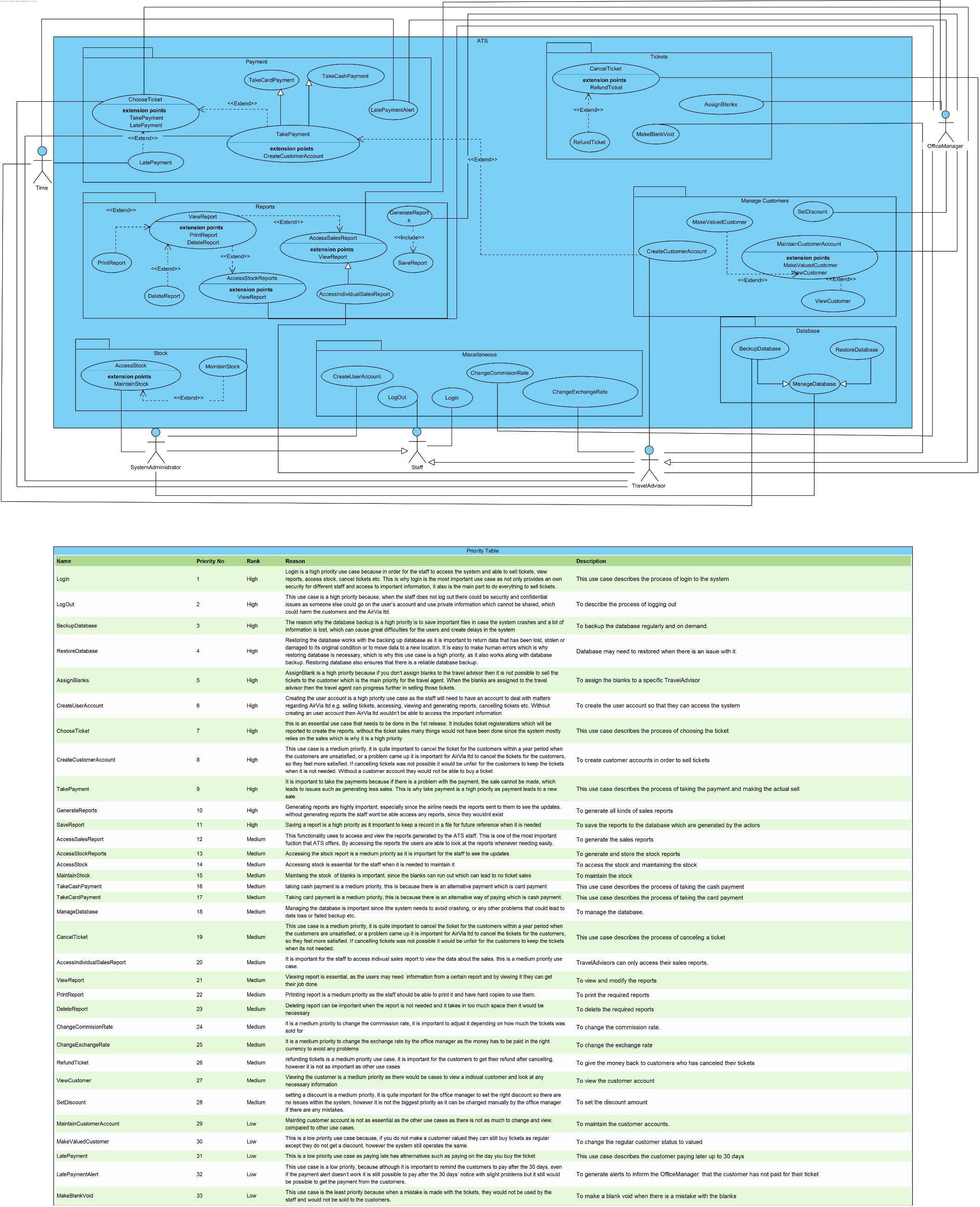
## Description of the Existing System

The software they have used for 20 years has reached the point where its further maintenance is problematic and they want a new software product to reflect the recent advances in user interfaces and more flexibility, which will eliminate the difficulties and limitations of the old software. AirVia Ltd want to evaluate the new software before they decide whether to make it mandatory for all travel agents selling their tickets.

The current system AirVia uses is an old system, it has been used for more than 20 years, the data in the system is reaching the edge of overflow and the Y2K problem may occur any time as the time system was fixed by "windowing" method rather than the rewrite. There is no graphical user interface for the current system, which makes the new employee, especially for the younger generation, feel hard to involve. For the current system, there is only one type of user that can have access of the entire system, which makes the data insecure and difficult to manage. If the customer or travel agent is paid by foreign currency the exchange rate has to be manually checked and updated by the office manager, AirVia often loses money by that. The commission rate for each travel agent might be different subject to different cases, it is very hard to change the hardcoded commission rate due to the version compatibility problem. Lastly, there are very limited ways to categories the customers which leads the difficulty in the business analysis.

## Use Case Diagram

Version 4



## Use Case Specifications

|  |  |
| --- | --- |
| **ID:** ATS1 | **Use Case:** Login |
| **Brief Description:** This use case describes the process of logging to the system to achieve the access permission to perform other functions. Every actor has different functions that they can perform. | |
| **Primary Actors:** System Administrator, Travel Advisor, Office Manager | |
| **Secondary Actors:** None. | |
| **Preconditions:** The system is operational. | |
| **Main Flow:**   1. Use case begins when the System Administrator, Travel Advisor or Office Manager activates the ATS GUI. 2. System asks for the user information: staff ID, password 3. System Administrator, Travel Advisor or Office Manager provides their unique staff ID and password 4. System encrypts the password. 5. System connects to the database 6. System checks for a matching staff ID and password.   IF there is a match, system gives permissions to the System Administrator, Travel Advisor, Office Manager  ELSE system shows the message depending on the situations. | |
| **Postconditions:** System Administrator, Travel Advisor or Office Manager gets permission if they provided a valid staff ID and password. | |
| **Alternative Flow:**  InvalidStaffID  IncorrectPassword  ForgetPassword | |

|  |
| --- |
| **Alternative Flow:** InvalidStaffID |
| **ID:** ATS1.1 |
| **Brief Description:** The information provided as a staff ID is invalid. |
| **Primary Actors:** System Administrator, Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** The system is operational. |
| **Alternative Flow:**   1. Use case starts when the system checks for the staff ID and password during the login process. 2. System searches the database and tries to match the staff ID with an existing staff ID. 3. System cannot match the staff ID with an existing staff ID. 4. System shows the message ‘The staff ID is not valid’ 5. Actor acknowledges the message. |
| **Postconditions:** The actor has informed that the staff ID invalid. |

|  |
| --- |
| **Alternative Flow:** IncorrectPassword |
| **ID:** ATS1.2 |
| **Brief Description:** The information provided as password is incorrect. |
| **Primary Actors:** System Administrator, Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** The system is operational. |
| **Alternative Flow:**   1. Use case starts when the system checks for the password during the login process. 2. System compares the encrypted passwords -the one in the database and the one that the actor provides- for a specific staff ID. 3. System could not match the passwords. 4. System shows the message ‘The password is incorrect’ 5. System Administrator, Travel Advisor or Office Manager acknowledges the message. |
| **Postconditions:** The actor has informed that the password is incorrect. |

|  |
| --- |
| **Alternative Flow:** ForgetPassword |
| **ID:** ATS1.3 |
| **Brief Description:** The system has functionality to recovering a forgotten password. |
| **Primary Actors:** System Administrator, Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** The system is operational. |
| **Alternative Flow:**   1. System Administrator, Travel Advisor or Office Manager chooses the ‘I forgot my password’ option 2. System asks for their staff ID and a security question.   IF the staff ID and the security question are matched, system sets the password to the default value.  ELSE system does not change the actors’ password.   1. System Administrator, Travel Advisor or Office Manager sets a new password for their account. |
| **Postconditions:** The account has recovered if the security question and the staff ID has matched. |

|  |  |
| --- | --- |
| **ID:** ATS2 | **Use Case:** Logout |
| **Brief Description:** This use case describes the process of logging out of a user. This use case ends the actors’ session. | |
| **Primary Actors:** System Administrator, Travel Advisor, Office Manager | |
| **Secondary Actors:** None. | |
| **Preconditions:**   1. The system is operational. 2. The actor has logged in. | |
| **Main Flow:**   1. Use case begins when the System Administrator, Travel Advisor or Office Manager selects logout option system provides. 2. The system ends the System Administrator, Travel Advisor or Office Managers’ session. 3. The system displays the login page. | |
| **Postconditions:** The system ends the System Administrator, Travel Advisor or Office Managers’ session. | |
| **Alternative Flow:** None. | |

|  |  |
| --- | --- |
| **ID:** ATS3 | **Use Case:** BackupDatabase |
| **Brief Description:** The database needs to be backed up regularly or on demand by the System Administrator. | |
| **Primary Actors:** System Administrator | |
| **Secondary Actors:** None. | |
| **Preconditions:** The actor has logged in. | |
| **Main Flow:**   1. Use case begins when the System Administrator activates the functionality to managing the database,    1. System Administrator chooses the “Backup” option system provides.   ELSE when the time comes to back up the database.   1. System connects to the database. 2. System backs up the database. 3. System disconnects to the database. 4. System shows a conformation message after a successful backup. | |
| **Postconditions:** Database is backed up. | |
| **Alternative Flow:** None. | |

|  |  |
| --- | --- |
| **ID:** ATS4 | **Use Case:** RestoreDatabase |
| **Brief Description:** The database fails to perform some actions such as storing new data, deleting data or retrieving data. It needs to be restored by the System Administrator. | |
| **Primary Actors:** System Administrator | |
| **Secondary Actors:** None. | |
| **Preconditions:** System Administrator must be logged in. | |
| **Main Flow:**   1. Users of the ATS informs the System Administrator that there is an issue with the database and it needs to be restored. 2. System Administrator backs up the database. 3. System Administrator gets exclusive access to the database. 4. System Administrator informs other ATS users how long they must avoid using the database. 5. System Administrator restores the database. 6. System Administrator saves the changes. | |
| **Postconditions:** Database is restored. | |
| **Alternative Flow:** None. | |

|  |  |
| --- | --- |
| **ID:** ATS5 | **Use Case:** AssignBlanks |
| **Brief Description:** Blanks are received by the airline and added to the stock of blanks. Office Manager allocates the blanks to each Travel Advisor before they can sell them to a customer. Unless a blank is assigned it cannot be sold. Blanks can also be reassigned from one Travel Advisor to another Travel Advisor. | |
| **Primary Actors:** Office Manager | |
| **Secondary Actors:** None. | |
| **Preconditions:** The system is operational and blanks are received from the airline and added to the stock of blanks with a unique blank ID. | |
| **Main Flow:**   1. Office Manager activated the functionality for assigning a blank. 2. IF the blank has never been assigned:   Office Manager selects a specific Travel Advisor to assigns the blanks.  ELSE  Office Manager selects a Travel Advisor to assign from, and another Travel Advisor to assign a new blank.   1. The system validates that each blank is assigned to a unique Travel Advisor. 2. The system shows the confirmation message after successful assign process. | |
| **Postconditions:** A blank is assigned to a unique Travel Advisor. | |
| **Alternative Flow:**  StockIsEmpty | |

|  |
| --- |
| **Alternative Flow:** StockIsEmpty |
| **ID:** ATS5.1 |
| **Brief Description:** The stock of blanks is empty therefore there is no blank to assign. |
| **Primary Actors:** System Administrator |
| **Secondary Actors:** None. |
| **Preconditions:** Actor must be logged in. |
| **Alternative Flow:**   1. The system prompts the System Administrator that the stock is empty. 2. System Administrator adds new blanks to the stock with a unique ID. 3. System saves the changes. |
| **Postconditions:** Blanks are added to the stock. |

|  |  |
| --- | --- |
| **ID:** ATS6 | **Use Case:** CreateUserAccount |
| **Brief Description:** ATS provides functionality to crate accounts for its users. Accounts are created by the System Administrator. | |
| **Primary Actors:** System Administrator | |
| **Secondary Actors:** None. | |
| **Preconditions:** No user exist in ATS for the new user. | |
| **Main Flow:**   1. The system administrator activates the functionality for creating an account. 2. The system places an empty user account in the session. 3. The system opens a form for entering the user details: staff ID, staff email, name and password. 4. The system administrator enters details.    1. System confirms their validity 5. System requires to choose the role of the newly created user.   5.1 The system admin specifies the role.   1. The system creates a new user record in ATS. | |
| **Postconditions:** User details are stored in the database: staff ID, staff email, name, password and staff type. | |
| **Alternative Flow:**  InvalidUserInformation  UnspecifiedRole | |

|  |
| --- |
| **Alternative Flow:** InvalidUserInformation |
| **ID:** ATS6.1 |
| **Brief Description:** The information provided for the new user is invalid or missing. |
| **Primary Actors:** System Administrator |
| **Secondary Actors:** None. |
| **Preconditions:** There must be an attempt of creating a new user account. |
| **Alternative Flow:**   1. Use case starts when the system administrator provides the information to the system. 2. System checks for the validity of the information provided.   IF required data is entered, system creates the user.  ELSE system asks for additional information or correction |
| **Postconditions:** User information is validated. |

|  |
| --- |
| **Alternative Flow:** UnspecifiedRole |
| **ID:** ATS6.2 |
| **Brief Description:** The System Administrator has not provided the information for the role of the new user. Unless the role is specified, new user account cannot be created. |
| **Primary Actors:** System Administrator |
| **Secondary Actors:** None. |
| **Preconditions:** There must be an attempt of creating a new ATS user. |
| **Alternative Flow:**   1. System asks for the role of the new user. 2. System administrator specifies the role. 3. System gives privileges to the user according to the type of the role. 4. System saves the changes. |
| **Postconditions:** Required information is provided. |

|  |  |
| --- | --- |
| **ID:** ATS7 | **Use Case:** ChooseTicket |
| **Brief Description:** Ticket sales are sold through the Travel Advisor or the Office Manager and are sold to the customers. | |
| **Primary Actors:** Travel Advisor,Office Manager | |
| **Secondary Actors:** None. | |
| **Preconditions:**   1. System is operational. 2. Check ticket price should be performed by The Travel Advisor or Office Manager. | |
| **Main Flow:**   1. The use case “ChooseTicket” starts when the Travel Advisor or Office Manager are beginning to sell the flight tickets to the customers. 2. The Travel Advisor or Office Manager select the desired performance from the system. 3. The actor looks up the information related to various airlines and checks the availability of seats on flights.    1. The system provides a list of items that matches the search keywords: departure-arrival date, time and location. 4. The Travel Advisor or Office Manager check that there are any available tickets.   4.1 The system provides more information about the selected item: blank type   1. The Travel Advisor or Office Manager enters ticket selling date and the ticket price for the desired performance. 2. If tickets are available the Travel Advisor or Office Manager choose to add item to cart    1. The system adds item to the cart. 3. The Travel Advisor or Office Manager enter the details about the customer who wants to buy the ticket. 4. The system checks for the validity of required data entry: customer email 5. Payment:   IF customer wants to pay immediately,  **Extension point (TakePayment)**  ELSE Customers that have an account can choose to pay later – up to 30 days  **Extension point (LatePayment)**   1. The system connects to the database. 2. The system writes data into the database. 3. The system shows a confirmation message after successful ticket sale. | |
| **Postconditions:**   1. Every ticket sold is registered in an Air Ticket Sales report. 2. The application verifies the authenticity of the username and password and then displays information related to various flights to the customer. 3. A new sale is recorded by the Travel Advisor or Office Manager 4. The appropriate commission amount is recorded. | |
| **Alternative Flow:**  NoSeatsLeft  BlanksFinished  InvalidCustomerData | |

|  |
| --- |
| **Alternative Flow:** NoSeatsLeft |
| **ID:** ATS7.1 |
| **Brief Description:** There is no empty seat left for the desired flight. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** There must be an attempt of selling a ticket. |
| **Alternative Flow:**   1. Use case begins when the system checks for availability of seats and informs the Travel Advisor or Office Manager that there are no seats left. 2. The Travel Advisor or Office Manager inform the customer. 3. Customer chooses alternative flight. 4. The Travel Advisor or Office Manager process to the next stage of selling the ticket. |
| **Postconditions:** Customer informed that there is no seat left for the specific flight and availability of other flights. |

|  |
| --- |
| **Alternative Flow:** BlanksFinished |
| **ID:** ATS7.2 |
| **Brief Description:** There are no blanks left for the Travel Advisor who want to sell tickets. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** The Travel Advisor or Office Manager attempted to sell a ticket, but no blanks left to sell. |
| **Alternative Flow:**   1. Use case begins when the system informs the Travel Advisor that there is no blank left to sell. 2. Travel Advisor informs the Office Manager that he/she needs to assign new blanks to the Travel Advisor. 3. Office Manager activates the functionality to assign new blanks.   **Extension Point: AssignBlanks**   1. Office Manager assigns the blanks to the Travel Advisor. 2. Travel Advisor sells to ticket for the customer who wants to buy the ticket. |
| **Postconditions:** New blanks are assigned to the customer. |

|  |
| --- |
| **Alternative Flow:** InvalidCustomerData |
| **ID:** ATS7.3 |
| **Brief Description:** Travel Advisor or the Office Manager enters the data about the customer but the system informs that some of the data provided is invalid or missing. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:**   1. There must be an attempt of selling a ticket. 2. Customer does not provide all the required data entry, or the Travel Advisor or Office Manager does not enter the information correctly. |
| **Alternative Flow:**   1. Use case begins when the customer wants to buy the ticket and provides information. 2. The Travel Advisor or Office Manager enter the details about the customer: customer email 3. System checks for the data provided. 4. System informs the Travel Advisor or Office Manager that there has a problem with the data provided. It might be missing or incorrect. 5. The Travel Advisor or Office Manager checks the data entry and reconfirms with the customer. 6. System checks the data again. 7. System confirms the validity. 8. The Travel Advisor or Office Manager process the next stage of selling the ticket. |
| **Postconditions:** The valid data about the customer is provided. |

|  |
| --- |
| **Alternative Flow:** LatePayment |
| **ID:** ATS11 |
| **Brief Description:** The customers, both valued and regular can decide to pay later, up t0 30 days if an account has been created for them. |
| **Primary Actors:** Time |
| **Secondary Actors:** None. |
| **Segment 1 Preconditions:**   1. The customer requests to opt for the late payment. |
| **Segment 1 Flow:**   1. The Travel Advisor or Office Manager choose the pay later option in the system. 2. The Travel Advisor or the Office Manager enters the details of the customers: customer email 3. The system saves the information and shows a confirmation message that the option of late payment has been accepted. |
| **Segment 1 Postconditions:**   1. The system will have an automatic alert after 30 days, to show that the customers’ payment is due. |

|  |  |
| --- | --- |
| **ID:** ATS8 | **Use Case:** CreateCustomerAccount |
| **Brief Description:** ATS provides functionality create accounts for its customer to keep information about them. | |
| **Primary Actors:** Travel Advisor, Office Manager | |
| **Secondary Actors:** None. | |
| **Preconditions:**   1. System is operational. 2. The Travel Advisor or Office Manager are logged in to the system. | |
| **Main Flow:**   1. Use case starts when the Travel Advisor or Office Manager activates the functionality to create a user account. 2. System asks for the customer information: name and email 3. The Travel Advisor or Office Manager enter the information provided by the customer. 4. System checks for the validity of the information. 5. System connects to the database. 6. The system writes data into database. 7. The system shows the confirmation message after successful account creation. | |
| **Postconditions:** A user account created with a unique email. | |
| **Alternative Flow:**  InvalidCustomerData | |

|  |
| --- |
| **Alternative Flow:** InvalidCustomerData |
| **ID:** ATS8.1 |
| **Brief Description:** Use case describes what happens when the data provided by the Travel Advisor or Office Manager is incorrect or missing. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** There must be an attempt of creating a user account. |
| **Alternative Flow:**   1. Use case begins when the Travel Advisor or Office Manager enters data to the system. 2. The Travel Advisor or Office Manager enter the details about the customer. 3. System checks for the data provided. 4. System informs the Travel Advisor or Office Manager that there has a problem with the data provided. It might be missing or incorrect. 5. The Travel Advisor or Office Manager check the data and if necessary, asks for additional information. 6. System checks the data again. 7. System confirms the validity. 8. The Travel Advisor or Office Manager process the next stage of creating a user account. |
| **Postconditions:** The new user details are provided as required by the system. |

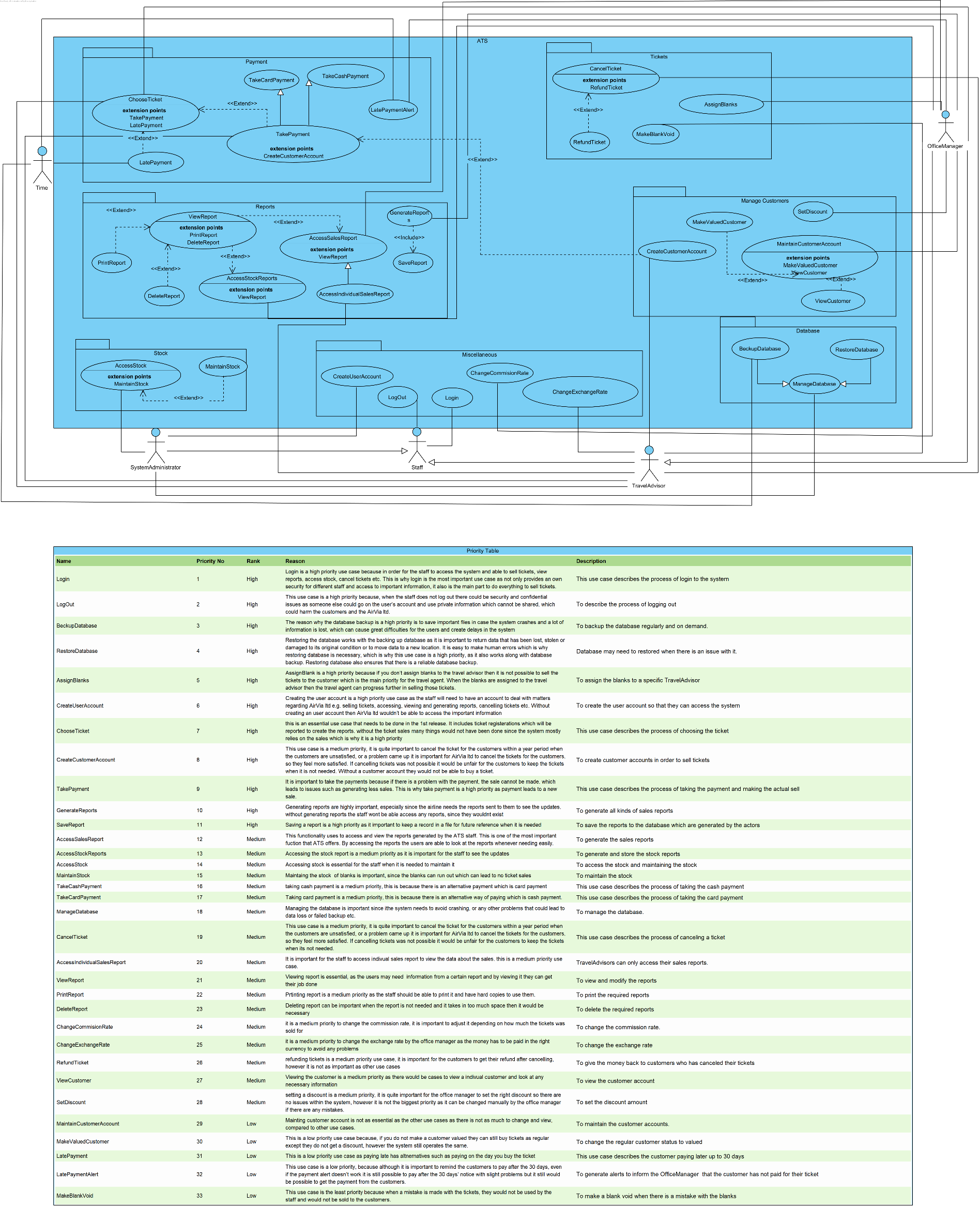
|  |  |
| --- | --- |
| **ID:** ATS9 | **Use Case:** TakePayment |
| **Brief Description:** This use case describes when the Travel Advisor or the Office Manager takes the payment for the wanted flight tickets for the customer. | |
| **Primary Actors:** Travel Advisor, Office Manager | |
| **Secondary Actors:** None. | |
| **Preconditions:**   1. System is operational. 2. Travel Advisor or Office Manager are logged in. 3. Customers have decided what ticket they want to buy. | |
| **Main Flow:**   1. The use case starts when the Travel Advisor or Office Manager process to checkout. 2. Travel Advisor or Office Manager activate the functionality to take the payment. 3. System calculates the total amount including taxes. 4. System asks for the customer account detail: customer email   **Extension Point: CreateCustomerAccount**  4.1 IF the customer is valued, the Travel Advisor or Office Manager gives discount  4.2 ELSE proceed to checkout with no discount when the customer is regular   1. System asks for the payment details.   5.1 IF the customer wants to buy by card, The Travel Advisor or Office Manager ask for the card details.  5.2 ELSE The Travel Advisor or Office Manager take cash payment.   1. The system records the payment details.   6.1 IF card payment: records account number, name on the card, card number, date, time, amount combined with the customer details.  6.2 ELSE cash payment: records amount, date and time combined with the customer details.   1. Payment must proceed and confirmation must be provided. 2. Ticket is registered to the Air Ticket Sales Report by The Travel Advisor or Office Manager. 3. System checks for all the data entry. 4. System connects to the database. 5. The system writes data into database. 6. The system shows the confirmation message after successful payment. | |
| **Postconditions:**   1. A new sale recorded in the database by the Travel Advisor or Office Manager. 2. System received whole amount at once. 3. Details about the payment is recorded. | |
| **Alternative Flow:**  WrongAmountReceived  InvalidCustomerData  IssueWithCardDetails | |

|  |
| --- |
| **Alternative Flow:** WrongAmountReceived |
| **ID:** ATS9.1 |
| **Brief Description:** The amount received from the customer at the checkout is less than expected. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** There must be an attempt of taking the payment. |
| **Alternative Flow:**   1. This use case begins when the system asks for the payment details. 2. Customer provides the amount of money which is required by the Travel Advisor or Office Manager. 3. Travel Advisor or Office Manager enter the amount to the system. 4. System checks and informs the Travel Advisor or Office Manager that the amount provided is incorrect. 5. The Travel Advisor or Office Manager inform the customer. 6. Customer provides the correct amount of money. 7. System checks and validates. |
| **Postconditions:** Correct amount of money is received. |

|  |
| --- |
| **Alternative Flow:** InvalidCustomerData |
| **ID:** ATS9.2 |
| **Brief Description:** The details about the customer is invalid. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** There has been an attempt of taking the payment but the information about the customer is invalid or missing. |
| **Alternative Flow:**   1. Use case begins when the customer provides information. 2. Travel Advisor or Office Manager enter the details about the customer: customer email 3. System checks for the data provided. 4. System informs the Travel Advisor or Office Manager that there has a problem with the data provided. It might be missing or incorrect. 5. Travel Advisor or Office Manager checks the data and if necessary, asks for additional data. 6. System checks the data again. 7. System confirms the validity. 8. The Travel Advisor or Office Manager process the next stage of payment. |
| **Postconditions:** System approved the data provided. |
| **Alternative Flow:** IssueWithCardDetails |
| **ID:** ATS9.3 |
| **Brief Description:** The details about the card which is being used for the payment cannot be accepted. The card might be expired etc. |
| **Primary Actors:** Travel Advisor, Office Manager |
| **Secondary Actors:** None. |
| **Preconditions:** There must be an attempt of taking the payment. |
| **Alternative Flow:**   1. Travel Advisor or Office Manager provides the card details to the system. 2. System checks the details and informs the user that the card cannot be accepted. 3. The Travel Advisor or Office Manager inform back to the customer. |
| **Postconditions:** No sales are made with the card which is not accepted by the system. |

|  |  |
| --- | --- |
| **ID:** ATS3 | **Use Case:** GenerateReport |
| **Brief Description:** The Office Manager creates sales report which contains all the sales transactions corresponding sub-totals and grand totals. | |
| **Primary Actors:** Office Manager, Travel Advisor | |
| **Secondary Actors:** none | |
| **Preconditions:**   1. Office Manager and Travel Advisor have logged in to the system. 2. System is operational. | |
| **Main Flow:**   1. Use case begins when the Office Manager activates the functionality to generate reports.    1. ELSE the Travel Advisor activates the functionality to generate individual reports 2. System calculates sub-totals and grand totals and displays Individual Sales Report. 3. System calculates sub-totals and grand totals and displays Global Sales Report. 4. System combines Individual-Global Reports and stock reports in a file. 5. Office Manager or Travel Advisor select save report. 6. System connects to the database. 7. The system writes data into database. 8. The system shows the confirmation message after successful generation of a report. | |
| **Postconditions:** The report is accessible, can be deleted, viewed and printed. | |
| **Alternative Flow:** None. | |

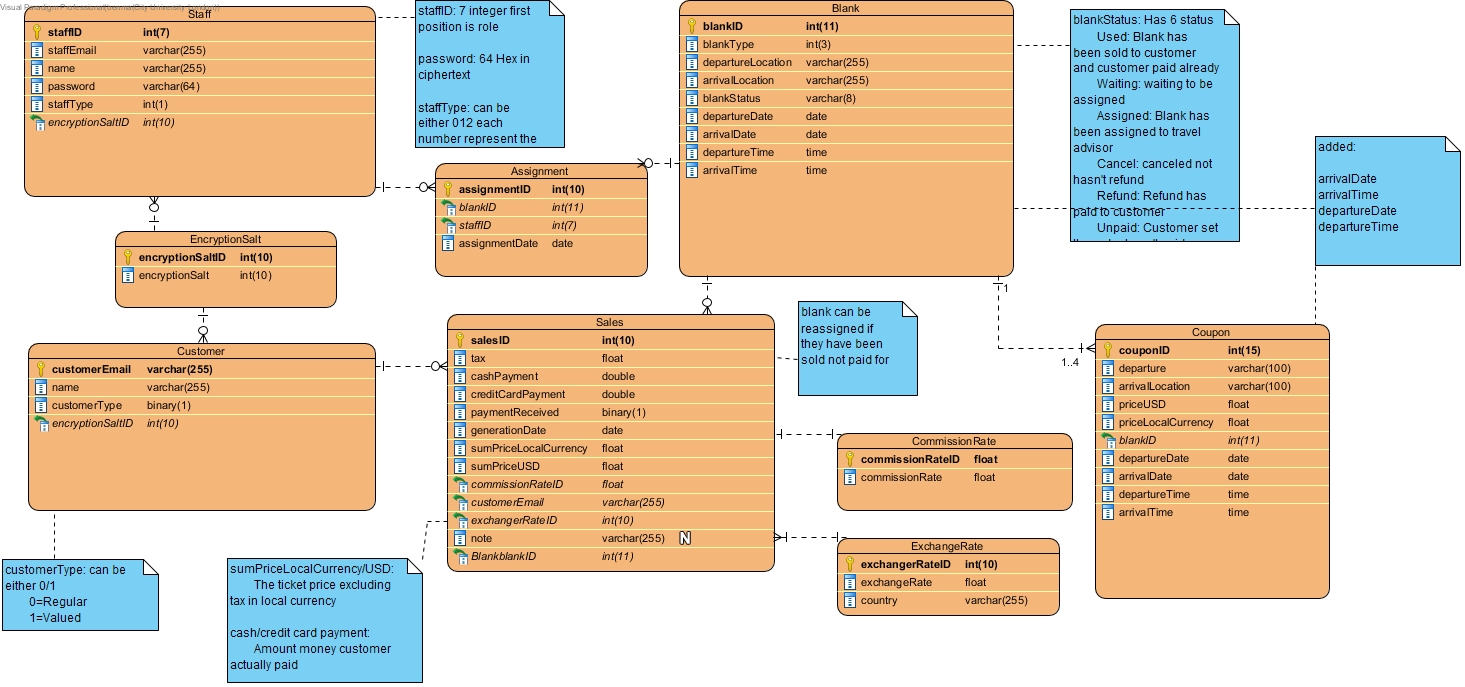
## Indexed List of Use Cases



# System Design

## A screenshot of a cell phone Description automatically generatedDesign Class Diagrams

## ER Diagram



## Database Schema

## SQL Statements

### Create Database

CREATE TABLE Staff (

staffID int(7) NOT NULL,

staffEmail varchar(255) NOT NULL,

name varchar(255) NOT NULL,

password varchar(64) NOT NULL,

staffType int(1) NOT NULL,

encryptionSaltID int(10) NOT NULL,

PRIMARY KEY (staffID),

FOREIGN KEY (encryptionSaltID) REFERENCES EncryptionSalt (encryptionSaltID));

CREATE TABLE Blank (

blankID int(11) NOT NULL,

blankType int(3) NOT NULL,

departureLocation varchar(255) NOT NULL,

arrivalLocation varchar(255) NOT NULL,

blankStatus varchar(8) NOT NULL,

departureDate date NOT NULL,

arrivalDate date NOT NULL,

departureTime time NOT NULL,

arrivalTime time NOT NULL,

PRIMARY KEY (blankID));

CREATE TABLE Customer (

customerEmail varchar(255) NOT NULL,

name varchar(255) NOT NULL,

customerType binary(1) NOT NULL,

encryptionSaltID int(10) NOT NULL,

PRIMARY KEY (customerEmail),

FOREIGN KEY (encryptionSaltID) REFERENCES EncryptionSalt (encryptionSaltID));

CREATE TABLE ExchangeRate (

exchangerRateID int(10) NOT NULL,

exchangeRate float NOT NULL,

country varchar(255) NOT NULL,

PRIMARY KEY (exchangerRateID));

CREATE TABLE CommissionRate (

commissionRate float NOT NULL,

commissionRateID float NOT NULL,

PRIMARY KEY (commissionRateID));

CREATE TABLE EncryptionSalt (

encryptionSalt int(10) NOT NULL,

encryptionSaltID int(10) NOT NULL,

PRIMARY KEY (encryptionSaltID));

CREATE TABLE Coupon (

couponID int(15) NOT NULL,

departureLocation varchar(100) NOT NULL,

arrivalLocation varchar(100) NOT NULL,

priceUSD float NOT NULL,

priceLocalCurrency float NOT NULL,

blankID int(11) NOT NULL,

departureDate date NOT NULL,

arrivalDate date NOT NULL,

departureTime time NOT NULL,

arrivalTime time NOT NULL,

PRIMARY KEY (couponID),

FOREIGN KEY (blankID) REFERENCES Blank (blankID));

CREATE TABLE Assignment (

assignmentID int(10) NOT NULL,

blankID int(11) NOT NULL,

staffID int(7) NOT NULL,

assignmentDate date NOT NULL,

PRIMARY KEY (assignmentID),

FOREIGN KEY (staffID) REFERENCES Staff (staffID),

FOREIGN KEY (blankID) REFERENCES Blank (blankID));

CREATE TABLE Sales (

salesID int(10) NOT NULL,

tax float NOT NULL,

cashPayment double NOT NULL,

creditCardPayment double NOT NULL,

paymentReceived binary(1) NOT NULL,

generationDate date NOT NULL,

sumPriceLocalCurrency float NOT NULL,

sumPriceUSD float NOT NULL,

commissionRateID float NOT NULL,

customerEmail varchar(255) NOT NULL,

blankID int(11) NOT NULL,

exchangerRateID int(10) NOT NULL,

note varchar(255),

PRIMARY KEY (salesID),

FOREIGN KEY (blankID) REFERENCES Blank (blankID),

FOREIGN KEY (exchangerRateID) REFERENCES ExchangeRate (exchangerRateID),

FOREIGN KEY (customerEmail) REFERENCES Customer (customerEmail),

FOREIGN KEY (commissionRateID) REFERENCES CommissionRate (commissionRateID));

### Operations

Staff login:

GUI typein: STAFFID(sid), PASSWORD(pw)---plaintext

SELECT EncryptionSalt.encryptionSalt FROM Staff INNER JOIN EncryptionSalt ON Staff.encryptionSaltID = EncryptionSalt.encryptionSaltID WHERE Staff.staffID = "sid";

// Identify the encryptionSalt where used to encrypt the password

SELECT password FROM Staff WHERE STAFFID="sid"

// Get the password user got

// If user enters the incorrect password system will reject the login

// If user enters the correct password system approves the login and change the encryptionSalt to improve the security.

System will generate a random number(xyz) as new encryption salt

UPDATE EncryptionSalt SET encryptionSalt = "xyz" FROM Staff INNER JOIN EncryptionSalt ON Staff.encryptionSaltID = EncryptionSalt.encryptionSaltID WHERE staffID="sid"

// update the encryption salt

system will generate new encrypted password(abc) based on the plaintext password and new encryption salt

UPDATE Staff SET password = "abc" FROM STAFF WHERE staffID="sid"

// update the new encrypted password in to the system

Add new sales

GUI typein: customerEmail(ce), blankID(bID), cashPayment(cash), creditCardPayment(card)

system calculate: tax(tax), generationDate(today), salesID(sID), sumPriceLocalCurrency(sumLocal), sumPriceUSD(sumUSD), blankID, ,commissionRateID(cid), exchangeRateID(eid), customerEmail(cEmail)

INSERT INTO Sales (salesID, tax, cashPayment, creditCardPayment, paymentReceived, generationDate, sumPriceLocalCurrency, sumPriceUSD,commissionRateID, exchangeRateID, customerEmail, blankID) VALUES ("sID", "tax", "cash", "card", "1", "today", "sumLocal", "sumUSD", "bID", "cID", "eID", "bID");

Add new blank(when Office manger first time create it)

GUI typein: blankType(bt), departureLocation(dl), arrivalLocation(al)

system generate: blankID(bID)

INSERT INTO Blank (blankID, blankType, departureLocation, arrivalLocation,blankStatus) VALUES ("bID","bt","dl","al","Waiting");

Delete overdue sales

input: todayDate(today)

system calculate the date which is 30 days before (date)

DELETE FROM Sales WHERE (generationDate = "date") AND (paymentReceived = "0");

Delete the sales that has been payed in past 30 days

Delete the blank need to return to airline

input: blankID(bID)

DELETE FROM Blank WHERE blankID = "bID";

### Sales Reports

Int sales report

data user will typein: period(start - end)

SELECT row\_number() over(order by Blank.blankID) as row\_number, blankID, sumPriceUSD, exChangeRate, sumPriceLocalCurrency, tax, （ cashPayment+creditCardPayment+ tax ）, cashPayment, creditCardPayment, (cashPayment + creditCardPayment), (cashPayment + creditCardPayment), tax FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.exChangeRateID = ExChangeRate.exChangeRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (444 OR 440 OR 420) AND "start"<Sales.generationDate<"end";

// list all the info about international blank

SELECT COUNT(blankID), SUM(sumPriceUSD), SUM(sumPriceLocalCurrency), SUM(tax), SUM(sumPriceLocalCurrency+creditCardPayment+tax),SUM(cashPayment), SUM(creditCardPayment), SUM(cashPayment+creditCardPayment+tax), SUM(cashPayment+creditCardPayment), SUM(tax) FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.exChangeRateID = ExChangeRate.exChangeRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (444 OR 440 OR 420) AND "start"<Sales.generationDate<"end";

// show the sum of data

SELECT SUM(cashPayment+creditCardPayment)\*CommissionRate FROM Sales INNER JOIN CommissionRate ON Sales.commissionRateID = CommissionRate.commissionRateID WHERE blankType = (444 OR 440 OR 420) AND "start"<Sales.generationDate<"end";

// calculate total commission rate

SELECT SUM(cashPayment+creditCardPayment)\*(1-CommissionRate), SUM(tax) FROM Sales INNER JOIN CommissionRate ON Sales.commissionRateID = CommissionRate.commissionRateID WHERE blankType = (444 OR 440 OR 420) AND "start"<Sales.generationDate<"end";

// calculate net amount

SELECT SUM(cashPayment+creditCardPayment+tax)-SUM(cashPayment+creditCardPayment)\*CommissionRate-SUM(tax) FROM Sales INNER JOIN CommissionRate ON Sales.commissionRateID = CommissionRate.commissionRateID WHERE blankType = (444 OR 440 OR 420) AND "start"<Sales.generationDate<"end";

// calculate Total net amount for bank remittance to air via

DOME

data user will typein: period(start - end)

SELECT row\_number() over(order by Blank.blankID) as row\_number, blankID, sumPriceLocalCurrency, sumPriceUSD, cashPayment, creditCardPayment, tax,（sumPriceLocalCurrency + tax）, (cashPayment + creditCardPayment), note FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.exChangeRateID = ExChangeRate.exChangeRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (201 or 101) AND "start"<Sales.generationDate<"end";

// list all the info about the domestic blank

SELECT COUNT(blankID), SUM(sumPriceLocalCurrency), SUM(sumPriceUSD),SUM(cashPayment),SUM(creditCardPayment), SUM(tax), SUM(sumPriceLocalCurrency+creditCardPayment+tax), SUM(cashPayment+creditCardPayment)FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (201 or 101) AND "start"<Sales.generationDate<"end";

// calculate total

SELECT SUM(cashPayment+creditCardPayment)\*CommissionRate FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (201 or 101) AND "start"<Sales.generationDate<"end";

// calculate total commission rate

SELECT SUM(cashPayment+creditCardPayment)\*(1-CommissionRate) FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (201 or 101) AND "start"<Sales.generationDate<"end";

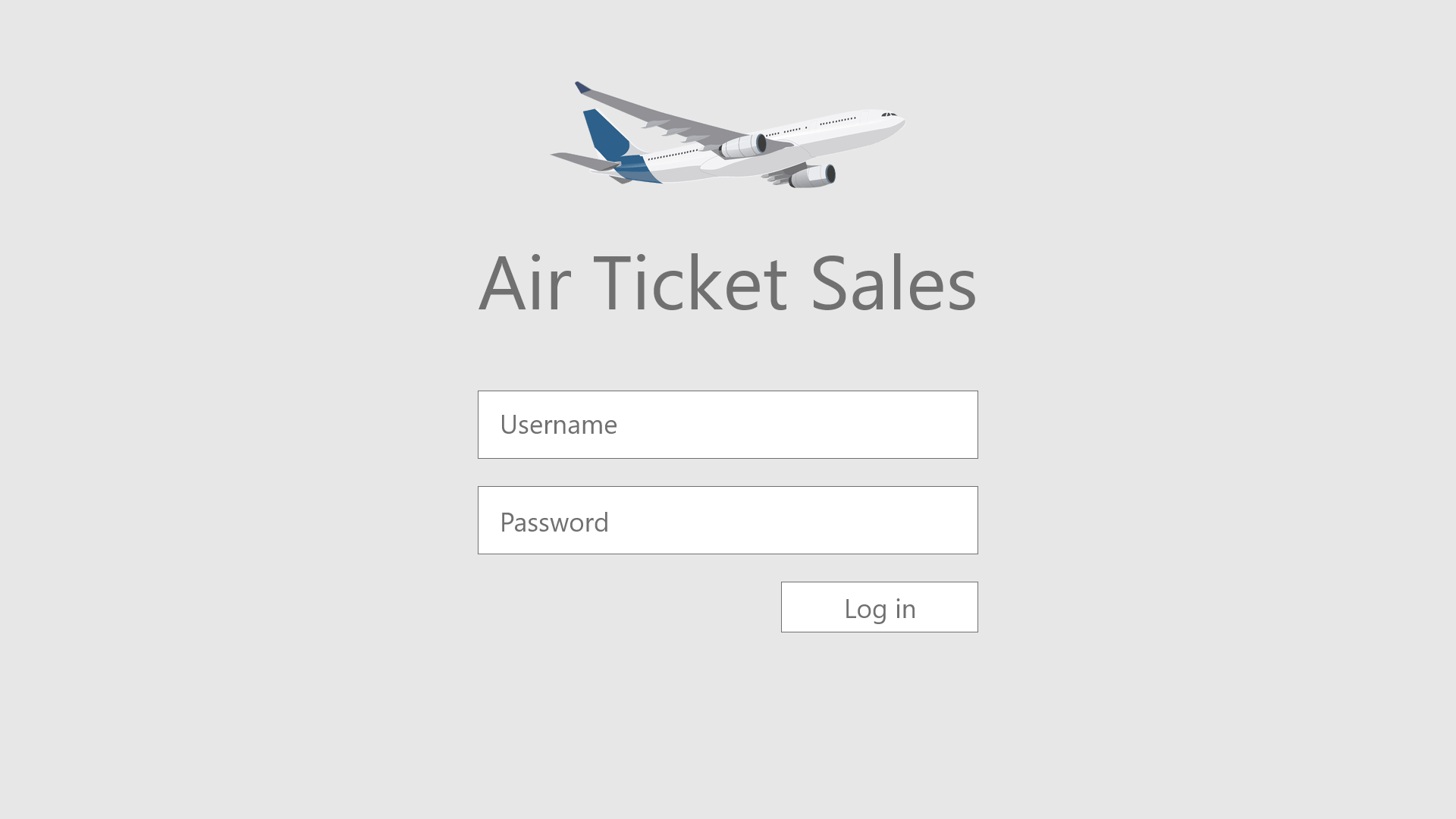
// calculate net amount for agent debit

SELECT SUM(sumPriceLocalCurrency+creditCardPayment+tax) - SUM(cashPayment+creditCardPayment)\*CommissionRate FROM Sales INNER JOIN Blank, CommissionRate, ExChangeRate ON (Sales.commissionRateID = CommissionRate.commissionRateID AND Sales.blankID = Blank.blankID) WHERE blankType = (201 or 101) AND "start"<Sales.generationDate<"end";

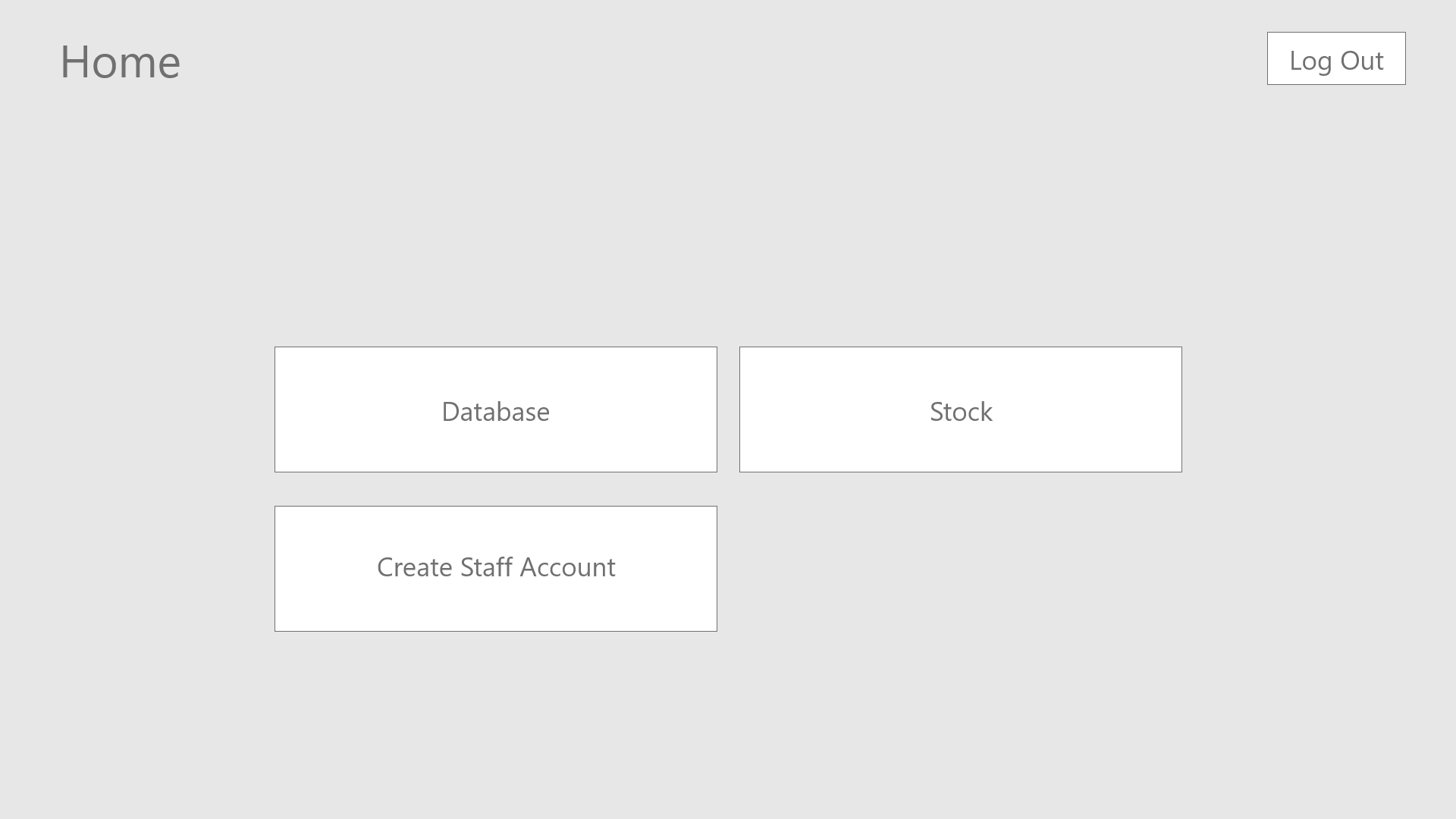
// calculate total net amount for bank remittance to air via

## GUI Designs

### Login Page



### System Administrator Home Page

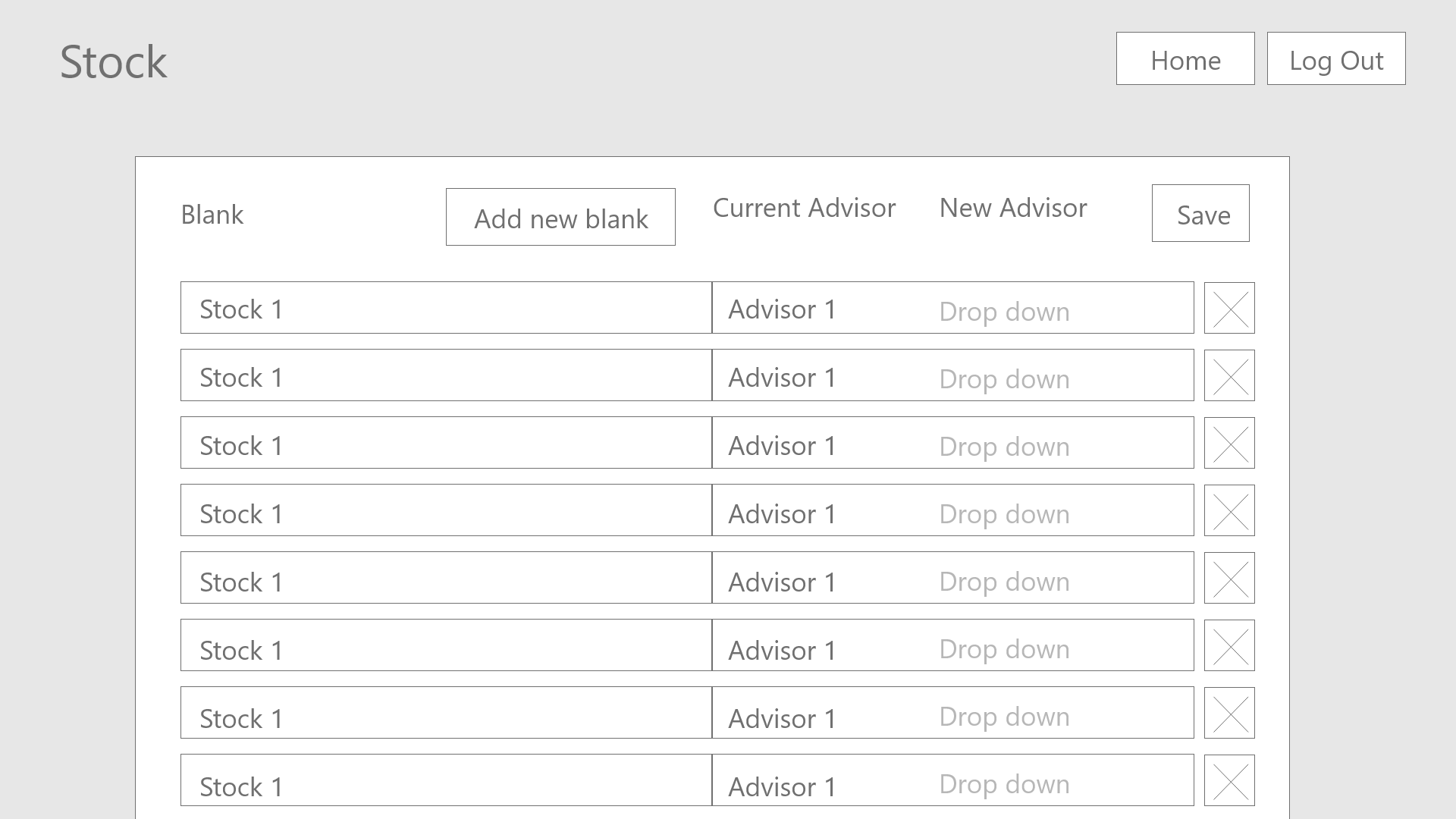


Boundary Class: SysAsminMainMenuGUI

### System Administrator Database

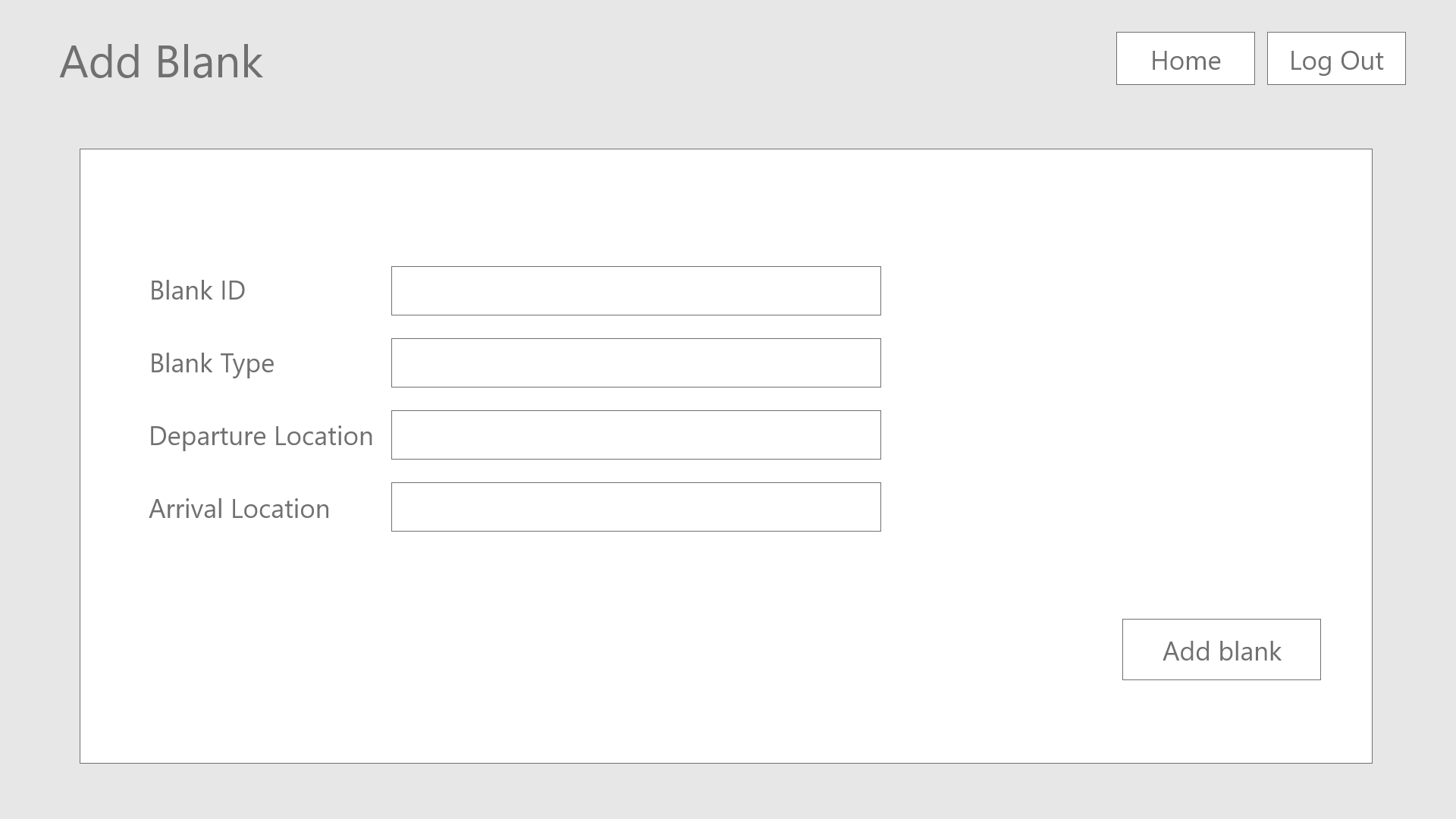
Boundary Class: DatabaseGUI

### System Administrator Stock



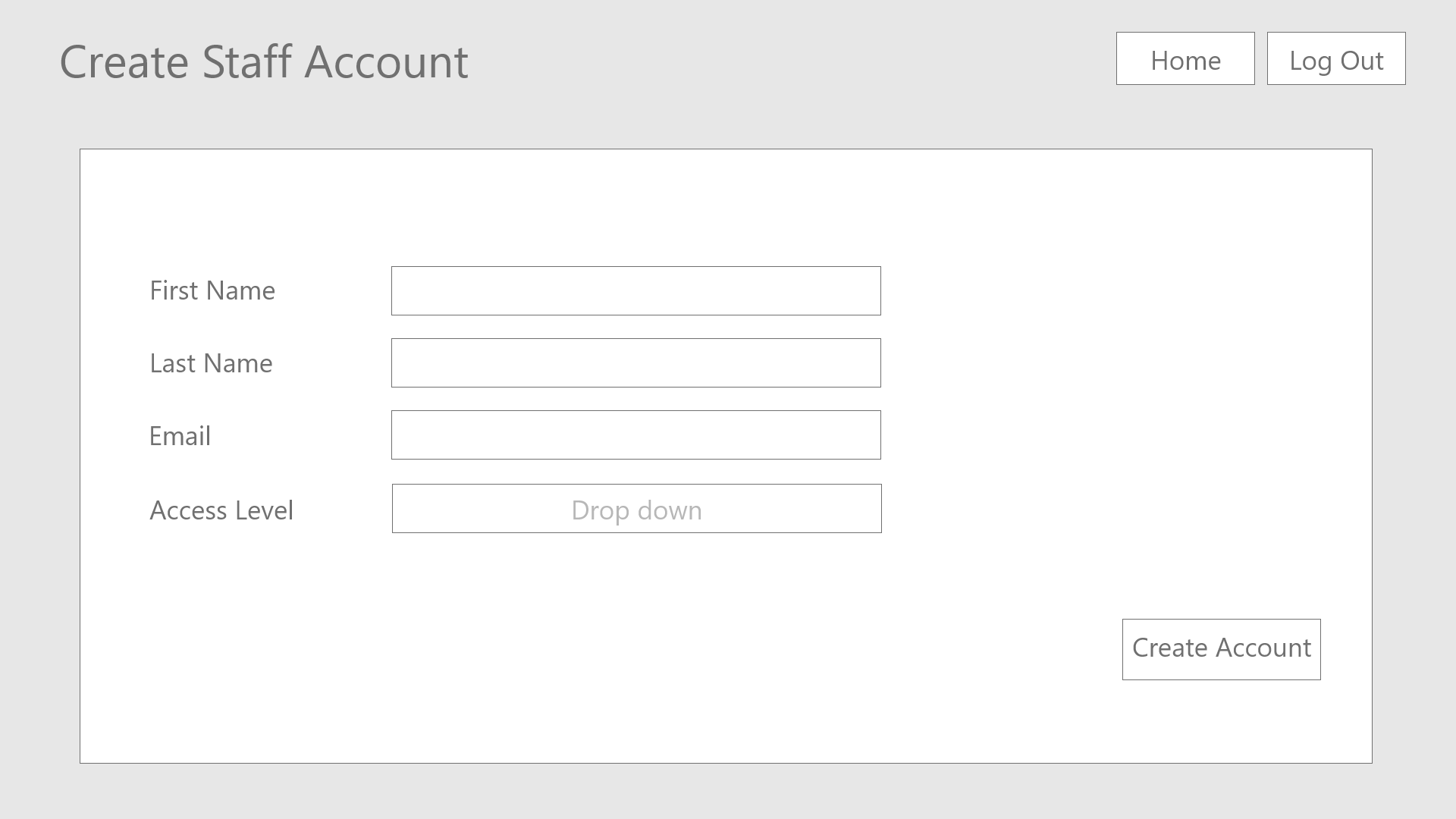
Boundary Class: StockGUI

### System Administrator Add Blank



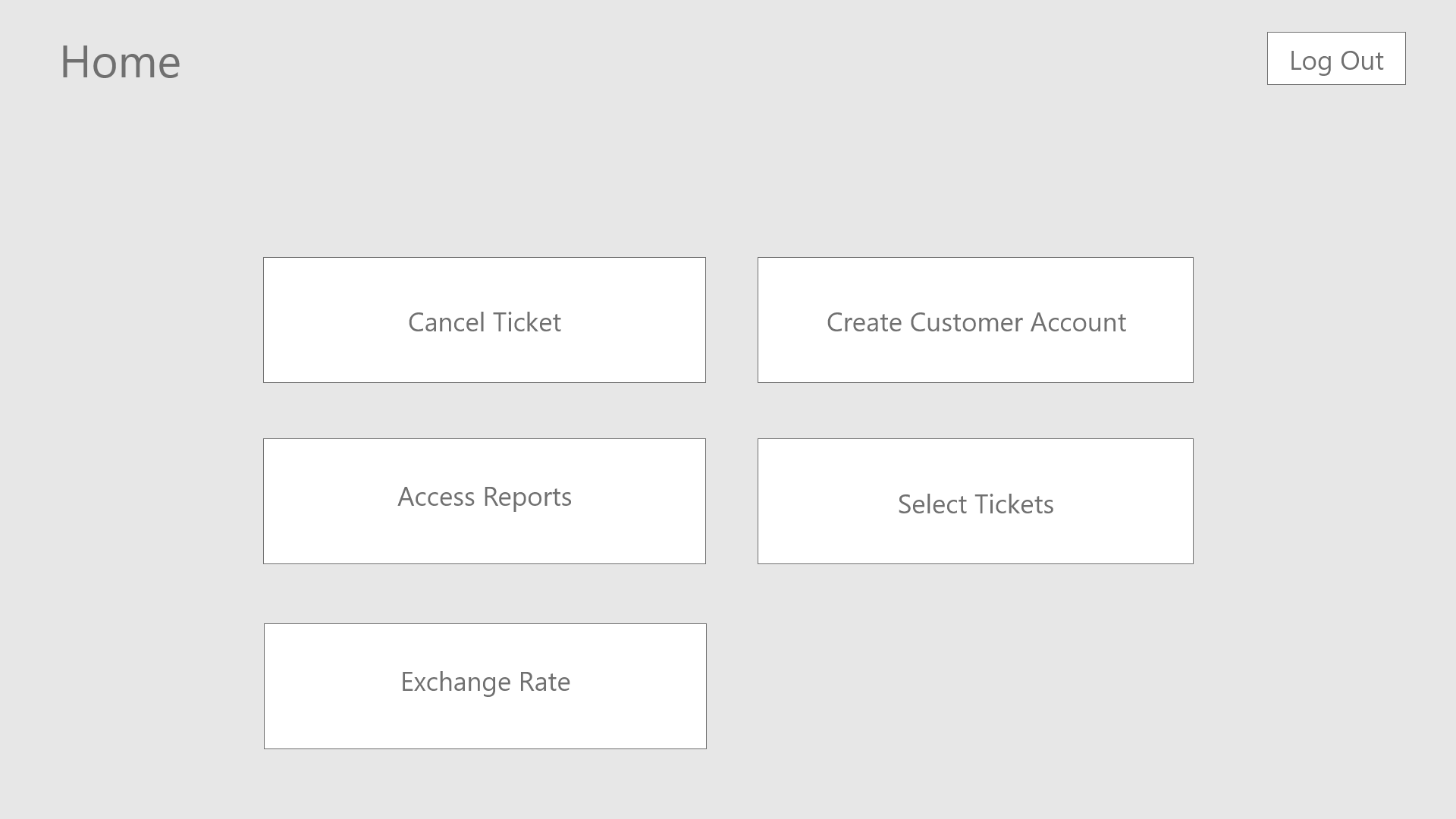
Boundary Class: AddBlankGUI

### System Administrator Create Staff Account



Boundary Class: CreateStaffAccountGUI

### Travel Advisor Home Page



Boundary Class: TravelAdvisorMainMenuGUI

### Access Reports

Boundary Class: AccessReportsGUI

### Cancel Ticket

Boundary Class: CancelTicketGUI

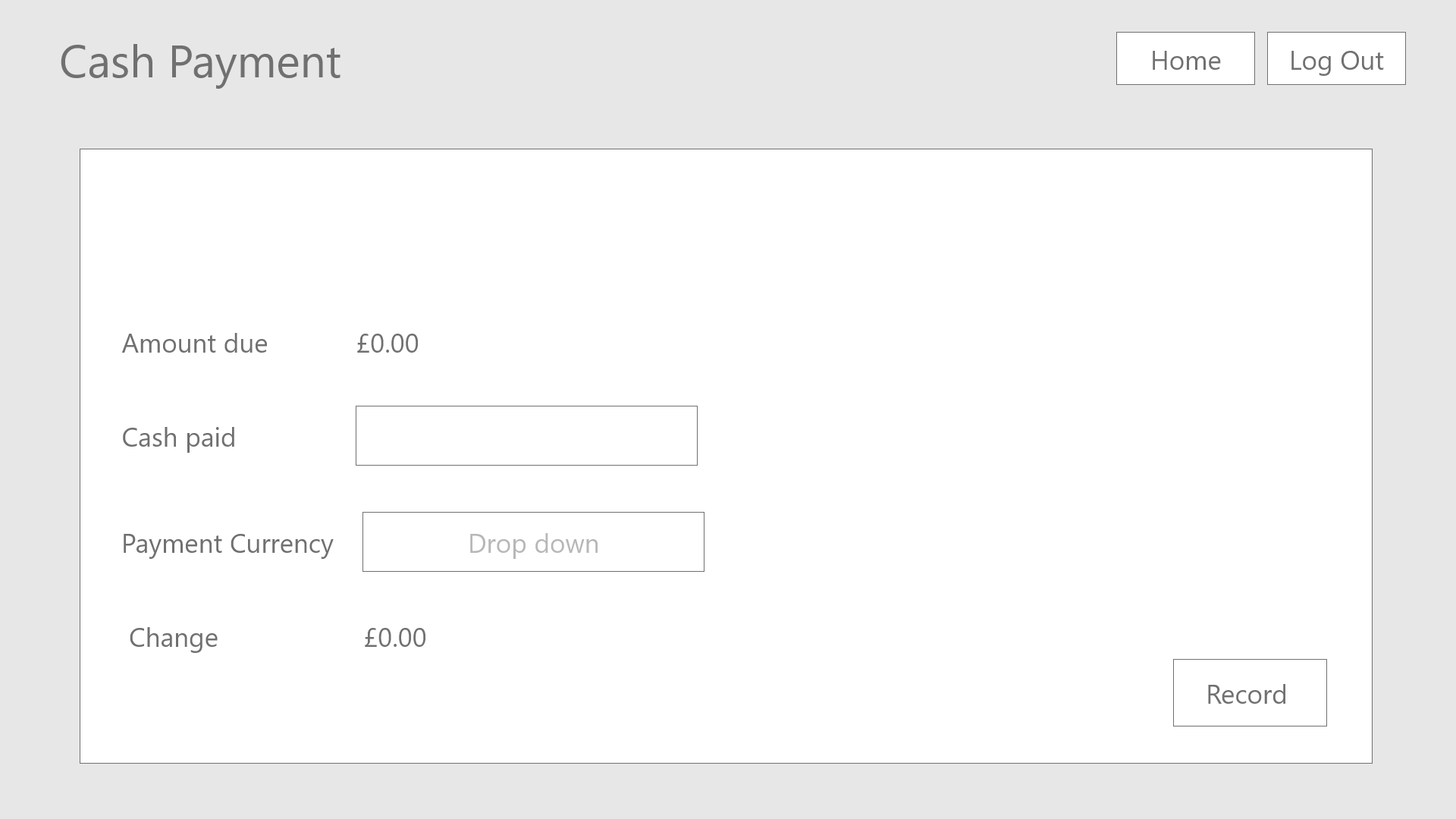
### Ticket Selection

Boundary Class: TicketSelectionGUI

### Card Payment

Boundary Class: CardPayment

### Cash Payment



Boundary Class: CashPayment

### Create Account

Boundary Class: CreateCustomerAccountGUI

### Exchange Rate

Boundary Class: ExchangeRateGUI

### Office Manager Home Page

Boundary Class: OfficeManagerMainMenuGUI

### Office Manager Tickets Menu

Boundary Class: OfficeManagerTicketsGUI

### Settings

Boundary Class: SettingsGUI

### Maintain Customer Account

Boundary Class: MaintainCustomerAccountGui

## Access

The System Administrator can access:

* Database
* Stock
* Add Blank
* Create Staff Account

Travel Advisor can access:

* Cancel Ticket
* Create Customer Account
* Select Tickets
* Card Payment
* Cash Payment
* Exchange Rate
* Access Reports (access restricted to individual reports only)

The Office Manager can access everything the Travel Advisor can access, plus:

* Void Blank
* Assign Blank
* Access Reports (all)
* Settings
* Maintain customer accounts

Users of the system will log in and be presented with their respective menu pages. From there they can navigate to whichever service they require, and then they will be shown the menu once again. Users can also log out and navigate to the home page (menu) at any point using the menu bar at the top of the application which persists independent of the screen being shown. If users log out at any time they are placed at the log in screen.

Example navigability for Travel Advisor:

1. Log in
   1. Enter staffID, password.
   2. Click Log in.
2. Home Page
   1. Click Select Tickets.
3. Select Tickets
   1. Use the drop down to select 3 tickets.
   2. Enter the customer email.
   3. Take Card Payment.
   4. Enter card details.
   5. Click Record.
4. Home Page
   1. Click Log Out.
5. end