Wolfie Airlines

Generated by Doxygen 1.9.8

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

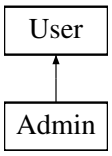
Class Documentation

4.1 Admin Class Reference

This class represents an admin user.

`#include <admin.h>`

Inheritance diagram for Admin:



Public Member Functions

- Admin (const std::string &username, const std::string &email, double discount, const std::string &discount↩ _type, const std::string &premium_card, const std::string &payment_method, mongocxx::client &client, const std::string &profession, const std::string &registration_date, double money_spent, double money_saved, int ticket_bought, const std::vector< bsoncxx::document::value > &user_flights, bool is_admin, std::string hashed_admin_password)

  Constructs a new Admin object.
- Admin (const User &user)

  Constructs a new Admin object from a User object.
- void AddFlight (User &user)

  Adds a flight.
- void AddVerificationQuestion (User &user)

  Adds a verification question.
- void ManageUsers (User &user)

  Manages users.
- void AddLuggageItem (User &user)

  Adds a luggage item.

Public Member Functions inherited from User

- User (mongocxx::client &client)

  Constructs a new User object.
- **User** (std::string username, std::string email, double discount, std::string discount_type, std::string premium_card, std::string payment_method, mongocxx::client &client, std::string profession, std::string registration_date, double money_spent, double money_saved, int ticket_bought, std::vector< bsoncxx↩ ::document::value > user_flights, bool is_admin)

  Constructs a new User object with specified parameters.
- void **Reset** ()

  Resets the user.
- mongocxx::collection & **GetCollection** ()

  Returns the collection.
- mongocxx::collection **GetSpecificCollection** (const std::string &collection_name)

  Returns a specific collection.
- std::string **GetPassword** ()

  Returns the password.
- void **SetPassword** (const std::string &password)

  Sets the password.
- void **SetPremiumCard** (User &user, const std::string &card)

  Sets the premium card.
- void **SetBlik** (const std::string &payment_method)

  Sets the Blik payment method.
- void **SetVisa** (const std::string &card_number, const std::string &card_cvv)

  Sets the Visa payment method.
- void **ChangeUsername** (const std::string &username)

  Changes the username.
- void **ChangeEmail** (const std::string &email)

  Changes the email.
- void **ChangePassword** (const std::string &password)

  Changes the password.
- void **SetDiscount** (double discount, const std::string &discount_type)

  Sets the discount.
- double **GetDiscount** () const

  Returns the discount.
- std::string **RecognizeDiscount** () const

  Recognizes the discount.
- void **AddTicketToUser** (const std::vector< int > &seats, const FlightConnection &flight_connection)

  Adds a ticket to the user.
- void **UpdateMoneySaved** (double normal_price, double discount_price)

  Updates the money saved.
- Admin * **LoginAsAdmin** ()

  Logs in as an admin.
- bool **CheckIfAdmin** () const

  Checks if the user is an admin.
- void **SetIsAdmin** (bool is_administrator)

  Sets whether the user is an admin.
- void **LuggageCheckin** (int flight_number)

  Checks in luggage.
- mongocxx::cursor **FindUserInDatabase** ()

  Finds the user in the database.
- template<typename T >
  void UpdateUserInDatabase (const std::string &value_in_database, const T &value_to_set)

  Updates the user in the database.

Public Attributes

- std::string **hashed_admin_password_**

  The hashed password of the admin.

Public Attributes inherited from User

- std::string **username_**

  The username of the user.
- std::string **profession_**

  The profession of the user.
- std::string **email_**

  The email of the user.
- std::string **discount_type_**

  The type of discount the user has.
- double **discount_**

  The discount rate of the user.
- std::string **premium_card_**

  The premium card of the user.
- std::string **payment_method_**

  The payment method of the user.
- std::string **registration_date_**

  The registration date of the user.
- double **money_spent_**

  The total money spent by the user.
- double **money_saved_**

  The total money saved by the user.
- int **ticket_bought_**

  The total tickets bought by the user.
- std::vector< bsoncxx::document::value > **user_flights_**

  The flights of the user.
- bool **is_admin_**

  Whether the user is an admin.

Additional Inherited Members

Protected Attributes inherited from User

- mongocxx::client & **_client**

  The MongoDB client.

4.1.1 Detailed Description

This class represents an admin user.

It inherits from the User class and adds additional functionality specific to admins.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Admin() [1/2]

```
Admin::Admin (
            const std::string & username,
            const std::string & email,
            double discount,
            const std::string & discount_type,
            const std::string & premium_card,
            const std::string & payment_method,
            mongocxx::client & client,
            const std::string & profession,
            const std::string & registration_date,
            double money_spent,
            double money_saved,
            int ticket_bought,
            const std::vector< bsoncxx::document::value > & user_flights,
            bool is_admin,
            std::string hashed_admin_password )
```

Constructs a new Admin object.

Parameters

username	The username of the admin.
email	The email of the admin.
discount	The discount available to the admin.
discount_type	The type of discount available to the admin.
premium_card	The premium card of the admin.
payment_method	The payment method of the admin.
client	The MongoDB client.
profession	The profession of the admin.
registration_date	The registration date of the admin.
money_spent	The amount of money spent by the admin.
money_saved	The amount of money saved by the admin.
ticket_bought	The number of tickets bought by the admin.
user_flights	The flights of the admin.
is_admin	A flag indicating whether the user is an admin.
hashed_admin_password	The hashed password of the admin.

4.1.2.2 Admin() [2/2]

```
Admin::Admin (
            const User & user ) [inline]
```

Constructs a new Admin object from a User object.

Parameters

user	The User object.

4.1.3 Member Function Documentation

4.1.3.1 AddFlight()

```
void Admin::AddFlight (
            User & user )
```

Adds a flight.

Parameters

user	The user to add the flight to.

4.1.3.2 AddLuggageItem()

```
void Admin::AddLuggageItem (
            User & user )
```

Adds a luggage item.

Parameters

user	The user to add the luggage item to.

4.1.3.3 AddVerificationQuestion()

```
void Admin::AddVerificationQuestion (
            User & user )
```

Adds a verification question.

Parameters

user	The user to add the verification question to.

4.1.3.4 ManageUsers()

```
void Admin::ManageUsers (
            User & user )
```

Manages users.

Parameters

user	The user to manage.

The documentation for this class was generated from the following files:

- admin/admin.h
- admin/admin.cpp

4.2 Authentication Class Reference

This class handles user authentication.

`#include <authentication.h>`

Public Member Functions

- Authentication (const std::string &uri_str, const std::string &db_name, const std::string &collection_name)

  Constructs a new Authentication object.
- bool RegisterUser (const std::string &username, const std::string &email, const std::string &password)

  Registers a new user.
- void AuthenticateUser (const std::string &username, const std::string &password, std::promise< bool > &&promise, User &user)

  Authenticates a user.

Static Public Member Functions

- static std::string HashPassword (const std::string &password)

  Hashes a password.

4.2.1 Detailed Description

This class handles user authentication.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Authentication()

```
Authentication::Authentication (
            const std::string & uri_str,
            const std::string & db_name,
            const std::string & collection_name )
```

Constructs a new Authentication object.

Parameters

uri_str	The URI string.
db_name	The database name.
collection_name	The collection name.

4.2.3 Member Function Documentation

4.2.3.1 AuthenticateUser()

```
void Authentication::AuthenticateUser (
            const std::string & username,
            const std::string & password,
            std::promise< bool > && promise,
            User & user )
```

Authenticates a user.

Parameters

username	The username of the user.
password	The password of the user.
promise	A promise for the result of the authentication.
user	The User object.

4.2.3.2 HashPassword()

```
std::string Authentication::HashPassword (
            const std::string & password )  [static]
```

Hashes a password.

Parameters

password	The password to hash.

Returns

The hashed password.

4.2.3.3 RegisterUser()

```
bool Authentication::RegisterUser (
            const std::string & username,
            const std::string & email,
            const std::string & password )
```

Registers a new user.

Parameters

username	The username of the user.
email	The email of the user.
password	The password of the user.

Returns

A boolean indicating if the registration was successful.

The documentation for this class was generated from the following files:

- authentication/authentication.h
- authentication/authentication.cpp

4.3 EnvParser Class Reference

This class is used for parsing environment variables.

```
#include <env.h>
```

Public Member Functions

- **EnvParser** ()
  Constructs a new EnvParser object.
- void **ParseEnvFile** ()
  Parses the environment file.
- std::string GetValue (const std::string &key) const
  Gets the value of a specific environment variable.

4.3.1 Detailed Description

This class is used for parsing environment variables.

4.3.2 Member Function Documentation

4.3.2.1 GetValue()

```
std::string EnvParser::GetValue (
            const std::string & key ) const
```

Gets the value of a specific environment variable.

Parameters

key	The key of the environment variable.

Returns

The value of the environment variable.

The documentation for this class was generated from the following files:

- env/env.h
- env/env.cpp

4.4 FlightConnection Class Reference

This class handles flight connections.

`#include <flight_connection.h>`

Public Member Functions

- FlightConnection (const std::string &uri_str, const std::string &db_name, const std::string &collection_name)

  Constructs a new FlightConnection object.
- FlightConnection (std::string flight_id, std::string departure_city, std::string destination_city, std::string departure_time, std::string arrival_time, int available_seats, double price)

  Constructs a new FlightConnection object.
- std::string GetDepartureCity () const

  Gets the departure city.
- std::string GetDestinationCity () const

  Gets the destination city.
- std::string GetDepartureTime () const

  Gets the departure time.
- std::string GetArrivalTime () const

  Gets the arrival time.
- std::string GetIdentifier () const

  Gets the flight identifier.
- int GetAvailableSeats () const

  Gets the number of available seats.
- double GetPrice () const

  Gets the price of the flight.
- std::vector< FlightConnection > FindAllConnections ()

  Finds all flight connections.
- FlightConnection FindConnection (const std::string &departure_city, const std::string &destination_city)

  Finds a flight connection by departure and destination cities.
- std::vector< FlightConnection > FindConnectionByPrice (double &min_price, double &max_price)

  Finds flight connections by price range.
- FlightConnection FindConnectionById (const std::string &id)

  Finds a flight connection by ID.
- std::vector< FlightConnection > FindConnectionsByDeparture (const std::string &departure_city)

  Finds flight connections by departure city.
- std::vector< FlightConnection > FindConnectionsByDestination (const std::string &destination_city)

  Finds flight connections by destination city.
- std::vector< int > GetSeatsTaken (const std::string &flight_identifier)

  Gets the seats taken for a specific flight.
- void UpdateSeatsTaken (const std::string &flight_identifier, const std::vector< int > &seats_taken)

  Updates the seats taken for a specific flight.

4.4.1 Detailed Description

This class handles flight connections.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FlightConnection() [1/2]

```
FlightConnection::FlightConnection (
            const std::string & uri_str,
            const std::string & db_name,
            const std::string & collection_name )
```

Constructs a new FlightConnection object.

Parameters

uri_str	The URI string.
db_name	The database name.
collection_name	The collection name.

4.4.2.2 FlightConnection() [2/2]

```
FlightConnection::FlightConnection (
            std::string flight_id,
            std::string departure_city,
            std::string destination_city,
            std::string departure_time,
            std::string arrival_time,
            int available_seats,
            double price )
```

Constructs a new FlightConnection object.

Parameters

flight_id	The flight ID.
departure_city	The departure city.
destination_city	The destination city.
departure_time	The departure time.
arrival_time	The arrival time.
available_seats	The number of available seats.
price	The price of the flight.

4.4.3 Member Function Documentation

4.4.3.1 FindAllConnections()

```
std::vector< FlightConnection > FlightConnection::FindAllConnections ( )
```

Finds all flight connections.

Returns

A vector of all flight connections.

4.4.3.2 FindConnection()

```
FlightConnection FlightConnection::FindConnection (
            const std::string & departure_city,
            const std::string & destination_city )
```

Finds a flight connection by departure and destination cities.

Parameters

departure_city	The departure city.
destination_city	The destination city.

Returns

The found flight connection.

4.4.3.3 FindConnectionById()

```
FlightConnection FlightConnection::FindConnectionById (
            const std::string & id )
```

Finds a flight connection by ID.

Parameters

id	The flight ID.

Returns

The found flight connection.

4.4.3.4 FindConnectionByPrice()

```
std::vector< FlightConnection > FlightConnection::FindConnectionByPrice (
```

```
            double & min_price,
            double & max_price )
```

Finds flight connections by price range.

Parameters

min_price	The minimum price.
max_price	The maximum price.

Returns

A vector of found flight connections.

4.4.3.5 FindConnectionsByDeparture()

```
std::vector< FlightConnection > FlightConnection::FindConnectionsByDeparture (
            const std::string & departure_city )
```

Finds flight connections by departure city.

Parameters

departure_city	The departure city.

Returns

A vector of found flight connections.

4.4.3.6 FindConnectionsByDestination()

```
std::vector< FlightConnection > FlightConnection::FindConnectionsByDestination (
            const std::string & destination_city )
```

Finds flight connections by destination city.

Parameters

destination_city	The destination city.

Returns

A vector of found flight connections.

4.4.3.7 GetArrivalTime()

```
std::string FlightConnection::GetArrivalTime ( ) const
```

Gets the arrival time.

Returns

The arrival time.

4.4.3.8 GetAvailableSeats()

```
int FlightConnection::GetAvailableSeats ( ) const
```

Gets the number of available seats.

Returns

The number of available seats.

4.4.3.9 GetDepartureCity()

```
std::string FlightConnection::GetDepartureCity ( ) const
```

Gets the departure city.

Returns

The departure city.

4.4.3.10 GetDepartureTime()

```
std::string FlightConnection::GetDepartureTime ( ) const
```

Gets the departure time.

Returns

The departure time.

4.4.3.11 GetDestinationCity()

```
std::string FlightConnection::GetDestinationCity ( ) const
```

Gets the destination city.

Returns

The destination city.

4.4.3.12 GetIdentifier()

```
std::string FlightConnection::GetIdentifier ( ) const
```

Gets the flight identifier.

Returns

The flight identifier.

4.4.3.13 GetPrice()

```
double FlightConnection::GetPrice ( ) const
```

Gets the price of the flight.

Returns

The price of the flight.

4.4.3.14 GetSeatsTaken()

```
std::vector< int > FlightConnection::GetSeatsTaken (
            const std::string & flight_identifier )
```

Gets the seats taken for a specific flight.

Parameters

flight_identifier	The flight identifier.

Returns

A vector of seats taken.

4.4.3.15 UpdateSeatsTaken()

```
void FlightConnection::UpdateSeatsTaken (
            const std::string & flight_identifier,
            const std::vector< int > & seats_taken )
```

Updates the seats taken for a specific flight.

Parameters

flight_identifier	The flight identifier.
seats_taken	A vector of seats taken.

The documentation for this class was generated from the following files:

- flights/flight_connection.h
- flights/flight_connection.cpp

4.5 FlightInfo Struct Reference

Contains information about a flight.

`#include <user_tickets_prints.h>`

Public Attributes

- int **flight_number**
  The flight number.
- std::string **flight_id**
  The flight ID.
- std::string **departure**
  The departure location.
- std::string **destination**
  The destination location.
- std::string **departure_time**
  The departure time.
- double **price**
  The price of the flight.
- std::vector< int > **seats**
  The seats in the flight.
- bool **checkin**
  Whether the user has checked in.
- bool **luggage_checkin**
  Whether the user has checked in luggage.

4.5.1 Detailed Description

Contains information about a flight.

The documentation for this struct was generated from the following file:

- user/user_functions/user_tickets/user_tickets_prints.h

4.6 Item Class Reference

This class represents an item in the airport system.

`#include <item.h>`

Public Member Functions

- Item (const std::string &item_name, const std::string &description, const std::vector< std::string > &hints, bool forbidden, bool registered_luggage, bool hand_luggage, bool pilot_allowance, double max_count, double weight, std::string &profession, std::string &category)

  Constructs a new Item object.
- Item (const std::string &item_name, const std::string &description, const std::vector< std::string > &hints, bool forbidden, bool registered_luggage, bool hand_luggage, bool pilot_allowance, double max_count, double weight, std::string &category)

  Constructs a new Item object.
- const std::string & GetItemName () const

  Gets the name of the item.
- const std::string & GetDescription () const

  Gets the description of the item.
- const std::string & GetProfession () const

  Gets the profession related to the item.
- const std::vector< std::string > & GetHints () const

  Gets the hints related to the item.
- bool IsForbidden () const

  Checks if the item is forbidden.
- bool IsRegisteredLuggage () const

  Checks if the item can be transported as registered luggage.
- bool IsHandLuggage () const

  Checks if the item can be transported as hand luggage.
- bool IsPilotAllowance () const

  Checks if the item needs pilot allowance to be transported.
- double GetMaxCount () const

  Gets the maximum count of the item.
- double GetWeight () const

  Gets the weight of the item.
- std::string GetCategory () const

  Gets the category of the item.

4.6.1 Detailed Description

This class represents an item in the airport system.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Item() [1/2]

```
Item::Item (
            const std::string & item_name,
            const std::string & description,
            const std::vector< std::string > & hints,
            bool forbidden,
            bool registered_luggage,
            bool hand_luggage,
            bool pilot_allowance,
            double max_count,
            double weight,
            std::string & profession,
            std::string & category )
```

Constructs a new Item object.

Parameters

item_name	The name of the item.
description	The description of the item.
hints	The hints related to the item.
forbidden	Indicates whether the item is forbidden.
registered_luggage	Indicates whether the item can be transported as registered luggage.
hand_luggage	Indicates whether the item can be transported as hand luggage.
pilot_allowance	Indicates whether the item needs pilot allowance to be transported.
max_count	The maximum count of the item.
weight	The weight of the item.
profession	The profession related to the item.
category	The category of the item.

4.6.2.2 Item() [2/2]

```
Item::Item (
            const std::string & item_name,
            const std::string & description,
            const std::vector< std::string > & hints,
            bool forbidden,
            bool registered_luggage,
            bool hand_luggage,
            bool pilot_allowance,
            double max_count,
            double weight,
            std::string & category )
```

Constructs a new Item object.

Parameters

item_name	The name of the item.
description	The description of the item.
hints	The hints related to the item.
forbidden	Indicates whether the item is forbidden.
registered_luggage	Indicates whether the item can be transported as registered luggage.
hand_luggage	Indicates whether the item can be transported as hand luggage.
pilot_allowance	Indicates whether the item needs pilot allowance to be transported.
max_count	The maximum count of the item.
weight	The weight of the item.
category	The category of the item.

4.6.3 Member Function Documentation

4.6.3.1 GetCategory()

```
std::string Item::GetCategory ( ) const
```

Gets the category of the item.

Returns

The category of the item.

4.6.3.2 GetDescription()

```
const std::string & Item::GetDescription ( ) const
```

Gets the description of the item.

Returns

The description of the item.

4.6.3.3 GetHints()

```
const std::vector< std::string > & Item::GetHints ( ) const
```

Gets the hints related to the item.

Returns

The hints related to the item.

4.6.3.4 GetItemName()

```
const std::string & Item::GetItemName ( ) const
```

Gets the name of the item.

Returns

The name of the item.

4.6.3.5 GetMaxCount()

```
double Item::GetMaxCount ( ) const
```

Gets the maximum count of the item.

Returns

The maximum count of the item.

4.6.3.6 GetProfession()

```
const std::string & Item::GetProfession ( ) const
```

Gets the profession related to the item.

Returns

The profession related to the item.

4.6.3.7 GetWeight()

```
double Item::GetWeight ( ) const
```

Gets the weight of the item.

Returns

The weight of the item.

4.6.3.8 IsForbidden()

```
bool Item::IsForbidden ( ) const
```

Checks if the item is forbidden.

Returns

True if the item is forbidden, false otherwise.

4.6.3.9 IsHandLuggage()

```
bool Item::IsHandLuggage ( ) const
```

Checks if the item can be transported as hand luggage.

Returns

True if the item can be transported as hand luggage, false otherwise.

4.6.3.10 IsPilotAllowance()

```
bool Item::IsPilotAllowance ( ) const
```

Checks if the item needs pilot allowance to be transported.

Returns

True if the item needs pilot allowance to be transported, false otherwise.

4.6.3.11 IsRegisteredLuggage()

```
bool Item::IsRegisteredLuggage ( ) const
```

Checks if the item can be transported as registered luggage.

Returns

True if the item can be transported as registered luggage, false otherwise.

The documentation for this class was generated from the following files:

- luggage/item/item.h
- luggage/item/item.cpp

4.7 Luggage Class Reference

This class represents a luggage in the airport system.

```
#include <luggage.h>
```

Public Member Functions

- Luggage (const std::vector< Item > &items, double total_weight)
  Constructs a new Luggage object.
- double ProcessItemsAndGetWeight ()
  Processes the items and gets the weight.
- std::tuple< bool, std::string > ConfirmItems (User &user)
  Confirms the items in the luggage.
- double CalculateOverweightFee (double weight) const
  Calculates the overweight fee.

Public Attributes

- const double **overweight_fee_per_kg_** = 2.0
  The overweight fee per kg.
- const double **euro_to_pln_** = 4.32
  The conversion rate from euro to pln.
- const double **max_allowed_weight_** = 32.0
  The maximum allowed weight.
- double **max_weight_** = 20.0
  The maximum weight.

4.7.1 Detailed Description

This class represents a luggage in the airport system.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Luggage()

```
Luggage::Luggage (
            const std::vector< Item > & items,
            double total_weight )  [inline]
```

Constructs a new Luggage object.

Parameters

items	The items in the luggage.
total_weight	The total weight of the luggage.

4.7.3 Member Function Documentation

4.7.3.1 CalculateOverweightFee()

```
double Luggage::CalculateOverweightFee (
            double weight ) const
```

Calculates the overweight fee.

Parameters

weight	The weight of the luggage.

Returns

The overweight fee.

4.7.3.2 ConfirmItems()

```
std::tuple< bool, std::string > Luggage::ConfirmItems (
            User & user )
```

Confirms the items in the luggage.

Parameters

user	The user confirming the items.

Returns

A tuple containing a boolean indicating if the confirmation was successful and a string message.

4.7.3.3 ProcessItemsAndGetWeight()

```
double Luggage::ProcessItemsAndGetWeight ( )
```

Processes the items and gets the weight.

Returns

The weight of the items.

The documentation for this class was generated from the following files:

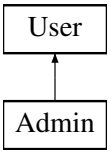
- luggage/luggage.h
- luggage/luggage.cpp

4.8 User Class Reference

Represents a user in the system.

`#include <user.h>`

Inheritance diagram for User:



Public Member Functions

- User (mongocxx::client &client)

  Constructs a new User object.
- **User** (std::string username, std::string email, double discount, std::string discount_type, std::string premium_card, std::string payment_method, mongocxx::client &client, std::string profession, std::string registration_date, double money_spent, double money_saved, int ticket_bought, std::vector< bsoncxx↩ ::document::value > user_flights, bool is_admin)

  Constructs a new User object with specified parameters.
- void **Reset** ()

  Resets the user.
- mongocxx::collection & **GetCollection** ()

  Returns the collection.
- mongocxx::collection **GetSpecificCollection** (const std::string &collection_name)

  Returns a specific collection.
- std::string **GetPassword** ()

  Returns the password.
- void **SetPassword** (const std::string &password)

  Sets the password.
- void **SetPremiumCard** (User &user, const std::string &card)

  Sets the premium card.
- void **SetBlik** (const std::string &payment_method)

  Sets the Blik payment method.
- void **SetVisa** (const std::string &card_number, const std::string &card_cvv)

  Sets the Visa payment method.
- void **ChangeUsername** (const std::string &username)

  Changes the username.
- void **ChangeEmail** (const std::string &email)

  Changes the email.
- void **ChangePassword** (const std::string &password)

  Changes the password.
- void **SetDiscount** (double discount, const std::string &discount_type)

  Sets the discount.
- double **GetDiscount** () const

  Returns the discount.
- std::string **RecognizeDiscount** () const

  Recognizes the discount.

- void **AddTicketToUser** (const std::vector< int > &seats, const FlightConnection &flight_connection)

  Adds a ticket to the user.
- void **UpdateMoneySaved** (double normal_price, double discount_price)

  Updates the money saved.
- Admin * **LoginAsAdmin** ()

  Logs in as an admin.
- bool **CheckIfAdmin** () const

  Checks if the user is an admin.
- void **SetIsAdmin** (bool is_administrator)

  Sets whether the user is an admin.
- void **LuggageCheckin** (int flight_number)

  Checks in luggage.
- mongocxx::cursor **FindUserInDatabase** ()

  Finds the user in the database.
- template<typename T >
  void UpdateUserInDatabase (const std::string &value_in_database, const T &value_to_set)

  Updates the user in the database.

Public Attributes

- std::string **username_**

  The username of the user.
- std::string **profession_**

  The profession of the user.
- std::string **email_**

  The email of the user.
- std::string **discount_type_**

  The type of discount the user has.
- double **discount_**

  The discount rate of the user.
- std::string **premium_card_**

  The premium card of the user.
- std::string **payment_method_**

  The payment method of the user.
- std::string **registration_date_**

  The registration date of the user.
- double **money_spent_**

  The total money spent by the user.
- double **money_saved_**

  The total money saved by the user.
- int **ticket_bought_**

  The total tickets bought by the user.
- std::vector< bsoncxx::document::value > **user_flights_**

  The flights of the user.
- bool **is_admin_**

  Whether the user is an admin.

Protected Attributes

- mongocxx::client & **_client**

  The MongoDB client.

4.8.1 Detailed Description

Represents a user in the system.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 User()

```
User::User (
            mongocxx::client & client )  [explicit]
```

Constructs a new User object.

Parameters

client	The MongoDB client.

4.8.3 Member Function Documentation

4.8.3.1 UpdateUserInDatabase()

```
template<typename T >
void User::UpdateUserInDatabase (
            const std::string & value_in_database,
            const T & value_to_set )  [inline]
```

Updates the user in the database.

Template Parameters

T	The type of the value to set.

Parameters

value_in_database	The value in the database.
value_to_set	The value to set.

The documentation for this class was generated from the following files:

- user/user.h
- user/user.cpp

Chapter 5

File Documentation

5.1 admin/admin.h File Reference

This file contains the declaration of the Admin class.

```
#include "../user/user.h"
```

Classes

- class Admin

  This class represents an admin user.

5.1.1 Detailed Description

This file contains the declaration of the Admin class.

5.2 admin.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_ADMIN_H
00007 #define AIRPORT_ADMIN_H
00008
00009 #include "../user/user.h"
00010
00016 class Admin : public User {
00017  public:
00036   Admin(const std::string &username,
00037         const std::string &email,
00038         double discount,
00039         const std::string &discount_type,
00040         const std::string &premium_card,
00041         const std::string &payment_method,
00042         mongocxx::client &client,
00043         const std::string &profession,
00044         const std::string &registration_date,
00045         double money_spent,
00046         double money_saved,
00047         int ticket_bought,
00048         const std::vector<bsoncxx::document::value> &user_flights,
00049         bool is_admin,
```

```
00050         std::string hashed_admin_password);
00051
00056   Admin(const User &user)
00057       : User(user), hashed_admin_password_("") {}
00058
00059   std::string hashed_admin_password_;
00060
00065   void AddFlight(User &user);
00066
00071   void AddVerificationQuestion(User &user);
00072
00077   void ManageUsers(User &user);
00078
00083   void AddLuggageItem(User &user);
00084 };
00085
00086 #endif  // AIRPORT_ADMIN_H
```

5.3 admin/admin_functions/admin_functions.h File Reference

This file contains the declarations of functions used in the admin dashboard.

```
#include "../../user/user.h"
```

Functions

- void HandleAdminDashboard (Admin &admin, User &user)

  Handles the admin dashboard.
- std::string ProcessAddingFlight ()

  Processes the addition of a flight.
- std::string CaptureInputWithValidation (const std::string &title, const std::string &message, const std::function< bool(const std::string &)> &validator)

  Captures user input with validation.
- std::string CaptureLineWithValidation (const std::string &title, const std::string &message, const std::function< bool(const std::string &)> &validator)

  Captures a line of input with validation.
- std::optional< bool > CaptureBoolWithValidation (const std::string &title, const std::string &message)

  Captures a boolean value with validation.

5.3.1 Detailed Description

This file contains the declarations of functions used in the admin dashboard.

5.3.2 Function Documentation

5.3.2.1 CaptureBoolWithValidation()

```
std::optional< bool > CaptureBoolWithValidation (
            const std::string & title,
            const std::string & message )
```

Captures a boolean value with validation.

Parameters

title	The title of the input field.
message	The message to display to the user.

Returns

An optional boolean value. If the input is valid, the value is the captured input. If the input is invalid, the value is std::nullopt.

5.3.2.2 CaptureInputWithValidation()

```
std::string CaptureInputWithValidation (
            const std::string & title,
            const std::string & message,
            const std::function< bool(const std::string &)> & validator )
```

Captures user input with validation.

Parameters

title	The title of the input field.
message	The message to display to the user.
validator	A function to validate the input.

Returns

The captured input.

5.3.2.3 CaptureLineWithValidation()

```
std::string CaptureLineWithValidation (
            const std::string & title,
            const std::string & message,
            const std::function< bool(const std::string &)> & validator )
```

Captures a line of input with validation.

Parameters

title	The title of the input field.
message	The message to display to the user.
validator	A function to validate the input.

Returns

The captured input.

5.3.2.4 HandleAdminDashboard()

```
void HandleAdminDashboard (
            Admin & admin,
            User & user )
```

Handles the admin dashboard.

Parameters

admin	The admin object.
user	The user object.

5.3.2.5 ProcessAddingFlight()

```
std::string ProcessAddingFlight ( )
```

Processes the addition of a flight.

Returns

A string representing the result of the operation.

5.4 admin_functions.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_ADMIN_ADMIN_FUNCTIONS_ADMIN_FUNCTIONS_H_
00007 #define AIRPORT_ADMIN_ADMIN_FUNCTIONS_ADMIN_FUNCTIONS_H_
00008
00009 #include "../../user/user.h"
00010
00016 void HandleAdminDashboard(Admin &admin, User &user);
00017
00022 std::string ProcessAddingFlight();
00023
00031 std::string CaptureInputWithValidation(
00032     const std::string &title,
00033     const std::string &message,
00034     const std::function<bool(const std::string &)> &validator);
00035
00043 std::string CaptureLineWithValidation(
00044     const std::string &title,
00045     const std::string &message,
00046     const std::function<bool(const std::string &)> &validator);
00047
00054 std::optional<bool> CaptureBoolWithValidation(const std::string &title, const std::string &message);
00055
00056 #endif //AIRPORT_ADMIN_ADMIN_FUNCTIONS_ADMIN_FUNCTIONS_H_
```

5.5 admin/admin_functions/validators.h File Reference

This file contains the declarations of validation functions used in the admin dashboard.

```
#include <string>
```

Functions

- bool ValidateFlightId (const std::string &flight_id)

  Validates a flight ID.
- bool ValidateCity (const std::string &city)

  Validates a city name.
- bool ValidateDate (const std::string &date)

  Validates a date.
- bool ValidateTime (const std::string &time)

  Validates a time.
- bool ValidatePrice (const std::string &price)

  Validates a price.
- bool ValidateNonEmpty (const std::string &input)

  Validates that an input is not empty.
- bool ValidateSolution (const std::string &solution)

  Validates a solution.

5.5.1 Detailed Description

This file contains the declarations of validation functions used in the admin dashboard.

5.5.2 Function Documentation

5.5.2.1 ValidateCity()

```
bool ValidateCity (
            const std::string & city )
```

Validates a city name.

Parameters

city	The city name to validate.

Returns

True if the city name is valid, false otherwise.

5.5.2.2 ValidateDate()

```
bool ValidateDate (
            const std::string & date )
```

Validates a date.

Parameters

date	The date to validate.

Returns

True if the date is valid, false otherwise.

5.5.2.3 ValidateFlightId()

```
bool ValidateFlightId (
            const std::string & flight_id )
```

Validates a flight ID.

Parameters

flight_id	The flight ID to validate.

Returns

True if the flight ID is valid, false otherwise.

5.5.2.4 ValidateNonEmpty()

```
bool ValidateNonEmpty (
            const std::string & input )
```

Validates that an input is not empty.

Parameters

input	The input to validate.

Returns

True if the input is not empty, false otherwise.

5.5.2.5 ValidatePrice()

```
bool ValidatePrice (
            const std::string & price )
```

Validates a price.

Parameters

price	The price to validate.

Returns

True if the price is valid, false otherwise.

5.5.2.6 ValidateSolution()

```
bool ValidateSolution (
            const std::string & solution )
```

Validates a solution.

Parameters

solution	The solution to validate.

Returns

True if the solution is valid, false otherwise.

5.5.2.7 ValidateTime()

```
bool ValidateTime (
            const std::string & time )
```

Validates a time.

Parameters

time	The time to validate.

Returns

True if the time is valid, false otherwise.

5.6 validators.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_ADMIN_ADMIN_FUNCTIONS_VALIDATORS_H_
00007 #define AIRPORT_ADMIN_ADMIN_FUNCTIONS_VALIDATORS_H_
00008
00009 #include <string>
00010
00016 bool ValidateFlightId(const std::string &flight_id);
00017
00023 bool ValidateCity(const std::string &city);
00024
00030 bool ValidateDate(const std::string &date);
00031
00037 bool ValidateTime(const std::string &time);
00038
00044 bool ValidatePrice(const std::string &price);
00045
00051 bool ValidateNonEmpty(const std::string &input);
00052
00058 bool ValidateSolution(const std::string &solution);
00059
00060 #endif //AIRPORT_ADMIN_ADMIN_FUNCTIONS_VALIDATORS_H_
```

5.7 admin/admin_prints/admin_prints.h File Reference

This file contains the declarations of functions used for displaying admin related information.

Functions

- void **DisplayAdminMenu** ()

  Displays the admin menu.
- void **DisplayAddingFlightInfo** ()

  Displays information related to adding a flight.
- std::string DisplayAdminMessageAndCaptureInput (const std::string &title_message, const std::string &text_message)

  Displays a message to the admin and captures their input.
- std::string DisplayAdminMessageAndCaptureLine (const std::string &title_message, const std::string &text_message)

  Displays a message to the admin and captures a line of their input.
- void **DisplayManageUsersMenu** ()

  Displays the manage users menu.

5.7.1 Detailed Description

This file contains the declarations of functions used for displaying admin related information.

5.7.2 Function Documentation

5.7.2.1 DisplayAdminMessageAndCaptureInput()

```
std::string DisplayAdminMessageAndCaptureInput (
            const std::string & title_message,
            const std::string & text_message )
```

Displays a message to the admin and captures their input.

Parameters

title_message	The title of the message.
text_message	The text of the message.

Returns

The input captured from the admin.

5.7.2.2 DisplayAdminMessageAndCaptureLine()

```
std::string DisplayAdminMessageAndCaptureLine (
            const std::string & title_message,
            const std::string & text_message )
```

Displays a message to the admin and captures a line of their input.

Parameters

title_message	The title of the message.
text_message	The text of the message.

Returns

The line of input captured from the admin.

5.8 admin_prints.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_ADMIN_ADMIN_PRINTS_ADMIN_PRINTS_H_
00007 #define AIRPORT_ADMIN_ADMIN_PRINTS_ADMIN_PRINTS_H_
00008
00012 void DisplayAdminMenu();
00013
00017 void DisplayAddingFlightInfo();
00018
00025 std::string DisplayAdminMessageAndCaptureInput(const std::string &title_message, const std::string
      &text_message);
00026
00033 std::string DisplayAdminMessageAndCaptureLine(const std::string &title_message, const std::string
      &text_message);
00034
00038 void DisplayManageUsersMenu();
00039
00040 #endif //AIRPORT_ADMIN_ADMIN_PRINTS_ADMIN_PRINTS_H_
```

5.9 authentication/auth_functions/user_authentication.h File Reference

This file contains the declarations of functions used for user authentication.

```
#include <string>
#include <tuple>
#include "../../user/user.h"
#include "../authentication.h"
```

Functions

- std::tuple< std::string, std::string, std::string, bool > RegisterUser ()

  Registers a new user.
- std::tuple< std::string, std::string, bool > Login ()

  Logs in a user.
- void HandleRegistration (Authentication &auth)

  Handles the registration process.
- bool HandleLogin (Authentication &auth, User &user)

  Handles the login process.

5.9.1 Detailed Description

This file contains the declarations of functions used for user authentication.

5.9.2 Function Documentation

5.9.2.1 HandleLogin()

```
bool HandleLogin (
            Authentication & auth,
            User & user )
```

Handles the login process.

Parameters

auth	The Authentication object.
user	The User object.

Returns

A boolean indicating if the login was successful.

5.9.2.2 HandleRegistration()

```
void HandleRegistration (
            Authentication & auth )
```

Handles the registration process.

Parameters

auth	The Authentication object.

5.9.2.3 Login()

```
std::tuple< std::string, std::string, bool > Login ( )
```

Logs in a user.

Returns

A tuple containing the username, password, and a boolean indicating if the login was successful.

5.9.2.4 RegisterUser()

```
std::tuple< std::string, std::string, std::string, bool > RegisterUser ( )
```

Registers a new user.

Returns

A tuple containing the username, password, email, and a boolean indicating if the registration was successful.

5.10 user_authentication.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_AUTHPRINTHANDLER_H
00007 #define AIRPORT_AUTHPRINTHANDLER_H
00008
00009 #include <string>
00010 #include <tuple>
00011 #include "../../user/user.h"
00012 #include "../authentication.h"
00013
00018 std::tuple<std::string, std::string, std::string, bool> RegisterUser();
00019
00024 std::tuple<std::string, std::string, bool> Login();
00025
00030 void HandleRegistration(Authentication &auth);
00031
00038 bool HandleLogin(Authentication &auth, User &user);
00039
00040 #endif  // AIRPORT_AUTHPRINTHANDLER_H
```

5.11 authentication/authentication.h File Reference

This file contains the declaration of the Authentication class.

```
#include <future>
#include <string>
#include "../user/user.h"
#include "mongocxx/v_noabi/mongocxx/client.hpp"
#include "mongocxx/v_noabi/mongocxx/database.hpp"
#include "mongocxx/v_noabi/mongocxx/instance.hpp"
```

Classes

- class Authentication

  This class handles user authentication.

5.11.1 Detailed Description

This file contains the declaration of the Authentication class.

5.12 authentication.h

Go to the documentation of this file.

```
00001
00006 #ifndef AUTHENTICATION_H
00007 #define AUTHENTICATION_H
00008
00009 #include <future>
00010 #include <string>
00011
00012 #include "../user/user.h"
00013 #include "mongocxx/v_noabi/mongocxx/client.hpp"
00014 #include "mongocxx/v_noabi/mongocxx/database.hpp"
00015 #include "mongocxx/v_noabi/mongocxx/instance.hpp"
00016
00021 class Authentication {
```

```
00022  private:
00023   mongocxx::client _client_;
00024   mongocxx::database _db_;
00025   mongocxx::collection _collection_;
00026
00027  public:
00034   Authentication(const std::string &uri_str, const std::string &db_name, const std::string
      &collection_name);
00035
00041   static std::string HashPassword(const std::string &password);
00042
00050   bool RegisterUser(const std::string &username, const std::string &email, const std::string
      &password);
00051
00059   void AuthenticateUser(const std::string &username,
00060                         const std::string &password,
00061                         std::promise<bool> &&promise,
00062                         User &user);
00063 };
00064
00065 #endif  // AUTHENTICATION_H
```

5.13 checkin/checkin_prints.h File Reference

This file contains the declaration of functions used for check-in operations.

```
#include "../user/user.h"
```

Functions

- void PrintCheckinScreen (User &user)

  Prints the check-in screen for a user.

5.13.1 Detailed Description

This file contains the declaration of functions used for check-in operations.

5.13.2 Function Documentation

5.13.2.1 PrintCheckinScreen()

```
void PrintCheckinScreen (
            User & user )
```

Prints the check-in screen for a user.

Parameters

user	The user for whom the check-in screen is printed.

5.14 checkin_prints.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_CHECKIN_FUNCTIONS_H
00007 #define AIRPORT_CHECKIN_FUNCTIONS_H
00008
00009 #include "../user/user.h"
00010
00015 void PrintCheckinScreen(User &user);
00016
00017 #endif  // AIRPORT_CHECKIN_FUNCTIONS_H
```

5.15 env/env.h File Reference

This file contains the declaration of the EnvParser class.

```
#include <string>
#include <unordered_map>
```

Classes

- class EnvParser

  This class is used for parsing environment variables.

5.15.1 Detailed Description

This file contains the declaration of the EnvParser class.

5.16 env.h

Go to the documentation of this file.

```
00001
00006 #ifndef ENVPARSER_H
00007 #define ENVPARSER_H
00008
00009 #include <string>
00010 #include <unordered_map>
00011
00016 class EnvParser {
00017  private:
00018   std::unordered_map<std::string, std::string> _env_map_;
00019
00020  public:
00024   EnvParser();
00025
00029   void ParseEnvFile();
00030
00036   [[nodiscard]] std::string GetValue(const std::string &key) const;
00037 };
00038
00039 #endif  // ENVPARSER_H
```

5.17 flights/flight_connection.h File Reference

This file contains the declaration of the FlightConnection class.

```
#include <string>
#include "mongocxx/v_noabi/mongocxx/client.hpp"
#include "mongocxx/v_noabi/mongocxx/collection.hpp"
#include "mongocxx/v_noabi/mongocxx/database.hpp"
```

Classes

- class FlightConnection

  This class handles flight connections.

5.17.1 Detailed Description

This file contains the declaration of the FlightConnection class.

5.18 flight_connection.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <string>
00009
00010 #include "mongocxx/v_noabi/mongocxx/client.hpp"
00011 #include "mongocxx/v_noabi/mongocxx/collection.hpp"
00012 #include "mongocxx/v_noabi/mongocxx/database.hpp"
00013
00018 class FlightConnection {
00019  private:
00020   std::string _flight_id_;
00021   std::string _departureCity_;
00022   std::string _destinationTime_;
00023   std::string _arrivalTime_;
00024   std::string _departureTime_;
00025   std::string _destinationCity_;
00026   int _availableSeats_{};
00027   double _price_{};
00028
00029   mongocxx::client _client_;
00030   mongocxx::database _db_;
00031   mongocxx::collection _collection_;
00032
00033  public:
00040   FlightConnection(const std::string &uri_str, const std::string &db_name, const std::string
      &collection_name);
00041
00052   FlightConnection(
00053       std::string flight_id,
00054       std::string departure_city,
00055       std::string destination_city,
00056       std::string departure_time,
00057       std::string arrival_time,
00058       int available_seats,
00059       double price);
00060
00065   [[nodiscard]] std::string GetDepartureCity() const;
00066
00071   [[nodiscard]] std::string GetDestinationCity() const;
00072
00077   [[nodiscard]] std::string GetDepartureTime() const;
00078
00083   [[nodiscard]] std::string GetArrivalTime() const;
```

```
00084
00089   [[nodiscard]] std::string GetIdentifier() const;
00090
00095   [[nodiscard]] int GetAvailableSeats() const;
00096
00101   [[nodiscard]] double GetPrice() const;
00102
00107   std::vector<FlightConnection> FindAllConnections();
00108
00115   FlightConnection FindConnection(const std::string &departure_city, const std::string
      &destination_city);
00116
00123   std::vector<FlightConnection> FindConnectionByPrice(double &min_price, double &max_price);
00124
00130   FlightConnection FindConnectionById(const std::string &id);
00131
00137   std::vector<FlightConnection> FindConnectionsByDeparture(const std::string &departure_city);
00138
00144   std::vector<FlightConnection> FindConnectionsByDestination(const std::string &destination_city);
00145
00151   std::vector<int> GetSeatsTaken(const std::string &flight_identifier);
00152
00158   void UpdateSeatsTaken(const std::string &flight_identifier, const std::vector<int> &seats_taken);
00159 };
```

5.19 functions/helpers.h File Reference

This file contains the declaration of various helper functions.

```
#include <string>
#include "ftxui/dom/table.hpp"
#include "ftxui/screen/color.hpp"
```

Functions

- std::string ExtractFileName (const std::string &path)

  Extracts the file name from a path.
- void Countdown (int seconds, const std::string &type)

  Performs a countdown.
- std::string HashString (const std::string &string_to_hash)

  Hashes a string.
- void SetCellColor (ftxui::Table &table, int col, int row, ftxui::Color color)

  Sets the color of a cell in a table.

5.19.1 Detailed Description

This file contains the declaration of various helper functions.

5.19.2 Function Documentation

5.19.2.1 Countdown()

```
void Countdown (
            int seconds,
            const std::string & type )
```

Performs a countdown.

Parameters

seconds	The number of seconds to countdown.
type	The type of countdown.

5.19.2.2 ExtractFileName()

```
std::string ExtractFileName (
            const std::string & path )
```

Extracts the file name from a path.

Parameters

path	The path to extract the file name from.

Returns

The extracted file name.

5.19.2.3 HashString()

```
std::string HashString (
            const std::string & string_to_hash )
```

Hashes a string.

Parameters

string_to_hash	The string to hash.

Returns

The hashed string.

5.19.2.4 SetCellColor()

```
void SetCellColor (
            ftxui::Table & table,
            int col,
            int row,
            ftxui::Color color )
```

Sets the color of a cell in a table.

Parameters

table	The table containing the cell.
col	The column of the cell.
row	The row of the cell.
color	The color to set the cell to.

5.20 helpers.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_HELPERS_H
00007 #define AIRPORT_HELPERS_H
00008
00009 #include <string>
00010 #include "ftxui/dom/table.hpp"
00011 #include "ftxui/screen/color.hpp"
00012
00018 std::string ExtractFileName(const std::string &path);
00019
00025 void Countdown(int seconds, const std::string &type);
00026
00032 std::string HashString(const std::string &string_to_hash);
00033
00041 void SetCellColor(ftxui::Table &table, int col, int row, ftxui::Color color);
00042
00043 #endif  // AIRPORT_HELPERS_H
```

5.21 info_prints.h

```
00001
00006 #ifndef FUNCTIONS_H
00007 #define FUNCTIONS_H
00008
00009 #include <string>
00010
00011 #include "../../user/user.h"
00012
00018 void PrintSuccessMessage(const std::string &title_message, const std::string &optional_message);
00019
00025 void PrintErrorMessage(const std::string &title_message, const std::string &optional_message);
00026
00031 void PrintLogout(User &user);
00032
00036 void PrintSeeya();
00037
00038 #endif  // FUNCTIONS_H
```

5.22 main_handler.h

```
00001
00006 #ifndef MAIN_FUNCTIONS_H
00007 #define MAIN_FUNCTIONS_H
00008
00009 #include "../authentication/authentication.h"
00010 #include "../flights/flight_connection.h"
00011
00019 void ProcessChoice(bool is_logged_in, Authentication &auth, User &user, FlightConnection
      &flight_connection);
00020
00021 #endif  // MAIN_FUNCTIONS_H
```

5.23 functions/main_prints/main_prints.h File Reference

This file contains the declaration of various display and input capture functions.

```
#include <memory>
#include "ftxui/dom/elements.hpp"
#include "../../user/user.h"
```

Functions

- void PrintScreen (const std::shared_ptr< ftxui::Element > &screen)

  Prints a screen.
- void PrintFullWidthScreen (std::shared_ptr< ftxui::Node > container)

  Prints a full width screen.
- void PrintNodeScreen (std::shared_ptr< ftxui::Node > container)

  Prints a node screen.
- std::string DisplayMessageAndCaptureStringInput (const std::string &title_message, const std::string &text↩ _message)

  Displays a message and captures string input.
- double DisplayMessageAndCaptureDoubleInput (const std::string &title_message, const std::string &text_↩ message)

  Displays a message and captures double input.
- std::string DisplayWarningAndCaptureInput (const std::string &title_message, const std::string &text_↩ message)

  Displays a warning and captures input.
- void DisplayUserMenu (User &user)

  Displays the user menu.
- void **DisplayMenu** ()

  Displays the main menu.

5.23.1 Detailed Description

This file contains the declaration of various display and input capture functions.

5.23.2 Function Documentation

5.23.2.1 DisplayMessageAndCaptureDoubleInput()

```
double DisplayMessageAndCaptureDoubleInput (
            const std::string & title_message,
            const std::string & text_message )
```

Displays a message and captures double input.

Parameters

title_message	The title of the message.
text_message	The text of the message.

Returns

The captured double input.

5.23.2.2 DisplayMessageAndCaptureStringInput()

```
std::string DisplayMessageAndCaptureStringInput (
            const std::string & title_message,
            const std::string & text_message )
```

Displays a message and captures string input.

Parameters

title_message	The title of the message.
text_message	The text of the message.

Returns

The captured string input.

5.23.2.3 DisplayUserMenu()

```
void DisplayUserMenu (
            User & user )
```

Displays the user menu.

Parameters

user	The user for whom the menu is displayed.

5.23.2.4 DisplayWarningAndCaptureInput()

```
std::string DisplayWarningAndCaptureInput (
            const std::string & title_message,
            const std::string & text_message )
```

Displays a warning and captures input.

Parameters

title_message	The title of the message.
text_message	The text of the message.

Returns

The captured input.

5.23.2.5 PrintFullWidthScreen()

```
void PrintFullWidthScreen (
            std::shared_ptr< ftxui::Node > container )
```

Prints a full width screen.

Parameters

container	The container to print.

5.23.2.6 PrintNodeScreen()

```
void PrintNodeScreen (
            std::shared_ptr< ftxui::Node > container )
```

Prints a node screen.

Parameters

container	The container to print.

5.23.2.7 PrintScreen()

```
void PrintScreen (
            const std::shared_ptr< ftxui::Element > & screen )
```

Prints a screen.

Parameters

screen	The screen to print.

5.24 main_prints.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_MAIN_PRINTS_H
00007 #define AIRPORT_MAIN_PRINTS_H
00008
00009 #include <memory>
00010
00011 #include "ftxui/dom/elements.hpp"
00012 #include "../../user/user.h"
00013
```

```
00018 void PrintScreen(const std::shared_ptr<ftxui::Element> &screen);
00019
00024 void PrintFullWidthScreen(std::shared_ptr<ftxui::Node> container);
00025
00030 void PrintNodeScreen(std::shared_ptr<ftxui::Node> container);
00031
00038 std::string DisplayMessageAndCaptureStringInput(const std::string &title_message, const std::string
      &text_message);
00039
00046 double DisplayMessageAndCaptureDoubleInput(const std::string &title_message, const std::string
      &text_message);
00047
00054 std::string DisplayWarningAndCaptureInput(const std::string &title_message, const std::string
      &text_message);
00055
00060 void DisplayUserMenu(User &user);
00061
00065 void DisplayMenu();
00066
00067 #endif  // AIRPORT_MAIN_PRINTS_H
```

5.25 luggage/item/item.h File Reference

This file contains the declaration of the Item class.

```
#include <string>
#include <vector>
```

Classes

- class Item

  This class represents an item in the airport system.

5.25.1 Detailed Description

This file contains the declaration of the Item class.

5.26 item.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_ITEM_H
00007 #define AIRPORT_ITEM_H
00008
00009 #include <string>
00010 #include <vector>
00011
00012 class User;
00013
00018 class Item {
00019   std::string item_name_;
00020   std::string description_;
00021   std::vector<std::string> hints_;
00022   bool forbidden_;
00023   bool registered_luggage_;
00024   bool hand_luggage_;
00025   bool pilot_allowance_;
00026   double max_count_;
00027   double weight_;
00028   std::string profession_;
00029   std::string category_ = "special";
00030
00031  public:
```

```
00046   Item(const std::string &item_name,
00047        const std::string &description,
00048        const std::vector<std::string> &hints,
00049        bool forbidden,
00050        bool registered_luggage,
00051        bool hand_luggage,
00052        bool pilot_allowance,
00053        double max_count,
00054        double weight,
00055        std::string &profession,
00056        std::string &category);
00057
00071   Item(const std::string &item_name,
00072        const std::string &description,
00073        const std::vector<std::string> &hints,
00074        bool forbidden,
00075        bool registered_luggage,
00076        bool hand_luggage,
00077        bool pilot_allowance,
00078        double max_count,
00079        double weight,
00080        std::string &category);
00081
00086   [[nodiscard]] const std::string &GetItemName() const;
00087
00092   [[nodiscard]] const std::string &GetDescription() const;
00093
00098   [[nodiscard]] const std::string &GetProfession() const;
00099
00104   [[nodiscard]] const std::vector<std::string> &GetHints() const;
00105
00110   [[nodiscard]] bool IsForbidden() const;
00111
00116   [[nodiscard]] bool IsRegisteredLuggage() const;
00117
00122   [[nodiscard]] bool IsHandLuggage() const;
00123
00128   [[nodiscard]] bool IsPilotAllowance() const;
00129
00134   [[nodiscard]] double GetMaxCount() const;
00135
00140   [[nodiscard]] double GetWeight() const;
00141
00146   [[nodiscard]] std::string GetCategory() const;
00147 };
00148
00149 #endif  // AIRPORT_ITEM_H
```

5.27 luggage/item/item_handler.h File Reference

This file contains the declaration of various item handling functions.

```
#include <vector>
#include "../../user/user.h"
#include "item.h"
```

Functions

- double GetDoubleValue (const bsoncxx::document::view &item, const std::string &key)

  Gets a double value from a BSON document.
- std::vector< std::string > GetArrayValue (const bsoncxx::document::view &item, const std::string &key)

  Gets an array value from a BSON document.
- std::string GetStringValue (const bsoncxx::document::view &item, const std::string &key)

  Gets a string value from a BSON document.
- std::vector< Item > GetItems (User &user)

  Gets the items for a user.

5.27.1 Detailed Description

This file contains the declaration of various item handling functions.

5.27.2 Function Documentation

5.27.2.1 GetArrayValue()

```
std::vector< std::string > GetArrayValue (
            const bsoncxx::document::view & item,
            const std::string & key )
```

Gets an array value from a BSON document.

Parameters

item	The BSON document.
key	The key of the value to get.

Returns

The array value.

5.27.2.2 GetDoubleValue()

```
double GetDoubleValue (
            const bsoncxx::document::view & item,
            const std::string & key )
```

Gets a double value from a BSON document.

Parameters

item	The BSON document.
key	The key of the value to get.

Returns

The double value.

5.27.2.3 GetItems()

```
std::vector< Item > GetItems (
            User & user )
```

Gets the items for a user.

Parameters

user	The user to get the items for.

Returns

The items for the user.

5.27.2.4 GetStringValue()

```
std::string GetStringValue (
            const bsoncxx::document::view & item,
            const std::string & key )
```

Gets a string value from a BSON document.

Parameters

item	The BSON document.
key	The key of the value to get.

Returns

The string value.

5.28 item_handler.h

Go to the documentation of this file.

```
00001
00006 #ifndef ITEM_HANDLER_H
00007 #define ITEM_HANDLER_H
00008
00009 #include <vector>
00010
00011 #include "../../user/user.h"
00012 #include "item.h"
00013
00020 double GetDoubleValue(const bsoncxx::document::view &item, const std::string &key);
00021
00028 std::vector<std::string> GetArrayValue(const bsoncxx::document::view &item, const std::string &key);
00029
00036 std::string GetStringValue(const bsoncxx::document::view &item, const std::string &key);
00037
00043 std::vector<Item> GetItems(User &user);
00044
00045 #endif  // ITEM_HANDLER_H
```

5.29 luggage/luggage.h File Reference

This file contains the declaration of the Luggage class.

```
#include <vector>
#include <iostream>
#include "item/item.h"
```

Classes

- class Luggage

  This class represents a luggage in the airport system.

5.29.1 Detailed Description

This file contains the declaration of the Luggage class.

5.30 luggage.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_LUGGAGE_H
00007 #define AIRPORT_LUGGAGE_H
00008
00009 #include <vector>
00010 #include <iostream>
00011 #include "item/item.h"
00012
00013 class User;
00014
00019 class Luggage {
00020   std::vector<Item> items_;
00021   double total_weight_ = 0.0;
00022
00023  public:
00029   Luggage(
00030       const std::vector<Item> &items,
00031       double total_weight
00032   ) : items_(items), total_weight_(total_weight) {}
00033
00038   double ProcessItemsAndGetWeight();
00039
00045   std::tuple<bool, std::string> ConfirmItems(User &user);
00046
00047   const double overweight_fee_per_kg_ = 2.0;
00048   const double euro_to_pln_ = 4.32;
00049   const double max_allowed_weight_ = 32.0;
00050   double max_weight_ = 20.0;
00051
00057   double CalculateOverweightFee(double weight) const;
00058 };
00059
00060 #endif //AIRPORT_LUGGAGE_H
```

5.31 luggage/luggage_handler.h File Reference

This file contains the declaration of the CheckIn function.

```
#include "../user/user.h"
```

Functions

- void CheckIn (User &user, int flightNumber)

  Checks in a user for a specific flight.

5.31.1 Detailed Description

This file contains the declaration of the CheckIn function.

5.31.2 Function Documentation

5.31.2.1 CheckIn()

```
void CheckIn (
            User & user,
            int flightNumber )
```

Checks in a user for a specific flight.

Parameters

user	The user to check in.
flightNumber	The number of the flight the user is checking in for.

5.32 luggage_handler.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_LUGGAGEHANDLER_H
00007 #define AIRPORT_LUGGAGEHANDLER_H
00008
00009 #include "../user/user.h"
00010
00016 void CheckIn(User &user, int flightNumber);
00017
00018 #endif  // AIRPORT_LUGGAGEHANDLER_H
```

5.33 luggage/luggage_prints/luggage_prints.h File Reference

This file contains the declaration of various luggage handling functions.

```
#include "../../user/user.h"
#include "../item/item.h"
#include "ftxui/component/component.hpp"
```

Functions

- std::vector< ftxui::Component > CreateGroups (const std::vector< ftxui::Component > &checkbox_components)

  Creates groups of components.
- void PrintAllItems (User &user)

  Prints all items for a user.
- void PrintSpecificItem (Item &item)

  Prints a specific item.
- void PrintWelcomeInCheckIn (User &user)

  Prints a welcome message in the check-in.

Variables

- const std::string **AIRPORT_NAME** = "WOLFI AIRPORT "

  The name of the airport.
- const std::string **ITEM_CARD** = "KARTA PRZEDMIOTU"

  The item card.

5.33.1 Detailed Description

This file contains the declaration of various luggage handling functions.

5.33.2 Function Documentation

5.33.2.1 CreateGroups()

```
std::vector< ftxui::Component > CreateGroups (
            const std::vector< ftxui::Component > & checkbox_components )
```

Creates groups of components.

Parameters

checkbox_components	The components to group.

Returns

The groups of components.

5.33.2.2 PrintAllItems()

```
void PrintAllItems (
            User & user )
```

Prints all items for a user.

Parameters

user	The user to print the items for.

5.33.2.3 PrintSpecificItem()

```
void PrintSpecificItem (
            Item & item )
```

Prints a specific item.

Parameters

item	The item to print.

5.33.2.4 PrintWelcomeInCheckIn()

```
void PrintWelcomeInCheckIn (
            User & user )
```

Prints a welcome message in the check-in.

Parameters

user	The user to print the welcome message for.

5.34 luggage_prints.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_LUGGAGE_PRINTS_H
00007 #define AIRPORT_LUGGAGE_PRINTS_H
00008
00009 #include "../../user/user.h"
00010 #include "../item/item.h"
00011 #include "ftxui/component/component.hpp"
00012
00013 const std::string AIRPORT_NAME = "WOLFI AIRPORT  ";
00014 const std::string ITEM_CARD = "KARTA PRZEDMIOTU";
00015
00021 std::vector<ftxui::Component> CreateGroups(const std::vector<ftxui::Component> &checkbox_components);
00022
00027 void PrintAllItems(User &user);
00028
00033 void PrintSpecificItem(Item &item);
00034
00039 void PrintWelcomeInCheckIn(User &user);
00040
00041 #endif  // AIRPORT_LUGGAGE_PRINTS_H
```

5.35 plane/plane.h File Reference

This file contains the declaration of the ProcessSeatSelectionAndPurchase function.

```
#include <vector>
#include "../flights/flight_connection.h"
#include "../user/user.h"
```

Functions

- void ProcessSeatSelectionAndPurchase (std::vector< int > seat_number, FlightConnection &flight_↩ connection, FlightConnection &found_connection, User &user)

  Processes the seat selection and purchase for a flight.

5.35.1 Detailed Description

This file contains the declaration of the ProcessSeatSelectionAndPurchase function.

5.35.2 Function Documentation

5.35.2.1 ProcessSeatSelectionAndPurchase()

```
void ProcessSeatSelectionAndPurchase (
            std::vector< int > seat_number,
            FlightConnection & flight_connection,
            FlightConnection & found_connection,
            User & user )
```

Processes the seat selection and purchase for a flight.

Parameters

seat_number	The seat numbers selected.
flight_connection	The flight connection.
found_connection	The found flight connection.
user	The user making the purchase.

5.36 plane.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_PLANE_H
00007 #define AIRPORT_PLANE_H
00008
00009 #include <vector>
00010
00011 #include "../flights/flight_connection.h"
00012 #include "../user/user.h"
00013
00021 void ProcessSeatSelectionAndPurchase(std::vector<int> seat_number,
00022                                      FlightConnection &flight_connection,
00023                                      FlightConnection &found_connection,
00024                                      User &user);
00025
00026 #endif  // AIRPORT_PLANE_H
```

5.37 qr_code/qrcode_prints.h File Reference

This file contains the declaration of QR code creation and printing functions.

```
#include "qrcodegen.hpp"
```

Functions

- void CreateQr (const std::string &email, const std::string &username, const std::string &flight_id, std::vector< int > seats)

  Creates a QR code based on user and flight information.
- void PrintQr (const QrCode &qr)

  Prints a QR code.

5.37.1 Detailed Description

This file contains the declaration of QR code creation and printing functions.

5.37.2 Function Documentation

5.37.2.1 CreateQr()

```
void CreateQr (
            const std::string & email,
            const std::string & username,
            const std::string & flight_id,
            std::vector< int > seats )
```

Creates a QR code based on user and flight information.

Parameters

email	The email of the user.
username	The username of the user.
flight_id	The ID of the flight.
seats	The seats selected by the user.

5.37.2.2 PrintQr()

```
void PrintQr (
            const QrCode & qr )
```

Prints a QR code.

Parameters

qr	The QR code to print.

5.38 qrcode_prints.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_QRCODE_PRINTS_H
00007 #define AIRPORT_QRCODE_PRINTS_H
00008
00009 #include "qrcodegen.hpp"
00010
00011 using qrcodegen::QrCode;
00012 using qrcodegen::QrSegment;
00013 using std::uint8_t;
00014
00022 void CreateQr(const std::string &email,
00023               const std::string &username,
00024               const std::string &flight_id,
00025               std::vector<int> seats);
00026
00031 void PrintQr(const QrCode &qr);
00032
00033 #endif  // AIRPORT_QRCODE_PRINTS_H
```

5.39 tickets/tickets.h File Reference

This file contains the declaration of various ticket handling functions.

```
#include "../flights/flight_connection.h"
#include "../user/user.h"
```

Functions

- void HandleTicketChoice (FlightConnection &flight_connection, User &user)

  Handles the ticket choice of a user.
- void HandleBuyTicket (int choice, FlightConnection &flight_connection, User &user)

  Handles the purchase of a ticket.
- void HandleFlightById (FlightConnection &flight_connection, User &user)

  Handles the flight by its ID.
- void HandleFlightByData (FlightConnection &flight_connection, User &user)

  Handles the flight by its data.
- void ProcessPurchase (FlightConnection &flight_connection, FlightConnection &found_connection, User &user)

  Processes the purchase of a ticket.

Variables

- const int **MAX_TICKETS** = 4

  The maximum number of tickets.
- const int **EMERGENCY_SEAT_ONE** = 37

  The first emergency seat number.
- const int **EMERGENCY_SEAT_TWO** = 45

  The second emergency seat number.

5.39.1 Detailed Description

This file contains the declaration of various ticket handling functions.

5.39.2 Function Documentation

5.39.2.1 HandleBuyTicket()

```
void HandleBuyTicket (
            int choice,
            FlightConnection & flight_connection,
            User & user )
```

Handles the purchase of a ticket.

Parameters

choice	The choice of the user.
flight_connection	The flight connection.
user	The user making the purchase.

5.39.2.2 HandleFlightByData()

```
void HandleFlightByData (
            FlightConnection & flight_connection,
            User & user )
```

Handles the flight by its data.

Parameters

flight_connection	The flight connection.
user	The user.

5.39.2.3 HandleFlightById()

```
void HandleFlightById (
            FlightConnection & flight_connection,
            User & user )
```

Handles the flight by its ID.

Parameters

flight_connection	The flight connection.
user	The user.

5.39.2.4 HandleTicketChoice()

```
void HandleTicketChoice (
            FlightConnection & flight_connection,
            User & user )
```

Handles the ticket choice of a user.

Parameters

flight_connection	The flight connection.
user	The user making the choice.

5.39.2.5 ProcessPurchase()

```
void ProcessPurchase (
            FlightConnection & flight_connection,
            FlightConnection & found_connection,
            User & user )
```

Processes the purchase of a ticket.

Parameters

flight_connection	The flight connection.
found_connection	The found flight connection.
user	The user making the purchase.

5.40 tickets.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_TICKETS_H
00007 #define AIRPORT_TICKETS_H
00008
00009 #include "../flights/flight_connection.h"
00010 #include "../user/user.h"
00011
00012 const int MAX_TICKETS = 4;
00013 const int EMERGENCY_SEAT_ONE = 37;
00014 const int EMERGENCY_SEAT_TWO = 45;
00015
00021 void HandleTicketChoice(FlightConnection &flight_connection, User &user);
00022
00029 void HandleBuyTicket(int choice, FlightConnection &flight_connection, User &user);
00030
00036 void HandleFlightById(FlightConnection &flight_connection, User &user);
00037
00043 void HandleFlightByData(FlightConnection &flight_connection, User &user);
00044
00051 void ProcessPurchase(
00052     FlightConnection &flight_connection,
00053     FlightConnection &found_connection,
00054     User &user);
00055
00056 #endif  // AIRPORT_TICKETS_H
```

5.41 user/discounts/discounts.h File Reference

This file contains the declaration of various discount handling functions.

```
#include "../user.h"
```

Functions

- double GetDiscount (std::string choice)

  Gets the discount based on a choice.
- void HandleDiscountChoice (User &user, std::string choice)

  Handles the discount choice of a user.
- void PrintDiscountCard (User &user)

  Prints a discount card for a user.

5.41.1 Detailed Description

This file contains the declaration of various discount handling functions.

5.41.2 Function Documentation

5.41.2.1 GetDiscount()

```
double GetDiscount (
            std::string choice )
```

Gets the discount based on a choice.

Parameters

choice	The choice of the user.

Returns

The discount as a double.

5.41.2.2 HandleDiscountChoice()

```
void HandleDiscountChoice (
            User & user,
            std::string choice )
```

Handles the discount choice of a user.

Parameters

user	The user making the choice.
choice	The choice of the user.

5.41.2.3 PrintDiscountCard()

```
void PrintDiscountCard (
            User & user )
```

Prints a discount card for a user.

Parameters

user	The user for whom the discount card is printed.

5.42 discounts.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_DISCOUNTS_H
00007 #define AIRPORT_DISCOUNTS_H
00008
00009 #include "../user.h"
00010
00016 double GetDiscount(std::string choice);
00017
00023 void HandleDiscountChoice(User &user, std::string choice);
00024
00029 void PrintDiscountCard(User &user);
00030
00031 #endif  // AIRPORT_DISCOUNTS_H
```

5.43 user/premium_cards/premium_cards.h File Reference

This file contains the declaration of various premium card handling functions.

```
#include "../user.h"
```

Functions

- void HandlePremiumCard (User &user)

  Handles the premium card of a user.
- void HandleCardChoice (const std::string &card, int price, User &user)

  Handles the card choice of a user.
- double GetCardDiscount (const std::string &card)

  Gets the discount of a card.
- std::string RecognizeDiscountCard (double discount)

  Recognizes a discount card based on a discount.

5.43.1 Detailed Description

This file contains the declaration of various premium card handling functions.

5.43.2 Function Documentation

5.43.2.1 GetCardDiscount()

```
double GetCardDiscount (
            const std::string & card )
```

Gets the discount of a card.

Parameters

card	The card to get the discount for.

Returns

The discount as a double.

5.43.2.2 HandleCardChoice()

```
void HandleCardChoice (
            const std::string & card,
            int price,
            User & user )
```

Handles the card choice of a user.

Parameters

card	The card chosen by the user.
price	The price of the card.
user	The user making the choice.

5.43.2.3 HandlePremiumCard()

```
void HandlePremiumCard (
            User & user )
```

Handles the premium card of a user.

Parameters

user	The user with the premium card.

5.43.2.4 RecognizeDiscountCard()

```
std::string RecognizeDiscountCard (
            double discount )
```

Recognizes a discount card based on a discount.

Parameters

discount	The discount to recognize the card from.

Returns

The recognized card as a string.

5.44 premium_cards.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_PREMIUM_CARDS_H
00007 #define AIRPORT_PREMIUM_CARDS_H
00008
00009 #include "../user.h"
00010
00015 void HandlePremiumCard(User &user);
00016
00023 void HandleCardChoice(const std::string &card, int price, User &user);
00024
00030 double GetCardDiscount(const std::string &card);
00031
00037 std::string RecognizeDiscountCard(double discount);
00038
00039 #endif  // AIRPORT_PREMIUM_CARDS_H
```

5.45 user/professions/profession_choice.h File Reference

This file contains the declaration of various profession choice functions.

```
#include "../user.h"
```

Functions

- void MusicProfession (User &user)
  Handles the music profession choice of a user.
- void MathProfession (User &user)
  Handles the math profession choice of a user.
- void InformaticProfession (User &user)
  Handles the informatics profession choice of a user.
- void DoctorProfession (User &user)
  Handles the doctor profession choice of a user.
- void PoliceProfession (User &user)
  Handles the police profession choice of a user.

5.45.1 Detailed Description

This file contains the declaration of various profession choice functions.

5.45.2 Function Documentation

5.45.2.1 DoctorProfession()

```
void DoctorProfession (
            User & user )
```

Handles the doctor profession choice of a user.

Parameters

user	The user making the choice.

5.45.2.2 InformaticProfession()

```
void InformaticProfession (
            User & user )
```

Handles the informatics profession choice of a user.

Parameters

user	The user making the choice.

5.45.2.3 MathProfession()

```
void MathProfession (
            User & user )
```

Handles the math profession choice of a user.

Parameters

user	The user making the choice.

5.45.2.4 MusicProfession()

```
void MusicProfession (
            User & user )
```

Handles the music profession choice of a user.

Parameters

user	The user making the choice.

5.45.2.5 PoliceProfession()

```
void PoliceProfession (
            User & user )
```

Handles the police profession choice of a user.

Parameters

user	The user making the choice.

5.46 profession_choice.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_PROFESSION_CHOICE_H
00007 #define AIRPORT_PROFESSION_CHOICE_H
00008
00009 #include "../user.h"
00010
00015 void MusicProfession(User &user);
00016
00021 void MathProfession(User &user);
00022
00027 void InformaticProfession(User &user);
00028
00033 void DoctorProfession(User &user);
00034
00039 void PoliceProfession(User &user);
00040
00041 #endif  // AIRPORT_PROFESSION_CHOICE_H
```

5.47 user/professions/profession_handler.h File Reference

This file contains the declaration of various profession related question handling functions.

```
#include <string>
#include "../user.h"
```

Functions

- bool GuessMusicAuthor (const std::string &music_link)

  Guesses the author of a music piece given a link.
- bool GuessDoctorQuestion (User &user)

  Guesses the answer to a doctor related question.
- bool GuessInformaticQuestion (User &user)

  Guesses the answer to an informatics related question.
- bool GuessMathQuestion (User &user)

  Guesses the answer to a math related question.
- bool DisplayPoliceProfession ()

  Displays information related to the police profession.

5.47.1 Detailed Description

This file contains the declaration of various profession related question handling functions.

5.47.2 Function Documentation

5.47.2.1 DisplayPoliceProfession()

```
bool DisplayPoliceProfession ( )
```

Displays information related to the police profession.

Returns

True if the operation is successful, false otherwise.

5.47.2.2 GuessDoctorQuestion()

```
bool GuessDoctorQuestion (
            User & user )
```

Guesses the answer to a doctor related question.

Parameters

user	The user making the guess.

Returns

True if the guess is correct, false otherwise.

5.47.2.3 GuessInformaticQuestion()

```
bool GuessInformaticQuestion (
            User & user )
```

Guesses the answer to an informatics related question.

Parameters

user	The user making the guess.

Returns

True if the guess is correct, false otherwise.

5.47.2.4 GuessMathQuestion()

```
bool GuessMathQuestion (
            User & user )
```

Guesses the answer to a math related question.

Parameters

user	The user making the guess.

Returns

True if the guess is correct, false otherwise.

5.47.2.5 GuessMusicAuthor()

```
bool GuessMusicAuthor (
            const std::string & music_link )
```

Guesses the author of a music piece given a link.

Parameters

music_link	The link to the music piece.

Returns

True if the guess is correct, false otherwise.

5.48 profession_handler.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_PROFESSION_HANDLER_H
00007 #define AIRPORT_PROFESSION_HANDLER_H
00008
00009 #include <string>
00010
00011 #include "../user.h"
00012
00018 bool GuessMusicAuthor(const std::string &music_link);
00019
00025 bool GuessDoctorQuestion(User &user);
00026
00032 bool GuessInformaticQuestion(User &user);
00033
00039 bool GuessMathQuestion(User &user);
00040
00045 bool DisplayPoliceProfession();
00046
00047 #endif  // AIRPORT_PROFESSION_HANDLER_H
```

5.49 user/professions/profession_prints/profession_prints.h File Reference

This file contains the declaration of profession information display and validation functions.

```
#include <string>
#include "../../user.h"
```

Functions

- int CreateProfessionScreen ()

  Creates a profession screen.
- std::string DisplayProfessionInfo ()

  Displays profession information.
- void **InvalidAnswer** ()

  Handles an invalid answer.
- void ValidAnswer (const std::string &category, User &user)

  Handles a valid answer.

5.49.1 Detailed Description

This file contains the declaration of profession information display and validation functions.

5.49.2 Function Documentation

5.49.2.1 CreateProfessionScreen()

```
int CreateProfessionScreen ( )
```

Creates a profession screen.

Returns

An integer representing the status of the operation.

5.49.2.2 DisplayProfessionInfo()

```
std::string DisplayProfessionInfo ( )
```

Displays profession information.

Returns

A string containing the profession information.

5.49.2.3 ValidAnswer()

```
void ValidAnswer (
            const std::string & category,
            User & user )
```

Handles a valid answer.

Parameters

category	The category of the answer.
user	The user providing the answer.

5.50 profession_prints.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_PROFESSION_PRINTS_H
00007 #define AIRPORT_PROFESSION_PRINTS_H
00008
00009 #include <string>
00010
00011 #include "../../user.h"
00012
00017 int CreateProfessionScreen();
00018
00023 std::string DisplayProfessionInfo();
00024
00028 void InvalidAnswer();
00029
00035 void ValidAnswer(const std::string &category, User &user);
00036
00037 #endif  // AIRPORT_PROFESSION_PRINTS_H
```

5.51 user/professions/user_profession_functions.h File Reference

This file contains the declaration of user profession handling functions.

```
#include "../user.h"
```

Functions

- void HandleProfessionChoice (int choice, User &user)

  Handles the profession choice of a user.
- void HandleProfession (User &user)

  Handles the profession of a user.

5.51.1 Detailed Description

This file contains the declaration of user profession handling functions.

5.51.2 Function Documentation

5.51.2.1 HandleProfession()

```
void HandleProfession (
            User & user )
```

Handles the profession of a user.

Parameters

user	The user whose profession is being handled.

5.51.2.2 HandleProfessionChoice()

```
void HandleProfessionChoice (
            int choice,
            User & user )
```

Handles the profession choice of a user.

Parameters

choice	The choice made by the user.
user	The user making the choice.

5.52 user_profession_functions.h

Go to the documentation of this file.

```
00001
00006 #ifndef AIRPORT_USER_PROFESSION_FUNCTIONS_H
00007 #define AIRPORT_USER_PROFESSION_FUNCTIONS_H
00008
00009 #include "../user.h"
00010
00016 void HandleProfessionChoice(int choice, User &user);
00017
00022 void HandleProfession(User &user);
00023
00024 #endif  // AIRPORT_USER_PROFESSION_FUNCTIONS_H
```

5.53 user/user.h File Reference

This file contains the declaration of the User class.

```
#include <string>
#include "../flights/flight_connection.h"
#include "../luggage/luggage.h"
#include "mongocxx/client.hpp"
```

Classes

- class User

  Represents a user in the system.

5.53.1 Detailed Description

This file contains the declaration of the User class.

5.54 user.h

Go to the documentation of this file.

```
00001
00006 #ifndef USER_H
00007 #define USER_H
00008
00009 #include <string>
00010
00011 #include "../flights/flight_connection.h"
00012 #include "../luggage/luggage.h"
00013 #include "mongocxx/client.hpp"
00014
00015 class Admin;
00016
00021 class User {
00022  private:
00023   mongocxx::database _db_;
00024   mongocxx::collection _collection_;
00025   std::string _password_;
00026
00027  protected:
00028   mongocxx::client &_client;
00029
00030  public:
00035   explicit User(mongocxx::client &client);
00036
00040   User(std::string username, std::string email, double discount,
00041        std::string discount_type, std::string premium_card,
00042        std::string payment_method, mongocxx::client &client,
00043        std::string profession, std::string registration_date,
00044        double money_spent, double money_saved,
00045        int ticket_bought, std::vector<bsoncxx::document::value> user_flights, bool is_admin);
00046
00047   std::string username_;
00048   std::string profession_;
00049   std::string email_;
00050   std::string discount_type_;
00051   double discount_;
00052   std::string premium_card_;
00053   std::string payment_method_;
00054   std::string registration_date_;
00055   double money_spent_;
00056   double money_saved_;
00057   int ticket_bought_;
00058   std::vector<bsoncxx::document::value> user_flights_;
00059   bool is_admin_;
00060
00061   void Reset();
00062   mongocxx::collection &GetCollection();
00063   mongocxx::collection GetSpecificCollection(const std::string &collection_name);
00064   std::string GetPassword();
00065   void SetPassword(const std::string &password);
00066   void SetPremiumCard(User &user, const std::string &card);
00067   void SetBlik(const std::string &payment_method);
00068   void SetVisa(const std::string &card_number, const std::string &card_cvv);
00069   void ChangeUsername(const std::string &username);
00070   void ChangeEmail(const std::string &email);
00071   void ChangePassword(const std::string &password);
00072   void SetDiscount(double discount, const std::string &discount_type);
00073   [[nodiscard]] double GetDiscount() const;
00074   [[nodiscard]] std::string RecognizeDiscount() const;
00075   void AddTicketToUser(const std::vector<int> &seats,
00076                        const FlightConnection &flight_connection);
00077   void UpdateMoneySaved(double normal_price, double discount_price);
00078   Admin *LoginAsAdmin();
00079   [[nodiscard]] bool CheckIfAdmin() const;
00080   void SetIsAdmin(bool is_administrator) { User::is_admin_ = is_administrator; }
00081   void LuggageCheckin(int flight_number);
00082   mongocxx::cursor FindUserInDatabase();
00083
00090   template<typename T>
00091   void UpdateUserInDatabase(
00092       const std::string &value_in_database,
00093       const T &value_to_set) {
00094     bsoncxx::document::value update_builder = bsoncxx::builder::basic::make_document(
00095         bsoncxx::builder::basic::kvp("$set", bsoncxx::builder::basic::make_document(
00096             bsoncxx::builder::basic::kvp(value_in_database, value_to_set))));
00097
00098     bsoncxx::document::view update_view = update_builder.view();
00099     bsoncxx::document::value filter_builder_email_password = bsoncxx::builder::basic::make_document(
00100         bsoncxx::builder::basic::kvp("email_", email_),
00101         bsoncxx::builder::basic::kvp("password", GetPassword()));
00102
00103     bsoncxx::document::view filter_view_email_password = filter_builder_email_password.view();
```

```
00104     _collection_.update_one(filter_view_email_password, update_view);
00105   }
00106 };
00107
00108 #endif  // USER_H
```

5.55 user/user_functions/user_payments/user_payment_functions.h File Reference

This file contains the declaration of user payment handling functions.

```
#include <iostream>
#include "../../user.h"
```

Functions

- void HandlePaymentOption (User &user)

  Handles the payment option of a user.
- bool AuthenticatePayment (User &user, const std::string &payment_method, const std::string &title_message, int target_price)

  Authenticates a payment made by a user.

5.55.1 Detailed Description

This file contains the declaration of user payment handling functions.

5.55.2 Function Documentation

5.55.2.1 AuthenticatePayment()

```
bool AuthenticatePayment (
            User & user,
            const std::string & payment_method,
            const std::string & title_message,
            int target_price )
```

Authenticates a payment made by a user.

Parameters

user	The user making the payment.
payment_method	The method of payment used by the user.
title_message	The title message for the payment.
target_price	The target price of the payment.

Returns

True if the payment is authenticated, false otherwise.

5.55.2.2 HandlePaymentOption()

```
void HandlePaymentOption (
            User & user )
```

Handles the payment option of a user.

Parameters

user	The user making the payment.

5.56 user_payment_functions.h

Go to the documentation of this file.

```
00001
00006 #ifndef USER_PAYMENT_FUNCTIONS_H
00007 #define USER_PAYMENT_FUNCTIONS_H
00008
00009 #include <iostream>
00010
00011 #include "../../user.h"
00012
00017 void HandlePaymentOption(User &user);
00018
00027 bool AuthenticatePayment(User &user, const std::string &payment_method, const std::string
      &title_message, int target_price);
00028
00029 #endif  // USER_PAYMENT_FUNCTIONS_H
```

5.57 user_prints.h

```
00001
00006 #ifndef AIRPORT_USER_PRINT_FUNCTIONS_H
00007 #define AIRPORT_USER_PRINT_FUNCTIONS_H
00008
00009 #include <string>
00010
00011 #include "../../user.h"
00012 #include "ftxui/dom/elements.hpp"
00013 #include "ftxui/screen/screen.hpp"
00014
00020 std::string DisplaySettingsMenu(const User &user);
00021
00026 int DisplayDefaultPaymentScreen();
00027
00032 void DisplayProfileScreen(User &user);
00033
00034 #endif  // AIRPORT_USER_PRINT_FUNCTIONS_H
```

5.58 user_settings_handler.h

```
00001
00006 #ifndef USER_SETTINGS_FUNCTIONS_H
00007 #define USER_SETTINGS_FUNCTIONS_H
00008
00009 #include <iostream>
00010
00011 #include "../../user.h"
00012
00017 void HandleSettingsOption(User &user);
00018
00019 #endif  // USER_SETTINGS_FUNCTIONS_H
```

5.59 user_tickets_prints.h

```
00001
00006 #ifndef AIRPORT_USER_TICKETS_PRINT_FUNCTIONS_H
00007 #define AIRPORT_USER_TICKETS_PRINT_FUNCTIONS_H
00008
00009 #include <string>
00010 #include <vector>
00011
00012 #include "../../user.h"
00013
00015 const int PAGE_SIZE = 4;
00016
00021 struct FlightInfo {
00022   int flight_number;
00023   std::string flight_id;
00024   std::string departure;
00025   std::string destination;
00026   std::string departure_time;
00027   double price;
00028   std::vector<int> seats;
00029   bool checkin;
00030   bool luggage_checkin;
00031 };
00032
00039 std::optional<std::string> CreateTicketsScreen(User &user, bool is_checkin = false);
00040
00041 #endif  // AIRPORT_USER_TICKETS_PRINT_FUNCTIONS_H
```

Index