SDI_CW_Final

Generated by Doxygen 1.8.17

Chapter 1

SDI-Project

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

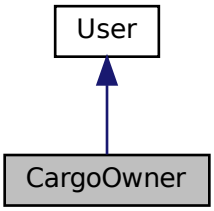
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

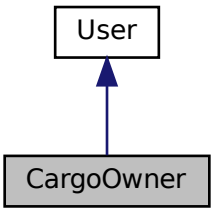
Class Documentation

4.1 CargoOwner Class Reference

Inheritance diagram for CargoOwner:



Collaboration diagram for CargoOwner:



Public Member Functions

- CargoOwner (std::string uNameInp, std::string pWordInp, std::string emailInp, std::string cNumInp)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 CargoOwner()

```
CargoOwner::CargoOwner (
            std::string uNameInp,
            std::string pWordInp,
            std::string emailInp,
            std::string cNumInp )
```

Used for setting cargo owner details

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

- SDI_CW_4/userclass.h
- SDI_CW_4/userclass.cpp

4.2 Controller Class Reference

Public Member Functions

- bool signIn (std::string username, std::string password)
- bool signOut ()
- std::vector< std::string > **getTransportcomp** ()
- bool validateAccount (std::string username, std::string password)
- bool validateUsername (std::string usernameInput)
- bool validatePassword (std::string password)
- void **passUpdatedUserDetails** (std::string updateString, int callLocation)
- bool validateEmail (std::string email)
- int validateOrderDimension (std::string orderDimensionsandWeight[ ], bool frozen, bool fragile, float transportcompRate)
- bool validateCnum (std::string contactNum)
- bool **validateOrder** (std::string sourceAddress[ ], std::string destinationAddress[ ], float order↩ DimentionsandWeight[ ], bool frozen, bool fragile)
- bool validateAddress (std::string numberandstreet, std::string townorcity, std::string county, std::string postcode)
- bool **ValidateDriverdetails** (std::string CPCnum, std::string drivLicID, std::string NInum, std::string regNum, bool lorryIndex)
- int validatelorryIndex (std::vector< bool >lorryIndexVec)
- bool validatePostCode (std::string postCode)
- void selectDriver (std::vector< std::string > SelectedTCorder)
- std::vector< std::vector< std::string > > **getTCOrderList** ()
- std::vector< std::vector< std::string > > getDriverOrders (std::string driverName, int callLocation)
- std::vector< std::vector< std::string > > **getOrderHistory** ()
- std::vector< std::vector< std::string > > **getcargoOwnerOrders** (std::string cargoOwnerName)
- std::string getUserType ()
- bool **validateCPCnum** (std::string CPCnum)
- bool **validateDrivLicID** (std::string drivLicID)
- bool **validateNInum** (std::string NInum)

- bool **validateRegNum** (std::string regNum)
- void passOrderAddresses (std::string sourceLocation[ ], std::string destinationLocation[ ])
- void passOrderDetails (std::string transportCompSelected)
- void acceptorRejectOrder (bool acceptOrder, std::string driverName, std::string orderID)
- void updateOrderStatus (int callLocation, std::string orderID)
- void setDriverChosen (std::string driverName, std::string orderID)
- std::string **passUsername** ()
- bool createUser (std::string userType, std::string username, std::string password, std::string email, std::string contactNumber, std::vector< std::string > driverStrings, std::vector< std::string > address, int lorryIndex)
- bool **CreateLorry** (std::string lorryType, std::string lorryRegNum)
- bool **deleteThisUser** ()
- bool **createOrder** ()
- bool **writeUserToDB** (std::shared_ptr< User > &user)
- bool **writeOrderToDB** (std::shared_ptr< User > &user, const Order &order)
- bool **writeMsgToDB** (const std::string &username, const std::string &msg)
- bool **writeLorryToDB** (const User &user, const Lorry &lorry)
- bool **deleteUserFromDB** (const std::string username)
- User **readUserFromDB** (const std::string &username)
- std::list< Order > **readOrdersFromDB** (const std::string &username)
- std::list< std::string > **readMessagesFromDB** (const std::string &username)
- Lorry **readLorryFromDB** (const std::string &username)

4.2.1 Member Function Documentation

4.2.1.1 acceptorRejectOrder()

```
void Controller::acceptorRejectOrder (
            bool acceptOrder,
            std::string driverName,
            std::string orderID )
```

acceptorRejectOrder passes the drivers choice after receiving and order and stores it in the database, it uses accept order to validate which option needs to be called.

4.2.1.2 createUser()

```
bool Controller::createUser (
            std::string userType,
            std::string username,
            std::string password,
            std::string email,
            std::string contactNumber,
            std::vector< std::string > driverStrings,
            std::vector< std::string > address,
            int lorryIndex )
```

createUser is used to create the User object and pass the user details to be saved on the database.

4.2.1.3 getDriverOrders()

```
std::vector< std::vector< std::string > > Controller::getDriverOrders (
            std::string driverName,
            int callLocation )
```

All the Get order functions return a vector of string vectors with order information for a given username.

4.2.1.4 getUserType()

```
std::string Controller::getUserType ( )
```

This returns the user type of the user object to the main window where it is used.

4.2.1.5 passOrderAddresses()

```
void Controller::passOrderAddresses (
            std::string sourceLocation[],
            std::string destinationLocation[] )
```

Setting order source and destination location.

4.2.1.6 passOrderDetails()

```
void Controller::passOrderDetails (
            std::string transportCompSelected )
```

Passes order details to the database to be stored.

4.2.1.7 selectDriver()

```
void Controller::selectDriver (
            std::vector< std::string > SelectedTCorder )
```

selectDriver receives an order vector from the user and converts the required details into floats with lexical cast, then it gets all drivers in the database and compares them to current values for best weight and volume match. Once the best match is found, it is passed to setDriverChosen where the selected driver is set for the orderid given.

4.2.1.8 setDriverChosen()

```
void Controller::setDriverChosen (
            std::string driverName,
            std::string OrderID )
```

Passing driver chosen to the database.

4.2.1.9 signIn()

```
bool Controller::signIn (
            std::string username,
            std::string password )
```

signIn calls the database readUser function in order to validate the user login info. If the user is found in the database the user pointer won't be null and true is returned.

4.2.1.10 signOut()

```
bool Controller::signOut ( )
```

signOut signs out the user as so that another can login or create an account. It does this by resetting the user pointer.

4.2.1.11 updateOrderStatus()

```
void Controller::updateOrderStatus (
            int callLocation,
            std::string orderID )
```

updateOrderStatus takes a call location and order id to allow the same database function to be called with different results. The function called updates the state of the order and can only be called by the driver if they have accepted the order.

4.2.1.12 validateAccount()

```
bool Controller::validateAccount (
            std::string username,
            std::string password )
```

ValidateAccount calls the database readUser function to see if the user is in the database, if they are found the user will be created and tempUser won't be null.

4.2.1.13 validateAddress()

```
bool Controller::validateAddress (
            std::string numberandstreet,
            std::string townorcity,
            std::string county,
            std::string postcode )
```

validateAddress takes the users address information validates it with regex and returns true of false.

4.2.1.14 validateCnum()

```
bool Controller::validateCnum (
            std::string contactNum )
```

validateCnum checks to see if the given contact number is the correct length and returns true if it is.

4.2.1.15 validateEmail()

```
bool Controller::validateEmail (
            std::string email )
```

ValidateEmail checks a given email against a regex pattern and returns the result.

4.2.1.16 validatelorryIndex()

```
int Controller::validatelorryIndex (
            std::vector< bool > lorryIndexVec )
```

validatelorryIndex takes all of the checkbox states from the UI, ensures only one is chosen and returns which has been chosen.

4.2.1.17 validateOrderDimension()

```
int Controller::validateOrderDimension (
            std::string orderDimensionsandWeight[],
            bool frozen,
            bool fragile,
            float transportcompRate )
```

This takes all the order information from the MainWindow and validates if its in the correct format, if it is the calculate shipping cost function is called in the order class and an order is instantiated. < regex needs to compare strings so the values are converted to floats to be used after.

Parameters

transportcompRate	returns true

4.2.1.18 validatePassword()

```
bool Controller::validatePassword (
            std::string password )
```

Validate password ensures the user enters a password within the bounds set.

4.2.1.19 validatePostCode()

```
bool Controller::validatePostCode (
            std::string postCode )
```

validatePostCode is used to validate the postcode separately from the address to be used in the order.

4.2.1.20 validateUsername()

```
bool Controller::validateUsername (
            std::string username )
```

ValidateUsername uses readUser like validateAccount however returns true if only the username is found to match.

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

- SDI_CW_4/controllerclass.h
- SDI_CW_4/controllerclass.cpp

4.3 Database Class Reference

Public Member Functions

- **Database** (std::string connectString)
- void startDBtest ()
- void **encryptAllPWS** ()
- std::vector< std::string > getTransportcompanys ()
- void writeUser (const std::shared_ptr< User > &user)
- void writeOrder (const std::shared_ptr< User > &user, const Order &order, std::string transportCompSelected)
- void addDriverSelectiontoOrder (std::string driverName, std::string OrderID)
- void acceptOrder (std::string driverName, std::string orderID)
- void rejectOrder (std::string orderID)
- void updateOrderStatus (std::string orderStatusChosen, std::string orderID)
- std::vector< std::vector< std::string > > readOrderHistory ()
- void **updateUserDetails** (std::string username, std::string updateString, std::string updateID)
- std::string **encryptorDecryptPassword** (std::string Password)
- bool **writeMsg** (const std::string &username, const std::string &msg)
- bool **deleteUser** (const std::string username)
- std::shared_ptr< User > readUser (const std::string &username, const std::string &password)
- std::vector< std::vector< std::string > > readCOOrders (const std::string &username)
- std::vector< std::vector< std::string > > readTCOrders (const std::string &username)
- std::vector< std::vector< std::string > > readDrivers ()
- std::vector< std::vector< std::string > > getDriverOrders (std::string driverName, int callLocation)

4.3.1 Member Function Documentation

4.3.1.1 acceptOrder()

```
void Database::acceptOrder (
            std::string driverName,
            std::string orderID )
```

acceptOrder is used to set order status to assigned for the order after it is accepted, while adding the drivers name in the correct column. < getting all drivers and their lorry index.

4.3.1.2 addDriverSelectiontoOrder()

```
void Database::addDriverSelectiontoOrder (
            std::string driverName,
            std::string OrderID )
```

addDriverSelectiontoOrder updates the given order with the driver chosen for it along with using the orderID to identify the location to change the results. < adding the selected driver to the driverChosen variable.

4.3.1.3 getDriverOrders()

```
std::vector< std::vector< std::string > > Database::getDriverOrders (
            std::string driverName,
            int callLocation )
```

getDriverOrders returns all the driver orders and filters which are returned with the call location, with call location 0 being for assigned orders and call location 2 being for all order states except delivered or created. < looping through results and adding them to a vector,then adding the result to another vector.

4.3.1.4 getTransportcompanys()

```
std::vector< std::string > Database::getTransportcompanys ( )
```

Gets all transport company prices and names, returning a vector with all entries. <looping throught the username results

4.3.1.5 readCOOrders()

```
std::vector< std::vector< std::string > > Database::readCOOrders (
            const std::string & username )
```

readCOOrders reads all cargoOwners orders and stores all which have not been marked as delivered. It returns a vector of vectors containing all of these string entries.

4.3.1.6 readDrivers()

```
std::vector< std::vector< std::string > > Database::readDrivers ( )
```

readDrivers selects the drivers username and lorryindex from all drivers, finds the drivers lorry details with lorryindex and returns a vector of vectors with their details. < includes the drivers username, their lorry weight, if they can carry frozen goods and their cargo volume.

< getting all drivers and their lorry index.

< looping through results and adding them to a vector,then adding the result to another vector.

4.3.1.7 readOrderHistory()

```
std::vector< std::vector< std::string > > Database::readOrderHistory ( )
```

readOrderHistory retreives all the orders which have been delivered and returns a vector of string vectors, it iterates though each row and column to add the responses to a vector, adding the created vector to orderHistoryList once all the columns have been added.

4.3.1.8 readTCOrders()

```
std::vector< std::vector< std::string > > Database::readTCOrders (
            const std::string & username )
```

readTCOrders selects all of the orders where the given transportation company have been assigned, looping through the results and returning a vector of vectors.

4.3.1.9 readUser()

```
std::shared_ptr< User > Database::readUser (
            const std::string & username,
            const std::string & password )
```

readUser uses the username and password given at login to search for users in the database, if found the user will be initialized to the corresponding user type. If the usertype is a driver the driver details will also be found from 'lorrys' where they are stored in the database. The passwords on the database are encrypted so a QT base64 conversion is used to compare the un encrypted password to the user input.

4.3.1.10 rejectOrder()

```
void Database::rejectOrder (
            std::string orderID )
```

rejectOrder is used to remove the driverchosen as so that the order can be reassigned a new driver.

4.3.1.11 startDBtest()

```
void Database::startDBtest ( )
```

Tests the database connection.

4.3.1.12 updateOrderStatus()

```
void Database::updateOrderStatus (
            std::string orderStatusChosen,
            std::string orderID )
```

updateOrderStatus is used to update the order with the status chosen by the driver such as delivered or loading.

4.3.1.13 writeOrder()

```
void Database::writeOrder (
            const std::shared_ptr< User > & user,
            const Order & order,
            std::string transportCompSelected )
```

writeOrder Writes an order to the database after it has been created and validated, calling a stored procedure on the database with the required values.

4.3.1.14 writeUser()

```
void Database::writeUser (
            const std::shared_ptr< User > & user )
```

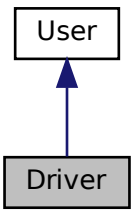
writeUser Writes all the user information to the database after a user has been created, this is done by passing all the validated information to a stored procedure on the database while calling the 'adddriverinfo' procedure if a drivers details also need to be added. The passwords are encrypted with the postgresql encode function in the adduser statement as so they are stored securely. !Setting the local user type to be the same as in the user object.

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

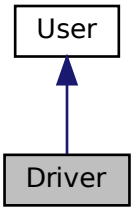
- SDI_CW_4/database.h
- SDI_CW_4/database.cpp

4.4 Driver Class Reference

Inheritance diagram for Driver:



Collaboration diagram for Driver:



Public Member Functions

- Driver (std::string uNameInp, std::string pWordInp, std::string emailInp, std::string cNumInp, std::string N↩ InumberInp, std::string drivLicIDInp, std::string CPCnumInp, std::string lorryRegNumInp, int lorryIndex)
- void **manageOrder** (int index)
- const std::string & **getCPCnumber** () const
- const std::string & **getNInumber** () const
- Lorry **getlorry** () const
- const std::string & **getDrivLicID** () const

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Driver()

```
Driver::Driver (
            std::string uNameInp,
            std::string pWordInp,
            std::string emailInp,
            std::string cNumInp,
            std::string NInumberInp,
            std::string drivLicIDInp,
            std::string CPCnumInp,
            std::string lorryRegNumInp,
            int lorryIndex )
```

Used for setting Driver details.

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

- SDI_CW_4/userclass.h
- SDI_CW_4/userclass.cpp

4.5 Lorry Class Reference

Public Member Functions

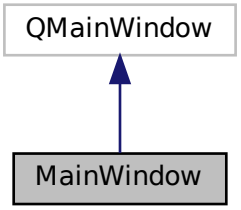
- **Lorry** (int typeIndex, std::string regNumInp)
- std::string **getregNum** ()
- int **gettypeIndex** ()

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

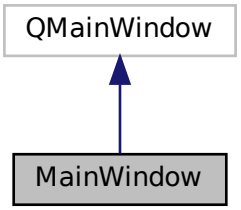
- SDI_CW_4/orderclass.h
- SDI_CW_4/orderclass.cpp

4.6 MainWindow Class Reference

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Member Functions

- **MainWindow** (QWidget *parent=nullptr)
- void **setUsertype** (std::string usertype)

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

- SDI_CW_4/mainwindow.h
- SDI_CW_4/mainwindow.cpp

4.7 Order Class Reference

Public Member Functions

- float calcShippingCost (float orderDimensionsandWeight[ ], bool frozen, bool fragile, float transportcompRate)
- void setSourceandDestination (std::string SourceAddress[ ], std::string DestinationAddress[ ])
- float **getWidth** () const
- float **getLength** () const
- float **getHeight** () const
- float **getWeight** () const
- float **getCost** () const
- bool **getFragile** () const
- bool **getFrozen** () const
- const std::string & **getSourceAddressLine1** () const
- const std::string & **getSourceAddressLine2** () const
- const std::string & **getSourceAddressLine3** () const
- const std::string & **getSourceAddressLine4** () const
- const std::string & **getDestinationAddressLine1** () const
- const std::string & **getDestinationAddressLine2** () const
- const std::string & **getDestinationAddressLine3** () const
- const std::string & **getDestinationAddressLine4** () const

4.7.1 Member Function Documentation

4.7.1.1 calcShippingCost()

```
float Order::calcShippingCost (
            float orderDimensionsandWeight[],
            bool frozen,
            bool fragile,
            float transportCompRate )
```

calcShippingCost gets order dimensions, fragile/frozen state and transportCompRate to calculate the order cost. The volumetric weight of the order is calculated and whichever is heavier the weight or volumetric weight is used to calculate the order cost. Additional variables such as fragile and frozen add cost to the order and the transport company fee is added at the end. < volumetric weight

< 300 miles set as a place holder distance

< width

< length

< height

< weight

4.7.1.2 setSourceandDestination()

```
void Order::setSourceandDestination (
            std::string SourceAddress[],
            std::string DestinationAddress[] )
```

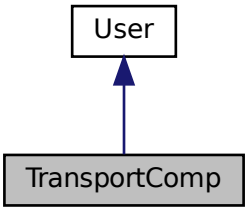
sets order source and destination.

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

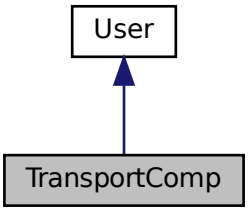
- SDI_CW_4/orderclass.h
- SDI_CW_4/orderclass.cpp

4.8 TransportComp Class Reference

Inheritance diagram for TransportComp:



Collaboration diagram for TransportComp:



Public Member Functions

- TransportComp (std::string uNameInp, std::string pWordInp, std::string emailInp, std::string cNumInp)

4.8.1 Constructor & Destructor Documentation

4.8.1.1 TransportComp()

```
TransportComp::TransportComp (
            std::string uNameInp,
            std::string pWordInp,
            std::string emailInp,
            std::string cNumInp )
```

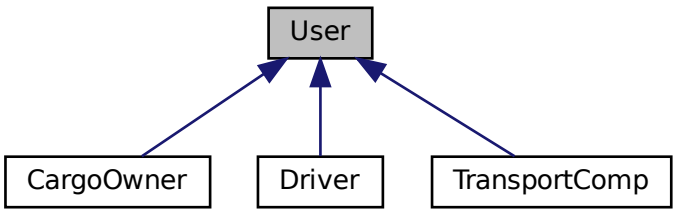
Used for setting TransportComp details.

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

- SDI_CW_4/userclass.h
- SDI_CW_4/userclass.cpp

4.9 User Class Reference

Inheritance diagram for User:



Public Member Functions

- virtual void setAddress (std::string numberAndStreet, std::string townOrCity, std::string county, std::string postCode)
- virtual const std::string & **getUsername** () const
- const std::string & **getPassword** () const
- const std::string & **getEmail** () const
- const std::string & **getContactNumber** () const
- const std::map< std::string, std::string > & **getAddress** () const
- const std::string **getAddressFormat** () const
- const std::string & **getAddressLine1** () const
- const std::string & **getAddressLine2** () const
- const std::string & **getAddressLine3** () const
- const std::string & **getAddressLine4** () const
- void **setUsername** (const std::string uNameInp)
- void **setPassword** (const std::string pWordInp)
- void **setEmail** (const std::string emailInp)
- void **setContactNumber** (const std::string cNumInp)

4.9.1 Member Function Documentation

4.9.1.1 setAddress()

```
void User::setAddress (
            std::string numberAndStreet,
            std::string townOrCity,
            std::string county,
            std::string postCode )  [virtual]
```

Used for setting the user address when they sign in or create an account.

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

- SDI_CW_4/userclass.h
- SDI_CW_4/userclass.cpp

Index