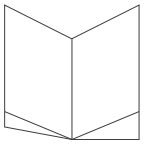
ICT Engineering
Project Report SEP 1X A16 Group 1
VIA University College



# Project Report

---

# Autobus

## Bus Transportation Management System version 1.0

Group no: 1

Mogens Bjerregaard, student number 254111

Nicolai Onov, student number 253739

Eugeniu Maloman, student number 253979

Adamantios Kounis, student number 253711

Course: SEP 1X A16

Class: IT-1X-A16

Supervisors: Ole Ildsgaard Hougaard and Birgitte von Fyren Balsløv

Date: 15 december 2016

# Abstract

*Given an imaginary problem case - describing a bus company's demanding operational tasks of managing their reservations, customers, passengers, chauffeurs, and associated resources – the defined assignment for this project, is to develop a single user software application, which helps the company manage their business processes.*

*The project report covers the progressive four-step process of developing the software system - from analysis, and design - to implementation, and testing.*

*The analysis section covers how the use case diagrams and descriptions are carried out, based on an interview with the bus company's MD. The design process gives some insight to considerations done to the GUI structure, in order to make it intuitive and easy to use, and which colours to use.*

*The implementation section puts focus on a couple of details of how the code was implemented, such as the structure of the class containing the main method and GUI components, as well as how data persistence is handled in the system, and the use of lambda expressions.*

*The testing phase lead to a result of 100% of the requirements was implemented, of which all are functioning as intended.*

*The project report concludes that the analysis and design phases, are perhaps the more important ones in the context of assigning adequate time, as it otherwise can result in unnecessary confusion and extra workload in the implementation phase.*

*The report also addresses the problems of a design with the main method, and the GUI components and event handlers in one single class - often resulting in a significant number of lines of code - and suggest a better design, where these parts are in separate classes.*

# Table of Contents

# Introduction

This project is based on a fictional problem case. The scenario is a bus reservation company (VIA Bus) which offers bus & chauffeur rental products and predefined tours to sites and events in Europe. The challenge is to help the bus company handle their operational and administrative processes, by developing a single user system for managing their portfolio of products and resources in an easier way.

The development method is a four-step process, which takes it beginning with an analysis of the customer's (the bus company) requirements for the system. Based on an interview with the customer, a list of requirements is formed and verified with the customer.

Next step is using the list of requirements to create use case diagrams and descriptions of each of the functions the system should be designed to perform. For every use case, an activity diagram is drawn, and subsequently the class diagram - explaining the relationship between the individual Java classes – and the sequence diagrams are constructed.

The development is now ready to commence the third step - the implementation of the Java code, and finally the last step of the development process - the testing of the system - can be executed.

# Analysis

## The requirements

Basis of the analysis process is the interview (appendix A) with the manager of VIA Bus company. The information gathered through the interview, is transferred to a list of requirements for the system to be developed.

The functional requirements are:

1. The user must be able to make reservations for predefined tours and reservations for a bus and chauffeur (bus and chauffeur service), and get a total price and a price per passenger. The system must coordinate the availability of resources (seats, buses, and chauffeurs) for a reservation.
2. The user must be able to search, add, update, and remove tours. Each tour must register destination, durance, extra services (all inclusive, breakfast, lunch, entrance tickets), and any pick-up stops.
3. The user must be able to search, add, update, and remove customized bus & chauffeur trips. Each bus reservation must register durance, and extra services.
4. The user must be able to search, add, update, and remove chauffeurs and register which tour type he can drive (one day only), if he is a full-time employee or external, his name, address, 5-digit employee-id, phone number, email.
5. The user must be able to search, add, update, and remove buses and register capacity, bus type: luxury bus, party bus, or standard bus, price per hour.
6. The user must be able to search, add, update and remove customers. Customers purchase history must be registered (discount can be applied). User must be able to register name, organisation name and type (company, school, private), address, phone number, birthday, email.
7. The user must be able to search, add, update, and remove passenger and register name, address, phone number, birthday, email.

The non-functional requirements are:

1. The system must be a single user system to run on a front desk computer only.
2. The system should be built in Java and have a standard looking program look (some fields to enter the data and some buttons, menus, or tabs), that can be used with a keyboard and mouse.
3. The system must be able to save all information.

## The use cases

The use case diagram (appendix B) is displayed below in a simplified edition, in the sense that the use cases concerning customers, passengers, chauffeurs, tours, and buses, is illustrated by a representative model which is similar for each of them – *delete*, *update*, *search*, and *new*.

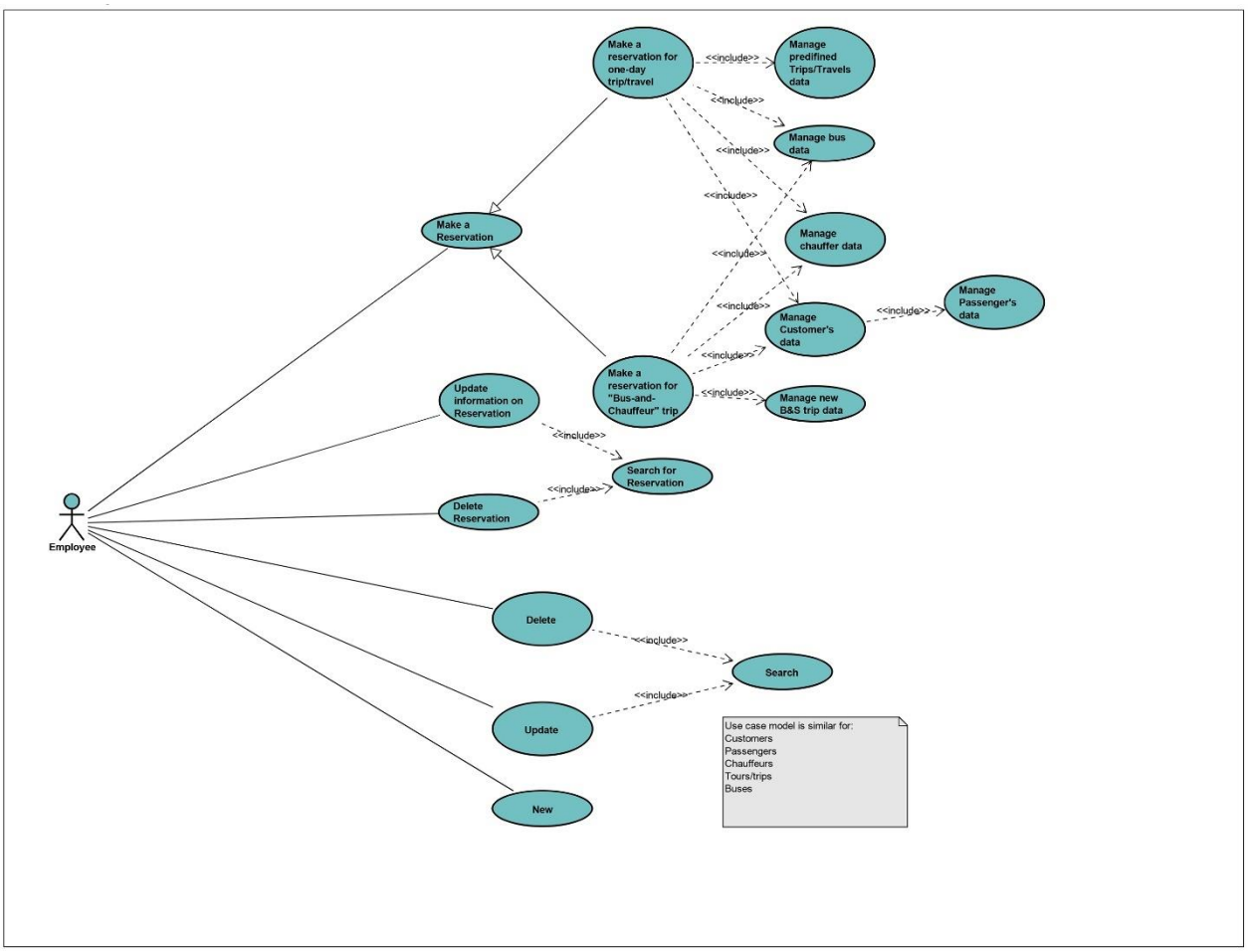



*Illustration 1:The image shows a shortened version of the use case diagram - the use cases concerning customers, passengers, chauffeurs, tours, and buses, is similar for each one of them. The full version of the use case diagram can be found in appendix B.*

The perhaps most complex use case is the reservation process. In the next example, the use case description of the bus & chauffeur reservation is featured. The use case descriptions can be found in appendix C.

| Item | Value |
|---|---|
| Use case | Make a reservation for bus & chauffeur |
| Postcondition | The reservation is completed and all the data regarding it is saved: type of trip/travel, customers data, passengers data, appointed bus and appointed chauffeur. |
| Base sequence | 1. Enter name of the trip, duration of the trip/travel, distance driven, pick-up/drop-off stops, departure time, arrival time to destination/destinations, arrival back to Terminal in Horsens, extra services(food, tickets, etc) (Use case: Manage new B&C trip data).<br>2. Verify the new B&C trip's data.<br>3. IF the new B&C trip's data is not correct THEN go to step 1, ELSE create a record of a new B&C trip in system. (Use case: Manage new B&C trip data).<br>4. Enter number of passengers, and date of the trip's start.<br>5. System returns a list of all available buses, suitable for this trip.(User case: Manage bus data)<br>6. Select from the list the bus to reserve.<br>7. System returns detailed information about the bus: type, model, level of comfort, number of seats.(Use case: Manage bus data).<br>8. IF bus cannot be accepted by Employee THEN go to step 4.<br>9. System returns the list of all available chauffeurs, appropriate for this trip.(Use case: Manage chauffeur data).<br>10. Select from the list the chauffeur to appoint.<br>11. The system returns detailed information on given chauffeur: name, address, 5-digit-employee-id, availability, phone number, email, wishes for trips.(Use case: Manage chauffeur data)<br>12. IF chauffeur cannot be accepted by Employee THEN go to step 8.<br>13. Verify dates, number of passengers, bus to reserve and chauffeur to appoint.<br>14. If dates/number of passengers/bus to reserve/chauffeur to appoint are not correct THEN go to step 4.<br>15. Enter the customer's name and phone number.<br>16. System searches for customer by name and phone number.(Use case: Manage customer data) |

| | |
|---|---|
| | 17. IF the customer is not found in the system THEN enter customers address and email address if there is one and create customer record.(Use case: Manage customer data).<br>18. IF there are more customers to enter, THEN go to step 15.<br>19. Enter passengers name and phone number.<br>20. System searches for passenger by name and phone number.(Use case: Manage passenger data)<br>21. IF the passenger is not found in the system THEN enter passengers name, address and email address if there is one and create a passenger record.(Use case: Manage passenger data)<br>22. IF there are more passengers to enter, THEN go to step 19.<br>23. Show information for the reservation for a final verification.<br>24. Store the reservation data in the system. |
| Exception sequence | No available buses, suitable for entered number of passangers is available for the selected trip in the given date interval.<br>1-7 as a base sequence<br>The returned list is empty and use case ends.<br><br>Employee will not reserve one of the available buses.<br>1-8 as a base sequence<br>No more buses to select and use case ends.<br><br>No available chauffeurs, suitable for the selected trip and bus in the given date interval.<br>1-9 as a base sequence<br>The returned list is empty and use case ends.<br><br>Employee will not appoint one of the available chauffeurs.<br>1-13 as a base sequence<br>No more chauffeurs to select, and use case ends. |
| Sub use case | Manage new b&c trip data<br>Manage customer's data<br>Manage bus data<br>Manage chauffeur data |
| Note | The reservation process can be cancelled at any time |

## The activity diagram

The activity diagram below, is showing the bus and chauffeur reservation process. The activity diagrams for the other processes, are available in appendix D, and the sequence diagrams in appendix E.

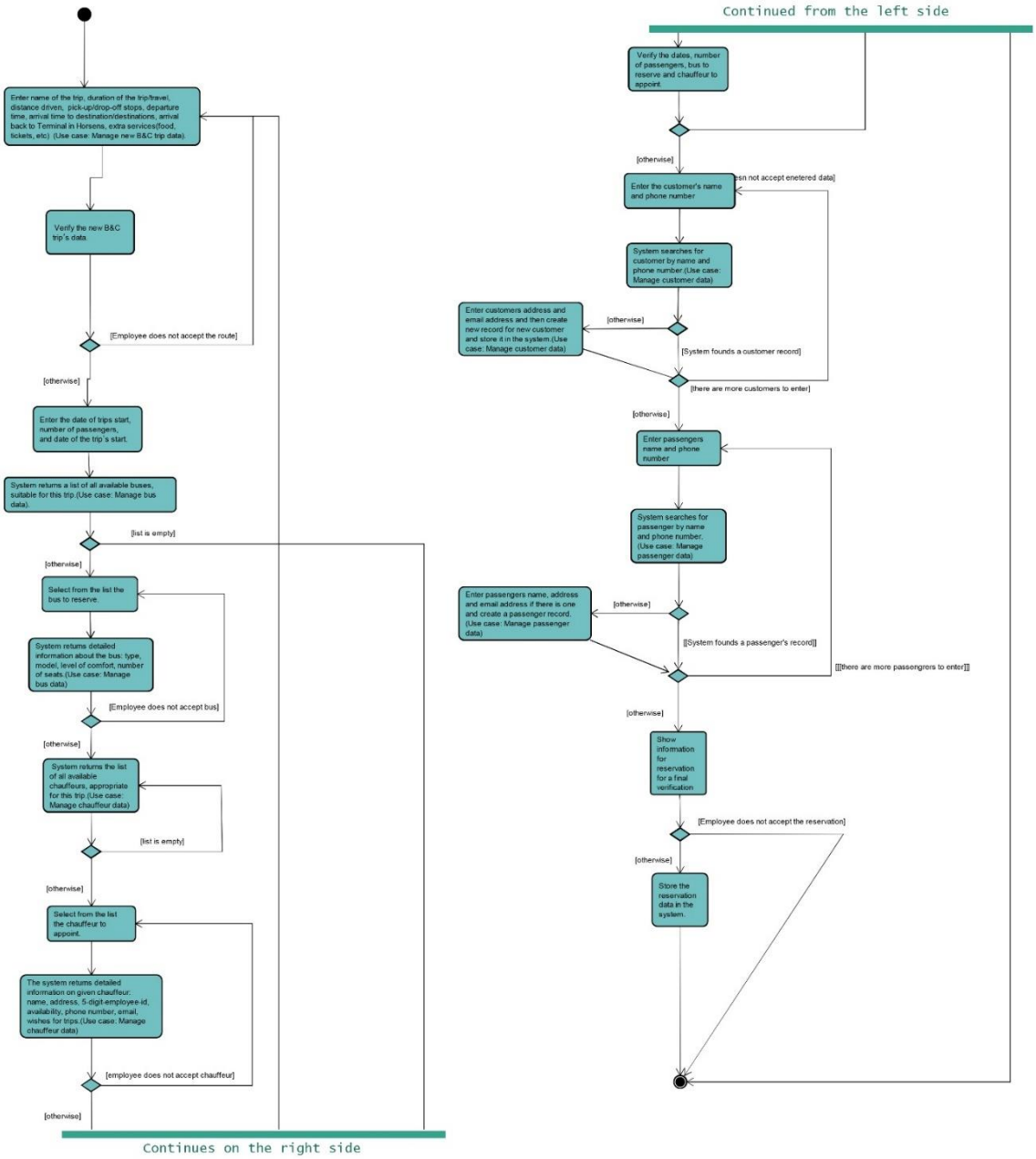



*Illustration 2: The activity diagram of the bus & chauffeur reservation process. The activity diagram is divided in two, the left part is continued in the right side of the illustration.*

# Design

## The GUI design

The importance of a good, effective graphical user interface design, is emphasized by several company studies (Galitz, 2007) that shows how well designed user interfaces in business software applications, can result in considerable productivity improvements as well as significant savings in operational costs. Understanding what comprises a good GUI design, and which elements and characteristics make it effective, requires knowledge of numerous aspects, such as how human beings see, understand, and think. Different guidelines to good GUI design, based on screen design research, practical experience, and knowledge, can be used. In this project considerations, such as colour usage and how the placement of menus, buttons and input fields are structured, has been taken into account.

### Colour in GUI design

One of the purposes of colour in a graphical user interface, is to provide a user experience with higher emotional impact (Rochelau, 2015). Proper use of colour in GUI design, can make the software application more interesting, but then again in contrast it can be distracting and cause visual fatigue when used improperly (Galitz, 2007).

Using a mix of multiple colours successfully in a single design can be complicated, thus (Tips for UI Design Colors and Color Matching Techniques, 2016) propose using 1-2 colours or a couple of different shades of a single colour for preserving visual consistency. Bright or dark colours in GUI designs are preferred over minimal white/black designs according to user study tests (Virgillito, 2016). The contrast between text colour and background colour is similarly an important influence to bear in mind – for instance a (too) light text colour on a bright background colour, can make it difficult for the user to read, while colours with low value contrast can make it challenging for colour blinds to distinguish. For the overall colour environment in the GUI design for the Autobus application, a dark teal green shade colour theme is being applied in combination with a contrasting white colour for text and other elements. The teal green colour is linked to emotional states of stability, freshness and reliability (Barker, 2016), and seems to be a suitable choice for the Autobus system's GUI design.


*Illustration 3: Teal green colour shades used in the GUI design*

### The structure of the GUI

Understanding the business workflow is essential when designing the structure of the GUI, to ensure it achieves the business objectives it is being created for. The design of the GUI must match the customer's expectations of the specific tasks required to be performed by the system. The use case descriptions in the analysis process chapter is helping determining the business functions, describing the activity tasks, and developing a conceptual model of the complete system.

Some of the best practices and design guidelines for graphical user interfaces (Lal, 2013) includes:

- Make sure the user can anticipate a user interface element behaviour from its visual properties
- Convey warnings, errors etc. in understandable language
- Use consistent theme for the windows and user interface elements and their behaviours
- Make it intuitive by using predictable and familiar actions and menus

For the purpose of helping defining the structure of the business functions, the tasks are divided in two: The *product* and the *resources*. The *product* is the product which the company offers and sells, and the *resources* are the resources necessary for selling the product.

The functions for the *product* contains the tasks:

1. **Create new reservations,** with sub functions such as creating *new Customer* and *new Passengers*.
2. **Access to the reservations archive**, with sub functions such as *View*, *Search*, *Update*, and *Delete* objects in the archive.

| | | Create new | Access to archive |
|---|---|---|---|
| *The product* | Tour reservations | New reservation, new customer, new passengers | View, search, update, delete |
| | Bus reservations | New reservation, new customer, new passengers | View, search, update, delete |

The functions for the *resources* include the tasks:

1. **Access to the archives of:** Buses, Chauffeurs, Customers, Passenger, and predefined Tours. These tasks embrace sub functions, such as creating a *New* object, *Search*, *Update*, and *Delete* if applicable.
2. **Updating the price list for extra services**

| | | Access to archive |
|---|---|---|
| *The resources* | Buses | New, Search, Update, Delete |
| | Chauffeurs | New, Search, Update, Delete |
| | Customers | Search, Update, Delete |
| | Passengers | Search, Update, Delete |
| | Tours | New, Search, Update, Delete |
| | Prices | Update |

GUI systems are profoundly menu oriented in the sense that they very often make use of a menu bar to navigate through the page structure and the functions available in the system. To create an intuitive menu navigation bar, the structure of the menu elements should be kept simple and have links to tasks in familiar and expected positions (Galitz, 2007). Many GUI systems are structured with a menu bar starting with *File*, which usually contains a drop down menu for tasks concerning files (*new*, *open*, *save*, *exit*), followed by *Edit*, which often has a drop down menu for tasks related to editing objects or parameters. The list of menu items in the top level can be extended if it serves a purpose for the logical understanding of the business structure.

Transferring the models for *the product* and *the resources* to an intuitive menu bar navigation structure, could look like this:

| File | Edit | Reservations |
|---|---|---|
| New Tour reservation | Buses | Tour reservations |
| New Bus reservation | Chauffeurs | Bus reservations |
| Exit | Customers | |
| | Passengers | |
| | Tours | |
| | Prices | |

The *New Bus reservation* task is shown next as an **example** of how this activity process was structured in the GUI design for the Autobus system. The purpose of the *New Bus reservation* function is to allow the user to create a new Bus reservation, using the following steps to build up the new reservation in an

intuitive right to left order, all surrounded with a titled border signposting which information should be entered:

1. The first step is to define the customer for the reservation, either by using *Search* (by phone number) for an existing Customer in the customers archive, or *Create* a new Customer by entering the required information. If the customer type is *Private*, the customer name field is automatically copied to the contact name field for convenience. Similarly, if the customer contact person is also a passenger (the *is passenger* checkbox is ticked), the required information from the customer fields are automatically copied to the passengers list.
2. Next step is to build the passenger list for the reservation by either adding existing passengers from the passengers archive by searching their phone number), or creating new passengers and sub sequentially add them to the passengers list for the reservation being created. When the passengers list is complete, the *Next* button is pressed to continue the reservation process. At this point the system has enough information to filter the buses that offers adequate capacity.
3. The third step is selecting the start and end date for the reservation. When the date interval has been entered and validated, the system has gathered sufficient information to filter the buses and chauffeurs available for the date interval selected.
4. The fourth step is selecting extra services for the reservation, such as accommodation services.
5. The fifth step is selecting a bus from a list, which specifies the price and type of bus (*standard*, *party*, or *luxury*).
6. And finally, the chauffeur for the reservation is chosen from a list. The system has now gathered all information necessary for displaying a summary calculating the total price and price per passenger. The user can either accept or cancel the reservation. If the reservation is cancelled, all fields are returned to default and the screen returns to the *New Bus reservation* start page. If the new reservation is accepted, the new reservation is saved to the reservations archive, and a confirmation message is displayed.

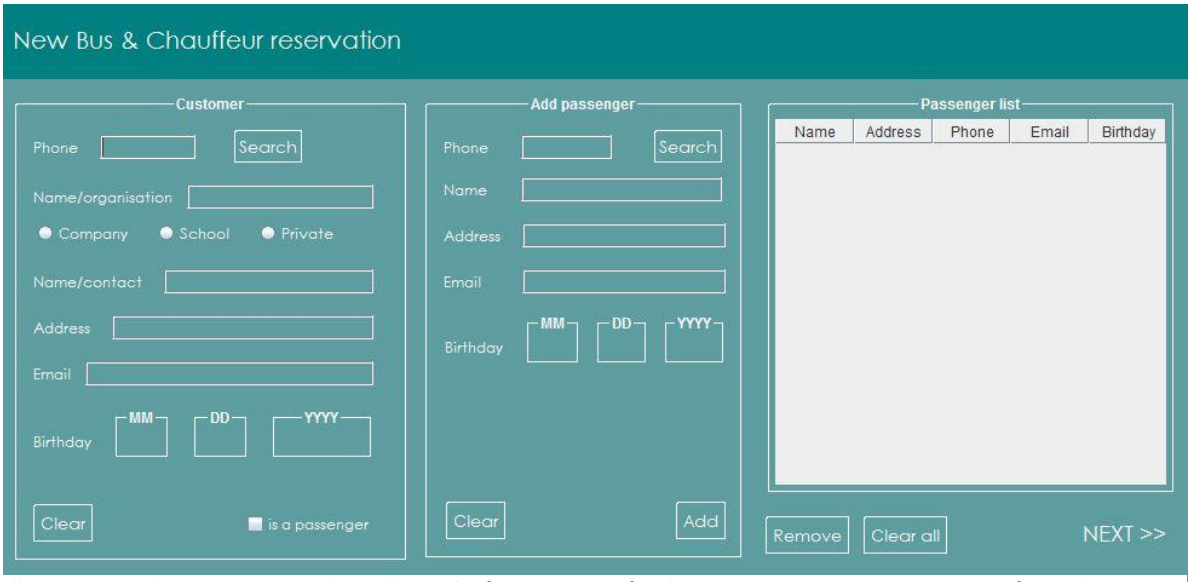



*Illustration 4: The screen image above shows the first two steps for the New Bus reservation process: Defining the customer, and building the passenger list from new or existing passengers.*

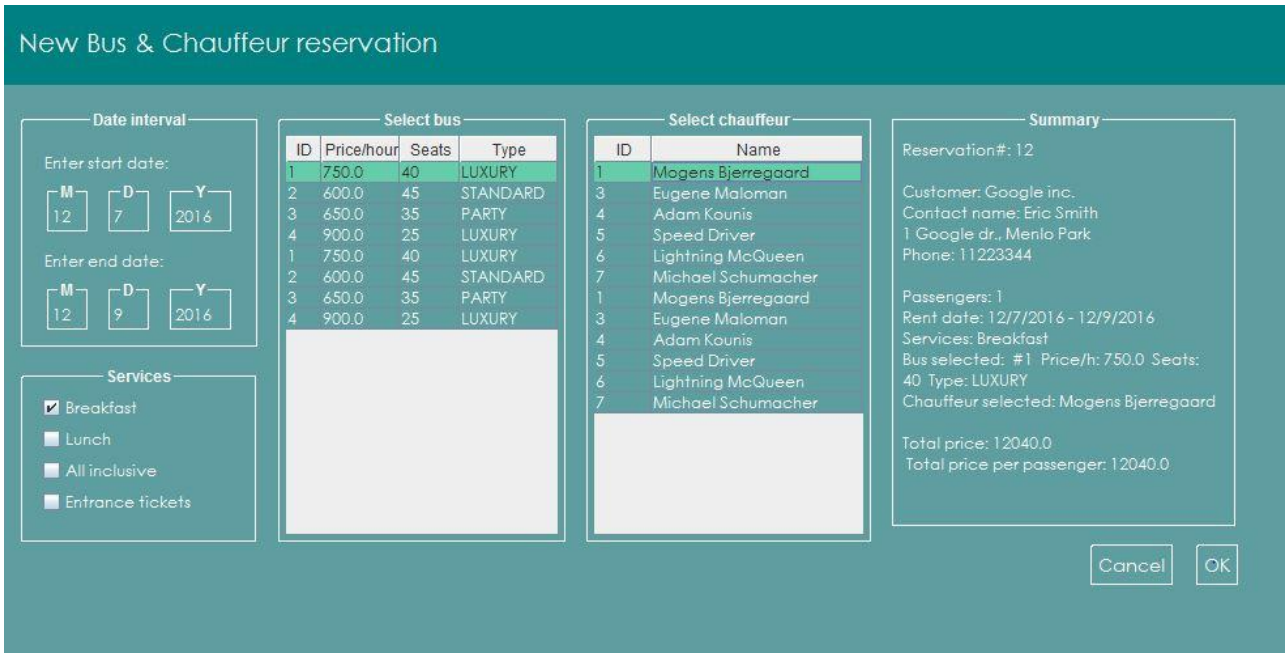



*Illustration 5: The image shows the second page of the process of making a new bus reservation, containing the steps 3-6: Specifying the date interval, Selecting extra services, Choosing the bus, and selecting the Chauffeur.*

The next **example** of how the use case and activity diagrams is being structured in the GUI design of the Autobus system, is the task for managing predefined tours. The window for the tours archive, is accessed from the menu bar Edit: Tours. The tours archive window allows the customer to create a new predefined tour, and updating, deleting and searching predefined tours already created in the archive.

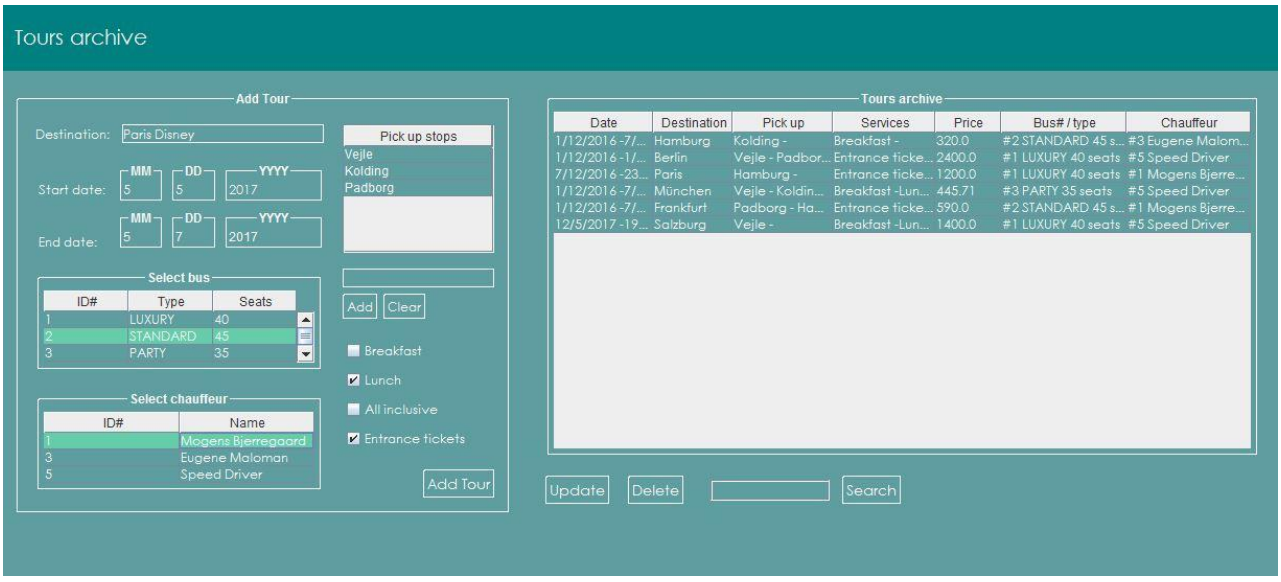



*Illustration 6: The screen image shows the Tours archive window. The left panel provides an option for the user to create and add a new predefined tour, specifying the destination, date interval, pick up stops, bus, chauffeur, and services included for the tour. The right panel displays a list of the tours archive. It allows the user to select and edit a parameter in a specific tour object, and update it – or delete the selected tour.*

## The class diagram

The class diagram is displayed below to illustrate the relations between the classes used in the Autobus system. Only the most important attributes and methods required for understanding the reciprocal relations are included, leaving out basic methods such as the *mutators* and *accessors*.

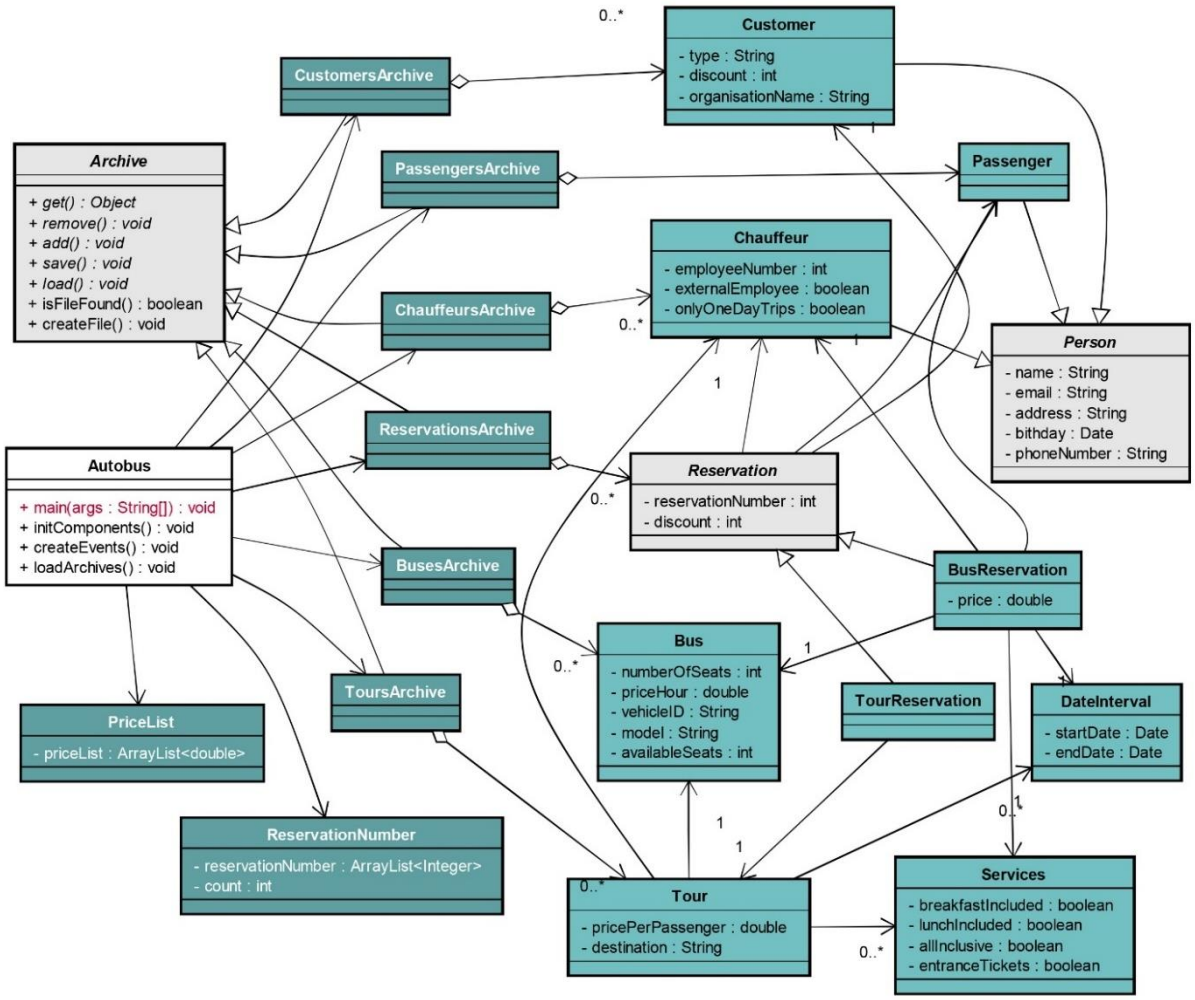



*Illustration 7: The class diagram showing only important attributes and methods.*

The darker green classes are implementing the *Archive* abstract class. Each of the archives uses ArrayList for collecting the objects related, for instance; the *CustomerArchive* has a collection (ArrayList) of *Customer* objects. The *BusReservation* class and the *TourReservation* class implement the abstract class *Reservation*. The abstract class *Person*, is implemented in the *Customer* class, the *Passenger* class, and the *Chauffeur* class.

# Implementation

In this chapter, some of the more interesting parts of the code will be explained further in detail. For versioning control during the code implementation, Github was used. The Github repositories can be found at this link: https://github.com/MogensBjerregaard/IT-SEP1X-A16_rc

### The Autobus class

The Autobus class contains the main method which initiates the Autobus constructor and set the frame visible at full screen size, using a lambda expression `EventQueue.invokeLater()`, since almost all code that creates or interacts with Swing components must run on the event dispatch thread (Oracle, 2015).

```
public static void main(String[] args) {
	EventQueue.invokeLater(() -> {
		Autobus frame = null;
		try {
			frame = new Autobus();
		} catch (Exception e) {
			e.printStackTrace();
		}
		frame.setVisible(true);
		frame.setExtendedState(MAXIMIZED_BOTH);
	});

}

public Autobus() throws Exception {

	initComponents();
	createEvents();
	loadArchives();

}
```

The Autobus constructor calls three methods:

1. `initComponents()` *for creating all the GUI components*
2. `createEvents()` *for creating all the event listeners*
3. `loadArchives()` *for reading the archived data from file*

### Data persistence

The purpose of the `loadArchives()` method is to deal with data persistence. As an example of handling data persistence, the following code snippet is displaying how the import of data related to the toursArchive at system launch is addressed.

```
toursArchive = new ToursArchive();

if (toursArchive.isFileFound()){
	toursArchive.load();
	listTours();
} else {
	toursArchive.createFile();
	listTours();
}
```

First an instance of the toursArchive is created, and the the system checks if there already exists a toursArchive data file, and then reads the data and displays them in a table. If no existing data is found, which would be the case when the system is launched initially, the system calls the `createFile()` method.

The code snippets below, shows how the check for an existing data file is done, and how the data import is handled.

```
public boolean isFileFound(){
	Path path = Paths.get("C:\\Autobus\\ToursArchive.dat");
	return (Files.exists(path));
}

public void load () throws Exception{
	FileInputStream fileInputStream =
	    new FileInputStream("C:\\Autobus\\ToursArchive.dat");
	ObjectInputStream objectInputStream =
	    new ObjectInputStream(fileInputStream);
	try {
	      ArrayList<Tour> otherToursArchive =
	          (ArrayList<Tour>)objectInputStream.readObject
	      this.toursArchive=otherToursArchive;
	} finally {
	      objectInputStream.close();
	}
}
```

The `createFile()` method is called when no data file can be found, and it creates an empty data file for storing and importing data, by adding an object to the ArrayList and saving the data file, and the removing the object that was just added and saving the data file once again. The code snippets for the `createFile()` and `save()` methods are displayed on the next page.

```
public void createFile() throws Exception{
	addTour(new Tour(""));
	save();
	toursArchive.remove(0);
	save();
}

public void save() throws Exception{
	Path path = Paths.get("C:\\Autobus");
	if (!Files.exists(path)){
		try {
			Files.createDirectory(path);
		} catch (Exception e) {
			JOptionPane.showMessageDialog(null,
			"Error: Unable to create directory!");
		}
	}
	FileOutputStream fileOutputStream =
		new FileOutputStream("C:\\Autobus\\ToursArchive.dat");
	ObjectOutputStream objectOutputStream =
		new ObjectOutputStream(fileOutputStream);
	try {
		objectOutputStream.writeObject(toursArchive);
	} finally {
		objectOutputStream.close();
	}
}
```

**Using lambda expressions in event listeners**

The last example of the code implementation is demonstrating how lambda expressions was used for some of the event handlers, as displayed in the code snippet below. It reduces the lines of code, since the anonymous inner class is omitted, and moreover makes it somewhat more readable (Naftalin, 2015).

```
// Prior to Java 8 syntax:

rdbtnNewBusCompany.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent event) {
		rdbtnNewBusPrivate.setSelected(false);
		rdbtnNewBusSchool.setSelected(false);
	}
});


// Java 8 lambda way:

rdbtnNewBusCompany.addActionListener(event -> {
	rdbtnNewBusPrivate.setSelected(false);
	rdbtnNewBusSchool.setSelected(false);
});
```

# Test

1. Test performed: Executing a reservation for a predefined tour and reservation for a bus and chauffeur.
    a. Result: All functions are working correctly, and the system coordinates the availability of resources (seats, buses, and chauffeurs) for a reservation, as well as calculating the total price and a price per passenger.
2. Test performed: Executing a search, add, update, and remove function related to the tour reservations.
    a. Result: All functions are working correctly, a tour registers destination, durance, extra services (all inclusive, breakfast, lunch, entrance tickets), and any pick-up stops.
3. Test performed: Executing a search, add, update, and remove function related to the bus reservations.
    a. Result: All functions are running correctly, and each bus reservation registers durance, and extra services.
4. Test performed: Executing a search, add, update, and remove function related to chauffeurs.
    a. Result: All functions are running correctly. The system registers if the chauffeur only drives one day only, if he is a full-time employee or external, name, address, employee-id, phone number, and email.
5. Test performed: Executing a search, add, update, and remove function related to buses.
    a. Result: All functions are running correctly. The system registers seat capacity, bus type (luxury, party, or standard), and price per hour.
6. Test performed: Executing a search, add, update and remove function related to customers.
    a. Result: All functions are running correctly. The system registers purchase history, and a discount can be applied automatically. The customers name, organisation name and type (company, school, private), address, phone number, birthday, and email are registered.
7. Test performed: Executing a search, add, update, and remove function related to passengers.
    a. Result: All functions are running correctly. The system registers name, address, phone number, birthday, email.

# Result

All 7 functional requirements listed has been implemented, tested, and is working.

The 3 non-functional requirements have all been implemented. Storing and retrieving data has been tested, and is working.

The abstract class *Archive* has not been implemented.

# Discussion

All though this might be considered a relatively small project task and the Autobus system a somewhat simple software application, all members of the project team agree that the project process has been a fairly complex challenge at certain moments – nevertheless it was expected, taken into account that it is a first semester project managed by novice developers.

Perhaps one of the more notable assets of the Autobus system worth highlighting, is the look and feel of the graphical user interface, and the simple structure of the navigation menu – the simplicity of it all give the impression of a system serving well as a helpful tool to manage the reservations.

On the list of possibly further obvious improvements – conceivably for a version 1.1 update – would be looking at a better way to structure the code for the GUI components and the event handlers.

# Conclusion

The project process has been a method of 4 sequential phases – analysis, design, implementation, and testing. It appears obvious, that the test results builds upon the quality of the implementation, analysis, and design processes. Being inexperienced project managers and developers, one of the difficult tasks was to estimate the time table for each of the 4 project phases.

In anticipation of the implementation phase being the most challenging and time consuming, there was put a lot of focus and work into processing the analysis phase as swiftly as possible.

Similarly, for the design phase, finishing quickly had highest priority. As a consequence, the implementation phase was commenced even before the design phase had been fully completed. During the progression, this caused the necessity for several changes to the design, which lead to some confusion to what had been changed, and what had been synchronized between the design diagrams and the implemented code.

In retrospect, dedicating more time to the analysis and design phases, would be well spent, as it appears to impact the time required for the implementation significantly - perhaps more than expected. Besides, a more thoroughly developed analysis and design phase, would most probably make it easier to distribute the work between members.

The testing result shows that 100% of the customer requirements have been implemented, and all of the implemented requirements are functional, and working as intended.

The Autobus class includes the code for the GUI components and its associated event handlers, and contains a little less than 15.000 lines in total. Separate classes for the GUI and event handlers, would most likely be a better design and make the implementation easier.

# References

Barker, S., 2016. *The Psychology of Color in Web Design - Vandelay Design*. [online] In Design Psychology. Available at: <http://www.vandelaydesign.com/the-psychology-of-color-in-web-design/>.

Galitz, W.O., 2007. *The essential guide to user interface design : an introduction to GUI design principles and techniques*. 3rd ed. Available at: <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=191679>.

Lal, R., 2013. *Digital design essentials : 100 ways to design better desktop, web, and mobile interfaces*. Design Essentials. Available at: <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=601915>.

Naftalin, M., 2015. *Mastering lambdas : Java programming in a multicore world*. McGraw-Hill Education.

Oracle, 2015. *Initial Threads (The Java™ Tutorials > Creating a GUI With JFC/Swing > Concurrency in Swing)*. [online] Available at: <https://docs.oracle.com/javase/tutorial/uiswing/concurrency/initial.html>.

Rochelau, J., 2015. *The Ultimate Guide to Color Theory for Designers*. [online] Available at: <http://www.vandelaydesign.com/ultimate-guide-color-theory-designers/>.

Snell, S., 2016. *Tips for UI Design Colors and Color Matching Techniques*. [online] Available at: <http://www.vandelaydesign.com/ui-design-colors/>.

Virgillito, D., 2016. *How to Choose the Right UI Design Colors for Your WordPress Site | Elegant Themes Blog*. [online] Elegant Themes Blog. Available at: <https://www.elegantthemes.com/blog/tips-tricks/how-to-choose-the-right-ui-design-colors-for-your-wordpress-site>.

# Appendix A

## The Interview

***"Please tell us about your company, Mr Driver."***
Trip Driver: *"Just call me Trip, we are not so formal here at VIA Bus."*

***"OK, Trip… Are you a travelling agency with bus travels?"***
Trip Driver: *"We are much more than just that. We see ourselves as specialists in bus tours so we offer both travels and trips. Currently we have one-day trips to Flensburg, Copenhagen, Skagen, Djurs Sommerland, and Legoland, plus occasionally other destinations when we feel that there could be a market for it. In the school holidays, we arrange one-week travels with all included. The service that we are especially proud of is our bus-and-chauffeur service."*

***"What is 'bus-and-chauffeur'?"***
Trip Driver: *"Thank you for asking. There is a big market for bus tours and we arrange it exactly how the customer wants it. A bus and a chauffeur will be reserved for a specific trip, for example a school trip, a company visit, a trip to an event or different locations the same day. A minibus or a luxury bus is often reserved for small or larger groups going to Tivoli, bowling centres, to parties or similar. Sometimes our party bus is reserved the whole day for a bachelor party, and sometimes for luxury transportations or trips – better and more extravagant than limousines, don't you think?"*

***"Well… Isn't it expensive?"***
Trip Driver: *"Absolutely not."*

***"What about the driver? Is he a part of your staff?"***
Trip Driver: *"Some of our chauffeurs work here full time, but we are also in contact with some other chauffeurs, who we only call at times when extra drivers are needed.*

***"Full time drivers. Don't they have a lot of wasted time, when there are not any trips?"***
Trip Driver: *"Yes, and no. Our chauffeurs are all specialists in their field, so besides driving they also clean the busses, drive them to service, and sometimes even fix small problems with the busses. In addition, they of course also spend some time writing down reservations from customers and prepare routes for trips. However, it is true that it is a problem for us that we pay a full salary even when a chauffeur is not driving or working with any of the other related jobs I just mentioned. This is not easy to fix without some kind of computer program to optimise the use of our chauffeurs. In other words, we could really use a better overview of when each of the chauffeurs is available."*

***"What kind of system do you have in mind?"***
Trip Driver: *"Oh, I don't know much about these kinds of things, but I'm expecting some sort of standard looking program that we can use with a keyboard and mouse. You know the kind: Some fields to enter the data and some buttons, menus or tabs to do things: Saving and searching and what have you. We don't want any accidents with unsaved data."*

***"So far I have understood that the system should help with determining when a driver is available for a trip."***
Trip Driver: *"Yes, and we would also like to have the system handle all our reservations for travels, trips and bus-and-chauffeur services."*

***"How do customers reserve a travel?"***
Trip Driver: *"You make a reservation for the two kinds of tours; travels and one-day trips just like at any other travel agency. Depending on the number of passengers we then select a suitable bus and find an available chauffeur."*

***"What about bus-and-chauffeur?"***
Trip Driver: *"Here you reserve a bus and a chauffeur for a day, or maybe just a couple of hours, by specifying your destination, or destinations. Sometimes we provide food like breakfast or lunch in the bus, or reserve a restaurant for dinner. In the party bus you can also reserve a party guide, and of course use everything available in the bus. Right now we offer a discount if you are a frequent customer of any of our travels, trips or bus-and-chauffeur services."*

***"Do you keep track of your customers?"***
Trip Driver: *"Normally we write down at least the name, address and phone number for the customer making the reservation. For tours like travels and one-day trips all passengers are registered with their name, address and birthday. Sometimes we also get an email address if he or she wants a newsletter from us. I have to say that it is not easy to locate a frequent customer, especially if it is a company."*

***"What is the difference between a customer and a passenger?"***
Trip Driver: *"A customer is the person paying for the reservation, typically also the person making the reservation. A reservation for a one-week travel or a one-day trip is normally made for more than one passenger, for instance a family, and the customer is then typically one of the passengers on the tour. We register the customer and all the passengers for a reservation like that. A customer for a bus-and-chauffeur service may be a private person, a company or a school, and in some cases a secretary will be making the reservation in which case we write down the secretary as the customer. This will be OK as long as we also know the company name in order to give a later discount to the company and not only to another trip reserved by the same secretary."*

***"OK, I see. What kind of data do you then store for drivers?"***
Trip Driver: *"We actually like to call them chauffeurs instead of drivers. The plan is to have a new system that is independent from our payroll system. From your system we would like to have their names, their address, the same 5-digit employee-id that is also used in our payroll system, when they are on a trip, when they are available, and their wishes for trips… and actually also their phone number and email address, because that is the way we contact them."*

***"Sorry about the driver-thing. You said something about wishes for trips."***
Trip Driver: *"Yes, we have one chauffeur who will not drive the party bus, one who wishes to have only one-day trips and others who like to take the summer week travels, in particular my oldest son Speed. We want to fulfil their wishes because a happy chauffeur gives happy bus passengers."*

***"…Speed Driver?"***
Trip Driver: *"Yes, that's my son, the best chauffeur we have – always on time. He says he likes these travels because he sees them as a sort of vacation for himself as well."*

***"Would you like to have an online Web system where customers can make a reservation on their own?"***
Trip Driver: *"No. We would like to have a system to run on the front desk computer only. No website, and only me and my employees using the program. Oh, and one more thing, my niece tells me she's learning something called Java, and if you could make the system using that, then I think it would be very helpful. Can you do that?*

***"No problem. We can make a single user system in Java for you."***
Trip Driver: *"Single user? There are more employees who should use the program – but of course not at the same time…. and I trust all my employees, so don't waste time making a login with passwords and all that."*

***"How do you set the prices for at trip, Trip?"***
Trip Driver: *"The price for a seat depends on the duration of a trip and the distance driven, and is independent of the passengers' age. Prices for extra services like food, accommodation and tickets, for example to Legoland, are normally different for adults and children. We have an idea for prices for standard bus-and-chauffeur services, but we set the prices for each of these services depending on the trip and extra services. There is no standard way to set the price and it would be perfectly OK for me if the system allows us to type in the price for each reservation."*

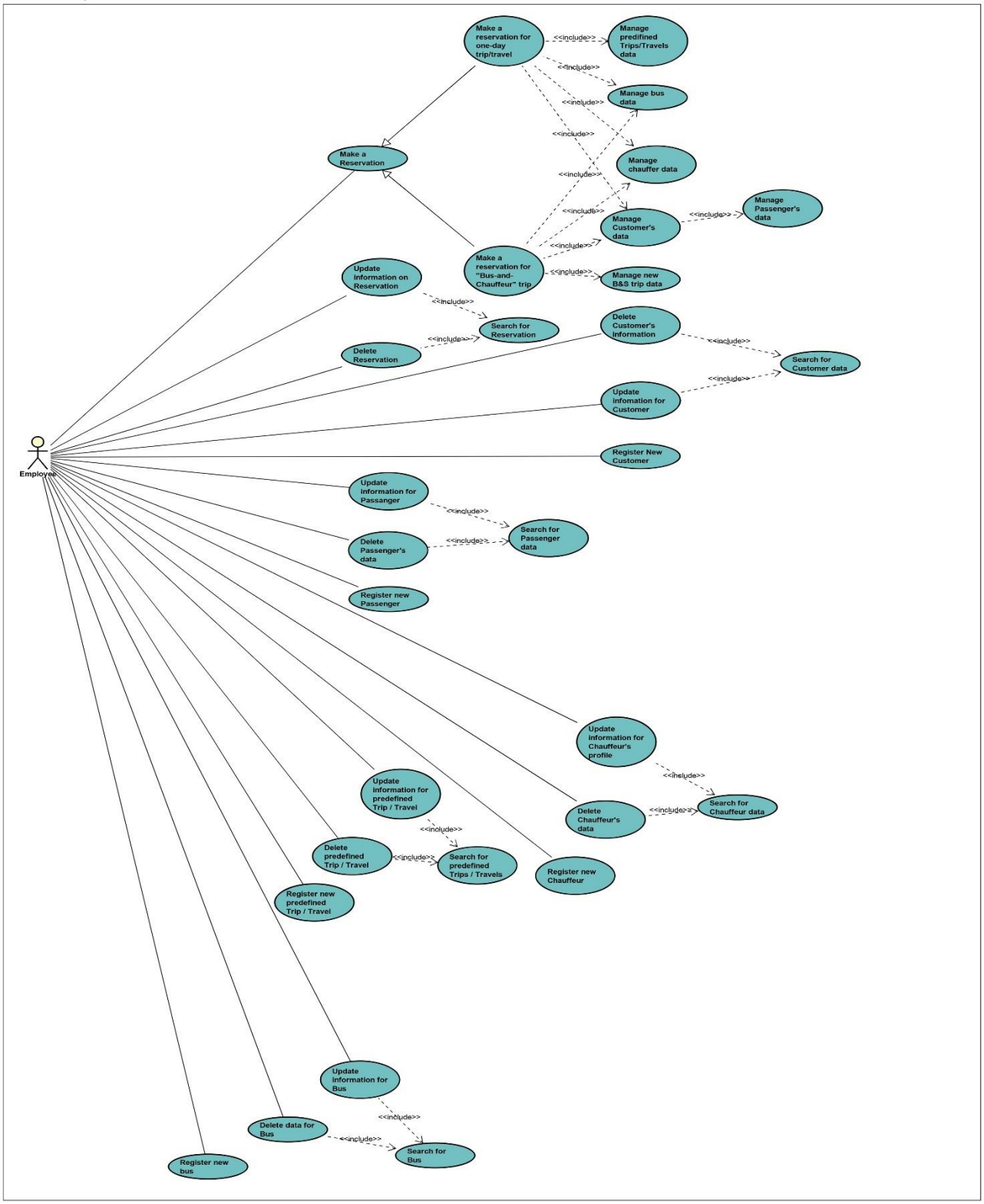
***"Let me try to sum this up: You need to be able to register your tours, both your own designed trips and travels, and trips renting a bus with driver… sorry chauffeur, register customers including their email address, register passengers for your own tours, register reservations including date and time, see available chauffeurs for a tour, find a suitable bus for a tour, and chauffeur wishes for tours. Am I missing anything? I suppose you need to be able to cancel or change a reservation."***
Trip Driver: *"Certainly, and we also need to be able to find the frequent customers or companies to give them a discount for their next reservation. Maybe, if possible, we could get a list of a chauffeur's day, when he is on each trip, approximately where to find him on the trip and if he is in between trips."*

***"Do you keep track of when you are where in all your trips?"***
Trip Driver: *"For the one-week travels and one-day trips we have approximate routes and schedules, some of which are pick-up/drop-off stops. For the bus-and-chauffeur services, we always have approximate time for departure, arrival to destination or destinations, and arrival back to our bus terminal in Horsens. Sometimes the route can be very precise and other times we know less about it."*

***"I think we have all that we need for now to start making your system. Thank you for your time, Mr … Trip."***
Trip Driver: *"No problem, and don't hesitate to return with any questions you might have."*

# Appendix B

## The use case diagram



# Appendix C

## The use case descriptions

Please see attached Astah files for the complete use case descriptions

# Appendix D

## The activity diagrams

Please see attached Astah files for the complete activity diagrams

# Appendix E

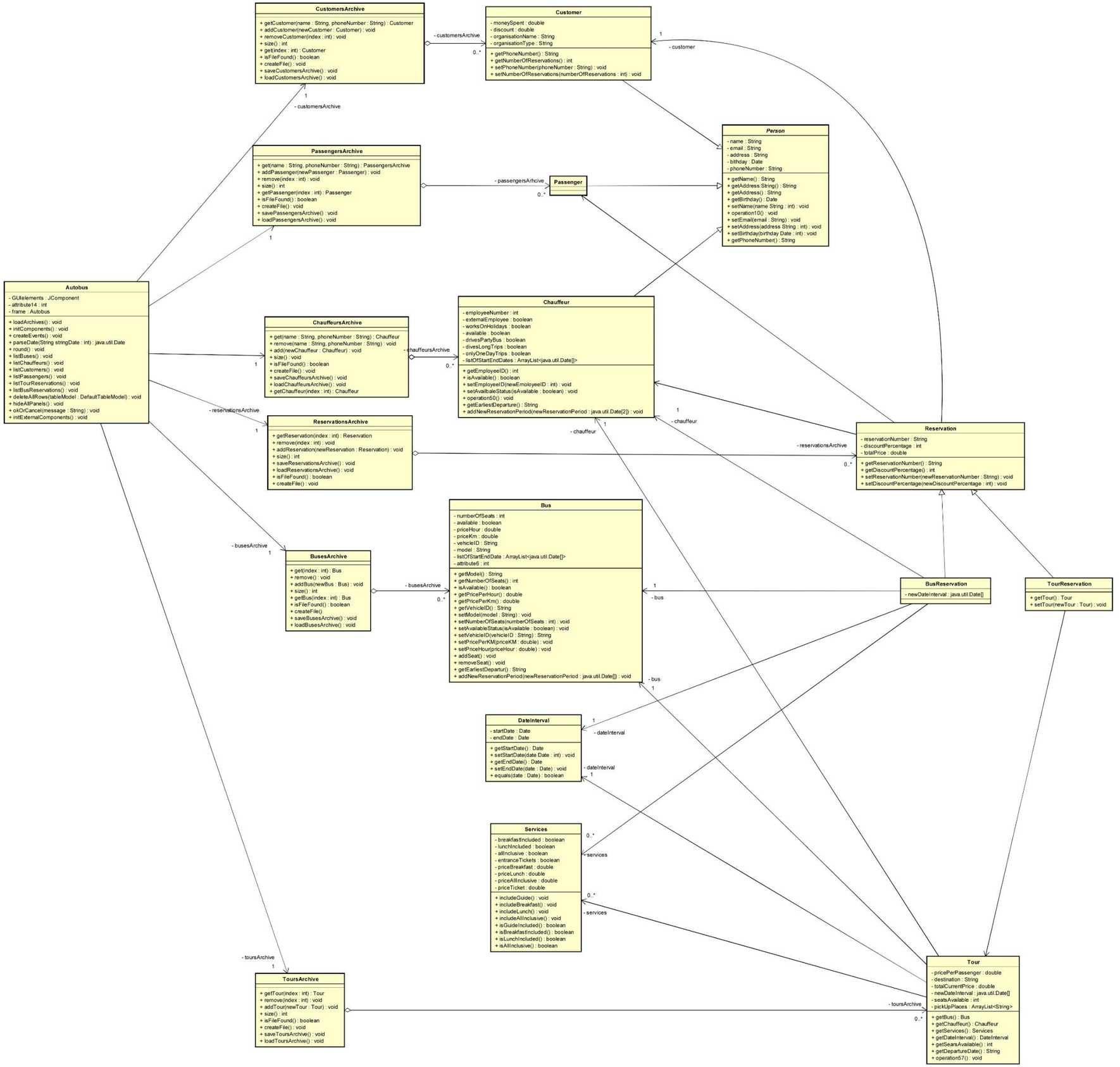
## The sequence diagrams

Please see attached Astah files for the complete sequence diagrams

ICT Engineering
Project Report SEP 1X A16 Group 1
VIA University College

# Appendix F

The class diagram



ICT Engineering
Project Report SEP 1X A16 Group 1
VIA University College

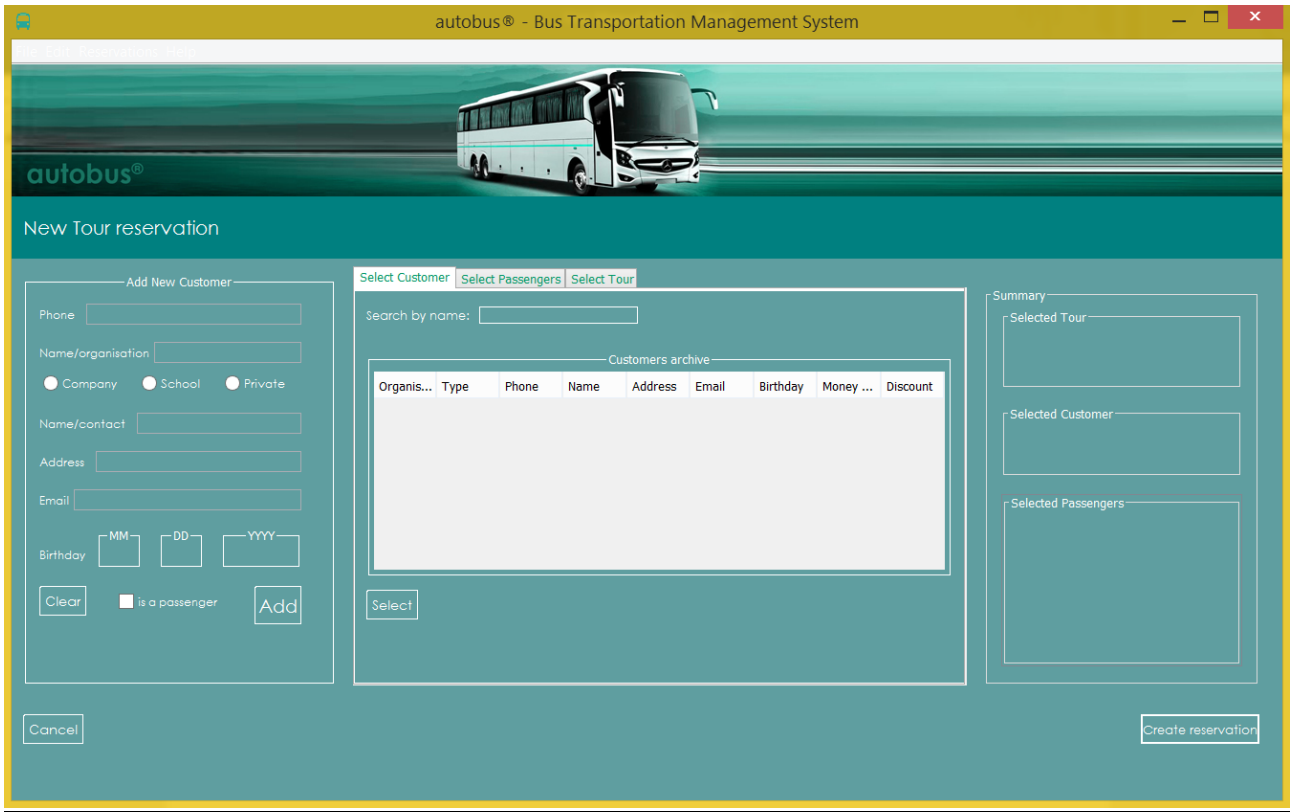
# Appendix G

## The user manual

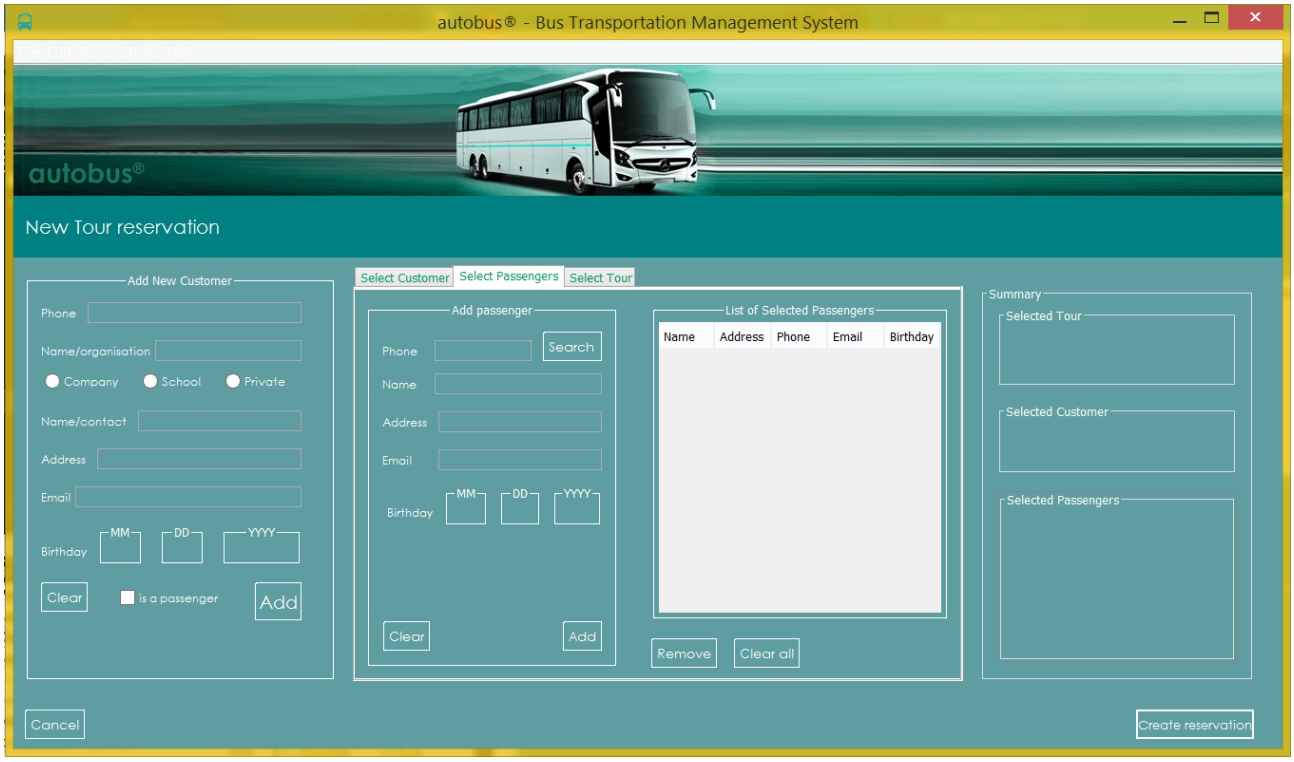
1. GENERAL INFORMATION
   1.1. SYSTEM OVERVIEW

   Autobus is an application, which allows collecting information about buses, Chauffeurs, Customers, Passengers, Tours and Prices. Using the collected data, it enables the user to make new tour and bus reservations. Furthermore, it has the option to update the information of the reservations as well as all the other collected data. The program is to be used by the owner of the bus company and the employees only.

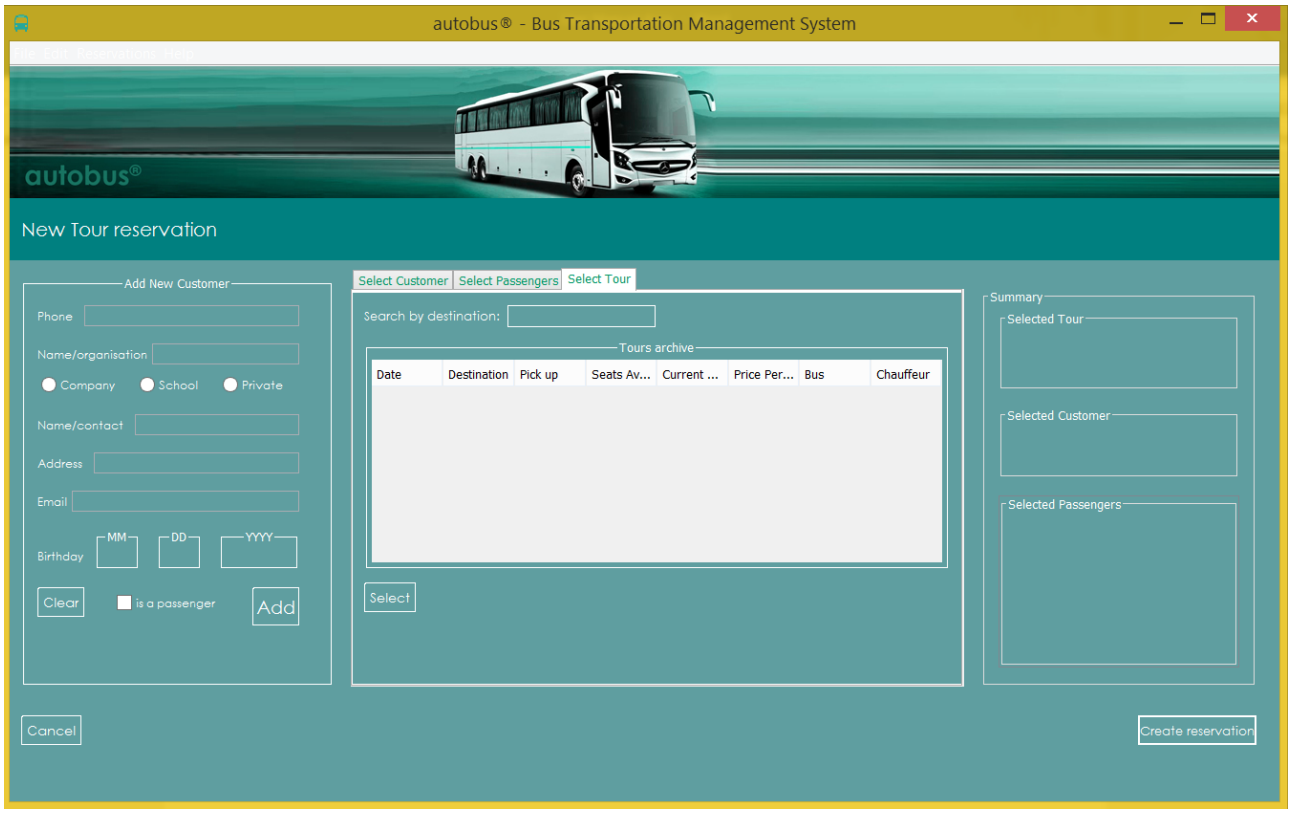
2. USING THE SYSTEM
   **2.1. FILE TAB**
   2.1.1.1. NEW TOUR RESERVATION TAB



In the new tour reservation tab, to add a new tour reservation, you need select on the customers from the “Customers Archive” table by right-clicking on them and then pushing the “Select Button”. The information about the selected customer will now appear in the “Summary” table. You can optionally add new customer by inputting the phone number, name, name/contact, Address, Email, Birthday information, checking which kind of an organisation the booking is made for, and check or uncheck the box if the customer is a passenger too. Lastly click on “Add” to add new customer record to the system. The new customer will now appear in the “Customers archive” table.

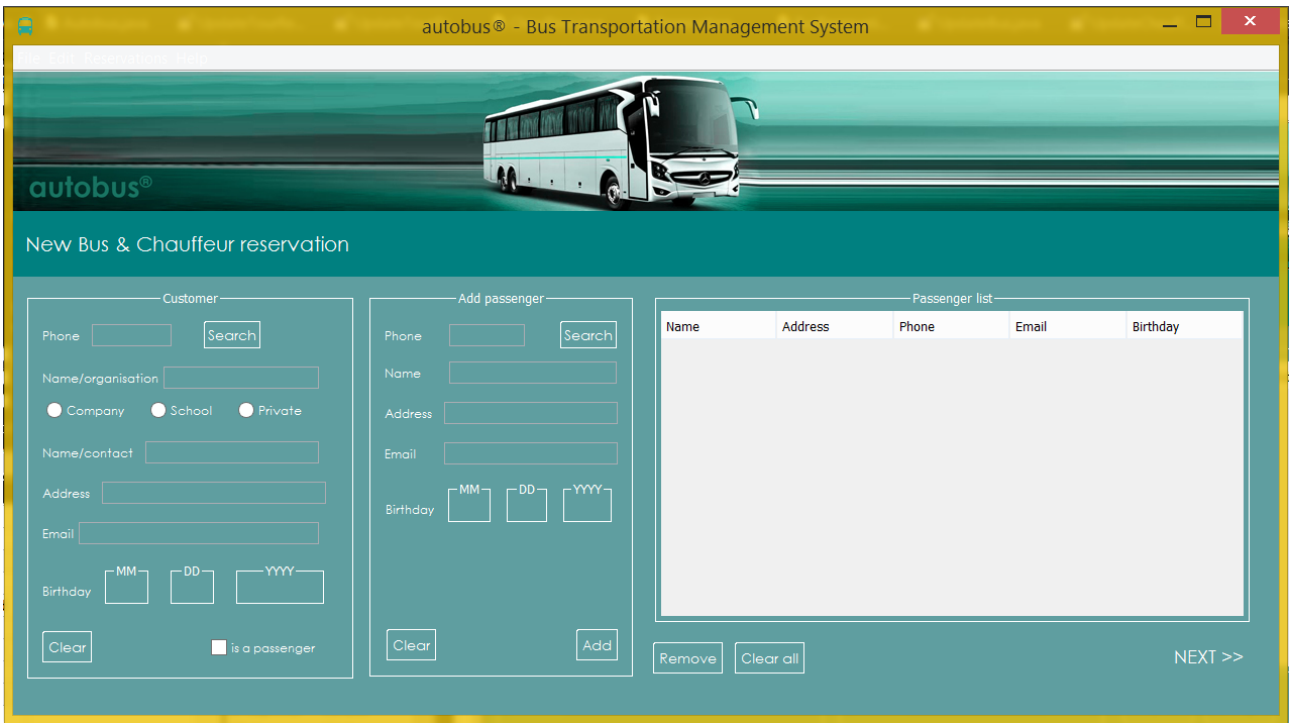



You can proceed to “Select passengers” tab. The “list of selected passengers” table is always empty in the beginning. To create a reservation, it is required to add at least one passenger to “list of selected passengers” table. To do this you can either search for the previously made passenger record by inputting passenger’s phone number and clicking “search” button, or you can create a new passenger record by filling all the information in the “Add Passenger” panel and clicking “Add” button. Either you way, if everything was done correctly, the passenger’s record will be added to the ”list of selected passengers” table. It is possible to add as many passengers as you want. You can edit the “list of selected passengers table” by using “remove” and clear all” buttons. Each time you add a passenger to “list of selected passengers” table, the “Summary” table will be updated with the main information about every passenger you have previously added.




Next, select one of the tours from the “Tours archive” table (located in the “Select tour”) tab by clicking on one of rows and then pushing “Select” button. After this, the “Summary” table will be updated with the main information of the selected tour. Finally, you can click on “Create reservation” button to create a new tour reservation and save it in the system.

2.1.1.2 NEW “BUS&CHAUFFEUR” RESERVATION TAB

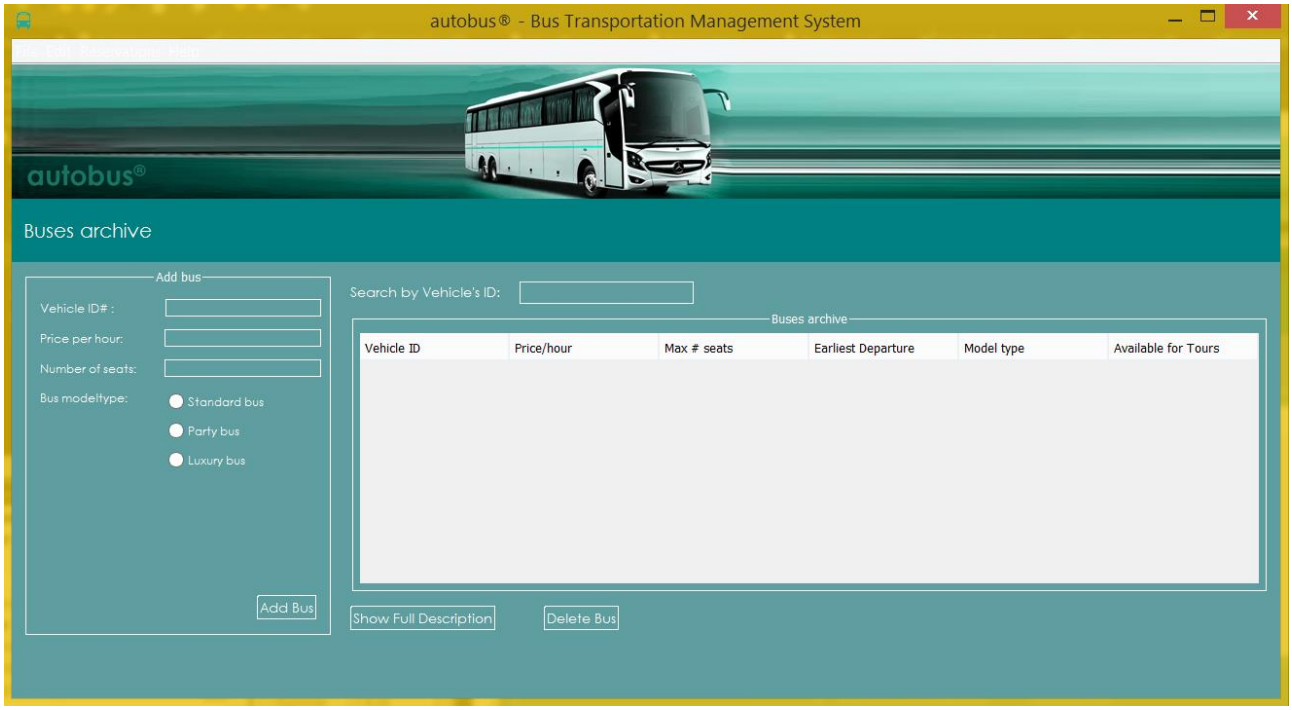



In the Bus reservation tab, to add a Customer, you need to input the phone number, click on “Search” so the customer’s information will show up and check the box if the customer is a passenger too. To add a Passenger, you need to input the phone number, click on “search” so the passenger’s information will show up and then click on “Add” to add the passenger or “Clear” to erase all the information. After you add the wanted number of passengers click on “NEXT” on the bottom left corner. After you click next a new tab will open in which you will need to input the starting and the ending date of the reservation and click on “search”. After that check for the services available for the trip and select the bus and Chauffeur from the two lists and then click “Select”. The Summary will be displayed on the right side and click “create” to finish the bus and chauffeur reservation process.

2.1.1.3. EXIT

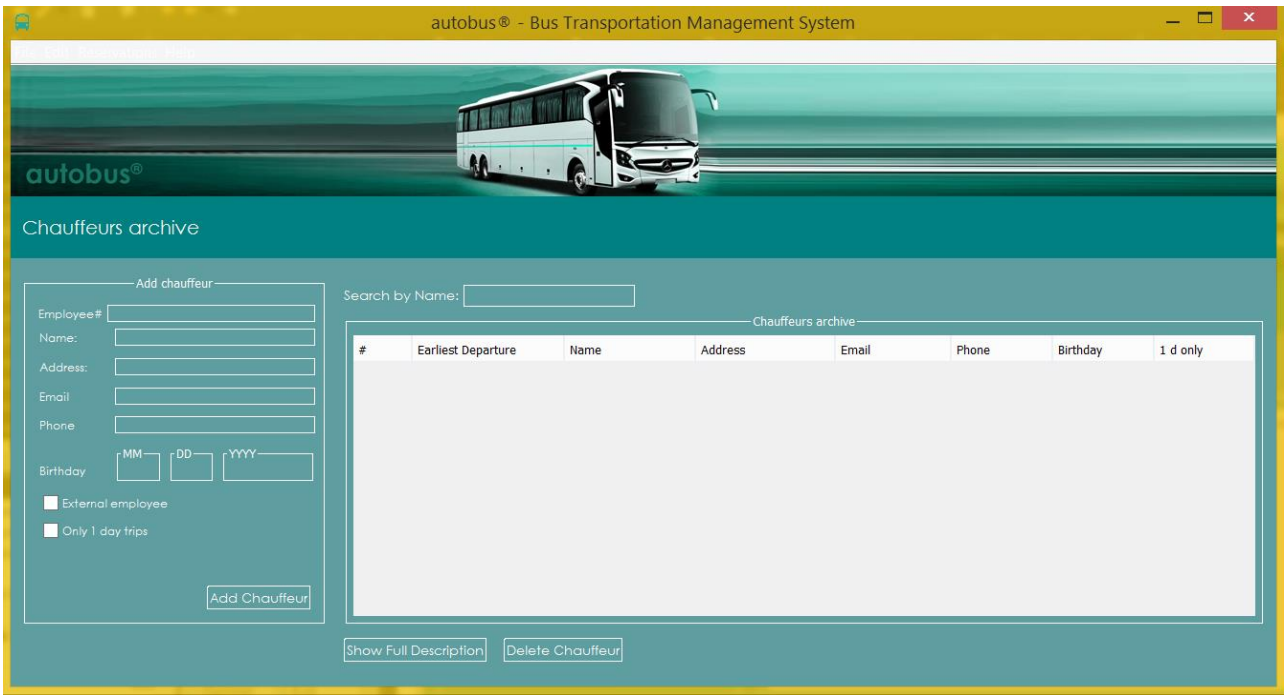
After you click on the exit tab the program will shut down. The same conclusion would happen if you click on the X on the top right corner of the window.

## 2.2. **EDIT TAB** 2.2.1.1 BUSES TAB



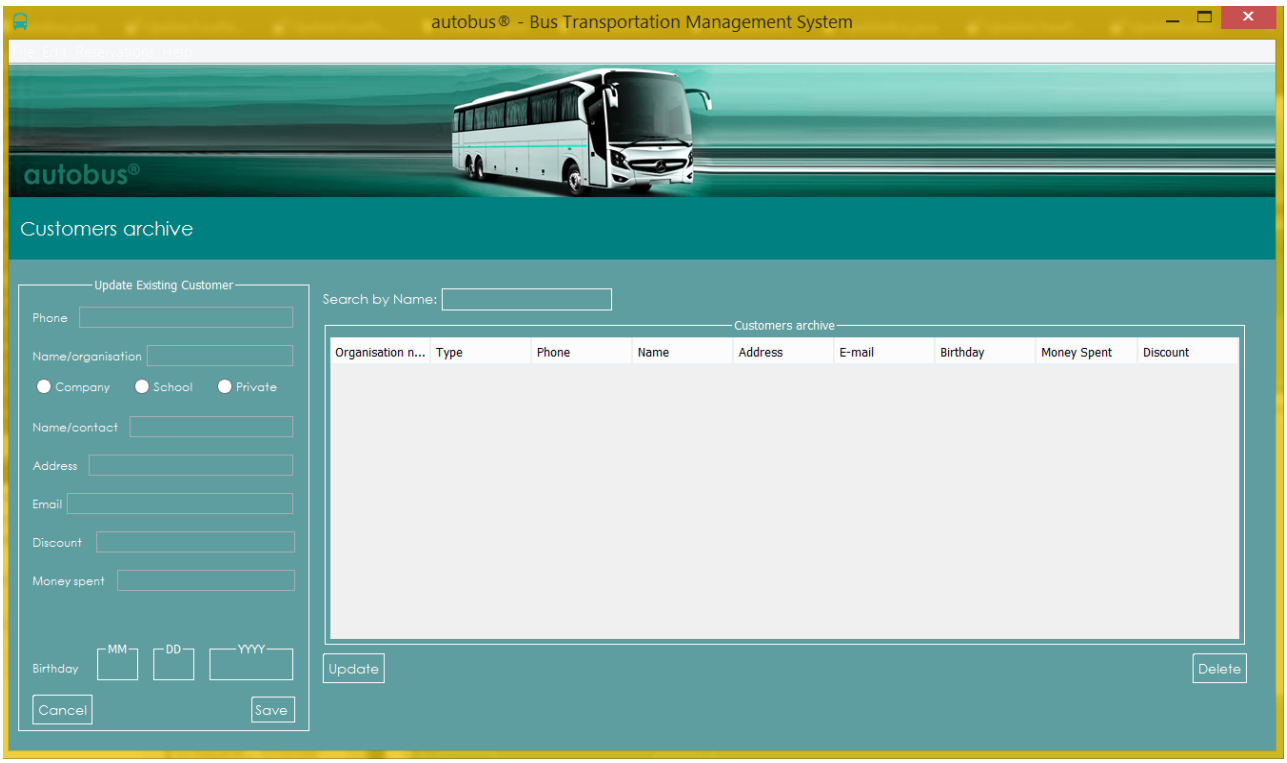

In the buses tab, to add a new bus, you need to input a Vehicle ID, a price per hour and the number of seats and you need to select the model/type of the bus. To add the bus, click on “Add Bus”. The new bus information will be shown in the table on the right side of the tab. There is also the option to search for a bus by inputting the vehicle’s ID on the search bar as well as deleting the information by clicking on “Delete Bus” of the bus or viewing the whole description by clicking on “Show Full Description”. After selecting one of the listed buses and clicking on “Show Full Description” you can see the buses information and you can update the information by changing the data and clicking on “Save Changes”. There is also the option to cancel the updating process by clicking “Cancel” or you can click “Back” button to return to initial buses tab.

2.2.1.2 CHAUFFEURS TAB



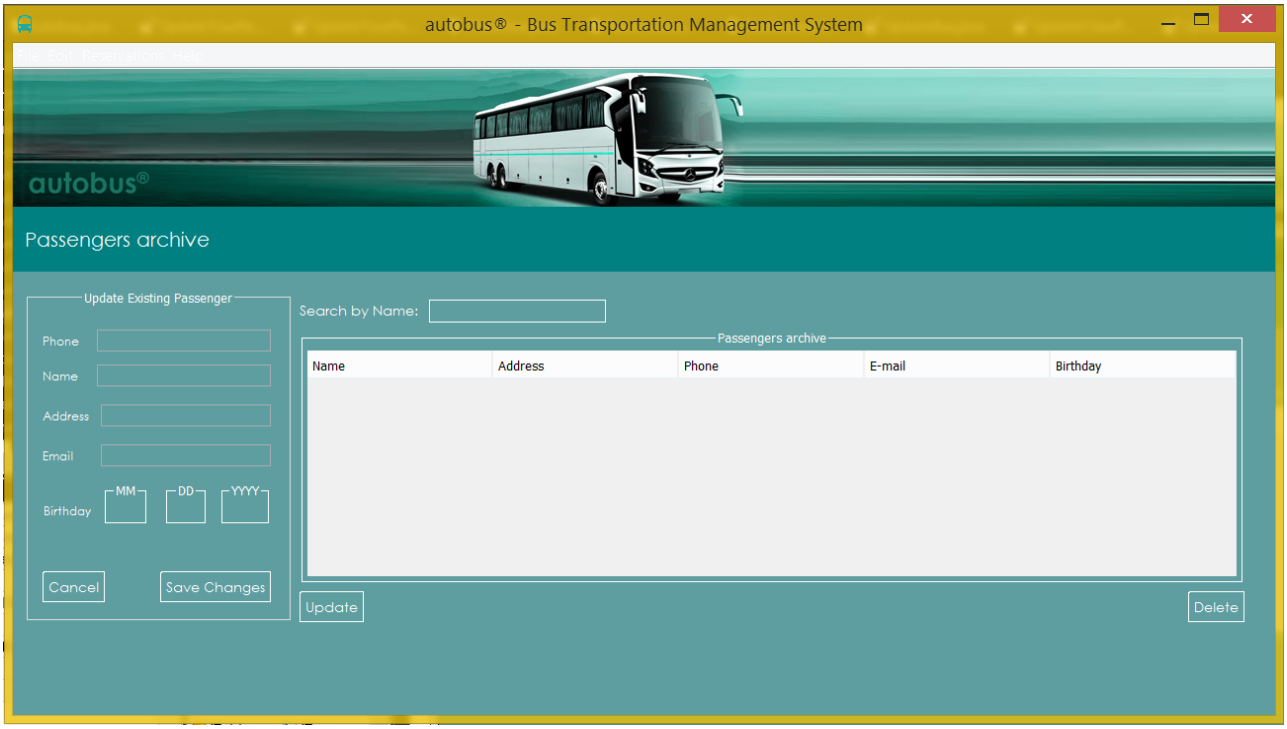

In the Chauffeur tab, to add a new Chauffeur, you need to input the Employee-ID number, name, Address, Email, Phone number, Birthday information and check or uncheck the boxes for “external employee” and “only 1 day trips”. Lastly click on “Add Chauffeur”. All the added information will be shown in the table on the right side of the tab. There is also the option to search for a Chauffeur by inputting the Chauffeur’s ID on the search bar as well as deleting the information by clicking on “Delete Chauffer” of the Chauffer or viewing the whole description by clicking on “Show Full Description”. After selecting one of the listed Chauffeurs and clicking on “Show Full Description” you can see the Chauffeur’s information and you can update the information by changing the data and clicking on “Save Changes”. Lastly, there is also the option to cancel updating process by clicking “Cancel” or you can click “Back” button to return to initial Chauffeur tab.

### 2.2.1.3 CUSTOMERS TAB



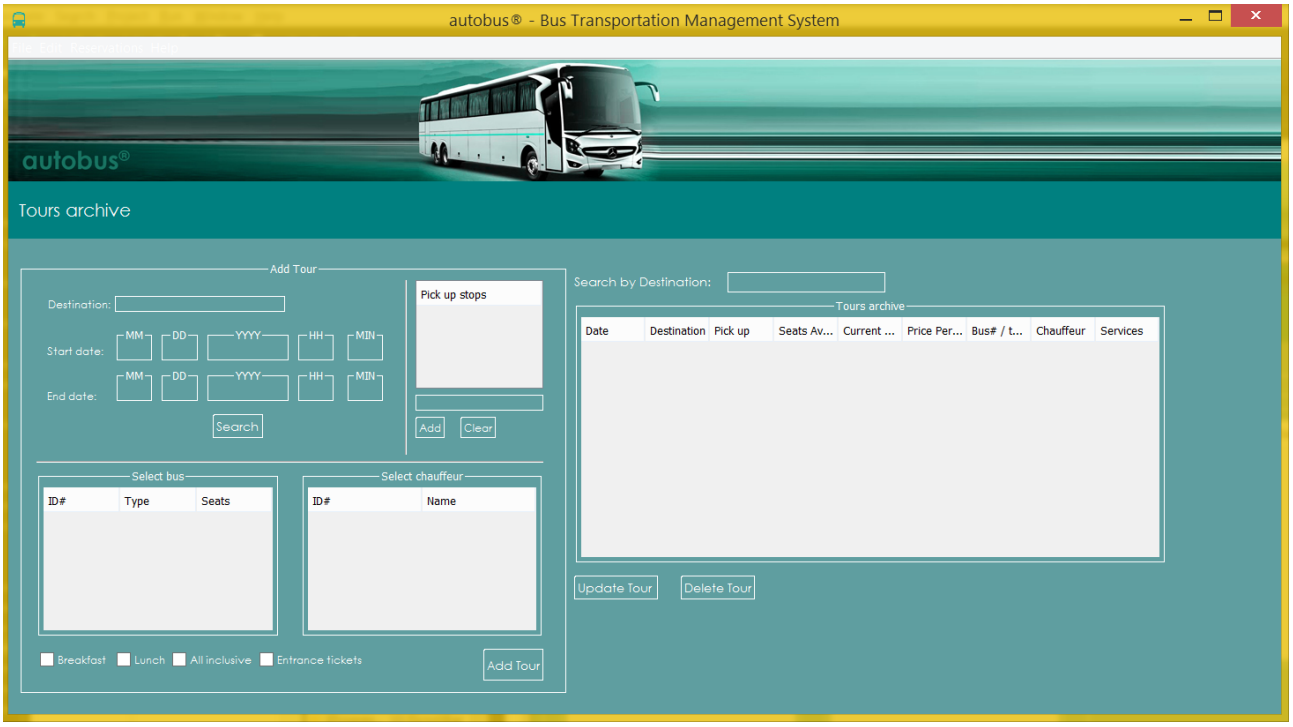

In the Customer tab, to add a new Customer, you need to input the phone number, name, check which kind of an organisation the booking is made for , name/contact, Address, Email and Birthday information. Lastly click on “Save”. All the added information will be shown in the table on the right side of the tab. There is also the option to search for a Customer by inputting the name of the customer on the search bar as well as deleting the information of the Customer by clicking on “Delete” or viewing the whole description by clicking on “Update”. After selecting one of the listed Customers and clicking on “Update” you can see the Customer’s information and you can update the information by changing the data and clicking on “Save”. Lastly, there is also the option to go back by clicking “Cancel”.

2.2.1.4 PASSENGERS TAB



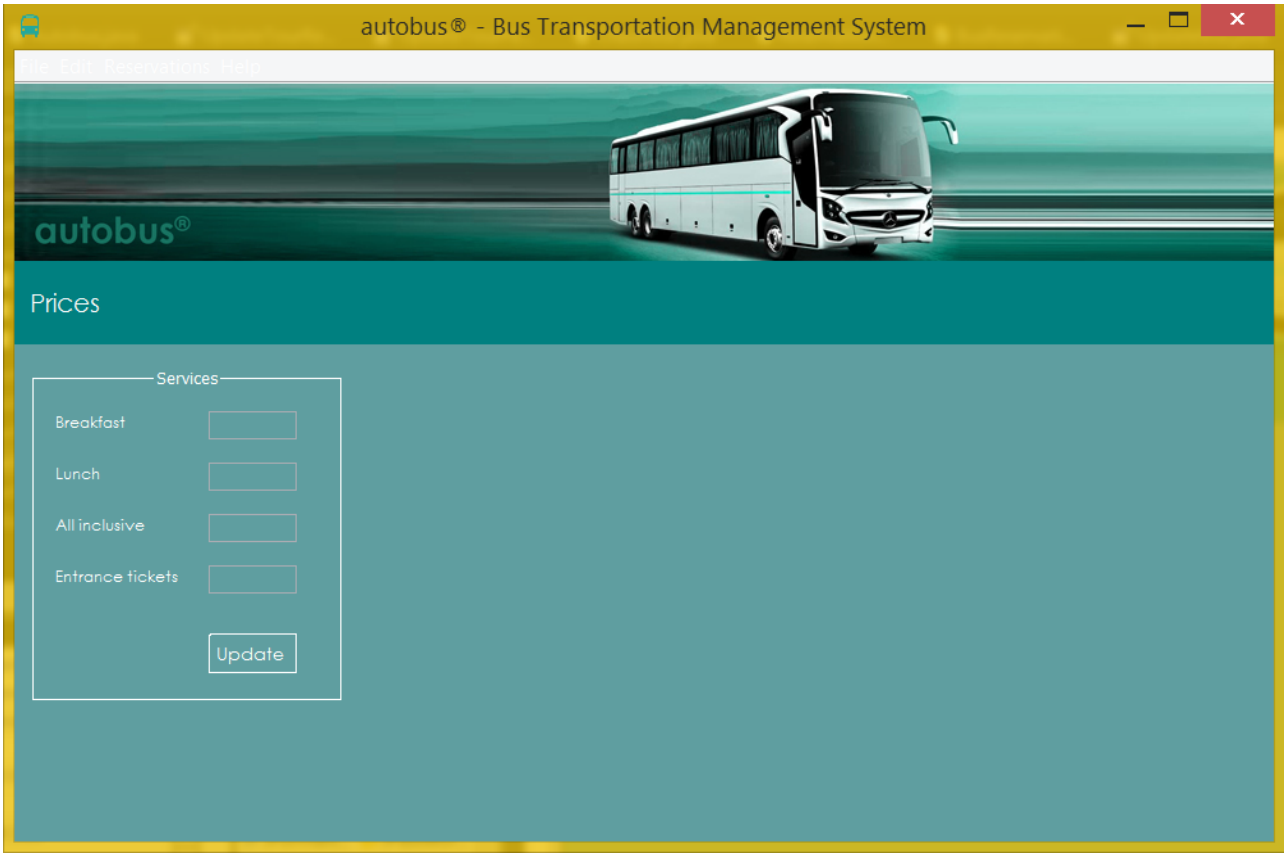

In the Passenger tab, to add a new Passenger, you need to input the phone number, name, Address, Email, and Birthday information. Lastly click on “Save Changes”. All the added information will be shown in the table on the right side of the tab. There is also the option to search for a Passenger by inputting the name of the Passenger on the search bar as well as deleting the information of the Passenger by clicking on “Delete” or viewing the whole description by clicking on “Update”. After selecting one of the listed Passenger and clicking on “Update” you can see the Passenger’s information and you can update the information by changing the data and clicking on “Save”. Lastly, there is also the option to go back by clicking “Cancel”.

### 2.2.1.5 TOURS TAB



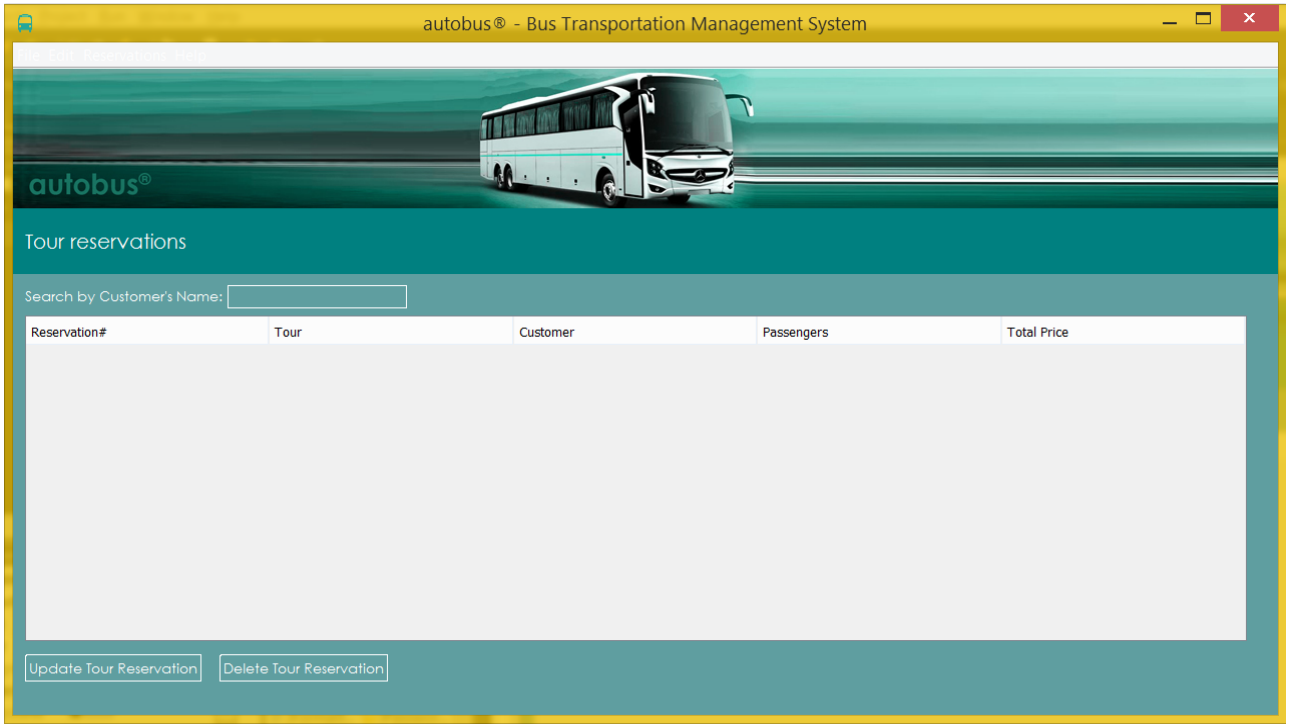

In Tours tab, you can add new tours, and view, update the existing ones. To create a new tour, you need first input tour's destination, start date and end date. You can then click the "Search" button, which will then show you every available bus and chauffeur in the given time in "Select bus" and "Select chauffeur" tables accordingly. You can now select one of the buses and one of the chauffeurs from the table by just right-clicking on them. Optionally, you can add pick-up places by inputting information for pick-up in text field located under the "Pick up stops" table and then clicking on "Add" button. You can clear the "Pick up stops" table by clicking on "Clear button" located under the "Pick up stops" table. You can see the most important information for every previously created tour in the "Tours archive" table. By selecting one of the rows from the "Tours archive" table you can click "Delete Tour" button to delete the tour from the system. After you have selected one of the rows from the "Tours archive" table you can also click "Update Tour" button to see more specific information about this Tour and optionally to make changes with this tour (Changing the dates, bus, or chauffeur for example).

2.2.1.6 PRICES TAB



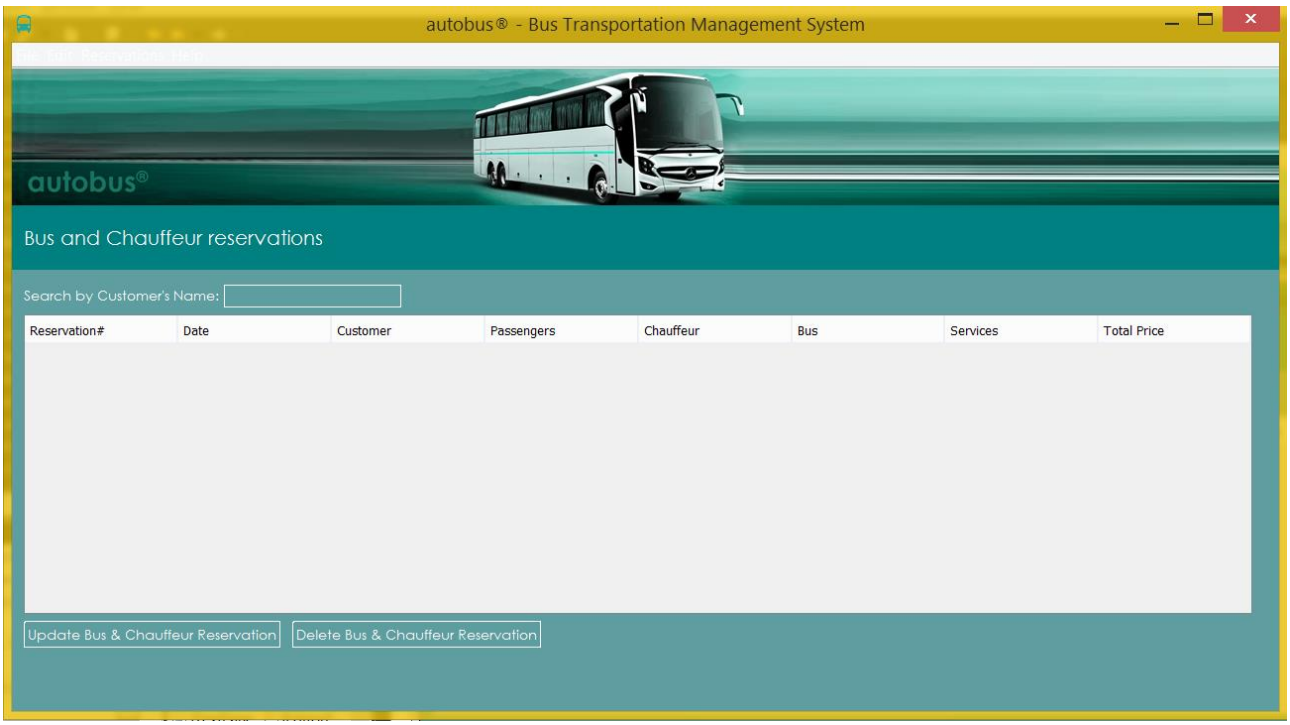

In the Prices tab, to input or change the prices for the services just click in the boxes the input the wanted prices. Finally click on “Update”, so the updates can be made.

## 2.3. **RESERVATION TAB** 2.3.1.1 TOUR RESARVATIONS TAB




In tour reservation tab, you can see all the Tour reservations. You can search for a specific tour reservation by typing the customer's name in the search bar. After you select a tour from the list you can update the information by clicking on "Update Tour Reservation" or delete it by clicking "Delete Tour reservation".

2.3.1.2 BUS RESARVATIONS TAB




In bus reservation tab, you can see all the bus and chauffeur reservations. You can search for a specific the bus and chauffeur reservation by typing the customer's name in the search bar. After you select a bus and chauffeur reservation from the list, you can update the information by clicking on "Update Bus & Chauffeur Reservation" or delete it by clicking "Delete Bus & Chauffeur reservation".

## 2.4. HELP TAB

2.4.1.1. ABOUT AUTOBUS

After you click on "About Autobus", a new window will show up showing information about the application. To go back you need to click on "OK" or the X on the top right corner of the window.

## 3. IMPORTANT TO DO WHEN STARTING THE PROGRAM FOR THE FIRST TIME

When starting the program for the first time, you need to select on the edit tab and add buses, Chauffeurs, Customers, Passengers, Tours and Prices that has been already decided and registered to the company. After this is done new tour reservation and new bus and chauffeur reservation can be made. This way all the reservations will be seen in the reservation's tab. For the program to work correctly, it is strongly recommended to avoid creating records of the same type with identical information (For example two customers with the same name and phone number).