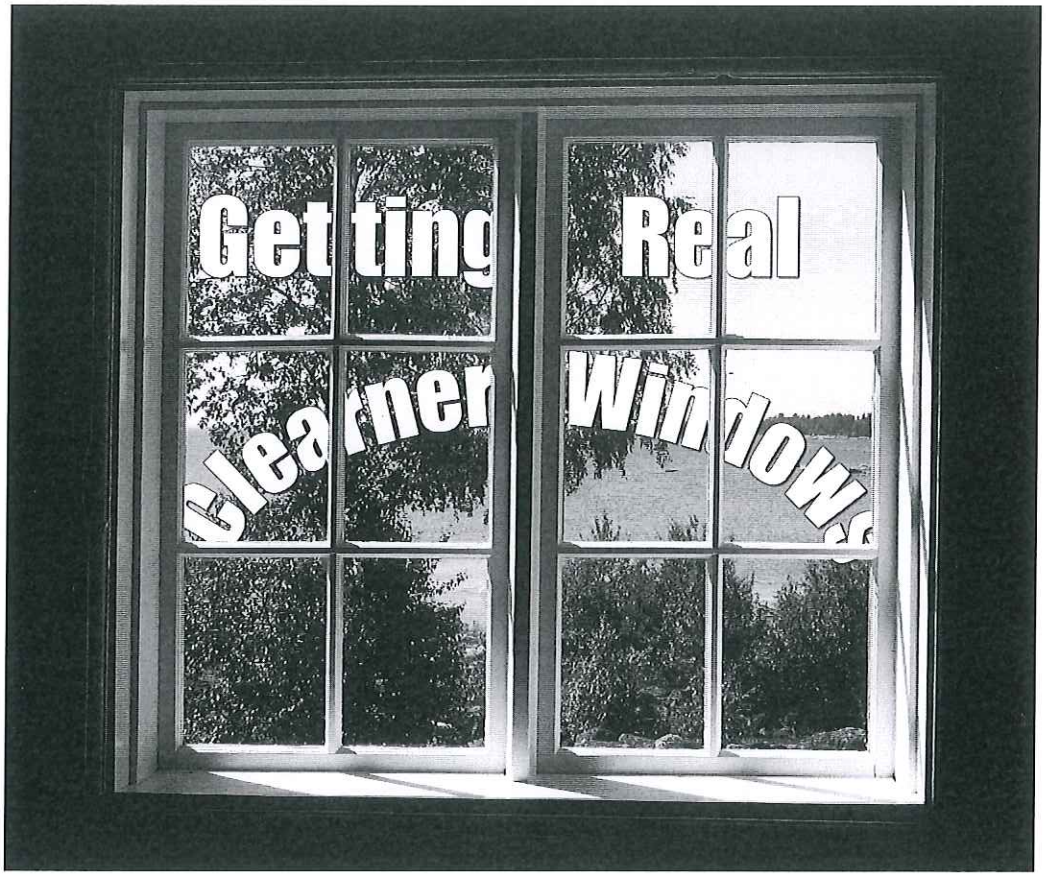
Project Getting Real



Jonas, Donato, Pedro, Loan
Class C - Group 1

Table of Content

Purpose

Our Product Owners currently uses a website-based subscription service to manage their daily business activities. It is quite costly as the company must pay for the subscription every month. The company has had a great growth rate during the past years and would like to expand its it system, thus they can save IT cost in the long term. Our mission is to help the company to develop a console app which can be used to manage employees' daily working schedules. This preliminary project helps our customers to have an overview concept about the actual app, which we hope to develop for them in the future

Business Model

Key Activities

CR Window Cleaning & Cleaning Service/CR Vinduespolering & Rengøring (CR) was founded in 1998 by Claus Rasmussen. He started his business quite simple with by running around and cleaning windows for its customers in Odense and Fyn.

The company kept growing steadily for many years. In 2009, CR grew to a certain size that Claus decided to employ Ole Lauritsen as a window cleaner. Initially Claus and Ole drove around together, but after a short period he had purchased another car thus both has more time to service all the new customers in a satisfactory manner.

Today Ole Lauritsen is the operation manager and he is responsible for the daily operation and at the same time has an overview of the company's vehicles and personnel consisting of 28 window cleaners and cleaners, and office staff

CR has almost 20 years of experience in cleaning industry. They offer window cleaning services as well as normal cleaning services for businesses and individuals, for commercial and private locations.

The main activities are Window cleaning for business and individual. CR Window Cleaning & Cleaning performs window cleaning and cleaning of window frames in all types of building, both inside and out. For both private and business from the branch situated in Fredericia, and for customers in Kolding, Vejle, Middelfart and Fredericia.

Key Partners

Employee: the company relies on the employee's turnover.

Customers: the company has customers in Odense as well as in Kolding, Fredericia and Middelfart. Their customers are private economic operators, professional associations, banks, municipal and private institutions, housing associations, etc.

Key Resources

- Employees (28)
- Cleaning equipment
- Vehicles
- IT System
- Knowledge of the market

Value Proposition

- Skilled and competent window washers with many years of experience
- Customize service: flexible times based on the specific customer's needs
- Cheap window cleaning service: the head office is in Odense, but the window cleaners are assigned cleaning tasks, which are closed to their resident locations. In other words, the company would save cost on transportation, thus they can offer a very affordable window cleaning service to their customers.
- Environmentally friendly products: detergents that are inexpensive, environmentally approved and have ecolabels - Svane logo
- Coverage a big area: all of Funen and much of Jutland
- Hight quality service: efficient cleaning by the appropriate use of machines and precision
- Maintain good relationship and keep regular contacts with their customers
- Customer satisfaction is continuing improvement are company's key strategies

Business Case

We help the company to make a customer schedule management system to better manage customer's appointments and more convenient for their employees to use the software. Thus, they can improve their service quality and customer's satisfaction

Use Case

- Use Case 1: Managers - as an administrator - can add or update customer information, date and time and assigns tasks for his employees on the system, he can also edit the input information

- Use Case 2: Employee - as a user - can log in the system daily and check their working schedule. However, the employees cannot change the information in the system

Personas

We've decided to focus our scenario's around the 3 main personas found in the company.

- Users who does not know much about computer and technology

  - can use technology such as smart-phones ect.
  - Relies on tech support for all issues
  - Prefers old methods such as paper slips and notes
- Users who are tech savvy and enjoy using computer
  - Contacts tech support for major issues only
  - Enjoys using online / tech based organization tools
- Users who does not speak Danish very well
  - See less tech savvy persona

Scenario

Use Case 1

Scenario 1

- Administrators log in the system with an username (email) and password, the system checks authorization and verify whether the input information is correct. If the input information is correct, the system allows the administrator to log in. The administrator now can add ***customer information** (customer name, customer address, customer postal code, customer city), ***working schedule** (starting time, closing time) and ***assign tasks** (employees name) to the users. The system updates and displays the input information on the screen

More than 1 scenario

Scenario 2

- Administrators log in the system with an incorrect username (email) and password, the system fails to verify the authorization and denies access to the system. The administrators can try to log in the system again. If the username and password are correct, the system allows the administrators to access the system and update ***customer information, *working schedule** and ***assign tasks** (employees name) to the users. If the administrators put incorrect username or password 3 times, the system will block the access for 10 minutes

Use Case 2

Scenario 1

- Users log in the system with an username (email) and password, the system checks authorization and verify whether the input information is correct. If the input information is correct, the system allows the users to log in the system and see information on the screen.

Scenario 2

- Users log in the system with an incorrect username (email) and password, the system fails to verify the authorization and deny access to the system. Users can try to log in again. The system will allow them to access the system once they input the correct username and password

Actors

- Users
  - Employees

- Admins
  - Office Staff
  - Managers

Stakeholders

Ole, and his co-partner (50/50 ??)
All the employees

Agile Method & Scrum Management

We developed the project based on the Agile Method and use Scrum Management to keep tracking processes of our project

We have had a meeting with our product owner to know about his requirements for the **Schedule Management Application.** Then we made 2 use cases **Manager** and **Employee** based on the product requirements. The product owner decided that we should prioritize the Manager Use Case to develop first

We have daily scrum meetings to discuss about our project

Plan **To Do Tasks** Board for the first Sprint from the beginning of the Sprint
Move the certain tasks to the **In Progress** Board when starting to work on it
Move the In Progress tasks to the **Done** Board when the tasks are finished

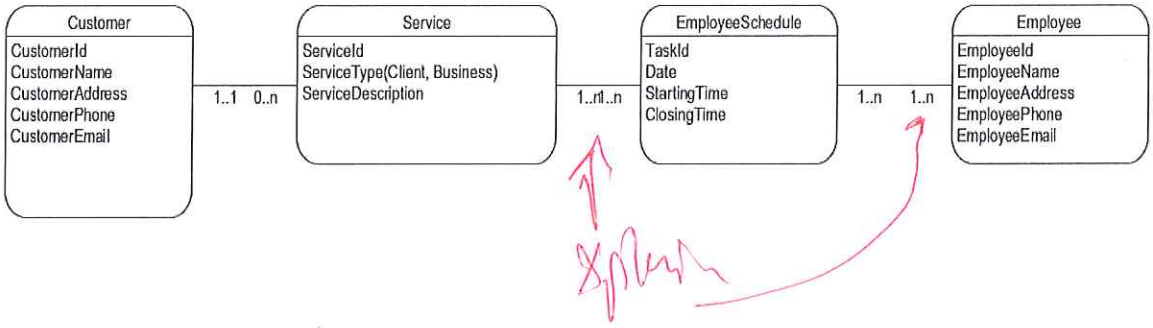
Based on the **Scenario** in the **Manager Use Case**, we had drawn the Domain Model, Object Model, SSD, SD and Class Diagram. After that, we started to write codes based on the **Class Diagram** and the other artifacts. During this agile development process, we will update the artifacts when necessary to make sure that they the artifacts and the codes are connected with each other

We then showed our product owner the finished demo of our Sprint and received feedback from him and improve the product based on his requirement if necessary

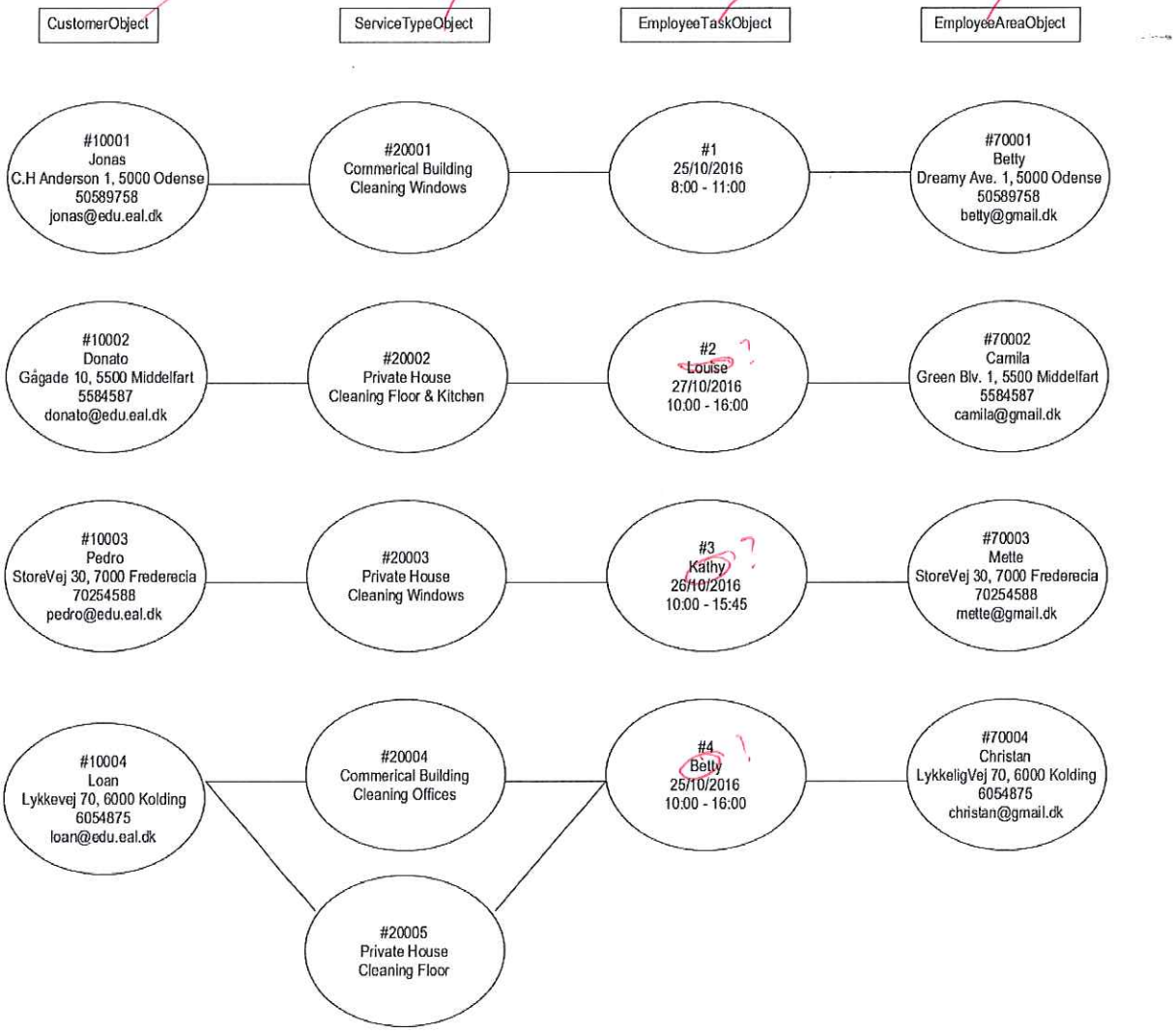
At the end of the Sprint, we will do a **Sprint Retrospective** to review our working processes during the Sprint. We will then continue a new Sprint until we finish our project with a similar approach

Scrum Board

Domain Model

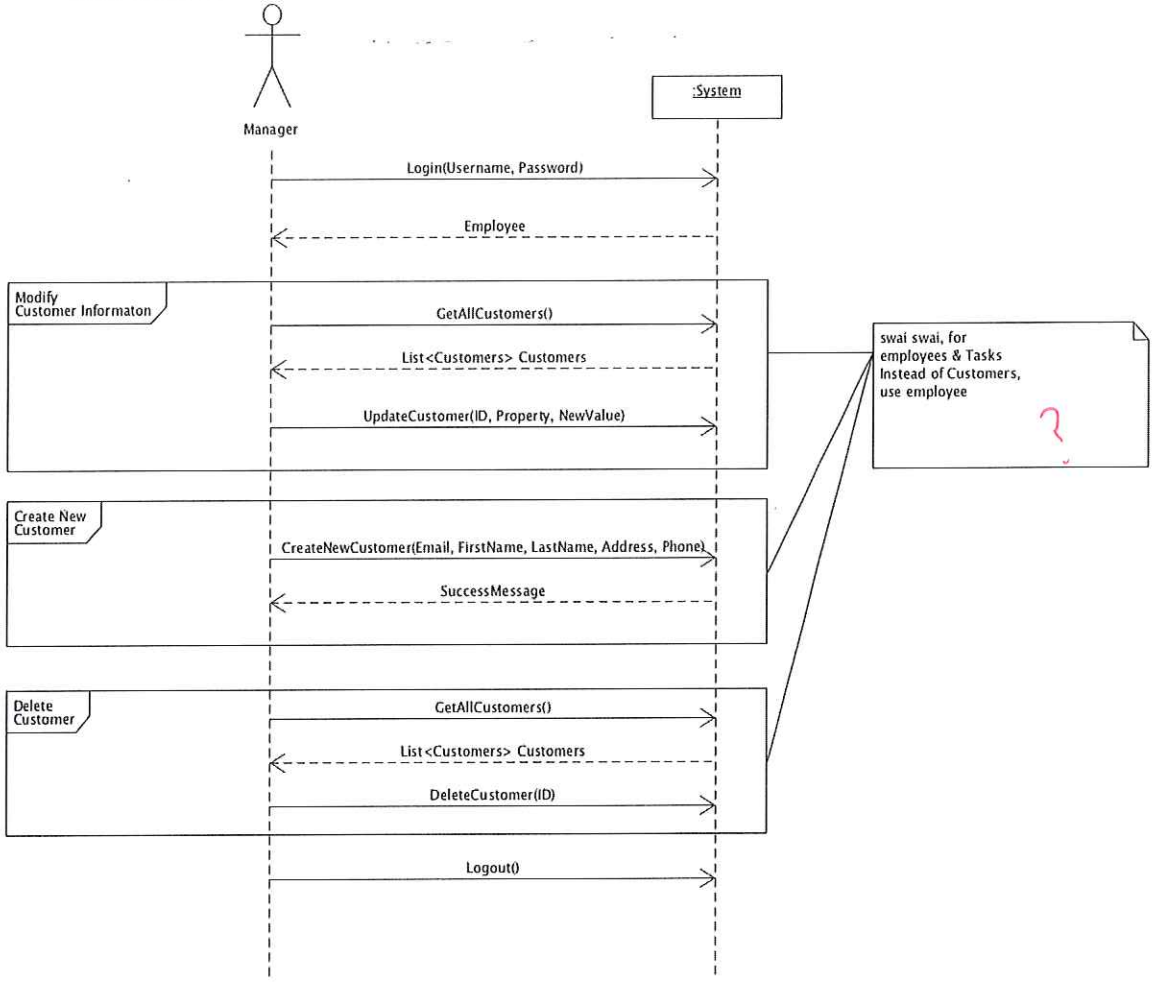


Object Model



System Sequence Diagram

- Use Case 1



System Operation Contract

Operation:	Login(username, password)
Cross References:	Use case 1, SSD – Manager, SSD – Employee
Preconditions:	User not already logged in
Postconditions:	Employee instance created (instance creation)

Operation:	GetAllEmployee()
Cross References:	UC1, SSD Manager
Preconditions:	Logged in as a manager
Postconditions:	List of all Employee

Operation:	UpdateEmployee(id, propery, newvalue)
Cross References:	SSD Manager
Preconditions:	Employee with ID Exists

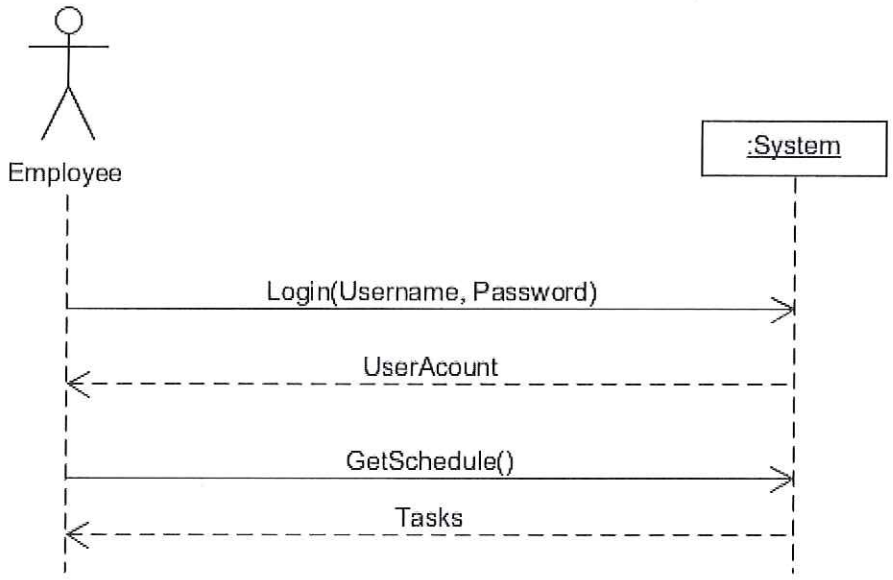
	Logged in as a manager
Postconditions:	Employee Information changed & saved

Operation:	CreateNewEmployee(..)
Cross References:	SSD Manager
Preconditions:	Logged in as a manager
Postconditions:	New Empoyee object (instantiation) Saved

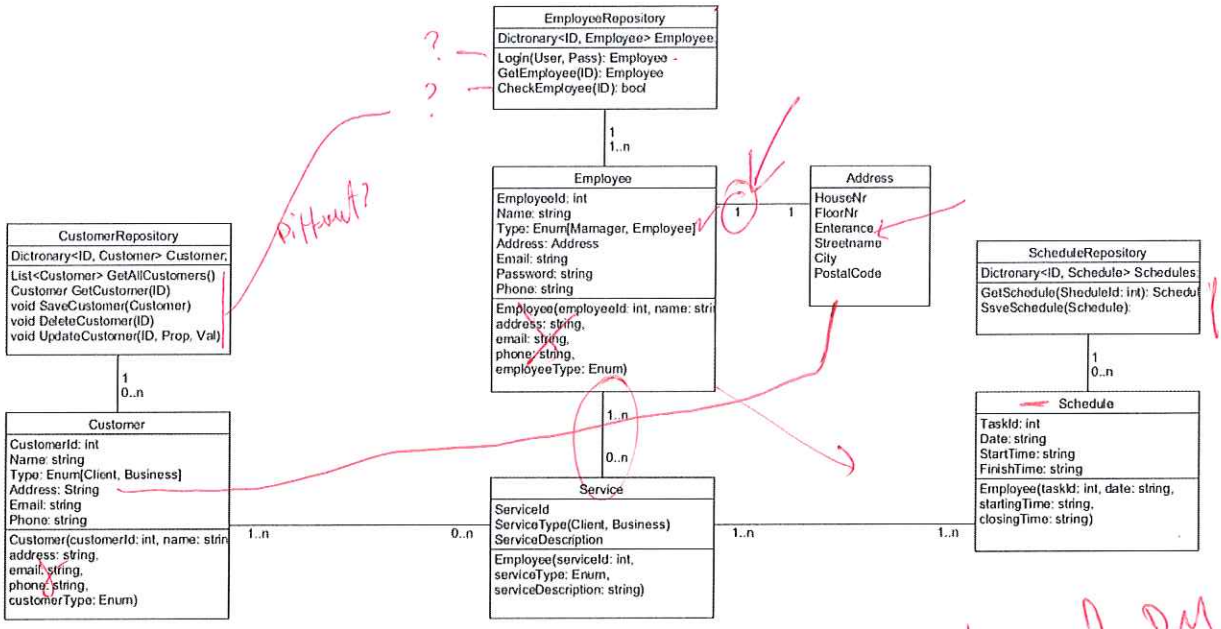
Operation:	DeleteEmployee(ID)
Cross References:	SSD Manager
Preconditions:	Logged in as a manager Employee of ID exists
Postconditions:	Employee of ID no longer exists

System Sequence Diagram

- Use Case 2

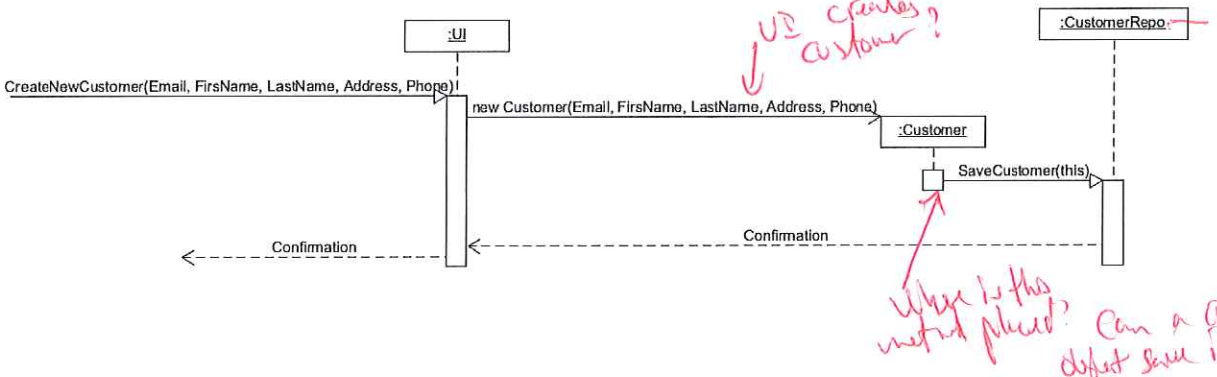


Class Diagram

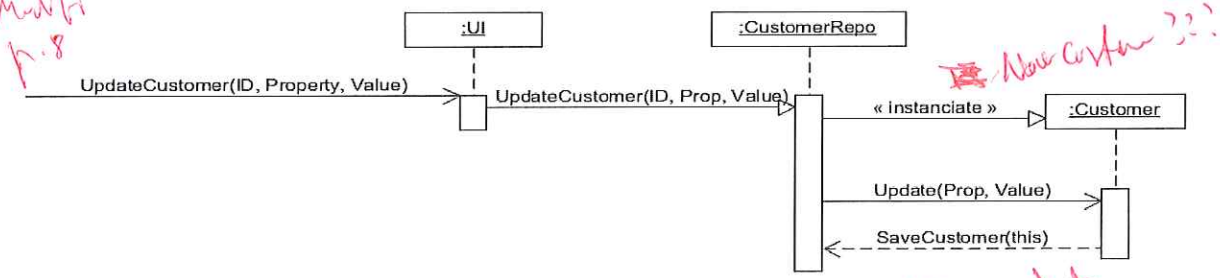


Sequence Diagrams

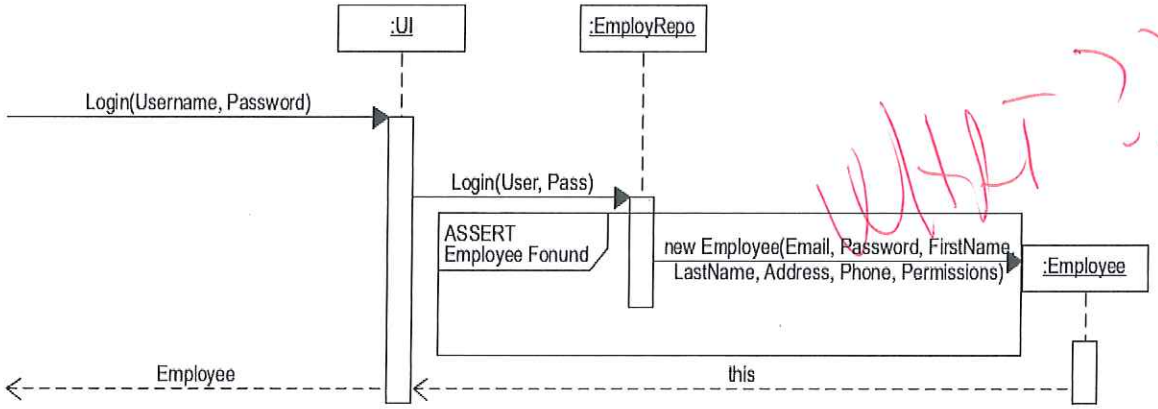
Create New Customers



Update Customer



Login


:UI
:EmployRepo
Login(Username, Password)
Login(User, Pass)
ASSERT
Employee Fonund
new Employee(Email, Password, FirstName,
LastName, Address, Phone, Permissions)
:Employee
Employee
this

Code Samples

Ensuring user is logged in.

```
38          internal void Run() {
39              ProgramRunning = true;
40              while (ProgramRunning) {
41                  while (LoggedIn == null) {
42                      Login();
43                  }
44
45                  string userInput;
46                  if (LoggedIn.Permissions == 2) {
47                      userInput = "5";
48                  } else {
49
```

Login function in Interface

Catch 2 custom exceptions, one to deal with invalid user credentials, and one if no employees are registered.

```
83          private bool Login() {
84              Console.Clear();
85              Console.WriteLine("Login:\n");
86              Console.Write("Username: "); string Username = GetInput();
87              Console.Write("Password: "); string Password = GetPassword();
88
89              try {
90                  LoggedIn = RepoEmp.Login(Username, Password);
91                  Password = null; // Lets take the password out of memory w
92                  return true;
93              } catch (InvalidLoginException) {
94                  Console.WriteLine("\n\nInvalid User Credentials!");
95                  Console.ReadKey();
96                  return false;
97              } catch (NoUserException) {
98                  bool Debug = true;
99                  if (Debug) {
100                     this.Init();
101                 } else {
102                     EmployeeUI EUI = new EmployeeUI();
103                     EUI.CreateEmployee();
104                 }
105                 return false;
106             }
107         }
```

NEVER do while(true) NEVER

Get Password

Method found on stack overflow, edited to not use securestring, because we could not deal with that.

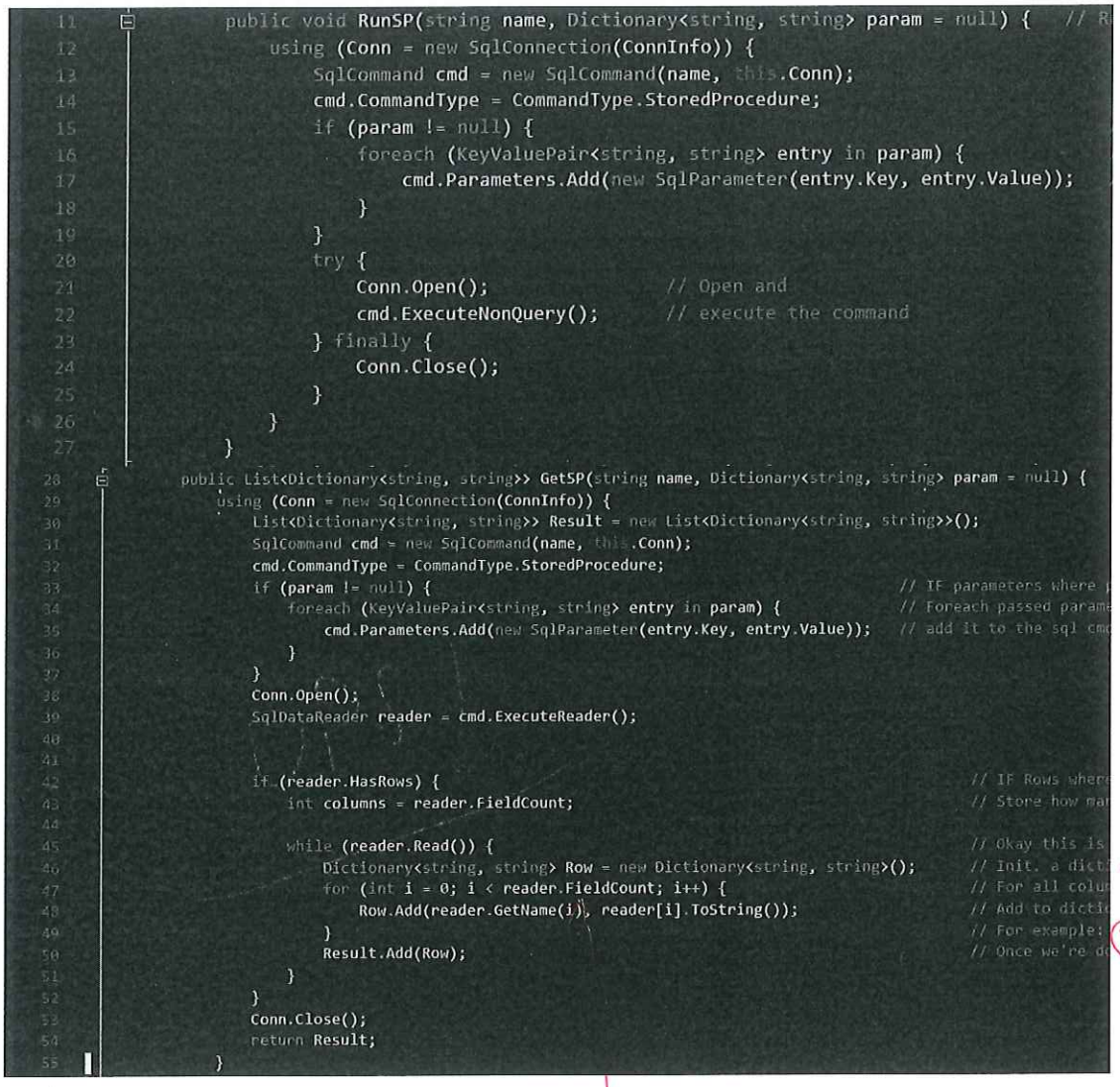
```
171         internal string GetPassword() { // Thanks StackOverflow: http:/
172             string pwd = "";            // Edited to remove securestrin
173             while (true) {
174                 ConsoleKeyInfo i = Console.ReadKey(true);
175                 if (i.Key == ConsoleKey.Enter) {
176                     break;
177                 } else if (i.Key == ConsoleKey.Backspace) {
178                 if (pwd != "") {
179                         pwd = pwd.Substring(0, pwd.Length - 1);
180                         Console.Write("\b \b");
181                     }
182                 } else {
183                     pwd += i.KeyChar;
184                     Console.Write("*");
185                 }
186             }
187             return pwd;
188         }
```

← BAD

UPS!

Database Methods

Our main method of dealing with the database is via the database class.
We use 2 methods, one for query's that don't return, and one that does return.



```
11      public void RunSP(string name, Dictionary<string, string> param = null) {   // R
12          using (Conn = new SqlConnection(ConnInfo)) {
13              SqlCommand cmd = new SqlCommand(name, this.Conn);
14              cmd.CommandType = CommandType.StoredProcedure;
15              if (param != null) {
16                  foreach (KeyValuePair<string, string> entry in param) {
17                      cmd.Parameters.Add(new SqlParameter(entry.Key, entry.Value));
18                  }
19              }
20              try {
21                  Conn.Open();                // Open and
22                  cmd.ExecuteNonQuery();      // execute the command
23              } finally {
24                  Conn.Close();
25              }
26          }
27      }
28      public List<Dictionary<string, string>> GetSP(string name, Dictionary<string, string> param = null) {
29          using (Conn = new SqlConnection(ConnInfo)) {
30              List<Dictionary<string, string>> Result = new List<Dictionary<string, string>>();
31              SqlCommand cmd = new SqlCommand(name, this.Conn);
32              cmd.CommandType = CommandType.StoredProcedure;
33              if (param != null) {                                          // IF parameters where
34                  foreach (KeyValuePair<string, string> entry in param) {   // Foreach passed parame
35                      cmd.Parameters.Add(new SqlParameter(entry.Key, entry.Value));   // add it to the sql cm
36                  }
37              }
38              Conn.Open();
39              SqlDataReader reader = cmd.ExecuteReader();
40
41
42              if (reader.HasRows) {                                         // IF Rows where
43                  int columns = reader.FieldCount;                          // Store how ma
44
45                  while (reader.Read()) {                                   // Okay this is
46                      Dictionary<string, string> Row = new Dictionary<string, string>();   // Init. a dict
47                      for (int i = 0; i < reader.FieldCount; i++) {         // For all colu
48                          Row.Add(reader.GetName(i), reader[i].ToString());   // Add to dicti
49                      }                                                     // For example:
50                      Result.Add(Row);                                      // Once we're d
51                  }
52              }
53              Conn.Close();
54              return Result;
55          }
```

KPI

Customer's Satisfaction:

Send survey questions about Customer's Satisfaction to Customers before implementing the new IT system, ranking the customer satisfaction index from 1(minimum) to 10(maximum) for every question

Send the survey questions to the Customers again after implementing the new IT system, using the same satisfaction index. After that, comparing the new Customer Satisfaction Index with the old one to see if there are any improvement by using the new IT system

Service Hours Per Day, Per Month:

How many hours the company served their customers per day, per month before implementing the new IT system. How many hours the company has served their customers per day, per month after implementing the new IT system.

We then compare the index for service hours between the two periods to see if the total amount of service hours have been increased

Time For Creating Task

Using a user's survey to compare "time for creating task" index before and after implementing the new IT system. Index ranks from (more slowly, no change, faster, much faster)

How does your system influence on the hours the company serve customers per day?

RETROSPECTIVE - SPRINT 1 Wk44/45

What went well?
We follow the sprint management schedules and fulfill the assigned tasks

What went wrong?
We did not strictly follow the Agile working method from the beginning. We spent quite a lot of time on artifacts. It seems more like a combination of waterfall and agile method

What can be done better?
We should strictly follow the Agile working method and allocate time properly for both artifacts and codes

RETROSPECTIVE - SPRINT 2 Wk46/47

What went well?
We work really well with each other as a team. Everyone can speak up their mind about any issues. When we don't completely agree in any solution, we will make the final solution by using democracy vote

We have learnt and improved ourselves during the process of making the group project. It is very efficiently to follow Agile model, we switch back and forth between the artifacts and the codes as it is necessary to update the artifacts to reflect what is presented in codes

What went wrong?
It is the first time we use Scrum Master to control our project, so there is still many rooms for improvement. We did not always prioritize the most important use case in every task we did, we quite often did tasks for both use cases at once instead

What can be done better?
We should have assigned every task to a certain use case. By sticking with the use case, which is prioritized by the product owner we can finish the most important use case first and increase the possibility to meet the deadline

RETROSPECTIVE - SPRINT 3 Wk48/49

What went well?
Everyone is engaged and work actively in the projects

What went wrong?
No major problems went wrong

What can be done better?

There are some certain tasks that take more time than expected. Give some extra time for unexpected situations when planning scrum

PROJECT LOG

✓ PRESPRINT Wk43

Wk43/ Day1
Discuss Working Methods & Tools. We use Trello as a co-working environment, Scrum Management, Materials Storage and Github for Code Sharing

Wk43/ Day 2
Make a group contract about general working rules and team members' roles

Wk43/ Day 3 & 4
Outline main points of Business Model and Business Analysis based on the company website

Wk43/ Day 5 & 6
Pre-draw Domain Model & Object Model based on assumption information from the company website

✓ SPRINT 1 Wk44 – Wk45

Wk44/ Day 1
Plan Sprint 1
Prepared interview questions for the Project Owner and had a meeting with him

Wk44/ Day 2
Business Model & Business Analysis
User Stories, Roles & Scenarios

Wk44/ Day 3
Product Backlog & Sprint Backlog

Wk44/ Day 4
Update Object Model & Domain Model based on the information from the Product Owner

Wk44/ Day 5
System Sequence Diagram
Set up the Getting Real Project in Visual Studio

Wk45/ Day 1
Programming Class & Business Lecture - No Group Project

Wk45/ Day 2
Class Diagram

Wk45/ Day 3
Outline Criteria for feedback

Wk45/ Day 4
Update Artefacts: Domain Model, Object Model, Class Diagram, Sequence Diagram

Wk45/ Day 5
Retrospective for Sprint 1

✓ SPRINT 2 Wk46 – Wk47

Wk46/ Day 1
Learning Database – No Group Meeting

Wk46/ Day 2
Update System Sequence Diagram

Wk46/ Day 3
International Student Event – No Group Meeting

Wk46/ Day 4
Update Sequence Diagram for Employee Login System
Create Codes for Employee Login System
Test Employee Login

Wk46/ Day 5
No Group Meeting

Wk47/ Day 1:
Learning Database: No Group Meeting

Wk47/ Day 2:
Prepare Project Presentation

Wk47/ Day 3:
Write Codes: Employee Class
Testing Codes: Employee Class: Add, Update, Delete
System Operation Contracts for Employee

Wk47/ Day 4:
Write Codes: Customer Class
Testing Codes: Customer Class: Add, Update, Delete

Wk47/ Day 5
Retrospective for Sprint 2

✓ SPRINT 3 Wk48 – Wk49

Wk48/ Day 1
Plan Sprint 3
Learning Programming

Wk48/ Day 2
Create Database for Employee Class

Wk48/ Day 3
Create Database for Customer Class

Wk48/ Day 4
Create Stored Procedures for Update Customer, Save Customer, Delete Customer, Update Employee, Save Employee, Delete Employee

Wk48/ Day 5
Create Stored Procedures for Update Customer, Save Customer, Delete Customer

Wk49/ Day 1
Learning Programming

Wk49/ Day 2
Connect Customer Class with Database

Wk49/ Day 3
Connect Employee Class with Database

Wk49/ Day 4
Create Customer Menu

Wk49/ Day 5
Retrospective for Sprint 3

✓ SPRINT 4 Wk50 – Wk51

Wk50/ Day 1
Planning Sprint 4
Create KPI

Wk50/ Day 2
Create Customer and Employee UI

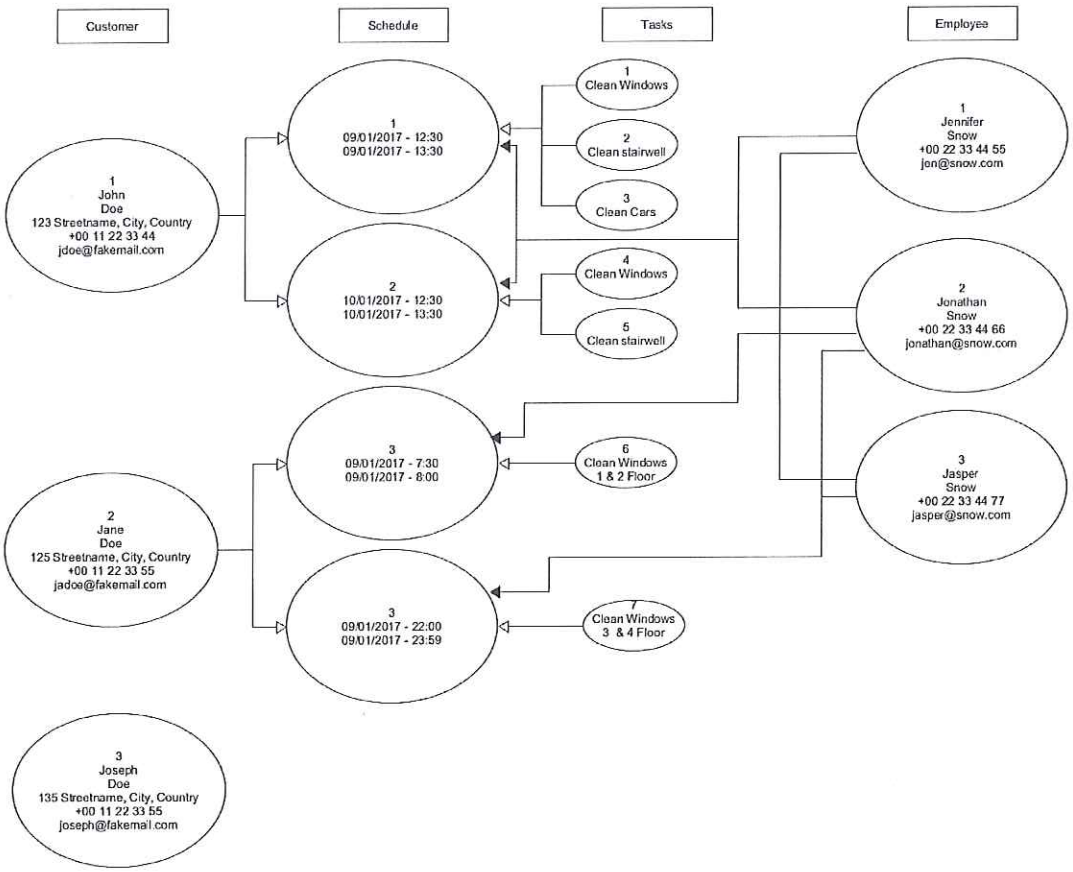
Wk50/ Day 3
Create Login UI
Create Table & Stored Procedure for Schedule

Wk50/ Day 4
Schedule Repository and Schedule Menu

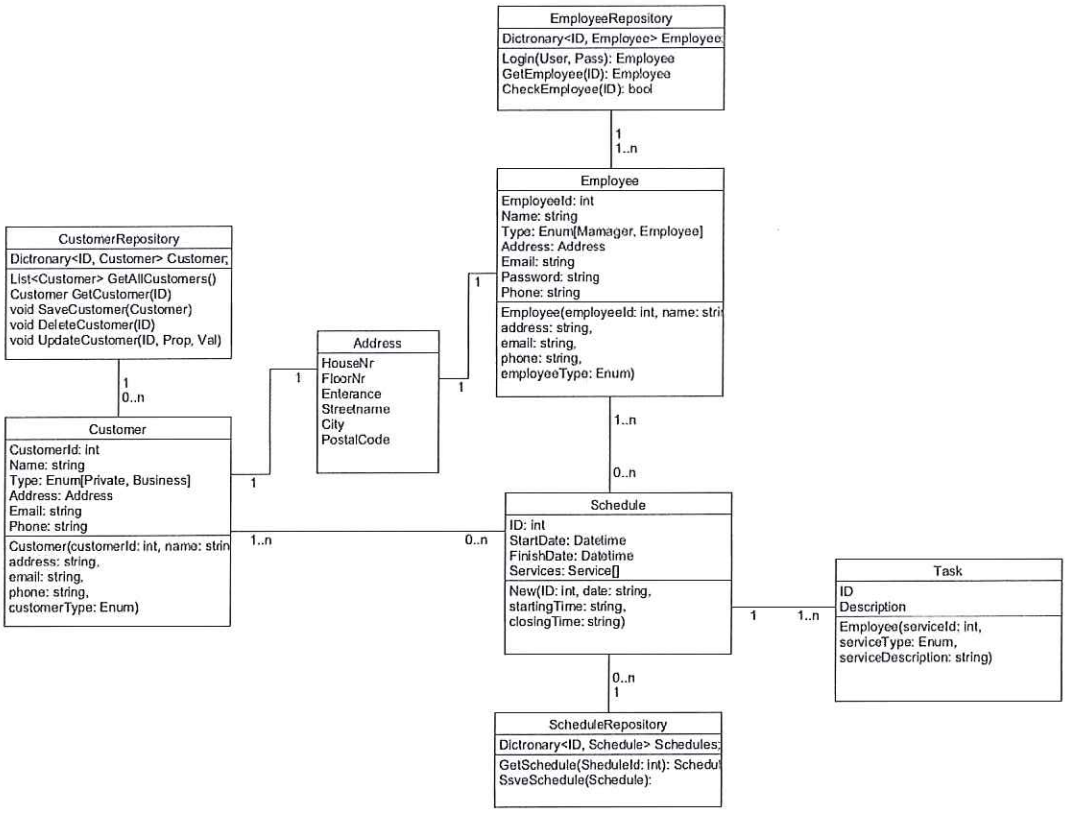
Wk50/ Day 5
No Group Meeting

Wk51/ Day 1
Update Report

Object Model

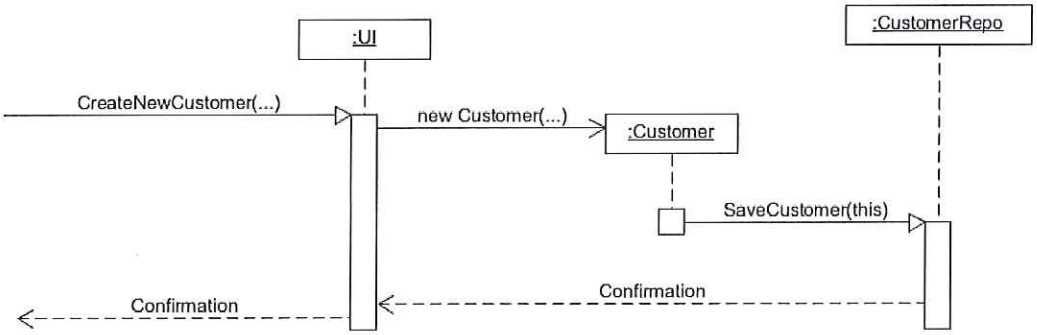


Class Diagram

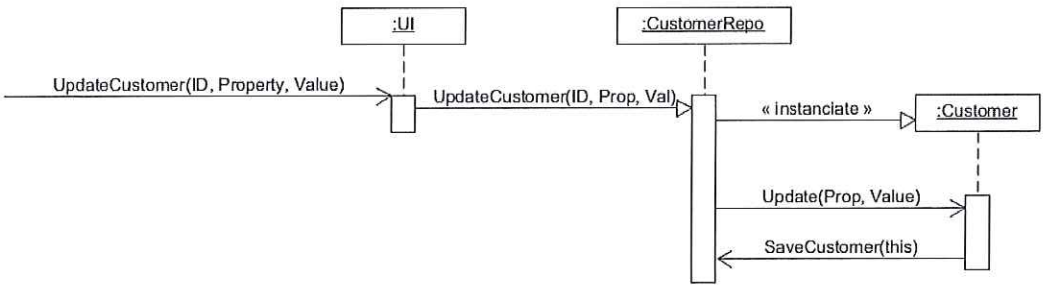


Sequence Diagrams

Create New Customer



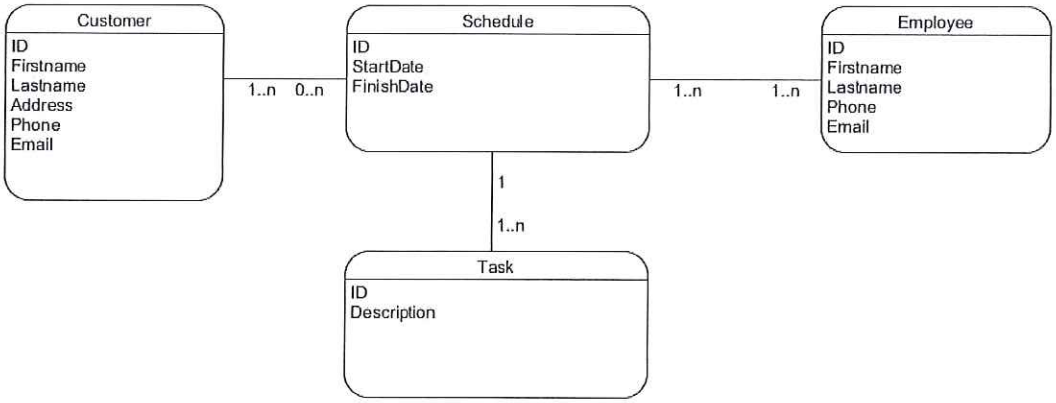
Update Customer



Stakeholders

Ole, and his co-partner (50/50 ??)
All the employees

Domain Model



Very Simple System.

Too little details in work flow

inconsistency between models

and model-code.

No DB design or quality insurance
ie normalisation

No test

No code quality insurance.

But many things
actually work good.