Project Report

Arturs Silins - 315226
Bhupas Gautam - 306411
Maximillian Marius Wallin - 315268
Ondrej Klimek - 315255
Siddhartha Jonathan Grasse - 315278

Supervisors:
Ib Havn
Jørn Martin Hajek

Software Technology Engineering
2nd Semester

Horsens, 2022

Table of Contents

List of figures and tables

Abstract

A Danish restaurant chain has requested to create an application that will be used to reserve tables and order food. Using Scrum and Unified Process, this application is implemented in Java code with MVVM architecture and RMI technology. JavaFX is used to create the graphical user interface, and the database is written in SQL. Astah was used to create the relevant diagrams.

Introduction

The client, Bob, is the head of a Danish restaurant chain, looking for a program that combines table booking and food ordering and digitalizes them to require less manual work. He wants a product that enables guests to book tables and pre-order food themselves while also being able to submit special requests, for example, specific diets or allergies.

Additionally, Bob wants to have a comment section where guests can express their wishes, for instance, if the occasion is a birthday or an anniversary. Guests should be able to decide if they will dine in or order their food for takeaway. If they choose to dine in, they can reserve a table. Also, they can choose a specific number of chairs and have baby chairs added if needed.

Bob would like the system language to be in English by default and have the option of Danish as the second language to accommodate a wide variety of guests. If multiple languages are not possible in the system, the client is satisfied with the system being in English.

The project's focus is to create a restaurant management system that is fast and easy to use. The guests and the staff should not have to spend too much time on the screen.

Unified Process and SCRUM have been used as a framework for this project. The Unified Process is based on the enlargement and refinement of a system having multiple iterations with cyclic feedback.(TheTeche, 2022) While Scrum is a framework that helps teams and organizations address complex problems step by step while delivering products. It is used by teams that consist of a Product Owner, Scrum Master, and Developers.(Digite, 2022)

Analysis

The analysis of the customer's needs is shown in this chapter. Relevant diagrams will be shown with short explanations. In this chapter, customers' requirements, use case diagram, domain model, and sequence diagram, which were all created using Astah, will be discussed. More diagrams are available in appendix B.

Actors

Different features of the applications are divided between the different actors; therefore, it is crucial to know the role of the different actors.

Guest: The guest is the most critical actor; therefore, the application is built around what is most important for the guest. The guest can access the following features; Create an account, edit an account, log in, make a reservation, order, and take out.

Staff: Staff are the people who work at the restaurant, and they have access to all the features of guest and the following extra features; see order, manage orders, See a reservation and Manage reservations.

Manager: The manager has the most influential role in the application, and they can access all the features available to staff and the following extra features; Manage table, manage menu items and manage users.

Requirements

When analyzing requirements, special attention was made to the SMART criteria. Requirements are grouped by actors and written using the MoSCoW method.(Product Plan, 2022) As the significance of these requirements can change during the project, they have not been ordered by importance in this section.

Functional Requirements:

1. As staff I want to see the table bookings, so that I know where the guests are seated.
2. As staff I want to see the orders, so that I know what needs to be cooked.
3. As staff I want to see comments, so that I can take them into consideration.
4. As staff I want to see special requests, so that I accommodate them appropriately.
5. As staff I want to see ordered food, so that the guests can receive their meal.
6. As staff I want to edit orders, special requests, and table bookings, so that I can correct mistakes or cancel reservations.
7. As guest I want to see the available tables, so that I can book them.
8. As a guest I want to book a table, so that I have secured seating when I arrive.
9. As staff I want to book a table, so that I can seat my guests.
10. As a guest I want to pre-order meals, so that the staff can prepare them.
11. As staff I want to order meals, so that the kitchen can prepare them.
12. As a guest I want to make comments, so that I can get my special wishes fulfilled.
13. As a guest I want to make special requests, so that I don't experience any difficulties.
14. As staff I want to make special requests, so that I can fulfil the customers dietary needs.
15. As a guest I want to create an account, so that I have my information saved for future visits.
16. As a manager I want to unregister staff from the system, so that unemployed staff cannot access the system.
17. As a manager I want to remove/add dishes to/from the menu, so that I can offer the food that I have in stock.

Quality Requirements:

18. As a user I want the system to be intuitive to make the booking process last less than 2 minutes.
19. As a manager I want the system language to be in english/danish to accommodate a wide variety of guests.

Use case

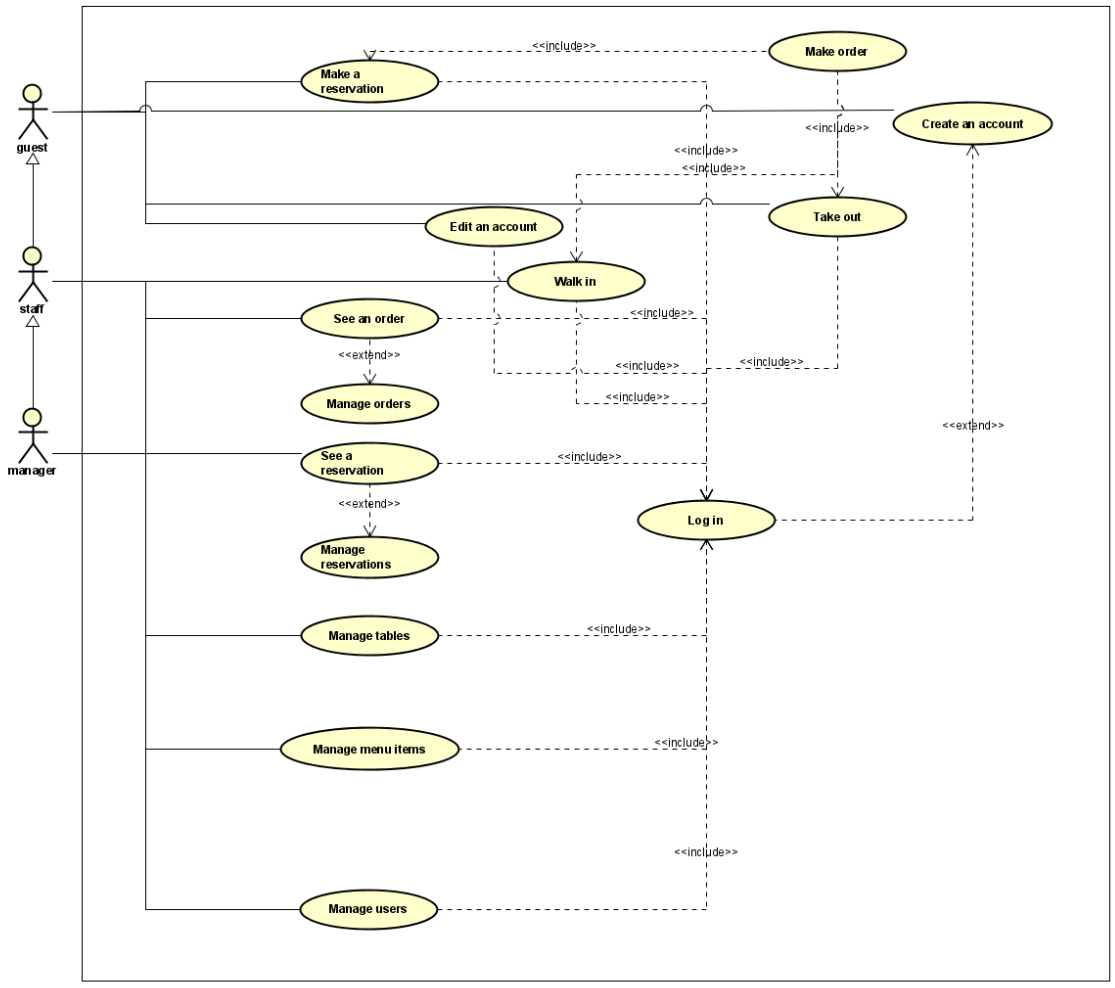


Figure 1 Use cases

Create an account: To use the application, the user must have an account and must be logged in; the user creates an account by filling in options such as name, phone number, and password.

Edit an account: The user can edit the name, phone number, and password.

Login: The user must log in using a phone number and password before using the application.

Take out: the guest can decide to order take out and get the food at the front desk. To create a takeout order user must be logged in.

Make a reservation: The User makes a reservation by filling out the necessary information like name, phone number, date and time.

Walk in: Staff can register a walk in when guests enter the restaurant without a reservation.

Make order: Users can order food, and beverages, make comments and communicate dietary needs.

See an order: Staff can view all the orders.

Manage orders: Staff can edit/delete an order.

See a reservation: Staff can view all the reservations.

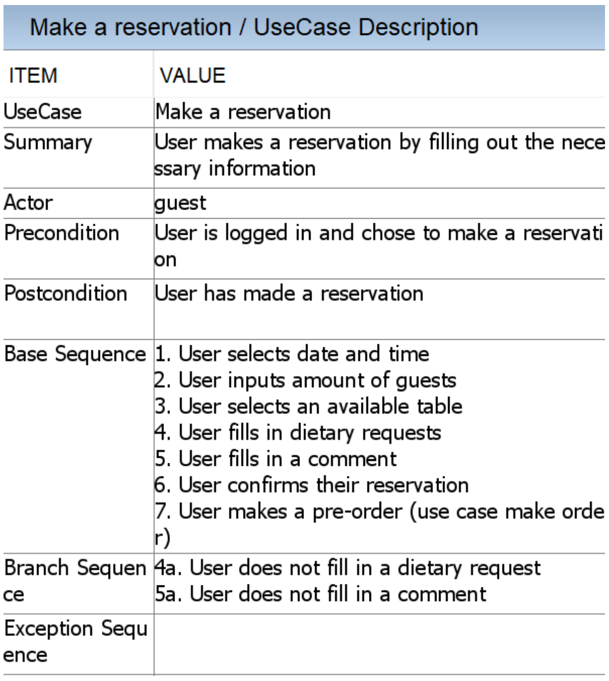
Manage reservations: Staff can edit/delete a reservation.

Manage tables: Manager can add/edit/delete tables.

Manage menu items: The manager can add/edit/delete menu items.

Manage users: The manager can change the user's position or delete their account.

All these use cases are further explained in the use case descriptions, which provide further details on every step of each use case. The use case description of making a reservation can be seen below. The remaining use case descriptions can be found in appendix B.



Make a reservation / UseCase Description

ITEM	VALUE
UseCase	Make a reservation
Summary	User makes a reservation by filling out the nece ssary information
Actor	guest
Precondition	User is logged in and chose to make a reservati on
Postcondition	User has made a reservation
Base Sequence	1. User selects date and time 2. User inputs amount of guests 3. User selects an available table 4. User fills in dietary requests 5. User fills in a comment 6. User confirms their reservation 7. User makes a pre-order (use case make orde r)
Branch Sequen ce	4a. User does not fill in a dietary request 5a. User does not fill in a comment
Exception Sequ ence	

Figure 2 UseCase Description

Domain Model

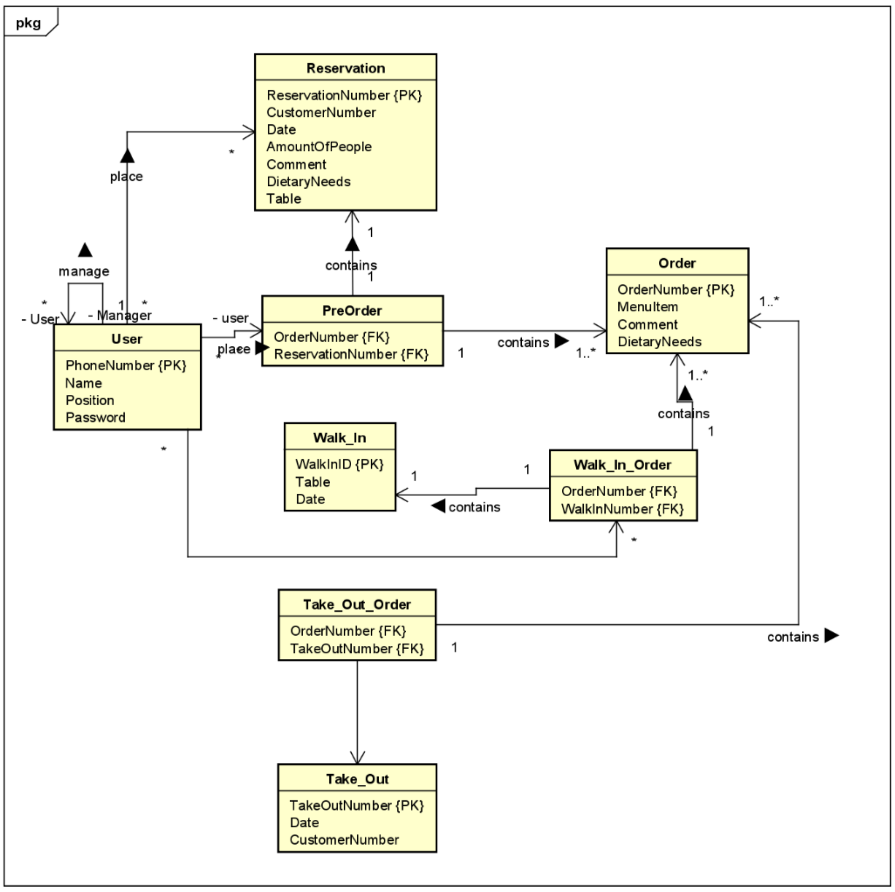


Figure 3 Domain Model

The domain model shows the data of the system and the data's relationships, the problem domain concepts, and attributes with their relationships.
Besides phone number, name, and password, the user has a position that can be either guest, staff, or manager. A user with the manager position manages other users.
A reservation can be made on its own but can also be part of a Pre-order and an Order. Additionally, an Order is used as part of the Walk_In_Order and a Take_Out_Order, with Walk_In and Take_Out, respectively.
With the Analysis part of the project explained, the design can be explained in the next chapter.

Design

In this chapter, the project's architecture, design patterns, and applied technologies will be explained, along with the relevant diagrams from Astah. The database was created using Postgres, and the code was written in java.

We had a choice between RMI or sockets for the network communication. The development team decided to use RMI because it is simpler to implement even though it is less customizable. MVVM was used as the primary architectural pattern for the project. The GUI was sketched on paper.

Architecture

MVVM was used as the primary architectural pattern for this project.(Martin, 2022) The model was distributed among server and client; Network was used to connect them. The model's right side shows "shared" containing util, networking, and the necessary transfer objects. To store the information, a database has been used. The left side shows the client and the server, both containing networking and model, with the client also containing core and view.

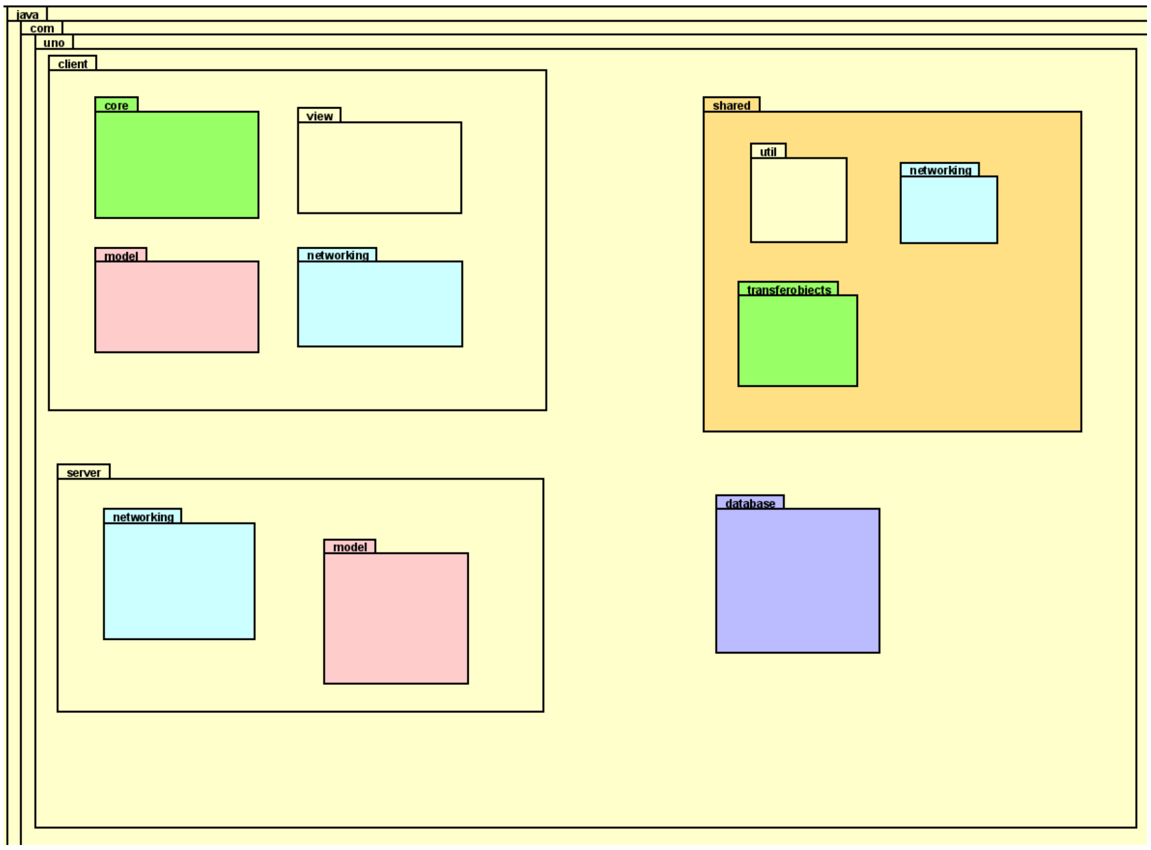


Figure 4 MVVM

The view contains the ViewModel as illustrated in Figure 5.

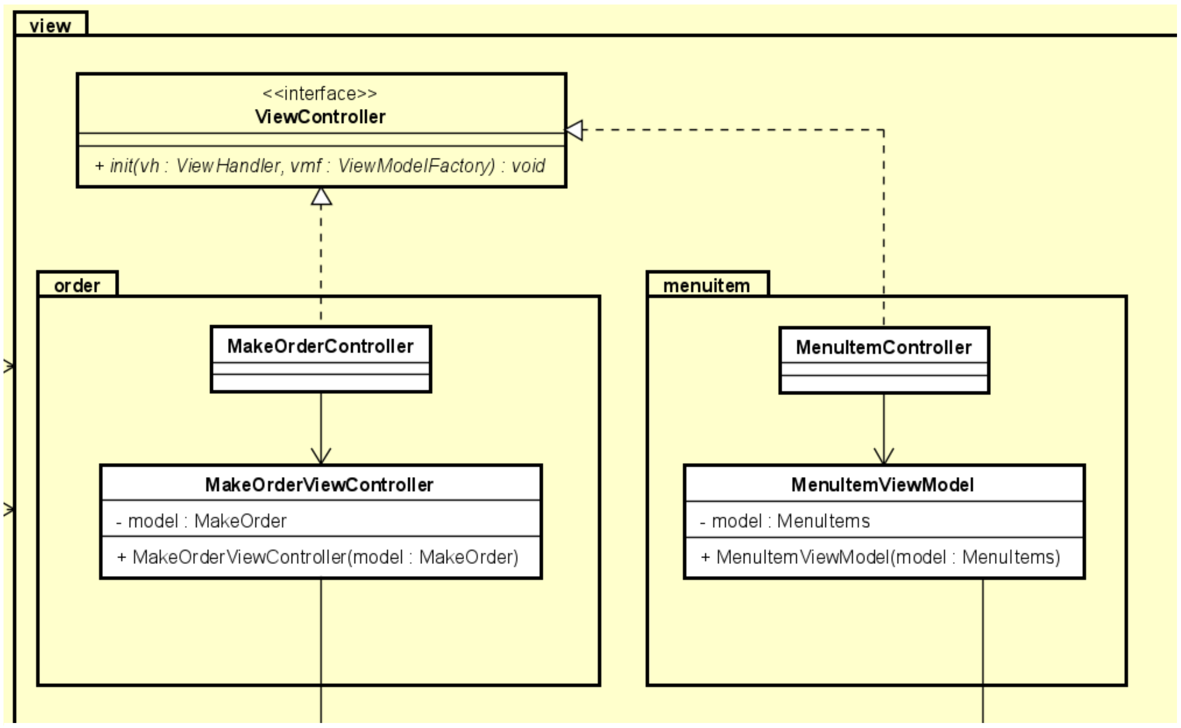


Figure 5 Viewmodel

Technologies

Our product had mentioned that we are required to make a client-server connection in this project. Developers had to decide between using RMI or Sockets for the networking.

Sockets

Usually, sockets are used for low-level network communication, making a simple call that interfaces with IP. Sockets are very customizable and can be used with other programming languages. Sockets require implementing their Threads, making code large and complex, thus making it harder to find mistakes. (Vaidya, 2022)

RMI

RMI is used for networking between Java applications. It is less customizable than Sockets, but RMI is simple to code because it handles most of the networking automatically.(Javatpoint, 2022) Our developers decided to use RMI instead of Sockets because it is easier to program and debug.

GUI

The GUI design was first sketched on paper before being created in Scenebuilder. The examples below show the Menu and Order views.

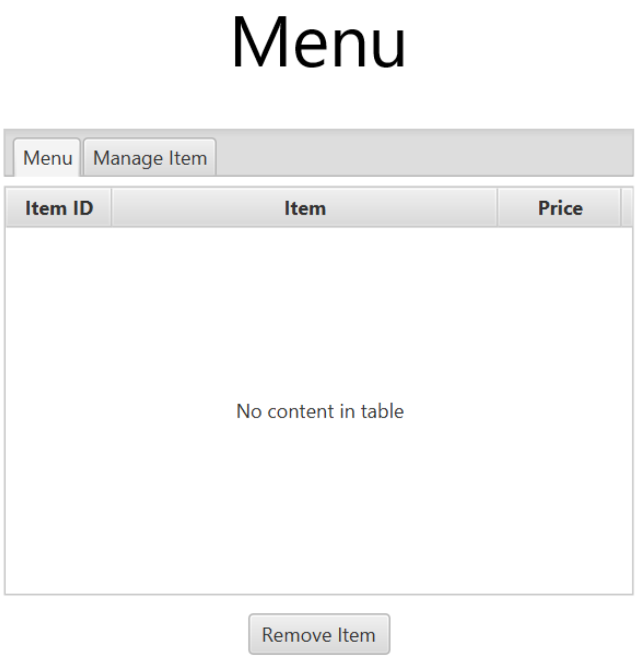


Figure 6 Menu GUI

As one of the requirements is to finish an order in under 2 minutes, the main focus of the GUI is to make it fast and easy to navigate and avoid unnecessary information and clutter.

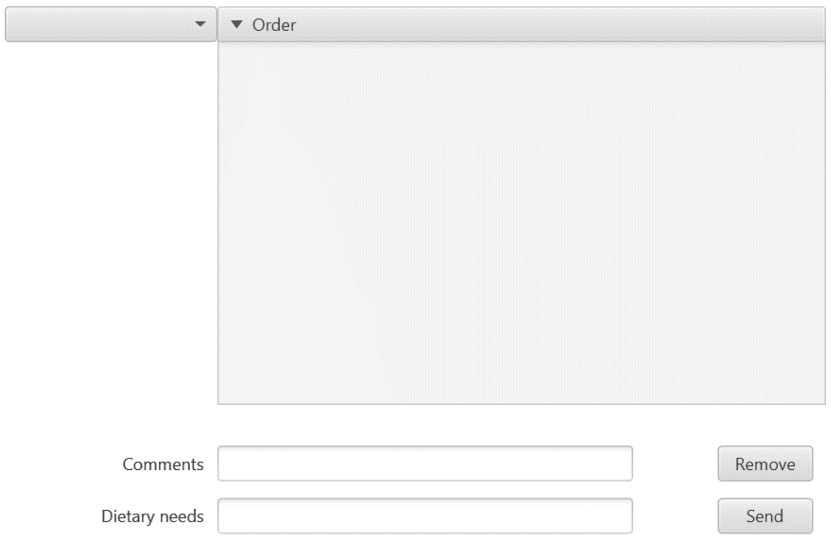


Figure 7 Order GUI

Design Pattern

The SOLID principle has been used to create more maintainable and flexible software, which will help to expand the system in the future if the client decides to do so. (Millington, 2022)

The database connection class is a singleton, and it establishes a connection to the database whenever needed to get an instance of it. It holds the information of the URL, the user, and the password.

Synchronization has been used so that it does not create multiple instances and fails.

The state pattern is used in the database handler to connect multiple individual methods with one interface. The states include AccountState, BookTableState, and ReservationState. The client factory creates the client, the ModelFactory creates all the model classes, and the ViewModelFactory creates the view models, which are the connections between the view and the model. Additionally, the view handler starts the individual views. The Observer pattern is used to notifications of new events, such as a new order or a new table booking.

All the classes have been written with encapsulation abstraction in mind, using interfaces whenever it helped simplify the code.

Class Diagram

In this chapter, essential classes will be explained in detail. More classes can be found in Appendix B.

MenuItem

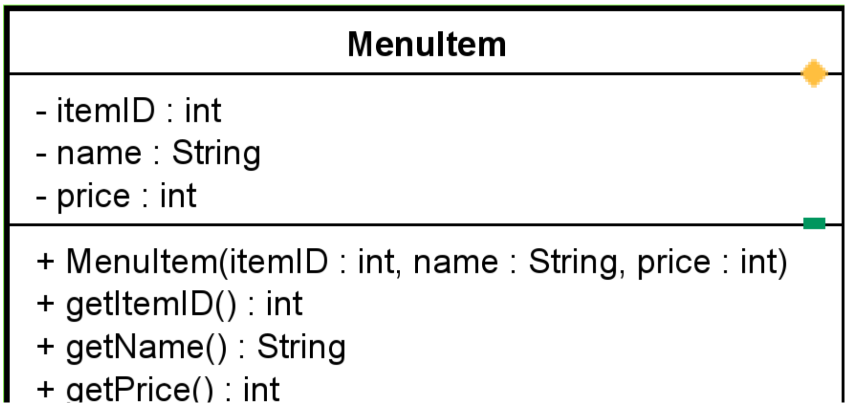


Figure 8 MenuItem Class Diagram

`itemID` is a number that indicates the specific menu item. `name` is the name of the actual item, while the `price` is its price. These details will be used throughout the system, for example, in the `Order transferobject` shown below.

Order

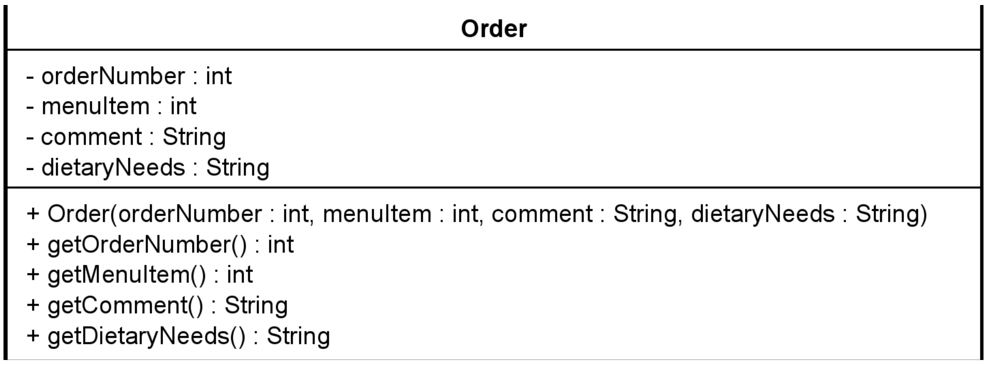


Figure 9 Order Class Diagram

The `orderNumber` indicates the number of the actual order, and the `menuItem` is the item ordered, the `comment` is the comment that the guest leaves for the staff if there is a special occasion, and the `dietaryNeeds` are, for example, an intolerance towards a specific ingredient.

MakeOrder

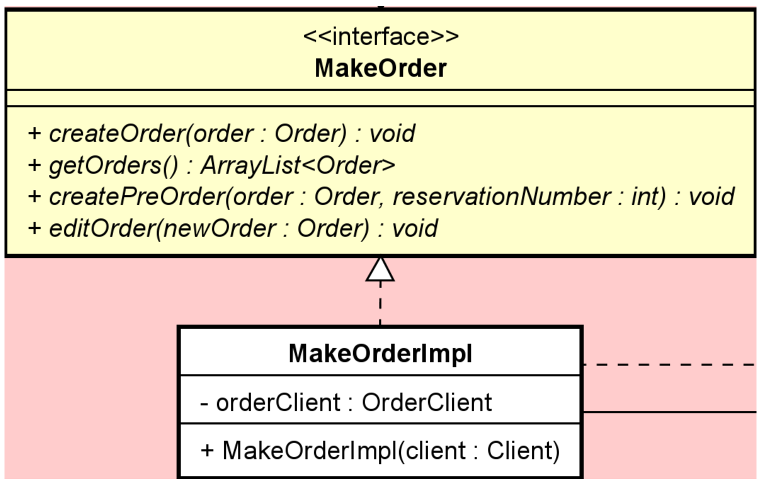

<<interface>>
MakeOrder
+ createOrder(order : Order) : void
+ getOrders() : ArrayList<Order>
+ createPreOrder(order : Order, reservationNumber : int) : void
+ editOrder(newOrder : Order) : void
MakeOrderImpl
- orderClient : OrderClient
+ MakeOrderImpl(client : Client)

Figure 10 MakeOrder Class Diagram

`MakeOrderImpl` inherits from the `MakeOrder` Interface, which contains a `createOrder` constructor that accepts a specific order as an argument. This order can also be edited with the `editOrder()` method. This method will be explained in more detail in the sequence diagram below.

Sequence Diagram

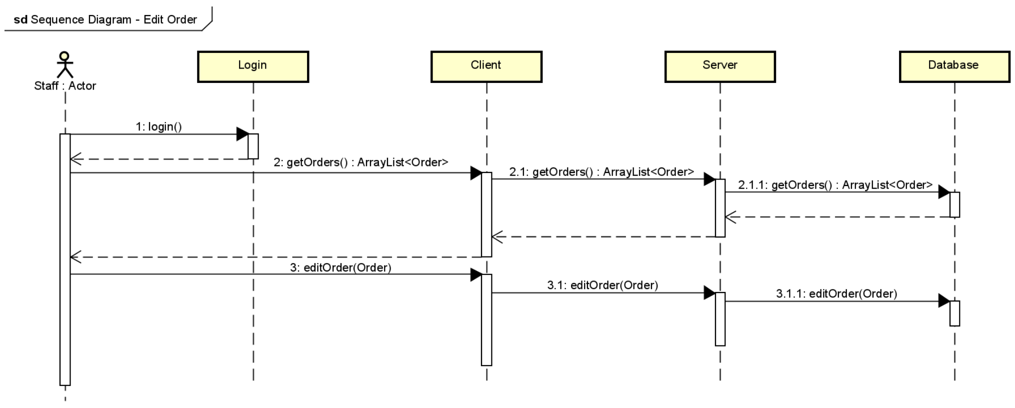

sd Sequence Diagram - Edit Order
Staff : Actor
Login
Client
Server
Database
1: login()
2: getOrders() : ArrayList<Order>
2.1: getOrders() : ArrayList<Order>
2.1.1: getOrders() : ArrayList<Order>
3: editOrder(Order)
3.1: editOrder(Order)
3.1.1: editOrder(Order)

Figure 11 Sequence Diagram

Figure 12 explains the interaction of the staff editing an order. First, the staff logs into the system through the login, then sending the `getOrders()` method to the client, which sends the `getOrders()` method to the server, which sends the `getOrders()` method to the database. The database returns the information to the server, which returns the information to the client, which sends the information to the staff.

With this information, the staff can select a specific order and send the `editOrder()` method referencing that order to the client, which sends the `editOrder()` method to the server, which sends the `editOrder()` method to the database. The model of the database will be illustrated in the next chapter.

Database Model

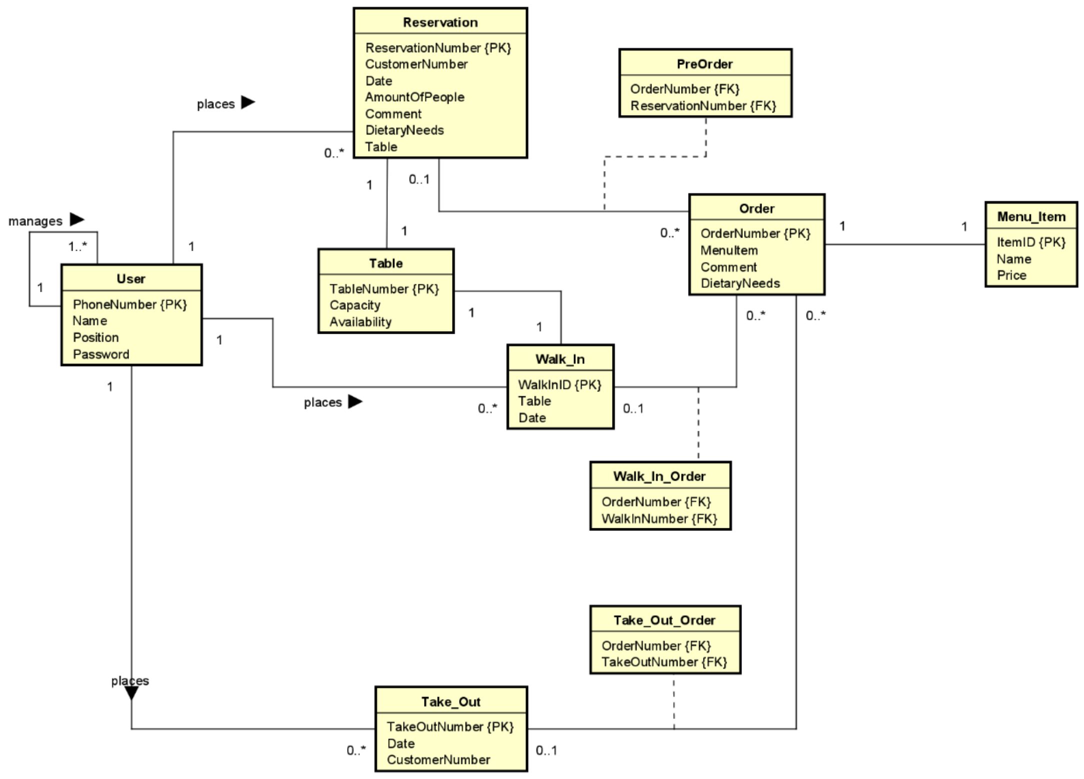


Figure 12 Database Model

The User Table contains a Name, a position which can be "Staff", "Guest", or "Manager", a password that is being hashed, and a phone number as a primary key. A user with the " Manager " position manages all the other users. Any user can place a reservation, which is saved in the "Reservation" table. This table contains a customer number, which is the phone number of a user with the position of "Guest", the date of the reservation, the number of people that are going to attend, a comment for the staff if the reservation is for a special occasion, dietary needs like intolerance to specific ingredients, a specific table that is chosen from the "Table" table, and a reservation number as the primary key. If the guest makes an order, this reservation can be used with that order to create a pre-order, which is saved in the "PreOrder" table.

The" Order" table contains a Menu Item that is being taken from the "Menu_Item" table, comments and dietary needs like the reservation table, and an order number as a primary key. The "Walk_In_Order" and "Take_Out_Order" tables work on the same principle. With the design finished, the next step of the process is the implementation.

Implementation

This chapter will discuss how the design is implemented into code, along with explanations.

Order

```
public class Order implements Serializable {

    private int orderNumber;
    private int menuItem;
    private String comment;
    private String dietaryNeeds;

    public Order(int orderNumber, int menuItem, String comment, String dietaryNeeds) {
        this.orderNumber = orderNumber;
        this.menuItem = menuItem;
        this.comment = comment;
        this.dietaryNeeds = dietaryNeeds;
    }

    public int getOrderNumber() {
        return orderNumber;
    }

    public int getMenuItem() {
        return menuItem;
    }

    public String getComment() {
        return comment;
    }

    public String getDietaryNeeds() {
        return dietaryNeeds;
    }
```

The `Order transferobject` contains the order number, the chosen menu item, a comment, the dietary needs, and the necessary get methods. This object gets sent to the `OrderClientImpl`.

OrderClientImpl

```
public class OrderClientImpl implements OrderClient{

    private OrderServer server;


    public OrderClientImpl(Server server) {
        try {
            this.server = server.getOrderServer();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }


    public void createOrder(Order order) {
        try {
            server.createOrder(order);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    @Override     public ArrayList<Order> getOrders() {
        ArrayList<Order> list = new ArrayList<>();
        try {
            list = server.getOrders();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        return list;
    }


    @Override     public void createPreOrder(Order order, Reservation reservation){
        try {
            server.createPreOrder(order, reservation);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    @Override     public void editOrder(Order newOrder) {
        try {
            server.editOrder(newOrder);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
```

The order is sent to the order server as a new order using the `editOrder`() method.

OrderServerImpl

```
public class OrderServerImpl implements OrderServer {

  private OrderHandler handler;

    public OrderServerImpl(OrderHandler handler) throws RemoteException {
      UnicastRemoteObject.exportObject(this, 0);
    this.handler = handler;
  }


  @Override

   public void createOrder(Order order) throws RemoteException {
    handler.createOrder(order);
  }

  @Override

   public ArrayList<Order> getOrders() throws RemoteException {
    return handler.getOrders();
  }

  @Override

   public void createPreOrder(Order order, Reservation reservation) throws
RemoteException {
    handler.createPreOrder(order, reservation);
  }

  @Override

   public void editOrder(Order newOrder) throws RemoteException{
    handler.editOrder(newOrder);
  }
```

The new order is sent to the order handler using the `editOrder`() method.

OrderHandlerImpl

```
public class OrderHandlerImpl implements OrderHandler{

  private Database database;

  public OrderHandlerImpl(Database database) {
    this.database = database;
  }

  @Override

   public void createOrder(Order order) {
    database.createOrder(order);
  }

  @Override

   public ArrayList<Order> getOrders() {
    return null;
  }

  @Override

   public void createPreOrder(Order order, Reservation reservation) {
    database.createPreOrder(order.getOrderNumber(),
reservation.getReservationNumber());
  }

  @Override   public void editOrder(Order newOrder) {
    database.editOrder(newOrder);
  }
}
```

The new order is sent to the database using the `editOrder()` method

DatabaseImpl

```java
public class DatabaseImpl implements Database {

  private DatabaseHandler currentState;
  private DatabaseHandler handler;

  public DatabaseImpl() {
    handler = new OrderState();
  }

  @Override

   public ArrayList<MenuItem> getMenuItems() {
    setState(new MenuItemsState());
    return (ArrayList<MenuItem>) currentState.getListFromDatabase();
  }

  @Override

  public void createOrder(Order order) {
    setState(new OrderState());
    currentState.sendToDatabase(order);
  }

  @Override

 public void bookTable(Table table)
  {

  }

  @Override

public void editTableBooking(Table oldBooking, Table newBooking) {

  }

  @Override

 public void editOrder(Order newOrder) {
    setState(new EditOrderState());
  }


  @Override

 public void createPreOrder(int orderNumber, int reservationNumber) {

    PreOrder preOrder = new PreOrder(orderNumber, reservationNumber);

    setState(new PreOrderState());
    currentState.sendToDatabase(preOrder);
  }

  @Override

 public void createAccount(Account account) {
    setState(new AccountState());
    currentState.sendToDatabase(account);
  }
  @Override   public void createReservation(Reservation reservation) {
    setState(new ReservationState());
    currentState.sendToDatabase(reservation);
  }

  @Override

public Account login(String phoneNumber, String password) {
    return null;
  }

  private void setState(DatabaseHandler state) {
    this.currentState = state;
  }
}
```

The new order is used in the `editOrder()` method, and the current state is set to the `EditOrderState`

`EditOrderState`

```
public class EditOrderState implements DatabaseHandler {

  public void sendToDatabase(Object object) {
    Order temp = (Order) object;

    try(Connection connection = DatabaseConnection.getInstance().getConnection()) {
      PreparedStatement statement =
          connection.prepareStatement("UPDATE Order SET MenuItem = ?, comment = ?,
dietaryneeds = ? WHERE ordernumber = ?");
      statement.setInt(1, temp.getMenuItem());
      statement.setString(2, temp.getComment());
      statement.setString(3, temp.getDietaryNeeds());
      statement.setInt(4, temp.getOrderNumber());
      statement.executeUpdate();
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }

  @Override

  public Object getFromDatabase() {
    return null;
  }

  @Override

 public void sendListToDatabase(Object list) {
  }

  @Override

public Object getListFromDatabase() {
    return null;
  }
}
```

Test

To avoid finding out about the bugs at the end of the program, it was decided to test each item using unit-testing once it was completed. When the test succeeds, that issue could be moved to the "Done" section in Jira. The test cases were made from the use case descriptions, and it was tested using the GUI to make sure that it worked as intended. Attention was paid special attention to ensure all cases were covered, along with the sunny scenario and exception sequences.

Test cases

Creating an account: Green

Step	Input	Response
1	User launches the program.	User is prompt with a login screen
2	User clicks Create account	User is shown a create account scene.
3	User types; name, number and password and clicks Create account	All the information is send and saved to a database with password hashed. System creates a user with given information.

Make Reservation: Red

Step	Input	Response
1	User enters number, password, and signs in.	User is prompt with a main menu.
2.	User clicks on book table button	None because we did not have enough time to implement it into the GUI.

Manage Booking Green

Step	Input	Response
1	User clicks on the View Booking	User is prompt with a View Booking screen
2	User clicks on the booking they wants to change and than clicks Edit Booking	System prompts user to a edit booking screen where they can change all the information about boking
3	User fills the necessary edits and clicks Update	System changes the original information to the new on the database.

Manage User Green

Step	Input	Response
1	User clicks on the View Users	User is prompt with a View Users screen
2	User clicks on the user they wants to change and than clicks Edit User	System prompts user to a edit user screen where they can change all the information about user
3	User fills the necessary edits and clicks Update	System changes the original information to the new on the database.

Make Take Away Order Red

Step	Input	Response
1	User clicks on Take Away Order	None because we did not have enough time to connect the main view with the controller.

Results and Discussion

This software is open-source; therefore, anyone is welcome to try it out and give feedback. The source code can also be found in Appendix E with the installation guide. The User Guide can be found in Appendix C, while all the diagrams and Astah are in Appendix B.

The section below shows the status of all the user stories mentioned with a small explanation if needed.

Requirement	Status
1. As staff I want to see the table bookings, so that I know where the guests are seated.	Implemented
2. As staff I want to see the orders, so that I know what needs to be cooked.	Implemented
3. As staff I want to see comments, so that I can take them into consideration.	Implemented/bugs
4. As staff I want to see special requests, so that I accommodate them appropriately.	Implemented/bugs
5. As staff I want to see ordered food, so that the guests can receive their meal.	Implemented
6. As staff I want to edit orders, special requests, and table bookings, so that I can correct mistakes or cancel reservations.	Implemented/bugs
7. As guest I want to see the available tables, so that I can book them.	Implemented/bugs
8. As a guest I want to book a table, so that I have secured seating when I arrive.	Not Implemented

9. As staff I want to book a table, so that I can seat my guests.	Not Implemented
10. As a guest I want to pre-order meals, so that the staff can prepare them.	Not Implemented
11. As staff I want to order meals, so that the kitchen can prepare them.	Implemented but not connected to the main menu.
12. As a guest I want to make comments, so that I can get my special wishes fulfilled.	Implemented but not connected to the main menu.
13. As a guest I want to make special requests, so that I don't experience any difficulties.	Not Implemented
14. As staff I want to make special requests, so that I can fulfill the customers dietary needs.	Implemented but not connected to the main menu.
15. As a guest I want to create an account, so that I have my information saved for future visits.	Implemented
16. As a manager I want to unregister staff from the system, so that unemployed staff cannot access the system.	Partially Implemented
17. As a manager I want to remove/add dishes to/from the menu, so that I can offer the food that I have in stock.	Not Implemented

As the table above shows, the system does not fulfill all the initially set goals and is not optimized. A significant number of planned features have been scrapped during the process, either because the client did not directly ask for them, because they were not necessary for the program to run or because it was impossible to implement them before the deadline. The edit and view functionalities are implemented, albeit some with minor bugs, but the methods necessary to actually add information like orders, bookings, comments and food to the system will need to be implemented in the next sprint.

Conclusion

The client, Bob, is the head of a Danish restaurant chain, which is looking for a restaurant management system that combines table booking and the ordering of food to lessen the workload of their staff.

The product owner of the development team has been in constant contact with the client to assess the exact needs and make sure they are met. Following the Unified Process and SCRUM, each of the customers' requirements has been analyzed and formulated as user stories before being turned into use case descriptions. Additionally, a Domain model to showcase the data interaction has been created. The use case cases were used in the design section to create functional classes, which are part of the MVVM system architecture, following the SOLID principle. The "editOrder()" method was used as a primary example. A System Sequence diagram shows a user's interaction with all the specific parts of this architecture when using this method. At the end of the design process, a model for implementing the database has been created. The class diagrams and the database model are used in the implementation phase to implement all the designs into the system. Code snippets show the exact classes and methods of the "editOrder()" functionality with the system. With the implementation of the specific methods finished, the code was tested to ensure it meets the assigned requirements. In the end, the "results and discussion" chapter compares these tests with the user stories and reviews the entire process. The account creation, login, view and edit functionalities have successfully been implemented, the add functionalities like “order”, “book table”, “make comment” however will need another sprint to be finished.

Project future

The program was written as a Java application, so it is only available on PC. The add functionalities, like “order”, “preorder” and “book table” have not been completed and will need to be worked on further. Additionally implementing future extensions on a website or as a mobile application is recommended to make it accessible for more users like current competitors do.

Bibliography

Digite, 2022. *What Is Scrum Methodology? & Scrum Project Management*. [online] Available at: <https://www.digite.com/agile/scrum-methodology/> [Accessed 13 May 2022].

Javatpoint, 2022. *Remote Method Invocation (RMI) - javatpoint*. [online] Available at: <https://www.javatpoint.com/RMI> [Accessed 29 May 2022].

Martin, M., 2022. *MVC vs MVVM: Key Differences with Examples*. [online] Available at: <https://www.guru99.com/mvc-vs-mvvm.html> [Accessed 17 May 2022].

Millington, S., 2022. *A Solid Guide to SOLID Principles | Baeldung*. [online] Available at: <https://www.baeldung.com/solid-principles> [Accessed 20 May 2022].

Product Plan, 2022. *What is MoSCoW Prioritization? | Overview of the MoSCoW Method*. [online] Available at: <https://www.productplan.com/glossary/moscow-prioritization/> [Accessed 19 May 2022].

TheTeche, 2022. *The Unified Process in Software Engineering Theteche.com*. [online] Available at: <https://theteche.com/the-unified-process-in-software-engineering/> [Accessed 24 May 2022].

Vaidya, N., 2022. *Socket Programming in Java | Java Networking Tutorial | Edureka*. [online] Available at: <https://www.edureka.co/blog/socket-programming-in-java/> [Accessed 27 May 2022].

Appendices

Bring ideas to life
VIA University College

Process Report

Arturs Silins - 315226
Bhupas Gautam - 306411
Maximillian Marius Wallin - 315268
Ondrej Klimek - 315255
Siddhartha Jonathan Grasse - 315278

Supervisors:
Ib Havn
Jørn Martin Hajek

Software Technology Engineering
2nd semester

Table of Contents

List of Figures and Tables

Introduction

This is the process report for our second-semester project in Software Technology Engineering. After fruitfully completing our first semester, we started the new semester with more knowledge and experience about software engineering and ourselves. We first added a fifth member to our group, enhanced our collective capabilities, and made the new experience of working as a 5-member group instead of 4. In SEP2, we learned how to characterize the individual personalities of all our group members and how to use these most efficiently. Additionally, we learned about the cultural differences of our group members and how to use them to strengthen our bond. We also acquired more conflict management and communication skills. In SDJ2, we improved upon our Java knowledge while learning SQL and how to create and manage databases in DBS1. Finally, in SWE1, we deepened our project work skills, particularly the design and analysis parts. We also learned about SCRUM and UP, which is the framework we are using this semester to develop our project, instead of the waterfall, which we used in the first semester.

Group Description

Our group consists of 5 members, Bhupas from Bhutan, now a Danish citizen, Max from Denmark grew up in France, Ondrej from Czechia, Arturs from Latvia, and Siddhartha from Germany. We also learned Java and how to write efficient code in the first semester due to our previous experiences in C++, C, HTML/CSS, JavaScript, and Python. Having already finished a one-semester project, we already learned a lot about ourselves and the process of working on a project together. We also learned about our different cultural backgrounds and personality types, which will be elaborated on below.

E-stimate Personal Profile

▲ Bhutan/Bhupas ▲ Latvia/Arturs ▲ Czechia/Ondrej ▲ Denmark/Maximillian △ France ▲ Germany/Siddhartha

The team wheel in figure Figure 1 shows the specific personality types of our group.
As you can see, there is a high concentration of green personality types, which significantly benefits the general group harmony.
.
Arturs is just as potent on his blue side as his green one, making him very systematic and a great SCRUM Master. On bad days he can be very skeptical and cool. Ondrej is a little greener than Arturs, which does not make an ideal product owner personality

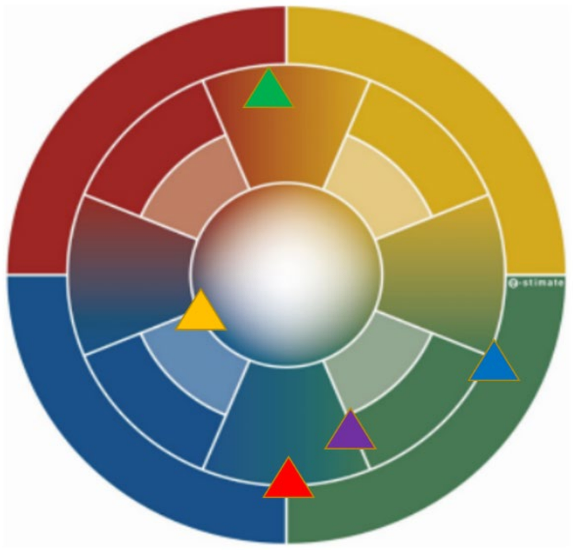


Figure 1 Personal profile

wise, but we chose him because he came up with the idea for our project. Ondrej supports the group with his agreeableness and sense of harmony. On bad days Ondrej acts hesitant and defeated. He is always happy to assist anyone who needs his help. Also, he is very systematic and reliable with a strong blue side. Max, on the other hand, is also very green but has a strong yellow side, this combination allows him to take care of the general well-being of the group. He spreads harmony and liveliness. He can be very uninhibited and unfocused on his bad days. Siddhartha has a strong red and yellow tendency, making him demanding and result-oriented while helping the group form and fulfill the necessary goals with his creativity and enthusiasm. On his bad days, he tends to show the weaknesses of his yellow side, seeking a lot of external contacts and quickly losing his focus. Bhupas combines all the above characteristics with a slight emphasis on the blue side, making him flexible enough to perform any needed role. He acts a little slow and evasive on his bad days.

Cultural Dimensions

Next, we will explain our cultures and how they impact our behaviors combined with our personality types. Figure 2 shows the cultural dimensions of the countries we grew up in, while Figure 3 shows our stances.

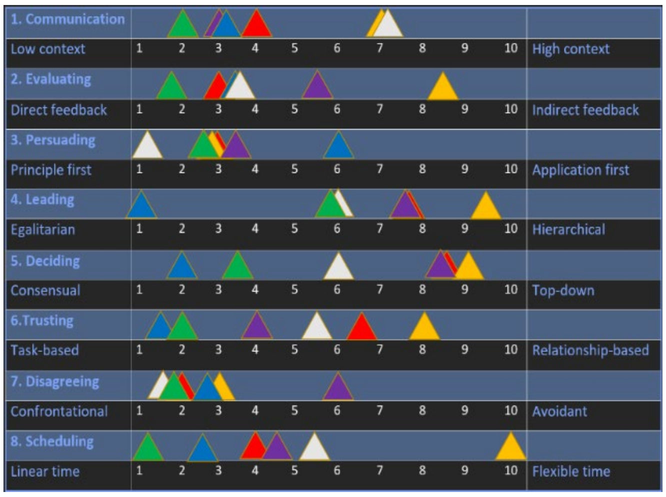


Figure 2 Cultural-dimensions

Our stances are affected by the culture we grew up in and our personality types. For example, Arturs's blue tendencies make him give feedback more directly, while his green tendency makes him less confrontational than his culture indicates.

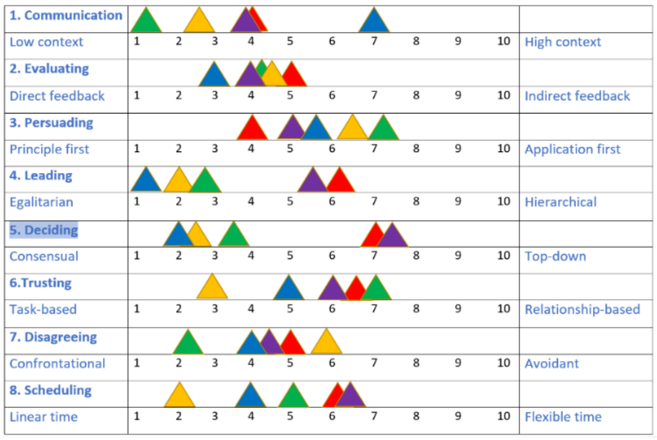


Figure 3 Cultural-dimensions-personal

Similarly, Ondrej's strong green tendencies make him give more indirect feedback and be less confrontational than the Czech culture. His French and Danish upbringing influenced Maximillian; the French side shows in his communication and trust, while his Danish side shows in his persuading and deciding. His scheduling is right between the French and Danish tendencies.
Siddhartha's red tendencies make him communicate, evaluate and persuade even more direct than in the German culture. In contrast, his yellow side makes his trust more relationship-based and his scheduling more flexible. Bhupas has been strongly impacted by his life in Denmark, aligning with most Danish characteristics, besides trusting, which shows a combination of both cultures. His level of disagreeing is very uncharacteristic of both the Danish and Bhutanese culture, probably impacted by his green tendencies.

Even though we all come from different cultures and have different personality types, our stances are very aligned, which helps us cooperate and efficiently resolve any conflicts that might arise. Especially Arturs and Ondrej share many characteristics both culturally and personality-wise, which makes them empathize and understand each other's perspectives.
Our group uses Douglas McGregor's Theory Y approach like the Danish school system.(Anon., 2022a) We are responsible for our work, only using the teacher as an assistant if we do not manage to solve our problems on our own. Our group does not have a clear leader, and we all have our freedom, which can be inefficient at times, but it is essential to hear everyone's opinion before deciding. We are also very aware of our strengths and weaknesses, using them to realize our best potential.
Additionally, we sometimes split ourselves up into smaller groups, but never with less than two people. We all have a powerful drive to complete the project to the best of our abilities, not just to get a good grade but to develop the necessary knowledge and skills to achieve our full potential in our careers as software engineers. Because of this, we implement the deep learning approach. We are working on this project to learn these skills in line with Denmark's problem-based learning system.

Project Initiation

As opposed to the first semester, we can now choose our topic, which we must complete according to the SCRUM method using UP. We had to fulfill a few requirements, the system has to be a client-server system using RMI or Sockets, and it must connect to a database. Each group member had to come up with two ideas each, after which we decided on two ideas to present to our supervisors. Our group members' actual jobs inspired these ideas: Ondrej and Siddhartha were chefs, Arturs was a dishwasher, and Maximillian was a waiter. We had a recipe finder system as our primary candidate. Still, our supervisor deemed it too simple, so we chose our secondary candidate, a restaurant booking system, as our semester project.

Project Description

As most of our group has much experience in the restaurant business, we all had many ideas for useful features we could implement into our program, which led to a long brainstorming session with Ondrej as project owner, always having the last word on what actually to use. Especially Maximillian, who has worked as a waiter for years, and Siddhartha, a chef, had many ideas of features they would like to implement into the system and how exactly to implement them. Some of them were accepted by Ondrej and put into the project description. We still added and removed a few features during the project.

Project Execution

The approach of our project execution was described under the semester description of SEP2. We used SCRUM to divide our project into different sprints, working on one sprint after another in the proper order.(Digite, 2022) Our supervisors were available whenever we needed their support, which happened a couple of times, especially when it came to using SCRUM. We applied the database skills we learned in DBS1, the design patterns and JAVA knowledge we learned in SDJ2, and the design, planning, and report writing specifications we learned in SWE1 and SEP2. Having learned from last semester, our group met every day except for Sundays while being split up into two teams for the majority of the time. We decided to study at VIA instead of working by ourselves at home because it creates a motivating learning atmosphere free from distractions. We had regular SCRUM meetings in which we caught each other up on our progress, resolved internal conflicts, and discussed how to handle specific situations in the future. We chose to individually test our code during the sprints and only mark them as done when it passed all the critical tests. Each sprint consisted of an elaboration phase, in which the design was planning was finished, a construction phase in which the actual code was implemented and a transition phase, which concluded the sprint with testing and finishing touches. Initially, we had a lot of high goals and expectations of our system, but we started seeing the project in a more realistic way when we began the sprints. During the sprints, we adjusted to these expectations and ended up with a viable product for the customer. All our sprints will be explained in more detail below.

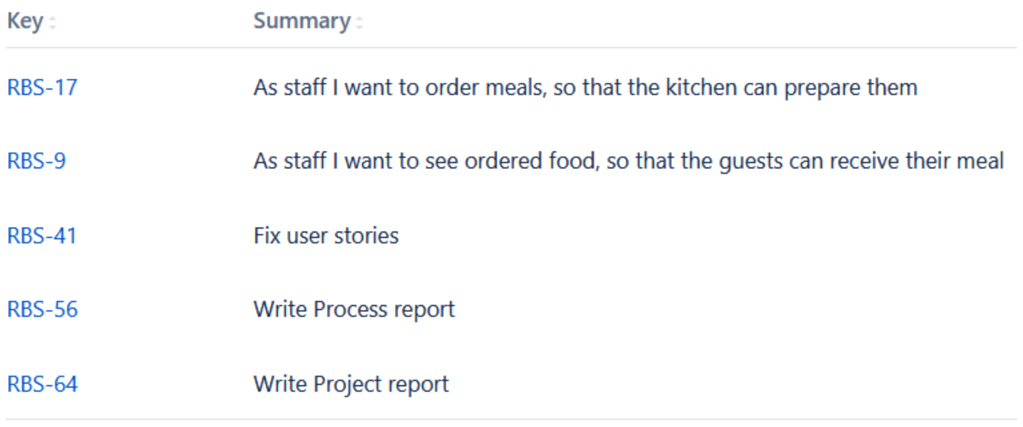
Jira and Git

Our goal in this project has been to work as close as possible to a real software firm; therefore, we wanted to use Jira software from the beginning.(Jira., 2022b) We had heard that Jira takes time to get used to; therefore, we gave total freedom to our scrum master to decide if he wanted to use Excel or Jira to document our workflow.

One of the first things we did in Jira was to put all our user stories in the Backlog and used the serialized names to name the branches on git. Additionally, we linked Jira to Git, which helped us keep an overview of our work and showed how many commits and pull requests have been made. By working in this way, we knew at all times which branches were responsible for which user stories; after finding a solution to the user stories, we merged the branch with the main branch. When creating a sprint, we used MoSCoW method to group the user stories.(Product Plan, 2022) We worked on the most periodized features decided by the project owner. We did not use all the features of Jira as intended; for example, while working on the user stories, we kept forgetting to update the board.

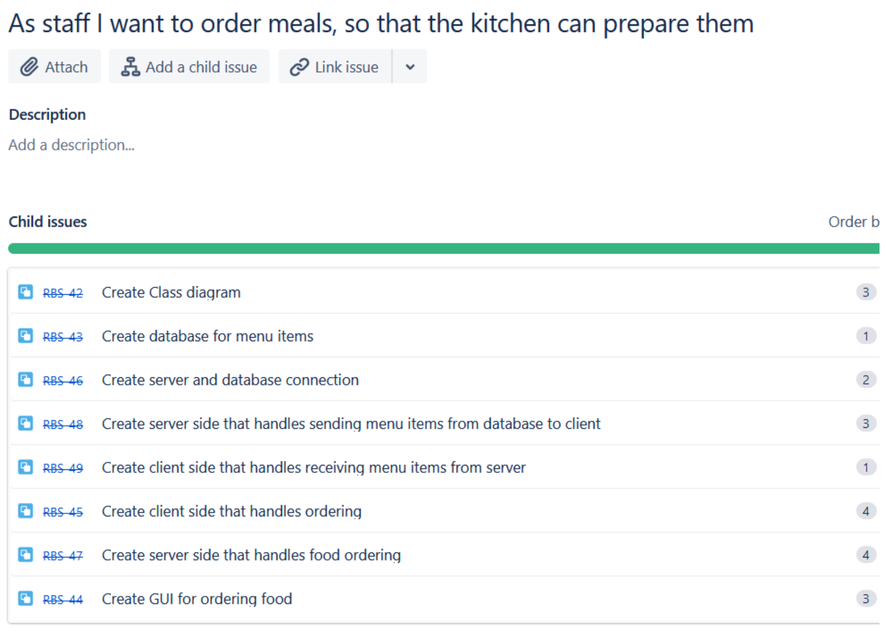
Sprint 1:

We used our user stories to create items for the product backlog. We asked our product owner about the essential tasks and selected these for the first sprint. These tasks include the functionality of ordering meals (21 Story points) and seeing the ordered meals (9 Story points). Additionally, we added the process report and project report writing, to which we assigned 2 story points each, to these items. We distributed this story points to the tasks by playing "Poker" on discord. Figure 4 shows this product backlog, while Figure 5 and Figure 6 show the sprint backlog of the two main products.
Altogether the first sprint consisted of 36 Story Points.



Key	Summary
RBS-17	As staff I want to order meals, so that the kitchen can prepare them
RBS-9	As staff I want to see ordered food, so that the guests can receive their meal
RBS-41	Fix user stories
RBS-56	Write Process report
RBS-64	Write Project report

Figure 4 Sprint 1 - Product backlog



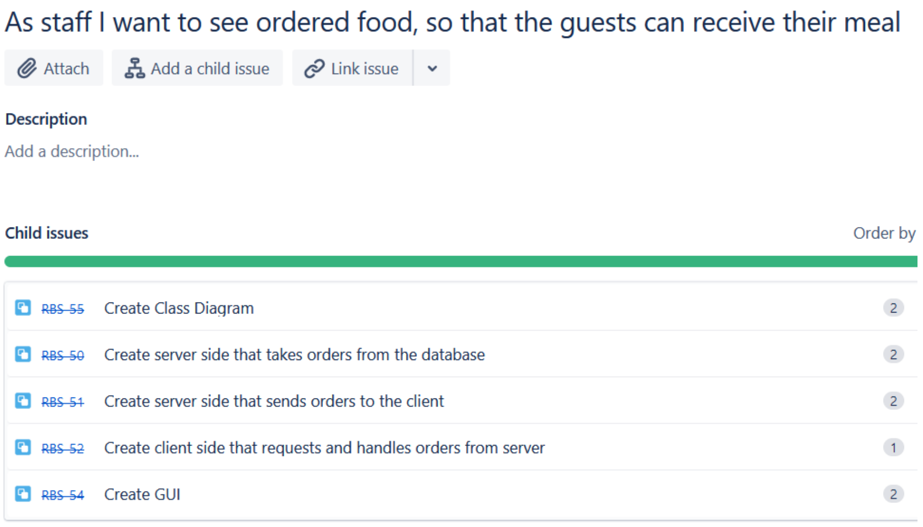
As staff I want to order meals, so that the kitchen can prepare them

Description
Add a description...

Child issues Order b

~~RBS-42~~	Create Class diagram	3
~~RBS-43~~	Create database for menu items	1
~~RBS-46~~	Create server and database connection	2
~~RBS-48~~	Create server side that handles sending menu items from database to client	3
~~RBS-49~~	Create client side that handles receiving menu items from server	1
~~RBS-45~~	Create client side that handles ordering	4
~~RBS-47~~	Create server side that handles food ordering	4
~~RBS-44~~	Create GUI for ordering food	3

Figure 5 Sprint backlog RBS-17



As staff I want to see ordered food, so that the guests can receive their meal

Attach | Add a child issue | Link issue

Description

Add a description...

Child issues Order by

~~RBS-55~~	Create Class Diagram	2
~~RBS-50~~	Create server side that takes orders from the database	2
~~RBS-51~~	Create server side that sends orders to the client	2
~~RBS-52~~	Create client side that requests and handles orders from server	1
~~RBS-54~~	Create GUI	2

Figure 6 Sprint backlog RBS-9

Sprint Review:

The GUIs for ordering food and seeing the ordered food do not perfectly match the original vision, but they fulfill the clients' requirements; it is easy to navigate and read, and the font needs to be adjusted. Minor Bugs were found in the system, and these will need to be fixed in the next sprint. The sprint was finished far later than estimated.

Sprint Retrospective:

Figure 7 shows the process of working through these tasks during the sprint. We completed most tasks on the first day, with the rest of the time being linear. We underestimated how long it would take us to finish these tasks, so we finished the sprint far later than expected. This indicates that we were unsure how to distribute the necessary story points and used less time than expected for some items and more time than others.

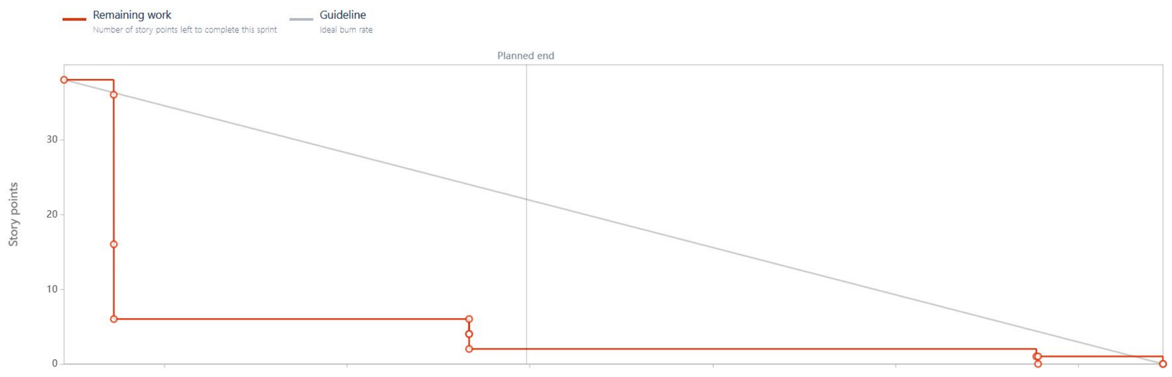


Figure 7 Sprint - Burndown chart

We have learned things that we should start doing, things that we should keep doing, and things that we should stop doing.

What to start doing:

- Properly discuss and estimate how long each task will take.

What to continue doing:

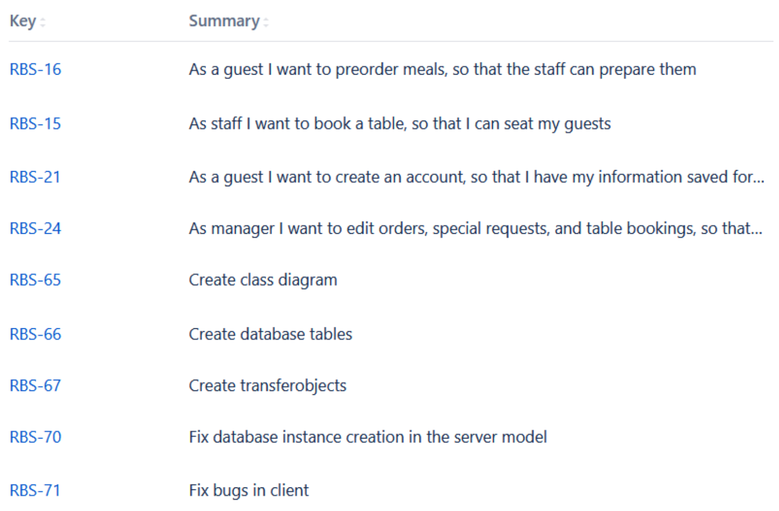
- Scrum Poker.
- Commenting on the code while implementing.
- Using git.
- Writing the reports while coding.

What to stop doing:

- Using the waterfall mindset for SCRUM.

Sprint 2:

With what we learned from the first sprint, we had a better idea of how to distribute our work. In the first sprint, we severely underestimated how many story points it would take to finish the class diagram; we had allocated only one-story point but needed 6. We finished the poker relatively quickly and ended up with 34 story points but had to add more during the sprint when we found critical bugs that needed to be fixed immediately. The total amount of story points for this sprint turned out to be 39. Figure 8 shows the product backlog of this sprint.



Key	Summary
RBS-16	As a guest I want to preorder meals, so that the staff can prepare them
RBS-15	As staff I want to book a table, so that I can seat my guests
RBS-21	As a guest I want to create an account, so that I have my information saved for...
RBS-24	As manager I want to edit orders, special requests, and table bookings, so that...
RBS-65	Create class diagram
RBS-66	Create database tables
RBS-67	Create transferobjects
RBS-70	Fix database instance creation in the server model
RBS-71	Fix bugs in client

Figure 8 Sprint 2 - Product backlog

Sprint Review:

The bugs from the first sprint have been fixed, and more functionality has been added to TUI, but no GUI has been implemented in this sprint, making it hard to assess the functionality. Additional bugs have been discovered that need to be fixed in the next sprint.

Sprint Retrospective:

As the burndown chart in Figure 9 shows, the whole team worked on the tasks consistently. Still, we found significant bug issues towards the end of the sprint, which needed to be solved immediately, so more story points had to be allocated for these.

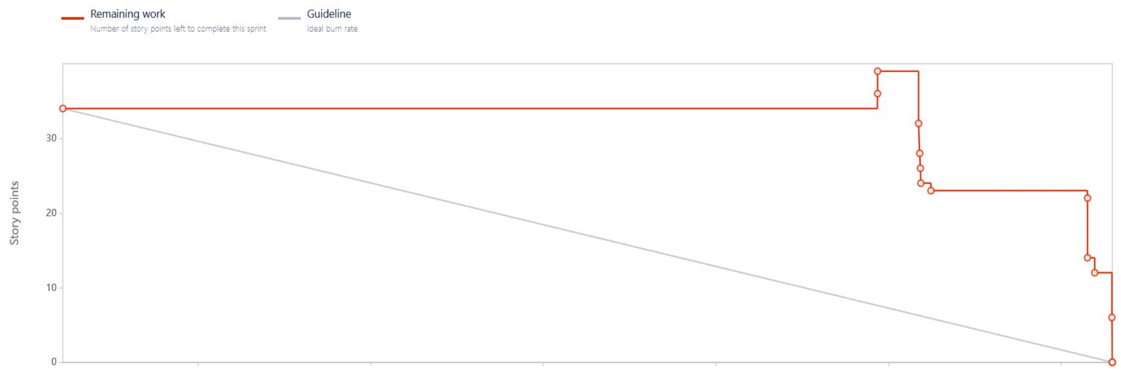


Figure 9 Sprint 2 - Burndown chart

We will now elaborate on the lessons we have learned from this sprint.

What to start doing:

-Test our code during implementation to avoid critical bugs.
-Properly distribute the workload so that some team members don't finish their tasks too early. In contrast, others struggled to finish theirs in time.

What to continue doing:

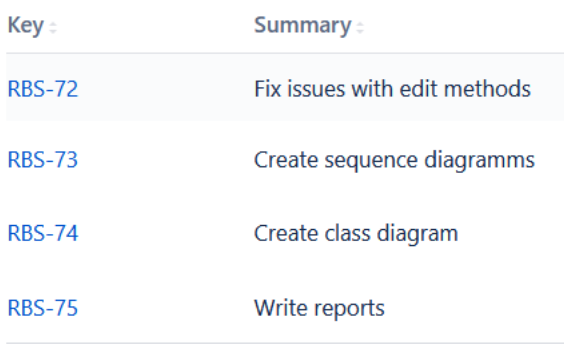
-Properly estimate the workload of each item.
-Distribute the coding and report writing among team members equally.
-Keep following the SOLID principle.

What to stop doing:

-Spend too much time on the SCRUM poker.

Sprint 3

Since we had primarily focused on programming on Sprint 2, we decided to use sprint 3 for report writing and bug fixing. Additionally, we need to create diagrams to assist us with implementing the methods. Since we already knew the issues that needed to be addressed in this sprint, we quickly finished the poker. We also used this sprint to review our code and reports, so we all had a better project overview. The product backlog of this sprint is shown in Figure 10.



Key	Summary
RBS-72	Fix issues with edit methods
RBS-73	Create sequence diagramms
RBS-74	Create class diagram
RBS-75	Write reports

Figure 10 Sprint 3 - Product backlog

Sprint Review:

This sprint has not produced any actual products and was finished late. Still, it has laid critical groundwork for the final sprint, having taken a lot of the planning and organizing out of the way. The product owner requested to add hashing to the account creation functionality completed in sprint 2. This will be implemented in the next sprint.

Sprint Retrospective:

Although this sprint has not included any significant programming progress, it turned out to be a critical part of the process. We managed to find and fix flaws that we were not aware of. Additionally, this sprint helped us understand each team member's workflow. As the Burndown chart in Figure 11 shows, the whole team has been working on their tasks consistently and finished them together, even if it was not when we originally planned to finish.

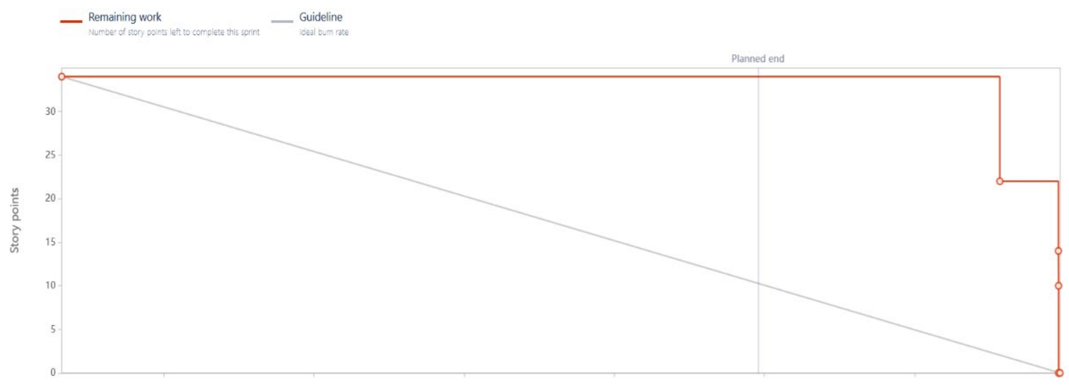


Figure 11 Sprint 3 - Burndown chart

What to start doing:

-Put more time into implementing the methods needed.

What to keep doing:

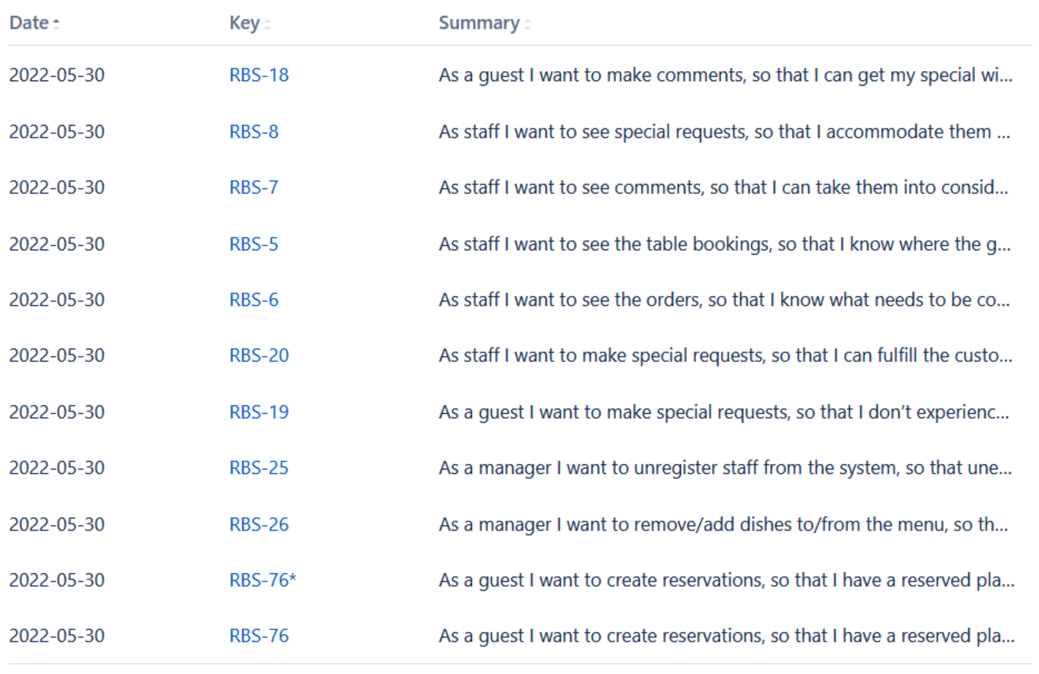
-Distribute the workload equally and support each other.

What to stop doing:

-Underestimate the time it takes to finish our tasks.

Sprint 4

After having finished all the groundwork in the previous sprints, all that is left is to implement the minor issues that have not been a priority so far. A list of these issues is visualized in the product backlog in Figure 12.



Date	Key	Summary
2022-05-30	RBS-18	As a guest I want to make comments, so that I can get my special wi...
2022-05-30	RBS-8	As staff I want to see special requests, so that I accommodate them ...
2022-05-30	RBS-7	As staff I want to see comments, so that I can take them into consid...
2022-05-30	RBS-5	As staff I want to see the table bookings, so that I know where the g...
2022-05-30	RBS-6	As staff I want to see the orders, so that I know what needs to be co...
2022-05-30	RBS-20	As staff I want to make special requests, so that I can fulfill the custo...
2022-05-30	RBS-19	As a guest I want to make special requests, so that I don't experienc...
2022-05-30	RBS-25	As a manager I want to unregister staff from the system, so that une...
2022-05-30	RBS-26	As a manager I want to remove/add dishes to/from the menu, so th...
2022-05-30	RBS-76*	As a guest I want to create reservations, so that I have a reserved pla...
2022-05-30	RBS-76	As a guest I want to create reservations, so that I have a reserved pla...

Figure 12 Sprint 4 - Product backlog

These tasks have been distributed equally among the team members using SCRUM poker.

Sprint Review:

Less than half of the planned features for this sprint have successfully been implemented, mainly the methods necessary to view and manage the added information.

However the methods that are necessary to actually add this information like “make special requests”, “create reservations” and “order food” have not been finished before the deadline, so they will need to be implemented in a fifth sprint.

Sprint Retrospective:

This sprint was very programming heavy, with many methods that still needed to be implemented. This made the workflow chaotic and hard to work on. While working on this sprint our programming process was a lot slower than expected. We had planned to finish implementation of all of our use case stories but because of the amount of items that needed to be implemented we could only make half of them work in this sprint. The burndown chart in Figure 13 shows that we had to complete this sprint without implementing all these items.. In this sprint we all were too focused on programming that we forgot to update the Jira board and ended up updating everything at the end. Instead of using more time to implement these items during this sprint, the development team will focus on the documentation and move the missing items to a fifth sprint.

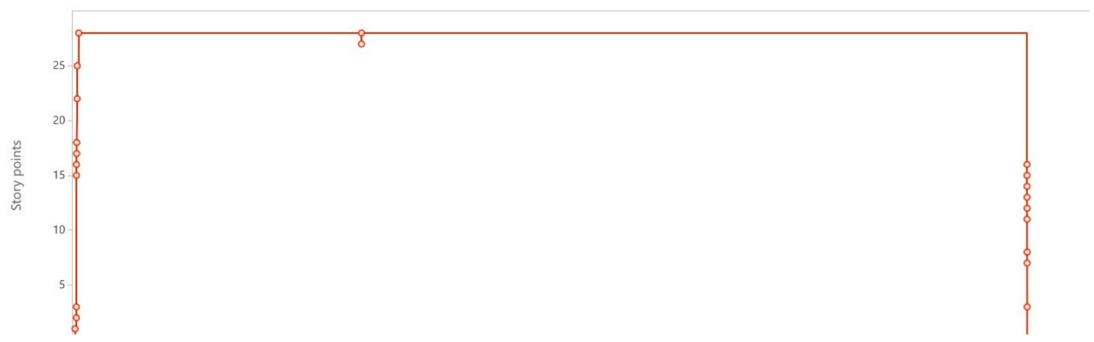


Figure 13 Sprint 4 - Burndown chart

What to start doing:

-Have more realistic expectations of necessary implementations
- Start to update the board on Jira as soon as task is finished.
- Make a better plan for next sprint.

What to keep doing:

-Support each other whenever problems arise

What to stop doing:

- Keeping too many tasks for the last sprint.

Personal Reflections

Arturs Silins

A restaurant booking system is something that most of the group had experience with, as 4 of the group members are working in a restaurant. The group was the same group as last semesters group, but we added another member who was already in our friendgroup from last semester, so it was easy to add him to the semester project group.

The group worked very well on the background description, but we lacked planning for the analysis, because everyone had their own personal problems, so the whole group was not able to work when the analysis phase should have been finished, so we lacked a little behind and only started working on the project when the project period had started.

For planning we used Jira, we had a backlog from which we could make sprints, we assigned user stories to group members, which helped to plan our sprints and our work. It was much easier to communicate who was working on what for how long. The working times and locations were very flexible, sometimes we worked in school and sometimes we worked from home. We defined our problem by discussing it with the product owner, who acted in a way to simulate a direct contact with the customer and was able to define the problems and needs the customer has. The goals we initially set were not fully realistic, we put a lot of different variables into the program, for example we accounted that there are dine in orders, but the customer can call the restaurant and make a reservation that way, so the staff would have to go into the system and make a reservation, or the customer can use the application to make the reservation, which adds more complexity to the system because one thing is not only used in one cases, but in multiple cases in different ways.

We tackled what appeared like the most critical problems in the beginning and then we followed them in the sprint plan, but due to lack of experience we ended up planning the sprints with more simple tasks and ended up leaving the difficult tasks to the end.

Funny enough I would have started earlier, but realistically if we had more time we would have had more sprints, but I think we planned it out as good as we could have considering it was our first time using SCRUM and Unified Process. If it was up to me I would have worked more with the class diagrams in the first sprint so that the following sprints could be more focused on the coding part, which would avoid some members having to wait for other members to finish their tasks before being able to move on.

Compared to the first semester project the second semester project is much more complex and feels like an actual realistic product that a company could use. If we had managed to implement all the functionality we were planning to and was more developed and designed it could actually be sold on the market.

I feel like the group contract was made a little to harsh, for example we were supposed to pay money if we were late or disturb the group members. I think the penalties should have been less extreme, which would have made it easier to actually enforce them.

Because the group contract was not enforced it was never taken seriously, so it did not impact any of the work, but it improved the relationship of the group members because it was the first thing that we did together.

This semesters group contract was not enforced at all, us being a friend group it is not something we wanted to do, so using less harsh penalties would make this a lot easier.

I was the SCRUM master of the group and I studied Jira before we started the SCRUM planning and made a backlog. I also set up rules on how to use github to avoid merge conflicts and code glitches. In github I also set up versions of the code, which were easy to follow after the sprints, so there were different releases after each sprint. Also I studied about version control, so that in JavaDocs we could actually write as close to real life standard of version control as possible.

The group members each contributed differently, they all used their own strengths in the project work.

What motivated us the most was the lack of work we had done and the hand in date getting closer and closer. I think that in the beginning of the project period some members were demotivated by the idea of how much time we will have until the hand in, which resulted in not putting in 100% of their power into the project.

We did not have any major challenges caused by cultural differences, because of being friends we created our own kind of culture, which does not have a negative impact for our group. I learned that I like to follow standards and have a systematic approach to things, which helps keep the group more organized. From this project period I have learned a few things about structuring work, from our profession I learned more about using github and version control, which I will work on and use in the further group projects, because it is something that we will definitely encounter when we work in a real company.

Working in a group while solving a problem helps to have different views on the problem, which results in the problem being solved more quickly and/or more correctly.

Problem based learning can create bad habits which leads you to solving a problem incorrectly or more inefficiently due to the lack of direct supervision.

But it teaches us to find the answers, as real work or life doesn’t always have answers in a book that you can just follow. By working with problem-based learning we can find what helps ourselves find the answers to our own problems. And at the end we can all be very proud of our accomplishments even if we did not manage to finish the program in time.

Bhupas Gautam

I selected the topic because I wanted to work on a project that is interesting in but not familiar with. I use the food ordering app a lot and always wanted to create something like that, but I was not familiar with the restaurant industry thus, I did not know how to do it. I decided to continue working in the same group as last semester because we worked together very well and complemented each other with our individual strengths and weaknesses. We also made a management app last year with the same people, and I wanted to continue sharpening my skills in that direction for future job opportunities.

For the project, we did much planning initially, perhaps too much. Still, it gave us an overview of everything that would happen in the project. Many of our group members we already familiar with the environment because of that, it took a long time to plan the project. At first, we wanted to create something very advanced and perhaps not realistic to our capabilities as programmers. Unfortunately, we had to remove many of our additionally conceived features to keep everything realistic.

In this project, we tried to use SCRUM with help from Jira. In the beginning, I didn't know how we were going to work on SCRUM because I was very used to the waterfall model. After the first sprint meeting, I preferred SCRUM a lot more than waterfall because it gave me an overview of what was happening through daily standup meetings and SCRUM meetings. The SCRUM method fits me perfectly because I am an allrounder in my colors while having a strong red side, so I always want to have an overview of everything. Jira was easy to use because I had used software made by the same company before. It gave me an overview of who is doing which specific task and how far they were along, thus helping me create a balanced group harmony. Even though our program ended up being quite simple and does not have all the features we initially conceived because we did not have enough time to implement many methods on the last sprint. It does most of what it needs to do in a good and efficient way, rather than doing everything, which I think the companies would prefer. Our group contract was less strict than last semester's contract, but it was still strict regarding time and communication. Unfortunately, just like last semester, we did not follow our group contract as thoroughly as possible, especially regarding the time. Despite that we understood each other's different ways of working and solving problems. None of our group members were penalized. But in the future, I feel like we should enforce the contract more.

Because of my personality, I like to have everything in order and make sure that everything is clear. It helps me understand my group members better and help them with what they need. We all had our strengths and weaknesses, which we already knew from the last semester. Even though we had a new group member, we were already familiar with his working style and capabilities. Perhaps because of that familiarity, we focused on what we were good at, which was both positive and damaging to the group. As a group, we contributed as much as possible. Still, as a member, I was only able to improve on what I was already good at instead of learning about new things that I am not yet good at. I want to change that in future semesters. We all had a goal of doing our best to our capabilities just like last semester and perhaps even improved on that. Our motivation was to push each other beyond what we were already capable of, thus always being together, even when we don't work in a group. This helped us push even more than what I initially conceived. As seen in the personal profiles, despite us coming from different backgrounds, there are many things we are similar in, especially regarding our ambition and education, therefore, our cultural background played little to no role in our group. I have learned that because of my red tendency, I would not always be satisfied with the group's work, which I would like to improve in the future.

For the next project, I would spend less time planning because it never goes as planned, and instead, I would start the project from the beginning. I would rather discuss and do a small task than discuss every detail. Personally, I feel like we discussed more than we worked on, which I would like to improve in our third semester. Group-based learning helped me understand a topic that I was unfamiliar with because one of the group members would explain it to me. I feel like SEP2 has improved me more as a programmer than what I had learned in this semester because SEP2 helped me combine all the knowledge and visualize what I have learned. I chose to study at VIA because VIA teaches its students problem-based learning, which I'm very familiar with since high school. This is also used in most Danish companies, and there is a reason for that. That reason is that problem-based learning helps me think critically and solve the problem even if I did not know how to do it initially. It forced me to learn new things.

Maximillian Marius Wallin

Even though I was the one new member of the group it felt natural, as we were all already friends previously. I had a very good idea about the other members worked. During the project initiation each member had to pick a few ideas for the project, a few of us picked restaurant themed projects, which fit perfectly because I had a very good idea about how booking systems and everything that had to do with restaurants, as I was a waiter for 4 years. I experience definitely helped with this project. I think that we had a very precise idea about how and when we wanted things to go, we had a very good planning that we would start early and finish a few days before so that we could finalize our project. But in reality we sadly got caught in an unproductive state of mind in the first week, which let us fall behind a little bit on all the things we aspired to do. We also underestimated the amount of coding the project would take. This was a problem at the end, because we did not manage to fulfill all the requirements set by the client. The main planning tool we used for this project was Jira. This tool made everything more organized, so that we could see when we had to do each task and which sub issues that task had. It had a broad access to tools, so we never had any doubt about how the planning tool worked. We wanted to make a fully working restaurant booking application, that would be easy and intuitive to use. We had set our expectations and requirements higher than what they should have been. In reality we did not allocate the time correctly, so the outcome of our project was sadly not exactly what we had in mind, but the base requirements were met nonetheless. I think that we bit off more than we could chew. The methods we used were SCRUM and UP, at first I had to get used to not using the waterfall model that we learned in the first semester, but once I got into the rhythm of using SCRUM and having our SCRUM master do all the planning for us, it made it more comfortable and efficient to use. We had daily standup meetings, which were a great way to follow up on where each person was at their task and what still needed to be done. This helped the whole team to unify in our process. Our group contract included a policy to not be late for a certain amount of time, if not followed there would be a penality that we had to fulfill. Only if necessary did we stay home instead of meeting up, but we still got the work done that we were supposed to do. I liked to come up with ideas for the project since I have some experience in the restaurant business, so I knew what I wanted to improve on or remove. I believe that I contributed a lot with my professional knowledge. Every member of our group has a field that they specialize in better than others, which is a good thing and a bad thing because for example the people who are good at specific things would usually stay in their field, which limits the amount of learning they can do in the fields that they don't excel in. So while it is good that we all specialize in one thing it would be better that we would all try to grow in every direction. We wanted the best possible outcome of our project, and we also wanted to learn the necessary skills it takes to become good software engineers. I did not notice any major cultural differences between the group members. In the group I believe I can follow through on the tasks I have been assigned without needing assistance from other group members. But that does not mean that I do not rely on others when it is needed. Problem based learning is a good experience in itself because it will accommodate you to working in companies where huge projects are an every day experience. Although we did not get everything done in time the current product is a big accomplishment that the group can be proud of.

Ondrej Klimek

I selected this topic because I work in a restaurant and the system we were using at the time was insufficient and inefficient. Most of the group was formed last semester, bcausee of good friendships we added one more member. We spent too much time on planning, which ruined our timeplans. We used Jira, which is a little bit confusing sometimes. The goal we set up was mostly realistic, the unrealistic goals were some requirements which we have not learned to implement yet. We started much later on the project execution than I would have preferred, and during the execution we would meet at 10, although I personally would prefer to meet earlier. 8am would be a good time to start. We used SCRUM and UP. In my opinion SCRUM is not appropriate for small projects such as our semester project, and for small teams. When it comes to unified process I prefered the waterfall way as we had everything ready before we took on the coding part.

I wish we had started on our project earlier so that we would have had enough time to implement more sophisticated software. One of the risks involved is lack of experience, which shows in inaccurate planning, because of that we had to get rid of some of the initially set requirements and did not manage to implement all the necessary functionality in time.

Unfortunately, the group contract was not taken too seriously. We set many rules, especially ones for being late, but we have not enforced any of them. We also set our own deadlines, which we also did not hold up to. I felt the responsibility of coming up with the right requirements since it was my idea to make a reservation and ordering system for a restaurant. Since the contract was not really enforced it did not have any impact on our project. When making the next group contract I would like to put more thought into it and make sure it is enforced. Next time I would like to include set times of the group meetings.

The group worked well together since we knew each other well from the previous semester. Some group members have contributed more than others, which is normal because not everyone is at the same level and different people have different experiences. I believe everyone did the best they could considering their private life situation.
The group was motivated by the idea of the final product, but on the other hand we were demotivated by each delay.
Our team was not that multicultural, as all of us have similar cultural backgrounds. Since there were no major differences there were not many different experiences to benefit from. I learned that there can be some things some people do that might be irritating to me, and if they repeat these things too often I get annoyed. The solution to that is not to keep this to myself and to express my feelings openly. Next time I will try to be more active during the lessons, and during the study period in general. The advantages of group work are that each person can bring new ideas, see the problem from a different point of view and there are more people to deal with the tasks. With problem based learning we learn how to incorporate the newly gained knowledge in practice. The disadvantages of group work could be disagreements between the group members and incompatibility between the personality types. I don’t see any significant disadvantages of problem based learning.

Advantages of creating a problem formulation could be to see the problem from the customers point of view. One of the disadvantages could be that we don’t have enough experience to know all the problems we could encounter in real life.

Siddhartha Jonathan Grasse

The group was formed early on with the same members as last semesters project group, with the new addition of Maximillian, who I already knew from class and considered a friend. He quickly showed spirit and a drive, additionally to a willingness to keep the learning atmosphere light hearted. I on the other hand realised some issues that showed during the process of the project, particularely from the yellow side of my personality. I ended up being distracted easily and generally very unfocused. I tried to fight this as much as possible and quickly figured out how to contribute to the project as much as I could. During the inception phase I had a lot of ideas and we initially decided for my project idea, a recipe finder system as the groups favorite, but unfortunately our supervisors deemed it too simple to use for this semester project. That's why we decided to use our secondary idea, a restaurant booking system, instead. Being a professional chef this was a great alternative for me and I contributed with a lot of relevant potential functionalities for our system. We did however spend a little too much time spitballing these ideas and should have finished that earlier by staying more realistic and focusing less on conceiving the most ideal and perfect system we could. That way we could have started the execution phase earlier, which would have ensured a stressfree work environment. We ended up scrapping a lot of the functionality in order to finish a simple and functional system and even then we did not manage to implement all the critical requirements set by the customer. The main methods we used during the project were SCRUM and UP, albeit both of them being confusing in the beginning turned out to be very helpful and actually very similar to how I thought of the process in the last semester. These methods helped us to stay flexible during the development and focus on one issue at a time. I am satisfied with the project results, as compared to last semester our program actually works, even with the limited functionality. During the project I realised that the main focus should be to apply the knowledge we had learned in school in a realistic way and not to create a program that would actually be used by a company. The group contract was followed by all members, except when it came to the time management part, like meeting on time and handing in the project a few days in advance. We however never felt a need to enforce any of the rules and regulations, which I think is a good thing, even though we should try to all arrive at the same time in future projects.

I felt very responsible for the group project, having a very high drive and being very goal oriented I tried to help everyone do their job as efficient as possible, albeit I sometimes forgot to pay attention to my own contributions. At the end we all worked together in harmony and got the job done to the best of our abilities. All the regulations of the group contract that have been followed were helpful to maintain a group harmony, but it was obvious that some members had issues with their own time management, which created some friction between these members and the members that decided to arrive to the meetings on time.

The only thing that needs to be changed in the next contract would be an actual enforcement of the meeting times.

Our group did a very good job utilising each members strength to the best of their abilities, this however created a specific niche for each member, which kept them from learning new things outside of their horizon. This is a decision we made heavy heartedly, in order to finish the project on time and with all the requirements met. I believe all of our members were highly motivated, not just in order to finish the project, but also to pick up the necessary skills to ensure a bright future as software engineers. Our cultural backgrounds did not differ in a very obvious way, so it did not influence our group behavior in a major way. I learned that I can be responsible for the motivation of my group partners, making them actually want to work on their tasks when I'm around, as long as I also contribute myself. Additionally I have a lot of creativity that I use to find new and interesting ways to solve problems we encounter during the project.

Next time I'm involved in a group project I will keep working on my weaknesses, like being distracted easily and losing focus.

Group work and problem based learning are fantastic principles, which prepare us for the professional life as software engineers. I am very grateful to experience this kind of education system, as I definitely prefer it over the german one. I feel very motivated and know exactly why I'm learning certain things, because I'm able to apply them right away. I also like working in groups as it is much easier for me to stay motivated than it is when working alone. I can not think of any disadvantages for this kind of system. Even without the critical requirements met I'm very grateful for all the experiences I have made during this project period and look forward to the next one.

Supervision

Ib Havn, our SWE1 teacher, and Jørn Martin Hajek, our SDJ2 teacher, mainly supervised this project. They were both very supportive whenever we asked them for help, which admittedly was not very often. In retrospect, we should have used this option more frequently, which would have made the process easier, and had allowed us to finish all the requirements in time. We want to thank them both for their support of us.

Conclusion

This project aimed to develop a table booking system using SCRUM and Unified Process. In the beginning, it was challenging for us to adjust from the waterfall method that we had learned in the previous semester to the SCRUM method that we learned in this one. Additionally, our group added a fifth member who changed the group dynamic of the prior project. However, this did not present any issues and improve our productivity in the long run. Our team split up more naturally into two halves, which could work on particular problems independently. It was essential to keep the communication open between these subgroups. This ensures that everyone understands why specific decisions are being made; we decided to meet and discuss our goals every day and work alongside each other whenever necessary. To make working with SCRUM easier, we held stand-up meetings every day. We utilized the software Jira, which helped us organize and review the individual sprints. In the future, we will continue to use SCRUM and Jira because it makes the workflow more efficient and natural. Also, we try to keep more realistic expectations for our project from the beginning. Because of our unrealistic expectations and many methods on sprint 4, we were not able to implement all the features from the use case stories. Despite of that we are proud of what we have created on this project.

Bibliography

Anon. 2022a. *Theory X and Theory Y - Team Management Training from MindTools.com*. [online] Available at: <https://www.mindtools.com/pages/article/newLDR_74.htm> [Accessed 20 May 2022].

Anon. 2022b. *Who uses Jira? | Atlassian*. [online] Available at: <https://www.atlassian.com/software/jira/guides/use-cases/who-uses-jira#how-software-development-teams-use-jira> [Accessed 25 May 2022].

Digite, 2022. *What Is Scrum Methodology? & Scrum Project Management*. [online] Available at: <https://www.digite.com/agile/scrum-methodology/> [Accessed 15 May 2022].

Product Plan, 2022. *What is MoSCoW Prioritization? | Overview of the MoSCoW Method*. [online] Available at: <https://www.productplan.com/glossary/moscow-prioritization/> [Accessed 1 June 2022].