Aalto University
Ohjelmoinnin peruskurssi Y2

Jet Ski Reservation System Plan

Prepared by

Tatu Rouhiainen

tatu.rouhiainen@aalto.fi

100810721

Informaatioteknologia, first year student

February 24, 2023

Table of Contents

General Description and Difficulty Level

General Description and Backstory

In May 2022 I founded a jet ski rental business with my friends https://hetijetti.fi - Hetijetti vesijetti vuokraus. We started off with one jet ski in the Helsinki metropolitan area, got another friend to join in on the business and we bought one jet ski for the area of Lahti. I created a wordpress website with a very simple form that people could use to “reserve” a jet ski, but it looked and felt very cheap. In that part especially, there was room for improvement, but the summer season went well and the business was profitable.

Later in 2022, because of irrelevant reasons for this project plan, I sold my shares alongside a couple of my friends that left two people running the company. For the 2023 season the business is operating solely in Lahti, with two jet skis, and the current owners are planning on to expand and buy a third one. How does this all relate to the project, you might wonder. This company will be the not-so-hypothetical company of this project, that I will design a reservation system for. This backstory, the real life usage possibility and my personal connection to this brand and project topic is what makes my project unique.

Difficulty Level

I’m going for the most challenging difficulty as the challenge excites me and the hardest version would be the most functional and suitable for commercial use. A clear, usable and informative user interface plays a key role in customer satisfaction. The faster and easier the customer gets to maneuver through the reservation system, the more trust they will have on the brand and the more likely they are to enjoy their experience and recommend the brand to their friends.

For the reservation system, I plan on developing at least the following key features:

- A graphical user interface

- 3 different jet skis (products)

- Reservable times that are different length (1h, 2h, 4h (half a day), 8h (full day), weekend

- Price according to the reservation length (69€, 119€, 199€, 299€, 499€)

- Possibility to order additional services (life vests, wet suits, water toys, etc.)

- Reservation confirmation email

- Automatic reminder 24h before reservation starts.

DISCLAIMER: At least for now, this plan won't make any assumptions if the jet skis have different pricings or not. In real life it also is a question mark for the 2023 season. Usually the smaller or less powerful the jet ski is, the cheaper it is to rent.

Use Case Description of the User Interface

User - Program communication

The user communication with the program should be made as simple as possible for several reasons. The easier and the simpler the communication is, the faster, more pleasant and smoother the customer experience will be. It encourages impulse buying or in this case impulse reserving.

To achieve maximum simplicity, the user will communicate with the program by clicking different options shown to the user (hand drawn pencil sketches included below). The only user input by typing will be when they fill their own personal details.

- User selects the jetski that they want to rent
- User selects the date by clicking a date from a calendar
- User selects the reservation length by clicking one of the five radio options
- User selects the start time of the reservation by clicking one of the free starting times
- User selects the additional services by clicking all the boxes of the services you want
- User fills in his/her/their personal details by filling in the respective fields.
- User confirms the reservation by clicking a confirmation button at the very end, after checking that the order details are correct and every required field is satisfied with correct information

DISCLAIMER: I’m not good at drawing, and these figures only give a very rough visualisation of the UI. A color palette for the brand already exists, and this booking calendar will inherit the Hetijetti brand’s styles, fonts and colors. These figures are meant to give an idea of the functionality of the booking calendar. Rough pencil sketches of how I vision the user interface:

The flow of the reserving is split into six pages:

FIRST PAGE

User chooses the jetski they want to rent. There are at least two jet skis in real life, but a possibility of a third. This figure is made with the assumption there will be three. This doesn't include the ability to select and reserve multiple at the same time.

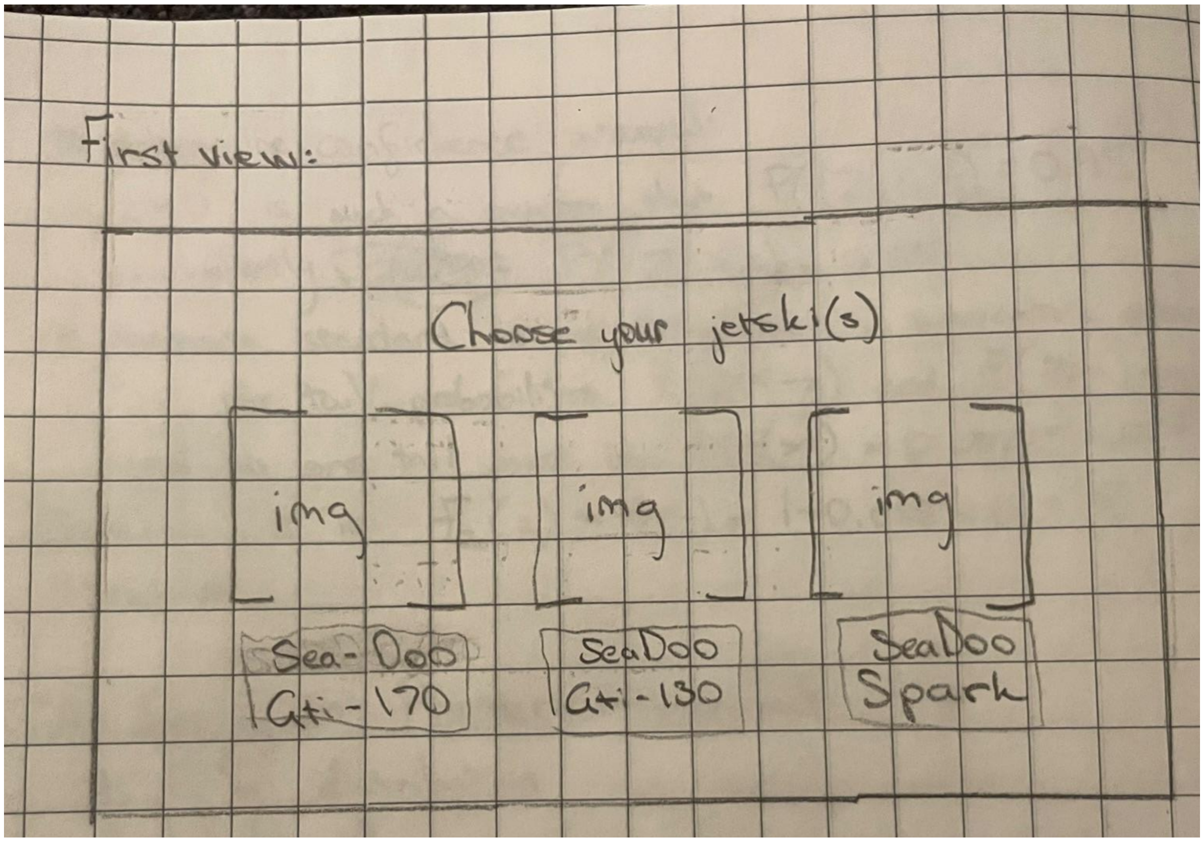


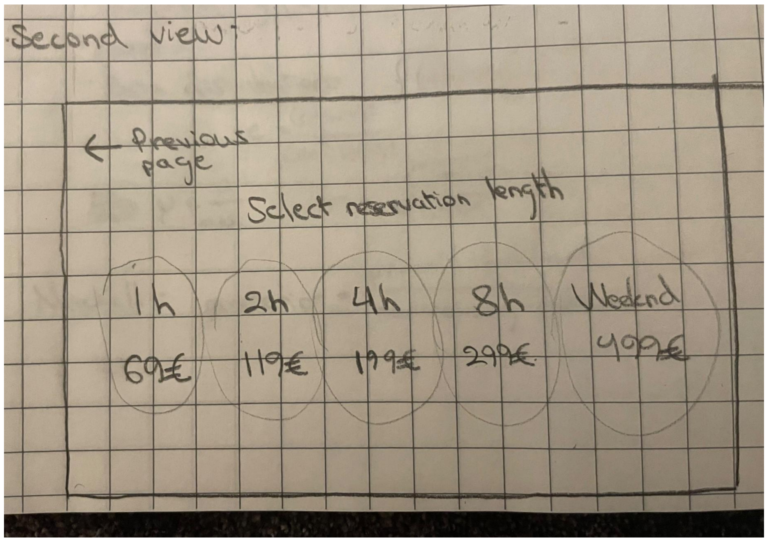
Image is a png / transparent image or a figure of the jetski in question

Below is a label for the image, the model of the jetski

Both Jetski label and figure will act as a button to select by clicking to move on to the next page

SECOND PAGE

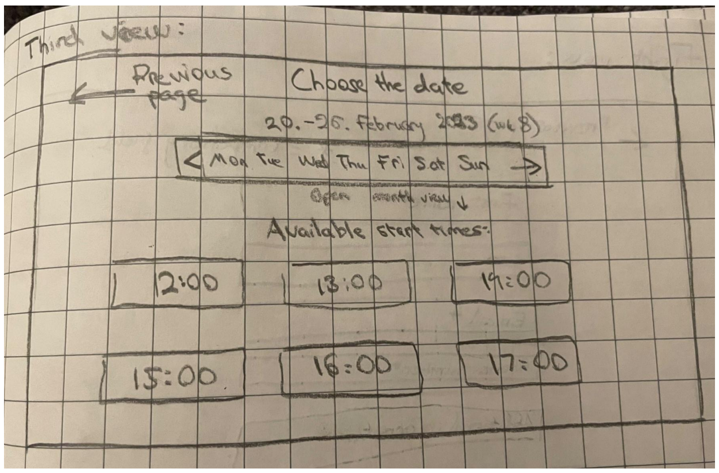
User chooses the reservation length out of the given options by clicking the length. Also note that from this page on the user has an ability to go back to the previous page to edit his/her/their selections.



THIRD PAGE

User chooses the date and the time. In my head it makes sense to keep these two selections on one page. Once the user selects a date, it will automatically display the free starting time slots for that day, for that jetski.

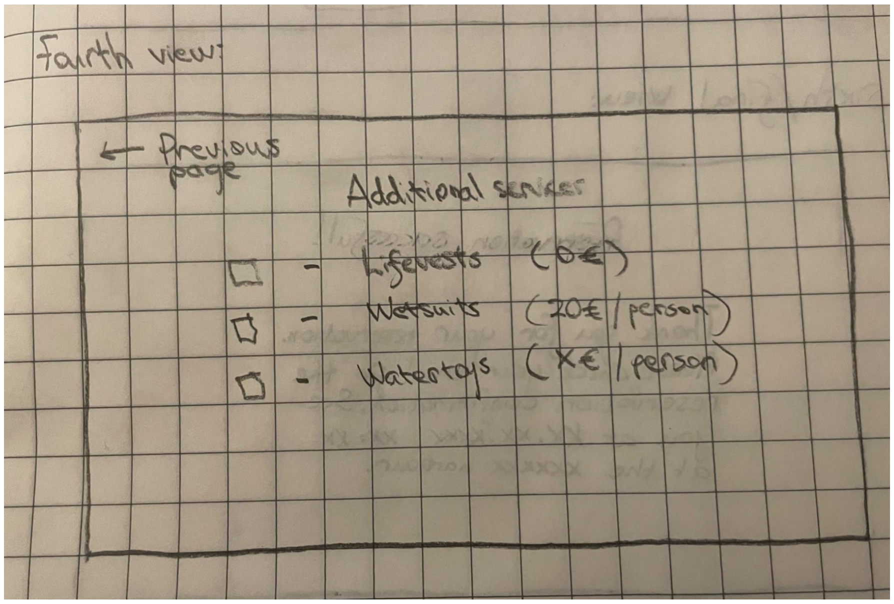
Figure is MISSING the button to move on to the next page. The button will most likely be in the top right corner but that is more of a UI question than a hard programming question



FOURTH PAGE

Users will get to select the additional services they want with their rental. Here is a couple to demonstrate functionality (ability to select multiple), but will not be the actual services. In reality life vests are always included automatically without extra charge.

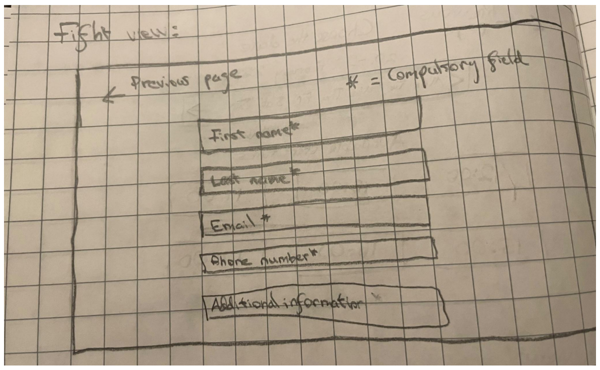
Figure is MISSING the button to move on to the next page. The button will probably be in the top right corner, but that is more of a UI question than a hard programming question.



FIFTH PAGE

The user fills in his personal information. The sizes will be altered, as the additional information requires a bit more space than the other fields. A * symbol represents that the field must be filled to move on to the next page.

Figure is MISSING the button to confirm reservation. The button will either be in the top right corner or bottom center, but that is more of a UI question than a hard programming question.

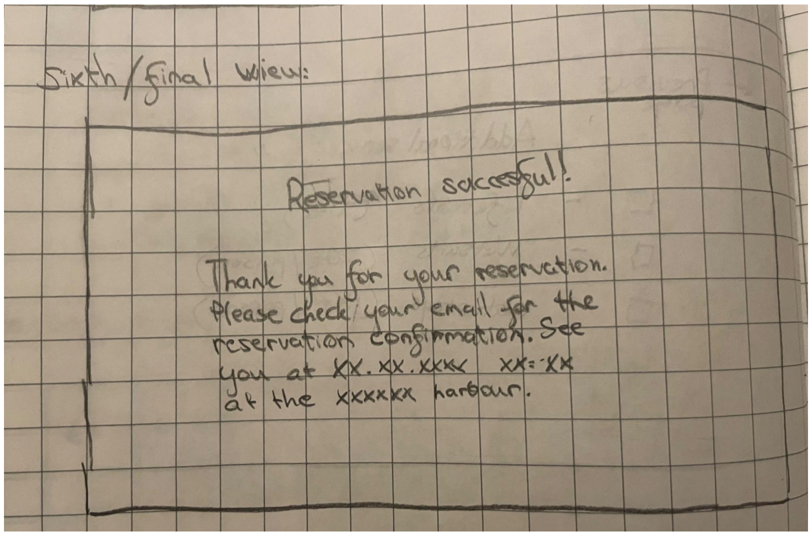


LAST MINUTE EDITOR’S NOTES:

A screen between the fifth and the sixth screen showing the confirmation details with the total price and the personal information with a big confirm reservation button that takes the customer to the last page. I don’t know how I did not think about this earlier, but that will most definitely be added into the project.

SIXTH PAGE

Reservation confirmation page, some text and some imagery most likely. Also instructions on how to move on from here: go check email / give a google maps link to the location where the reservation starts / possibility to add the event to their google calendar / etc.



Thank you for your reservation. Please check your email for the reservation confirmation. See you at XX.XX.XXXX XX:XX at the XXXXXX harbour.

Program’s structure plan

Here is a brief idea of the classes I am planning to implement to the project for the functionality of the project:

Class: Reservation

- This class represents a single reservation made by a customer. It will contain the information about the start time, end time, reservation length, price, any additional services ordered by the customer, but also the jetski(s) the customer reserved.

Class: Customer

- This class represents a customer who makes a reservation. It will contain information about the customer's first name, last name, email and phone number.

Class: JetSki

- This class represents a single jet ski available for rental. It will contain information about the jet ski's ID, availability and any additional information about the jet ski that may be relevant to the reservation process.

Class: Inventory

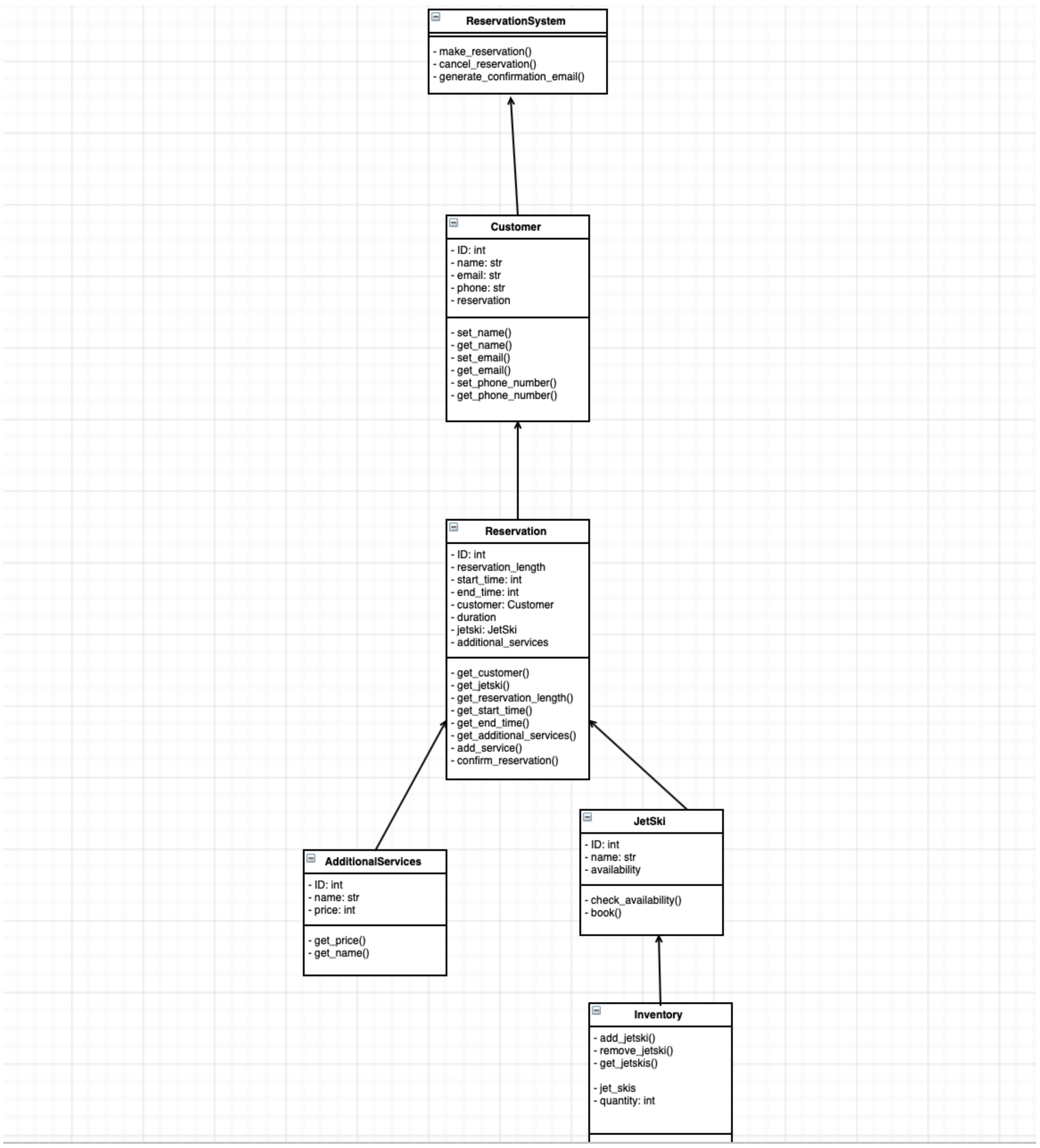
- This class represents the inventory of jet skis available for rental. It will contain a list of JetSki objects and methods for adding, removing and updating the inventory.

Class: AdditionalServices

- This class represents the additional services to add to the reservation. It will contain a list of services with an ID, name and price.

Class: ReservationSystem

- This class represents the reservation system itself. It will contain methods for making a reservation, cancelling a reservation and generating a confirmation email.


ReservationSystem
- make_reservation()
- cancel_reservation()
- generate_confirmation_email()
Customer
- ID: int
- name: str
- email: str
- phone: str
- reservation
- set_name()
- get_name()
- set_email()
- get_email()
- set_phone_number()
- get_phone_number()
Reservation
- ID: int
- reservation_length
- start_time: int
- end_time: int
- customer: Customer
- duration
- jetski: JetSki
- additional_services
- get_customer()
- get_jetski()
- get_reservation_length()
- get_start_time()
- get_end_time()
- get_additional_services()
- add_service()
- confirm_reservation()
AdditionalServices
- ID: int
- name: str
- price: int
- get_price()
- get_name()
JetSki
- ID: int
- name: str
- availability
- check_availability()
- book()
Inventory
- add_jetski()
- remove_jetski()
- get_jetskis()
- jet_skis
- quantity: int

Here is a brief idea of the classes that describe the UI of the program. Disclaimer, UI is still very unknown to me and I have only a very small understanding if it requires this many classes or not to develop my vision of having different screens for each part of the reservation process.

Class: JetSkiScreen

- This class represents the first screen of the reservation system, where customers can select a jet ski for rental. It will contain buttons for each available jet ski, that takes the user for the next screen.

Class: ReservationLengthScreen

- This class represents the second screen of the reservation system, where customers can select a reservation length. It will contain buttons for each available reservation length, that take the user to the next screen, as well as a “Previous Page” button to go back to the previous screen.

Class: DateSelectionScreen

- This class represents the third screen of the reservation system, where customers can select a date for their reservation. It will contain a calendar widget for selecting the date, as well as the available time slots displayed as selectable buttons for the respective date. It contains a “Confirm Date Selection” button to move on to the next screen and a “Previous Page” button to go back to the previous screen.

Class: AdditionalServicesScreen

- This class represents the fourth screen of the reservation system, where customers can select any additional services they want. It will contain checkboxes for each available service and a “Next” button to move on to the next screen, as well as a “Previous Page” button to go back to the previous screen.

Class: CustomerInformationScreen

- This class represents the fifth screen of the reservation system, where customers can enter their personal information for the reservation. It will contain input fields for the customer’s first name, last name, email and phone number as well as an open text field for additional information they might want to add to their reservation. E.q “for a birthday”, “bachelor party”, whatever it might be.

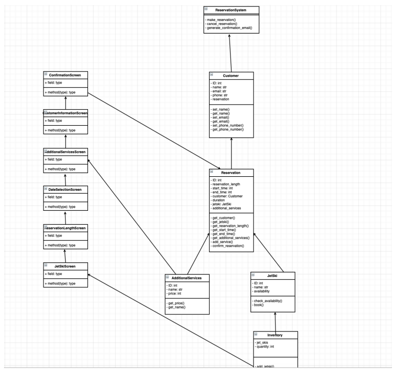
Class: ConfirmationDetailsScreen

- This class represents the second to last screen of the reservation system, where customers can view the confirmation details along with the total price of the reservation. There will be the customers last chance to view that the details they input were correct, and go back and edit their selection or personal details. There is a confirm reservation button that takes the customer to the last screen.

Class: ConfirmationScreen

- This class represents the final screen of the reservation system, where customers can view a confirmation message indicating that the reservation was successful if the confirmation email was sent successfully to the customer’s email and the confirmation went through in the code. It could display the reservation info for the customer, say that the same information was sent to their email address successfully and provide some instruction moving forward.

To be completely honest, I have no idea how the relationships between these UI classes are displayed on a UML diagram, or what variables or methods they include. I have yet to start the PyQT section or Round 6. I imagine that each class will get passed the information from the previous screen, until in the confirmation screen a customer variable and a reservation is created. Adding these UI classes the UML diagram could look something like this:



Data Structures

Lists

Lists can be used to store the JetSki objects in the Inventory class, as well as the list of reservations made by a customer in the Customer class. Lists are mutable, which means that their size can be changed, which is useful because new JetSkis objects could be added or removed from the inventory or an existing customer makes a new reservation.

Dictionaries

Dictionaries can be used to store the reservation data in the ReservationSystem class. A dictionary can store the reservation data for a single reservation with keys representing the reservation ID, start time, end time, customer information, jet ski information, reservation length, additional services and price. Dictionaries are useful for storing and retrieving data based on a unique key, such as the reservation ID.

Sets

Sets can be used to store the list of available reservation lengths and additional services in the JetSki and Reservation classes. Since sets only allow unique elements, they can ensure that each reservation length and additional service is only listed once.

Tuples

Tuples can be used to store the start and end times of a reservation in the Reservation class, as well as the date selected by the customer in the DateSelectionScreen class. Since tuples are immutable, which means that their size and values can't be changed, they can ensure that the start and end times of a reservation are not accidentally modified.

Linked lists

Linked lists could be useful for storing data that needs to be frequently added or removed from the the middle of the list, but could be very complex to implement as I don't have too much experience working with linked lists.

Arrays

Great for storing data that needs to be accessed randomly and quickly, but their size is fixed at the time of creation and adding or removing elements can be slow and inefficient.

Hash tables

Hash tables are useful for quick access to data based on a key, but can be complex to implement as I don't have too much experience working with hash tables. Hash tables also don't maintain order so that might limit the usage possibilities in the project.

Files and File Formats

The reservation system might require storing and loading data from files, such as customer information, jet ski inventory and reservation details. We will need images to build a clean, simple and effective user interface.

PNG, JPEG

We will store images as PNG or JPEG files depending on the purpose. PNG stands for Portable Network Graphics, and PNG images can have a transparent background or in general have transparent pixels in the image. For example the icons or a transparent figure of the jet ski could be stored as a PNG or perhaps the logo of the brand. JPEG stands for Joint Photorgraphy Expert Group, and JPEG images usually take less space, and in that way make the program more efficient as the smaller the file size is, the faster it loads. Backgrounds or other images that don't require transparency could be stored as JPEGs.

CSV

Customer information can be stored in text files, with each record separated by a delimiter such as a comma. A "fancier" file for this type of data could be a CSV file, which stands for comma separated values. CSV files also can be imported and exported to Excel or Google Sheets for example. The fields in a CSV file will include the customer data that is collected on the personal details page. It will at least include their unique id, first name, last name, email, phone number and a reservation id.

We also could store reservation details to a CSV file that contains the fields of the data related to reservations. This would mean the reservation id, customer id, price, date, start time, end time, additional services, jet ski and more data that we might want to include in the reservation.

Binary files

Binary files could offer a lot of efficiency, but my personal knowledge of working with them is close to none so they don't seem like a logical choice. Binary files could be useful for storing images or the reservation data for fast access and minimal disk space usage. The data could be stored in a binary format that uses fixed-length records, where each record contains all the information for a single reservation. This could make it easier to read and write the data to and from the disk. In general, the use of binary files can help reduce the size of the program and improve performance.

Algorithms

The algorithms chosen for this project are designed to ensure that the reservation process is efficient and user-friendly, while also providing customers with the necessary information about their reservation and any associated costs. The implementation of these algorithms may involve the use of various data structures (as described above) but the goal is to create a seamless user experience that meets the needs of both business and its customers.

Reservation Price Calculation Algorithm

If the prices were based on a set hourly price, the total cost could be calculated with the length of the reservation and the hourly price, but in this company they are not. The prices are set for each time length individually, to make longer reservations more worthy and encourage customers to reserve longer times.

The reservation price calculation algorithm will be very simple, but it will use the set price according to the length of the reservation, according to the jet ski and the additional services selected to calculate the total cost of the reservation. The formula will be something like:

$$Total\ Cost\ =\ (JetSki\ Rental\ Price\ for\ The\ Selected\ Length)\ +\ (Additional\ services\ cost)$$

This algorithm will be used in the ConfirmationDetailsScreen class to display the final price to the customer before confirming the reservation.

Calendar View Algorithm

The calendar view algorithm will display a graphical user interface showing each day of the week, or month, that the customer can rent the jet skis. According to the selected date, the available times for the selected jet ski will be displayed. This algorithm will use the data stored in the Inventory class to determine the availability of each jet ski for each day of the week and will display this information on the screen. Customers will be able to click a specific start time for their previously selected reservation length. The algorithm will need to be able to calculate with the reservation length the possible start times. If there is an existing reservation from 18-20 and the current customer has selected a 4 hour reservation, the possible start times could be 10am, 11am, 12am, 13am, but no later than 13 am.

This algorithm has to take into consideration the following rules:

- Earliest possible start time is 10 am
  (the business owners probably don't want to wake up earlier)

- Latest possible reservation end time is 10pm
  (the business owners probably don't want to pick up the jet skis any later than that) This means that a 4 hour reservation cannot start any later than 6 pm for example.

- There must be a 1 hour gap between every reservation
  (The business owners have to handle paperwork with the new customer before their booking starts, introduce them to the jet skis and handle payment in real life. All of this is done outside the reservation length.)

- Same day reservations are not possible through the reservation system. Instead, the reservation system should instruct them to make them through a phone call.

Automatic Reminder Algorithm

The automatic reminder algorithm will send reminders to customers before their scheduled reservation time to remind them of their reservation and ensure they do not forget, and it is a great possibility to remind them to be in the jet ski rental place 20 minutes before the reservation starts. This algorithm can be implemented using a simple timer function that sends an automated email or a text message to the customer a specified amount of time before their reservation. The algorithm could help to reduce no-shows and enhance the customer experience. The reminder is sent 24 hours before the start time. If there is less than 24 hours between the time of the booking and the start time of the reservation, automatic reminder is not sent.

Artificial intelligence - hypothetical

In an alternate universe, where I would know how to work with all of these following aspects of programming I will talk about, we could make a machine learning algorithm to predict customer behavior and preferences, allowing the system to recommend specific additional services based on past customer data. Natural language processing algorithms could also be used to improve the customer experience by allowing customers to make reservations using voice commands. This would allow blind people to make reservations easier, but I guess blind people wouldn't also drive jet skis.

Testing Plan

To test the final program at the system level, we need to consider the central functionality of the program and what it must be able to do. Right now, my mind could think of a zillion different methods to test the various parts of the program, but we will focus on a few of the most central parts of the functionality. The program's core functionality will most likely involve reading and parsing text / csv files, searching for text within those files and displaying the results to the user. Therefore, some of the testable properties that could be tested for the final program include:

Add Reservation

This method tests adding a new reservation to the system. Some test cases for this method could be:

Case 1: A valid reservation object with all required fields

Case 2: An invalid reservation object with missing or incorrect fields

Case 3: A reservation that conflicts with an existing reservation (same time slot, jet ski not available)

Expected output for each input respectively would be:

Case 1: The method should add the reservation to the system and update the relevant jet ski's availability status

Case 2: The method should not add the reservation to the system and should return an error message

Case 3: The method should not add the reservation to the system and should return an error message

Get Available Start Times

This method returns a list of available start times for a given date, reservation length and jet ski. Some test cases for this method could be:

Case 1: A valid date and length with available start times

Case 2: A valid date and length with no available start times

Case 3: An invalid date or length

Expected output for each input respectively would be:

Case 1: The method should return a list of available start times

Case 2: The method should return an empty list

Case 3: The method should return an error message

Calculating The Total Price

This method calculates the price of the total price for a reservation and returns the total price. Some test cases for this method could be:

Case 1: Empty list of jet ski object's, rental duration of 1 hour

Case 2: List of 1 jet ski object, rental duration of 1 hour

Case 3: List of 2 jet ski objects, rental duration 1 hour
(Two jet skis might not be possible to reserve in my program, even though in real life this is very much a possibility, but this would test it anyways)

Expected output for each input respectively would be:

Case 1: The method should return an error message

Case 2: The method should return 119

Case 3: The method should return 238

Creating A Customer Object

This method creates a new customer object. Some test cases for this method could be:

Case 1: Empty dictionary of customer data

Case 2: Dictionary with all required information: id, name, email, phone number

<u>Case 3</u>: Dictionary missing some required information or invalid information (like wrong formatted email)

Expected output for each input respectively would be:

<u>Case 1</u>: The method should return an error message

<u>Case 2</u>: A new customer object with the provided information is created and added to a csv file

<u>Case 3</u>: The method should return an error message

Libraries and Other Tools

PyQt

As much as I am able to do with PyQt, I will try to do it. I don’t have much knowledge of other libraries or python libraries in general so for learning purposes it might be best to focus on one. Researching about PyQt also made me confident that I will not need to rely on other libraries.

I would primarily use the PyQt library for building the user interface. PyQt is a Python binding for the Qt toolkit, which provides a comprehensive set of GUI libraries and tools for building desktop applications.

With PyQt, I can easily create windows, buttons, text boxes and other GUI elements needed for the jet ski reservation system. I can also define the functionality of these GUI elements using event-driven programming, where user actions such as button clicks can trigger specific functions to execute.

Schedule

Based on the requirements, checkpoints and the 80h that is budgeted for this project, here is a tentative schedule for the development of the jet ski reservation system.

Phase 1: Planning and Design (Estimated time: 10 hours)

- Understanding project requirements and specifications	1h
- Identifying necessary features and components of the system	1h
- Planning the implementation of the system's classes and methods	1h
- Sketching the user interface	1h
- Creating this plan that you are reading right now	∞h

Phase 2: Implementing Core Functionality (Estimated time: 40 hours)

- Creating and implementing the customer class	3h
- Creating and implementing the jet skis and inventory classes	2h
- Creating and implementing the reservation class	5h
- Implementing the methods for each class	20h
- Building the reservation system's workflow, from the first page to the order confirmation page	10h

Phase 3: User Interface Implementation (Estimated time: 20 hours)

- Developing user interface components using PyQt	5h
- Implementing logic to connect UI components with system functions	10h
- Creating a great user experience for the customer through a simple but functional user interface, that is pleasant to look at	5h

Phase 4: Testing and Debugging (Estimated time: 10 hours)

- Running unit tests on system components and methods	2h
- Manually testing the system's workflow and user interface	2h
- Debugging and fixing issues that arise during testing	6h

Here is how the phases will be completed in the context of dates and the two checkpoints (1st checkpoint: 24th of March 2023, 2nd checkpoint: 14th of April 2023)

Phase 1: 20th of February 2023 – 3rd of March 2023
Phase 2: 4th of March 2023 – 23rd of March 2023 | CHECKPOINT 1
Phase 3: 24th of March 2023 – 14th of April 2023 | CHECKPOINT 2
Phase 4: 15th of April 2023 – 30th of April 2023
FINAL DEADLINE: 12th of May 2023 at 14:00

Literature References and Links

hetijetti.fi website developed by me, the not-so-hypothetical company of this project
https://hetijetti.fi

A+, Y2 course material
https://plus.cs.aalto.fi/y2/2023/?hl=en

MyCourses, Y2 course material
https://mycourses.aalto.fi/my/

UML class diagram creation
https://gitmanager.cs.aalto.fi/

w3schools, general knowledge about python syntax and OOP
https://www.w3schools.com/python/

Realpython, reading about data structures
https://realpython.com/

PyQt general information
https://riverbankcomputing.com/software/pyqt/

PyQt5 Documentation
https://www.riverbankcomputing.com/static/Docs/PyQt5/

Wikipedia, what the abbreviations PNG, JPEG, CSV stand for
https://www.wikipedia.org

Techtarget, article about binary files
https://www.techtarget.com/whatis/definition/binary-file

Python documentation
https://www.python.org