Candidate number: 15

MH130

User Interface Programming and Architectures

Individual Written Home Examination

Kristiania University College

Smart Parking Reservation System for Majorstuen

Fall 2024

15.11.2024

Table of Contents

List of Figures

UML-Diagram

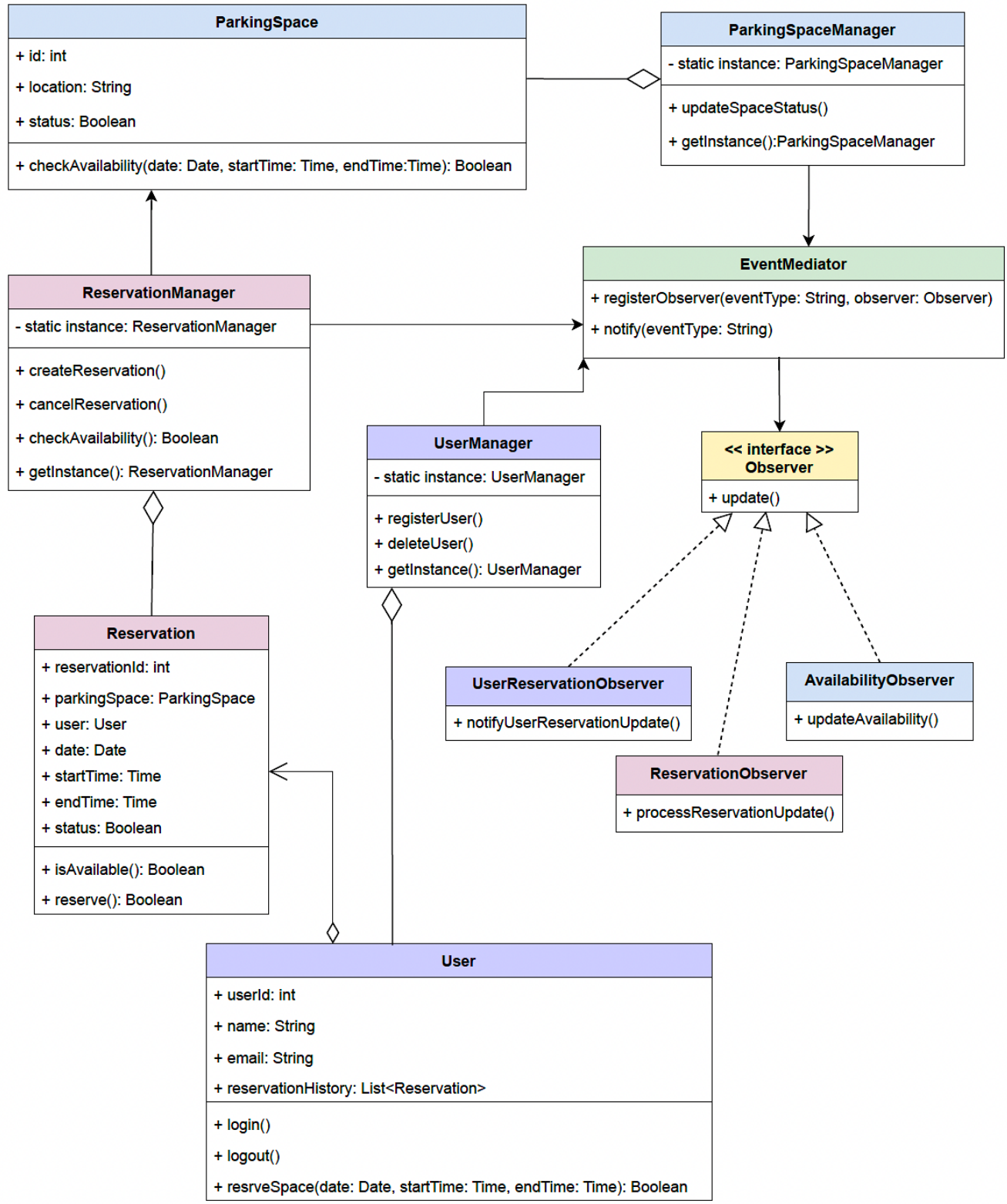

Figure 1: UML Class Diagram

The class diagram illustrates the use of an event-driven architecture with a mediator topology. The design patterns employed are Singleton and Observer.

Parking Components

The parking functionality is represented by the classes ParkingSpaceManager, ParkingSpace, and AvailabilityObserver. The ParkingSpace class models a parking space with attributes for location and status, and includes the checkAvailability() method to determine if a parking space is available at a given time, which is essential for preventing double-booking. ParkingSpaceManager is a Singleton class responsible for managing ParkingSpace objects, including real-time status updates through the updateSpaceStatus() method. When the status of a parking space changes, ParkingSpaceManager notifies EventMediat*or*, which then forwards the update to AvailabilityObserver. This observer pattern ensures that the system is dynamically updated and distributes real-time parking availability information across the system, contributing to a reliable, up-to-date view of available parking spaces.

Reservation Components

The reservation system is represented by the classes ReservationManager, Reservation, ParkingSpace, and ReservationObserver. The Reservation class represents an individual reservation, with methods such as isAvailable() to check if a reservation is possible and reserve() to book the space if available. ReservationManager is a Singleton class that handles the creation and cancellation of reservations, checking parking space availability with checkAvailability() before a reservation is made to avoid double-booking. By notifying EventMediator of changes (such as new reservations or cancellations), ReservationManager ensures that ReservationObserver and UserReservationObserver are updated. ReservationObserver manages system-related updates about reservations, while UserReservationObserver handles user-related notifications, such as reservation confirmations or cancellations.

User Account Components

User account management is represented by the classes User, UserManager, and UserReservationObserver. The User class models a system user with functions such as logging in, logging out, and using the reserveSpace() method to reserve parking spaces. UserManager is a Singleton class responsible for creating and deleting user accounts. UserReservationObserver receives updates from EventMediator regarding changes in user reservations and can, for instance, send notifications to users about updates in their reservation status. This structure provides an efficient and seamless interaction between the user and the reservation system, while ensuring that user information is kept up to date.

Report

Discussion of the Solution

The Oslo City Council desired a modern design and a robust system in the process of reserving parking spaces at Majorstuen. The case required implementation of some essential key functions, such as an efficient and user-friendly search function, the ability to reserve parking spaces and review reservations afterward, as well as preventing misunderstandings and user errors. The solution addresses the various key functions and requirements set by the Oslo City Council. The solution offers that users can locate available parking spaces through a search field, and it provides informative messages throughout the reservation processes to minimize user errors. Additionally, the system addresses the need to prevent double-booking, ensuring users cannot reserve the same parking space more than once.

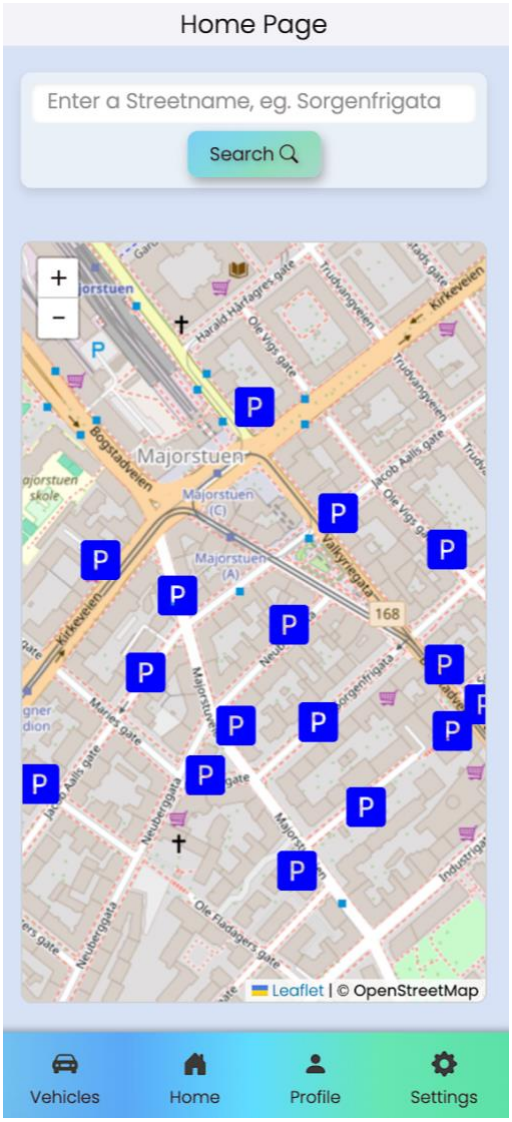

Figure 2: Home Page on an iPhone 12 Pro

To offer a modern interface, a map from OpenStreetMap has been implemented (as figure 1 illustrates above) alongside the JavaScript framework Leaflet, providing interactive functionality within the map. OpenStreetMap contributes detailed and updated map data [1]. Leaflet enables easy integration with features like zooming and pop-ups, making the map experience intuitive and responsive [2]. Together, Leaflet and OpenStreetMap enable an anticipated feature with a customized map view and real-time information updates, creating a dynamic and flexible interface for search, reservations, and navigation in the area. To allow users to search for addresses directly on the map, the geocoding service Nominatim is used to convert addresses into coordinates [3], allowing the search function to provide quick and efficient access to parking spaces in the area.

Bridging the Gulf of Execution and Evaluation

To ensure that users can effectively navigate the system and understand the outcomes of their actions, several measures have been implemented to bridge the Gulf of Execution and Evaluation. To reduce the gulf of execution, established UI conventions and design patterns, as suggested by Ko [4], have been implemented to enable users to achieve their goals intuitively. To minimises the gulf of evaluation, the system is designed to provide continuous and meaningful feedback [4].

Search Function

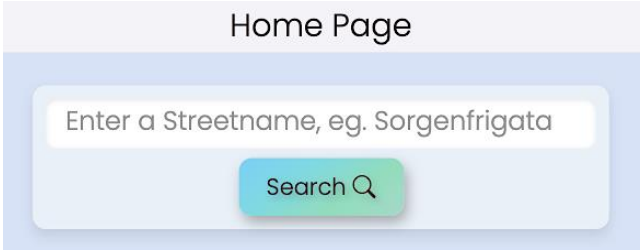

Figure 3: The Search Field

The main page features a prominent search field above the map, providing an intuitive way for users to start their search process. The search field has a clear affordance [5], since it affords the users to enter and search for a desired location, as shown in figure 2. The system is designed to be understandable with familiar symbols such as the magnifying glass on the search button, placeholder text in the search field, and the appearance of the text cursor. These elements act as signifiers [5], directing users towards the actions they need to take to find available parking spaces. After the search button is clicked, the system provides confirmatory feedback in the form of either a result on the map or an informative message indicating that no parking spaces are available at the searched address. The search results are displayed on the map as available or occupied parking spaces,

represented by blue icons for available and burgundy for occupied. This clearly signals the search outcome, helping users make informed decisions without delay and bridging the gulf of evaluation.

Parking Space Reservation

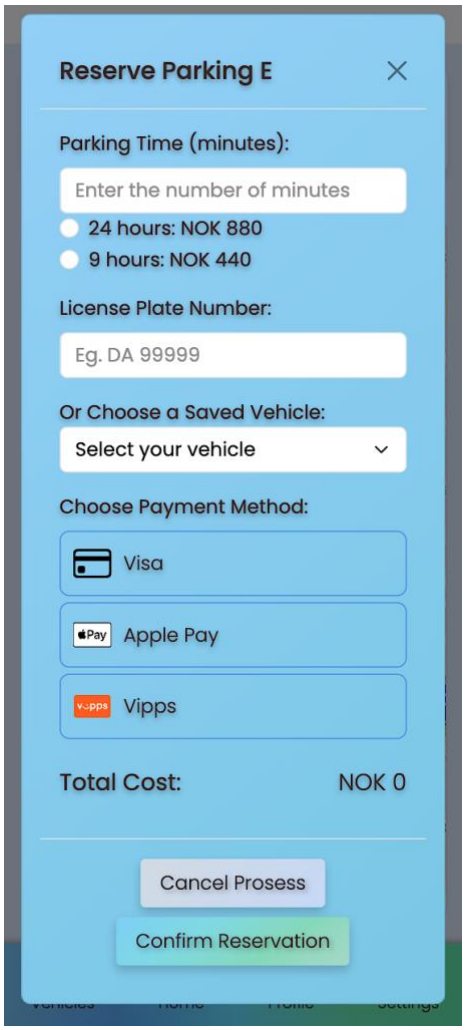

Figure 4: The Reservation Form of a Paring Space

As figure 4 shows, the system is designed to easily select time slot, vehicle plate, and payment method through a form in a modal window. Users can fill in information in input fields or choose preset options via radio buttons and a dropdown menu of saved vehicles, providing flexibility in interaction and clear affordances that indicate how these elements can be used. Visual effects, such as a colour change on hover and the pointer cursor function as signifiers. These features make it easy for users to navigate and understand the reservation process. The system provides feedback at each step of the reservation process to prevent the gulf of evaluation. If users attempt to reserve an occupied space, a message clearly indicates that the space is unavailable. After the “Confirm Reservation” button is clicked, an overlay displays a confirmation of the reservation. Additionally, a brief alert message provides immediate feedback on the success of the action. This gives users a

sense of control and understanding throughout the process, assuring them that the reservation is finalised.

Overview of Reservations

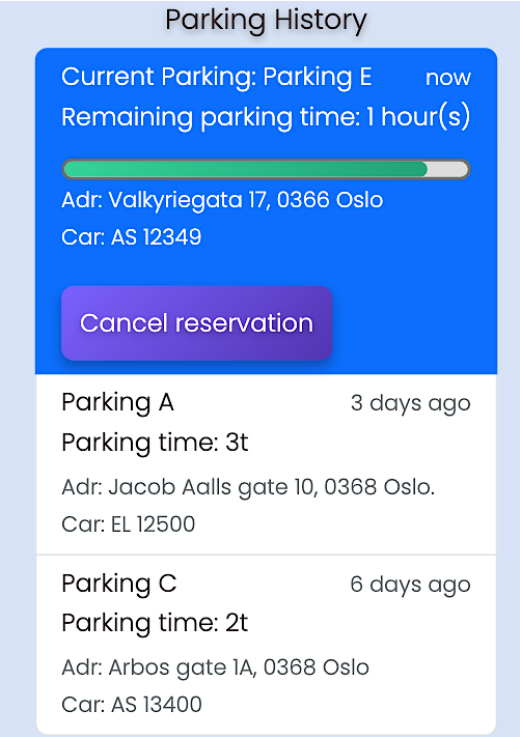

Figure 5: Parking History

The profile page affords an overview of active and past reservations, giving users control over their parking history, as figure 5 demonstrates. The 'Cancel Reservation' button provides a clear affordance for cancellation, with a hover effect that includes shadow and enlargement as signifiers of clickability. The X-icon on the reservation overlay affords an option to close the view without cancelling, offering users a choice in their interaction. After cancelling a reservation, a confirmation message appears, and the parking icon reverts to blue, signifying availability. These affordances and signifiers ensure users can perform desired actions and understand outcomes, creating a cohesive, user-friendly experience.

User Interface Design Choices and Universal Design Principles

To provide a user interface that incorporates universal design, ensuring the GUI is accessible to as many users as possible regardless of functional ability, the 7 principles of universal design have been implemented [6], and complemented by WCAG 2.2 principles for enhanced web accessibility [7]. By integrating principles from both universal design and WCAG, the application is designed to be human-centered, user-friendly, and inclusive.

Equity in Use

The app ensures equal accessibility for all users by implementing ARIA roles and attributes, making it compatible with screen readers and beneficial for people with diverse abilities. The interface offers equal opportunities for navigation and functionality, allowing sufficient time to read and interpret content, such as alert messages. The navigation structure is clear, with descriptive page titles and icons to assist users. This conscious decision aligns closely with WCAG guidelines 2.2 and 2.4, which focus on making content both operable and user-friendly [7], and supports the principle of providing an inclusive and accessible experience with appealing design [6].

Flexibility in Use

The reservation process allows users to enter a license plate manually or select from saved vehicles. Users can customize the interface to their preferences, including language and dark/light mode. Offering multiple ways to access the same function is a deliberate choice, enhancing navigation flexibility and addressing guidelines 2.a in [6] and 2.4 in [7]. The app aligns with the Robust principle in [7] by supporting screen readers through ARIA attributes and semantic HTML. A critical aspect is that the use of Bootstrap for a modern and responsive design [8], somewhat diminishes semantic HTML. This choice makes the UI more visually consistent and accessible, but it may come at the expense of comprehensibility for screen readers.

Simple and Intuitive Use

The app has a clear layout and consistent design, with familiar icons and simple instructions that make it accessible to all users, regardless of knowledge or experience level, in line with the principle's guidelines [6]. For example, the search button features a clear search icon next to the text "Search". To prevent misuse, input fields include clear examples, ensuring information is easily comprehensible. These deliberate design choices make the application intuitive and meet the WCAG principle of understandability [7].

Perceivable Information

To adhere to principle 4 in [6], information is presented clearly and in multiple formats, such as icons with explanatory text, high-contrast colours for readability, and both visual and textual feedback when a parking space is reserved. The use of ARIA attributes further ensures accessibility for visually impaired users. Together, these design choices support the WCAG principle of perceivability [7], enabling individuals with cognitive and sensory challenges to use the app effectively.

Error Tolerance

The app minimizes user errors by following the guidelines in [6], incorporating clear error messages and 'fail-safe' features to discourage unintended actions. For instance, if a user attempts to reserve an occupied space, a warning prevents double booking. In the reservation form, users can select only one method to set the time, either manually or through radio buttons. This intentional design aligns with the WCAG principles of understandability and operability, making it easier for users to correct mistakes [7].

Low Physical Effort

The app is optimised for ease of use, requiring minimal clicks to reserve a parking space. The map automatically zooms in on the searched area, eliminating the need for manual adjustment, and simplified options allow quick completion of the reservation form. This design benefits users with reduced mobility but may limit those preferring more control. Overall, this choice makes the app faster and easier to use for the majority, though there is a potential compromise between simplicity and control. This sixth design principle, supports the WCAG operability principle by reducing physical effort, especially advantageous for users with reduced mobility.

Size and Space for Approach and Use

Large buttons and adequate spacing between elements ensure that the app is easy to operate, even for users with motor challenges. Proper design provides good sightlines and reach to all components, making navigation comfortable. This reflects the seventh principle in [6] and supports WCAG requirements for both perceivability and operability [7], ensuring that all users can navigate and operate the app with ease.

System Architecture and Design Choices

The system architecture of this smart parking reservation system is based on an event-driven architecture with a mediator topology [9] and incorporates the Observer and Singleton design patterns from the Gang of Four design patterns [10].

The Singleton pattern is employed to ensure that only a single instance of the management classes exists [10]. This is critical in a system handling real-time data to prevent conflicting updates, data duplication, and to maintain consistency throughout the system. For instance, having multiple instances of 'ReservationManager' capable of creating and cancelling reservations independently, could lead to double-booking and data inconsistency.

The Observer pattern is used to implement event-driven communication within the system and is central to the event-driven architecture. The Observer pattern is ideally suited for this system as it enables dynamic updating of different parts of the system as changes occur [10]. This provides flexibility and facilitates system extensibility. For example, when a parking spot is updated or a reservation is created, observers automatically receive updates. If a new feature requires event notifications, a new observer can be added and registered with the 'EventMediator' without impacting the rest of the system, thereby establishing a loosely coupled system.

The mediator structure, which centralizes event distribution through the 'EventMediator', reduces dependencies between components, enhancing modularity and simplifying maintenance [9]. This architecture, utilizing the Singleton and Observer patterns within an event-driven mediator topology, creates a robust system for real-time updates of parking spots, reservations, and user management.

List of References

[1] OpenStreetMap, “OpenStreetMap,” n.d. [Online]. Available: https://www.openstreetmap.org/. [Accessed: October 28, 2024].

[2] Leaflet, “Leaflet,” n.d. *Leaflet - an open-source JavaScript library for mobile-friendly interactive maps*. [Online]. Available: https://leafletjs.com/. [Accessed: October 28, 2024].

[3] Nominatim, “Nominatim,” n.d. *Open-source geocoding with OpenStreetMap data*. [Online]. Available: https://nominatim.org/. [Accessed: October 29, 2024].

[4] A. J. Ko, J. O. Wobbrock, and E. Whitmire, “User Interface Software and Technology.,” 2024, ch. 2. [Online]. Available: https://faculty.washington.edu/ajko/books/user-interface-software-and-technology/.

[5] D. Norman, *The Design of Everyday Things*, New York: Basic Books, 2013.

[6] M. F. Story, “Maximizing Usability: The Principles of Universal Design,” *Assistive Technology*, vol. 10, no. 1, pp. 4-12, 1998. doi: 10.1080/10400435.1998.10131955.

[7] Web Accessibility Initiative (WAI), *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, 5 Oct. 2023. [Online]. Available: https://www.w3.org/TR/WCAG/. [Accessed: November 4, 2024].

[8] Bootstrap, “Bootstrap,” n.d. *Build fast, responsive sites with Bootstrap.* [Online]. Available: https://getbootstrap.com/. [Accessed: October 31, 2024].

[9] M. Richards, *Software Architecture Patterns*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015, ch. 2, “Event-Driven Architecture”.

[10] A. Shvets, *Design Patterns*. Refactoring.Guru, 2024. [Online]. Available: https://refactoring.guru/design-patterns . [Accessed: November 9, 2024].