TECHNIUM

*Technium Vol. 4, Issue 5 pp.26-38 (2022)*
*ISSN: 2668-778X*
*www.techniumscience.com*

# uPark: a progressive web parking application

**Konstantinos Panos, Evi Papaioannou, Christos Kaklamanis**

University of Patras and CTI "Diophantus", Greece
papaioan@ceid.upatras.gr

**Abstract**. We present uPark, a progressive web parking application for the online purchase of slots of pre-paid parking time, which remains associated to client accounts rather than bounded to particular parking spots or vehicles. uPark integrates three sub-systems, one for drivers, one for inspectors and one for administrators. It optimizes the overall parking process for citizens/clients from a financial and a practical point of view and technically facilitates the work of inspectors. It is a lightweight, easy-to-use, responsive application. Moreover, as a progressive web application, uPark can be used both as a web site and a mobile app on any device. uPark can be used for various areas, like city center, mall parking area or university campus, making a time- and effort-saving, eco-friendly and useful progressive web application.

**Keywords**. progressive web application, parking application, cost-benefit ratio for citizens, eco-friendly, open source, Node.js, JavaScript, TypeScript, PostgreSQL, Express, Leaflet.

## 1. Introduction

Parking cards form a practical means for dealing effectively with parking issues in populated urban areas. The fact that the vast majority of the population owns a portable smart device, e.g., a smartphone, wirelessly connected to the Internet coupled with several e-parking applications helped in facing to some extent several drawbacks of paper parking cards, related to issues like perishable material substance, ease of purchase, duration modification, validity checking, etc. E-parking systems have gained popularity for both general (e.g., city center) and specialized (mall parking areas) contexts. However, it is often the case that the set-up and use of such e-parking systems becomes too sophisticated, requires the use of special devices (e.g., for inspectors) and is usually limited to simulating the paper parking cards philosophy, in the sense that purchase of cards is made electronically rather than physically. Nevertheless, drawbacks related to the quality of the process as far as the maximization of the citizen/client benefit is concerned require closer attention and further improvement.

In this work, we present uPark, an online parking application which optimizes the overall process for citizens/clients both from a financial and practical point of view, while, at the same time, technically facilitating the work of inspectors. uPark utilizes a central database structure where all available parking spots of an area are kept (via their coordinates). Users registered to uPark can use any computing device (e.g., smartphones, tablets, desktop computers or laptops) to purchase slots of pre-paid parking time, associate it with parking spots and use it for typical vehicles (i.e., having a typical plate). However, this parking time remains associated to client accounts rather than bounded to particular parking spots or vehicles. uPark keeps track of the remaining purchased e-parking time per user account and makes it available for use by multiple physical persons and for multiple vehicles as long as the same uPark account is used. As far as inspectors are concerned, once registered to uPark,

they can carry out their task by simply using their smartphone with no requirement for any additional material or special equipment. uPark is a lightweight, easy-to-use, responsive application with a very attractive and user-friendly interface where several functionalities can also be directly performed on interactive maps. In addition, uPark is a progressive web application and, thus, can be used as a web site and mobile app on any device. Users can add uPark to their home screen and launch it from there as a top-level, full-screen application. uPark has been developed mainly using Node.js, a free, open-sourced, cross-platform JavaScript run-time environment that executes JavaScript code outside a web browser and TypeScript, a strongly typed programming language that builds on JavaScript. The uPark database has been built on PostgreSQL, an open source object-relational database system while the Express framework has been used for the communication with the uPark interface. React JavaScript library, Leaflet and OpenStreetMap have been used for the development of interactive map functionalities. uPark can be used to serve any particular area, like a city center, a mall parking area or a university campus, making a time- and effort-saving, eco-friendly and useful progressive web application.

Similar in spirit parking applications in Greece include myAthensPass (http://parkinathens.gr/) available in the city of Athens, ParkPal (https://thesi.gr/) which is part of the THESI parking system of the city of Thessaloniki, Flowbird (https://flowbirdapp.com) available in Municipality of Rafina and ParkAround (https://www.parkaround.gr/) used for private parking facilities. myAthensPass is a free mobile app available for Android and iOS devices that allows drivers to pay from their mobile device for parking spaces in controlled zones within the City of Athens. ParkPal is a free mobile app available for Android and iOS devices for vehicle parking in Thessaloniki. Flowbird is a free parking service available for Android and iOS devices; it can also be used as a website via a browser on desktop computers. In Greece, Flowbrd is available only in Rafina (a city in the East Attica region). ParkAround is a free parking service available for Android and iOS devices; it can also be used as a website via a browser on desktop computers. ParkAround works only for collaborating private parking agencies in Greece. It is currently available in several cities (i.e., Athens, Thessaloniki, Chalkida, Ioannina, Larissa, Agios Nikolaos, Agrinion) and points of interest (i.e., Athens and Thessaloniki airports). All these parking applications require user registration and support functionalities which include overview of parking spots on a map, exact location detection via GPS or via QR code scanning, selection of parking duration, instant purchase of parking time available also for later use, push notifications shortly before parking time expiration, extension of parking time remotely, access to previous transactions. However, all of them are available for particular locations, and, to the best of our knowledge, none of them has been built as a progressive web application. Furthermore, uPark can be easily used for any location as long as a plan of available parking posts per sector is provided.

The rest of the paper is structured as follows. In Section 2, we address in detail uPark design and implementation. In Section 3, we present a use scenario for demonstrating uPark subsystems and supported functionalities. We conclude in Section 4 also discussing future plans.

## 2. System design and implementation

In what follows we present design and implementation details. We also describe the user interface for each of the three uPark subsystems.

### *2.1. System design*

uPark can be used as a parking management application for all places for which available parking spots per sector is provided and stored in its database. Sectors can be streets or roads or any part of a place where there are predefined parking spots. uPark requires registration and supports three types of users, namely Drivers, Inspectors and Administrators, thus making an integrated solution for the whole parking process lifecycle. There is a separate user interface revoked according to the category a registered user belongs to. In particular, Drivers can access uPark at https://strong-taiyaki-58c81d.netlify.app/, Inspectors can access uPark at https://cerulean-druid-66a5aa.netlify.app/ and Administrators can access uPark at https://startling-tapioca-84e54e.netlify.app/.

Technium Vol. 4, Issue 5 pp.26-38 (2022)
ISSN: 2668-778X
www.techniumscience.com

uPark has been designed and implemented as a progressive web application (PWA) [1,2]. PWAs are web applications developed using particular technologies and patterns to allow them to exploit features from both web and native applications. A progressive web application is essentially a website, except that it can be "installed" on the user's device, giving the feel of a native application. This means that users can add uPark to their home screen and use as a website and mobile app on any device.

A progressive web application is built using the shell and the content as the two main structural components. The shell describes the infrastructure of the application and contains all the static files needed by an application to function. In the context of web development, such files include html, css, JavaScript and image files. The content refers to data that can change which is, thus, excluded from the application shell. This is because this data is dynamic and could potentially be already old when the application is loaded. Content is typically exposed via an API service and is easy to query. The service worker is responsible for managing this content in the application so that it is ensured that the freshest content is available upon request.

The implementation of uPark as a progressive web application offers an improved experience to users and developers. PWAs are characterized by speed, security and easy access, thus contributing to the improvement of the user experience. Users can easily access favorite apps by simply tapping their icons, rather than navigating to them using a browser. Moreover, PWAs are built using modern APIs in order to deliver enhanced capabilities, reliability and installability and to be reachable by anyone, anywhere, on any device while providing developers with a single codebase.

The design of uPark is outlined in Figure 1. uPark users, i.e., clients, who can be Drivers, Inspectors and Administrators access uPark via their personal device. User requests are processed by the server, which in turn retrieves the necessary information from the system database in order to display the necessary information to users.

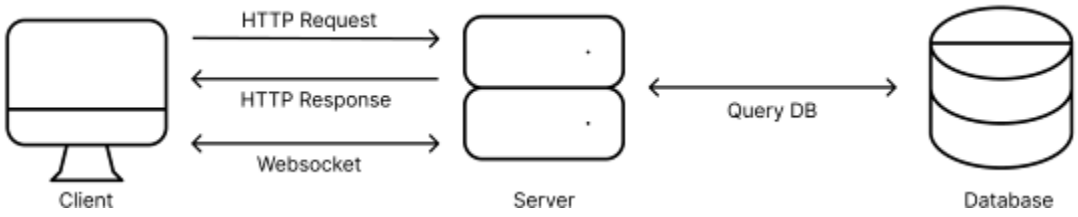



Figure 1: uPark PWA design

*2.1.1. Driver subsystem.* uPark home screen when accessed by a Driver is depicted in Figure 2. Drivers can sign up and create a new account or sign in to an existing account at https://strong-taiyaki-58c81d.netlify.app/.

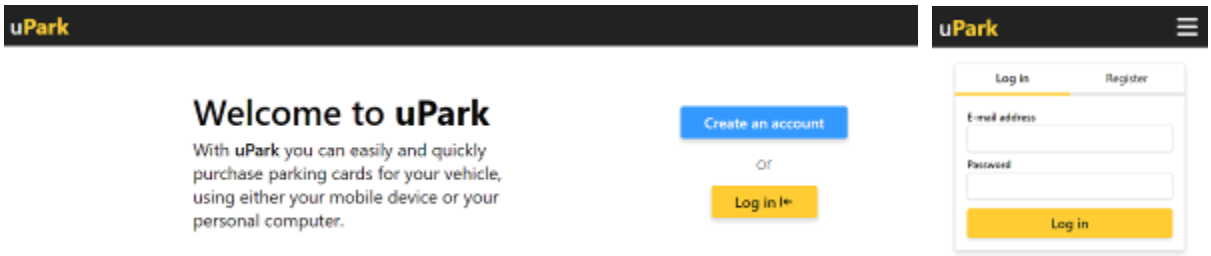



Figure 2. Home screen for desktop (left) and mobile users (right)

Technium Vol. 4, Issue 5 pp.26-38 (2022)
ISSN: 2668-778X
www.techniumscience.com

A successfully signed-in Driver is presented with the screen depicted in Figure 3 where apart from automatic position detection on an interactive map, the following functionalities are available: Home, Manage vehicles, Parking history, Settings, Log out..

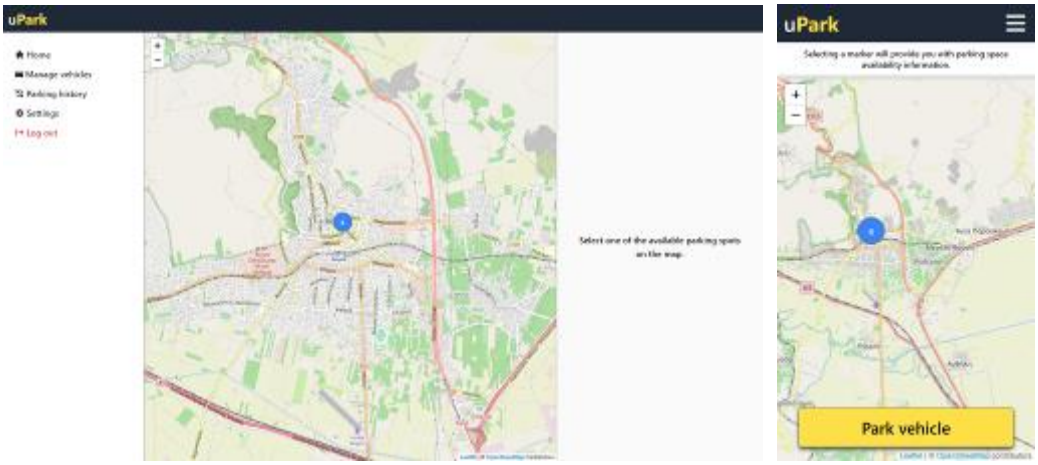



Figure 3. Home screen for authenticated Drivers (left: desktop, right: mobile)

Once logged in, Drivers can see on interactive map available parking spots in predefined sectors of an area, e.g., a city, (Figure 4) and proceed with parking spot selection and purchase of parking time. Parking spots can be selected in three ways depending on whether a desktop or a mobile user makes the spor selection. In particular, desktop users can select parking spots directly on the interactive map. Mobile users can select a parking spot either by scanning with their mobile device the QR code physically placed on the parking sector to which the desired parking spot belongs or (if the camera on their device is not functioanal) by simply typing the number of the parking sector of interest.

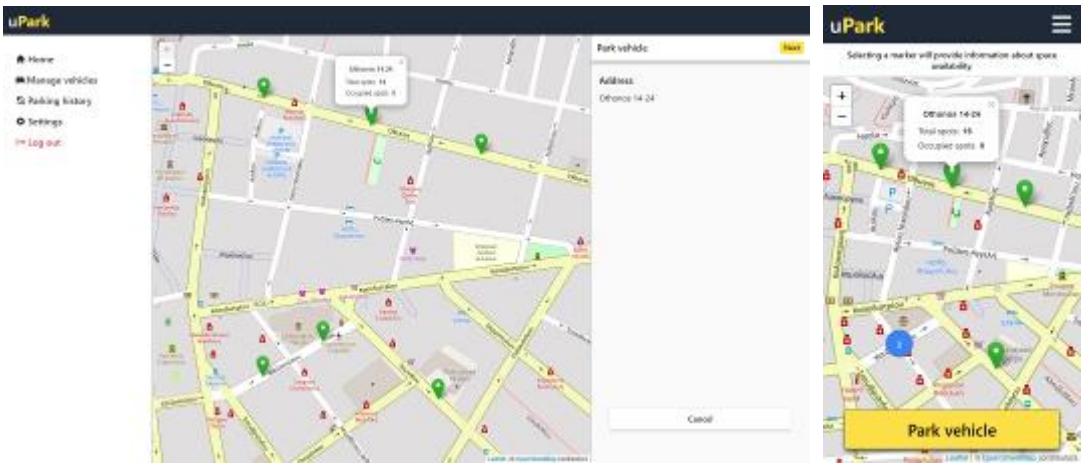



Figure 4. Information on available parking spots (left: desktop, right: mobile)

Via their dashboard, Drivers can store information about their vehicles via the functionality Manage vehicles. In this way, for purchasing parking time, Drivers can simply select a vehicle from their list of stored vehicles, instead of having to manually enter the requested information each time they wish to make a new purchase. As shown In Figure 5, Drivers can add a new vehicle to the list, and modify or remove an existing vehicle. In addition, a unique QR code is generated for each vehicle.

Technium Vol. 4, Issue 5 pp.26-38 (2022)
ISSN: 2668-778X
www.techniumscience.com

Drivers can optionally print and place his QR code on some visible spot of their corresponding vehicle, so that Inspectors can easier and faster check parked vehicles.

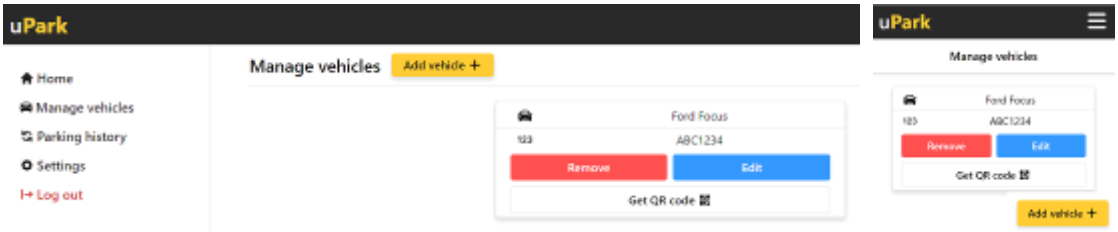



Figure 5. Vehicle management functionalities via the Driver dashboard (left: desktop, right: mobile)

Drivers can also have access to a brief or detailed history of parking spots they used (Figure 6) via the Parking history option. Drivers are first presented with an overview of previously used parking spots. By selecting a particular parking spot, detailed information appears including exact postal address, parking date, vehicle make and license plate, parking duration and cost, as well as status (expired or active).

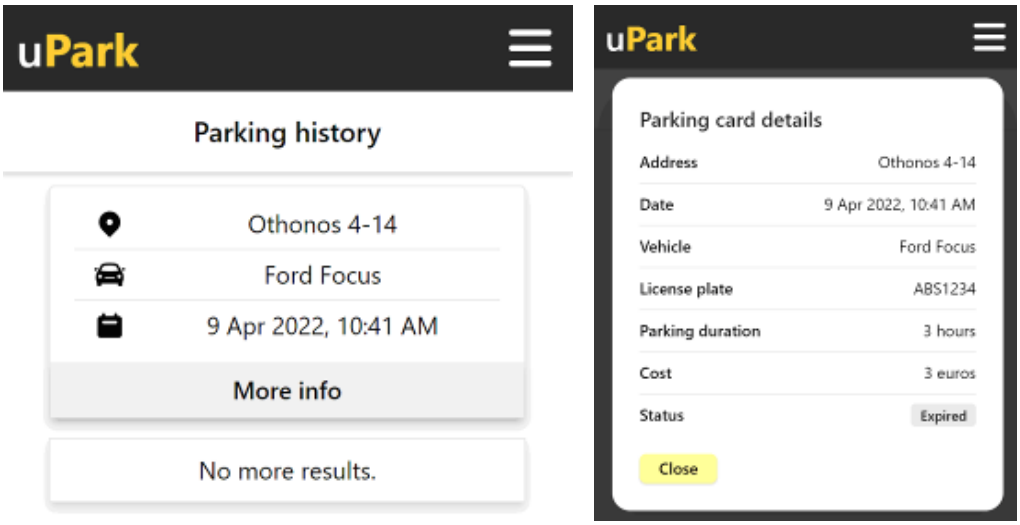
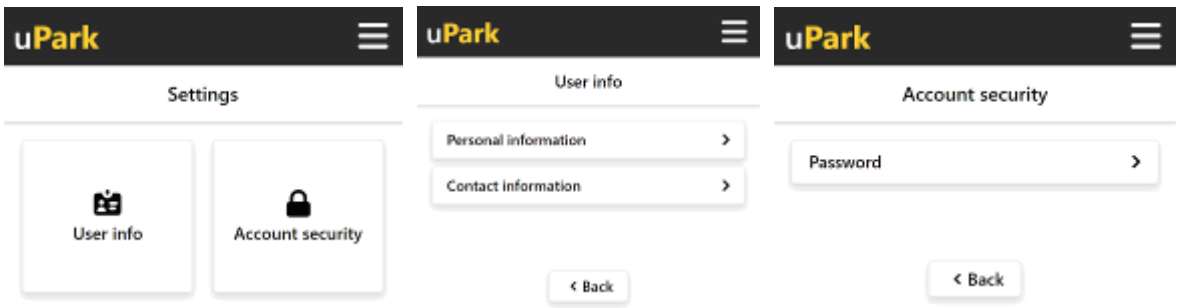


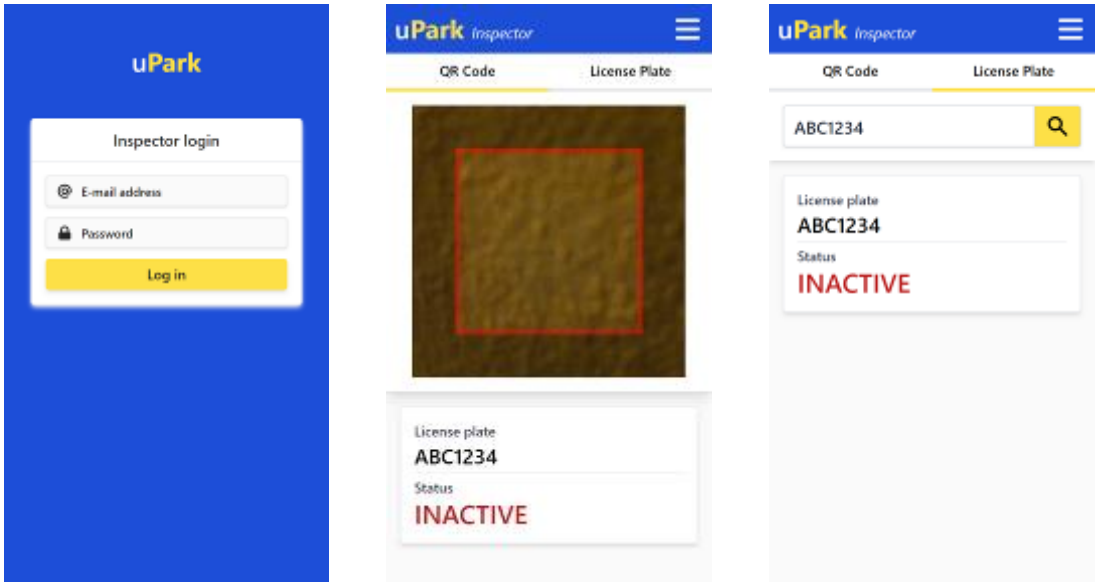

Figure 6. Brief history (left) and detailed information for already used parking spots.

Drivers can edit profile information via the Settings option (Figure 7). In particular, Drivers can edit personal information (first and last name) and contact information (email address) via User info as well as password via Account security.



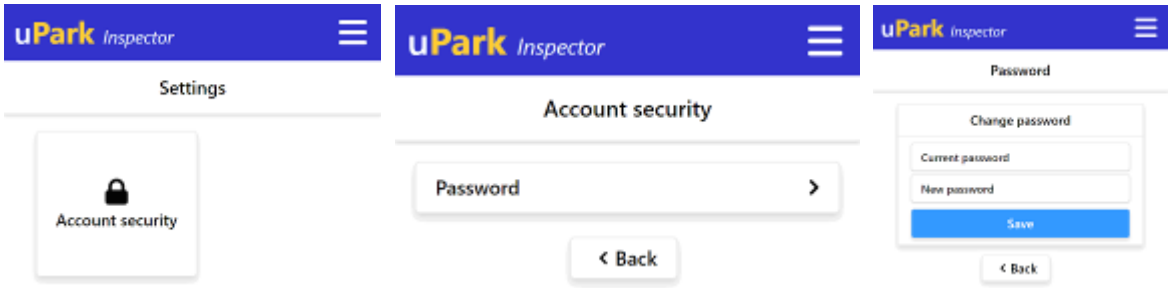

. Figure 7. Profile edit options for Drivers.

*2.1.2. Inspector subsystem.* uPark home screen when accessed by an Inspector is depicted in the left part of Figure 8. Inspectors must sign in at https://cerulean-druid-66a5aa.netlify.app/ using an email address and a password provided by the uPark Administrator. If successfully signed-in, an Inspector is presented with a checking functionality for parked cars either via scanning the QR code on the vehicle (middle part of Figure 8) or by typing in the plate number of a car (right part of Figure 8).




. Figure 8. uPark home screen (left) and checking functionalities (middle and right) for Inspectors.

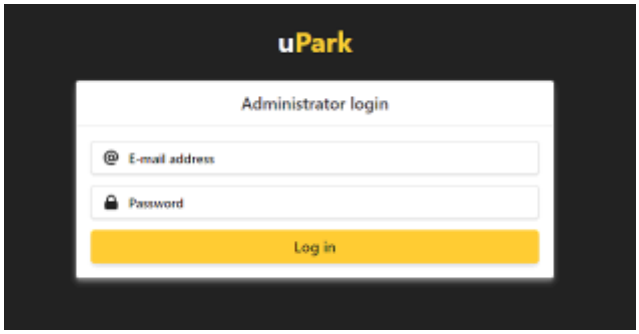
Inspectors can change their account password via the option Account security under Settings as shown in Figure 9.



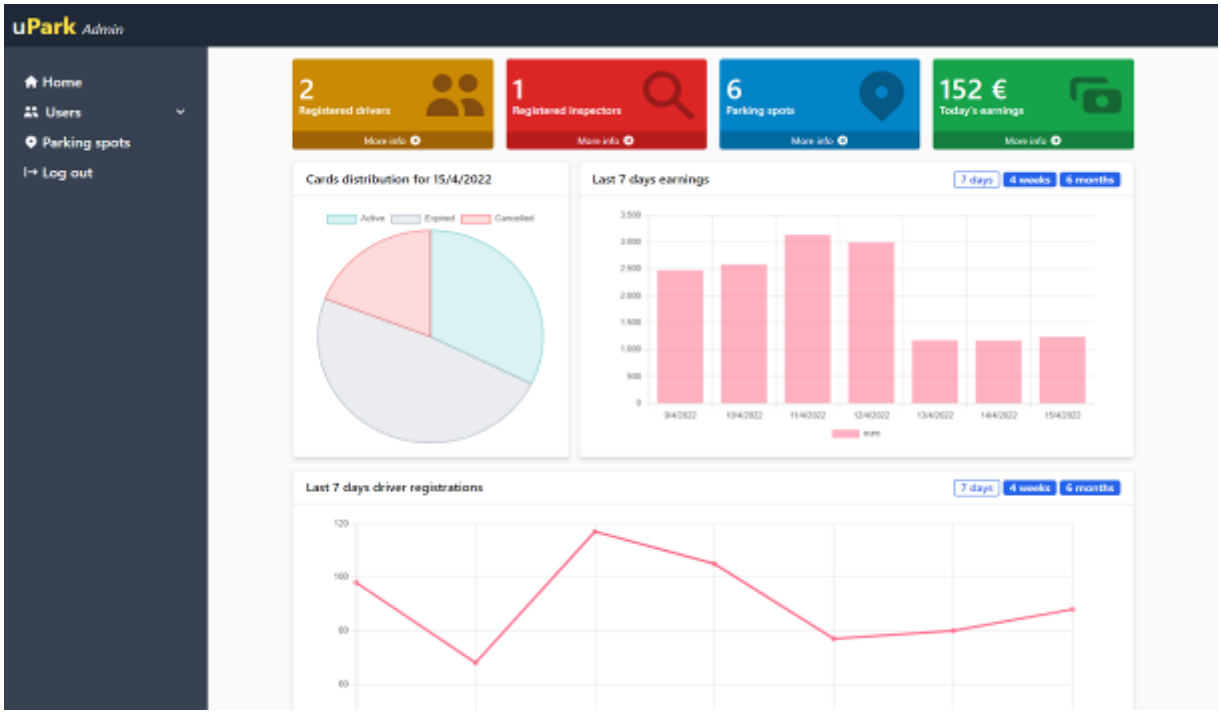

. Figure 9. Change password functionality for Inspectors

*2.1.3. Administrator subsystem.* For accessing uPark, Administrators must first sign in at https://startling-tapioca-84e54e.netlify.app/ (Figure 10). They are then presented with the dashboard depicted in Figure 11. Via the dashboard, Administrators have access to various information including system users (registered Drivers and Inspectors), parking spots and revenues, both in plain text and graphical format. In this way, Administrators can have a complete overview of uPark performance and proceed to corrective actions when necessary.

Technium Vol. 4, Issue 5 pp.26-38 (2022)
ISSN: 2668-778X
www.techniumscience.com




. Figure 10. uPark log-in screen for Administrators

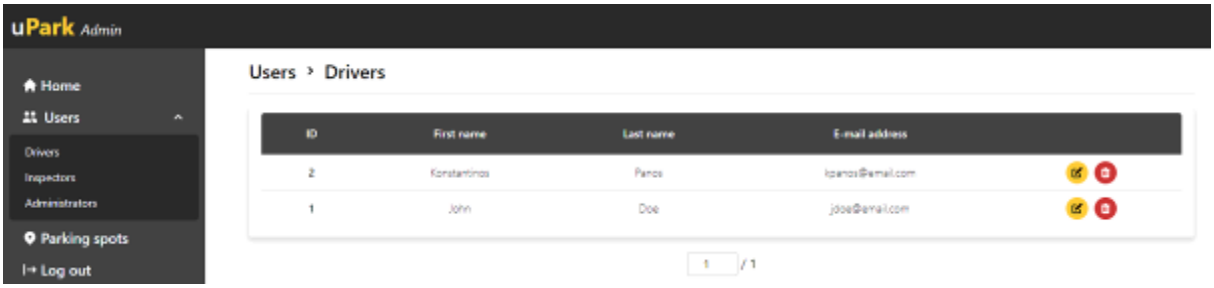



.

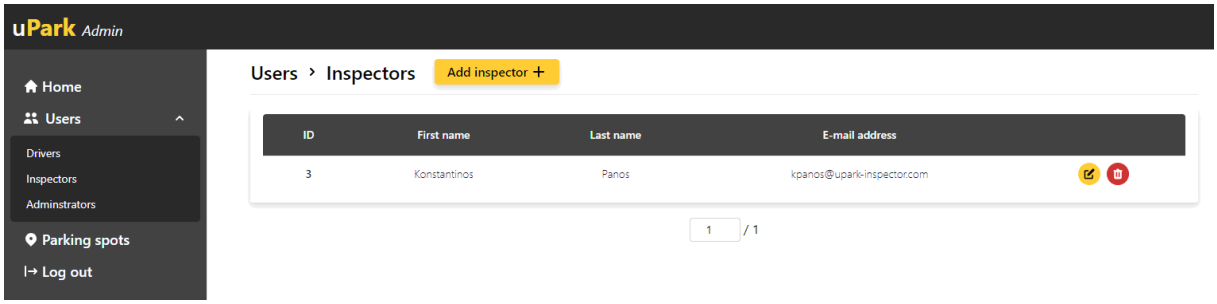
. Figure 11. uPark Administrator dashboard

Administrators can manage all registered users and parking spots. Administrators have access to the list of all registered Drivers, Inspectors and Administrators. They can edit or remove Drivers from uPark (Figure 12). They also have access to the list of all registered Inspectors (Figure 13) and can add, edit (name, email, password reset) or remove Inspectors. As shown in Figure 14, Administrators have access to the list of all parking spots registered in the system database. They can add new parking spots, edit or delete existing ones. Administrators can view on a map every parking spot registered in the uPark database; they can also edit postal address, coordinates and number of available parking spots per sector.

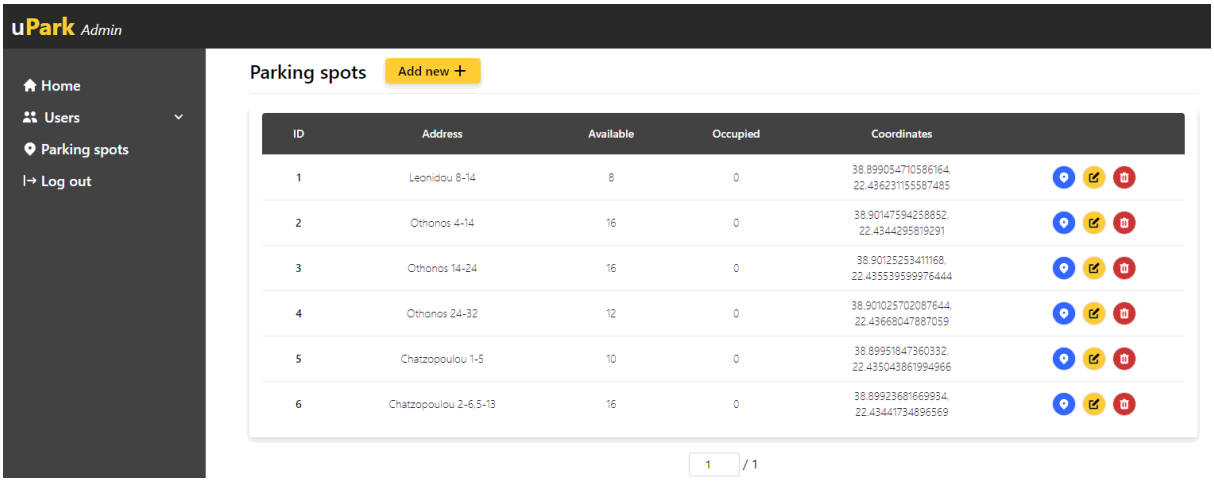
Technium Vol. 4, Issue 5 pp.26-38 (2022)
ISSN: 2668-778X
www.techniumscience.com




. Figure 12. Driver management dashboard




. Figure 13. Inspector management dashboard




. Figure 14. Parking spot management dashboard

*2.2. System implementation*

In this section, we overview programming languages, frameworks and other software used for uPark development. uPark, as a progressive web application, is actually a website whose development invlolves web technologies. uPark is composed of two parts: front-end, i.e., the client-side, and back-end, i.e., the server-side. The front-end part deals with how things are presented to the user and, actually corresponds to the user interface. The back-end refers to the involved server which processes requests, queries the database and produces responses to the browser. Both parts were developed using Node.js [3,4] and the Typescript programming language [5,6]. In addition, React.js library [7] was used for the front-end and Express.js library was used for the back-end [9]. We used PostgreSql for the uPark database [3].

Node.js [3,4] is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine which converts JavaScript code into machine code. It is lightweight, scalable, fast,

and data-intensive. An Node.js application is executed in onle one process and does not start a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking. In general, Node.js libraries significantly limit blocking behavior since they are based on non-blocking paradigms. More precisely, when Node.js performs I/O operations, like reading from the network, accessing a database or the file system, instead of blocking the thread wasting time, it resumes the operations after a response returns. This practice allows Node.js to manage thousands of concurrent connections with a single server without needing to deal with thread concurrency and, thus, avoiding lots of potential bugs. Node.js supports the development of web-application using just the JavaScript programming language, instead of different languages for server-side and client-side development.

Typescript [5,6] is a programming language developed and maintained by Microsoft and a set of tools. It is a typed superset of JavaScript compiled to JavaScript. That is, TypeScript is JavaScript enriched with additional features for application-scale development. All TypeScript code is converted into its JavaScript equivalent for execution either at the client-side or at the server-side. TypeScript overcomes the largest issue with JavaScript, which has to do with the fact that a problem can only be detected during runtime in JavaScript. TypeScript removes this problem by checking for any issue at compile time, thus helping to detect errors before a script is executed.

React.js [7,8] is a free and open-source front-end JavaScript library for building compex user interfaces from small and non-interacting parts of code called “components”. React can be used as a base in the development of mobile or server-rendered applications. However, React is only concerned with state management and their rendering to the Document Object Model (DOM). A significant feature of React is the use of a virtual DOM. React uses reconciliation, i.e., it creates an in-memory data-structure cache, computes the resulting differences and then efficienly updates the browser displayed DOM. This allows programmers to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change. This selective rendering provides a major performance boost saving effort of recalculating the CSS style and page layout.

Express.js [9] is a back-end web application framework for Node.js. It is a free, open-source software under the MIT license. It is designed for building web applications and APIs. Express is a minimal and flexible Node.js web application framework that provides broad features for building web and mobile applications. Express is the back-end component of popular development stacks together with the MongoDB database software and a JavaScript front-end framework or library. Express provides a simple routing for requests made by clients as well as a middleware responsible for making decisions to give the correct responses to client requests. Also, based on the V8 engine of the Google Chrome browser, it can make a very good choice for applications that require very fast response and request processing, as well as for real-time applications such as live-chat services.

For developing uPark, we extensively used the CSS framework TailwindCSS (https://tailwindcss.com/), a framework for rapidly building custom user interfaces. Compared to other widely used frameworks, like for example Bootstrap, which provides developers with predefined elements such as buttons and menus, TailwindCSS is a utility-first CSS framework, offering smaller building blocks (CSS utility classes) which can be combined to produce larger and specialized elements that best meet particular requirements without writing CSS as in traditional approaches. Furthermore, TailwindCSS supports the development of responsive user interfaces for different screen sizes and a uniform way of naming CSS classes.

Passport (https://www.passportjs.org/) is authentication middleware for Node.js which can be used by any Express-based web application. It supports several login types which can then be combined for user authentication via optional authentication services (e.g., google, facebook) or combinations of external authentication services. We also used passport-local (https://www.passportjs.org/packages/passport-local/), a module allowing authentication using a username and password in Node.js applications.

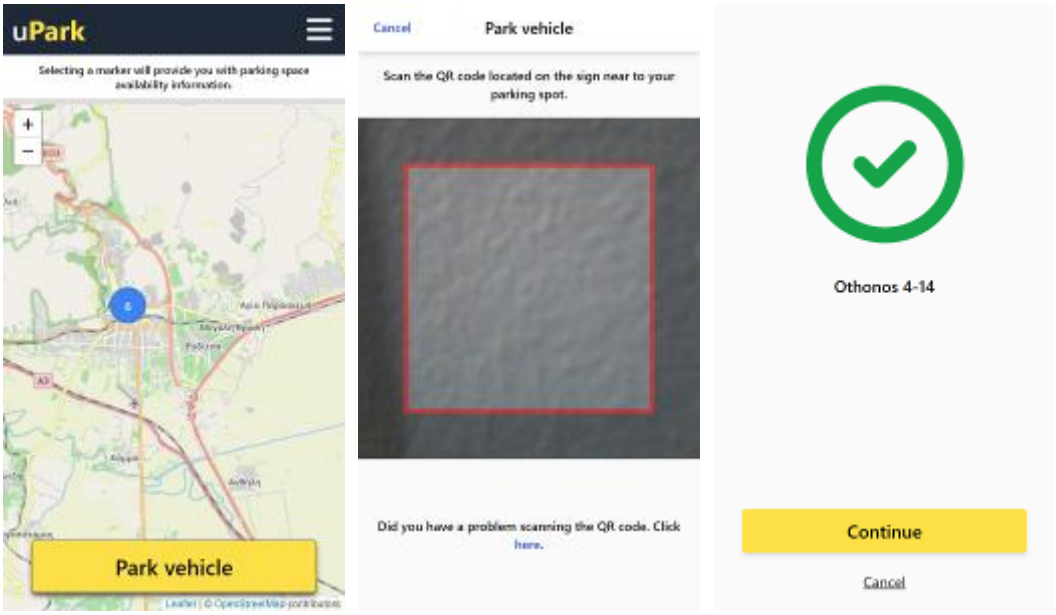
We used node-postgres (https://node-postgres.com/), a collection of Node.js modules, for interfacing with our PostgreSQL database and Socket.IO (https://socket.io/), an event-driven JavaScript library for real-time web applications, which enables real-time, bi-directional communication between web clients and servers. It includes a client-side library running in the browser and a server-side library for Node.js.

For the management and acceptance of online payments, we used stripe.js (https://js.stripe.com), a JavaScript library providing APIs to tokenize customer information, collect sensitive payment details using customizable stripe elements while offering advanced fraud functionalities. We used Leaflet (https://leafletjs.com/), a very popular open-source JavaScript library for mobile-friendly interactive maps and for the map refresh process we used React Router DOM (https://reactrouter.com/), a client and server-side JavaScript routing library for React for building user interfaces enabling the implementation of dynamic routing in a web app, i.e., pages are never refreshed but the content is dynamically fetched based on the URL. For the communication of uPark with its server, we used Axios (https://axios-http.com/), a JavaScript library used to make HTTP requests from Node.js or from the browser and supports the Promise API. We used the react-qr-reader.js (https://www.npmjs.com/package/react-qr-reader) for scanning QR codes from web-browser-based applications and Chart.js (https://www.chartjs.org/), a free, open-source JavaScript library, for data visualization. We used Visual Studio Code (https://code.visualstudio.com/) as our main a code editor.

## 3. uPark demonstration scenario

In this section, we demonstrate some of the functionalities of uPark via a scenario according to which a registered Driver with a mobile device with a functional camera wishes to purchase parking time for a parking post just occupied by one of his vehicles.

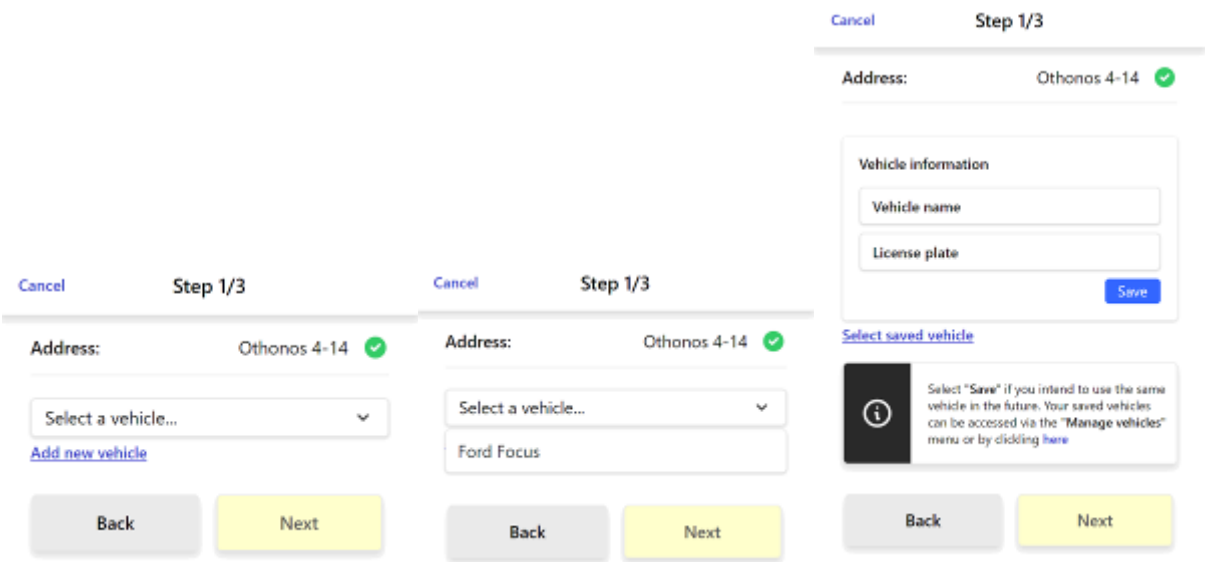
The registered Driver must first sign in to uPark. After signing in, uPark automatically detects the Driver's position (Figure 15, left part) and prompts the user to scan the QR code of the current parking sector (Figure 15, middle part). A message confirms a successful scan (Figure 15, right part). An indicative sign indicating the QR code assigned to a parking sector is depicted in Figure 16.




. Figure 15. A Driver selecting a parking spot before purchasing parking time

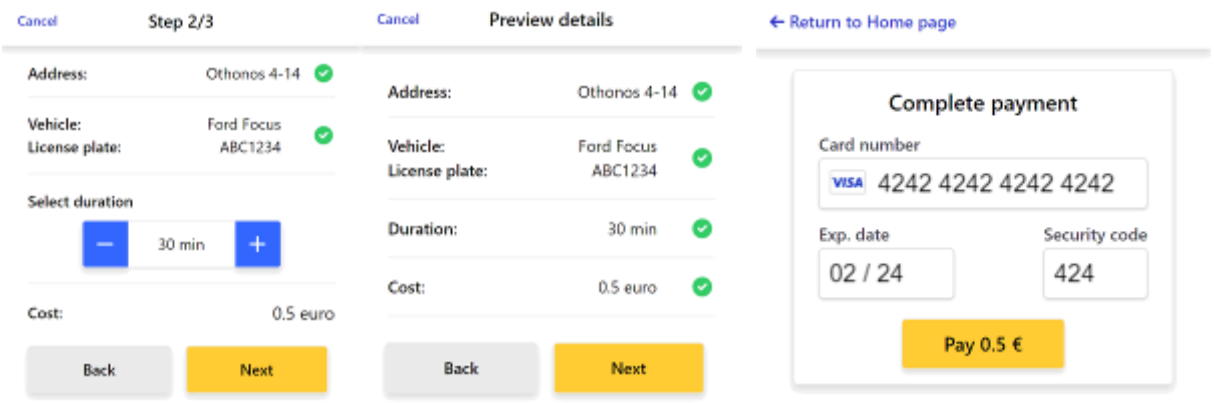

. Figure 16. A sign indicating the QR code assigned to a parking sector

Then, the Driver must declare the vehicle just parked in order to purchase parking time (Figure 17, left part). The Driver can either select the vehicle of interest from the list of the Driver's registered vehicles (Figure 17, middle part) or manually enter information regarding the vehicle of interest (Figure 17, right part).



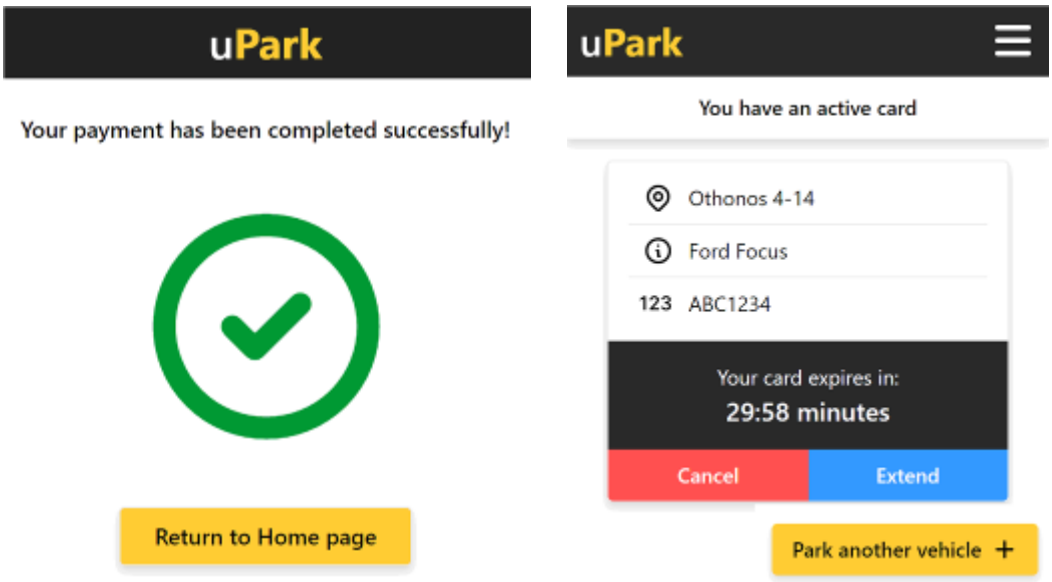

. Figure 17. Vehicle selection

Then, the desired parking duration must be selected (Figure 18, left part). Available parking duration ranges from 30 minutes to 5 hours, with a step which is currently set to 30 minutes. Then, an overview of the Driver's purchase request appears (Figure 18, middle part). After confirming, the Driver is transferred to the payment page (Figure 18, right part).




. Figure 18. Selection of parking duration

After the payment is successfully completed, a confirmation message appears (Figure 19, left part). The Driver can see at the uPark home screen full details of the parking time just purchased with options to pause or extend it; an option for purchasing parking time for another vehicle is also available (Figure 19, right part).




. Figure 19. A successfully completed parking time purchase

## 4. Concluding remarks

We designed and developed uPark, a progressive web parking application, integrating three separate subsystems, one for Drivers available at https://strong-taiyaki-58c81d.netlify.app/, one for Inspectors available at https://cerulean-druid-66a5aa.netlify.app/ and one for Administrators available at https://startling-tapioca-84e54e.netlify.app/. As a progressive web application, uPark is essentially a website but can also be "installed" on the user's device, giving the feel of a native application. uPark complements and extends other, similar in spirit, existing applications, since, apart from being a PWA, uPark is very flexible and can function for any public or private parking area in a city, mall or campus, as long as available parking sectors and parking spots are provided. From a technical point of view, uPark fruitfully combines a wide range of state-of-the-art web technologies and frameworks into an elegant, responsive and completely functional application environment.

It is within our future plans to make uPark publicly available first within our academic community for testing within our university campus and then at our city municipal authority for testing at parking facilities at the city port, the city center, a local mall as well as at private parking facilities. Observing uPark in practice can provide valuable feedback for improvements and extensions.

**References**

[1] M. J. WARGO. Learning Progressive Web Apps. *ISBN: 9780136485674*. Addison-Wesley Professional, 2020.
[2] Progressive web apps (PWAs) | MDN. Available at https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
[3] G. LIM. Beginning Node.js, Express & MongoDB Development. *ISBN 10: 1078379556, ISBN 13: 9781078379557*. Independently published, 2019.
[4] Introduction to Node.js. Available at https://nodejs.dev/learn/introduction-to-nodejs
[5] B. CHERNY. Programming TypeScript: Making Your JavaScript Applications Scale. ISBN-10: 1492037656, ISBN-13: 978-1492037651. O'Reilly Media Inc., Sebastopol, CA, USA, 2019.
[6] TypeScript: JavaScript With Syntax For Types. Available at https://www.typescriptlang.org/
[7] React – A JavaScript library for building user interfaces. Available at https://reactjs.org/
[8] React: The Virtual DOM . Available at https://www.codecademy.com/article/react-virtual-dom
[9] Express - Fast, unopinionated, minimalist web framework for Node.js. Available at https://expressjs.com/