6/9/2022

Project Report

IoT Web Based Smart Parking

Supervised By:

Pr. Abdelatif RAHMOUN
Dr. Hamdan BENSENANE

Team Members:

Members	Roles
BELFAR Ilyas	Front-end Developer
BOUNAB Abdelmounaim	Hardware
SEGOUAT Mohcen Abdelouahed	Back-end Developer
ZELLAGUI Iskander	Team Leader
ZIANE Mohamed	Hardware

Table of Contents:

Content	Page
1. Team Members	1
2. Table of Contents	2
3. Abstract	3
4. Introduction	4
5. Internet of Things (IoT)	5
6. IoT Smart Parking	6
7. Our IoT Smart Parking	7
• Working Principle	7
• Hardware Materials Used	8
• Programming Languages and Software Used	10
• iParking Diagrams	14
• iParking User Interface	18
• iParking Applications	24
• iParking Advantages	24
• Bill of Materials and Cost	24
• Parking Model	25
8. Conclusion and Perspectives	26
9. Table of Figures	27

Abstract:

Rapid demographic growth is always the source of many problems, including the increase in transport needs, which directly leads to traffic jams on one hand and the parking problem on the other.

Parking space has become a very rare thing to come by, many people found themselves forced to roam the streets in the search of it, risking maybe getting a fine or being late to work.

Through our project, we aim to offer an IoT car parking system, using connected objects, which allows our customers to know the place to go to.

Keywords: Parking System, Connected Objects, IoT

Introduction:

Imagine, you arrive at your destination twenty minutes early for a very important meeting. You have plenty of time as long as you can find a place to park.

The parking lot for the building is full. You drive around desperately looking for a space on the street but do not find one. You try the underground parking lot of the building across the street. Driving in, you suddenly have to stop. There is plenty of traffic ahead of you trying to do the same thing. You attempt to call the meeting to say you will be late and there is no cell phone signal in the underground parking garage.

It takes a half-hour to find a space. When you finally arrive at the office for the meeting, you are sweating profusely and out of breath. The receptionist tells you that everybody already left. Your meeting was cancelled and you have to deal with serious losses.

Well you are not alone. An average driver in the US wastes $345 per year, which results in over $70 billion annually nationwide.

Recent research is predicting that up to 68% of the people in the world will live in major metropolitan cities by 2050. This could have a direct impact on how car owners park in cities.

With the rise of connected devices, IoT-based smart parking systems are becoming more and more popular. These parking systems empower drivers with advanced information about available parking lots, the payment method can be easily automated and if these parking systems are private (for customers only) that means that you would have a bigger chance of finding an available lot to park.

Internet of Things (IoT):

The basic definition of an Internet of Things (IoT) can be defined as anything which could be connected to internet results into “Internet of Things”.

The things in Internet of Things are sensors, actuators, RFID tags. The things could be tracked, controlled or monitored using remote computers connected through Internet.

IoT extends the use of the Internet, providing the communication, and thus inter-network of the devices and physical objects, or “Things”.

IoT, in general consist of the devices and physical objects, number of objects can gather the data at remote locations and communicate to units managing, acquiring, organizing and analysing the data in the processes and services.

It provides a vision where things (wearable, watch, alarm clock, home devices, surrounding objects with) become smart and behave alive through sense computing and communicating with embedded small devices which interact with remote objects or persons through connectivity. Due to high scalability in the cloud any number of nodes could be added or removed from the IoT system on a real time basis and IoT is well known to reduce human effort storage at extent.

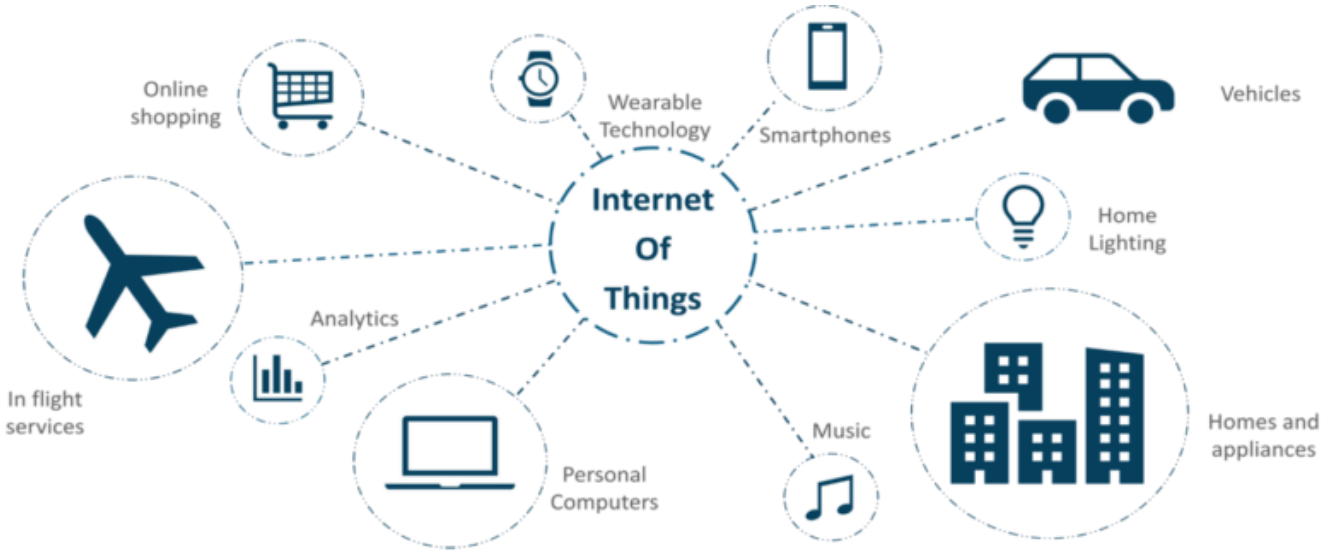

Figure1: IoT’s Domains of use

IoT Smart Parking:

There has been a tremendous growth in the number of vehicles on road in past few years. But unfortunately the road networks and road widths have not grown in proportion to vehicle numbers. This has created in huge parking crisis especially in urban areas. At such times smart online parking systems are the need of the hour.

This system aims at replacing the conventional parking system with an IoT-based smart parking system by using an automated access control (Microcontroller, Sensors and Actuators) which accordingly it can either authorize or deny the access of clients to the park. The users will be provided with an application (mobile or web), using which they can know about the availability of the parking lots. The system will have an automated process of payment in this way, it will help reduce human effort and time by using automation technology.

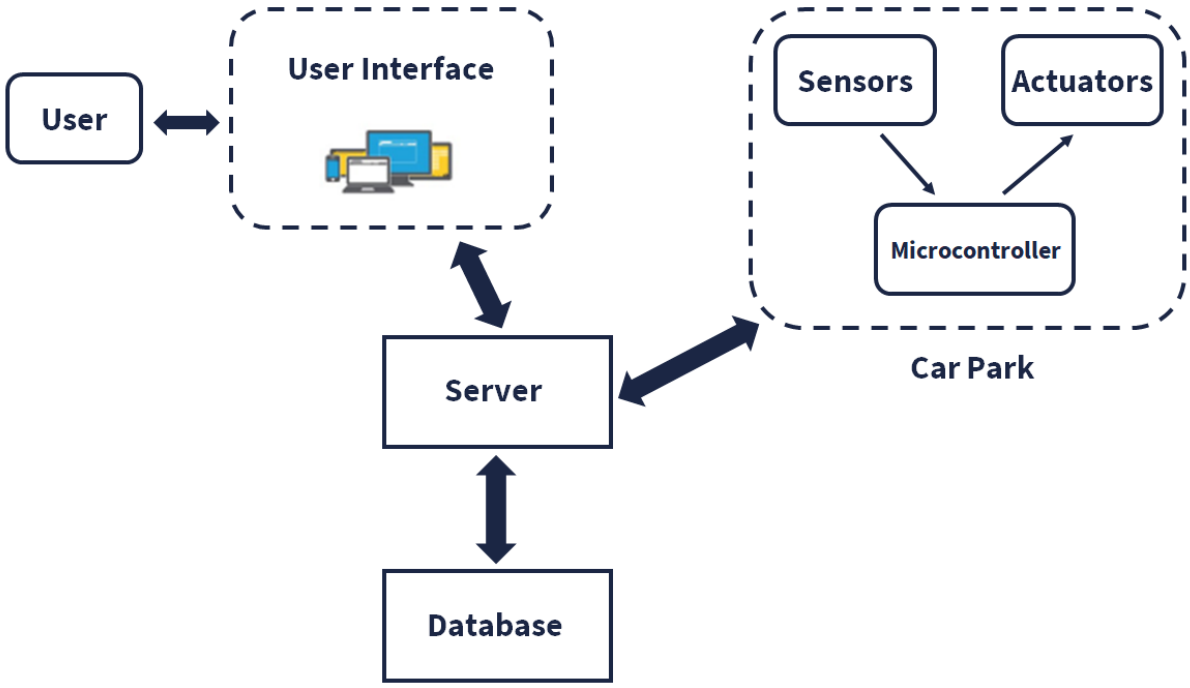

Figure2: IoT Smart Parking System Architecture

Our IoT Smart Parking:

1. Working Principle:

In order to make our version of an IoT Smart Parking we decided that our Access Control System will be based on RFID (Radio-Frequency-Identification), the client will show his own RFID tag at the entrance of the parking building, the RFID Reader will read the Id of that tag and send it to our web server through the microcontroller we have chosen (NodeMCU8266).

Once reached to the server, we will compare the tag's Id with those in our database (MongoDB), when we make sure that the tag exists in our DB we make other verifications related to the balance and debts.

If all constraints are verified, the user will be authorized to have access and the barriers of the entrance will be opened (Servo Motor) otherwise it will be denied.

At the exit of the parking building, the same method will be applied. We send the RFID tag Id using RFID Reader to the web server in order to register the date of exit so that parking cost can be calculated, now this will happen only if the client Tag indicates that he was in the parking area.

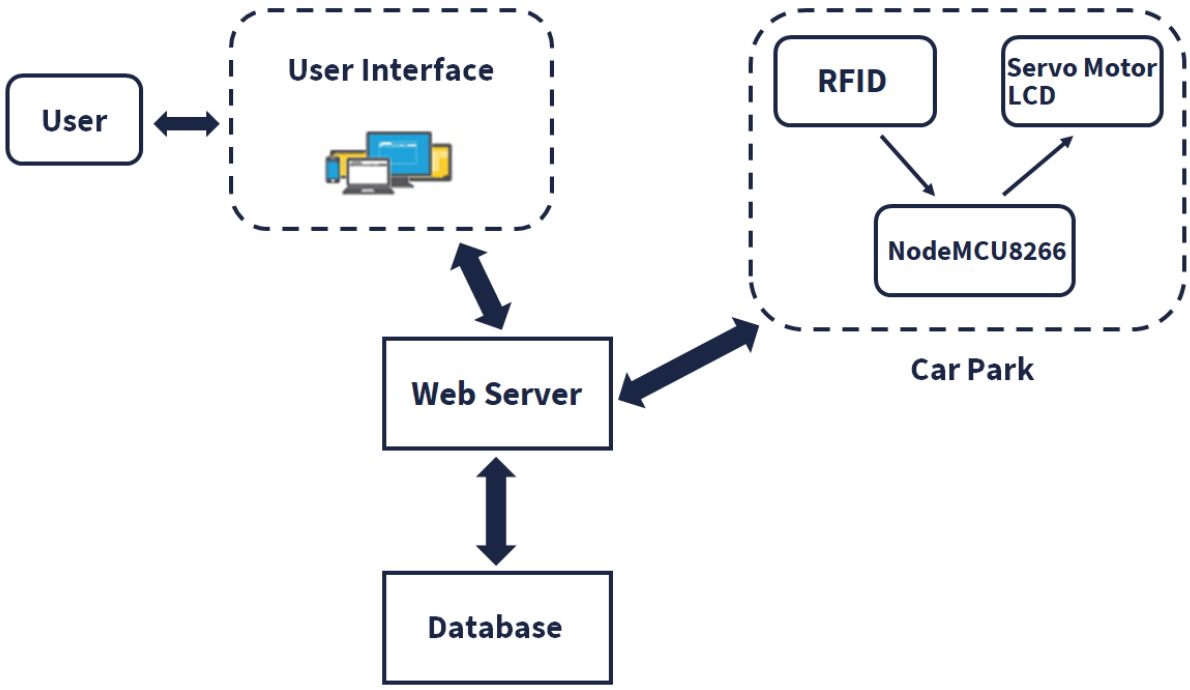

Figure3: Our Version of the IoT Smart Parking System Architecture

2. Hardware Materials Used:

- **PC HP Elite book G1:**
  - **Processor:** Intel® i5, 4th Generation.
  - **RAM:** 8GB.
  - **System:** Windows 10, 64 bits.
- **NodeMCU ESP8266:** The NodeMCU (Node Microcontroller Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (Wi-Fi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

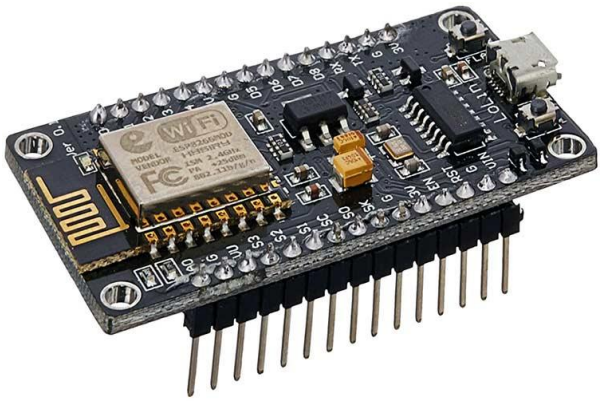

Figure4: NodeMCU ESP8266

- **Radio Frequency Identification:** RFID refers to a wireless system comprised of two components: tags and readers. The reader is a device that has one or more antennas that emit radio waves and receive signals back from the RFID tag.

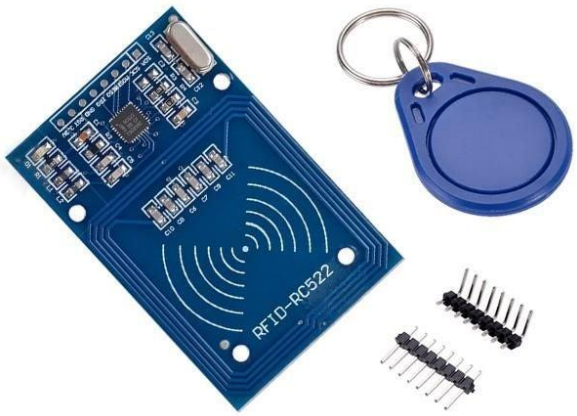

Figure5: RFID Reader and RFID Tag

- **Servo Motor:** A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor.

Figure6: Servo Motor

- **Liquid-Crystal Display:** LCD is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in colour or monochrome.

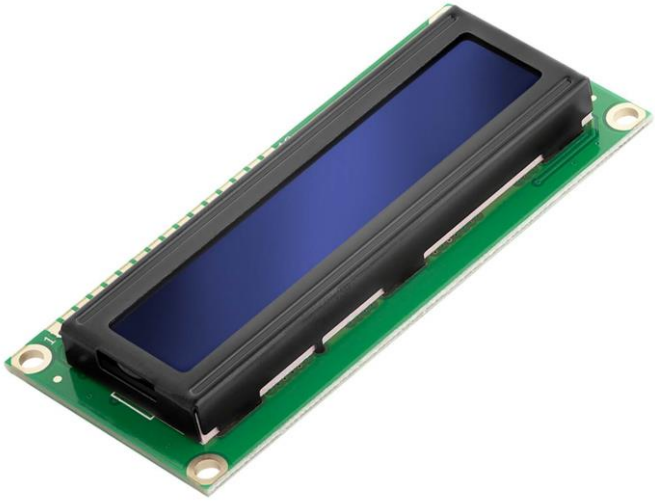

Figure7: LCD

3. Programming Languages and Software Used:

- **Python:** Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Figure8: Python logo

- **Flask:** Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Applications that use the Flask framework include Pinterest and LinkedIn.

Figure9: Flask logo

- **C++:** C++ is a general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. C++ was designed with an orientation toward systems programming and embedded, resource-constrained software and large systems, with performance, efficiency, and flexibility of use as its design highlights.

Figure10: C++ logo

- **HTML:** The Hypertext Mark-up Language or HTML is the standard mark-up language for documents designed to be displayed in a web browser.
- **CSS:** Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a mark-up language such as HTML.
- **JavaScript:** JavaScript often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behaviour, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

Figure11: HTML5 logo

Figure12: CSS3 logo

Figure13: JavaScript logo

- **Bootstrap:** Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Figure14: Bootstrap logo

- **Arduino IDE:** Arduino IDE (Integrated Development Environment) is used to write computer code and upload that code to the physical board. Arduino IDE is very simple and this simplicity is probably one of the main reasons why Arduino has become so popular.

Figure15: Arduino IDE logo

- **MongoDB:** MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL) which is deemed non-free by several distributions.

Figure16: MongoDB logo

- **Hypertext Transfer Protocol:** HTTP is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser.

Figure16.1: HTTP logo

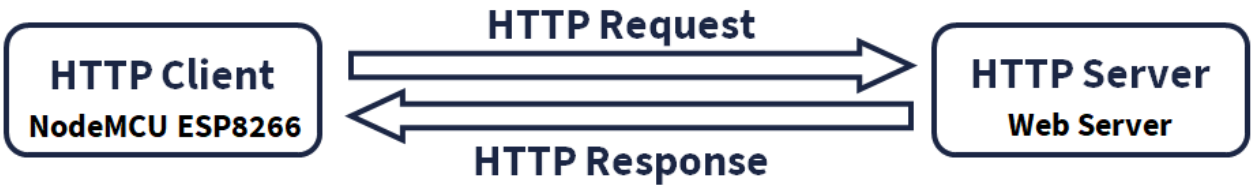

Figure16.2: HTTP Request/Response

Request Method	Request has payload?	Response has payload?	Safe?
GET	Optional	Yes	Yes
HEAD	Optional	No	Yes
POST	Yes	Yes	No
PUT	Yes	Yes	No
DELETE	Optional	Yes	No
CONNECT	Optional	Yes	No
OPTIONS	Optional	Yes	Yes
TRACE	No	Yes	Yes
PATCH	Yes	Yes	No

Figure16.3: Properties of request methods

4. Diagrams:

- **Use Case Diagram:** A use case diagram is a graphical depiction of a user's possible interactions with a system.

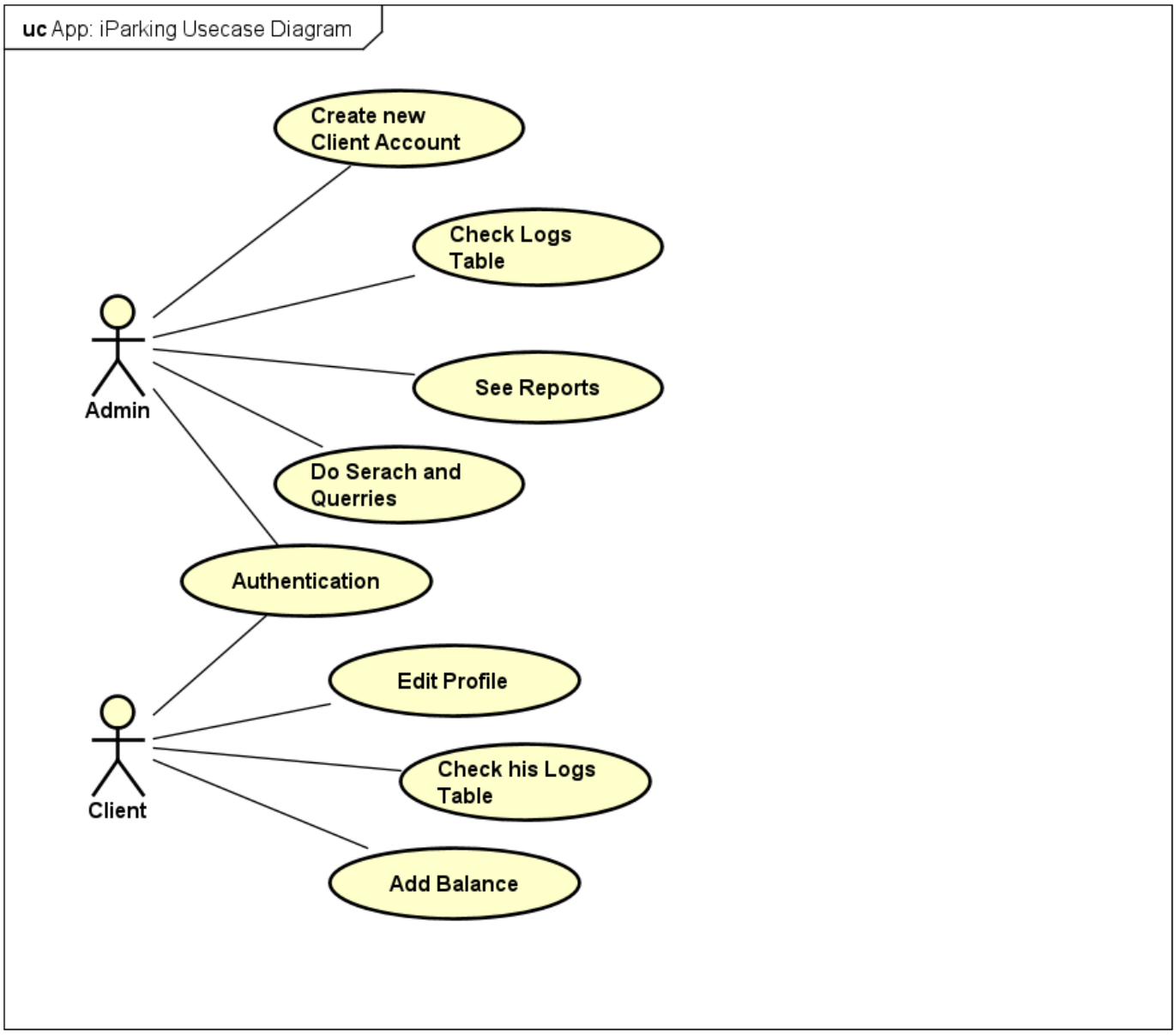

Figure17: Use Case Diagram

- **Class Diagram:** In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

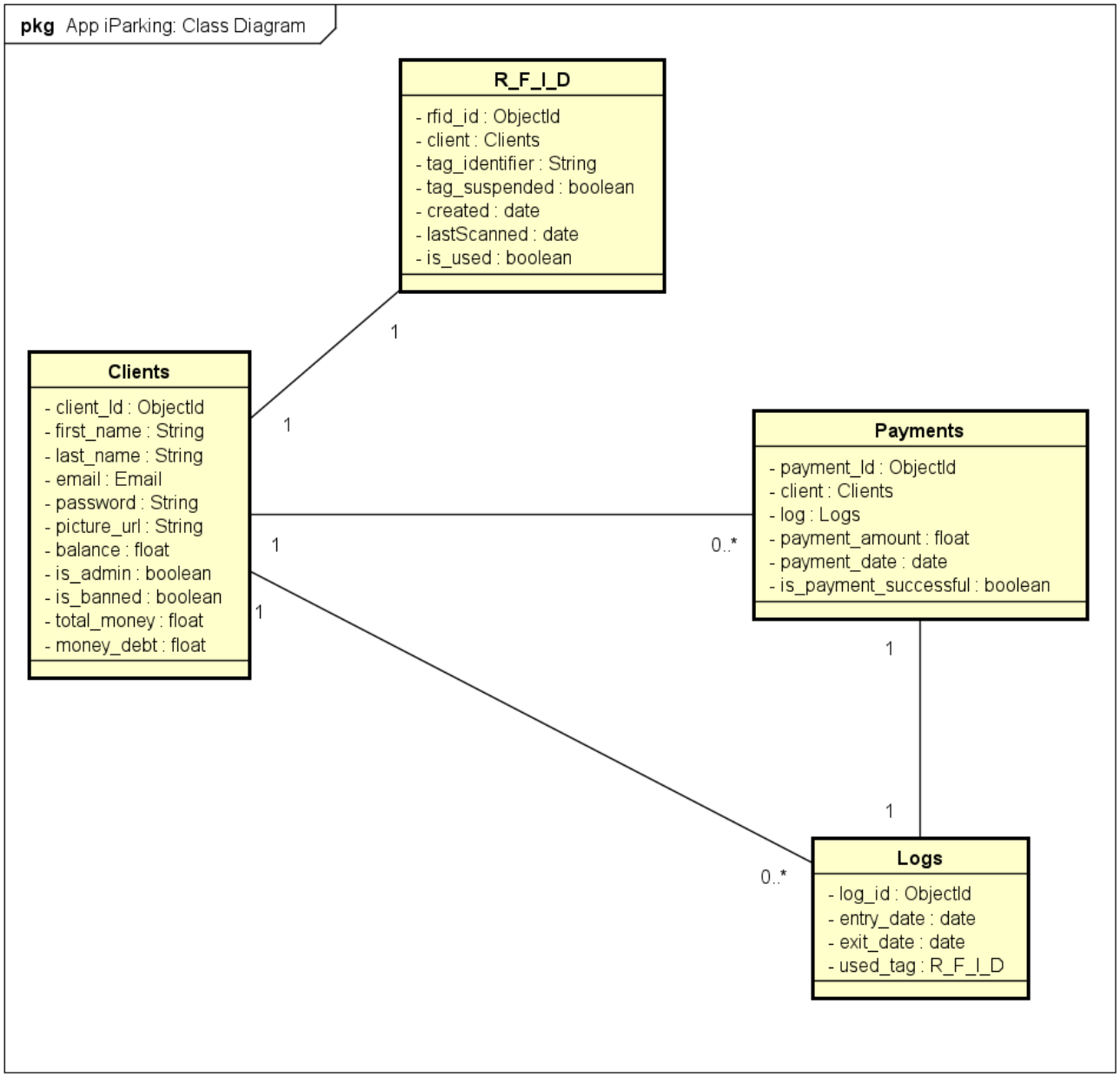

Figure18: Class Diagram

- **Activity Diagram:** Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

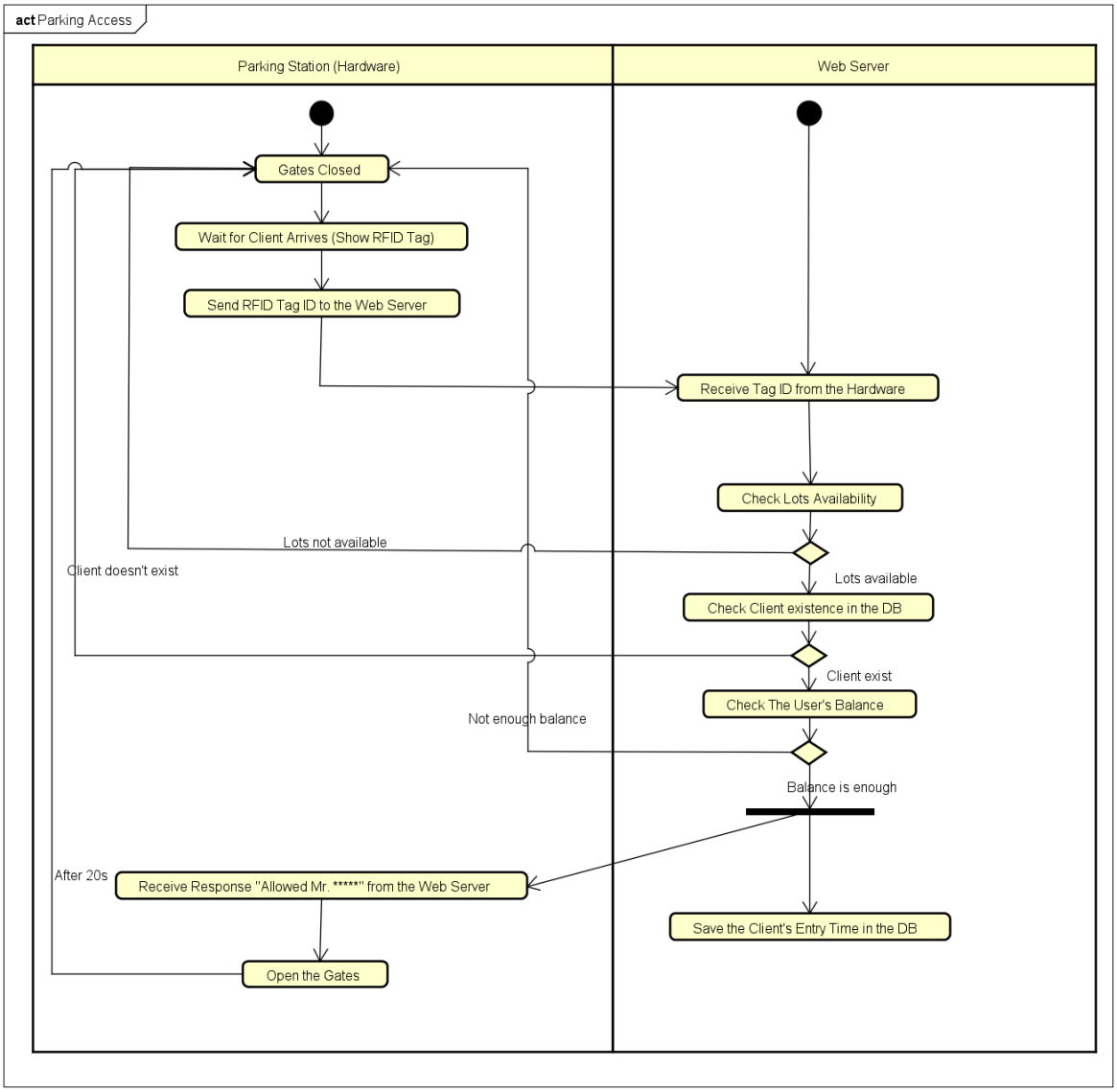

Figure19: Activity Diagram for "Parking Access"

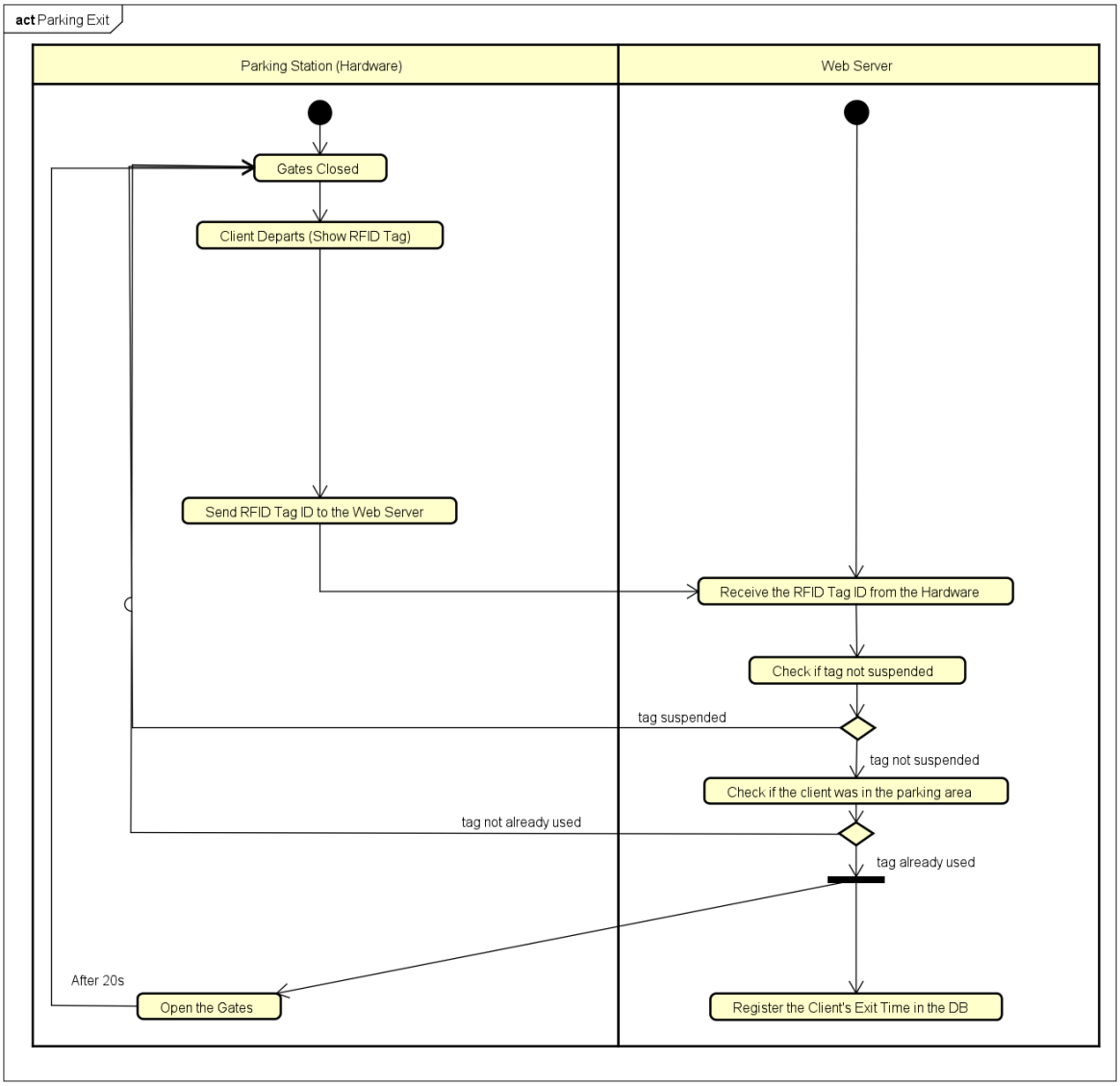

Figure20: Activity Diagram for "Parking Exit"

5. User Interface (iParking):

The app we have created (iParking) has two dashboards: one for the admin and the other for the client, each one has its unique design and features

Figure21: iParking Logo

- **Client Dashboard:**

Figure22: Login UI

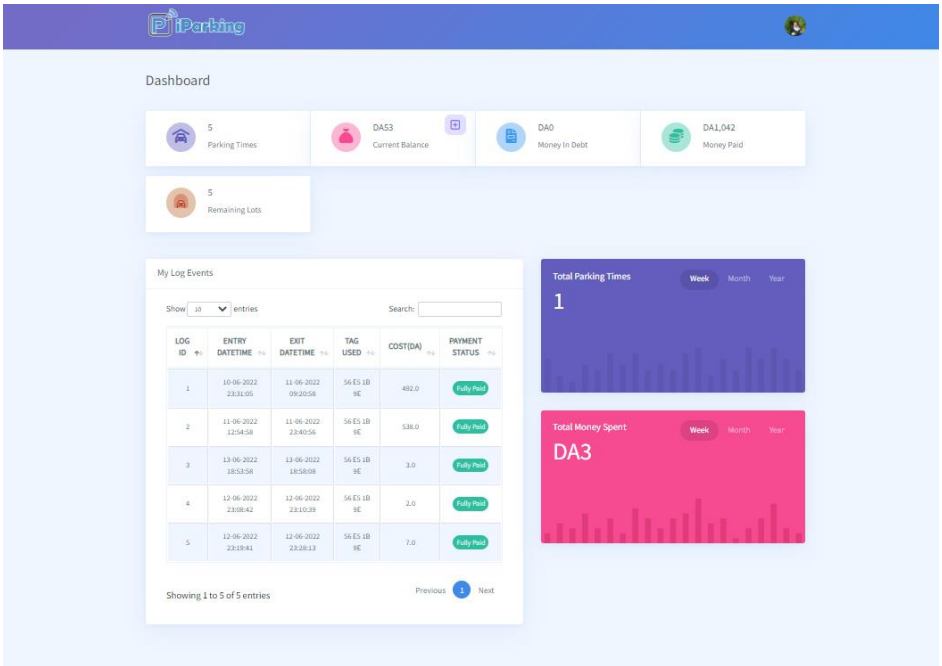

Figure23: User Dashboard UI

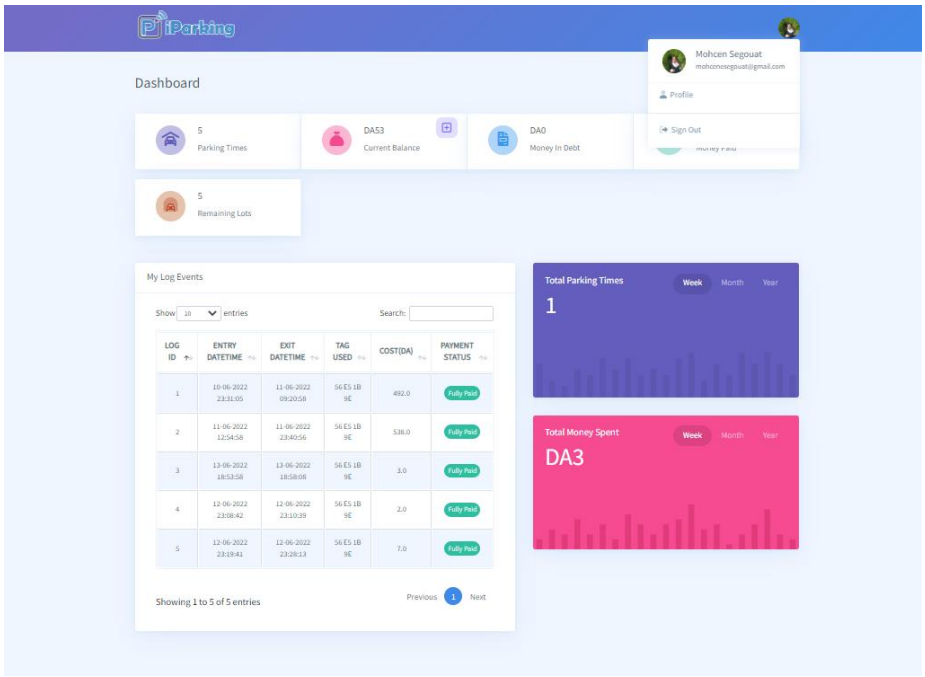

Figure23.1: User Dashboard UI

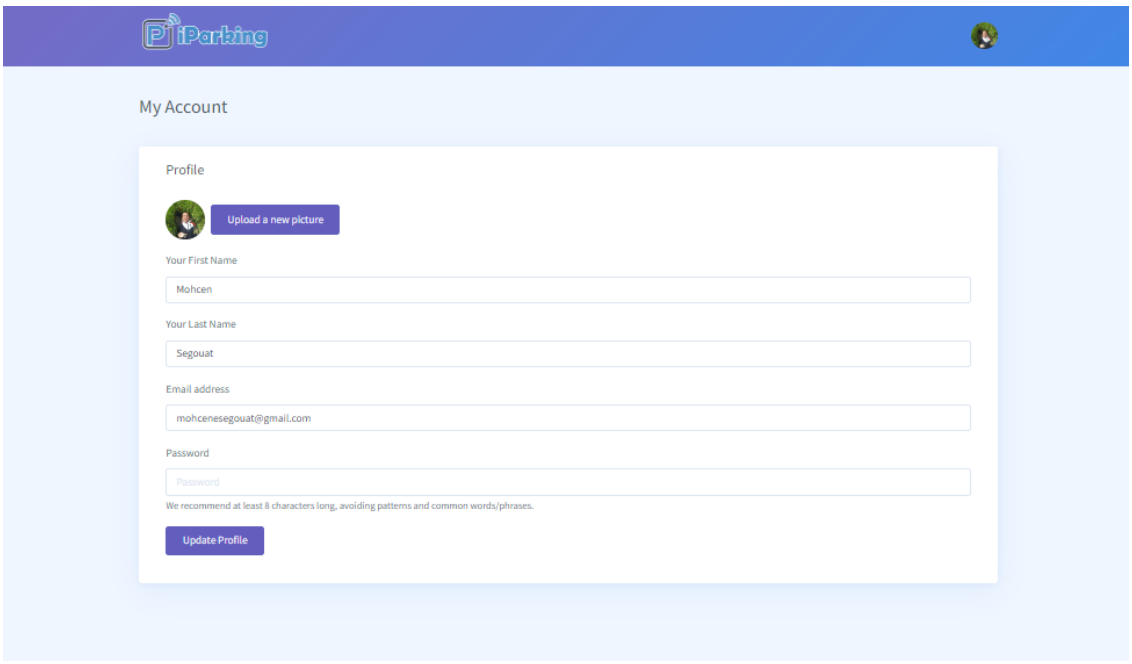

Figure24: Edit Profile UI

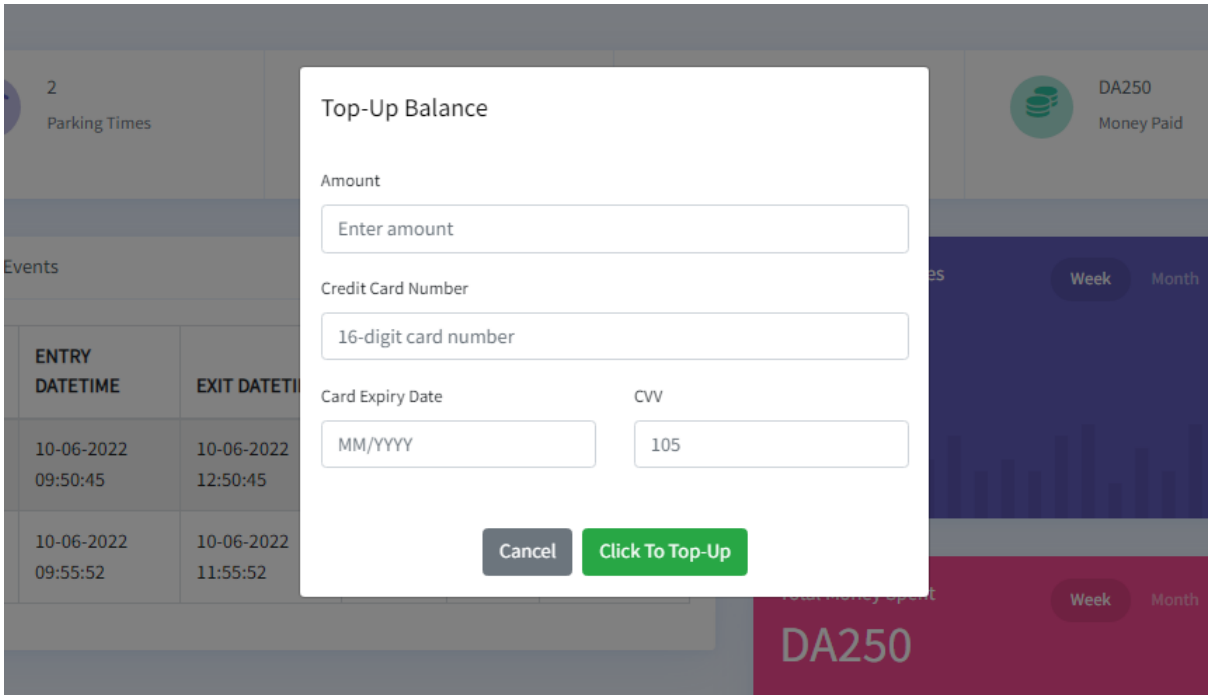

Figure25: Add Balance UI

- **Admin Dashboard:**

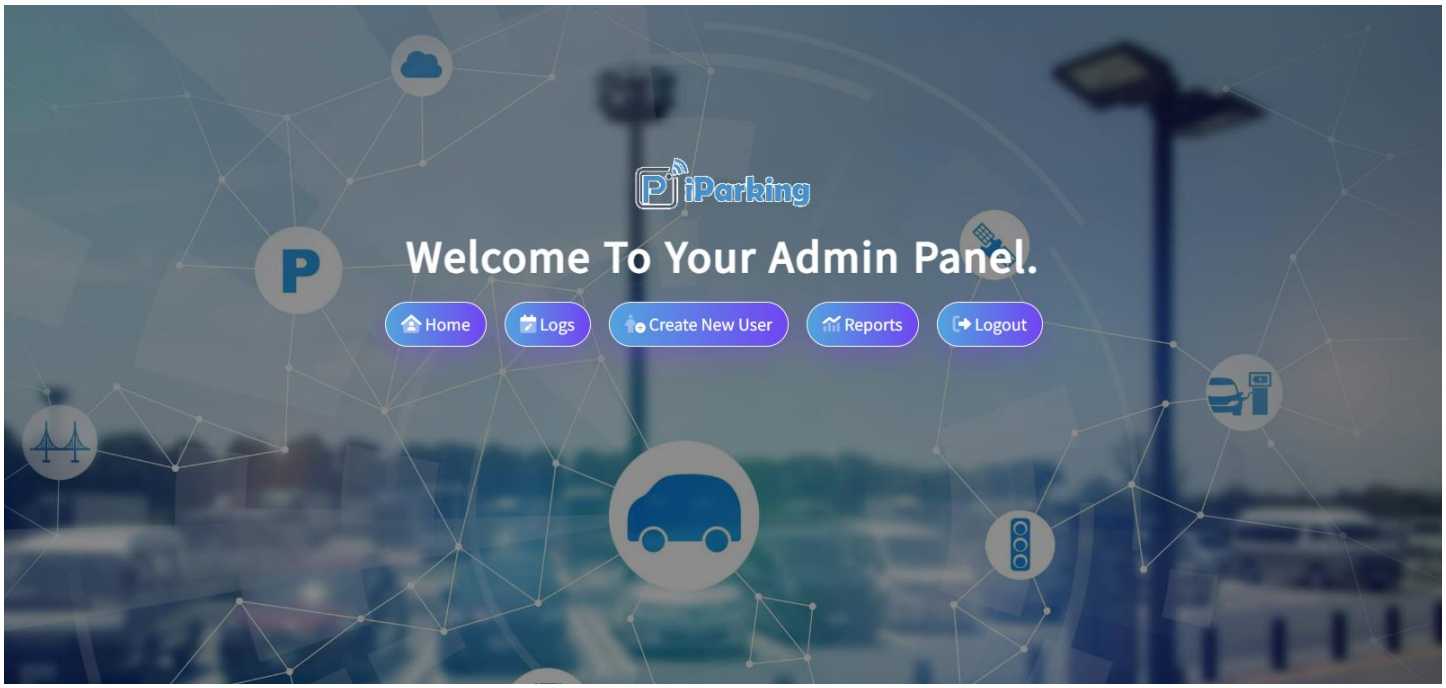

Figure26: Admin Dashboard UI

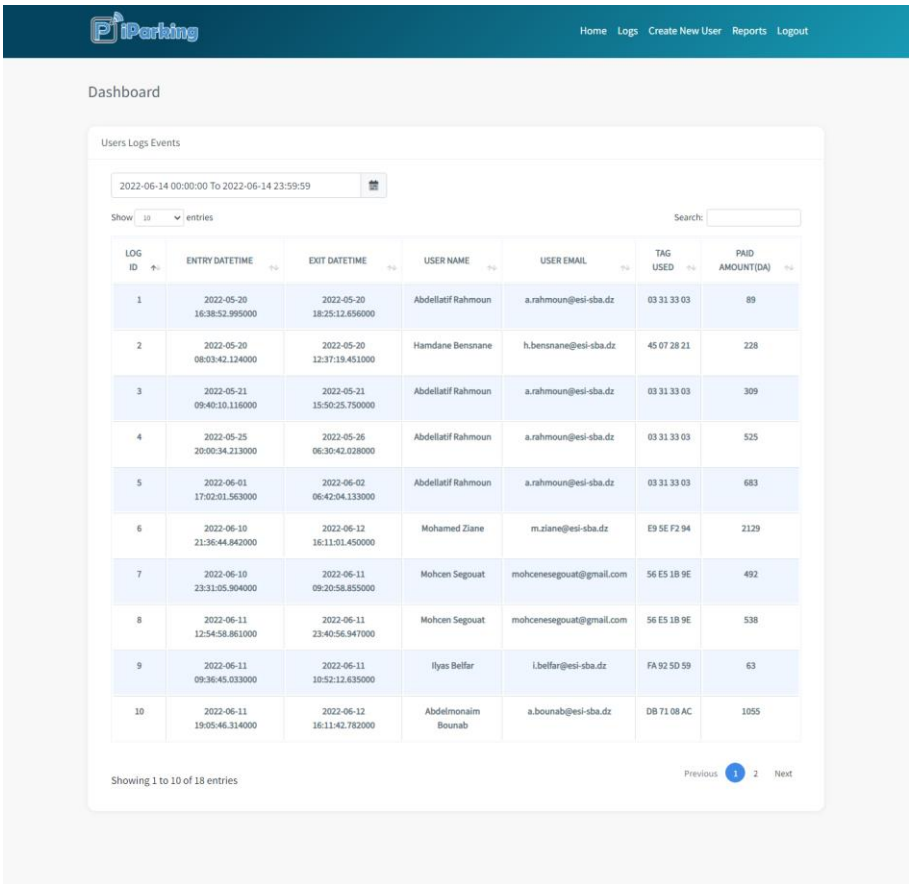

Figure27: Users Logs UI

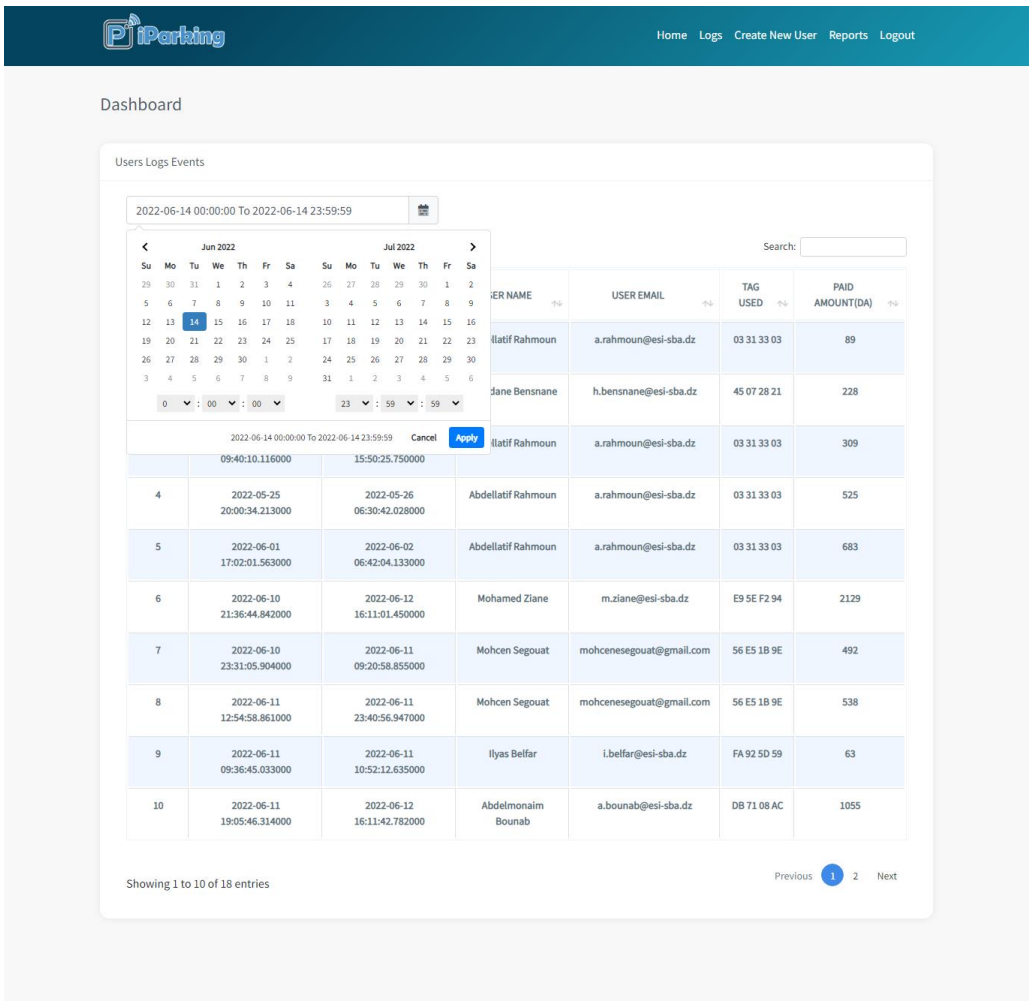

Figure27.1: Users Logs UI

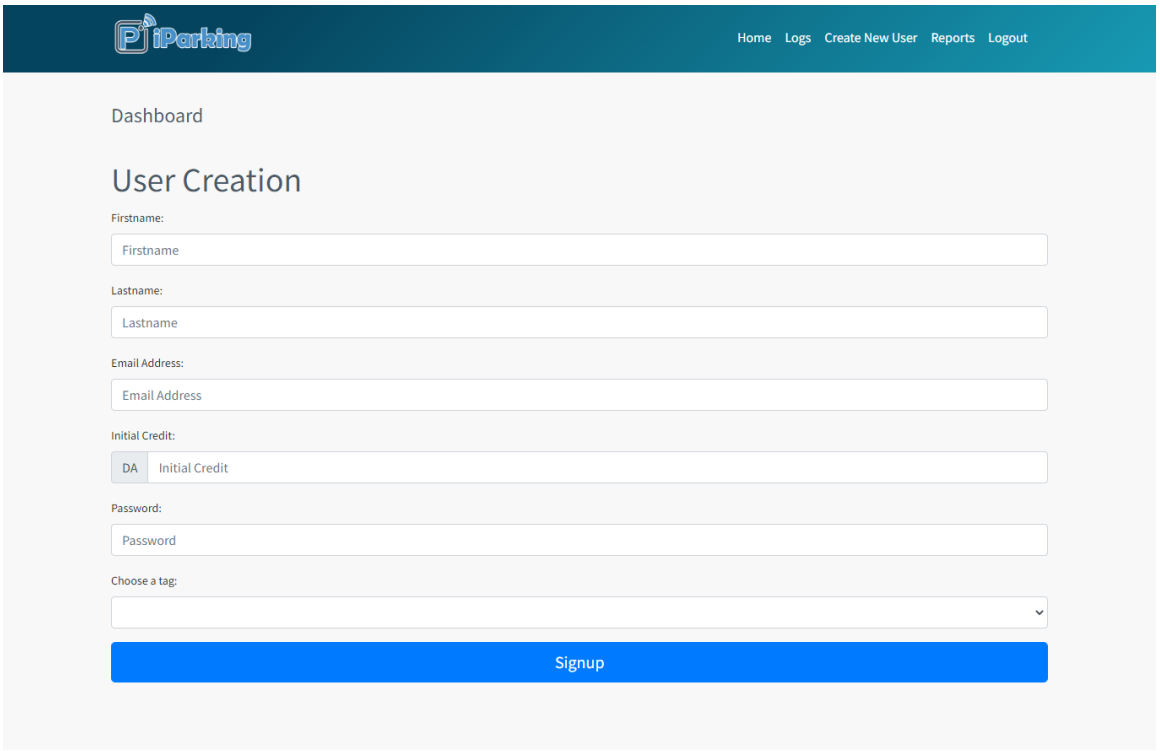

Figure28: Create New User UI

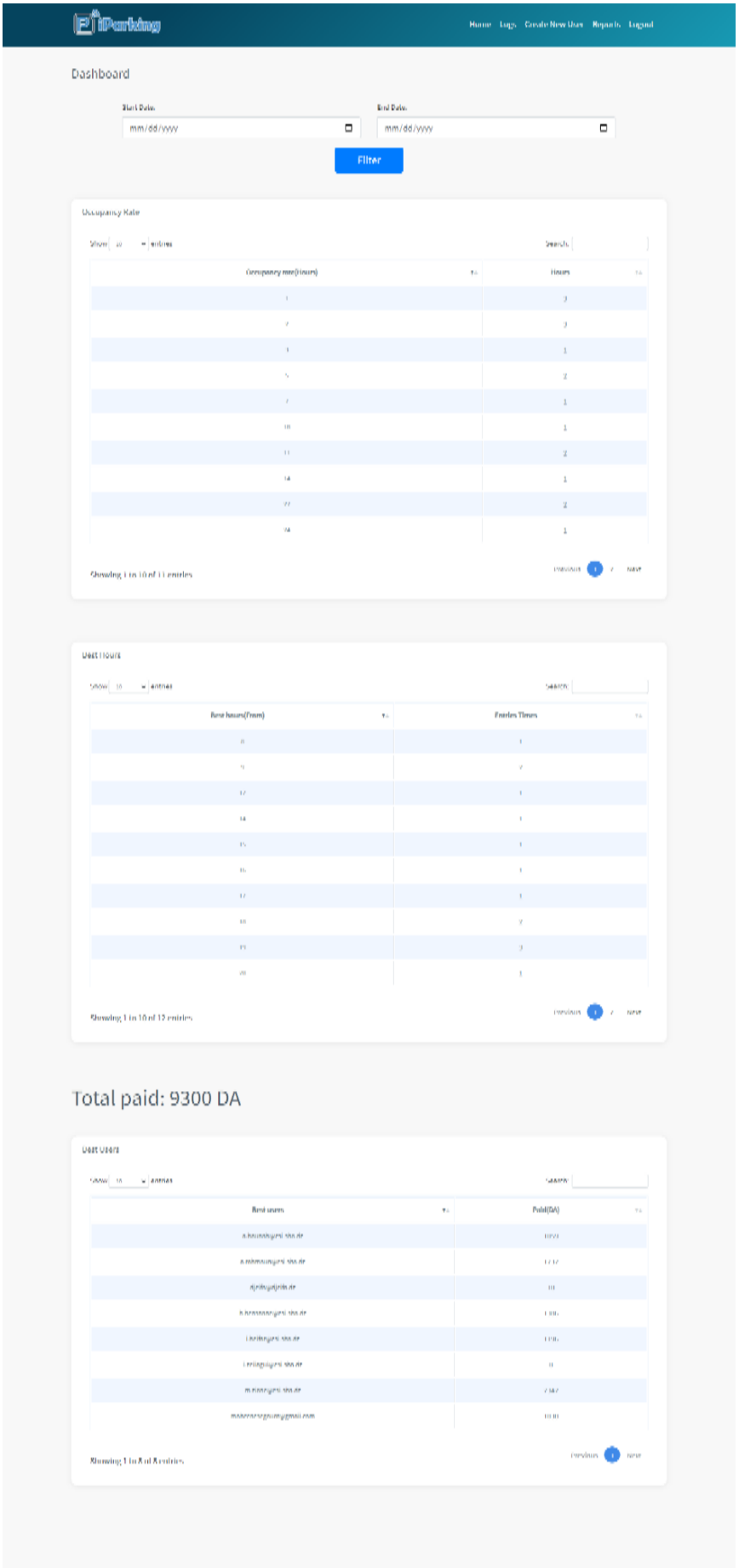

Figure29: Reporting UI

6. iParking Applications:

Our IoT Parking System is designed to give access to already registered clients that we have their accounts and can charge them, so our app will fit to be a private IoT Parking System that has its own clientele that can have access.

- Residential Parking (Only residents have access).
- Parking Spaces in office buildings (Only employees have access).
- Airport/Railway Parking (Only registered clients have access).
- Mall Parking (Only registered clients have access)… etc.

7. iParking Advantages:

This IOT smart parking system provides the following advantages:

- Automatic Parking System with Zero Human Intervention.
- RFID Scan for Access Verification.
- Automatic Gate Barriers for Entry/Exit.
- Online Parking Slot availability.
- Easy to Use System.

8. Bill of Materials and Cost:

Part	Supplier	Unit Price ($)	Quantity	Total Prince($)
NodeMCU ESP8266	ESI-SBA	5.5	2	11
RFID Reader RC522	ESI-SBA	8	2	16
Servo Motor SG90	ESI-SBA	3	2	6
LCD I2C	ESI-SBA	9	1	9
RFID Tags	ESI-SBA	2	10	20
Parking Model Materials	-	20	1	20
Energy / PC	-	15	1	15
			Total	**97 USD**

Figure30: The Project's Bill

9. The Model:

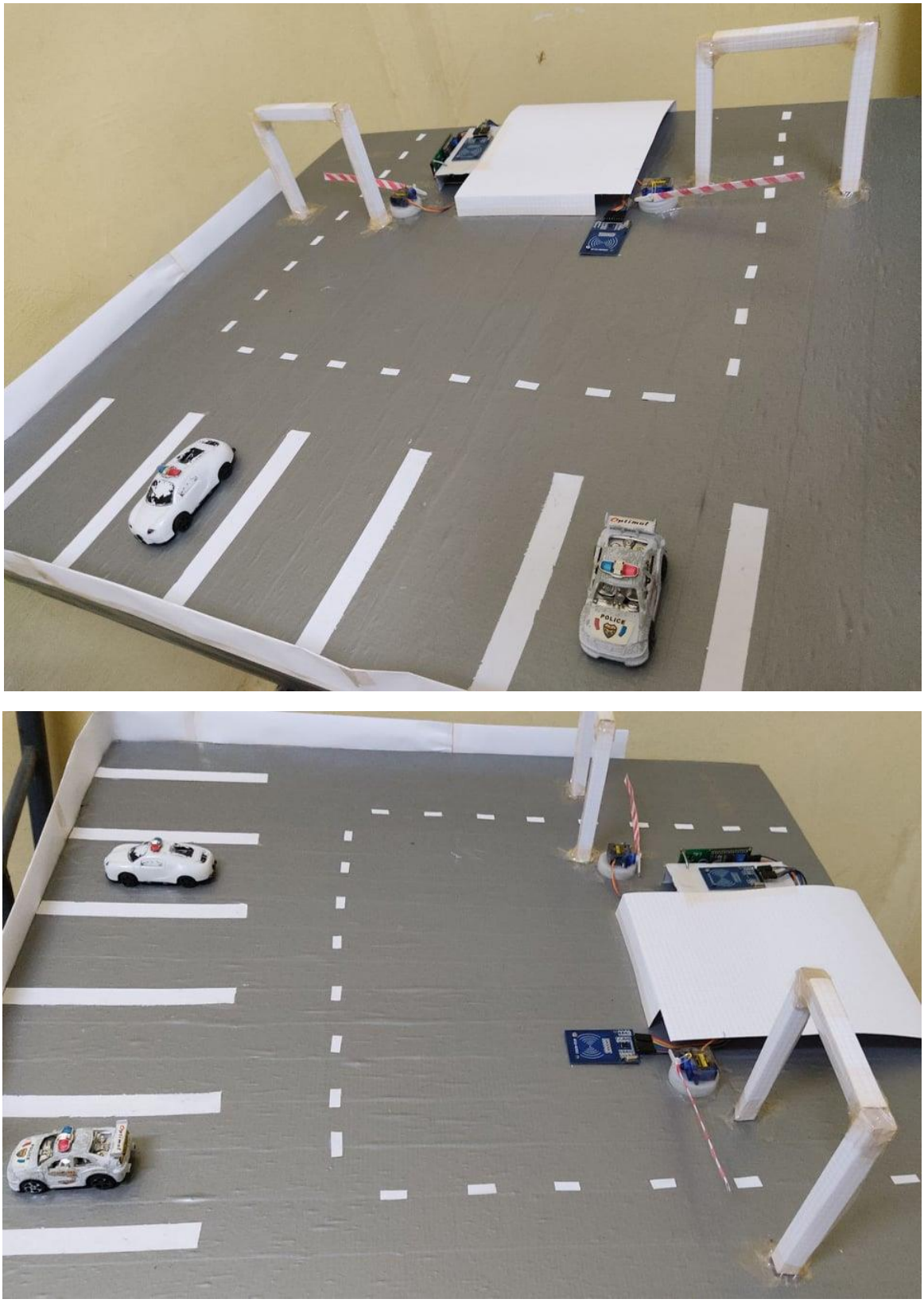

Figure31: Parking Model

Conclusion:

Overall, the implementation of the Smart Parking system was a success. This senior design project has been a valuable learning experience over the past semester, providing insight into design philosophies and the Internet of Things. Information learned from this project will be useful moving forward into our 5th in school, and will lead to better engineering in the future.

Perspectives:

Several improvement proposals are possible for our smart parking system solution, among them we can name:

- Expand the project with several car parks.
- Use Google Map to locate parking areas.
- Create a model similar to parking spaces in the application, to guide the driver to his destination.
- Implement a more improved system that allows users to reserve, cancel or update their parking reservation remotely.
- Use different intelligence technologies to detect and monitor cars.

Table of Figures:

Figure	Title	Page
01	IoT's domains of use	05
02	IoT Smart Parking System Architecture	06
03	Our IoT Smart Parking System Architecture	07
04	NodeMCU ESP8266	08
05	RFID Reader and RFID Tag	08
06	Servo Motor	09
07	LCD	09
08	Python Logo	10
09	Flask Logo	10
10	C++ Logo	11
11	HTML5 Logo	11
12	CSS3 Logo	11
13	JavaScript Logo	11
14	Bootstrap Logo	12
15	Arduino IDE Logo	12
16	MongoDB Logo	12
16.1	HTTP Logo	13
16.2	HTTP Request/Response	13
16.3	Properties of Request Methods	13
17	Use Case Diagram	14

Table of Figures:

Figure	Title	Page
18	Class Diagram	15
19	Activity Diagram “Parking Access”	16
20	Activity Diagram “Parking Exit”	17
21	iParking Logo	18
22	Login UI	18
23	User Dashboard UI	19
23.1	User Dashboard UI	19
24	Edit Profile UI	20
25	Add Balance UI	20
26	Admin Dashboard UI	21
27	Users Logs UI	21
27.1	Users Logs UI	21
28	Create New User UI	22
29	Reporting UI	23
30	The Project’s Bill	24
31	Project Model	25