DUBLIN BUSINESS SCHOOL

REQUIREMENT ANALYSIS COURSE WORK

NAME: MAUSAM SUDHIR JADHAV

STUDENT NO: 10605724

DBS PARKING BOOKING APP

TABLE OF CONTENTS:

7. APPENDIX

1. INTRODUCTION AND SCOPE:

As per data, Drivers spend around 17 hours per year searching for empty spaces, resulting in wasted time fuel and vehicle emission. With ever-growing number of cars, the vacant spots remain limited and hard to find. Its challenges increase when finding parking spot around an Institution.

Institution like Colleges, Shopping Malls, Corporate Houses, Restaurants, Stadiums, and Residential Complexes has congested area wherein they are situated. So if you are just visiting for a while, it can be really hectic to find a vacant parking space during peak hours. Dublin Business School is one such Institution that is located in largely populated urban area where finding a proper parking spot is bit more frantic. The Drivers visiting this institution are in distress most of the time because of traffic conditions and trouble finding Parking Space. Furthermore, the process to find an empty parking space is frustrating because the drivers need to take a detour and wait for a parked car to release a parking space. Thus in this fast-paced growing society we have to solve this issue using Tech-friendly approach for speed, efficiency and customer convenience.

With the emergence of mobile technologies, which allow a mobile device to communicate with each other and even with computerized system. We can develop an application which contributes towards exploring the solution to solve parking management issue.

This project covers the requirement and development of an app that book parking spots per person available at Dublin Business School. It will analyze and put forward important aspects with predetermined stages and execute it with constant improvisation through feedback at each stage.

2. SOFTWARE DEVELOPMENT LIFE CYCLE:

SDLC is the process used as the framework for software development. We can also define it as a Blueprint for completing each step of the lifecycle for software development. Each step of SDLC is called a Phase.

Following are its phases which are implemented in the sequence as shown:

1. Requirement Analysis
2. Design
3. Development
4. Testing and Validation
5. Deployment
6. Maintenance

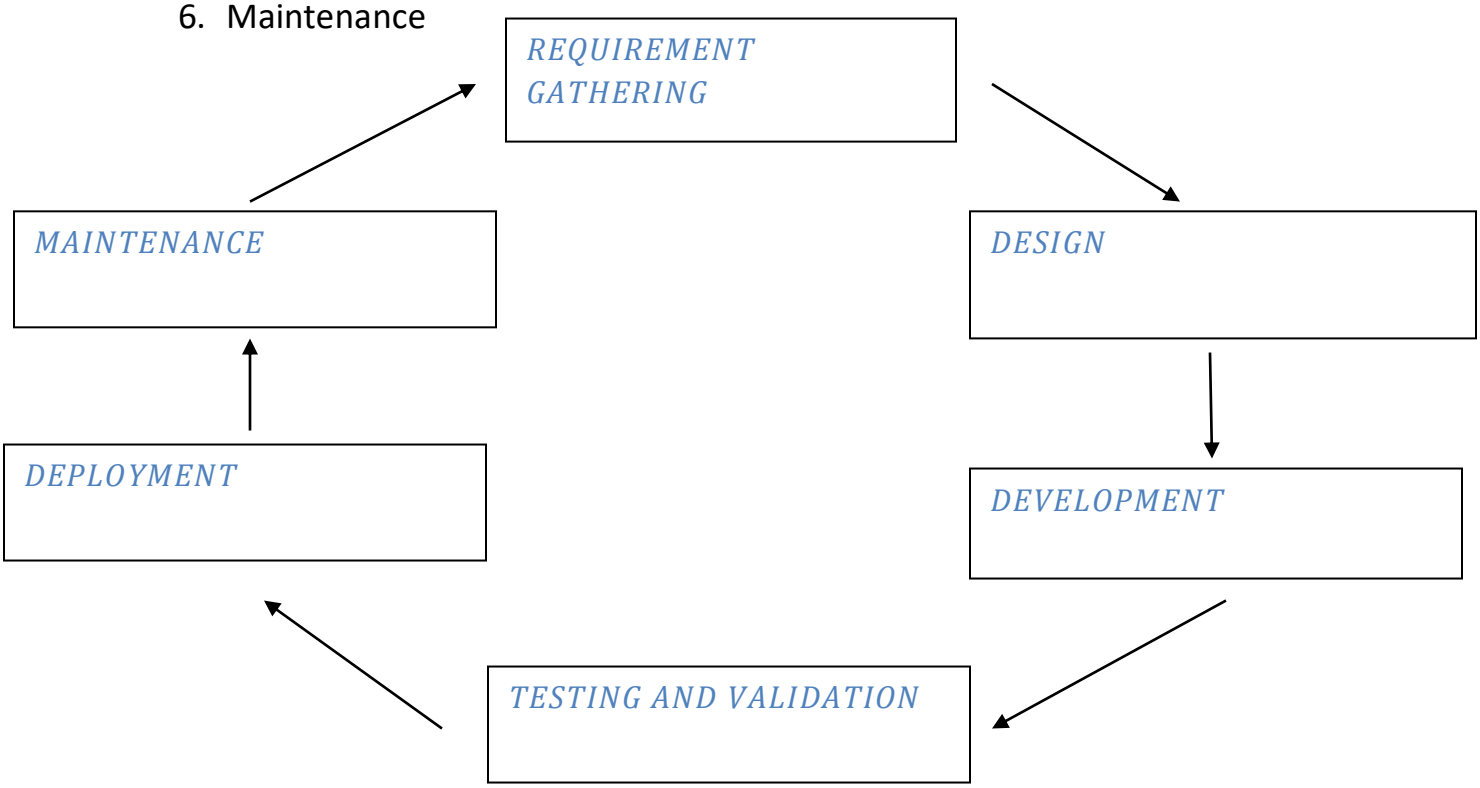


Fig2.1. A Pictorial Representation of the SDLC Model

They are also various SDLC models that follow different steps unique to itself to ensure for success, they are as follows

1. Waterfall

2. Incremental Model
3. V-Shape Model
4. Spiral Model
5. Agile

REFERENCE: Doctor Appointment System - Sample Assignment.pdf

3. METHODOLOGY:

3.1 AGILE DEVELOPMENT

The term agile development refers to the chain of rapid development, design and deployment to get desired outcome through constant feedback. It is the opposite of plan-driven management structure, where the project manager lays plan to the developers' team and the plan workflows. However, in Agile, the first section is always the planning part, so we know what we are about to build and then we break down the whole application into small chunks of codes or services and we work on small services, one service at a time, Ensuring we first follow micro service model and then macro service model and don't affect application in general. So we plan, design, develop, Test, Deploy and review each aspect of the project with collaboration and cross-functionality among separate unit teams and customers and with constant upgrading of Software. We do all this in Iterations or number of smaller cycles called sprints, with every iteration new features is added to the product, resulting in gradual project growth. Also specific set of task or requirement have to be completed in priority so that we don't worry about other micro services at the same time.

At the end of each Sprint a potentially shippable product is delivered.

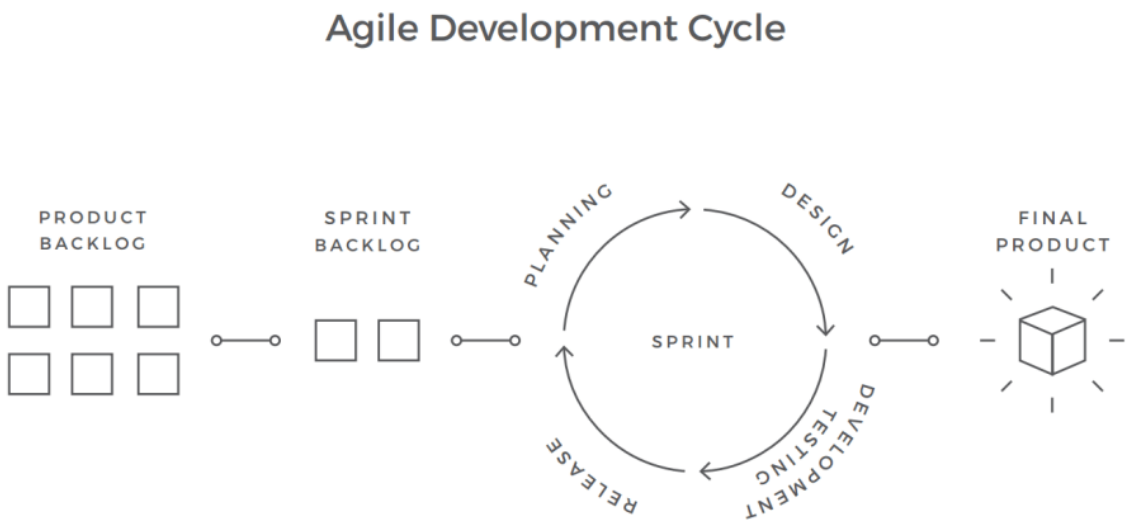


Fig. 3.1 AGILE DEVELOPMENT CYCLE DIAGRAM

REFERENCE: https://www.altexsoft.com/whitepapers/agile-project-management-best-practices-and-methodologies/

3.2 WHY AGILE DEVELOPMENT MODEL FOR AN APP THAT BOOKS PARKING SPACE

Responsiveness, adaptability, and efficient project management all seem ideally designed for an industry like parking management, which involves so many different individual actors, enforcement objectives, and changing circumstances. An Agile approach to parking management can be designed around tasks that are time- and team-oriented, with daily reviews and constant upgrades that emphasize how everyone's been handling their assigned tasks, as opposed to worrying solely about what someone in charge wants them to do. As opposed to top-down, parking professionals–from field officers to IT teams to managers–can keep track of themselves with an agile methodology and respond quickly to necessary changes or issues that develop as they develop. In light of the way

technology itself is changing the game for parking lot solutions with new systems like pay-by-plate or emerging new technologies like Parking Assist System that takes control of car from driver, judge the size of the space via sensors and help you maneuver into the spot, the Agile methodology seems like an obvious fit.

We plan and implement this parking application in iterations or sprints as we look forward to align with evolving technology and suit consumer demands through constant feedback from them.

REFERENCE: https://www.gtechna.com/bringing-agile-software-methodology-to-parking/

4. FEASIBILITY STUDY AND REVIEW

Our primary role was to build a design concept for a parking app for Dublin Business School, so at the start of the project, 3 UI/UX designers and a project manager were involved in this process. It took 2-3 weeks for designers to conduct a detailed analysis of competitors and research to build a unique UI design and clickable prototypes for investors' presentation.

As we progress and the model prototype is well defined, we go further with core software building on iOS platform. Major Developers will be part of this team like iOS and backend developers, along with QA engineers.

REFERENCE: https://cadabra.studio/blog/how-we-built-parking-app-design/

5. SOFTWARE IMPLEMENTATION AND ENGINEERING:

5.1 REQUIREMENT ELICITATION AND ANALYSIS

5.1.1 BUSINESS AND SYSTEM REQUIREMENTS

There are three main stages in conducting a thorough requirements analysis:

1. The first step is to gather the requirements by collecting business process documentation and conducting interviews with stakeholder and for Investments, general Policies, and special functionalities.

2. We also created a survey using this questionnaire form link for Stakeholders: https://docs.google.com/forms/d/e/1FAIpQLScrghnaZkB_wXK0rfVm_79ufFJUJApEiNddZ2d554O42IPPUA/viewform?usp=sf_link

2. Next, we analyze and validate the requirements from stakeholders and survey responses, evaluating whether they're clear, complete, consistent, and unambiguous.

3. Finally, record the requirements and monitor their implementation throughout the project.

Further we divide the requirements into two types based on Survey and Stakeholders criteria:

5.1.1.1 FUNCTIONAL REQUIREMENTS:

- These requirements specify the services system should provide, how The system should react to particular inputs, and how the system should behave in particular situations.
- Meeting these requirements is obligatory under any circumstances.

Following are some of the functional requirements for this app:

a) Nearly precise cost estimation of all the basic hardware like processor, laptops or computers for project team, software development like interface, Advanced GPS System graphics, contract with customers or institution.
b) Ease of registration or sign up interface. Users should be allowed to sign in through their Google accounts or integrated profiles to speed up their registration process.
c) Details of how transaction should be occurred between user and administrator.

d) The App should display accurate number of parking spaces available on the Interface screen. Also it should be able to show every minute parking status to the user.
e) If the parking area is full, it should be able to calculate the earliest time at which a certain parking slot in that area will be empty and process it as message to the user.
f) Defining Time constraints of booking a slot like maximum parking duration allowed and closing time.
g) The time frame of booking a slot should be current or future, past dates can't be permitted.
h) Specifics related to legal or regulatory compliance
i) The External Interface of the system and its mobility and convenience for users.
j) Procurement and modification of technology that monitors and records the parking space availability.
k) Should not allow more vehicles than allocated space for each category.
l) Implementation of QR code Scanner technology at parking lots if the person wants to pay the cash.
m) Linking the mobile application with parking monitoring systems in the respective Institutions.
n) Developing GPS technology within the app with good Navigation Features and mobility so that user can access it with ease.
o) The Staff should be able to get discount on parking rates by putting identification credentials on the app.
p) Incase if there is mismanagement in allocation of parking space for example, if a certain parking lot is allocated to a certain driver and if another person occupies it before without authorization then the monitoring system should immediately inform the concerning authorities and tow the vehicle.
q) Another challenge we needed to solve is to ensure that information about traffic jams and the travel time will be synchronized. For example, if you need to book a spot in 30 minutes, and you are 20 minutes away from the

parking lot, but there are traffic jams on your route, the system will notify you that you cannot make it, and you need to change the booking time.

5.1.1.2 NON-FUNCTIONAL REQUIREMENTS

- Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users.
- It is mainly concerned with the performance of the system and focuses on quality of system.
- These non-functional requirements usually specify or constrain the characteristics of system as a whole.
- They may relate to the crucial system properties like reliability, response time and memory use.

Following are some of the Non-Functional requirements for this app:

a) The Database of this app should be maintained for several years.
b) Our app is entitled primarily for Dublin Business School so it should partly incorporate its signature color or style in User interface design
c) Able to serve a million users smoothly and system should not crash due to overload.
d) App should be quicker
e) System should be operated with ease and shouldn't receive backlash from users.
f) Advance booking payment options should be implemented along with extension of session and transaction history features should be available.
g) The legal compliance should be taken into consideration while designing the app, like the European privacy law.
h) The app should be advertised well in the media showcasing its benefits and offers so that it reaches maximum people.

i) The app should be able to identify the employee or student of certain institutes by cross checking their name and id on that institute that have mutual contract with stakeholders. So that they are able to get discount.

5.1.2 HARDWARE AND SOFTWARE TOOLS

Following were the tools used for software development:

- Android Studio V.
- Software Development was performed using ASP.NET Microsoft Visual Studio Framework.
- The database was implemented using Microsoft SQL Server and SQL Server Management Studio (SSMS).
- Intuvision VA Software for Parking Space Tracking.

REFERENCE:
https://www.temjournal.com/content/94/TEMJournalNovember2020_1357_1363.pdf

4.2 ARCHITECTURAL AND SYSTEM DESIGN:

As we go through this applications design process it is of utmost importance that we develop this in iterations or step so that we can analyze challenges and upgrade the system.

5.2.1 INTERFACE DESIGN WITH ARCHITECTURAL LAYOUT

At First Iteration our team conducts research and data gatherings of whole parking lot infrastructure in Dublin Business School. We then create a representation of abstract model view pattern of this system for its generalized overview

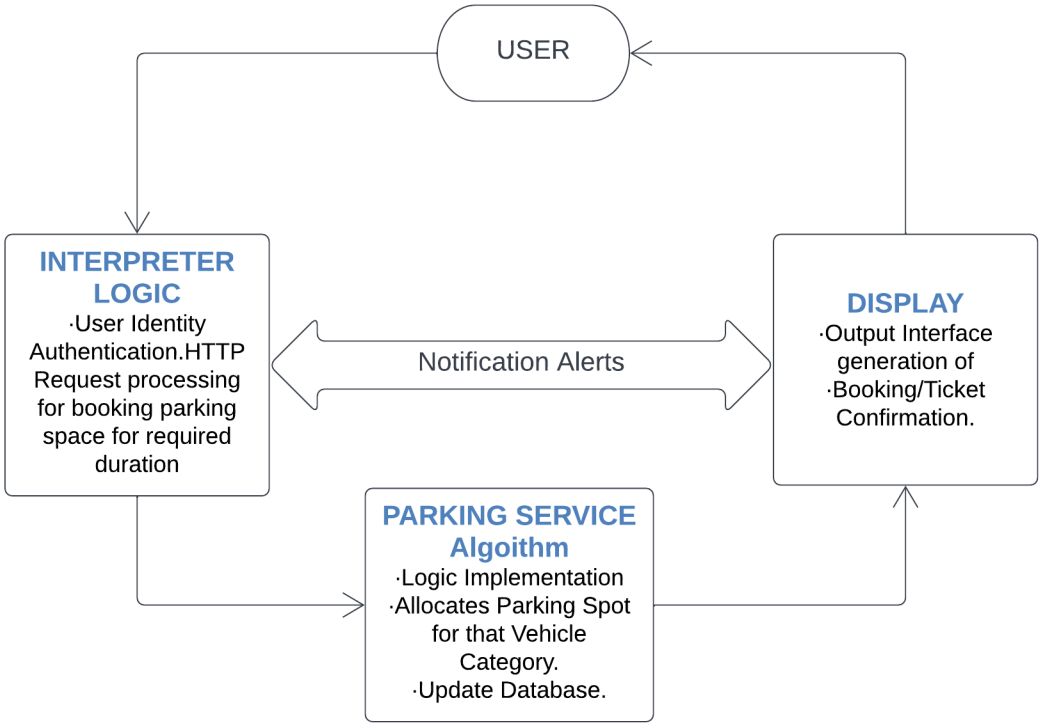


FIG 5.2.1 MODEL-VIEW-PATTERN DIAGRAM

We inspect the monitoring system and gather data of size and total no of parking slots per floor and the blueprint of entire area. We also check if the parking slots in each floor can accumulate and make space for three main categories of vehicles i.e. Small (e.g. Bike), Medium (e.g.: Cars) and Large vehicles (e.g.: Bus). We also determine the scale and if it's feasible to put in one machine

we summarize the data as follows:

For a Parking Lot in Dublin Business School:

5 Floors and 200 parking slots:
5*100 = 1000 Parking Slots

Data size for parking slot in one parking lot:

(Total Parking Slots)*(Data Size per parking slot)
1000*1K = 5M (which fits in one machine)

Data Size for Ticketing per day:

(No of Parkings in a day)*(Size per ticket)
10k*100bytes = 1MB

Data for 10 years:

10*365*1MB = 3.65GB (Which fits in one Machine)
So data for slots data and tickets data potentially fits in one machine.

Now we construct APIs which are generally two types with regards to this Application:

PUBLIC APIs

```
PARKING SERVICE {
Ticket entry (VehicleType type):// returns ticket
Double exit (long ticketID):// returns price
}

Ticket {
Longid:
Date entryTime, ExitTime;
ParkingSlot slot;
String VehicleNumber;
}
```

ParkingSlot {
Long id:
VehicleType type:
State state;
//..other metadata about floor layout.
}

enum VehicleType{Small, Medium, Large}
enum State{Occupied, Unoccupied, UnderRepair}

5.2.2 SYSTEM DESIGN

In Second Iteration we develop a more sophisticated Layout of Parking Service Process as Shown in Diagram:

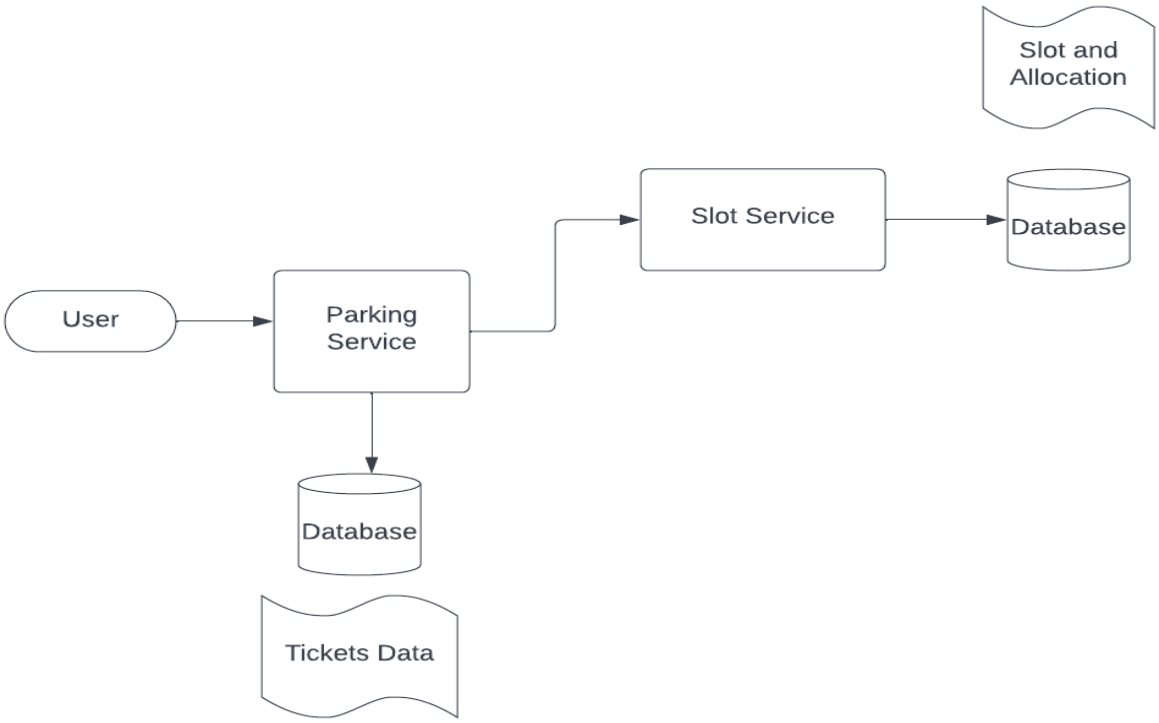


FIG 5.2.2 HIGH-LEVEL SYSTEM DESIGN

The above Figure illustrates the High level Design of Parking Process. We have Four Major Components here i.e. User, Parking Service Model, Slot Service, Database. Each of this components have specified task.
The process flow takes place with the user login in to the DBS parking app for parking their car. The parking service displays the number of available parking spaces and when user clicks on a particular empty parking space, the parking service asks user to input the type of vehicle and its number and allocates that parking slot to the vehicle. Once it receives the parking slot, it creates ticket entry in ticketing database with vehicle number and unique Id and returns the Ticket to the user. The parking service model also informs the user whether the parking area is full or not and if it's full then it notifies the user the time at which a particular parking slot or any parking slots will be empty.
The Slot Allocation Service is CCTV Monitoring and Management System which Enables to find empty slot for that vehicle. It allocates the Parking Slot for that Driver on basis of its management strategy. After allocating a parking slot the parking service updates the state of parking slot to Occupied in database. At exit the parking service or Admin gets the entry and exit time and Slot no from ticketing database and it updates the state of that slot to empty. It also computes the price on input details like duration and Vehicle type and based on policy it generates price on Ticket. However the users who have monthly subscription can show their Digital Id or scan their digital ticket on ticket Counter.

We earlier created Public APIs for Parking Service but while designing we also need to create APIs for two more service i.e. Slot Service and Parking Service.

INTERNAL APIs

```
Slot Service{
ParkingSlot allocate(VehicleType type);
boolean unallocate(long ParkingSlotId);
}

Pricing Service{
```

double Calculate(VehicleType type, Date in Time, Date outTime);
}

5.2.3 UML DIAGRAMS

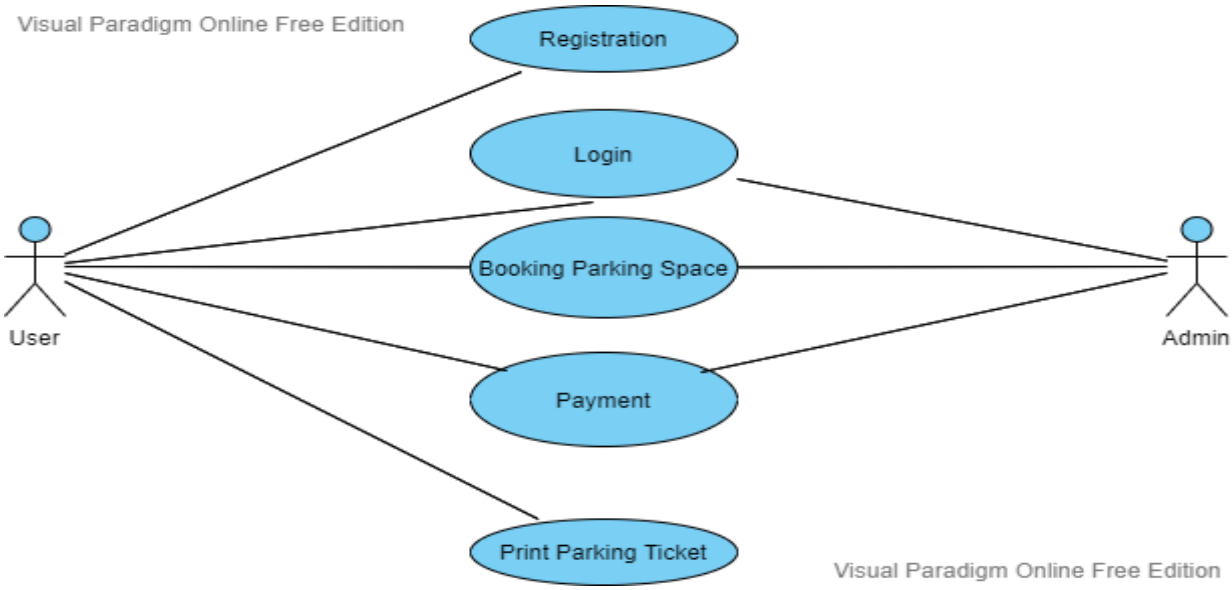


FIG 5.2.3.1 USE-CASE DIAGRAM

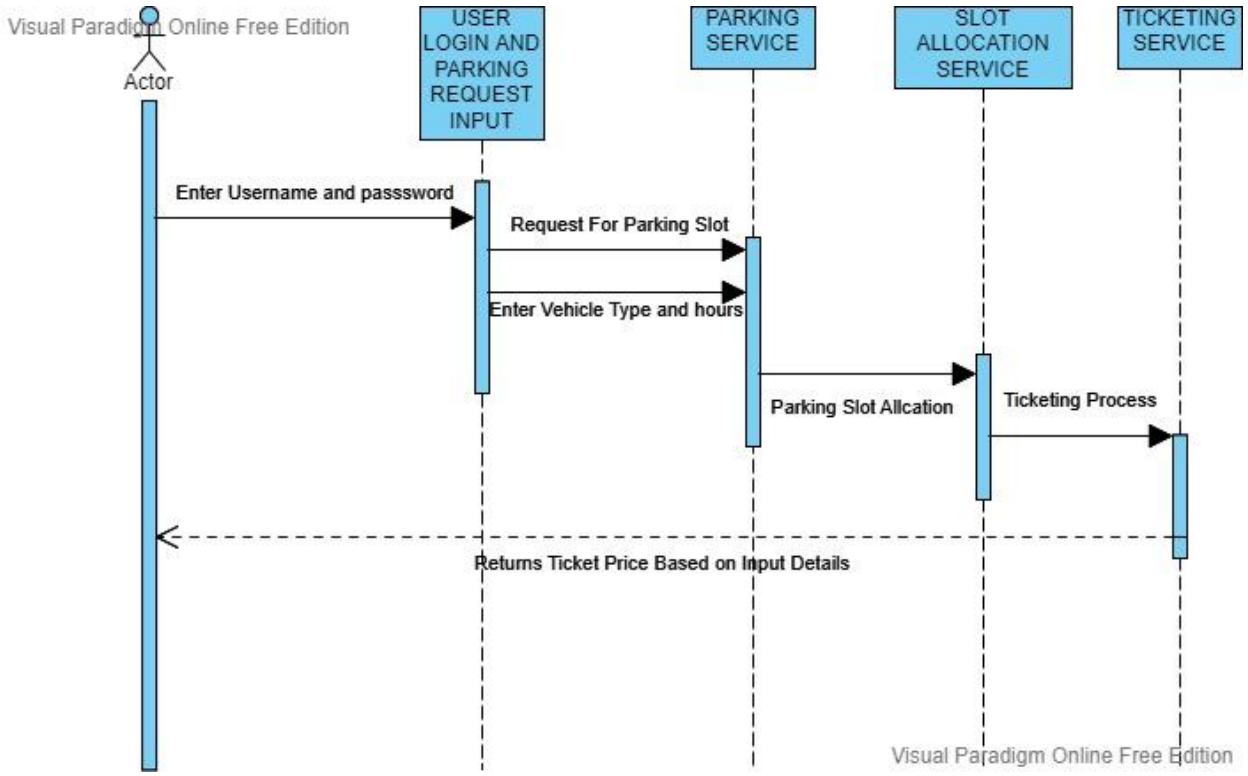


FIG 5.2.3.2 SEQUENCE DIAGRAM

5.3 SYSTEM INTEGRATION AND TESTING:

5.3.1 TEST CASES

The 'black box' testing technique was used to check if the system functionality meets user expectations and to ensure that the application was working correctly. Firstly all the components of the System were integrated right after carrying out Unit testing. After that ten users interacted with the system interface, to test the following five test scenarios:

- Log in to the application using DBS ID.
- Create a new booking by choosing time.
- Immediate Slot Allocation upon entry using monitoring system with god management Capabilities.

- The application should be able to detect the current user when they close the application without signing out.
- Alert message should notify the user before they confirm the reserve or the checkout button.

REFERENCE: TEMJournalNovember2020_1357_1363

6. DEPLOYMENT

The Proposed System was show to have passed all tests and operated according to expected specification allowing its use for Dublin Business School Staff members and Students and the events of increasing scalability.

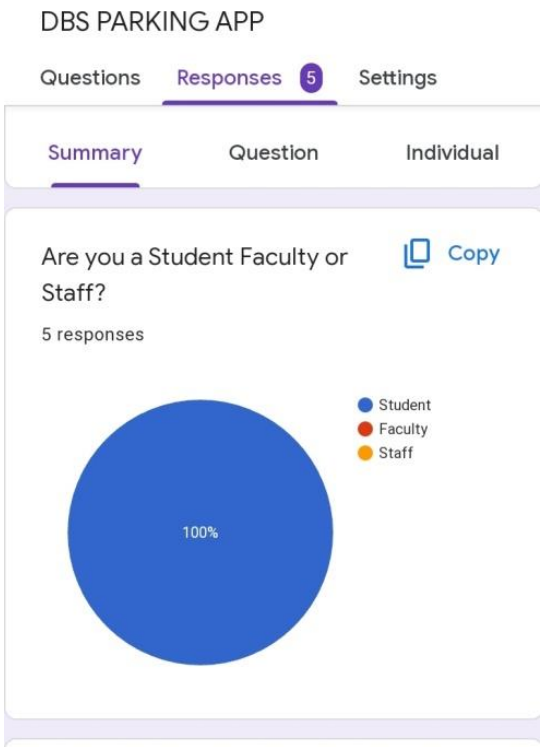
Finally, we deploy this application on Dublin Business School main Website and Google Playstore. We also conduct an informative session for students and faculty members of DBS regarding this app.

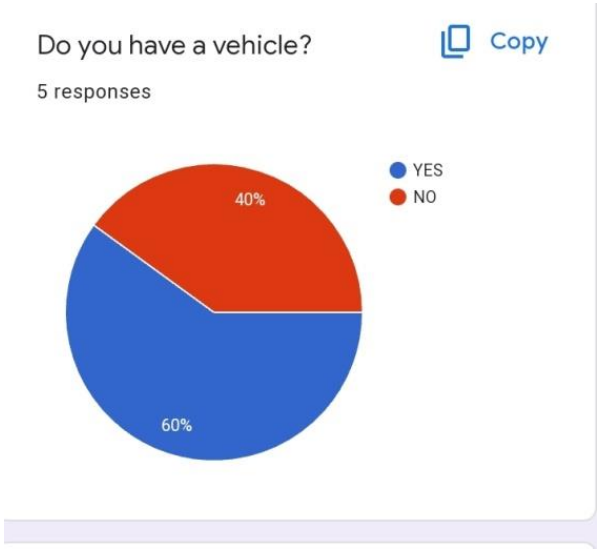
REFERENCE: TEMJournalNovember2020_1357_1363

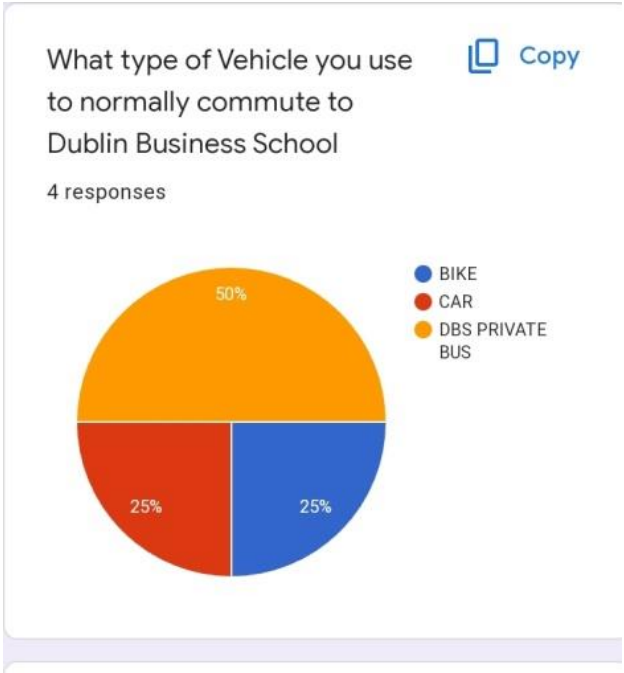
7. APPENDIX

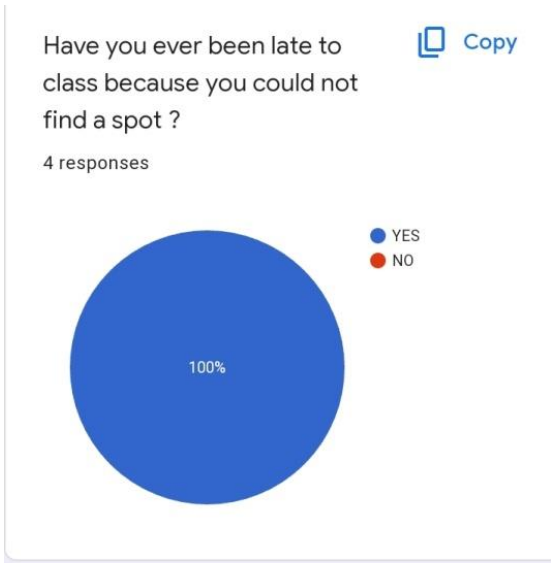
7.1 SURVEY RESULT:

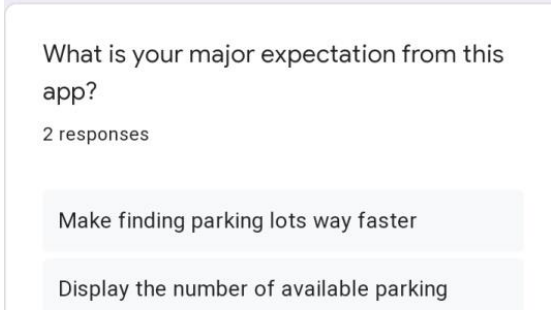
So we get the responses from the questionnaire form as given in charts below:

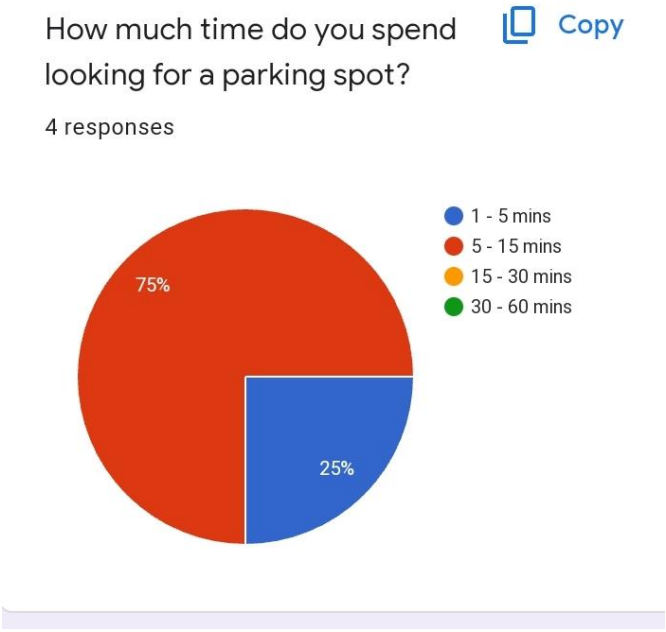












REFERENCE FOR SURVEY QUESTIONS: https://www.surveymonkey.com/r/57F9R9J