International School

**Capstone Project 1**

*CMU-SE 450*

**ARCHITECTURE DOCUMENT**

*Version: 1.2*

*Date: 25/10/2020*

**DTU SMART PARKING SYSTEM**

**Submitted by:**

Nguyen Thi Bich Ni

Nguyen Huu Tuong

Luong Phan Thanh Dat

Duong Le Duc Thinh

**Approved by**

Msc. Nguyen Duc Man

Bsc. Nguyen Chien Thang

**Proposal Review Panel Representative:**

Name Signature Date

**Capstone Project 1- Mentor:**

Name Signature Date

MSc. Nguyen Duc Man

BSc. Nguyen Chien Thang

**PROJECT INFORMATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project acronym** | DSParking | | |
| **Project Tittle** | DTU Smart Parking System | | |
| **Start Date** | 24/08/2020 | **End Date** | 06/12/2020 |
| **Lead Institution** | International School, Duy Tan University | | |
| **Project Mentor & contact details** | MSc. Nguyen Duc Man  Email: mannd@duytan.edu.vn  Tel: 0904235945  BSc. Nguyen Chien Thang  Email: chienthangplus@gmail.com | | |
| **Product Owner** | Luong Phan Thanh Dat  Email: mark.dat.code@gmail.com  Tel: +84 702 303 444 | | |
| **Scrum Master & contact details** | Nguyen Huu Tuong  Email: nguyenhuutuong999@gmail.com  Tel: +84 766 693 571 | | |
| **Team members** | **Name** | **Email** | **Tel** |
| Nguyen Huu Tuong | nguyenhuutuong999@gmail.com | +84 766 693 571 |
| Nguyen Thi Bich Ni | [bichninguyen1999.nn@gmail.com](mailto:bichninguyen1999.nn@gmail.com) | +84 702 303 444 |
| Duong Le Duc Thinh | [duongthinh1999317@gmail.com](mailto:duongthinh1999317@gmail.com) | +84 967 893 387 |
|  | Luong Phan Thanh Dat | mark.dat.code@gmail.com | +84 369 187 157 |

|  |  |  |  |
| --- | --- | --- | --- |
| **DOCUMENT NAME** | | | |
| **Document Title** | Architecture Document | | |
| **Author(s)** | C1SE.03 Team | | |
| **Role** | Team members | | |
| **Date** | 06 – Sep – 2020 | **File name:** | Architecture\_Design\_v1.2 |
| **URL** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **REVISION HISTORY** | | | |
| **Version** | **Person** | **Date** | **Description** |
| **1.0** | Dat, Luong Phan Thanh  Thinh, Le Duong Duc | 25 – Sep – 2020 | Draft |
| **1.1** | Dat, Luong Phan Thanh  Thinh, Le Duong Duc | 15 – Oct – 2020 | update |
| **1.2** | Dat, Luong Phan Thanh  Thinh, Le Duong Duc | 25 –Nov– 2020 | update |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Approval**  The following signatures are required for approval of this document | | | |
| **Mentor** | MSc Man, Nguyen Duc | **Signature:** |  |
| **Date:** |  |
| BSc Thang, Nguyen Chien | **Signature:** |  |
| **Date:** |  |
| **Scrum Master** | Tuong. Nguyen Huu | **Signature:** |  |
| **Date:** |  |
| **Team Member(s)** | Dat, Luong Phan Thanh | **Signature:** |  |
| **Date:** |  |
| Thinh, Le Duong Duc | **Signature:** |  |
| **Date:** |  |
| Ni, Nguyen Thi Bich | **Signature:** |  |
| **Date:** |  |

**TABLE OF CONTENTS**

[**Project Information 1**](#_heading=h.1ci93xb)

[**Architecture Document 2**](#_heading=h.3whwml4)

[**Document Approvals 3**](#_heading=h.qsh70q)

[**1.**](#_heading=h.3as4poj) **Introduction 5**

[1.1 Project overview 5](#_heading=h.1pxezwc)

[1.2 Purpose 5](#_heading=h.49x2ik5)

[**2.**](#_heading=h.2p2csry) **Architecture driver 6**

[2.1 Business constraints 6](#_heading=h.147n2zr)

[2.2 Technical constraints 6](#_heading=h.3o7alnk)

[2.3 Functional requirement 7](#_heading=h.23ckvvd)

[**3.**](#_heading=h.ihv636) **Architecture overview 7**

[3.1 System context 7](#_heading=h.32hioqz)

[3.2 Component and connector 9](#_heading=h.1hmsyys)

[3.3 Workflow QR scan 12](#_heading=h.41mghml)

[3.4 Module view 13](#_heading=h.2grqrue)

[3.5 Allocation view 14](#_heading=h.vx1227)

# Introduction

## Project overview

DSParking - smart parking management system is a software to manage motorbikes in/out at the parking lot of Duytan University by using a combination of QR code and surveillance camera system. The smart parking system quickly processes the image and fully identifies the license plate. It is a multi-feature intelligent solution, which helps to manage quickly, simply, and easily with advanced features such as online payments, reports, statistics... In addition, the Software has the ability to integrate with many different software applications such as myDTU to retrieve personal information of students, link payment gate, and other software applications

## Purpose

This specification covers following:

* Brief specification of the project, high level requirement.
* Detail quality attribution.
* System context, sequence diagrams.
* Architecture presented by various view types: Component an Connect, Module view and Allocation view.
  1. **Business driver**

**Business Problems:**

The current, parking was used for parking tickets (paper tickets), which could clearly many disadvantages when using paper tickets such as:

* Relatively safe: It is easy for bad guys to steal the motorbike if the user accidentally drops the ticket.
* The number of students is too much, the parking lot becomes overloaded and difficult to manage.
* Parking as well as taking the motorbikes take quite a lot of time.
* Causing congestion at rush hour.
* Difficulty in managing revenue.
* Unprofessional and costly.

**Business Need:**

* Absolutely safe security.
* Easily control the number of motorbikes in the parking lot.
* Parking and taking motorbikes take place quickly
* Minimize traffic congestion.
* Easy to statistic and manage revenue.
* Cost savings.
* Highly professional, in line with current trends.

.

# Architecture driver

## 2.1 Business constraints

* Sources: 4 people.
* Project was started on: 24/08/2020.
* Project will be ended on: 08/12/2020.
* Project will be finished in 105 days (790 hours).
* Cost: $1896.

## 2.2 Technical constraints

**Technical to develop:**

* Programing Language: JavaScript, C#, Java
* Frameworks / Libraries: ReactJS, NodeJS, Bootstrap 4.0, SVM, Emgu, OpenCVDotNet, Aforge, Zxing.
* Database Management System: SQL Server 2014, Firebase
* Web server:

**Environment:**

* Web browsers: Google Chrome.
* Operation systems: Microsoft Windows 10, Android 4.2 and up
* IDE: Visual Studio 2019, Visual Studio Code, Android Studio.

## 

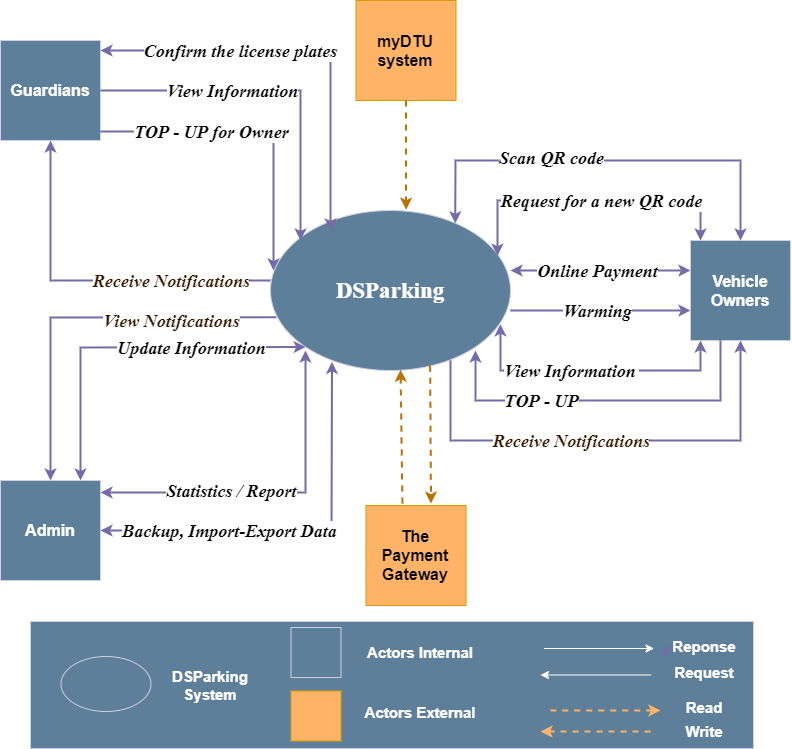
## 2.3 Functional requirement

References to Product Backlog specification of ProductBacklogV1.0.docx

# Architecture overview

This section shows the diagrams which bounds our target system and describes the architecture and interaction between components

## 3.1 System context



***Figure 1: System Context Overview***

**Students:**

* Request for a QR code
* Scan QR code
* Request Login/Logout to the system, System check authentication and response
* Request to view user information
* Request view list account-balance of user in system
* Request to display come in and out history
* Top up into user account
* Online payment
* Get notifications

**Lecturers:**

* Request for a QR code
* Scan QR code
* Request Login/Logout to the system, System check authentication and response
* Request to display come in and out history
* Request to view user information
* Get notifications

**Guardians**

* Request Login/Logout to the system, System check authentication and response
* Enter the student's username to:
  + Request to view user information
  + Request to display come in and out history
  + Top up into user account
* Enter the license plate in some cases cannot be identified

**Administrators:**

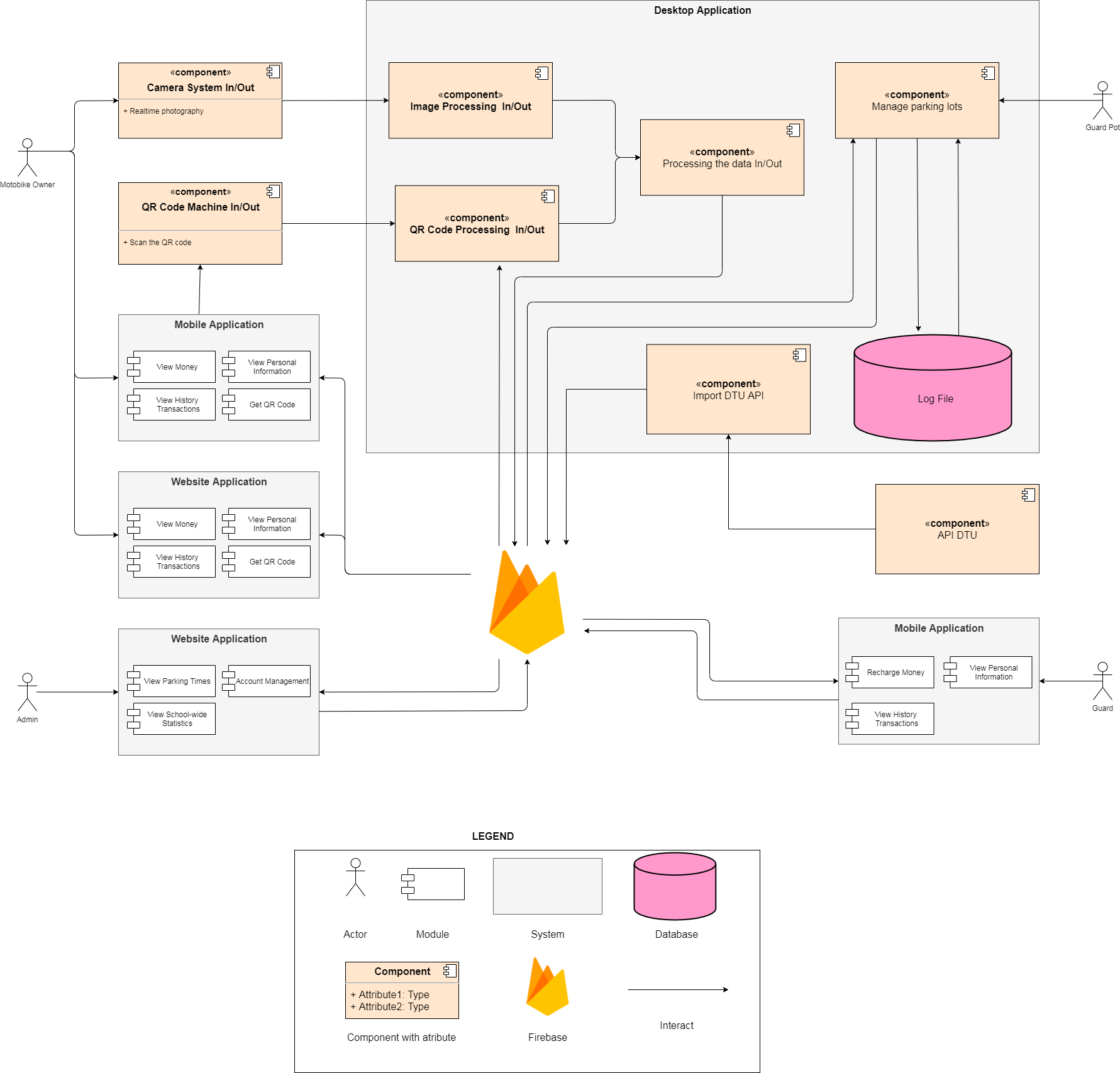
* Request Login/Logout to the system, System check authentication and response
* Insert, edit, and delete user accounts
* Revenue and total motorbike turn statistics
* Backup, import - export data.
* Get notifications

## 3.2 Component and connector

We mainly used a C&C view to argue and reason about architectural properties, quality attribute requirements, and functional requirements that the system must add here.

This view type partitions the system into components that have some runtime presence such as processes, objects, data stores, and connectors or that represent pathways of communication such as data flows and access to shared storage.

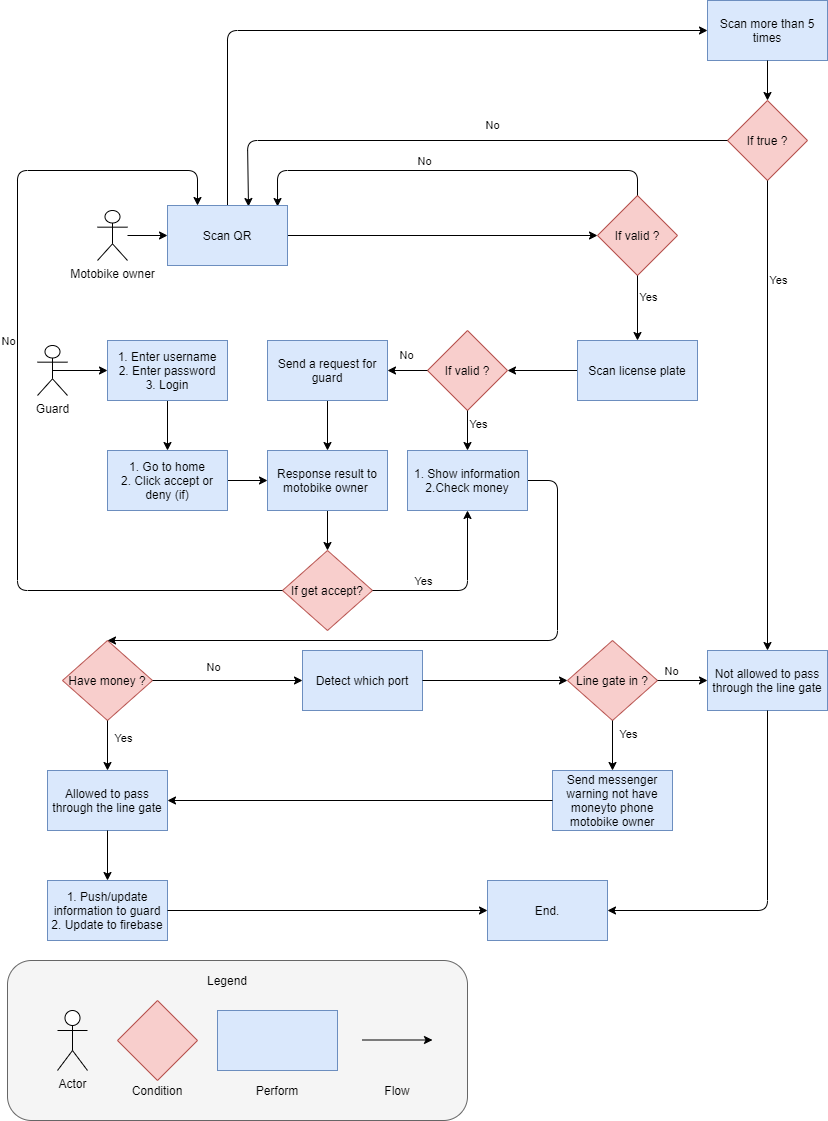
***Figure 2: Component and Connector View***



**Prose**

|  |  |
| --- | --- |
| **Element** | **Responsibilities** |
| Mobile application | Mobile application consists of two parts for the motorbike owner to use and protect. The motorbike owner uses the app to view parking history, view personal information and receive QR. Protection uses the app mainly to recharge the car holder. |
| Website application | Website application consists of two parts for the motorbike owner to use and the admin use. The motorbike owner uses the web to view parking history, view personal information, receive QRs and view statistics. Admin uses the web to manage users' accounts, advertise parking visits, view parking statistics,secure account management. |
| API DTU | API DTU is the address to provide information of students, faculty or security to the system |
| Desktop application | Desktop application is the place for processing QRs and images, then the processed data is sent to the firebase. Protect pot can use a desktop app to view detailed parking history of the user. |
| Database | Database is a component which contains information of parking man. All data was parked at that block. |
| Firebase | Receive data and interact with system in order to create notification in real time |

## 3.3 Workflow QR scan



***Figure 3: Workflow Qr scan***

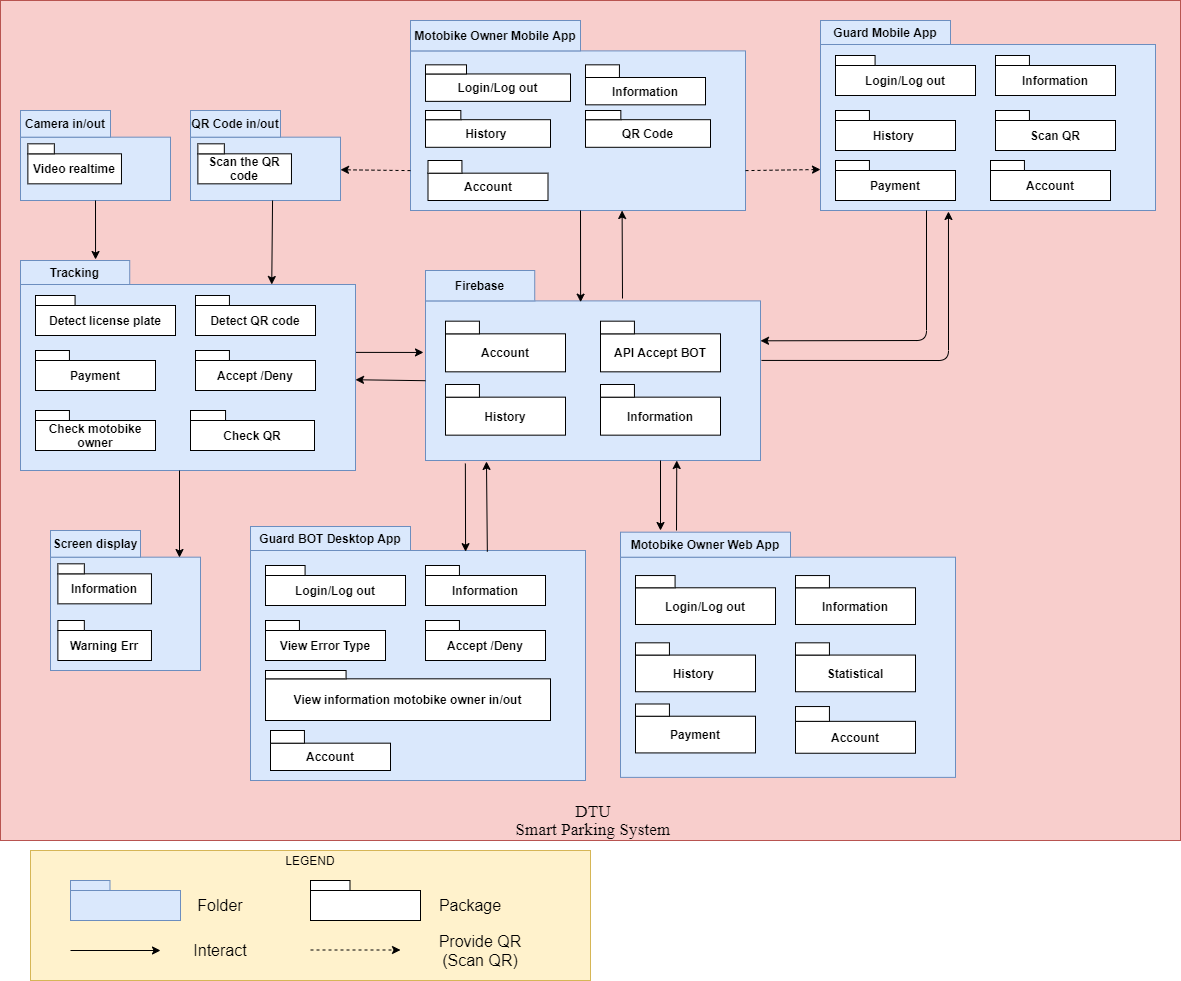
## Module view

This view type partitions the system into a unique non-overlapping set of

decomposable implementation modules. The goal is to show how the source code is

decomposed, as well as the dependencies between modules. In other words, it shows the

decomposition.



***Figure 4: Module View***

The elements in this view type are packages or classes or interfaces. Packages are hierarchically decomposable; hence, placement of a class or package inside another package in a diagram implicitly indicates an is-part-of relation. This view type also shows dependencies among implementation units. It tells developers what other modules must exist in order for their portion of the system to work correctly.

This view type was selected because it helps the following roles:

## · The project manager, who must define work assignments, form teams, and formulate project plans and schedule, knowing which modules are more critical in terms of dependencies.

## · Testers who use the modules as their unit of work to create test cases and perform the tests.

## · The configuration manager who is in charge of maintaining current and past versions of the units in consistent and functional package assemblies, being able to produce a running version of the system.

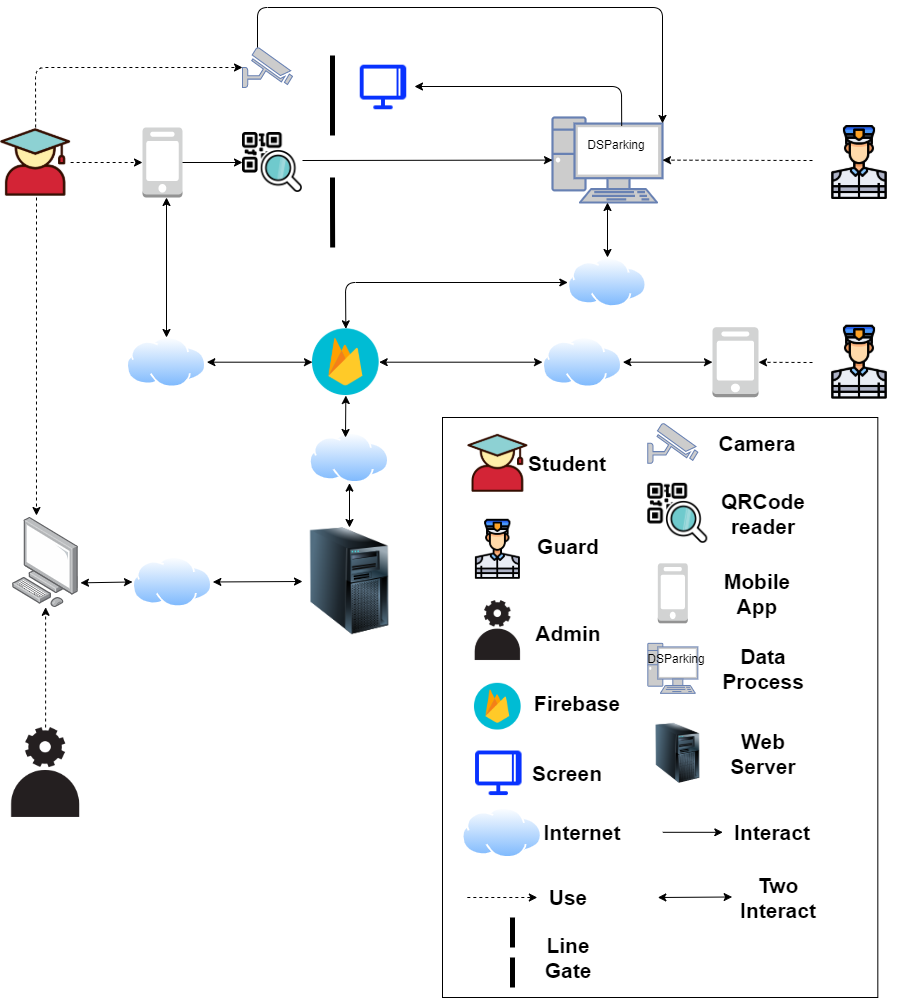
## · Developers, who are required to implement the elements.

## · Maintainers, who are tasked with modifying the software elements

## 

## Allocation view

The allocation view models the run-time architecture of a system. It shows the configuration of the hardware elements when the system is deployed.



***Figure 5: Allocation view***

**Prose**

|  |  |
| --- | --- |
| **Element** | **Responsibilities** |
| Student, Guard, Admin | The person who wants to use our system |
| Camera | Take photo of license plate |
| QRCode reader | Perform scan to get QR code information |
| Screen | Screen is used to display information when scanning a QR code to the user |
| Data process | the data process processes the QR code and the image of the license plate allowing the user to enter or exit |
| Web service | Web service is where the user logs in to the system to view information |
| Mobile app | Mobile app is also where the user logs in to the system to view information |
| Firebase | Receive data and interact with system in order to create notification in real time |