INSTITUTE OF TECHNOLOGY, TALLAGHT (ITT)

Use Cases and Logical Architecture

Mobile Car Parking System

X00118543

Rubielyn Nachor

Table of Contents

1 Online Parking System Use Cases

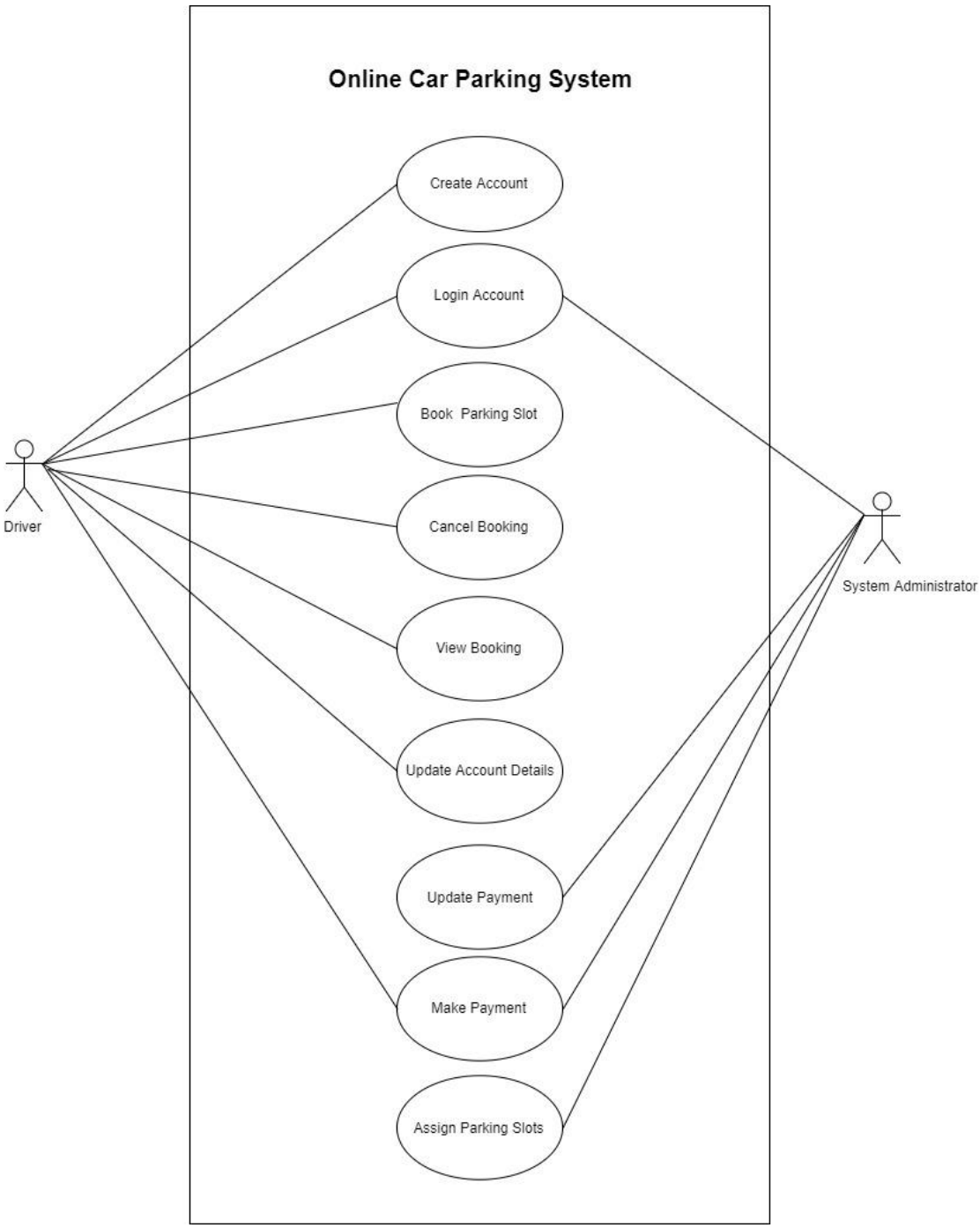

Figure 1: Use Case diagram for online parking system

1.1 Use Case 1

Title (goal)	Create Account
Primary Actor	Customer
Story	A new customer wants to create an account to online parking system.

1.2 Use Case 2

Title (goal)	Login Account
Primary Actors	Driver, System Administrator
Story 1	When a new user has finished creating an online Car Parking account. The user can login to the system with a valid username and password.
Story 2	The System Administrator wants to login to the system to perform configuration.

1.3 Use Case 3

Title (goal)	Book Parking Slot
Primary Actor	Driver
Story	A registered driver wants to book a parking slot to the system with his desired location, start time, and end time.

1.4 Use Case 4

Title (goal)	Cancel Booking
Primary Actor	Driver
Story	The driver changed his mind and wants to cancel his booked slot.

1.5 Use Case 5

Title (goal)	View Booking

Primary Actor	Driver
Story	The driver wants to view his reference number, start time, end time, and duration of is booked slot.

1.6 Use Case 6

Title (goal)	Update Account Details
Primary Actor	Driver
Story	The driver wants to update his name/phone number/email address/Valid bookings

1.7 Use Case 7

Title (goal)	Update Payment
Primary Actor	System Administrator
Story	The Administrator updates the price of booking a car parking slot in the system.

1.8 Use Case 8

Title (goal)	Make Payment
Primary Actor	Driver
Secondary Actor	System Administrator
Story 1	The driver makes payment to his booked slot by entering his valid card details; card number, expiration date, card holder name
Story 2	The System Administrator accepts the payment and issue a ticket receipt to the driver.

1.9 Use Case 9

Title (goal)	Assign Parking Slots
Primary Actor	System Administrator

Story	The system administrator wants to assign new parking slots to the system.

2 Prototype Schedule, Winter Semester 2017

Iteration #1, Complete: 25/10/2017	**Use Case 1**: Create Account **Use Case 2**: Login Account **Use Case 9**: Assign Parking Slots
Iteration #2, Complete: 15/11/2017	**Use Case 9**: Assign Parking Slots **Use Case 3**: Book Parking Slot **Use Case 5**: View Booking **Use Case 4**: Cancel Booking
Iteration #3 Complete: 13/12/2017	**Use Case 6**: Update Account Details **Use Case 8**: Make Payment **Use Case 7**: Update Payment

3 Parking system mobile Application Logical Architecture

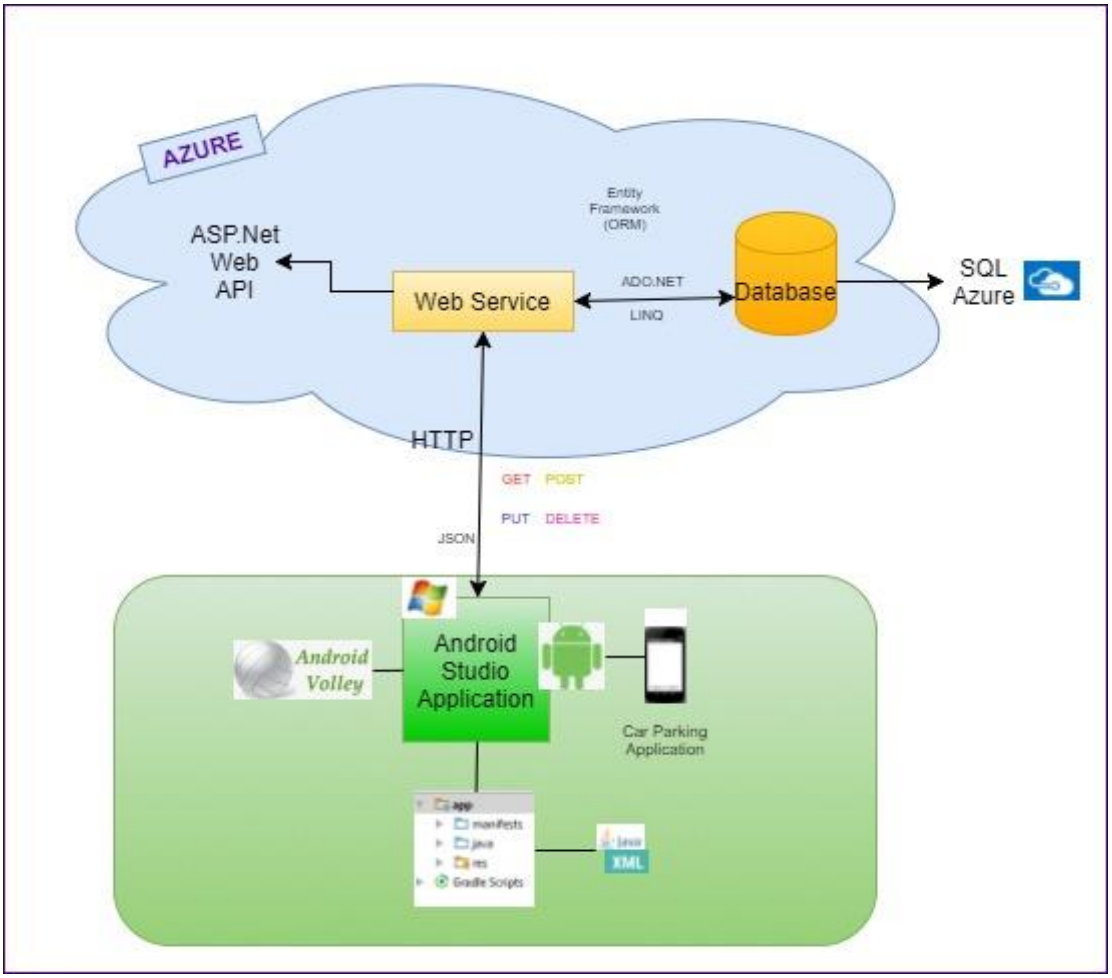

Figure 2: Online Parking System logical Architecture

3.1 Azure

3.1.1 ASP.NET Web API

The ASP.NET Web API is a fully supported and extendible framework for building HTTP based endpoints. ASP.NET Web API supports mobile application that needs services layer. The Web API connects with the database through Entity Framework and a Call from Android Studio.

3.1.2 Entity Framework (EF)

The Entity Framework in Azure is an object-relational mapper (ORM) for the .NET Framework. EF maps objects to objects in Relational Object Database, and serves as a bridge between data from the database and data from the code. (*Microsoft, 2016)*

The advantages of using EF are; it reduces impedance mismatch between classes and Relational Database and developers don't have to worry about foreign key constraints. (*Microsoft, 2016)*

3.1.3 EF Work Flow

According to Rowan Miller the Entity Framework in Azure has two ways of creating a database; the first one is by creating a new database and the second is using an existing database, both choices have 2 different method such as; Designer Centric and Code Centric. The Designer Centric method behaves like a Relational diagram that allows classes to be auto generated from the model. However, this is not a preferred approach as it causes conflict with having too many tables. My preference for my project is the Code Centric method as it defines classes and special mapping in code and it also applies migration changes to database. *(Rowan Miller, 2015)*

3.1.4 ADO.NET Code First

In this project I will be using ADO.NET which is a data access technology from the Microsoft .NET Framework that allows relational and non-relational systems communicates to each other. I will specifically use ADO.NET Code first which will allow me to map to an existing database from an SQL Server.

The following are the steps to connect to an existing database:

1. Create Project
2. Install EntityFramework package. After installation I can add "using System.Data.Entity above my Classes.

   - Project -> Manage NuGet Packages->EntityFramework->Install

3. Adding LINQ query will allow me to retrieve all the data from the database.
4. To connect to the database

   - **Right Click Data Connections > Add Connection > Microsoft SQL Server**

5. Connect to LocalDb (**(localdb)\v11.0**)
6. Enter a Database.

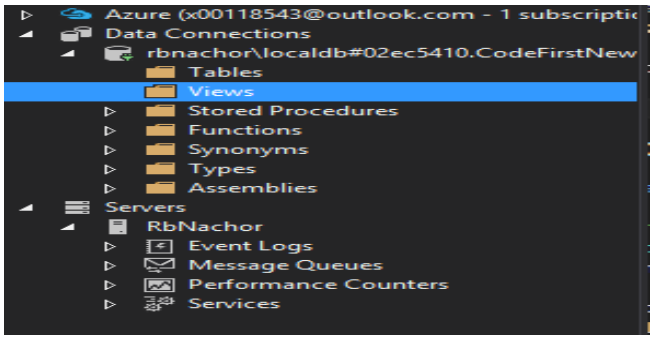

Figure 3: Local DB

Enable Code First Migrations for the project

7. **Tools -> Library Package Manager -> Package Manager Console**
8. Enter Enable-Migrations in the command in Package Manager Console.

   -a New Migration folder has been added to the project. *(Microsoft, 2016)*

3.2 ANDROID STUDIO

The IDE that I will be using for this project is the Android Studio Application which is based on IntelliJ IDEA. Android Studio contains a HTTP library called Volley. The volley library will be used to populate a User Interface (UI) for my .Net Framework implementations and display the data. Volley integrates easily with HTTP protocol and comes out of the box with support for raw strings, images, and JSON. The advantage of using volley is that it makes networking easier for Android applications, this works by calling the HTTP GET, POST,PUT methods in the API. *(Android, 2017)*

Android Studio provides an Android Emulator which will allow me to display the User Interface I have implemented from .Net Framework and test my app on an Android mobile phone device. I will be using the latest Android Operating system which is the Android Oreo 8.0 in Pixel_API_26.

3.2.1 Main files in Android Studio

The Android studio has two main files such “app” and “Gradle Scripts”.

The “app” file is composed of 3 files; manifest, java, and res.

app > manifests > AndroidManifest.xml

The manifest file describes the fundamental characteristics of the app and defines each of its components.

app > java > com.example.myfirstapp >MainActivity.java

This file is the main activity as it contains the entry point of the app “main”. When running the app, the system launches an instance of this Activity and loads its layout.

app > res > layout > activity_main.xml

This XML file defines the layout for the activity’s User Interface (UI). It contains a TextView element that contains a string value.

Gradle Scripts > build.gradle

This file contains two files with this name: one for the project and one for the app. Android Studio Gradle is an advanced build toolkit, to automate and manage the build process and allows defining flexible custom build configuration. Each build configuration can define its own set of code and resources. Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific for building and testing applications. *(Android, 2017)*

4 References

Microsoft Developer Network, 16 October 2013. *Entity Framework* [online], Retrieved from: https://msdn.microsoft.com/en-us/library/gg696172(v=vs.103).aspx [1st October 2017]

Rowan, M. 08 July 2015. *Introduction to Entity Framework*. Microsoft Channel 9 [online] Retrieved from: https://channel9.msdn.com/Blogs/Seth-Juarez/An-Introduction-to-Entity-Framework-with-Rowan-Miller [1st Oct 2017]

Microsoft Developer Network, 23 Oct 2016. *Entity Framework Code First to a New Database* [online], Retrieved from: https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx [6th Oct 2017]

Android Developer, 24 Jan 2017. *Transmitting Network Data Using Volley* [online], Retrieved from: https://developer.android.com/training/volley/index.html [1st Oct 2017.]

Android Developer, 3 March 2017. Create an Android Project [online], Retrieved from: https://developer.android.com/training/basics/firstapp/creating-project.html [1st Oct 2017]