Car Sharing on Demand

Ricardo Duarte

*BSc Hons in Contemporary Software Development* 2024/2025

# 

Computing Department, ATU Donegal, Port Road, Letterkenny, Co. Donegal, Ireland.

Car Sharing on Demand:

A car sharing platform for users and providers

Author: Ricardo Duarte

Supervised by: Pauric Dawson

June 2024

# Declaration

I hereby certify that the material, which l now submit for assessment on the programmes of study leading to the award of Bachelor of Science in Applied Computing, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution. I understand that it is my responsibility to ensure that I have adhered to ATU’s rules and regulations.

I hereby certify that the material on which I have relied on for the purpose of my assessment is not deemed as personal data under the GDPR Regulations. Personal data is any data from living people that can be identified. Any personal data used for the purpose of my assessment has been psudonymised and the data set and identifiers are not held by ATU. Alternatively, personal data has been anonymised in line with the Data Protection Commissioners Guidelines on Anonymisation.

I consent that my work will be held for the purposes of education assistance to future students and will be shared on the ATU Computing website (http://www.atucomputingdonegal.com/) and Research THEA website (https://research.thea.ie/). I understand that documents once uploaded onto the website can be viewed throughout the world and not just in Ireland. Consent can be withdrawn for the publishing of material online by emailing Thomas Dowling; Head of Department at jade.lyons@atu.ie to remove items from the ATU Computing website and by email emailing Denise McCaul; Systems Librarian at denise.mccaul@atu.ie to remove items from the Research THEA website. Material will continue to appear in printed formats once published and as websites are public medium, ATU cannot guarantee that the material has not been saved or downloaded.

Signature of Candidate: Date:

# Acknowledgements

I would like to thank…

# Abstract

Written here is no more than 250 words of summary of the problem, hypothesis and main conclusions. The abstract will entice people to read the rest of the document.

# Acronyms

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Page |
| JWT  V2X  V2P  JVM | JSON Web Token  Vehicle 2 Everything  Vehicle 2 Person  Java Virtual Machine | 1  1  3  3 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Table of Contents

[Declaration 4](#_Toc195223440)

[Acknowledgements 5](#_Toc195223441)

[Abstract 6](#_Toc195223442)

[Acronyms 7](#_Toc195223443)

[Table of Contents 8](#_Toc195223444)

[Table of Figures 10](#_Toc195223445)

[Table of Tables 11](#_Toc195223446)

[Table of Code Listings 11](#_Toc195223447)

[1. Introduction 1](#_Toc195223448)

[1.1. Purpose 1](#_Toc195223449)

[1.2. Background 2](#_Toc195223450)

[1.2.1. V2X 2](#_Toc195223451)

[1.2.2. JWT 3](#_Toc195223452)

[1.2.3. Kotlin Multiplatform 3](#_Toc195223453)

[1.3. Aims & Objectives 4](#_Toc195223454)

[1.4. Report Outline 5](#_Toc195223455)

[2. Design 5](#_Toc195223456)

[2.1. Requirements 5](#_Toc195223457)

[2.1.1. User Application 5](#_Toc195223458)

[2.1.2. Server 6](#_Toc195223459)

[2.1.3. Car Application 7](#_Toc195223460)

[2.2. High Level Design 8](#_Toc195223461)

[2.2.1. Use Case Diagram 8](#_Toc195223462)

[2.2.2. Sequence diagrams 9](#_Toc195223463)

[2.3. Low Level Design 12](#_Toc195223464)

[2.3.1. Class diagrams 12](#_Toc195223465)

[2.4. Agile Methodology 12](#_Toc195223466)

[2.4.1. Kanban Board 13](#_Toc195223467)

[2.4.2. Epics 14](#_Toc195223468)

[2.4.3. Stories 14](#_Toc195223469)

[Appendices 16](#_Toc195223470)

[Appendix A: References i](#_Toc195223471)

[Appendix B: Code Listing iii](#_Toc195223472)

# Table of Figures

[Figure 1 System use case diagram 8](#_Toc195222441)

[Figure 2 Car Server sequence diagram 9](#_Toc195222442)

[Figure 3 User server sequence diagram 10](#_Toc195222443)

[Figure 4 User Car sequence diagram 11](#_Toc195222444)

[Figure 5 Project’s kanban board 13](#_Toc195222445)

[Figure 6 Car Server Jira epic 14](#_Toc195222446)

[Figure 7 Jira story for the creation of the server skeleton code 14](#_Toc195222447)

# Table of Tables

**No table of figures entries found.**

# Table of Code Listings

**No table of figures entries found.**

# Introduction

Car sharing today presents today, according to (Turoń, 2023) “one of many solutions to help solve mobility in dense urban areas”, sine it aims to provide an easy and affordable solution for short-term trips and allow an individual to easily access a car sharing service over a smartphone in real time. The author of the article also highlights that in 2023 car sharing solutions existed in “over 59 countries offered by hundreds of different providers”, which could highlight the scalability of such solutions in today’s world and evidence how much this numbers could grow soon (JetBrains, 2024) given the trend.

In 2024 in Germany car sharing services “covered more than 1200 cities with over 300 providers”, according to (Wandering, 2024), which highlights the extension of the car sharing services on of the largest EU countries. He also highlighted that car sharing services can be “a valid option for people who don’t own a car or drive sporadically” and that base free “free floating” providers are a great alternative to “more typical car rentals when it comes to short trips” (Wandering, 2024).

Giving that, the purpose of this project is to introduce a generic car-sharing platform, that allows to provide a general, safe and easy to use solution into the car sharing model landscape.

## Purpose

The purpose of this project is to provide a generic unified solution for car sharing with a centralized backend that aims to establish secure sessions between user and car relying on a non-standard user session authentication method instead providing a secure token (JWT) for each session. This solution fits within a V2X design where the vehicle will be able to communicate with both the server to self-register and provide a status update at a constant rate at the same time as it can communicate with the user once a session as been established to execute diverse set of operations part of the car sharing flow (such as car lock and unlock for example).

## Background

The world today is ever more interconnected, especially in urban canters, which provides the need for more ways of transportation and shared mobility are not excluded, for example shared scooters and bikes and cars are among the most convenient forms of transportation, according to (Turoń, 2023), however in the same article the author explains that in the adopted markets the implementation of car sharing services “one can observe the number of closures, takeovers of companies, or implementation of services on only a pilot basis” (Turoń, 2023, p.1185), which indicates that either the available services are monopolized or inexistent on a commercial way, and according to the author, the arguments behind it are: “the system had an ill-matched business model, funds were not properly managed, the project turned out to be economically un-proﬁtable, information found about the ﬂeet of vehicles was improperly matched to the needs of users, customer expectations were not met, or that the scope of operation of the services was incorrectly deﬁned” (Turoń, 2023, p.1186).

Therefore, the purpose of this solution is the implementation of a cross-platform solution that aims to implement a scalable and generic car sharing platform, which allows for adjustment based on the user needs and that provides a seamless and secure of connecting the user to the car, the solution aims to address both the needs of the user and provider by allowing more flexibility in the offer. This project is going to be implemented in Kotlin, with a design based on a V2X which favours the usage of JWT for password less authentication between user and car.

### V2X

The constant evolution of information networks, such as 5G, pushes for to design vehicle wireless technologies capable of communicating with everything from the cloud, people or even other vehicles, and, according to (Hasan *et al*., 2020), V2X as a technology aims to “improve trafﬁc and resource efﬁciency, incidents and road pollution”. However, this technology is not free from challenges and the author mentions that the biggest concerns are regarding “security and privacy”. In this project the idea is to use V2X to connect vehicles to users both through an off-board entity (cloud server) and directly V2P using JWT as a technology for a safe and password less authentication between user and car.

### JWT

According to (Jones, *et al.*, 2015), the standard defines a JSON Web Token as a “URL-safe means of representing claims to be transferred between two parties”, a JWT is composed by an header containing the signing algorithm and type, a payload which represents the data object (in JSON format) being transmitted between entities, and the signature that validates the token itself generated via HMAC-SHA256 algorithm using a private key. According to (Rana, *et al.*, 2023) which also states that “JWT with the HMAC-SHA256 algorithm provides a high level of security in data exchange”, which in this project is used to mitigate the security challenges from the V2X nature of the project. The JWT can also be used for authorization and authentication which will allow for a user to initialize a safe and unique session desired vehicle.

### Kotlin Multiplatform

Kotlin multiplatform is a Kotlin capability that aims for “reducing time spent writing and maintaining the same code for different platforms”, according to (JetBrains, 2024), this allows for the same code to be shared for both web, mobile and desktop while attempting to preserve most of the benefits of native code. Kotlin provides multiple frameworks that will assist this project such as Compose, which is a cross-platform framework that allows developers to easily develop cross platform applications with shared code for both the models (in Kotlin) and UI (declarative). Kotlin also provides KTOR which is a framework for the web, which allows to both write client and server code, KTOR allows to compile for the JVM and natively which provides a great tear of flexibility for the solution being developed.

## Aims & Objectives

The main goal of this project is to develop a car sharing system that focus on simplicity, safety and scalability as well as generalization which aims to provider users and providers flexibility in offer. The main aims for this project are:

* To create a user application that can request a car service to a service provider by:
  + By using Kotlin Multiplatform to create a cross-platform application.
  + By allowing the user to select available cars on demand.
  + By requesting a safe JWT token to a server to create a session with a car and request commands.
* To create a server
  + By using Kotlin KTOR to implement the server-side code and JWT authentication flow.
  + By requesting the car for updated status.
* To create a test car application
  + By using Kotlin native and KTOR for the https client code.
  + By self-registering to a server as a provider.
  + By establishing a connection with a user once a valid request with a valid token is done.
  + By providing the server with up-to-date status.

## Report Outline

Provide a summary of what each of the following chapters will entail

# Design

In order to produce the desired solution introduced in the previous chapter, first a design must be created in order to understand how the project can be implemented, the design section is composed gathering of requirements (both functional and non-functional), the creation of high-level design (flow diagrams), low level design (class diagrams and sequence diagrams) and the necessary agile project management for the implementation phase by creating an agile board with agile artifacts (such as epics and user stories).

## Requirements

When analysing the aims and goals of project is possible to obtain a high-level view of how the system should be but it lacks details necessary for its implementation, therefore this next topic is focused on gathering and categorizing the functional and non-functional requirements which will be the fundamental guidelines for the overall design and implementation.

### User Application

One of the parts of the desired system is a cross-platform application that can ran in multiple platforms, where a user can fetch for available cars, get the car details such as availability, amount of fuel and location for example, and request a session for a specific car, lock and unlock the car and end the current session.

#### Functional Requirements

* Be cross-platform application
* The application should be able to fetch for cars, from the server and list them to the user.
* The listed cars must provide the user the current car status (such as availability, charge or fuel, and exact location) in a list (or map) and cost.
* The application should allow the user only to select a car and it must be available.
* The application must be able to request the server for a secure token to start a session with a selected car.
* The application must allow the user to unlock, lock the car and end a session once a session has been established.

#### Non-Functional Requirements

* The application must be developed using Kotlin Multiplatform (with compose)
* The application must use KTOR framework for the https client-side code.
* The application must use JSON as the data encoding to communicate with both the server and the car
* The application must be able to encode and decode JWT

### Server

The second and most central component of the system is a server application that can fetch data from the car app for the car details, and feed the user application, accordingly, allow to get request from a user to authenticate with a specific car, generate a signed JWT and provide it to the user so it can authenticate with the car.

#### Functional Requirements

* Have a server capable of accepting car and user registration.
* The server must be capable of providing a public key to a car on registration to validate incoming requests from users.
* The server must be capable of providing the user with a secure token on session request.
* The server must be able to gather data from registered cars data on a regular basis.
* The server must be capable of providing the fetched car data at real time to the user applications registered.

#### Non-Functional Requirements

* The server must be implemented in Kotlin.
* The server must use KTOR for the https implementation and token capability.
* The server must use JWT as the technology for secure token.
* The server must respond to car data updates every 30 seconds.

### Car Application

The last part of the system is a car application that constantly sends status data back to the server, it can interact with the user app to initialize a session, receive commands for lock and unlock and end the session on user demand.

#### Functional Requirements

* Have an application that can communicate with the server.
* The application should be capable to register by itself as a provider to the server.
* The application should be capable of receiving a public key from the server to validate future requests from a user.
* The application should be able to provide data about its status to the server.
* The application should be capable of establishing a session with a user application.
* The application should be capable of receiving commands from the user once a session has been established (lock and unlock).
* The application must provide a report back to the client once the session has been terminated.

#### Non-Functional Requirements

* The application must be developed using Kotlin native.
* The application must use KTOR as the framework for the https client code.
* The application must be capable of decoding JWT from the client and validate its signature using the public key.
* The application must send a status to the server every 30 seconds.

## High Level Design

The high-level design dictates a broader image of how the system looks like, which allows for the existence of an overview of what is desired to be implemented which match the defined functional requirements, this was designed using a use case diagram and sequence diagrams.

### Use Case Diagram

The use case diagram provides a broad description of the interactions between the different actors and the designed system. In this case the project main components are represented by three different actors who interact with each other, and each interaction is a use case, this is better highlighted in the following image:

A diagram of a company

AI-generated content may be incorrect.

Figure 1 System use case diagram

### Sequence diagrams

The use case diagram itself only represents the interaction between the actors of the system, which alone do not provide enough information of how the system behaves. In this case a sequence diagram is required to provide information how the system interactions happen on a time sequence, which can provide a sense of linearity and better understanding of the system requirements and how the components of the system interconnect with each other in more detail. In this project three main sequence diagrams where extracted, which can be observed in better detail in the figures that follow:

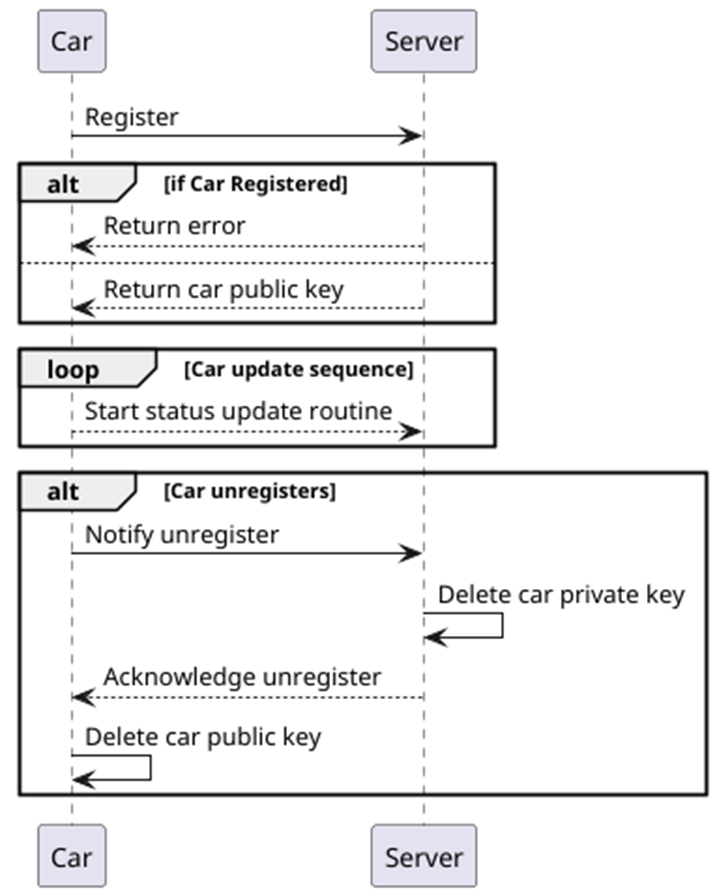


Figure 2 Car Server sequence diagram

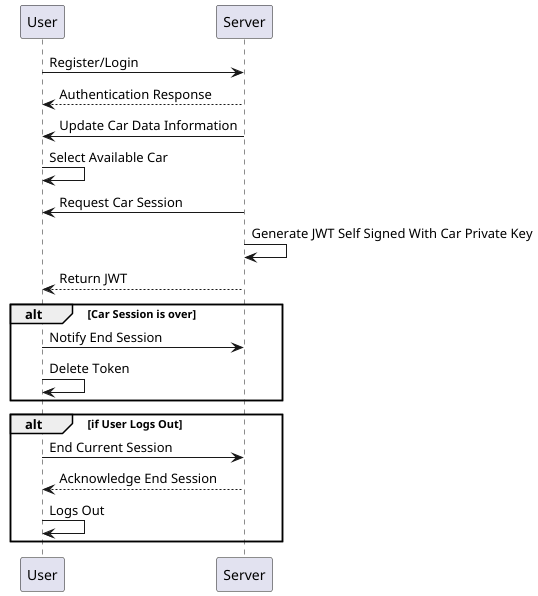


Figure 3 User server sequence diagram

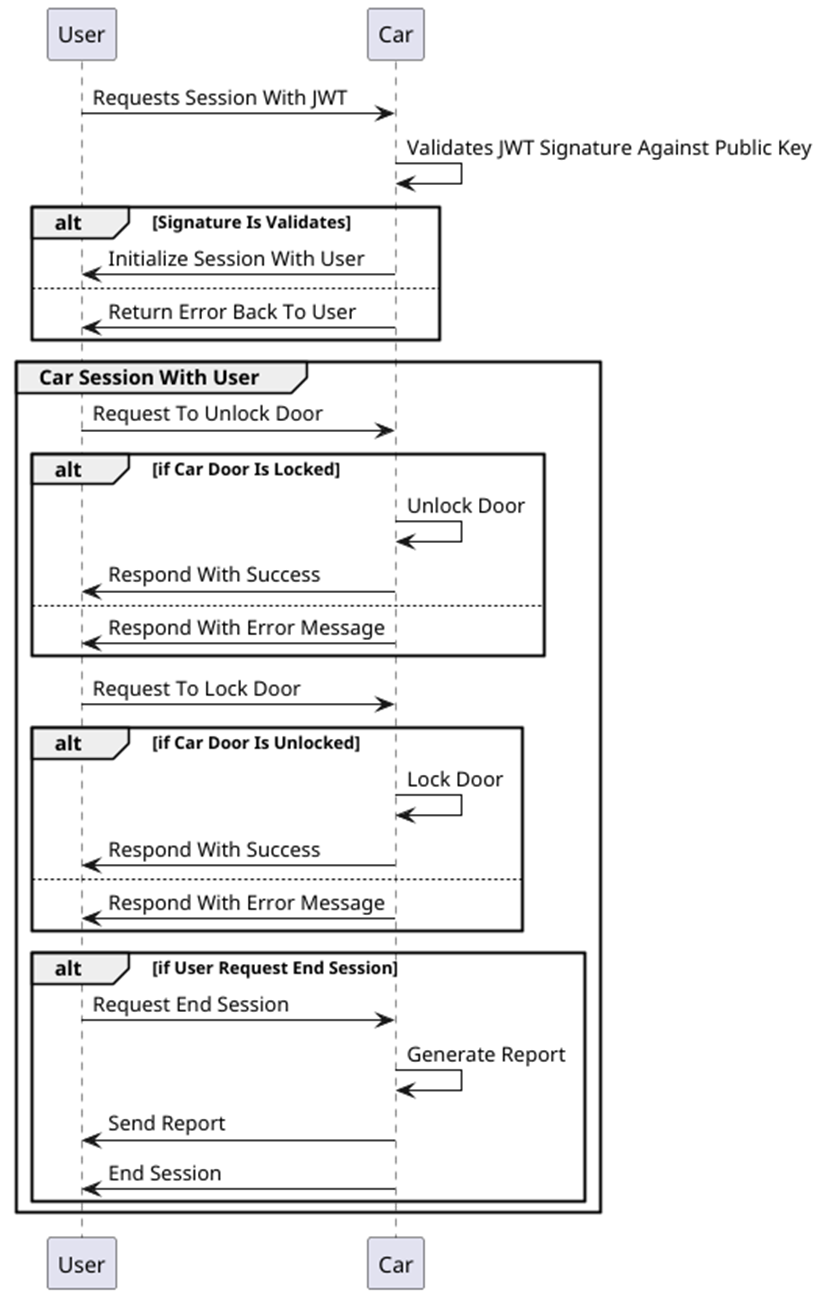


Figure 4 User Car sequence diagram

## Low Level Design

The low level design is a fundamental component of the design chapter which aims to give a more in depth understanding of how the implementation should be, and focus on higher technical details of the development, an example is the creating of class diagrams which is a diagram that represents the components in the system in an object orient manner (units or components represented in classes and instantiated as objects) and how they the relation between them (inheritance, composition). The system class diagrams are:

### Class diagrams

## Agile Methodology

Given the nature of the project being developed and its complexity, some form of project management is required so the development can meet the needs of requirements as much as possible. As far as methodology is concerned there are major methodologies in software development today, such as waterfall which according to (Falade rhoda, *et al.*, 2025) focus on “Detailed upfront planning, fixed scope, phased progression” but “Waterfall resists change, risking irrelevance” and “Scope issues surface during testing, inflating costs if misaligned with needs”, on the other end agile which key feature are “Iterative planning, continuous stakeholder feedback, adaptive scope refinement” while it risks “Over-Flexibility: Without discipline, Agile can invite endless changes”. Given the pros and cons of both methodologies the one selected for this project was the agile methodology since it provides higher flexibility deliverable increments and nature of the project helps mitigate ant risk of endless change given its straightforward nature. Within the agile methodology there exist several frameworks such as Kanban, SCRUM, SAFe, etc.., for this specific project Kanban was selected given the size of the team and nature of the segments of work (linear workload).

### Kanban Board

Kanban as many other frameworks, presents its artefacts in a board called Kanban board, in which columns separate the flow of work, in this project the flows defined were, *Ready* which represents any story or task that contains a complete description and is ready to be started, *In Progress* which holds any story or task in active development and finally *Done* which represents every task which have been merged to the main branch. The project’s board can be visualized in the figure below:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5 Project’s kanban board

### Epics

An epic is an agile artifact that represents a large feature, usually an epic is written in a very high-level perspective mostly focused on a larger scope of the feature, highlighting the feature’s functional requirements. This project contains three epics Server, User App and Car App. The image below is of one of the project features:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6 Car Server Jira epic

### Stories

Stories in agile represent a smaller part of a feature, usually carries a more granulated description written in the perspective of the developer or user and aims to implement one or more functional requirements. The figure bellow represents one of many tasks of this project:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7 Jira story for the creation of the server skeleton code

# Appendices

# Appendix A: References

Falade rhoda, A., Philip, T. & Ava, A., 2025. Agile vs. Waterfall: Which Methodology Best Reduces Scope Creep?. p. 5.

JetBrains, 2024. *Kotlin Help.* [Online]   
Available at: https://kotlinlang.org/docs/multiplatform-intro.html  
[Accessed 2025].

Jones, M. B., Bradley, J. & Sakimura, N., 2015. *JSON Web Token (JWT).* [Online]   
Available at: https://www.rfc-editor.org/rfc/rfc7519.html  
[Accessed 2025].

Monowar, H., Sibin, M., Takayuki, S. & Hongsheng, L., 2020. Securing Vehicle-to-Everything (V2X) Communication Platforms. *IEEE Transactions on Intelligent Vehicles,* 5(4), p. 22.

Rana, M., Pandey, A., Mishra, A. & Kandu, V., 2023. Enhancing Data Security: A Comprehensive Study on the Efficacy of. *International Journal on Recent and Innovation Trends in Computinng and Communication,* 11(9), pp. 4409-4416.

Turoń, K., 2023. Factors Affecting Car-Sharing Services. *Smart Cities,* p. 17.

Wandering, S., 2024. *Carsharing 2024: Providers Compared.* [Online]   
Available at: https://www.navit.com/resources/carsharing-2024-providers-compared  
[Accessed 2025].

# Appendix B: Code Listing