TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Mihhail Kohhantšuk 193974IAIB

Technical Solution and a Prototype for Linking License Plate with a Mobile Device

Bachelor's thesis

Supervisor: Innar Liiv
PhD
Janno Stern
BSc

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mihhail Kohhantšuk 193974IAIB

Numbrimärgi ja mobiilseadme sidumise tehniline lahendus ja prototüüp

Bakalaureusetöö

Juhendaja: Innar Liiv
Doktorikraad
Janno Stern
Bakalaurusekraad

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Mihhail Kohhantšuk

02.05.2022

Abstract

The aim of this thesis is to build a solution that would expand the potential of license plate recognition (LPR) by linking the car's license plate to the owner's mobile device.

Since car related services are often directly related to the car or the driver, the license plate is sufficient to identify the person behind the wheel and allow access to the service offered with the help of the driver's mobile device.

This thesis presents a technical solution with a prototype that can be offered as a service. Prototype consists of three parts:

1. License plate recognition application.
2. Server side running application to manage users and license plates with the goal of sending out push notifications to users as license plates are being detected.
3. Mobile application for user registration that allows the user to add license plates under his name as well as receive push notifications and confirm the receipt to start the use of a related service.

This thesis is written in English language and is 32 pages long, including 6 chapters and 13 figures.

Annotatsioon

Numbrimärgi ja mobiilseadme sidumise tehniline lahendus ja prototüüp

Lõputöö eesmärgiks on luua lahendus, mis laiendaks numbrimärgi tuvastuse võimalusi ühendades numbrimärgi omaniku mobiilse seadmega.

Kuna autodega seotud teenused on enamasti seotud kas auto või juhiga, siis piisaks numbrimärgist, et identifitseerida juht ja võimaldada seotud teenuse alustamist mobiilse seadme kaudu.

Lõputöö jooksul töödeldakse valmis lahenduse prototüüp, mida saaks pakkuda teenusena. Prototüüp koosneb kolmest osast:

1. Numbrimärgi tuvastuse tarkvara.
2. Serveri poolne rakendus, mis haldaks kasutajaid ja nendega seotud numbrimärke, saates „push notification“-eid kasutajatele, kelle numbrimärk oli tuvastatud.
3. Mobiilne rakendus kasutajale konto loomiseks ja numbrimärke salvestamiseks. „Push notification“-ite vastu võtmiseks ja seotud teenuse kasutamise alustamise kinnitamiseks.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 32 leheküljel, 6 peatükki ja 13 joonist.

List of abbreviations and terms

LPR	License Plate Recognition
SaaS	Software as a Service
URL	Uniform Resource Locator, web address
MVP	Minimum Viable Product
DNN	Deep Neural Network
GPU	Graphics Processing Unit
UI	User Interface design
UX	User Experience design
API	Application Programming Interface
SDK	Software Development Kit
AI	Artificial Intelligence
FCM	Firebase Cloud Messaging
CPU	Central Processing Unit
USB	Universal Serial Bus
CSI	Camera Serial Interface
RTSP	Real Time Streaming Protocol
JWT	JSON Web Token
UUID	User Unique Identifier
Regex	Regular Expression
OOP	Object Oriented Programming
FP	Functional Programming
FRP	Functional Reactive Programming
ORM	Object Relational Mapping
REST	Representational state transfer
BLE	Bluetooth Low Energy

Table of contents

List of figures

1 Introduction

1.1 Background and problem

License plate recognition (LPR) is used widely across the world nowadays. Author feels that there is still a lot more untapped potential in LPR for regular users.

So far, the majority of LPR related advancements have been aimed at helping government institutions with issues such as traffic management and potential criminal activity detection. However, vast majority of regular users are so far limited to only using these advancements in parking related services. As the technology is evolving rapidly, the use of mobile devices as a payment method is increasing in popularity. Car related services are often directly related to the driver behind the wheel, and therefore the license plate should be enough to identify the driver and allow the use of a related service.

Author believes that the possibility of identifying the user through license plate recognition has not been explored so far mainly due to technology limitations. Regardless of which car-involving service users wanted to access, they were always forced to leave the vehicle in order to process payment. The general public is growing accustomed to the idea of paying for the service using the mobile device. This in turn allows us to completely reimagine how transactions are being made. Today people should be able to pay for the service without leaving the vehicle using only their mobile device.

This change also greatly reduces waste by relying on paperless receipts. Previously drivers used to pick up the ticket at the entrance and the same ticket was used for payment on their way out. Author believes that this use of tickets is not efficient, as every single ticket completely loses its value as soon as the service is paid for. The stress on the driver is also greatly reduced as there is no need to worry about making sure that the ticket is stored safely.

1.2 Goal

To expand the potential of LPR cameras for the general public and to eliminate the need of leaving the vehicle in order to pay for the service, as well as omit the usage of parking tickets due to them being completely useless once the parking is paid for, author's aim is to create a solution that would recognize a license plate, identify a driver's mobile device related to that vehicle and send a push notification to the mobile device to start the use of related service. Mobile application will be developed to allow user registration, so that users could add respective license plates under their name and confirm the start of the related service use through the application.

Another issue author is trying to tackle is the maintenance of LPR cameras and software. It takes a lot of resources for a company to build an in house LPR solution, especially if producing LPR software is not a company's first goal. This also addresses the issue of the need for human resource required to keep the system running. With that in mind it makes sense to produce a solution which can be offered as a service, as it is much more rational and cost-efficient for a company.

The main goals of this thesis are:

- To create a technical solution with a scalable architecture:
    - Create a direct link between the mobile device and the license plate;
    - develop a mobile application that would allow users to register, link the license plates to their accounts and start the use of services with the help of push notifications;
    - run LPR software on Jetson Nano, a powerful small form factor computer, further explored in Chapter 4.1.2;
    - develop a server-side application for handling requests coming from the LPR software, as well as fully support the mobile application.
- To create a prototype that with further development can be offered as a service either with or without support for the mobile application.

2 Analysis

As a part of an analysis the required functionality is going to be explained, existing solutions are going to be researched and work is going to be split into parts.

2.1 Required functionality

Aim of the thesis is to develop a solution which will be offered as a service, which means it can be consumed in two ways and have different goals that require different levels of functionality.

Firstly, it can be offered purely as LPR Software as a Service (SaaS). This means that the only goal of the application will be to read the license plate and send respective license plate to the correct address (URL). The benefit of this approach to the potential client is that it eliminates the need of maintaining the LPR software and purely focus on the implementation of the user interaction whenever their license plate has been read. This approach is most often offered by providers of similar solutions. The downside of this approach is that the company will have to implement users' license plate handling within their system which can be time-consuming and possibly unnecessary and cost-inefficient depending on the service being provided.

Secondly, it can be offered alongside the mobile application developed in the scope of this thesis. This approach will eliminate the need for the consumer company to implement a complex users' license plates handling solution. Implementation of this solution will produce a following user flow:

- Prior to using this service users will need to have downloaded a mobile application and added a respective license plate under their name.
- LPR software detects an approaching vehicle, reads its license plate and sends a respective request to our server-side application.
- The server will receive a request and try to find a respective user from the database.

- If the user has been found it will take the device token and send a respective push notification to the user's mobile device.

- Upon receiving a push notification, the user can open it and will be taken to the mobile application, where they will be able to confirm receiving a push notification and be taken to the consumer company's service.

Author believes that user will prefer having a single mobile application that will allow starting the use of different services rather than having to manually search for the correct application every time. It also has the potential to attract new service providers, that haven't had the means to develop a full solution which would allow the use of LPR within their sphere.

The downside of this approach is that users must be aware that an application needs to be downloaded which is not directly related to the company offering the service they wish to use.

Implementing the second approach in turn also creates all the necessary components for the first one to work, so it will be implemented in the scope of this thesis. The implementation will consist of following components:

- LPR software running on Jetson Nano. It is sole purpose is to read the license plate and send respective data to server.

- Database to handle users, their respective license plates and push notifications, as well as information about installed LPR cameras and their service providers.

- Server-side application. It will fully support mobile application functionality such as user registration, user license plates and device token handling for push notifications, storage of information about the LPR cameras in use, handling of information about services and service providers, notifying users when their license plate has been read, as well as allowing to use the solution as pure LPR software, by sending data to a correct address instead of a push notification.

- Mobile application. The purpose of the mobile application is for the user to register and keep track of their saved license plates, as well as confirmation of receiving a push notification. Registration could have been omitted since only

device token is required to send the notification, but it allows the application to successfully handle scenarios where the device token has been updated. This also allows to store the necessary user data, such as email which can be used to inform user about possible issues or interruptions of the service.

2.2 Existing similar solutions

Plate recognizer[1] offers LPR software that can use either images or live video stream as input. It offers a paid subscription service but does allow free 2500 image lookups per month or 3 live camera streams for only 1 month.

Axis[2] offers LPR solution with the use of their purpose-built cameras with pre-installed Axis software.

OpenALPR[3] is offering opensource API as well as cameras made specifically for LPR. Different types of LPR solutions are offered some examples include:

- Gathering statistics as a security measure to track which vehicles passed the camera.
- Detect which customer arrived at drive-through to prioritize their order.
- Paying for car wash based on license plate.

All the services are hidden behind a paid subscription with only 14 days of free trial. This is the closest implementation that author has found which tries to widen the use cases of LPR.

Autlo[4] offers a parking application to use in Baltics and different solutions, which are all directly related to parking. All the solutions are paid, but prices are all set after a consultation with the CEO.

---

[1] https://platerecognizer.com/

[2] https://www.axis.com/solutions/license-plate-recognition

[3] https://www.openalpr.com/

[4] https://autlo.com/

Rollet[1] offers a completely seamless parking solution. Bundled with a mobile application to register the license plate and setup the payment method, it allows the user to enter and leave paid parking lots, all while the payment is automatically handled in the background. The cost of implementing this service is set only after getting in touch with Rollet directly.

After looking at existing solutions author has found that only Rollet tries to link user's license plate to the mobile device to identify them and allow the use of the service through their mobile device. Main downside being that it is a fresh start-up, so it is very localized for use in Hungary. All other solutions are either only LPR API or are targeted at installing the cameras to identify only the vehicle itself and leave all the license plate handling implementation to the consumer company.

2.3 Relevance

At the time of working on the thesis on 28.04.2022 author found a news article about Circle K implementation of a very similar solution to the one being produced in the scope of this thesis.

Solution developed by Circle K allows the user to pay for filling up at the gas station with the help of a mobile device. LPR software detects a license plate after which it tries to find the user and checks if the user has sufficient funds on their card. If the checks are successful, then the respective gas pump will unlock and allow the user to fuel the vehicle. After the user has finished filling up, they can just drive off and funds will be taken from the bank account automatically [1].

This backs up the idea that the expanding use cases of LPR with the help of a mobile device is an issue with growing relevance right now and should be explored further.

Solution produced in the thesis is different, as it would allow a single application to be used across different similar service providers as well as not being limited to just gas stations or parking.

[1] https://www.rollet.hu/

2.4 Defining minimum viable product

Initial first round of minimum viable product (MVP) was defined by the components that were necessary in order to complete full user flow:

Author decided on the approach of two rounds of MVP for multiple reasons:

- To better understand, how the user flow is supposed to work and how the involved steps are going to be implemented.
- To have a more foolproof solution. Due to the small timeframe making one MVP may lead to some unexpected issues being discovered right before the thesis submission deadline.
- This allowed more time to focus on the technical aspects that author had no previous experience with. These aspects include setting up Jetson Nano and running software on it as well as LPR implementation with the help of Nvidia DeepStream SDK. This in turn also allows to better polish issues in the areas that author is more familiar with like Flutter and NestJS.

Initial MVP was finished on 5th of April 2022. Its purpose was to allow the simplest implementation of the user flow to be complete.

As the result of the first round of MVP following components were implemented:

- LPR application running on Jetson Nano getting pre-recorded video files as an input and live output of all the license plates as they appear.
- Running server-side application capable of saving new users based on their device-token and saving relations between users and license plates as well as sending out push notifications when it is told that an LPR application has found a license plate.
- Python script reading live output of LPR application in order to tell backend to try and find a respective user as soon as a license plate has been identified.
- Flutter application working on both IOS and Android allowing users to save license plates under their name and delete existing ones as well as receiving push

notifications. Registration has been omitted at this step, as it was not necessary to retrieve device-token which is used to send push notifications.

- Firebase project with running Firebase Cloud Messaging service to allow the usage of push notifications.

2.5 Conclusions from the initial MVP

During and after implementation of the initial MVP several issues came to light that needed fixing or implementing in the second round of MVP:

- User registration is necessary, as Firebase Cloud Messaging uses device tokens to identify devices and sends push notifications based on that. The issue is that these tokens may be refreshed or updated at random intervals and need to be handled manually inside the applications, otherwise push notifications would never reach the mobile device anymore [2].

- Multiple car user problem highlighted in Chapter 4.3

- A confirmation screen view is needed inside mobile application to show info related to where the push notification came from and show info about potential service. It also allows for quicker and more fluid user interaction, as it eliminates the need to click on the push notification itself and can show the necessary info straight away.

- Data about which camera information comes from should also be saved in the database to differentiate between potential different cameras and to have the location of the camera stored which allows to find the closest car owner.

- Implementation of live stream support using Nvidia DeepStream SDK is more complex than initially expected due to author having no prior experience working with applications developed in C with the help of Gstreamer.

3 License Plate Recognition

3.1 How it works

Due to LPR being explored for many years different approaches have emerged to tackle this issue. However, there are still general steps that every single solution must fulfil [3].

1. Detecting the vehicle and subsequently it's license plate. This is done by looking for rectangular objects and filtering out those that have unfitting aspect ratios, for example being too wide. It also needs to account for possibility of the license plate being skewed, as the car could be approaching from the side and not face the camera directly.

2. Segmentation of characters is done in order to help split license plate letters into multiple areas of interest to make subsequent steps easier and more precise.

3. Optical character recognition is the process of translating the captured image into an alpha numeric text entry. Its result must comply with the required set regulations of the order that letters and numbers must be in. For example, an Estonian license plate can not contain only numbers or only letters, both must be present.

3.2 Existing use cases

With how accessible LPR has become, it is being used extensively and is essential in the smart cities. Smart city uses information and communication technology to improve operational efficiency, share information with the public and provide a better quality of government service and citizen welfare [4].

Some of the use cases of LPR are:

- Law enforcement – police forces often use LPR technologies to see if vehicle is registered and if it has any traffic violations, as well as tracking vehicle location in real time with the help of street cameras [5].

- Car parking management – use of LPR has extensive benefits in parking management. It helps keep track in real time, how many cars are occupying the parking lot at any point in time. Gathering essential data of how long drivers stay in the parking lot on average and seeing trends of which vehicles and when usually arrive, which could be crucial to some marketing campaigns. It also allows for an automated payment system, omitting the need of parking tickets and saving drivers the stress of having to make sure that their parking ticket is safe [5].

- Journey time analysis and traffic management – the ability to track vehicles throughout the city helps to understand, how congested some of the streets are, which is essential in city planning [5].

- Tollbooth records – manual tollbooth management on highways is still present in some parts of the world. LPR eliminates that need by allowing to pay automatically thus greatly reducing congestion and productivity [5].

- Paying at gas stations – vehicle is identified, and respective user is found allowing to pay for the gas through the mobile application [1].

3.3 Potential use cases

Author believes that there are multiple use cases that are yet to be implemented.

3.3.1 Car wash

This is offered as a solution by OpenALPR; however, it does not seem to be widely used outside of America, so there is still a lot of potential to offer this in other parts of the world.

Another possibility is offering it in self-service car washes. Right now, the user must manually input the timeframe they wish to use the machine for. Using license plate-based solution would allow users to enter, use the washing cycles as they see fit and as they leave, they will be prompted with the option to pay for the period the machine was in use for.

3.3.2 Restaurant orders / drive-through

This is also offered by OpenALPR, however this solution does not seem to be very popular outside of America. Bringing food out based on the license plate greatly reduces load on the drive-through line as well as eliminates the need for user to aimlessly wait inside the restaurant until their order is complete.

3.3.3 Gas stations

Paying for fuel using a phone application has been adopted in some parts of the world, for example Russia. To the best of authors knowledge in Estonia an ability to pay for gas using a phone application is only available through Cloudics[1] and Easy fuel[2] applications which support respectively Alexela and Circle K gas stations in Estonia. Cloudics application is limited to user having to manually search for the correct gas station through the mobile device. Author believes this step could be handled easily with the use of LPR cameras, further eliminating the possibility of human error, similar to the recently developed solution by Easy Fuel.

3.3.4 Seamless parking

Identifying the driver behind the wheel and having access to the mobile device also allows for a possibility to pay for parking without even taking the mobile device out of the pocket. This is achieved by saving the bank details inside the application prior to parking and having clear times of when parking started and ended i.e., when the driver entered the parking lot and when the driver left it so the driver can be automatically billed. This solution has been well implemented by Rollet[3].

3.4 LPR with the help of machine learning

Difficulty of recognizing the license plate comes from many outside factors that could affect visibility and readability of the plate itself. These factors can differ from having

[1] https://cloudics.eu/en/

[2] https://play.google.com/store/apps/details?id=com.circlek.innovation&hl=en&gl=US

[3] https://www.rollet.hu/

rain or snow coming down from the sky to simply having dirt on the license plate or having some of the letters scratched from road debris.

This is where machine learning comes in to help train the model that could be prepared for these occurrences and offer better performance than an algorithm on its own.

LPR falls somewhere between computer vision and deep learning subsets of machine learning.

3.4.1 Computer vision

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand [6].

In the context of LPR computer vision's aim is to improve detection of the vehicle and its license plate on the image.

3.4.2 Deep learning

Deep learning is a subset of machine learning where neural networks — algorithms inspired by the human brain — learn from large amounts of data. Deep learning algorithms perform a task repeatedly and gradually improve the outcome through deep layers that enable progressive learning [7].

In the context of LPR deep learning is aimed at improving recognition of characters on a detected license plate, as there are lots of small factors affecting the quality of them, such as dirt, rain, image quality, scratches etc.

4 Building the application

4.1 Technologies

4.1.1 Push notifications

Push notifications are being sent out with the help of Firebase Cloud Messaging (FCM). It is a free cross platform messaging solution provided by Google.

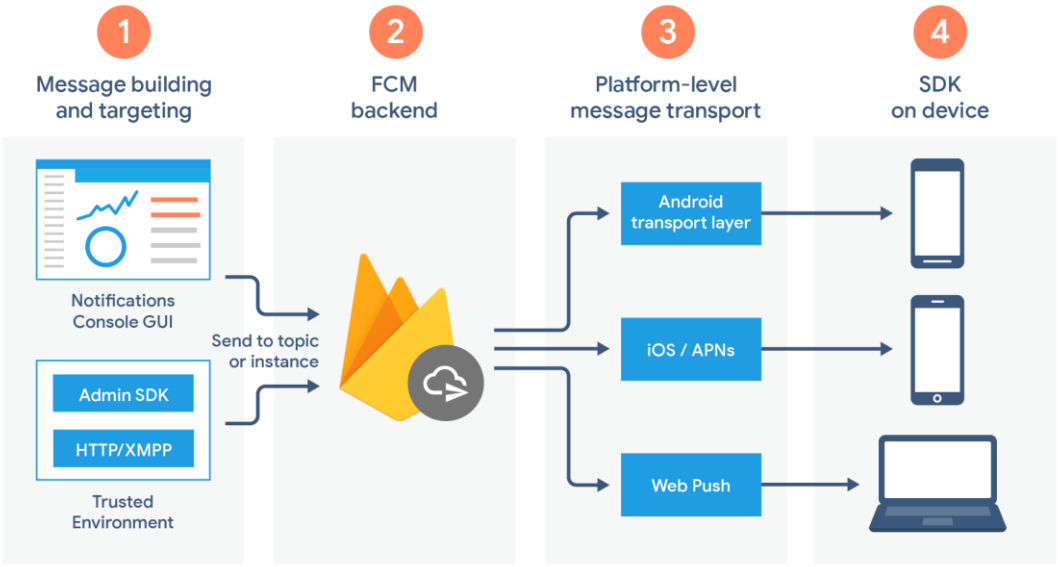

Figure 1 *FCM push notifications architecture [8]*

FCM was chosen due to being a very widely used solution, as well as doing all the heavy lifting, allowing the application to send out push notifications using simple requests. On Figure 2 our server-side application plays a role of a Trusted Environment. An example request server-side application sends to FCM has the following structure:

```
https://fcm.googleapis.com/fcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA
body: {
  to: <deviceToken>,
    notification: {
      title: 'Title of the push notification',
      body: 'Body of the push notification'
    },
    data: {
      foo:  'bar'
    },
  },
}
```

Figure 2 *FCM Payload to send a push notification*

Device token must be known in order to send a push notification to a specific device. It can be retrieved inside the mobile application, and it is developer's responsibility to store that token inside the database and keep track of that token, as it can change, for example if the user reinstalls the application [2].

FCM also allows to freely pass data to the device as long as the payload is smaller than 4000 bytes [9], which is helpful in determining how the application should handle incoming requests. In the scope of the thesis, information about the camera and service related to it is passed inside the data tag.

FCM also allows the use of silent push notifications using the following request structure:

```
https://fcm.googleapis.com/fcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA
body: {
  to: <deviceToken>,
  content_available: true,
  data: {
    foo:  'bar'
  },
}
```

Figure 3 *FCM payload to send a silent push notification*

Silent push notifications allow to fire some handlers on mobile device in the background even when the application is closed or minimised. This is especially useful in our case,

since we need to gather last known user location to account for multiple car user problem mentioned in Chapter 4.3.

4.1.2 Jetson Nano

NVIDIA has a subdivision dedicated to producing advanced AI embedded systems. Its aim is to give developers all the tools to develop and deploy AI-powered autonomous machines.

For that NVIDIA has released a series of embedded computing boards called Nvidia Jetson modules. These modules are small form-factor, high-performance computers running on specifically designed Jetson Software. They are compatible will the same AI software and cloud-native workflows used across other NVIDIA platforms [10].

Figure 4 *Jetson Nano Developer Kit [11]*

The Jetson Nano Developer Kit is used for this thesis. It is the best value-for-money compromise powerful computer that runs on NVIDIA JetPack SDK and is compatible with NVIDIA AI platform for training and deploying AI software.

Jetson Nano uses a microSD (minimum recommended is a 32 GB UHS-1) card as a boot device as well as main storage and runs off Micro-USB power supply (5V=2A) making

it extremely compact and power efficient [12]. Also having 4 USB 3.0 ports and both HDMI and DisplayPort support on the board makes the use extremely easy and intuitive.

Although Jetson Nano allows to train machine learning models, it lacks processing power to justify that. It is specifically made to run the trained models with respective software.

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from that data [13].

Jetson Nano is used as a part of this thesis, as it allows the use of a very powerful DeepStream SDK which in turn eliminates the need of creating LPR from scratch. Jetson Nano is also a very powerful tool on its own, allowing to run software on GPU, rather than the CPU making it much more efficient. Running LPR software on Jetson Nano also eliminates the need of paying extra for more powerful servers, as computation heavy tasks will be happening on it.

4.1.3 License plate recognition

Two approaches were considered when implementing license plate recognition.

First option is to implement it from scratch using Python or C++ with the help of OpenCV.

OpenCV is an open source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate use of machine perception in the commercial products [14].

Second option is implementing it with the help of Nvidia deep learning software.

NVIDIA offers lots of software options to train models and deploy them. Optionally NVIDIA also has a lot of pre-trained models available to use. The only downside is that you require a supported Nvidia graphics card (GPU) to run them, as application developed during the thesis is aimed to run on Jetson Nano, this is not an issue.

First approach was deemed excessive and was left out of the scope at the time of writing the thesis, mainly due to the software being run on Jetson Nano favouring the use of second option.

Implementation is based upon the sample LPR application provided by NVIDIA. To run it multiple Nvidia deep learning software components will be required:

1. DeepStream SDK.

DeepStream is a streaming analytic toolkit to build AI-powered applications. It takes the streaming data as input - from Universal Serial Bus (USB)/Camera Serial Interface (CSI) camera, video from file or streams over Real Time Streaming Protocol (RTSP) and uses AI and computer vision to generate insights from pixels for better understanding of the environment. DeepStream SDK can be the foundation layer for multiple video analytic solutions like understanding traffic and pedestrians in smart city, health and safety monitoring in hospitals, self-checkout and analytics in retail, detecting component defects at a manufacturing facility and others [15].

RTSP is an application-level network communication system that transfers real-time data from multimedia to an endpoint device by communicating directly with the server streaming the data [16].

DeepStream SDK supports development in C/C++ and in Python through the Python bindings.

2. Car detection model.

TrafficCamNet model detects one or more physical objects from four categories within an image and returns a box around each object, as well as a category label for each object. The four categories of objects detected by this model are – car, persons, road signs and two-wheelers. It was trained on a proprietary dataset with more than 3 million objects for car class. Most of the training dataset was collected and labelled by NVIDIA in-house from several traffic cameras in a city in the US [17].

3. Car license plate detection model.

LPDNet detect one or more license plate objects from a car image and return a box around each object, as well as an LPD label for each object. The license plate models are trained on a proprietary dataset with over 45000 US car images [18].

4. Car license plate recognition/text extraction model.

LPRNet is license plate recognition network, which aims to recognize characters in license plates from cropped RGB license plate images. LPRNet was trained on a proprietary dataset with over 310000 US license plates images [19].

LPR software used in the thesis is a sample application provided by NVIDIA.

Figure 5 *Multiple vehicles and license plate detected by NVIDIA LPR sample application*

```
Frame Number = 12 Vehicle count = 6 Person count = 0 License Plate Count = 1
License Plate 808AXF
```

Figure 6 *NVIDIA LPR sample application terminal text output at the same time as Figure 5 detected the license plate*

Sample application runs three different Deep Neural Network (DNN) models simultaneously. First DNN handles locating the vehicle itself. Second DNN tries to locate a license plate on the vehicle it has previously located. Third DNN reads the located license plate and gathers it's text. Working together in a pipeline application processes the entire flow from reading video input to writing the detected numberplate to the output.

4.1.4 Mobile application

Flutter is a free, open-source cross-platform mobile software development kit (SDK), built on Dart, maintained by Google and is rapidly gaining traction online [20].

Dart is a programming language made with fast client development in mind, maintained by Google since 2011. The goal of fast development means that Dart comes with an extensive set of built-in tools. Dart allows for code to be compiled into native language, so no special environment is required to run it. Also offering great support for asynchrony, making asynchronous operations simple and intuitive [21].

Flutter allows for easy and seamless cross-platform mobile application development. It also offers a vast library of provided UI elements, making development very fast and intuitive. Syntax is comparable to JavaScript and performance is very fast.

For the reasons mentioned above and due to author having prior experience with the language, it was the main language of choice for fast and fluid development of mobile application that would work out of the box on both Android and IOS devices.

A mobile application was developed with the idea of scalability and easily implementable further improvements in mind. At the time of development UI/UX were not priorities as the aim of the thesis was to get the prototype working and produce a usable solution.

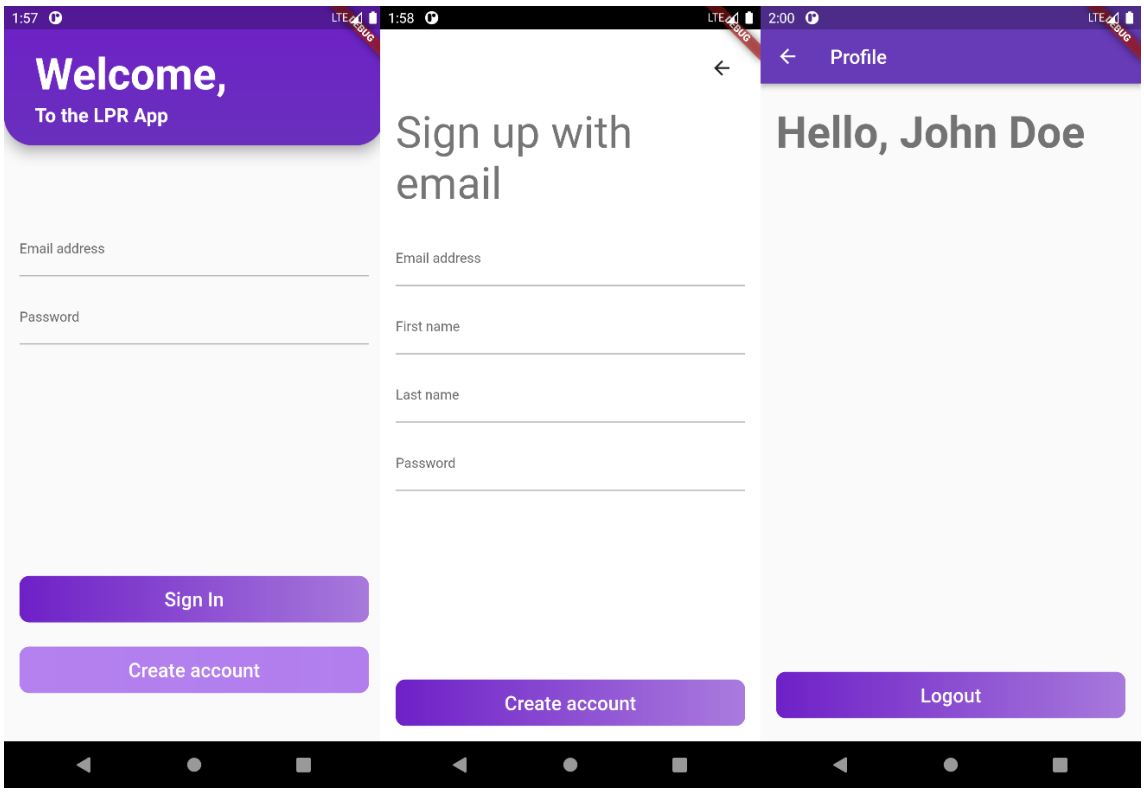

Figure 7 *User registration, login and logout views inside mobile application*

To have better user management registration was added as well as an ability to login/logout. User handling happens with the help of JWT tokens. A JWT token is returned upon successful registration or login and is then stored on the user's mobile device.

Even though UUID is used to identify users in backend, email must be unique as in the future it will be possible to change the email of the user.

Regex validation is checking email validity as a first layer of protection. Further checks will happen in backend thanks to NestJS built-in email validator. Currently the only password requirement is being longer than 6 characters.

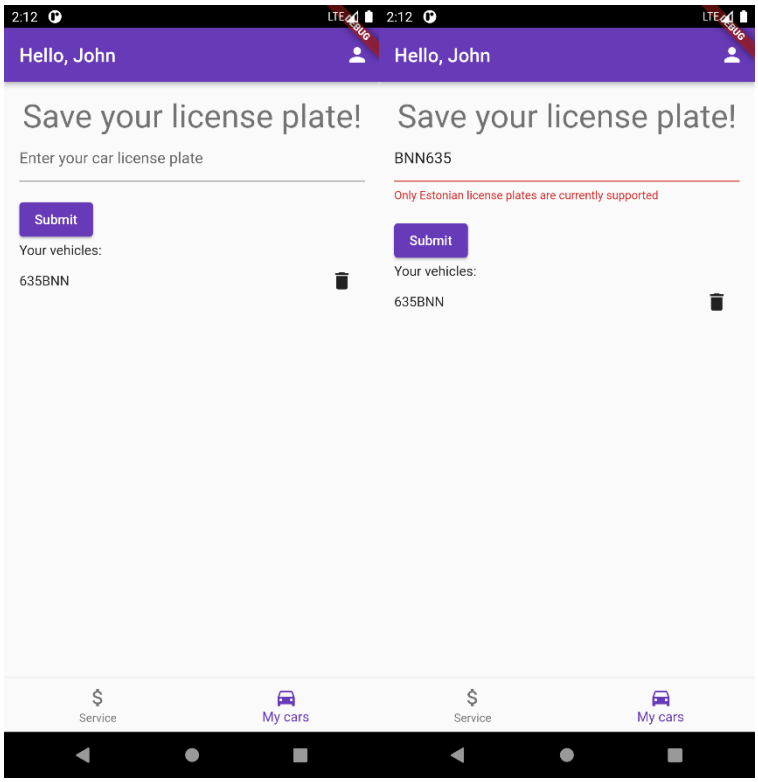

Figure 8 *Saving a license plate view and its invalid input error handling example*

User has an ability to add multiple license plates under their name, as well as remove them if such need arises.

As custom Estonian license plate detecting model is yet to be trained. Only simple Estonian license plates are currently supported like 123ABC. Further development will make it possible to support even custom numbers such as ACTION1.

Simple error handling has been implemented by applying Regex that only allows inputs of 3 numbers followed by 3 letters.

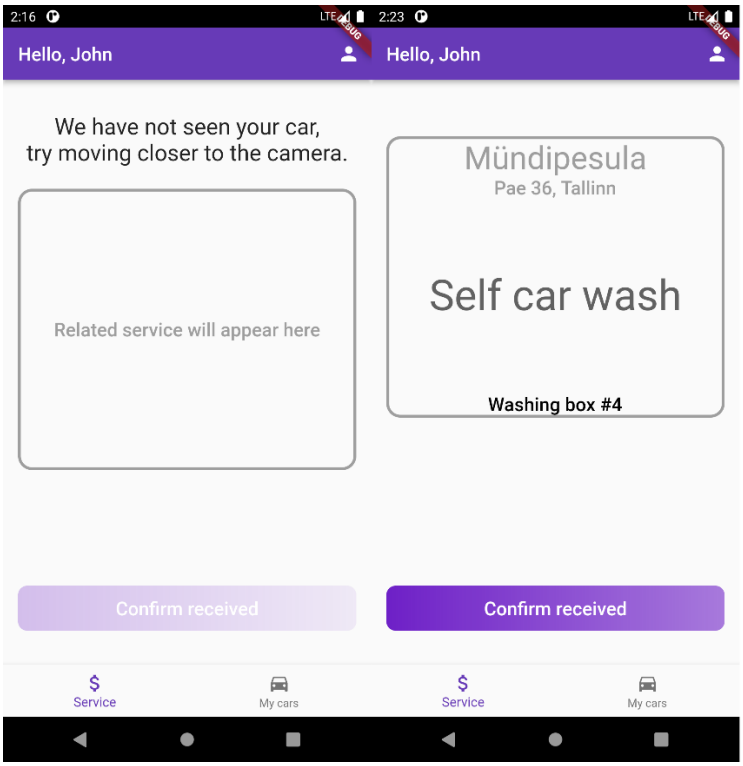

Figure 9 *Service confirmation view*

Confirmation view is a point where related service will appear once the push notification comes in. It will appear regardless of whether the user has clicked the push notification manually or just had this view open on his mobile device.

Clicking on confirmation button currently sends a request to backend saying that it has been pressed. Further implementation will direct the user to the service-related application or just open web browser window with the link saved under the service.

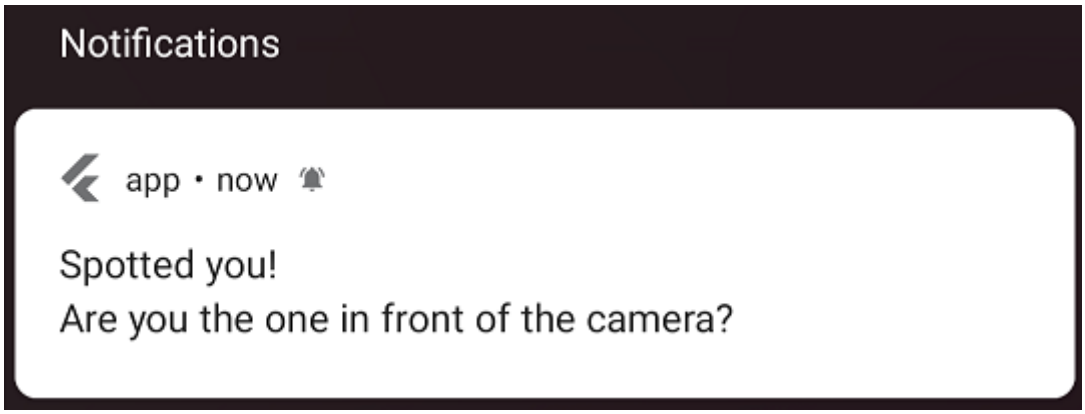

Figure 10 *Push notification arriving upon license plate being read*

Push notifications are fully supported. They take user to the confirmation screen inside the application. The information about related service is stored in cache as soon as the push notification arrives and is removed as soon as confirmation button is pressed. This is done to ensure that information is displayed inside the application correctly regardless of how the user had reached it. Silent notifications are also working. If silent notification arrives then the user's current or last known location is silently sent to backend in the background.

4.1.5 Backend

NestJS was the framework of choice for backend, due to author having prior experience in using it which made it stand out compared to other Node.js frameworks.

NestJS is a framework for building efficient and scalable Node.js server-side applications. It uses progressive JavaScript, is built with, and fully supports TypeScript and combines elements of Object Oriented Programming (OOP), Functional Programming (FP) and Functional Reactive Programming (FRP). Architecture of NestJS was heavily inspired by Angular [22].

Using NestJS allows the use of a very powerful tool called TypeORM. TypeORM brings Object Relational Mapping (ORM), a technique creating a layer between the language and the database, helping programmers work with data without the OOP paradigm [23].

Writing raw SQL queries can be extremely time consuming, especially as the database grows. ORMs create a model of the object-oriented program with a high-level of abstraction. In other words, it makes a level of logic without the underlying details of the

code. Mapping describes the relationship between an object and the data without knowing how the data is structured. The model can then be used to connect the application with the SQL code needed to manage data activities. This “plumbing” type of code does not have to be rewritten, saving the developer a tremendous amount of time [23].

Database of choice was PostgreSQL. It is a powerful relational database that has huge community support is still actively being updated. It also has a very sophisticated way of generating unique values thanks to UUID data type.

User sensitive endpoints are protected by JWT authentication.

4.2 Architecture

Application architecture was designed with support for different use cases in mind. Backend should be able to handle requests coming in from the LPR application about the license plates as well as requests coming in from the user’s mobile device. Implementation of new features must not hinder existing ones.

4.2.1 Database architecture

The architecture of the database was done with the intention of being able to expand every single table further and add more tables without affecting existing relations.

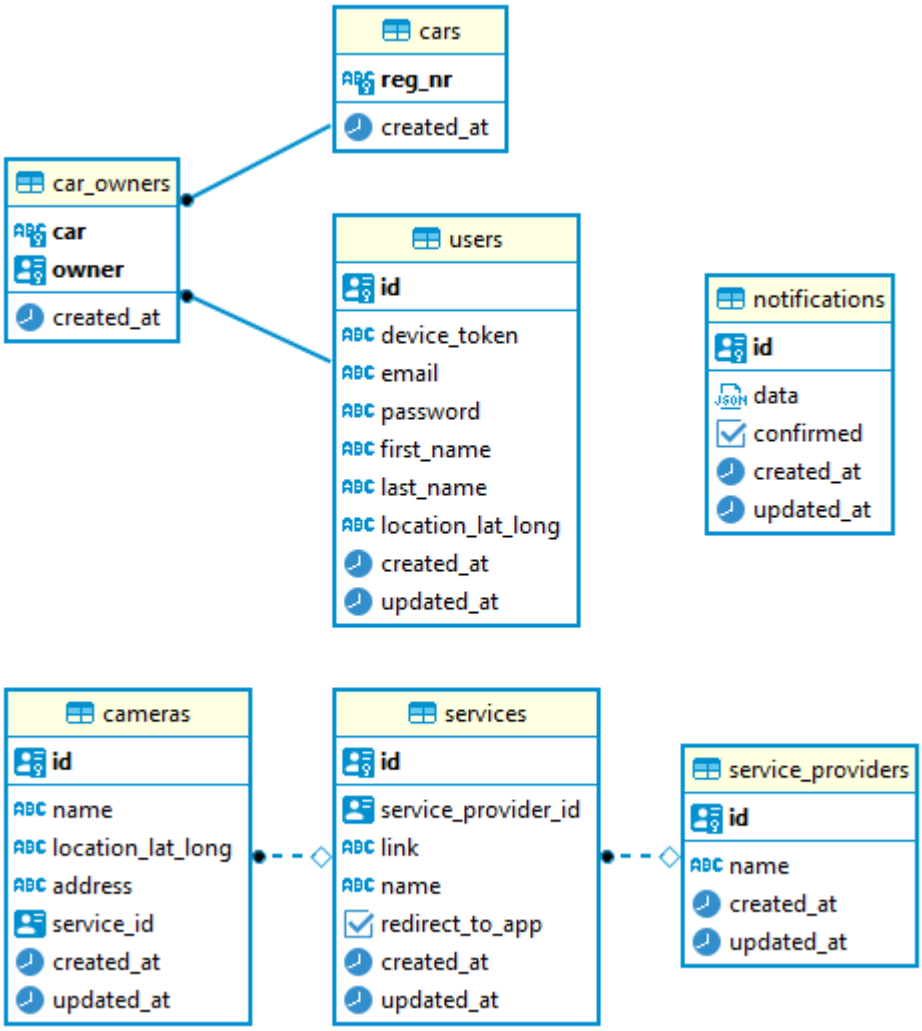

Figure 11 *PostgreSQL database architecture*

The idea of this architecture was to explicitly separate services from the user related information. This is done since the LPR software can be offered independently from the mobile application. Since a single camera can only be related to one service being offered, knowing id of the camera that read the license plate is enough to tell which service is being provided and by whom.

Notifications table store basic information about the push notifications that were sent out and *confirmed* field acts as a confirmation and gets updated when the user clicks the respective button inside the application. That is done with the idea of making possible problem detection easier, as there will be a clear backlog of unconfirmed notifications in case something went wrong.

With scalability in mind some tables like *cars* and *service_providers* currently don't store any extra information; however, they are already fighting against duplication of information.

4.2.2 Application architecture

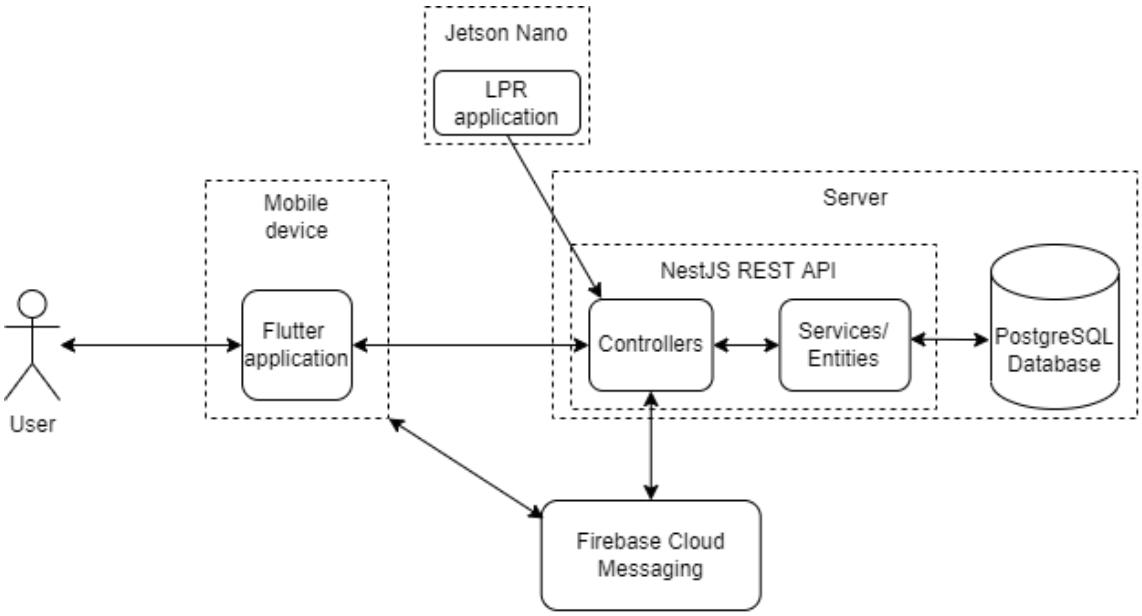

Figure 12 *Application architecture drawing*

Backend NestJS REST API is responsible for handling two different types of tasks.

1. Accepting requests coming in from the LPR software.

These request all follow the same structure, the only variables are what camera the information came from and what license plate has been read.

Using the information about the camera it is then decided whether there will be a request sent straight to the service provider or will the application try to find the user from the database related to that license plate and then attempt to send a push notification

Sending a push notification is as simple as sending a respective request similar to the one mentioned in Figure 3. If a license plate has multiple related users, then the difficulty rises, and steps mentioned in Chapter 4.3.1 are performed.

2. Support for the mobile application

Flutter mobile application communicates with the backend through the NestJS REST API.

To allow user registration and handling, authentication had to be implemented. This has been done with the help of JWT tokens which store user email and the time they were issued inside the payload. This is also very handy as it allows to make requests without manually specifying user who is making the request, as with the help of email inside the JWT payload user can easily be found.

Multiple endpoints allow addition and removal of license plates and accepting confirmation of receiving push notifications.

4.3 Multiple car user problem

An interesting problem came to light during the implementation of sending out push notifications. If a license plate has multiple related users, then the user who will receive the push notification has to be selected.

4.3.1 Developed solution

Author’s solution makes use of the location of the mobile device to determine who is the closest one to the location of the camera.

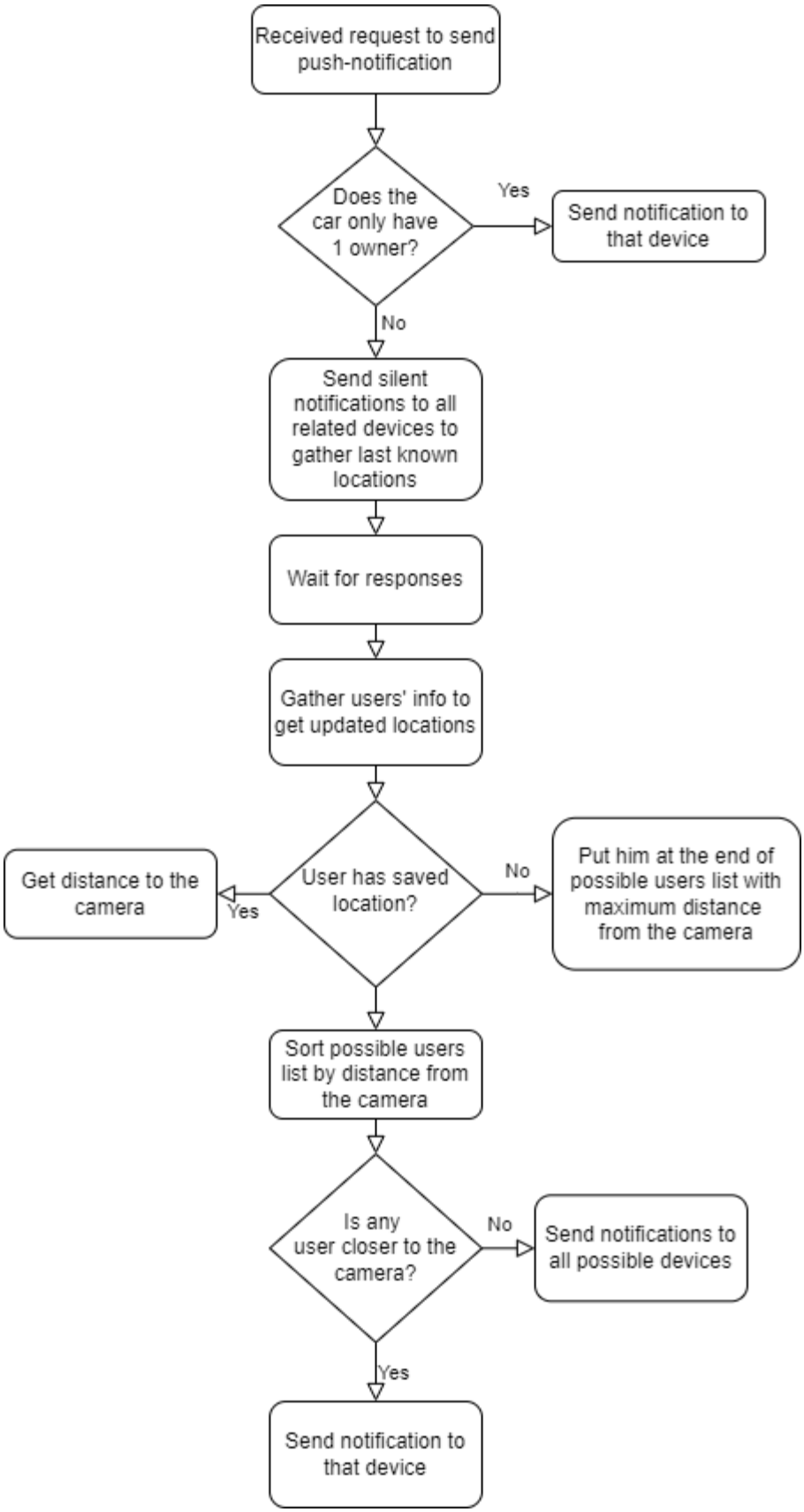

Figure 13 *Multiple car user problem solution implementation steps*

Getting the distance from the camera to the location is done using Haversine formula.

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ is latitude, λ is longitude, R is earth's radius and d is the distance between 2 points on the surface of a sphere [24].

Users whose location is unknown are still accounted for as possible candidates, as for example if the car has 2 users and it is known that 1st user is further than 1 km away from the camera then the 2nd user is the one in front of the camera.

Developed solution produces accurate enough results, as the most likely scenarios are that only 1 of the users' is in the car so the other ones are far away or there are multiple users in the car in which case either of them can handle starting the service through a mobile device.

4.3.2 BLE beacon

Another more bulletproof solution would be using a Bluetooth Low Energy (BLE) beacon.

A Bluetooth beacon is a small wireless device that works based on Bluetooth Low Energy. It's kind of like a lighthouse: it repeatedly transmits a constant signal that other devices can see. Instead of emitting visible light, though, it broadcasts a radio signal that is made up of a combination of letters and numbers transmitted on short, regular intervals. A Bluetooth-equipped device like a smartphone, gateway, or access point can "see" a beacon once it's in range, much like sailors looking for a lighthouse to know where they are [25].

BLE is mostly used in short-range applications. Most Bluetooth beacons can reliably transmit up to approximately 30 meters without any physical obstructions. A typical operating range is around 2 to 5 meters, depending on the transmit power. The higher the range, the higher the battery consumption [26].

Since most mobile devices nowadays turn on Bluetooth automatically and keep it on all the time, a beacon can be used to send a silent signal to the mobile device. Thanks to that and working in short range a signal can be sent out by BLE beacon as soon as LPR has read a license plate and confidently find the correct mobile device.

The only downside is requiring more infrastructure by having to maintain the BLE beacon together with the camera. However, this change is minor and receiving a completely confident results in return makes this option very appealing.

At the time of writing the thesis this solution was left out due to fear of not making the working prototype in time for the thesis submission deadline, as well as not having experience and access to BLE beacon, however, future developments will definitely attempt to implement BLE beacon into the architecture.

4.4 Integration into other applications

In order to offer LPR software on its own as a service, requests coming from the LPR software running on the camera need to be handled separately. Ideally these requests would still follow the same structure to avoid creating extra endpoints and duplicating unnecessary information.

Knowing, that there is a direct relation between the camera and the service. Service entity was used to define how the request coming from the camera should be handled. Currently a simple boolean value saved in the database on service table is used to direct the flow of the incoming request.

If service provider only wants to use the LPR software without the mobile application and push notifications support, then huge part of sending push notifications can be omitted and data about the read license plate will be sent straight to the service provider for them to handle on their own.

5 Potential avenues for future research and development

This solution had a defined MVP that needed to be achieved. Author believes there is still a lot of potential to grow the application and wishes to continue to develop this application further.

These features are prioritised and are planned to be developed next:

1. Add support for custom numberplates inside the application, as currently it only supports traditional Estonian numberplates like 123ABC.

2. Creating an admin interface/API. It is clear with this prototype that it is essential as currently only author could create new camera entities and add/update existing service information which could clearly be handled by the consumer company. As well as adding some graphics and statistics of how often this service is being used.

3. Explore the possibility of not needing to pick up the phone at all and make the use of some services completely autonomous, as it was done by Rollet[1] and parking. As this would complete the initial vision of the author making use of all available resources and eliminating the need to worry about whether parking is paid for or not, having to constantly make sure that you have the parking ticket on you or in some cases not worrying about whether you need to put the parking clock up or not. This heavily ties the application with parking applications.

4. Find a more foolproof solution to make sure that the license plate has not been read incorrectly, as it would raise major concerns for users' if they unexpectedly received a notification.

5. Generating test data from Estonian numberplates to train a better custom LPR model tailored towards vehicles with Estonian license plates.

[1] https://www.rollet.hu/

6 Summary

The aim of the thesis was to create a solution that would allow the client to link his phone with his car's license plate. This change would allow to greatly expand possibilities and use cases of LPR.

In the scope of the thesis providers of similar solutions were explored. It has been confirmed that popular solution providers mainly offer only LPR as a software and the idea of making a connection between the user's mobile device and a vehicle is not yet properly explored.

Developed solution can use the license plate to identify the user and send a push notification to open a respective view inside mobile application which directs the user to start using the related service. It can also be offered purely as a service similar to what existing solutions are proposing. This solution is preferred as it is running on Jetson Nano which eliminates the need of paying for powerful servers, as this small device does all the work.

During the development I realised how important it is to clearly set the goal and properly plan ahead of time as there are always things that you could develop and improve which in turn make it harder to focus on one task, since you are always looking into new features that could be implemented. I also clearly underestimated the difficulty of implementing LPR software, as many implementations fall short as soon as the conditions get a little more difficult, so a lot of the time was spent looking into possibilities of reimplementing LPR instead of focusing on areas I had more experience with, which could have produced a more refined solution.

The main contributions of this thesis are:

- Creating a technical solution with a scalable architecture:
    - Creating a connection between the mobile device and the license plate;

    - a mobile application that allows for user registration and gives them an option to link license plates to their account, as well as confirmation of incoming push notifications to start the use of the related services;
    - running LPR software on Jetson Nano;
    - server-side application capable of simultaneously handling requests coming from LPR software and the mobile application.
- A prototype that with further development can be offered as a service either with or without support for the developed mobile application.

References

[1] https://forte.delfi.ee/artikkel/96517681/video-uus-lahendus-eestis-kaivitati-tana-ennelounal-autonumbri-pohised-maksed

[2] https://firebase.google.com/docs/cloud-messaging/manage-tokens

[3] https://www.licenseplatesrecognition.com/how-lpr-works.html

[4] https://www.twi-global.com/technical-knowledge/faqs/what-is-a-smart-city

[5] https://viso.ai/computer-vision/automatic-number-plate-recognition-anpr/

[6] https://www.ibm.com/topics/computer-vision

[7] https://www.ibm.com/cloud/watson-studio/deep-learning

[8] https://firebase.google.com/docs/cloud-messaging/fcm-architecture

[9] https://firebase.google.com/docs/cloud-messaging

[10] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

[11] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[12] https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#prepare

[13] https://docs.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model

[14] https://opencv.org/about/

[15] https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_Overview.html

[16] https://www.techtarget.com/searchvirtualdesktop/definition/Real-Time-Streaming-Protocol-RTSP

[17] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/trafficcamnet

[18] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/lpdnet

[19] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/lprnet

[20] https://flutter.dev/

[21] https://inlab.fib.upc.edu/en/blog/what-dart-programming-language

[22] https://docs.nestjs.com/

[23] https://www.altexsoft.com/blog/object-relational-mapping

[24] https://www.movable-type.co.uk/scripts/latlong.html

[25] https://kontakt.io/what-is-a-beacon/

[26] https://www.novelbits.io/overview-bluetooth-beacons-part-1

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Mihhail Kohhantšuk

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Technical Solution and a Prototype for Linking License Plate with a Mobile Device”, supervised by Innar Liiv and Janno Stern
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

02.05.2022

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.