1. **Table of contents**

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)
- [Login](#)
- [List of vehicles](#)
- [Read Registration](#)
- [Vehicle's List Of Documents](#)

[Key Considerations](#)
- [How will your app handle data persistence?](#)
- [Describe any edge or corner cases in the UX.](#)
- [Describe any libraries you'll be using and share your reasoning for including them.](#)
- [Describe how you will implement Google Play Services or other external services](#)

[Next Steps: Required Tasks](#)
- [Task 1: Project Setup](#)
- [Task 2: Implement UI for Each Activity and Fragment](#)
- [Task 3: Connect to API(S)](#)
- [Task 4: UI Testing](#)
- [Task 5: Gradle Test Automation](#)
- [Task 6: Make It Material](#)

GitHub Username: Radsen

Autollow

Description

Some cities in the world use cameras to traffic control, not only to check if you crossed when the light was red, also to verify if your documents are up to date. To own a vehicle in my country you need to have some documents: The license, the registration, a mandatory insurance(which does not cover everything), the complementary insurance, the emission control certificate. Besides that you have to pay taxes and a special tax for roads.

If for some reason you do not have all the documentation in order, the cameras or the officers can give you a ticket that will make your wallet miserable.

Problem:
Keep track of the necessary documents for a vehicle to avoid tickets or overcharges due to documents missing or expired.

Proposed Solution:
The user will sign up to enter the application and he'll create the vehicle(s) he wants to keep track of its documentation. Once the vehicles are added, he'll need to add the documents for that vehicle (Some countries provide this information through web services and is public, but the person ID is required for validation). The documents with expiration day will create alarms to remind the user the documents are close to expire and they need to be update it.

Intended User

Vehicle owners (Motorcycles, Automobiles, Trucks, etc...) which require the documents up to date.

Features

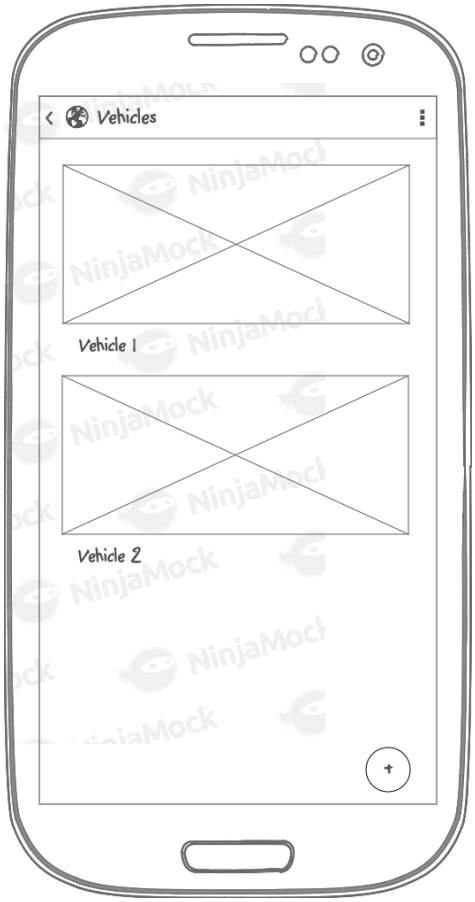
- Saves information
- Takes pictures
- Notifies user document expiration
- Connects to API(s)

User Interface Mocks

Login

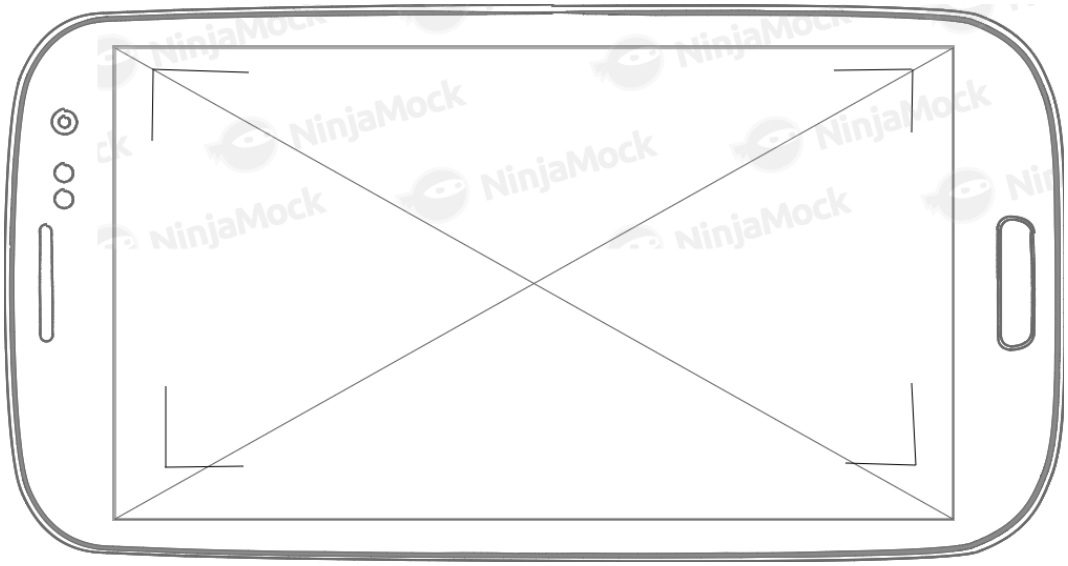
If the user is not authenticated, the app will display this screen for authentication.

List of vehicles



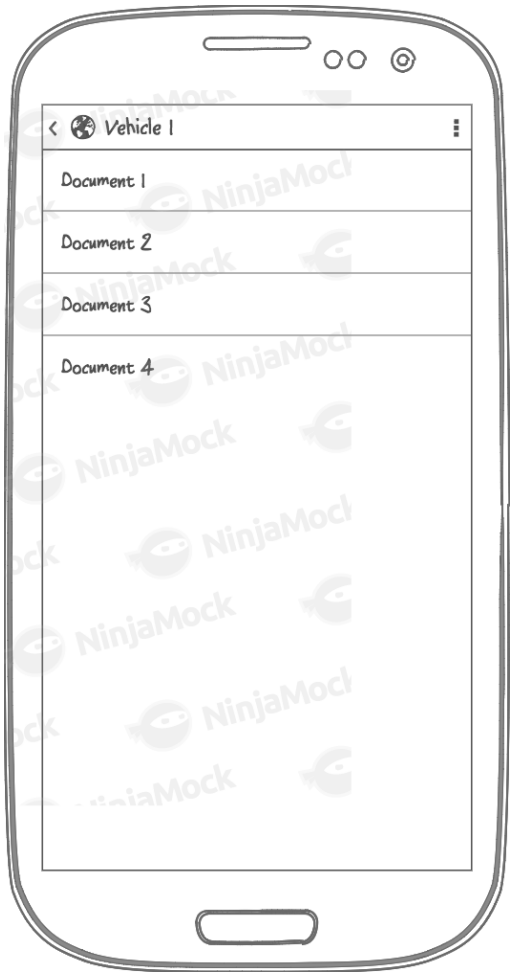
This screen once authenticated will display all the vehicles the user is tracking, but if no vehicles are listed pressing the "+" button he'll be able to add a vehicle.

Read Registration



Here the user would scan the registration card and take all the relevant information to be saved and used by the app (I’m still not sure if using mobile vision I can get the data from my country’s registration card but is worth to try).

Vehicle's List Of Documents



The list of required documents for a given vehicle.

Widget

The widget will show the vehicle the user has marked as default or in use and it will tell him if he could use it that day (Remember we have a curfew to use or vehicles in my country). The placeholder could show a checkmark or a forbidden to let him know if it is possible to use the vehicle that day or just change the background color for a color like green for available, red for can't be used and maybe yellow or orange if the time of the curfew has not started yet.

Key Considerations

How will your app handle data persistence?

The app requires the information to be available when the user wants to see it, regardless he is online or offline and also this information should be synchronized with the cloud or a server. Seems to be a good candidate to use Firebase Real Time Database. However, the documents

have different content and I'm not sure how easy would be to model a non SQL DB to store documents based on location or language.

Describe any edge or corner cases in the UX.

Once the registration is read, we need to present the information captured with the camera to the user to validate if the data is correct and whether is placed in the right fields. If not, the user should be able to edit any data that is not right and save it.

Describe any libraries you'll be using and share your reasoning for including them.

Some libraries I could use for the app are:
Butterknife: Very handy, and since I used it for the first time in the projects I learned to love it (Not that I can't use findWiewById everywhere and everytime I need it)

Picasso: Although, I don't think the app will be requesting too many images from apis. I like to have it around to deal with caching and download images. So far, the only image I think we are going to be using is the one the user could take from its vehicle for the vehicle profile in the list of vehicles.

Timber: To log things for debug.

Mobile Vision: For what I read I could be an interesting choice to use for capturing the data from the registration card. It says, it can read text from business cards, credit cards and works fine with latin based languages.

Describe how you will implement Google Play Services or other external services

- Google places

I don't know yet how to use Google play services in this app. But one thing I could add is the maps and geo fences(My country is one of the few which has a vehicular curfew for pollution and mobility and this is one of the most popular tickets in my city).

- AddMob

Another thing we could incorporate to the app is adds related to vehicles, but I'm not sure if I want to have adds for this application given that the notifications and also the publicity seems like a little too much and I don't want the user to be overloaded with information from the app.

Describe how you will connect to the APIs

Depending on the information we want to present we could use IntentService or JobDispatcher. Looking at the apis I have found the documents don't change very often. Then, it make sense to use an IntentService to bring the documents. However, if we want the tickets info makes more sense to have the JobDispatcher, because a ticket could be given to you by a camera and you would never known you got one unless you enter the app and request that information. Therefore, it should be retrieved every 24 hours or every week to let you know if you have received one.

As a user I'm thinking I won't be committing violations all the time which makes this a candidate for a push mechanism more than a pull request basis. What I'm saying is I'm not thinking all the time "Wow let me see if today I broke the traffic law", but I really want to know if I made a mistake and I did not know.

In that case using firebase to get the information from the APIs and then push the data to the device makes sense.

Note: Looking for push server you need to pay for most of them if is not all of them. So, Probably it could be better if I stick to the JobDispatcher approach. (Suggestions would be welcomed)

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

1. Create github repository.
2. Create development branch
3. Clone the repository
4. Add the libraries:
    - Butterknife
    - Mobile Vision
    - Picasso
    - Timber
    - Espresso

- Firebase
- Google Play Services (Places and Adds)

Task 2: Implement UI for Each Activity and Fragment

1. Build LoginActivity
2. Build LoginFragment
3. Build MainActivity (Vehicle List)
4. Build MainFragment (Vehicle List)
5. Build Vehicle Adapter
6. Build RegistrationReaderActivity
7. Build VerificationActivity
8. Build VerificationFragment
9. Build VehicleCardActivity (Document List)
10. Build VehicleCardFragment
11. Build Document Card Adapter

Task 3: Connect to API(S)

My country provides web services to retrieve information for some of the already mentioned documents and most of the data to query them can be found in the vehicle registration card.

1. Retrieve Document Information for document one
2. Save Document to DB (Not sure how to do this with Firebase Real-Time Database)

I'm not sure if I have to call different end points to bring all docs or just one, if more than one needs to be called then tasks 1 and 2 will be multiplied by the number of end points.

Task 4: UI Testing

1. Test login messages
2. Test if after login goes to Vehicle List
3. Test if Vehicle list is empty
4. Test if vehicle list has data
5. Test if Add Vehicle works
6. Test verification view is displayed after capture
7. Test Document List is empty
8. Test if Document list has data

Task 5: Gradle Test Automation

1. Create Automation Script

Task 6: Make It Material

1. Create App Theme
2. Create Styles
3. Apply Metrics
4. Add Material Icons
5. Add Meaningful Motion

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"