# *Acceptance testing – Car Yard Assist*

Since the GPS on most tablet emulators in Android Studio does not work properly this appendix will show you how to fix this problem and also how to run the application.

- If you intend to run the application on a tablet emulator, you can skip **Step S10**.
- If you intend to run the application on a real device, you can skip **Step S5-S9**.
- If you already have Android Studio installed and a tablet emulator with a GPS that works, you can skip **Step S1**, **S5-S10**.

The application can be tested on a phone emulator but the UI is not designed for a phone screen.

Since the application is not avaliable in the Google Play-store, the instructions down below will have to be completed first in order to test the application.

## Setting up the application

### S1.

Download Android Studio from the following link: https://developer.android.com/studio?gclid=EAIaIQobChMI9KOC49ON7gIV5hoGAB2phA-pEAAYASAAEgL98fD_BwE&gclsrc=aw.ds

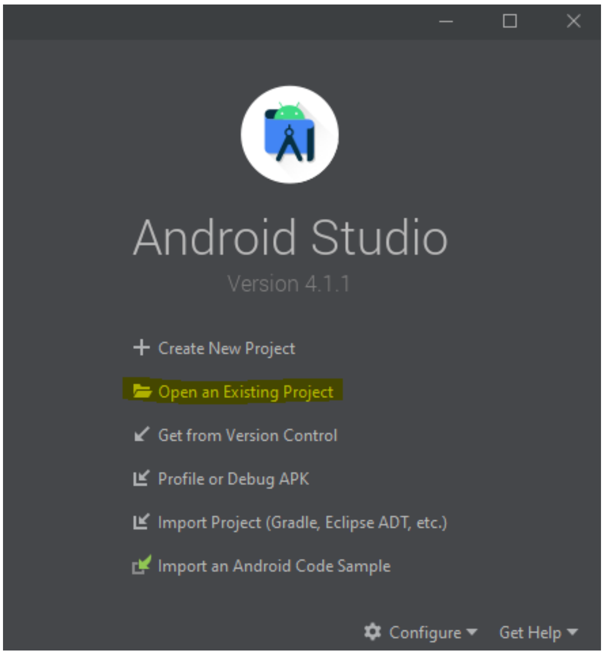
### S2.

Download the project from the following link: https://github.com/Niklas-Rig/DVD313Project3

Press the green ”**Code**”-button and then press ”**Download ZIP**”. Once the project has been downloaded, unzip the files.

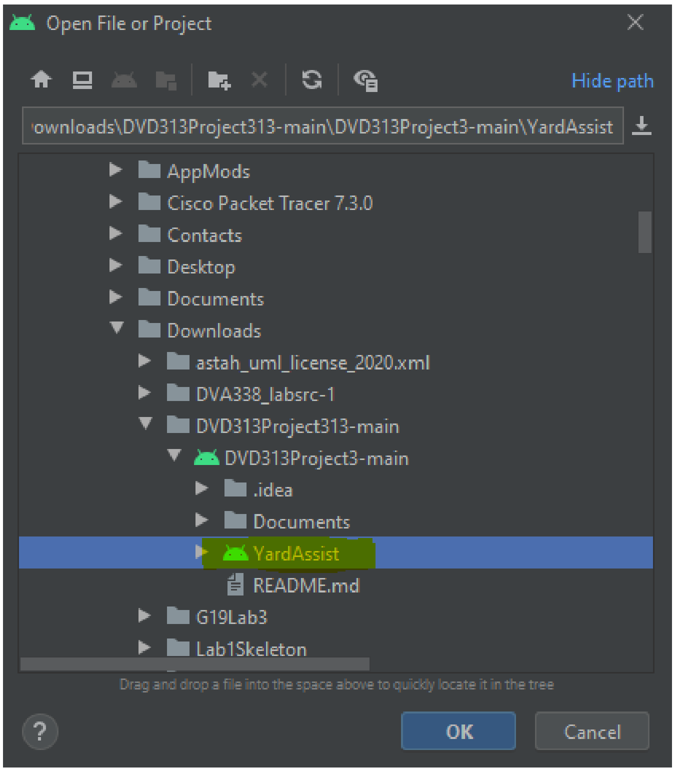
### S3.

Open Android Studio and click **”Open an Existing Project”**



**S4.**

Select the project that was downloaded and unzipped in **Step S2**, make sure to only import the **”YardAssist”** folder as shown in the picture below.



Open File or Project

Hide path

ownloads\DVD313Project313-main\DVD313Project3-main\YardAssist

- AppMods
- Cisco Packet Tracer 7.3.0
- Contacts
- Desktop
- Documents
- Downloads
  - astah_uml_license_2020.xml
  - DVA338_labsrc-1
  - DVD313Project313-main
    - DVD313Project3-main
      - .idea
      - Documents
      - YardAssist
      - README.md
  - G19Lab3
  - Lab1Skeleton

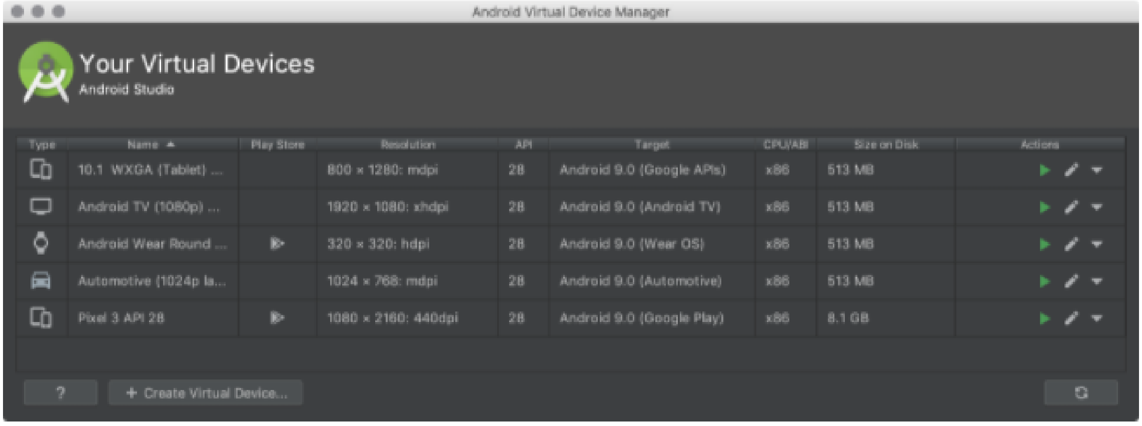
Drag and drop a file into the space above to quickly locate it in the tree

**S5.** (For using an emulator)

Follow the steps below in order to create an emulator in order to be able to run the application.

To create a new AVD:

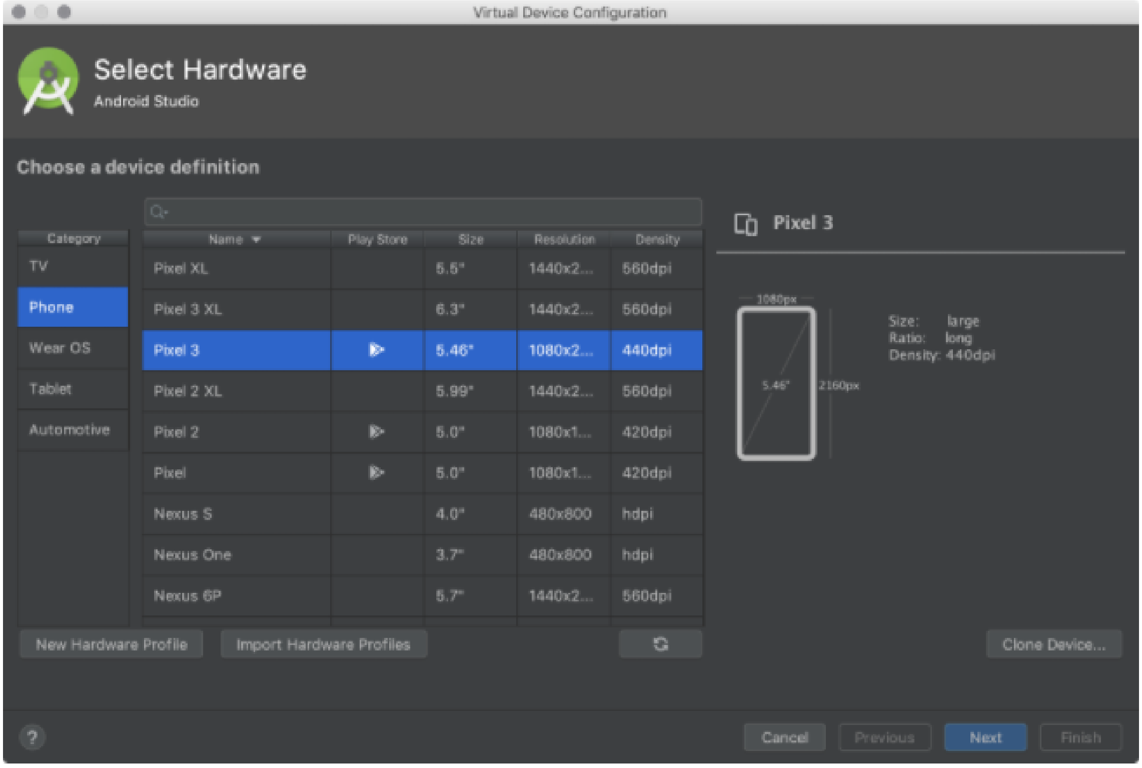
1. Open the AVD Manager by clicking **Tools > AVD Manager**.



| Type | Name ▲ | Play Store | Resolution | API | Target | CPU/ABI | Size on Disk | Actions |
|---|---|---|---|---|---|---|---|---|
| | 10.1 WXGA (Tablet) ... | | 800 × 1280: mdpi | 28 | Android 9.0 (Google APIs) | x86 | 513 MB | ▶ ✎ ▾ |
| | Android TV (1080p) ... | | 1920 × 1080: xhdpi | 28 | Android 9.0 (Android TV) | x86 | 513 MB | ▶ ✎ ▾ |
| | Android Wear Round ... | ▶ | 320 × 320: hdpi | 28 | Android 9.0 (Wear OS) | x86 | 513 MB | ▶ ✎ ▾ |
| | Automotive (1024p la... | | 1024 × 768: mdpi | 28 | Android 9.0 (Automotive) | x86 | 513 MB | ▶ ✎ ▾ |
| | Pixel 3 API 28 | ▶ | 1080 × 2160: 440dpi | 28 | Android 9.0 (Google Play) | x86 | 8.1 GB | ▶ ✎ ▾ |

2. Click **Create Virtual Device**, at the bottom of the AVD Manager dialog.

   The **Select Hardware** page appears.



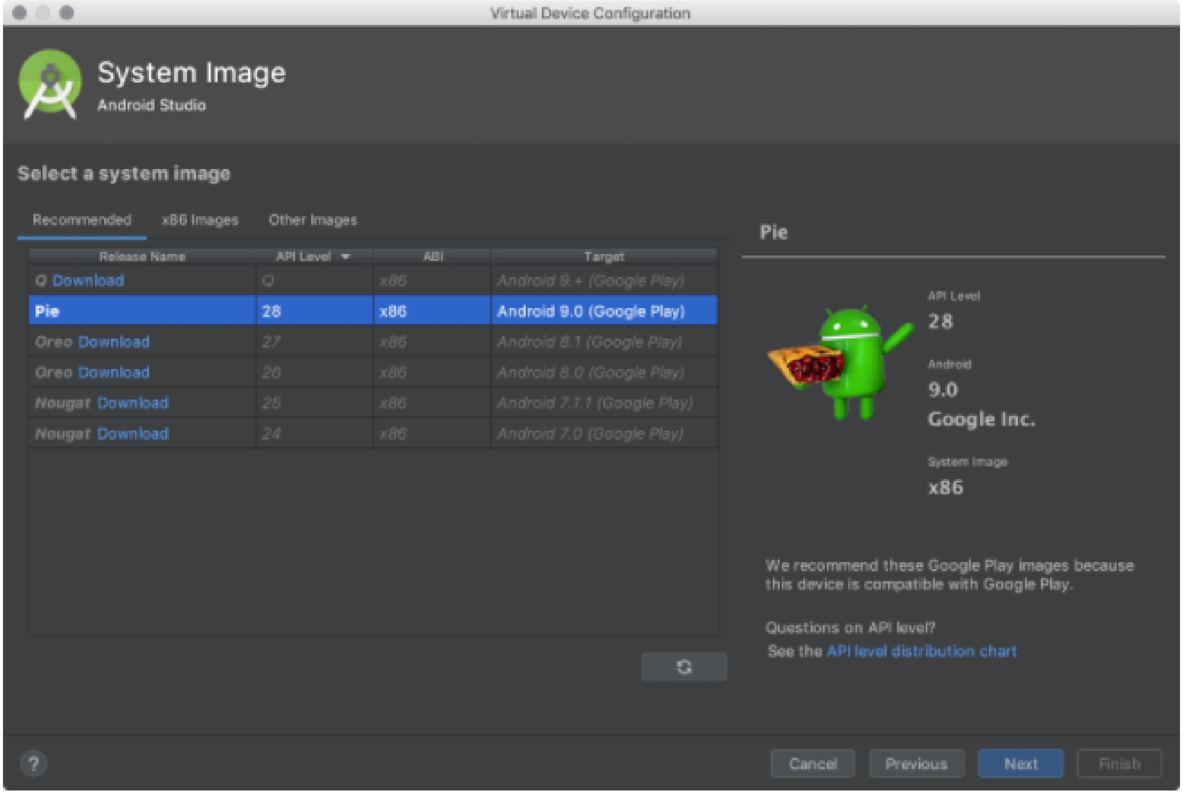
| Name ▼ | Play Store | Size | Resolution | Density |
|---|---|---|---|---|
| Pixel XL | | 5.5" | 1440x2... | 560dpi |
| Pixel 3 XL | | 6.3" | 1440x2... | 560dpi |
| Pixel 3 | ▶ | 5.46" | 1080x2... | 440dpi |
| Pixel 2 XL | | 5.99" | 1440x2... | 560dpi |
| Pixel 2 | ▶ | 5.0" | 1080x1... | 420dpi |
| Pixel | ▶ | 5.0" | 1080x1... | 420dpi |
| Nexus S | | 4.0" | 480x800 | hdpi |
| Nexus One | | 3.7" | 480x800 | hdpi |
| Nexus 6P | | 5.7" | 1440x2... | 560dpi |

   Notice that only some hardware profiles are indicated to include **Play Store**. This indicates that these profiles are fully CTS compliant and may use system images that include the Play Store app.

3. Select a hardware profile, and then click **Next**.

   If you don't see the hardware profile you want, you can create or import a hardware profile.

The **System Image** page appears.



Select a system image

| Release Name | API Level | ABI | Target |
|---|---|---|---|
| Q Download | Q | x86 | Android 9.+ (Google Play) |
| Pie | 28 | x86 | Android 9.0 (Google Play) |
| Oreo Download | 27 | x86 | Android 8.1 (Google Play) |
| Oreo Download | 26 | x86 | Android 8.0 (Google Play) |
| Nougat Download | 25 | x86 | Android 7.1.1 (Google Play) |
| Nougat Download | 24 | x86 | Android 7.0 (Google Play) |

Pie

API Level
28

Android
9.0
Google Inc.

System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the API level distribution chart

. Select the system image for a particular API level, and then click **Next**.

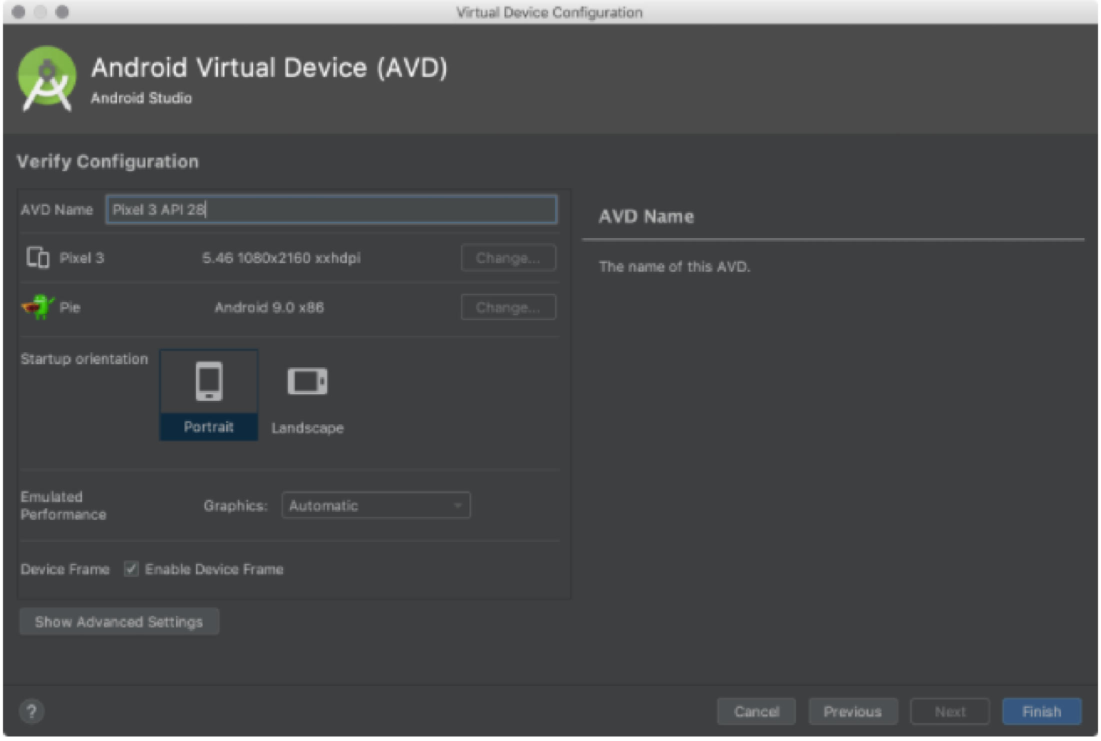
The **Recommended** tab lists recommended system images. The other tabs include a more complete list. The right pane describes the selected system image. x86 images run the fastest in the emulator.

If you see **Download** next to the system image, you need to click it to download the system image. You must be connected to the internet to download it.

The API level of the target device is important, because your app won't be able to run on a system image with an API level that's less than that required by your app, as specified in the `minSdkVersion` attribute of the app manifest file. For more information about the relationship between system API level and `minSdkVersion`, see Versioning Your Apps.

If your app declares a `<uses-library>` element in the manifest file, the app requires a system image in which that external library is present. If you want to run your app on an emulator, create an AVD that includes the required library. To do so, you might need to use an add-on component for the AVD platform; for example, the Google APIs add-on contains the Google Maps library.

The **Verify Configuration** page appears.



Virtual Device Configuration

Android Virtual Device (AVD)

Android Studio

Verify Configuration

AVD Name: Pixel 3 API 28

Pixel 3 5.46 1080x2160 xxhdpi

Pie Android 9.0 x86

Startup orientation: Portrait

Emulated Performance Graphics: Automatic

Device Frame ☑ Enable Device Frame

AVD Name

The name of this AVD.

5. Change AVD properties as needed, and then click **Finish**.

   Click **Show Advanced Settings** to show more settings, such as the skin.

   The new AVD appears in the **Your Virtual Devices** page or the **Select Deployment Target** dialog.

To create an AVD starting with a copy:

1. From the **Your Virtual Devices** page of the AVD Manager, right-click an AVD and select **Duplicate**.

   Or click Menu ▼ and select **Duplicate**.

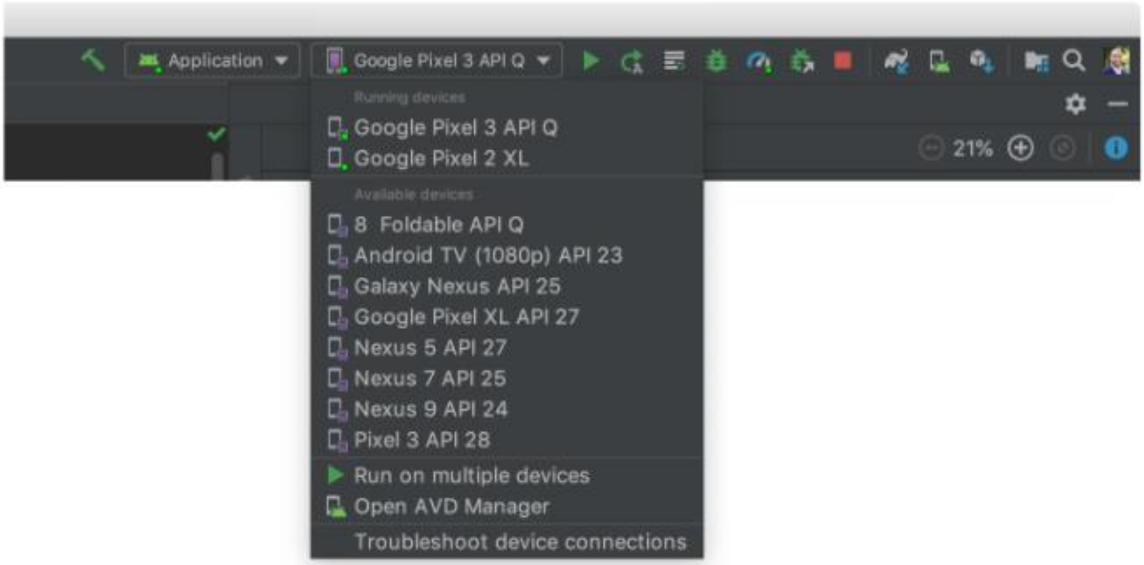
   The **Verify Configuration** page appears.

2. Click **Change** or **Previous** if you need to make changes on the **System Image** and **Select Hardware** pages.
3. Make your changes, and then click **Finish**.

   The AVD appears in the **Your Virtual Devices** page.

Run the app on an emulator as follows:

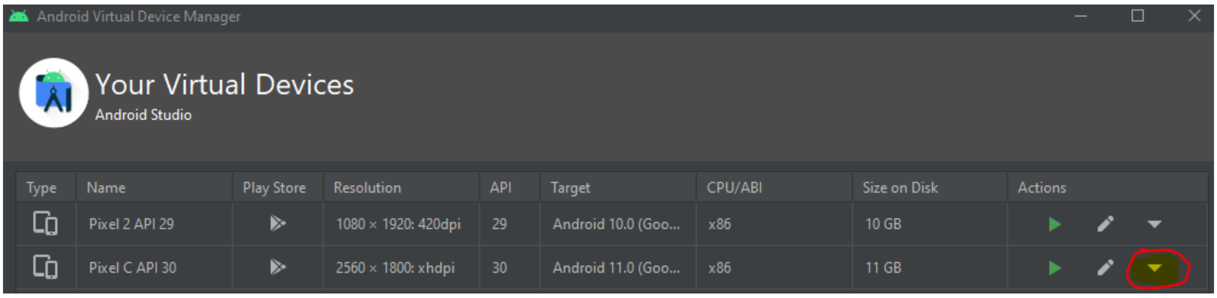
1. In Android Studio, create an Android Virtual Device (AVD) that the emulator can use to install and run your app.
2. In the toolbar, select your app from the run/debug configurations drop-down menu.
3. From the target device drop-down menu, select the AVD that you want to run your app on.



4. Click **Run** ▶.

## S6.

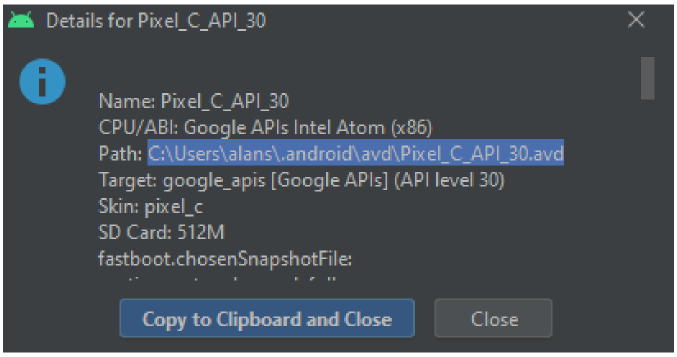
Press the red ”**stop**”-button in Android Studio to stop the application. Now open the AVD-manager again. Then press the arrow pointing downwards in the **”Actions”-**column as shown in the picture below.



| Type | Name | Play Store | Resolution | API | Target | CPU/ABI | Size on Disk | Actions |
|---|---|---|---|---|---|---|---|---|
| | Pixel 2 API 29 | ▶ | 1080 × 1920: 420dpi | 29 | Android 10.0 (Goo... | x86 | 10 GB | ▶ ✎ ▼ |
| | Pixel C API 30 | ▶ | 2560 × 1800: xhdpi | 30 | Android 11.0 (Goo... | x86 | 11 GB | ▶ ✎ ▼ |

### S7.

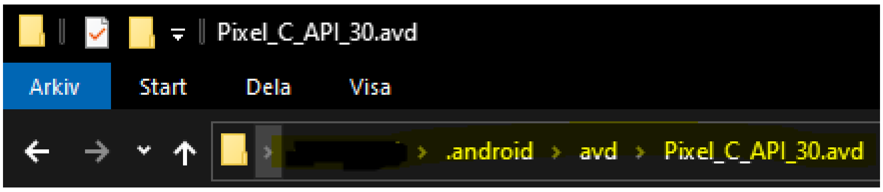
Select ”**View details**” in the dropdown list, then copy the file shown next to ”**Path**:”



Name: Pixel_C_API_30
CPU/ABI: Google APIs Intel Atom (x86)
Path: C:\Users\alans\.android\avd\Pixel_C_API_30.avd
Target: google_apis [Google APIs] (API level 30)
Skin: pixel_c
SD Card: 512M
fastboot.chosenSnapshotFile:

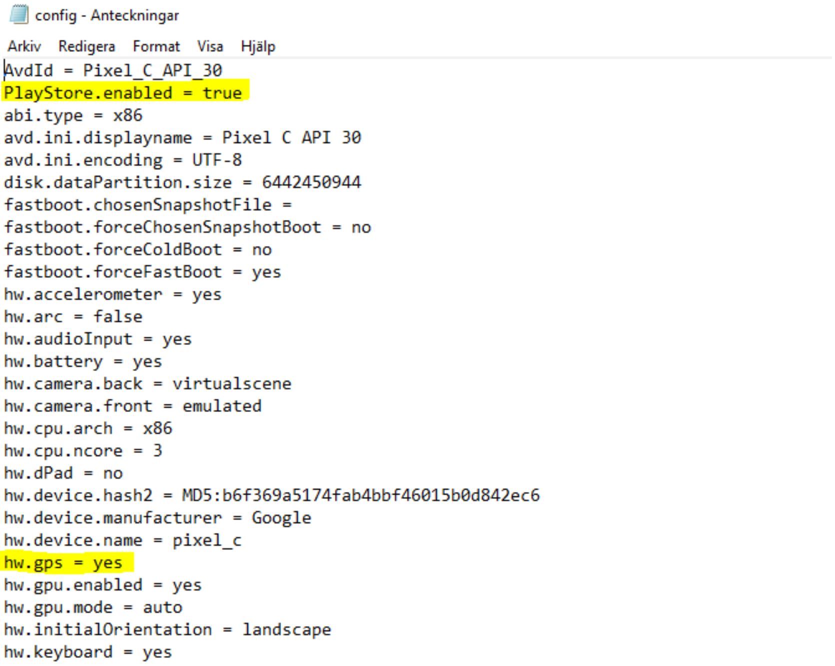
### S8.

Open your file explorer and paste the filepath in the search bar



### S9.

Open the file called ”**config**” and set **PlayStore.enabled** to true and **hw.gps** to yes



```
AvdId = Pixel_C_API_30
PlayStore.enabled = true
abi.type = x86
avd.ini.displayname = Pixel C API 30
avd.ini.encoding = UTF-8
disk.dataPartition.size = 6442450944
fastboot.chosenSnapshotFile =
fastboot.forceChosenSnapshotBoot = no
fastboot.forceColdBoot = no
fastboot.forceFastBoot = yes
hw.accelerometer = yes
hw.arc = false
hw.audioInput = yes
hw.battery = yes
hw.camera.back = virtualscene
hw.camera.front = emulated
hw.cpu.arch = x86
hw.cpu.ncore = 3
hw.dPad = no
hw.device.hash2 = MD5:b6f369a5174fab4bbf46015b0d842ec6
hw.device.manufacturer = Google
hw.device.name = pixel_c
hw.gps = yes
hw.gpu.enabled = yes
hw.gpu.mode = auto
hw.initialOrientation = landscape
hw.keyboard = yes
```

**S10.** (For using a real device)

1. Connect your device to your development machine with a USB cable. If you developed on Windows, you might need to install the appropriate USB driver for your device.
2. Perform the following steps to enable **USB debugging** in the **Developer options** window:

   a. Open the **Settings** app.

   b. If your device uses Android v8.0 or higher, select **System**. Otherwise, proceed to the next step.

   c. Scroll to the bottom and select **About phone**.

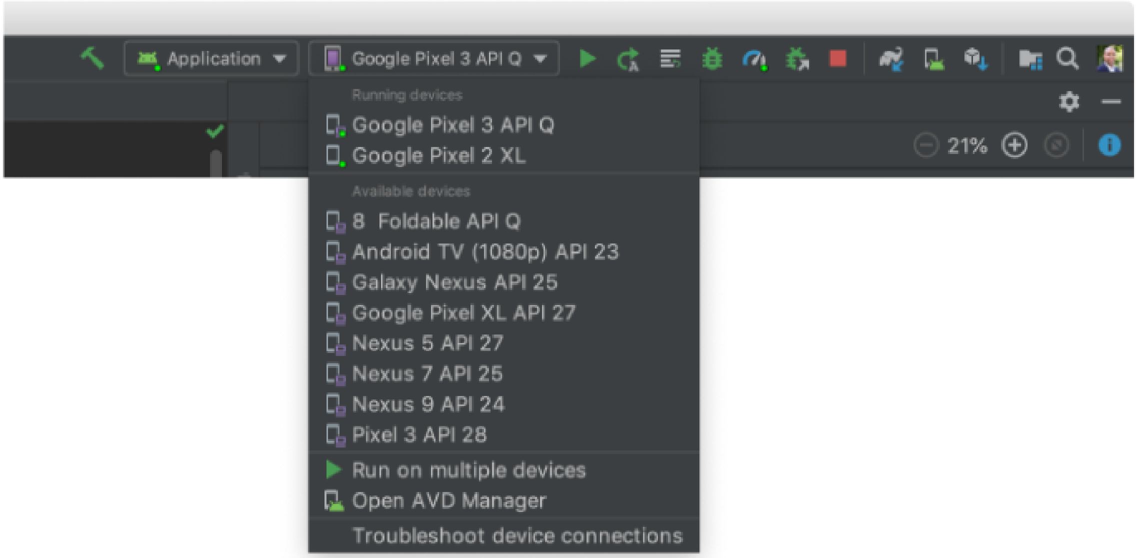
   d. Scroll to the bottom and tap **Build number** seven times.

   e. Return to the previous screen, scroll to the bottom, and tap **Developer options**.

   f. In the **Developer options** window, scroll down to find and enable **USB debugging**.

Run the app on your device as follows:

1. In Android Studio, select your app from the run/debug configurations drop-down menu in the toolbar.
2. In the toolbar, select the device that you want to run your app on from the target device drop-down menu.



1. Click **Run** ▶.

*Everything should now be set up in order to get into the testing part.*

# Testing

These are the requirements that will be tested:

- The system should be able to display the location of a product/vehicle(R1).
- Live updates of the users location (R2)
- Products should be identified by an unique ID (R3).
- The system should provide operators with the shortest path to a vehicle or the place to park the vehicle. (R4)
- The system should provide a description of the products/vehicles containing:color, registration number, etc. (R5)
- The system should display a To-do list of tasks for the Operator and Manager (R6).
- Only the manager should be able to add tasks to the task-list (R7).
- Only the admin should be able to create and delete accounts for operators (R8).
- All users of the system should be able to search for a product/vehicle(R9).

**T1.** If the setup was done correctly you should be presented with this screen after pressing the **”Run”**-button in **Android Studio**.

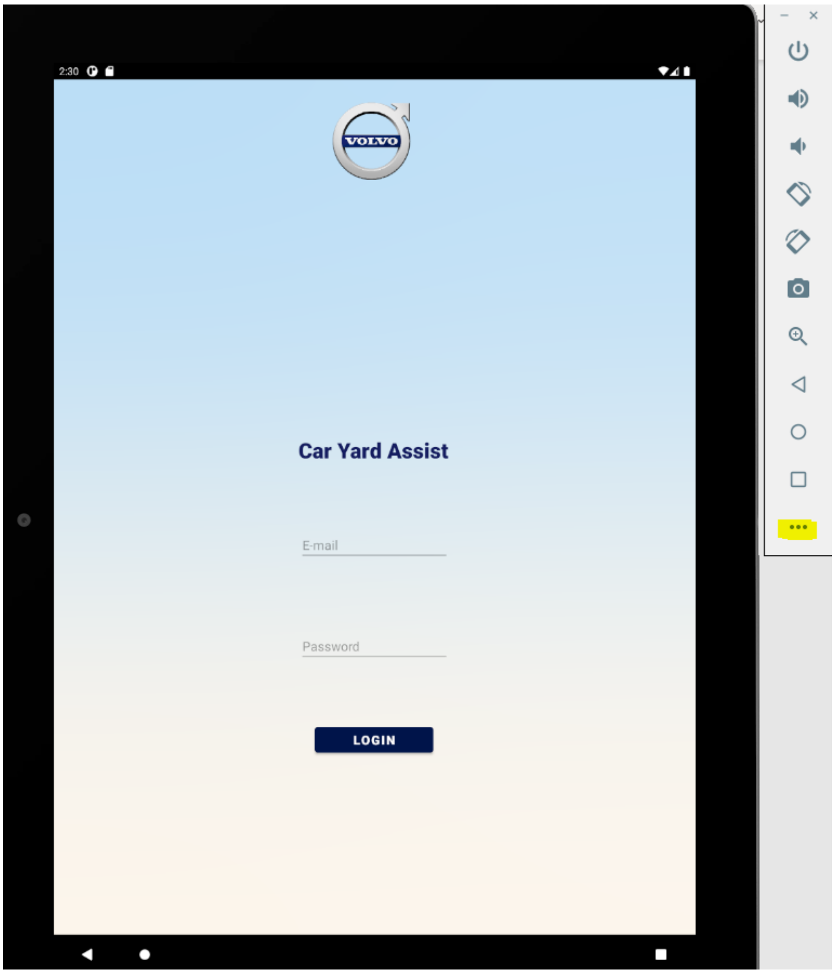
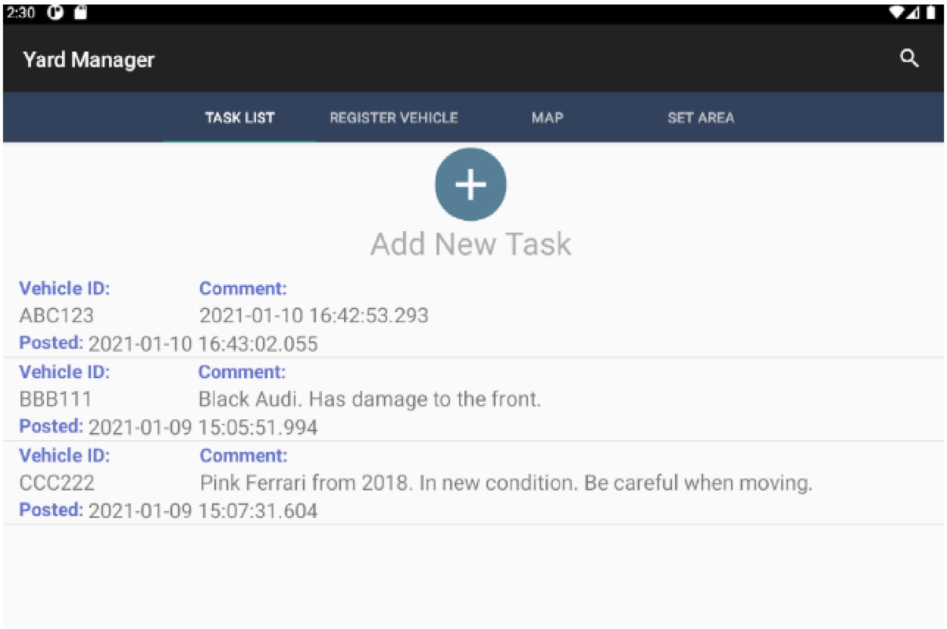


*Figure 1. The login screen of the application*

**T1.** Login as a manager, e-mail**: testmanager@gmail.com** and password**: 123456**. If logged in correctly, you should be presented with the following screen (the tasks displayed might not be the same):



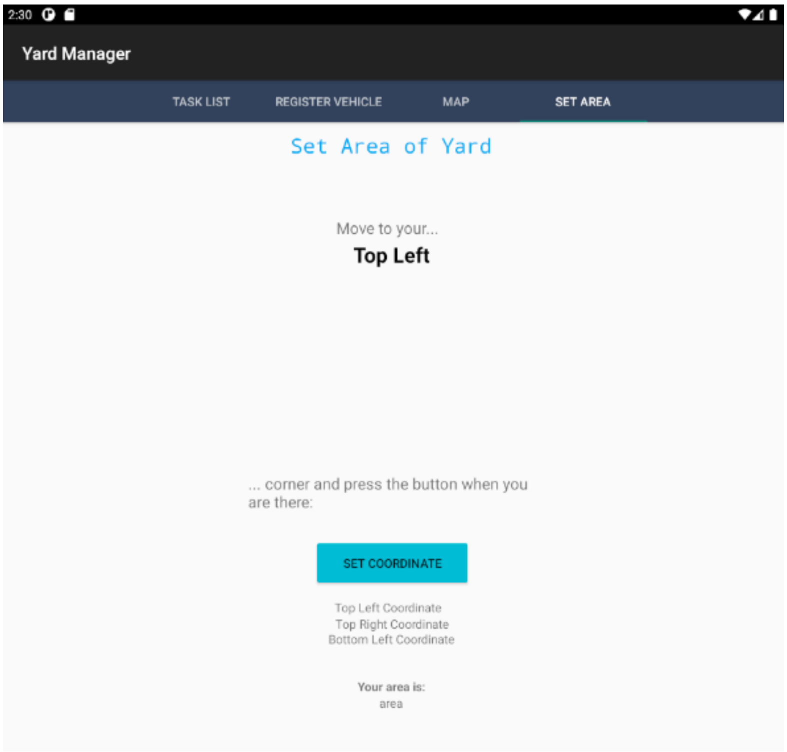
**Vehicle ID:** ABC123
**Comment:** 2021-01-10 16:42:53.293
**Posted:** 2021-01-10 16:43:02.055

**Vehicle ID:** BBB111
**Comment:** Black Audi. Has damage to the front.
**Posted:** 2021-01-09 15:05:51.994

**Vehicle ID:** CCC222
**Comment:** Pink Ferrari from 2018. In new condition. Be careful when moving.
**Posted:** 2021-01-09 15:07:31.604

*Figure 2. The task-list screen when logged in as a manager*

**T2.** Press ”Set Area”, you should now be presented with the following screen



Set Area of Yard

Move to your...
**Top Left**

... corner and press the button when you are there:

Top Left Coordinate
Top Right Coordinate
Bottom Left Coordinate

**Your area is:**
area

*Figure 3. Screen for marking the area of a yard*

**T3.**

***For testing purposes we recommend that you mark an area about as big two football fields.***

**If you are using a real device**

Position yourself and the device in the position where you want the **top left corner** of your yard to be and then press the ”Set Coordinate”-button.

Do the same for the **top right corner** and then the **bottom left corner**, the application will calculate the **bottom right corner**.

Now your rectangular shaped yard should be saved to our database.

**If you are using an emulator**

Press the three dots(marked with yellow in figure 1) next to the emulator to open the extended controls, then select **”Location”** from the menu on the left.

Choose a location to where you want your **top left corner** of the yard to be and press **”Set Location”** in the bottom right corner. Wait 10 seconds and then press **”Set Coordinate”**-button in the application.

Change your location again through the extended controls to where you want the **bottom right corner** of the yard to be, wait 10 seconds and then press **”Set Coordinate”**.

Do the same as above for the **bottom left corner**, the application will calculate the **bottom right corner**.

Now your rectangular shaped yard should be saved to our database.

**T4.** Now press **”Map”**

**If you are using a real device**

Move around inside the area that you marked, your location should be displayed by a green square.

**If you are using an emulator**

Change your location through the extended controls to any place within the area that you marked, your location should be displayed by a green square. Big changes of your location might take a few seconds to calculate.

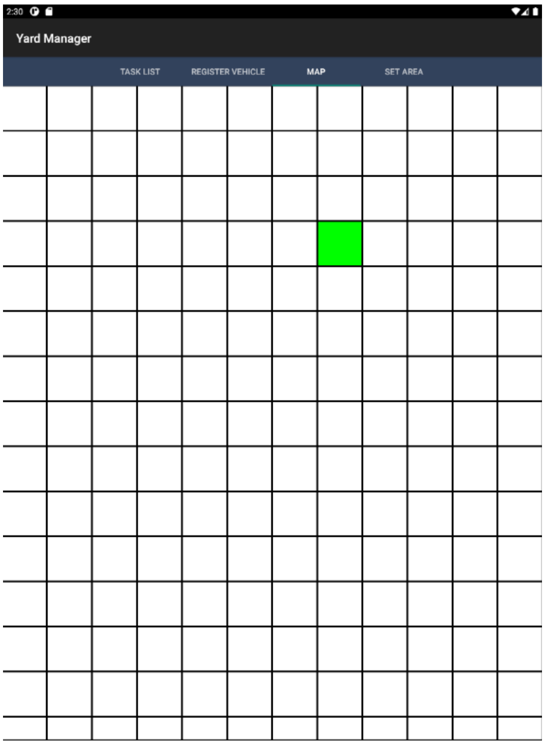
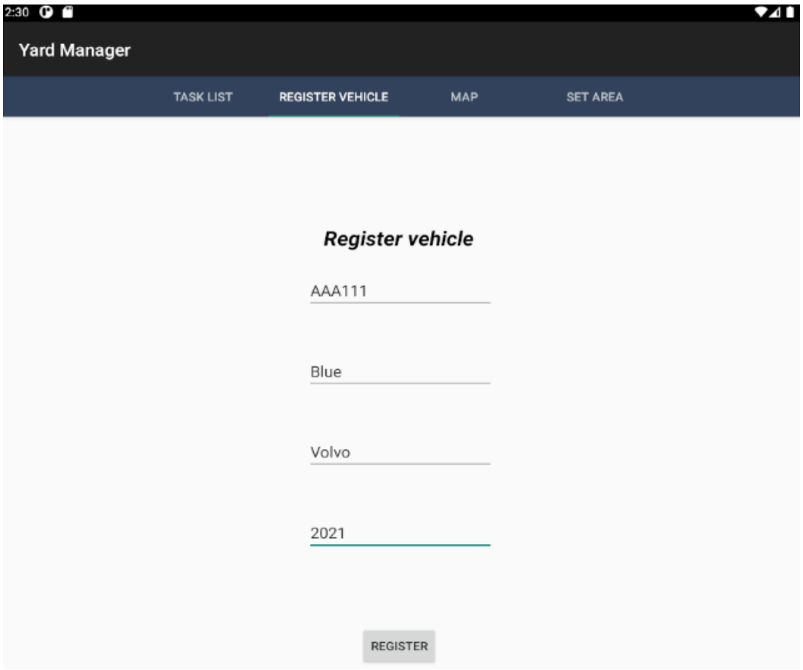


*Figure 4. Live update of the users location*

**T5.** Now press **”Register Vehicle”**. Enter any registration number, e.g IJK123, color, model and year, then proceed by clicking the **”Register”**- button. The vehicle should be placed in the square that you are currently inside. When you move out of from that that cell(e.g when you move approximately 3m in either direction) the cell should be red, indicating that it is occupied by a vehicle. We suggest adding approximately 10 vehicles to populate the map properly.



***Register vehicle***

AAA111

Blue

Volvo

2021

*Figure 5. Register a vehicle, it will be placed on the current location of the users device*

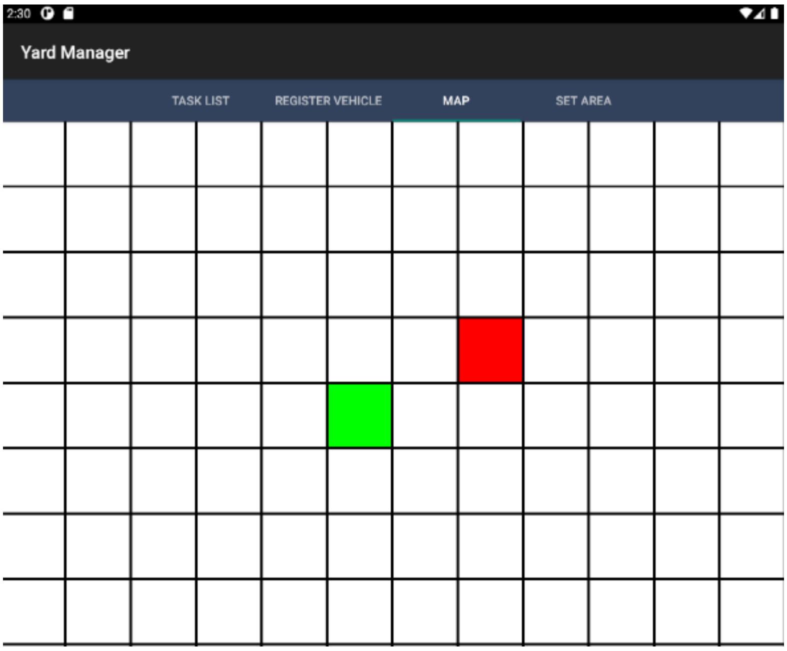


*Figure 6. When the user moves from the cell that the vehicle was registerd in, the ”old” cell will be red indicating that it is occupied by a vehicle*

**T6.** Now press **”Task list”** to go to the task list-screen, then press **”Add New Task”.** You should be taken to to the screen below. Enter the same Vehicle ID as you did in when registering the vehicle and then press the **”Mark goal via map”**-button.

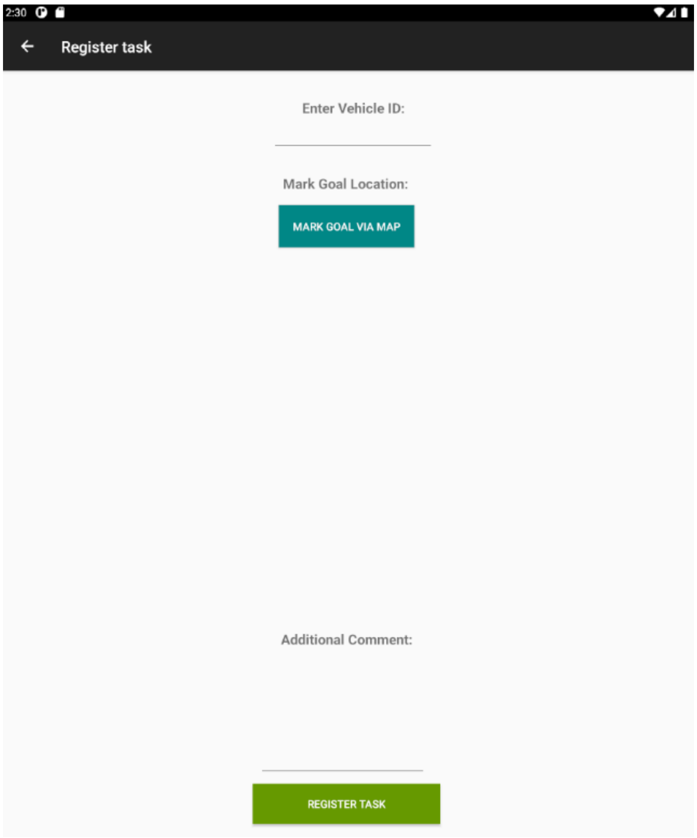


*Figure 7. Register task screen*

**T7**. The green square in the picture below is your location and the grey square is the square that has been chosen as the destination where the vehicle should be parked. Click any square in the grid that has been presented to you, then press the **”Set coordinates”-**button at the bottom of the screen.

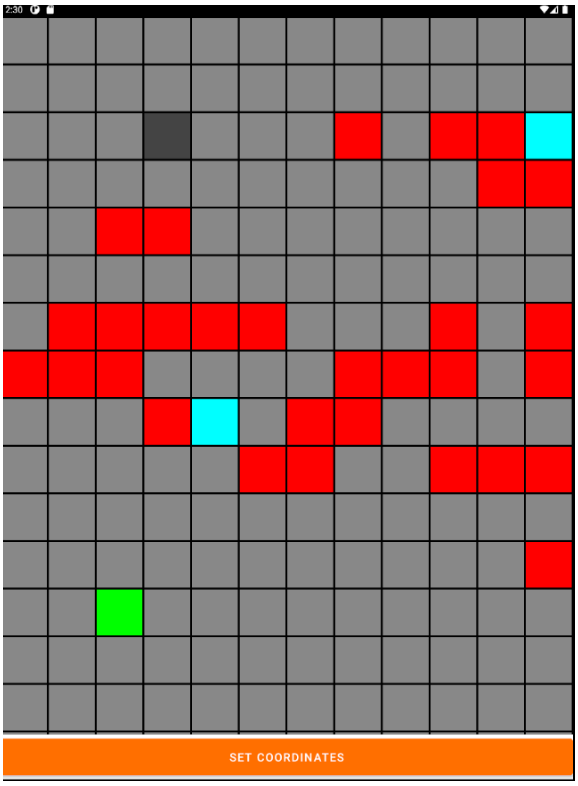
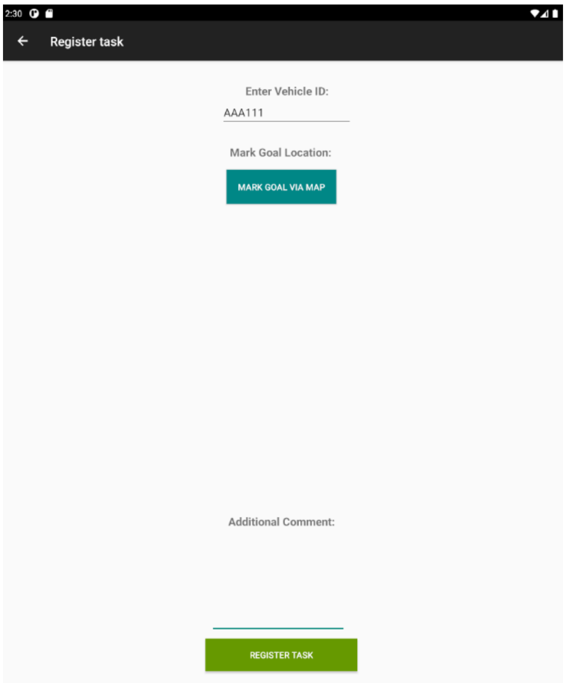


*Figure 8. The screen for marking where a vehicle should be parked*

**T8.** You will now be taken back to the **”register task”**-screen where you will have to option to add a comment to the task. Press the **”register task”** –button to complete the registration and then press the **arrow in the topleft** screen to go back to the **”task list”-**screen.



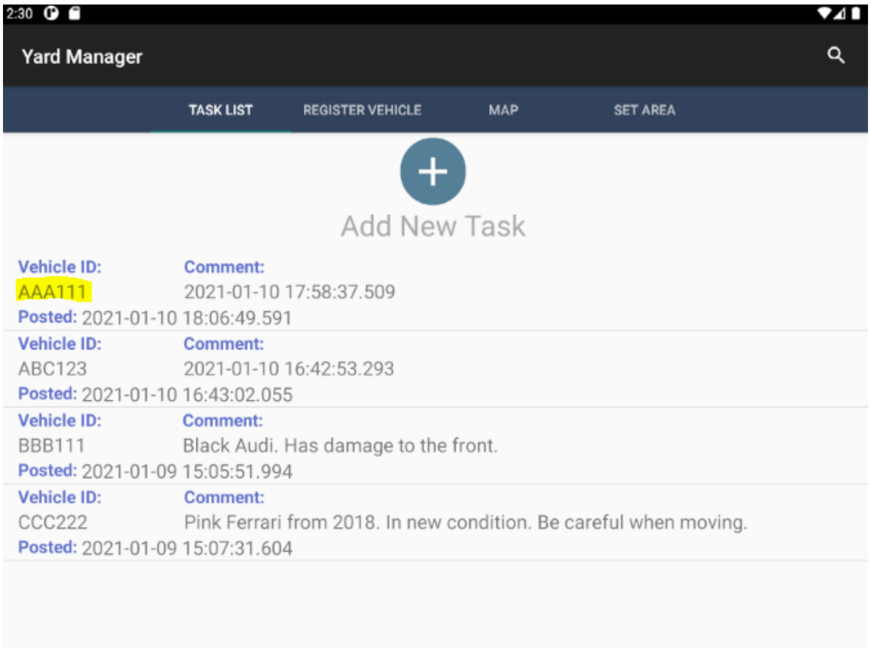
Enter Vehicle ID:
AAA111

Mark Goal Location:

Additional Comment:

*Figure 9. The register task screen*

**T9**. Your task should now be added and displayed in the task list. **Press the task** and then press **”Start the task”** in the popup window that has been presented to you, this should provide you with the shortest path to the vehicle from your current location.



Vehicle ID: AAA111 Comment: 2021-01-10 17:58:37.509
Posted: 2021-01-10 18:06:49.591

Vehicle ID: ABC123 Comment: 2021-01-10 16:42:53.293
Posted: 2021-01-10 16:43:02.055

Vehicle ID: BBB111 Comment: Black Audi. Has damage to the front.
Posted: 2021-01-09 15:05:51.994

Vehicle ID: CCC222 Comment: Pink Ferrari from 2018. In new condition. Be careful when moving.
Posted: 2021-01-09 15:07:31.604

*Figure 10. The task list*

**The following steps marked with * denotes that the instructions may be difficult to complete when using an emulator. We recommend these to be done with a real device.*

***T10**. Navigate yourself to the vehicle with the help of the navigation shown in the application, this should be the shortest path to the vehicle.

***T11**. When you are in the same cell as the vehicle, press the ”Pick up vehicle”-button. The application should now provide you with the shortest path to the cell where the vehicle should be parked.

***T12**. When you have navigated yourself to the cell where the vehicle should be parked, press the ”Drop off vehicle”-button

**The steps T1-T12 should now have verified that the requirements R1-R6 have been met.*

**T13**. Press the **”Run”**-button in **Android Studio** to restart the application, the same as was done in **Step S4.4.**

**T14**. The application should now restart and present you with the login screen again. Login as an **admin** this time, email**: testadmin@gmail.com** and password:**123456**.

**T15**. You should now be presented with the ”Create Account”-screen.

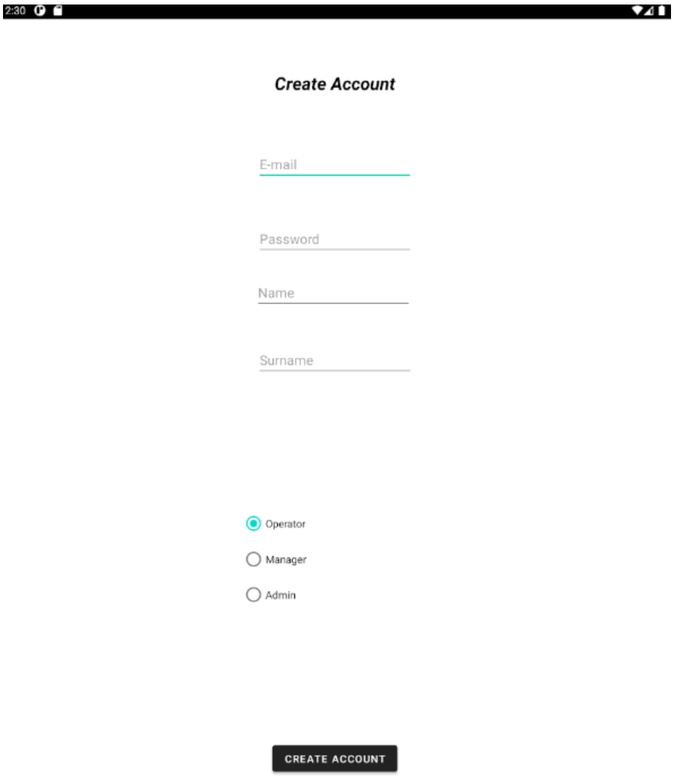


*Figure x. The screen for creating accounts, only admins can acess this page*

Fill in the textfields with any values:

*Email: ____@gmail.com*

*Password: Test*

*Name: Example*

*Surname: Example*

**T16**. Select **Operator** and press the **”Create Account”**- button. Then repeat **Step T8(restart the application)**.

**T17**. You should now login with the account that you created in the previous steps. You now have some of the functionality that the Manager has, this can be verified by repeating some of the earlier steps. When signed in as an Operator you cannot add new tasks to the task list and you can not set the area of the yard.

The steps **T13-T17** should verify that requirement **R7** has been met, half of **R8**(the admin can not delete accounts) has been met and **R9** has not been met.