Maintenance Document

Version	:	1.1
Date	:	16/10/2018
Team Name	:	Magnito (TP072)
Team Members	:	Ajmal Salim Asif Khan Clover Cheng Shen Yang

Version Control

Date	Version	Owner	Comments
15/10/18	V 1.0	Magnito	Initial Draft
16/10/18	V 1.1	Magnito	Final Draft

Table of Contents

Overview

The maintenance document describes maintenance personnel with the detailed information necessary to maintain the system effectively. Our product is an **Android app** named **Aways**, this document includes system application, system organization, security, equipment environment, computer hardware, support software, database characteristics, error conditions, testing information, the roles and responsibilities of program modules.

In addition to the items identified for inclusion in the maintenance document, additional information may be provided to improve the maintenance and modification of the system.

1.Introduction

This maintenance document offers the detailed information necessary for maintenance personnel to maintain the **Aways** app effectively.

1.1 Purpose

The purpose of this manual is to describe important information and guidelines that will help any future personnel to understand how to maintain our app.

1.2 Target Audience

The target audience of this maintenance manual is the programming developer's team who take the responsibility to maintain or make some changes for this app in the future.

2.System Description

2.1 System Application

Aways provides a simple way to get the users to the car parking spots and bike share stations and by they would be able to get directions through a quick click. By the built-in auto-rerouting system, there will be no other concerns during the driving. Live notifications, nearby disruption, parking timer and daily notifications are all we are making to let Aways become your powerful backup.

The detailed functions Aways provides are listed as below:

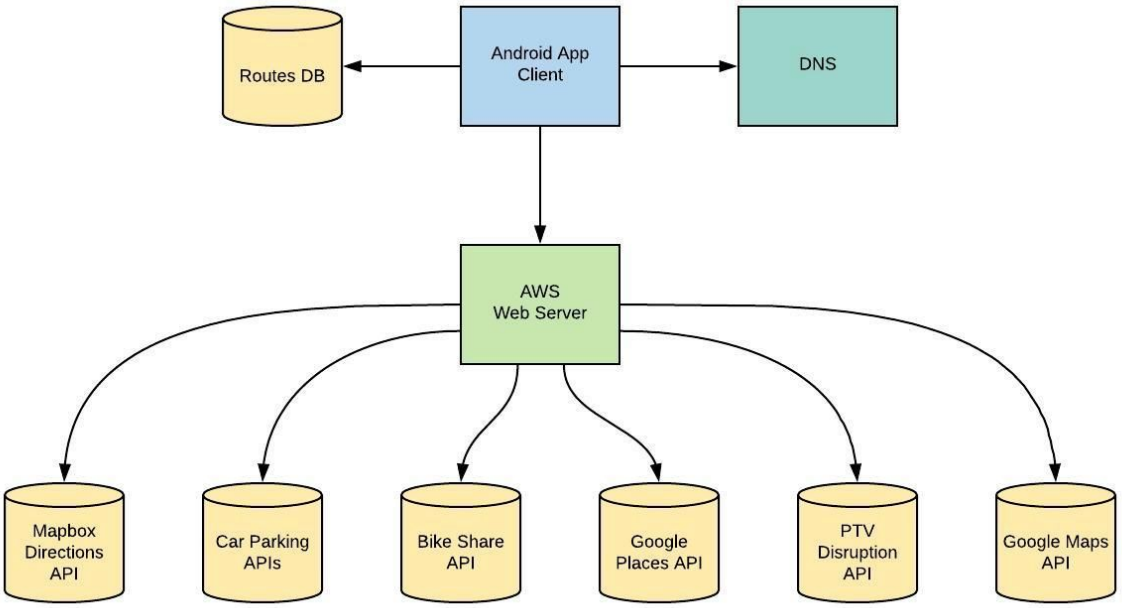
- Finding nearby and destination car parking spots, bike share stations and different types of disruptions
- Getting directions to the car parking spots and the bike share stations
- Auto-reroute the users when the current car parking spot becomes unavailable
- Store user favourite routes
- Daily notifications of the routes saved in the system
- Live train lines notifications
- Parking expiry timer
- Multi-language support
- Search Radius

Aways is not just limited to providing a solution covering the overall aspects of the transport disruptions, but to create an efficient and comforting travel experience for the commuters to the CBD
for work.

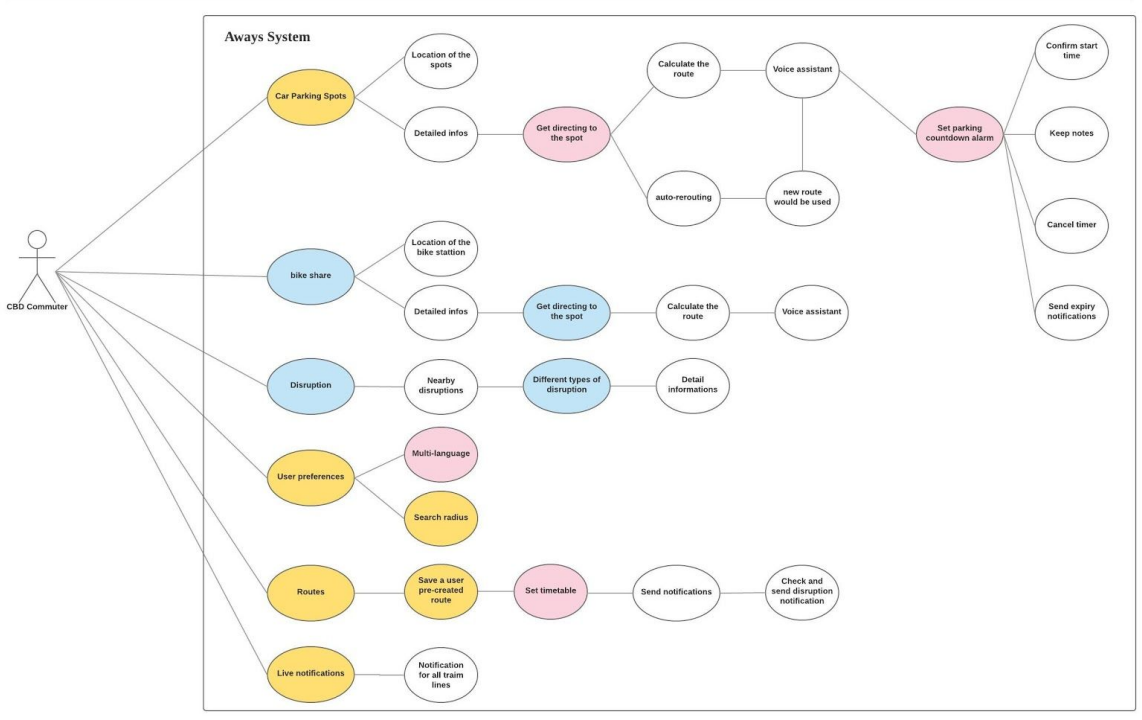
2.2 System Organisation

The system primarily consists of an Android app and a web server. The web server fetches the required data from the source APIs and delivers it to the app through a REST API. The overall system design is as follows:

2.2.1 System Architecture Diagram



2.2.2 Use Case Diagram



2.3 Security

2.3.1 Security Standards

Name	Details	Mitigation	Status
Insufficient Transport Layer Protection	In the case of an insufficient transport layer, a hacker can gain access to the data and modify or steal it on her/his will. This results in frauds, identity threats etc.	Use SSL and TLS to encrypt the communication.	Done
Reverse Engineering	Intruders can reverse engineer the app executable (.apk) and decompile classes to gain knowledge to source code or even sensitive details like API tokens.	Use ProGuard to obfuscate the code and make the process extremely harder	Done
Open API	If the API is not secured with app tokens, any public users can get access to the API and misuse it	Secure the API with a middleware and use JWT	Done
Client Side Injection	Execute malicious codes on the client side on the mobile device/API	Strict Header Validation	Done

2.3.2 Accounts and Credentials

Provider	Credentials	Comments
Gmail	Username : easygo.magnito@gmail.com Password : z^^DG\:8WrT]t~g)	Google Mail account used for all other account creation
BitBucket	Username : easygo.magnito@gmail.com Password : CEKbf3=<W%?;c/dn	Used for version controlling
AWS Web Server	Username : easygo.magnito@gmail.com Password : D?/H)'x*S3fffh^]	Used for Web Server
MariaDB	Username : root Password : _E!0I52qiQ2DrlziT3&t	MySQL Database Credentials
Melbourne Data	Username: easygo.magnito@gmail.com Password:	Used for accessing Open Datasets from Melbourne Data Portal

	CEKbf3=<W%?;c/dn	
NameCheap	Username: Clover111 Password : lucky1234	Used Namecheap service to register domain and SSL
Mapbox	Username: aways Password: Q3<`%\(WY85h`BRM	Used for Map requests and Navigation SDK
PTV Timetable API	Developer ID and Key should be requested from https://www.ptv.vic.gov.au/about-ptv/data-and-reports/datasets/ptv-timetable-api/	Used to get information about disruptions
Firebase Crashlytics	Username : easygo.magnito@gmail.com Password : z^^DG\:8WrT]t~g)	Used to get more information about runtime crashes
Google Maps	Username : easygo.magnito@gmail.com Password : z^^DG\:8WrT]t~g)	Used to show Google Maps

3. Support Environment

3.1 Equipment Environment

The Android app is developed using native Java targeting Android SDK version 27 (Oreo) with a minimum SDK support from SDK 19 (KitKat). The web server is using LAMP (Linux Apache MySQL PHP) stack on an AWS EC2 server. Lumen framework is used to develop the REST API.

3.1.1 Computer Hardware

For the proper development and maintenance of the system, the minimum hardware requirements for the system is as follows:

Hardware Requirements	**Developer Machine:** ***Windows*** Microsoft® Windows® 7/8/10 (32- or 64-bit) 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system

	image) 1280 x 800 minimum screen resolution ***Mac*** Mac® OS X® 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra) 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution ***Linux*** GNOME or KDE desktop Tested on Ubuntu® 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications) 64-bit distribution capable of running 32-bit applications GNU C Library (glibc) 2.19 or later 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution **Minimum Web Server Requirements:** 1 GB Memory, 1 vCPU, 25 GB SSD Disk, 100 GB Bandwidth Transfer, CentOS/Ubuntu/Amazon Linux
Operating Systems	**Developer Machine**: Windows, Linux, Mac **Web Server**: CentOS/Ubuntu/Amazon Linux

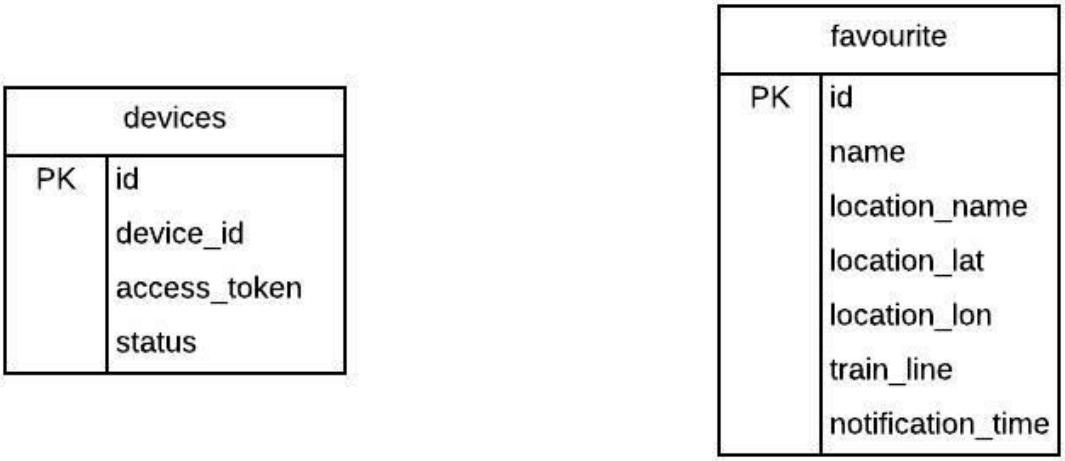
3.2 Support Software

The software requirements for the system application development and maintenance is shown below:

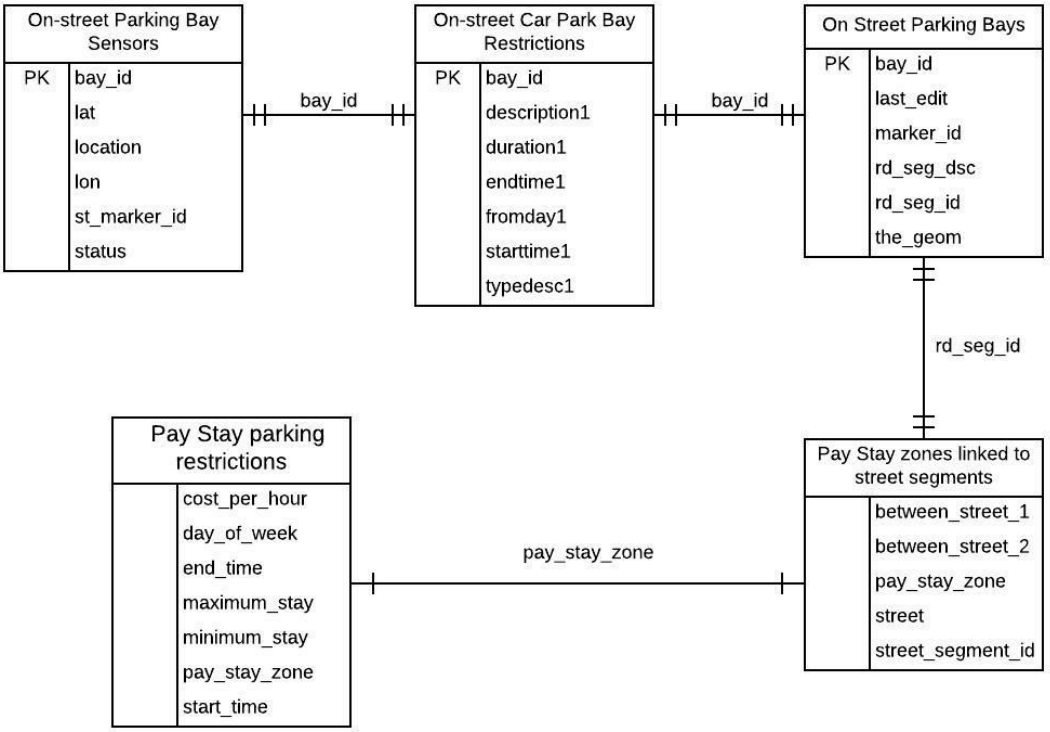
Android Client: Android Studio 3.1.4, Java 1.8, Git
Web Server: PHP, Apache, MySQL, Composer, Lumen

3.2.1 ER Models

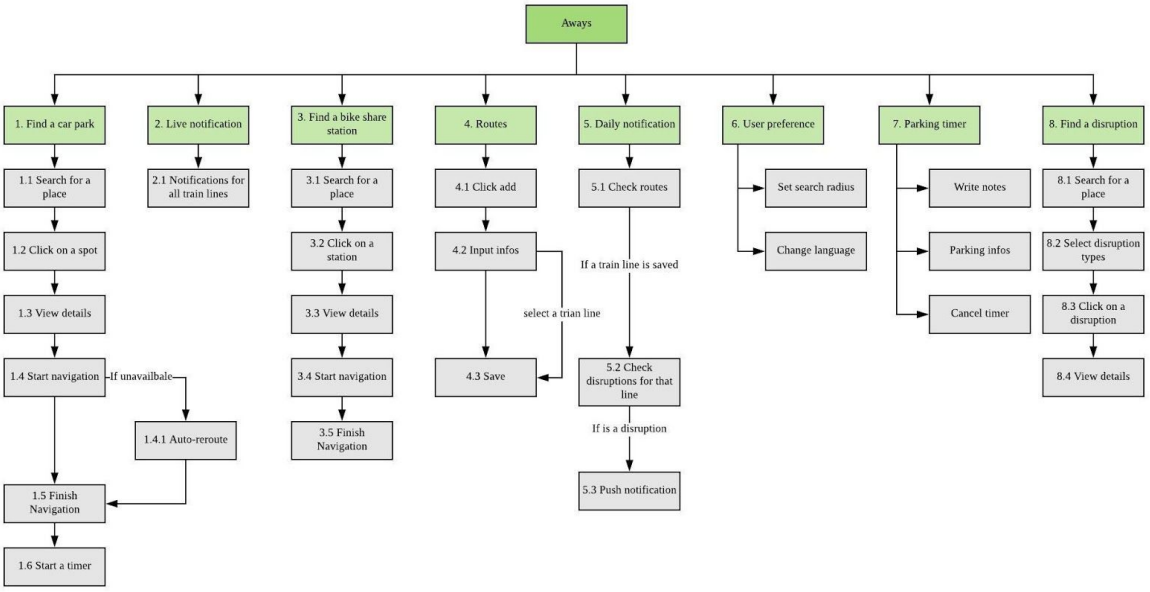
ER Diagram for App and Website



ER Model for Car Parking Dataset



3.2.2 Functional Decomposition



3.2.3 Aways API Documentation

Base URL : `https://aways.me/api/public/`			
Type	**Endpoint**	**Parameters**	**Sample Response**
POST	`auth`	`api-token` `device_id`	`"access_token": "$2y$10$WCoRvz4orjeSBf9LgU4lXuWOpWpG65dTuRTYHlFV4HGRMiJykWIia"`

POST	`parking`	`api-token`
		`lat`
		`lon`
		`search_radius`
			`"bay_id": "3515",`
			`"title": "Geographe Street",`
			`"description": "N/A",`
			`"price": "$0",`
			`"duration": "N/A",`
			`"duration_in_min": "N/A",`
			`"lat": "-37.80865880458556",`
			`"lon": "144.95837673606894"`

POST	`notification`	`api-token`
		`line`
			`"disruption_id": 142498,`
			`"title": "Frankston line stations",`
			`"description": "There will be full and partial temporary`

			long-term and short-term closures of station car parks and changes to pedestrian access in and around stations on the Frankston line, due to works as part of the Level Crossing Removal Project.", "disruption_status": "Current", "disruption_type": "Planned Closure", "published_on": "2018-07-10T01:34:18Z", "last_updated": "2018-10-04T01:45:54Z", "from_date": "2018-07-10T01:16:00Z", "to_date": "2018-12-30T16:00:00Z",
POST	bike_share	api-token lat lon	"id": "10", "title": "St Paul's Cathedral - Swanston St", "avail_bikes": "5", "empty_docks": "4", "terminal": "60007", "lat": 144.967409, "lon": -37.817189, "updated_at": "1 week ago"
POST	disruption	api-token lat lon route_type	"disruption_id": 142498, "title": "Frankston line stations", "description": "There will be full and partial temporary long-term and short-term closures of station car parks and changes to pedestrian access in and around stations on the Frankston line, due to works as part of the Level Crossing Removal Project.", "disruption_status": "Current", "disruption_type": "Planned Closure", "published_on": "2018-07-10T01:34:18Z", "last_updated": "2018-10-04T01:45:54Z", "from_date": "2018-07-10T01:16:00Z", "to_date":

			"2018-12-30T16:00:00Z",

3.3 Database Characteristic

3.3.1 MySQL

MySQL is an open source relational management system. Aways uses MySQL to keep record of the devices registered and their access tokens.

More details about MySQL can be found at https://www.mysql.com/

3.3.2 SQLite

SQLite is a simple relational database management system for Android and the only default option provided by the Android Architecture. Aways uses GreenDAO on top of it to manage seamless databases transactions whilst taking advantage of ORM features provided by GreenDAO.

More details about SQLite can be found at https://www.sqlite.org/index.html

3.3.3 Shared Preferences

An alternative way to store data permanently in Android devices is to use Shared Preferences. Shared Preferences can come in handy for simple data persistence and avoid the hassle of using SQLite. Aways uses Shared Preferences to store user token and other preferences like car parking search radius, language preferences etc.

More details can be found at https://developer.android.com/training/data-storage/shared-preferences

3.3.4 Datasets Used

Data Source(s) Details						
Name	**Description**	**Module**	**Type**	**URL**	**Frequency of Iteration System Updates**	**Frequency of Source Updates**
On-Street Parking Bay Sensors	The status of the parking spot can be found here, i.e. whether a car is present or not, the spatial	Car Parking	API	https://data.melbourne.vic.gov.au/Transport-Movement/On-street	2 mins	2 mins

	coordinate and the street maker id. All can be found from in-ground car parking bay sensors across the city.			-Parking-Bay-Sensors/vh2v-4nfs?_ga=2.208428310.525775592.1534694663-2077461 76.1532676435		
Google Places API	This allows users to select a place on an interactive map. The present place displays a list of places where the user's device was last located along with an indication of the relative likelihood for each place. As the user's type, it automatically fills in the name and/or address of a place. It displays high-quality images of a place and more.	Car Parking	API	https://developers.google.com/places/android-sdk/start	Real-time	Monthly
On-Street Car Park Bay Restrictions	This shows the restrictions that apply to each bay with a sensor, at various times of the week.	Car Parking	SODA API/ CSV/ JSON	https://data.melbourne.vic.gov.au/Transport-Movement/On-street-Car-Park-Bay-Restrictions/ntht-5rk7/data?_ga=2.214671951434373441.1534996554-20700	Daily	Daily

				30246.153413013 5		
On-Street Car Park Bays	This shows the location and shape of every bay, including the nearly 20,000 spots without sensors.	Car Parking	SODA API/ GeoJSON	https://data.melbourne.vic.gov.au/Transport-Movement/On-street-Parking-Bays/crvt-b4kt/data?_ga=2.162332700.1434373441.1534996554-2070030246.1534130135	Daily	Daily
On-street Car Parking Meters with Location	This provides Information about the parking meter model type, if the model accepts credit card and tap 'n go facilities.	Car Parking	SODA API/ CSV	https://data.melbourne.vic.gov.au/Assets-Infrastructure/On-street-Car-Parking-Meters-with-Location/vdsi-4gtj	Monthly	Monthly
Melbourne Bike Share stations, with the current number of free and used docks	Melbourne Bike Share is a joint RACV/Victorian Government bicycle hire scheme. This dataset contains the locations of all of the bike share pods, the number of bikes located at each pod and the number of empty slots at the pod.	Bike Share	SODA API/ CSV	https://data.melbourne.vic.gov.au/Transport-Movement/Melbourne-Bike-Share-stations-with-current-number-/tdvh-n9dv	Every 15 mins	Every 15 mins

MapBox API for directions	Navigation and Directions SDK	Car Parking and Bike Share	API	https://www.mapbox.com/navigation/	Real-time	Real-time
PTV Timetable API	The API returns scheduled timetable, route and stop data for all metropolitan and regional train, tram and bus services in Victoria, including Night Network.	Disruption	API	http://timetableapi.ptv.vic.gov.au/swagger/ui/index	Real-time	Real-time

3.4 Server Environment Setup

The following procedures help you install an Apache web server with PHP and MySQL support on your Amazon Linux instance (sometimes called a LAMP web server or LAMP stack).

> *For complete and full reference for this tutorial, please see:*
> *https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html*

This is an overview of how to set up the server and might require knowledge about Linux systems and servers to carry out all the necessary tasks.

To install and start the LAMP web server with the Amazon Linux AMI

Step 1: Connect to your instance

Step 2: Update all the software packages

```
[ec2-user ~]$ sudo yum update -y
```

Step 3: Install the Apache web server, MySQL, and PHP software packages.

```
[ec2-user ~]$ sudo yum install -y httpd24 php70 mysql56-server php70-mysqlnd
```

Step 4: Start the Apache web server

```
[ec2-user ~]$ sudo service httpd start
Starting httpd:                                            [  OK  ]
```

Step 5: Use the **chkconfig** command to configure the Apache web server to start at each system boot.

```
[ec2-user ~]$ sudo chkconfig httpd on
```

Step 6: Add a security rule to allow inbound HTTP (port 80) connections

a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

b. Choose **Instances** and select your instance.

c. Under **Security groups**, choose **view inbound rules**.

d. You should see the following list of rules in your default security group:

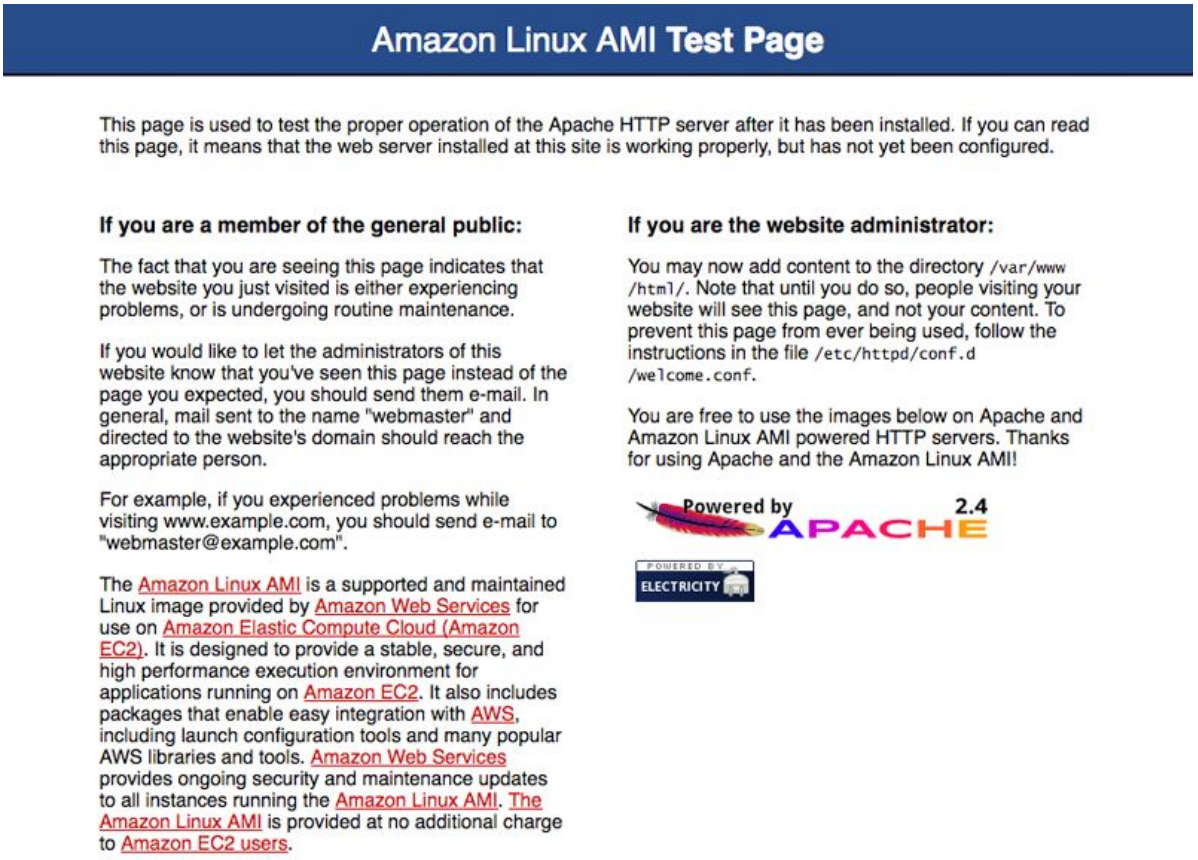
```
Security Groups associated with i-1234567890abcdef0
Ports     Protocol     Source     launch-wizard-N
22        tcp          0.0.0.0/0          ✔
```

Using the procedures in Adding Rules to a Security Group, add a new inbound security rule with the following values:

- **Type**: HTTP
- **Protocol**: TCP
- **Port Range**: 80
- **Source**: Custom

Step 7: Success

Test your web server. In a web browser, type the public DNS address (or the public IP address) of your instance. If there is no content in /var/www/html, you should see the Apache test page.

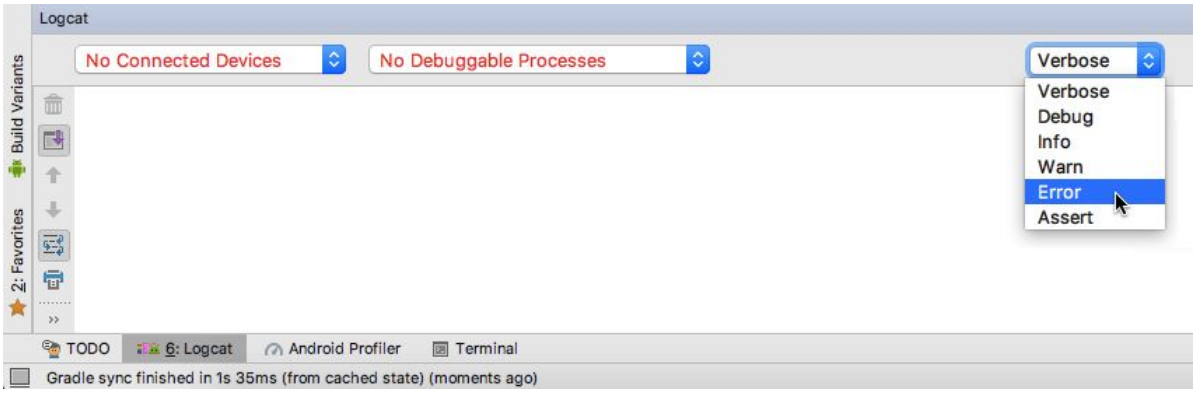


4. Error Conditions

4.1 Android Errors

Testing is a vital part of the app development process to identify all the potential bugs, errors and other issues that can hinder the performance. It is essential to get access to the stack trace in order to find the issue.

Android Studio by default ships with a debugger installed and one can make use of the Logcat to identify any errors. You can select the error stack trace by selecting the option as shown below:



The following are the main 5 common issues that one can encounter while developing or maintaining the app:

1) R.layout.main Cannot Be Found / Cannot Resolve Symbol R

This error is caused when Android Studio can't generate your R.java file correctly, and it can often crop up out of nowhere—one minute everything will be working fine, and the next minute every part of your project is failing to compile.

To fix the error:

Often, the most effective solution is the simplest: clean and rebuild your project. Select `Build > Clean Project` from the Android Studio toolbar, wait a few moments, and then build your project by selecting `Build > Rebuild Project.`

2) Please Choose a Valid JDK Directory

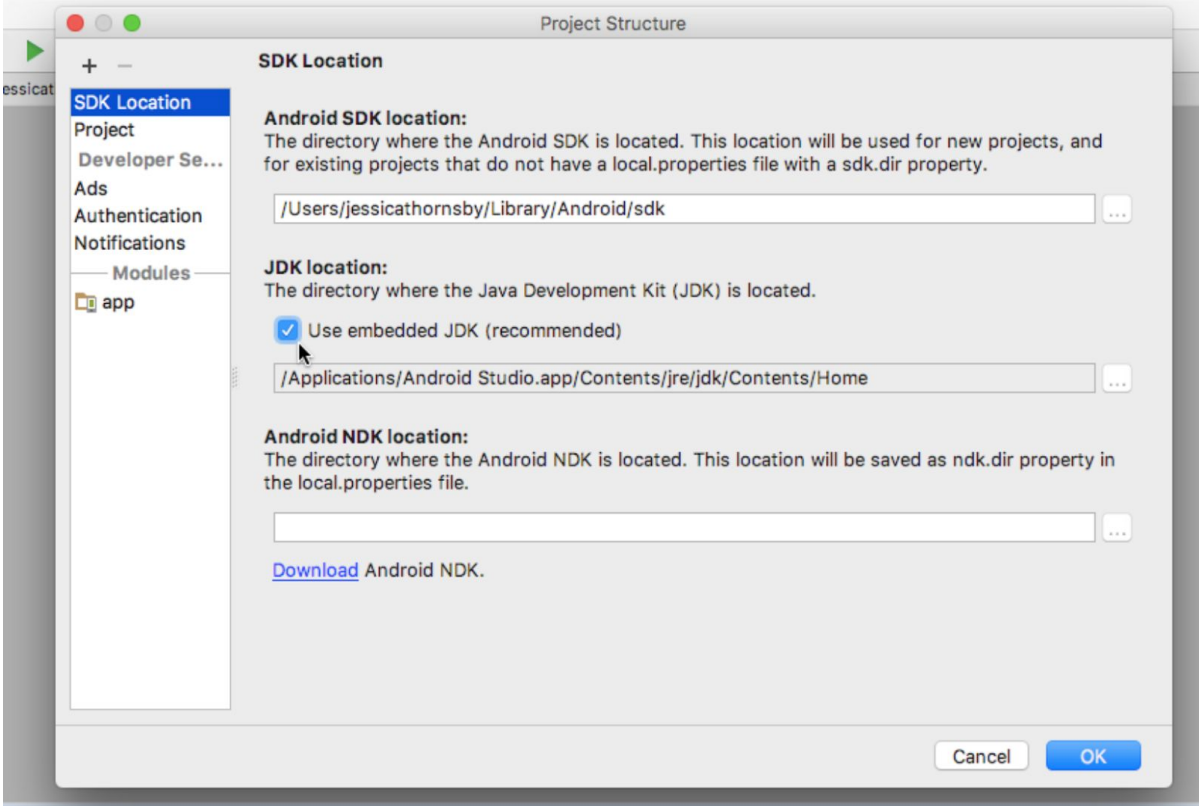
If you're getting a JDK error whenever you try to build your app, then it means Android Studio is struggling to find where the JDK is installed on your development machine.

To fix this error:

Select `File > Project structure…` from the Android Studio toolbar.

Select SDK Location from the left-hand menu.

Make sure the Use embedded JDK checkbox is selected.



3) Error Installing APK

Android Studio's ability to recognize a connected Android device is notoriously hit and miss. It is possible that you've attached your device to your development machine but is encountering an Error

installing APK message whenever you try to install your APK, or your device isn't even appearing in the Select Deployment Target window,

To fix the error:

Check USB debugging is enabled.
Check your smartphone or tablet screen.
Make sure you have the correct USB driver installed.
Check that your device meets your project's minimum SDK requirements.
Try restarting your adb (Android Debug Bridge) process.

4) ActivityNotFoundException

An ActivityNotFoundException occurs when a call to startActivity(Intent) or one of its variants fails because the Activity can't execute the given Intent.

To fix the error:

Firstly, if you encounter this error after moving an Activity class from one package to another, then it's possible that you've confused Android Studio and just need to clean and rebuild your project.
An ActivityNotFoundException can also be caused if an error in the target Activity is not loading correctly. To check whether this is occurring in your project, put your intent code inside a try-catch block.

5) NullPointerException

NullPointerException means that you've tried to use a reference that's pointing at a null value, as though it were referencing an object.

To fix the error:

Make sure all your objects are initialized before you attempt to use them, and always verify that a variable isn't null before you request a method or field from that object.

4.2 Aways API Errors

Aways uses REST API to communicate with the several third-party API data providers and with the Aways App. There are 5 main errors that can relate to the Aways API:

1) Using http:// instead of https://

Aways uses SSL/TLS encryption to secure the data transferred across the system. However, it is also possible to get the same API results without using the **https://** but it is not that the API will return the required output all the time as it can depend on the HTTP client we are using to parse the response.

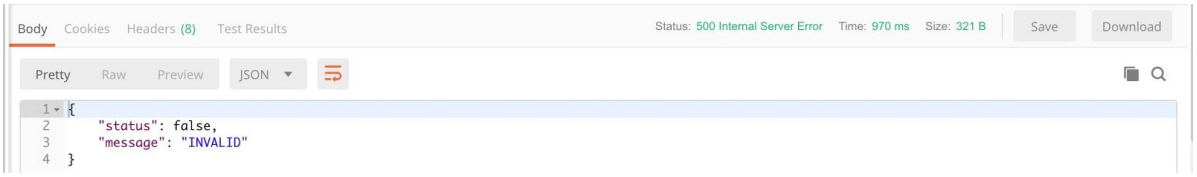
2) Unexpected error codes

Aways API can return unexpected status codes depending upon the error. HTTP defines over 40 standard status codes that can be used to convey the results of a client's request.

The status codes are divided into the five categories presented here:

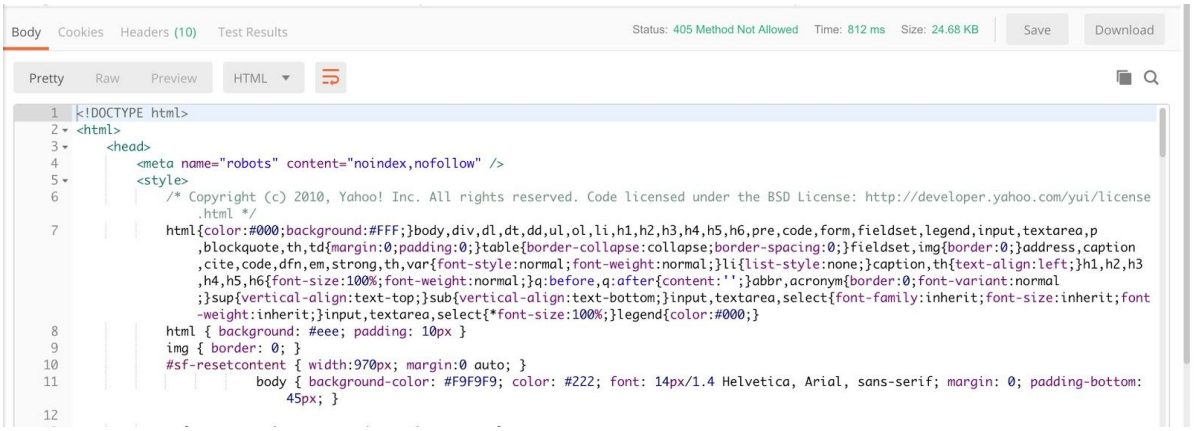
1xx: Informational - Communicates transfer protocol-level information
2xx: Success -Indicates that the client's request was accepted successfully.
3xx: Redirection - Indicates that the client must take some additional action in order to complete their request.
4xx: Client Error - This category of error status codes points the finger at clients.
5xx: Server Error - The server takes responsibility for these error status codes.

Aways API will return 200 for all the success API calls and 500 for any unexpected errors. For example, if the API requires some fields and the developer misses one or more of the fields, it can return in an invalid response.



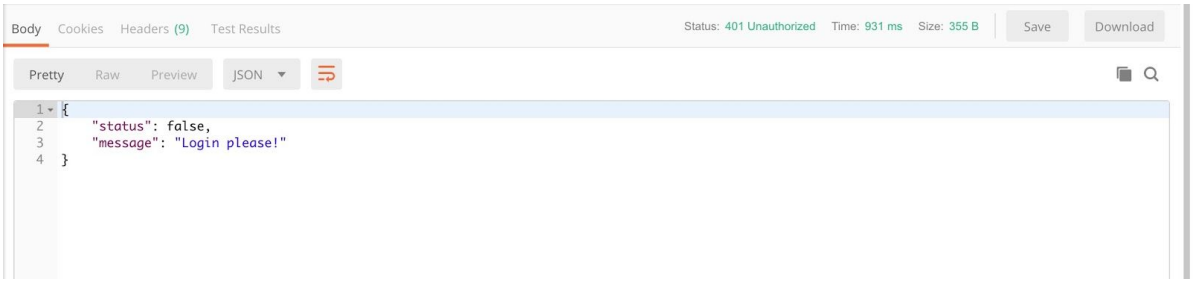
3) Using the wrong HTTP method

This is one of the major pitfalls developers can come across. Even though there are several standard HTTP methods with their own purposes, Aways API contains only POST methods. For some reasons, if one is trying to access the endpoint with any other HTTP method, then it can result in an error response. The error code for this response will be 405 and an example is shown below:



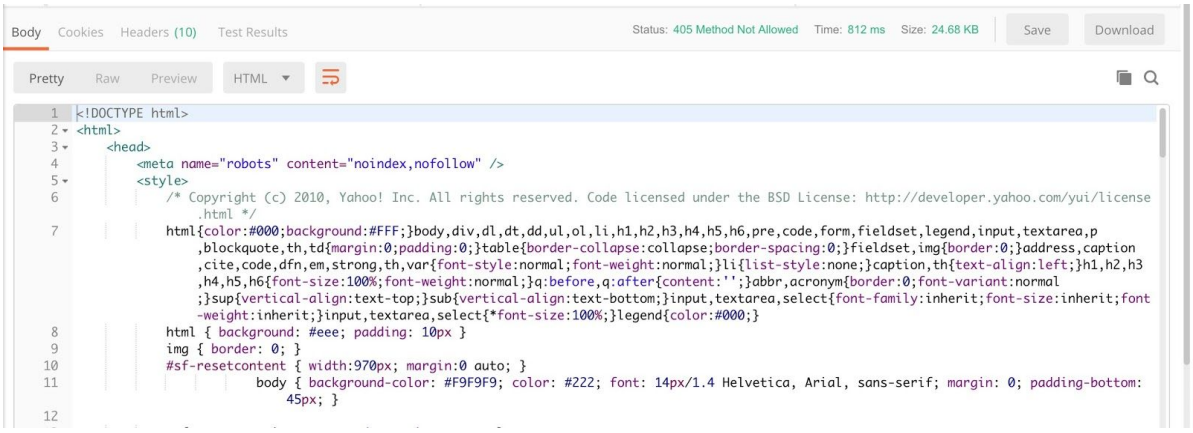
4) Sending invalid authorization credentials

APIs that implement JWT usually require the developer to include an **Authorization** header for each request. It's important that you're using the correct word. Aways use the word **api-token** instead of **Authorization** and can return a 401 error response if the authentication header name is wrong or the authentication token is invalid.



5) APIs returning invalid content type when there is an error

For any other unexpected issues, aways can return the error page in HTML itself. This happens when the error is not addressed or the developer makes some critical mistake while making some modifications. One example is when the developer uses invalid function names or makes a typo in a function call. A sample error page is shown below:



5.Testing information

There are several testing methods that include various aspects: Usability, Acceptance, Security

Test case description-

Test Case ID	Test Case Name	Test Steps	Expected Result	Actual Result	Status (Pass/Fail)
1	Open the app	Open the app	Display the map on the home page	Success, the app displays the map	**Pass**

2	Set Perences	#Open the drawer on the home page # Select Home Address # Enter Address #Select Work Address #Enter Address	The app should save the home and work addresses	Success on saving the preference	**Pass**
3	View Parking Spots	#Select the return arrow #Move the map around	Show parking spots in the CBD based on the user's preferences	Success in showing the parking spots	**Pass**
4	Select a parking spot	Select a parking spot on the screen	Show Details of the parking spot	Success showing the price and parking duration	**Pass**
5	Disruption Notification	Open the app and return to the home screen	A disruption notification appears at the top of the screen	Disruption notification received	**Pass**
6	Routes during disruption	#The user needs to start the app # The user needs to search for a place or click on the favorites and choose a certain parking spot #The user needs to select get direction	#The home screen should appear with multiple options # The user will get the available parking spots around QV # The user receives real-time route	#Home screen appears #Results turn back # Real-time route is displayed	**Pass**
7	Voice Assistance	The user needs to select get directions	Real-time voice assistance received	Voice resistance received	**Pass**

8	Parking Expiry Notification	#The user needs to finish his/her journey # The user clicks 'Yes' #Counting down finished	# The parking timer alert should pop up #Timer should start and parking detail page pops up # A new expiry notification would appear	#The parking time alert pops-up # The timer starts counting down in the notification bar and user can see the parking details #A new expiry notification appears	**Pass**
9	Favourites	#User opens the Routes on Home page and click the add button #User opens the parking page	# The favourite place will be stored and show in the page # User stored favourite places should appear on that page	#The place show in the list #The places appear in the list	**Pass**
10	Track Parking Spot	#User starts the timer after the journey completion #User clicks on the notification when it appears #User put notes in the notes area #User click stop timer	#Timer expiry reminder should show in the notification #Notification would not disappear and will open the parking details page #User can put note area and the note will keep showing during counting down #The timer notification should	#Timer expiry reminder shows in the notification and countdown starts #Notification remaining in the notification bar and parking details open #Whenever user open this page, the notes keep showing on the page #The timer notification disappeared	**Pass**

			disappear in the status bar		
11	Searching radius	#Open the Favorites on Routes page #Change prefer radius	#The app can only show the spots in a default radius #The app can only show the spots within '150' radius	As expected	**Pass**
12	Parking Auto-Rero uting	#The user needs to start the app # The user needs to search for a place or click on the favourites and choose a certain parking spot #The user needs to select get direction	#The home screen should appear with multiple options #The user will get the available parking spots around QV #The user receives real-time route	#Home Screen Appears #Results turn back #Real- time route is displayed	**Pass**
13	Language Selection	#The user should select Preferences #Select Change Language #Choose Chinese	#See the Preferences Screen #See a popup with option of selecting English and Chinese #App will switch to Chinese language	#See Preferences Screen #See a popup with option of selecting English and Chinese #App will switch to Chinese language	**Pass**
14	Bike Sharing	#The user selects the Bike Sharing option from Home #The User can search for Bike Share stations for specified address	#Bike Sharing screen opens and shows nearby bike share options #User sees the bike share options nearby Flinders Street	#Bike Sharing screen opens and shows nearby bike share options #User sees the bike share	**Pass**

				options nearby Flinders Street	
15	Navigation to bike share	#The user selects the Bike Sharing option from Home #The User can search for Bike Share stations for specified address #User selects one of the result #Click on "Get Directions	#Bike Sharing screen opens and shows nearby bike share options #User sees the bike share options nearby Flinders Street #The details page will be shown with button "Get Directions" #Walking navigation should start automatically to the bike share location	#Bike Sharing screen opens and shows nearby bike share options #User sees the bike share options nearby Flinders Street #The details page will be shown with button "Get Directions" #Walking navigation should start automatically to the bike share location	**Pass**
16	Disruption	#User selects the "Disruptions" option from Home #User searches for an address and hit search	#User sees the Disruption Screen with nearby disruptions #User will see potential PTV disruptions near the specified address	#User sees the Disruption Screen with nearby disruptions #User will see potential PTV disruptions near the specified address	**Pass**

6. Documented testing carried out

6.1 Integrity(Acceptance Testing):

Activity (Commuter)	Pass/ Fail	Comments
Epic: I can receive transportation disruption notifications, and also check for disruptions in any place in Victoria		
User Story 1: **As a commuter** to the CBD for work/appointment, **I want to** receive notifications of any public transport disruptions, which may affect my way to the CBD, **so that** I could plan my journey in advance.	**Pass**	
Successful opening of the app		
Select live notifications		
View Disruption information		
Scroll the live notification		
User Story 2: **As a commuter** to the CBD, **I want to** receive daily notifications of the train line disruptions, which may affect my way to the CBD, **so that** I could plan my journey in advance	**Pass**	The time should be in the format xx:xx instead of xx:x
Successful opening of the app		
View Routes		
Add Routes		
Name the Route		
Search for the Route		
Select a train line		
Pick a time		
Save the route		
Receive Notification at the selected time		
EPIC: I can set my preference, view the parking spots and navigate to the parking spot with voice assistance.		

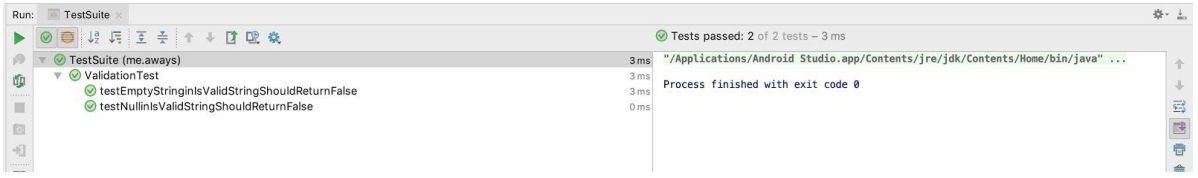
User Story 1: As a senior commuter, I want to customize my journey to the CBD by setting my preference in the app, **so that** I could save time and find a satisfactory parking spot.	**Pass**	
View Routes		
Add Routes		
Name the Route		
Search for the Route		
Select a train line		
Pick a time		
Save the route		
View Preferences		
Set Radius		
View Parking		
Select previously saved name		
View Parking Information		
User Story 2: As a car owner, I want to receive navigation through voice assistance once I select a parking spot, **so that** I could drive to the selected parking spot	**Pass**	
View Parking		
View Saved Route Or Search For Parking		
Select Navigation		
Voice assistance received for navigation		
User Story 2: **As a car owner** who is driving, and have found that the initially displayed parking spot is exhausted, **I want** to be auto rerouted to the next possible parking area directly, **so that** I do not need to perform extra actions on my phone.	**Pass**	

View Parking		
Voice update once the parking spot becomes unavailable		
Auto-rerouted to new parking spot		
User Story 3: **As a PTV commuter, I want** to search for any potential public transport disruptions all over Victoria, **so that** I can plan my itinerary better.	**Pass**	
View Disruptions		
Search Disruptions		
View/Scroll Information		
User Story 4 : **As a commuter** who is a Mandarin speaker, **I want** to use the app in my language, **so that** I can use it much more comfortably.	**Pass**	
View Languages		
Change Languages		
Select Mandarin		
View Information in Mandarin		
User Story 5 : **As a commuter,** who works in the CBD and has catch-ups in the CBD, **I want** to know about the nearby bike stations and the availability of the bikes, **so that** I can go catch-up with friends/family	**Pass**	
View Bike Sharing		
Search Bike Sharing		
View/ Scroll Information		
User Story 6 : **As a commuter,** who works in the CBD and commutes by train, I want to receive navigations to the bike stations, **so that** I can take a bike to catch-up with friends/family at a restaurant/bar near me	**Pass**	

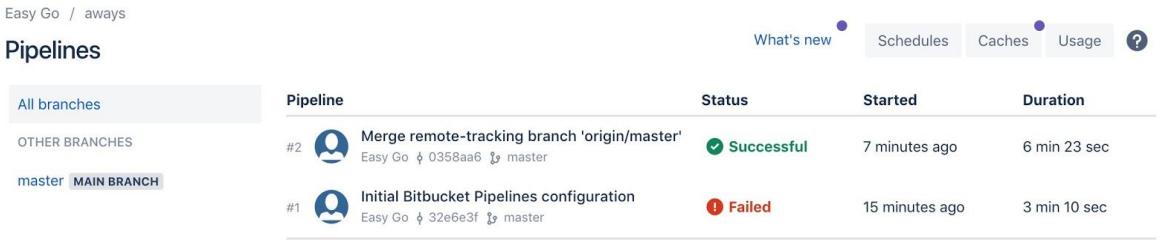
View Bike Sharing		
Search Bike Sharing		
View/ Scroll Location		
Select Location		
Start Navigation		

6.2 Unit Testing and Regression Testing

Unit Testing is essential in debugging process and it can help a lot in conjunction with Continuous Integration Tools. Aways uses unit testing and Bitbucket Pipeline for continuous integration. Developers will be able to run all the test classes using the Test Suite and verify if all the test cases are passed or not. A sample image of unit testing is shown below:



Whenever a new commit is pushed to the VCS, the Bitbucket Pipeline will automatically invoke the Regression Testing procedures and the results can be visible in the Bitcucket Repository under **Pipelines** options. A sample image is shown below:



6.3 Stress Testing

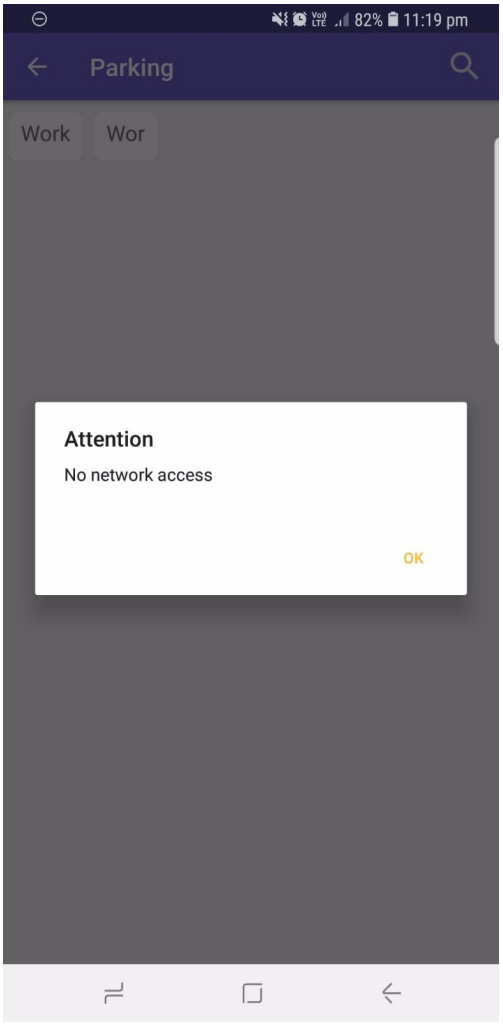
Stress testing is essential in determining the robustness of the app by testing above the normal operation scenarios and push it more than an average user operation. We have focussed mainly on 4 types of stress testing:

6.3.1 Network Connectivity

Simply by turning OFF the WiFi or turning ON the airplane mode (for total connectivity loss) and using the app should not crash. Here, we want to test things like:

- Will the app crash if a network request was already in progress?
- Notify the user that the connection was lost?

Aways handled all the cases correctly and promptly responded with informative messages



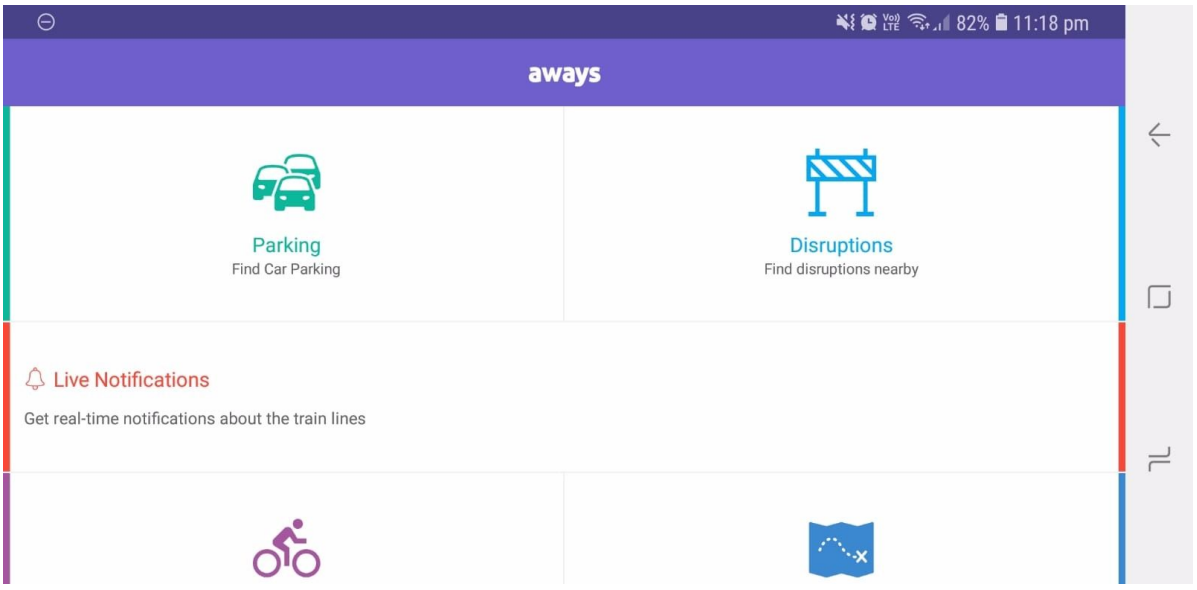
6.3.2 Orientation Change

Another major stress we did was with the screen orientation changes. In Android, the activity or the screen is recreated every time when the device is rotated leaving heaps of rooms for potential memory leaks and null pointer exceptions (NPE).

We want to check the impact of orientation change on the performance and behaviour of the app:

- What happened to any ongoing requests?
- Was the app state restored correctly?
- Was the correct related landscape/portrait layout/resources loaded?

Aways app did not crash throughout this test



6.3.3 App killed in the background

Android has its own garbage collector and can kill apps to clean up memory for other running apps. When we go back to the app all the dependencies required by the current screen should be re-initialized if needed.

Ways to reproduce:

- Open the app
- Press device Home button
- Run the following command in terminal
  `adb shell am kill me.aways`

Aways re-initialised the required screens as required without crashing or behaving unexpectedly.

6.3.4 App Update

In this test, we have checked if the app is backward compatible when updating to a new version. As Aways uses SQLite and SharedPreferences to store the user data inside the phone, there is a chance of getting this data cleared while updating to a newer version.

Aways app passed this test as the `autoBackup` variable is set to `true` in the `AndroidManifest.xml` file and uses GreeenDAO for all SQLite operations.

7. References

- Jessica, T. (2017, May 9). How to Solve Android's 13 Most Common Error Messages. *Envatotus+*. Retrieved Oct 15, 2018, from https://code.tutsplus.com/tutorials/how-to-solve-androids-most-common-error-messages--cms-28706

- Gazar, M. (2016, Dec 21). Stress-testing Android apps. *Medium*. Retrieved Oct 15, 2018, from https://medium.com/default-to-open/stress-testing-android-apps-601311ebf590