trackle: track your vehicle.! Always remember where you parked

Neha Mahajan[*] nsmahajan@wpi.edu
Arun Vadivel[*] avadivel@wpi.edu
Kiran Mohan[**] kmohan@wpi.edu

[*]Department of Computer Science, Worcester Polytechnic Institute
[**]Department of Robotics Engineering, Worcester Polytechnic Institute

ABSTRACT

In today's tight-scheduled life, the average person has the need to remember many things, one among which is the location of the parked vehicle. Surveys show that one in every two drivers (52%) has forgotten where they parked their vehicle and the total time spent in searching for their parked vehicle is estimated to be 200 man-years. To minimize this ineffective time expenditure, we propose *trackle*, a GPS based Android application that automatically identifies the location of the parked vehicle. A user study and survey with 8 participants indicates a positive response to *trackle*. Seven of the eight participants in the study found it to be implementable in their everyday life.

Keywords

Parking location reminder, Smartphone sensing, GPS.

1. INTRODUCTION & MOTIVATION

It is common for people to forget where they parked their vehicle. As a study by insurance.com[1] shows, out of 2000 drivers (1000 male and 1000 female), about 52% (44% of total male and 59% of total female drivers) forgot where they parked their vehicles. Another survey[2] shows that the total time spent searching for parked vehicles is estimated at 200 man-years. This is a major time expense considering the fact that the first automobile was manufactured in 1886 by Benz Patent Motor Car which means that cars have been around for only 130 years. This indicates that for every 1 year spend in manufacturing vehicles; 1.53 years are spent just searching for parked cars. Moreover, as the number of public go-to places increases every year with an even higher increase in the sales of vehicles and increasing congestion in the parking lots of urban cities, the time spent in searching for parked vehicles is bound to increase.

Given this background of time wasted in searching for parked vehicles, we attempt to tackle this problem with *trackle*, an Android based smartphone application developed to automatically or manually record the vehicle's location after it has been parked. Users can then find their parked vehicle and the shortest route to get there with the click of one button.

The contribution of this paper is two-fold. First, we present our novel approach used in *trackle* that detects and saves the vehicle's parked location without external devices or peripheral equipment. Second, we present the results from the user study that resulted in an overall positive response regarding integrating the application to the user's everyday life.

In the remainder of this paper, we discuss related work in Section 2. Next we define the objectives of the application in Section 3. We then present our design approach for *trackle* in Section 4 and implementation specifics in Section 5. Section 6 describes the user study and survey. Discussions are included in Section 7 and we conclude in Section 8.

2. RELATED WORK

ParKing Reminder: Find my car[3] is an application that performs a similar function. However the major difference is that ParKing Reminder forces the user to manually tag the vehicle's parking location with a button click. In cases where the user's vehicle is Bluetooth enabled and paired with the smartphone, the application saves the location at the time of disconnection of the vehicle's Bluetooth device indicating turning off of the vehicle's ignition. However, this makes the application dependent on external factors such as the Bluetooth on the vehicle while *trackle* works independently as a self reliant application.

ParkSense[4] is an application developed for a parallel type of problem. This application addresses the issue of searching for vacant parking spots. The application also considers energy efficiency and hence utilizes relative WiFi signal strengths as the core parameter to detect vehicle movement from a parked location and indicates that a parking slot has become vacant and this information is passed around to all other users of the application thus reducing the burden of searching for parked locations. However, our application deals with the issue of finding the vehicle once it has been parked. This is a complementary problem to the one considered in ParkSense.

Typical systems that currently address this issue require some additional peripheral hardware with GPS features installed on the vehicle. The mobile application communicates with this peripheral device to request the GPS location of the vehicle.

Other systems use external devices in the vehicle such as an audio system with Bluetooth capability. Mind Mobile[5] is one such system which uses the Bluetooth and GPS of an additional device that also functions as an in-vehicle USB charger. Mind Mobile uses the external device to send a trigger to the smart phone when the vehicle is changed to park mode and the ignition is turned off. The application on the mobile device saves the location at the time instant when the trigger is received.

The major difference between these systems and *trackle* is the design decision to make *trackle* a self-sufficient, standalone application that does not require any external or peripheral devices to be fully functional. This reduces potential failure modes of the additional external devices and hence improves reliability of the application.

3. OBJECTIVES

The primary objective of *trackle* is to automatically or manually save the parked location of the user's vehicle and to display the path back to the parked vehicle upon user's request.

The secondary objective is to make the application convenient to use by providing relevant features such as history tracking, capability to capture photos and notes and parking cost calculator.

The tertiary objective of the application is to make its UI (User Interface) minimalistic and elegant.

4. DESIGN APPROACH

The design of the application was done in two phases – first was choosing the APIs required and second was implementing them. While choosing the APIs required, several options were considered and based on certain requirements, the APIs chosen were as shown in Fig.1.

The decision to go with the Fused Location Provider API instead of the regular Google Play Location Services API was a crucial one. This is because the newer Fused Location Provider API is quicker in providing location updates and is more power efficient. This API handles power consumption by requesting location updates only when required as compared to the interval-based location updates requested by the older Location Services API.

There are two modes of operation of the application – Manual and Auto. Both modes operate in the same way except for one difference. In the Auto mode, the parked location is automatically saved while in the Manual mode, the user has to save the location of the parked vehicle manually by clicking a button. We explain the design approach of the Manual mode first and subsequently that of the Auto mode. A flow chart of the manual mode is shown in Fig.2 and for the auto mode in Fig.3.

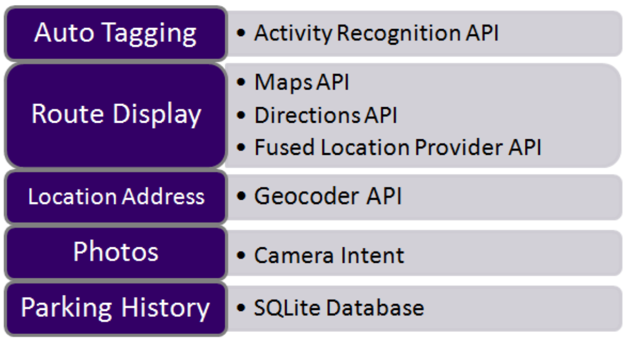


Fig.1: Design Approach

In the Manual mode, the user clicks the "Park" button once the vehicle is parked. The Fused Location Provider API is used to get the latitude and longitude of the location. The application creates a new Universally Unique Identifier (UUID) and saves the latitude and longitude of the user's location when the button is clicked. This UUID and its corresponding latitude and longitude is then stored to the database. For database functionality, the SQLite Database is used. The application provides the ability to capture three photos and also take notes on the parked location. Camera Intent was used for capturing photos. If the user has parked in a paid parking lot, the duration of parking and the cost can be entered using sliders. All of this is also stored in the database.

In the Auto mode, the parked location is automatically detected. This is done using the Activity Recognition API. The website in [6] gives a list of activities that the Activity Recognition API can recognize. The user's Activity is predicted every one second. The application checks if the predicted activity transitions from "IN_VEHICLE" to any other state. If this is true, the location at the instant that the transition occurs is saved temporarily and a flag is set to true. Once the flag is set to true and the user's location changes 30 meters or greater away from the temporarily saved location without a transition back to IN_VEHICLE, this is indicative of the user having parked his vehicle and this is the **"parked" condition**. If the activity of the user transitions to IN_VEHICLE before 30 meters, the flag is reset to false and hence, the temporarily saved location is not saved to the database. In the case that "parked" condition as mentioned above is detected, the "park" button click is simulated using the performClick() function and thus, all the operations that occur when the "park" button is clicked in manual mode are performed.

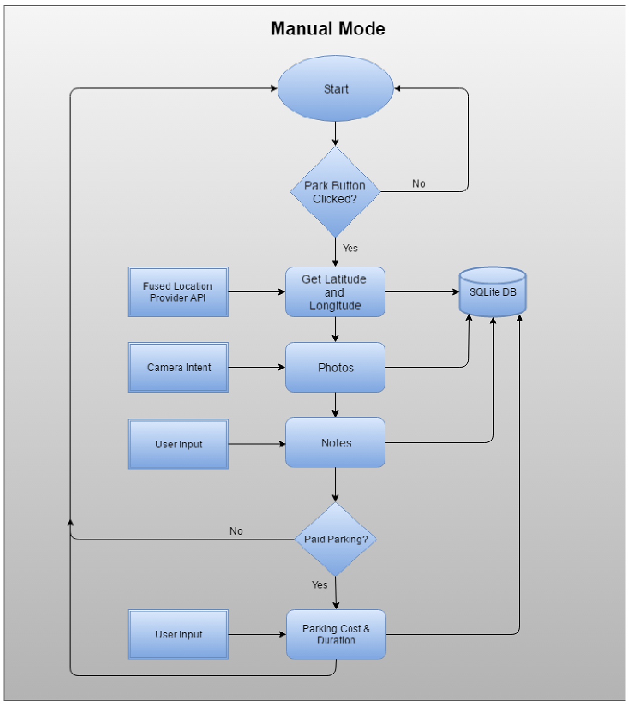


Fig.2: Flow Chart of Manual Mode

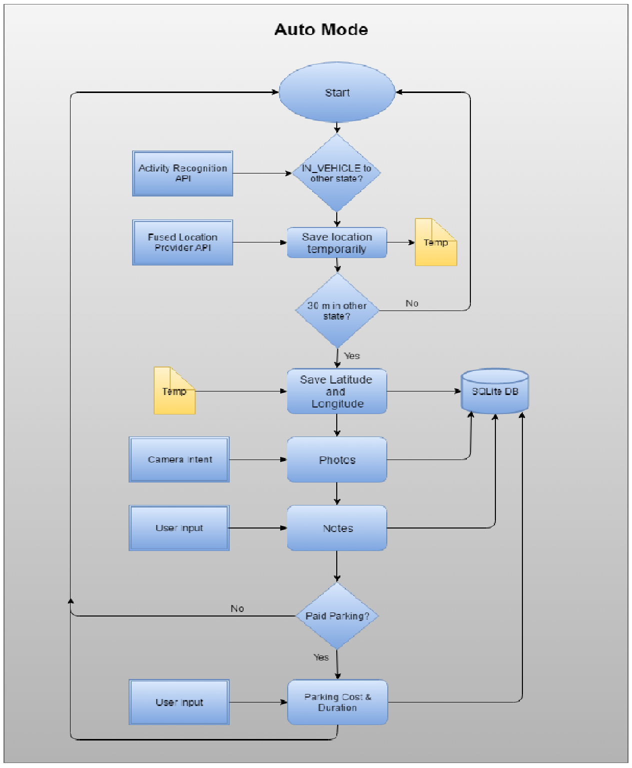


Fig.3: Flow Chart of Auto Mode

4.1 FEATURES

trackle comes with the following features.

Saves the location of user's parked vehicle

The latitude and longitude of the user's parked vehicle is saved either manually or automatically depending on the mode of operation as selected by the user.

Displays shortest path & ETA back to vehicle

The Directions API was used to obtain the shortest path and distance to the vehicle. The Estimated Time of Arrival (ETA) was calculated based on the average human walking speed and the distance to the vehicle as obtained from the Directions API.

Saves history of parked locations

The user has the option to save the history of his parked locations. Every time the "park" button is clicked or every time *trackle* automatically detects that the vehicle has been parked, the location along with the photos, notes, parking cost and duration is automatically saved to the database provided that the Location History has been enabled in the Settings Menu.

Has capability to capture and store photos & notes

At every parking location, the user can take photos and notes and also make a note of his parking duration and cost using sliders. All of this information is stored in the database for future retrieval.

Has a minimalistic UI design

Throughout the application development process, the User Interface was given as equal importance as the functionality of the application. Floating Action Buttons were used for elegance and the theme of the application was kept uniform. The main idea was that the theme of the application should complement the functionality.

Reduced battery consumption

Power efficiency was considered while developing the application. As *trackle* is dependent on GPS for its functioning, we were concerned with the battery consumption that would be incurred by running *trackle*. Doing some research, it was found that the Fused Location API would request for location updates only when requested. This helped in addressing power efficiency issues.

5. IMPLEMENTATION SPECIFICS

The entire application was built using Android Studio v2.0. We present the number of lines of Java code, number of classes, number of XML files and number of icons. The icons were generated using GIMP (GNU Image Manipulation Program).

The application was developed using:

- 2075 lines of Java code
- 14 Java class files
- 14 XML layout files
- 11 icons

The in-house testing of the application was done on a Samsung Galaxy Note 5 SM-N9200 and also on a Google Nexus-5 smartphone. All application functionality was tested and the application worked flawlessly on both devices.

6. SCREENS & BUTTONS

The following figures are screen-shots from the application. They show the flow from one screen to the other by clicking respective buttons.

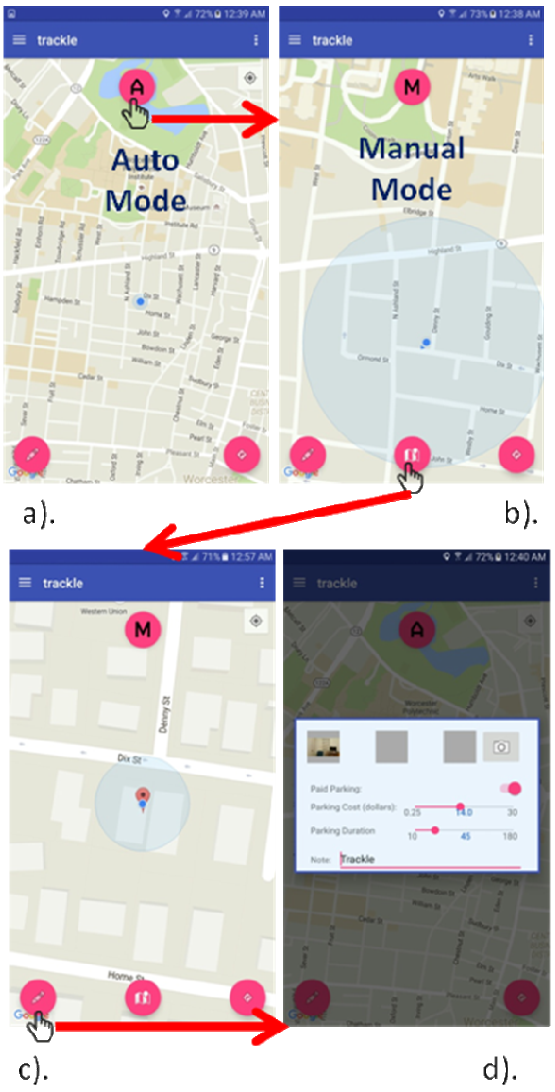


Fig.4: a) & b).Auto / Manual switching. c).Park button clicked. d).Edit button clicked.

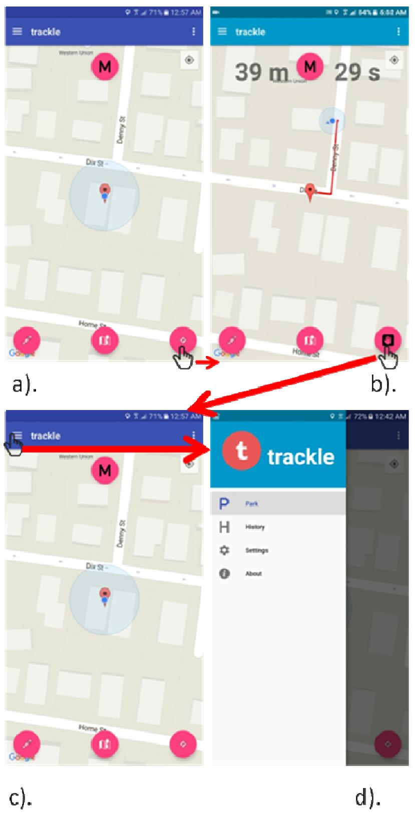


Fig.5:
a).Manual mode.
b).Directions button clicked.
c).Done button clicked
d).Navigation Drawer clicked

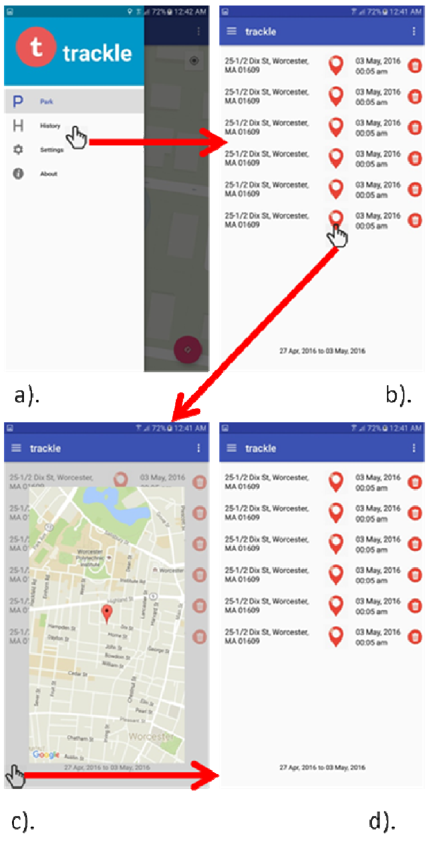
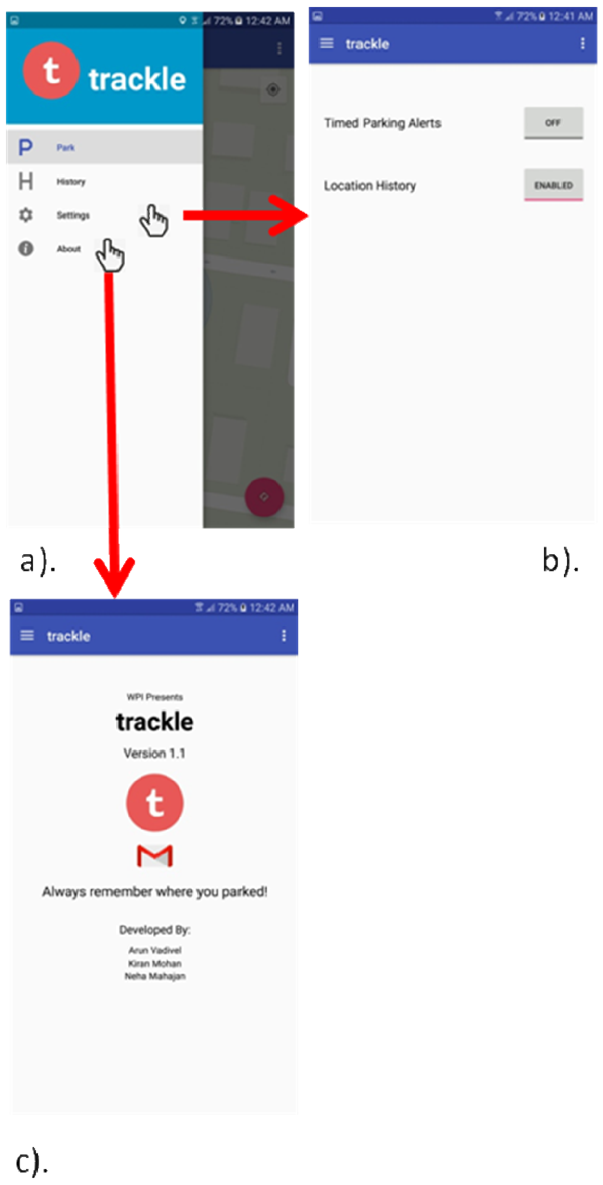


Fig.6:
a).Navigation Drawer
b).History Button clicked
c).View Location clicked
d).Outside map dialog clicked



Timed Parking Alerts OFF

Location History ENABLED

WPI Presents
trackle
Version 1.1
Always remember where you parked!
Developed By:
Arun Vadivel
Kiran Mohan
Neha Mahajan

Fig.7: a).Navigation drawer b).Settings button clicked c).About button clicked

The first screen is the fragment_park which is the main screen. This screen has four floating action buttons. The button on the top center is the button to toggle between the manual and automatic modes of the application. The three buttons at the bottom (from left to right) are the Edit, Park and Directions buttons. If automatic mode is selected, the Park button is removed from the screen as the parking action would now be done automatically and this button would lose its functionality and become redundant. In the manual mode, this button is enabled and shown to the user. If the user has parked his vehicle and clicks on the "Park" button, the location marker appears on the map on the user's current location. The location is saved in the Database using a newly created UUID. In the automatic mode, the same events occur when the "parked" condition is satisfied as explained in the Design Approach (Section 4).

The Edit floating action button can be clicked to bring up a dialog in which the user can take photos, notes and also set the parking duration and cost. Once the user clicks out of the dialog, the data in the dialog is saved to the SQLite Database to the corresponding UUID.

When the user wants to return to his parked vehicle, the "Directions" button is clicked and the shortest path to the parked vehicle is displayed along with a dynamic estimation of the distance to the destination and Estimated Time of Arrival (ETA).The path displayed is also dynamic and this takes care of re-routing if the user takes a route that is not the shortest path. When the directions button is clicked, the same floating action button changes to the Done button.

Once the user has reached his vehicle, the Done button can be clicked and the dynamic path is removed and dynamic estimation of ETA and distance is stopped.

The Navigation Drawer can be opened by sliding from left to right on the fragment_park. The navigation drawer has three options – Settings, History and About.

In the Settings screen, the user has 2 toggle buttons – to enable / disable parking alerts and to enable / disable location history. The parking alerts feature has not been implemented and hence, the first button is non-functional. Toggling the Enable / Disable Location history toggles whether the application saves the history of parked locations.

The History screen contains the history of parked locations along with the Address obtained using the Geocoder API, photos and notes taken at that parking location, a view button to view the location on a map dialog and a delete button to delete that particular history item.

The About screen contains and displays details about the application such as version number and developers of the application.

7. EVALUATION

A user study was conducted with 8 participants. All 8 participants had their own Android smartphones. *trackle* was installed on their smartphones and the users tested the application for four days. After the four days of testing, the participants were given a survey in which four questions were asked. The questions were as follows.

1. Which mode do you find most effective? Manual or Auto?
2. Was *trackle* helpful?
3. Would you use *trackle* in your everyday life?

4. In a few words, please describe what aspect of *trackle* would make it more usable.

A sample survey is shown in Fig.8.

7.1 Results of Evaluation

The results of the survey were positive and the feedback received was valuable. A table of results is shown in Table-1. The pie charts for the results are shown in Fig.9, Fig.10 and Fig.11.

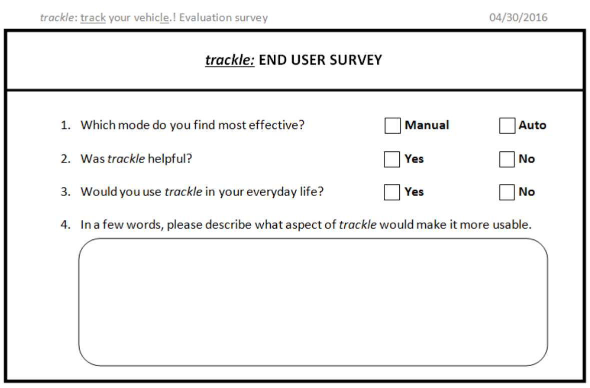
All eight participants said that the Manual mode was more effective than the Auto mode. This was expected because the Auto mode depends a lot on the quality of the sensors on-board the smart phone.

Q No	Question	Auto	Manual
1	Which mode do you find most effective?	0	8
Q No	**Question**	**Yes**	**No**
2	Was *trackle* helpful?	7	1
3	Would you use *trackle* in your everyday life?	6	2

Table-1: Results of survey

Seven out of eight participants (87.5%) responded that *trackle* was helpful. While one participant said that it wasn't helpful, the reason for this was found to be that the participant was looking forward to test only the Auto mode and since the functioning of the Auto mode is dependent on the Activity Recognition API and the sensors on-board, the user experienced issues such as not tagging at the right spot and tagging at wrong spot. This is beyond our control as there are inaccuracies in the prediction capability of the Activity Recognition API and also inaccuracies of on-board sensors.

Six out of eight participants (75%) said that they would use *trackle* in their everyday lives. The remaining two participants had a limited mobile network data plan and they felt that our application was consuming considerable amount of data. We are yet to take statistics on the amount of data consumed by the Fused Location Provider API. Once this data is collected, we would be able to quantify the data consumption of our application.


trackle: track your vehicle.! Evaluation survey 04/30/2016

trackle: END USER SURVEY

1. Which mode do you find most effective? ☐ Manual ☐ Auto
2. Was *trackle* helpful? ☐ Yes ☐ No
3. Would you use *trackle* in your everyday life? ☐ Yes ☐ No
4. In a few words, please describe what aspect of *trackle* would make it more usable.

Fig.8: Sample user survey

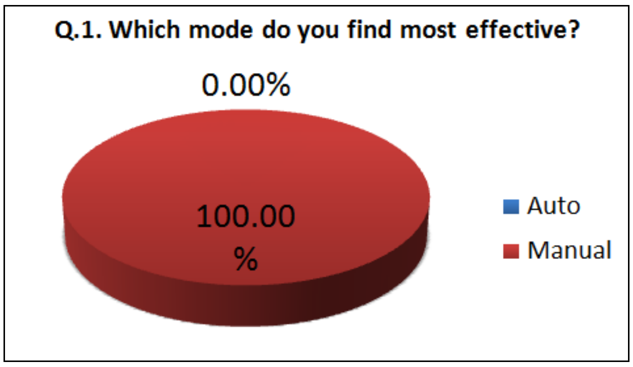


Fig.9: Pie chart for results of Q1 of survey

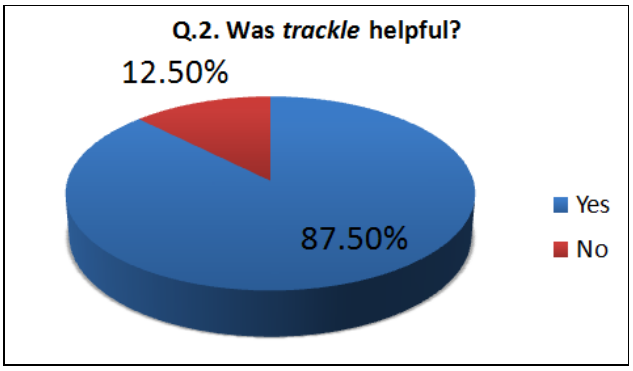


Fig.10: Pie chart for results of Q2 of survey

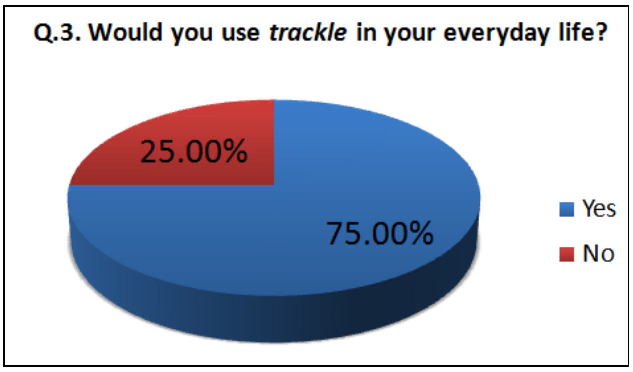


Fig.11: Pie chart for results of Q3 of survey

8. DISCUSSION & FUTURE WORK

In this section, we present the pending work and also the future work which can be continued by researchers based on the foundation provided by *trackle*.

The Auto mode does not work in the background i.e. *trackle* needs to be opened in the foreground for the Auto mode to function. Another situation that *trackle* does not function at its best is when GPS and Network services are unavailable. The application seems to crash when tested at such situations. If this is handled, the application can be used to just take photos and notes without having the exact location of the parked vehicle. One noteworthy point is that for Android 6.0 and up, manual permissions need to be given by the user by navigating to Settings – Application Manager – *trackle* – Permissions – Enable Location and Storage Permissions. The application would fail with a toast saying "Unable to Load Map" if these permissions are not given manually. It is possible to integrate the permission request into the application. However this was not implemented due to time constraints.

Future research work can include using cloud storage services like the Amazon Web Services (AWS) to store the parking history data with a dedicated user login. Another significant contribution would be to perform predictive analysis on stored and real time streaming data to predict parking availability and traffic at different locations in real-time. Integration with automobile manufacturers and Maps services like Google Maps and Here Maps can lead to providing live parking traffic updates at different locations and live parking fare updates to users.

9. CONCLUSION

In conclusion, we present the features that were proposed versus what was actually achieved. Table-2 shows the list of

A video of the application at work is attached with this report. The video has also been published to YouTube and can be found at https://youtu.be/yyCF0J0q6ZQ

Feature	Proposed	Achieved
Auto-Tag	✓	✓
Manual-Tag	✓	✓
History DB	✓	✓
Photos & Notes	✓	✓
Cost Calculation	✓	✓
Time Expiration Alert	✓	✗
ETA & Estimated Distance	✗	✓
Dynamic Path Change	✗	✓
Power efficiency	✗	✓
Shortcut to Email Developers	✗	✓

Table-2: Proposed vs. Achieved features

All proposed features, except for the time expiration alert were implemented. Four extra features that were not proposed were also included in the application – the first being the Estimated Time of Arrival at the destination and Estimated Distance to the destination. The second extra feature is the dynamic path change where the path to the destination would change dynamically i.e. the path displayed would shrink or grow as the user walks towards or away from the destination. This also helps in re-routing if the user decides to take a different path to the destination.

The third extra feature bundled into the application is the Power efficiency. This is handled by the Fused Location Provider API which implements requirement-based location updates instead of interval-based updates as explained in our Design Approach (Section 4). The fourth extra feature is the Shortcut to Email Developers. This is present in the above screen and when clicked takes the user to the Gmail application with the emails of the developers pre-defined in the "To:" field. Users can provide feedback directly to the developers using this feature.

10. ACKNOWLEDGMENTS

We express our thanks to Satish Narkhede for his inputs into the functionality and usability of the application during the initial brainstorming sessions.

We also extend our gratitude to Professor Emmanuel Agu, Instructor of CS528 at Worcester Polytechnic Institute for motivating us throughout the project timeline.

We thank the eight participants who helped us evaluate our application and gave us honest feedback.

11. REFERENCES

[1] http://www.insurance.com/about-us/news-and-events/2014/05/biggest-driving-embarrassments-forgetting-where-one-parked-and-driving-over-curbs.html

[2] http://www.dailymail.co.uk/news/article-2688266/The-200-years-spend-looking-lost-cars-One-seven-drivers-admit-forgotten-parked-vehicle.html

[3] https://play.google.com/store/apps/details?id=il.talent.parking&hl=en

[4] Sarfraz Nawaz, Christos Efstratiou, Cecilia Mascolo. Smart Sensing Systems for the Daily Drive, IEEE computer society.

[5] Mind Mobile App http://dev.goldcrownelectronics.com/?page_id=360

[6] https://developers.google.com/android/reference/com/google/android/gms/location/DetectedActivity#constant-summary