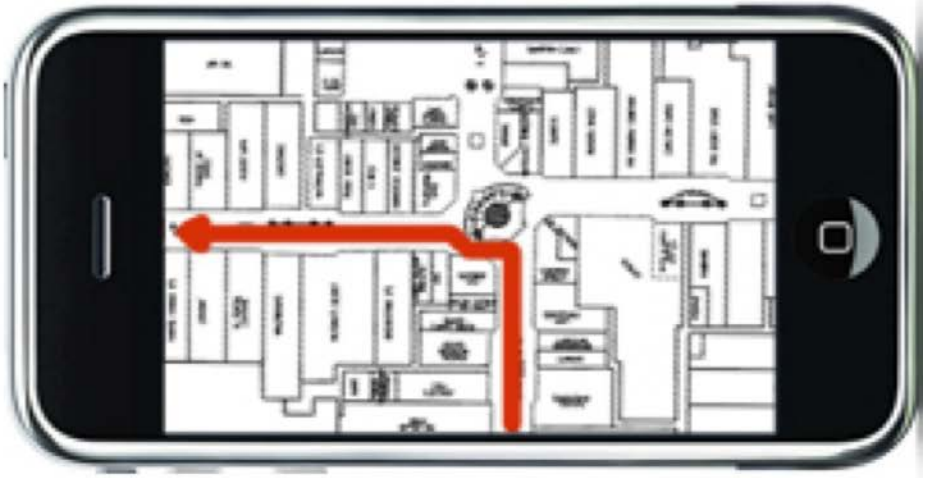
# Project Report

# Take Me There - Indoor Navigation Framework




Alok Nerurkar
Jigar Patel

Instructor: Prof Norman Sadeh

Mobile and Pervasive Computing Services

# Contents

## Acknowledgements

We extend our sincere thanks to Professor Norman Sadeh for all the guidance he has provided us during the development of this project. We would also take this opportunity to thank the TA's for their insights and helpful tips. We also would like to thank Linda Francona for coordinating our meetings with Professor Sadeh. We would also like to thank all the people listed below whose research was very helpful in implementing our key feature - tracking the user. Especially it helped us in not spending time on things which were already tried on and whose results were not much encouraging.

1. Oliver J. Woodman
   http://www.cl.cam.ac.uk/research/dtg/www/publications/public/ojw28/Main-PersonalRedist.pdf
2. Ubejd Shala, Angel Rodriguez
   http://hkr.diva-portal.org/smash/get/diva2:475619/FULLTEXT02.pdf
3. Kurt Seifert and Oscar Camacho
   http://www.freescale.com/files/sensors/doc/app_note/AN3397.pdf
4. Paul
   http://www.thousand-thoughts.com/2012/03/android-sensor-fusion-tutorial/3/

## Executive Summary

"Take Me There" is an android based mobile application which helps users to find destination within a building. As outdoor navigation has become part of day-to-day life, the next thing what users want is indoor navigation especially in large buildings. In this document we will go through the details of the application covering all issues related to app development though mainly it focuses on technical issues as that was where we spent most of our time initially.

There are many approaches to solve this problem each using different set of hardware and external infrastructure. But the one which we were trying was using sensors available only on the smartphone and which does not require any expensive installation of external infrastructure.

The application has two major parts. First part shows the path to the user on a map. And second part tracks the user as he walks towards destination and show progress on the map. First part requires building a database of possible destinations and routes using digital maps. Second part was about identifying where the user is based on accelerometer, gyroscope, magnetometer, QR Codes and map.

During first half, we focused extensively on second part of the application i.e. tracking the user and tried various ways but none of them was accurate in determining exact location of user. Using Sensor Fusion techniques we were able to get the change in direction of the user. But tracking all kinds of movement of user just using mobile based sensors accurately was not working. Finally, we decided to use Step Counter sensor to count number of steps users has taken and use data from maps to estimate where the user is. This solves the problem in best case scenario when the user is walking towards destination but leaves the problem of precise tracking in all other scenarios unsolved.

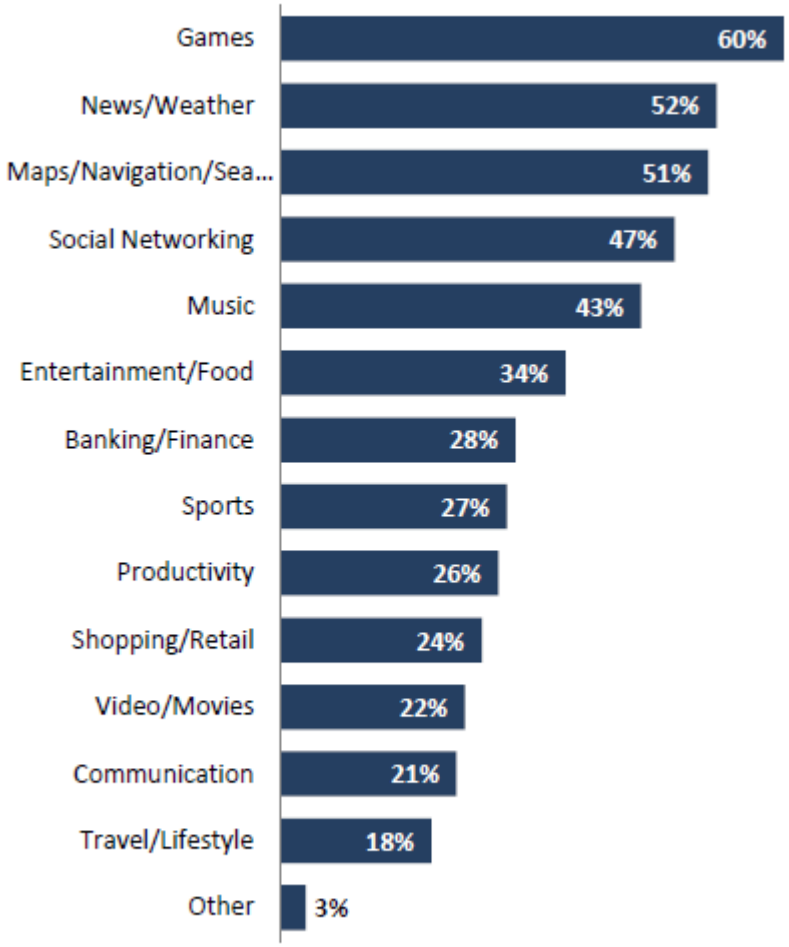
In the second half of the semester, we focused more on first part of the application and created the database and implemented the route finding user interface and connected it to the database and sensor service built during first half.

This application may not solve all cases but can be used by those who feel completely lost when they enter new building and can get to the destination if they follow the path shown in the map. Also, we feel that as the smartphone sensors are getting more and more sophisticated and with the advent of wearable devices like the Samsung Smart Watch or the Qualcomm Togo, step counters and other such features will get more and more accurate and our approach can be extended. We would like to take this forward to cover even more cases and make tracking more accurate.

## Market Study

Currently 1 out of 4 people in the US actively use mobile apps. Out of them 51% use navigation apps on their mobile. This means navigation currently lies in the top three apps used by people today.

*% of Nielsen recent downloaders who have used each category of apps in the past month...*



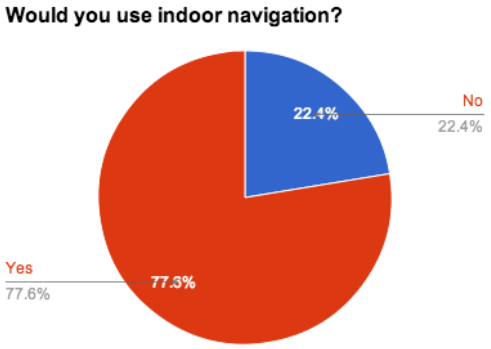

**Source:** The Nielsen App Playbook, December 2009. N=3,962 adults who have downloaded an app in the 30 days prior to the survey.

Social traffic and navigation app Waze now has 45 million users. Clearly people like their phone to navigate them around. With the outdoor component of navigation fully developed we are clearly ignoring a huge market - Indoor Navigation. All the existing solutions for indoor navigation require the use of external devices like Wi-Fi access points, Bluetooth LE devices etc. As a result, people are reluctant to incur the additional hardware cost.

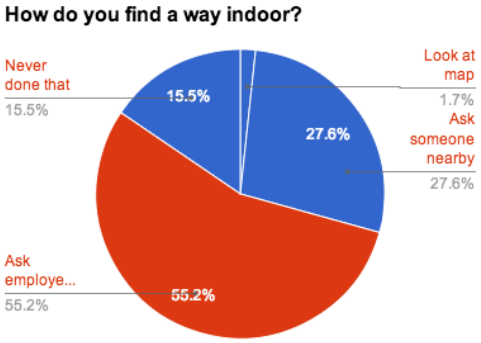
## User Survey

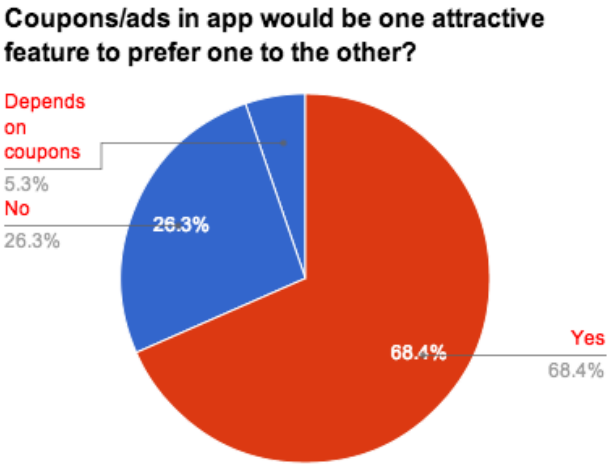
We did the survey and got 58 answers.
- 85% were lost in the building and asked to the people near them or to the employees.
  → They did not use the indoor maps.

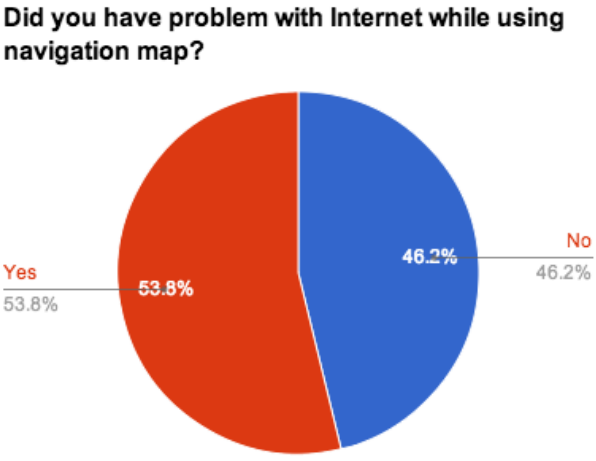
-77.3% answered they would use indoor navigation!




→ Our app is useful for them.




## Customer Survey

**PittMills Mall (Galleria)**

Reply from PittMills

1. If this will provide value to your business ?

It would be a value if used correctly. We would like the customers to know of the sales and specials the stores have but not to the point that they are overwhelmed when they are walking the mall by getting information on every store they pass.

2. Has anyone already reached out to you regarding something similar ?

No

3. Has any customer come up to you and requested such a product ?

No but mainly because they do not know it may exist.

4. And lastly would you be interested in implementing this at your mall ?

Possible but I can see issues. Our common area of the mall is 260,000 square feet. The shoppers do not go right from the mall entrance to only one store. They may take hours and walk in different directions. They also may walk back and forth which means no two customers will walk the mall the same way to the same stores. How accurate will it be if it is based on foot steps?

## Existing solutions


The Meridian NavKit SDK allows you to embed maps and navigation features of the Meridian app into your own custom iOS and Android apps. Coupled with the proper hardware infrastructure (Wi-Fi Access points), the BluDotKit SDK enables the use of indoor location awareness. It provides visitors with their location inside your building & monitor visitor traffic patterns.


Localization using BLE (Bluetooth Low Energy) devices provides the exact position inside buildings and notifies the application when the user enters, leaves or changes floor level inside a building. This feature can be used to accurately locate yourself and others in extraordinarily useful contexts such as airports, shopping malls, conference centers, etc.


It also uses Wi-Fi access points. But instead of relying on a static fingerprint model, Redpin does not make a difference between training and usage. Redpin can be used right away and allows its user to enter the symbolic labels if necessary. As this is an ongoing process, Redpin can also easily adapt to changes in the environment, caused for example by replaced access points.


Based upon the user's needs, wifarer will suggest one or a combination of the indoor positioning engines that they support - Mobile Centric Wifi, Network Centric Wifi (Cisco MSE) or Bluetooth LE / iBeacon.

## Motivation

- We feel there is a need for a more universal solution. If you design a framework based on Bluetooth LE devices, you will get good accuracy, but every building will not have BLE devices.
- Plus due to the different technologies, a building with enough Wi-Fi access points cannot be serviced with a framework devised for BLE devices. Thus there is not enough reusability.
- With the smartphones getting more and more sophisticated, we feel using the phone sensors will be the direction to go. There are highly accurate step counters available.
- As more and more wearable technologies get added to the market, this is the direction to go forward. Recently Samsung launched the S5 with support for the Smart Watch. This can calculate the steps more accurately.

## Business Model

### Target End Users

**Visitors** - Whenever visitors enter large building like CMU campus then they can easily get lost esp. when sign boards are missing or not located at appropriate places. They can find out where a particular building is using Google maps but once they enter the building GPS based apps are of no help and the user is left all on its own to figure out the way. Our app targets especially these set of users to help them navigate indoors.

**Unfamiliar sections** - Even daily users of a building may get lost within certain sections of a building if they are visiting that area for first time. They can also benefit from the app.

### Target Customers

The customers of the application will be the owners of large buildings like University campuses, Shopping Malls, Airports etc.

## Revenue Model

The application will be downloaded from the marketplace for free by the end users. But the customers (building owners) will be charged for feeding their building's data into the database. The charges would depend on the amount of data and type of potential end users for that building (Commercial establishments would be charged more than university).
One of the key things we are trying to achieve is to have zero or minimal cost on part of the customers in installing external infrastructure like iBeacons, Wifi Access points etc. And the alternative we explored is using QR Codes.

### Risks

Our solution works only on latest smart phones which have gyroscopes and step counters available. But the smartphone market trend is clearly showing that in a couple of years all the smartphones will have these sensors.

## Application Flow

- The end user will enter the building and start the application.
- He will then scan a QR Code located near the entrance.
- This QR Code will contain information about that location. It will inform the app about the building, floor and starting point within that floor. Based on this information, application will show the map of that floor and also show a list of important landmarks in that building.
- The user will select the destination from the landmark list or provide input using either voice or typing it.
- The app will show the path to destination from the starting point by drawing the path in yellow over the map.
- Once the user starts walking, the map will be updated to show a red line which indicates the user walking over the path.
- When user is supposed to make a turn, the app will indicate that by both voice and toast message.
- When user reaches destination, the app will stop tracking and indicate the same to user.

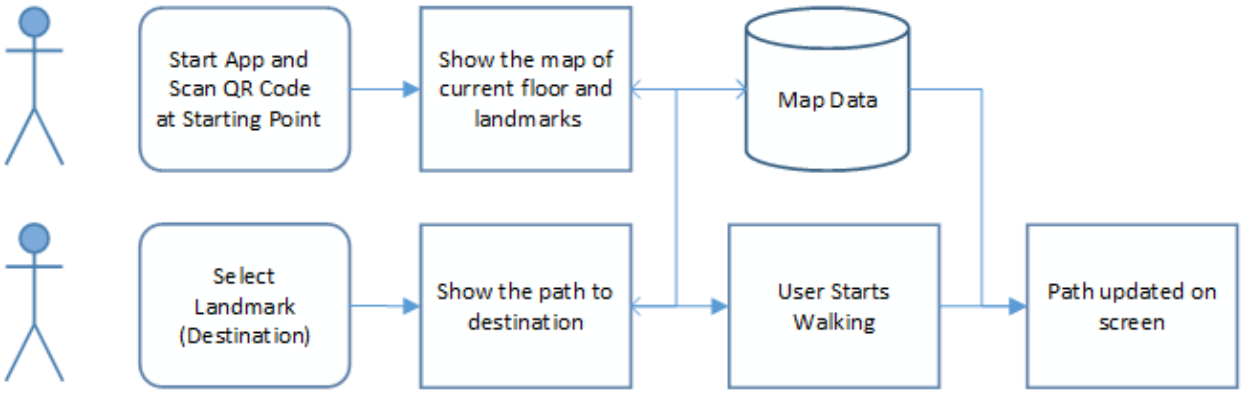
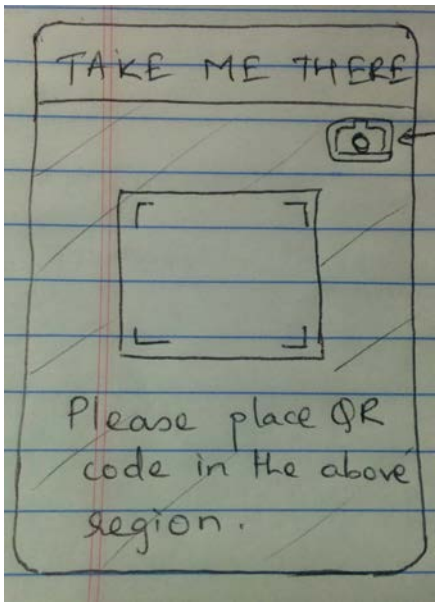



**Figure 1. User Application Flow**

## UI Mockups

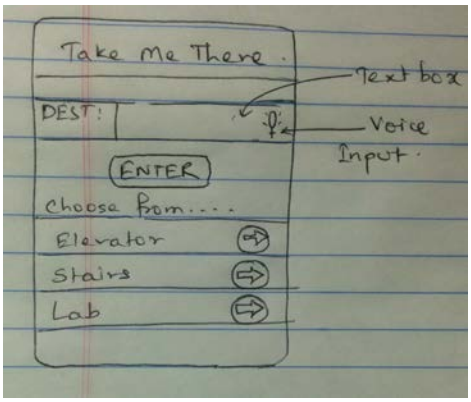
### 1. Scanning QR code to give locations

As the user enters the building he scans the QR code at the entrance he wants to enter from. Each QR code will have embedded information about the starting point on that particular place. It will also download the floor maps for the building.




### 2. Select Destination

Then he will prompted by a screen which gives the most popular destinations. Along with that he will be given a space to provide destination through text for with a voice command.

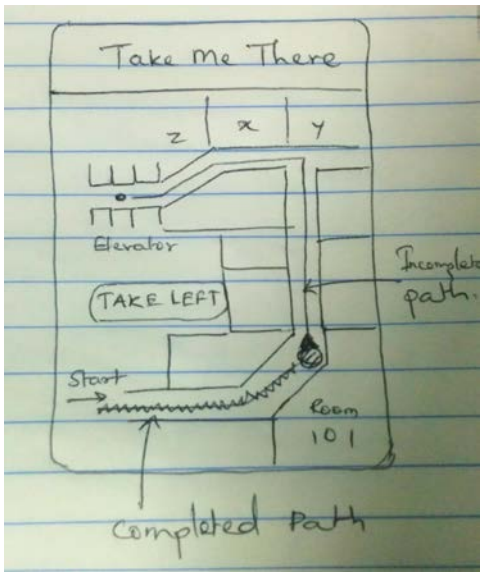


### 3. Get directions and path

After the user selects the destination, a set of directions will be provided along with the paths. The destination can be on a different floor. The path will be colored yellow.

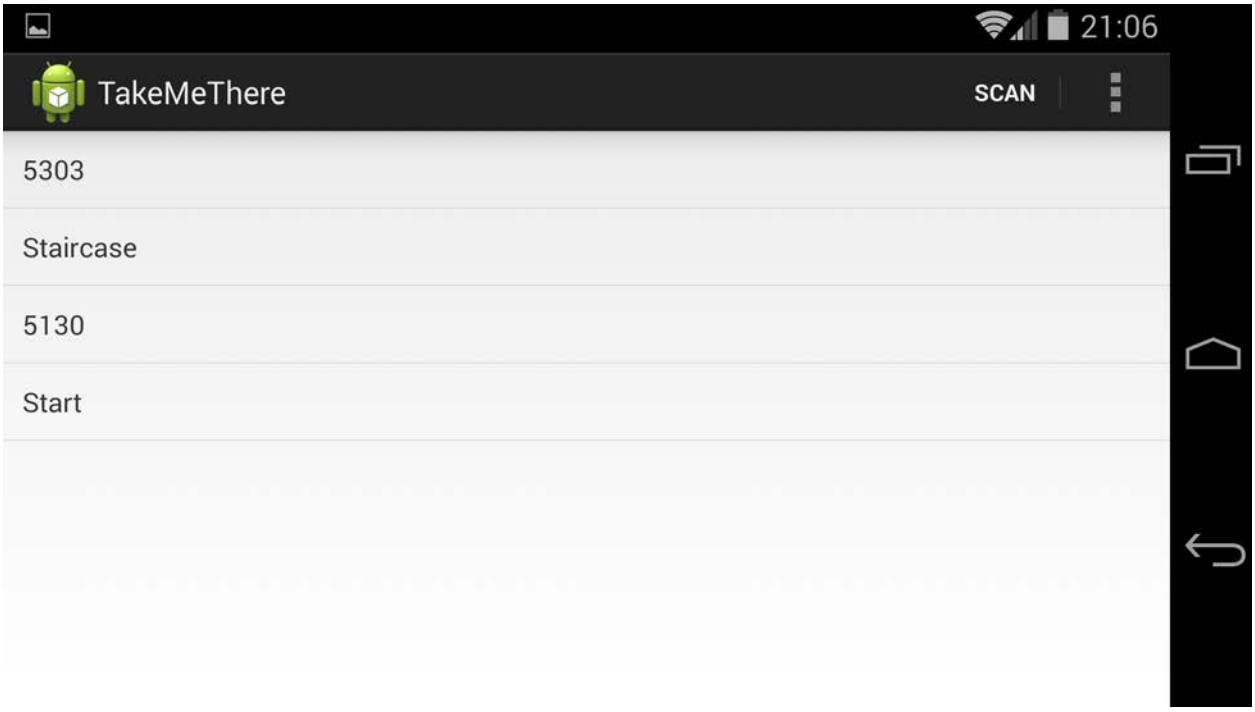
### 4. Start walking on the path

As the user starts walking on the path we color the completed path in red.

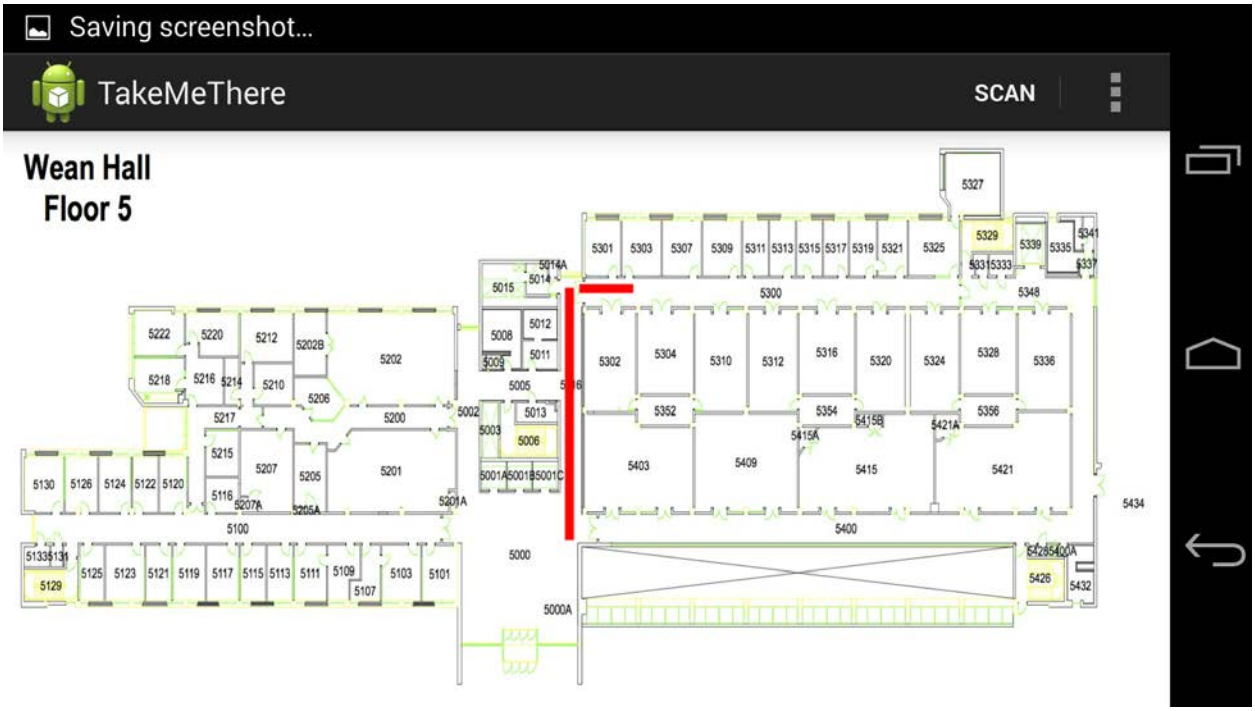


## Final UI Screenshots

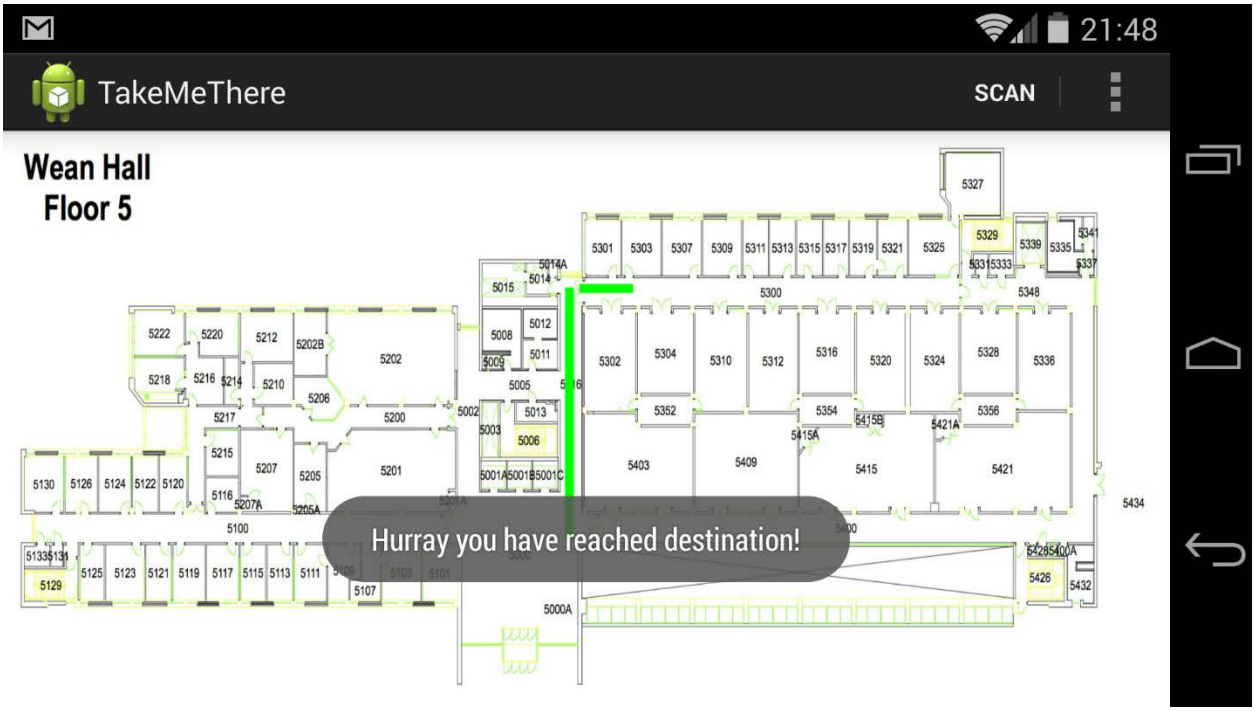

1. **Scanning QR code**
2. **Select Destination**



## 3. Get directions and Path




## 4. Start Walking



## Application Architecture

The application has been divided into two major parts

1. Route Finder
2. Movement Tracker

### 1. Route Finder

This component will draw the route over map given start and end points. A Route consists of multiple paths. Each path is the smallest possible walkway where the user does not have to take any turn. With a given map, we can create multiple paths for each floor. Each path will have starting and end point (in map coordinates) and will also have distance associated with it.

We will pre-compute all the routes from all possible start points to destinations in that building. We had 2 options here - either to pre-compute all possible routes or do it at run time. First option will be heavy on memory but high on performance as compared to the other option. We decided in favor of earlier because our app is target to high-end smartphones which will have enough memory and also for the fact that within indoors, response time of the app becomes critical.

As shown below, we modeled each Building as one entity. Each building will have multiple floors, elevators and staircases. Each elevator/staircase will be attached to multiple floors. And each floor will consist of paths as described above. A route will consist of multiple paths within a floor. Each starting or destination point is modelled as Location. And hence each route will consist of Starting and Ending Location but each path may not be as paths can be between two intersections at which there might not be any location available. The initial database design is shown below. This design is supposed to change as we start implementing it.

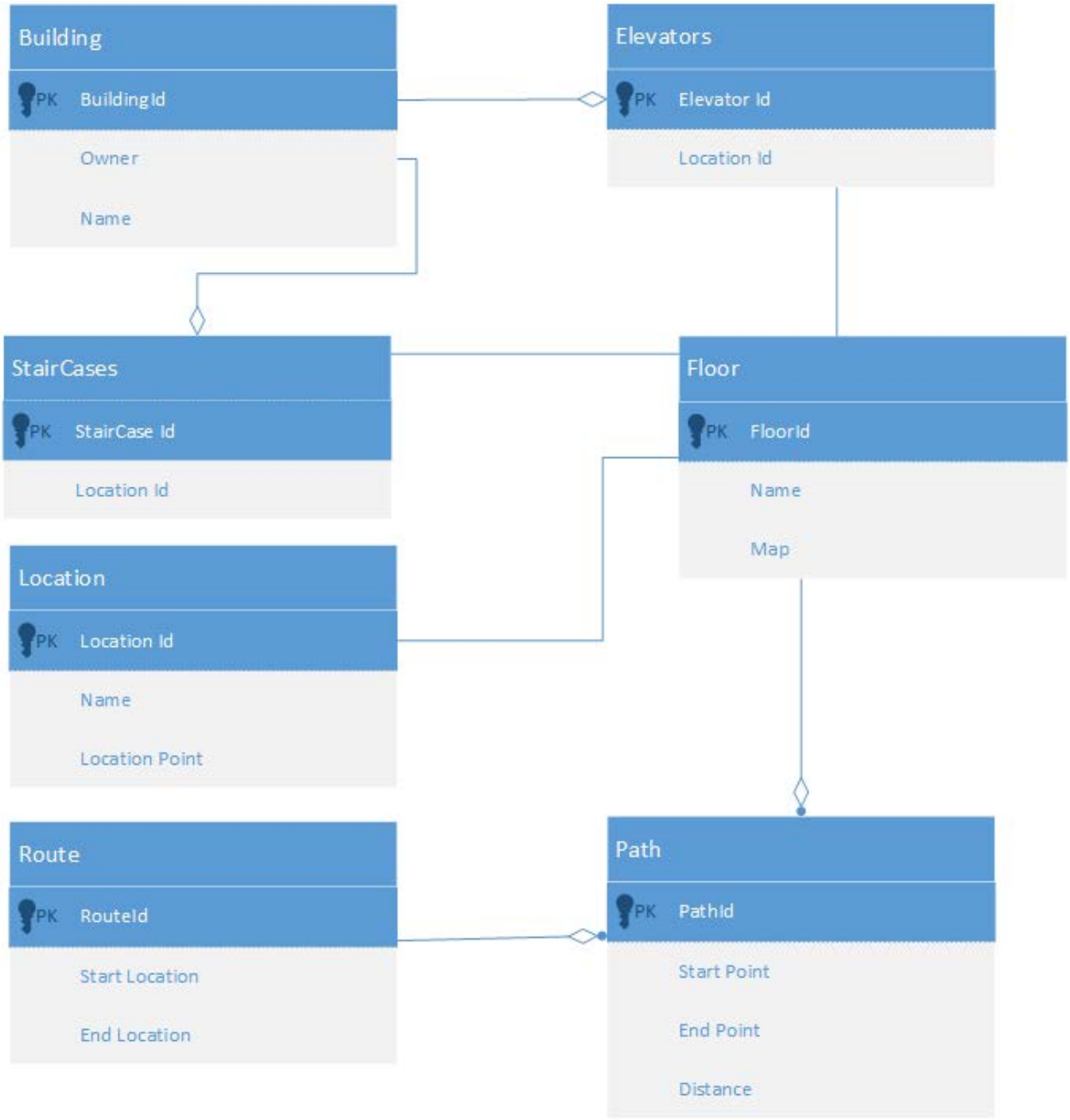



**Figure 2. Initial ER Diagram**

## 2. Movement Tracker

It helps in identifying where the user is with respect to starting point. The objective was to help the user in reaching destination and not to precisely track the user. To help the user in navigating, we show his progress over map as he starts walking towards destination. This was the most challenging part of the project and we spent nearly 80% of our time on this.

Initial design was to use Beacons to locate the user. With Apple launching iBeacon support in its devices, it looking very promising to use them for location tracking. With iBeacons installed at regular intervals, we can get the precise location of user using the receiver applications on smartphones. And these receive applications can be built for both iOS and Android (though there will be 2 different applications).

But then we realized that it would be highly costly to install and maintain iBeacons as their range was limited to 50 meters. In order to avoid these costs we turned out entire attention to the sensors available only on the mobile phone. Most modern smartphones come with accelerometer, gyroscope and magnetometer (compass). Brief overview of each sensor is given below:

**Accelerometer:** It measures acceleration in all three axes including force of gravity. Conceptually, an acceleration sensor determines the acceleration that is applied to a device (Ad) by measuring the forces that are applied to the sensor itself (Fs) using the following relationship:

$$Ad = - \sum Fs / mass$$

However, the force of gravity is always influencing the measured acceleration according to the following relationship:

$$Ad = -g - \sum F / mass$$

for this reason, when the device is sitting on a table (and not accelerating), the accelerometer reads a magnitude of g = 9.81 m/s2. Similarly, when the device is in free fall and therefore rapidly accelerating toward the ground at 9.81 m/s2, its accelerometer reads a magnitude of g = 0 m/s2. Therefore, to measure the real acceleration of the device, the contribution of the force of gravity must be removed from the accelerometer data.

**Gyroscope:** The gyroscope values are in radians/second and measure the rate of rotation around the x, y and z axis. Rotation is positive in the counterclockwise direction.
To get the actual orientation those speed values need to be integrated over time. This is done by multiplying the angular speeds with the time interval between the last and the current sensor output. This yields a rotation increment. The sum of all rotation increments yields the absolute orientation of the device. During this process small errors are introduced in each iteration. These small errors add up over time resulting in a constant slow rotation of the calculated orientation, the gyro drift.

**Geomagnetic Field (Magnetometer):** The geomagnetic field sensor lets you monitor changes in

the earth's magnetic field. This sensor provides raw field strength data (in μT) for each of the three coordinate axes. The output values are affected by noise and hence not 100% reliable in all situations.

**Orientation Sensor:** The orientation sensor derives its data by using a device's geomagnetic field sensor in combination with a device's accelerometer. Using these two hardware sensors, an orientation sensor provides data for the following three dimensions:

- Azimuth (degrees of rotation around the z axis). This is the angle between magnetic north and the device's y axis. For example, if the device's y axis is aligned with magnetic north this value is 0, and if the device's y axis is pointing south this value is 180. Likewise, when the y axis is pointing east this value is 90 and when it is pointing west this value is 270.

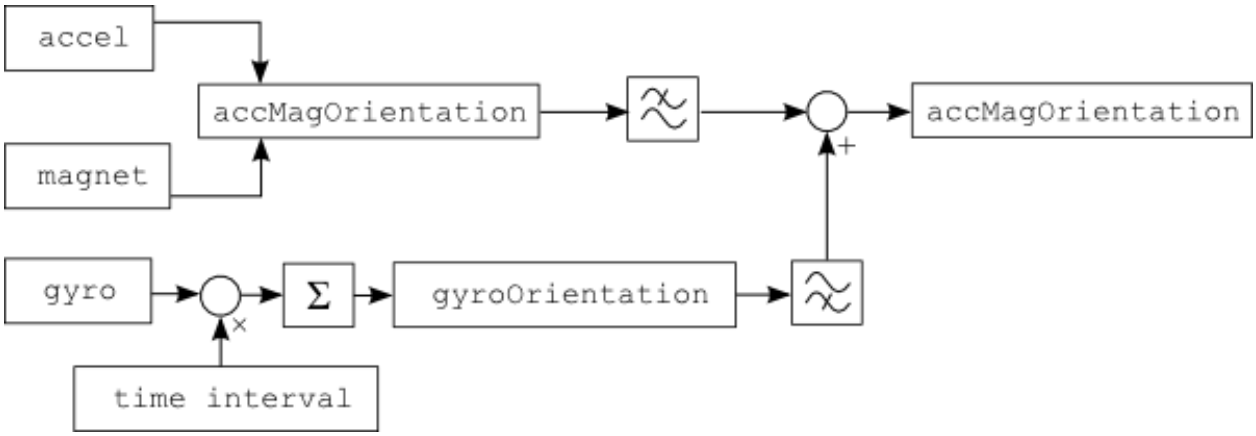
- Pitch (degrees of rotation around the x axis). This value is positive when the positive z axis rotates toward the positive y axis, and it is negative when the positive z axis rotates toward the negative y axis. The range of values is 180 degrees to -180 degrees.

- Roll (degrees of rotation around the y axis). This value is positive when the positive z axis rotates toward the positive x axis, and it is negative when the positive z axis rotates toward the negative x axis. The range of values is 90 degrees to -90 degrees.

Tracking problem has two sub problems namely detecting direction and detecting movement.

## 2.1 Detecting orientation (direction)

We can use gyroscope or orientation sensor for this but they both suffer from errors. To avoid both types of errors i.e. gyro drift and noisy orientation, the gyroscope output is applied only for orientation changes in short time intervals, while the magnetometer/accelerometer data is used as support information over long periods of time. This is equivalent to low-pass filtering of the accelerometer and magnetic field sensor signals and high-pass filtering of the gyroscope signals. The overall sensor fusion and filtering looks like this:




The low-pass filtering of the noisy accelerometer/magnetometer signal (accMagOrientationin the above figure) are orientation angles averaged over time within a constant time window. The high-pass filtering of the integrated gyroscope data is done by replacing the filtered high-frequency component from accMagOrientation with the corresponding gyroscope orientation values.

### 2.2 Detecting user movement

This part deals with identifying displacement of user. It is here where we kind of hit a road block. In theory, displacement can be obtained by integrating accelerometer data twice but in practice the values which we got were not correct. They were off not just by 10-20% but by more than 50%. To improve the accuracy, we tried to apply different filters but the accuracy was no were near desired accuracy. And hence finally we decided to not integrate the acceleration data.

The other alternative was to detect steps taken by user and estimate where he is. We started off by measuring spikes in acceleration data and for every change from positive to negative acceleration values we counted it as one step. Initially it showed some good results but when we tested it under different conditions it was not giving proper results.

Then we tried to calibrate values on startup and sample the values (i.e. take average of 10 samples of accelerometer data) to avoid odd spikes like we did in finding orientation. Though it was better than earlier but the results were still not acceptable.

Looking at other ways, we found that most of them use some foot based sensor which is not subject to any other movements like a smartphone is. Finally, we got hold of Step Counter sensor released in latest Android phones. This sensor accurately (~90%) determines the number of steps taken by the user if he does not shake the phone in between. As this sensor is also based on accelerometer readings it is not fully free from issues as acceleration changes can occur even while user is not walking. But this was the most accurate we found until now and we

decided to go ahead with it.

In short, the problem is that mobile phone acceleration can be caused by multiple factors and not just by walking. So it is difficult to identify if the user is walking or not just based on accelerometer and other sensors available.

**Using constraints:** But using these sensors and constraints of the map we can estimate the user's location. As the user is walking on the path which we had shown, we know how much distance each path will take. And based on user's average step length we can find out how many steps would be required by the user to complete one path.
Below is the general outline of the algorithm to track the user based on step counter and map constraints.

Data inputs - Route consisting of multiple paths with each path having distance associated with it, Average Step length and Initial Values of Sensor

Two types of events will be raised by sensors one on step detection and another on rotation of phone. We are considering angle change of 45 degrees or more only as that will help in ease of implementation and at the same time is more realistic also.

1) For each step detection,
    a) Calculate distance traveled based on step counter and average step length
    b) If distance traveled less than or equal to "current path".distance then
        i) update GUI
    c) Else
        i) Ignore the event
        ii) And stop incrementing step counter and distance traveled
2) For each angle change (>45 degrees)
    a) If distance traveled less than "current path".distance then
        i) Ignore the event (in future ask user to pause tracking)
    b) Else
        i) Mark Turn

1) Reset step counter, distance traveled

2) Change current path to next path (from route or end if no more path)

3) Update GUI

## Summary of the challenges in tracking user using sensors

- **Difficult to distinguish between walking and other activities**

  **What we did:** As the user is walking the sensors might show us at some point that the user is at a position which is physically not possible. As outlined above, we are going to use constraints on the path a user can travel indoors.


- **Propagating errors of the sensors**

  The raw values provided by sensors are not usable because of errors. Although we can apply filters to make them usable, they are not fool-proof. There are bound to be errors. Even the smallest of errors have the property of propagating and causing problems.

  **What we did:** We tackled this by frequent resetting of values. The routes we provide are actually divided into small paths. These small paths are about 10 steps length. After the user completes a path segment we reset all the sensor values. What this does is, although there might be errors of one or two steps we do not allow the errors to propagate to the next path. We have got good results after testing this approach.

- **Difference in step length of different users**

  **What we plan:** We plan to calibrate the step length of a user using samples. When the user installs the app he will be asked to complete a simple path. We will know the distance and calculating the steps we can find out the step length. This is done in most of the pedometers you find in the market. So a pretty standard solution.

- **Walking side-ways**

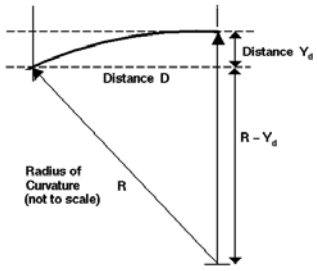
  Our current solution fails to recognize the difference between walking straight and walking sideways.

  **What we plan:** Try to isolate the accelerations in different directions. Thus ignoring the acceleration in the side-ways direction. Currently our filters are capable of finding the peaks in the accelerometer data to get the steps. So somehow we plan to isolate the accelerations in different directions. Thus we can ignore the values in the horizontal directions to counter this.

- **Curved paths**

  We also have to take care of curved paths.

  **What we plan:** Using geometry to determine the difference in the distance travelled in straight and curved path. Thus approximating the distance covered.




## Security and Privacy

From security we are mainly concerned about the map data of the building. We identified three places where we need to take some action

1) The map data which will reside on server should be in encrypted form and

accessible only from our application. The options is to use public key cryptography to authenticate the application requests on server.

2) Initially when the data is downloaded from the server to mobile phone we need to use SSL in order to ensure that no one can snoop on map data.

3) Once the user leaves the building, we need to erase the map data from the mobile phone.

With regards to privacy, we do not get any PII other than user location. And even that information will be in memory for the time the app is on and not stored to disk so once the user closes the app all the location information will be lost. The applications is not sending the location information to any other service. In fact, other than QR Code scanning to identify the start point and initial loading of database, application is not going to need network at all. And hence privacy issue will not be that important in this application. We will also come up with a privacy policy following FIPS guidelines.

## Usability Considerations

**1. Minimize start-up time**

As discussed earlier, we will pre-compute all the routes from all possible start points to destinations in that building. This will help in improving response time of the app which is critical in an indoor environment.

**2. User can make mistakes and the mistake should be fixable.**

Well we need to assume that the user can make mistakes. He can leave the path we give him and we should be able to handle this situation also.

**3. Cannot assume user's undivided attention.**

We want to give audio commands to the user. So even if he is looking anywhere else we can steer him using audio commands.

**4. Consistent and simple UI**

We feel our app will not require a complex UI. We need to modify the maps to be clean and user should be able to zoom. There should be a screen which clearly wants the user to enter a destination and then till the user reaches the destination not burden him with any other options.

**5. Things that look the same should act the same.**

We want to minimize the decision time by designing the buttons and instructions to be clearly understood with respect to navigation. For example a chequered flag for destination, directed arrows while navigating etc. following standard navigation symbols.

## Assumptions

1. User will be willing to scan QR Codes at building entrances and willing to enter the starting and destination locations.

2. User will follow the path as shown in the app and not deviate from it.

3. There is a map available for each building which the app supports.

4. User has latest android based smartphone (version 4.3 and above)

5. User will not shake the phone while using our app.

## App Design Changes

**Database related**

1) Earlier we modeled Path as consisting of start and end point only. But then we realized that paths are not just straight lines but can be of any shape. Hence to draw them properly on screen, we added another column to path table. This new column is called endpoints and is a comma separated list of all the points on map which fall between start and end point of the path.

2) Another change was related to routes_paths table. Here we stored the route and paths associated with it initially without considering order of the paths in the route. Hence while fetching them from the database we had to rely on the order in which they were returned (and assume that they were inserted in correct order). We have decided to add sortorder column to the routes table which will tell us which path is in which order.

**User Interface related**

Initially we created a new bitmap for every screen update. But as we were frequently updating the screen we decided to draw over the bitmap instead of recreating bitmap ever time. Also this approach was more consistent with the database we created and hence it was easy to integrate front and backend.

## Implementation Challenges

1) Updating User Interface frequently

When the use walks one segment of the route (i.e. path) we updated the screen to indicate where user is. But the sensor service running in background was constantly giving inputs on steps and direction which delayed screen updates and hence the entire route was getting drawn at the end.

2) Managing different versions of the sqlite on android phone

We used sqlite asset helper library to make it easy for us to change database and upload it to the android device. It allowed us to automatically copy the sqlite file from assets folder to the data directory of the phone

## Future development

1. User Interface Modification

After talking to users and customers we know the need to change the UI to be more user friendly and attractive. We have a good idea now what the users are looking for and we will implement the UI according to that.

2. More precise tracking

We will try to handle lateral movement, digression from shown route and walking backwards if feasible

3. Handling multiple floors

For this, we will extend the current logic of finding route. So a route across two floors will be comprised of minimum 2 routes - route one from start floor to nearest elevator. And then from elevator on end floor to destination. The only issue here is again in tracking the user i.e. to detect change in floor

4. Rigorous testing esp. of following cases

1. User walks sideways

This scenario considers that the user has to move sideways due to some reason. Our current case counts sideways motion as a step. As we have primarily considered sensor data as our source of step count we get the readings for sideways motion to be similar. So it adds steps when it shouldn't. We need to come up with a solid logic to avoid this.

2. User leaves the path

Currently we are not allowing the user to leave the path we have given. As our service would be a comprehensive navigation system, we cannot place a restriction on the user to follow only one path. He should be free to explore other paths. We should be able to detect this and if possible re-route.

3. User waits for a long time at a particular point

We have handled this scenario to some extent. However we need to test this more rigorously. If the userehensive navigation system, we cannot place a restriction on the user to follow only oneuse the program to stop and start again which will cause all state information to be lost. We need to be able to recover from these errors.

4. User moves backward on the same path

We have to consider the case where the user turns back and moves on the same path backwards. This should update the travelled path. We have not considered this case. We have orientation to be highly accurate. So this case should be possible.

## Development Tools




**Bitbucket.org**

Bitbucket is a web-based hosting service for projects that use Git revision control system. We used it to collaborate our code.

**Eclipse**

We used eclipse as our Interactive Development Environment (IDE), since it provides very good support and tools that makes programming and Java very easy.

**Android SDK**

This is a software development toolkit provided by Android which can be used to develop applications for Android platform.

**Java**

We wrote all our code in Java, which is programming language for Android development.

**Sql Lite**

We used Sql Lite as our database to store all the map related data

## Appendix

Poster