CLOUD COMPUTING PROJECT
IOT GROUP G-2

INTERNET OF THINGS ENABLED SMART BUILDING SYSTEM PROTOTYPE

INSTRUCTOR:

I. L. Yen

NAME		NET ID	Spring 2014

1. ARUN KUMAR RAJAPPAN : @utdallas.edu
2. DEEP SHAH : @utdallas.edu
3. NAVYA PADALA : nxp131930@utdallas.edu
4. VEERA VENKATA RAVI TEJA ARRABOLU : vxa132930@utdallas.edu

TEACHING ASSISTANTS:

1. Guang Zhou

2. Jinwei Yuan

CS 6301.501

INDEX

INTERNET OF THINGS

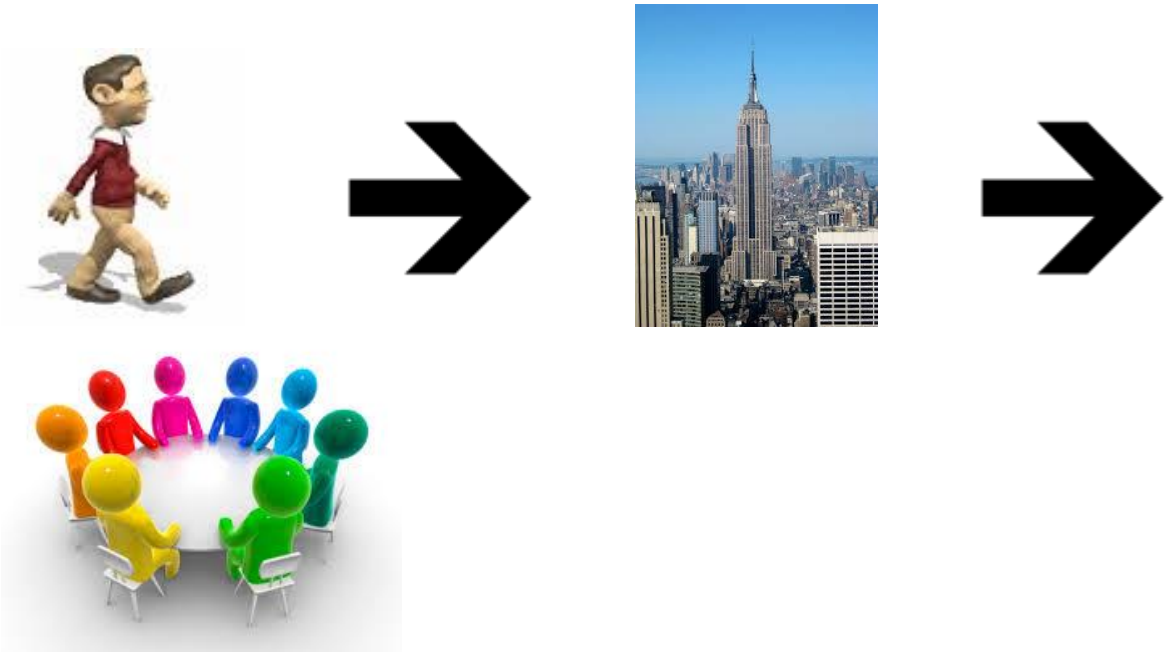
1. INTRODUCTION

The Internet of Things (**IoT**) refers to the ever-growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other Internet-enabled devices and systems.

2. MOTIVATION

In today's fast pacing business world the crave to deploy new technologies and successful models to smoothen the business flow and lure the end user with the finest art form of technology has driven the need for innovation and internet related technologies so that applications could cater to the need of customers connected globally to the internet. Here everything device has access to the Internet.

3. PURPOSE



1. User attends an appointment in a building to meet his host for a meeting.
2. Our standalone application at the host building verifies the visitor's identity and validates the meeting schedule.
3. Our smart phone application will aid indoor navigation.

4. WHY ANDROID

- **Android based on Java.**

  As android is a mobile application development framework based on Java. It is compatible with mobile, tablet and even desktop.

- **Android Community**

  Android has a large developer community so we had good reference to all development challenges we faced during the application development.

- **Android Market**

  Android has a huge market share of about 50% as per Yahoo answers. So our application could be reachable to huge number of end user's. This was our main motivitaion.

- **Android Support Google Frameworks**

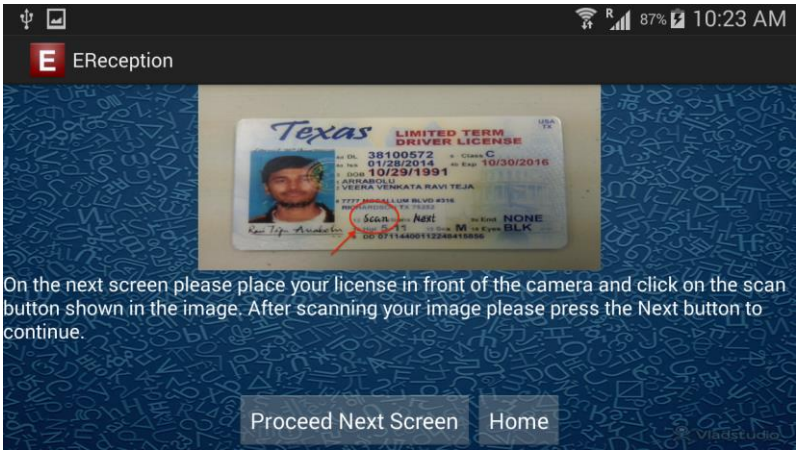
  Android is a framework that uses Google services by default. Google enabled services like Google App Engine, Google Calendar, Google Voice and Frameworks to support Cloud service integration.

5. BACKGROUND STUDY

5.1 OPTICAL CHARACTER RECOGNIZATION

EXPLORATION:

- We had explored for many face recognition apis listed at the site mentioned below
  - http://www.ocr-it.com/document-conversion-service/corporate-litigation-support-ediscovery-pricingpricing
  - http://ocrsdk.com/plans-and-pricing/
  - http://ocrapiservice.com/rates/
- Out of all the apis we explored we found the cloud based service provided by Abby's Cloud based OCR service to be better and accurate. Hence we decided to use its SDK for implementing face recognition
  - http://ocrsdk.com/

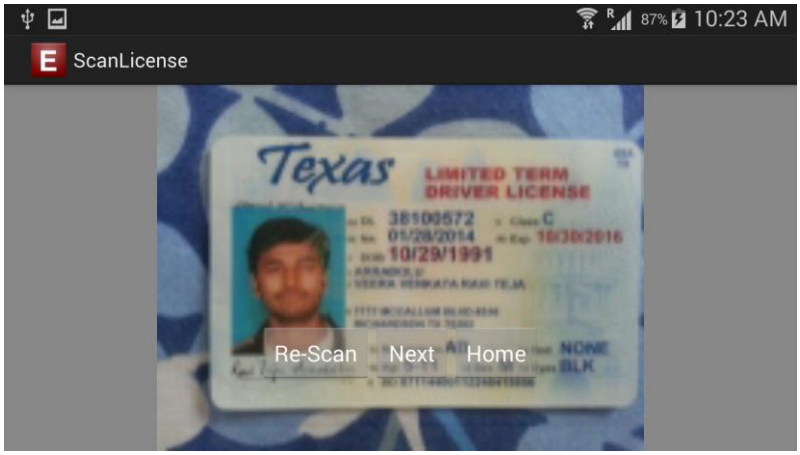


IMPLEMENTATION:

- We implemented an camera based application to take the picture
- Once the picture is captured we used an Bitmap compression algorithm to compress the captured image to KB of size, so that its caused less network overhead while calling the cloud service
- We used the Abby's OCR SDK to call the cloud service and process the JSON result returned by the service. If the process was successful then we would navigate to next activity

ISSUES:

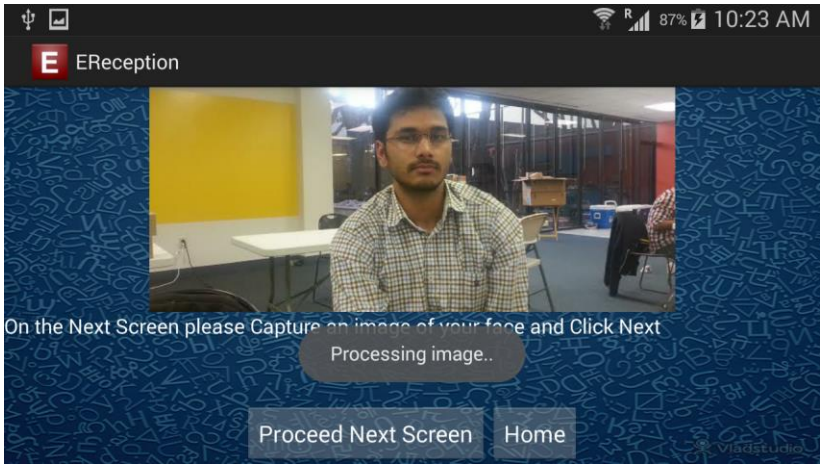
- Initially we were supposed to get the license scanned via scanner but due to compatibility issues of scanner with android devices we had to fall back to camera based solution
- To get a better results from OCR we required a high pixel camera for scanning the license
- Due to the effect of surrounding like lightning issues etc we need to process the image with Image processing algorithms for blurring, gray scaling, light balancing etc before sending it to OCR service for processing.



5.2 FACE RECOGNIZATION

EXPLORATION:

- We had explored for many face recognition apis listed at the site mentioned below
  - http://blog.mashape.com/post/53379410412/list-of-50-face-detection-recognition-apis
- Out of all the apis we explored we found the cloud based service provided by FacePlusPlus to be better and accurate. Hence we decided to use its SDK for implementing face recognition
  - http://www.faceplusplus.com/term-of-use/
    https://github.com/FacePlusPlus

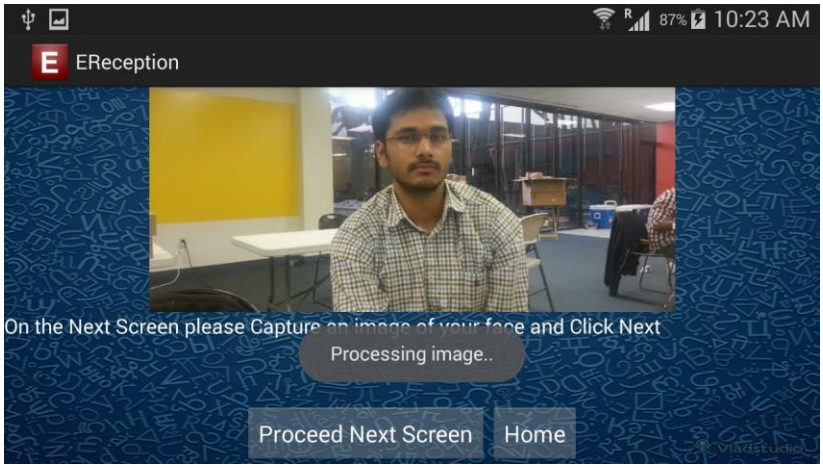


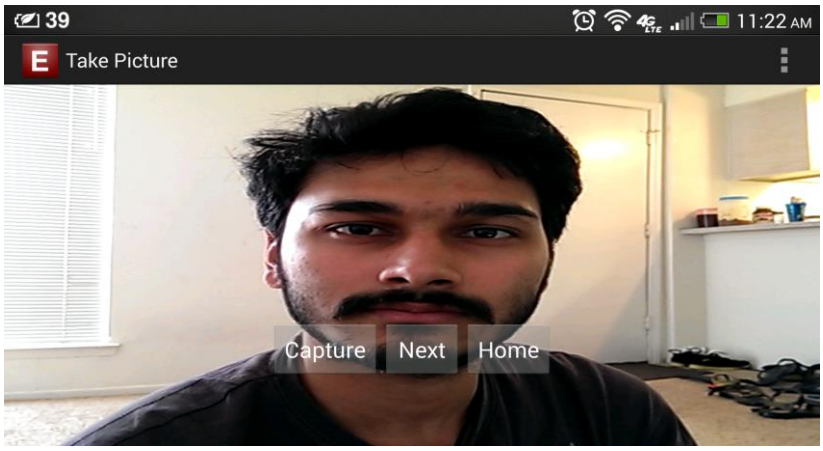
IMPLEMENTATION:

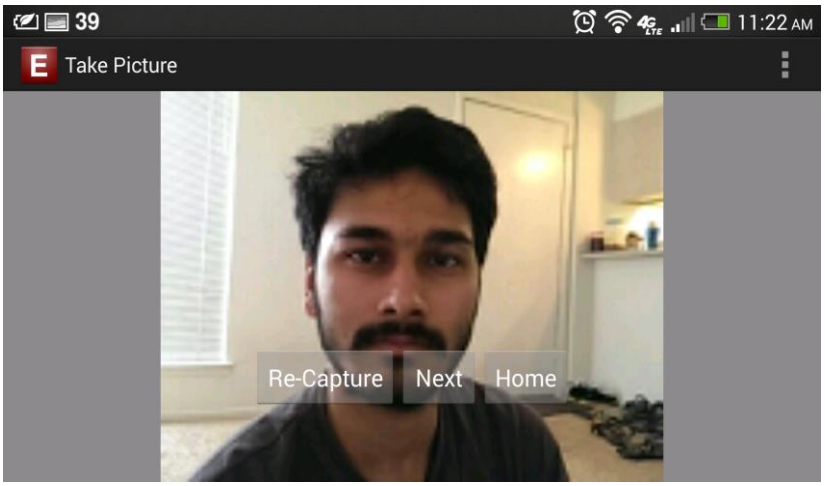
- We implemented an camera based application to take the picture
- Once the picture is captured we used an Bitmap compression algorithm to compress the captured image to KB of size, so that its caused less network overhead while calling the cloud service
- We used the Face++ SDK to call the cloud service and process the JSON result returned by the service. If the process was successful then we would navigate to next activity

ISSUES:

- Due to the effect of surrounding like lightning issues etc we need to process the image with Image processing algorithms for blurring, gray scaling, light balancing etc before sending it to Face Recognition service for processing.







5.3 SPEECH TO TEXT & TEXT TO SPEECH

EXPLORATION

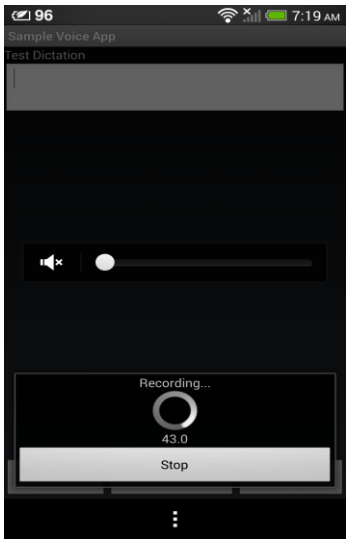
- The use of the **Speech to Text** and **Text to Speech** services in android has been crucial for verification of user details when validating with the Google calendar for scheduled appointments of the visitor with any host in the building.
- The user after successfully passing the initial stages of verification with the OCR module, and the Face Recognition module, validation of details would be backed by final scrutiny of details by the user himself/herself.
- The details displayed for the user are the fetched OCR scan results of the driver's license ID of the visitor.
- In order to implement the best reliable service for **Speech to Text** and **Text to Speech,** the following have been explored:

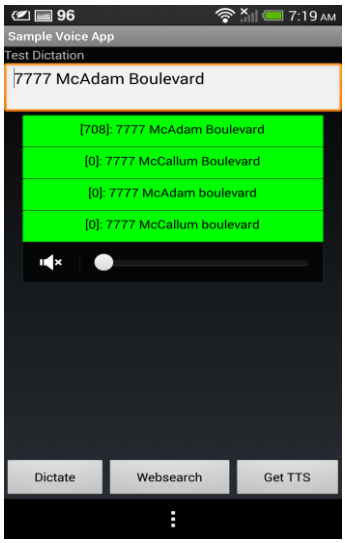
  1. NUANCE NDEV android SDK
  2. Google Voice android SDK

1. **NUANCE NDEV android SDK**

   - Nuance android application development SDK provides the services based on REST API implementation and user API keys.
   - However, it had features useful, it was not a reliable service as the results were containing errors.

- It had features to use both the Text to Speech and Speech to Text.
- **Screenshots**: The above screenshots show the NUANCE dragon based android application.





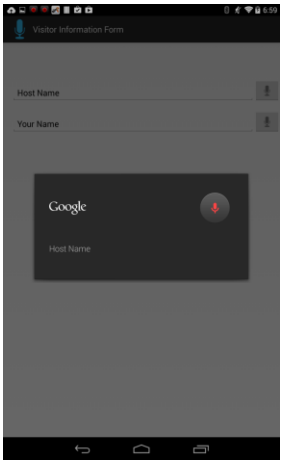
- The left screen indicates the voice recording being processed by the SDK, and the right indicates the results being displayed by the NUANCE after it is being processed by the NUANCE cloud service being called by the REST API.
- The result for the address field was tested as shown above.

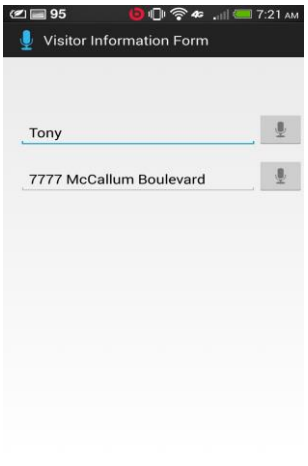
Issues:

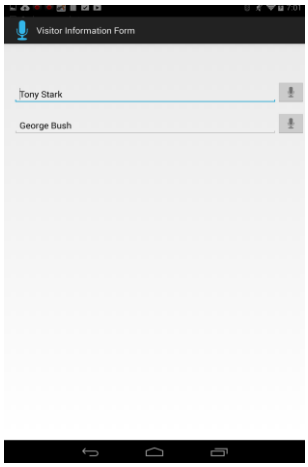
- After several tests were performed, it has shown very unreliable performance in delivering the right text for the speech dialogue.
- It also had problems during recording the user's speech dialogue, and failed to record some times, which indicated poor quality of service for free developer cloud service.
- The service assured good quality of service for paid version, and offered additional features like voice based commands assistance, which can be used for customizability of the application.
- However, the Text to Speech offered accurate results but it could not be a threaded process.
- As enabling it as a thread throughout the application lifetime in the background causes an overhead for the transmission of data for other modules (Face++,OCR).

2. GOOGLE Voice android SDK

- The best possible implementation for the speech to text and text to speech services available for android.
- Android has special intents that have native frameworks supported by Google's server for speech to text and text to speech.
- Speech to text has been implemented as a Service while the Text to Speech has been implemented using the local TTS engine on the android Smartphone, pertaining to the thread conditions.
- For smooth user interaction we had threaded the services in the android application such the User does not need to wait for an activity to complete to go to the next activity.
- Threading of TTS would be required during whole lifecycle of the process, so it causes an overhead for the other services which are running in threads.
- Initial tests for the Google based android application gave successful results.
- **Screenshots**: Below indicated is the same scenario tested for address, visitor name.

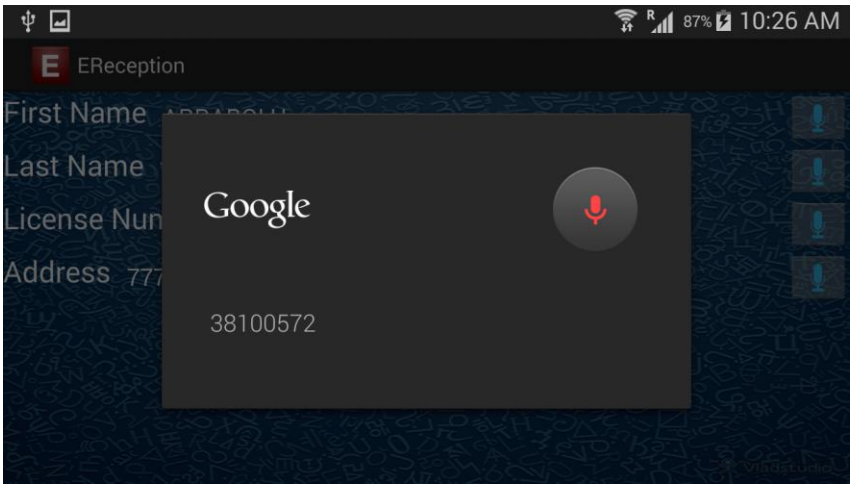




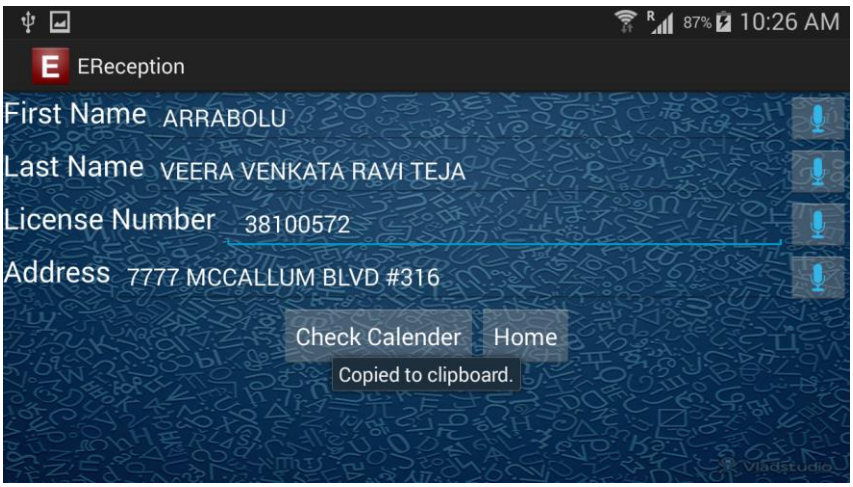


IMPLEMENTATION

- The Google Voice based OCR details validation form has been integrated into the E-reception application.
- It fetches data from the OCR service and the listed details of the user can validated by the user here by giving voice dialogue inputs to make modifications.
- **Screenshots:** The following screenshots provides the scenario of the visitor's scanned license ID details, which can be modified using the voice input. Here the license number of the user is being modified.



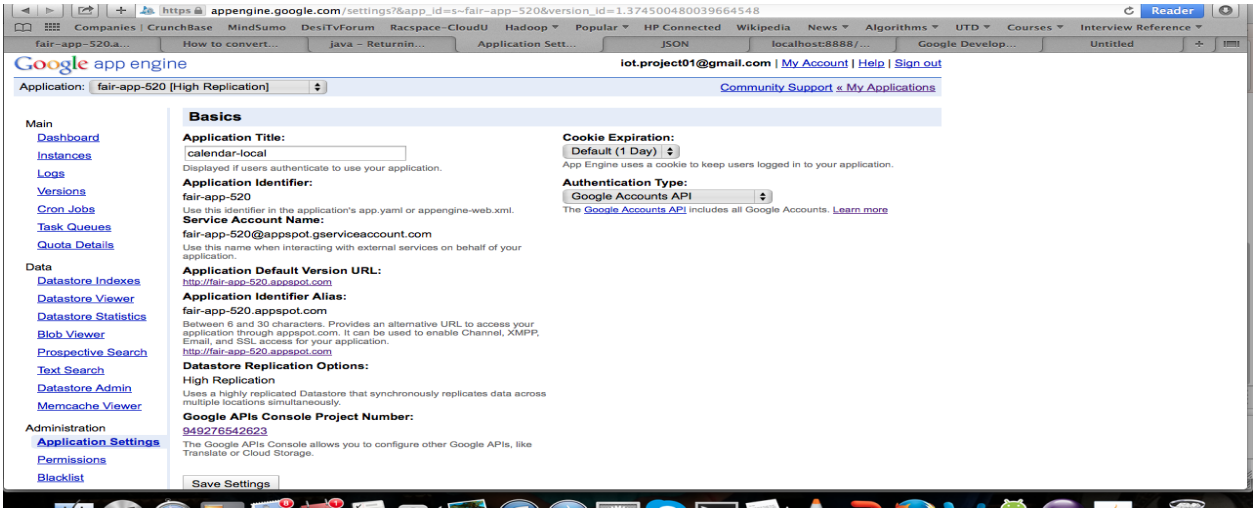
Screen after validation of details, clicking check calender proceds to calender event verification.



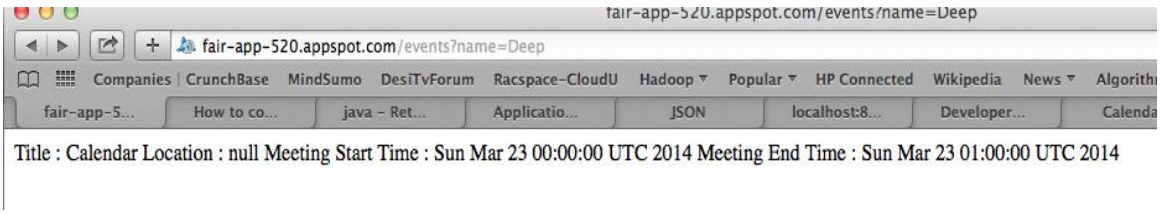
5.4 CALENDAR (GENERATION AND VERIFICATION)

EXPLORATION: As per our requirements we had a need to use Google Calendar Service and byfar the best service for Scheduling Meetings and Verifications. It was also natively supported by android, hence we implemented Google. Initially we intended the service to be web application so we developed a servlets that can insert events and can verify events.

- Using Google Calendar to Make Appointments and Verifying Visits
- Implemented Using Google Calendar API Paramters.
- Hosted on Google App Engine : http://fair-app-520.appspot.com/

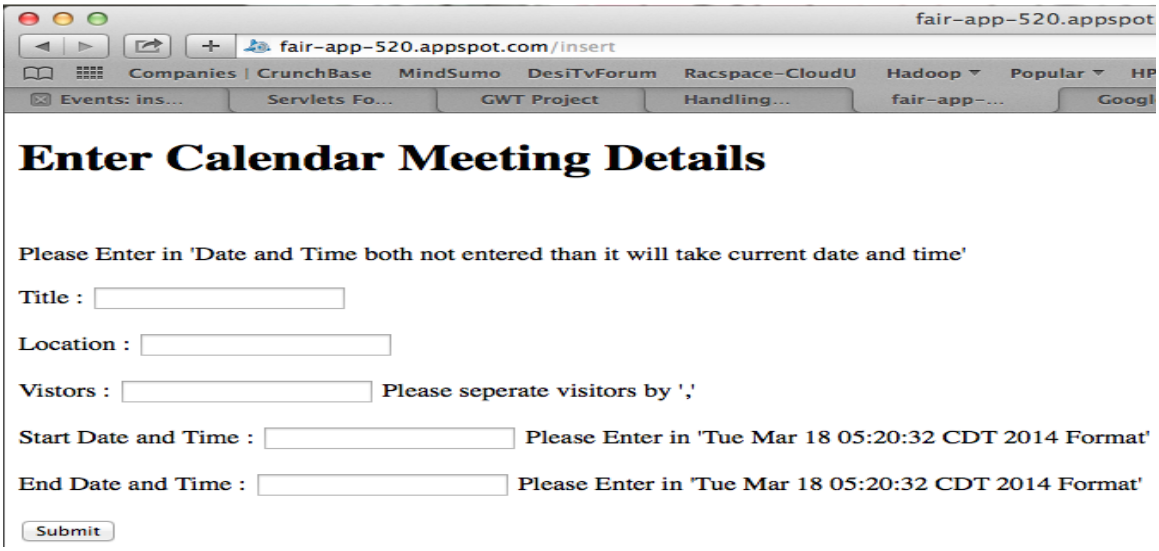


Verification of events :



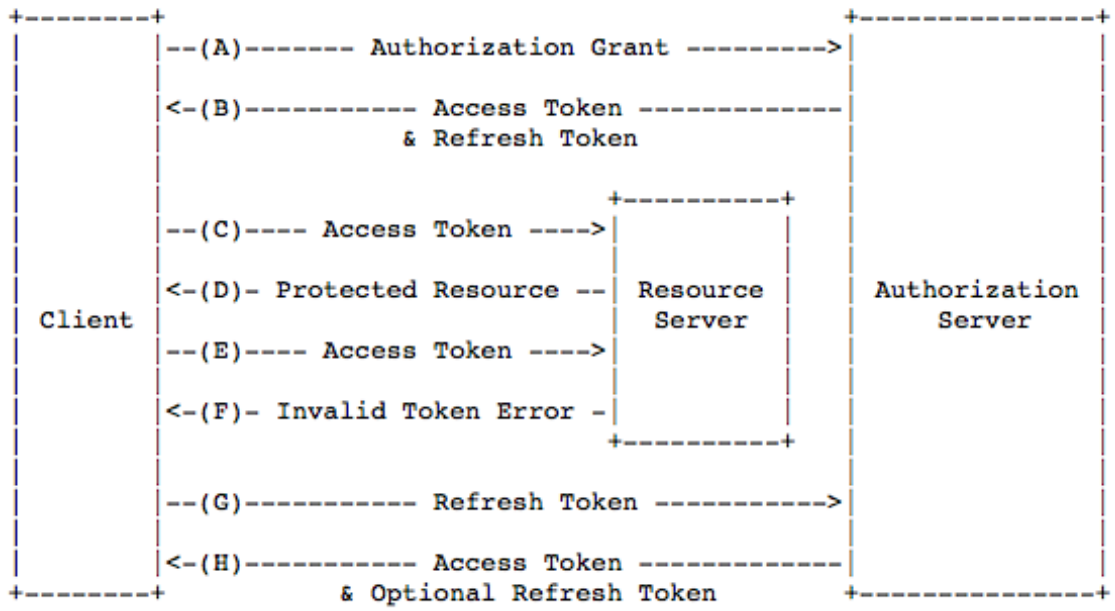
- Vistor's name from the OCR will be taken and in return he will be provided with list of all meetings he is going to attend.
- Vistor will select anyone event from the list.

Meeting Insert :



ISSUES :

In order to avoid unnecessary user and redundant user interaction to fetch credentials during the use of the application we implement the OAuth 2.2 Authentication.



OAuth Work Flow:

The flow illustrated in Figure 2 includes the following steps:

(A) The client requests an access token by authenticating with the authorization server, and presenting an authorization grant.

(B) The authorization server authenticates the client and validates the authorization grant, and if valid issues an access token and a refresh token.

(C) The client makes a protected resource request to the resource server by presenting the access token.

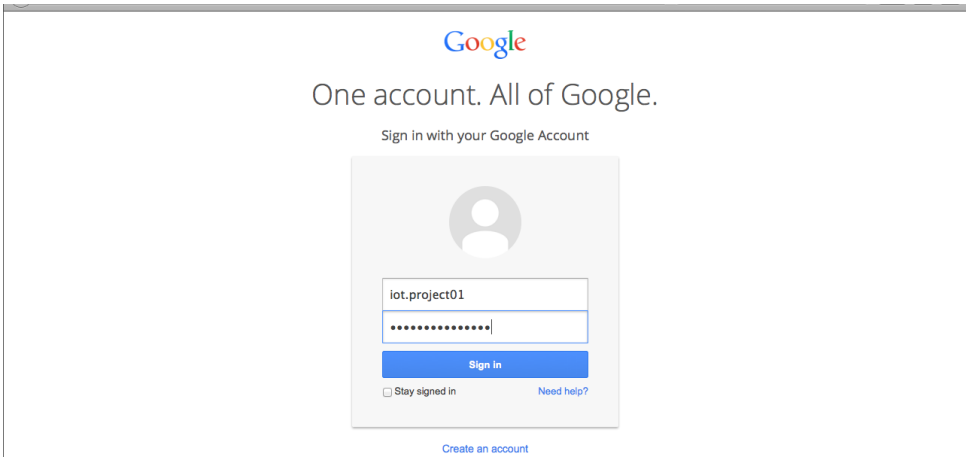
(D) The resource server validates the access token, and if valid,
serves the request.

(E) Steps (C) and (D) repeat until the access token expires. If the client knows the access token expired, it skips to step (G), otherwise it makes another protected resource request.

(F) Since the access token is invalid, the resource server returns an invalid token error.

(G) The client requests a new access token by authenticating with the authorization server and presenting the refresh token. The client authentication requirements are based on the client type and on the authorization server policies.

(H) The authorization server authenticates the client and validates the refresh token, and if valid issues a new access token (and optionally, a new refresh token).



IMPLEMENTATION: As we moved from Web to Android. Now we have integrated the calendar services into our process framework with the aid of developing 2 applications. Each has been utilized to deliver specific user (Host/ Visitor) requirements.
The 2 applications are :

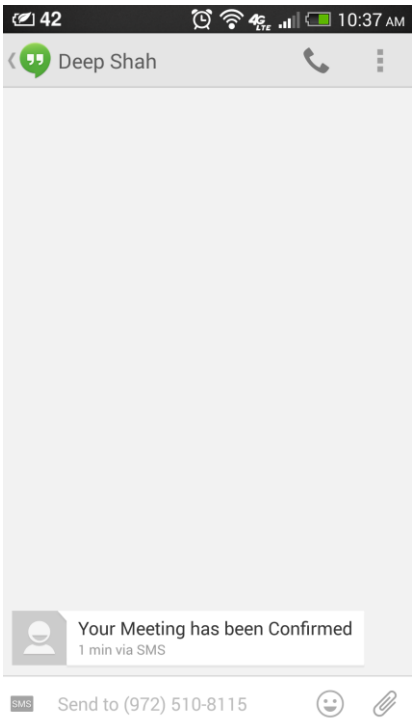
1. Calendar Meeting Scheduler for the host
2. Integrated Module for Calendar Event Verification

1. **Calendar Meeting Scheduler for the host:**
   The host uses this application to schedule meetings for his guest(s). Once the host(s) schedules the meeting. The guest(s) are notified about the meeting schedule and details by an SMS.

- Host has to insert following fields inorder to Schedule a Meeting:
  a. Location

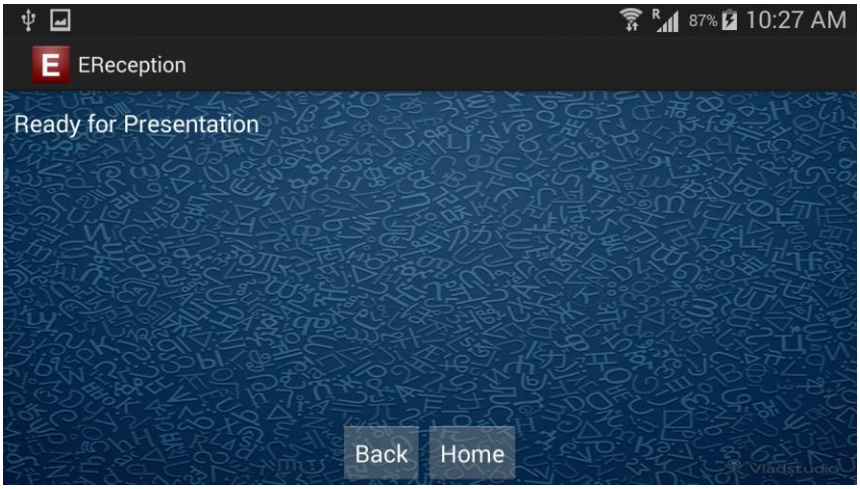
b. Start Date and Time
c. End Date and Time
d. Invitees
e. Mobile Number

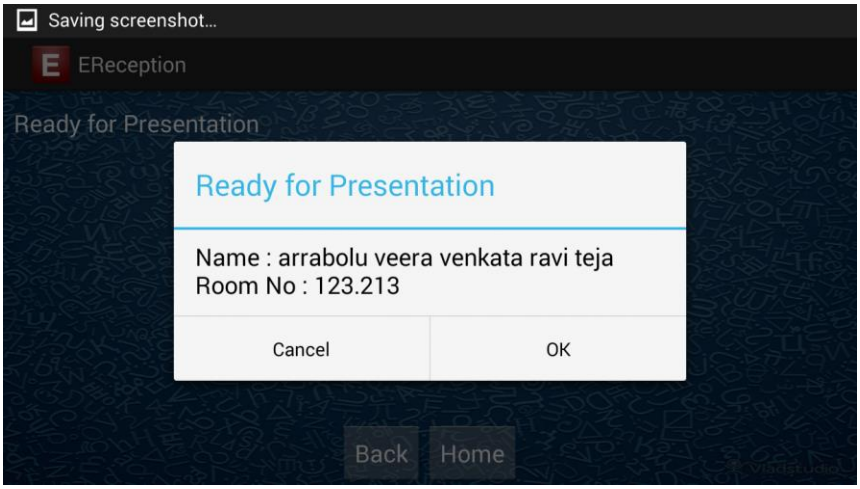
- SMS Alert to User if the event is Scheduled



2. Integrated Module for Calendar Event Verification:

After the details have been validated by user himself. They are been passed for verification they are being sent toGoogle Calendar using Google Calendar API (v3). On Verification it lists all the Meetings where user has been invited. Clicking the meeting user can see the location. Finally when accepts the meeting QRCode is generated.





5.5 QR CODE (GENERATION AND READER)

EXPLORATION:

- In order to implement better security functionality and M2M data transfer we have utilized the QR code.
- The QR code for authentication is an approach which is currently a best practice.
- However the code can be readable by any QR reader, the security provided by the QR code is still better as details like front gate authentication pass phrase to enter the building can be passed through the QR code.
- The QR code is generated locally at the device using the cloud API from http://goqr.me/
- The API provided utilizing the REST API calls to the interface with our parameters(Room Number), and display the code generated to the user, so that he/she can scan it using our smart phone indoor navigation application, and proceed for indoor navigation.
- QR code has been utilized in two phases:
  1. QR code Generator
  2. QR code Reader

1. QR code Generator

- ❖ The generation of the code is implemented as a result of the data obtained after the verification from calendar service which gives the user destination's details.

2. QR code Reader

- ❖ The scanned code details enabled the Smartphone application to plot the path depending on the room number scanned.

- The cloud services available for QR code generation was numerous but two of them were tested for implementation.

  1. Google Charts API
  2. QRServer API

- However both were implementable, QRServer API was simpler and did not need authentication and complicated integration as the Google Charts API needed.

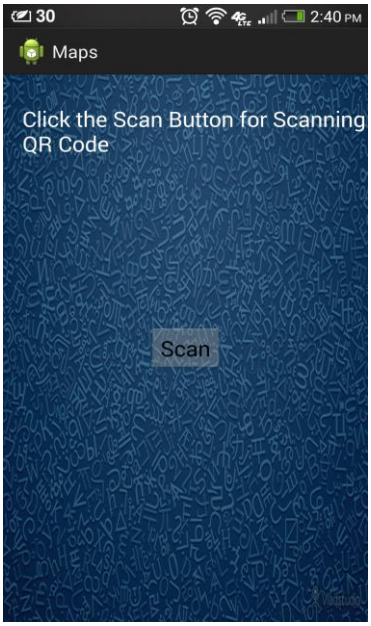
Implementation:

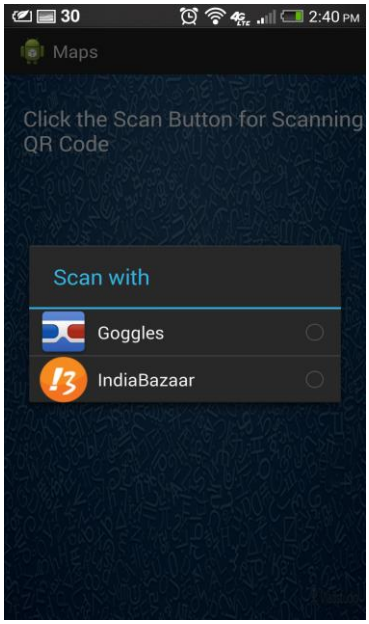
QR Generation:

- Screenshots: Below is the implementation of a QR code generated for a room number.

QR Reader:

- The QR reader integrated in the Smartphone application uses the ZXING open source library which depends on an external camera scanner application support.
- The Google Goggles application is being called during the invocation of the scan process to aid the scanning of the QR code.
- Screenshots: Below is the scanning application calling the aid of another installed Goggles application for scanning the code.



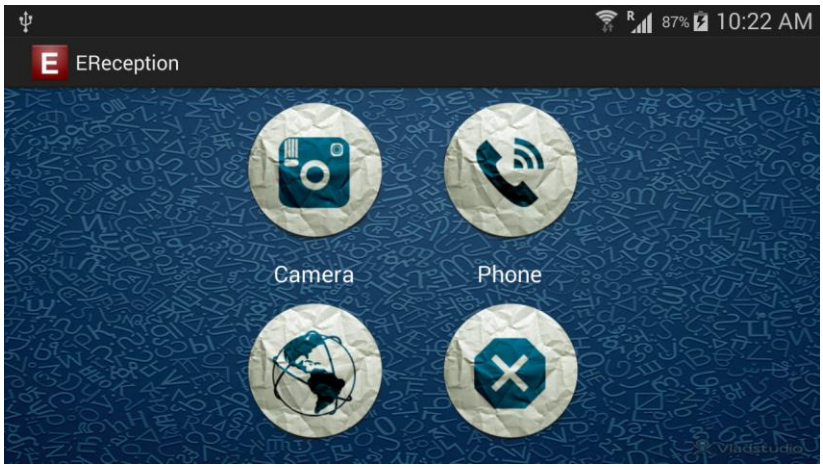


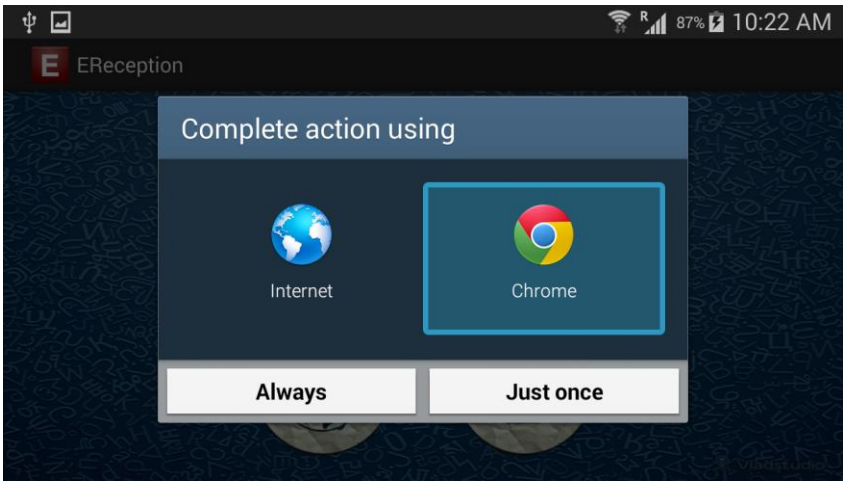
5.6 UI DEVELOPMENT AND INTEGRATION

ISSUES: UI Integration was a tedious task. Each Mobule was implemented independently hence understanding their code and integrating with the current flow was crucial task. Each service was having its own threads. In order to provide smooth user interaction we had to put service threads in background for synchronization and calling other UI Activities in front. Also to start and stop Text to Speech when user changes the acitivity in between.

Along with this we alos provided features :

- Main Menu
- Call For Help
- About Us





5.7 BLUETOOTH (BLE)

EXPLORATION

- In order to implement the maps we have considered exploring two Bluetooth technologies existing to extract the best results.
- The Bluetooth signal strength measured is termed as Received Signal Strength Indication (RSSI) which defines the strength of the signal measured by the device (Smartphone) with respect to the Bluetooth beacons.
- This RSSI plays a crucial role in determining the person's positioning indoors and aids in navigation.
- In order to better understand about the positioning we had considered the following relevant papers, and had adopted it.

1. Development of an Android APK for Bluetooth Localization - Carlos Carrera Carbó, Czech Technical University in Prague.

2. A Smartphone Localization Algorithm Using RSSI and Inertial Sensor Measurement Fusion - William Wei-Liang Li et al.

- The paper for Bluetooth based android APK development served as a guide in helping to understand how the RSSI is useful in utilizing the positioning of the person.
- However, the Smartphone Localization paper seemed less feasible when compared to the prior.
- The Bluetooth technology has diversified recently which is the Bluetooth prevalent in the devices having Bluetooth<4.0 and Bluetooth>4.0 version
- Sampling can be done with both the Bluetooth.
- However, the Bluetooth above version 4.0 is Bluetooth Low Energy which is considered to be more efficient especially for indoor navigation applications.
- We have sampled RSSI with both Bluetooth version based applications for understanding the differences.
- Both delivered accurate performance, but BLE delivered stable results, and shown significantly less battery power drain.

Screenshots: Shown below is a traditional Bluetooth implementation for sampling the RSSI values.

5.8 MAPS API

Exploration

- Maps have been a crucial aspect for the indoor positioning.
- We had considered four different approaches to achieve an appropriate map for our application.

1. Open Street Maps for Android and creating indoor maps on Sails cloud

2. Google Maps API
3. Canvas Based Maps
4. Image based plotting Map

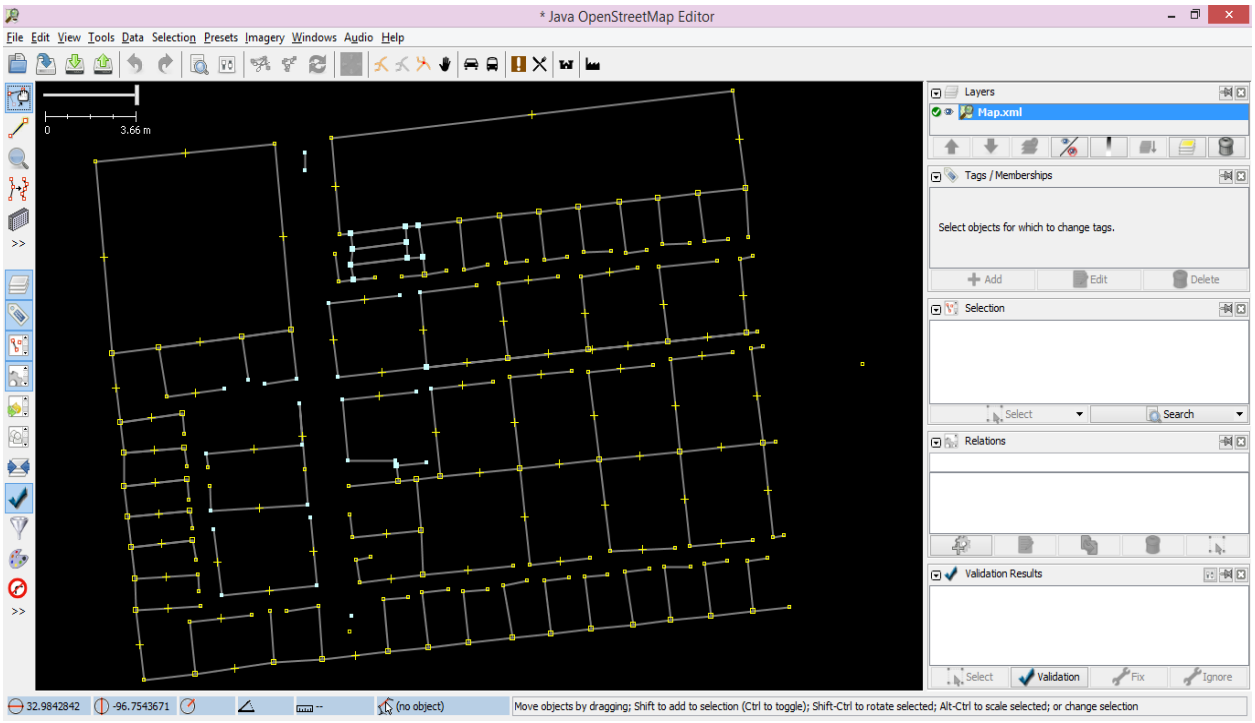
1. Open Street Maps for Android and creating indoor maps on Sails cloud

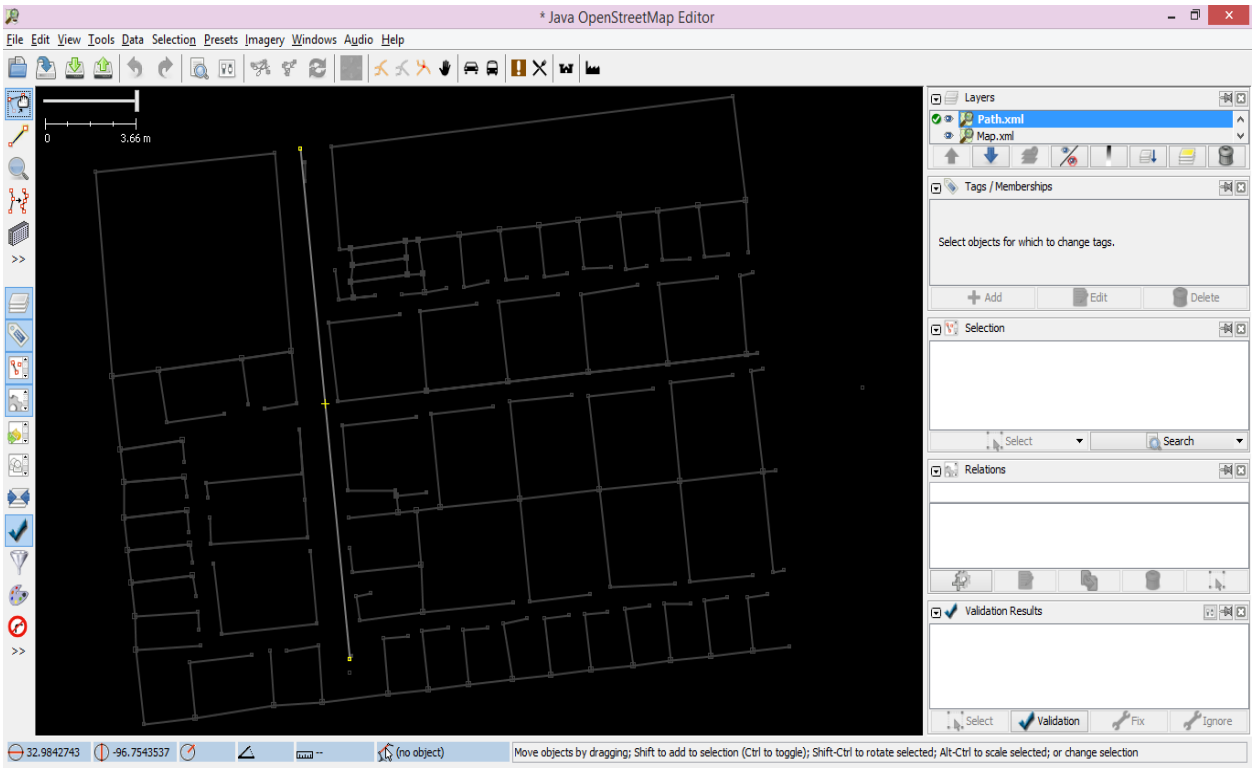
Exploration:

- The approach was based on using OSM maps and creating the own indoor map.
- OSM maps were integrated with the apps from Sailstech to upload into their could after development.

Implementation:

- Using JOSM editor, created an indoor map and the path routing files.
- Created building project in SAILS cloud and uploaded the created map and path routing files.
- Rendered the hence created maps using the Map Render Editor from Sailstech.
- Used SAILS BuildNGO to record the RSS finger print data.





Issues:

- The indoor map and path routing files were XML data about the node's latitudes and longitudes in the map.
- The Maps navigation which is used in BuildNGO wasn't bluetooth based.
- It was an unsuccessful attempt to use the XML data with other navigation techniques. Mapping of XML data back to an image data was not feasible and hence we chose the Google Maps API.

2. Google Maps API

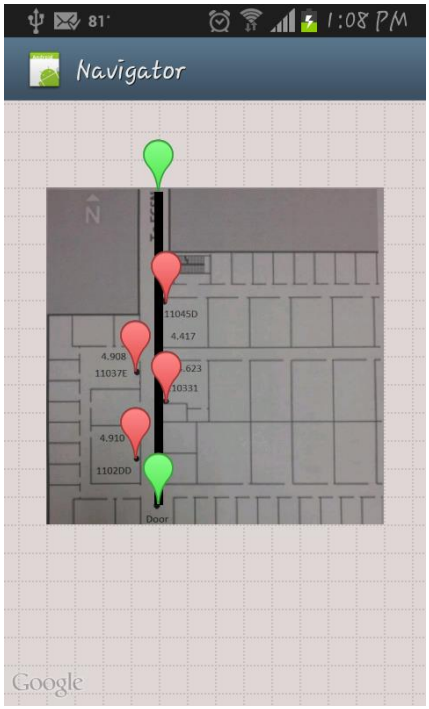
Exploration:

- Google Maps Android API V2 was used to build the maps.

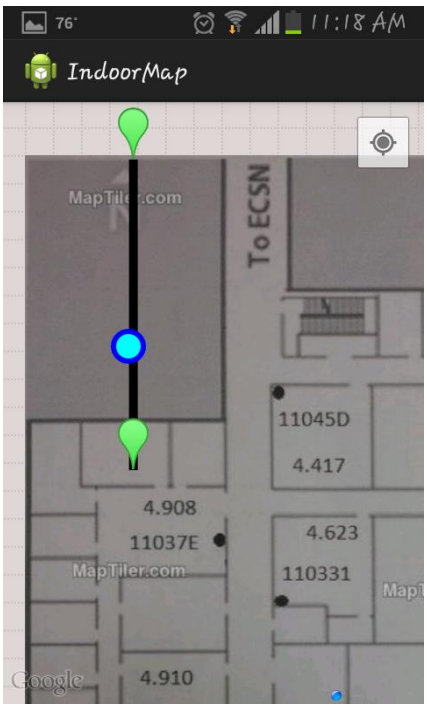
Implementation:

- TileOverlay feature of the google maps API was used mainly for building the custom map.
- Declaring the map_type of the google maps as 'none' was the crucial step in implementing the custom maps.
- The map image is divided into multiple tiles according to the zoom level.
- Zoom levels from 0 to 4 are implemented for the zoom in and zoom out of the map.
- Positioning of the markers was done using the markers in the API and the path was drawn using the polylines feature of the api.
- Camera positioning was also implemented so that when the map opens, the camera is positioned to the start position of the map.

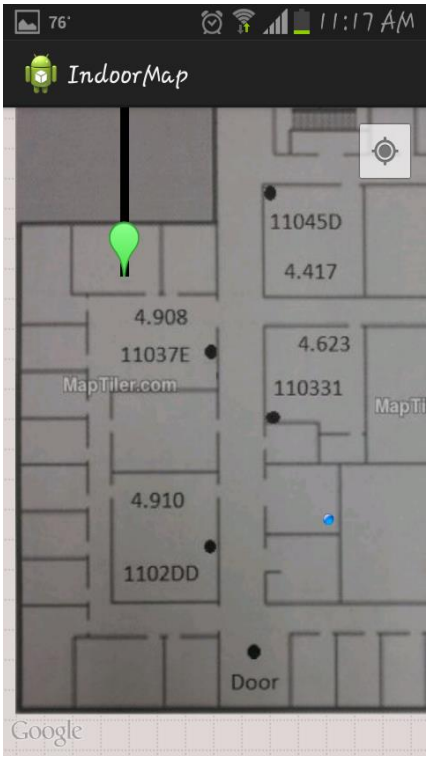
- We faced an issue of tile being repeated horizontally as we scroll towards left or right of the map. This was default set by the google maps as the world map is shown like a spherical map in google maps. This feature was not changeable.
- To overcome the above problem, implementation of Box bounding was done to not allow the map to go out of bound of the custom map. Box bounding helps in getting back the map on the screen even if scrolled.
- The start and destination markers were set using the latitudes and longitudes of each tile.

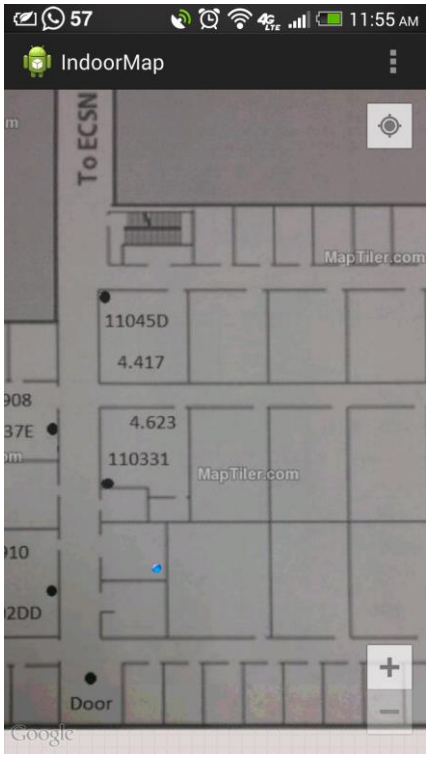


First version of google maps with markers

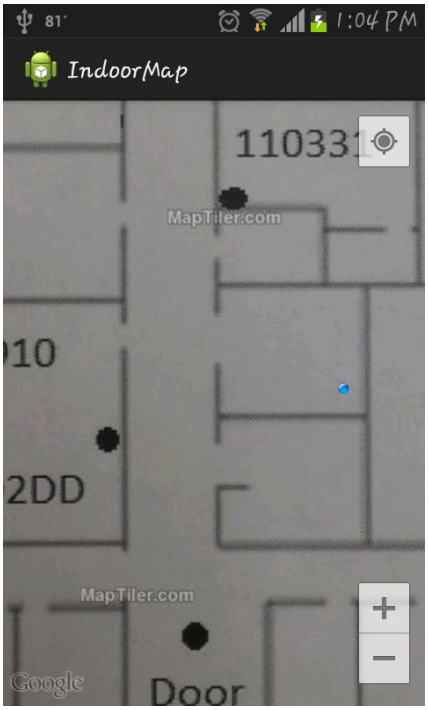


Markers using lattitude and longitude





Scrollable map with markers and path



Zoom level 3

After Box bounding of the Map at zoom level 2

Zoom level 4

Issues:

- As the tile overlay is used on top of the google world map, latitudes and longitudes are to be specified for each point on the map like start, destination, beacon points, etc.
- Finding the latitudes and longitudes was a difficult part for each point.
- Another issue faced with the google maps was the navigation, simulation was possible but the proper navigation wasn't possible even after tiling the image.
- The image was repeating horizantally due to google's unchangable default settings of spherical map. The solution of using box bounding wasn't totally perfect as it wasn't working similarly on all the devices.

4. Image Based Map Plotting

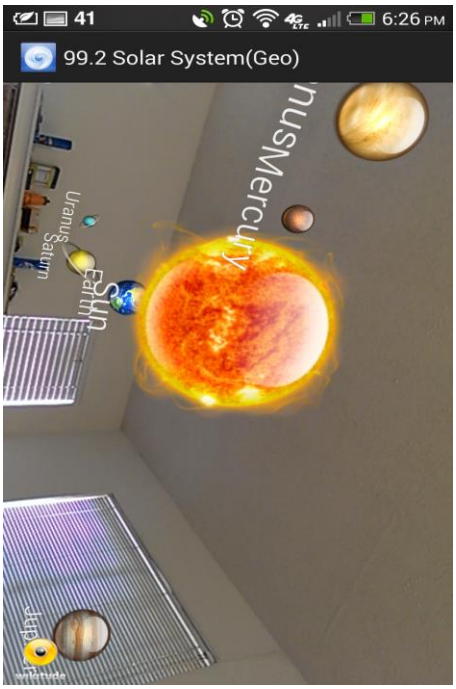
Exploration:

- The approach was based on implementing using either Image View or Virtual Reality.
- However, we had implemented the Image View based map plotting.

Virtual Reality based Map approach

- To implement virtual reality the following SDK has been explored, provided by Wikitude.
- However, Metacio is another SDK provider which does support the same but has limited documentation for working on complicated applications like maps unlike Wikitude.
- Both offered free "Cloud based Virtual Reality service for developers".

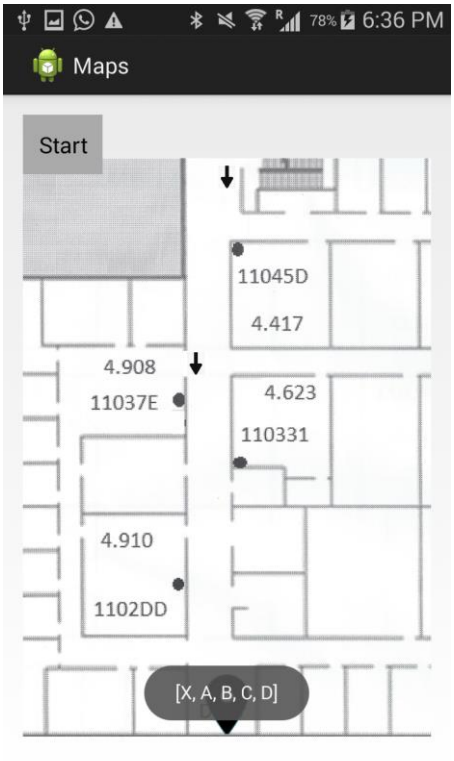
- Wikitude has showcased a custom Virtual Reality Navigation application based on GPS, and Triangulation for outdoor positioning, named "Navigator" which was the motivation for choosing wikitude.
- The possibility of implementing a Virtual Reality based View Map instead of the Image View would be feasible.
- But we had limited resources about how we can make a tangible outcome considering the positioning factor using the Bluetooth RSSI.
- Screenshots: Left image show utilizable the point of interests for map plotting and finding user's position. Right image shows solar system overlaid on camera view, and in a similar way the path obtained after djikstra could be plotted here to the camera view.
- A similar approach to how wikitude Navigator had been implemented for GPS navigation.

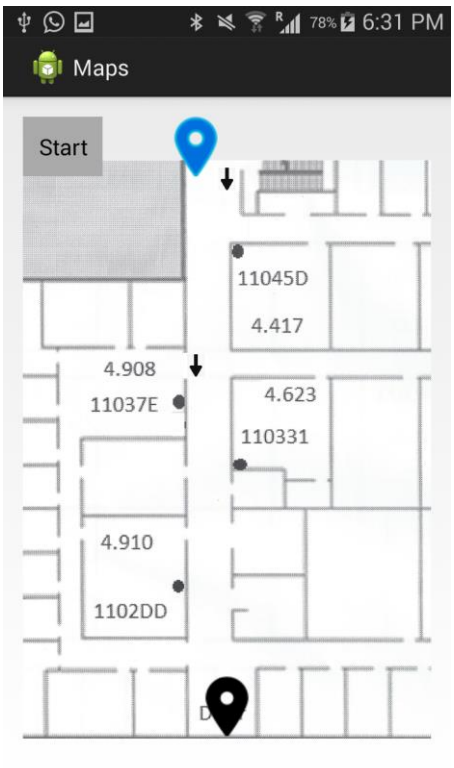


Implementation:

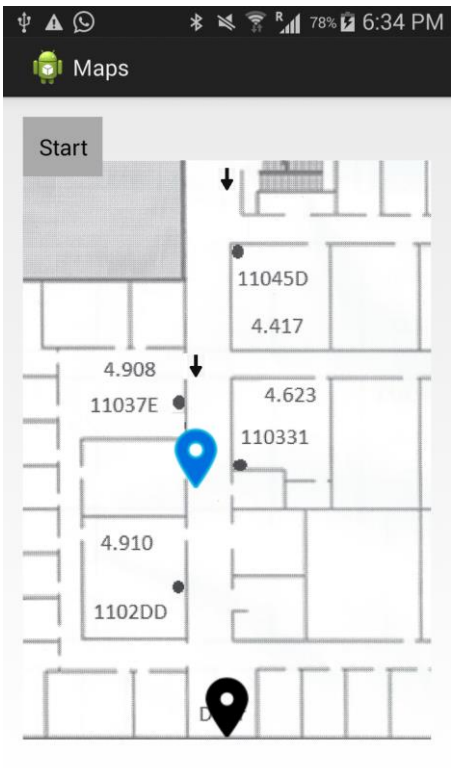
- We considered the map to be an image uploaded in the Image View using the android intent and had plotted it with image markers to show start and destination positions.
- Our approach was based on dividing the Map into zones and finding the position of the user relevant to whichever zone proximity closest the user falls in.
- The Map is divided into phases and sub phases.
- A total of five phases till his destination point in this case.
- Each phase has two sub phases which determine if the user is entering the phase or exiting the phase.
- The Map contains paths to different locations, so we made the map locations as nodes and used a connected weighted graph to plot all connected paths.
- We have implemented the path generation using the **Dijkstra** Algorithm to find the shortest path, and the resultant path is plotted on the map.
- To show the resultant path on the map we show the path using arrows.
- Before we plot the map, we show the path on the screen with a toast message showing the path from which zones it is connected.

- The destination is announced to the visitor and also the turns in the path when he approaches a turn.
- **Screenshots**: The first image shows a toast of the path calculated to the destination D from source X. The images in the sequence show our approach of arrows indicating the path directing to the destination.

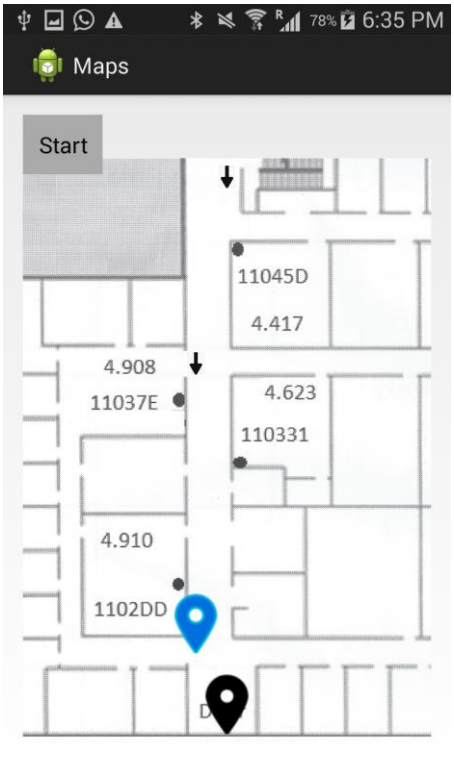



78% 6:31 PM
Maps
Start
11045D
4.417
4.908
11037E
4.623
110331
4.910
1102DD


Saving screenshot...
Maps
Start
11045D
4.417
4.908
11037E
4.623
110331
4.910
1102DD


78% 6:34 PM
Maps
Start
11045D
4.417
4.908
11037E
4.623
110331
4.910
1102DD


78% 6:34 PM
Maps
Start
11045D
4.417
4.908
11037E
4.623
110331
4.910
1102DD



Issues:

- The possibility of implementing the maps on Image View had difficulties in plotting the path calculated by the Djikstra algorithm.
- The path co-ordinates could have been plotted using polyline functionality of Google maps if this could be employed with the API.
- However, Google Maps had complicated flow scenarios and complex mapping functions based on longitudes and latitudes but with limited resources about how they can be utilized, the implementation would be very difficult.

5. Map Navigation using Android Sensors (compass, accelerometer and step counter)

Implementation:

- We loaded the floor map on a ImageView/Canvas
- Now instead of just using the BLE proximity logic to detect user position we are using compass, accelerometer and step counter android sensors for real time positioning of a person inside the building.
- This approach smoothens the tracker motion on the map (instead of having sudden jumps in the position)
- In this the user location continuously updates as the visitor is on move and whenever he is close to any BLE device it announces that BLE device is nearby.
- Just like the previous canvas version we will be scanning the source and destination based on the QR code and will determine the path using Dijakstras algorithm
- For calculating the distance from the BLE device we took serveral samples of RSSI reading from all the devices and then took an average of all. After this for calculating the distance we used a solution used by the iBeacons for getting the distance which uses the formula of distance as

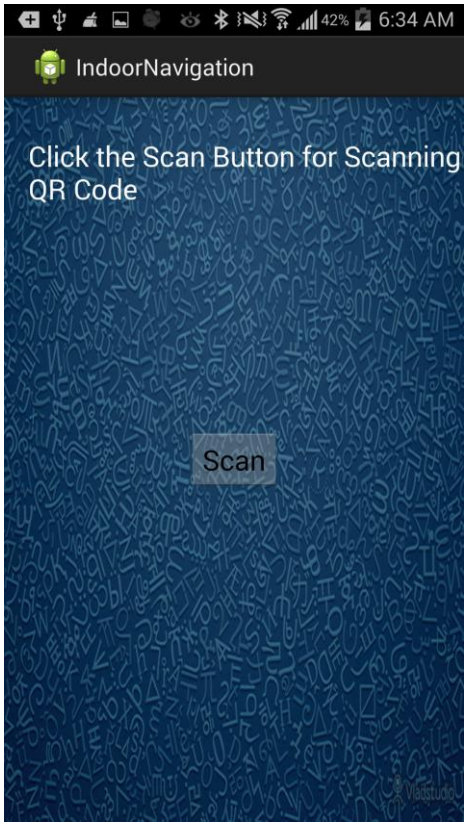
Distance = Received Power(RxPower)/Transmission Power (TxPower)

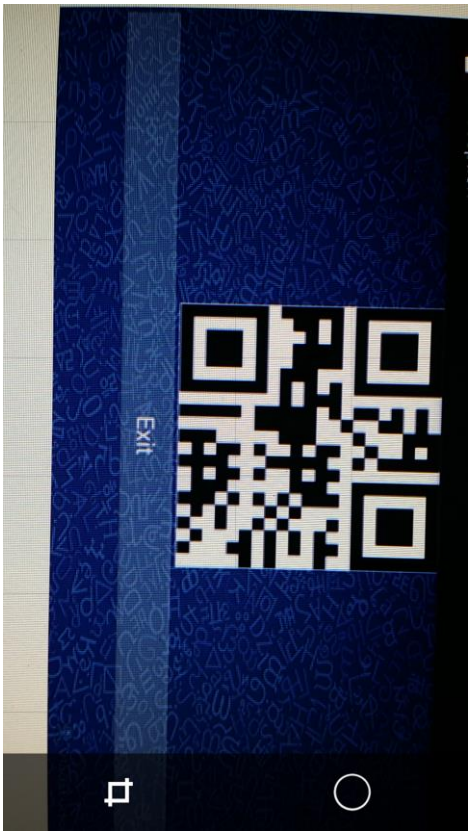
Here in our case Received power = average of samples and transmission power we defaulted to -49 dBm

This gave us approx. correct results for distance with error threshold of 1 to 2 meters

Issues:

- The main issue faced in this implementation is to draw user position based on the sensor reading specially the Orientation of the visitor and how to navigate smoothly like u-turns etc
- If a user closes his application logic to restart or position based on the BLE proximity must be implemented to detect the current location of user
- Unlike previous map solution we don't need to update visitors position based on proximity of BLE as we are updating the position using sensors. But surely using proximity readings we can rectify any positioning errors as mentioned in my second point

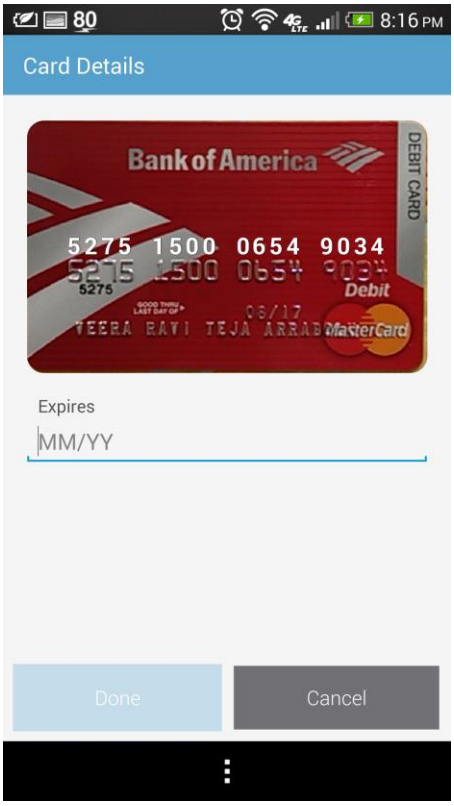




5.9 Miscellaneous

- **Card.io :**

  - ❖ Android SDK which is a card auto detector and reader framework.
  - ❖ It can be integrated to detect cards but relevant documentation on how to customize it was unavailable.
  - ❖ However, if customized it could be used as an functionality for auto detected the visitor's driver license ID and auto scan it.
  - ❖ Screenshots: It was customizable to detect credit cards, which is shown below.



- **<u>Scanner compatibility with android :</u>**

  - The scanner provided in the lab wasn't compatible with the android devices.
  - Explored many apps for compatibility, but there we no apps that could make that particular version of scanner compatible with android.
  - The main issue was with the scanner drivers.
  - Hence we found an alternative to use the camera to take a picture of the id proof submitted (license) and use it for OCR and facial recognition.

6. References:

For OCR:

- http://www.ocr-it.com/document-conversion-service/corporate-litigation-support-ediscovery-pricingpricing
- http://ocrsdk.com/plans-and-pricing/
- http://ocrapiservice.com/rates/

For Face Recognition:

- http://blog.mashape.com/post/53379410412/list-of-50-face-detection-recognition-apis
- http://www.faceplusplus.com/term-of-use/
- https://github.com/FacePlusPlus

For QRcode:

- http://goqr.me/api/
- https://developers.google.com/chart/infographics/docs/qr_codes

For Maps:

- http://support.sailstech.com/kb/tutorial/step1-create-indoor-map-and-path-routing-file-by-using-josm
- https://developers.google.com/maps/documentation/android/

For Virtual Reality :

- http://dev.metaio.com/
- http://www.wikitude.com/products/wikitude-sdk/
- WIKITUDE NAVIGATOR APPLICATION LINK : http://www.wikitude.com/showcase/wikitude-navigation/

For Positioning:

- Development of an Android APK for Bluetooth Localization - Carlos Carrera Carbó, Czech Technical University in Prague.
- A Smartphone Localization Algorithm Using RSSI and Inertial Sensor Measurement Fusion - William Wei-Liang Li et al.
- http://stackoverflow.com/questions/10230502/alternativeto-google-maps-open-source-map-api-that-provides-a-database-of-node
- http://bingmapsandroidsdk.codeplex.com/
- http://stackoverflow.com/questions/11651931/how-to-design-an-android-indoor-map-of-a-building
- https://indoor.io/
- http://wiki.openstreetmap.org/wiki/Indoor/Projects
- http://www.slideshare.net/yswellam/indoor-navigation-24339326
- http://googlemapsmania.blogspot.com/2011/06/how-to-create-indoor-shopping-mall-maps.html
- http://www.youtube.com/watch?feature=player_embedded&v=ATfdi-oYWzw
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=15&cad=rja&uact=8&ved=0CEwQFjAEOAo&url=http%3A%2F%2Frun.unl.pt%2Fbitstream%2F10362%2F9193%2F1%2FTGEO0101.pdf&ei=IeFNU-P8OoqSyAS6soGoDA&usg=AFQjCNFT_P1FlVDFS85LCWcz1OmS26wytw
- http://www.creativebloq.com/inspiration/top-seven-alternatives-google-maps-api-7122779

- http://www.developereconomics.com/mapping-apis-location-based-apps/

For Google App Engine:

- https://developers.google.com/appengine/?csw=1

For Calendar:

- https://developers.google.com/google-apps/calendar/

For Android:

- http://developer.android.com/index.html

For Bluetooth:

- http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html
- https://developer.android.com/reference/android/bluetooth/BluetoothGatt.html

7. CONTRIBUTIONS

1. Arunkumar Rajappan
   - OCR:
     - Implemented the full feature module for scanning the license and processing the image before sending it to OCR service and to send the returned OCR result to next module
   - Face Recognition:
     - Implemented the full feature module for taking the picture of the visitor and processing the image before sending it to face recognition service and navigation to next module if face recognition was successful (along with data containing OCR results)
   - Bluetooth Positioning
     - Explored both Bluetooth API function for detecting both normal and BLE devices. I found API for BLE device to be more accurate and stable for working with BLE devices
     - Came up with a solution for finding the distance between the person and BLE device (as mentioned in section for "Map Navigation using Android Sensors") and implemented the Bluetooth positioning module based on that solution, which seems to provide approximately accurate results. In this if received signal was < then -89 dBM then the device was considered out of range or at a distance > 10 mts

https://community.estimote.com/hc/en-us/articles/201029223-RSSI-Range-Zones-and-Distance-Accuracy

- Map Navigation using Android Sensors (Canvas version)
  - Implemented a Map navigation module based on the android sensors (i.e. compass, accelerometer and step counter) for real time and smoothened positioning of a person. Few links used for exploration.
    http://www.indoorlbs.com/p/indoor-navigation-systems.html
- Map Navigation using Android Sensors (Google Maps version)
  - Explored various options for navigation on Google Maps module implemented by "**Navya Padala**", i.e. for sensor based navigation and updation of user location .
    http://www.movable-type.co.uk/scripts/latlong.html
- Integration and UI
  - Integrated the OCR module, face recognition module and the maps module (i.e. the modules implemented by me with other dependent modules to complete application flow as well the flow of data/results from module by another).

    NOTE: In this once I was done the major role of integrating ALL the modules and giving life to applications UI was done by "**Deep Shah**" and he did a ***commendable*** job in completing the task as well giving all module a proper design/layouts, due to his efforts application's UI has a better and life-like User Interface.

2. Deep Shah

- Google Calendar:
  Implemented the full feature module for Google Calendar. It included Meeting Scheduling with SMS feature and Meeting Verification from Google Calendar using Google Calendar API.
- Maps:
  Implementing Image Based Map with features like pointing Directions, End Point plotting. Also integrated Dijkstra Algorithm of "**V. V Ravi Teja"** with the actual Maps.
- Integration and UI:
  Integrated all functionalities "**V. V Ravi Teja", "Arunkumar" and** "**Navya"** along with **myself** into main application. Integrated from Starting Menu till End Maps so user can have smooth interactions.
- Special Features like Call for Help, SMS, Web Call:
  Also included Call for Help, SMS Alert Event Scheduling and Google Web Page Calling for About Us.

3. Navya Padala

- Scanner compatibility for android:
  Explored different possibilities and apps for making the scanner compatible with android to scan the id proof and use it for OCR and facial recognition with **"Arunkumar".**
- Maps:
  Explored different possible apps like micello, OSM for android etc., for implementing indoor maps.
  Explored and implemented the OSM maps using SAILS cloud for building the indoor map with navigation.

Implemented indoor maps using the Google maps android API v2 with **“V. V Ravi Teja”.**

4. V. V Ravi Teja Arrabolu

- Speech to Text, Text to Speech :
  Explored Nuance dragon, Google services available for speech to text and text to speech, implemented Google service based voice module, provided native TTS support throughout the application.
- Maps :
  Exploring about Google maps services, implementing Google maps with "**Navya"**; experimenting with Virtual Reality based tools (Metacio, Wikitude) for mapping, Exploring positioning methods in publications, utilizing Djikstra algorithm for finding shortest path, implementing the Image based Map plotting with "**Deep Shah"**.
- Bluetooth :
  Exploration about Bluetooth technologies available, RSSI sampling application for measuring Bluetooth signals during exploration, analyzing and comparing signal strength to positioning from a beacon based on the paper [1] referred above in positioning section of reference.
- License ID Scan :
  Explored about possible card detection application for extracting image with better quality, refer miscellaneous section for relevant screen shots.
- QR code:
  QR code exploration, design utilization, QR code cloud API based generator module development, QR code reader module development.