Group #1

Jackie Utitus, Jon Nagamine, Maria Morales, Navid Harandi and Utsav Jain

California State University, Long Beach

Fall 2013

CECS 491

http://www.VisualDialer.com

Abstract

Visual Dialer is a mobile application that attempts to solve two common problems that callers face when calling companies. The first problem that it solves is by displaying a company's entire phone menu system visually as a menu in the mobile app. This allows the caller to select their menu option before the call takes place. As a result, this removes the need for the caller to listen to the entire phone menu system before making their selection. The second problem that the Visual Dialer mobile application solves is that it allows the end user to send pertinent information to the company's phone representative with a click of a button. Rather than spelling out every word, for example the caller's first and last name, the app will automatically send the information to the representative. The representative can access the information via the Visual Dialer Representative Widget. The widget will allow the representative to easily copy/paste information sent by the customer to any other application on the company's computer.

Sometimes it is the little things that make a big difference. By solving these two problems, customers should have less frustration contacting companies. In turn companies should see higher customer satisfaction.

Table of Contents

1. Introduction and Background

1.1 Statement of Problem Area

Today, companies provide many options for their customers to get in contact with them. When it comes to the most common one, that being a phone call, companies have created in many cases complex phone menus before a customer is able to reach a live person. The goal of Visual Dialer is to make that experience less frustrating. Another problem is that in many cases the customer has a hard time providing information to a representative without having to spell every word. With the Visual Dialer Representative Widget, the customer can send information directly to the representative from their mobile device.

1.2 Previous and Current Work, Methods and Procedures

N/A. No previous work.

1.3 Background

The idea for this project came about from many frustrating calls to various companies. From computer companies to department stores to government agencies. Most of them seemed to purposely make it difficult for a customer to talk to a representative. On top of that, there is an inherent problem with having a customer try to interact with a computer over a phone. In particular they have to listen to an entire set of menu options before choosing. This is time conserving and adds to the overall frustration by a caller. Once a caller is finally connected to a representative, in many instances, the representative has a hard time making out the exact spelling of various things that they need to input in the company's computer system. This can be the caller's name, address, phone number or various other things. So with these frustrating problems in today's phone systems, Visual Dialer was born.

1.4 Brief Project Description

As part of the class project, we chose this project because it involved various different technologies. These technologies presented a good set of skills that the team would be introduced to. These include:

- mobile app for user interface

- web-based support widget

- IVR integration for dialing mechanism

- server side web services for interaction with the mobile app, support widget and IVR

- SQL Server to store company phone menus

- social network integration to provide a way to virally advertise Visual Dialer

Of course, like most projects, the scope of the project can grow limitless. As part of the project for this class, we as a team decided to keep it to the main functionalities. Providing a way for an end user to see a list of businesses and their phone menus, be able to dial out to a selected menu option, be able to share their selected menu item with friends, and be able to interact with the customer representative by passing necessary information directly to their computer.

1.5 Purpose/Objectives/justification of Project

Visual Dialer gives user a more friendly experience when working with business call menus and will make user more incline to go through the menu to solve their problems then just taking to a representative. Visual Dialer makes communication with businesses easier than before with options for convenience such as search for hard to find businesses and favorites for businesses that the user calls frequently. Visual Dialer also allows users to send the information they want to send such as specific comments or for those with more challenging names it creates a quicker connection between user and representative that guarantees the representative gets the data that the user input for their information.

1.6 Team work assignment and accomplished

Our team agreed on using SCRUM Agile methodology. Each release was considered a sprint and the team had a weekly standup to provide status.

The following is a list of major areas of the application and an overview of each team member's contributions:

- Android Mobile UI
    - Majority of development:
        - Jackee Utitus
        - Jon Nagamine
        - Maria Morales
    - JUnit Framework Setup
        - Utsav Jain
    - Some graphics work
        - Navid Harandi
    - API client:
        - Utsav Jain
- Representative Widget
    - Majority of development:
        - Utsav Jain

        - Some additional contributions:
            - Maria Morales
            - Navid Harandi
- WebServices and Database
    - Utsav Jain
- IVR Call Processing
    - Navid Harandi

The following is a list of sprints and what the team accomplished:

- Sprint 1: Alpha Release (Sept.16th to Oct. 7th)
    - Android Development
        - Main menu
        - Business menu
        - Phone directory menu
        - Action page
    - WebServices and Database
        - Setting up the database infrastructure
        - Setting up the tables in the database
        - Setting up a restful WebService using MS Web API
    - IVR Development
        - Setting up the IVR Cloud Interface
- Sprint 2: Beta 1 Release (Oct. 8th to Oct. 28th)
    - Android Development
        - Connectivity with Web API
        - Favorites functionality
        - Sharing mechanism(email and text message)
        - Added search feature for the phone menus
        - Various work on the local DB
    - Representative Widget Development
        - End to end completion of Representative Widget
        - User on Android can send data to representative by calling the Web API
        - Representative Widget will continuously call Web API to see if there are any messages by the Android device
        - Completed UI development of sample widget site
    - Web API Development
        - Implemented Microsoft's Entity Framework
        - Completed code for Android (phone menus, dialing, and status checking) and RepConnect
        - Completed code for Twilio integration using their .NET MVC Helper Library
    - IVR development
        - Setup Twilio to connect to our Web API server upon end user calls
        - Android app dials into IVR which will then interact with the

Web API to determine what business to dial and the dial sequence

- Sprint 3: Beta 2 Release (Oct. 29th to Nov. 18th)
    - Android Development
        - Google+ Integration
        - Ability to send custom messages via RepConnect
        - Added Unit Tests using JUnit
        - Various bug fixes (ex: Favorites and Menu order)
        - Further refined the UI by adding some more graphics
        - Error handling related to getting categories, businesses, phone menus
        - Caching and syncing mechanism
    - Representative Widget Development
        - Injectable widget via chrome extensions
        - Ability to receive custom messages
    - Web API Development
        - Added unit tests and refactored code as a result
        - Setup procedure for pushing up code, database schema changes to production
        - Added more content for phone menus
- Sprint 4: Final Release (Nov. 19th to Dec. 16th)
    - All Teams:
        - Complete documentation
        - Test the entire application
    - Android Team:
        - Continue cleaning up the UI code and making it look pretty
    - Representative Widget Development
        - Look into security measures
        - Look into ability for representative to easily copy/paste data from the widget

2. System Functional Specification

2.1 Functions Performed

Calling

Calling is the main and primary functionality of Visual Dialer allowing the user to quickly visually transverse their call menu's and then when calling get directed directly to the option they are looking for. For this they must first select a category and a business. This allows them to call the business directly or chose the phone menu option they want.

Connecting to Representative

The connect to rep button allows a user to directly communicate with the business they are attempting to get in touch with. This function also allows the user to have preset information that they can send the the representative such as name, address, city, etc. However, the user is able to edit this information or delete information they do not want sent before they send it to the representative they want to get in touch with. In addition the user is given an optional comments section so the user can put in their own writing what they want to connect the representative about. This is able to be used by any company that downloads the Visual Dialer Google Chrome add-on. The representative then gives the user a specialized code and the user can safely and securely send their information to the representative.

Sharing

Sharing allows the user to share their use of Visual Dialer with their friends. Users can send pre generated messages, that have the option to be edited, to their friends through email, text messages or Google+.

Search

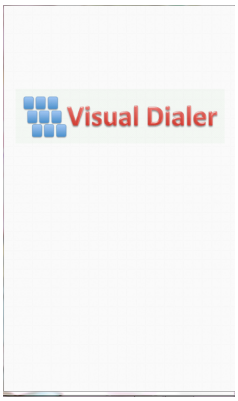
Search is used for the users convenience when looking for businesses or call menu options that they are not able to find or they do not want to look for their position within Visual Dialers menus. The search function searches for results for all words individually from the database and returns anything that matches any word from the user inputted data. The user then can either chose a result or re-enter new information for searching. When the user selects a result they are either directed to the menu if that option still has sub menus for the user to select from of bring them to the action menu for further options.

Favorites

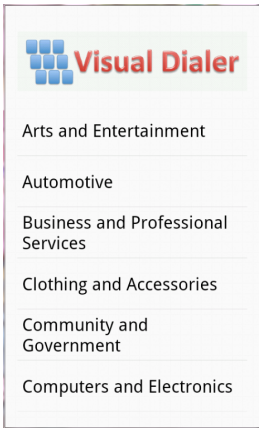
Favorites is used for the user to mark what they consider to be items that they will call multiple times by adding it to their favorites list. This list is able to be accessed through the menu button on the phone. Once a favorites list item has been selected the user gets sent to the action menu where many other features are available. The user also have the option to press the menu button on the phone and clear their favorites list.

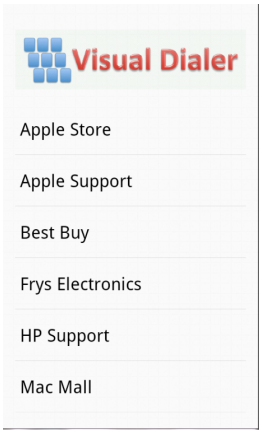
2.2 User Interface Design

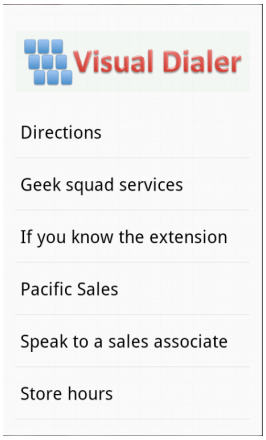
The user starts off at the splash screen which promotes Visual Dialer do the user. This is where the application calls the server check for a status update and if needed updates the SQLite database stored on the phone.



The user then gets directed to the main menu where the user is able to select from different categories of businesses. Dependent upon the category that is chosen businesses are generated that are under that category. Then the user is directed to the businesses sub menu where the user can chose the sequence they want Visual Dialer to call for them. Here you can see possible sequences that user can choose.

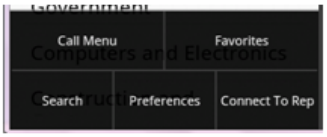






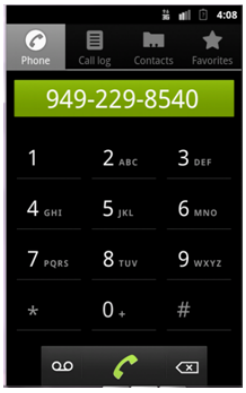
After the user selects the sequence to get in touch with the business they are looking for they are directed to the action menu where the user is given the chance to select from multiple options. They are able to call the business or business sub menu that they have selected and they are also able to add that selected item to a favorites list for quicker references. They are given the option to share their experiences through the share option and they are given the option to send their information to a representative through the

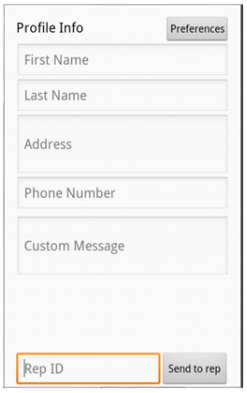
connect to rep option.



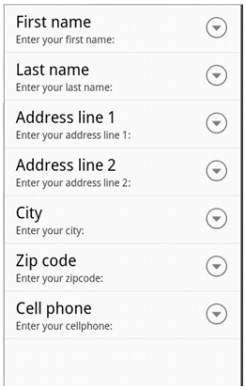
The application also works with the phones menu button. When this button is pressed and either the main menu or the action menu is a menu with sever more option for the user to take advantage of. This allows the user to go to the action menu (labeled as call menu), go to their favorites list, search through the businesses, update their preferences or go to the connect to screen.

If the user choses to press the dial now button from the action menu they are directed to their on phone dialing system. This can vary dependent upon the android system the user is using and what calling application the user has on their phone. The phone number is previously generated by making calls to the server where the desired directory is sent and a phone number is returned. However, the application does not call out automatically it waits for the user to press the call button. This allows the user the choice to go back and not call the business that was selected.





At the action menu they can also choose to connect to a representative. If the user chooses this item they are redirected to the connect to representative screen. This allows them enter data that they want to send to the representative. This guarantees that the information the user enters is what the representative receives, assuming the representative is using the Visual Dialer Google Chrome add-on.



From the connect to representative screen they are able to access their preferences; preferences can also be access through the phones menu button at the action menu or the main menu. The preferences are saved through out the application and gives the user the option to save static information such as their name, address, city, etc. If the preferences are filled out the connect to representative screen will be pre filled out with an option to edit.

At the action menu the user can choose the share button which will direct them to a new share menu options. This share menu allows the user to let others know they are using the Visual

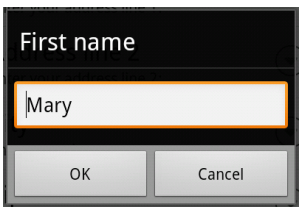
Dialer application. Users are given the option to either share through email, text messages or with Google+ for sharing on a social network.

For the share functions when sending to someones email or texting someone there is a popup box that is displayed for the user where all the user has to do is enter in their friends information and the rest is done for them. However, Google+ is different. When you share with Google+ you are directed to a Google+ sign in page. Here, when the user signs in the application will gather your gmail and google acount information from the phone. This allows the application to use this information to sign into their Google+ account. The user then has the option to post which allows the user to edit information such as text, the link to the website, location and other such things provided by their Google+ account.

The user can add items to their favorites list when they get to the action menu as long as they have selected a business or business item. When they press the add to favorites button the selected business or business sub menu will be added to the favorites list. The user can access their favorites list when they are in the main menu or the action menu by pressing the phones menu button and selecting favorites. They user can select any item on their favorites list and after selecting an item they are directed to that items action menu. Users are also given the options to clear their favorites list by pressing the menu button on the phone and then choosing clear all.

The user has the option to search for the business or business sub menu they are looking for. The search is accessible by pressing the menu button on either the call menu or the main menu. Search treats each entered word as a different entity and searches for any element in the database whose name contains any of the entered words. All of the results are shown to the user for them to choose from. Then they can simply choose which item they were looking for and they will either get redirected to the main menu if there are sub menus still available or to the action menu where they are given many more options.

2.2 User Input Preview

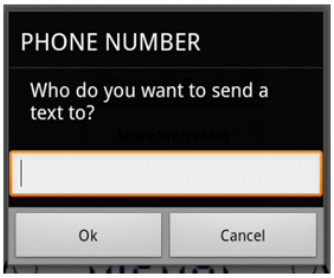


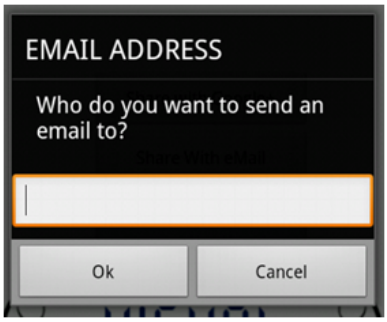
When the user is at the preference screen they are able to edit individual information. When a user selects a given

section to edit they click on the right most tab. Then a box will open giving the user a chance to edit this information. If this section was previously filled out all they have to do is delete the old information and enter new information. The information here will always be saved in the application.

When the user transitions from the preference screen to the connect to rep screen they will noticed that any fields they have filled out in the preferences screen is already filled out in the connect to rep screen. Though some information is already filled out all fields are still able to be updated. Here specifically, the user is able to edit a specific field for comments. This is where the user can put in precisely what they want to talk to the representative about. The user does need to enter in a connect to representative code so they can transfer their information directly to the representative. It should be noted that the information that has specifically edited here will not remain when the user leaves the connect to representative screen nor is it saved on the phone for so the user is given more privacy.

When the user is at the share screen they are given the chance to share via text messaging. When the user selects this option they will see a box pop up that allows them to enter in their friends phone number. The user can enter any information they want in this box, however, only the numbers will remain and if what is entered is an invalid number simply the application will not send the message.





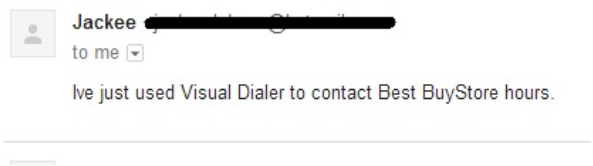
When the user is at the share screen they are also given the opportunity to share through email. If the user selects this options a pop up box will appear that allows the m to enter in the email address of their friends they want to share Visual Dialer with. WHen the user presses OK here they are redirected to the phones email application where they are able to customize the email content and email address.

When the user goes to the share menu they are prompted for input. This input is split into individual words and each word is treated differently. The database is then searched for any name matching or containing part of any word the user inputs. All of the results are then displayed for the user to

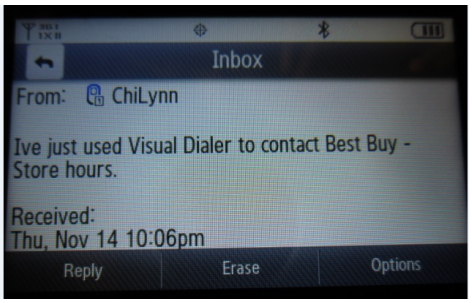
see. The user has the option to delete the entered words and enter new input. If the user enters input the old search results are lost and new results are generated in it's place. When the user selects a result displayed from the search they are then directed to the location of the selects. If the selection does not have any sub menu's they are then directed to the selected options action menu.

2.3 User Output Preview

The users output is not visible on their own device. When the user uses the share function it will send an email to another person persons, and the user themselves do not have access to the email they sent unless they sent a copy to themselves. Before the user clicks the send button there is a screen that shows what will be sent in the message. They can however view the information if the person they sent to is willing to share the email.

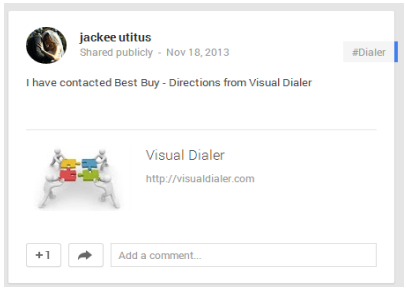


or



Another output is the text function in the share menu. This function sends a text to whomever the user wants to send a text to saying that they have used the application to call whoever they have called. Like the email the user can view the information to be sent but once it is sent they do not see the end output, unless they sent a copy to themselves, or are able to view one they sent to someone else.

The post to Google+ function enables the user to login to Google Plus and to post on their page that they have used the application. This function, like the text and emails, allows the user to edit what they post. Unlike the others though, this post can be viewed by the user on their own Google Plus page without the need to "send a to themselves" or to contact someone they sent it to in order to view the post.



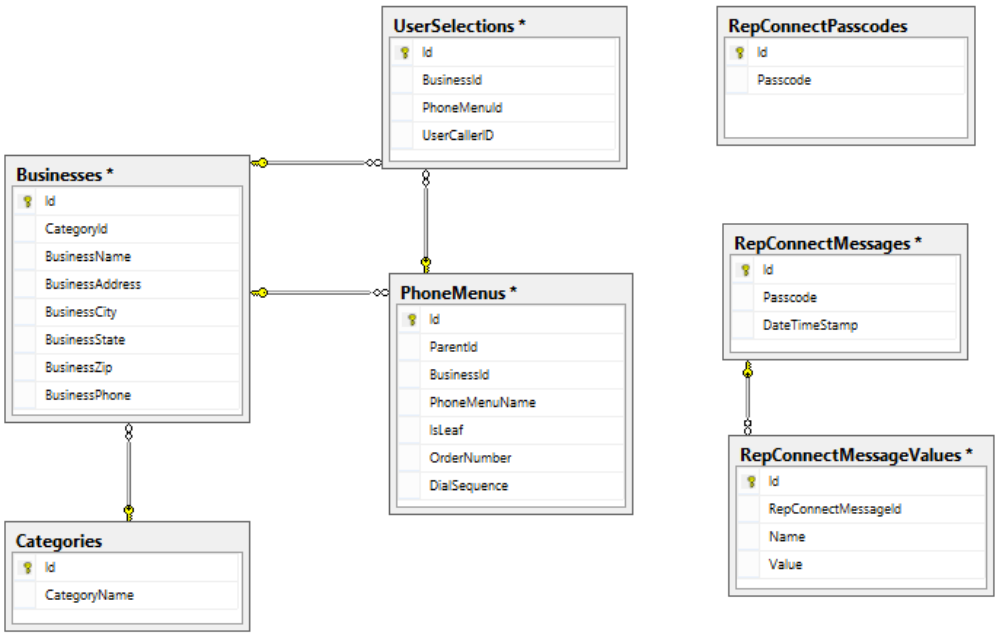
With the connect to rep function, the user can edit the information that they want to send to the representative for the representative to process the users request. The user may be required to enter

certain information to send to the representative such as an ID number or other personal information that the representative may need, or to remove information that the user does not want to send to the representative if it is not necessary.

2.4 System Data Base/File Structure Preview

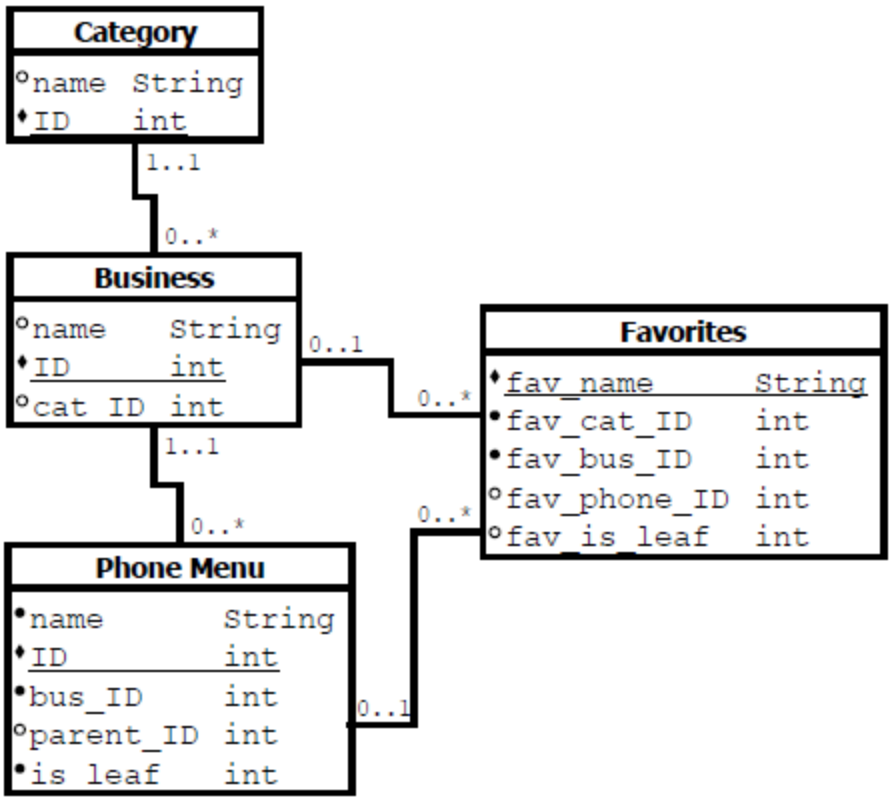
There are two databases used within Visual Dialer. On the Web API server side, we are using Microsoft SQL Server 2012. On the Android side, we are using a local SQLite database.

The server side SQL server is automatically generated by code using the Microsoft Entity Framework. Here's a diagram showing the table structures:



SQLite database:

The SQLite database is created using the SQLite database libraries within the android framework. The structures are connected as such:



2.5 External and Internal Limitations and Restrictions

One of the primary restrictions is the requirements of the libraries used. Google Play services, a library used by our application requires that the application be installed on a device running Android 2.3 or higher. Development can be continued on a device running Android 2.2 but an additional library is needed to do so. In order to be up to date, the application requires a connection to the internet to access our servers in order to have the most current information about the companies contact information.

2.6 User Interface Specification

2.6.1 Interface Metaphor Model

Refer to section 2.2

2.6.2 User Screens/Dialog

The Opening screen is the first screen the user accesses when opening the Visual Dialer application. This screen both advertises the application and gives time for the application to access the server side database to load data into the SQLite database. The opening screen transitions the user to the Main Menu.

The Main Menu is the first interactable screen the user goes to. This screen provides the user the opportunity to find businesses by category and transverse their phone menu visually. The user is given many other options through the phone menu button. The user can be redirected to many screens through the menu button however if they transverse all the way through the menu they will get directed to the Action Menu.

The Action Menu is mainly for the user to interact with the sequences they have chosen. The user is given the option to call the business they have selected, the option to connect to a representative, add this business and it's phone menu to the favorites list and the ability to share with their friends their use of Visual Dialer. These options are only available when a user has selected a business. The user then is directed based off of the selection they have chosen.

The Favorites screen allow the user to select from items they have added to their favorites list. The user also has the option to press the phones menu button and clear their list of favorite items. If a user selects a favorites item they get redirected to the Action Menu.

The Search screen gives the user the ability to enter in information that gets split up by each work and searched for in the database. Then a list of results are shown where the user can interact with the given items and get redirected to the Action Menu or the Main menu dependent upon whether the selected item has a submenu.

The Share screen gives the user the options to select what kind of way they would like to share Visual Dialer with their friends. The user has the option to share with text message, email, or Google+. Dependent upon what the user chooses the user is either prompted for more information in order to send the messages to their friends or are redirected to the Google+ screen.

The Google+ screen gives the user the ability to sign on to Google+ using their Google account. Once the user signs in they are able to post a message to their Google+ account saying they have been using the Visual Dialer application and directing other to the Visual Dialer website. SInce the user is using Google+ they are given other options before accepting the post to add information such as tagging people or adding a location.

The Preferences screen allow the user to update personal information that is repeatable used in the application that are saved internally on the phone. This information is used for connecting to representatives.

The Connect to Rep screen has text fields that the user can fill in to send to a representative they are talking to. Each field is optional and can be edited. The rep ID is needed from speaking to a representative to guarantee that the users information is only sent to the representative they are in contact with. The Connect to Rep screen also allows people to fill out remembered personal information by pressing the preference button thus redirecting you to the Preferences screen.

See section 2.2 for further detail.

2.6.3 Report Formats/Sample Data

N/A. There are no reports or sample data associated with this project.

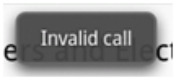
2.6.4 On-line Help Material

N/A. Visual Dialer is a mobile application and the whole point of it is that it’s user friendly enough where it doesn’t require an on-line help material.

2.6.5 Error Conditions and System Messages

At the opening splash screen the application makes a call to the server and checks for an update. If the application is not able to connect to the internet instead of closing the application or running with no data it works of off the old previous data that is stored in the phones internal SQLite database. If there is an update the application clears it’s old data and makes another call which will receive the new data from the database linked to the server.

When the user clicks on an item in the main menu some categories or businesses don't have sub menus. When the user tries to interact with these sub menu items then receive a message notifying them that they are not able to interact with that sub menu.



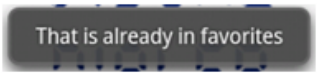
When you use the phones menu button to go to the action menu but you have not selected a business you will receive a warning letting you know what you have selected is not an item that you are able to call, share or add to your favorites list.

At the action menu users will only be able to call out when they have

actually selected a business or a business sub menu item. Otherwise an error message will pop up notifying them that calling is not an option.

The user will also not be able to add items to their favorites list if it is not an item that is able to be called. If a user attempts to add an item to their favorites list and it is an invalid item an error message will appear.

In the favorites list everything is unique since there is no need to have an item in the favorites list multiple times they are not able to click the add to favorites button for the same item multiple times. This does not include adding different sub menus within a business since each of them count as different items in the favorites list. For the users convenience there is an error message that shows when the user tries to add an item multiple times.



When the user is in the search menu they are able to enter anything they want however not everything the user enters will render results. When they search does not find anything related to the user's search a message is displayed letting the user know that nothing was found with the search they entered.

2.6.6 Control Functions

N/A to this project.

3. System Performance Requirements

3.1 Efficiency

The application makes use of a local caching mechanism where it determines if there are updates to the phone menues on the server. If there are updates, the local database is updated, otherwise it continues on without the need to communicate with the server to pull down data. This makes the user experience much more efficient.

The JSON requests are tiny and allow for an efficient means of communication between the client and server.

3.2 Reliability

3.2.1 Description of Reliability Measures

Visual Dialer is very reliable. In most areas that the user could run into areas such as selecting invalid options or inputting wrong data the user is not able to access. The user will be displayed with error messages preventing them from accessing invalid options. Computational areas are surrounded with try and catch blocks preventing users from selecting options that could potentially crash the the application. The application is set up so if one section does crash it will display a message and return the user to the previous screen that they were at.

3.2.2 Error/Failure Detection and Recovery

The SQLite database stores the data it previously retrieved and if for any reason that the user cannot access the internet the database works off of information that was previously installed on the system. This makes sure that the application does not crash and is still runnable when the user has a lack of internet access. However, in this situation the user will be unable to make calls using Visual Dialers services and the user cannot Share through email or Google+ with their friends.

There are several areas that the user cannot access such as calling a category. When this happens the user is given a message informing them of their error and prevents the application from crashing. See section 2.6.5 for further detail of potential problems.

3.2.3 Allowable/Acceptable Error/Failure Rate

It is not acceptable at all for the device to crash because of the Visual Dialer application. The Google Play services should not be crashing. If a screen does crash, returning the user to a previous

screen and preventing the entire application from crashing is acceptable. The Google API console needs to be up and fully functional, however, this is something that is out of our control.

3.3 Security

3.3.1 Hardware Security

The hardware security is the security the user has on the phone itself. Some devices have a 4-digit password, others have a connect the dots password. Some people don't set up security on their phones and there is no password or code, but just slide a bar and they have full access to the phone and all its contents. Another form of hardware security is how well you secure the device, leaving it lying around is not very secure even if there is a password.

3.3.2 Software Security

In terms of software security, on the server side one security measure that we have in place is the access to the SQL Server. As best practices dictate, our Web API layer is the only code that has access to our SQL Server. It serves as a gatekeeper. There are other areas that require security measures which was not part of the scope for this class project, but should be implemented as one of the first items if this project is to be released to the public. In particular, the Web API calls that are made by the mobile app and support widget. The calls should be made via SSL using https rather than http. The reason our team decided not to include this in the scope for the class project was because of the cost of SSL certificates.

3.3.3 Data Security

As far as data security goes, the information that is sent from the end user to the representative requires encryption/decryption logic from both the client app and the support widget. In other words, the Visual Dialer servers should at no point have access to the data. In case of an intrusion, the customer’s sensitive data would not be available. This item was also not part of the scope of the class project.

3.3.4 Execution Security

Primary execution security is the users login information on Google

Plus and the security on their device, such as any password on the phone itself.

Google API utilizes the OAuth 2.0 protocol for authorization and authentication purposes. The process starts by registering the application with Google Developer Console to have access to manage the applications access to various APIs and to present information about the app to the user. Once registered, a client id is assigned to the application that is known to both Google and the application. This id is used to authenticate the user and gain access to the Google APIs. Our application uses the Google+ Sign-In library to help with the authentication of the users and the accessing of the APIs. When the user signs in they are prompted to select who they want to be able to see the posts made by the application. Once the user finishes the consent screen an authorization code is returned from the Google Servers. This code is exchanged for a token and the token is used to call the Google API.

3.4 Maintainability

The code is written in an object-oriented fashion. In addition, the Web API layer is written using MVC patterns. The entity framework also provides an increased means of maintaining it. So all in all, the code is very much maintainable. There's definitely room for some refactoring in all areas. However, that's expected from a project that was worked on for a school project.

3.5 Modifiability

The application is largely data-driven and dependent on the local SQLite database rather than platform-specific features. The separation of the database layer from the client application layer makes it easy to modify the Android application without affecting the database. Because the Android application presents a view of the information stored in the database, changes to the local database are reflected in the application menus.

3.6 Portability

The application is designed for Android devices, more specifically for Android version 2.3 and later. The device need to have Google Play Services on it in order to run the application. As of now the application is not available for use on any other platform (iOS, Windows Mobile, etc).

The Web API on the server side is written in Microsoft .NET using MVC 4

and Microsoft Entity Framework. So this server-side piece will only run on Microsoft operating systems and Microsoft web servers (IIS). The only way this code can be portable is if it's compiled via Mono C#. This has not been tested from our end and from our understanding, Mono C# does not include every .NET library. So at the least, there may be some tweaking required to make it work.

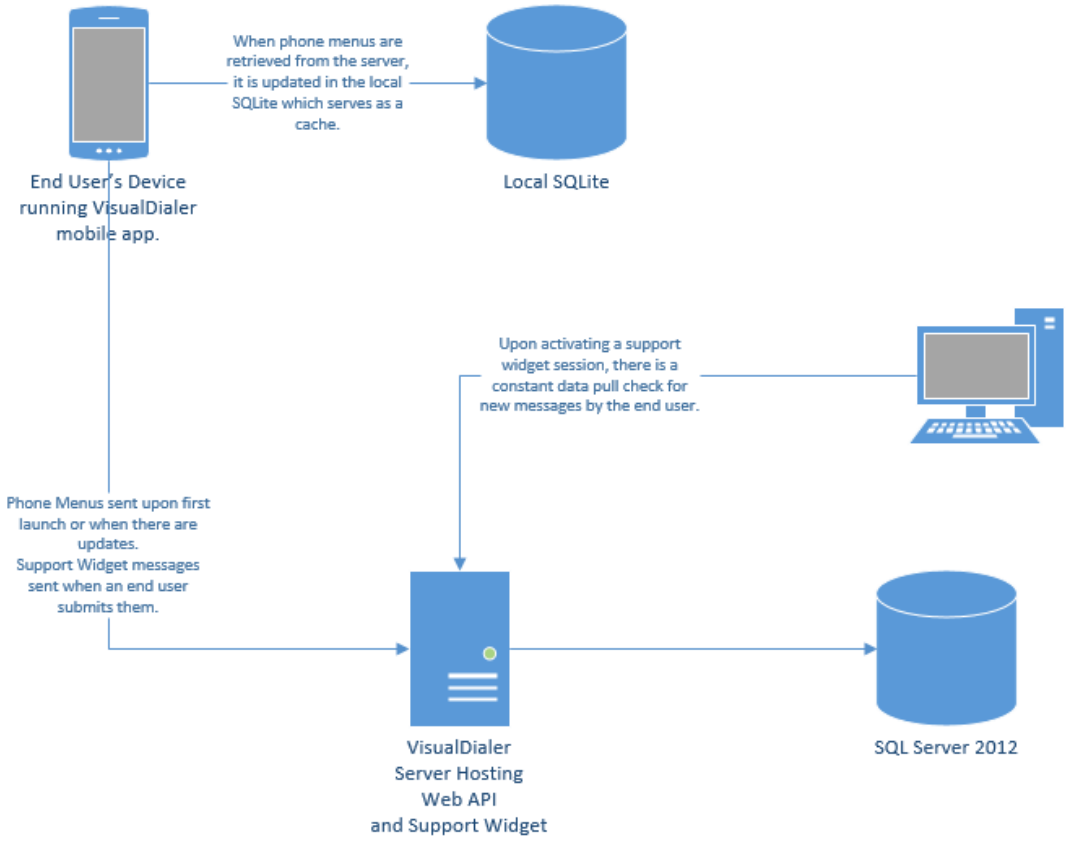
3.7 Others

N/A.

4. System Design Overview

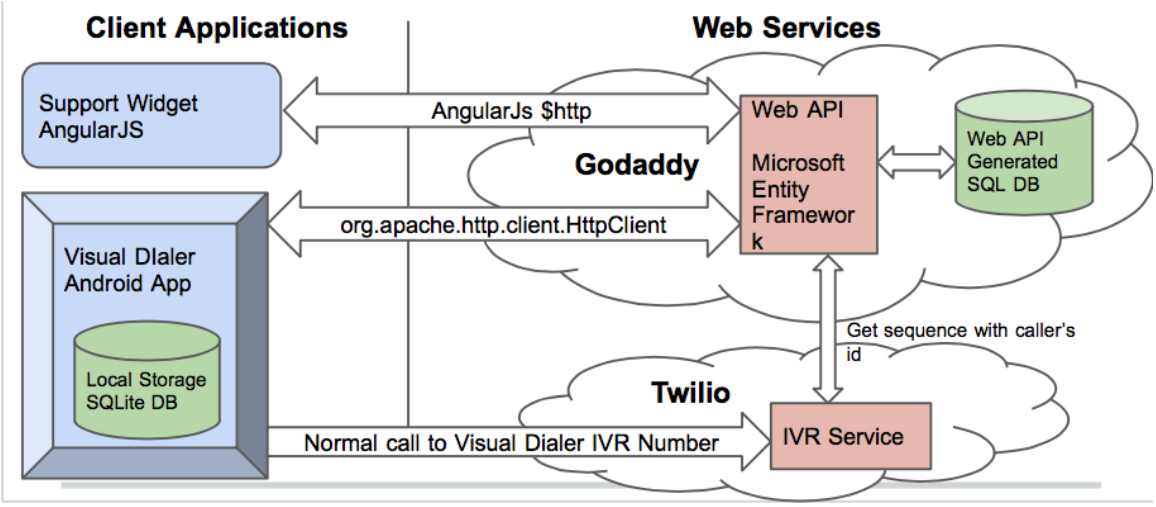
4.1 System Data Flow Diagrams

Here is the Visual Dialer data flow diagram:



4.2 System Structure Charts

Here is a chart of the system structure:



4.3 System Data Dictionary

The system is comprised of the following data elements:

- Category:

    - Categories of businesses which are displayed to the end user as a menu. Ex: “Computers and Electronics” is a category which contains businesses that fall in the computers and electronics industry.

- Business:

    - Businesses that contain a PhoneMenu to be used in VisualDialer. Upon selecting a Category, a list of businesses are displayed to the end user to select.

- PhoneMenu:

    - PhoneMenus that may contain other PhoneMenus in a parent/child relationship or an end point which allows an end user to dial, save as favorite or share with friends. PhoneMenus contain the dial sequence needed to connect an end user to the selected phone menu item.

- UserSelection

    - Stores the user’s selected PhoneMenu. Upon calling the IVR, the last UserSelection item is queried for the user based on their callerID. The system will then connect the user to the selected PhoneMenu using it’s dial sequence.

- RepConnectPasscode:

    - Upon starting a Support Widget instance, a Passcode is

generated. This passcode allows the end user's device to connect to the correct representative's Support Widget.

- RepConnectMessage:

    - Associates a message item with the passcode. The end user's message is sent to the Support Widget.

- RepConnectMessageValue:

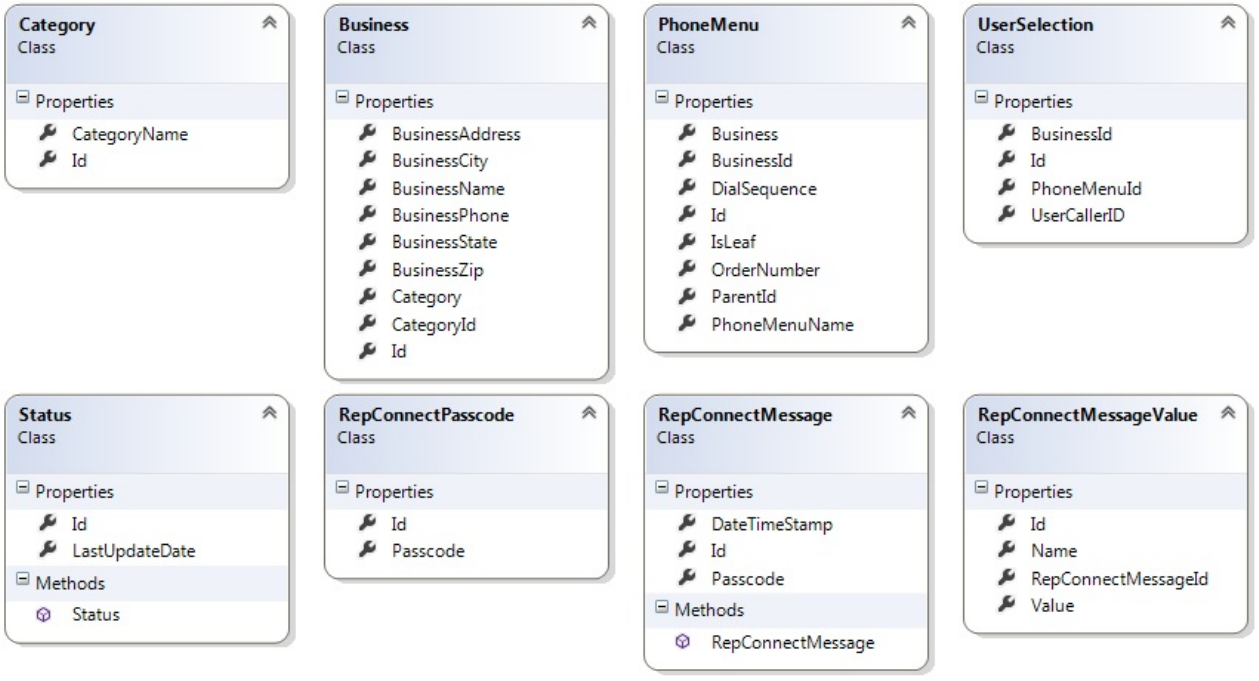
    - Associates name/value pairs to a RepConnectMessage item to be sent to the Support Widget from an end user's device.

- Status:

    - Provides that datetimestamp of the last update to the database related to the Categories, Businesses and PhoneMenues. This datetimestamp is used for syncing by the device with it's internal database as a cache.

4.4 System Internal Data Structure Preview

Below is the class diagram showing the internal data structure of the Web API layer:



4.5 Description of System Operation (high level)

The full system consists of four layers, the client applications (Android app and support widget), the web API services, the database, and the web IVR service. Communication between the clients and web API happens via Asynchronous HTTP requests. The requests range from simple gets to

sending JSON data. The responses, if any, are always in JSON format. When the Android app is ready to dial a menu sequence, it calls the Visual Dialer Twilio IVR service which gets a dial sequence from the Web API based on the caller's id. That sequence is repeated to the destination number. The support widget is a simple JavaScript UI that gets integrated into the website of an existing customer support portal. When a new session is started, the widget will continuously poll for new messages that are sent via the Android application. When the session is ended by the support rep, the polling stops.

4.6 Equipment Configuration (diagram and description)

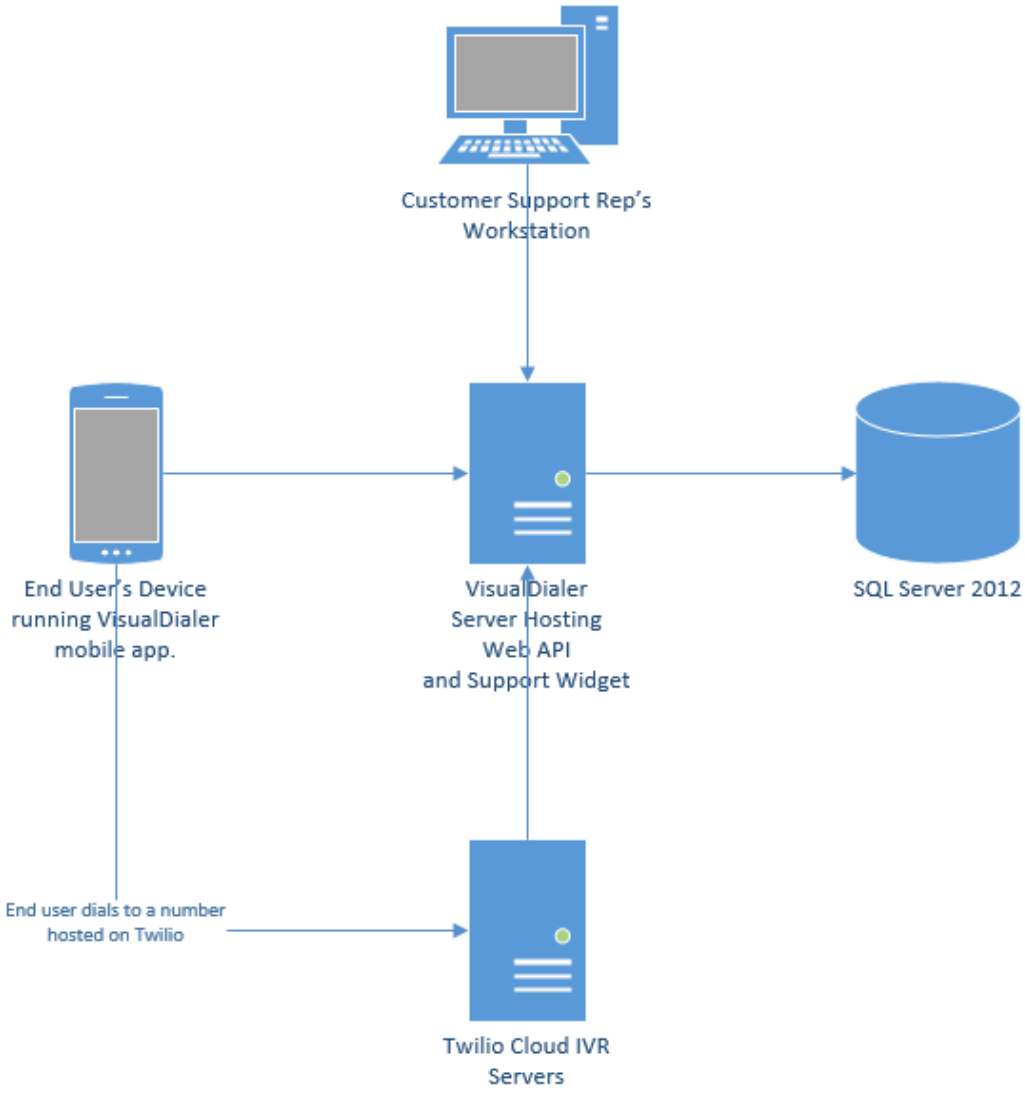
The Android application is based on a phone with sdk version 2.3 or higher and requires network permissions to be fully functional.

```
<manifest xlmns:android...>
 ...
 <uses-permission
android:name="android.permission.INTERNET"></uses-permissio
n>
</manifest>
```

The Web API is based on Microsoft .NET 4.5 using MVC 4 and Microsoft Entity Framework. It is hosted on a Microsoft Operating System using MS Windows 2012 Server. Currently it is hosted on Godaddy's shared web account. The Web API communicates with a SQL Server 2012 instance, also hosted by Godaddy. By using MS Entity Framework, the database is automatically generated for us in the dev environment (local machine). Upon release to production (Godaddy Servers), the database is replicated there.

The support widget is written in Angular JS and also hosted on the same Godaddy shared web account. It is accessed by a customer support rep's workstation by means of an injectible module in their browser.

Below is a diagram showing the various equipment:



4.7 Implementation Languages

The Visual Dialer application is implemented as an Android app that is compatible with 2.3 devices or higher. We chose Android as the first platform because our team is more familiar with Java vs. Object C, and the Android userbase has surpassed all other mobile platforms (greater than 80 in November of 2013 - see http://www.businesskorea.co.kr/article/2200/mobile-os-competition-android-market-share-surpasses-80). It also supports simple HTTP requests which works well with a REST Api. Once the Android app is ready for the market, we will implement an iOS version.

The Customer Support Widget is written in AngularJS 1.2.4. We chose AngularJS because it allows for fast development, has a good support

community, is open source, and can be used with the Google Closure compiler for optimization. Since it utilizes JavaScript, it also allows for an easy integration into any existing web applications via some script tags and a few lines of JavaScript. JavaScript also allows for creation of a Chrome Extension utilizing the same code.

Android specific tests are written in Junit3 (http://junit.sourceforge.net/junit3.8.1/) because Junit4 (http://junit.org/) is not supported without jumping through hoops. Non-Android specific Java test are written in JUnit4. JavaScript tests are written in Jasmine 1.3 (http://pivotal.github.io/jasmine/).

The Web API is written using ASP.NET 4.5 in C#. We have implemented MVC 4 and also Microsoft Entity Framework. We had initially selected Java for the webservices portion, however, because of limitations by Godaddy's servers with Java, we chose .NET. It turned out to be a good solution even though we have other parts written in Java. If we were to go back and do it all over, it would probably make sense to go with Java all around.

The server Web API layer interacts with SQL Server 2012. Specifically, the Entity Framework automatically generates the necessary tables for us there. We chose Microsoft SQL Server 2012 because of the ease of use with .NET and it was available to us by Godaddy.

On the client android side, the mobile app makes use of a local SQLite database as a means to cache the phone menus for the end user. We chose the local SQLite because it provided a good user experience for the end user. It is the default database system available.

For the IVR phone processing, we decided to go with Twilio's offering. It is very inexpensive and you pay for what you use. That seemed ideal for us since we didn't have the money to setup IVR servers from scratch.

4.8 Required Support Software (pre-existing)

In order to run the application, the user has to have Google Play Services on their phone. This is free from Google.

In order to run the Support Widget, the customer representative user needs a Chrome browser. This software is also available free from Google.

5. System Data Structure Specifications

5.1 Other User Input Specification

5.1.1 Identification of Input Data

Users can store contact information in the Android application that can be later sent to customer service representatives via the Connect to Rep feature. The Connect to Rep feature allows users on the mobile app to share information with representatives using the browser widget and make it easier for representatives to help customers.

This information can be entered through the application Preferences or through the Connect to Rep screen.

5.1.2 Source of Input Data

Contact information is entered manually by users. This allows users to share only the information that they are comfortable sharing. Additionally, all text fields in the Preferences and Connect to Rep screens are optional.

5.1.3 Input Medium and/or Device

Contact information is input into the application through the Android text input device (e.g. on-screen keyboard).

5.1.4 Data Format/Syntax

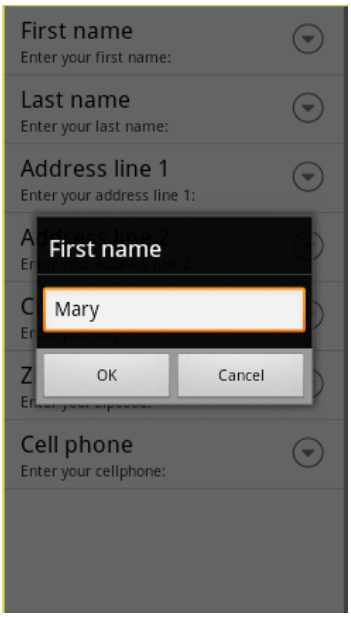
Information is input into text fields. For information stored in the Preferences screen, each text field is associated with a Preference which is saved for future use. Text fields in the Connect to Rep screen are not associated with any Preference and can be filled for that specific customer service session. No restrictions are placed on syntax, allowing users to input any variety of string relevant to their situations.

5.1.5 Legal Value Specification

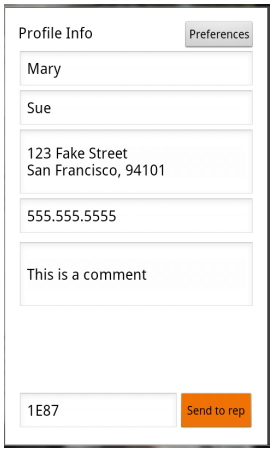
Any text input is accepted as a legal value for the text fields so that users can store any type of information relevant to their particular customer service experience.

5.1.6 Examples

Preference screen with sample input. A pop-up text field appears when a user selects a Preference (First name, Last name, etc) to edit:



Connect to Rep screen with sample input:



5.2 Other User Output Specification

5.2.1 Identification of Output Data

The share function sends miscellaneous data to Google+ such as email and texts. This data has no real purpose save to share with friends and be free marketing for our application.

The connect to rep data is information that may actually be

important. This information is information that the representative needs to assist the user with the issues that they may have been having. BY having this information it is also easier to input the information since the representative can copy and paste the information into the appropriate fields as opposed to trying to understand the user spelling out their name or trying to say a 25 digit id number and such.

5.2.2 Destination of Output Data

There are several types of output destinations. There is the text destination, the email destination, the Google plus post destination, and the connect to rep data destination.

The first is the text share function, This functions send the output to another phone, the person that the user designates as the recipient of the message. The email destination is the email of whomever the user of the application decides to send an email to. Another output destination is the Google Plus page of the user. This post is visible to those that the user gave permission to view when they first signed in with the application.

The email output is sent to the recipient, who is chosen by the user, via email. The destination is whatever mail system the user is using for their email services. The connect to rep information is sent to the representative to whom the user is in contact with, from whatever company they are currently in contact with. The representative's screen is the final destination of the connect to rep data.

5.2.3 Output Medium and/or Device

The output medium/device for the Google Plus post and the email is any device that can view Google Plus posts and any device that supports the recipients email services.

The medium for the connect to rep is a Chrome add-on that requires any computer that can use the Chrome web browser. The add-on is on the representatives computer and it will pop up the data that the user sends via the application.

Text message output devices are any device that can receive text messages, such as cellular devices, tablets with the appropriate applications and other such devices.

5.2.4 Output Format/Syntax

The format is dependant on what the user chose to share, such as which business, if they chose a specific submenu and such. The Google Plus share function formats the text as plain format.

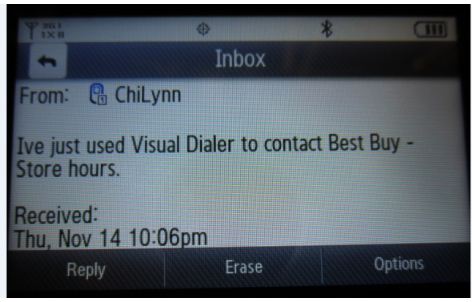
5.2.5 Output Interpretation

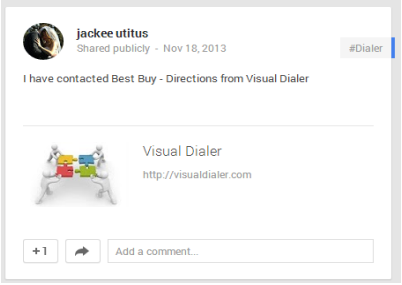
The meaning of the output is just for marketing purposes, and to allow users who chose to to share another random aspect of their lives. It exposes individuals who may have never heard of our application to be introduced to our application. The Google Plus posts have a link to our website which will give those interested a bit more information about what our application does.

5.2.6 Examples

This is an example of the email output screen. This shows the email that Jackee sent to me (Jon) from the application.

This is an example of the text output screen. This shows a text that Chilynn sent to a test phone from the application.





This is an example of the Google Plus output screen. This is showing a post Jackee made on her Google Plus wall

This is an example of the connect to rep output screen. The user is a test file, whose name is Mary Sue, and her address is visible, and the message field is customizable.

For further information about these screens see section 2.3 The User Output Preview.

5.3 System Data Base/File Structure Specification

5.3.1 Identification of Data Base/Files

The Visual Dialer SQLite database is saved within the DatabaseTables.java names as the VisualDialerdb. This is done using the android frameworks SQLite database methods.

The server-side SQL Server 2012 is hosted by Godaddy as part of their shared hosting platform. SQL Server files are stored as mdf and ldf files. We have not implemented anything else such as stored procedures or triggers.

5.3.2 (Sub)systems Accessing the Data Base

The SQLite database is created when the android application is launched. It maintains information even after the application is closed. The database is updated by checking the status of the server side database and if out of sync the tables within the SQLite drop the tables and retrieves the new data from the server side database. The SQLite database is used in most of the java android files every time saved data about the categories, businesses or submenus are accessed. The frequency is high given that on each menu transaction through the main menu the SQLite database is accessed.

On the SQL Server 2012 side, we store tables which are generated by the Microsoft Entity Framework based on our models in our code. The Entity Framework automatically generates the create, retrieve, update and delete methods and ties them to the database. Upon compiling the code, the tables are automatically created on the development environment. Upon fully testing the app, it can be pushed up to the production SQL Server. The code on the production environment will not drop/create tables automatically.

5.3.3 Logical File Structure

The SQLite database mimics the server side database but is more lightweight. Because of the constraints implemented in the server

side database and since the SQLite database retrieves its data directly from the server database, then those constraints don't need to also be put on the SQLite database. This would cause added complexity on the application that are not needed.

On the server-side, all data is stored on the SQL Server 2012 database in tables using typical types such as int and string variables. We do not use blob or text types.

5.3.4 Physical File Structure

The SQLite database is saved on the internal storage of the users android phone device that they have the application saved on. The SQLite database itself does not block other things from accessing it allowing ease of access. Since the SQLite database does not store any private or personal information giving everything access to this database does not create privacy issues. To access this information a DatabaseTables object needs to be created and the open method needs to be called then any queries that are within the class can be used.

On the server-side, all data is stored on the SQL Server 2012 database. Therefore, we don't have any type of physical file structure as far as data goes.

5.3.5 Data Base Management Subsystems Used

The SQLite database itself is not managed in the way that a typical database would be. The SQLite database is an internal storage go between for the server side database that allows the user to work regardless of the internet. It also creates a quicker access to the data for the user making their no need for the user to repeatedly make calls to the internet slowing down the menu transactions within the application.

We use Godaddy's shared web hosting solution. So, the SQL Server 2012 database is fully managed for us.

5.3.6 Data Base Creation and Update Procedure

The system is created in creation of the SQLite database and stored within the phone's internal memory. The update procedure is based off of the server side database. See section 5.3.2 for further details.

On the SQL Server 2012, the database and tables are created by the system. However, the phone menu data is updated manually at

the moment. In the future, we may implement a crawler that dials businesses and automatically populates our database.

5.4 System Internal Data Structure Specification

5.4.1 Identification of Data Structures

The data structure on the Web API layer is written using the MVC pattern. The Microsoft Entity Framework manages the glue between the code and the database. The idea is to write code and let the Entity Framework manage the database and table creation. In general, models have a corresponding table in the database.

5.4.2 Modules Accessing Structures (creating, updating, using)

As stated above, the Entity Framework by Microsoft is used to deal with the database and table generation. In addition, the Entity Framework also automatically generates the create, retrieve, update and delete methods for us. Although, in some cases it is necessary to add to or modify these auto generated methods.

5.4.3 Logical Structure of Data (format, organization, access, rationale, examples)

The data is stored in SQL Server 2012 in tables. Again, everything is managed by the Entity Framework. They are organized with best practices in mind. For example, every table has an id and is associated to other tables using the id field to define relationships. Only the Web API code has access to the SQL Server 2012.

6. Module Design specifications

6.1 Module Functional specification

6.1.1 Functions Performed

- Search
- Action Menu
    - Add to Favorites
    - Share
        - Google+
        - Text
        - Email
- Preference
- Connect To Representative

6.1.2 Module Interface Specifications

Search took in the users input and used it to search the applications SQLite database for any names that was equal to or contained part of the search. If something was found and selected the global variables got updated so the user got redirected back to the menu class where they were able to continue on.

The Action Menu contains the link to the share menu and the add to favorites button. The share enables the user to share their usage of the application with whatever method that they choose, via email, via text, or via Google+. The add to favorites function allows the user to save any company or phone menu item in a favorites list so that the user can reference back to it later.

6.1.3 Module Limitations and Restrictions

The Search function itself is not limited to anything in terms of input; users can search for names, non-case sensitive, or letters or any combination. The user is not able to search with nothing in the search bar and return results.

The add to favorites has several restrictions. the first is that in order to save an item into the favorites list it must be at least a business, the function will not allow the user to save a category. The other restriction is that the application will not allow the user to save the same item.

6.2 Module operational Specification

6.2.1 Locally Declared Data Specifications

The search function takes in data from the text box provided for the user. The data entered into this box does not have any specific format the user has to adhere to however, if the data entered is not specific to names the search might not render any results.

The data entered in the preference screen is allowed to enter any information they want. There is no validation check however, doing this would prove to be not very useful because of the lack of information that can be received from such an act.

The data entered at the connect to rep screen is able to be edited any way the user likes expect for the phone number box. This box sends the user to their number pad for interacting with to enter information.

The share screen where the user enters their friends phone numbers or email address is not limiting on what the user can enter. However, before the email/text is sent the data is formated and verified and if this data is invalid then the email/text is simply not sent.

6.2.2 Algorithm Specification

The search function take each word within the search bar. Then each pertaining section of the SQLite Database is searched for any name containing or equal to any word that was entered in by the user. All possible outcomes are then added to a list and displayed. The general coding is similar to:

```
for (Category i: DatabaseData){
        for(String x: words){
                if(i.get_name().contains(x))
                        found.add(i)
        }
}
for (Business i: DatabaseData){
        for(String x: words){
                if(i.get_name().contains(x))
                        found.add(i)
        }
```

```
}
for (Phone i: DatabaseData){
        for(String x: words){
                if(i.get_name().contains(x))
                        found.add(i)
        }
}
```

The add to favorites function works by getting the selected item to store, checking if its a valid item, checking if its in the list, the adding it. The psuedocode looks something like this:

```
runFav()
{
        get item
        check if item is a category, business or a business submenu item
        check if item is in the favorites list
        add item to list
}
```

The email function works by having the user input the email address of the person they wish to send the email to. The general coding of the email function is:

```
email()
{
        prompts user for email address
        set up email message and sends to email address
}
```

The text function works similarly to the email function. The user inputs the number to which they want to send the text. The general coding for the text function is:

```
sendText()
{
        get phone number
```

```
        set up text message and sends
    }
```

The share with Google+ button opens a menu with three buttons, Sign In , post and sign out. The sign in button works by having the user input their login information from google. The code looks something like this:

```
    signIn()
    {
        requests sign in from google play services
        user sings in
        user client is connected
    }
```

The post works by taking first checking what type of item was selected, then posting. The code looks like this:

```
    post()
    {
        check what type of phone item was selected
        format message with phone item
        posts message
    }
```

The sign out works by simply disconnecting the user client.

```
    signOut()
    {
        client.disconnect
    }
```

6.2.3 Description of Module Operation

See section 2.6.2

7. System Verification

7.1. Items/Functions to be Tested

Android Mobile UI:

- Preferences
- Menu Controller
- Search
- Action Menu

Support Widget:

- visualdialer.supportwidget.bootstrap
- visualdialer.supportwidget.common
- visualdialer.supportwidget.MainCtrl
- visualdialer.supportwidget.Message

7.2. Description of Test Cases

Unit Id	Test Case Id	Test Case Description
Preferences	testMappingPause	Ensure that preference inputs have been properly saved and still persist when the user returns to the paused preference activity.
Preferences	testMappingDestroy	Ensure that preference inputs have been properly saved and still persist after the preference activity has been terminated and restarted.
MenuController	testSetUpCategoreis	Ensures that the data that is in the category menu in the database is being displayed to the user.
MenuController	testSetUpBusinesses	Ensures that the business data that is added in the database is what is getting displayed on the screen for when a category associated to it is selected

MenuController	testSetUpPhoneMenu	Ensures that the phone menu that is displayed is what is in the database the a given business that it is associated to.
MenuController	testSetUpInnerPhoneMenu	Ensures that the if any items are within the phone menu item selected that it is properly retrieved from the database.
MenuController	testOnItemClicked	Ensures that given the menu position the next menu displayed is properly generated.
Search	testSearch	Ensure that the search retrieves an item given it has the same name as what is entered.
Action_Menu	testAddToFav	Checks that no duplicates are added to the favorites list.
visualdialer.supportwidget.bootstrap	testBootstrapInit	Verifies bootstrap adds support widget to element with given id.
visualdialer.supportwidget.common	testCommonInit	Verifies map of supported message to element ids is defined.
visualdialer.supportwidget.MainCtrl	testMainCtrlInit	Checks initial state of main controller.
visualdialer.supportwidget.MainCtrl	testMainCtrlGetPassCode	Verifies call to service for getting a new passcode.
visualdialer.supportwidget.MainCtrl	testMainCtrlDontGetCustomerDataUntilPassCode	Verifies that customer data is not requested from service until a passcode has been retrieved.
visualdialer.supportwidget.MainCtrl	testMainCtrlGetCustomerDataAfterPassCode	Verifies that customer data is requested from service and that elements are set with the returned customer data once a passcode has been retrieved.
visualdialer.supportwidget.	testMessageInit	Checks initial state of message.

Message		

7.3. Justification of Test Cases

Android Mobile UI:

Preferences are evaluated with two test cases (testMappingPause and testMappingDestroy) to ensure that the latest information set by the user has been saved. These preferences will be sent to a customer service user using the Representative Widget so it is important that the information has been properly mapped.

MenuController is evaluated for all the set up test cases guaranteeing that what is in the database is what is getting displayed. Menu controller is also evaluated for the onItemClicked method making sure that the correct menu is being displayed when the user is requesting it.

Search is tested for the checkTables method checking that the item that is in the database is able to be found given that items name is used to search for it.

The Action Menu is tested to ensure that the Favorites table in the Database is not able to have duplicate data. In doing this we don't give the user unneeded information and reduce the clutter on their favorites list.

Support Widget:

visualdialer.supportwidget.bootstrap is evaluated with one test case to verify that a new instance of the Support Widget is added to the element with given id. It also checks that the given map of supported message to element ids is saved.

visualdialer.supportwidget.common is evaluated with one test case to verify that map of supported message to element ids is defined. This map is updated by visualdialer.supportwidget.bootstrap and used by visualdialer.supportwidget.MainCtrl.

visualdialer.supportwidget.MainCtrl is evaluated with four test cases to check the initial state, to verify the call to service for getting a new passcode is executed as expected, to verify that customer data is not requested from service until a passcode has been retrieved, and to verify that customer data is requested from service & that elements are set with the returned customer data once a passcode has been retrieved. All the controller logic for the Support Widget is defined in visualdialer.supportwidget.MainCtrl.

visualdialer.supportwidget.Message is evaluated with one test case to check the initial state. visualdialer.supportwidget.Message is a POJO based on the web API common message JSON.

7.4. Test Run Procedures and Results

Test Case Id	Expected Result	Actual Result	Remarks
testMappingPause	Test data input in preferences (name, address, etc.) exists in preferences after a `callActivityOnResume` call	PASS	
testMappingDestroy	Test data input in preferences (name, address, etc.) exists in preferences after an activity `destroy` call	PASS	
testSetUpCategories	The category object is the same as what is retrieved when `setUpCategories` is called	PASS	
testSetUpBusinesses	The business object is the same as what is retrieved when `setUpBusinesses` is called	PASS	
testSetUpPhoneMenu	The category object is the same as what is retrieved when `setUpPhoneMenu` is called	PASS	
testSetUpInnerPhoneMenu	The category object is the same as what is retrieved when `setUpInnerPhoneMenu` is called	PASS	
testOnItemClicked	Given a menu position and a prefilled displayed names the correct menu is displayed when `onItemClick()` is called	PASS	
testSearch	Given an item in the database with the same name you are searching for the `checkTable` call retrieves it.	PASS	

testAddToFav	Returns a message that the item is already in the favorites list	PASS	
testBootstrapInit	A new Support Widget instance is created and added to element with given id. Saves given map of supported message to element ids.	PASS	
testCommonInit	Defines global map of supported message to element ids.	PASS	
testMainCtrlInit	Clears current passcode and sets status messages to “Getting passcode...”.	PASS	
testMainCtrlGetPassCode	Calls service for getting a new passcode.	PASS	
testMainCtrlDontGetCustomerDataUntilPassCode	Does not call service to get customer data until passcode has been retrieved.	PASS	
testMainCtrlGetCustomerDataAfterPassCode	Calls service to get customer data once passcode has been retrieved. Sets element text using retrieved customer data.	PASS	
testMessageInit	Sets member variables based on given JSON.	PASS	

7.5 Discussion of Test Results

- Preferences

The Preferences were tested to guarantee that information being entered into the Preference screen’s text fields persisted while the application was in different states. Preferences were tested for the case in which the user pauses the current Activity (the Preference screen) and then returns to the Activity. Preferences were also tested for the situation in which the Activity has been terminated (eg. the Visual Dialer application has terminated) and then restarted. The test input strings for the Preference text fields are compared with those

stored in the application Preferences after the interrupting action (`callActivityOnResume` or `destroy`) has taken place.

The success of these two tests guarantees that the information input by a user into the Preferences screen is mapped to the Preferences in different Activity states. Preferences are an important feature of Visual Dialer because storing contact information in the application can make the Connect to Rep process more efficient for users and representatives.

- Category and Business Menus

The Menu controller was tested to guarantee that the information that was added to the database was actually being retrieved and displayed to the user. This was tested for categories, businesses and businesses submenus. The Menu Controller was also tested for items being clicked. Making sure that given data in the database at each stage the item being clicked would properly needed to the items to be displayed at the next stage. This also check to make sure the selected business and the selected business items, which are used outside of this class, are properly selected.

The Search menu was tested to show that given an inputted testing string breaking it up and checking that information is found. Given that we know what is in the database and we use it's name to search for that item that it returns the item and displays it to the screen.

- Action Menu

The add to favorites was tested to check that if a phone menu item trying to be added to the favorites list is already in the list it would not allow it to be entered another time. We check to see if an item is in the list by using the phone items name and enters it in the favorites list if it is not in the list, and returns a message if the selected item is already in the list.

8. Conclusions

8.1 Summary

In general, the team accomplished what it wanted to with this project. We had envisioned having a mobile application that would allow an end user to easily see a business' phone menu system and allow them to traverse through the menu visually instead of listening over the phone. Once an item is selected, they are automatically connected. We also envisioned a way to save a menu item into favorites and also be able to share with friends. A second major feature was a way for an end user to send pertinent information about themselves to a business' representative with a click of a button. All these were achieved and in the midst we learned a lot of new technologies. Some of the major ones included, Java Angular JS, Android, IVR phone processing, Unit Testing, Microsoft SQL Server, Microsoft .NET Web API, and Microsoft Entity Framework.

The team also followed the Agile SCRUM methodologies. This allowed us to stay focused and efficient through out. Each release was considered a sprint (3-5 weeks). We had weekly standup meetings using Google hangouts. Almost 100% of the items that the team committed to was delivered on time. Others where obstacles were seen had to be researched and successfully resolved to meet our releases.

8.2 Problems Encountered and Solved

We did run into a few issues. Initially we had decided on Java technologies for the web services and the use of Godaddy's shared hosting. While attempting to setup an initial framework, we realized that Godaddy does not support Java on their shared hosting systems. So we decided to switch and use Microsoft .NET.

Another issue that we ran into was the integration of Google Play Services for the sharing feature.The main issue we had with integrating the Google Play Services was that the version that we had installed had become outdated a few days after installing it. The solution was to install the most recent version and once the new version was installed some of the coding had to be modified in order to run. Another issue arose from the modification of the code as well though. Between the machines in our group, one member had to have a value hardcoded, while the others were able to run the application using a reference call.

8.3 Suggestions for Better Approaches to Problem/Project

One area that we wish we would have done differently was the way information is transmitted between the end user and the representative for the support widget. We used SQL server as a means to store information

in a temporary table. This would have been much suited for Queues. If this project continues to be worked on after this class, Queues will be implemented.

8.4 Suggestions for Future Extensions to Project

If this project continues after this class, an iOS version would definitely be developed. In addition, the team feels that a means for entering and managing phone menus for businesses is also necessary. One option is to create a crawler, similar to how search engines crawl the web, but with the use of the IVR servers to dial businesses and use voice recognition to read in the phone menus. A person would then need to verify the data and mark it as valid. Finally, an sdk can be opened up to other developers and companies so they can integrate Visual Dialer into their own applications. Our Web API layer has been developed in a way where it would be pretty easy to setup an SDK.

9. Appendices

9.1 Use Cases

Use Case ID:	001
Use Case Name:	Open Application
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Primary User Interaction
Actor:	User
Stakeholder:	User wants to load his application.
Description:	Application loads and loads the information to the SQLlite database from the server.
Preconditions:	User must have application installed on their android smart phone version 2.3 or higher
Post conditions:	Menu loads
Fail-condition:	Application cannot connect to the internet
Priority:	High
Frequency of Use:	Often
Normal Course of Events:	1. Player opens application from their phone. 2. Visual Dialer is Displayed. 3. Application gets status from server. 4. Application transitions to Menu.
Alternative Courses:	3.a Application cannot get status from server. Application uses cached data for the menu. 3.b Application gets a newer status from the server. Application adds new data to the SQLite database.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	002
Use Case Name:	Transition through menu
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Primary user interaction
Actor:	User
Stakeholder:	User wants to transverse the menu
Description:	The user touches the menu and transitions between categories, businesses and phone menu options for businesses to call the desired business with the correct dial sequence.
Preconditions:	Data needs to be added to the SQLite database from the server.
Post conditions:	User gets sent to an action menu.
Fail-condition:	Menu item does not have items that branch off but does not lead to a dial sequence.
Priority:	High
Frequency of Use:	Often
Normal Course of Events:	1. User selects the category for the business they are looking for. 2. User selects the business they are looking for. 3. User selects a phone menu item. 4. User gets redirected to an action menu for further options.
Alternative Courses:	4.a User transverses through inner phone menu items. User gets to the end of the phone menu items and goes to the action menu.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	003
Use Case Name:	Search
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to search for a specific item.
Description:	The user enters in what they are looking for.
Preconditions:	Data needs to be added to the SQLite database from the server.
Post conditions:	User gets set to an action meu.
Fail-condition:	User enters information that is not in the database.
Priority:	Low
Frequency of Use:	Seldom
Normal Course of Events:	1. User types in what they are looking for. 2. Everything from the database that contains part of any word search is returned. 3. User selects the item that they were looking for. 4. User gets redirected to an action menu for further options.
Alternative Courses:	2.a Nothing is found from the search user must try again. 4.a User selects an item with a future internal items. Use case 002 Transition Through Menu
Special Requirements:	Android API 2.3 or higher

Use Case ID:	004
Use Case Name:	Action Menu
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Primary user interaction
Actor:	User
Stakeholder:	User wants to call a selected item.
Description:	The user is at an action menu were they can either call now if they selected a business or a business's menu item. They can add the current selection to a favorites list for quicker access. They can connect to a representative if they business they are calling access's visual dialers chrome add-on. Or, they can share with their friends through email, text message or Google+ that they are using our application.
Preconditions:	Data needs to be added to the SQLite database from the server.
Post conditions:	User gets sent to share, add to favorites, connect to representative or call now.
Fail-condition:	Sequence is not a valid selected item.
Priority:	High
Frequency of Use:	Often
Normal Course of Events:	1. User presses the dial now button. Use case 009 Call Now
Alternative Courses:	1a. User presses the favorites button. Use case 006 Add to favorites. 1b. User presses the connect to rep button. Use case 013 Connect to rep 1c. User presses the share button.

	Use case 014 Share.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	005
Use Case Name:	Favorites Menu
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to call a business they call frequently.
Description:	The user is given a list of items they added to favorites and wants to call a business.
Preconditions:	User has added some items to their favorites list.
Post conditions:	User gets sent to an action menu.
Fail-condition:	The user has not added any items into their favorites list.
Priority:	Low
Frequency of Use:	Seldom
Normal Course of Events:	1. User selects the business or business menu item that they want to call. 2. User is directed to the action menu with the selected item that from the favorites list. Use case 004 Action Menu
Alternative Courses:	1a. User presses the menu button on the phone. User presses the clear button from the menu.

	The user's favorites list is emptied.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	006
Use Case Name:	Add to favorites
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to quickly call a business.
Description:	The user tries to add the current business or business menu item to their favorites list for quicker calling purposes.
Preconditions:	User is currently at the action menu.
Post conditions:	User stays at the action menu.
Fail-condition:	User has not selected a business.
Priority:	Low
Frequency of Use:	Seldom
Normal Course of Events:	1. The user presses the favorites button. 2. The current item is added to the favorite menu.
Alternative Courses:	2a. The current item is already in the favorites menu. The item is not added and a message is displayed to the user. 2b. The user has not selected a business. The item is not added and a message is displayed to the user.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	007
Use Case Name:	Menu Button - Main Menu/Action Menu
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to access the action menu or other options.
Description:	The user touches the menu button on the phone and is given the option to: go to the favorites menu, go to the call menu(action menu), go to the search menu, update their user preferences, or connect to a representative.
Preconditions:	The user is either at some point in the main menu or at the action menu.
Post conditions:	User gets sent to an action menu.
Fail-condition:	User doesn’t select one of the menu items.
Priority:	Medium
Frequency of Use:	Occasionally
Normal Course of Events:	1. User selects call menu button. Use case 004 Action Menu
Alternative Courses:	1a. User selects the favorites button. Use case 005 Favorites 1b. User selects the search button. Use case 003 Search 1c. User selects the preferences button.

	Use case 015 Preferences 1d. User selects the Connect to rep button. Use case 013 Connect to rep.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	008
Use Case Name:	Menu Back Button
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Primary user interaction
Actor:	User
Stakeholder:	User wants to go back a stage in the main menu.
Description:	The user presses the back button on the phone and the main menu goes back to the previous state of the main menu.
Preconditions:	Data needs to be added to the SQLite database from the server.
Post conditions:	User gets stays in the main menu.
Fail-condition:	Previous menu items cannot be recovered.
Priority:	High
Frequency of Use:	Often
Normal Course of Events:	1. User is at a stage at the main menu and presses the back button. 2. Previous items from the menu are shown and the menu position is decremented. 3. Steps 1 and 2 are repeated till at the desired location

	in the menu.
Alternative Courses:	3a. User presses back at the category submenu of the main menu. Application is exited.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	009
Use Case Name:	Call Now
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Primary user interaction
Actor:	User
Stakeholder:	User wants to call a business
Description:	User calls a business and the sequence of buttons are pressed for the user then the user is connected to the call.
Preconditions:	User need to be at the action menu.
Post conditions:	User is calling the business.
Fail-condition:	A business has not been selected.
Priority:	High
Frequency of Use:	Occasionally
Normal Course of Events:	1. User has pressed the call now button. 2. User gets redirected to a menu where they can call. 3. User presses the call button. 4. User calls and waits will the correct sequence of

	buttons are pressed. 5. User is connected to the call where they are directed to the department they want to talk to.
Alternative Courses:	3a. User presses the back button and gets directed back to the action menu. Use Case 004 Action Menu
Special Requirements:	Android API 2.3 or higher

Use Case ID:	010
Use Case Name:	Share with Google+
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to transverse the menu
Description:	The user touches the menu and transitions between categories, businesses and phone menu options for businesses to call the desired business with the correct dial sequence.
Preconditions:	Data needs to be added to the SQLite database from the server.
Post conditions:	User gets sent to an action menu.
Fail-condition:	Menu item does not have items that branch off but does not lead to a dial sequence.
Priority:	Low
Frequency of Use:	Seldom

Normal Course of Events:	1. User presses the Google+ button. 2. User is directed to the Google+ page. 3. User Presses the Sign in button. 4. User then presses the post button. 5. A pop up window gives the user the option the edit the text, linked page, location, tag friends and other options that Google+ provides. 6. User presses the post button from Google+. 7. User presses the sign out button. 8. User is directed back to the share menu.
Alternative Courses:	3a. User presses the sign out button. User is directed back to the share menu. 3b. User presses the post button. An error message pops up.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	011
Use Case Name:	Share with email
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to share visual dialer with their friends.
Description:	The user wants to share visual dialer with their friends and does so by sending them an email.
Preconditions:	User is in the share menu.
Post conditions:	User stays in the share menu.
Fail-condition:	User enters an invalid email address.
Priority:	Low

Frequency of Use:	Seldom
Normal Course of Events:	1. User presses the Share with email button. 2. A pop up box shows prompting the user for an email address. 3. User enters an email address. 4. User presses OK. 5. User gets sent to phone's email application. 6. User is given a chance to edit the text of the email. 7. User presses the send button. 8. User gets redirected back to the share menu.
Alternative Courses:	4a. User presses cancel User gets redirected back to the share menu. 7a. User presses cancel User gets redirected back to the share menu.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	012
Use Case Name:	Share with text message
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to send a text about Visual Dialer
Description:	The user is able to enter a phone number to text their friends about Visual Dialer.
Preconditions:	User need to be at the share menu
Post conditions:	User stays at the share menu
Fail-condition:	User enters an invalid phone number

Priority:	Low
Frequency of Use:	Seldom
Normal Course of Events:	1. User presses the Share with Text button. 2. A pop up box shows prompting the user for a phone number. 3. User enters a phone number. 4. User presses OK. 5. A generated message is created and sent. 6. User is directed back to the share menu.
Alternative Courses:	4a. User presses cancel User gets redirected back to the share menu.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	013
Use Case Name:	Connect to rep
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Primary user interaction
Actor:	User
Stakeholder:	User wants to send information to a business.
Description:	The user is able to send information that they can edit to a business given they have the Visual Dialer Google Chrome add-on.
Preconditions:	User is at the connect to rep menu.
Post conditions:	User stays at the connect to rep menu.
Fail-condition:	Business does not have the Google Chrome add-on.
Priority:	Low

Frequency of Use:	Occasionally
Normal Course of Events:	1. User presses the connect to rep button. 2. The user enters information in the given boxes. 3. User gets the Rep ID from the business. 4. User enters the Rep ID 5. User presses the send to rep button. 6. The information that is filled out is sent to the rep.
Alternative Courses:	2a. The information is present and the user doesn't have fill in the boxes.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	014
Use Case Name:	Share
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to share Visual Dialer with their friends.
Description:	The user wants to share Visual Dialer with their friends either through email, text message, or Google+.
Preconditions:	User is at the share menu.
Post conditions:	User gets redirected to one of the share sub menus.
Fail-condition:	
Priority:	Low
Frequency of Use:	Seldom
Normal Course of Events:	1. User presses the share with Google+ button. Use case 010 Share with Google+.

Alternative Courses:	1a. User presses the share with email button. Use case 011 share with email 1b. User presses the share with text button. Use case 012 share with text
Special Requirements:	Android API 2.3 or higher

Use Case ID:	015
Use Case Name:	Preferences
Created By:	Jackee Utitus
Date Created:	11/22/13

Scope:	Visual Dialer
Level:	Secondary user interaction
Actor:	User
Stakeholder:	User wants to pre-save information for the connect to rep button.
Description:	The user is given the option to save preset information such at the user's name, phone number, address, and other information.
Preconditions:	The user is at the preference menu.
Post conditions:	User returns to the previous menu.
Fail-condition:	The user enters invalid information.
Priority:	Low
Frequency of Use:	Seldom
Normal Course of	1. The user selects a section that they want pre set.

Events:	2. A box pops up gathering information for the given selection. 3. User enters information. 4. User presses ok. 5. Repeat steps 1 through 4 till all the desired selected section have data. 6. User presses the back button on the phone. 7. The user is returned to the previous menu.
Alternative Courses:	3a. Information is already there.User deletes the old data and updates the information. 4a. User presses cancel. Updated information is not saved.
Special Requirements:	Android API 2.3 or higher

Use Case ID:	016
Use Case Name:	Support Rep Session
Created By:	Utsav Jain
Date Created:	11/22/13

Scope:	Visual Dialer Support Widget
Level:	Primary user interaction
Actor:	Support Rep
Stakeholder:	Support Rep gets a call from a customer and wants to start a new session.
Description:	The lifecycle of a support session from the point of view of the support rep.
Preconditions:	The user is not currently in a session.
Post conditions:	User clicks the button to stop the current session.
Fail-condition:	There is no Internet connection of the web API is nt available.

Priority:	High
Frequency of Use:	Often
Normal Course of Events:	1. The support rep is given the option to get a new passcode. 2. When the support rep clicks the button, and call to the web API is made to get a new passcode. 3. Once it comes back, the Support WIdget displays the passcode in the UI and starts polling for customer data. 4. The support rep can now tell the user to passcode so they can connect via Visual Dialer (use case 015). 5. As the user sends information via Visual Dialer the Support Widget displays it in the same page. 6. Once the support session is done the support rep stops the session by clicking a button.
Alternative Courses:	
Special Requirements:	

10. Program Listing

Here's a list of internal API calls that are available from the WebAPI layer. The calls can be made to http://<web_api_root>/ followed by below listing.

PhoneMenus

GET api/PhoneMenus
GET api/PhoneMenus/{id}
GET api/PhoneMenus?businessId={businessId}
PUT api/PhoneMenus/{id}
POST api/PhoneMenus
DELETE api/PhoneMenus/{id}

RepConnectPasscodes

GET api/RepConnectPasscodes
GET api/RepConnectPasscodes/{id}
PUT api/RepConnectPasscodes/{id}
POST api/RepConnectPasscodes
DELETE api/RepConnectPasscodes/{id}

UserSelections

GET api/UserSelections
GET api/UserSelections/{id}
GET api/UserSelections?userCallerID={userCallerID}
PUT api/UserSelections/{id}
POST api/UserSelections
DELETE api/UserSelections/{id}

Categories

GET api/Categories
GET api/Categories/{id}
PUT api/Categories/{id}
POST api/Categories
DELETE api/Categories/{id}

RepConnectMessages

GET api/RepConnectMessages
GET api/RepConnectMessages/{id}

GET api/RepConnectMessages?passcode={passcode}
PUT api/RepConnectMessages/{id}
POST api/RepConnectMessages
DELETE api/RepConnectMessages/{id}

Businesses

GET api/Businesses
GET api/Businesses/{id}
GET api/Businesses?categoryId={categoryId}
PUT api/Businesses/{id}
POST api/Businesses
DELETE api/Businesses/{id}

RepConnectNameValues

GET api/RepConnectNameValues
GET api/RepConnectNameValues/{id}
GET api/RepConnectNameValues?repConnectMessageId={repConnectMessageId}
PUT api/RepConnectNameValues/{id}
POST api/RepConnectNameValues
DELETE api/RepConnectNameValues/{id}

RepConnectMessageValues

GET api/RepConnectMessageValues
GET api/RepConnectMessageValues/{id}
GET api/RepConnectMessageValues?repConnectMessageId={repConnectMessageId}
PUT api/RepConnectMessageValues/{id}
POST api/RepConnectMessageValues
DELETE api/RepConnectMessageValues/{id}

Main Menu

The first interactable screen that you will get to is the main menu. This menu allows you to navigate the menu looking through categories for business and looking through businesses for particular options that are available to call within that business. This allows you to visual go through call menu's to find exaction what you are looking for.

Navigating the Menu

Initially, you are at the Category Menu screen, see figure 1, where you can select the category to which the business you are looking for is under. If initially you do not see the category that you are looking for you can scroll down to see more options. To scroll through the menu you simply slide your finger in an upwards manor to see more options. Once you find the correct category you believe your business to be under you simply Select that options. To do this, simply touch the section of the screen you see the category at. In the example below the Computers and Electronics sections has been pressed.

Once you have selected a category that you believe your business is under you are directed to the Business Menu screen, see figure 2; this is where you can select a business you are looking for. Same as for category if you would like to scroll down to see more options simply slide your finger in an upwards manor. If you do not find the business you are looking for press the phones back button and you will return to the Category Menu screen where you can choose again. If you do find the business you are looking for select it by touching that section of your screen. In the example below Best Buy was selected for the business.

If you have selected a business then you are directed to that businesses phone menu, see figure 3. Here you can see the initial phone menu options. Some options have inner options themselves. Do not be afraid to select an item, since, at any point in the menu if you believe you have selected the wrong option you can press the back button on your phone. At any point in this process you can select more options by pressing the menu button, see Phone Menu Button under the Main Menu sections. Once a phone Menu has been selected that does not have a sub menu the user is redirected to the action menu, see the Action Menu section.

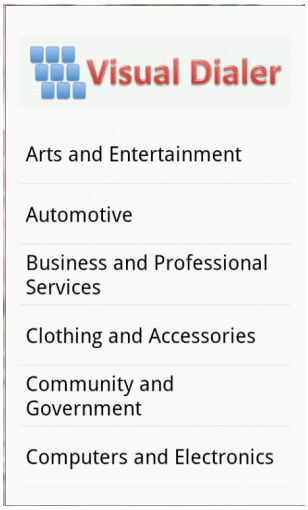


Figure 1: Category Menu

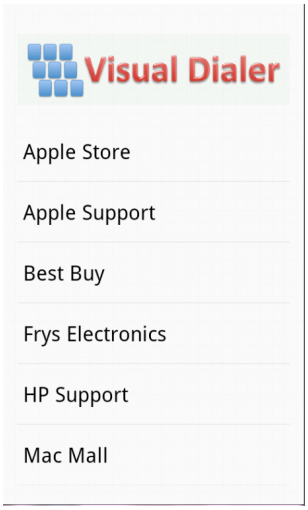


Figure 2: Business Menu

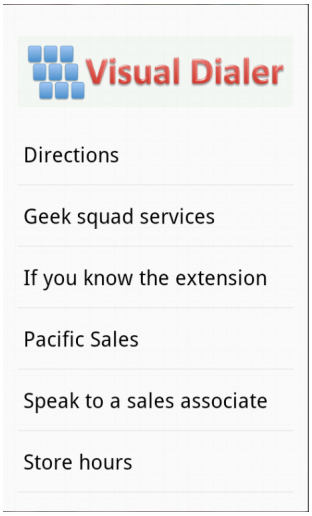


Figure 3: Phone Menu

Phones Menu Button

At any point in the Main Menu or the Action Menu, see Action Menu, you have the option of pressing the Menu button on the phone. The Menu button is location at the bottom portion of your phone and looks like a list of items, see figure 4. If you press this button you are given five additional options, see figure 5. You are given the option to go to the Call Menu, see Action Menu, you are given the option to go to Favorites, see Favorites, you are given the option to Search, see Search, you are given the option to update your Preferences, see Preferences, and you are given the option to Connect To Rep, see Connect to Rep. Any of these options can be selected by touching the portion of the screen the menu item you choose is located at.

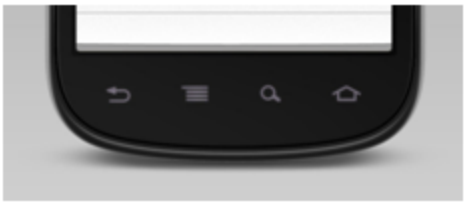


Figure 4: Menu Button

Figure 5: Menu screen

Action Menu

At this point you are at the Action Menu also referred to at the Call Menu, see figure 6. This menu is where you have the option of action interacting with the selected business or business phone menu you have selected. If you have gotten here without selecting a business or business phone menu item then many options will be disabled for you. If this is the case messages will appear across your screen notifying you. If you have selected a business the name will be displayed across the top. In this example you can see Best Buy – Direction

which was selected in the previous example, see Main Menu. Here you are able to interact with such buttons such as the Dial Now button, see Calling, the Connect to Rep button, see Connect to Rep, the Share section, see Share, and the Add To Favorites button, see adding to Favorites.

Figure 6: Action Menu

Calling

At this point you are directed to your phones call application. At this application you have the option to press the phone icon to connect the call. You can also press the phones back button this gives you the option to not make the call. Once the call goes through you will be waiting because the call will press the given sequence for you and connect you once you are at the sequence that you have chosen. While this call is being made, you will not hear anything till you are connected. For items that have a longer sequences this might take a longer time so be patient and you will be connected to the given sequence.

Adding to Favorites

You have the option to touch the Add to Favorites button. This option will take whatever you have selected to get to the Action Menu and add that sequence into the favorites list. If the sequence you have selected is already in the favorites list then you are not able to add it in a second time and you will have a message displayed letting you know that selected is already in the favorites list.

Sharing

You have the option to touch the Share button. Here you are directed to the Share Menu, see figure 7. You are able to share with your friends that you are using the Visual Dialer application. Here you are given the option to Share with Google+, see Share through Google+, the option to Share With eMail, see Share through Email, and the option to Share with Text, see Share through Text.

Share through Text

Here you will see a pop up box that asks you for your friends phone number where you can text them at. Enter in the ten digit number of your friend and press the Ok button. This will automatically generate a text message and send it to them. If you press the cancel button you are directed back to the Share Menu.

Figure 7: Share Menu

Share through Email

After you press the Share with eMail button a pop up box will appear prompting you for your friends email address. Enter in your friends email address and press the Ok button. You also have the option to touch the cancel button in which you are redirected back to the Share Menu. If you press Ok you will be redirected to your on phone email application where you are able to edit the pre-generated email message and the email address. You are given the option to press send or back out of the application and get directed back to the Share Menu.

Share through Google+

If you press the Share with Google+ button you are redirected to a Google+ sign in page, see figure 8. When you press the Sign In button, you will be automatically signed into your Google+ account based off of your information stored on your phone. If it is your first time using this function you might be prompted to allow access to your Google account, this is because of the setting of your phone. If you press the Post button an automatically generated post is displayed for you. The content of the post is displayed showing your use of Visual Dialer, however, you have the option to edit this. Also the post has attached to it the website for Visual Dialer this too can be changed. You can also edit the location, who you are with and other such sections by touching the option and entering in information.

Figure 8: Google+

Favorites

You are able to save specific businesses or businesses phone menu options that are used most frequently to a favorites list. To add a business to this list you must do this through the Add to Favorites button in the Action Menu, see Add to Favorites. Your favorites list does not have a maximum size so you can store as many businesses you think are needed for convenience of access. Only one of each business or business phone menu option can be added to the favorites list. For example, your favorites list can consist of Best Buy and Best Buy – Directions since they are different sections because of the sub menu. However, your favorites list cannot consist of Best Buy, Best Buy, and Best Buy – Directions since the business by itself cannot be added more than once. You have the option to clear your favorites list, see Clear Favorites, and you have the option to interact with an item in your favorites list, see Click an Item

Clicking an Item

At the favorites list your main option is to touch an item in your list. If you have added an item to the list and do not see it simply scroll down by touching your screen and moving your finger upwards. When you select an item on the list, you are directed to that items Action Menu, see Action Menu, where you are given many new options.

Clearing Favorites

The Clear Favorites function is available under the Favorites section of Visual Dialer. To get to this option you must press the menu button on your phone. This will bring up a menu where you are given the option to select the Clear Favorites

button, see figure 9. If you press the button your entire Favorites list will be deleted but you have the option to re-enter the information that was previously in this list.

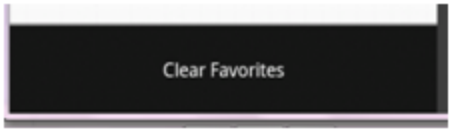


Figure 9: Clear

Search

At the search function you are able to enter in information that you can search for, see figure 10. At the box in the upper screen, simply touch the region to start entering text that you want to search for. Once you have entered in the text that you want to search for touch the Search button. You can enter multiple words to enter what you are looking for. Each word entered is checked independently of each other. The search checks for all data in its database and shows you want words contain any part of what was entered. In figure 10, you can see that simply ‘a’ was entered and the results returned all options that contained the letter ‘a’. If this did not render the results desired you can touch the text box and enter in new information.

Figure 10: Search

Selecting an Item

Once you have entered in the information that you are looking for and have pressed the Search button results are shown to you in a list. Some results may not be shown, thus, scroll down the list by moving your finger across the screen in an upwards direction. If there is a result that you would like to interact with simply touch that portion of the screen on your phone. Once a selection has been made you will either be redirected to the Main Menu, see Main Menu, if there are further sub menus to be displayed or you will be redirected to the Action Menu, see Action Menu, where you are given more options to choose from.

Connect to Rep

Once at the Connect to Rep screen you will see a screen similar to figure 11.

This is information that the user can send directly to the representative thus guaranteeing information such as name, address etc does get transferred as they mean it to be. Here you are given many different option of boxes to fill to to sent to a representative. You also have the option of clicking the Preferences button, see Preferences, to have this information auto filled out for each time using the send to representative option of Visual Dialer.

Figure 11: Connect to Rep

Entering Information

Each box is completely optional. Some boxes will be pre-filled out if you have previously set up your preferences.

The data can be change in each box by interacting with them and entering new information. You are also given an option to add comments which are not saved. This comment section is data that you can send to the representative that is not changed. Here you can send information such as account numbers, issues with products you are having or any other information. This information is not saved through Visual Dialer so your information is secure. All information that is entered into the comments section is completely voluntary and not required.

Sending Information

To send information you actually need to be in contact with a representative who is using the Visual Dialer Google Chrome add-on. When contacting this representative they will give you a code needed to send the information to the representative you are in contact with and that they will be the only one getting it. This code is entered into the bottom most box in the connect to rep screen; in figure 11 the code given is 1E87. After you enter this code you press the Send to rep button at the bottom right. This button will send all entered information shown

in the boxes above. Any information that you wish not to be shared, merely remove the information to be shared. If you wish the exit this screen, press the phones back button and you will be returned to the previous screen you were at.

Preferences

The preferences screen allows you to have key information that might be used multiple times as seen in figure 12. This information is only used in the Connect to Rep screen, see Connect to Rep, and never stored outside of your personal device. These preferences are for convenience purposes only and are completely voluntary with the ability to enter and update information at any time.

Entering/Updating information

To enter information the user must click the symbol on the right side of the screen associated with the information you want to update. Touching this simple brings up a pop up box. If preferences was previously filled out then you have the option of deleting the old information and adding new or leaving the information alone. If this is the first time filling out the preferences section then just enter in the information that you would like stored by touching the box and using the phones keyboard device.

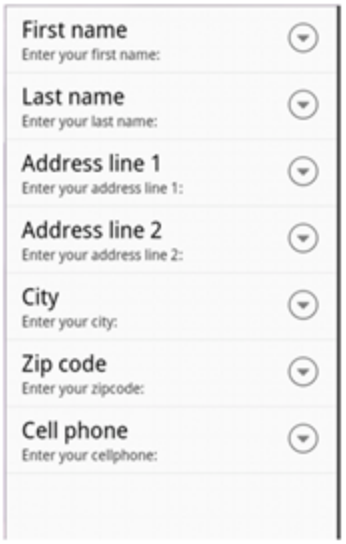


Figure 12: Preferences