

Software Design Document

Smart Dialer Application

CSC 4330

Prepared by Cameron Moore, Omar Qasem,
Patrick Richardson, Matthew Spedale, and Jeffrey Tolliver

Submission Date: April 2nd, 2017

Table of Contents

1. Introduction
2. Main System Architecture
3. Sub-System Architecture
4. Logical View
5. Development View
6. Physical View
7. Data View
8. Work Assignment View
9. User Interface
10. Element Catalog

1. Introduction

1.1 Purpose

This document outlines the implementation for the Android application "Smart Dialer" as described in the SRS Document. It shall inform all developers of the intended form of the application. It also outlines the main system and sub-system architectures, the logical, development, physical, data, and work assignment views, and the user interface of the project and application. An element catalog is available at the end of this document to define the icons that are used in each architecture view diagram.

1.2 Scope

The Smart Dialer application is designed to run on Android mobile devices. This document displays information about the application, specifically concerning the software design and architecture used to develop it. The product is intended to simplify mobile phone calls, and provide more information to the application user upon receiving or making a call than what is provided by the default caller application that comes preinstalled on Android devices. The main feature of the application is an automated reverse phone number (411) lookup. When the user makes a phone call, Smart Dialer stores the number inputted, performs a reverse phone number lookup on it for relevant information (company name, location, thumbnail image, and email) and stores it in the history list. Additionally, features such as a contact list, call logs, and speed dial are also integrated into Smart Dialer to ensure that users do not need to use both Smart Dialer and the default caller application depending on what functions they want to use.

1.3 Reference

- IEEE Software Engineering Standards Committee, IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- Smart Dialer SRS Document
- Dr. Nash's System Design PDF Presentation
- "Data Model as an Architectural View", Paulo Merson, Oct. 2009. Retrieved from <http://www.sei.cmu.edu/reports/09tn024.pdf>

2. Main System Architecture

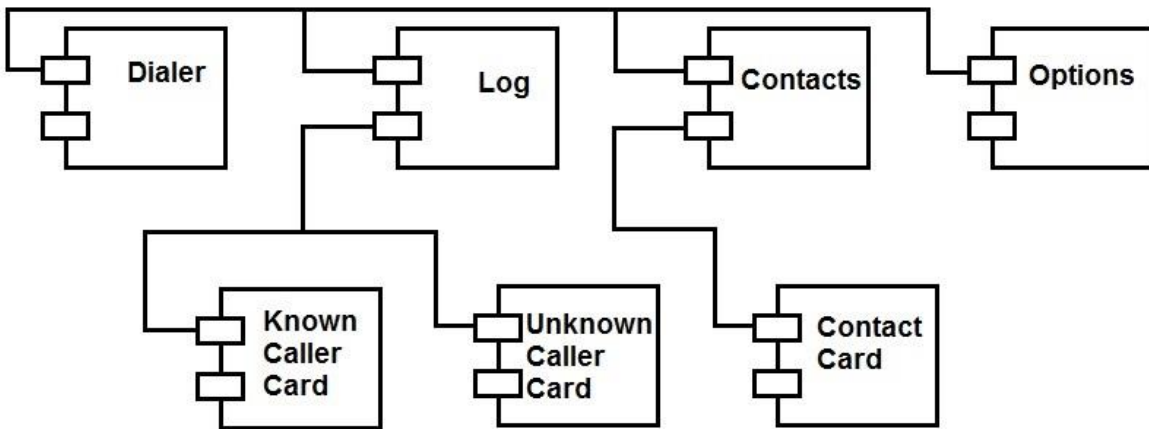
The Smart Dialer application utilizes Object Oriented Design philosophies to ensure that we can simplify and encapsulate as much of our code as possible. Encapsulating the code allows us to classify and separate entities by their functions, making individual work much simpler. In addition, the application also uses Implicit Invocation philosophies, as event-based interactions with agents and listeners are necessary for an interactive application. Smart Dialer is developed using the Android Studio IDE using the Java programming language.

The application opens onto the main screen, which is the dialer component. From here, the user can input phone numbers to make calls, similarly to how most built-in phone calling applications work. Using a tab system at the top of the screen, the user can also change to the log or contacts screens. The log screen contains a history of sent, received, and missed phone calls. Each call will contain the phone number and name of the caller, and can be tapped to call back, or held down to open another panel with more information, such as a thumbnail image, location, and email. The contacts screen contains the user's saved phone contacts, displaying the name, phone number, and thumbnail image of each contact. These contacts can be called by tapping them, and holding them allows editing. The user can create a speed dial list of their contacts for quick calling as well, and add additional contacts using a button at the bottom of the screen. Inside of the application, there is also an options screen that allows users to change settings to fit personal preferences.

3. Sub Systems Architecture

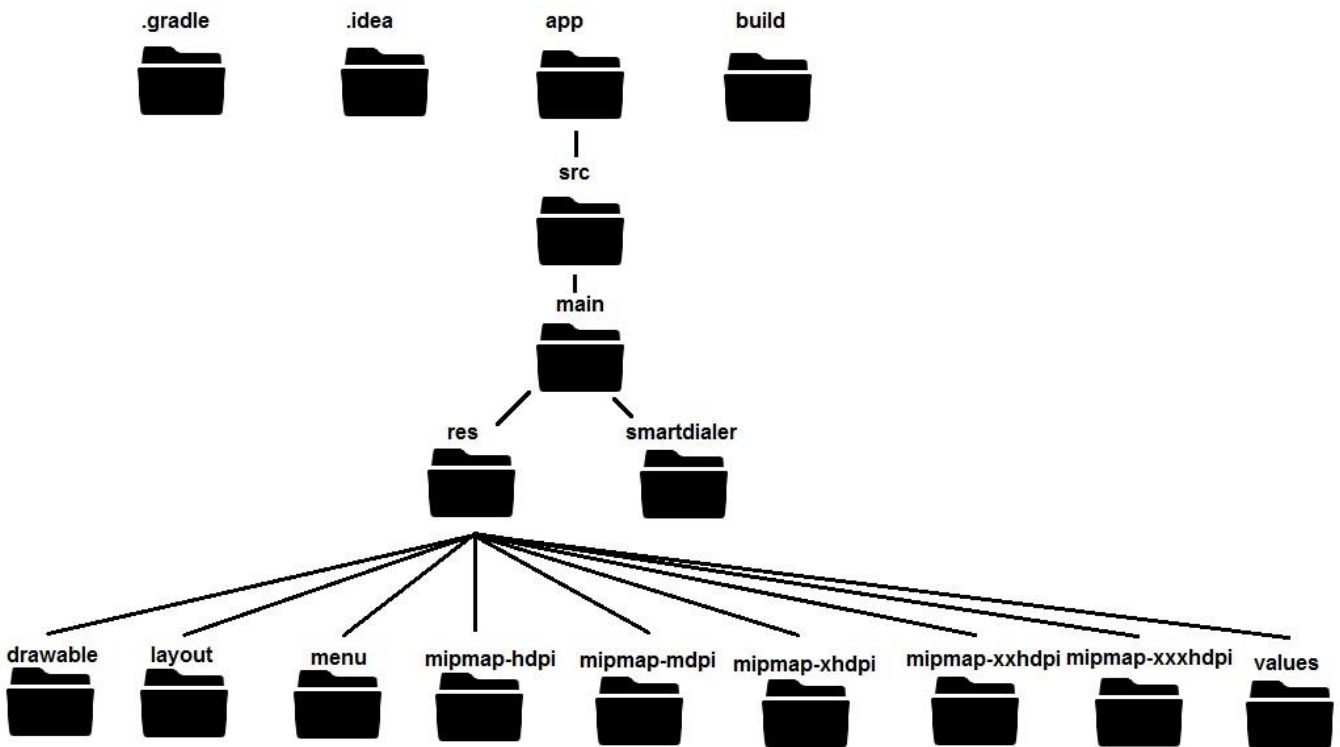
The sub-system architecture of Smart Dialer involves the use of the Whitepages Pro API Database, as well as integration of the default caller application that comes preinstalled on any Android device. Smart Dialer uses the Whitepages API when performing the reverse phone number lookup. Whitepages will provide the phone numbers, contact information, email and physical address for each call the user makes. This allows us to have a reliable database with a consistent format, making referencing information to pull much easier than trying to use a more generic service such as Google. The integration of the default caller application involves sending call logs, contact information, and history to the Smart Dialer application, consolidating call information between both applications for easy use.

4. Logical View



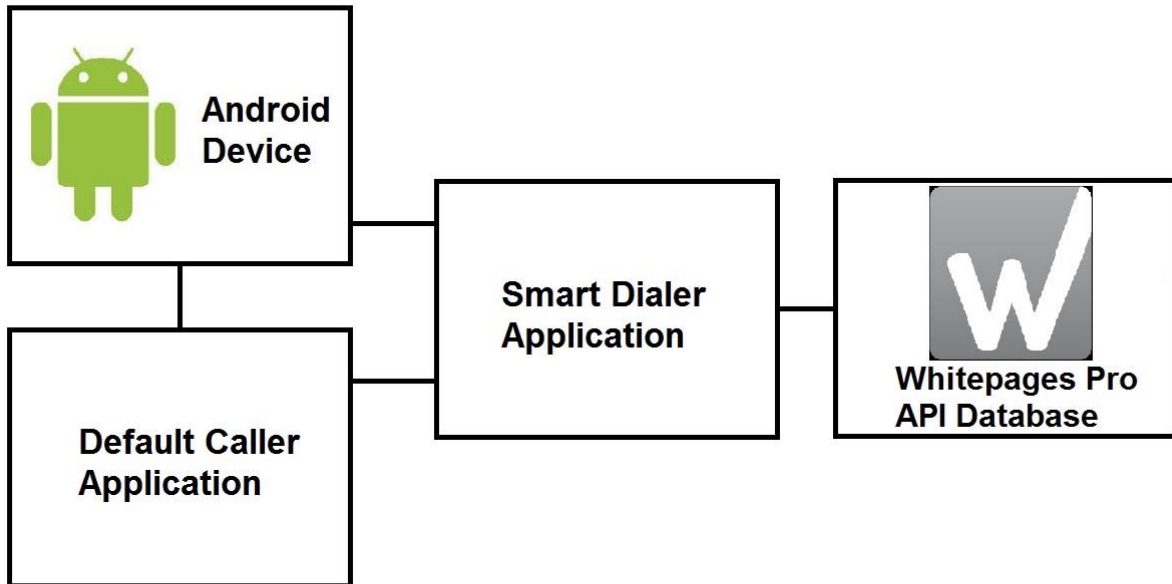
The Object-Oriented Design architecture style was chosen for this project.

5. Development View



This view represents the repository of our Smart Dialer application project. The app folder contains code to execute our program. The .gradle, .idea, and build are all Android Studio generated project folders.

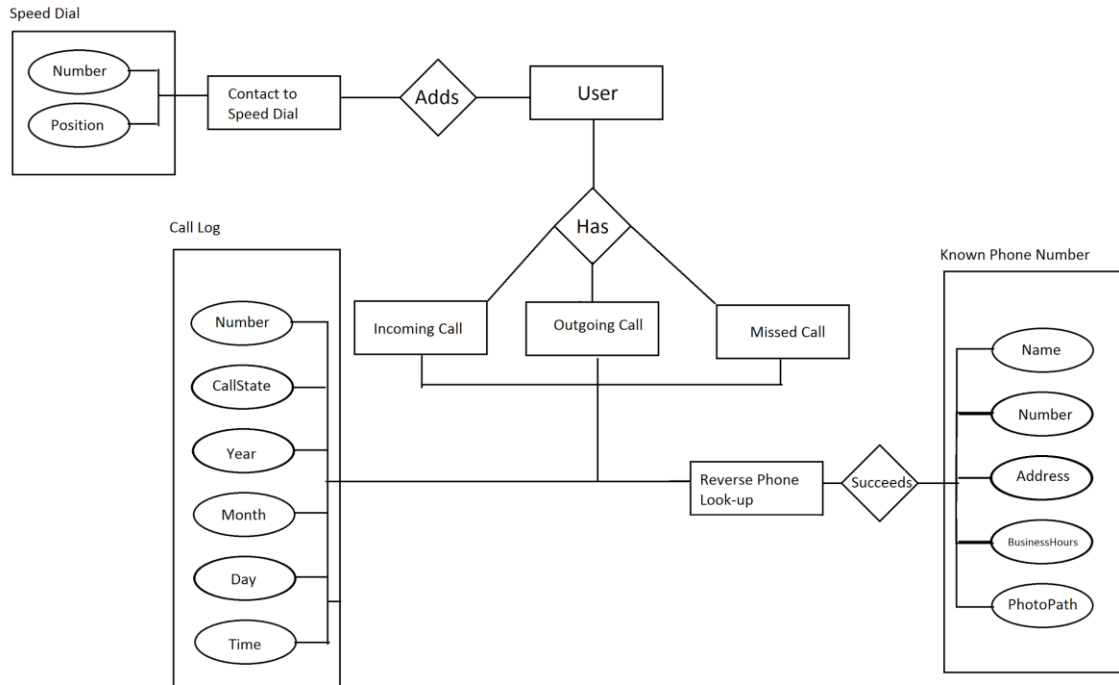
6. Physical View



The Smart Dialer application runs on any Android Device with the Lollipop operating system version 5.0 or newer. The application interfaces with the default caller application preinstalled on Android devices, pulling information for the user's call log and contacts for use in Smart Dialer. The Smart Dialer application also pulls information from the Whitepages Pro API Database to perform reverse phone number lookups.

7. Data View

Smart Dialer has 3 main databases, the Log database, which contains all phone calls and their states (outgoing, incoming, missed), the Speed Dial database, which contains a the phone number and speed dial position. The phone number is used to reference the appropriate contact, and the Known Phone Numbers database, which contains the appropriate name, number, address (if applicable), business hours (if applicable), and photo file-path (if applicable).



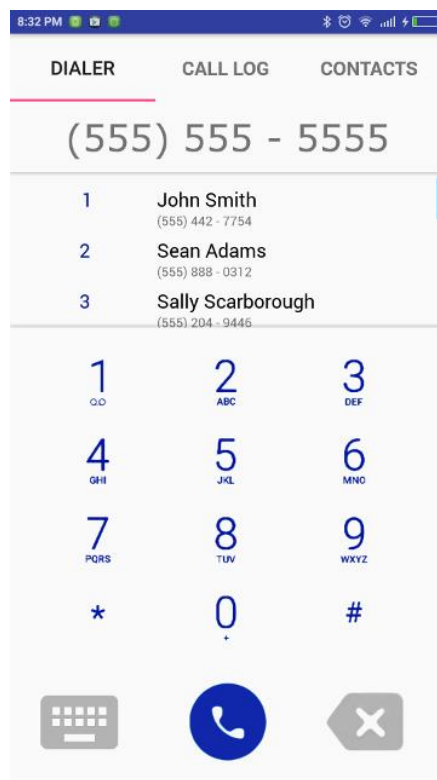
8. Work Assignment View

Smart Dialer's development team is comprised of members Cameron Moore, Omar Qasem, Patrick Richardson, Matthew Spedale, and Jeffrey Tolliver. Each member of the team is responsible for a screen in the Smart Dialer application, while the main backend development of the system is handled by Patrick Richardson, the team leader. Additionally, documentation for the project is handled by Matthew Spedale.

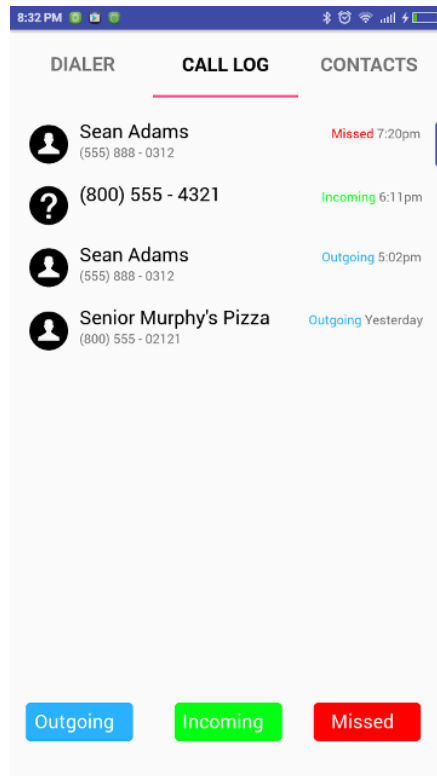
9. User Interface

The user interface of Smart Dialer is designed to mimic the interfaces of traditional mobile phone calling applications that are provided by Apple and Google on their iPhone and Android devices. The rationale behind this is to make our application as simple and intuitive to use as possible. Seeing as Smart Dialer is a mobile application, it is safe to assume most users will be familiar with the default phone dialer application on their device, and thus transitioning to Smart Dialer should feel familiar to them.

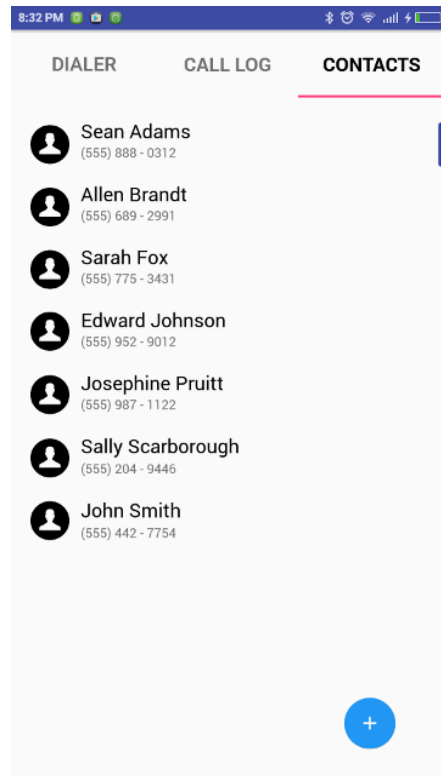
As specified in the Main System Architecture, the application contains three primary screens: the Dialer, the Log, and the Contacts. Each screen contains a tab at the top of the screen that can be pressed to move to the other screens, for a total of three tabs.



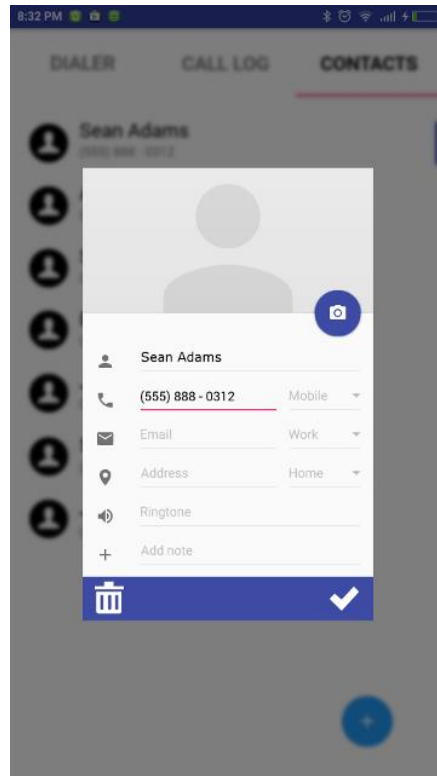
The Dialer screen contains a box in the middle of the screen containing the main display, which shows the text input of the phone number the user has dialed. Beneath the text input is a list containing the user's speed dial options. Tapping on an entry, or inputting the appropriate number and pressing dial will automatically call that number. The bottom of the screen contains the dialer itself, which is based on the T9 telephone keypad common to many cell phones.



The Log screen contains a historical list of calls, with details on each call entry. On the bottom of the screen are filter buttons that allow the user to filter the list. Tapping on an entry will populate the Dialer's input text with the associated phone number as the app switches to the Dialer screen, ready for the user to tap the call button to initiate the phone call. Holding down on an entry will bring up the Edit Contact overlay so that the user may edit (or save) the associated contact.



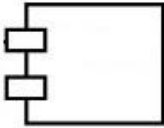

The Contacts screen contains the user's contacts, listed alphabetically by last name. Here the user may tap on a name to populate the Dialer's text input and holding down on the name will bring up the Edit Contacts overlay (same behavior as in the call log).



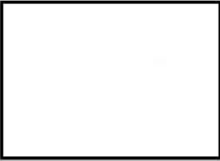



The Edit Contact overlay allows the user to add, remove, or edit any information of the user. If it is an existing contact or a known number that has been successfully searched through reverse-lookup, fields will be already populated with relevant information. This overlay contains three buttons. The middle-right allows the user to select a photo for the contact, the bottom left deletes the contact (after a confirmation window), and the bottom right saves contact with the updated information.

10. Element Catalog



Logical View Catalog:

	Object
	Object Association line

Physical View Catalog:

	Component
	Component Association line
	Android logo
	Whitepages logo

Development View Catalog:

	Folder
	Folder-Subfolder Association line