* 1. Introduction
  + 1.1 Purpose
  + 1.2 Document Conventions
  + 1.3 Intended Audience and Reading Suggestions
  + 1.4 Product Scope
* 2. Overall Production
  + 2.1 Product Perspective
  + 2.2 Product Functions
  + 2.3 User Classes and Characteristics
* 3. Other Nonfunctional Requirements
  + 3.1 Performance Requirements
* 4. Implementation
  + 4.1 Functional Requirements
  + 4.2 Preferred Requirements
  + 4.3 Nonfunctional Requirements

**1. Introduction**

**1.1 Purpose**

Our purpose for IPhoneBook is to provide users with a unique app that allows the user to categorize their contacts into specific groups. Whereas other phonebook/contact apps store contacts and allow users to add contacts and specify certain contacts, for example with a favorites, our app allows users to place contacts within specific groupings, such as friends, family, and colleagues, to allow for a more clean and concise way to store contacts.

**1.2 Document Conventions**

This proposal will us the Times New Roman font for all text. The bodies of our different sections will have a 12 point font. Subsection headings will be 14 point bold and ordered numerically corresponding to the topics listed in the headings. Section headings will be in 18 point bold and numerically ordered.

**1.3 Intended Audience and Reading Suggestions**

Our intended audience for IPhoneBook

**1.4 Product Scope**

After users enter the name and contact information, the user may organize the people in the contacts into different groups based on the user’s preference. Additionally the user may search for the contact information of a specific person based on their name, phone number, or email. Aside from storing contact information, the user may also sort their contacts based on the user’s criteria and remove certain contacts as well.

**2. Overall Production**

**2.1 Product Perspective**

IPhoneBook is a mobile application designed to provide for users a contact application that allows the user to store, categorize, sort, and remove contacts. IPhoneBook was conceived as a tool for users to categorize contacts easily, allowing for quick search for the appropriate contact within different groupings.

The system of IPhoneBook will print for the user a series of commands upon opening to prompt the user to input a command such as add, load, or remove. Afterwards the system will perform certain tasks corresponding to the users input, allowing the user to add contacts to specific groups.

The contact information will be stored in a file, allowing the user to retrieve, search, and sort contacts. A feature unique to IPhoneBook is the ringtones appended to specific contacts as another layer of categorizing.

**2.2 Product Function**

* Our users will be able to choose from a variety of commands
  + Save
  + Load
  + Add
  + Remove
  + List group
  + Remove phone
  + Show phone
  + Search name
  + Alphabetization based on contact name
  + Sort email
* The system will read the users input and execute the corresponding command
  + If the user’s input is invalid they will be prompted again to enter the correct format

**2.3 User Classes and Characteristics**

IPhoneBook is meant to allow the user to quickly save to and retrieve from a file, the contact information of the users friends, family, or colleagues.

**3. Other Nonfunctional Requirements**

**3.1 Performance Requirements**

IPhoneBook should be able to quickly retrieve and display contact information to the user and perform the appropriate search and sort based on the user’s criteria. The displayed contact information will include the contacts name, phone number, and email address.

**4 Implementation**

**4.1 Functional Requirements**

1. **Adding Contact Information** will allow the user to input contact information such as name, phone number, email, category for such contact, ringtone
2. **Saving Contact Information** to File will allow the user to be able to save all the necessary contact information to a txt file for later use
3. **Loading Contact Information** will allow the user to load saved information from the text file they have previously worked on
4. **Help** will allow the user to view the list of commands that will state all the possible options that the user is able to do
5. **Listing Contact Information** will allow the user to list contact information for the entire phonebook or possibly one category
6. **Removing Contact Information** will allow the user to edit their phonebook by removing certain contacts from the file
7. **Showing Contact Information** will allow the user to see a specific contact entry based on one of the subsets in the contact entry
8. **Searching Contact Information** will allow the user to be able to look up a certain contact based on either name or phone number or email, etc
9. **Sorting the Contacts** will be a feature that will be sorting the contacts as they are entered based on specific categories that the user has designated.
10. **Groups** will be allowing three categories for the user to store contact information in (friends, family, colleagues)
11. **Create new ArrayLists** for groups
12. **Allow for sorting** of arraylists for groups by name and email

**4.2 Preferred Requirements**

1. **Alphabetizing** will show the user the contact information in a specific category alphabetized by name
2. **Category Choice** will allow the user to decide the names of the categories they want to implement in their phonebook
3. **Ringtone** would allow the user to set one of the ringtone options given to certain users and/or categories

**4.3 Nonfunctional Requirements**

* Language: Java
* Design Pattern: MVC pattern, a main class that runs the program and a phonebook class that contains all the methods: adding, sorting, loading, etc.
* Scalability: data structures will be scalable in order for the user to add new contacts and store them in an arraylist
* Maintainability: a clear and clean structure that will allow the implementation of new code and the ability to quickly resolve issues that may arise during further development
* Usability: UI enables intuitive and easy interactions to add contacts and sort
* Write to a txt file the contact information that’s being added
* Read from a txt file the contact information that’s added