Software Requirements and Analysis

Team 2: Neighborhood Watch Development Team

Software Requirements Specification

Daniel Botdorf, Emily Bardwell, Jacob Pace, Kayode Gbogi-Emmanuel, Sam Rossilli

Version: (1.0.1) **Date: (03/10/2022)**

(Intentionally left blank)

Table of Contents

Revision History

Name	Date	Reason For Change	Version
Daniel Botdorf, Emily Bardwell, Jacob Pace, Kayode Gbogi-Emmanuel, Sam Rossilli	3/10/22	Initial Version	1.0.1

1 Introduction

1.1 Purpose

This product documents the design and procedure necessary to develop an application for the homeowners of XYZ neighborhood. It is divided into an Introduction, Specific Requirements, and Supporting documentation.

The document will be effective for use by the customer, product owners, and the technical implementation team.

1.2 Scope

1.2.1 Software Version Matrix

The software produced will assist in the function of a mobile application. Such software may include:

Software	Version
RHEL 8.5	4.18.0-348
SQLite	3.38.0
MySQL	8.0.28
Python	3.9

1.2.2 New Software Capabilities

This application will allow the user to see reported violations, document violations, and schedule neighborhood watch taskings. The app will not have a dialing function that works similar to 911.

1.3 Definitions, acronyms, and abbreviations

Variables Table

Name	Description
<ADMIN_USERID>	The UserID of the individual admin.
<ADMIN_PW>	Admin Password. Must be unique for each admin user.
<USER_USERID>	User Password. Must be unique for each user.
<USER_PASSWORD>	User Password. Must be unique for each user.

1.4 References

IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications* The Institute of Electrical and Electronics Engineers, Inc. 345 East 47th Street, New York, NY 10017-2394, USA, ISBN 0-7381-0332-2.

1.5 Overview

Describe what the rest of the SRS contains (see TOC?)

2 Overall descriptions

This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state-specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3 of the SRS, and makes them easier to understand.

This section usually consists of six subsections, as follows:
a) Product perspective;
b) Product functions;
c) User characteristics;
d) Constraints;
e) Assumptions and dependencies;
f) Apportioning of requirements.

2.1 Product perspective

The product is a communication platform among the neighborhood watch and other members of a community. Through the application, neighborhood members will be able to view the status of the members of their neighborhood watch. The application provides an easy tool for the neighborhood watch and other members of the neighborhood to report incidents. The reports will immediately alert the community. The application will advise/remind whoever is making a report to immediately call the police if it is a dangerous situation.

The product will document recent incident reports on a map that encompass the entire neighborhood. It will also feature trend reports about the occurrence of criminal activities and their locations. Personnel who will have access to the trend reports is to be determined after discussions with stakeholders about privacy issues and access to sensitive information.

2.1.1 System interfaces

This should list each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

2.1.2 User interfaces

Users will be able to access the platform through an application downloaded on their mobile devices. The user interface will be extended to and accessible from browsers such as Microsoft Edge, Google Chrome, DuckDuckGo, etc.

The graphical user interfaces will be implemented using software packages such as Java Swing, Python GUI, etc.

The user interface will utilize the following elements:

Input controls: checkboxes, buttons, text fields, date field, list boxes, dropdown lists

Navigational components: search fields, tags

Information components: icons, progress bars, notifications, message boxes

Navigating the platform:

Design (color and theme) is currently in progress

- Assessing the platform through the website or the application will transfer the users to a login page
- The login page will request the user to enter their chosen username and password.
- There is a button in the login page which the users can click to be delivered to a page for obtaining their forgotten passwords
- A sign-up button in the login page will direct the users to a page where users can create their accounts
- After signing in, the user is directed to the home page. The home page will display recent neighborhood news and information.
- The **Home Page** features a dropdown list that lists a series of options including reports, discussion boards, neighborhood watch calendar, crime statistics, crime map, etc.
- The **Report** feature opens a page where users can report an incident. The user must fill every part of the page to complete the report. The first is a severity rating from 1 to 5. The second is a dropdown list that contains possible incident types. The third is a location which can be described by the user or entered through a GPS. The fourth is a text field for a short description with a max length of 100 words. After the users document their incident, they can click the report button which will transfer them to a progress page confirming the action.
- The **Discussion** feature opens a page where users can connect and communicate with their fellow neighbors. The page lists up to 10 discussions. A user can either create a discussion or comment to a discussion. The page has a button to create a discussion. By clicking the button, users are directed to a page where they fill in the title of the discussion, and their comment. Users can select a discussion where they have an option to reply to other people's comments.
- The **Neighborhood Watch Calendar** feature directs the user to a new page with a calendar. The calendar will be in standard mm/dd/yy format. Each day on the calendar displays and will simply display the names of those of the watch members who are on duty. The designated leader of the neighborhood watch is provided with special privileges(password) to assign calendar days to the watch members by clicking the add button on the page. The add button directs the watch leader to a new window where the leader can delegate some member of the neighbor watch a day and time of duty.
- The **Crime Statistics** features trend graphs that should be automatically updated when reports are added to the system.(This needs discussion with customers for specifics on how crimes are displayed and organized).
- The **Crime Map** is a map display that will present pins where crimes have occurred given a certain radius. Users can select a pin to display a small summary of the incident. Users can sort incidents by distance, incident type, and date from a dropdown list on the window.

The HOA requested that we alert those on neighborhood watch when there is an incident in their area. We need to discuss what this will look like in further detail with the stakeholders.

2.1.3 Hardware interfaces

Communication protocols such as DNS (Domain name system), DHCS (Dynamic host configuration protocol), SMTP (Simple mail transfer protocol).

Supported devices include mobile devices such as smartphones, tablets, and laptops. The mobile device must have access to the internet, and it can either utilize the software as an application or has a browser that can access the software and maintain the graphical user interface.

Other hardware interfaces such as servers for data security, and database management purposes will need to be discussed further with stakeholders.

2.1.4 Software interfaces

The application shall utilize the following software interfaces:

Database Service: SQLite version 3.38.0

The purpose of using this interface is to allow application data to be created, updated, read, and destroyed for data such as users, incidents, and reports. It will act as an interface between application and database services. SQLite is used for compatibility on devices running iOS.

The interface can be defined in the SQLite documentation.

2.1.5 Communications interfaces

The app shall utilize the following communication interfaces, provided by the operating system on the device the app is running on:

IEEE 802.11 Wi-Fi Protocol

3G Protocol

The purpose of using this communication interface is to allow for the application to be available on a mobile device as well as a desktop device.

Java Database Connectivity - JDBC 4.3

The purpose of using this interface is to allow the application to interface and communicate with the database engine. The Java Database Connectivity is used to allow for easy use in the Android operating system.

The interface can be defined in the JDBC documentation.

2.1.6 Memory constraints

This should specify any applicable characteristics and limits on primary and secondary memory.

2.1.7 Operations

The application, under normal operation shall allow for the user to

access the application through an iPhone app

access the application through Android app

access the application through a webpage

create incidents to add to the map

choose data display methods (list vs map)

sort incidents by date

sort incidents by distance

The system automatically shall, under normal operation

- verify user identity with a login
- check user permissions to remove incidents
- remove data from the incident database 30 days after creation

2.1.8 Site adaptation requirements

The application shall initialize the user interface with:

- A view of an incident map
- A button to ‘login to add incidents’
- Incidents in list view sorted by date

The application shall initialize the user accounts upon creation as

- A default ‘neighbor’ level account without permissions to remove incidents

2.2 Product functions

The application shall have the following functionality.

The ability to log in – Login page –

- Log in shall be through the application
- Log in shall be through email and password
- Security verification – undetermined -

The Home page

- Undetermined
- -potential- announcements from the HOA
- -potential- quick glance crime statistics what has occurred in the last week

The Crime Statistics page

- Ability to view crime reports within a certain radius
- The ability to sort crimes based on -undetermined-
- The ability to look at crime trends

Crime map page

- The ability to view all crime reports on a map
- The ability to view a quick summary of the crime when clicking on a pin

The Report page

- The ability to create a report of a crime
- The ability to add a location and description of the crime
- The ability to alert the neighborhood watch association when a crime is reported
- This report should be added to the crime maps page and crime statistics page

The Neighborhood watch calendar page

- The ability to view a monthly calendar of who is on duty on any give night
- The ability to add different people to the calendar
- The ability to take people off the calendar
- –potentially- The ability to sign up for the watch

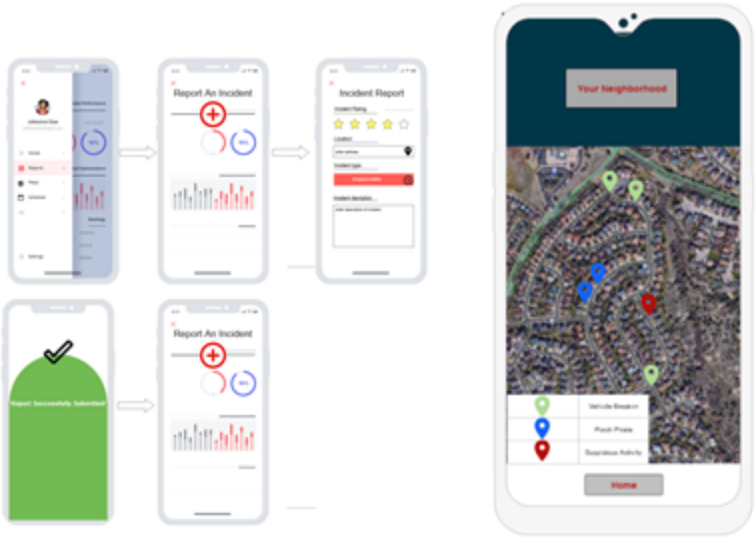
The discussion boards

- Allow neighbors to start discussions on any issues they may have
- Allow neighbors to comment on discussions
- -potentially- allow HOA to moderate discussions

Crime Report for police

- The ability to print a weekly crime report for the police so the HOA may discuss issues

We have some diagrams, but we want to work on them more and get the HOA involved on the design so it looks nice. Here are a few we've already shown, but need a lot of work.



2.3 User characteristics

Users of the product are of average education, meaning some are college graduates, others have some college and still others in the neighborhood have little to no college. They are not extremely experienced in the technical field and lack any IT department. This is something we will have to carefully consider during the design process and will most likely affect the database interface. They are however able to interface with smart devices such as iPhones with ease and expertise.

2.4 Constraints

In the production of the neighborhood watch scheduler, the main constraint for our team is the transmission of the reports provided by the software to the local police. As this is a major reason for the creation of this software it is crucial that some kind of communication is available, however, we need to ensure that our system is not going to impede police systems or flood police databases. In addition, this system is going to be used for personal safety. We need to ensure that the software is reliable and secure so the user feels safe to use it.

2.5 Assumptions and dependencies

In the production of the neighborhood watch scheduler the main constraints for our team are is the communication of our software to the local police. As this is a major reason for the creation of this software it is crucial that some kind of communication is available however, we need to ensure that our system is not going to impede police systems or flood police databases. In addition to this because this system is going to be used for personal safety we need to ensure that the software is reliable and secure so the user feels safe to use it.

2.6 Apportioning of requirements

Based on the dependencies above the features most likely to be postponed until later in development are police notification/documentation systems.

3 Specific requirements

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:

a) Specific requirements should be stated in conformance with all the characteristics described in 4.3.
b) Specific requirements should be cross-referenced to earlier documents that relate.
c) All requirements should be uniquely identifiable.
d) Careful attention should be given to organizing the requirements to maximize readability.

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in 3.1 through 3.7.

3.1 External interfaces

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in 2 and should not repeat information there.

It should include both content and format as follows:

a) Name of item;
b) Description of purpose;
c) Source of input or destination of output;
d) Valid range, accuracy, and/or tolerance;
e) Units of measure;
f) Timing;
g) Relationships to other inputs/outputs;
h) Screen formats/organization;
i) Window formats/organization;
j) Data formats;
k) Command formats;
l) End messages.

3.2 Functions

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with “The system shall…”

These include

a) Validity checks on the inputs
b) Exact sequence of operations
c) Responses to abnormal situations, including
    1) Overflow
    2) Communication facilities
    3) Error handling and recovery

d) Effect of parameters
e) Relationship of outputs to inputs, including
    1) Input/output sequences
    2) Formulas for input to output conversion

It may be appropriate to partition the functional requirements into subfunctions or subprocesses. This does not imply that the software design will also be partitioned that way.

3.3 Performance requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:
a) The number of terminals to be supported;
b) The number of simultaneous users to be supported;
c) Amount and type of information to be handled.

Static numerical requirements are sometimes identified under a separate section entitled Capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions. All of these requirements should be stated in measurable terms.

For example,

95% of the transactions shall be processed in less than 1 s.

rather than,

An operator shall not have to wait for the transaction to complete.

NOTE – Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.

3.4 Logical database requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:
a) Types of information used by various functions;
b) Frequency of use;
c) Accessing capabilities;
d) Data entities and their relationships;
e) Integrity constraints;
f) Data retention requirements.

3.5 Design constraints

This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.5.1 Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

a) Report format;
b) Data naming;
c) Accounting procedures;
d) Audit tracing.

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

3.6 Software system attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. Subclauses 3.6.1 through 3.6.5 provide a partial list of examples.

3.6.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

3.6.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

3.6.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to

a) Utilize certain cryptographical techniques;
b) Keep specific log or history data sets;
c) Assign certain functions to different modules;
d) Restrict communications between some areas of the program;
e) Check data integrity for critical variables.

3.6.4 Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

3.6.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

a) Percentage of components with host-dependent code;
b) Percentage of code that is host dependent;
c) Use of a proven portable language;
d) Use of a particular compiler or language subset;
e) Use of a particular operating system.

3.7 Organizing the specific requirements

For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in Section 3 of the SRS. Some of these organizations are described in 3.7.1 through 3.7.7.

3.7.1 System mode

Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency. When organizing this section by mode, the outline in A.1 or A.2 should be used. The choice depends on whether interfaces and performance are dependent on mode.

3.7.2 User class

Some systems provide different sets of functions to different classes of users. For example, an elevator control system presents different capabilities to passengers, maintenance workers, and fighters. When organizing this section by user class, the outline in A.3 should be used.

3.7.3 Objects

Objects are real-world entities that have a counterpart within the system. For example, in a patient monitoring system, objects include patients, sensors, nurses, rooms, physicians, medicines, etc. Associated with each object is a set of attributes (of that object) and functions (performed by that object). These functions are also called services, methods, or processes. When organizing this section by object, the outline in A.4 should be used. Note that sets of objects may share attributes and services. These are grouped together as classes.

3.7.4 Feature

A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs. When organizing this section by feature, the outline in A.5 should be used.

3.7.5 Stimulus

Some systems can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections for loss of power, wind shear, sudden change in roll, vertical velocity excessive, etc. When organizing this section by stimulus, the outline in A.6 should be used.

3.7.6 Response

Some systems can be best organized by describing all the functions in support of the generation of a response. For example, the functions of a personnel system may be organized into sections corresponding to all functions associated with generating paychecks, all functions associated with generating a current list of employees, etc. The outline in A.6 (with all occurrences of stimulus replaced with response) should be used.

3.7.7 Functional hierarchy

When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data Flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data. When organizing this section by functional hierarchy, the outline in A.7 should be used.

3.8 Additional comments

Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification. For example, see A.8 for an organization combining user class and feature. Any additional requirements may be put in a separate section at the end of the SRS.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, Finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; and when organizing by functional hierarchy, data Flow diagrams and data dictionaries may prove helpful.

In any of the outlines given in A.1 through A.8, those sections called “Functional Requirement *”* may be described in native language (e.g., English), in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, and Outputs.

4 Supporting information

The supporting information makes the SRS easier to use. It includes the following: Table of contents;
a) Index;
b) Appendixes.

4.1 Table of contents and index

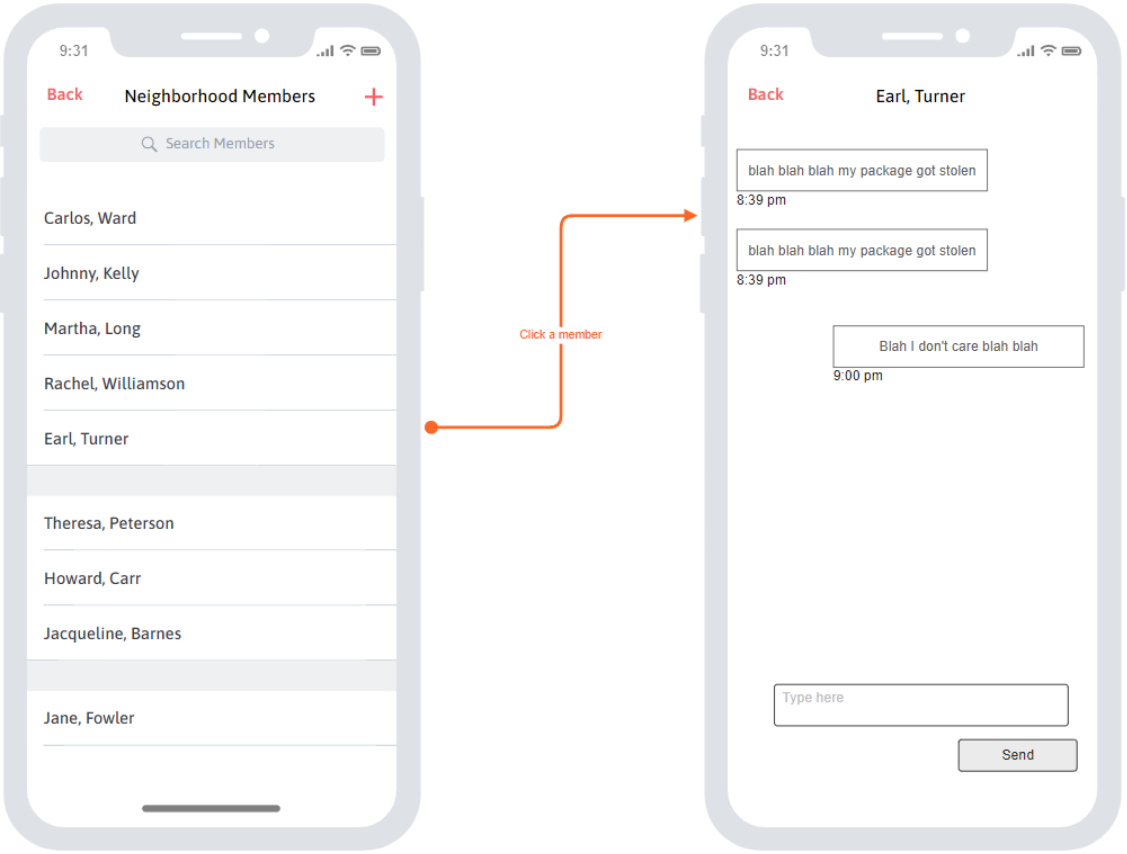
The table of contents and index are quite important and should follow general compositional practices.

4.2 Appendices

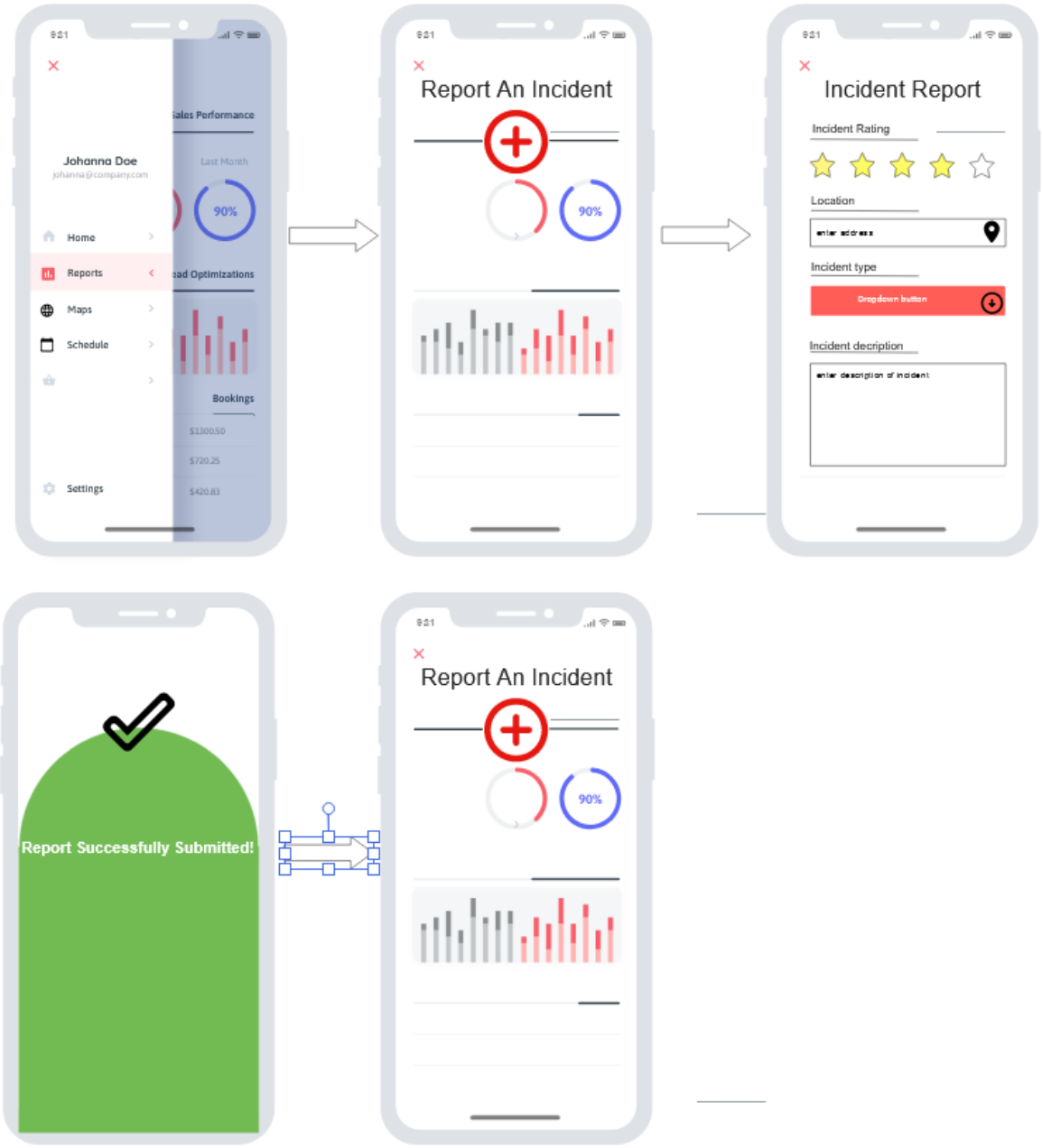
4.2.1 Mockups/Story boards

Map-view

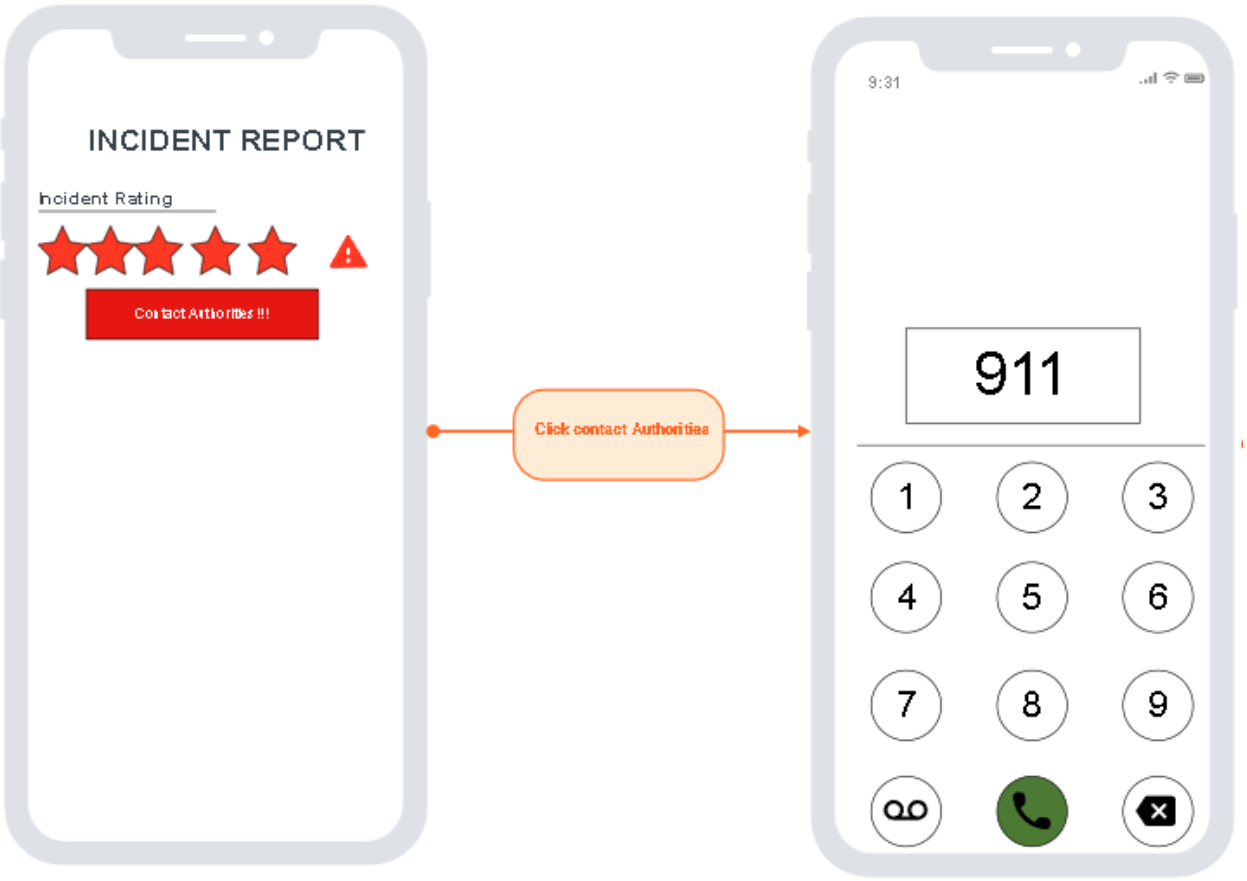
Connect with members of the neighborhood



Add an incident in the database



Law enforcement notification



Patrol Schedule