SANE App

Drew Shoemaker

UMKC SCE

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Author** | **Date** | **Revision** |
| D. Shoemaker | 9/14/2015 | Initial Release |
|  |  |  |

Table of Contents

[Vision Statement 4](#_Toc430021439)

[Requirements 5](#_Toc430021440)

[Categories of Users 5](#_Toc430021441)

[Actor-Goal List 5](#_Toc430021442)

[User Stories 5](#_Toc430021443)

[Sprint #1 11](#_Toc430021444)

[Iteration #1 Review Meeting 11](#_Toc430021445)

[Iteration #1 Retrospective 11](#_Toc430021446)

[Design 13](#_Toc430021447)

[Coding Standards 14](#_Toc430021448)

# **Vision Statement**

Sexual Assault remains one of the most underreported crimes in the United States, with only 32% of Sexual Assaults actually being reported. It happens in many forms of relationships and the type of assault varies (e.g. obvious, ambiguous, verbal, physical, etc.). The Justice Department has identified Sexual Assault as one of the few forms of violent crime that has not benefited from the advent of technology, specifically smartphones. One reason for the lack of reporting when it comes to this form of crime is fear, embarrassment and shame; often sexual assaults are perpetrated by people that know and are in a intimate relationship with the victim. These factors often delay reporting until the statute of limitations as already passed, if the victim ever even reports at all. The psychology of domestic violence also provides obstacles.

The SANE application aims to overcome or mitigate these obstacles by providing an outlet for women to identify sexual assault when it occurs. The app will also serve as an anonymous link for women to a Sexual Assault Nurse Examiner (SANE) and Law Enforcement Officers (LEO). Reports created by users will be anonymously sent to SANEs and LEOs to be reviewed. If a SANE determines that the case is serious and worthy of reporting, the application will inform the user that the case has been determined to be serious and prompt them to open a case. From an educational perspective, the application will vary in color from green (good), yellow (suspicious) and red (bad) as the user fills out a report. The option to open a case will be provided to users whenever the report turns red.

# **Requirements**

## Categories of Users

**Users** – The only category of user for the application. These users will create reports, view previous reports and open cases when there is a serious incident reported.

## Actor-Goal List

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| User | Log In |
|  | Validate User Credentials (probably just a pin number for privacy) |
|  | Create A Report |
|  | Answer questions as they are prompted from the app |
|  | Question tree will be traversed based on user answers |
|  | Report will change color based on severity of incident |
|  | User can save any type of report |
|  | User can open case for Red reports |
|  | Review Previous Reports |
|  | Open cases on old reports that are Red |
|  |  |

## User Stories

S1

|  |
| --- |
| **As a user, I want to a form of authentication that protects my information from unwanted users of the application** |

S2

|  |
| --- |
| **As a user, I want to be able to create a new report to record a new incident** |

S3

|  |
| --- |
| **As a user, I want to save a report no matter the severity** |

S4

|  |
| --- |
| **As a user, I want to open a case for serious reports of assault that I create** |

S5

|  |
| --- |
| **As a user, I want to review all previous reports that I have created** |

S6

|  |
| --- |
| **As a user, I want to open a case on a report that I created in the past** |

S7

|  |
| --- |
| **As a user, I want to be prompted by the application to open a case when I have filed a report that is more serious than I believed it to be** |

Product Backlog

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Story** | **Story Points** | **Priority** | **Status** |
| S1 | Allow user to login with a PIN | 4 | 7 |  |
| S2 | Allow user to create a new report | 12 | 1 |  |
| S3 | Allow user to save a created report | 7 | 2 | Completed  8/3/2013 |
| S4 | Allow user to open a case for Red reports | 10 | 3 |  |
| S5 | Allow user to view all previously created reports | 8 | 4 |  |
| S6 | Allow user to open a case while viewing an old report | 5 | 5 |  |
| S7 | Prompt user to open a case on serious reports | 12 | 6 |  |

|  |  |
| --- | --- |
| **Title:** | Find and subscribe to quizzes |
| **Use case ID:** | UC001 |
| **Actor:** | Quiz Taker |
| **Description:** | This use case describes the system interactions needed to find and subscribe to a new quiz. |
| **Basic Flow:**   1. This use case begins when a quiz taker elects to search for a new quiz. 2. The system responds by showing a complete list of available quizzes. Quizzes that the quiz taker is currently subscribed to are shown but not selectable. Quizzes the user has never taken, or has taken in the past but is not currently subscribed to, are selectable. The system shows the title of available quizzes. 3. The quiz taker selects one of the selectable quizzes. 4. The system shows the selected quiz as selected. 5. The quiz taker elects to exit the search function. The quiz taker is now subscribed to the selected quizzes. | |
| **Alternate Flows:**  2a. There are no quizzes available.  1. The system displays a message that no quizzes are available.  3a. The quiz taker selects more than one quiz.  1. All selected quizzes are shown as selected.  3b. The quiz taker wants to unselect a quiz.  1. The quiz taker repeats the select operation on the quiz he or she wants to unselect.  2. The quiz is shown as unselected. | |
| **Exceptions:**  2. There is a format error in quiz data.   1. The quiz taker is presented with an error message identifying the quiz or quizzes that are incorrectly formatted. The error message identifies the incorrectly formatted quiz without giving away any of the data on specific questions. Service continues with data from correctly formatted quizzes. | |
| **Open issues:**   1. If there are no quizzes available, should the quiz taker be given instructions on how to gain access to quizzes? This could be useful in situations where the quiz data files have been misplaced, moved or deleted. | |

When documenting detailed requirements at the beginning of a project, consider using system shalls to specify requirement detail that isn’t conveniently captured by stories and use cases.

There are an incredible amount of detailed decisions that have to be made even for simple applications. For example:

1. If a user enters a valid ID but invalid password should the system be helpful and respond that only the password was incorrect or report back a generic “logon attempt failed”?
2. How many logon attempt failures for a valid ID are allowed before the system blocks attempts? During what timeframe?
3. Should blocking be temporary? If so, how long is “temporary”?

That’s 5 questions that need to be answered and we haven’t even got past the login. Documenting these decisions in use case specifications will likely make them unreadable. These detailed requirements that aren’t part of the abstract or logical behavior of the system can be specified in a long list of “system shalls”.

Example:

1. While taking a quiz, a quiz taker shall be able to save partial results.
2. The system shall provide the quiz author the ability to create multiple choice, true/false and multiple selection questions.
3. The system shall automatically grade multiple choice, true/false and multiple selection questions.
4. The system shall provide the quiz author the ability to create essay questions.
5. The system shall automatically grade essay questions.
6. The system shall give authors the ability to create pass/fail quizzes.
7. The system shall give the author the ability to limit the feedback given to quiz takers. (For example, the quiz author might want to prevent the quiz taker from seeing his or her score.)
8. The system shall keep quiz answers confidential. Users should not be allowed unauthorized access to answers.

It may also be helpful to document decisions or facts from the problem domain that might be useful in specifying requirements. Doing so helps avoid repeatedly asking the customer and can avoid reopening discussions/debates. For example, here are some general business rules for a loyalty rewards program:

1. Customer earns points on every purchase at participating merchants at a rate of 1 point per $1 spent.
2. Points are redeemed in units of 100. Every 100 points is worth $5 off select items
3. Merchants can decide what items can be purchased at a discount using points
4. Customers are not allowed to earn points when redeeming points (on cash balance of transition)

If there are important non-function requirements, consider documenting them separately.

Example:

The system shall be easy to use. A representative user fitting the profile of Volunteer described above without any prior experience with the application should be able to find and subscribe to a specific quiz in less than 2 minutes.

# **Sprint #1**

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S2 | Create Report Activity | 2 |  |
|  | Add Question Fragment | 3 |  |
|  | Create Layouts for Question Views | 2 |  |
|  | Add logic to determine which question branch to go to | 7 |  |
|  | Determine severity points (color change) based on questions | 5 |  |
| S1 | Add intro screen for app with dummy login | 2 |  |

## Iteration #1 Review Meeting

<Minutes/notes from feature demo (product review).>.

## Iteration #1 Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

You probably also want to calculate velocity for the just-completed iteration Velocity is the expected number of story points that can be completed during an iteration. For example, assume after the first iteration story 3 (16 story points) and just the first task of story 1 (4 story points) was completed. In this case your new velocity is 16 (total number of story points of completed stories). Notice you don’t get credit for partially completed stories. If you think this will misstate your progress, break large stories into smaller ones.

Project velocity: 16

A nice visual way of tracking velocity is to maintain a graph like the following:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 25 |  |  |  |  |  |  |
| 20 | 16 | 18 |  |  |  |  |
| 15 |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |
|  | **Sprint 1** | **Sprint 2** | **Sprint 3** | **Sprint 4** | **Sprint 5** | **Sprint 6** |

# **Design**

[What is the overall structure of the solution? What are the major modules of code? What are the dynamics of communication between these modules? The most common way of depicting this information is with static and dynamic models augmented with short narrative descriptions of design.]

# **Coding Standards**

Coding standards improve readability. They make it easier to understand code written by others. Good coding standards also improve reliability.

TBD