Software Engineering
Software Requirements Specification (SRS) Document

ScamGuard

26 September 2023

Version 1

By: Derek Fox, Alex Jasper, Kenny Banks

[Honor Code]

Table of Contents

1. Introduction

1.1. Purpose

The goal of the ScamGuard application is to help individuals identify and avoid common scams that are affecting people in their community. The application will allow users to post about scams that they are aware of/have experienced, as well as allow businesses to raise awareness for scams that may affect their customers.

1.2. Document Conventions

The purpose of this Software Requirement Document (SRD) is to describe the client-view and developer-view requirements for the ScamGuard application. Client-oriented requirements describe the system from the client's perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer's perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

1.3. Definitions, Acronyms, and Abbreviations

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
MySQL	Open-source relational database management system.
HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
NetBeans	An integrated development environment (IDE) for Java. This is where our system will be created.
API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.

1.4. Intended Audience

1.5. Project Scope

The goal of the software is to provide an easy-to-use interface for the targets of scams, businesses related to those scams, and administrators. This aligns with the fact that many of the targets of scams are elderly or otherwise not technologically adept individuals.

The benefits of the project to users include:

- Spread information about common scams in one's area or demographic group.
- Allow businesses to promote their trustworthiness by identifying potential scams that may be affecting their customer base.
- Decrease the total impact of scams on the community and society as a whole.

1.6. Technology Challenges

1.7. References

2. General Description

2.1. Product Perspective

ScamGuard found its origin in the desire to combat the rising prevalance of scams, especially among members of society who are not as techincally literate. Maintaining an easy-to-access compilation of common scams which can update in real time to respond to potential outbreaks of certain scam tactics will help this cause.

2.2. Product Features

The product features the ability for users to view common scams in their area, submit new scams for consideration, and comment on scams that others have experienced. Businesses can also release scam warnings to help their customers avoid targeted scams. Administrators will have the ability to remove incorrect information and review new additions to the database.

2.3. User Class and Characteristics

Our product does not expect our users to have any prior knowledge of a computer apart from using a web browser. Specifically, this software is actually intended for users that may not be adept with technology, as they may be at higher risk of certain scam techniques.

2.4. Operating Environment

The application is designed to operate on the web across many different devices.

2.5. Constraints

2.6. Assumptions and Dependencies

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans.

3. Functional Requirements

3.1. Primary

- FR0: The system will allow the user to lookup scams based on location, targets, or delivery method. The information will include the number and demographics of those who are being targeted.

- FR1: The system will allow the user to enter a new scam into the database, to be reviewed by a moderator. These entries can be created by those targeted by a scam, or by a business to warn their customers of a potential scam.
- FR2: The system will allow users to “like” or “dislike” scams, to mark whether they believe them to be legitimate or not.
- FR3: The system will allow moderators to remove scams from the database that have been submitted in error or fradulently.
- FR4: The system will allow comments to be posted underneath scams

3.2. Secondary

- Password protection for information only accessible to employees, managers, and each individual table.
- Authorization scheme so that customers can only alter and see their orders and no other customers’ orders.
- All customers will be required to verify their identity through an authetication/login system in order to guarantee scammers themselves aren't determining the legitmacy of scams.

4. Technical Requirements

1.1. Operating System and Compatibility

The application will be compatible with any modern operating system that is able to view and to interact with traditional web pages.

1.2. Interface Requirements

1.2.1. User Interfaces

The ScamGuard user interface is designed with a clean, intuitive layout that prioritizes user experience and navigation ease. Upon accessing the main page, users are greeted with a dashboard that presents trending scam reports and a search bar for quick lookups. For each report, there are interactive buttons allowing users to upvote, comment, or flag content, ensuring community engagement. Messages displayed on screens are crafted with clarity to guide the user effortlessly. To maintain consistency and branding, a cohesive style guide is followed across all pages. This encompasses a harmonious color palette, a legible font family, and responsive design elements to ensure compatibility across various devices.

1.2.2. Hardware Interfaces

ScamGuard, at its core, is designed to be universally accessible. It leverages responsive design practices to ensure seamless usability across various hardware devices. The following outlines the hardware interfaces and related criteria.

1.2.3. Communications Interfaces

ScamGuard will primarily use HTTP and its secure variant HTTPS for all web communications, ensuring dara privacy and integrity. Additionally for third-party integrations or future mobile app versions, ScamGuard will employ RESTful APIs to communicate data. WebSockets could be employed to allow for real-time feature enhancements.

1.2.4. Software Interfaces

We will use Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

2. Non-Functional Requirements

2.1. Performance Requirements

- **NFR0(R):** The local copy of the scam report database will consume less than 50 MB of memory.
- **NFR1(R):** The ScamGuard system (including the local copy of the scam report database, user profiles, and business responses) will consume less than 150MB of memory.
- **NFR2(R):** A novice user will be able to report a scam and receive a confirmation in less than 3 minutes.
- **NFR3(R):** An expert user will be able to report a scam and receive a confirmation in less than 45 seconds.
- **NFR4(R):** The search functionality, which allows users to look up scams, will return results in less than 2 seconds for common queries.
- **NFR5(R):** Businesses responding to a report on their page will receive an acknowledgment of their response in under 10 seconds.

2.2. Safety Requirements

- Info dialog prompting users to stay alert while reviewing scams as to not fall victim to one.
- Provide list of common methods to identify scammers to subsequently boost user judgement when examining potential scams.
- Warn users that although a service may be designated as **not** being a scam, there is no guarantee it is a trusted resource.

2.3. Security Requirements

- The system will only be usable by authorized users.
- Bot protection to ensure scammers are not identifying their own scams as “safe” on a mass-scale.
- Verification system that allows scammers to be separated from true customers.

2.4. Software Quality Attributes

2.4.1. Availability

High uptime (99.9%) with contingencies in place to handle server failures and/or other unexpected issues. Regular backups and distributed server infrastructure (load balancing) can be used to enhancing availabilty.

- System uptime of 99.9%
- Rapid failover in the case of server failures
- Regularly scheduled backups of data

2.4.2. Correctness

ScamGuard must provide accurate and updated information about scams. Data integrity is paramount; any inaccuracies could harm users or businesses. The system should have validation checks to ensure only accurate and verified information is disseminated.

- Implement validation measures for user submissions to prevent the spread of misinformation.
- Allow users and businesses to flag incorrect information for review by moderators.
- Use a trust scoring system to rate the reliability of sources.

2.4.3. Maintainability

As the scam landscape evolves, ScamGuard must adapt. The system should be structured in a modular manner, allowing for easy updates and modifications. Furthermore, clear documentation should be maintained to aid future developers.

- Modular codebase that separates functionalities.

- Comprehensive documentation covering system architecture, module details, and update procedures.
- Regular code reviews and refactoring sessions to ensure code health.

2.4.4. Reusability

Components of ScamGuard, like scam detection algorithms or user verification processes, might be reusable in other applications or modules. Designing with reusability in mind ensures a more efficient development process for future expansions or new products.

- Encapsulate functionalities into separate modules or microservices.
- Use consistent coding standards and conventions to ensure components are easily understandable and reusable.
- Store reusable components in a shared library or repository.

2.4.5. Portability

To reach a wider audience, ScamGuard should be usable across different platforms and devices. Whether a user accesses it through a web browser on a desktop, a mobile app, or even via third-party platforms, they should have a consistent experience.

- Develop with a responsive design to cater to both desktop and mobile users.
- Ensure compatibility across major web browsers: Chrome, Firefox, Safari, Edge, etc.
- Consider building APIs to allow third-party platforms or apps to retrieve and display ScamGuard's data.

2.5. Process Requirements

2.5.1. Development Process Used

Agile Scrum Model

2.5.2. Time Constraints

The project timeline is constrained by the academic calendar, with a deadline for the final project deliverable set for the end of the semester (approximately 15 weeks from the start date).

- Initial Planning and Requirements Gathering: 2 weeks
- Design and Prototyping: 3 weeks
- Development and Iterative Testing: ~ 6 - 8 weeks
- Final Testing and Deployment: 2-4 weeks

2.5.3. Cost and Delivery Date

As this is an academic project, the primary cost considerations are not financial but rather resource-based, focusing on the time investment from team members. However, we might incur minor costs for hosting services or third-party APIs

2.6. Other Requirements

Data Migration and Backup:

- ScamGuard should offer a method for data migration for future scalability and also have a robust backup system to protect against data loss.

Accessibility:

- The application should be accessible, following WCAG (Web Content Accessibility Guidelines) to ensure that individuals with disabilities can use the system.

Legal and Compliance:

- ScamGuard needs to adhere to data protection laws, such as GDPR in Europe or CCPA in California, to ensure user data is securely stored and managed.

Security Audits:

- Regular security audits should be conducted to ensure that the application is free from vulnerabilities.

Third-Party Integrations:

- ScamGuard should support third-party API integrations for features like social media sharing, data analytics, or payment processing (if applicable).

Documentation:

- Comprehensive user guides, admin guides, and API documentation should be available.

Multilingual Support:

- While the initial release may be in English, ScamGuard should be designed to easily support multiple languages in the future.

Disaster Recovery:

- A disaster recovery plan should be in place to restore service as quickly as possible in case of system failures.

User Training:

- While the application aims to be user-friendly, introductory tutorials or tooltips should be available to guide users in performing common tasks.

Software Updates:

- ScamGuard should offer an easy way to update the software for end-users, including patch management and update notifications.

Monitoring and Analytics:

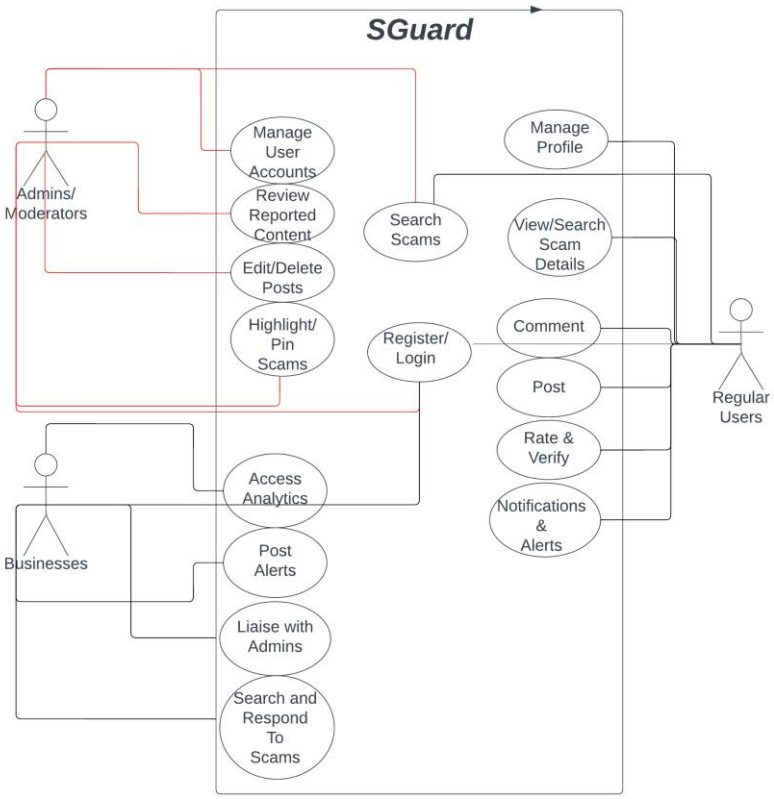
- Real-time monitoring should be in place to track system performance, user engagement, and other relevant metrics.

Feedback Mechanism:

- Users should have an easy way to provide feedback or report issues directly within the application.

TBD

2.7. Use-Case Model Diagram



2.8. Use-Case Model Descriptions

2.8.1. Actor: Business Users (Derek Fox)

- **Use-Case Name: Respond To Scam Allegations**
  - Businesses can provide clarifications or dispute scam reports related to their services or products.
- **Use-Case Name: Monitor Brand Reputation**
  - Businesses can view and analyze reports mentioning them to understand public sentiment and potential areas of concern.
- **Use-Case Name: Post Scam Alerts**
  - Businesses can post alerts for potential scams that they believe may target customers of their services.

2.8.2. Actor: Regular Users (Kenny Banks)

- **Use-Case Name: Report a Scam**
  - Users can post details of scames they've encountered, including evidence, communication logs, etc.
- **Use-Case Name: Search and View Scam Reports**
  - Users can search the database for scams based on various criteria (e.g., location, business name) and view details of reports.

2.8.3. Actor: Admins/Moderators (Alex Jasper)

- **Use-Case Name: Review Scam Reports**
    - Admins/Moderators review scam reports flagged by users or the system's automated checks, verifying their authenticity.
- **Use-Case Name: Manage Users**
    - Admins/Moderators can approve new user registrations, ban malicious users, or grant special permissions.

2.9. Use-Case Model Scenarios

2.9.1. Actor: Admins/Moderators (Alex Jasper)

- **Use-Case Name**: Review Scam Report
    - **Initial Assumption**: A scam report has been submitted by a user and is pending approval.
    - **Normal**: Admin/Moderator reviews the report has been submitted by a user and is pending approval
    - **What Can Go Wrong:** The scam report lacks sufficient evidence, contains inappropriate content, or is false.
    - **Other Activities**: Sending feedback to the user about the status of their report.
    - **System State on Completion**: Scam report is either approved and visible to the public or rejected and archived.
- **Use-Case Name**: User Account Management
    - **Initial Assumption**: A user account has been flagged for suspicious activity.
    - **Normal**: Admin/Moderator reviews the account activity, issues a warning or suspends the account.
    - **What Can Go Wrong**: Legitimate user accounts might be mistakenly flagged.
    - **Other Activities**: Notifying the user about the status of their account and any actions taken.
    - **System State on Completion**: Suspicious account is either flagged with a warning or suspended.

2.9.2. Actor: Regular Users (Kenny Banks)

- **Use-Case Name**: Report a Scam
    - **Initial Assumption**: User has encountered a potential scam and wants to report it.
    - **Normal**: User fills out the scam report form with details and submits it for review.
    - **What Can Go Wrong**: User provides incomplete or inaccurate information.
    - **Other Activities**: User can attach evidence (screenshots, links) to support their claim
    - **System State on Completion**: Scam report is submitted and awaits moderation
- **Use-Case Name**: Search for Scams
    - **Initial Assumption**: User wants to check if a particular incident is a known scam.
    - **Normal**: User enters the relevant keywords, and browses through the results.
    - **What Can Go Wrong**: No relevant results are found.
    - **Other Activities**: User can filter or sort search results.
    - **System State on Completion**: User has the list of relevant scam reports or a message saying no scams were found.

2.9.3. Actor: Businesses (Derek Fox)

- **Use-Case Name**: Respond to a Report
    - **Initial Assumption**: A scam report related to a business has been submitted.

        - **Normal**: Business logs in, reviews the report, and provides their side of the story or clarification.
        - **What Can Go Wrong**: Business response is aggressive, promotional, or inappropriate.
        - **Other Activities**: Busin
        - **System State on Completion**:
    - **Use-Case Name**:
        - **Initial Assumption**:
        - **Normal**:
        - **What Can Go Wrong**:
        - **Other Activities**:
        - **System State on Completion**:

3. Design Documents

3.1. Software Architecture

3.2. High-Level Database Schema

- Two example database tables, feel free to modify.

Scams Table:

SCAM_ID	LOCATION	TARGET	DELIVERY_METHOD
1	united_states	senior, corporation	phone
2	india	teenager	text
3	south_korea	bank	phone

Users Table:

USER_ID	FIRST_NAME	LAST_NAME	SCAM_HISTORY
1	John	Cena	{1, 2, 3}
2	Paul		{1}
3	Gregory	Smith	{}

3.3. Software Design

3.3.1. State Machine Diagram: Actor Name (Responsible Team Member)

3.3.2. State Machine Diagram: Actor Name (Responsible Team Member)

3.3.3. State Machine Diagram: Actor Name (Responsible Team Member)

3.4. UML Class Diagram

4. Scenario

4.1. Brief Written Scenario with Screenshots