**Software Design**

**Document**

**for**

**Home and Office and Security Scanner (H.O.S.S.)**

**Version 1.1 approved**

**Prepared by** Garret Stewart, Minsu Lee, Asahel Monroy, Ricardo Castellanos, Luis Cortez, Diego Flores, Ashley Pablo, Guang Wu, Jivan Karapetian, Adrian Holloway

Advisor: Dr. Huiping Guo

**Liaison**: Anson Fong

05/12/2023

Table of Contents................................................................................................................... <pg 2>

Revision History................................................................................................................-.. <pg 4>

1. Introduction................................................................................................................ <pg 5>
   1. Purpose........................................................................................................... <pg 5>
   2. Document Conventions……………………................................................. <pg 5>
   3. Intended Audience and Reading Suggestions................................................ <pg 5>
   4. System Overview........................................................................................... <pg 6>
2. Design Considerations............................................................................................... <pg 6>
   1. Assumptions and dependencies..................................................................... <pg 6>
   2. General Constraints........................................................................................<pg 7>
   3. Goals and Guidelines.....................................................................................<pg 7>
   4. Development Methods................................................................................... <pg 8>
3. Architectural Strategies............................................................................................<pg 8>

3.1. Products Used................................................................................<pg 8>

3.2 Reuse of Existing Software...............................................................<pg8>

3.3. Future Plans for Improving the Software......................................<pg 8>

3.4 User Interface Paradigms………..................................................<pg 8>

3.5. Hardware/Software Interface Paradigms.......................................<pg 8>

3.6 Error Detection and Recovery.......................................................<pg 9>

3.7. Memory Management Policies......................................................<pg 9>

3.8 Data Storage Management and Persistence...................................<pg 9>

3.9. Distributed Data and Control over a Network...............................<pg 9>

3.10 Generalized Approaches to Control...........................................<pg 10>

3.11. Concurrency and Synchronization……...................................<pg 11>

3.12 Communication Mechanisms………….......................................<pg 11>

1. System Architecture.................................................................................................<pg 12>
   1. .....................................................................................................................<pg 12>
2. Policies and Tactics..................................................................................................<pg 13>
   1. Specific Products Used.................................................................................<pg 13>
   2. Requirements traceability.............................................................................<pg 13>
   3. Testing the software.....................................................................................<pg 13>
   4. Protocols.......................................................................................................<pg 13>
   5. Maintaining the software..............................................................................<pg 14>
   6. Interfaces......................................................................................................<pg 14>
   7. System's deliverables....................................................................................<pg 15>
3. Detailed System Design...........................................................................................<pg 17>

6.1 Main Module….............................................................................................<pg 17>

6.2 UI Module.....................................................................................................<pg 17>

6.3 Scanning Module...........................................................................................<pg 18>

6.4 Scoring Module….........................................................................................<pg 18>

1. Detailed Lower level Component Design

7.x Name of Class or File....................................................................................<pg 15>

7.x.1 Classification........................................................................................<pg 15>

7.x.2 Processing Narrative(PSPEC)..............................................................<pg 15>

7.x.3 Interface Description............................................................................<pg 15>

7.x.4 Processing Detail...................................................................................<pg 15>

7.x.4.1 Design Class Hierarchy.................................................................<pg 16>

7.x.4.2 Restrictions/Limitations................................................................<pg 16>

7.x.4.3 Performance Issues.......................................................................<pg 16>

7.x.4.4 Design Constraints.......................................................................<pg 16>

7.x.4.5 Processing Detail For Each Operation.........................................<pg 16>

1. Database Design.......................................................................................................<pg 18>

8.1 ER Diagram....................................................................................<pg 19>

1. User Interface
   1. Overview of User Interface.........................................................................<pg 20>
   2. Screen Frameworks or Images....................................................................<pg 20>
   3. User Interface Flow Model.........................................................................<pg 23>
2. Requirements Validation and Verification..............................................................<pg24>
3. Glossary....................................................................................................................<pg 26>
4. References................................................................................................................<pg 27>

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| All HOSS group members | 09/12/22 | Initial draft of Software Design Document | 1.0 |
| All HOSS group members | 05/11/23 | Final draft incorporating all updates to HOSS software. | 1.1 |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

**1.1 Purpose**

The purpose of this document:

1. To analyze and scan the commonly used and brand new IOT devices on wifi networks
2. To educate the reader on the programs/technology needed to create and run this project
3. To describe how to properly utilize this software application
4. To highlight the current features and how a user may interact with them

This design document covers both the front-end design, back-end design and structure with sufficient level of detail. Some diagrams and user interface mock-ups are provided as well as descriptions of the main functions of the software application. However, this document does not go into detail as to how these functions are implemented.

**1.2 Document Conventions**

The Software Design Specification Document (denoted as SDS) is written in a way to help highlight information with special significance. Each section has a heading written in the biggest font and in bold. It will provide the reader with the title of the section and what it will touch on. Then, there are subheadings, in a slightly smaller font than the headings, also in bold. Then, it is followed by paragraphs, images, and lists that go in-depth for that specific topic. Visualizations like photos and diagrams are added to provide the reader with illustrations on the topic.

**1.3 Intended Audience and Reading Suggestions**

The SDS is intended for a broad range of readers: such as developers, project managers, marketing staff, testers, and general HOSS users that are interested in an in-depth description of this software application. With the inclusion of the glossary, it is assumed that all readers have sufficient knowledge to understand the content.

It is recommended that developers and project managers read the full document from beginning to end in order to have a strong understanding of the expected product, constraints, and range of freedom in terms of development. Marketing staff can focus on Section 9: User Interface, which contains UI mockups and a possible logo for the software application. Users are suggested to first read Section 11: Glossary in order to understand the terminology used in this document before heading to the sections they wish to read. Testers can skip ahead to Section 10: Requirements Validation and Verification for a simple overview of requirements that must be met.

**1.4 System Overview**

HOSS is a software application that scans a network for all connected devices. HOSS will utilize the NMAP library to scan and the Fing DevRecogAPI to identify the devices by model and type, then return the list of devices within the user’s network. After retrieving the list, HOSS will cross-reference its own database with historical data regarding the security/privacy settings of each device. After HOSS cross-references the database, it compiles and aggregates two security scores. The scores will be based on a 5 point rating based on the security and privacy of each device.

**2. Design Considerations**

**2.1 Assumptions and Dependencies**

2.1.1 Software Used

2.1.1.1 NMAP library

2.1.1.2 Javascript

2.1.1.3 HTML5

2.1.1.4 CSS

2.1.1.5 MongoDB

2.1.1.6 Python

2.1.1.7 Bootstrap

2.1.1.8 Fing DevRecogAPI

2.1.1.9 CVE API

2.1.2 End-user Characteristics

2.1.2.1 Users are assumed to have access to a network and a stable internet connection

in order to use the software application as intended

2.1.2.2 Users are assumed to have IoT devices that will be scanned by the software application.

2.1.2.3 Users should be able to recognize all their connected devices.

2.1.2.3.1 In case of any unknown device, the device shall still be stored in our database for our back-end to further research and manually update for future scans.

2.2.2.3.1 It is the User’s responsibility to address any unknown device to assure it is a device they intend to have in their network.

**2.2 General Constraints**

2.2.1 HOSS is required to run on a user's device because the NMAP library cannot scan devices in a network across the internet.

2.2.2 Manually inputting data for devices into the system database may not be feasible in the

future if there is a substantial increase in users and thus an increase in undocumented

devices

2.2.3 Users may have certain device or network settings enabled that can block the main

functionality of the application

2.2.3.1 If possible, the application should detect these cases and suggest to the user what

they should do if they wish to use its function

2.2.4 As of this time, there are no plans for a mobile application. The scope of this application is

assuming the user has a home computer.

2.2.4.1 Additionally, the scope of this application assumes the user will be scanning one

home router.

**2.3 Goals and Guidelines**

The main goal of this software application is to provide its users with relevant and important information about the devices connected to their network. Users can learn about the privacy and security terminology associated with internet of things devices.

This software application does not make any changes to a user's devices or attempts to fix their vulnerabilities. Its only purpose is to highlight potential problems in security and privacy for the user to then make their informed decision/action.

**2.4 Development Methods**

2.4.1 Agile Development

2.4.1.1 The developers shall incorporate Agile Development principles in the making of

our application. The primary methodology utilized will be responding to change over following a plan, since we anticipate several changes during the development of our application.

**3. Architectural Strategies**

**3.1 Products Used**

3.1.1 JavaScript

3.1.2 MongoDB

3.1.3 HTML

3.1.4 NMAP library

3.1.5 Python

3.1.6 BootStrap

3.1.7 Github

3.1.8 Fing DevRecogAPI

3.1.9 CVE API

**3.2 Reuse of Existing Software**

3.2.1 NMAP library

3.2.2 MongoDB

3.2.3 BootStrap

3.2.4 Fing DevRecogAPI

3.2.5 CVE API

**3.3 Future Plans for Improving the Software**

3.3.1 Monitor user’s network for traffic to build security score

3.3.1.1 Implement a server-traffic tracking mechanism that captures the volume of inbound and outbound traffic associated with a given device's MAC address, and incorporate this information as a key factor in determining the device's overall security score

3.3.2 Provide users with recommendations

3.3.2.1 If the user receives a low score on either their privacy or security score, HOSS

will provide the user with devices that are known to have a higher score to help improve

their security and privacy information.

3.3.3 Optimize software application to be more inclusive and provides an equitable user experience for everyone, regardless of their abilities

**3.4 User Interface Paradigms**

3.4.1 The menu-driven user interface focuses on the principal scan function and supplements the

user with a glossary section through the input of menu options and output through a display of information.

3.4.2 The form based interface interacts with the user’s input/output data.

**3.5 Hardware and/or Software Interface Paradigms**

3.5.1 Besides the requirement of our User owning a home computer and a functional network router, HOSS does not have any hardware requirements besides the requirements necessitated by the User. Thus, there are no hardware paradigms.

3.5.2 Software Interface Paradigms: HOSS shall allocate focus to each of the Software Interface Paradigms. We shall allocate 20% to design, 20% to requirements, 20% to coding, and 40% to testing. All of these allocations insure an efficient and error-free deployment of our program.

**3.6 Error Detection and Recovery**

We locked the navigation bar when the user scans for devices to prevent the application from losing functionality during its scans as well as preventing errors. Therefore the navbar will only be functional after the scan is complete to further prevent the loss of functionality.

**3.7 Memory Management Policies**

The NodeJS Express framework, which uses MongoDB for its database, helps organize the software application data/memory in a dynamic manner. This allows for both constant storage and allocation of devices and their respective model/MAC address/type/and host-name, which are retrieved from scan. The NodeJS Express framework also allows for a streamlined User experience, allowing the User to view all past devices scanned in their network. Thus, memory shall be allocated to the HOSS device’s page, along with general MongoDB memory allocation.

**3.8 Data Storage Management and Persistence**

The HOSS software application uses MongoDB for its database, and is organized in the following manner:

3.8.1 Device Data

3.8.1.2 Device Name

3.8.1.3 Device Brand

3.8.1.4 Device encryption method

3.8.1.5 Device authentication method

3.8.1.6 Historical data on any data breach or hack the device may have had

3.8.1.7 Device security and privacy score

3.8.1.8 Device Brand security and privacy score

3.8.1.9 Last date score was updated

**3.9 Distributed data or control over a network**

HOSS will not have any control over the connected network and scanned data will only be shown locally, and not saved online. The saved devices are only presented to the User relative to their local network, despite also being stored in our MongoDB for future use and reference.

**3.10 Generalized Approaches to Control**

The only form of User input which involves transfer of data is via the User interacting with the “Scan” button on HOSS’s scan page. To ensure problem flow of control and data, the User cannot press the “Scan” button again until the first network scan is complete. All other factors of User involvement are going from one HOSS tab to another, which are static pages.

Users also cannot access database information unless given special permissions by a HOSS server-side admin. This allows only a select few to be able to manipulate the database information, and prevents random users from affecting a device’s information.

**3.11 Concurrency and Synchronization**

MongoDB processes, stores, and retrieves data to and from the server-side and client-side It also enforces that all data displayed in the front-end is correct to all users using the site

**3.12 Communication mechanisms**

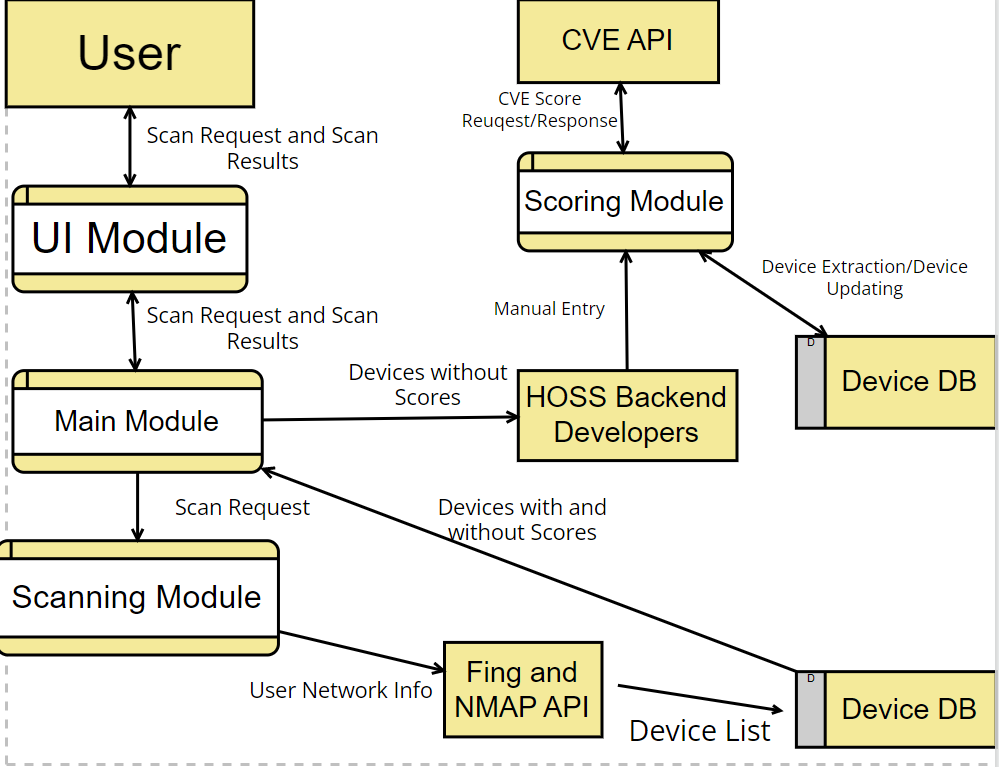
4.12.1 Wi-Fi

4.12.2 LAN

**4. System Architecture**

Below is our Data Flow Diagram for HOSS.

4.1 DFD Level-1



The explanation and rationale for the DFD Level shall follow. The initial step is the user interacting with our UI model by submitting a scan request.

**5. Policies and Tactics**

**5.1 Choice of which specific products used**

5.1.1 Programming interpreter

5.1.1.1 JavaScript

5.1.1.2 Python

5.1.2 Database

5.1.2.1 MongoDB

5.1.3 IDE

5.1.3.1 Visual Studio Code

5.1.4 Libraries

5.1.4.1 Eel

5.1.4.2 NMAP

5.1.4.3 Pymongo

5.1.5 Markup Language

5.1.5.1 HTML5

5.1.6 Framework

5.1.6.1 BootStrap

5.1.7 API

5.1.7.1 Fing

5.1.7.2 CVE

**5.2 Plans for ensuring requirements traceability**

5.2.1 Database

5.2.1.1 The database shall ensure HOSS maintains a constant repository for known IoT devices in order to provide as detailed and descriptive information as possible to the user.

5.2.2 Scanner

5.2.2.1 HOSS shall maintain a constant connection with the NMAP library in order to consistently perform IoT scans for any user that interacts with our software.

**5.3 Plans for testing the software**

5.3.1 Plans for maintaining software

5.3.1.1 Provide software updates periodically and provide database management. Distribution of updates and management will be tested

5.3.2 Interfaces for end-users, software, hardware, and communications.

5.3.2.1 Will be tested on a weekly basis

5.3.2.2 There shall be a backend for HOSS developers to constantly add new IoT devices on a weekly basis that we did not have enough information on during a user scan.

5.3.3 Tactics for importing code

5.3.3.1 We will have a generic DBservice class for our Database Code, assuring that in the event we need to use a different SQL compiler such as MySQL or PostGreSQL, the only adjustment needed would be adjusting the semantics of the DSservice class. All other code shall be specific to our designed programming languages.

5.3.4 How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

5.3.4.1 N/A: Javascript and Python being interpreted languages mean there is no need for compilation or linking/loading as it is accomplished at runtime.

**6. Detailed System Design**

**6.1 Main Module**

**6.1.1 Responsibilities**

The main module will interact with the UI module, scoring module and scanning module. This module will provide the communication between all modules and act as the middle man.

**6.1.2 Constraints**

N/A

**6.1.3 Composition**

The subcomponents are the UI Module, Scoring module and Scanning module.

**6.1.4 Uses/Interactions**

The main module collaborates with the scoring module and requests score data and sends a list of identified devices to the scoring module. The main module also collaborates with the scanning module by requesting and receiving NMAP/Fing devices from the scanning module. The UI interface will also collaborate with the main module by requesting and receiving scan requests.

**6.1.5 Resources**

N/A

**6.1.6 Interface/Exports**

N/A

**6.2 UI Module**

**6.2.1 Responsibilities**

The UI module will request a scan and get a scan response. The UI module will be the main interface Users will interact with.

**6.2.2 Constraints**

The User needs to be connected to a network in order to interact with the UI module.

**6.2.3 Composition**

The subcomponents will be the outside agent, the User, and main module.

**6.2.4 Uses/Interactions**

The UI module will send a scan request to the main module and receive a scan result from the main module.

**6.2.5 Resources**

N/A

**6.2.6 Interface/Exports**

Will receive data as forms of double, ints and strings.

**6.3 Scanning Module**

**6.x.1 Responsibilities**

The primary responsibility of the Scanning Module is to request and receive NMAP devices that are connected to the customer’s WiFi network. It will then have NMAP send these devices to our Fing API for specific model information. After this, the devices shall be stored into our MongoDB for further evaluation and retrieval by our Scoring Module.

**6.x.2 Constraints**

Scanning module will just scan IoT devices connected to the customer’s WiFi Network. There must be a stable, secure WiFi connection.

**6.x.3 Composition**

The database contains all the devices known by HOSS.

**6.x.4 Uses/Interactions**

The scanning module will interact with both ends of the software. It will scan the network and interact with our database to either check if the device exists or to add a new device.

**6.x.5 Resources**

N/A

**6.x.6 Interface/Exports**

N/A

**6.4 Scoring Module**

**6.x.1 Responsibilities**

The scoring module is responsible for three things: retrieving a privacy and security score from the database, and averaging the two scores for a total score. The third responsibility will be to send a flag to the backend of the software in the event that a requested device does not yet have any scores in the database, so the administrators of the software can manually research and enter scores. The security score is an average between a device’s brand score and a device’s model score. Another factor incorporated in the security score is the list of all vulnerabilities returned by our CVE API.

Our privacy score is determined by a brand’s privacy policies regarding if they sell a User’s data, and what type of sensitive information a device contains about a User. This information includes, but is not limited to: address, social security number, DOB, and payment information.

Security score is determined by encryption method, default settings, authentication, angst CVE vulnerabilities. A weighted score is assigned to each category.

**6.x.2 Constraints**

Scoring data from the database will be stored and returned as an integer, and the calculated average will be stored and returned as a double.

**6.x.3 Composition**

The database contains all the devices known by HOSS and their respective scores, if applicable at the time of scanning, or a null value otherwise.

**6.x.4 Uses/Interactions**

The scoring module will interact with both the database and with the backend of the software. The database interaction will consist of fetching scores in the database to return to the user, and the backend interaction will consist of sending a flag to indicate the necessity of manual entry in the event that there is no score for a given device in the database.

**6.x.5 Resources**

N/A

**6.x.6 Interface/Exports**

N/A

**7. Detailed Lower level Component Design**

**7.x Component Design**

**7.x.1 Classification**

N/A

**7.x.2 Processing Narrative (PSPEC)**

N/A

**7.x.3 Interface Description**

**7.1.3** See Section 9 for description and aesthetic representation on our user interface.

**7.x.4 Processing Detail**

N/A

**7.x.4.1 Design Class Hierarchy**

See our alpha ER Diagram for an initial perspective on the Design Class hierarchy.

**7.x.4.2 Restrictions/Limitations**

7.1.4.2 See performance issues and design constraints

**7.x.4.3 Performance Issues**

7.1.4.3 The only issue that may hinder the performance of HOSS is the user’s internet connection, which may compromise HOSS’s ability to scan efficiently and expediently.

7.2.4.3 If the User has an excessive amount of IoT devices within their home network, the Scan may take longer than expected. The ideal Scan time is 10 - 15 seconds.

**7.x.4.4 Design Constraints**

7.1.4.4.1 The largest constraint for HOSS’s design is the ability to capture all known IoT devices, which is impossible given the large number of IoT devices. To circumvent this constraint, HOSS developers will add IoT device information to our database on a weekly basis.

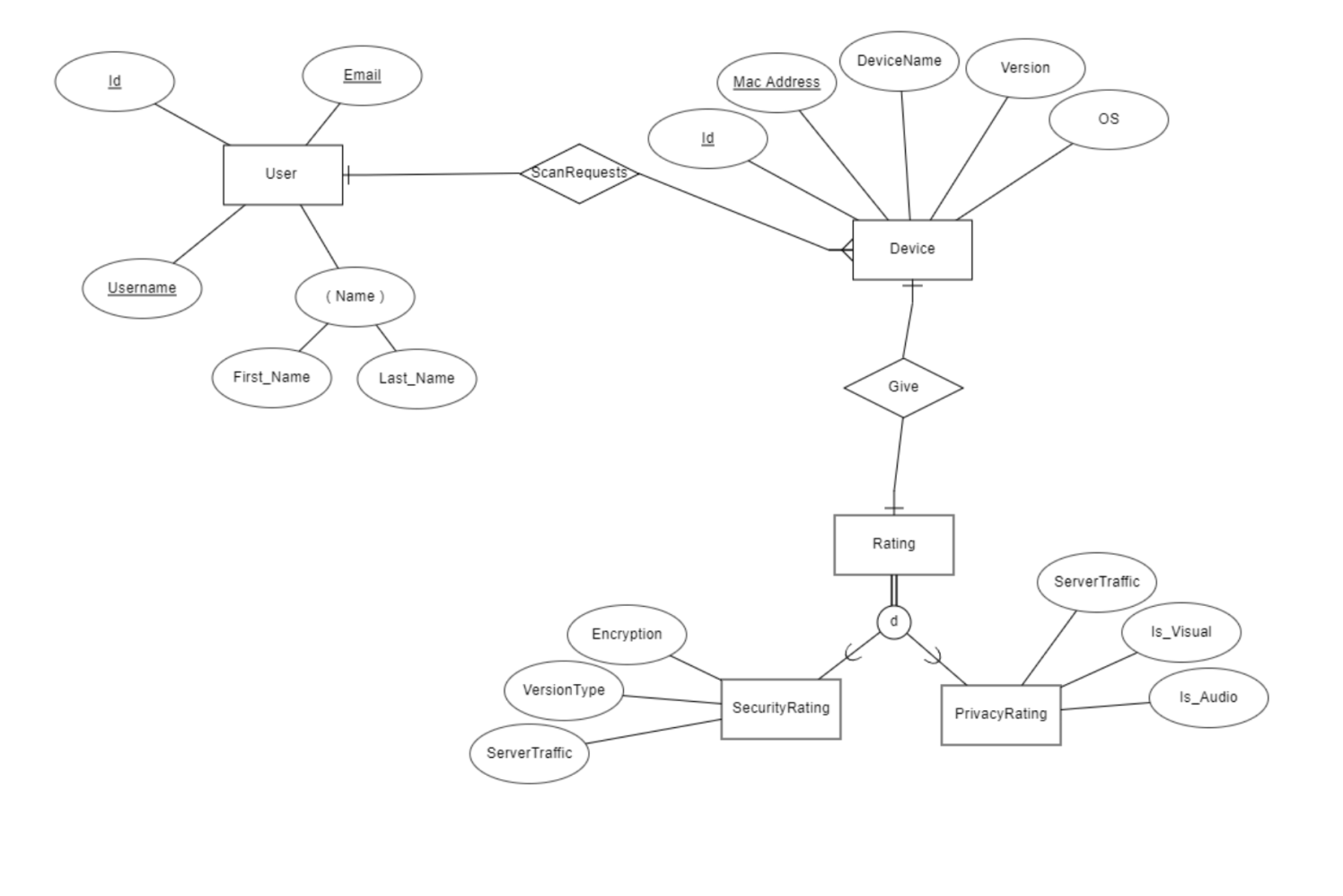
**7.x.4.5 Processing Detail For Each Operation**

N/A

**8. Database Design**

MongoDB is a platform used for implementing our database design. A preliminary draft of our Entity-Relationship Diagram is presented below. Since MongoDB is a versatile, JSON-based query language, it is not always assumed that each entity will contain the exact number of attributes presented below. This is because of the sheer amount of IoT devices, we are expecting to retrieve a fair amount of null and unexpected values from each scan.

8.1 ER Diagram



Our Database shall contain three entities: User, Device, and Rating. Each factor will have their own respective attributes that will be accounted for in our relational schema and database. Rating extends into two subclasses: SecurityRating and PrivacyRating. The attributes of these subclasses are tentative and may include more attributes once more NMAP Scan results are conducted. It was initially discussed amongst our group if our Glossary required a database, but since our Glossary shall be static with known terminology, it does not require a database for the storage of its information.

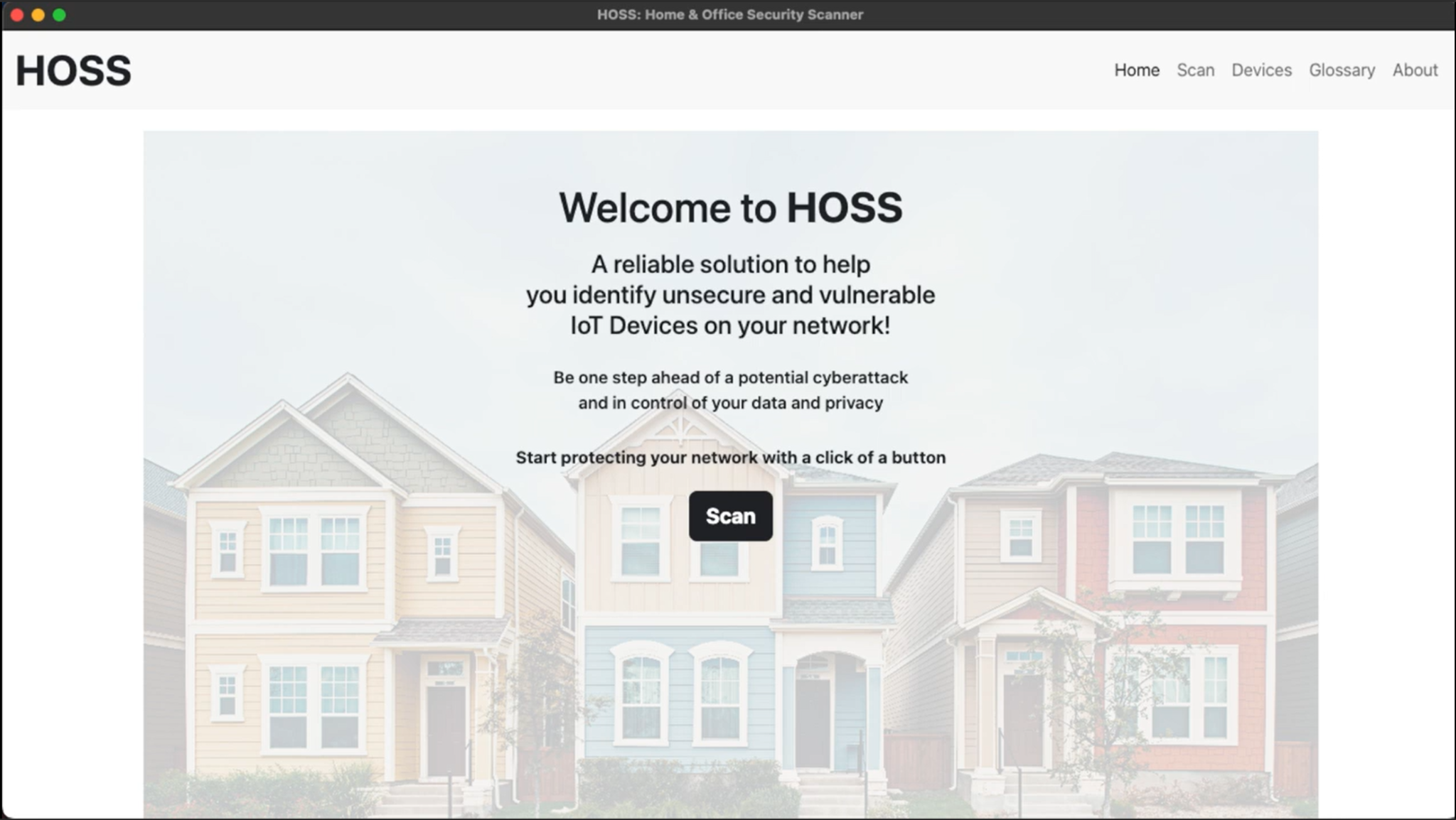
**9. User Interface**

**9.1 Overview of User Interface**

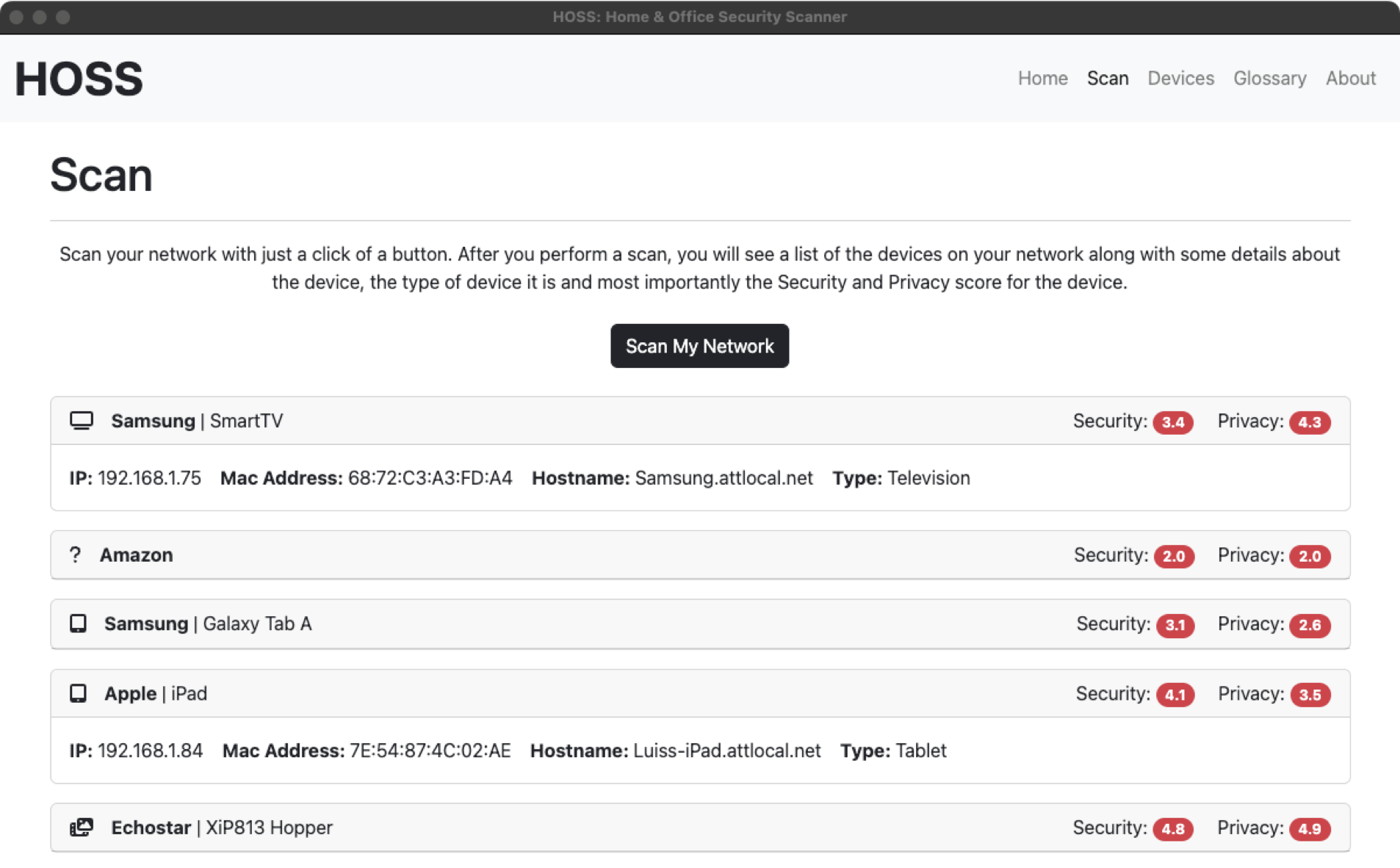
See 9.2 for User Interface Overview, Flow Diagram, and preview of all important HOSS user interface pages.

**9.2 Screen Frameworks or Images**

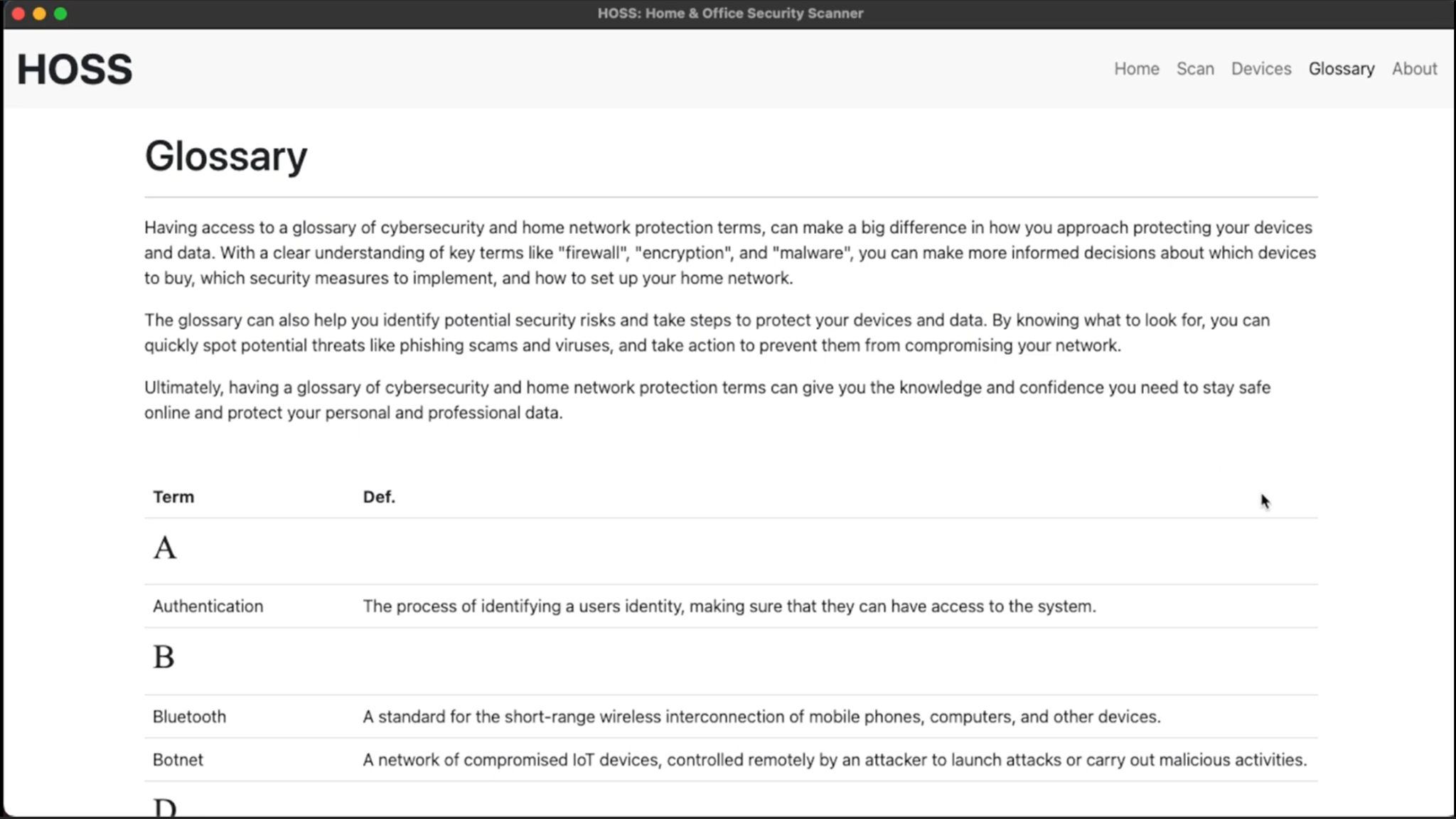
9.2.1 Home Page



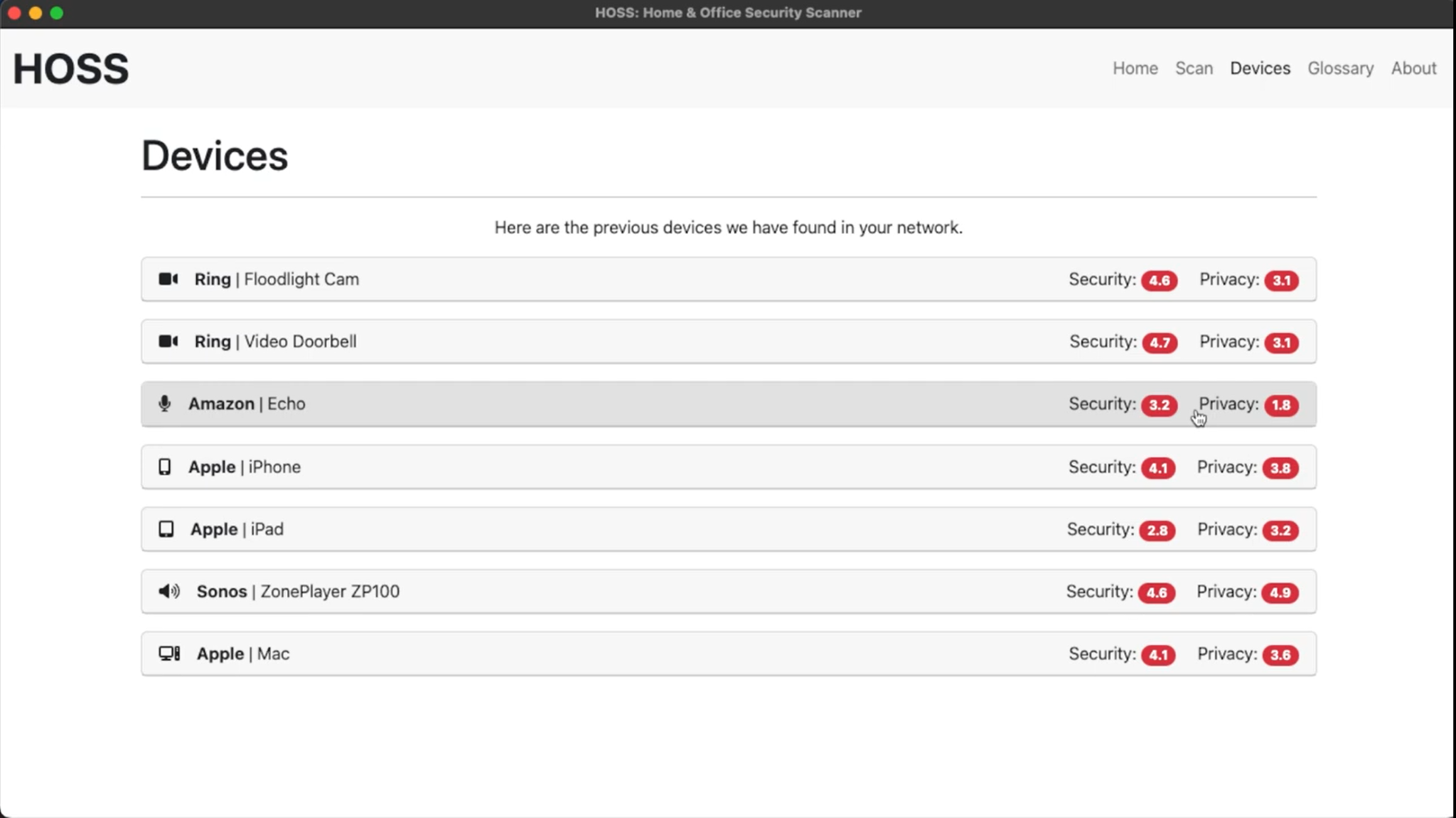
9.2.2 Scan Results Page



9.2.3 Glossary Page



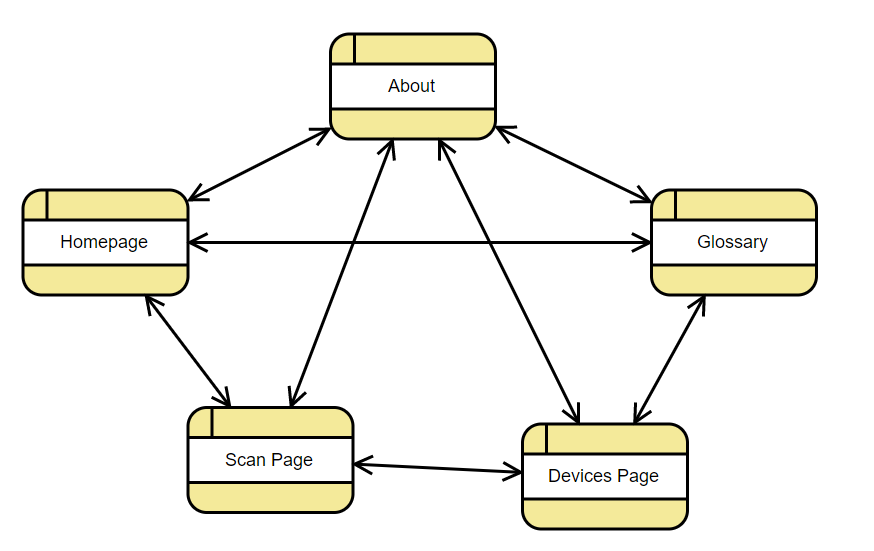
9.2.3 Devices Page



9.2.4 About Page



**9.3 User Interface Flow Model (updated)**



HOSS has an extremely versatile User Interface flow. As shown above, the User may go from any desired tab to any other desired tab at any time besides when conducting a scan. HOSS shall wait for the scan to conclude prior to the user going to another tab. While it is not shown above, since HOSS is a software application, the user may also exit from HOSS at any time. As shown in the User Interface pictures above, the red “exit” button on the top left is always present on every screen.

**10. Requirements Validation and Verification**

Due to the extensive amount of requirements HOSS contains, we have chosen a select amount of requirements which are deemed of high importance to include in this section.

|  |  |  |
| --- | --- | --- |
| 4.1.1.1 The application shall be supported by modern computers | Our entire HOSS program must satisfy this requirement since none of the modules would run if HOSS isn’t run on a modern device. | HOSS shall be tested on multiple modern platforms. |
| 4.1.1.2 The application shall provide a main page to the user | Our user interface module shall satisfy this requirement. | HOSS shall ensure the main page is both user friendly and is extensively tested by the front-end. |
| 4.1.1.3 The application shall provide a scan results page | Our user interface module, scanning module, and scouring module shall satisfy this requirement. | HOSS shall perform preliminary and mock scans to ensure the scan results page provides correct information. |
| 4.1.1.4 The application shall provide a Glossary page | Our user interface module shall satisfy this requirement. | HOSS shall ensure the glossary page opens properly from both the home page and the scan results page. |
| 4.2.1.2.1 The scan results page shall display a list of all devices connected to the user’s network | Our scanning module shall satisfy this requirement. | Since our scanning module utilizes NMAP, the scan results page will display a list of all connected devices since NMAP will capture these devices which will then be cross-referenced in our database. |
| 4.2.1.2.2 The scan results page shall list the devices in such a way that the device name, security rating, and privacy rating are visible | Our scanning module and scoring module shall satisfy this requirement. | Our scoring module shall be tested by assuring accurate scores and provided for each category to assure these scores are correctly presented to the user. |
| 4.2.1.3.1 The glossary page shall display a list of relevant terms related to privacy and security and their definitions | Our user interface module shall satisfy this requirement. | HOSS shall experiment by explaining its glossary terminology to people with minimal experience in the Computer Science field, to assure the glossary explains all terminology in an easy-to-understand manner. |
| 4.2.1.3.2 HOSS shall store a list of previously scanned User devices in the devices page. | Our User interface module shall satisfy this requirement. | The devices page successfully stores all devices from past scans. |
| Scan result shall identify all devices on network | Our Scans successfully identify all IoT devices in a User’s network. | While HOSS may not always identify a device’s name and model, it shall always identify the presence of a device. |
| Scan table shall have info regarding each devices model and brand | Our Scan table contains info regarding each device’s model and brand. | In any edge case where we are unable to identify the brand or model, a flag is sent to our back-end to manually input this information. |
| Scan table shall have info regarding each devices Mac address and IP address | HOSS is successfully able to retrieve a device’s Mac and IP address | There is no edge case where HOSS shall be unable to retrieve this info. |

**11. Glossary**

* **Bootstrap** - CSS Framework used for developing responsive and mobile-first websites.
* **Fing DevRecogAPI** - Network scanner that identifies devices based on brand, model, and type.
* **HOSS** - Home and Office Security Scanner
* **IoT** - Internet of Things, which describes the network of physical objects that have software, sensors, and other technologies.
* **JavaScript** - A programming language that creates dynamic and interactive content for web content. It manipulates, calculates, and validates data.
* **MongoDB** - A non-relational database that uses JSON-like documents. It’s designed to store large amounts of data and let the user work with it efficiently.
* **NodeJS** - Backend javascript framework used for server side programming.
* **NMAP** - Network scanner that checks for vulnerabilities and discovers hosts by sending packages and analyzing the response.
* **Python** - General purpose programming language used in web development and software development.
* **SDS** - Software Design Specification

**12. References**

NMAP API Website - <https://nmap.org/book/nse-api.html>

Fing API Documentation - <https://www.fing.com/images/uploads/general/Fing_Cloud_API_v1.0.pdf>

CVE API Website - <https://nvd.nist.gov/developers/vulnerabilities>

MongoDB Documentation - <https://www.mongodb.com/docs/>