System Requirements Specification

for

AutoPen

Version 5.1 Approved

Prepared by Michael Allen

Caleb Hall

Calla Robinson

Myles Scott

Joshua Buscher

3/3/2023

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

1. Revision History

Name	Date	Reason For Changes	Version
Michael Allen	9/21/23	Rewording Section 1.4 and 2	1.0
Michael Allen	9/26/23	Section 4 and 5	1.1
Michael Allen	9/28/23	Section 3 and 6	1.2
Caleb Hall	9/29/23	Overall Editing and formatting	1.3
Myles Scott	10/26/23	Changing title from 1.0 to 2.0, adding use case diagram	2.0
Caleb Hall	10/31/23	Editing 1.5, 3.3	2.1
Michael Allen	10/31/23	Editing Title Page, Table of Contents	2.2
Joshua Buscher	11/21/23	Editing Section 2.5, 2.7	3.1
Caleb Hall	11/21/23	System diagram, operating environment, editing	3.2
Joshua Buscher	2/5/24	Removing all usage of words “AI”/”Artificial Intelligence”	4.0
Caleb Hall	2/5/24	Editing, applying change in scope	4.1
Joshua Buscher	3/2/24	Altering Reasons for Changes, editing requirements,	5.0
Calla Robison	3/3/24	Requirements 3.1, 3.2, 3.3, 3.4	5.1
Michael Allen	3/3/24	Formatting the document	5.1

1. Introduction

Section 1 is an introduction to the concepts and ideas to be discussed later in the document, and covers the purpose of the document in 1.1, the convention of the document in 1.2, the audience the product is intended for in 1.3, the scope of the product in 1.4, and the references used for the product in 1.5.

1.1 Purpose

With the need for stronger, faster security with the increase of cybersecurity threats, AutoPen aims to revolutionize the field of penetration testing by leveraging BurpSuite to conduct automated, comprehensive penetration tests. The primary objective is to provide businesses and organizations with a quick, cost-effective and thorough way to identify and rectify potential vulnerabilities in their networks and systems. This SRS covers both the integrated testing suite and the website that will be used to access said application.

1.2 Document Conventions

1. The document will be in Times New Roman size 12 font.
2. The requirements have their own authority.

1.3 Intended Audience and Reading Suggestions

This documentation on AutoPen is designed specifically for web developers and web testers as users of this system.

The outline for the entire document is as follows: Section 1 provides an initial overview of the document, explaining its purpose, conventions, and scope. Section 2 delves into a comprehensive understanding of the product, its functions, characteristics, and constraints. Section 3 details the specifications for how the system interacts with its users, hardware, other software, and communications networks. Section 4 describes individual features or functionalities of the system in depth. Section 5 enumerates the system's performance, safety, security, quality, and business rules ensuring it meets established standards and stakeholder expectations. Section 6 contains additional information in appendixes.

1.4 Product Scope

The scope of this product is having a functional automated penetration tester using the Burp Suite API, and a functional website to be used to call it. The product will not contain any Artificial Intelligence to do said penetration testing, and will not be using a custom made web browsing virtual machine to function.

1.5 References

- The URL of the web application is https://www.autopentest.net/
- Jira site for agile sprint planning can be found at: https://autopentest.atlassian.net/jira/software/projects/PEN/boards/1
- Documentation for BurpSuite, the APT used to run penetration testing, can be found at: https://portswigger.net/burp/documentation
- Github for the project can be found at: https://github.com/Caleb-Hall-1015/AutoPen

2. Overall Description

Section 2 covers the overall description of the product and is composed of the perspective of the product in 2.1, the product function in 2.2, user information and classes in 2.3, the operating environment in 2.4, the design and implementation constraints in 2.5, documentation on how users will be able to learn how to use the product in 2.6, and assumption and dependencies in 2.7.

2.1 Product Perspective

The integration of automated penetration testing represents a new paradigm shift in the field of cybersecurity. By replacing manual systems and innovating with advanced technologies, organizations can enhance their security posture, reduce vulnerabilities, and stay ahead of potential cyber threats. The increased efficiency, accuracy, and scalability offered by automated penetration testing make it an indispensable tool in today's digital landscape. Embracing automation in penetration testing is not only a step towards better security but also a strategic investment in the long-term success and resilience of an organization's IT infrastructure.

The image below (Figure 1) shows a system diagram for the product. It is split into two main sections, the Virtual Machine or VM, and Digital Ocean. The VM is made up of the web app that is being tested, and the Burp Suite API that tests the web app. Digital Ocean is comprised of a vulnerability database, Django REST API and a browser. The vulnerability database that stores all the vulnerabilities the Burp Suite API found. The Django REST API that shares the details of the targeted website, and returns the results of the penetration testing. Finally, the browser is what allows the user to input the URL to be tested, and what shows the user the results of the tests.

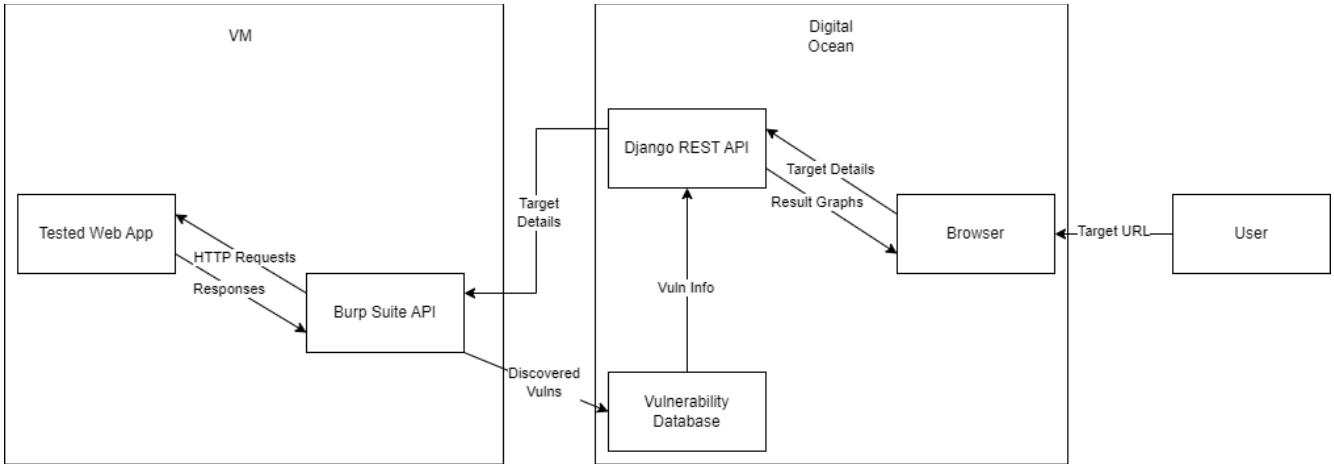

Figure 1: System Diagram

- Figure 1 presents the system architecture, illustrating the flow of data and interactions between different components. At the core is a web application tested through a Virtual Machine (VM), which communicates with Burp Suite API to handle HTTP requests and responses, and to identify vulnerabilities. The discovered vulnerabilities are logged into a Vulnerability Database. Additionally, a Django REST API is employed to manage target details and vulnerability information, which can be accessed via a browser interface hosted on Digital Ocean. This browser interface allows the user to input a target URL and visualize result graphs and target details.

2.2 Product Functions

1. The product must allow users to test a website for vulnerabilities
2. The product must be a functional website, allowing any user to visit using the internet

2.3 User Classes and Characteristics

The users fall into two categories: end user and internal developer.

2.3.1 End User

End users are the web app developers who are testing their own site.Also, End users only have read/write permissions on their own profiles,

2.3.2 Internal Developer

Internal developers are AutoPen employees. In addition, internal developers have admin access across the site. Both classes are equally important to satisfy.

2.3.3 Use Case & Relationships

User Registration: This use case allows unregistered users to create an account by providing necessary details such as username, email, and password. The system then validates this information and stores it, enabling the user to access the system with their new credentials. No prerequisites exist for this action, and upon completion, the user's data is securely stored in the system.

User Login: Registered users can log into the system by entering their credentials, which the system verifies. If the credentials are correct, the system grants access, establishing a user session. This process requires that a user account already exists and results in the user being logged in.

Initiate Penetration Test: Authenticated users can start a penetration test by specifying a target URL and the depth of the test. The system then processes this request and begins the test, requiring the user to be logged in prior to this action. Once initiated, the penetration test seeks to identify vulnerabilities.

Generate Test Report: Upon completion of a penetration test, authenticated users can request a detailed report from the system. This report includes findings on vulnerabilities, their severity, and suggested remediations. The generation of this report depends on the prior completion of a penetration test.

Payment for Premium Features: This use case allows registered users to pay for premium features through a payment gateway. The system processes these payments and, upon success, grants access to the requested premium features. Users must be logged in and have requested premium features for this process to be initiated.

External Vulnerability Database Query: The "AutoPen" system autonomously queries an external database to fetch updated vulnerability definitions and related data. This process is integral to the system's functionality, ensuring that the penetration tests are conducted with the latest information. There are no user-related pre-conditions for this action.

Cloud Storage for Report Storage: The system interacts with cloud storage solutions to manage the storage and retrieval of penetration test reports. This backend process facilitates efficient report management, requiring a system-level request to store or retrieve specific reports.

System Notification: The "AutoPen" system sends notifications to users based on specific events, such as the completion of tests, availability of reports, or system updates. These notifications ensure users are informed about relevant activities and results within the system. Notifications

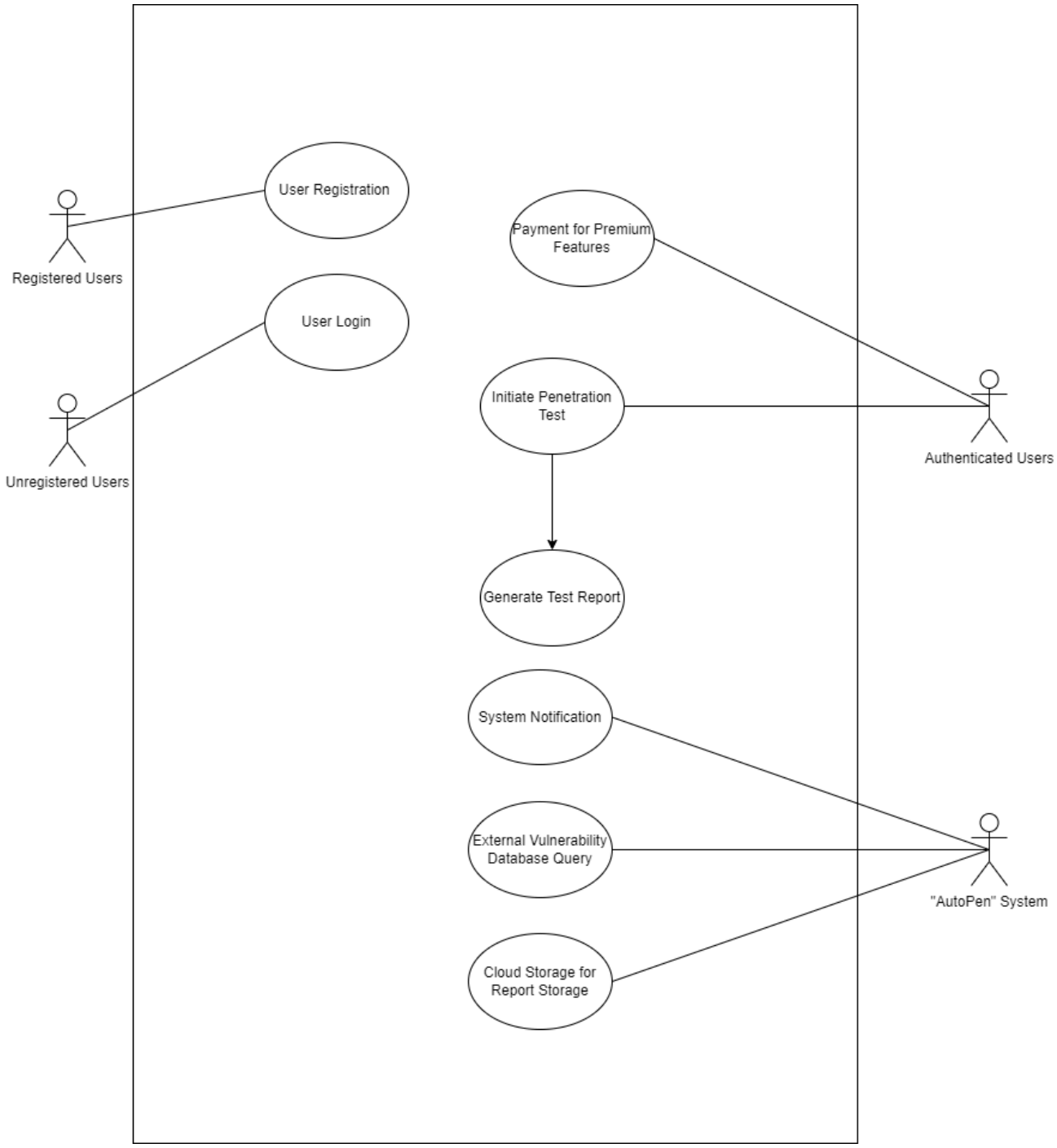

Figure 2: Use Case Diagram

- The diagram, denoted as Figure 2, outlines the use case flow for a security testing service. It differentiates between registered and unregistered users, showing that registered users can proceed to user login, while new users must undergo user registration. Once authenticated, users have the option to pay for premium features. The primary functionality for authenticated users is to initiate a penetration test, which leads to the generation of a test

report. Subsequently, the system issues a notification, queries an external vulnerability database, and stores the report in cloud storage. This process is part of the 'AutoPen' system that orchestrates the security testing workflow.

2.4 Operating Environment

To effectively utilize the required system, the user must have the following operating environment:

1. Any form of computer that can run a web page
   a. Windows, Apple, or Linux Desktop/Laptop
   b. Android or Apple Mobile Smartphone
2. Stable Internet Connection
   a. At least 1 Mb/s
3. Up-To-Date Website/Web Browser
   a. Latest version of Chrome, Firefox, Safari, or equivalent browser

2.5 Design and Implementation Constraints

The design and implementation of AutoPen must consider the potential legal and ethical constraints associated with the device. By addressing these concerns, such as intellectual property rights, privacy, consumer protection, plagiarism, and human interaction, AutoPen can be developed and used responsibly. It is crucial to prioritize legal compliance and ethical considerations to ensure the successful integration and acceptance of AutoPen in various industries and domains.

Time is another constraint, as the full software development lifecycle must fall within a single academic year. From project vision to final deliverable, this project can take no longer than 10 calendar months, limiting the scope and depth of the project.

A third constraint is financing. Limited funds dictate that all tools used are either provided by the university, or are available for a generally low price. As such, the project is unable to offer the full gamut of cybersecurity tools, rather it relies on the professional version of BurpSuite and the basic tier of DigitalOceans.

The final constraint is skill, as the team is made up of intermediately skilled software engineers. As such, development of in-house penetration testing tools and advanced algorithms is not feasible.

2.6 User Documentation

An instruction panel and video tutorial will be implemented into our website. The AutoPen website aims to enhance user experience, provide clear guidance, and support users throughout their penetration testing journey. This addition will ensure that users can effectively utilize the platform's capabilities and maximize the benefits of automated penetration testing.

2.7 Assumptions and Dependencies

2.7.1 The system shall adhere to General Data Protection Regulation (GDPR) and similar regional data protection laws.

2.7.2 The system shall aim for security certifications to establish trustworthiness.

2.7.3 The system shall use a hosting platform that has integrated or allows the import of virtual machines.

2.7.4 The system must use a hosting platform that has the ability to host back-end algorithm tasks.

2.7.5 The system shall use Django REST API to generate visual references to display.

3. External Interface Requirements

Section 3 covers the External interface requirements associated with the product. The format of Section 3 is as follows: Section 3.1 covers the user interface requirements, Section 3.2 covers the hardware interface requirements, Section 3.3 covers the software interface requirements, and Section 3.4 covers the communications interface requirements.

3.1 User Interfaces

3.1.1 Main Interface for Test Execution

Description: The main interface of the software shall include the following elements:

3.1.1.1 Input Fields for Target Specifications:

1. There shall be input fields provided to capture and specify the target specifications required for the test execution.
2. The input fields should be appropriately labeled and allow users to enter relevant information.

3.1.1.2 Start Test Button:

3.1.1.2.1 The interface shall feature a "Start Test" button, which initiates the test execution process when activated.

3.1.1.2.2 The button should be prominently displayed and easily identifiable for user interaction.

3.1.1.3 Progress Bar:

3.1.1.3.1 A progress bar shall be incorporated into the interface to visually represent the status and progression of the test.

3.1.1.3.2 The progress bar should update in real-time, providing users with a clear indication of the test execution progress.

3.1.1.4 Results Display Area:

3.1.1.4.1 An area designated for displaying test results shall be included in the interface.

3.1.1.4.2 The results display area should present relevant information in a clear and organized manner, facilitating easy interpretation by the user.

3.1.1.4.3 System shall display a summary of vulnerabilities detected and potential fixes.

Dependencies: The proper functioning of the main interface is dependent on the availability of relevant input data for target specifications.

Acceptance Criteria:
- Upon entering valid target specifications and initiating the test by clicking the "Start Test" button
- The progress bar should display incremental updates
- The results display area should showcase the relevant test results upon completion.

3.1.2 GUI Standards

Description: The system shall adhere to modern web application design principles, ensuring ease of use and intuitive navigation in the graphical user interface (GUI).

3.1.2.1 Consistent Design:

3.1.2.1.1 The system shall maintain a consistent design throughout the GUI, following established web application design principles.

3.1.2.1.2 The system shall have all pages containing "Help," "Home," and "Settings" buttons.

3.1.2.2 Intuitive Navigation:

3.1.2.2.1 The GUI shall be designed to facilitate intuitive navigation, allowing users to easily locate and interact with system features.

3.1.2.3 Responsive Layout:

3.1.2.3.1 The system shall implement a responsive layout to fit various screen sizes and mobile devices

3.1.2.4 User-Friendly Controls:

3.1.2.4.1 The GUI shall incorporate user-friendly controls, such as buttons, forms, and input fields, designed for ease of interaction.

3.1.2.5 Feedback Mechanisms:

3.1.2.5.1 Interactive elements within the GUI shall provide clear feedback to users, confirming their actions or indicating the system's response.

Acceptance Criteria:
- Users shall be able to navigate through the system's GUI effortlessly.
- The GUI shall maintain a cohesive and visually appealing design.
- Controls and interactive elements shall behave consistently, providing a predictable user experience.
- The GUI shall be accessible and responsive across different devices and screen sizes.

3.1.3 Error Messages

Description: The system shall generate error messages in the form of red pop-up banners that will be displayed at the top of the screen. These error messages shall provide concise explanations to communicate issues or errors encountered during system operation.

Acceptance Criteria:
- Error messages shall be visually represented as red pop-up banners.
- The placement of error messages shall be consistently at the top of the screen.
- The content of error messages shall be concise and clearly explain the nature of the encountered issue.
- Users shall be able to dismiss error messages or take appropriate actions based on the information provided.

3.1.4 Components with UI

The system shall incorporate the following components with user interfaces:

3.1.4.1 Main Dashboard:

3.1.4.1.1 The system shall display a main dashboard with essential information, such as real-time data and key metrics relevant to the user's role.

3.1.4.1.2 The main dashboard shall be customizable, allowing users to arrange and prioritize displayed elements based on individual preferences.

3.1.4.2 Settings:

3.1.4.2.1 The system shall include a settings interface accessible from the main navigation.

3.1.4.2.2 Users shall be able to modify settings related to system behavior, notifications, and personalization.

3.1.4.2.3 Changes made in the settings interface shall persist across user sessions.

3.1.4.3 Results Page:

3.1.4.3.1 The system shall present a results page with a clear layout, providing detailed information on completed processes, tests, or relevant activities.

3.1.4.3.2 Users shall be able to filter and sort results based on various parameters for better analysis.

3.1.4.4 User Profile:

3.1.4.4.1 The system shall feature a user profile interface accessible from the main menu.

3.1.4.4.2 Users shall be able to view and edit their profile information, including but not limited to name, contact details, and password.

3.1.4.4.3 Changes to the user profile shall be reflected throughout the system

Acceptance Criteria:

- Prompt loading and customization of the main dashboard with persistent user-specific settings
- Accessible and immediately effective settings modifications
- Clear display and user-friendly options for results, including export capabilities
- User profile accessibility, editable details, and accurate reflection of modifications throughout the system

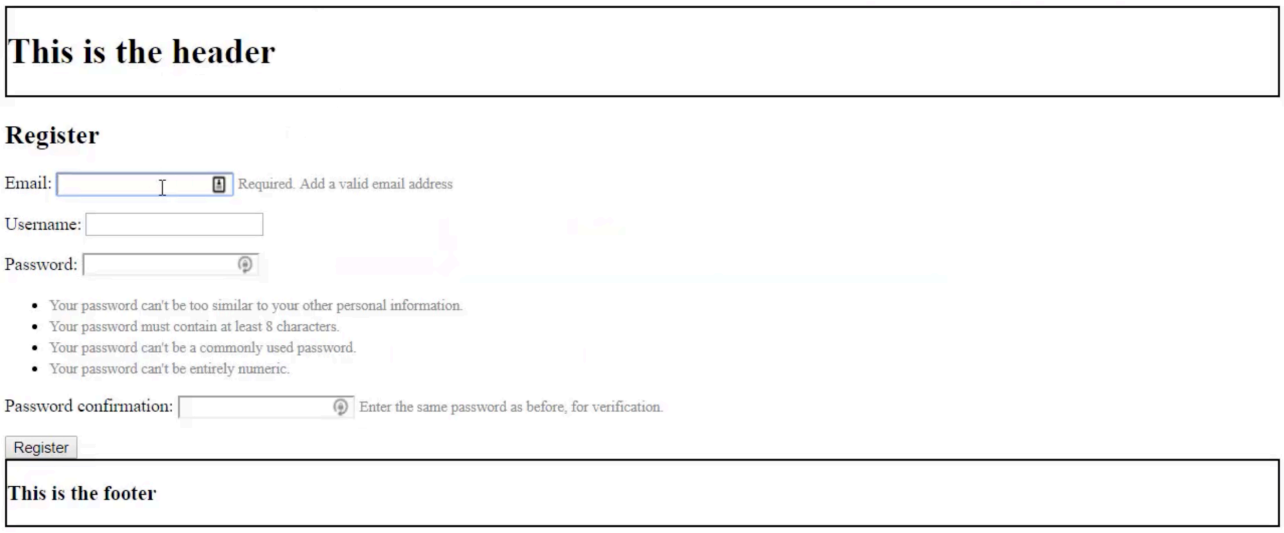

Figure 3: Example User Interface Screen

- Figure 3 depicts a screenshot of an example user interface for a registration page. The interface includes a header at the top and a footer at the bottom, framing the registration form. The form fields prompt the user for an email address, with a note indicating that a valid email address is required, a username, and a password, along with a password confirmation field. Accompanying the password field are instructions specifying the password criteria, such as a minimum length and restrictions on similarity to other personal information. The layout is simple and user-centric, designed to facilitate an easy and secure registration process.

3.2 Hardware Interfaces

3.2.1 The interface shall allow users to access the website seamlessly across multiple devices, including PCs, laptops, tablets, and mobile phones

3.2.2 The application shall be hosted on Siteground's web hosting platform, and user interaction with the system shall primarily occur through the website's servers, which are allocated a minimum of 20 GB web space

3.2.3 The system shall employ Secure HTTP (HTTPS) for all web communications to ensure data security. Additionally, real-time updates shall be facilitated through the use of WebSockets

3.3 Software Interfaces

3.3.1 Connected Software:

3.3.1.1 The system shall utilize a PostgreSQL database hosted on Digital Ocean

3.3.1.2 The system shall operate within Sitegrounds' web hosting environment and be compatible with a Kali Linux VM leveraging Digital Ocean Cloud services, Droplets

3.3.1.3 The backend of the system shall be developed using the Python Django Web Framework

3.3.1.4 The frontend of the system shall be developed using the React Web Framework

3.3.2 Data In/Out:

3.3.2.1 The system shall accept incoming data including user credentials, target specifications, and user configurations

3.3.2.2 The system shall generate outgoing data in the form of test results, progress updates, and error messages

3.3.3 Services & Communications:

3.3.3.1 The system shall employ a Django RESTful API for seamless communication between the frontend and backend components

3.3.3.2 Comprehensive API documentation shall be made available in a separate document for reference and integration purposes

3.3.4 Shared Data Mechanism:

Given the constraints of web space, the system shall prioritize the optimization of data storage and retrieval processes

3.3.4.1 Database caching strategies shall be implemented to enhance efficiency in data storage and retrieval

3.3.4.2 The system shall integrate with the Burp Suite API to facilitate vulnerability and penetration testing services

3.4 Communications Interfaces

3.4.1 Functions:

3.4.1.1 The system shall support communication functions for email notifications, web browser alerts, and server-to-server communications

3.4.2 Message Formatting:

3.4.2.1 The system shall utilize JSON for data interchange between different components of the system

3.4.3 Communication Standards:

3.4.3.1 The system shall employ Secure FTP (SFTP) for secure file transfers

3.4.3.2 HTTPS shall be used for secure web communications, adhering to specific configurations provided by Sitegrounds

3.4.4 Security/Encryption:

3.4.4.1 All communications within the system shall be encrypted using TLS 1.3 or as supported by Sitegrounds to ensure data security and integrity.

3.4.5 Data Transfer Rates:

3.4.5.1 The system shall be optimized for broadband connections, with a minimum recommended speed of 5Mbps to facilitate efficient data transfer.

3.4.6 Synchronization:

3.4.6.1 The system shall employ WebSocket for real-time data synchronization between the client and server components, contingent on Sitegrounds' support for WebSocket technology

4. System Features

Section 4 is about the system feature of the product, describing what they do, their stimuli and requirements. The formatting for Section 4 is as follows: 4.1 is about automated penetration testing, and Section 4.2 is about user profile management.

4.1 System Feature: Automated Penetration Testing

4.1.1 Description and Priority

Allows for the penetration testing of websites. Priority: medium

This section uses a score from 1 to 10 where 1 is the worst and 10 is the highest
Benefits: 9 (Significant user value)
Penalty: 4 (Users may opt for competitors if not implemented well)
Cost: 7 (Complex to implement, but feasible)
Risk: 8 (Potential for misuse or false results)
Overall: 7

4.1.2 Stimulus/Response Sequences

1. User inputs target specifications.
2. System validates input and prompts confirmation.
3. User initiates the test.
4. System displays progress and provides real-time updates.
5. System presents results upon completion.

4.1.3 Functional Requirements

4.1.3.1 System shall allow users to input target specifications/target URL.

4.1.3.2 System shall validate the target specifications/target URL for correctness and safety.

4.1.3.3 System should provide real-time feedback on the penetration testing progress.

4.1.3.4 System must handle potential errors, such as unreachable targets, gracefully by notifying the user.

4.1.3.5 System shall ensure user authentication before initiating any penetration tests.

4.2 System Feature: User Profile Management

4.2.1 Description and Priority

Allows users to create, edit, and manage their profiles. Priority: Medium

Benefits: 7 (Enhances user experience)
Penalty: 5 (Users might be dissatisfied with static profile settings)
Cost: 3 (Standard feature in web applications)
Risk: 7 (Data privacy concerns)

4.2.2 Stimulus/Response Sequences

1. Users will log in or sign up.
2. System presents a dashboard or user profile page.
3. User edits profile information.
4. System validates and saves changes, then confirms with the user.

4.2.3 Functional Requirements

4.2.3.1 System shall provide user registration and login functionality.

4.2.3.2 System should allow users to edit their profile information, including name, email, and contact details.

4.2.3.3 System must securely store and encrypt user passwords.

4.2.3.4 System must allow password recovery through a secure method, such as email verification.

5. Other Nonfunctional Requirements

Section 5 is about covering all nonfunctional requirements that were not mentioned in earlier sections. The formatting of Section 5 is as follows: 5.1 is about requirements relating to performance, 5.2 is about requirements related to safety, 5.3 is about requirements related to security, 5.4 is about general software quality attributes, and 5.5 is about the business rules related to the product.

5.1 Performance Requirements

5.1.1 The system shall process user inputs within 2 seconds.

5.1.2 The system shall make regular, real-time progress updates with no lag exceeding 3 seconds.

5.1.3 The system shall support simultaneous penetration tests from 100 users without performance degradation.

5.1.4 The website shall load within 3 seconds.

5.2 Safety Requirements

5.2.1 The system shall strictly adhere to the target specified by the user, not performing penetration tests on websites or domains other than the one specified under any circumstances.

5.2.2 The system shall incorporate rate limiting to prevent accidental or intentional flooding of a target.

5.2.3 The system shall provide clear disclaimers and guidance to users on ethical use.

5.3 Security Requirements

5.3.1 The software shall encrypt all user data using industry-standard protocols for password hashing.

5.3.2 The software shall protect user sessions against session hijacking.

5.3.3 The software shall protect user sessions against cookie theft.

5.3.4 The software shall implement multi-factor authentication for enhanced user account security.

5.3.5 The software shall host all algorithms on a separate server, isolated from the primary website, with direct access to these models by external entities being strictly prohibited.

5.3.6 The software shall conduct regular security audits.

5.3.7 The software shall encrypt all data communication.

5.3.8 The software shall only allow registered and authorized users to view penetration test reports, with unauthorized access attempts being logged and reported.

5.3.9 The software shall have different levels of system functionalities and data should be accessible depending on the user role.

5.4 Software Quality Attributes

The following requirements are basic goals that the systems should attempt to follow, and should not be viewed as rigid requirements like other requirement sections. If any requirements in this section result in conflicts, 5.4.2 and 5.4.7 should be held in a higher priority than all other software quality attribute requirements.

5.4.1 The software should be able to accommodate changes in penetration testing methodologies or emerging threats with minimal modifications.

5.4.2 The system should always be accessible 99.9% of the time for users when they require penetration testing.

5.4.3 The penetration testing results shall have an accuracy rate of at least 95%.

5.4.4 The system shall be made so it is feasible to integrate new tools or functionalities without major architectural changes.

5.4.5 The system shall be able to seamlessly interact with common third-party platforms or services, if needed, such as vulnerability databases.

5.4.6 The system shall be web based, with any auxiliary tools or scripts working across different OS platforms.

5.4.7 The system shall not crash during a penetration test, handling errors gracefully.

5.4.8 The system shall be designed in a way such that they can be used in different contexts or projects.

5.4.9 The system shall handle unexpected inputs or situations without failing

5.4.10 The system shall have provisions to be tested easily, both for individual units and end-to-end functionality.

5.4.11 The user interface shall be intuitive with a preference towards ease of use, even if it comes with a slight learning curve.

5.5 Business Rules

5.5.1 Only admins or creators of the pentesting software shall be allowed to alter system mechanisms.

5.5.2 The only functionalities a normal user has are initiating tests and viewing initiated tests

6. Other Requirements

Section 6 is about covering all miscellaneous requirements not covered earlier. The formatting for Section 6 is as follows, 6.1 is about requirements relating to reuse objectives, 6.2 is about requirements related to the database, 6.3 is about internationalization requirements, 6.4 is about legal requirements, and 6.5 is about requirements related to accessibility.

6.1 Reuse Objectives

6.1.1 Components of the system, especially the machine learning models and data processing units, shall be modularly designed for potential reuse in other related projects.

6.1.2 RESTful APIs shall be made to be able to be used internally and potentially opened up for third-party integrations or other projects.

6.2 Database Requirements

6.2.1 The database shall be scalable to handle an increasing number of user records and penetration test results.

6.2.2 The database shall make regular backups, both incremental and full.

6.2.3 The database shall be scalable to handle an increasing number of user records and penetration test results.

6.2.4 The database shall have strict role-based access to ensure that only authorized personnel can view or modify the database.

6.2.5 The system shall use a third-party vulnerability database to retrieve vulnerability metrics.

6.3 Internationalization Requirements

6.3.1 The system should support multiple languages, starting with English, Spanish, and French.

6.3.2 The system should be able to handle and convert multiple currencies if there is a payment system in place.

6.3.3 The system shall have user profiles and scheduling features with accurate user time zones

6.4 Legal Requirements

6.4.1 The system shall adhere to international data protection regulations like GDPR (General Data Protection Regulation) for European users and CCPA (California Consumer Privacy Act) for California residents.

6.4.2 The system shall have features to prevent misuse, with users needing necessary permissions to conduct penetration tests on the target.

6.4.3 The system shall properly license, and follow the terms and conditions of all third party tools, libraries and services.

6.5 Accessibility Requirements

6.5.1 The website shall be designed following W3C Web Content Accessibility Guidelines.

6.5.2 The website shall have adequate contrast ratios and readable fonts.

6.5.3 The website shall be navigable using just the keyboard.

7. Appendix A: Glossary

Penetration Testing (Pentest): The practice of testing a computer system, network, or web application to find vulnerabilities that attackers could exploit.

SRS: Software Requirements Specification. A document that describes the features, behaviors, and attributes of a software system.

TLS: Transport Layer Security. A protocol ensuring privacy and data security between two communicating applications.

RESTful API: Representational State Transfer. A set of rules that developers follow when they create their API, allowing for interaction between systems using HTTP.

8. Appendix B: Analysis Models

Stakeholders:

1. Users (Registered and Unregistered)
2. Developers and IT Administrators
3. External vulnerability databases
4. Payment gateway providers
5. Hosting service providers

System Boundaries:

1. The system interacts with external entities, including external vulnerability databases, payment gateways, and cloud storage services.
2. Users interact with the system via a web-based interface, which includes data entry screens and data display screens.
3. The system operates within a cloud-based environment, including cloud servers for hosting and data storage.

Non-Functional Requirements:

1. Security and compliance with legal constraints, including access control, encryption, and adherence to relevant laws and regulations.
2. Scalability to handle increased user load and data storage needs.
3. Availability to ensure that the system is accessible and operational.
4. Performance to provide efficient penetration testing and report generation.

5. Usability with a user-friendly web interface.
6. Reliability for accurate and consistent penetration testing results.
7. Maintainability for ongoing updates, patches, and system enhancements.

9. Appendix C: To Be Determined List

1. Database Selection: Specific database technology to be used hasn't been finalized.
2. Third-Party Integration: Deciding on whether to integrate third-party vulnerability databases directly.
3. Pricing Model: How users will be charged for the service is TBD.
4. Notification System: How and when users receive notifications about test results or system updates.
5. User Data Retention Policy: Duration and conditions under which user data and reports will be stored.

(These are placeholders and might change based on the actual pending decisions in a real-world project.)