**AutoPen**

**System Design Document**

**Version 6.1**

Calla Robison

Caleb Hall

Michael Allen

Myles Scott

Joshua Buscher

| **Version Number and Editor Name** | **Date Modified** |
| --- | --- |

| Version 1.0 / Joshua + Michael | 10/24/2023 |
| --- | --- |
| Version 2.1 / Joshua + Myles | 10/31/2023 |
| Version 2.2 / Caleb + Calla + Michael | 10/31/2023 |
| Version 3.1 /Michael + Caleb + Joshua + Calla + Myles | 11/21/2023 |
| Version 4.0 / Joshua + Myles | 2/2/2024 |
| Version 4.1 / Michael | 2/5/2024 |
| Version 5.0 / Joshua | 2/13/2024 |
| Version 5.1 / Joshua | 3/2/2024 |
| Version 5.1 / Calla + Myles + Michael + Joshua + Caleb | 3/3/2024 |
| Version 6.0 / Joshua | 4/2/2024 |
| Version 6.1 / Michael, Caleb | 4/19/2024 |

TABLE OF CONTENT

[1 INTRODUCTION 3](#_heading=)

[1.1 Purpose and Scope 3](#_heading=h.1fob9te)

[1.2 Project Executive Summary 3](#_heading=)

[1.3 Document Organization 6](#_heading=)

[1.4 Project References](#_heading=) 7

[1.5 Glossary 7](#_heading=)

[2 SYSTEM ARCHITECTURE 7](#_heading=)

[2.1 System Hardware Architecture 7](#_heading=)

[2.2 System Software Architecture 8](#_heading=h.gquo4jlvgfa9)

[2.3 Internal Communications Architecture 9](#_heading=h.s261j5gckqbb)

[3 HUMAN-MACHINE INTERFACE 10](#_heading=)

[3.1 Inputs 10](#_heading=)

[3.2 Outputs 18](#_heading=)

[4 DETAILED DESIGN 21](#_heading=)

[4.1 Software Detailed Design 21](#_heading=)

[4.2 Internal Communications Detailed Design 25](#_heading=)

[5 EXTERNAL INTERFACES 25](#_heading=)

[5.1 Interface Architecture 25](#_heading=)

[5.2 Interface Detailed Design 26](#_heading=)

[6 SYSTEM INTEGRITY CONTROLS 29](#_heading=)

[7 SYSTEM DISCLAIMERS 30](#_heading=h.18nmbql1n23q)

SYSTEM DESIGN DOCUMENT

# INTRODUCTION

Cybersecurity threats are evolving rapidly, requiring more agile and thorough security systems. AutoPen aims to revolutionize the field of penetration testing by leveraging Burp Suite to conduct automated, comprehensive penetration tests. The primary objective is to provide businesses and organizations with a quicker, cost-effective, and more thorough method of identifying and rectifying potential vulnerabilities in their networks and systems. AutoPen additionally aims to meet secondary goals. The first of these secondary goals is prioritizing the small and medium enterprises (SMEs) market. The second is to cover common vulnerability areas, such as OWASP Top 10. Thirdly, AutoPen aims to support both on-demand and scheduled autonomous scans, enabling users to initiate tests at their convenience while also setting up periodic assessments based on predetermined intervals. AutoPen’s final secondary goal is to adhere to established cybersecurity frameworks and regulations, ensuring that all operations align with industry standards and best practices.

## Purpose and Scope

This document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces for AutoPen.

## Project Executive Summary

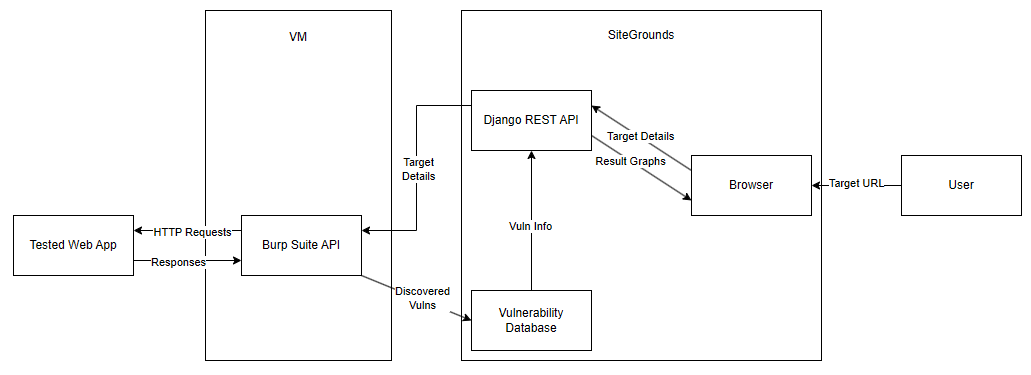
This section provides a managerial overview of AutoPen, specifically the system design.

### **System Overview**

With the increasing complexity and sophistication of cyber threats, organizations require comprehensive security measures to protect their assets and sensitive information. Traditional manual penetration testing methods can be time-consuming, costly, and prone to human error. AutoPen aims to streamline and optimize the penetration testing process, providing organizations with accurate and efficient assessments of their security posture.

The automated penetration test app leverages advanced technologies such as Burp Suite to simulate real-world cyber-attacks. By automating the testing process, our application can rapidly scan and analyze large volumes of data, identifying vulnerabilities, misconfigurations, and potential entry points that could be exploited by malicious actors.

The web application will be hosted on the cloud, allowing users to access the system. The web app will contain multiple pages, each with a different function such as configuring tests, launching tests, and viewing test results.



**Figure 1: System Diagram**

Figure 1 (shown above) is a basic system overview for Autopen. The virtual machine (VM) is responsible for hosting the tested application and the Burp Suite API that performs the testing. SiteGround is responsible for connecting the user to the AutoPen website and consists of the user’s browser, the Django REST API that prints the pentest results, and the vulnerability database that stores the found vulnerabilities for the REST API to print out.

The primary programming languages are divided by usage: client-side programming and server-side programming. The list below describes these two categories and is split into subsections. The specific subsections are Javascript with HTML and CSS using the React framework for client-side programming, and Python Django Framework for server-side programming.

* Client-side programming: The client-side programming handles presentation layers and user interaction. It is responsible for rendering web pages, handling user input, and making the user experience dynamic and responsive.
  + Javascript: This is used via the React framework to accept input and dynamically alter pages
  + HTML: This is used to render web pages and display content
  + CSS: This is used to set the visual style of pages
* Server-side programming: Server-side programming handles server logic, data processing, and database interactions. Server-side code is responsible for generating dynamic content, handling user authentication, and managing data.
  + Python: This is used with the Django framework to facilitate communication between the frontend, databases, and the Burp Suite application

### **Design Constraints**

To ensure the effectiveness and reliability of AutoPen, several design constraints need to be considered. These design constraints are separated into four distinct categories: Management Design Constraints, which address project timelines, the development team's expertise, and budgetary limits; Technical Design Constraints, which encompass limitations arising from the use of Burp Suite for vulnerability scanning and the chosen SiteGround Hosting plan; Legal Design Constraints, which include adherence to cyber law regulations such as the United States’ Computer Fraud and Abuse Act, the Electronic Communications Privacy Act, and the EU General Data Protection Regulation, as well as general liability and mandatory documentation for penetration testing; and Assumptions, where the legality of the project's scope and the sufficiency of hosting platform resources are presumed.

Management Design Constraints:

* Time Limitation: The project must be fully designed, developed, and tested within a single academic year. This period includes all stages of development, from initial planning to final presentation.
* Skill Constraints: The team consists of students with intermediate-level knowledge in software development and cybersecurity. Therefore, the project will avoid the development of custom penetration testing tools or highly complex algorithms that are beyond the team's current skill level.
* Financial Constraints: The project's budget is strictly limited to resources already provided by the academic institution or are accessible for a low fee. This includes software tools, development environments, and hosting platforms.

Technical Design Constraints:

* Burp Suite: The project will use the professional version of Burp Suite for web vulnerability scanning. Consequently, the capabilities of the project are confined to the features and limitations specified within this tier of the product.
* SiteGround Hosting: The project will be deployed on a basic tier of SiteGround Hosting, which provides essential services within the project's budget. The project will not utilize higher-tier services that incur additional costs, limiting the computing power of the assets.

Legal Design Constraints:

* United States Computer Fraud and Abuse Act: The project must follow the United States Computer Fraud and Abuse Act which makes unauthorized access to protected computers a crime.
* Electronic Communications Privacy Act: The project must follow the Electronic Communications Privacy Act which regulates the disclosure and interception of communications related to electronics.
* EU General Data Protection Regulation: The project must follow the EU’s General Data Protection Regulations which gives strict requirements for the protection and processing of personal data.
* General liability: The product must be liable for any damages caused by the penetration test.
* Documentation: The product must create detailed documentation that lists the scope of the pentest, the methods, and its results, the latter being needed to ensure proper procedures were met.

Assumptions Design Constraints:

* Legality: The team assumes that this project is legal to be made and used in its current scope.
* Hosting platform resources: The team assumes that the hosting platform dedicates enough resources to ensure the efficient operation of our programs

### **Future Contingencies**

The design and implementation of AutoPen comes with potential contingencies that may shift its developmental trajectory. Anticipating these challenges and devising alternate strategies ensures a smoother progression of the project. Five potential issues have been identified within AutoPen that need contingency plans. This section is formatted to first present the issue's name, then detail the specific scenario where the issue could arise, and finally, delineate the planned workaround for resolution.

Unintended Style Disruption:

* Scenario: A user mistakenly selects a penetration test style or configuration, causing disruption or breakdown of the target site.
* Possible Workaround: Implement a pre-test configuration validation mechanism. Restrict user choices to predefined, vetted penetration test configurations. Offer clear guidelines and warnings for configurations known to be aggressive.

Accidental Targeting of External Websites:

* Scenario: Due to misinterpretation of user input, AutoPen mistakenly tests a website other than the intended target, leading to possible ethical and legal concerns.
* Possible Workaround: Introduce a two-step validation procedure. Before starting a test, users are required to reenter the web address of the site they aim to test. AutoPen verifies this against the original target URL to ensure alignment.

Web Hosting Limitations:

* Scenario: The current web host, for reasons such as inadequate hosting capabilities, legal issues, or strategic decisions, becomes unsuitable for hosting AutoPen.
* Possible Workaround: Investigate alternative, more robust web hosting providers. Begin initial groundwork for transitioning to another provider to ensure minimal disruption. Concurrently, explore the feasibility of using a privately maintained physical server as an alternative or backup. Regularly evaluate hosting needs versus the host's capabilities to preemptively address limitations.

Transition to a Different Hosting Provider:

* Scenario: A strategic decision is made to migrate AutoPen to another hosting provider for enhanced performance, cost-efficiency, or other benefits.
* Possible Workaround: Ensure that the new hosting provider meets all the requirements of AutoPen. Begin the transition process during off-peak hours to minimize disruption. Implement a robust backup and migration strategy to prevent data loss.

Consideration of a Private Physical Server:

* Scenario: Due to growing needs or for enhanced control, there's a proposition to move AutoPen onto a private server managed by the development team.
* Possible Workaround: Conduct a thorough feasibility study weighing the pros and cons of maintaining a private server. Address challenges such as maintenance costs, security considerations, and scalability. If pursued, establish a transition roadmap and backup strategies.

## Document Organization

This design document offers an in-depth look into the architecture and blueprint of AutoPen. Beginning with the product's functionalities, it delves into its limitations, interactions, interfaces, hardware and software designs, and wraps up with security measures. As for what each specific section describes, Section 1 provides an introduction to the system, Section 2 provides information regarding the system architecture, Section 3 provides information regarding the human-machine interfaces, Section 4 provides info about detailed design, Section 5 provides information about external interfaces, and Section 6 provides information about the system’s integrity controls.

## Project References

The following are external resources used by AutoPen:

### Documentation for Burp Suite, the API used to run penetration tests, can be found at: https://portswigger.net/burp

### AutoPen Github can be found at: https://github.com/Caleb-Hall-1015/AutoPen

## Glossary

Penetration Testing (Pentest): The practice of simulating an attacker and testing a computer system, network, or web application to find vulnerabilities.

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

TLS: Transport Layer Security. A protocol for 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.

OWASP TOP 10: A widely recognized standard awareness document developed for the cybersecurity industry. It is primarily written for developers and web application security professionals. The standard represents a broad consensus regarding the most critical security risks currently facing web applications.

# SYSTEM ARCHITECTURE

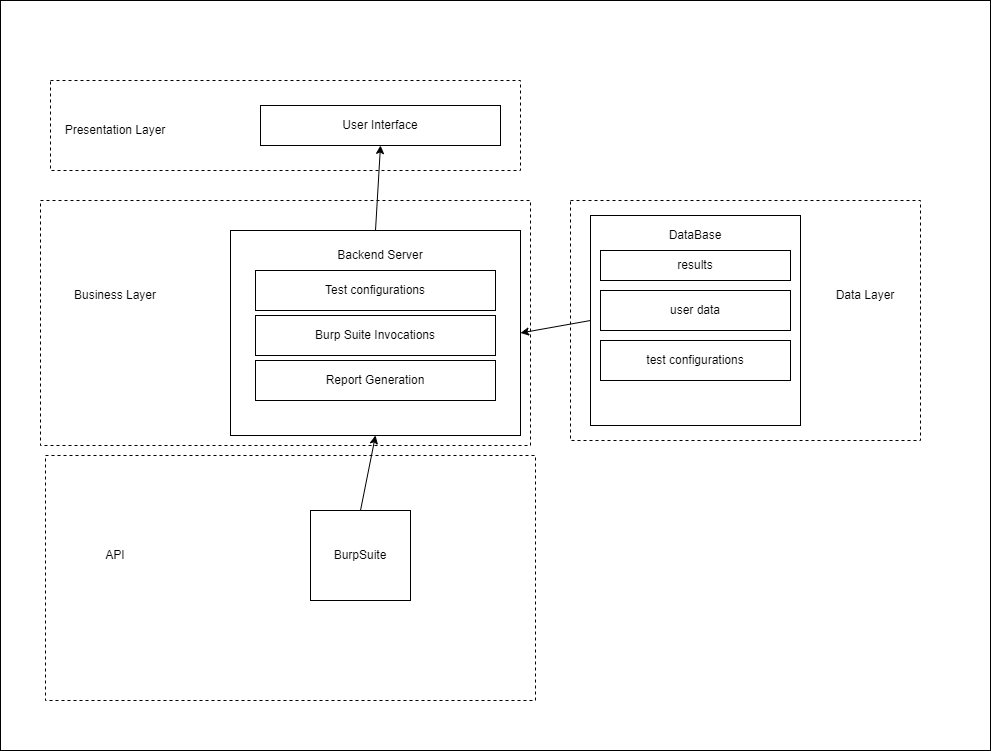
Section 2 concerns the general architecture of AutoPen. The formatting of Section 2 is as follows: Section 2.1 is about the hardware architecture of the system, Section 2.2 covers the software architecture of the system, and Section 2.3 details the internal communications architecture of the system.

## System Hardware Architecture

AutoPen is designed to operate entirely within the cloud, with all assets securely stored and managed in cloud-based servers. Unlike traditional hardware architectures, AutoPen does not require any physical infrastructure to function effectively. This cloud-based approach offers several advantages, including increased flexibility, scalability, and accessibility.

## System Software Architecture

The Burp Suite API-powered Penetration Testing system is structured with a four-tier architecture: Presentation Layer (Web UI), Business Logic Layer (Backend Server), Data Layer (Database), and API (Burp Suite vulnerability analysis tool) as shown in Figure 2. User interactions occur within the Presentation Layer, where functionalities such as registration, login, test initiation, and results viewing are facilitated. This layer interfaces directly with the Business Logic Layer, which handles critical operations including Burp Suite invocations, test configurations, and report generation. The Data Layer, constituting the database, manages the storage and retrieval of results, user data, and test configurations.



**Figure 2: Internal Communication Flow**

Figure 2 (shown above) describes the architecture of the AutoPen system and the layers that it uses. The presentation layer is the topmost layer that users interact with, which connects to the business layer, which is powered by the backend server. The business layer uses information from the API and Data Layer, to send results to the user.

### **Software Modules**

The software modules for the product include the User Management Module, the Testing Initialization Module, the Reporting Module, and the Notification Module. The User Management module is in charge of handling user registration, login, and profile management. The test initialization module handles initiating and configuring penetration tests. The reporting module handles the generation and display of test results to the user. The notification module helps to alert users about system updates and test completions.

### **Coding languages**

The coding languages can be split into two sections, frontend and backend languages. Frontend languages are utilized to develop the User Interface within the Presentation Layer, enabling functionality for components directly interacted with by users. The frontend languages for AutoPen include HTML, CSS, and JavaScript, specifically React.js. Backend languages refer to languages that function behind the scenes to perform tasks outside the view of users, like cloud services. The backend languages for the product are Python 3.11 and Python Django.

### **Tools**

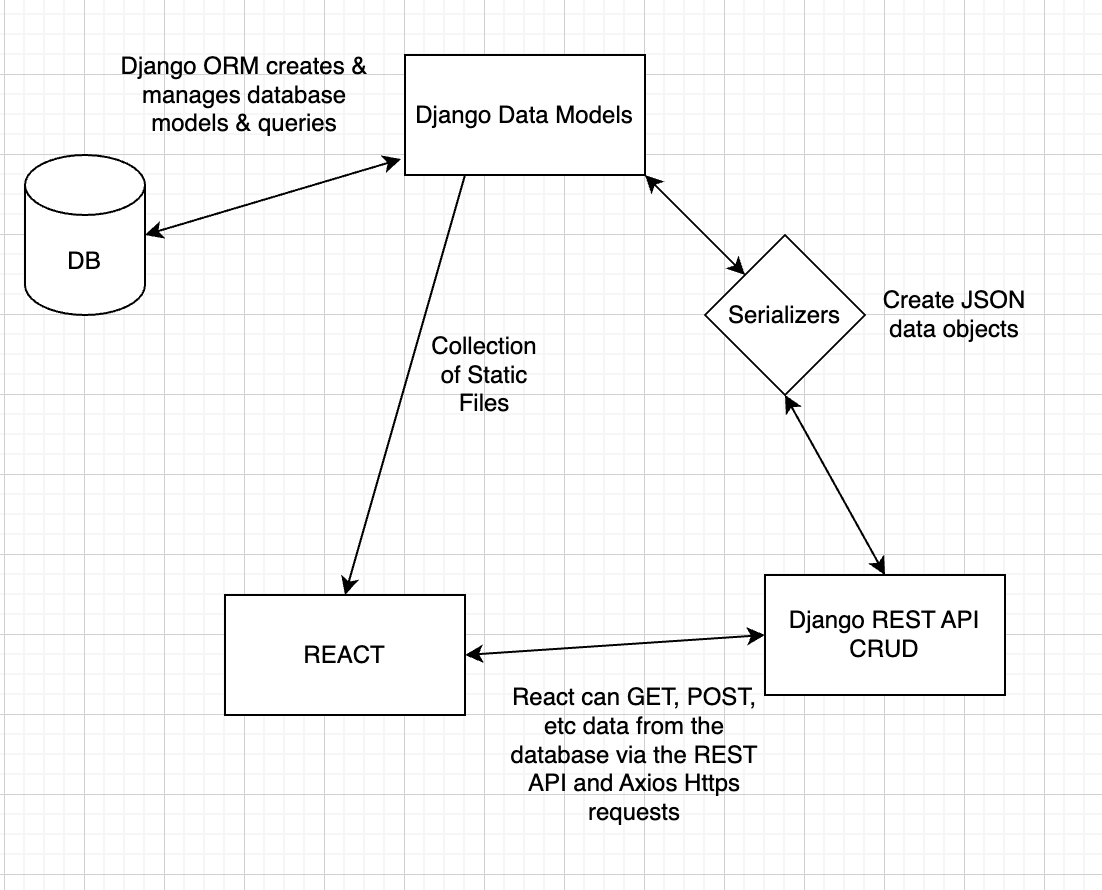
AutoPen employs a suite of tools to support its development and operation, including Git, Docker, SiteGround IaaS, GitHub Actions, Kali Linux, and VMWare. Git is used for version control, managing changes to the project codebase. Docker aids in containerizing application components and helps keep consistent deployment and testing across environments. SiteGround IaaS hosts the backend and facilitates the integration of the Burp Suite penetration testing API into the web application, while also managing cloud storage. GitHub Actions enhances the project by automating workflows for continuous integration and deployment. Kali Linux is used to host the Burp Suite API. VMWare is used for the management of the Kali Linux virtual machine.

## Internal Communications Architecture

The AutoPen system primarily relies on a web-based communication system, utilizing a combination of HTTP/HTTPS requests for standard interactions and WebSockets for real-time updates. In addition, when a user accesses the web application, and their browser initiate a penetration test, where real-time feedback is essential, a WebSocket connection is established. This ensures users receive immediate updates about the test's progress. The backend, in turn, communicates with the database to store or fetch data as required. The below list contains all communication architectures. Figure 3 is a communication architecture diagram outlining the relationships of the list of communication architectures.

Communication Architectures:

* HTTP/HTTPS: Used for standard request-response interactions between the frontend and backend.
* WebSocket: A protocol providing full-duplex communication channels over a single TCP connection, facilitating real-time updates.



**Figure 3: Internal Communication Flow**

Figure 3, shown above, shows how Django Data Models interact with the Database that stores queries, Serializers that Create JSON data objects, and REACT that stores static files. REACT and Serializers also interact with Django REST API CRUD, which obtains data from databases.

# HUMAN-MACHINE INTERFACE

Section 3 describes the inputs users give to the product and the outputs the users receive from the product. The formatting of Section 3 is as follows: Section 3.1 describes the inputs of the product while Section 3.2 describes the outputs.

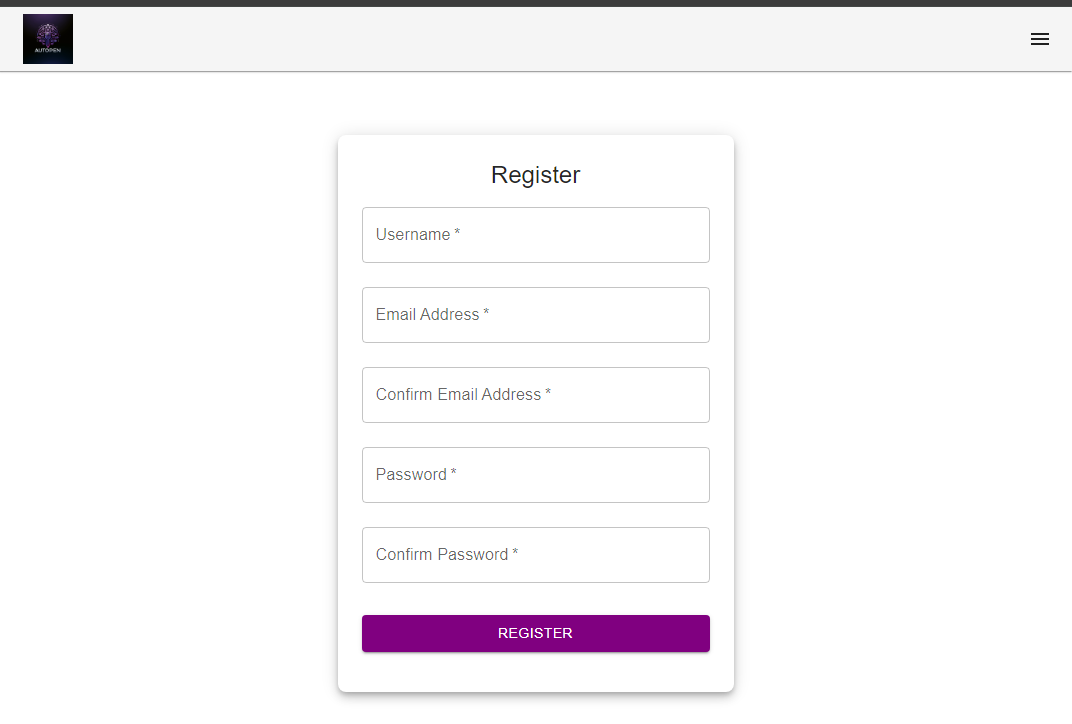
## Inputs

In the AutoPen system, user interaction is facilitated through four key data entry screens. Section 3.1 delves into these interfaces, mapping out the user's journey from initiating contact with the software to executing commands. Subsection 3.1.1 details the interaction points: 3.1.1.1 User Registration Screen for onboarding new users; 3.1.1.2 Login Screen for user authentication; 3.1.1.4 Launch/Progress Screen for configuring tests; and 3.1.1.5 Results Screen for displaying scan outcomes. Additionally, Subsection 3.1.2 addresses the system's feedback mechanisms through miscellaneous messages, and Subsection 3.1.3 defines transaction codes that streamline the operational flow within AutoPen.

### **Data Entry Screens:**

### **User Registration Screen:**

* Data Elements: Username, Email, Password, Confirm Password, optional user profile details like contact number
* Mandatory fields: Username, Email, Password.
* Constraints:
  + Passwords should match the Confirm Password field and have a minimum of 8 characters.
  + Emails must adhere to standard email formats.
* Controls: Users are barred from proceeding without completing mandatory fields. Passwords undergo strength validation.
* Access Restrictions: Exclusively accessible by unregistered users.

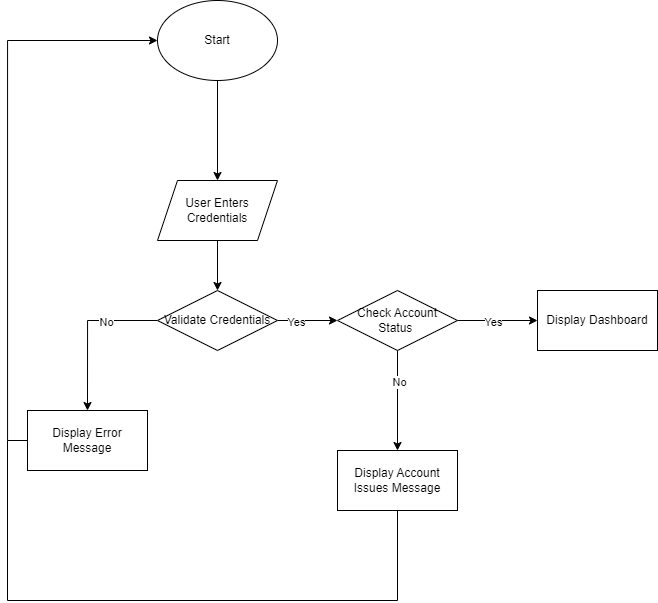


**Figure 4: User Registration Page**

Figure 4 depicts a screenshot of the user interface for AutoPen’s 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 a username, and an email address, with a note indicating that a valid email address is required, a field for confirming their address again, a password, and a password confirmation field. The layout is simple and user-centric, designed to facilitate an easy and secure registration process.

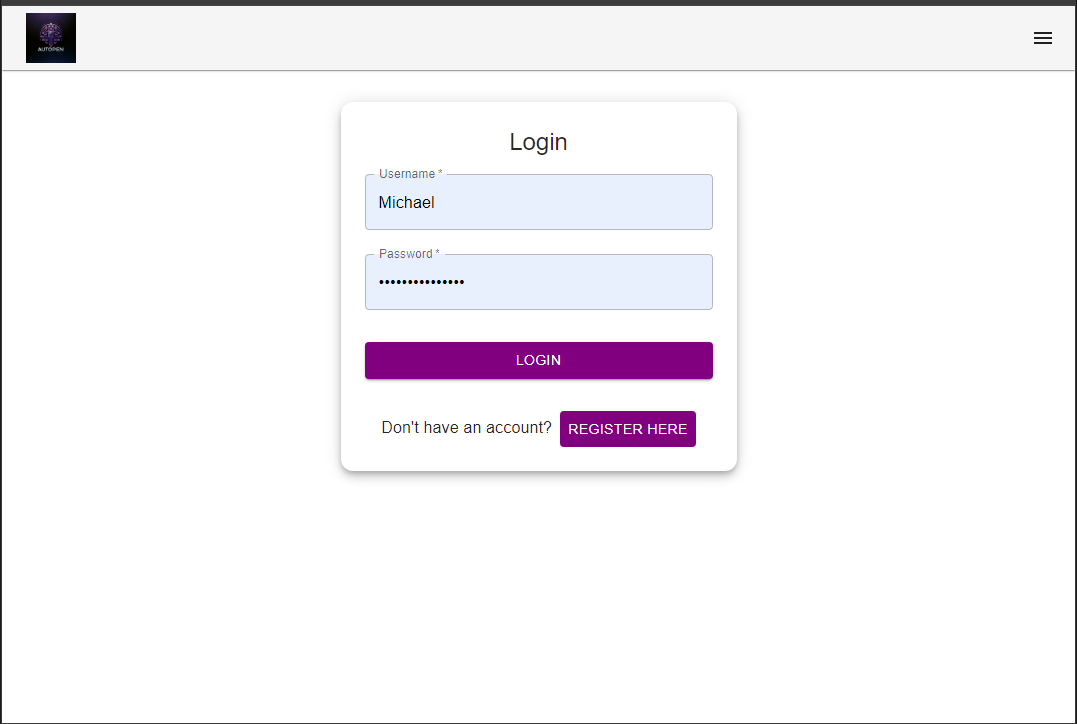
### **Login Screen:**

* Data Elements: Username/Email, Password.
* Mandatory fields: Username/Email and Password.
* Constraints: At this time, there are no constraints for this screen.
* Controls: Users are barred from proceeding without correct credentials.
* Access Restrictions: Exclusively accessible by unregistered users or logged-out users.



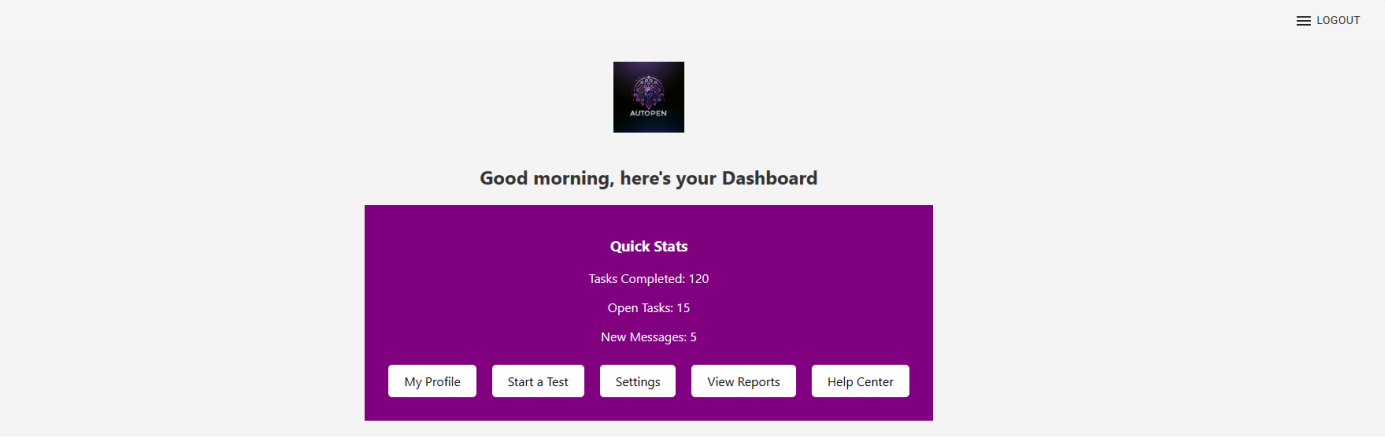
**Figure 5: Login Workflow**

Figure 5 illustrates the login workflow for a system or application. The process begins at 'Start', leading to a 'Login Page' where the user is prompted to enter their username or email, followed by their password. After submitting the credentials, the system checks for their correctness. If the credentials are incorrect, the workflow directs to 'Login Unsuccessful' where an error message is displayed and the user remains on the login screen. If the user has forgotten their password, there is a branch from the 'Enter Username/Email' step to a 'Forgot Password' option, which leads to a 'Redirect to Password Reset Page'. If the credentials are correct, the user is directed to 'Login Successful' and then redirected to the dashboard of the application.

****

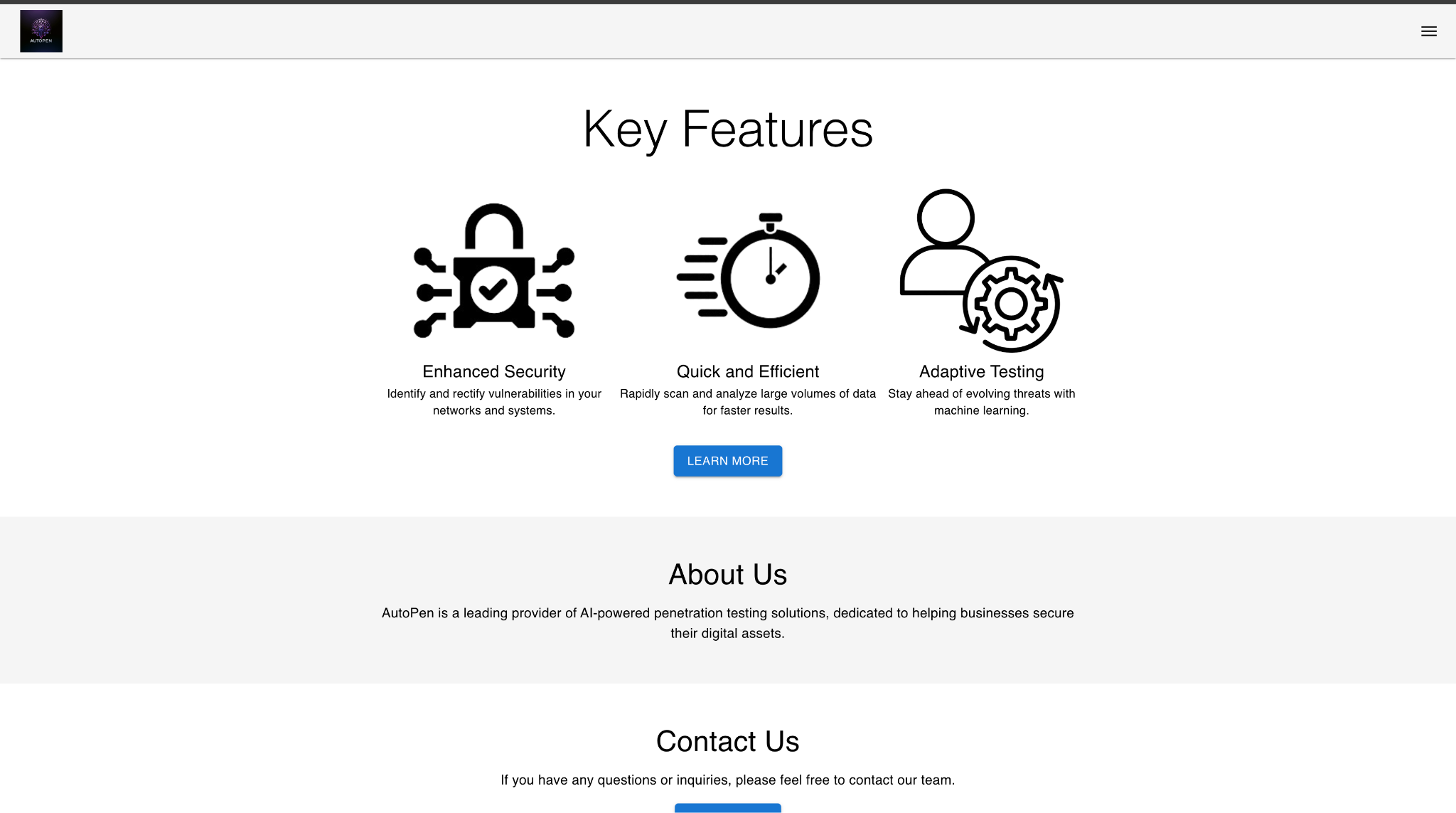
**Figure 6: Login Page**

Figure 6 showcases a user interface for the login page. It features two input fields, one for the 'Username' and another for the 'Password'. Below the input fields is a 'Login' button to submit the credentials and a ‘Register’ button for new users.



**Figure 7: Dashboard Page**

Figure 7 displays a screenshot of the user dashboard page. The page features five buttons, including: ‘My Profile’ for accessing a user's personal information, ‘Start Scan’ for navigating to the ‘Launch Scan Page’, ‘Settings’ for modifying user preference settings such as light/darkmode, ‘View Reports’ for viewing past scan reports, and ‘Help Center’ which navigates to the site’s FAQ page. The dashboard also contains information concerning the number of scans completed to-date as ‘Tasks Completed’, tasks still in progress as ‘Open Tasks’, and unseen reports as ‘New Messages’.

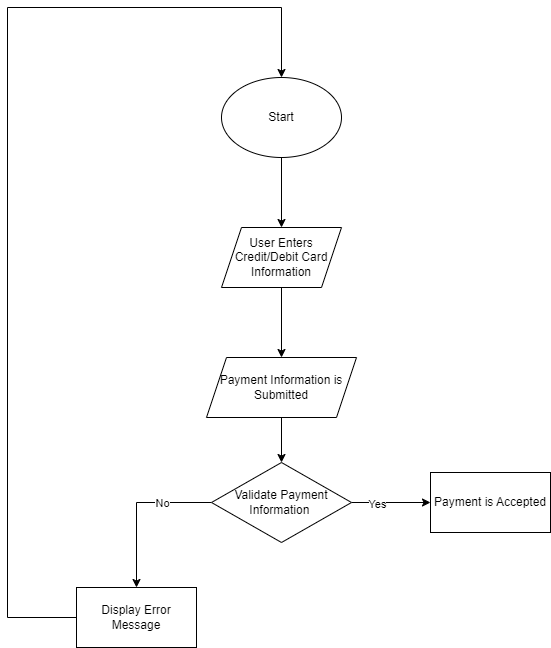
****

**Figure 8: Home Page/Landing Page**

Figure 7 displays a screenshot of the home page from a website, which is designed to provide a clean and informative user experience. The top portion of the page includes a navigation bar with a drop-down with links to the 'Register', 'Login', 'About us', 'Contact Us', 'Pricing', and 'FAQ' sections. The main content area highlights the 'Key Features' of the service, which are 'Enhanced Security', 'Quick and Efficient', and 'Adaptive Testing', each accompanied by an icon and a brief description. A 'Learn More' button is present under the features for further user engagement. The lower part of the page includes sections for 'About Us', giving a brief introduction to the company, and 'Contact Us', inviting user inquiries with a button for contact initiation. The overall design employs a clean, modern aesthetic with a focus on usability and easy access to information.

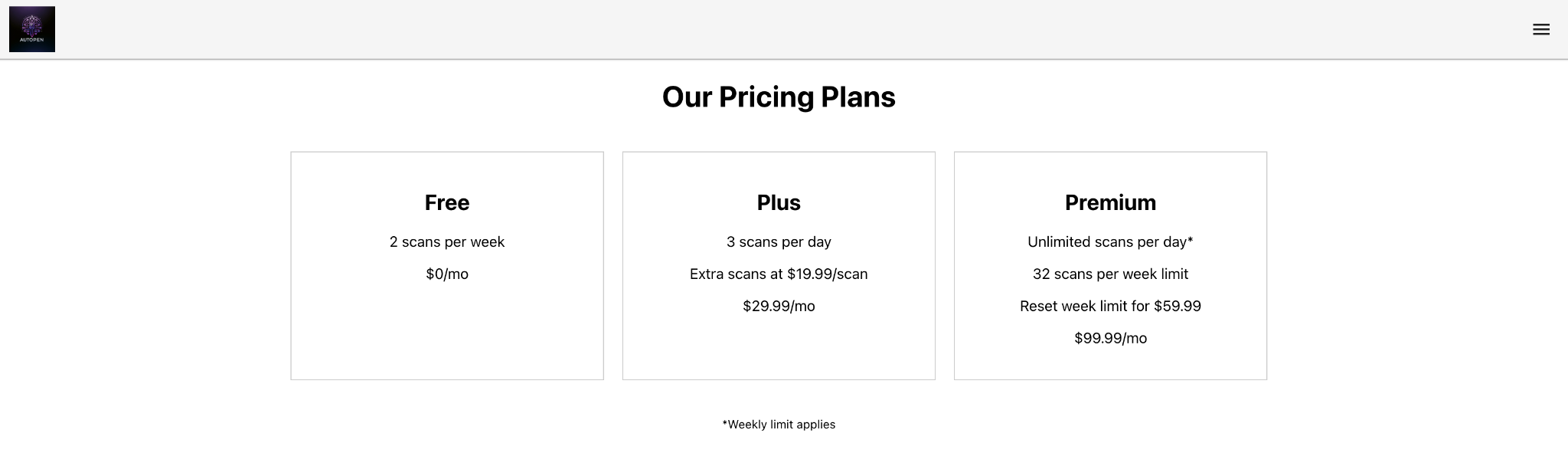
### **Payment Screen:**

* Data Elements: Credit/Debit Card Number, CVV, Expiry Date, Billing Address.
* Mandatory Fields: Credit/Debit Card Number, Card Expiration Date, Billing address
* Constraints:
  + Credit/Debit Card Number should adhere to card standards.
  + CVV should be 3 or 4 digits depending on the credit card.
* Controls: Payment cannot be processed without valid payment details.
* Access Restrictions: Only registered users access premium features or services.



**Figure 9: Payment Workflow**

Figure 8 depicts a payment workflow diagram, beginning with the 'Start' node and leading to a 'Payment Screen'. The process involves sequential steps where the user is required to enter their credit/debit card number, expiration date, billing address, and CVV. After these details are submitted, the system evaluates whether all payment information is valid. If any information is incorrect or incomplete, the workflow progresses to 'Display Error Message', prompting the user to correct the input. Conversely, if all the information is correct, the payment is accepted, completing the transaction process.

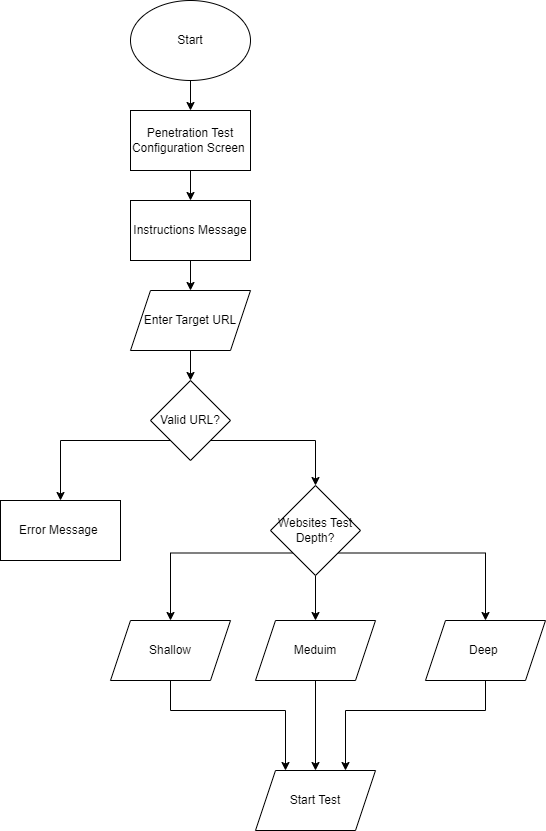
****

**Figure 10: Pricing Page**

Figure 9 showcases a screenshot of a pricing page from a website. The top of the page features a navigation bar with links to various sections of the site, including 'Home', 'About Us', 'Contact Us', 'Pricing', and 'FAQ/Help'. Below the navigation bar, the main heading 'Our Pricing Plans' introduces three distinct service packages: 'Free', 'Plus', and 'Premium'. Each plan is presented in its own card, detailing the specifics of the offer, such as the number of scans per week or day, and the price. The 'Free' plan offers 2 scans per week, the 'Plus' plan offers 5 scans per day with extra scans at $1.99 each, and the 'Premium' plan offers unlimited scans per day with a note about extra scans per week limit and the total cost. The layout is designed for clear comparison between the plans, facilitating an informed choice for the website's users.

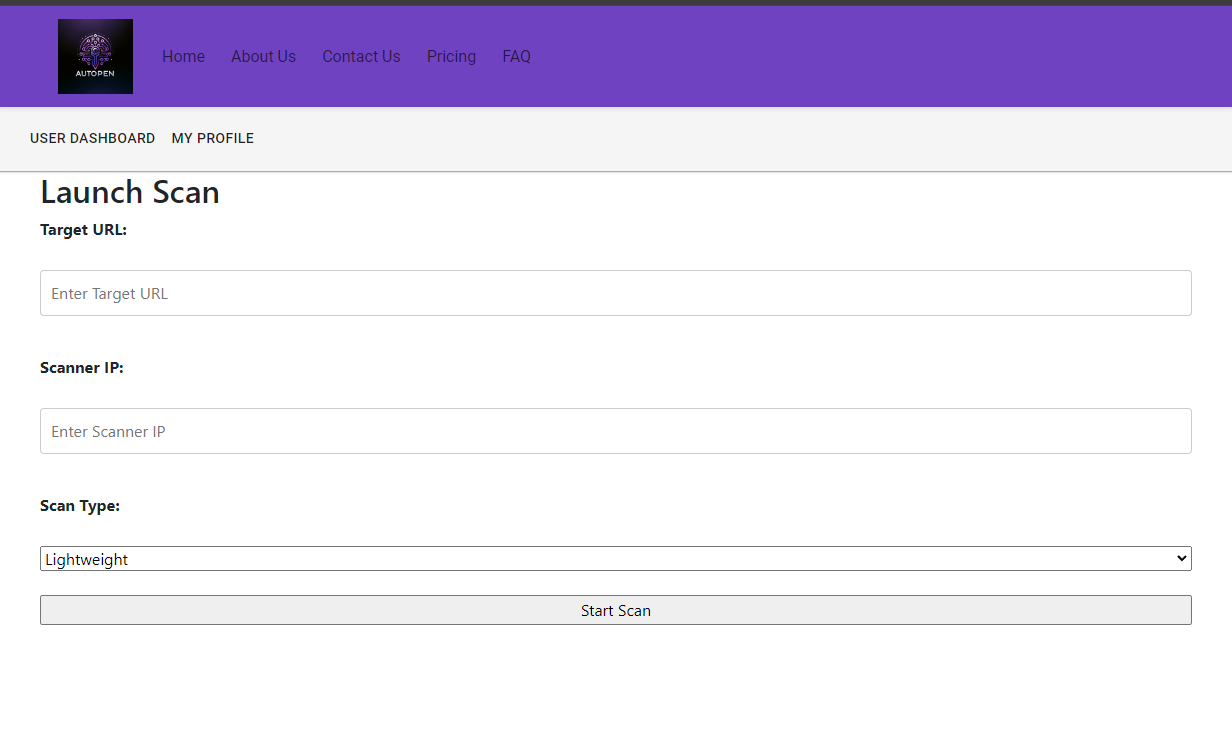
### **Launch/Progress Screen:**

* Data Elements: Target URL, Test depth (choices: shallow, medium, deep), Optional notes or instructions.
* Mandatory field: Target URL.
* The target URL must adhere to standard URL formats.
* Test depth is selectable via a dropdown menu.
* Controls: Tests cannot be started without a valid target URL.
* Access Restrictions: Accessible only to registered and authenticated users.



**Figure 11: Penetration Configuration Workflow**

Figure 9 displays a flowchart that details the penetration test configuration workflow. The sequence begins at 'Start', leading to a 'Penetration Test Configuration Screen' where users receive an 'Instructions Message'. The user is then prompted to 'Enter Target URL'. A decision point follows, checking if the entered URL is valid. If not, an 'Error Message' is displayed. If the URL is valid, the workflow proceeds to a decision point where the user must select the depth of the website test, with options for 'Shallow', 'Medium', or 'Deep' penetration testing. Once the depth is selected, the workflow culminates with 'Start Test', initiating the penetration test process.

****

**Figure 12: Launch Scan Page**

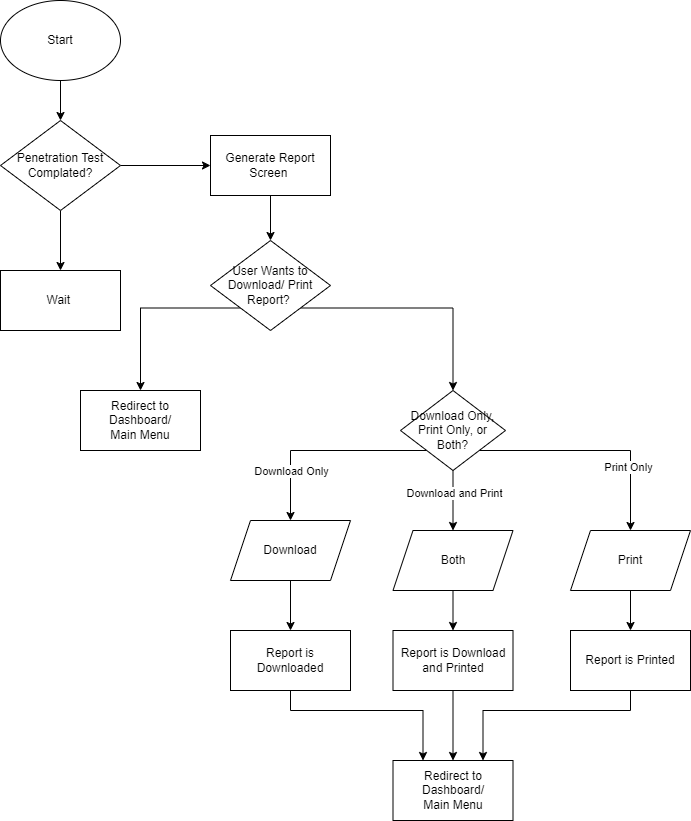
The 'Launch Scan' interface, depicted in Figure 11, provides a user-friendly layout for initiating network scans. At the top, the title 'Launch Scan' sets the context for the user's action. Below this, two input fields request the 'Target IP' and 'Scanner IP' addresses. Additionally, a dropdown menu for 'Scan Type' gives users the option to select the intensity of the scan, with 'Lightweight' being the default selection. Users simply click the 'Start Scan' button after entering the required information to begin the scanning process. The design of this page aims to streamline the scanning operation, minimizing complexity for the user.

## Outputs

This section presents the outputs generated by the AutoPen system, encompassing the data displays and reports that users receive post-interaction. The section is methodically sectioned into 3.2.1 Test Result Screen, where users review the detected vulnerabilities and suggested actions; 3.2.2 User Profile Screen, a centralized space for personal data and test history; and 3.2.3 System Notification Screen, the system's alert interface. Each screen is constructed with identification codes serving as unique screen identifiers, detailed contents, defined purposes, and access restrictions.

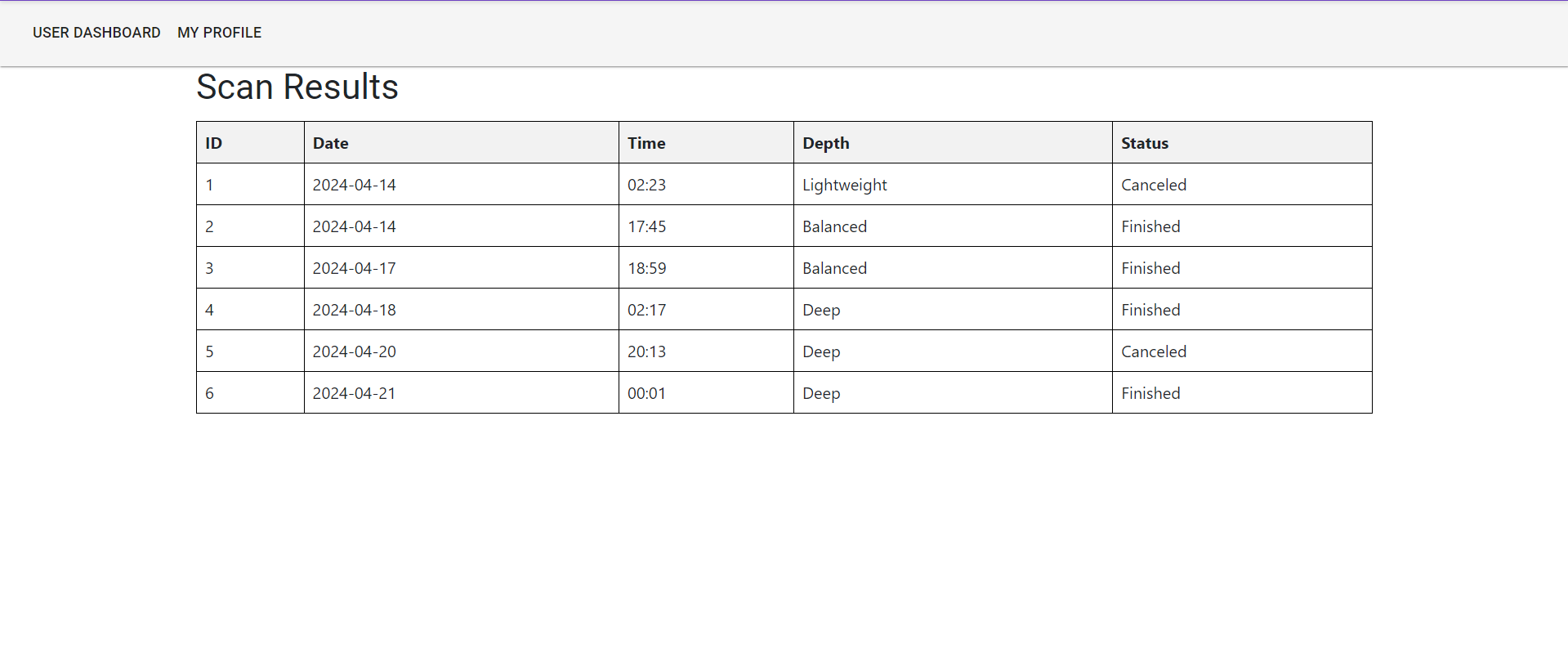
### **Basic Result Screen:**

* Data Elements: Table with columns: ID, Date, Time, Depth, and Status.
* Constraints: No constraints at this time.
* Purpose: List the completed scans.
* Access Restrictions: Available post-completion of a penetration test for the user who initiated the test.



**Figure 13: Generate Report Flow**

In the sequence outlined in Figure 12, the user's journey through the report generation phase post-penetration testing is charted. The workflow triggers an inquiry into the test's finalization. A holding pattern ensues until the test's end. Subsequently, the 'Generate Report Screen' poses a query regarding the user's preference for report acquisition—be it download, print, or a combination of both. The pathway diverges based on this preference: downloading, printing, or executing both operations. Completion of this action circulates the user back to the central hub of the 'Dashboard/Main Menu'. The workflow encapsulated in Figure 11 is instrumental in navigating the report generation process

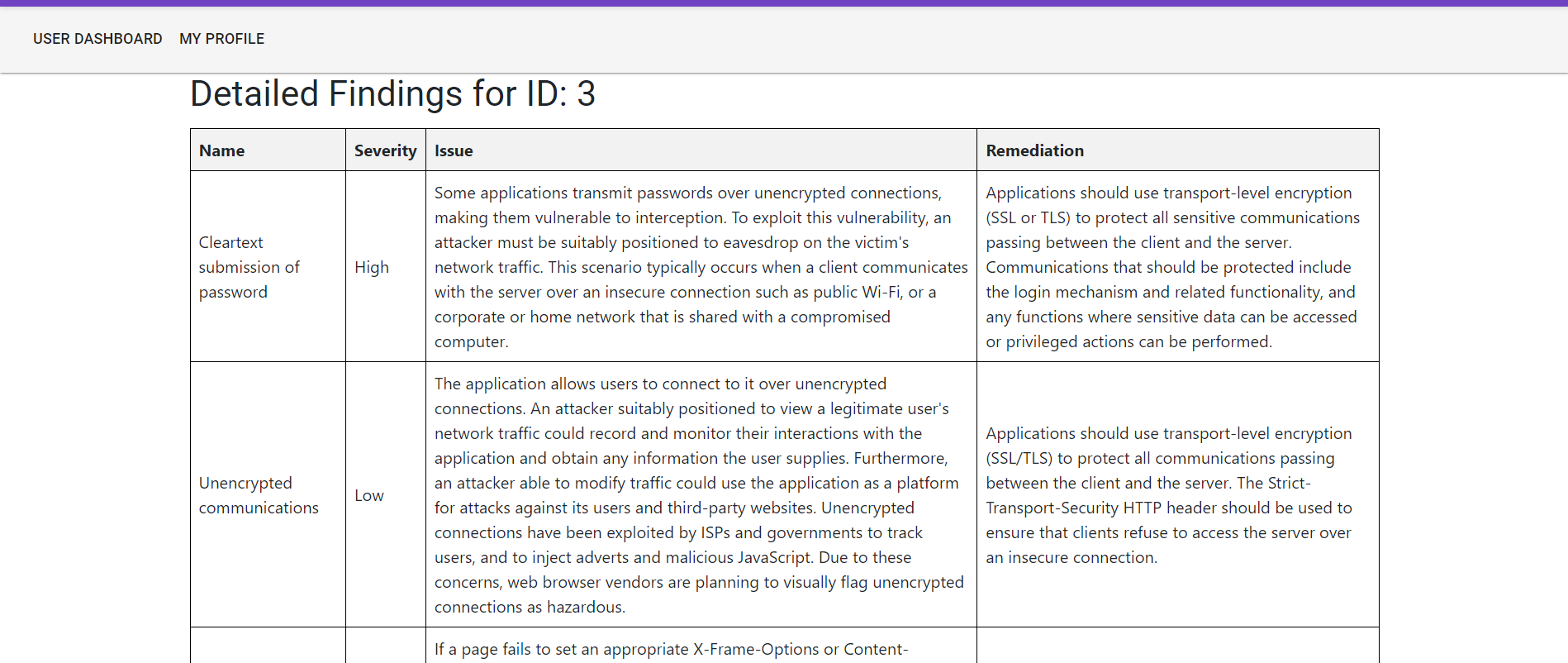
****

**Figure 14: Basic Result Page**

Featured in Figure 13 is an interface design of a 'Results Page' from a cybersecurity application. The layout is segmented into two primary headings: 'Results' crowning the page and 'Scan History' positioned just below. A succinct instruction, 'Select a previous scan to show results', invites users to interact with the system to retrieve and display the results of prior scans. The interface's streamlined design emphasizes functionality and ease of use so that historical scan data is readily accessible to the user.

### **Detailed Result Screen:**

* Data Elements: Table with columns detailing vulnerabilities detected, severity levels, vulnerability description, and potential fixes.
* Constraints: No constraints at this time.
* Purpose: To provide users with insights into their system's vulnerabilities and guide them on mitigation.
* Access Restrictions: Only the user who initiated the test can view its results.



**Figure 15: Detailed Result Page**

Featured in Figure 13 is an interface design of the 'Detailed Result Page' giving a more detailed report on vulnerabilities for a specific scan. The layout is segmented into four primary columns: 'Name' designates the vulnerability by name, ‘Severity’ designates the vulnerability by threat level, ‘Issue’ describes the vulnerability, and ‘Remediation’ suggests possible resolutions to fix the vulnerability.

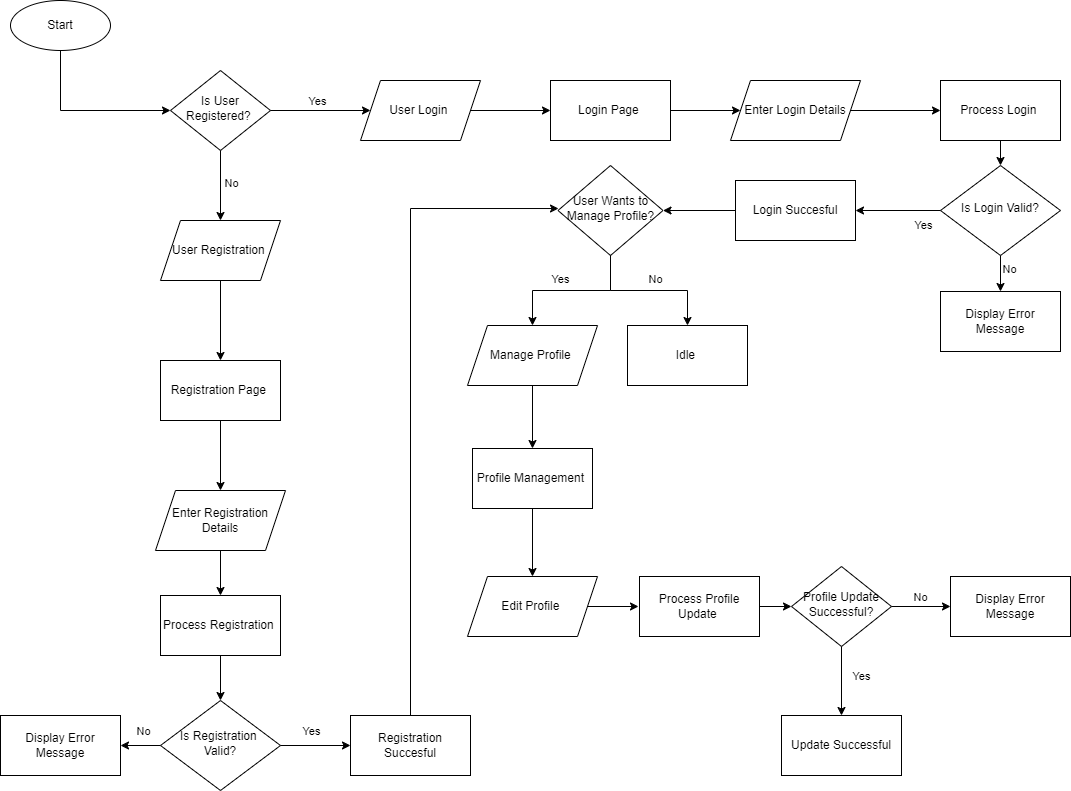
# DETAILED DESIGN

This section outlines the detailed design of the AutoPen product, focusing on the complexities of both its hardware and software components. Section 4.1 Software Detailed Design provides an in-depth examination of the software's architecture, broken down into three key modules: Section 4.1.1 User Management Module, which handles user interactions such as registration and authentication; Section 4.1.2 Burp Suite Penetration Testing Module, which focuses on security testing using Burp Suite; and Section 4.1.3 Database Module, which manages all data storage and retrieval operations for system integrity and security. Section 4.2 Internal Communications Detailed Design discusses the internal communication strategies necessary for effective data exchange and system operations within AutoPen.

## Software Detailed Design

AutoPen is structured around multiple software modules that collectively facilitate its primary function: automated penetration testing via a web-based platform. Each module focuses on distinct functionalities such as registering/logging in to a user profile, viewing a user dashboard, viewing informational pages (About us, FAQ, Pricing), running a pentest on a user’s web application, displaying a risk analysis report via graphs and charts to the user, viewing previously run tests, and customizing user preferences and settings.

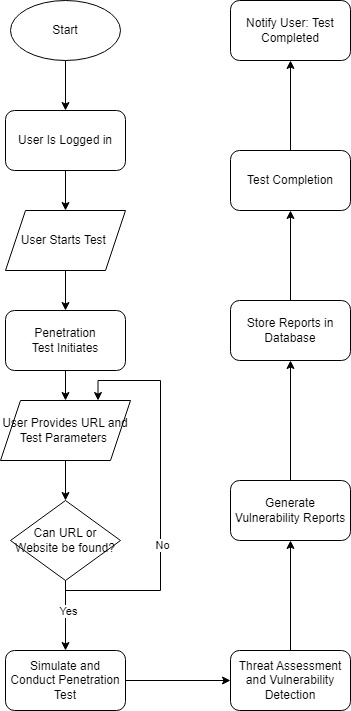
**4.1.1 User Management Module:** This module is important in managing user interactions with the system, encompassing user registration, authentication, and session management. It interfaces directly with the Database Module to handle data such as usernames, and passwords hashed and salted using Bcrypt encryption, email addresses, and user profile details.

****

**Figure 16: User Management Module Flowchart**

In Figure 14, the depicted flowchart breaks down the registration and login procedures into a series of decision-based steps. The process starts by verifying the user's registration status. If unregistered, the user is prompted to complete registration details on the appropriate page and proceed to the registration validation phase. Any discrepancies during validation prompt an error message, whereas correct details affirm registration completion. For registered users, the flow progresses to login, where credentials are verified. An invalid logon triggers an error, while a valid login allows for profile management. Within this section, users can opt to edit their profile, with successful edits acknowledged accordingly and errors flagged for user attention.

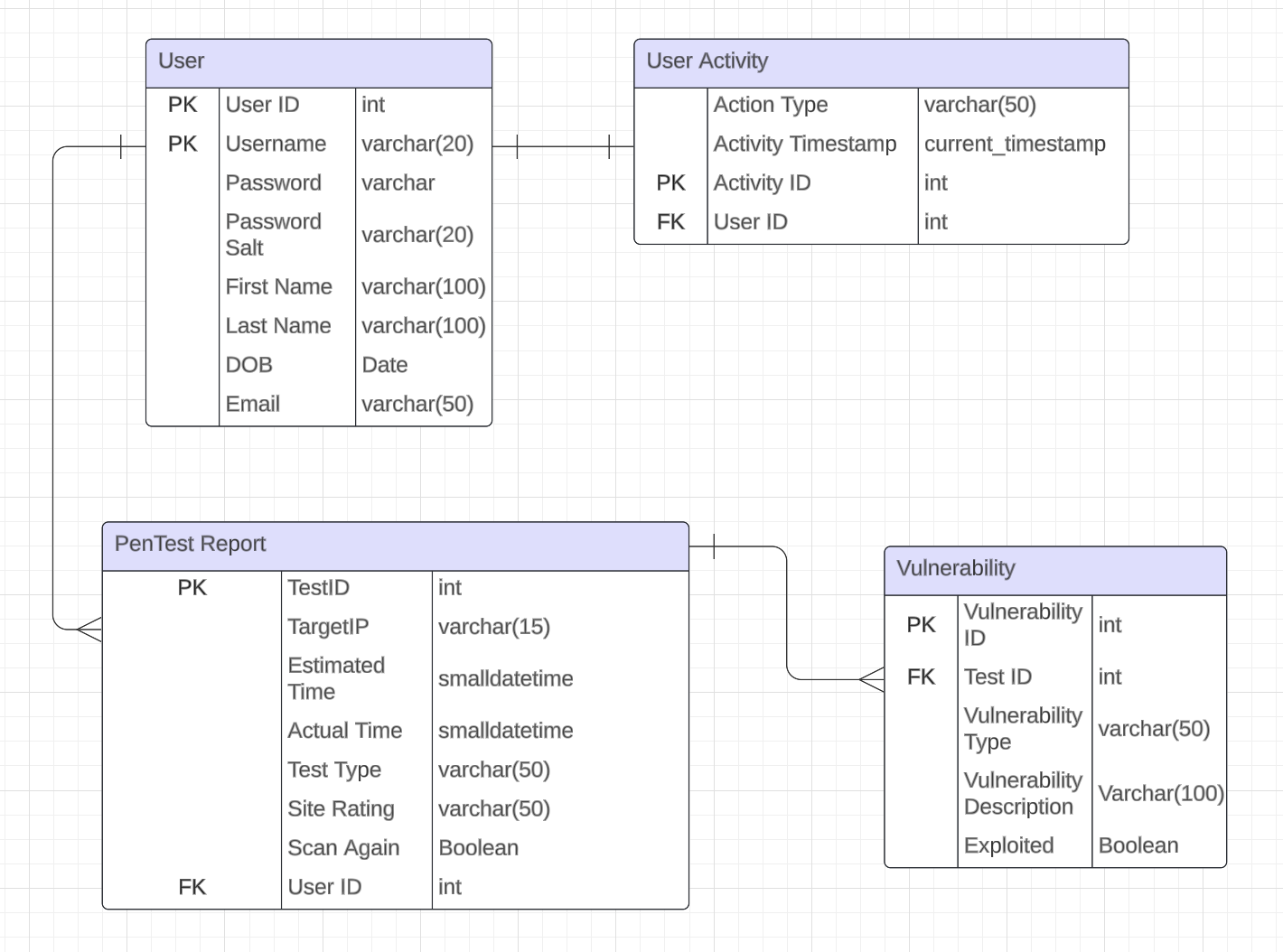
**4.1.2 Burp Suite Penetration Testing Module:** This module simulates cybersecurity attacks using the Burp Suite tool to identify vulnerabilities. It stores configurations and results in the Database Module, handling data elements like target URLs, test parameters, and reports on detected vulnerabilities.



**Figure 17: Burp Suite Penetration Testing Module Flowchart**

Figure 15 illustrates the procedural steps within the Burp Suite Penetration Testing Module via a flowchart. The process commences at the 'Start' node, advancing only after confirming that the user is logged in. The subsequent stage involves the user initiating the test, which then prompts the entry of the URL and test parameters. A negative result in the URL validation halts the process, while a positive result leads to the simulation and execution of the penetration test. Following the test phase, the module performs a threat assessment and detects vulnerabilities, which form the basis of the vulnerability reports. These reports are archived in a database. The completion of the test triggers a notification to the user, marking the end of the process.

**4.1.3 Database Module:** Serving as the central repository for all data, this module ensures the integrity, security, and availability of data across the system. It supports all other modules by providing necessary data management functions, including storage and retrieval of user data, test configurations, and system logs. The relational database schema illustrated below in Figure 16 shows how data is interconnected across tables, defining relationships and dependencies that are crucial for efficient data management.



**Figure 18: Web Database Schema**

Figure 16 provides an overview of a Web Database Schema, highlighting the relational structure between 'User', 'User Activity', 'PenTest Report', and 'Vulnerability' tables. Each table is defined by a set of attributes, with 'User ID' serving as a primary key (PK) in the 'User' table and as a foreign key (FK) in both the 'User Activity' and 'PenTest Report' tables, indicating relational links. 'PenTest Report' and 'Vulnerability' tables also share a relationship via the 'Test ID'. The schema illustrates fields such as usernames, passwords, personal information, test results, and vulnerability details, along with data types and constraints. This layout is essential for understanding the data management within the penetration testing framework.

## Internal Communications Detailed Design

AutoPen’s internal communications framework is designed for secure interactions within the system, particularly focusing on the elements of backend-server to database interactions, real-time data exchange, and the handling of both synchronous and asynchronous tasks. This section delineates the precise configurations and mechanisms that define the communication protocols, data handling methods, and network structures. Detailed descriptions and specifications of these configurations offer help in understanding how AutoPen maintains high levels of data integrity and operational responsiveness.

**4.2.1 Servers and Clients:** AutoPen utilizes a cloud-based architecture that dynamically adjusts the allocation of servers based on real-time user demand. Each interaction on the platform is managed as a separate client session, allowing for scalability and responsive service delivery. This design supports a high volume of simultaneous users without degradation of performance, leveraging cloud capabilities to scale resources up or down as needed.

**4.2.2 Data Exchange Formats:** For API interactions, AutoPen predominantly uses JSON (JavaScript Object Notation), chosen for its efficiency in parsing and lightweight data structure which reduces the overhead on network and processing power. This format facilitates structured and straightforward data transfers, enabling seamless integration and interoperability between different system components and external interfaces.

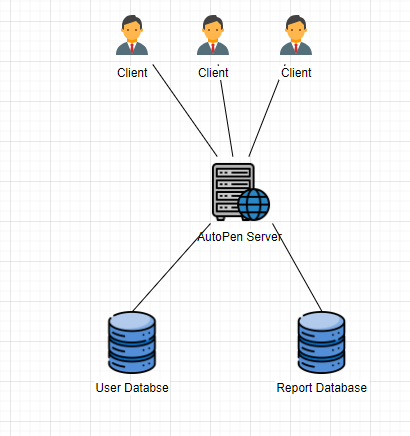
# EXTERNAL INTERFACES

This section is about the external interfaces of the product. The format of Section 5 is as follows: Section 5.1 is about the architecture of the external interfaces, and Section 5.2 goes into detail about the design of the external interfaces.

## Interface Architecture

For AutoPen to function efficiently, it interfaces with external systems. These include payment gateways for subscription services to external databases for vulnerability definitions and cloud storage solutions for report storage. This section elaborates on the electronic interplay between AutoPen and these external systems.

For the Interface Architecture, AutoPen interfaces via web-based APIs for real-time interactions and scheduled batch transfers for periodic data syncing or backup. These interfaces give data exchange and functional interplay. As for the general architecture of AutoPen, Wide Area Networks (WAN) for expansive data transfers are used, with gateway interfaces being used for payment processes and other third-party services.

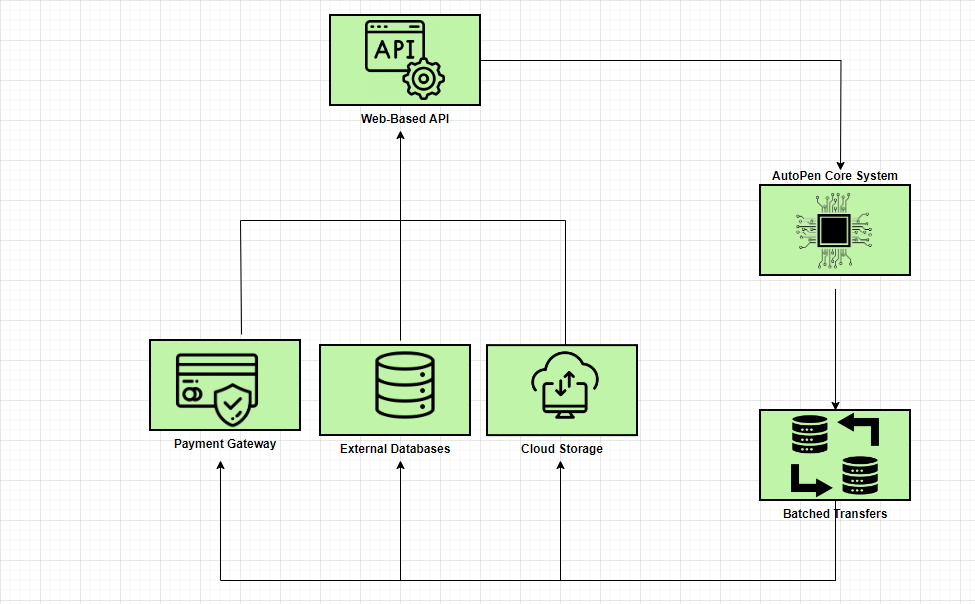


**Figure 19: Client-Server Communication**

Figure 17 provides an overview of the communication scheme of the clients, servers, and data stores. The client runs pentesting operations on the server wherein the server aggregates and stores the data from the operation in a database, from there, the server pulls data from the database and transforms it into visual graphics for the client.

## Interface Detailed Design

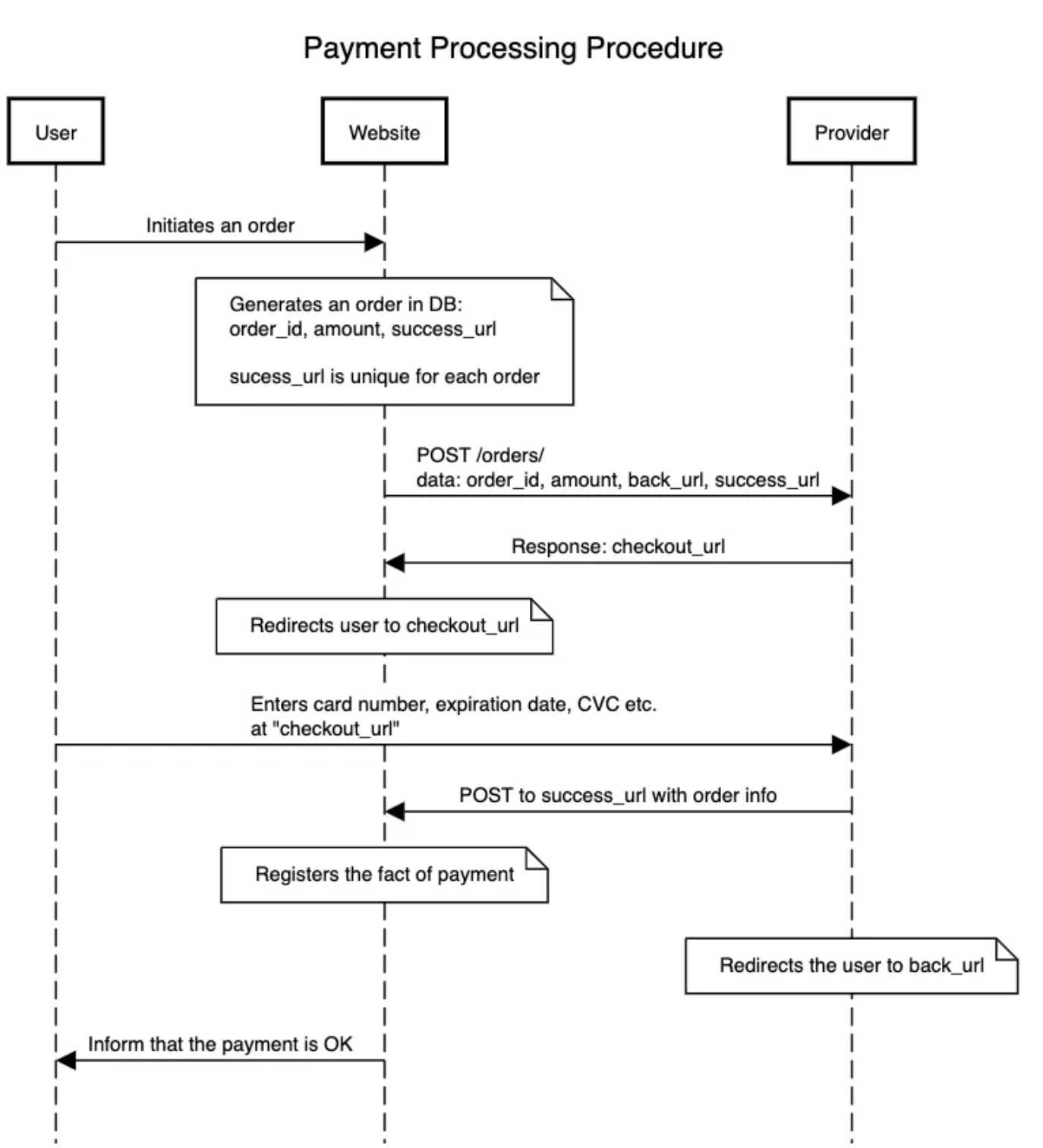
This section describes AutoPen's external interfaces, which are segmented into three distinct subsections, each dedicated to a specific type of interaction and characterized by specific communication attributes. Subsection 5.2.1 Payment Gateway Interface, describes AutoPen's method of handling financial transactions, detailing the data structures, transactional message sequences, and error logging mechanisms employed. Subsection 5.2.2 External Vulnerability Database Interface, specifies the protocol for systematic data retrieval from security databases, including synchronization strategies and integrity checks. Finally, Subsection 5.2.3, Cloud Storage Interface, focuses on data storage conventions, including file transmission confirmation sequences and contingency measures for data transfer discrepancies while attempting report archiving and access.



**Figure 20: External Interfaces**

Figure Y provides a high-level overview of the AutoPen system’s external interfaces, encapsulating six main components: Web-based API, AutoPen Core System, Batched Transfers, Payment Gateway, External Databases, and Cloud Storage. Arrows indicate the flow of data and control between the components. The Web-Based API facilitates immediate interactions with the Core System, while Batched Transfers ensure periodic syncing between AutoPen and external data services. The Payment Gateway, External Databases, and Cloud Storage represent key parts of the interface for financial transactions, security data, and report storage respectively.

**5.2.1 Payment Gateway Interface:** The Payment Gateway Interface of AutoPen employs a Standard Payment Data Format, encompassing user billing details, transaction IDs, payment statuses, and timestamps. Hand-shaking protocols include an initial request from AutoPen, acknowledgment of receipt from the gateway, and subsequent transmission of transaction status. In the event of payment discrepancies or failures, error reports are generated and logged within AutoPen. Queries can be executed using transaction IDs to fetch the corresponding status.



**Figure 21: Payment Process through Django**

Depicted in Figure 17 is the workflow for processing a payment via a Django-based system. The interaction starts when a user places an order, prompting the website to generate a corresponding record in the database with specific identifiers, including an order ID and a success URL. Following this, the website sends these details to the payment provider via a POST request and receives a checkout URL in return. The user is then directed to this checkout URL to input their payment credentials. Upon completion, the website sends the payment data to the success URL, acknowledges the transaction, and guides the user back to a predetermined URL, confirming the payment's success.

**5.2.2 External Vulnerability Database Interface:** This interface deals with the exchange of vulnerability definitions, metadata, timestamps, severity rankings, and suggested mitigation measures. Hand-shaking protocols are initiated by an AutoPen request followed by acknowledgment, leading to data transfer and a final acknowledgment upon completion. Discrepancies or failures during data transfer trigger error logs within the system. Querying the system with a vulnerability ID returns detailed information on that specific vulnerability.

**5.2.3 Cloud Storage Interface for Report Storage:** AutoPen's Cloud Storage Interface manages file formats, user IDs, timestamps, report data, and encryption details. The file upload process begins with an initiation protocol, followed by an acknowledgment from the cloud service, progression updates during the upload, and a final confirmation upon successful transfer. Should there be any upload failures or data mismatches, error notifications are promptly issued. Retrieval of specific reports is facilitated by queries based on user ID and timestamp, providing users with access to their data.

# SYSTEM INTEGRITY CONTROLS

The AutoPen system enforces a stringent protocol where access to critical data items is exclusively reserved for the creators and developers of the system, a measure that guarantees only those with authorized credentials are empowered to view and manipulate sensitive information. This restrictive approach should substantially lower the risk of unauthorized access or potential data breaches. Within the framework of Internal Security for Critical Data Access, the application adopts a Role-Based Access Control (RBAC) strategy, designed to allocate access and functional capabilities in strict accordance with the user's assigned role, effectively minimizing the chances of unauthorized data disclosure. The application delineates clear roles in alignment with operational requirements and security policies, ensuring operational functionality is maintained without compromising security integrity. These roles are crafted to reflect the specific responsibilities within the system, such as administrators who are given system access for user management, test configurations, and viewing all penetration test results, managers with permissions to configure tests, view results for their team's projects, and manage project settings, and users, specifically penetration testers, who are allowed to configure and initiate penetration tests and view their own test results. Each role is endowed with permissions that precisely mirror their responsibilities, a system that ensures the security measures do not impede the operational efficiency of the application.

Implementing authentication and authorization mechanisms, such as Multi-Factor Authentication (MFA), are critical components of this structure. MFA requires users to undergo multiple verification steps during login, which includes the use of a strong, unique password that meets industry-standard complexity requirements and token-based authentication. Following the authentication phase, the application conducts an authorization process to verify if the user holds the necessary permissions, as defined by their role, to carry out the requested action or access certain data, ensuring that the user's interactions with the application are confined to the privileges associated with their roles. To maintain the integrity and security of the application, audit procedures are put in place. These procedures are designed to flag any unauthorized access attempts immediately, generating instant notifications to designated administrators to allow for rapid response to any potential security incidents, allowing AutoPen systems to preserve a competent security posture.

# SYSTEM DISCLAIMERS

To protect both users and owners of AutoPen, this document includes a list of disclaimers concerning the use of the AutoPen application. For users to operate within AutoPen terms of service they must first review and confirm their understanding of all AutoPen disclaimers.

Disclaimer 1: AutoPen test results that are generated are intended solely for the users who have requested the specific assessment. Please keep in mind that these results are confidential and should not be shared without proper authorization. If you have any questions or concerns regarding the test results, we encourage you to contact the AutoPen support team for further assistance.”

Disclaimer 2: Parts of the document contain statements written with the help of ChatGPT.