AutoPen

System Design Document

Team members: Calla Robison, Caleb Hall, Michael Allen, Myles Scott, Joshua B.

Version/Author	Date
Version 2.0 / 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

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

Cybersecurity threats are rapidly evolving, requiring more agile and robust security systems. AutoPen aims to revolutionize the field of penetration testing by leveraging artificial intelligence to conduct automated, comprehensive, and adaptive 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.

Scope:

- Designed primarily for small and medium enterprises (SMEs).
- Covers common vulnerability areas, including OWASP Top 10.
- Ability to adapt to new threats by machine learning.
- The option to run tests periodically or on demand.
- Compliance with standard cybersecurity frameworks and regulations.

1.2 Project Executive Summary

With the increasing complexity and sophistication of cyber threats, organizations are in need of robust 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 machine learning, artificial intelligence, and big data analytics 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.

1.2.1 System Overview

AutoPen utilizes machine learning to train a model to conduct penetration testing. The system will be hosted on the cloud, with a web app 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.

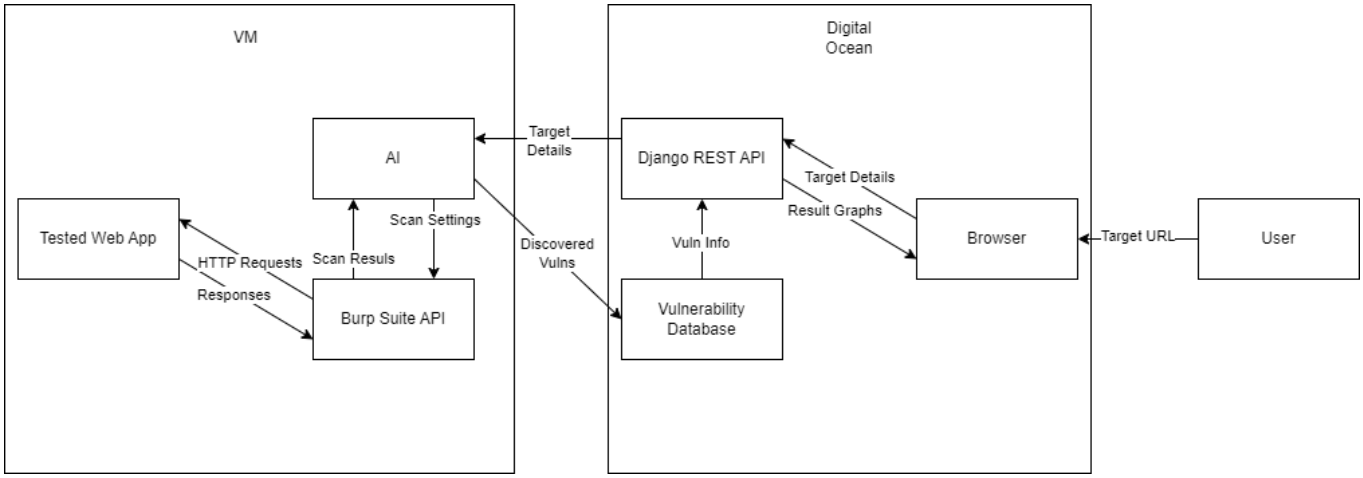

Figure 1: System Diagram

Primary programming languages:

- Client-side programming: Presentation layers and user interaction. Client-side code is responsible for rendering web pages, handling user input, and making the user experience dynamic and responsive.

  Python, Javascript and HTML/CSS

- 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: Django framework

  JavaScript: React

  Data Visualization: React (chart.js) and Django REST API

AutoPen utilizes the Python programming language due to the available packages for machine learning and other computations such as NumPy, Pandas and TensorFlow.

Area of Penetration Testing: Web application pen-testing

1.2.2 Design Constraints

AutoPen is an advanced AI system designed to test multiple vulnerabilities within a company's website. In order to ensure the effectiveness and reliability of AutoPen, there are several design constraints that need to be considered. The primary focus lies in developing an AI that is capable of identifying and exploiting vulnerabilities while simultaneously ensuring its own security and preventing unauthorized access.

Internal:

- 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.

External:

- Burp Suite: The project will use the professional version of Burp Suite for web vulnerability scanning. Therefore the project is limited in abilities to the offerings of this product.
- Digital Ocean Hosting: The project will be deployed on a basic tier of Digital Ocean Hosting, which provides essential services within the project's budget. The project will not utilize higher-tier services that incur additional costs, limiting the compute power of the assets.

Legal Constraints: The product must be compliant with the United States' Computer Fraud and Abuse act that makes unauthorized access to protected computers a crime, and the Electronic Communications Privacy Act which regulates the disclosure and interception of communications related to electronics, as well as comply with the EU's General Data Protection Regulation that gives strict requirements for the protection and processing of personal data. In addition, the product must be liable for any damages caused by the penetration test, and 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.

There are two main assumptions, that being the legality of this tool and the hosting platform dedicating enough resources to ensure efficient operation of our programs.

1.2.3 Future Contingencies

Overview:
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.

One possible contingencies that might arise in the design of the system is that a user accidentally uses the platform in an unintended way and breaks the site to be tested, with a possible workaround being to force the application to do a certain type of pentest. In addition, the application may end up accidentally attacking websites other than the one being tested, so the user must provide an URL of the website that the application must solely attack.

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.

1.3 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.

1.4 Project References

- Documentation for Metasploit, the tool that was previously used for the ai before changing to burp suite, can be found at: https://www.metasploit.com/

- Documentation for burp suite, the current tool used for the AI creation, can be found at: https://portswigger.net/burp
- Github for the project can be found at: https://github.com/AutoPen/AutoPenProj

1.5 Glossary

- AI: Artificial Intelligence. It refers to the capability of a machine to imitate intelligent human behavior.
- 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 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.

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

Overview:
The AutoPen system 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.

2.2 System Software Architecture

Overview:
The AI-powered Penetration Testing system is designed with a three-tier architecture: Presentation Layer (Web UI), Business Logic Layer (Backend Server), and Data Layer (Database).

Software Modules:

- User Management Module: Handling user registration, login, and profile management.
- Test Initialization Module: Initiating and configuring penetration tests.
- AI Testing Module: Incorporating AI models and algorithms for penetration testing.
- Reporting Module: Generating and displaying test results to users.
- Notification Module: Alerting users about test completions or system updates.

Languages:

- Frontend: HTML, CSS, JavaScript (React.js)
- Backend: Python 3.11 (using Python Django for web services)

Tools:

- Git: For version control.
- Docker: Containerizing application components for consistent deployment and testing.
- Digital Ocean IaaS: A cloud hosting platform and services to integrate the pentesting AI into the web application interaction and data visualization
- Digital Ocean Droplets: Linux-based virtual machines (VMs) that run on top of virtualized hardware. Each Droplet is a new server, either standalone or as part of a larger, cloud-based infrastructure
- Github Actions: For continuous integration and continuous deployment (CI/CD). Using a deploy.yaml that will consistently update repo changes to the cloud service Digital Ocean
- Kali Linux: Linux distro with pre-built tools and software for penetration testing
- VMWare: Virtualization software for hosting VMs
- Metasploit: A tool suite made for penetration testing and IDS signature development, that was later replaced by burp suite
- Burp suite: A tool suite made for penetration testing and IDS signature development
    - WMap: Web application exploit/vulnerability scanner

Narrative:
The user interacts with the Presentation Layer, where they can register, log in, initiate tests, and view results. This communicates with the Business Logic Layer, where the main functionalities like AI model invocations, test configurations, and report generations take place. The results, user data, and test configurations are stored and fetched from the Data Layer which is the database.

2.3 Internal Communications Architecture

Overview:
Our 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.

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.
- Diagram Reference:

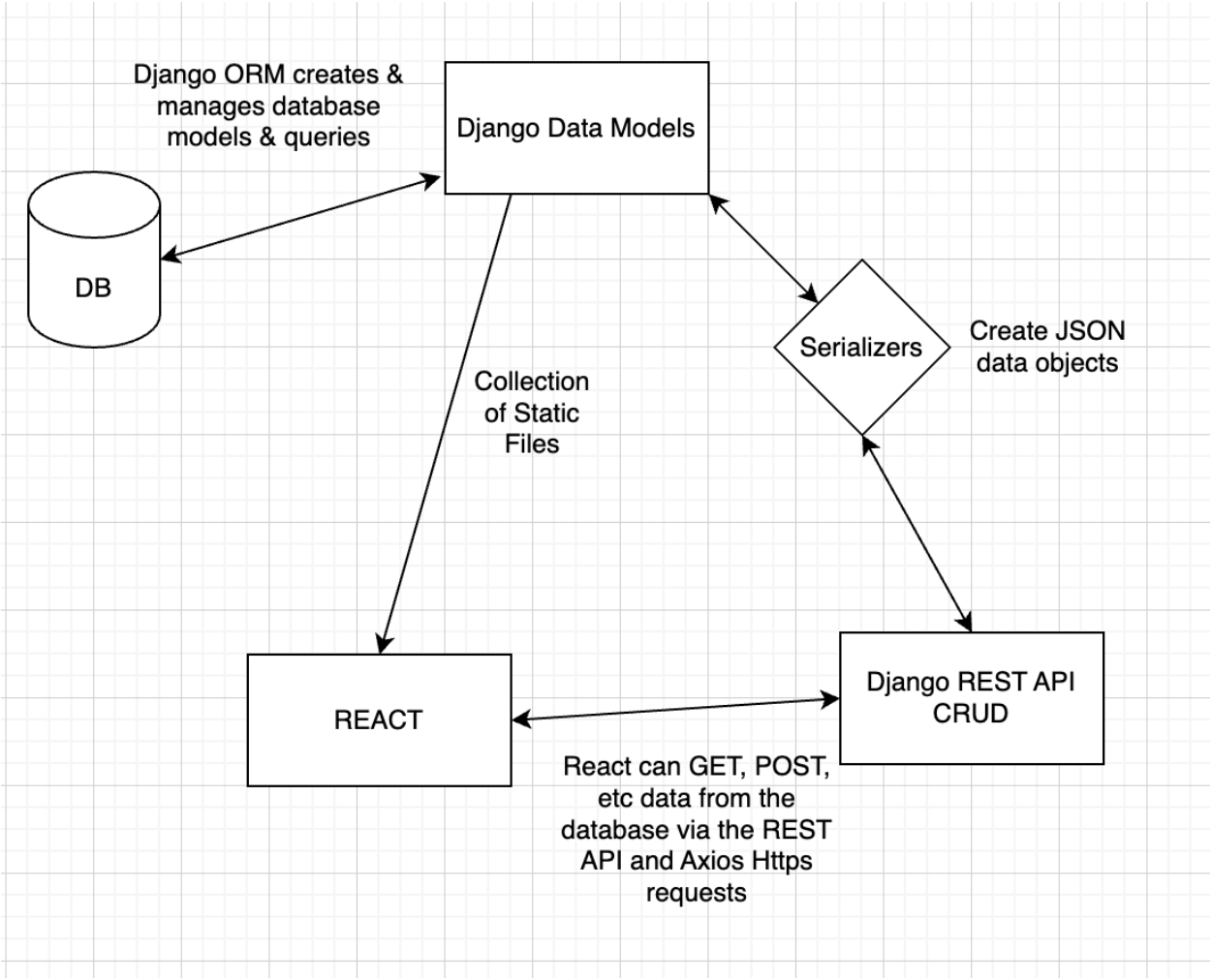

Figure 2: Internal Communication Flow

Narrative:
When a user accesses the web application, their browser communicates with the backend server using HTTP/HTTPS requests. For tasks like initiating 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.

3 HUMAN-MACHINE INTERFACE

3.1 Inputs

Overview:
The main input methods are through data entry screens on the web application. These input mechanisms directly map to the high-level data flows described in the System Overview section.

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.
    - 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.

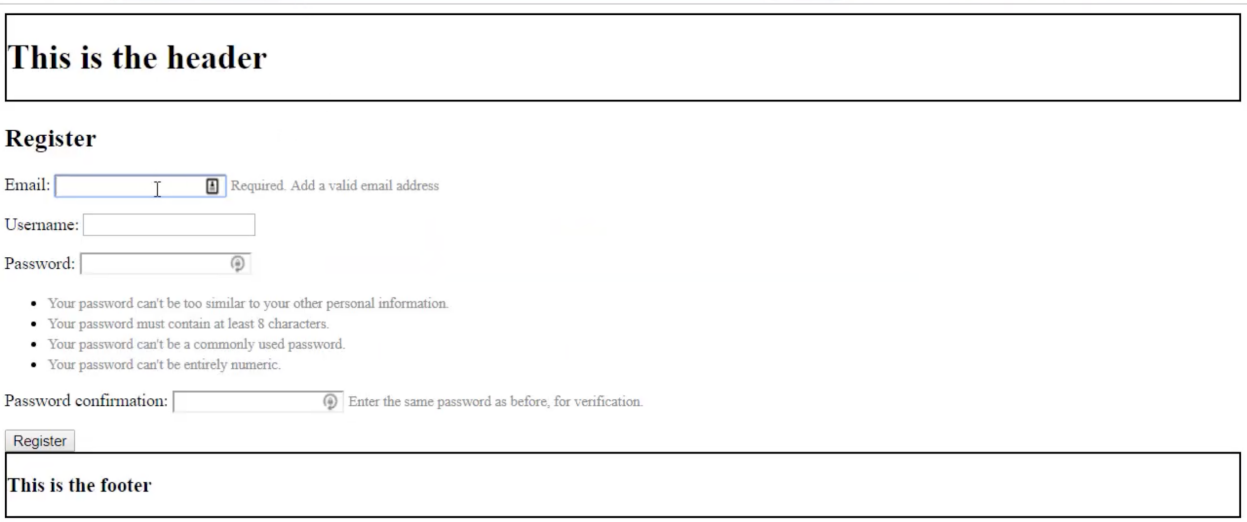

Figure 3 : User Registration Screen

- Login Screen:
    - Data Elements: Username/Email, Password.
    - Mandatory fields: Username/Email and Password.
    - Controls: Users are barred from proceeding without correct credentials.
    - Access Restrictions: Exclusively accessible by unregistered users or logged-out users.

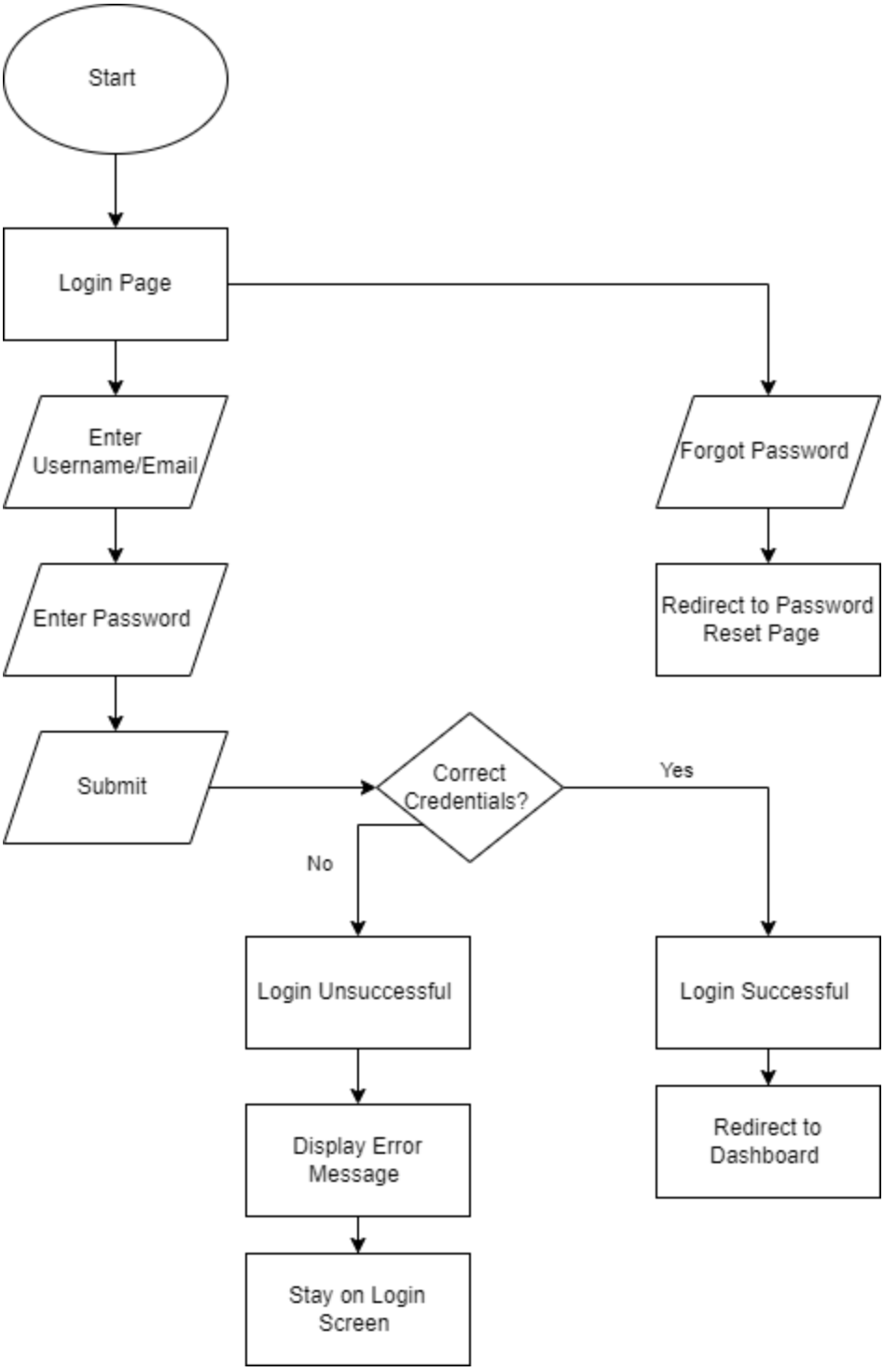

Figure 4: Login Workflow

- Payment Screen:
    - Data Elements: Credit/Debit Card Number, CVV, Expiry Date, Billing Address.
    - All fields are mandatory, and include:
        - Credit/Debit Card Number should adhere to card standards.

        - Card expiration date
        - Billing address
        - CVV should be 3 or 4 digits.
    - Controls: Payment cannot be processed without valid payment details.
    - Access Restrictions: Only registered users access premium features or services.

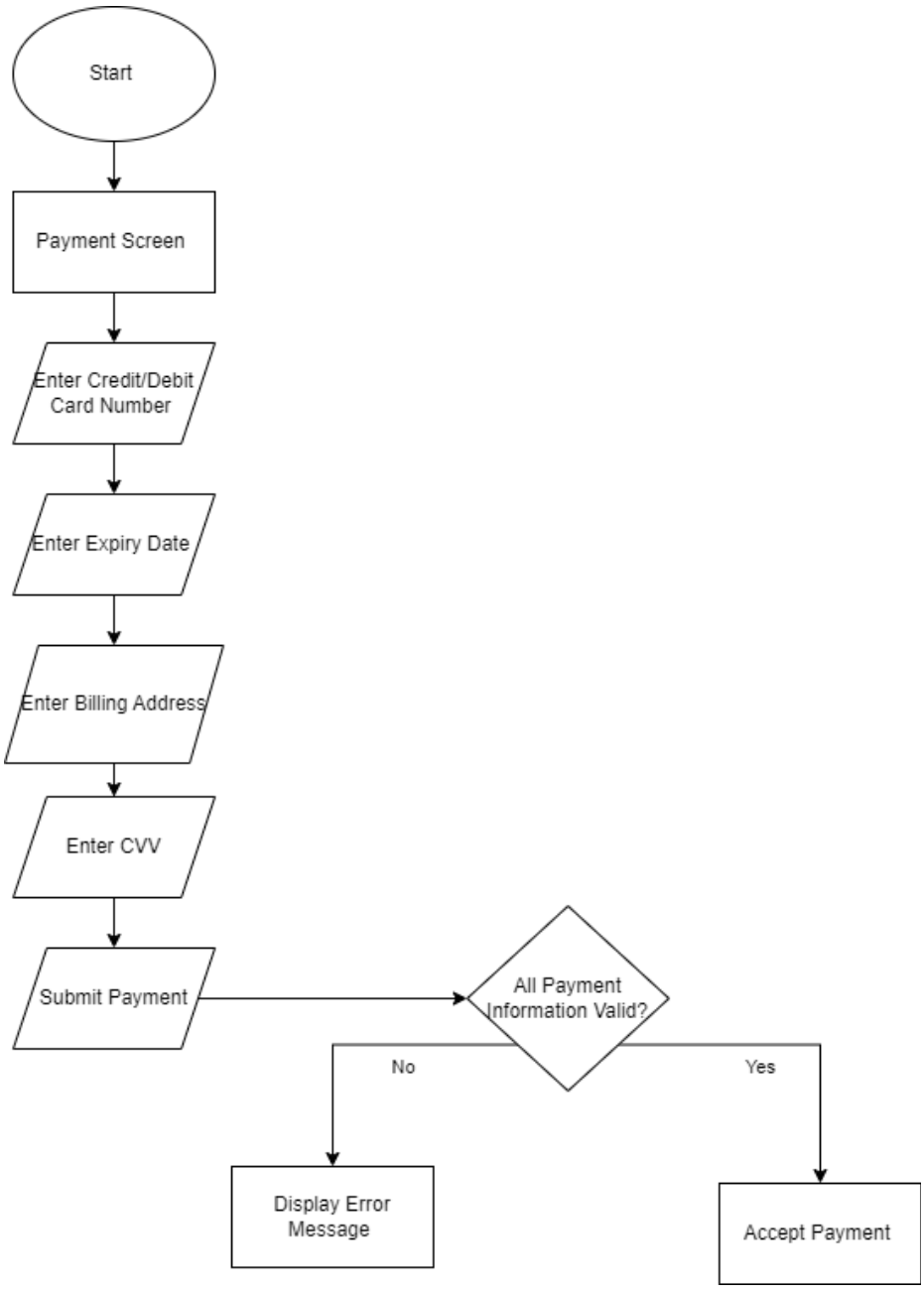

Figure 5: Payment Workflow

- Launch/Progress Screen:
    - Data Elements: Progress bar, Status Messages (like "Scanning", "Analysis in progress"), Estimated Time Left.
    - Controls: Users can potentially pause or cancel ongoing tests.
    - Access Restrictions: Available to registered users who have initiated a penetration test.
- Generate Report Screen:
    - Data Elements: Report contents, Vulnerabilities Found, Suggested Fixes, Download Report Button.
    - Controls: Users can download or print the generated report.
    - Access Restrictions: Available post-completion of a penetration test.

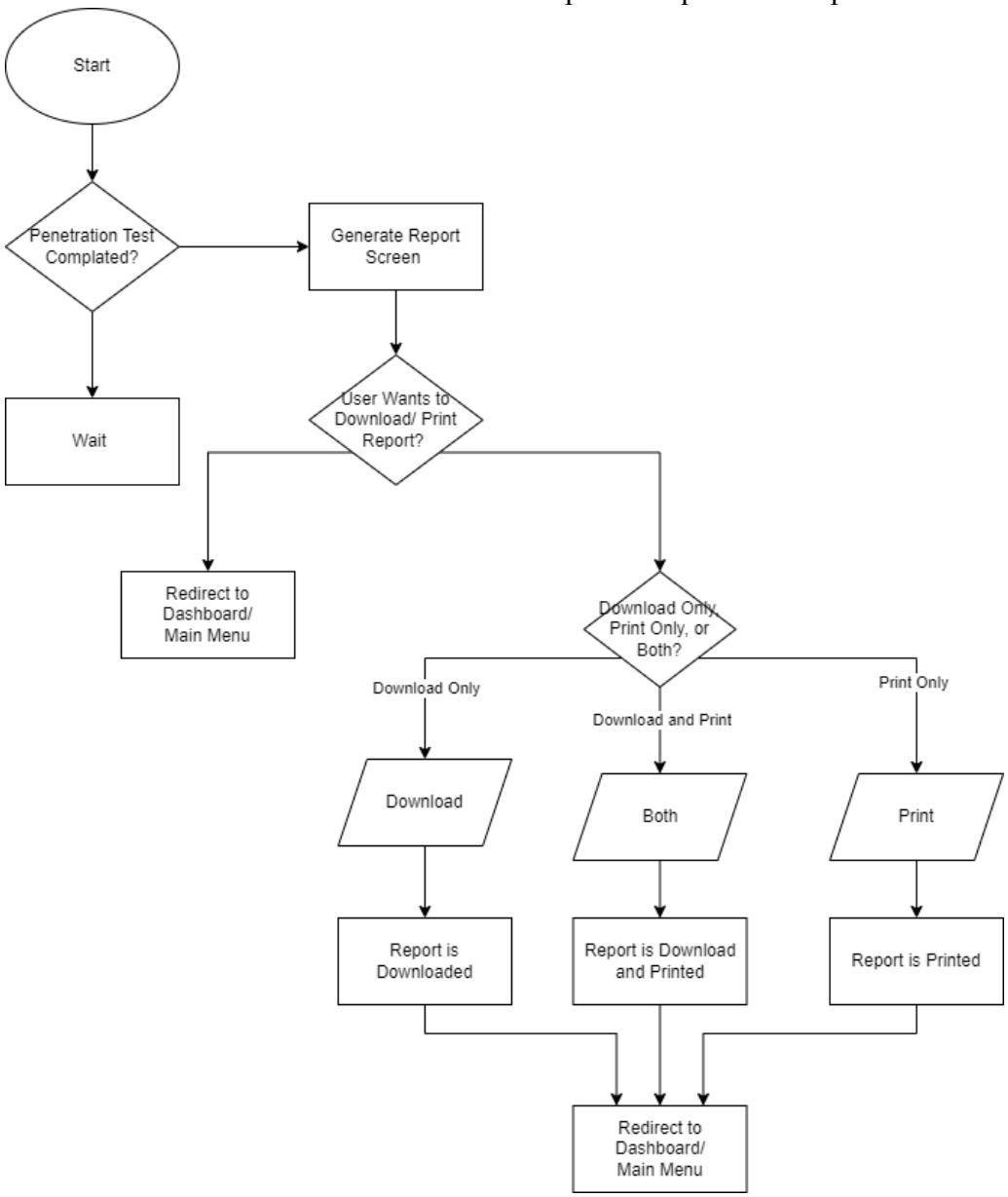

Figure 6: Generate Report Flow

- Penetration Test Configuration 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.

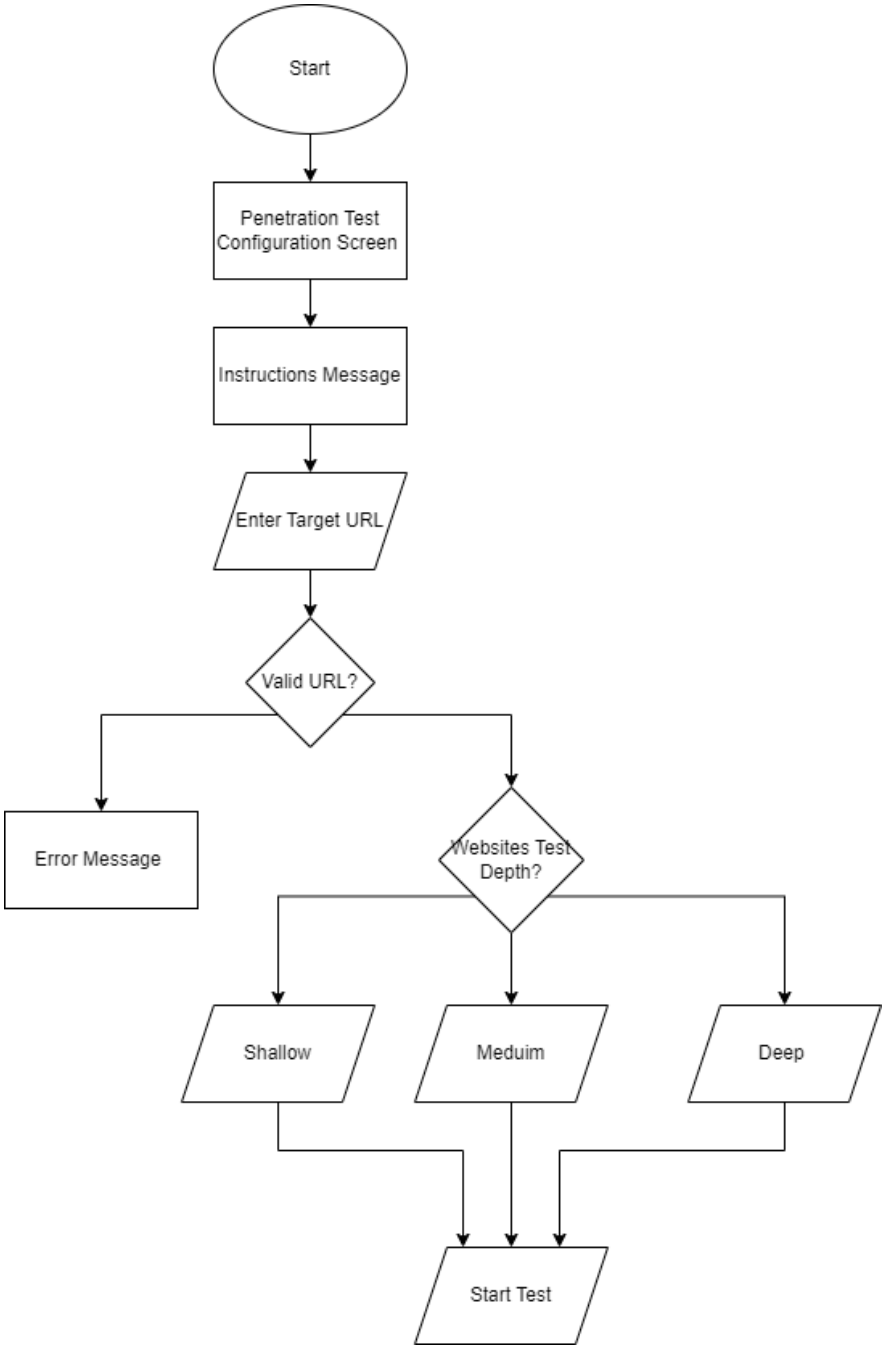

Figure 7: Penetration Configuration Workflow

Miscellaneous Messages:
- Success messages upon successful registration or test initiation.
- Error messages detailing the nature of the error, e.g., "Invalid email format" or "Passwords do not match."

Transaction Codes:
- REG_USER: User registration
- INIT_TEST: Initiate penetration test

3.2 Outputs

Overview:
System outputs primarily consist of data display screens showing test results, user profiles, and system notifications.

Test Result Screen:
- Identification Code: **TEST_RES**
- Contents: Vulnerabilities detected, severity levels, potential fixes, and an overall security rating. This will be represented in a graphical format using line graphs and pie charts.
- Purpose: To provide users with insights into their system's vulnerabilities and guide them on mitigation.
- Primary Users: Registered users who initiated the tests.
- Access Restrictions: Only the user who initiated the test can view its results.

User Profile Screen:
- Identification Code: **USER_PROF**
- Contents: User's registered details, past test history, and configurations.
- Purpose: Allows users to view and edit their details and review past tests.
- Primary Users: Registered users.
- Access Restrictions: Users can only view their own profiles.

System Notification Screen:
- Identification Code: **SYS_NOTIF**
- Contents: System-related notifications, updates, or alerts.
- Purpose: To keep users informed about system-related updates or changes.
- Primary Users: All users of the platform.
- Access Restrictions: Varies based on the nature of the notification. Some might be general, while others might be user-specific.

4 DETAILED DESIGN

4.1 Hardware Detailed Design

Overview:
"AutoPen" is an innovative web-based application designed to automate the penetration testing process. At its heart, it leverages a Python-based backend that orchestrates and drives the penetration tests, while users interact through a responsive web interface. This detailed design focuses on the underlying cloud infrastructure specifications essential for "AutoPen" and the user device requirements to interact with the system optimally.

Cloud Server Component for "AutoPen":
"AutoPen" primarily functions as a web application with its core logic and backend operations developed in Python. Its hosting on cloud infrastructure ensures scalability, redundancy, and efficiency. Here are the specifications tailored for the application:

Instance Type:
- High-compute cloud instance optimized for tasks such as AI model operations, especially given Python's data processing requirements.

Memory:
- Website host does not specify an exact number.

Storage:
- High-speed SSD storage, at 20GB, ensuring swift data read/write operations which are crucial for real-time penetration testing analytics.

Processor:
- Multi-core CPUs that cater well to Python's processing needs, ensuring rapid response times for user requests.

User Device Requirements for Accessing "AutoPen" Web Interface:
Users access "AutoPen" via its web interface. Optimal experience demands:

Desktop Systems/Laptops:
- Browser: Up-to-date web browser such as Chrome, Firefox, Safari, or others capable of supporting modern web standards and JavaScript.
- Memory: At least 4GB RAM for smooth functioning.
- Processor: Dual-core processor or better to ensure quick rendering and processing of web content.
- Monitor resolution: 1366x768 pixels or higher for the best display experience.

Tablets/Smartphones:
- Browser: Default mobile browsers, like Safari for iOS and Chrome for Android.
- Memory: 2GB RAM or more for fluid navigation and interaction.
- Screen resolution: A responsive design of "AutoPen" ensures adaptability to various screen sizes, from smartphones to tablets.

4.2 Software Detailed Design

Overview:
"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, ensuring optimal performance and streamlined operations.

Core Software Modules:
- User Management Module:
    - Narrative Description: Manages user registration, authentication, profile management, and session handling. It ensures secure and seamless user interactions.

- Interfaces: Interacts with the PostgreSQL Database Module for user data storage and retrieval and the Notification Module for alerts.
- Data Elements: Username, Password (hashed and salted), Email, User profile details.
- Graphic Representation:

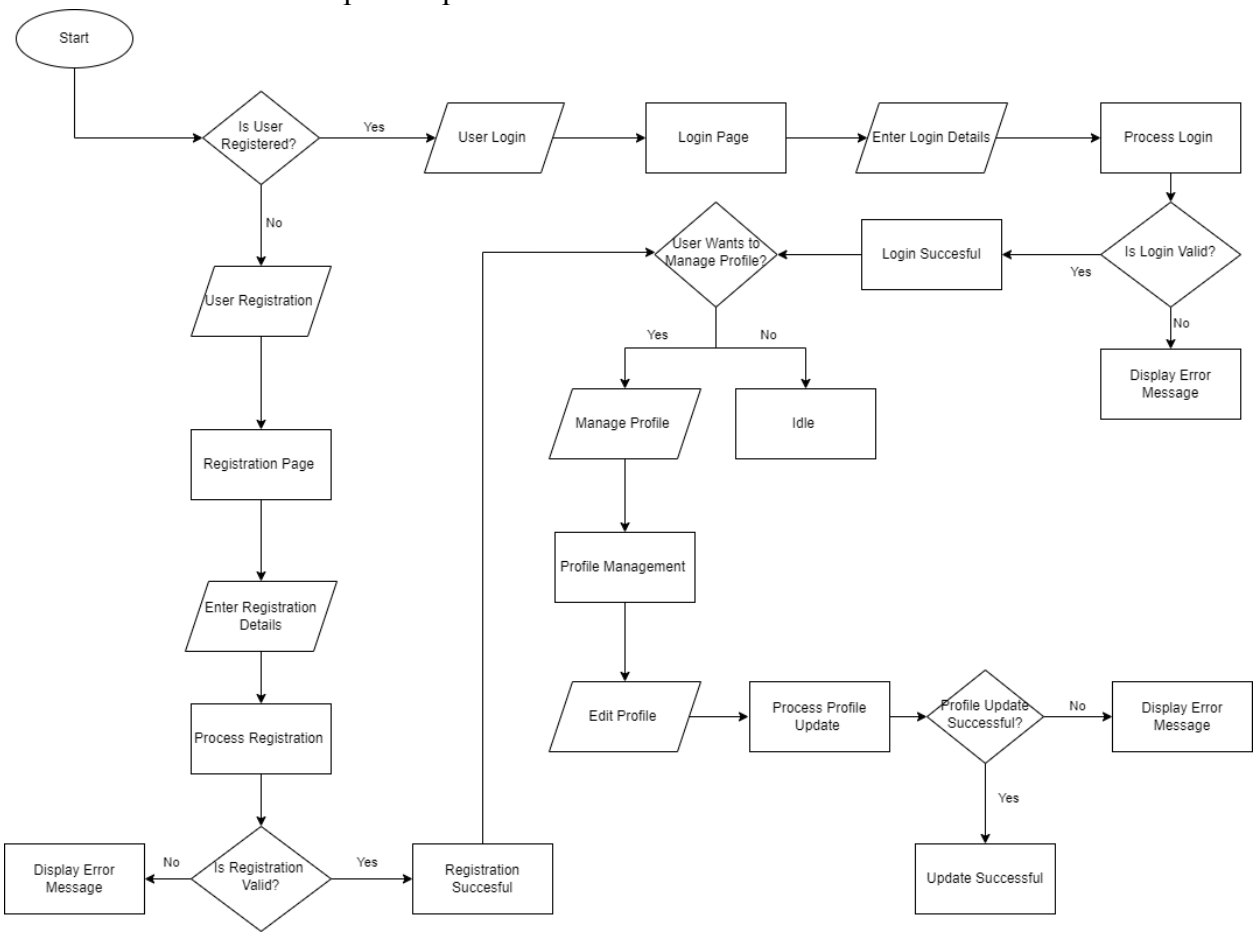

Figure 8: User Management Module Flowchart

- AI Penetration Testing Module:
  - Narrative Description: The heart of "AutoPen", this module handles the AI-driven penetration tests. It deploys various AI algorithms and models to simulate and conduct tests.
  - Algorithms: Proprietary AI algorithms tailored for penetration testing, threat modeling, and vulnerability detection.
  - Interfaces: Communicates with the PostgreSQL Database Module for storing test configurations and results.
  - Data Elements: Target URL, Test parameters, AI model outputs, Vulnerability reports.
  - Graphic Representation:

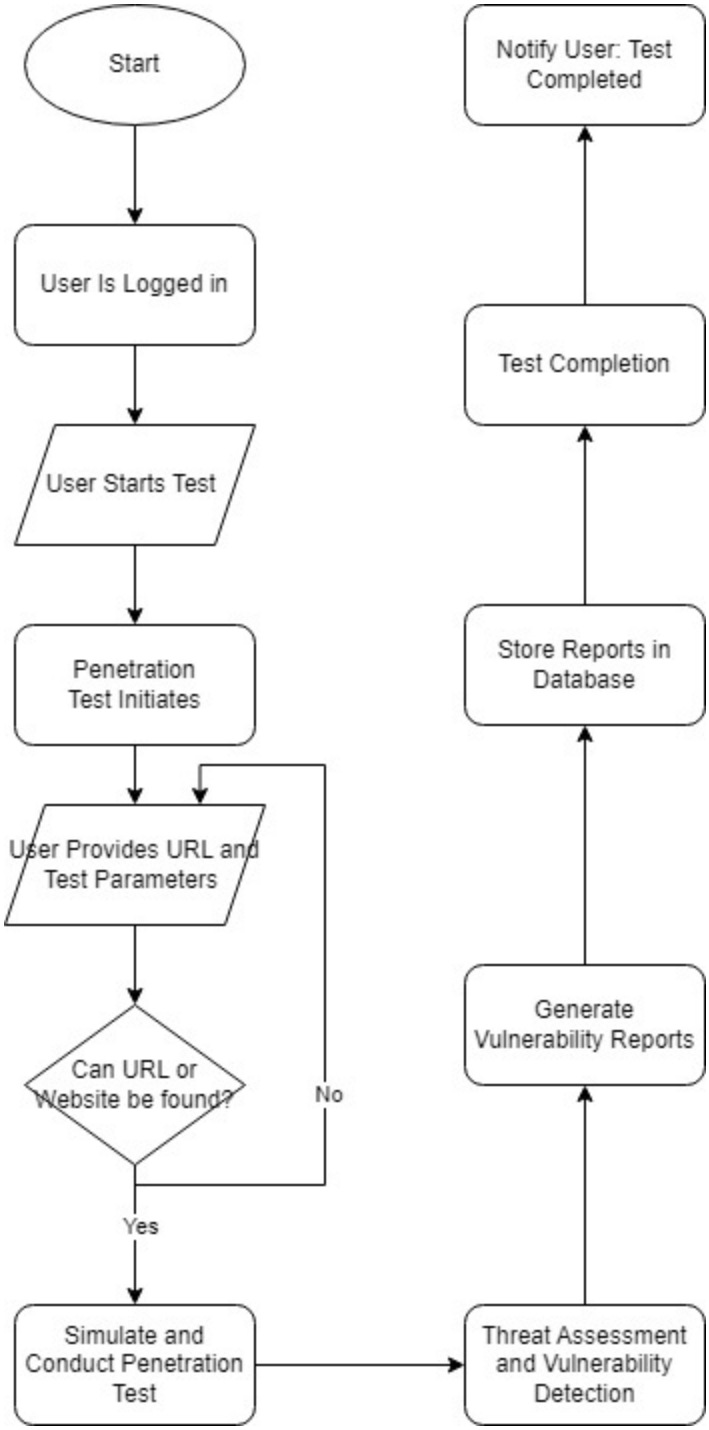

Figure 9: AI Penetration Testing Module Flowchart

- Database Module:
    - Narrative Description: Centrally manages all data storage and retrieval operations, ensuring data consistency, integrity, and security.
    - Interfaces: Serves all other modules, providing them with necessary data storage and access functionalities.

- Data Elements: User data, test configurations, AI-generated reports, system logs.
- Graphic Representation:

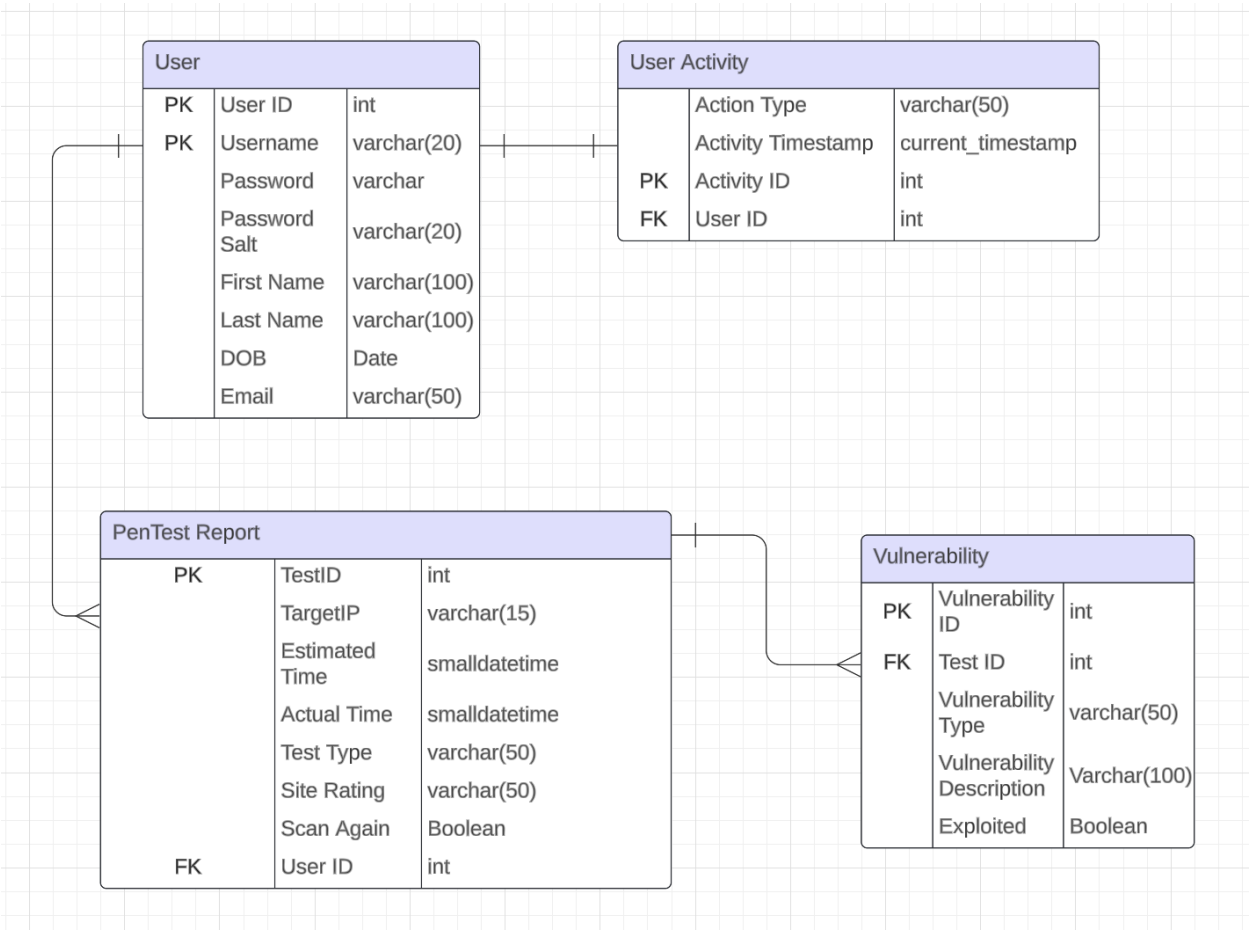

Figure 10: Web Database Schema

- Notification Module:
  - Narrative Description: Responsible for sending real-time updates, alerts, and notifications to users.
  - Interfaces: Mainly communicates with the User Management and AI Penetration Testing modules.
  - Data Elements: Notification content, User contact details, Notification logs.
  - Graphic Representation: (A flowchart detailing the notification triggering and delivery process would be provided.)

4.3 Internal Communications Detailed Design

Overview:
Given that "AutoPen" is a web-based application, internal communications play a pivotal role, particularly when discussing backend-server to database interactions, real-time data exchange, and synchronous/asynchronous task handling.

- Number of Servers and Clients: Given it's cloud-hosted, there's a dynamic allocation of servers based on demand. Every user accessing the platform is considered a client.
- Specifications for Bus Timing and Control: Standard cloud infrastructure communication protocols, optimized for minimal latency.
- Data Exchange Formats: Primarily JSON for API interactions, ensuring lightweight and structured data transfers.
- Graphical Representation: (A network diagram showcasing the cloud server clusters, data centers, and potential CDNs would be illustrated, providing clarity on data flow and infrastructure layout.)
- LAN Topology: In a cloud environment, virtual LANs (VLANs) ensure data segregation and traffic optimization.

5 EXTERNAL INTERFACES

5.1 Interface Architecture

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

Interface Architecture:
"AutoPen" interfaces primarily via web-based APIs for real-time interactions and scheduled batch transfers for periodic data syncing or backup. These interfaces ensure seamless data exchange and functional interplay.

Architecture:
Wide Area Networks (WAN) for expansive data transfers.
Gateway interfaces for payment processes or other third-party services.
Diagram: A diagram would illustrate the connection pathways between "AutoPen" and external systems, emphasizing data flow directionality and potential protocols. This would correlate with context diagrams presented earlier.

5.2 Interface Detailed Design

Payment Gateway Interface:

- Data Format: Standard Payment Data Format which includes user billing details, transaction IDs, payment status, and timestamps.
- Hand-shaking Protocols: Initial request from "AutoPen" to payment gateway, acknowledgment receipt from gateway, followed by transaction status.
- Error Reports: Payment failures or discrepancies trigger error reports. These are logged in "AutoPen" and potentially forwarded to users via notifications.
- Query and Response: Transaction ID query fetches transaction status.

External Vulnerability Database Interface:

- Data Format: Vulnerability definitions, metadata, timestamps, severity rankings,

and suggested mitigation measures.
- Hand-shaking Protocols: Initiation request, acknowledgment, data transfer, and completion acknowledgment.
- Error Reports: Any discrepancies or data transfer failures result in error logs.
- Query and Response: Query by vulnerability ID returns specific vulnerability details.

Cloud Storage Interface for Report Storage:
- Data Format: User ID, timestamp, report data, file format, and encryption details.
- Hand-shaking Protocols: File upload initiation, acknowledgment, upload progress, and completion status.
- Error Reports: Failed uploads or data mismatches lead to error notifications.
- Query and Response: File retrieval by user ID and timestamp fetches specific reports.

6 SYSTEM INTEGRITY CONTROLS

Overview:
Access to critical data items in AutoPen is strictly limited to the creators and developers of the system. This ensures that only authorized personnel have the ability to view and manipulate sensitive information. By restricting access to these individuals, the risk of unauthorized access or data breaches is significantly reduced.

AutoPen Disclaimer:
"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 our support team for further assistance."

Internal Security for Critical Data Access:

Role-Based Access Control (RBAC)
- Define distinct roles (e.g., administrator, manager, user, guest).
- Assign specific access permissions
- Only users with the assigned roles can access the data

Authentication & Authorization
- Use multifactor authentication mechanisms, such as passwords combined with biometrics or token-based authentication.

Audit Procedures

Regularly Performed Audits
- Audit logs should include data access, data changes, login attempts (both successful and unsuccessful), and any other system activities.
- Flag unauthorized access attempts and generate instant notifications to designated administrators.

Disclaimer 1: Parts of document contain statements written with the help of ChatGPT
Disclaimer 2: Information included in this document is subject to change during the process of system design.