A Report on Major Project

**NEXUS – A Software Authentication Checker Tool for Computer Networks**

*SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF*

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER ENGINEERING OF**

**VISHWAKARMA INSTITUTE OF TECHNOLOGY**

## Savitribai Phule Pune University

*BY*

**Aniruddha Kulkarni (GR No. 11910240)  
Shreyas Habade (GR No. 11910447)   
Niharika Rathi (GR No. 11910555)**

**Anushka Shinde (GR No. 11910334)**

*UNDER THE GUIDANCE OF*

Prof. Dr. S. R. Shinde



DEPARTMENT OF COMPUTER ENGINEERING

#### BANSILAL RAMNATH AGARWAL CHARITABLE TRUST’S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PUNE - 411037

2022 - 2023

**BANSILAL RAMNATH AGARWAL CHARITABLE TRUST’S**

## VISHWAKARMA INSTITUTE OF TECHNOLOGY

**(An Autonomous Institute affiliated to Savitribai Phule Pune University)**

## PUNE – 411037



**CERTIFICATE**

This is to certify that the Major Project titled Nexus-A Software Authentication Checker Tool for Computer Networks submitted by **Aniruddha Kulkarni (GR No. 11910240), Shreyas Habade (GR No. 11910447), Niharika Rathi (GR No. 11910555), Anushka Shinde (GR No. 11910334)** is in partial fulfillment for the award of Degree of Bachelor of Technology in Computer Engineering of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This project report is a record of bonafide work carried out by him under my guidance during the academic year 2022-2023.

### Prof. Dr. S. R. Shinde Prof. Dr. S. R. Shinde

Dept. of Computer Engg. Vishwakarma Institute of Technology, Pune.

### Guide Head of Computer Department

**Sign of External Examiner Date**

Bansilal Ramnath Agarwal Charitable Trust’s

### Vishwakarma Institute of Technology, Pune-37

**Department Of Computer Engineering**

## PROJECT SYNOPSIS

Group No: 18 Group Members:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roll No** | **Name** | **Class** | **Contact No** | **Email-ID** |
| 10 | Aniruddha Kulkarni | CS - B | 9822890892 | [aniruddha.kulkarni191@vit.edu](mailto:aniruddha.kulkarni191@vit.edu) |
| 01 | Shreyas Habade | CS - B | 7888264184 | [shreyas.habade19@vit.edu](mailto:shreyas.habade19@vit.edu) |
| 09 | Niharika Rathi | CS - C | 9766034306 | [niharika.rathi19@vit.edu](mailto:niharika.rathi19@vit.edu) |
| 09 | Anushka Shinde | CS - D | 7218250997 | [anushka.shinde19@vit.edu](mailto:anuksha.shinde19@vit.edu) |

Academic Year : 2022-23

Project Title : Nexus- A software authentication check tool

Project Area : Computer Networks, SMART Systems

Sponsor Company : -

Company Address : -

Internal Guide : Prof. Dr. Sandip R. Shinde

Name of the External Guide:

Contact No:

**Signature of Internal Guide**

## ACKNOWLEDGEMENT

#### It gives us great satisfaction to be able to present this project Nexus- A software authentication checker tool. We would like to express our deep gratitude towards our project guide **Prof. Dr. Sandeep Shinde**, for all the guidance and cooperation, without whom this project would have been an uphill task.

**Aniruddha Kulkarni,**

**Shreyas Habade,**

**Niharika Rathi,**

**Anushka Shinde**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Table of contents** | **Pg. no** |
| 1 | Software Project Synopsis | 6 |
| 2 | Feasibility Status Report | 8 |
| 3 | Use Case Analysis Document | 11 |
| 4 | Software Requirements Specification | 13 |
| 5 | Software Project Plan | 18 |
| 6 | Software Implementation Document | 22 |
| 7 | Conclusion | 40 |
| 8 | References | 41 |
| 9 | Acronyms | 42 |

|  |  |  |
| --- | --- | --- |
| **Sr. no.** | **List of figures** | **Pg. no** |
| 1 | System Architecture | 23 |
| 2 | Training Accuracy of all Algorithms | 35 |
| 3 | Testing Accuracy of all Algorithms | 36 |
| 4 | Models of highest training accuracy of each type | 38 |
| 5 | Models of highest testing accuracy of each type | 38 |
| 6 | Models of highest training accuracy | 39 |
| 7 | Models of highest testing accuracy | 39 |

# Software Project Synopsis



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

1. **CONTEXT**

The Nexus project is an extensive and versatile software authentication checker tool designed specifically for computer networks. Developed using the Python programming language, this project incorporates a wide range of functionalities to address the critical need for software authentication and data collection in networked environments. By combining an API created with the Django REST Framework, intuitive graphical user interfaces (GUI) built using PyQt5 and Qt5 Designer, and additional Python scripts, the Nexus tool empowers users to authenticate software, gather essential data from computer systems, store the collected information in a secure database, and perform various operations based on their respective roles within the network.

The primary objective of the Nexus project is to provide a user-friendly and efficient solution for software authentication and comprehensive data collection in computer networks. By implementing a secure and robust system architecture, the tool ensures the verification of software authenticity across both Windows and Unix machines connected to the network. Additionally, it facilitates the collection of relevant software and system information, enabling users to analyze and make informed decisions based on the obtained data.

The graphical user interfaces (GUI) play a crucial role in enhancing the overall user experience and interaction with the Nexus tool.

Data collection stands as a key feature of the Nexus tool. It encompasses the comprehensive gathering of software details, system information, and authentication status from both Windows and Unix machines. The collected data then undergoes a meticulous authentication process to identify trusted and untrusted software. In cases where untrusted applications are detected, the tool leverages the SMTP library in Python to send email notifications to users, suggesting the uninstallation of untrusted software and encouraging installation from trusted sources.

To ensure accessibility and scalability, the Nexus project is designed to be deployed on popular platforms such as Heroku for hosting the API component and GitHub for efficient source code management. This deployment strategy allows users to access the tool from various environments while enabling seamless scalability based on the network's evolving requirements.

### PROBLEM

In computer networks, the verification of software authenticity and the collection of comprehensive data from connected systems pose significant challenges. Existing solutions often lack a comprehensive approach to authenticate software and collect relevant information, leading to potential security vulnerabilities and inefficiencies in software management. Moreover, the absence of a user-friendly tool that integrates various functionalities further hampers the overall efficiency and effectiveness of software authentication and data collection processes.   
The need arises for a Python-based solution with an API, GUIs, and scripts to authenticate software, collect data from Windows and Unix machines, securely store information, and provide intuitive interfaces for user registration, login, and data retrieval. Enhancing security and efficiency in software management within computer networks is the primary challenge to address.

### SOLUTION

The Nexus project proposes a comprehensive software authentication checker tool for computer networks. By leveraging Python, the project integrates an API, GUIs, and scripts to address the challenges of software authentication and data collection. The solution offers a user-friendly interface for user registration, login, and data retrieval, allowing for seamless authentication and data gathering

from Windows and Unix machines. The collected data is securely stored in a database, enabling efficient software management. With its holistic approach, the Nexus tool aims to enhance security, streamline software authentication, and improve overall efficiency in computer network management.

# Feasibility Study Report



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

1. **CURRENT SYSTEMS AND PROCESSES**

Not a lot of work has been done that is like Nexus in terms of Software authentication and data collection. Some companies make an authentication system to keep a tab on their own company network and their own software’s. For example, some software’s can only be accessed when you connect through their VPN so they can monitor the activity and collect data.  
An example of the same could be Microsoft System Centre Configuration Manager (SCCM): SCCM is a comprehensive system management tool by Microsoft that offers features for software deployment, inventory management, and software updates. It provides capabilities for software authentication and data collection from networked devices.

### SYSTEM OBJECTIVES

The primary objectives of the Nexus project are as follows:

1. Implement a robust API using Django REST Framework to handle authentication and data storage.
2. Develop graphical user interfaces (GUI) using PyQt5 and Qt5 Designer for seamless user interaction.
3. Collect and store data from Windows and Unix machines, including software details, authentication status, and system information.
4. Authenticate software by verifying its authenticity and identifying any untrusted applications.
5. Enable administrators to fetch data for their devices and other registered devices.
6. Provide teachers with the ability to push their device details to the API for authentication.

### ISSUES

There can be several issues that can arise while developing a Software Authentication Checker Tool for Computer Networks, some of them are listed below:

1. Compatibility: Ensuring compatibility across different operating systems, software versions, and network configurations can be a major challenge. The tool needs to work seamlessly on Windows and Unix systems and handle various software environments.
2. Data Collection: Collecting accurate and comprehensive data from connected systems can be complex. Different operating systems and software may have unique data structures, making data collection and standardization a challenge. Handling large amounts of data efficiently and securely is also crucial.
3. Software Authentication: Implementing robust software authentication mechanisms requires careful consideration. Verifying the authenticity of software installations, detecting unauthorized or tampered applications, and distinguishing between genuine and counterfeit software can be difficult.
4. Security: Security is a critical aspect of the tool's implementation. Ensuring secure data transmission, storage, and access is essential to protect sensitive information from unauthorized access or data breaches. Proper authentication and authorization mechanisms need to be in place to control user access to the tool and its functionalities.
5. Scalability: As the tool is designed to work in computer networks, scalability becomes crucial. It should be able to handle a growing number of connected systems, user registrations, and data entries without compromising performance.
6. User Experience: Developing an intuitive and user-friendly interface is important for user adoption and satisfaction. The tool should provide clear instructions, error handling, and meaningful feedback to guide users through the authentication and data collection processes.
7. Maintenance and Updates: The tool needs to be regularly updated to address security vulnerabilities, compatibility issues, and to incorporate new features or improvements. Proper maintenance and support must be provided to ensure the tool remains functional and up-to-date.

### ASSUMPTIONS AND CONSTRAINTS

* 1. Assumptions
     1. The system must predict whether a software is authentic or not.
     2. The device running the software must have python installed on it.
  2. Constraints
     1. This system only works on Unix and Windows based systems.

### ALTERNATIVES

There are currently no software’s that can check the authenticity of an application after installation which is a key feature of Nexus. Though there are software’s that can detect unauthentic software’s while installation, Nexus allows users to check the authenticity once installation is complete and it also allows the admin to take monitor and notify the users on the same so that they can rectify the situation.

# Use Case Analysis Document



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

**USE CASE TEMPLATE**

|  |  |  |
| --- | --- | --- |
| USE CASE | Checking the authentication of all the installed software’s on a device | |
| **Goal** | To detect if a software is authentic/notarized or not | |
| **Purpose** | To develop a system that can efficiently monitor the authenticity of applications on a particular device within a computer network | |
| **Preconditions** | The device running the software must have python installed on it. | |
| **Success Condition** | Software is able to detect the authenticity of applications installed successfully. | |
| **Failed Condition** | Software is not fully able to detect the authenticity of applications installed. | |
| **Primary Actors** | Admin and node machine | |
| **Secondary Actors** | Active network connection, TCP/IP Socket and liveliness of hosted API | |
| **Trigger** | Logging in/Signing up onto the VIT/any computer network | |
| **DESCRIPTION** | **Step** | **Basic Course of Action** |
|  | 1 | Log In/Sign Up |
|  | 2 | Pushing the data to the hosted API |
|  | 3 | Running the code to identify authentication of application |
|  | 4 | Connecting all the modules to a common GUI |
|  | 5 | Check the result and notify the users in case of any discrepancies |

# Software Requirements Specification



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

1. **SCOPE**

The Nexus project focuses on providing authentication and data collection capabilities for software running on Windows and Unix machines. It includes features for user registration, login, role-based access control, and GUI interfaces for user interaction.  
The project also covers the development of an API using the Django REST Framework, which allows external systems to interact with Nexus programmatically. Additionally, the project includes the creation of Python scripts to facilitate data collection from connected systems.

However, it's important to note that the Nexus project does not aim to address every aspect of software management or provide extensive monitoring and analysis functionalities. While it collects data from computer systems, the project does not delve into in-depth analysis or reporting capabilities.

The scope of the Nexus project primarily revolves around authentication, data collection, and basic user management within computer networks. It focuses on providing a reliable and user-friendly tool that simplifies software authentication processes and facilitates data collection from connected Windows and Unix machines.

Future enhancements and scalability considerations may involve expanding compatibility to other operating systems, incorporating advanced analysis features, and integrating with additional network components. However, these aspects are outside the current scope of the Nexus project and may be considered for future development and iterations.

### REQUIREMENTS

1. Python3: Python3 is a widely used high-level programming language known for its simplicity and readability. It provides a versatile and extensive standard library, making it suitable for various applications, including web development, data analysis, and scripting.
2. Django: Django is a powerful and popular Python web framework that follows the model-view-controller (MVC) architectural pattern. It simplifies web development by providing built-in features for database management, URL routing, user authentication, and template rendering, allowing developers to build robust and scalable web applications efficiently.
3. REST Framework: Django REST Framework is a toolkit that extends Django's capabilities to build RESTful APIs. It provides a set of powerful tools and libraries to serialize and validate data, handle authentication and permissions, and manage API endpoints. REST Framework simplifies the process of building APIs in Django and promotes the development of scalable and maintainable web services.
4. Shell Scripting: Shell scripting involves writing scripts in a command-line interpreter (shell) to automate tasks and execute commands on a computer system. It provides a convenient way to interact with the operating system, execute commands, perform file operations, and combine multiple commands into scripts for improved efficiency and automation.
5. PyQt5: PyQt5 is a Python binding for the Qt toolkit, which is a powerful framework for building graphical user interfaces. PyQt5 allows developers to create cross-platform desktop applications with rich features and a modern look and feel. It provides a wide range of widgets, layouts, and event handling mechanisms, making it a popular choice for GUI development in Python.
6. SQLite3: SQLite3 is a lightweight and embedded relational database management system. It provides a simple and efficient way to store, query, and manipulate data within a single file, making it ideal for small to medium-sized applications. SQLite3 is widely used and supported, including within the Python ecosystem, and offers ACID-compliant transaction support.
7. Qt5 Designer: Qt5 Designer is a graphical user interface (GUI) design tool provided with the Qt framework. It allows developers to visually design and create interfaces by dragging and dropping widgets, defining layouts, and connecting signals and slots. Qt5 Designer helps streamline the GUI development process and promotes rapid prototyping and efficient UI design.
8. Heroku: Heroku is a cloud platform that enables developers to deploy, manage, and scale applications easily. It supports various programming languages and frameworks, including Python and Django. Heroku provides a straightforward way to host web applications, handle database management, and automate deployment processes, allowing developers to focus on building and delivering their applications.
9. GitHub: GitHub is a web-based hosting platform for version control and collaboration. It allows developers to store, manage, and share their code repositories. GitHub provides features like version control, issue tracking, pull requests, and project management tools, facilitating collaborative development workflows. It is widely used in the software development community for code sharing, open-source contributions, and team collaboration.
10. asgiref: Provides compatibility between different versions of the ASGI specification.
11. autopep8: Automatically formats Python code to conform to the PEP 8 style guide.
12. boto3: Provides an interface to Amazon Web Services (AWS) services.
13. botocore: Provides low-level support for AWS services.
14. certifi: Provides a list of trusted certificate authorities.
15. charset-normalizer: Provides support for Unicode character normalization.
16. django-cors-headers: Provides a way to add cross-origin resource sharing (CORS) headers to Django responses.
17. django-multiselectfield: Provides a multi-select field for Django forms.
18. gunicorn: A web server that is often used to serve Django applications.
19. idna: Provides support for Internationalized Domain Names (IDNs).
20. jmespath: Provides a parser for the JMESPath query language.
21. pycodestyle: Checks Python code for style violations.
22. python-dateutil: Provides extensions to the standard Python datetime module.
23. pytz: Provides support for time zones.
24. requests: Provides an easy-to-use interface to the HTTP protocol.
25. s3transfer: Provides a high-level interface to Amazon S3.
26. six: Provides compatibility between different versions of Python.
27. sqlparse: Parses SQL statements.
28. tomli: Provides a parser for the TOML configuration file format.
29. typing\_extensions: Provides backported support for some new typing features in Python 3.9.
30. urllib3: Provides an easy-to-use interface to the URL library.
31. whitenoise: Provides middleware for serving static files over HTTPS.
32. Windows SDK: The Windows SDK (Software Development Kit) is a collection of tools, libraries, and documentation provided by Microsoft for developing applications specifically for the Windows operating system. It includes a set of development tools, compilers, headers, and libraries that allow developers to create Windows applications using various programming languages, such as C++, C#, and Visual Basic. The Windows SDK provides APIs (Application Programming Interfaces) that enable developers to access and utilize the functionality of the Windows platform, including features like user interface components, file system operations, networking capabilities, and more. It also includes debugging and profiling tools to aid in the development and troubleshooting process.
33. Windows Sign Tool: The Windows Sign Tool is a command-line utility provided by Microsoft as part of the Windows SDK. It allows developers and software publishers to digitally sign their executable files and libraries to verify their authenticity and integrity. Digital signing is a process that involves adding a digital signature to a file using a private key, which can be verified by the recipient using the corresponding public key. By signing their files, developers can provide a level of assurance that the software has not been tampered with and that it originates from a trusted source. The Windows Sign Tool supports various signing algorithms and certificates, and it can also timestamp signed files to ensure their validity even after the certificate has expired. Digital signing is particularly important for distributing software securely and ensuring that users can trust the software they install on their Windows systems.

# Software Project Plan



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

1. **OVERVIEW**

We embraced the Agile software development methodology for our project, utilizing a sprint duration of one week. At the conclusion of each week, we conducted code reviews and merged all the code implemented during the sprint. Additionally, we engaged in planning sessions to outline the upcoming tasks for the subsequent week, ensuring a continuous and iterative development process.

### PROJECT GOALS

The main goal of the project was to build A Software Authentication Checker Tool for Computer Networks that will check if applications on a device are authentic or not. The Nexus project proposes a comprehensive software authentication checker tool for computer networks. By leveraging Python, the project integrates an API, GUIs, and scripts to address the challenges of software authentication and data collection. The solution offers a user-friendly interface for user registration, login, and data retrieval, allowing for seamless authentication and data gathering from Windows and Unix machines. The collected data is securely stored in a database, enabling efficient software management. With its holistic approach, the Nexus tool aims to enhance security, streamline software authentication, and improve overall efficiency in computer network management

|  |  |  |
| --- | --- | --- |
| **Project Goal** | **Priority** | **Comment/Description/Reference** |
| **Business Goals:** |  |  |
| Build the Software | 3 | Deploying the API and the software so that it can be readily integrated into any computer network |
| **Technological Goals:** |  |  |
| Fetching information of a device and figuring out the authenticity of applications | 1 | Writing appropriate codes in python along with shell scripting to fetch data from each of the individual devices on the network |
| Deploying the API | 2 | Using the Heroku GitHub pipeline to deploy the API to store the data received from the devices |
| Building GUI | 3 | Developing the front-end UI of the application |

### LITERATURE REVIEW

1. In the thesis paper (Software Licensing Analysis Tool, Tomáš Radej, 2013 ([link](https://is.muni.cz/th/324820/fi_m/fithesis.pdf))), Tomáš Radej has explained about the problems faced for detection and scanning of open licenses.
2. In the research paper (Comparison of Open Source License Scanning Tools, Hailing Zhang ([Link](https://www.diva-portal.org/smash/get/diva2:1463853/FULLTEXT01.pdf))), authors have provided insights for popular FOSS (Free and Open Source Software) license scanning tools.
3. In the paper (Automated software license analysis, [Timo Tuunanen](https://link.springer.com/article/10.1007/s10515-009-0054-z), [Jussi Koskinen](https://link.springer.com/article/10.1007/s10515-009-0054-z) & [Tommi Kärkkäinen](https://link.springer.com/article/10.1007/s10515-009-0054-z), 2009 ([link](https://link.springer.com/article/10.1007/s10515-009-0054-z))), authors have described an automated approach for OSS(Open source software) license analysis.
4. In the research paper (The Study of Digital Signature Authentication Process, Unnati Patel, Ashaben Patel, Falguni Suthar, Acharya Motibhai Patel ([Link](https://www.researchgate.net/publication/336603564_THE_STUDY_OF_DIGITAL_SIGNATURE_AUTHENTICATION_PROCESS))), authors have provided insights for digital signature technology for software authentication.
5. In the research paper (A survey on Digital Signatures, Rabeya Sultana, Tashrifa Shahid (Link)), authors have provided insights into the importance of digital signatures in software used in various technologies like cyber security, finance, etc.

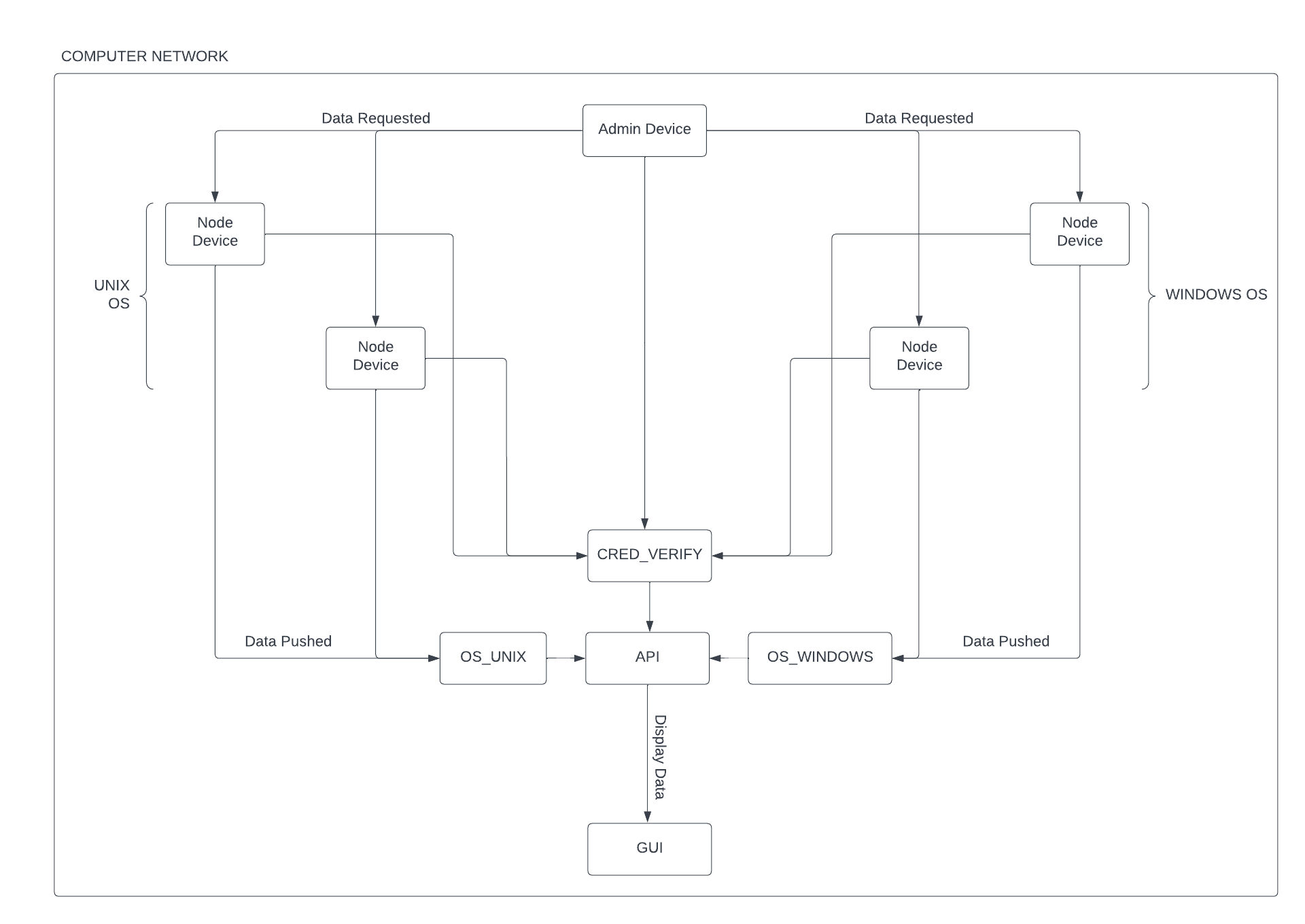
# System Implementation Document



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

1. **SYSTEM ARCHITECTURE**



##### Fig. 1: System Architecture

The Nexus system follows a modular architecture that consists of various components working together to provide software authentication and data collection capabilities for computer networks. It comprises an API (mjrprj\_api) developed using Django REST Framework, graphical user interfaces (GUI) built with PyQt5, and additional Python scripts.  
  
**Component Overview**

1. API: The API component serves as the backend for the Nexus tool, providing endpoints for user authentication, data collection, and software verification. It is built using Django REST Framework, which offers a robust framework for creating RESTful APIs.
2. Database (db.sqlite3): The Nexus tool utilizes the SQLite3 database to store user authentication details, collected data from computer systems, and software verification results. The database structure is designed to ensure efficient data retrieval and management.
3. GUI: The GUI component utilizes the PyQt5 library to develop user-friendly interfaces for user registration, login, and data retrieval. It enables users to interact with the Nexus tool efficiently and intuitively.
4. Node Device: All devices apart from the admin devices on a computer network
5. Admin Devices: The central device in the network, this device can be used to fetch data from the node devices
6. CRED\_VERIFY: The cred\_verify app within Nexus focuses on storing and managing user authentication information. It includes models, views, and serializers that facilitate the registration, login, and role-based access control functionalities.
7. OS\_WINDOWS: The os\_windows app handles the collection of data from Windows machines. It utilizes the Windows SDK and Shell Scripting to gather relevant system information, which is then stored in the database for further analysis.
8. OS\_UNIX: The os\_unix app is responsible for collecting data from Unix machines. It employs shell scripting techniques to retrieve system details and securely stores the collected information in the database for subsequent processing.

### IMPLEMENTATION

* 1. User Authentication and Authorization
     1. User Registration: The Nexus tool provides a user registration feature that allows new users to create accounts. Users are required to provide necessary information and choose their desired roles during the registration process.
     2. User Login: Registered users can log in to the Nexus tool using their credentials. The login mechanism verifies the provided information and grants access to the tool's functionalities based on the user's assigned roles and permissions.
     3. User Roles and Permissions: Nexus incorporates role-based access control, allowing users to have different levels of access and permissions within the system. Roles such as admin, user, and guest are defined, each with specific privileges. Admins have full access to all functionalities, including user management and data retrieval. Users have limited access based on their assigned roles, while guests have restricted access for basic interactions.
  2. Graphical User Interface
     1. GUI Signup: The GUI Signup interface provides a user-friendly form for new users to register their accounts. It prompts users to enter their details and choose their desired roles during the registration process. The interface ensures a smooth and intuitive user experience.
     2. GUI User Signup: The GUI User Signup interface allows admin users to create new user accounts. It includes fields for entering user information and selecting appropriate roles. This interface streamlines the user creation process, making it convenient for administrators.
     3. GUI Login: The GUI Login interface enables users to securely log in to the Nexus tool. It presents a login form where users can enter their credentials. Upon successful authentication, users gain access to the tool's functionalities based on their assigned roles.
     4. Admin Panel: The Admin Panel interface provides admins with a centralized dashboard to manage user accounts, roles, and permissions. It offers a comprehensive overview of the system's user base and facilitates efficient user management operations.
  3. Data Collection and Verification
     1. Collecting Windows Machine Data: The Nexus tool includes functionality to collect comprehensive data from Windows machines. It leverages the Windows SDK and Shell Scripting to extract system information such as hardware specifications, installed software, and network configurations. This data is essential for software authentication and network analysis.
     2. Collecting Unix Machine Data: For Unix machines, the Nexus tool utilizes shell scripting techniques to gather relevant system details. It retrieves information regarding hardware, software, processes, and network settings. The collected data contributes to software verification and system analysis within computer networks.
     3. Authenticating Software: The software authentication feature of Nexus verifies the authenticity of installed software on the connected machines. It compares software signatures and checks for known vulnerabilities or unauthorized versions. The authentication process helps identify potential security risks and ensures only authorized software is running.
     4. Sending Email Notifications: To keep users informed, Nexus incorporates email notification functionality. It sends notifications regarding successful user registration, important updates, and verification results. This feature improves communication and enhances user experience within the system.
  4. Deployment and Hosting:  
     1. Hosting the API (Heroku): The Nexus API is hosted on the Heroku platform, providing a scalable and reliable hosting solution. Heroku allows seamless deployment and ensures the availability of the API to users accessing the Nexus tool from various locations.
     2. Repository Hosting (GitHub): The source code for the Nexus project is hosted on GitHub, a widely used platform for version control and collaboration. GitHub enables effective code management, facilitates team collaboration, and provides a platform for community contributions and issue tracking.

### DEVELOPMENT AND TESTING

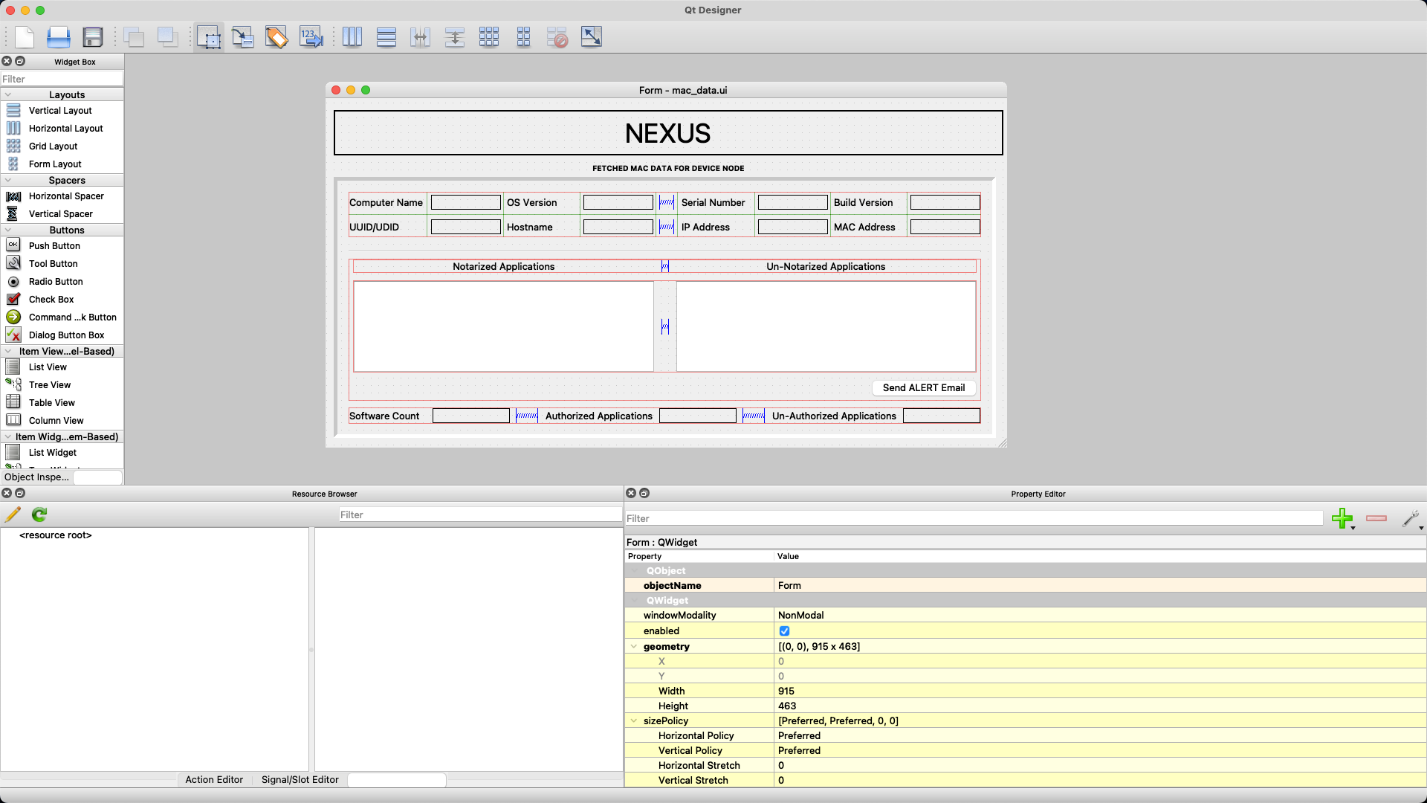
The development of the Nexus project followed the Agile methodology, which allowed for iterative and incremental development. The team planned sprints of one week each, where they identified tasks, implemented features, and resolved issues within the designated time frame. During the development phase, the team focused on implementing the various components of the Nexus tool, including the API, GUI, and scripts. We utilized Python3 as the primary programming language and integrated frameworks and libraries such as Django, Django REST Framework, PyQt5, and SQLite3.

Django/Django REST Framework:



##### Fig. 2: mjrprj-api created using Python & Django REST Framework

PyQT5:



##### Fig. 3: GUI designing using Qt5 Designer

Shell Scripting:



##### Fig. 4: Shell script to get details for UNIX systems

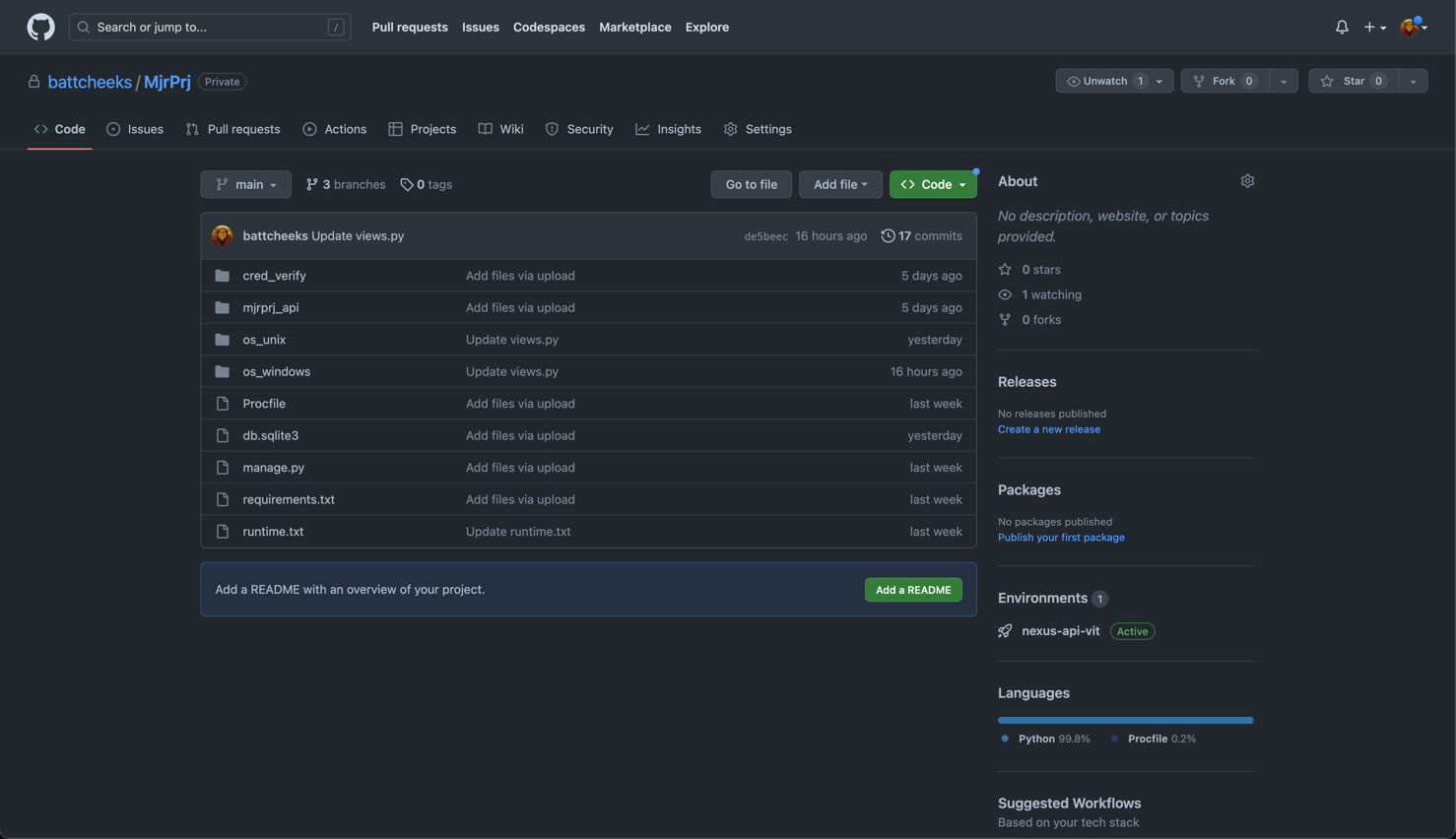
##### ../Desktop/47934d6c-faa4-40ee-ac7e-53a9ab4b5bc2.jpeg

##### Fig. 5: Scripting to get details for Windows systems

##### The development process involved designing and implementing the system architecture, defining the database structure, creating user interfaces, implementing authentication and authorization mechanisms, and integrating the necessary functionalities for software authentication, data collection, and user management.

Version control using Git and GitHub facilitated collaboration among the development team members. Regular code reviews, testing, and debugging were performed to ensure the quality and functionality of the codebase. Continuous integration practices were employed to automate the build and testing process, ensuring that the codebase remained stable and reliable throughout development.

Django REST API Code Hosted on GitHub:



##### Fig. 6: Django API Codes saved in a repository on GitHub

##### App created on Heroku for API Deployment:

##### /var/folders/9h/z85h8z095w5f2136_b467_gm0000gn/T/TemporaryItems/NSIRD_screencaptureui_lI4Bp6/Screenshot 2023-05-13 at 5.30.00 AM.png

##### Fig. 7: Heroku app created for API Deployment

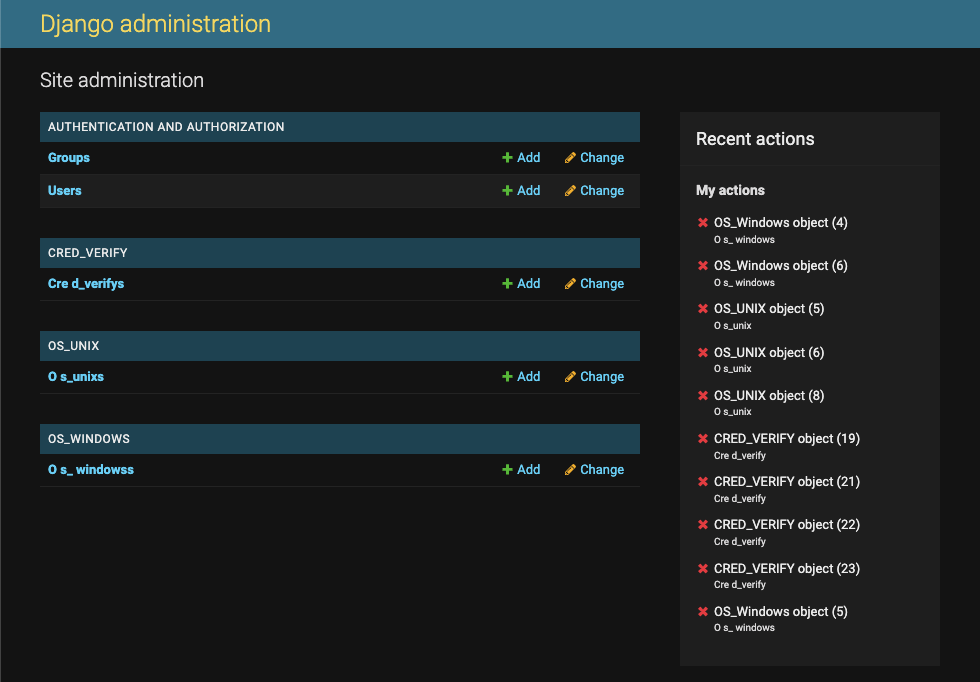
### RESULTS

NEXUS App created:

****

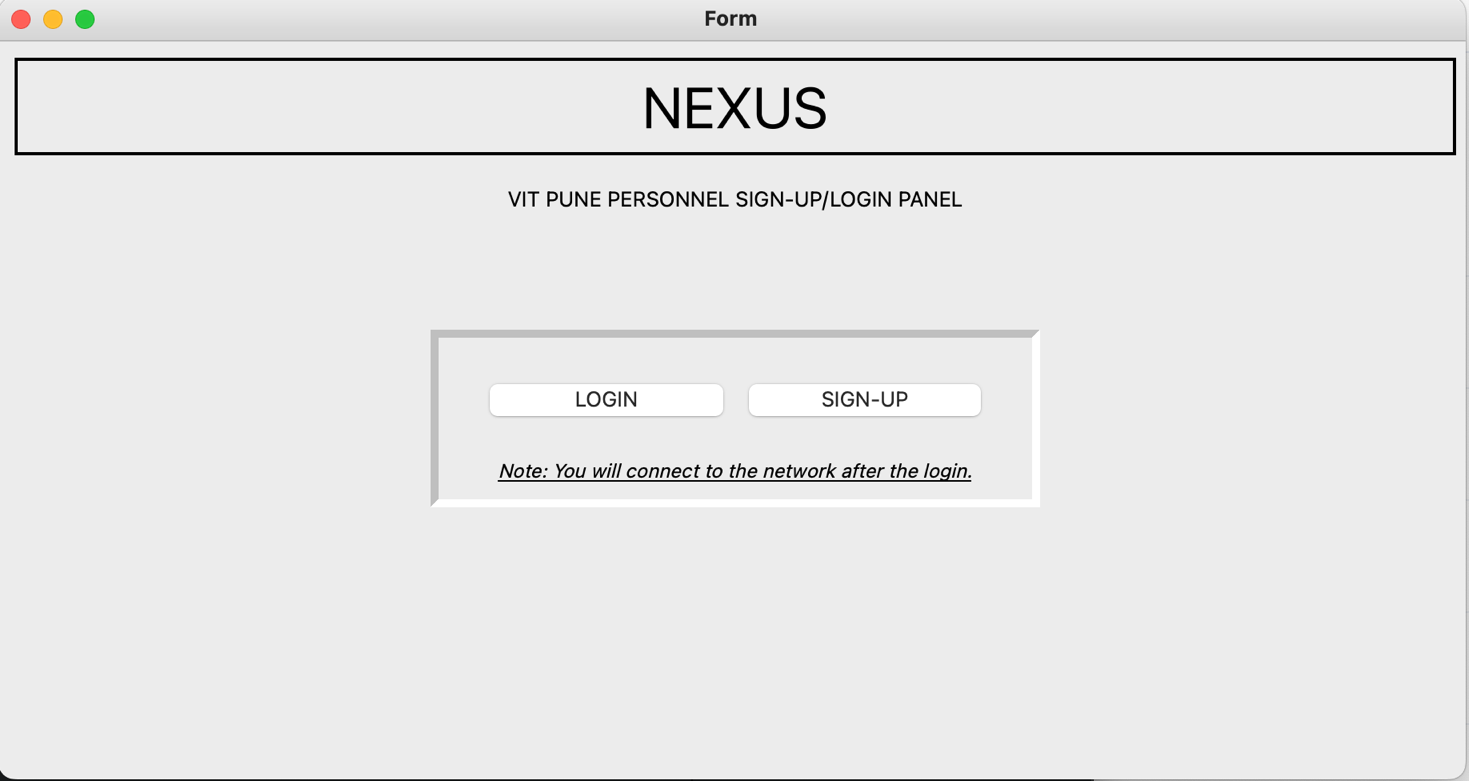
##### Fig. 8: Nexus Application created

API Hosted on Heroku:



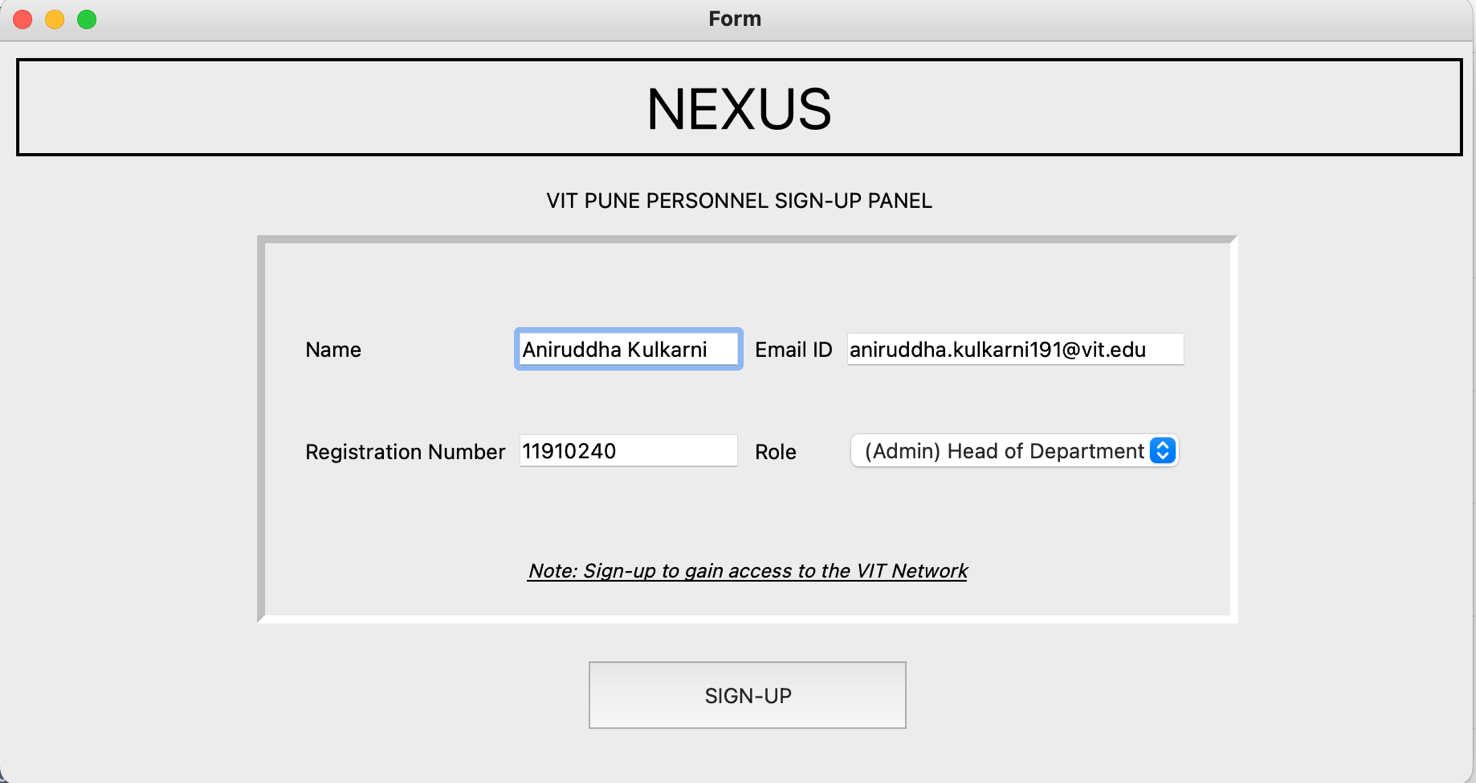
##### Fig. 9: API hosted using GitHub - Heroku pipeline

GUI displayed on application opening:



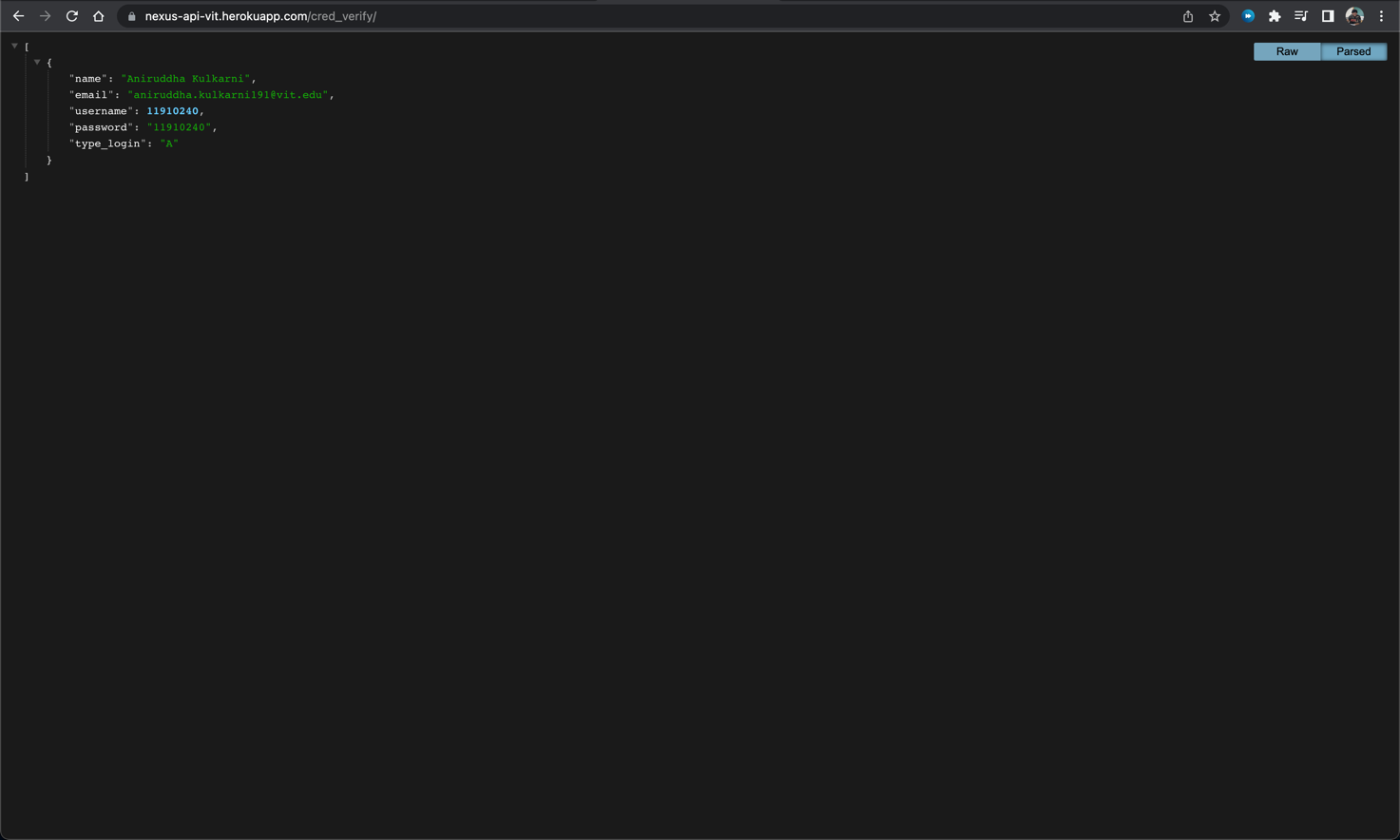
##### Fig. 10: GUI displayed when Nexus app is opened for user to log in or sign-up into the system.

GUI displayed for user sign-up:



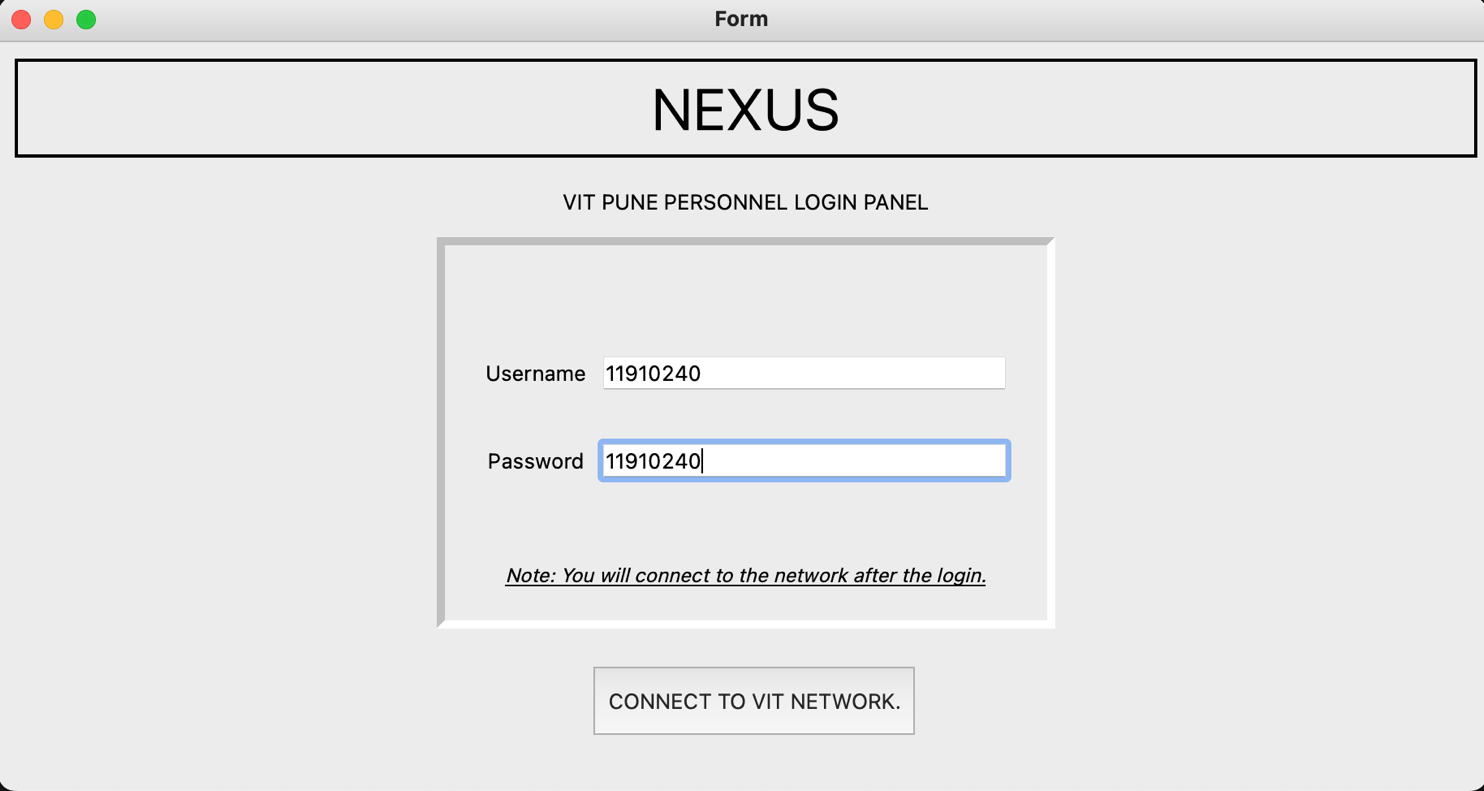
##### Fig. 11: Signing up into the software

New user data pushed to the API



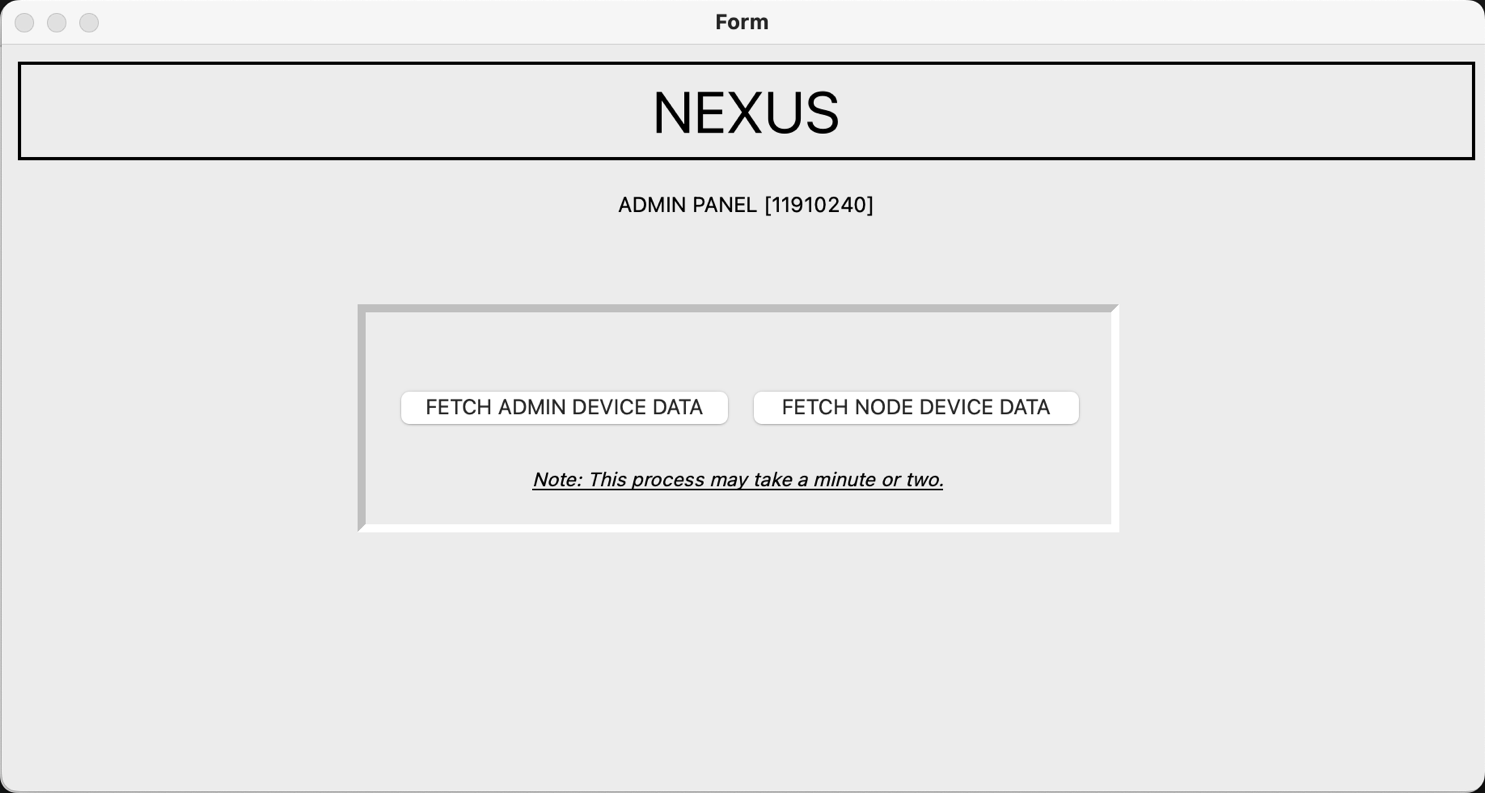
##### Fig. 12: Pushing new user data to the API

GUI Displayed to log into the system:



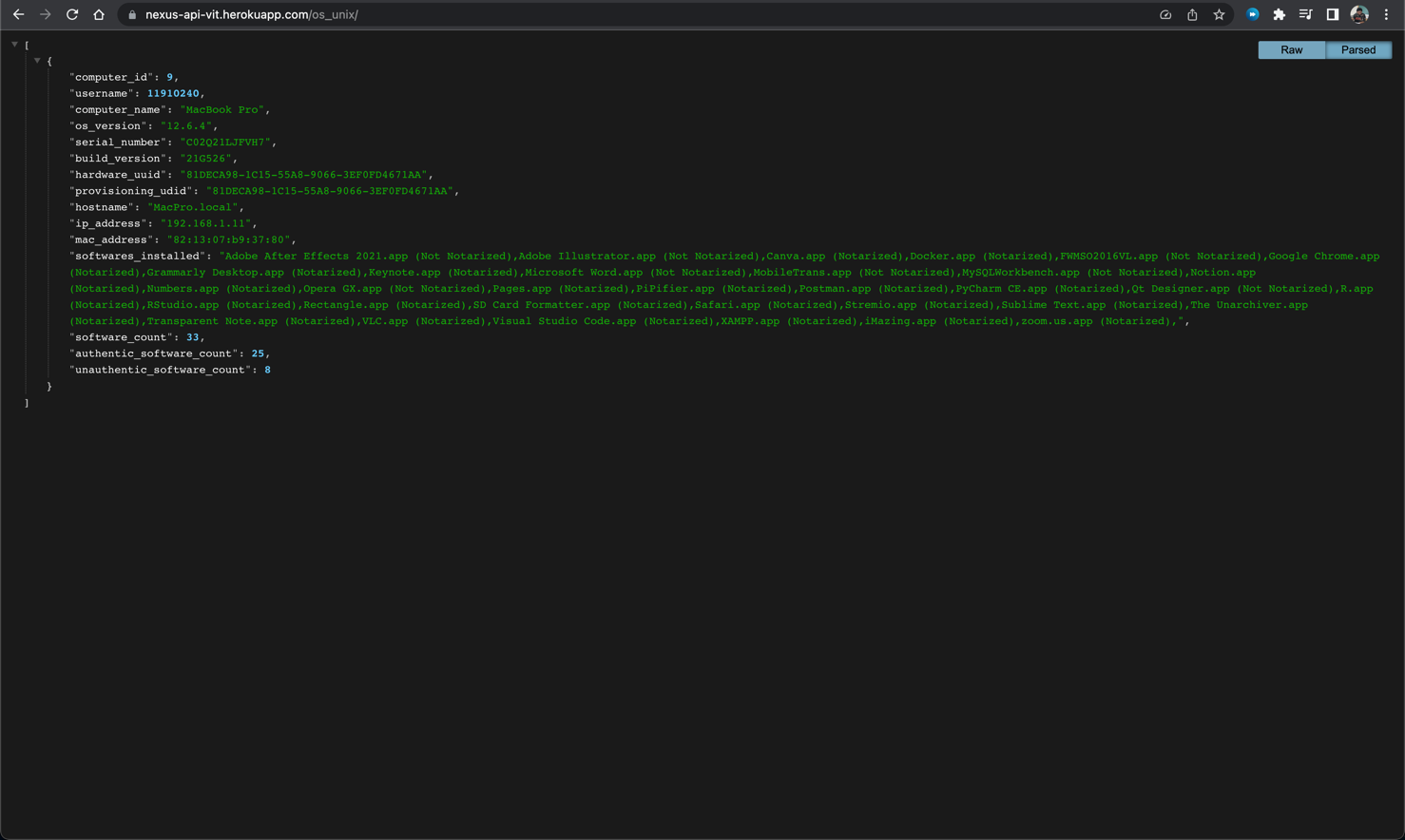
##### Fig. 13: Logging in into the system using credentials

Admin Panel GUI displayed to fetch data:



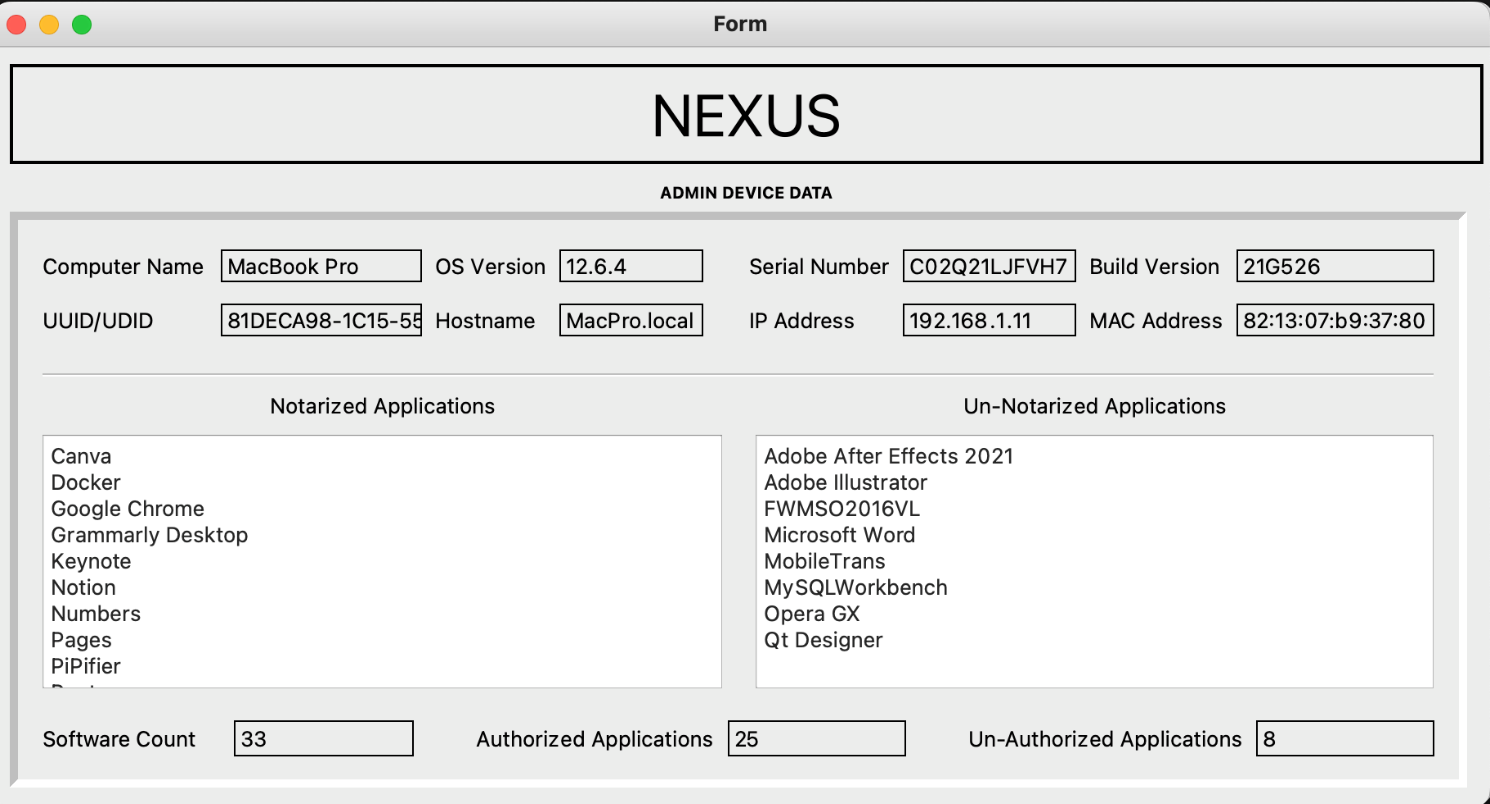
##### Fig. 14: Admin panel GUI generated when admin logs into the system.

Data pushed to the API when admin device data is requested:



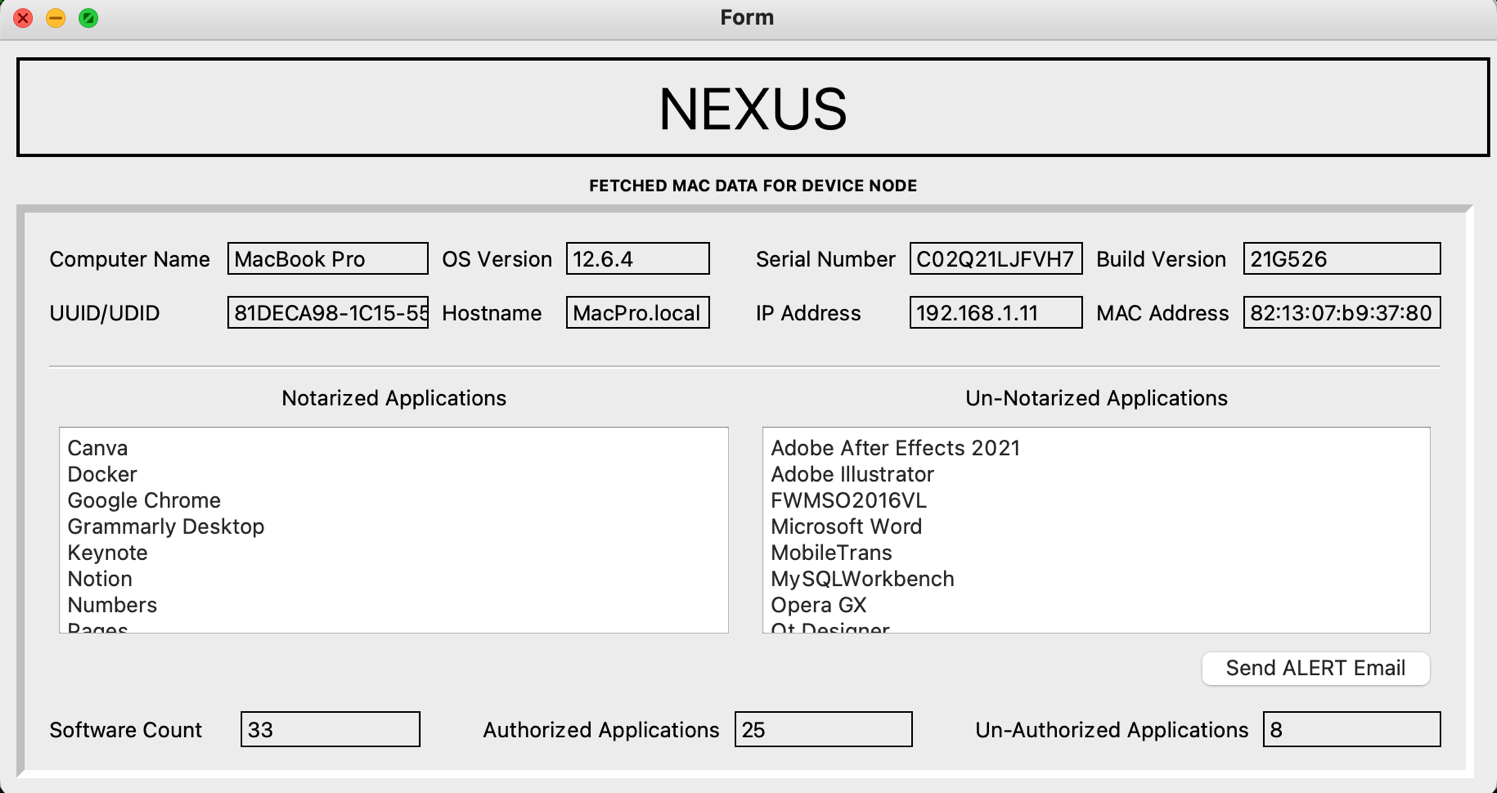
##### Fig. 15: UNIX Data pushed into the API

Admin Device Data displayed in GUI:



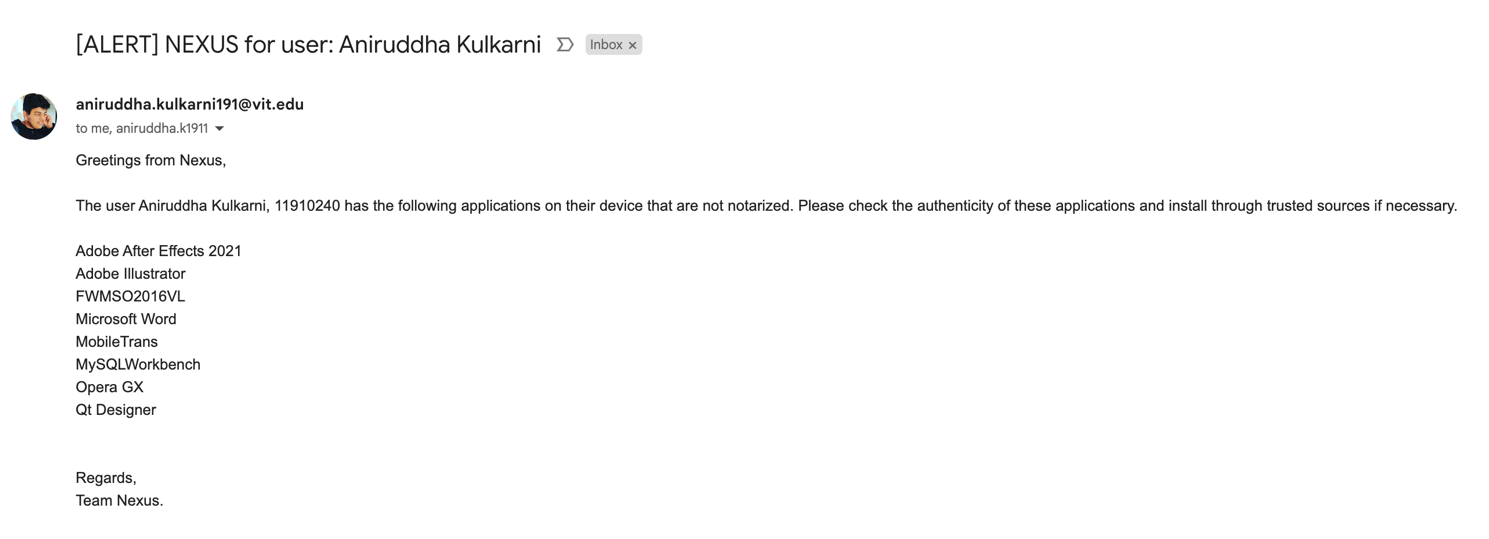
##### Fig. 16: GUI displaying the data received for the admin device from the API

Node Device Data displayed in GUI:



##### Fig. 17: GUI displaying data received for device node.

Alert Email sent to the user:

****

##### Fig. 18: Mail notification chain sent to user and admin to notify of un-notarized applications.

### CHALLENGES

During the development process, several challenges were encountered, including handling different operating systems, ensuring compatibility with various software versions, and managing user roles and permissions effectively. Overcoming these challenges required thorough testing, robust error handling, and continuous improvement.

### FUTURE SCOPE

1. Integration with Software Management Systems: Integrate Nexus with existing software management systems or version control tools to streamline the authentication and data collection processes.
2. Advanced Data Collection: Expand the data collection capabilities to gather more detailed information from computer systems, including hardware specifications, network configurations, and software dependencies.
3. Integration with Cloud Services: Integrate Nexus with popular cloud services like AWS, Azure, or Google Cloud Platform to enable seamless authentication and data collection in cloud-based environments.
4. Real-time Monitoring and Alerts: Implement real-time monitoring of software authenticity and system data, allowing administrators to receive immediate alerts in case of any unauthorized access or suspicious activities.
5. Machine Learning and Anomaly Detection: Utilize machine learning algorithms to analyze collected data and identify anomalies or potential security breaches. This can help in proactive detection and prevention of security incidents.

By focusing on these future developments, Nexus can evolve into a more powerful and comprehensive software authentication checker tool, providing enhanced security, usability, and efficiency in computer networks.

### CONCLUSION

In conclusion, the Nexus project has successfully developed a comprehensive software authentication checker tool for computer networks. By adopting the Agile methodology, we were able to efficiently plan and execute the development process, resulting in a functional and user-friendly tool.

Throughout the project, we achieved the goals of providing software authentication, data collection, and user management functionalities. The role-based access control system ensures secure access and proper authorization for users. The integration of GUI interfaces and API with Django and PyQt5 enhances the usability and flexibility of the tool.

The system architecture and component overview provide a clear understanding of the Nexus tool's structure and organization. The database storage and app modules facilitate efficient data management and storage. The graphical user interfaces facilitate seamless user interaction and streamline the authentication and data collection processes.

The development and testing phases ensured the quality and reliability of the Nexus tool. Thorough testing, including unit testing and integration testing, helped identify and address any issues or bugs.

The future development possibilities discussed in this report offer opportunities for further enhancement and expansion of the tool's capabilities.

In conclusion, the Nexus project has addressed the challenges in software authentication and data collection in computer networks. It provides a robust solution that improves security, efficiency, and usability in software management. The successful implementation of the Nexus tool lays a strong foundation for future advancements and improvements in the field of software authentication and network security.

### REFERENCES

.

1. [https://codesigningstore.com/how-to-digitally-sign-executable- files#:~:text=Executable%20signing%20certificates%2C%20commonly%20referred,executable%20file's%20integrity%20and%20authenticity](https://codesigningstore.com/how-to-digitally-sign-executable-%20files#:~:text=Executable%20signing%20certificates%2C%20commonly%20referred,executable%20file's%20integrity%20and%20authenticity)
2. <https://www.ibm.com/docs/SSER83_8.1.0/erp.db2/c_dperp_d_digitalsigning.html>
3. <https://developer.microsoft.com/en-us/windows/downloads/windows-sdk/>
4. <https://learn.microsoft.com/en-us/windows/win32/seccrypto/signtool>
5. <https://codesigningstore.com/what-is-microsoft-signtool-how-to-use-it>
6. <https://www.xtivia.com/blog/common-causes-of-software-license-noncompliance/>
7. <https://www.apple.com/legal/sla/docs/macOS1012TouchID.pdf>
8. <https://realpython.com/django-hosting-on-heroku/>