Tribhuvan University

Institute of Science and Technology

Bhaktapur Multiple Campus Dudhpati, Bhaktapur

Internship Report

On

"Password Reset Bot"

at

QuickFox Consulting

Under the Supervision of

Asst. Prof. Surya Bam

Submitted By:

Radil Koju (20254/075)

Submitted To:

Institute of Science and Technology

Tribhuvan University

In Partial Fulfillment of the Requirement for

Bachelor's Degree

In

Computer Science and Information Technology

November, 2023

Tribhuvan University Institute of Science and Technology

BHAKTAPUR MULTIPLE CAMPUS

Dudhpati, Bhaktapur

SUPERVISOR'S RECOMMENDATION

I hereby recommend that the report prepared under my supervision by **Radil Koju (TU Exam Roll No. 20254/075)** entitled **"Password Reset Bot**" at QuickFox Consulting in partial fulfilment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for evaluation.

……..…….

Asst. Prof. Surya Bam

Supervisor

Bhaktapur Multiple Campus

Dudhpati, Bhaktapur

Tribhuvan University Institute of Science and Technology

BHAKTAPUR MULTIPLE CAMPUS

Dudhpati, Bhaktapur

CERTIFICATE OF APPROVAL

This is to certify that this project prepared **Radil Koju (TU Exam Roll No. 20254/075)** entitled **"Password Reset Bot"** at QuickFox Consulting in partial fulfilment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as an internship for the required degree.

Asst. Prof. Surya Bam
Supervisor
Bhaktapur Multiple Campus
Dudhpati, Bhaktapur

Mr. Sushant Paudel
Head Of Department,
Bhaktapur Multiple Campus
Dudhpati, Bhaktapur

External Examiner

ACKNOWLEDGEMENTS

I am delighted to take this opportunity to extend my heartfelt gratitude to my esteemed mentor, Mr. Prashanna Nepal, for his unwavering guidance and support during my internship journey. His patience, enthusiasm, and continuous motivation have been instrumental in shaping me into a better version of myself with each passing day. My sincere appreciation goes out to the entire team **QuickFox Consulting**. for their unyielding support, encouragement, and guidance that not only bolstered my strengths but also helped me address and improve upon my weaknesses.

I would also like to extend my deep gratitude to my dedicated supervisor, **Asst. Prof. Surya Bam**, whose guidance and motivation have been pivotal throughout my internship. I am equally thankful to **Mr. Sushant Poudel** for his consistent support and invaluable guidance that facilitated the successful completion of my project.

Furthermore, I wish to express my sincere appreciation to the staff and faculty at Bhaktapur Multiple Campus for their continuous support and cooperation during this entire journey. My gratitude extends to my friends and colleagues whose direct or indirect contributions played a significant role in the culmination of this project.

Sincerely,
Radil Koju
(20254/075)

ABSTRACT

During my internship, I actively contributed to the development of an innovative Robotic Process Automation (RPA) project within a company specializing in RPA solutions for the banking and telecom industries, alongside website design and front-end and back-end development. The focal point of my internship involved the creation and refinement of the "Password Reset Bot." This RPA bot was designed to streamline the password reset process for help desk operations. The bot effectively addressed user requests for password resets with a high degree of efficiency and security, thanks in part to the utilization of the Selenium library.

This internship project not only allowed me to gain hands-on experience with cutting-edge RPA technologies but also provided valuable insights into automation strategies tailored to the specific needs of banking and telecom systems. Through this project, I improved my skills in RPA development, security, user interaction, and leveraging Selenium for web automation. These skills contributed to the company's mission of delivering efficient and secure automation solutions.

Keywords: Robotic process automation (RPA), Selenium Library

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
API	:	Application Programming Interface
CEO	:	Chief Executive Officer
CSS	:	Cascading Style Sheets
CV	:	Computer Vision
ECC	:	Electronic Cheque Clearing
HTML	:	Hyper Text Markup Language
IDE	:	Integrated Development Environment
IPS	:	Interbank Payment System
JS	:	JavaScript
ML	:	Machine Learning
NCHL	:	Nepal Clearing House Ltd
PAN	:	Personal Account Number
RPA	:	Robotic Process Automation
SQL	:	Structured Query Language

CHAPTER 1

INTRODUCTION

1.1 Introduction

The "Password Reset Bot" is a Robotic Process Automation (RPA) system designed to assist bank employees in resetting their passwords without the need to contact the IT support team. This tool holds significant value for both the bank's staff and the IT department, as it saves time and minimizes frustration for all involved parties. For banks, the password reset bot serves as a solution to alleviate the workload on IT support teams, allowing IT representatives to allocate their resources towards more complex issues. Moreover, this automation tool enhances employee satisfaction by offering a quick and hassle-free method to reset passwords.

The opportunity to intern at QuickFox Consulting from April 3, 2023, to June 5, 2023, provided an enriching experience and enabled the author to make an active contribution to the practical application of skills. During this internship at QuickFox Consulting, the author's involvement in the "Password Reset Bot" project was a pivotal part of their tenure. In the current landscape, where cyber threats pose a significant concern, this project emerged as an innovative endeavor aimed at streamlining and improving the password reset process.

1.2 Problem Statement

In the traditional banking landscape, a burdensome process existed where bank staff had to initiate password reset requests for various banking applications. These requests, once initiated, were then forwarded to the IT teams for resolution. The IT teams, in turn, had to manually reset passwords for each individual request. This outdated approach not only consumed valuable time and resources but also introduced the potential for human errors and security vulnerabilities.

Recognizing the need for a more efficient and secure solution, our team embarked on a transformative journey to automate the password reset process. By automating this critical aspect of our operations, we aimed to eliminate the bottlenecks associated with manual resets, enhance security, and empower bank staff with a streamlined and user-friendly experience.

1.3 Objective

The aim of this system is to automate the repetitive manual task increasing efficiency and accuracy. The objectives of this project are:

- To create a password reset bot
- To reduce the workload on IT support teams

1.4 Scope and Limitation of the Project

The scope and limitation of the project are as follows:

1.4.1 Scope:

- Password Reset Automation: The primary scope of the bot is to automate the password reset process, making it quicker and more efficient for users and reducing the burden on IT teams. This can encompass various aspects, including generating temporary passwords, sending reset links, and ensuring password policy compliance.

1.4.2 Limitations:

- Dependency on User Information: The bot typically relies on accurate user information and authentication methods for password resets. If user information is outdated or authentication methods are compromised, it can limit the bot's effectiveness.
- Security Risks: While the bot aims to enhance security, it can also introduce security risks if not implemented and maintained properly. For instance, if the bot's authentication process is weak, it may be vulnerable to exploitation.
- User Training: Users need to be educated about how to use the bot correctly, as well as the importance of maintaining the confidentiality of their authentication information.
- Monitoring and Maintenance: Continuous monitoring and maintenance are essential to ensure RPA bots remain optimized for their intended tasks.

1.5 Report Organization

The project report is structured into four main chapters. In Chapter One, titled "Introduction," it encompasses the problem statement, objectives, and the scope and limitations of the project. Chapter Two, "Organization Details," introduces the organization, its hierarchical structure, and

its working domain. Chapter Three, "Internship Activities," focuses on the practical aspects of the internship, including roles and responsibilities, a weekly log of activities, and descriptions of the projects undertaken during the internship. Chapter Four, "Conclusion and Learning Outcomes," provides a summative evaluation of the system and the lessons learned throughout the internship experience. Lastly, all the references are mentioned in the APA format. This structured framework ensures a comprehensive and informative documentation of the project and internship journey.

CHAPTER 2

ORGANIZATIONAL OVERVIEW AND LITERATURE REVIEW

2.1 Introduction

QuickFox Consulting, based in New Baneshwor, Kathmandu, Nepal, is a software development and consulting firm that emerged in 2019, driven by a team of seasoned professionals passionate about innovation and technology.

The flagship product of QuickFox Consulting is QuickRPA, a prominent Robotic Process Automation (RPA) platform. QuickRPA is widely adopted across diverse industries and organizations of varying sizes to streamline repetitive and time-consuming tasks. The company specializes in crafting RPA bots hosted within the QuickRPA platform, facilitating seamless interaction with an organization's systems for task automation. The development of these RPA bots leverages a range of technologies, including RobotFramework (RPAFramework), Python, AWS, and Docker.

Beyond its core RPA offerings, QuickFox Consulting provides a comprehensive suite of services encompassing:

- Software development
- IT consulting
- Business process automation
- Data analytics
- Cloud computing

QuickFox Consulting is a dynamic and rapidly expanding entity with a robust foothold in the Nepalese market. Additionally, it is actively extending its presence into neighboring regions and countries.

Table 2.1 Detail of the company

Company Name	QuickFox Consulting Pvt. Ltd.
Contact	+977 9865365130
Mail	hello@quickfoxconsulting.com
Website	https://quickrpa.ai/

QuickRPA, developed by QuickFox Consulting, is a versatile Robotic Process Automation (RPA) platform designed to assist organizations in automating repetitive and time-consuming tasks. This platform enables users to create and deploy bots using an RPA framework, making it a valuable resource for streamlining various processes. QuickRPA's applications span across a wide range of tasks, including data entry, order processing, customer service, and IT support.

What sets QuickRPA apart are its robust features, such as integration with artificial intelligence (AI) and machine learning (ML), orchestration capabilities, and centralized management and monitoring. These features empower organizations to automate complex workflows and enhance operational efficiency.

QuickRPA finds utility across industries, serving organizations of all sizes, including those in banking, finance, healthcare, insurance, and manufacturing. Its customers have reported significant benefits, with several real-world use cases highlighting its effectiveness. For instance, Ncell, a mobile network operator in Nepal, relies on QuickRPA to streamline customer onboarding and provisioning. NMB Bank and Himalayan Bank, both commercial banks in Nepal, use QuickRPA to automate back-office processes like loan processing and monthly loan interest updates. Additionally, Nabil Bank leverages QuickRPA to automate password resets for various applications used by its staff.

In summary, QuickRPA offers a powerful automation solution with a diverse range of applications, making it a valuable tool for organizations seeking to optimize their business processes and improve operational efficiency. Its successful implementations in various industries, as demonstrated by customer examples, underscore its effectiveness in driving productivity and reducing manual workloads.

2.2 Organizational Hierarchy

QuickFox Consulting has a hierarchical organization structure. It has different departments to facilitate the business process that have good coordination among each other which is shown in the figure below. CEO of QuickFox Consulting is responsible for directing over the operations and Business Development team is responsible for marketing and communicating with clients and Engineering Management team is responsible for managing the engineering team.

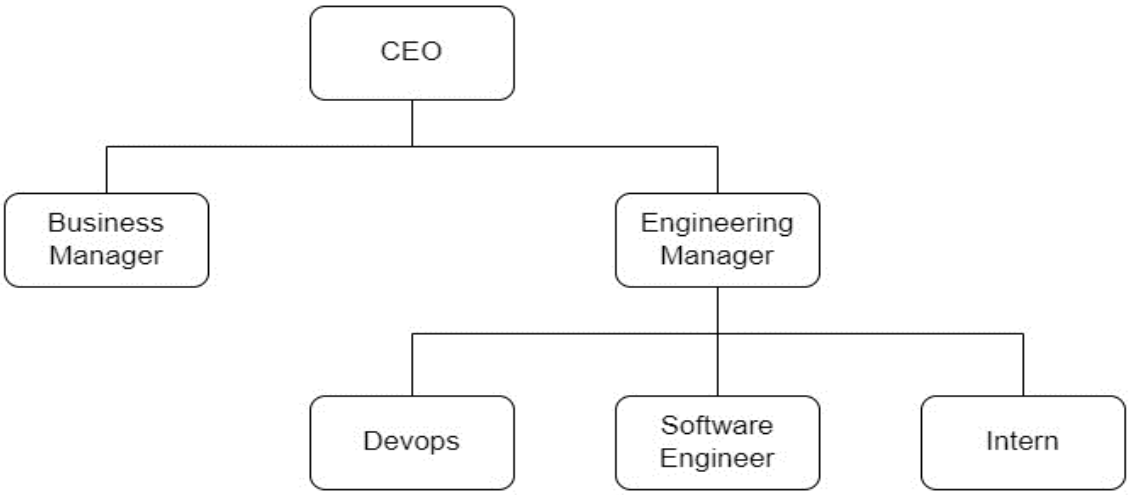

Figure 2.1Organizational Hierarchy

2.3 Working Domains of Organization

QuickFox Consulting provides multiple services in the area of Robotic Process automation, software development. The areas of expertise of the organizations are:

1. Robotic Process Automation
2. Software Development
    a. Python
    b. HTML/CSS/JS
    c. Mobile Application Development
3. Database Management
    a. MySQL
    b. Postgres
    c. Sqlite3
    d. Oracle
4. IT Consulting
5. Cloud Computing

2.4 Description of Intern Department

I worked for a total of 12 weeks as an intern at QuickFox Consulting. The details of my internship period in the organizations are summarizes in the following table:

Table 2.2 Internship Description

Start Date	3 April, 2023
End Date	5 June, 2023
Total Duration	12 Weeks
Position	Intern as Software Developer
Supervisor	Er. Prashanna Nepal
Office Hour	9:00 AM – 6: 00 PM

2.5 Literature Review

According to Ahuja and Tailor, RPA entered the vocabulary in organizations, in 2000, and its development began in 1990. Semantically, each word in Robotic Process Automation means: Robotic: systems that copy human behavior and perform tasks, Process: steps that lead to the fulfillment of a task, Automation: any task that's performed with assistance and not manually (Ahuja S, 2021)

Robotic Process Automation (RPA) stands out as a highly advanced technology encompassing computer science, electronics and communications, mechanical engineering, and information technology. It represents an emerging facet of business process automation, revolving around the concept of software robots and artificial intelligence (AI) workers. RPA has emerged as the contemporary lingua franca of the business world, showcasing its exceptional potency among the technologies of the 21st century. (Somayya Madakam, 2019)

UiPath is a notable RPA vendor dates back to 2005, in what was back then an outsourcing company. In response to a significant market demand, they recognized the necessity of developing an industry-standard platform dedicated to the training and orchestration of software robots, thereby embarking on this endeavor. (Ruchi Issac, 2018)

In 2001, a team of process automation specialists came together to establish Blue Prism with the aim of creating technology capable of enhancing the efficiency and overall effectiveness of various organizations. They directed their attention toward the white-collar back-office domain, identifying a substantial and unmet demand for automation solutions in this sector.(Ruchi Issac, 2018)

RPA provides dramatic improvements in accuracy and cycle time and increased productivity in transaction processing while it elevates the nature of work by removing people from dull, repetitive tasks. (Can Tansel KAYA, 2019)

RPA is an automation technology based on software tools that could imitate human behavior for repetitive and non-value-added tasks such as tipping, coping, pasting, extracting, merging and moving data from one system to another. The main benefits of RPA are cost reduction, increasing process speed, error reduction and productivity improvement. (Santiago Aguirre, 2017)

CHAPTER 3

INTERNSHIP ACTIVITIES

3.1 Roles and Responsibilities

During the twelve-week internship, the role held was that of a Python developer with specific job responsibilities centered on various aspects of Robotic Process Automation using RPAFramework, data manipulation utilizing pandas and NumPy, research and development in the field of CAPTCHA, and contributions to the News Scraping project for the QuickSamachar application. The primary tasks revolved around the development and enhancement of the Password Reset Bot, ensuring its reliability and functionality. The responsibilities can be summarized as follows:

- **Understanding Company Operations**: The internship began with an immersion into the company's operations, including the workflow, tools, and best practices related to the Password Reset Bot project.
- **Project Management**: The primary duty was to take ownership of the Password Reset Bot project, working independently to enhance its capabilities and address any issues that arose.
- **Execution of Assigned Tasks:** Actively participating in tasks delegated by the supervisor, particularly those related to the improvement of the Password Reset Bot's functionalities. This involved coding and testing to ensure the bot's effectiveness in resetting passwords securely.
- **Regular Progress Reporting:** To maintain effective communication, regular updates were provided to the supervisor regarding the status of the Password Reset Bot project and any tasks associated with it.
- **Continuous Learning and Mentorship**: Throughout the internship, there was a commitment to ongoing learning and skill development in the context of the Password Reset Bot project. Seeking guidance from mentors and senior colleagues was a key aspect of professional growth.
- **Creation of Professional Solutions:** The primary objective was to deliver high-quality solutions aligned with the organization's coding standards. This entailed producing well-structured and maintainable code to ensure the Password Reset Bot's reliability and security.

3.2 Weekly Logs

Table 3.1 Weekly activity log

Week	Activities
1st Week	Presentation on Git and GitHub, Coding practices, SOLID Principle and Zen of Python. Learned about RPAFramework.
2nd Week	Developed IMDb Web scrapper as per instructed by the Supervisor.
3rd Week	Familiarized with QuickRPA platform by integrating IMDb bot in the platform.
4th Week	Worked on web scraping of different news sites for QuickSamachar
5th Week	Continued working on web scraping
6th Week	On-premise visit for Functional requirement of Password Reset Bot
7th Week	Built process flow and finalized FRS document
8th Week	Started development on Password Reset Bot, developed helpdesk component, Finacle component , MIS component and factory pattern
9th Week	Developed Database component, ECC component, NCHL component, Bank Central component and Finished development of Password Reset Bot
10th Week	Tested on UAT server and addressed required changes
11th Week	Deployed bot on live server
12th Week	Focused on concluding activities and ensuring the Password Reset Bot is working as desired.

3.3 Description of the Projects Involved During Internship

3.3.1 Feasibility Study

Feasibility study is an analysis of how successfully the project can be completed. It is an assessment of the practicality of a proposed project or a system. A feasibility study aims to uncover the strengths and weaknesses of an existing business or proposed venture objectively and rationally, opportunities and threats present in the natural environment.

3.3.1.1 Technical Feasibility:

Technology Stack: Evaluate the technical requirements, including the programming languages, libraries (like Selenium), and tools needed to build and maintain the bot. Ensure that the required technologies are available and suitable for the task.

Integration: Assess whether the bot can integrate seamlessly with the existing systems and infrastructure, such as the help desk software and user authentication mechanisms.

3.3.1.2 Operational Feasibility:

User Training: Consider the ease with which help desk personnel and end-users can interact with the bot. Ensure that the bot's operation is user-friendly and doesn't require extensive training.

Maintenance: Determine the ongoing operational costs, such as software updates, security patches, and bot maintenance. Ensure that the bot can be efficiently maintained over time.

3.3.1.3 Financial Feasibility:

Cost-Benefit Analysis: Calculate the costs associated with developing, deploying, and maintaining the bot compared to the expected benefits, including cost savings from reduced help desk workload and improved efficiency.

Return on Investment (ROI): Determine the expected ROI by estimating how long it will take for the bot to pay back the initial investment through cost savings and productivity gains.

3.3.1.4 Legal and Compliance Feasibility:

Data Privacy: Ensure that the bot complies with data privacy regulations, especially when dealing with sensitive user information like passwords.

Authorization and Access Control: Verify that the bot's authorization process aligns with company policies and industry best practices to prevent unauthorized access to user accounts.

3.3.2 Requirement Collection

In the requirement collection phase, the original system for recommendation was studied and understood. Upon understanding the existing system, the possible functional and non-functional

requirements of the systems are determined that the system must meet. For requirement collection following conditions are followed:

3.3.2.1 Functional Requirement:

Functional requirements for a "Password Reset Bot" should specify the features and capabilities the bot must have to perform its intended task effectively and efficiently. Here are some key functional requirements for a password reset bot:

- **Credential Storage:** The bot should securely store login credentials (e.g., usernames and passwords) for various applications or systems that require password resets.
- **User Interaction:** The bot must communicate with users to gather necessary information for password resets, such as the user's identity and the system or application for which the reset is requested.
- **Authorization Verification:** The bot should check whether the user making the request has the necessary authorization to initiate a password reset for a specific account or system.
- **Password Reset Workflow:** The bot should perform the necessary steps to reset a password, including logging into the target system, generating a new password, and updating the user's account with the new credentials.

3.3.2.2 Non-Functional Requirements:

- **Security:** User credentials and sensitive data must be encrypted during storage and transmission.
- **Performance:** The bot should respond to password reset requests within a reasonable time frame to maintain user satisfaction.
- **Reliability**: The bot should be available and operational whenever users require password resets, minimizing downtime.
- **Error Handling:** Implement effective error handling and recovery mechanisms to ensure system stability and prevent data loss.
- **Usability:** The bot should have an intuitive and user-friendly interface for both help desk personnel and end-users.
- **Auditability and Logging:** Maintain comprehensive logs of all bot activities and password reset transactions for auditing purposes.

3.3.3 System Design

System Design focuses on how to accomplish the objective of the system. System Design is to deliver the requirements as specified in the feasibility report. System Design is the process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements.

3.3.3.1 Process Flow Diagram

A process flow is a visual representation of a sequence of steps or activities in a workflow. It illustrates how tasks are interconnected, showcasing the order and dependencies between them. Through symbols and arrows, it provides a clear and concise overview of the process, aiding in understanding, analysis, and optimization. Process flows are invaluable in various fields, such as project management, manufacturing, and software development, facilitating efficient communication and decision-making among team members. Following Process diagram is followed by the Password Reset Bot:

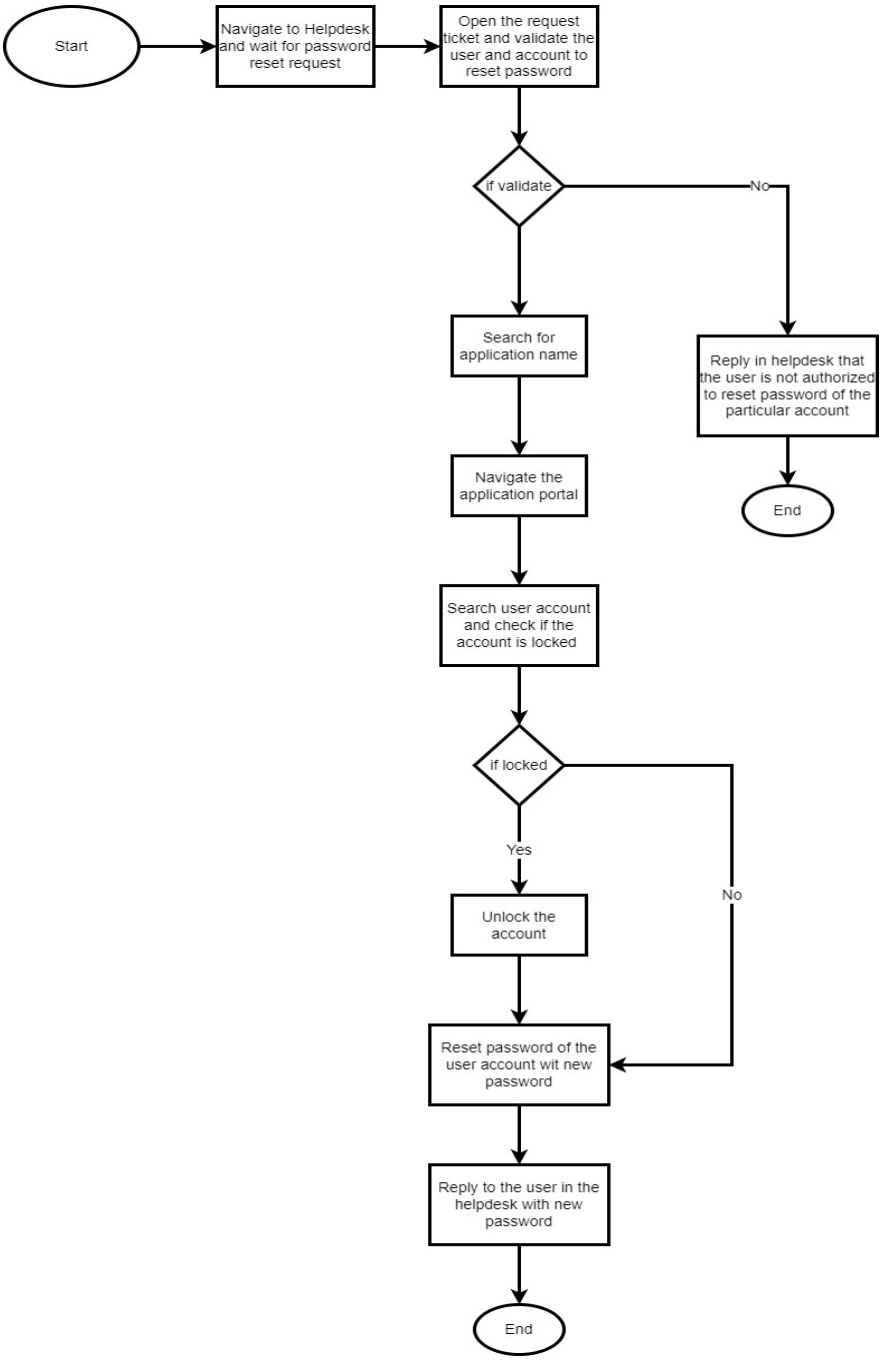

Figure 3.1 Process Flow Diagram

3.4 Task and activities performed

During the internship at QuickFox Consulting, the intern adeptly employed a diverse set of Tools and Technologies to engineer the Password Reset Bot.

Integrated Libraries and Technologies:

- **RPA Framework:** The project involved RPA Framework to facilitated efficient and structured automation of tasks within the Password Reset Bot.
    - **RPA.Browser.Selenium:** The intern leveraged the Selenium library to empower the bot in engaging with web-based user interfaces, optimizing workflows associated with password reset.
    - **RPA.Desktop :** The intern used RPA.Desktop to facilitated automation of desktop actions, contributing to a comprehensive solution for password reset functionality.
- **Pyperclip:** Pyperclip library was used for clipboard operations, enhancing data transfer and manipulation
- **SQLite3:** Integrating SQLite3 into the project provided the database solution to store information gathered by bot and store logs.

Through the integration of the technologies, the intern successfully constructed a robust Password Reset Bot. The seamless combination of these technologies resulted in a stable and efficient solution**.** The contributions of the intern are:

- **Requirement Collection and FRS Preparation**: Initiating with an on-premise visit to ascertain functional requirements, a detailed process flow was crafted, culminating in the finalization of the Functional Requirements Specification (FRS) document. Upon client approval of the document, the development of the Password Reset Bot was commenced.
- **Bot Module**: Bot module defines a Python class Bot that extends the QRBot class from the qrlib module. It also includes a PWD_RST_Process class from the PWD_RST_Process module. The Bot class has two main methods:
    - **start:** Initiates the bot by configuring platform components using setup_platform_components(), performing pre-run setup with before_run(), and then executing the run with execute_run().

```
10. def start(self):
11.         self.setup_platform_components()
12.         self.process.before_run()
13.         self.process.execute_run()
```

- **teardown:** Performs post-run cleanup using after_run().

```
15. def teardown(self):
16.         self.process.after_run()
```

- **PWD_RST_Process Module:** The module defines a class PWD_RST_Process that orchestrates a password reset process from help desk tickets. Builtin Libraries like time, datetime are imported. This class serves as the main driver for automating password resets, managing help desk tickets, and updating a database. Important methods of the class are:
  - **before_run:** The method initializes the Chrome driver path, creates a database, and logs into the IT help desk.

```
43.     def before_run(self, *args, **kwargs):
44.         """Fetch from Vault and Open Browser
45.             login to IT_HELP_DESK
46.         """
47.         get_chrome_driver_path()
48.         self.database.create_database()
49.         self.helpdeskcomponent.login()
```

  - **before_run_item:** Reads ticket information, validates the row, and manages ticket skipping based on various conditions.

```
51.     def before_run_item(self, *args, **kwargs):
52.         """Collect Row information"""
53.         self.info = {}
54.         self.rowvalid = False
55.         self.sleep1minute = False
56.
57.                 self.rowvalid,  self.info  =  self.helpdeskcomponent.read_ticket(args[0],
    self.skip_list)
58.         if self.rowvalid:
59.             self.row_num = 0
60.             self.empty_table_count = 0
61.         elif not self.rowvalid and self.info['ticket_id']:
62.             self.skip_list.append(self.info['ticket_id'])
63.         elif not self.info['ticket_id']:
64.             self.row_num = 0
65.             self.empty_table_count += 1
66.             self.sleep1minute = True
67.
68.         if self.info['ticket_id']:
69.             self.last_row_id = self.database.insert_into_database(self.info)
```

- **execute_run_item:**
  - The method resets passwords, updates the database, and replies to help desk tickets based on the outcome.
  - Checks if a ticket should be skipped based on its presence in a skip list or if it's empty.

```
def execute_run_item(self, *args, **kwargs):
            self.newpass = ""
            self.password_reset_sucess = False
            if Constants.INVALID_TEMPLATE:
                self.info['Remarks'] = "Invalid Template"
            elif self.rowvalid:
                try:
                                                    self.password_reset_sucess,self.newpass   = self.factory.run_factory(self.info)
                if self.password_reset_sucess and 'Password Reset' in self.info['template']:
                        self.info['Remarks'] = "Password reset Success"
                    elif self.password_reset_sucess:
                        self.info['Remarks'] = "Unlocked successfully."
                    elif not self.password_reset_sucess and Constants.USER_ALREADY_ACTIVE:
                        self.info['Remarks'] = "User already active."
                    elif not self.password_reset_sucess and Constants.USER_EXPIRED:
                        self.info['Remarks'] = "User Account Expired."
                    elif not self.password_reset_sucess and Constants.USER_NOT_AVAILABLE:
                        self.info['Remarks'] = "User Account Not available."
                    else:
                        self.info['Remarks'] = "Password reset unsuccessful"
                except Exception as e:
                    self.info['Remarks'] = "Error in Application"
            try:
                if not self.password_reset_sucess:
                    if self.info['ticket_id']:
                       BuiltIn().log_to_console(f"appending {self.info['ticket_id']} in the skip ticket list")
                        self.skip_ticket_list.append(self.info['ticket_id'])
                        self.info['Ticket Success'] = "Skipping Ticket"
                        self.database.update_database(self.last_row_id,self.info)
                else:
                    if self.rowvalid and self.password_reset_sucess:
                        ticket_success = self.helpdeskcomponent.reply_ticket(self.rowvalid, self.newpass, self.info, self.password_reset_sucess)
                      self.info['Ticket Success'] = "Success" if ticket_success else "Error"
                        self.database.update_database(self.last_row_id,self.info)
                        run_item.set_success()
                    elif self.rowvalid and not self.password_reset_sucess:
                        ticket_success = self.helpdeskcomponent.reply_ticket(self.rowvalid, self.newpass, self.info, self.password_reset_sucess)
                      self.info['Ticket Success'] = "Success" if ticket_success else "Error"
                        self.database.update_database(self.last_row_id,self.info)
                        run_item.set_error(trace=False)
                        self.database.update_database(self.last_row_id,self.info)
                    if self.info['ticket_id']:
                        run_item.report_data = self.info
                        run_item.post()

```

```
            except Exception as e:
                BuiltIn().log_to_console("Error occor while replying")
                if self.info['ticket_id']:
                    run_item.set_error(trace=False)
                    run_item.report_data = self.info
                    run_item.post()
```

- **after_run:** Logs out from the IT help desk and closes the browser.

```
    def after_run(self, *args, **kwargs):
        """Logout IT_HELP_DESK and
        Close the browser
        """
        self.helpdeskcomponent.logout()

```

- **execute_run:** Runs the password reset process in a loop, handling conditions such as time-based termination and empty ticket tables.

```
     def execute_run(self):
         while True:
                          if  str(datetime.datetime.now()).split()[-1]  >  '19:00:00'  or str(datetime.datetime.now()).split()[-1] < '08:00:00':
                 break
             self.row_num += 1
             self.before_run_item(self.row_num)
             self.execute_run_item()
             if self.sleep1minute:
                 time.sleep(30)
                 if self.empty_table_count >= 15:
                      BuiltIn().log_to_console("Ticket Table is empty so breaking out of while and terminating bot")
                     break
```

- **Factory Pattern Module:** The module defines an ApplicationFactory class that acts as a component for orchestrating the password reset process for various applications. This class serves as a centralized component for handling password resets across different applications, providing a modular and extensible approach to the password reset functionality.
    - **run_factory:** It accepts application information (info) as input and checks if the specified application is present in the sources dictionary. If available, it retrieves the corresponding password reset process from sources and executes it using the main method. The method logs success or failure messages based on the outcome

of the password reset process. It returns a tuple indicating whether the password reset was successful and the new password if successful.

```
•   def run_factory(self,info):
•   10.         self.new_pass = ""
•   11.         try:
•   12.             if info['application_name'] in sources:
•   13.               self.logger.info(f"Working Application is ---> {info['application_name']}")
•   14.                 reset_process = sources[info['application_name']]
•   15.                 password_reset_sucess, self.new_pass = reset_process.main(info)
•   16.                 if password_reset_sucess:
•   17.                     self.logger.info("Password Reset Success")
•   18.                 else:
•   19.                     self.logger.error("Password Reset Failed")
•   20.             else:
•   21.                 self.logger.error(f"{info['application_name']} not available ")
•   22.         except:
•   23.             self.logger.error("Password Reset Failed")
•   24.             password_reset_sucess = False
•   25.         return password_reset_sucess, self.new_pass
```

- **HelpDesk Component:** The component defines a Help_Desk_Component class that automates login/logout and ticket processing on a help desk system. It uses the RPA framework for browser automation and desktop interactions. The script utilizes Selenium for browser automation and Desktop for simulating keyboard shortcuts, interacting with the browser, and capturing screenshots in case of errors. It also includes error handling and logging for improved reliability and troubleshooting.
    - **login:** The method is responsible to navigate to the help desk login page, inputs credentials fetched from Vault, and logs in.

```
 1. def login(self):
 2.         try:
 3.             self.helpdesk_url = QREnv.VAULTS['helpdesk_vault']['url']
 4.             self.helpdesk_user = QREnv.VAULTS['helpdesk_vault']['username']
 5.             self.helpdesk_pass = QREnv.VAULTS['helpdesk_vault']['password']
 6.                                                                 chromedriver_path      =
"C:\\\\Users\\\\bot\\\\Documents\\\\Chromedriver_path\\\\chromedriver.exe"
 7.                             self.browser.open_browser(url=self.helpdesk_url,  browser="chrome",
executable_path=Constants.CHROME_PATH, options=option)
 8.                     wait_and_input_text(browser_lib=self.browser,  xpath=xpath["username_xpath"],
text=self.helpdesk_user, timeout=Constants.TIMEOUT)
 9.                 wait_and_input_password(browser_lib=self.browser,  xpath=xpath["password_xpath"],
password=self.helpdesk_pass, timeout=Constants.TIMEOUT)
10.             BuiltIn().log_to_console("Password Input Success")
11.             if "help" in self.helpdesk_url:
12.                  wait_and_click(browser_lib=self.browser, xpath="//div[contains(@class,'select2-
container')]", timeout=Constants.TIMEOUT)
```

```
13.                                                 wait_and_input_text(browser_lib=self.browser,
xpath="//input[contains(@class,'select2-input')]",           text="Local            Authentication",
timeout=Constants.TIMEOUT)
14.                 self.browser.press_keys(None,"RETURN")
15.                             wait_and_click(browser_lib=self.browser,  xpath=xpath["login_xpath"],
timeout=Constants.TIMEOUT)
16.             BuiltIn().log_to_console("Login Success")
17.             self.desktop.press_keys("ALT","SPACE")
18.             time.sleep(.5)
19.             self.desktop.press_keys("N")
20.
21.         except Exception as e:
22.             self.browser.capture_page_screenshot()
23.             BuiltIn().log_to_console("Failed to login")
24.             raise e
25.
```

- **logout:** It logs out from the help desk system.

```
 1. def logout(self):
 2.         try:
 3.             BuiltIn().log_to_console("Logging out...")
 4.                     wait_and_click(browser_lib=self.browser,  xpath=xpath["user_profile_xpath"],
timeout=Constants.TIMEOUT)
 5.             BuiltIn().sleep(1)
 6.                         wait_and_click(browser_lib=self.browser,  xpath=xpath["logout_xpath"],
timeout=Constants.TIMEOUT)
 7.             BuiltIn().log_to_console(f'Logout Succesful...')
 8.             self.browser.close_browser()
 9.
10.         except Exception as e:
11.             self.browser.capture_page_screenshot()
12.             self.browser.close_browser()
13.             raise e
14.
```

- **read_ticket:** it is responsible for reading and processing information from help desk tickets, validating and skipping as needed. The method returns fetched information as a dictionary.

```
 1. def read_ticket(self, row_num, skip_list):
 2.         info = {
 3.                 "ticket_id":"",
 4.                 "requester_name":"",
 5.                 "employee_id":"",
 6.                 "application_name":"",
 7.                 "application_id":"",
 8.                 "template":""
 9.                 }
10.
11.                                                                                 try:
self.browser.go_to("https://helpdesk.nabilbank.com/WOListView.do?viewID=617&globalViewName=Requests&
requestViewChanged=true")
12.             time.sleep(2)
13.             first_row = self.browser.get_webelements(ROW_XPATH)
14.             BuiltIn().log_to_console(f"first row is {first_row}")
15.             if first_row:
16.                 try:
```

```
                    ticket_id = get_element_text(browser_lib=self.browser, xpath=TICKET_ID_XPATH,
timeout=Constants.TIMEOUT)
                except Exception as e:
                    self.browser.capture_page_screenshot()
                    return False,info
                                    requester_name  =  get_element_text(browser_lib=self.browser,
xpath=REQUESTER_NAME_XPATH, timeout=Constants.TIMEOUT)
                if ticket_id in skip_list:
                    return False, info
                           wait_and_click(browser_lib=self.browser,  xpath=OPEN_FIRST_ROW_XPATH,
timeout=Constants.TIMEOUT)
                try:
                                    applicationname  =  get_element_text(browser_lib=self.browser,
xpath=xpath["applicationname1_xpath"], timeout=Constants.TIMEOUT)
                except:
                                    applicationname  =  get_element_text(browser_lib=self.browser,
xpath=xpath["applicationname2_xpath"], timeout=Constants.TIMEOUT)
                                     applicationid  =  get_element_text(browser_lib=self.browser,
xpath=xpath["applicationid_xpath"], timeout=Constants.TIMEOUT)
                                        employeeid  =  get_element_text(browser_lib=self.browser,
xpath=xpath["employee_id_xpath"], timeout=Constants.TIMEOUT)
                                   employeecheckid  =  get_element_text(browser_lib=self.browser,
xpath=xpath["employeeid_to_check_xpath"], timeout=Constants.TIMEOUT)
             template = get_element_text(browser_lib=self.browser, xpath=xpath["template_xpath"],
timeout=Constants.TIMEOUT)
                info = {
                    "ticket_id":ticket_id,
                    "requester_name":requester_name,
                    "employee_id":employeeid,
                    "application_name":applicationname,
                    "application_id":applicationid,
                    "template":template
                }
                self.desktop.press_keys("ALT","SPACE")
                time.sleep(.5)
                self.desktop.press_keys("N")
                if applicationname == "Finacle":
                    if employeecheckid == employeeid :
                        BuiltIn().log_to_console("Row is valid and application name if Finacle")
                        return True,info
                    else:
                        return False,info
                elif ticket_id:
                    return True,info
                BuiltIn().log_to_console("Row is invalid")
                return False,info

            else:
                BuiltIn().log_to_console("Row is not visible")
                return False,info
        except Exception as e:
            self.browser.capture_page_screenshot()
            Constants.INVALID_TEMPLATE = True
            return False,info

```

- **reply_ticket:** It replies to help desk tickets, changing their status to "Closed" and providing appropriate responses based on the ticket's content and processing outcomes.

```
def reply_ticket(self, valid, newpass, info, password_rst_success):
```

```
        try:
            if valid:
                if 'Password Reset' in info['template'] and password_rst_success:

                    reply = f"Your New password is {newpass}"

                elif not password_rst_success and Constants.INVALID_TEMPLATE:
                    Constants.INVALID_TEMPLATE = False
                    reply = "Invalid Template, Please select correct template."

                elif not password_rst_success and Constants.USER_EXPIRED:
                    Constants.USER_EXPIRED = False
                    reply = "User Account is Expired. Please get necessary approval to reinstate Finacle account."
                elif not password_rst_success:
                    reply = "Error occured while resetting Password, Please Contact IT"
            else:
                  reply = f"You are not authorized person(EmployeeID {info['employee_id']})  to reset password of ApplicationID {info['application_id']}."
            self.browser.unselect_frame()
                        wait_and_click(browser_lib=self.browser,  xpath=xpath["status_xpath"], timeout=Constants.TIMEOUT)
            self.browser.press_keys(None, 'Closed')
            self.browser.press_keys(None, 'RETURN')
              wait_and_input_text(browser_lib=self.browser, xpath=xpath["comment_status_xpath"], text=reply, timeout=Constants.TIMEOUT)
                wait_and_input_text(browser_lib=self.browser,  xpath=xpath["closure_comment"], text=reply, timeout=Constants.TIMEOUT)
              wait_and_click(browser_lib=self.browser,  xpath=xpath["close_request_btn_xpath"], timeout=Constants.TIMEOUT)
            self.logger.info(f"Ticket Replied Succesfully")
            return True

        except Exception as e:
            self.browser.capture_page_screenshot()
            return False

```

- **Database Component**: The component defines a Database class that interacts with a SQLite database for storing information related to password reset requests. This class serves as a component for managing the SQLite database operations related to password reset requests, providing functionality for initialization, insertion, and updating of records.
    - **create_database**: The method checks if the SQLite database file exists. If it doesn't exist, it creates a connection to the database and initializes a table (bot_table) with specific columns. It also calls create_trigger to create a trigger for updating the updated_date field when the ticket_status is modified.

```
def create_database(self):
        if not os.path.exists(self.DBName):
            try:
                con = sqlite3.connect(self.DBName)
```

```
                cur = con.cursor()
                cur.execute(f"""CREATE TABLE IF NOT EXISTS {self.table_name}(
                                                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                                                    ticket_id TEXT,
                                                    requester_name TEXT,
                                                    employee_id TEXT,
                                                    application_name TEXT,
                                                    created_date TEXT,
                                                    updated_date TEXT,
                                                    ticket_status TEXT,
                                                    remarks TEXT)
                            """)
            except Exception as e:
                raise e
            cur.close()
            con.close()
        self.create_trigger()

```

- **insert_into_database:** It inserts a new row into the bot_table table with information from a password reset request.

```
def insert_into_database(self, info):
            con = sqlite3.connect(self.DBName)
            cur = con.cursor()
            insert_sql = f"""INSERT INTO {self.table_name}(
                                                    ticket_id,
                                                    requester_name,
                                                    employee_id,
                                                    application_name,
                                                    created_date,
                                                    ticket_status
                                                    )
                                                    VALUES (?,?,?,?,?,?)
                                            """
                    info_tuple = (info['ticket_id'], info['requester_name'], info['employee_id'],
info['application_name'], datetime.datetime.today(), "pending")
            cur.execute(insert_sql, info_tuple)
            lastrowid = cur.lastrowid
            con.commit()
            cur.close()
            con.close()
            BuiltIn().log_to_console("Rows inserted in DB")
```

- **create_trigger:** The method creates a trigger in the database to automatically update the updated_date field when the ticket_status is modified.

```
def create_trigger(self):
        con = sqlite3.connect(self.DBName)
        cur = con.cursor()
        trigger_query = f"""
                        CREATE TRIGGER IF NOT EXISTS updatedatetrigger
                        AFTER UPDATE OF ticket_status ON {self.table_name}
                        FOR EACH ROW
```

```
                        WHEN old.ticket_status <> new.ticket_status
                        BEGIN
                            UPDATE {self.table_name}
                            SET updated_date = datetime('now', 'localtime')
                            WHERE id = old.id;
                        END;
                        """
        cur.execute(trigger_query)
        con.commit()
        cur.close()
        con.close()
```

- **update_database**: It updates the ticket_status and remarks fields for a specific row in the bot_table table.

```
    def update_database(self,last_row_id,info):
        con = sqlite3.connect(self.DBName)
        cur = con.cursor()
        update_sql = f"UPDATE {self.table_name} SET ticket_status= ?, remarks= ?  WHERE id = ?"
        cur.execute(update_sql, (info['Ticket Success'],info['Remarks'],last_row_id))
        BuiltIn().log_to_console("Database updated succesfully")
        con.commit()
        cur.close()
        con.close()

```

- **MIS Component**: The component defines a class MIS_Component that represents the password reset functionality for the "MIS" application
  - **main(info)**: Main method for password reset, invokes password_reset method.

```
def main(self,info):
        self.retrieve_vault()
        try:
            status , new_pass = self.password_reset(info)
        except Exception as e:
            new_pass = ""
        finally:
            return status, new_pass

```

  - **retrieve_vault():** It retrieves connection details from the vault.

```
def retrieve_vault(self):
        self.mis_user = QREnv.VAULTS['mis_vault']['user']
        self.mis_pass = QREnv.VAULTS['mis_vault']['password']
        self.mis_server = QREnv.VAULTS['mis_vault']['server']
        self.mis_port = QREnv.VAULTS['mis_vault']['port']
        self.mis_service = QREnv.VAULTS['mis_vault']['service_name']

```

- **password_reset(info):** The method resets the password for the "MIS" application using Oracle database queries.

```
def password_reset(self,info):
        try:
            self.logger.info(f"Resetting MIS password of {info['application_id']}")
            new_pass = " "
            new_pass = str(password_generator())
            application_id = str(info['application_id']).upper()
            user_exist_query = f"""SELECT * FROM mis_user_mst
                                   WHERE user_code='{application_id}'
                                   """
            sql_update_query = f"""UPDATE mis_user_mst
                                  SET user_password=MIS_PWD.FN_ENCRYPT_CHARACTER('{new_pass}'),
                                   first_login='Y',
                                   User_Lock='N',
                                   Login_atmpt='0'
                                   WHERE user_code='{application_id}'
                                   """
            self.logger.info("Connecting to Database")
                    connection  =  oracledb.connect(user=self.mis_user,  password=self.mis_pass,
dsn=f"{self.mis_server}:{self.mis_port}/{self.mis_service}")
            self.logger.info("Database Connection Succesful")
            cursor = connection.cursor()
            user_exist = cursor.execute(user_exist_query).fetchall()
            if not user_exist:
                self.logger.info(f"MIS User not available")
                Constants.USER_NOT_AVAILABLE = True
                return False, new_pass
            cursor.execute(sql_update_query)
            connection.commit()
            connection.close()
            self.logger.info(f"MIS password reset Success")
            return True, new_pass
        except Exception as e:
            self.logger.error(f"Password reset Failed for {info['application_id']} ")
            return False, new_pass

```

- **Finacle Component**: The component defines a class Finacle_Component that handles password reset and user unlocking for the "Finacle" application using Selenium. This component is part of a broader system for handling password resets across various applications and uses Selenium for web automation tailored to the Finacle application's structure.
    - **main(info):** It is a main method for password reset, invokes login, unlock_user, and password_reset methods.

```
def main(self,info):
        self.login()
        try:
```

```
            new_pass = " "
            status = self.unlock_user(info)
            if 'Password Reset' in info['template']:
                self.logger.info("Resetting Password")
                status, new_pass = self.password_reset(info)
            self.logout()
        except Exception as e:
            raise e
        finally:
            self.browser.close_browser()
        return status, new_pass

```

- **login():** The method logs into the Finacle application using provided credentials.

```
def login(self):
        try:
            self.finacle_url = QREnv.VAULTS['finacle_vault']['url']
            self.finacle_user = QREnv.VAULTS['finacle_vault']['username']
            self.finacle_pass = QREnv.VAULTS['finacle_vault']['password']
                                                                    chromedriver_path     =
"C:\\\\Users\\\\bot\\\\Documents\\\\Chromedriver_path\\\\chromedriver.exe"
                    option  =  "add_argument('--ignore-certificate-errors');add_argument('--start-
maximized')"
                                self.browser.open_browser(url=self.finacle_url,  browser='chrome',
executable_path=Constants.CHROME_PATH, options=option)
            self.browser.select_frame(xpath["loginFrame_iframe_xpath"])
                    wait_and_input_text(browser_lib=self.browser,  xpath=xpath["username_xpath"],
text=self.finacle_user, timeout=Constants.WAS_TIMEOUT)
                wait_and_input_password(browser_lib=self.browser,  xpath=xpath["password_xpath"],
password=self.finacle_pass, timeout=Constants.TIMEOUT)
                            wait_and_click(browser_lib=self.browser,  xpath=xpath["login_xpath"],
timeout=Constants.TIMEOUT)

        except Exception as e:
            self.browser.capture_page_screenshot()
            raise e

```

- **logout()**: The method logs logs out of the Finacle application.

```
def logout(self):
        try:
                            wait_and_click(browser_lib=self.browser,  xpath=xpath["logout_xpath"],
timeout=Constants.TIMEOUT)
            self.browser.handle_alert('ACCEPT')
            self.browser.close_browser()
            self.logger.info("Finacle browser closed successfully...")

        except Exception as e:
            self.browser.capture_page_screenshot()
            self.logger.error("Failed to logout")
            raise e

```

- **unlock_user(info):** It unlocks a user account in Finacle and performs necessary checks.

```
def unlock_user(self,info):
        try:
            self.browser.unselect_frame()
            self.browser.select_frame(xpath["loginFrame_iframe_xpath"])
                    wait_and_input_text(browser_lib=self.browser,  xpath=xpath["urmuim_xpath"], text="URMUIM", timeout=Constants.TIMEOUT)
            self.browser.press_keys(None, "RETURN")
            self.browser.unselect_frame()
                   self.browser.wait_until_element_is_visible(xpath["loginFrame_iframe_xpath"], timeout=Constants.TIMEOUT)
            self.browser.select_frame(xpath["loginFrame_iframe_xpath"])
                wait_and_select_from_list(browser_lib=self.browser,  xpath=xpath["action_xpath"], value_or_label="M", timeout=Constants.TIMEOUT)
                    wait_and_input_text(browser_lib=self.browser,  xpath=xpath["userid_xpath"], text=info['application_id'], timeout=Constants.TIMEOUT)
                     wait_and_click(browser_lib=self.browser,  xpath=xpath["go_button_xpath"], timeout=Constants.TIMEOUT)
            try:
                  wait_and_click(browser_lib=self.browser, xpath=xpath['continue_button_xpath'], timeout=Constants.TIMEOUT)
            except:
                self.logger.info("User is not available")
                return False
            self.browser.press_keys(xpath['expiry_date_xpath'], 'CTRL+X')
              wait_and_click(browser_lib=self.browser, xpath=xpath['user_ac_expiry_date_xpath'], timeout=Constants.TIMEOUT)
            self.browser.press_keys(xpath['user_ac_expiry_date_xpath'],'SHIFT+HOME')
            self.browser.press_keys(None,'CTRL+C')
            user_ac_expiry_date = pyperclip.paste()
            user_ac_expiry_date = datetime.datetime.strptime(user_ac_expiry_date, "%d-%m-%Y")
            if user_ac_expiry_date < datetime.datetime.today():
                self.browser.capture_page_screenshot()
                return False
                   wait_and_click(browser_lib=self.browser,  xpath=xpath['submit_button_xpath'], timeout=Constants.TIMEOUT)
            self.browser.unselect_frame()
            self.browser.select_frame(xpath["loginFrame_iframe_xpath"])
              wait_and_click(browser_lib=self.browser, xpath=xpath['sso_administration_button'], timeout=Constants.TIMEOUT)
                                                    wait_and_click(browser_lib=self.browser, xpath=xpath['reset_user_login_details_button'], timeout=Constants.TIMEOUT)
                                                wait_and_select_from_list(browser_lib=self.browser, xpath=xpath['modify_select_button'], value_or_label="Modify", timeout=Constants.TIMEOUT)
                    wait_and_input_text(browser_lib=self.browser,  xpath=xpath['user_id_input'], text=info['application_id'], timeout=Constants.TIMEOUT)
                           wait_and_click(browser_lib=self.browser,  xpath=xpath['go_sso_btn'], timeout=Constants.TIMEOUT)
                                                wait_and_select_from_list(browser_lib=self.browser, xpath=xpath['select_reset_type_btn'], value_or_label='modUnlockUser', timeout=Constants.TIMEOUT)
                         wait_and_click(browser_lib=self.browser,  xpath=xpath['sso_submit_btn'], timeout=Constants.TIMEOUT)
            self.browser.wait_until_element_is_visible(xpath["alert_xpath"], Constants.TIMEOUT)
            unlock_text = self.browser.get_text(xpath["alert_xpath"])
            if USER_ALREADY_ACTIVE in unlock_text:
                self.browser.capture_page_screenshot()
                return False
            try:
```

```
43.                 if unlock_text == USER_NOT_LOCK_MSG:
44.                     raise UserNotLockedException()
45.             except UserNotLockedException:
46.                 BuiltIn().log_to_console('User is not locked. No need to unlock.')
47.             return True
48.         except Exception as e:
49.             self.browser.capture_page_screenshot()
50.             BuiltIn().log_to_console('User is not locked. No need to unlock.')
51.             return False
52.
```

- **password_reset(info):** The function resets the password for a user in Finacle and generates a new password.

```
 1. def password_reset(self,info):
 2.         try:
 3.                                                 wait_and_select_from_list(browser_lib=self.browser,
xpath=xpath['modify_select_button'], value_or_label="Modify", timeout=Constants.TIMEOUT)
 4.                     wait_and_input_text(browser_lib=self.browser,  xpath=xpath['user_id_input'],
text=info['application_id'], timeout=Constants.TIMEOUT)
 5.                             wait_and_click(browser_lib=self.browser,  xpath=xpath['go_sso_btn'],
timeout=Constants.TIMEOUT)
 6.
 7.                                                 wait_and_select_from_list(browser_lib=self.browser,
xpath=xpath['select_reset_type_btn'], value_or_label='modUserPw', timeout=Constants.TIMEOUT)
 8.
 9.              '''Generating password for password reset with random 4 digit number behind 5 digit
random alphabets.'''
10.             generate_password = password_generator()
11.
12.               wait_and_input_text(browser_lib=self.browser, xpath=xpath['reset_password_input'],
text=generate_password, timeout=Constants.TIMEOUT)
13.                                                     wait_and_input_text(browser_lib=self.browser,
xpath=xpath['confirm_reset_password_input'], text=generate_password, timeout=Constants.TIMEOUT)
14.
15.                           wait_and_click(browser_lib=self.browser,  xpath=xpath['sso_submit_btn'],
timeout=Constants.TIMEOUT)
16.             '''Reading alert to confirm whether the user was unlocked or not'''
17.
18.             PASSWORD_CHANGE_MSG = "User updated successfully."
19.
20.                         self.browser.wait_until_element_is_visible(xpath["alert_for_reset_xpath"],
Constants.TIMEOUT)
21.             unlock_text: str = self.browser.get_text(xpath["alert_for_reset_xpath"])
22.             if not unlock_text.strip() == PASSWORD_CHANGE_MSG:
23.                 raise PasswordResetException()
24.             return True, generate_password
25.         except:
26.             self.browser.capture_page_screenshot()
27.             return False
28.
```

- Other components: Similar to Finacle Component, ECC Component, NCHL Component, Bank Central Component defines a class ECC_Component, NCHL_Component, Bank_Central_Component that handles the password reset of their respective application

and unlock/ activate the account if needed. Similar to Finale Component, the methods are built using selenium and RPA Framework.

- **Utils**: These utility functions provide reusable components for interacting with web elements and generating passwords in the context of a password reset bot. The password_generator function is particularly focused on generating secure and random passwords for use in the password reset process. The methods defined in Utils are called by different components throughout the Password Reset process
    - **wait_and_click:** It waits for an element to be visible and then clicks on it.

```
1. def wait_and_click(browser_lib, xpath, timeout):
2.     browser_lib.wait_until_element_is_visible(locator=xpath, timeout=timeout)
3.     browser_lib.click_element(locator=xpath)
4.
```

    - **wait_and_input_text:** The function waits for an input element to be visible and inputs the given text.

```
1. def wait_and_input_text(browser_lib, xpath, text, timeout):
2.     browser_lib.wait_until_element_is_visible(locator=xpath, timeout=timeout)
3.     browser_lib.input_text(locator=xpath, text=text)
4.
```

    - **password_generator:** It generates a password for password reset with a random 4-digit number behind a 5-digit random alphabet.

```
1. def password_generator():
2.     str_size = 5
3.     allowed_chars = string.ascii_uppercase
4.     random_string = ''.join(random.choice(allowed_chars) for x in range(str_size))
5.     password = f'{random_string}@{str(random.randint(1000,9999))}'
6.     return password
7.
```

- **Testing and Deployment:** The final phase involved rigorous testing on the UAT server, resulting in a few necessary adjustments such as the order of selecting tickets and improved exception handling. Following successful UAT testing, the bot was transitioned to the live server to implement real password changes for staff users.

CHAPTER 4

CONCLUSION AND LEARNING OUTCOMES

4.1. Conclusion

The internship experience at QuickFox Consulting has proven to be transformative. The involvement in the Password Reset Bot project has provided significant insights into the ever-evolving field of Robotic Process Automation (RPA) and its practical applications. This project served as a platform to delve into the technical complexities of RPA and its palpable advantages to organizations, including the optimization of operational efficiency, fortification of security measures, and the provision of self-service options to users.

Collaborating with a dedicated professional team at QuickFox Consulting has significantly contributed to the broadening of knowledge and an environment fostering growth and innovation. The project demanded meticulous attention to detail, problem-solving skill, and adaptability, all of which have played an integral role in personal and professional development. The integration of Git and GitHub for version control has added a crucial layer of organization and collaboration to the project, enhancing its overall efficiency.

The opportunity to work on-site with clients during the internship was a uniquely valuable experience. It allowed for the practical application of technology solutions within real-world business settings. Direct client interaction facilitated a profound understanding of their specific requirements and the challenges they encountered in their daily operations. This experience underscored the pivotal roles played by effective communication, adaptability, and flexibility in the realm of technology consulting. It also reinforced the significance of problem-solving and troubleshooting skills, particularly in client environments where unexpected challenges often surface.

In conclusion, the internship at QuickFox Consulting, particularly the involvement in the Password Reset Bot project and the on-site client engagement, has been a transformative journey. It deepened the understanding of RPA and its potential impact on organizations, improved technical and project management capabilities, and provided a comprehensive perspective on technology solutions in practical business contexts. The experience has been immensely valuable, and the acquired knowledge and skills are eagerly anticipated to be applied in future career endeavors.

4.2. Learning Outcome

The internship experience at QuickFox Consulting proved to be a significant and enlightening period within the organization, affording a firsthand exposure to the intricacies of a genuine working environment. It served as a valuable platform for gaining insights into the principles, commitment, challenges, and ethical considerations that are fundamental in a real-world organizational setting. Throughout the internship, a range of interpersonal and professional skills were cultivated, contributing to my development. Some of these acquired skills are delineated below:

- **Proficiency in RPA:** A solid foundation was laid in Robotic Process Automation (RPA), a skill of paramount importance in the technology sector.
- **Technical Expertise**: Substantial technical knowledge was acquired in the domains of software development and automation.
- **Project Management**: The ability to meticulously plan and execute projects with efficiency was developed.
- **Effective Communication**: A significant enhancement in communication skills, particularly crucial in the context of technology consulting.
- **Problem Solving**: Proficiency was achieved in troubleshooting and resolving real-world challenges that arise in professional settings.
- **Adaptability**: The experience of working with diverse clients instilled adaptability and flexibility in handling varying work contexts.
- **Client Interaction**: Proficiency in building and maintaining positive client relationships was acquired.
- **Business Context**: A comprehensive understanding of how technology integrates into the broader business context was developed.
- **Teamwork**: Strong teamwork skills were nurtured through collaborative efforts with fellow professionals.

This internship opportunity has undoubtedly played a pivotal role in the acquisition and refinement of these valuable skills, preparing for a successful and productive journey in the professional world.

REFERENCES

Ahuja S, a. T. (2021). Performance evaluation of Robotic Process Automation on waiting lines of toll plazas. *NVEO Natural Volatiles & Essentials Oils*, 8(5).

Can Tansel KAYA, M. T. (2019). Impact of RPA Technologies on Accounting Systems. *The Journal of Accounting and Finance*, 253-250.

Ruchi Issac, R. M. (2018). Delineated Analysis of Robotic Process Automation Tools. *International Conference on Advances in Electronics, Computer and Communications* .

Santiago Aguirre, A. R. (2017). Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study. *Springer International Publishing*, 65–71.

Somayya Madakam, R. M. (2019). THE FUTURE DIGITAL WORK FORCE: ROBOTIC PROCESS AUTOMATION (RPA). *Journal of Information Systems and Technology Management, 16*.