A FINAL YEAR PRESENTATION

ON

ROBOTIC PROCESS AUTOMATION (RPA) FOR BUSINESS PROCESSES."

BY

EZICHI CHUKWUNONSO FRANCIS
REG NO: NS/CSC/20/6406

SUBMITTED TO

THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF NATURAL AND APPLIED SCIENCES TANSIAN UNIVERSITY UMUNYA, ONITSHA ANAMBRA STATE

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.sc) DEGREE IN COMPUTER SCIENCE

SUPERVISOR

MRS ANYARAGBU HOPE.

TABLE OF CONTENTS

SECTION FOUR: DESIGNING RPA SYSTEMS FOR BUSINESS PROCESSES

4.1 Objectives of RPA Implementation

The primary objectives of implementing Robotic Process Automation (RPA) in business processes are to enhance operational efficiency, reduce costs, improve accuracy, and free up human resources for higher-value tasks. RPA aims to automate repetitive and rule-based tasks traditionally performed by human workers, thereby increasing productivity and allowing employees to focus on more strategic activities.

Key objectives of RPA implementation include:

1. Efficiency Improvement: By automating routine tasks, businesses can significantly reduce the time required to complete processes, leading to faster turnaround times and improved service delivery.

2. Cost Reduction: Automation helps lower operational costs by minimizing the need for manual labor, reducing errors, and decreasing the costs associated with rework and corrections.

3. Accuracy and Consistency: RPA bots perform tasks with a high degree of precision, eliminating the risk of human error and ensuring consistency in process execution.

4. Scalability: RPA solutions can be easily scaled to handle increased workloads without a proportional increase in costs, making it easier for businesses to grow and adapt to changing demands.

5. Compliance and Risk Management: RPA ensures that processes are carried out in compliance with regulatory standards and internal policies, reducing the risk of non-compliance and associated penalties.

6. Enhanced Customer Experience: By speeding up processes and improving accuracy, RPA contributes to better customer service and satisfaction.

7. Resource Optimization: By automating mundane tasks, businesses can reallocate human resources to more complex and value-adding activities, fostering innovation and growth.

4.2 Identifying Business Processes for Automation

The success of RPA implementation heavily depends on selecting the right business processes for automation. Not all processes are suitable candidates for RPA; therefore, it is crucial to conduct a thorough analysis to identify processes that will yield the maximum benefits when automated.

Key criteria for identifying business processes for automation include:

1. Repetitiveness: Processes that involve repetitive and routine tasks are ideal for automation. These tasks are often mundane and time-consuming for human workers but can be efficiently handled by RPA bots.

2. Rule-based: Processes that follow clear, predefined rules and decision logic are suitable for RPA. Since RPA operates based on structured logic, tasks with well-defined steps and decision criteria can be automated effectively.

3. High Volume: Processes that are performed frequently and in large volumes offer significant opportunities for efficiency gains through automation. High-volume tasks, such as data entry and invoice processing, are prime candidates for RPA.

4. Standardization: Processes that are standardized and have little variation in their execution are easier to automate. Uniform processes with consistent inputs and outputs can be more readily adapted to RPA solutions.

5. Error Prone: Tasks that are susceptible to human error due to their repetitive nature or complexity can benefit greatly from automation. RPA reduces the likelihood of errors, improving the overall quality and reliability of the process.

6. Manual Data Handling: Processes that involve significant manual data handling, such as data extraction, transformation, and entry, are well-suited for RPA. Automation can streamline these activities, reducing the risk of errors and speeding up data processing.

7. Integration Needs: Processes that require integration with multiple systems or applications can be automated to facilitate seamless data flow and interaction between different platforms. RPA can act as a bridge, automating the transfer of data across disparate systems.

To identify processes for automation, businesses can follow a systematic approach:

1. Process Mapping: Document and map out the existing processes to understand their workflows, inputs, outputs, and dependencies. This helps in visualizing the entire process and identifying potential areas for automation.

2. Process Assessment: Evaluate the processes based on the criteria mentioned above. Assess the feasibility and potential benefits of automating each process, considering factors such as complexity, volume, and impact on operations.

3. Stakeholder Input: Engage with stakeholders, including process owners and employees, to gather insights and identify pain points. Stakeholders can provide valuable information on which processes are most time-consuming and prone to errors.

4. Prioritization: Prioritize the processes based on their suitability for automation and the potential benefits they offer. Focus on automating high-impact processes that will deliver the most significant efficiency gains and cost savings.

5. Pilot Projects: Start with pilot projects to test the feasibility and effectiveness of RPA in selected processes. This allows for experimentation and refinement before scaling up automation across the organization.

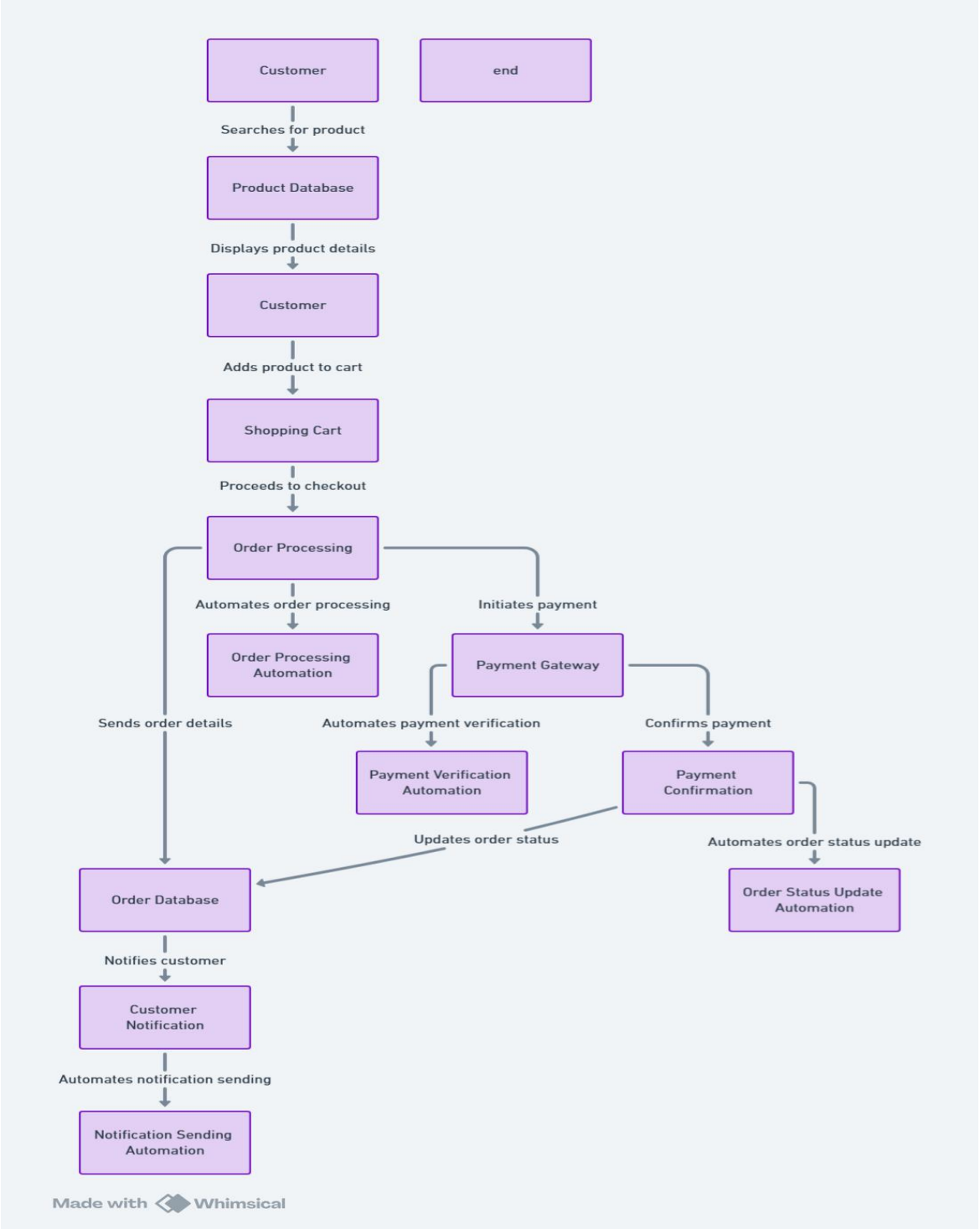
By carefully selecting and prioritizing processes for automation, businesses can maximize the benefits of RPA and achieve their objectives of improved efficiency, cost reduction, and enhanced operational performance.

4.3 Data Flow Diagrams for RPA

Data Flow Diagrams (DFDs) are essential tools in system design, as they provide a graphical representation of how data moves through a system. For Robotic Process Automation (RPA) in business processes, DFDs illustrate the interaction between various components, including data sources, processes, and data stores, as well as the automation elements introduced by RPA. These diagrams help in understanding the flow of data, identifying potential bottlenecks, and ensuring that the automation aligns with business objectives.

In the context of an e-commerce system, the DFD outlines the process flow from the customer searching for a product to the final order notification. The automation points in the system are highlighted to show where RPA can streamline operations.

4.3.1 Expanded DFDs to Show RPA Processes and Data Stores



The provided diagram represents an expanded DFD for an e-commerce system with integrated RPA processes. Below is a detailed explanation of the different components and the flow of data within the system.

1. Customer Searches for Product:
   - The process begins with the customer searching for a product in the product database.
   - Data Store: Product Database
   - Process: Displays product details to the customer.

2. Customer Adds Product to Cart:
   - After viewing the product details, the customer adds the product to the shopping cart.
   - Data Store: Shopping Cart

3. Proceed to Checkout:
   - The customer proceeds to checkout, triggering the order processing workflow.
   - Process: Order Processing
   - RPA Process: Order Processing Automation, which handles the verification and validation of order details automatically.

4. Initiates Payment:
   - The order processing step initiates payment through the payment gateway.
   - Process: Payment Gateway
   - RPA Process: Payment Verification Automation, which automates the verification of payment details.

5. Confirms Payment:
   - The payment gateway confirms the payment, updating the order status.
   - RPA Process: Payment Confirmation, which ensures the payment is processed correctly.
   - Data Store: Order Status Update Automation, which automatically updates the status of the order in the system.

6. Updates Order Status:
   - The system updates the order status based on the confirmation received from the payment gateway.
   - Data Store: Order Database

7. Notifies Customer:
   - The system sends a notification to the customer regarding the order status.
   - Process: Customer Notification

- RPA Process: Notification Sending Automation, which automates the process of sending notifications to customers.

Detailed Flow:

1. Customer Interaction:
   - Customer -> Product Database: Searches for product.
   - Product Database -> Customer: Displays product details.
   - Customer -> Shopping Cart: Adds product to cart.
   - Shopping Cart -> Order Processing: Proceeds to checkout.

2. Order Processing:
   - Order Processing -> Order Processing Automation: Automates order processing.
   - Order Processing -> Payment Gateway: Initiates payment.

3. Payment Handling:
   - Payment Gateway -> Payment Verification Automation: Automates payment verification.
   - Payment Gateway -> Payment Confirmation: Confirms payment.
   - Payment Confirmation -> Order Status Update Automation: Automates order status update.
   - Order Status Update Automation -> Order Database: Updates order status.

4. Customer Notification:
   - Order Database -> Customer Notification: Notifies customer.
   - Customer Notification -> Notification Sending Automation: Automates notification sending.

Conclusion

The expanded DFD for the e-commerce system with RPA processes demonstrates how data flows through the system and highlights the points where automation can significantly enhance efficiency and accuracy. By automating tasks such as order processing, payment verification, and customer notifications, the system can reduce manual intervention, minimize errors, and provide a seamless experience for customers. This approach not only improves operational efficiency but also ensures that the business can handle higher volumes of transactions with consistent quality and reliability.

4.4 Database Design for RPA Systems (e-commerce)

Designing a database for an RPA (Robotic Process Automation) system in an e-commerce context involves ensuring efficient data handling, scalability, and integration with various components of the e-commerce platform. Here's a comprehensive approach to creating such a database:

1. Identify Key Entities and Relationships

The first step is to identify the main entities involved in the e-commerce operations and their relationships. Some key entities typically include:

- Users
- Products
- Orders
- Payments
- Inventory
- Shipping
- Customer Support

2. Define Database Tables

Based on these entities, define the necessary tables and their attributes. Here's a basic schema outline:

Users Table
- `UserID` (Primary Key)
- `Username`
- `Password`
- `Email`
- `FirstName`
- `LastName`
- `Address`
- `PhoneNumber`
- `UserType` (Customer, Admin, etc.)
- `CreatedAt`
- `UpdatedAt`

Products Table
- `ProductID` (Primary Key)
- `ProductName`
- `Description`
- `Price`

- `Category`
- `StockQuantity`
- `CreatedAt`
- `UpdatedAt`

Orders Table
- `OrderID` (Primary Key)
- `UserID` (Foreign Key)
- `OrderDate`
- `TotalAmount`
- `OrderStatus` (Pending, Shipped, Delivered, etc.)
- `ShippingAddress`
- `PaymentID` (Foreign Key)
- `CreatedAt`
- `UpdatedAt`

OrderDetails Table
- `OrderDetailID` (Primary Key)
- `OrderID` (Foreign Key)
- `ProductID` (Foreign Key)
- `Quantity`
- `Price`

Payments Table
- `PaymentID` (Primary Key)
- `OrderID` (Foreign Key)
- `PaymentMethod` (Credit Card, PayPal, etc.)
- `PaymentDate`
- `PaymentStatus` (Pending, Completed, Failed)

Inventory Table
- `InventoryID` (Primary Key)
- `ProductID` (Foreign Key)
- `Quantity`
- `LastRestockDate`

Shipping Table
- `ShippingID` (Primary Key)
- `OrderID` (Foreign Key)
- `ShippingMethod`
- `ShippingDate`
- `ExpectedDeliveryDate`
- `ShippingStatus` (In Transit, Delivered, etc.)

CustomerSupport Table
- `SupportID` (Primary Key)
- `UserID` (Foreign Key)
- `OrderID` (Foreign Key)
- `IssueDescription`
- `IssueStatus` (Open, Closed, In Progress)
- `CreatedAt`
- `UpdatedAt`

3. Normalization and Relationships

Ensure the database is normalized to avoid redundancy and maintain data integrity. Establish the relationships between tables using foreign keys. For instance:

- `Orders` table references `Users` and `Payments` tables.
- `OrderDetails` table references `Orders` and `Products` tables.
- `Inventory` table references `Products` table.
- `Shipping` table references `Orders` table.
- `CustomerSupport` table references `Users` and `Orders` tables.

4. Indexes and Optimization

Create indexes on frequently queried fields to improve performance. For example:

- Index on `UserID` in `Orders` table.
- Index on `OrderID` in `OrderDetails` table.
- Index on `ProductID` in `Inventory` table.

5. RPA Integration Points

Identify points where RPA will interact with the database:

- Order Processing: Automate order creation, status updates, and payment processing.
- Inventory Management: Automate stock level checks and restocking processes.
- Customer Support: Automate responses to common queries and issue tracking.
- Shipping Updates: Automate shipping status updates and notifications to customers.

6. Security and Compliance

Implement security measures to protect sensitive data:

- Use encryption for storing passwords and payment details.
- Ensure compliance with relevant regulations like GDPR for data privacy.

7. Backup and Recovery

Set up regular backups and a recovery plan to prevent data loss and ensure business continuity.

4.5 RPA Workflow and Module Specification

Creating an RPA (Robotic Process Automation) workflow and module specification for an e-commerce database involves identifying key processes that can be automated, defining the steps for each process, and specifying the modules required for implementation. Below is how to approach this task.

1. Order Processing Workflow

Description: Automate the process of handling new orders from placement to fulfillment.

Steps:
1. Order Placement: Triggered when a new order is placed.
2. Order Validation: Check order details for completeness and validity.
3. Inventory Check: Verify product availability.
4. Payment Processing: Process payment using the provided payment method.
5. Order Confirmation: Send order confirmation to the customer.
6. Order Packaging: Notify the warehouse for order packaging.
7. Shipping: Arrange shipping and generate shipping labels.
8. Order Tracking: Update order status and provide tracking information.
9. Post-Purchase: Send follow-up emails and request feedback.

Modules:
- Order Management Module
  - Functions: Validate order, update order status.

- Inventory Management Module

- Functions: Check stock levels, update inventory.

- Payment Processing Module
  - Functions: Process payments, handle payment failures.

- Notification Module
  - Functions: Send emails/SMS for order confirmation, shipping, etc.

- Shipping Module
  - Functions: Generate shipping labels, track shipments.

2. Inventory Management Workflow

Description: Automate the management of product inventory, including restocking and stock level alerts.

Steps:
1. Inventory Monitoring: Regularly check inventory levels.
2. Restock Notification: Alert when stock levels fall below a threshold.
3. Supplier Order: Automatically place orders with suppliers for restocking.
4. Inventory Update: Update stock levels upon receipt of new inventory.
5. Report Generation: Generate regular inventory reports.

Modules:
- Inventory Monitoring Module
  - Functions: Monitor stock levels, trigger restock alerts.
- Supplier Management Module
  - Functions: Place orders, manage supplier details.
- Inventory Update Module
  - Functions: Update stock levels, handle inventory receipt.
- Reporting Module
  - Functions: Generate and distribute inventory reports.

3. Customer Support Workflow

Description: Automate the handling of customer support tickets and inquiries.

Steps:
1. Ticket Creation: Automatically create support tickets from customer inquiries.
2. Ticket Categorization: Categorize and prioritize tickets.

3. Auto-Response: Send automated responses for common issues.
4. Ticket Assignment: Assign tickets to support agents.
5. Resolution Tracking: Track the resolution progress.
6. Customer Follow-up: Send follow-up emails to customers post-resolution.
7. Feedback Collection: Collect feedback from customers.

Modules:
- Ticket Management Module
  - Functions: Create and categorize tickets, update ticket status.

- Auto-Response Module
  - Functions: Generate responses for common issues.

- Assignment Module
  - Functions: Assign tickets to agents.

- Follow-up Module
  - Functions: Send follow-up communications, collect feedback.

Detailed Module Specifications:

❖ Order Management Module
- Functions:
  - `validateOrder(orderID)`: Checks if the order details are complete and valid.
  - `updateOrderStatus(orderID, status)`: Updates the status of an order.

❖ Inventory Management Module
- Functions:
  - `checkStock(productID)`: Returns the current stock level of a product.
  - `updateInventory(productID, quantity)`: Updates the inventory after a sale or restock.

❖ Payment Processing Module
- Functions:
  - `processPayment(orderID, paymentDetails)`: Processes the payment for an order.
  - `handlePaymentFailure(orderID)`: Handles scenarios where payment processing fails.

❖ Notification Module

- Functions:
  - `sendEmail(recipient, subject, body)`: Sends an email notification.
  - `sendSMS(phoneNumber, message)`: Sends an SMS notification.

❖ Shipping Module
- Functions:
  - `generateShippingLabel(orderID)`: Generates a shipping label for the order.
  - `trackShipment(shippingID)`: Provides tracking information for a shipment.

By automating these workflows, the RPA system can streamline operations, reduce manual errors, and improve efficiency in the e-commerce platform.

4.6 Input and Output Requirements for RPA

Robotic Process Automation (RPA) systems require clearly defined input and output specifications to function effectively. These specifications ensure that the RPA bots interact correctly with other systems, handle data appropriately, and produce the desired results. Properly defining these requirements is crucial for the success of RPA implementations, as it ensures that the automated processes align with business objectives and operate seamlessly within the existing IT infrastructure.

4.6.1 Output Specifications for RPA Processes

Output specifications define what the RPA bots should produce as a result of their automated tasks. These outputs must meet certain criteria to be considered successful and beneficial for the business process.

1. Accuracy:
   - The output data generated by RPA bots must be accurate and free of errors. This is especially critical for tasks such as financial transactions, data entry, and reporting, where precision is paramount.

2. Format:
   - Outputs should be in the required format compatible with other systems or processes. For instance, financial data should be formatted according to accounting standards, and reports should follow the specified templates.

3. Timeliness:
- Outputs must be produced within the expected timeframes. RPA bots should meet performance benchmarks to ensure that business processes are not delayed.

4. Consistency:
- The outputs should be consistent across different instances of the same process. This helps in maintaining uniformity and reliability in operations.

5. Completeness:
- All required data should be included in the output. Incomplete outputs can lead to process failures or the need for additional manual intervention, which negates the benefits of automation.

6. Compliance:
- Outputs should comply with regulatory and organizational standards. For example, data handling and reporting should adhere to GDPR or other relevant regulations.

4.6.2 Input Specifications for RPA Processes

Input specifications define the data and information required by RPA bots to perform their tasks. These inputs must be accurately defined and provided to ensure the smooth operation of the automated processes.

1. Data Sources:
- Identify and define the data sources from which the RPA bots will retrieve information. This includes databases, spreadsheets, web services, and other repositories.

2. Data Format:
- Inputs must be provided in a format that the RPA bots can process. This may involve specifying the structure of data files, such as CSV, JSON, XML, or database records.

3. Data Accuracy:
- Ensure that the input data is accurate and validated before processing. Inaccurate inputs can lead to errors in the output and may compromise the integrity of the automated process.

4. Completeness:
   - Inputs must be complete, providing all necessary information required for the process. Incomplete data can cause the RPA bots to fail or produce incorrect outputs.

5. Data Accessibility:
   - RPA bots need uninterrupted access to the data sources. This involves setting appropriate permissions and ensuring that the bots can access the required data without manual intervention.

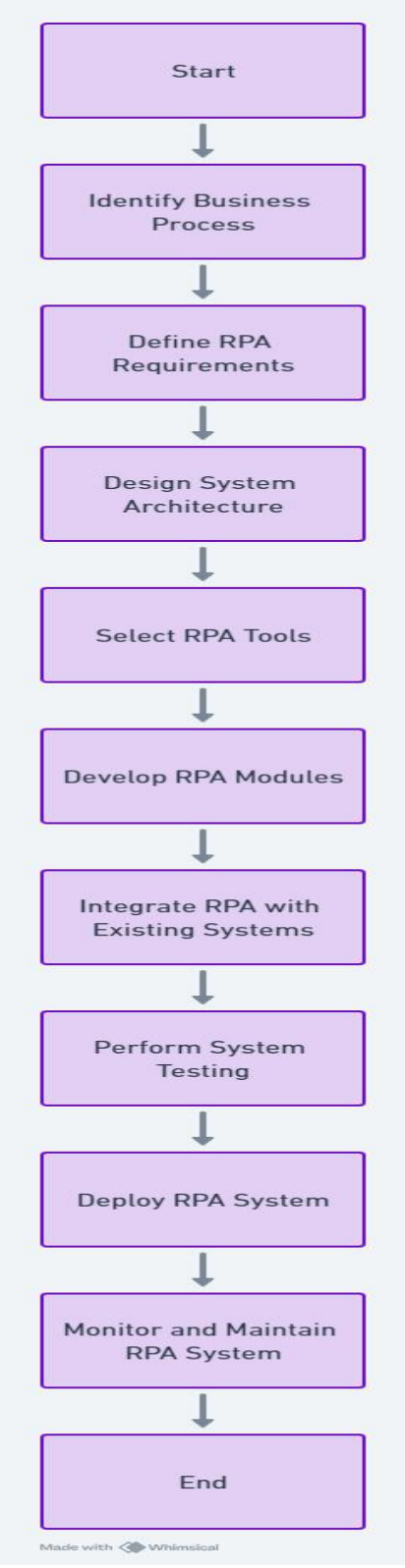
6. Real-time Data:
   - For processes that require real-time or near-real-time data, ensure that the inputs are updated and available as needed. This is crucial for processes like order processing and customer notifications (Marr, 2018).

Conclusion

Defining clear input and output specifications is critical for the successful implementation of RPA. By ensuring that inputs are accurate, complete, and accessible, and that outputs are accurate, timely, and compliant, businesses can maximize the benefits of RPA and achieve significant improvements in efficiency and effectiveness.

4.7 System Flowchart for RPA Implementation

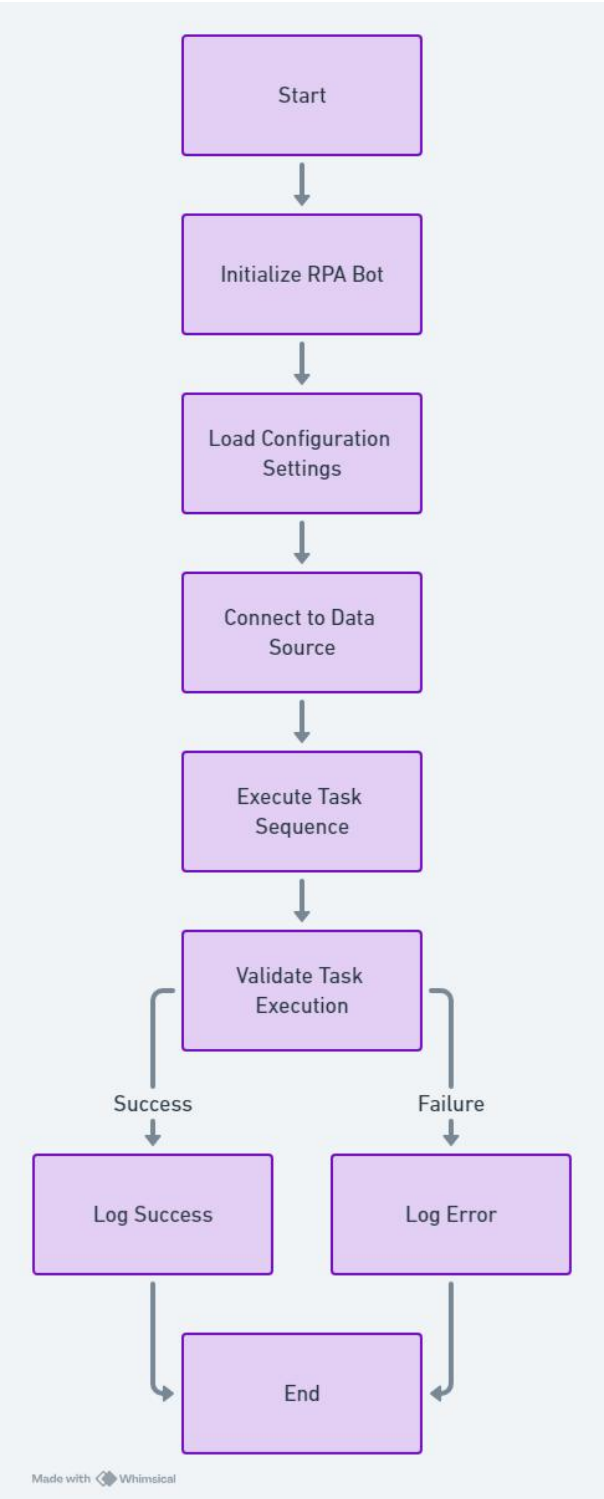
This flowchart outlines the steps involved in implementing an RPA system. It starts with identifying the business process to be automated and ends with monitoring and maintaining the deployed RPA system. The key steps include defining requirements, designing system architecture, selecting tools, developing and integrating RPA modules, performing system testing, and finally deploying the RPA system.



4.8 Program Flowchart for RPA Bots

This flowchart describes the program flow for an RPA bot. The process begins with initializing the RPA bot and loading configuration settings. The bot then connects to the data source, executes the task sequence, and

validates task execution. Depending on the success or failure of the task, it logs the result and ends the process.



4.9 Data Dictionary for RPA

A data dictionary is a critical component in any data management system, including RPA (Robotic Process Automation). It defines the structure, type, and constraints of data elements used in the system, ensuring consistency and clarity. Here is a comprehensive data dictionary for an RPA system designed for an e-commerce platform.

❖ Data Elements for an e-commerce platform

Field Name	Data Type	Description	Constraints
CustomerID	Integer	Unique identifier for each customer	Primary Key, Auto Increment
CustomerName	String	Full name of the customer	Not Null, Max 100 characters
CustomerEmail	String	Email address of the customer	Not Null, Max 100 characters, Unique
CustomerPhone	String	Contact phone number of the customer	Max 15 characters
ProductID	Integer	Unique identifier for each product	Primary Key, Auto Increment
ProductName	String	Name of the product	Not Null, Max 100 characters
ProductDescription	String	Detailed description of the product	Max 500 characters
ProductPrice	Decimal	Price of the product	Not Null, Positive value
StockQuantity	Integer	Available stock quantity of the product	Not Null, Non-negative
OrderID	Integer	Unique identifier for each order	Primary Key, Auto Increment
OrderDate	DateTime	Date and time when the order was placed	Not Null
OrderStatus	String	Current status of the order (e.g., Pending, Processed, Shipped)	Not Null
PaymentID	Integer	Unique identifier for each payment	Primary Key, Auto Increment
PaymentMethod	String	Method of payment (e.g., Credit Card, PayPal)	Not Null
PaymentStatus	String	Status of the payment (e.g., Completed, Failed)	Not Null
ShippingAddress	String	Address where the order will be shipped	Not Null, Max 200 characters
NotificationID	Integer	Unique identifier for each notification	Primary Key, Auto Increment
NotificationType	String	Type of notification (e.g., Email, SMS)	Not Null
NotificationStatus	String	Status of the notification (e.g., Sent, Failed)	Not Null
InventoryUpdateID	Integer	Unique identifier for each inventory update	Primary Key, Auto Increment
UpdatedStockQuantity	Integer	New stock quantity after the update	Not Null, Non-negative

❖ Relationships

- Customer to Order: One-to-Many (A customer can place multiple orders)
- Order to Product: Many-to-Many (An order can contain multiple products, and a product can be part of multiple orders)
- Order to Payment: One-to-One (Each order has one associated payment)
- Order to Notification: One-to-Many (An order can trigger multiple notifications)
- Product to Inventory Update: One-to-Many (A product can have multiple inventory updates)

4.10 Choice of Programming Language for the RPA Solutions Website

For the development of an RPA solutions website, the choice of programming language and platform is crucial to ensure functionality, performance, scalability, and ease of maintenance. Based on these criteria, Python is the recommended programming language.

❖ Reasons for Choosing Python

1. Versatility and Ease of Use:
   - Python is known for its simplicity and readability, making it accessible for developers of all skill levels.
   - It allows rapid development and prototyping, which is essential for the dynamic needs of an RPA solutions website.

2. Extensive Libraries and Frameworks:
   - Python boasts a rich ecosystem of libraries and frameworks, such as Django and Flask for web development, which streamline the development process.
   - Libraries like Selenium, PyAutoGUI, and BeautifulSoup are particularly useful for RPA tasks such as web scraping, automation, and data extraction.

3. Integration Capabilities:
   - Python integrates well with other technologies and platforms, which is essential for building an RPA solution that needs to interact with various systems and APIs.
   - Its ability to handle RESTful APIs and connect with databases seamlessly makes it a suitable choice for backend development.

4. Strong Community and Support:
- Python has a vast and active community, providing extensive resources, tutorials, and third-party tools.
- This community support helps in troubleshooting, knowledge sharing, and staying updated with the latest advancements.

5. Scalability and Performance:
- Python, with its frameworks, can handle high traffic and large-scale applications.
- It supports modularity and code reuse, which enhances the maintainability and scalability of the website.

Summary

Choosing Python as the programming language for developing an RPA solutions website offers numerous advantages, including versatility, extensive libraries, integration capabilities, community support, and scalability. By leveraging Python and frameworks like Django, developers can build robust, scalable, and efficient RPA systems that automate critical e-commerce processes, enhancing operational efficiency and customer satisfaction.