Design and Implementation of an Enhanced Real-Time Computer Inventory Management System: A Case of Rwanda Revenue Authority

Innocent Nsabimana[1]; Dr. Bugingo Emmanuel (PhD)[2]

[1,2] Masters of Science with Honors in Information Technology at University of Kigali, Rwanda

Publication Date: 2025/05/01

Abstract; The continuous development of technology and increase in IT environments have created a demand for the creation of efficient systems to manage these IT assets. This research covered the design and implementation of an enhanced Real-Time Computer Inventory Management System that handles tracking, security, and optimization of resource problems of IT assets. This system uses Laravel to develop an intuitive interface and backend, while powershell scripts are utilized to automate the gathering of real-time data in hardware and software inventories (computer name once changed after manual registration, IP Address, Operating system details, RAM, storage, CPU, logged-in users, and some software information such as currently installed applications,…). This conceptual framework hence guided the research in a manner that underlined the integration of real-time data gathering, scalability, and security. The results of semi-structured interviews with IT professionals explore qualitative findings on the vulnerabilities of existing systems and the ability for real-time solutions to improve IT governance. Major findings bring out that scalable and secure inventory systems create better decision-making, the accuracy of asset records, real-time data, and operational efficiency. The thematic analysis was used to identify some of the recurring themes on stakeholder engagement, data accuracy, and automation regarding contemporary IT asset management practices. This upgrade enhanced both the quality and quantity of IT asset recording, as well as their management and maintenance processes, furthering the mission of RRA being "To efficiently mobilize revenue and facilitate trade through transparent and innovative services to drive economic growth." (RRA, Mission and Vision statement, 2024)

***Keywords**: Power Shell, Laravel, CPU, Operating System.*

How to Cite: Innocent Nsabimana; Dr. Bugingo Emmanuel (PhD); (2025). Design and Implementation of an Enhanced Real-Time Computer Inventory Management System: A Case of Rwanda Revenue Authority. *International Journal of Innovative Science and Research Technology*, 10(4), 1936-1954. https://doi.org/10.38124/ijisrt/25apr1009

I. INTRODUCTION

The rapid advancement of technology and the expansion of IT infrastructures have made the efficient management of computer assets a critical concern for modern organizations. In public institutions like the Rwanda Revenue Authority (RRA), where transparency, accountability, and operational efficiency are paramount, manual or semi-automated inventory systems are no longer sufficient. These systems often suffer from data inaccuracies, lack of real-time updates, and inefficiencies in asset tracking and reporting.

This research focuses on the design and implementation of an enhanced Real-Time Computer Inventory Management System, aimed at overcoming the limitations of the existing system at RRA. By integrating PowerShell scripts for automated data collection and Laravel as a robust backend framework, the proposed system offers real-time tracking, centralized asset management, and improved data accuracy. It addresses key challenges such as delayed updates, manual data entry, and poor system integration, thereby improving IT governance and supporting informed decision-making.

The system was developed following a qualitative research approach, engaging IT professionals, administrators, and logistics staff to understand existing challenges and expectations. The findings highlight the importance of automation, real-time visibility, and security in managing IT assets effectively. This study not only enhances the internal operations of RRA but also contributes to broader discussions on modernizing IT asset management in public institutions.

➢ *Problem Statement*

With the rapid growth of technology and increasing reliance on IT infrastructure, public institutions like the Rwanda Revenue Authority (RRA) face growing challenges in managing their computer assets efficiently. The current inventory system at RRA is largely manual or semi-automated, which leads to delays in updates, inaccurate data, and difficulties in tracking hardware and software changes. These inefficiencies hinder timely decision-making, reduce transparency, and expose the institution to compliance and security risks.

This research addresses these issues by designing and implementing an enhanced Real-Time Computer Inventory Management System. The proposed system uses PowerShell scripts to automate the collection of system information and Laravel to provide a secure, user-friendly interface. By introducing automation and real-time data tracking, the new system minimizes manual entry, improves data accuracy, and integrates more smoothly with existing operations.

The solution was developed with input from IT and logistics staff to ensure it meets organizational needs. Ultimately, this system aims to boost operational efficiency, strengthen IT governance, and support RRA's goal of offering transparent and innovative public services.

II. LITERATURE REVIEW

The literature review explores real-time computer inventory management systems (RTCIMS) with a focus on their theoretical foundations, technical components, and practical applications. These systems are critical for enhancing IT asset visibility, minimizing operational inefficiencies, and supporting informed decision-making, particularly within government institutions such as the Rwanda Revenue Authority (RRA).

From a theoretical perspective, Information Systems Theory emphasizes the importance of real-time data in improving productivity and operational efficiency. The Resource-Based View further suggests that properly managed IT resources can provide a competitive advantage. Automation Theory supports the reduction of manual processes through tools such as PowerShell, which enhances the accuracy and efficiency of IT asset tracking. Real-Time Systems Theory highlights the value of instantaneous data processing in enabling swift responses to organizational changes. Additionally, Relational Database Models, particularly Codd's model, underpin modern database systems like MySQL, ensuring data integrity and consistency in dynamic environments. Web development frameworks such as Laravel, built on the MVC architecture, contribute to the scalability, maintainability, and user-friendliness of RTCIMS platforms.

Key concepts in RTCIMS include real-time tracking, scalability, system integration, and security. These systems enable automated asset discovery, centralized data management, and proactive maintenance strategies. Integration with helpdesk tools, CMDBs, and financial systems ensures a comprehensive and synchronized IT environment. Features like encryption, authentication, and Data Loss Prevention (DLP) are essential for securing sensitive asset data. Real-time tracking and updates reduce the reliance on manual audits and enhance asset accuracy. The transition from manual to automated systems marks a significant evolution, leading to increased operational efficiency and reduced costs.

Empirical studies confirm the benefits of real-time IT asset management. Automated discovery tools have reduced asset-related data inaccuracies, while integration with other IT systems has improved issue resolution times and reduced redundant procurements. Real-time systems contribute to improved compliance and security by enabling continuous monitoring and timely response to threats. Financially, organizations implementing RTCIMS report substantial cost savings and returns on investment, driven by optimized resource allocation and reduced asset downtime. Scalability ensures these systems can grow with organizational needs, while user-friendly interfaces increase adoption and satisfaction.

The application of RTCIMS in government settings, such as the RRA, enhances transparency, accountability, and public trust. Technologies like Laravel, MySQL, PowerShell, and real-time notification tools contribute to building efficient, secure, and scalable systems. As public institutions continue to evolve digitally, RTCIMS offer a reliable framework for effective IT asset management.

III. CONCEPTUAL FRAMEWORK

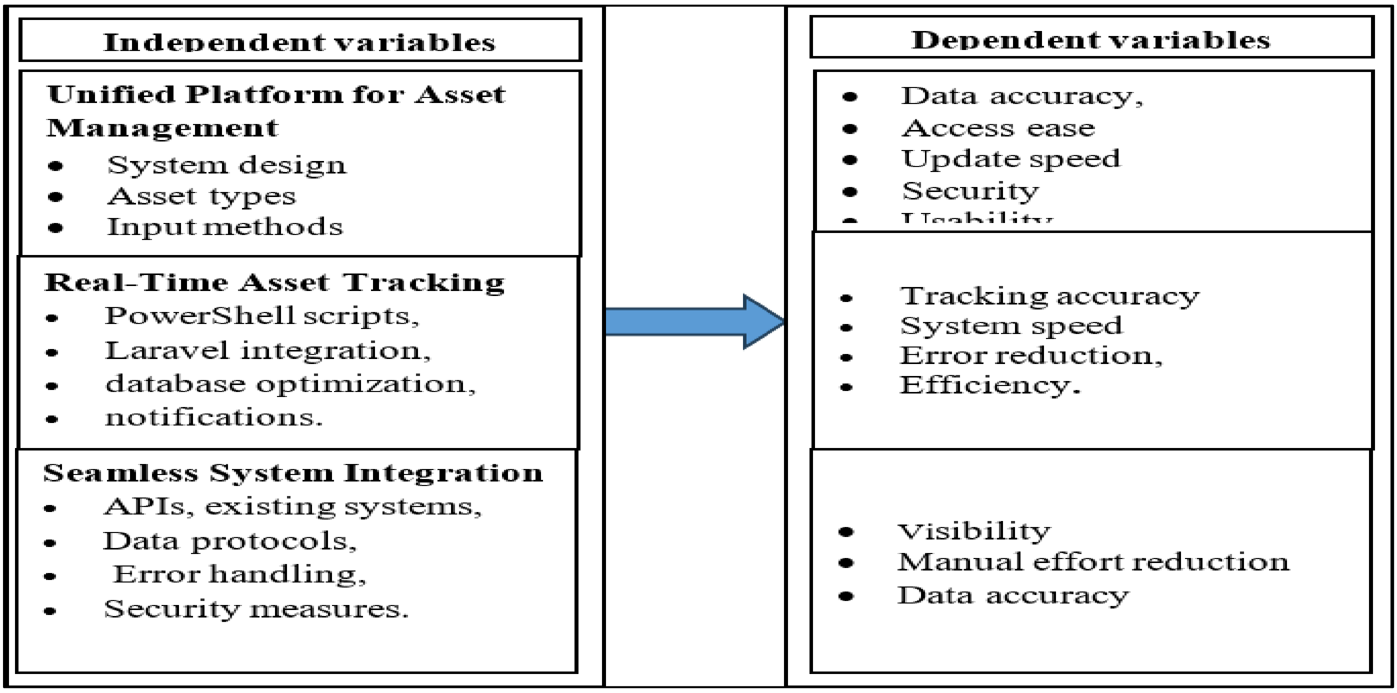


Fig 1 Conceptual Framework

➢ *Existing Model*

Big institutions currently utilize an IT Asset management system that helps in tracking and managing its computer inventory. This system likely incorporates a mix of manual process, basic software tools, or a centralized database form maintaining asset records.

The existing system uses manual entry to records IT equipment from the procurement to disposal where IT manager has to register manually the new equipment on reception date and later he keeps updating according to the new transaction.

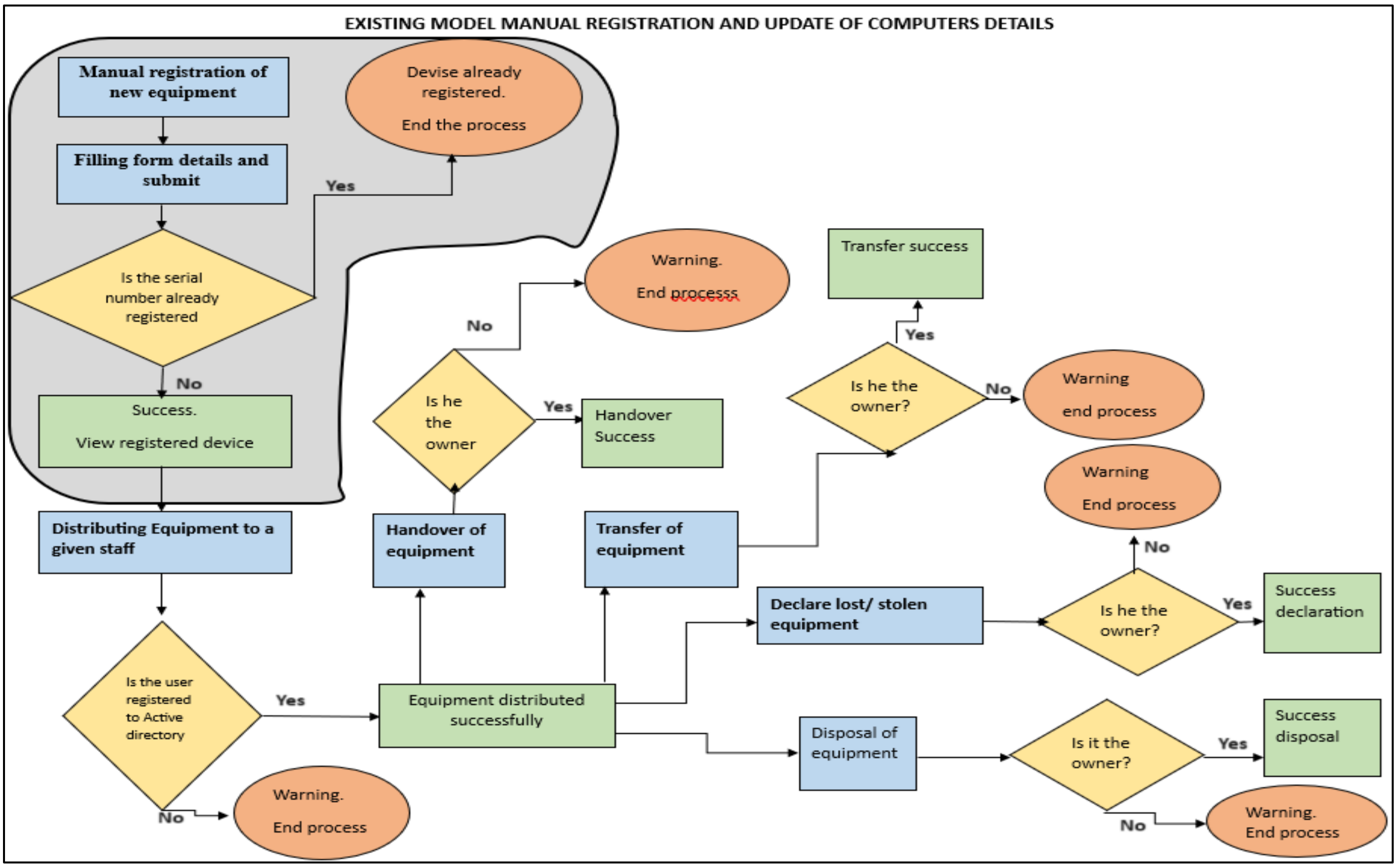


Fig 2 Existing model

➢ *Challenges and limitations*

- The current system might rely on periodic updated, leading to outdated or inaccurate data.
- Heavily reliance on human intervention, increasing the risk of errors and inefficiencies
- The system may not automatically detect new or disposed devices, making it hard to track changes dynamically.
- The system may not integrate with other tools like procurement platforms, HR systems or network monitoring tools
- It does not provide detailed, real-time insights into hardware specifications or software licenses in use

✓ **New proposed system model**

The proposed system introduces automation, real-time monitoring, and smart alerts to improve IT asset management at RRA. It uses PowerShell scripts and Laravel to track hardware and software changes, maintain a centralized database, and support integration with existing platforms, ensuring timely updates and proactive issue handling.

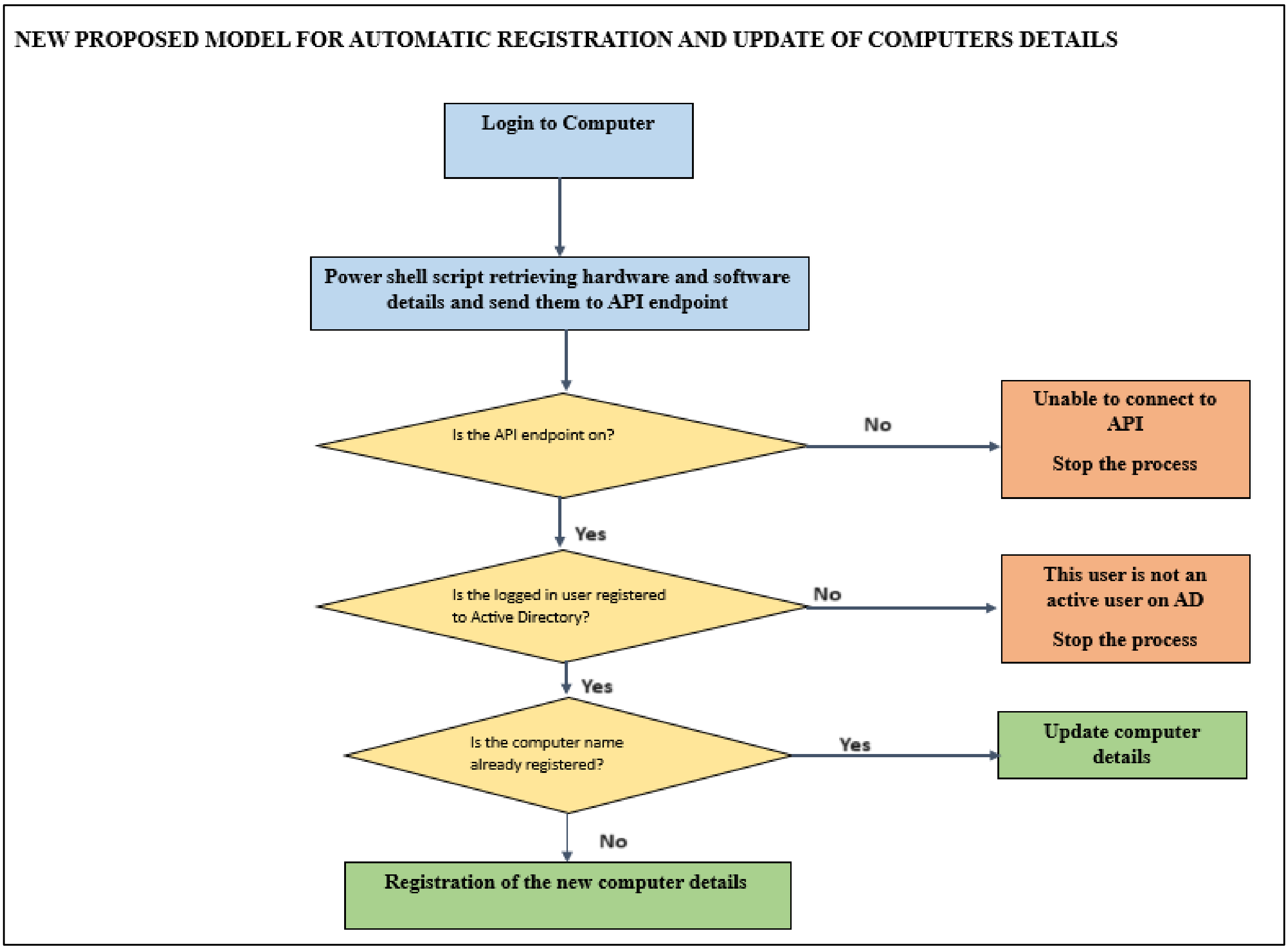

NEW PROPOSED MODEL FOR AUTOMATIC REGISTRATION AND UPDATE OF COMPUTERS DETAILS
Login to Computer
Power shell script retrieving hardware and software details and send them to API endpoint
Is the API endpoint on?
No
Unable to connect to API
Stop the process
Yes
Is the logged in user registered to Active Directory?
No
This user is not an active user on AD
Stop the process
Yes
Is the computer name already registered?
Yes
Update computer details
No
Registration of the new computer details

Fig 3 New Proposed Model

IV. RESEARCH METHODOLOGY

This study adopted a qualitative research methodology to design and implement a real-time computer inventory management system tailored for the Rwanda Revenue Authority (RRA). The chosen approach focused on exploring stakeholders' experiences, challenges, and expectations regarding IT asset management, emphasizing in-depth understanding rather than hypothesis testing.

A qualitative research design was employed to gather rich, descriptive insights from professionals directly involved in IT asset management at the RRA. The research population consisted of 119 individuals from two departments: the IT and Digital Transformation Department (87) and the Administration and Logistics Department (32). Purposive sampling was used to select participants based on their expertise and relevance to the topic.

Data were collected through semi-structured interviews and questionnaires. Semi-structured interviews allowed for flexible, open-ended discussions, enabling participants to express their views freely and explore topics beyond predefined questions. Primary data was gathered directly from participants, while secondary data came from organizational reports, audits, and policy documents to provide contextual background.

Thematic analysis was used to analyze the qualitative data, identifying recurring themes and patterns within participants' responses. This method was suitable for uncovering nuanced insights into IT asset management practices and the potential impact of a real-time solution.

Ethical considerations were strictly adhered to, including informed consent, confidentiality, and participants' right to withdraw at any stage. The study also acknowledged limitations, including potential bias in responses and resource constraints in data collection. Measures such as anonymity assurance were put in place to enhance the credibility of the findings.

V. PRESENTATION AND ANALYSIS OF RESEARCH FINDINGS

The study presents findings based on qualitative insights from IT, administration, and logistics staff regarding a real-time inventory system. It highlights the limitations of the current manual asset tracking system at RRA, which lacks automation, real-time updates, and integration with other platforms. Users face delays, data inconsistencies, and security risks due to the system's inability to automatically detect changes. Thematic analysis emphasizes the need for a centralized, automated solution to improve efficiency, accuracy, and informed decision-making.

➢ *Flowchart of the Existing System*

The flowchart illustrates a manual inventory process where IT staff record asset details during registration, but updates rely on scheduled checks or user reports. This approach causes delays and inaccuracies due to the lack of real-time automation.

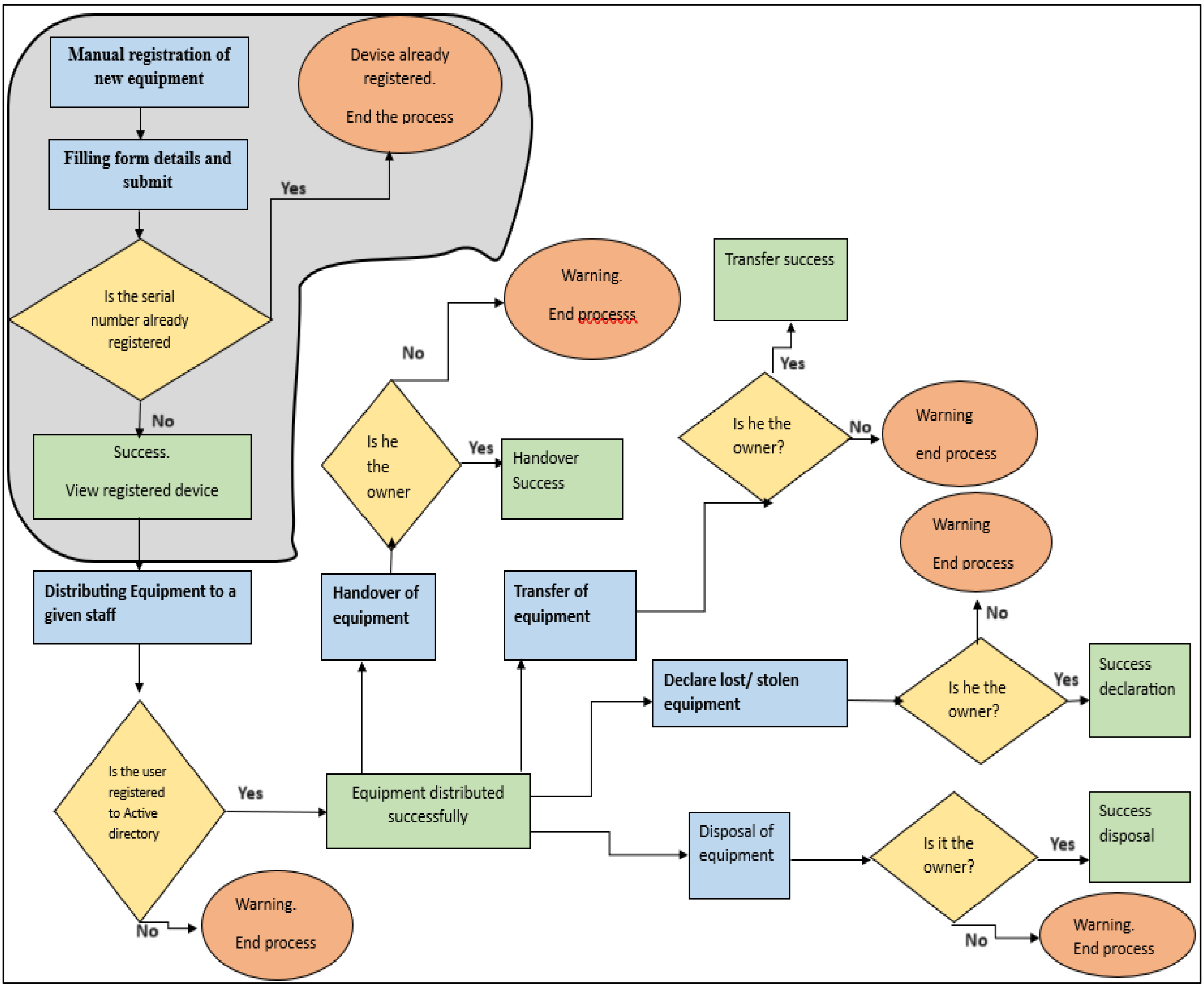


Fig 4 Existing System for IT Asset Management System

- *Flowchart of New Proposed Model*

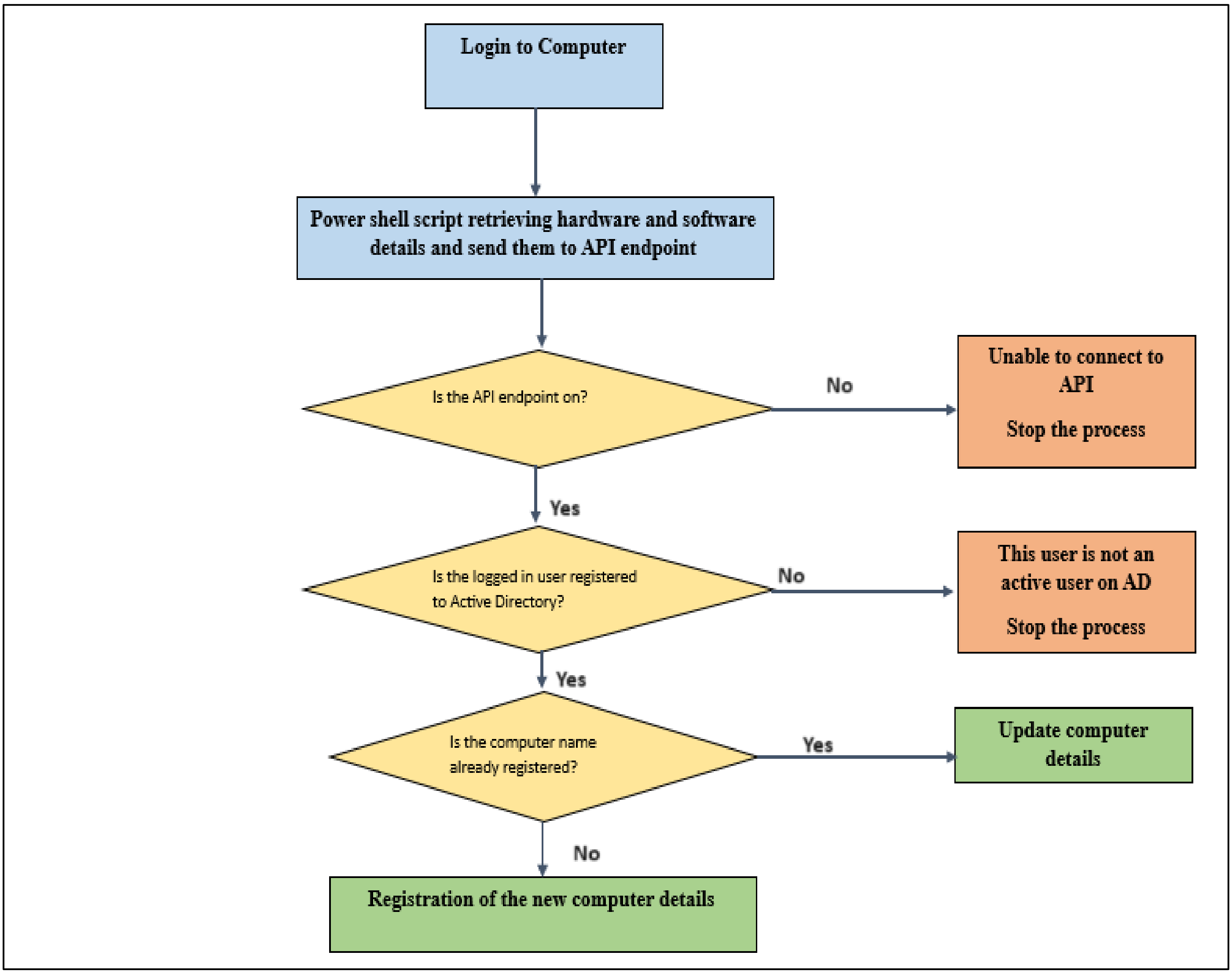

Login to Computer
Power shell script retrieving hardware and software details and send them to API endpoint
Is the API endpoint on?
No
Unable to connect to API
Stop the process
Yes
Is the logged in user registered to Active Directory?
No
This user is not an active user on AD
Stop the process
Yes
Is the computer name already registered?
Yes
Update computer details
No
Registration of the new computer details

Fig 5 New Proposed Model

- *Purpose and Importance of ULM*

UML (Unified Modeling Language) is essential in software development as it simplifies complex systems, enhances communication between technical and non-technical stakeholders, and promotes consistency among development teams. It includes structural diagrams like class and component diagrams, and behavioral diagrams such as use case and activity diagrams. In this project, key diagrams used include Class, Use Case, Sequence, and Activity diagrams to model and understand system behavior and requirements.

.

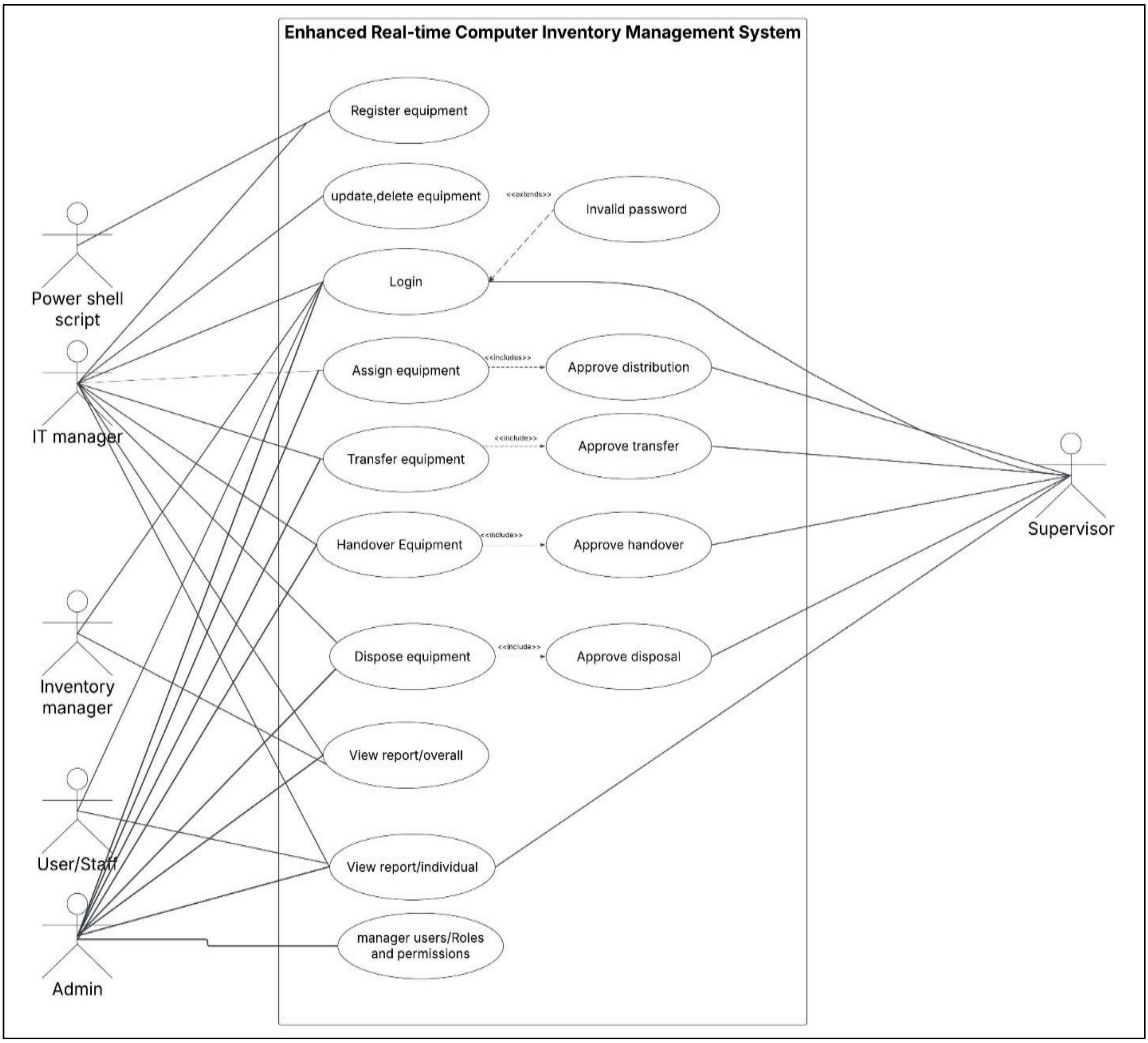


Fig 6 Use Case Diagram of the New Proposed Model

- *Class Diagram*

A class diagram is a core UML tool that illustrates the static structure of a system by defining classes, attributes, methods, and relationships like inheritance or association. It helps visualize object-oriented design, making systems easier to understand, develop, and maintain.

- *Importance of Class Diagrams in Software Engineering*

Class diagrams are crucial in modern software engineering as they enable teams to design and organize systems before actual coding begins. They also serve as a visual map for developers, enabling them to visualize the structure and relations of different components. Class diagrams also facilitate code maintainability, as they provide a clear view of the system, enabling code debugging and modification to be simpler in the long run.

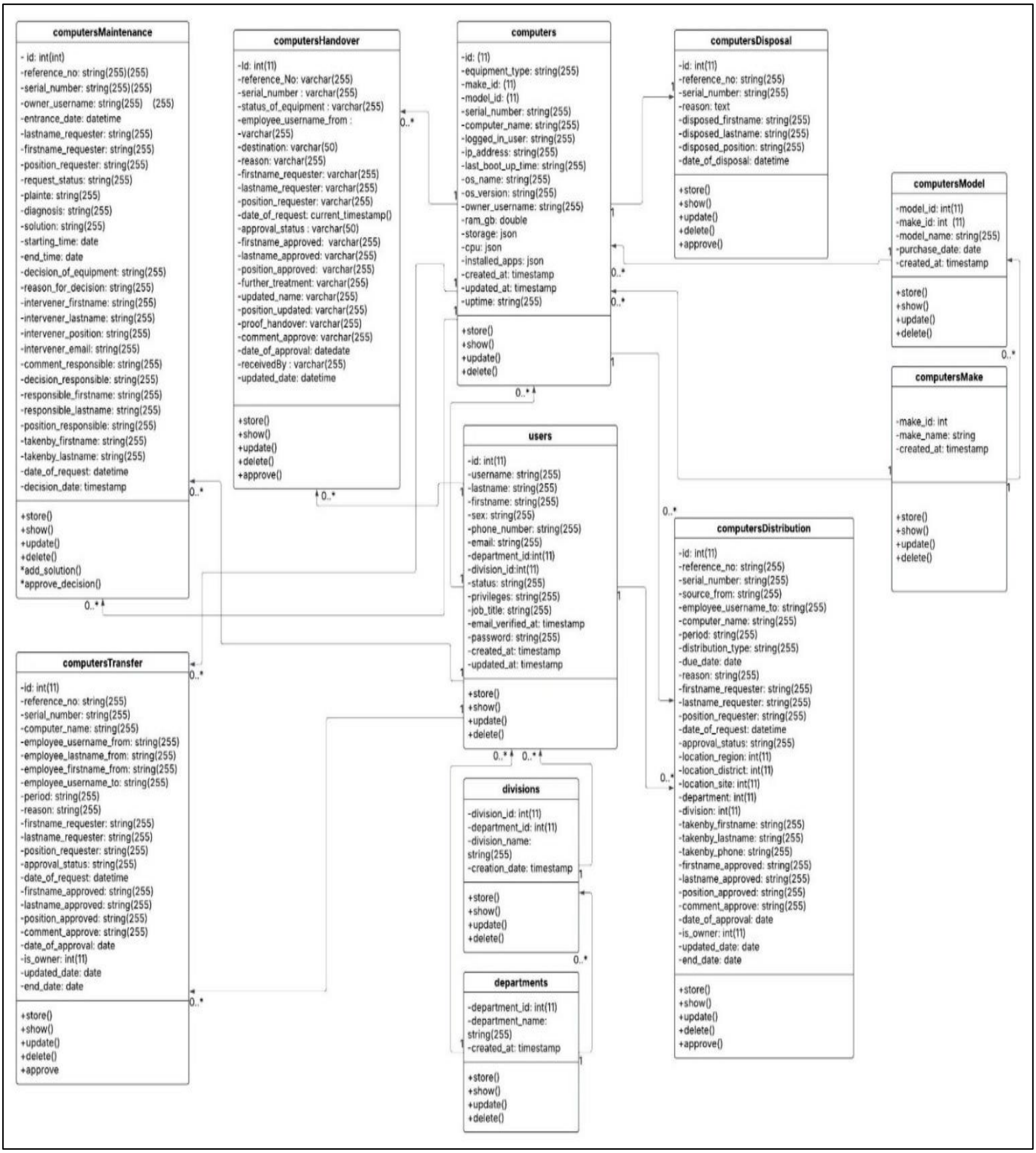

computersMaintenance
- id: int(int)
-reference_no: string(255)(255)
-serial_number: string(255)(255)
-owner_username: string(255) (255)
-entrance_date: datetime
-lastname_requester: string(255)
-firstname_requester: string(255)
-position_requester: string(255)
-request_status: string(255)
-plainte: string(255)
-diagnosis: string(255)
-solution: string(255)
-starting_time: date
-end_time: date
-decision_of_equipment: string(255)
-reason_for_decision: string(255)
-intervener_firstname: string(255)
-intervener_lastname: string(255)
-intervener_position: string(255)
-intervener_email: string(255)
-comment_responsible: string(255)
-decision_responsible: string(255)
-responsible_firstname: string(255)
-responsible_lastname: string(255)
-position_responsible: string(255)
-takenby_firstname: string(255)
-takenby_lastname: string(255)
-date_of_request: datetime
-decision_date: timestamp
+store()
+show()
+update()
+delete()
*add_solution()
*approve_decision()
computersHandover
-Id: int(11)
-reference_No: varchar(255)
-serial_number : varchar(255)
-status_of_equipment : varchar(255)
-employee_username_from :
-varchar(255)
-destination: varchar(50)
-reason: varchar(255)
-firstname_requester: varchar(255)
-lastname_requester: varchar(255)
-position_requester: varchar(255)
-date_of_request: current_timestamp()
-approval_status : varchar(50)
-firstname_approved: varchar(255)
-lastname_approved: varchar(255)
-position_approved: varchar(255)
-further_treatment: varchar(255)
-updated_name: varchar(255)
-position_updated: varchar(255)
-proof_handover: varchar(255)
-comment_approve: varchar(255)
-date_of_approval: datedate
-receivedBy : varchar(255)
-updated_date: datetime
+store()
+show()
+update()
+delete()
+approve()
computers
-id: (11)
-equipment_type: string(255)
-make_id: (11)
-model_id: (11)
-serial_number: string(255)
-computer_name: string(255)
-logged_in_user: string(255)
-ip_address: string(255)
-last_boot_up_time: string(255)
-os_name: string(255)
-os_version: string(255)
-owner_username: string(255)
-ram_gb: double
-storage: json
-cpu: json
-installed_apps: json
-created_at: timestamp
-updated_at: timestamp
-uptime: string(255)
+store()
+show()
+update()
+delete()
computersDisposal
-id: int(11)
-reference_no: string(255)
-serial_number: string(255)
-reason: text
-disposed_firstname: string(255)
-disposed_lastname: string(255)
-disposed_position: string(255)
-date_of_disposal: datetime
+store()
+show()
+update()
+delete()
+approve()
computersModel
-model_id: int(11)
-make_id: int (11)
-model_name: string(255)
-purchase_date: date
-created_at: timestamp
+store()
+show()
+update()
+delete()
computersMake
-make_id: int
-make_name: string
-created_at: timestamp
+store()
+show()
+update()
+delete()
users
-id: int(11)
-username: string(255)
-lastname: string(255)
-firstname: string(255)
-sex: string(255)
-phone_number: string(255)
-email: string(255)
-department_id:int(11)
-division_id:int(11)
-status: string(255)
-privileges: string(255)
-job_title: string(255)
-email_verified_at: timestamp
-password: string(255)
-created_at: timestamp
-updated_at: timestamp
+store()
+show()
+update()
+delete()
computersDistribution
-id: int(11)
-reference_no: string(255)
-serial_number: string(255)
-source_from: string(255)
-employee_username_to: string(255)
-computer_name: string(255)
-period: string(255)
-distribution_type: string(255)
-due_date: date
-reason: string(255)
-firstname_requester: string(255)
-lastname_requester: string(255)
-position_requester: string(255)
-date_of_request: datetime
-approval_status: string(255)
-location_region: int(11)
-location_district: int(11)
-location_site: int(11)
-department: int(11)
-division: int(11)
-takenby_firstname: string(255)
-takenby_lastname: string(255)
-takenby_phone: string(255)
-firstname_approved: string(255)
-lastname_approved: string(255)
-position_approved: string(255)
-comment_approve: string(255)
-date_of_approval: date
-is_owner: int(11)
-updated_date: date
-end_date: date
+store()
+show()
+update()
+delete()
+approve()
computersTransfer
-id: int(11)
-reference_no: string(255)
-serial_number: string(255)
-computer_name: string(255)
-employee_username_from: string(255)
-employee_lastname_from: string(255)
-employee_firstname_from: string(255)
-employee_username_to: string(255)
-period: string(255)
-reason: string(255)
-firstname_requester: string(255)
-lastname_requester: string(255)
-position_requester: string(255)
-approval_status: string(255)
-date_of_request: datetime
-firstname_approved: string(255)
-lastname_approved: string(255)
-position_approved: string(255)
-comment_approve: string(255)
-date_of_approval: date
-is_owner: int(11)
-updated_date: date
-end_date: date
+store()
+show()
+update()
+delete()
+approve
divisions
-division_id: int(11)
-department_id: int(11)
-division_name:
string(255)
-creation_date: timestamp
+store()
+show()
+update()
+delete()
departments
-department_id: int(11)
-department_name:
string(255)
-created_at: timestamp
+store()
+show()
+update()
+delete()
0..*
1

Fig 7 Class Diagram of the New System

- *Sequence Diagram*

- A sequence diagram is a UML tool that shows how objects and actors interact through message exchanges over time to accomplish specific system functions. It helps model the control flow and timing of interactions within use cases.

- *User Registration*

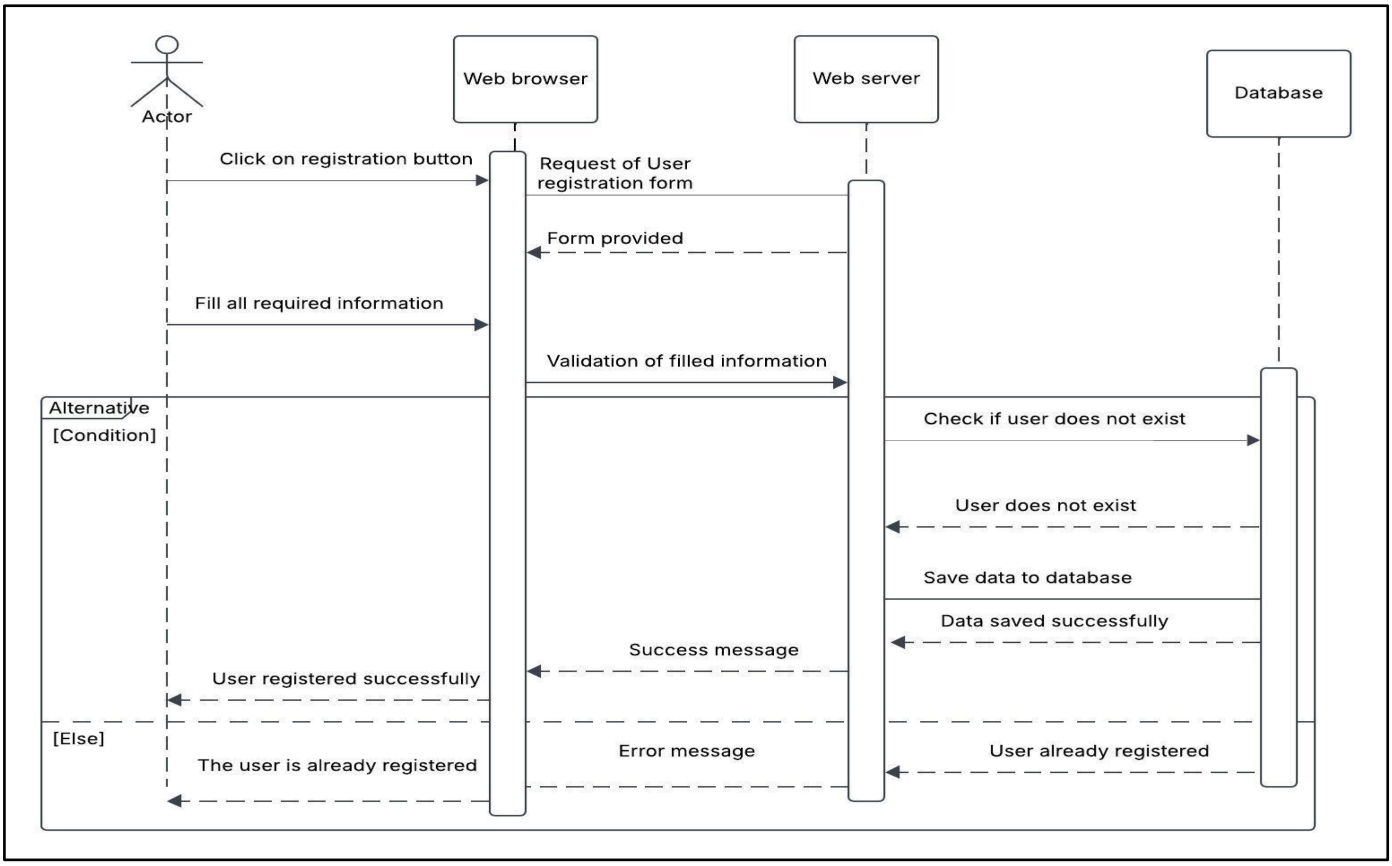

Actor
Web browser
Web server
Database
Click on registration button
Request of User registration form
Form provided
Fill all required information
Validation of filled information
Alternative
[Condition]
Check if user does not exist
User does not exist
Save data to database
Data saved successfully
Success message
User registered successfully
[Else]
Error message
User already registered
The user is already registered

Fig 8 Sequence Diagram of User Registration

- *Login*

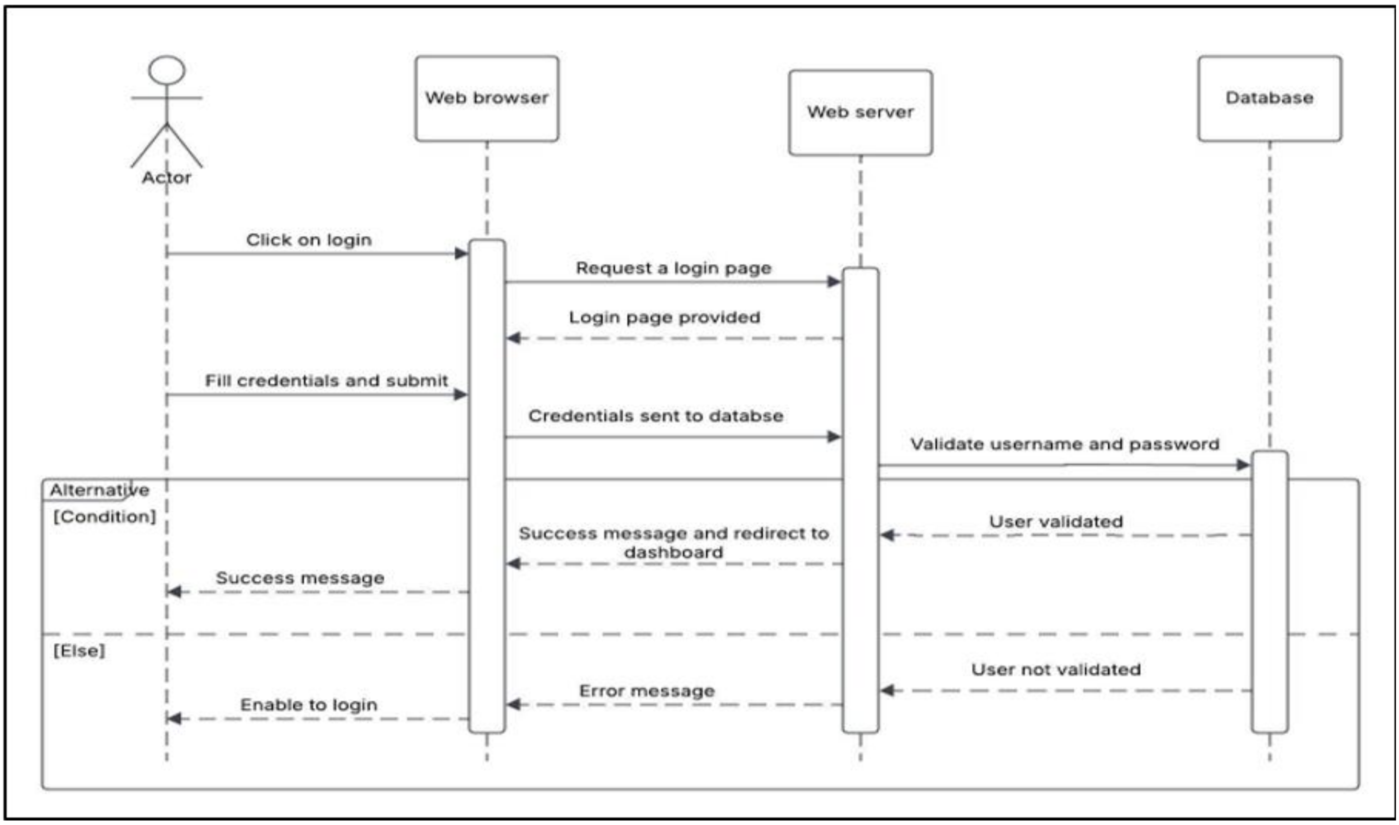


Fig 9 Login Sequence

➢ *Computer Registration*

- *Using API*

When a domain joined computer is booted at first time on the Institution network, all computer hardware and software information are retrieved by PowerShell script and sent to API end point and be saved in database using Laravel controller. When a computer serial number already exists, it updates the request otherwise it is saved as new record.

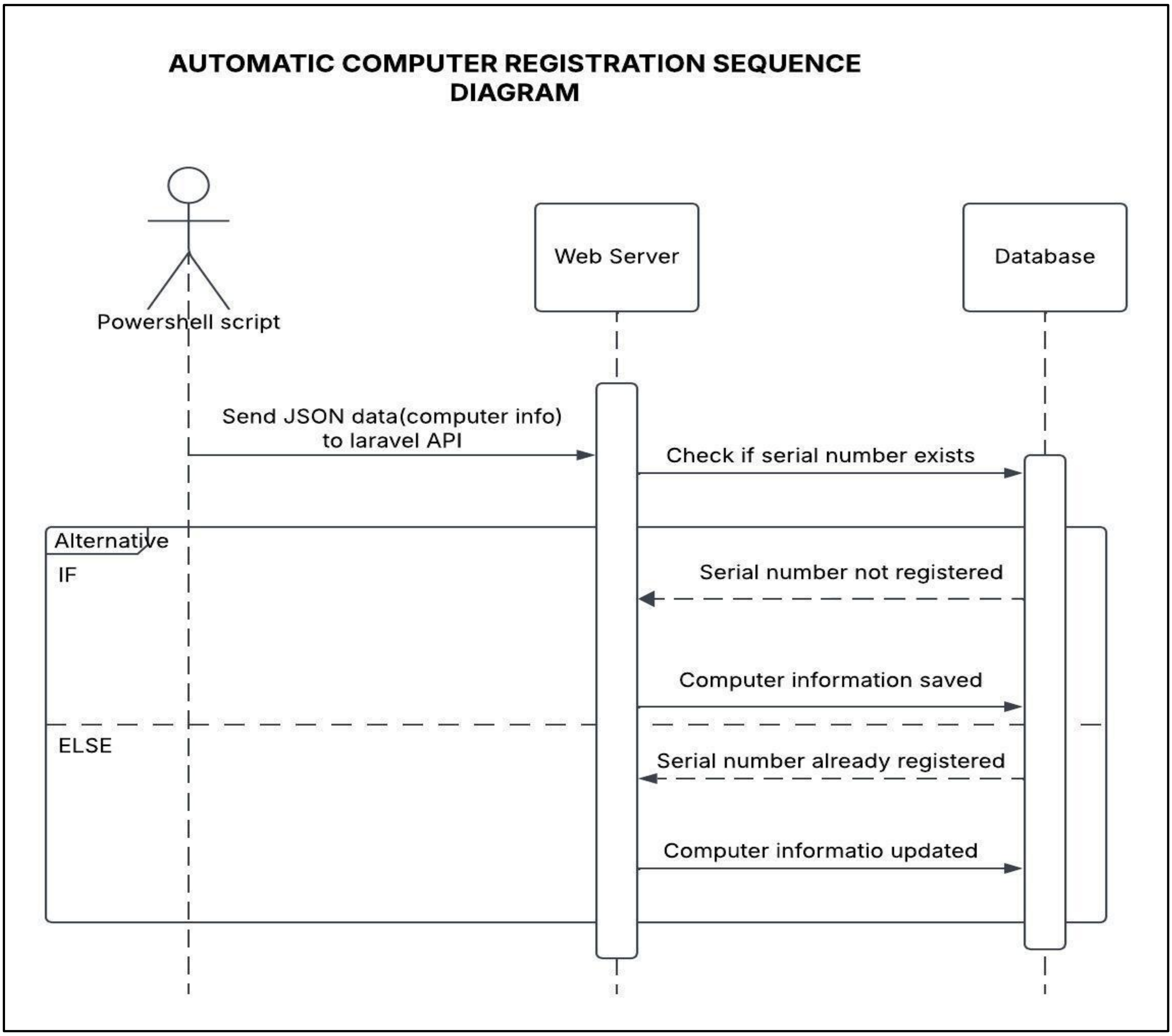

AUTOMATIC COMPUTER REGISTRATION SEQUENCE DIAGRAM
Powershell script
Web Server
Database
Send JSON data(computer info) to laravel API
Check if serial number exists
Alternative
IF
Serial number not registered
Computer information saved
ELSE
Serial number already registered
Computer informatio updated

Fig 10 Automatic Computer Registration

- *Using Interface*

An IT manager in charge of inventory can register a computer before it joins a domain. For example, new computers in store. He opens a web interface and fill in required information and then submit. After submission the Laravel controller validates all inputs and if no error it send data to database otherwise it shows the error message to the browser so that the IT manager can modify the wrong information accordingly.

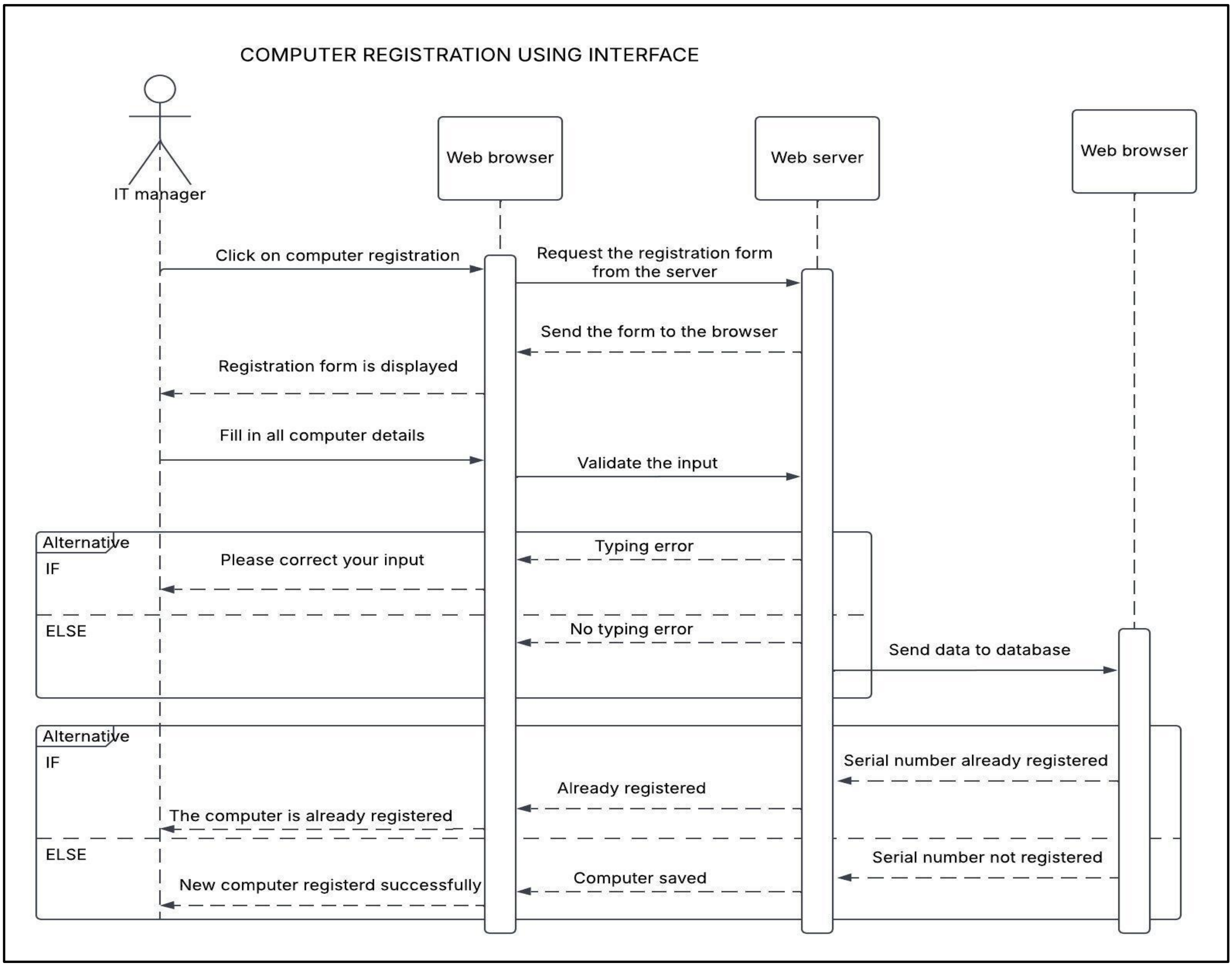

COMPUTER REGISTRATION USING INTERFACE
IT manager
Web browser
Web server
Web browser
Click on computer registration
Request the registration form from the server
Send the form to the browser
Registration form is displayed
Fill in all computer details
Validate the input
Alternative
IF
Typing error
Please correct your input
ELSE
No typing error
Send data to database
Alternative
IF
Serial number already registered
Already registered
The computer is already registered
ELSE
Serial number not registered
Computer saved
New computer registerd successfully

Fig 11 Computer Registration Using Interface

➢ *Computer distribution*

- *Request Initiation*

When a computer is already registered, the next step is to assign it to the user.

An IT manager is in charge of making request of all transactions related to computers (Registration, distribution, transfer, handover and disposal).

The user to assign to must be registered in the system otherwise the system returns an error message saying that the assigned user is not a registered user.

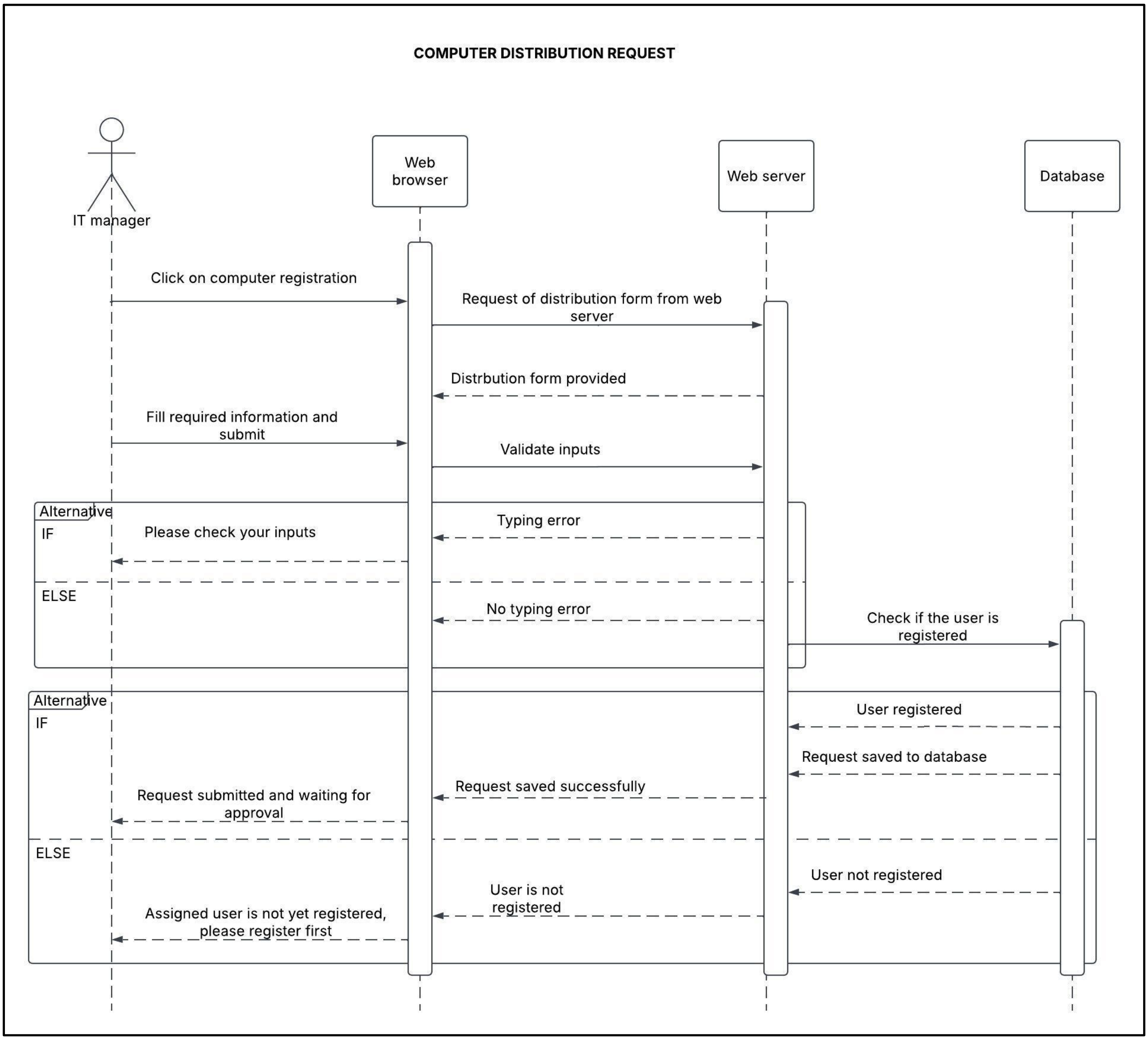

COMPUTER DISTRIBUTION REQUEST
IT manager
Web browser
Web server
Database
Click on computer registration
Request of distribution form from web server
Distrbution form provided
Fill required information and submit
Validate inputs
Alternative
IF
Please check your inputs
Typing error
ELSE
No typing error
Check if the user is registered
Alternative
IF
User registered
Request saved to database
Request saved successfully
Request submitted and waiting for approval
ELSE
User not registered
User is not registered
Assigned user is not yet registered, please register first

Fig 12 Computer Distribution Request

- *Request Approval*

After the request is sent, the next step is to approve. The supervisor goes to dashboard and search the pending request using reference no. If the reference no is valid the request details are displayed with the option of approve or reject. After choosing the decision, he submits the request and this submission changes the status of the request from Pending to Approved or Rejected accordingly.

The system sends notification to the user telling him/her to come and pick a computer and distribution form including the contact person.

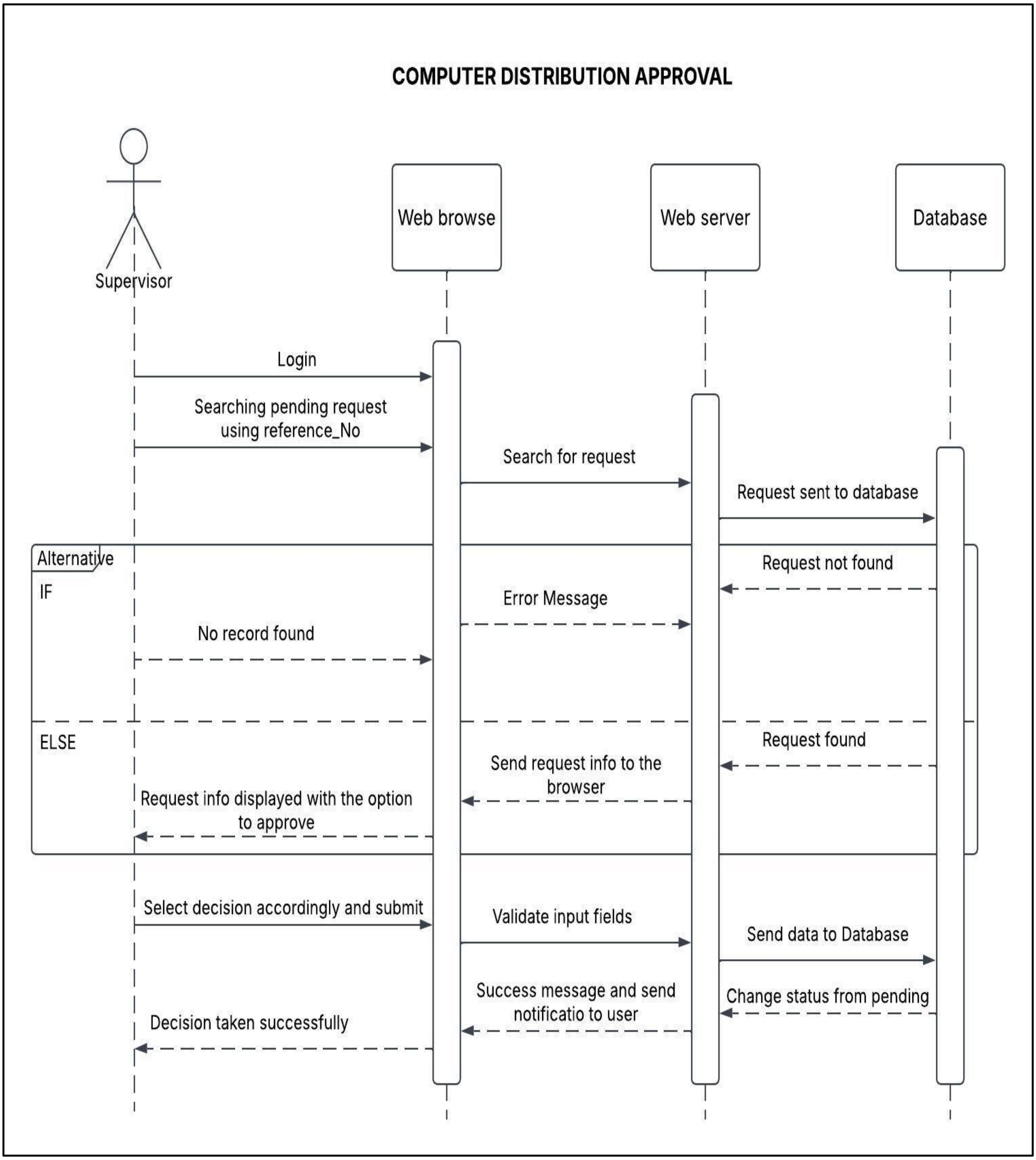
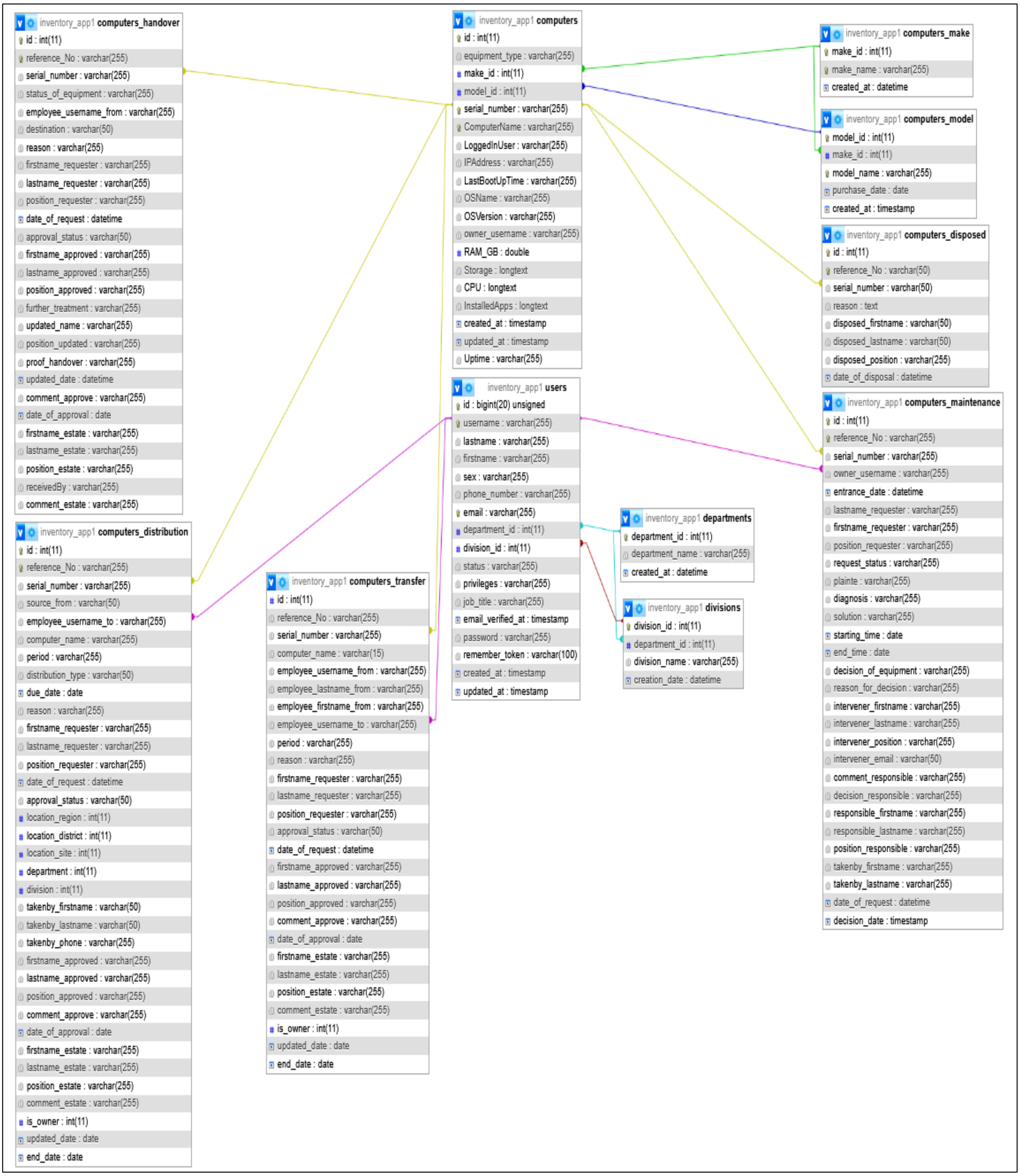

COMPUTER DISTRIBUTION APPROVAL
Supervisor
Web browse
Web server
Database
Login
Searching pending request using reference_No
Search for request
Request sent to database
Alternative
IF
Request not found
Error Message
No record found
ELSE
Request found
Send request info to the browser
Request info displayed with the option to approve
Select decision accordingly and submit
Validate input fields
Send data to Database
Change status from pending
Success message and send notificatio to user
Decision taken successfully

Fig 13 Computer Distribution Approval Sequence Diagram

➢ *Database diagram*

Database diagram is a graphic presentation of the database structure, which shows tables, columns, keys, and data entity associations (Elmasri & Navathe, 2020). It is an overview plan of database system design, understanding, and upkeep for data integrity, normalization, and efficiency (Connolly & Begg, 2019). Database diagrams are used heavily in relational database management systems (RDBMS) such as MySQL, PostgreSQL, Microsoft SQL Server, and Oracle (Silberschatz, Korth, & Sudarshan, 2021). In our case we have used MySQL database.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1009



inventory_app1 computers_handover
id : int(11)
reference_No : varchar(255)
serial_number : varchar(255)
status_of_equipment : varchar(255)
employee_username_from : varchar(255)
destination : varchar(50)
reason : varchar(255)
firstname_requester : varchar(255)
lastname_requester : varchar(255)
position_requester : varchar(255)
date_of_request : datetime
approval_status : varchar(50)
firstname_approved : varchar(255)
lastname_approved : varchar(255)
position_approved : varchar(255)
further_treatment : varchar(255)
updated_name : varchar(255)
position_updated : varchar(255)
proof_handover : varchar(255)
updated_date : datetime
comment_approve : varchar(255)
date_of_approval : date
firstname_estate : varchar(255)
lastname_estate : varchar(255)
position_estate : varchar(255)
receivedBy : varchar(255)
comment_estate : varchar(255)

inventory_app1 computers_distribution
id : int(11)
reference_No : varchar(255)
serial_number : varchar(255)
source_from : varchar(50)
employee_username_to : varchar(255)
computer_name : varchar(255)
period : varchar(255)
distribution_type : varchar(50)
due_date : date
reason : varchar(255)
firstname_requester : varchar(255)
lastname_requester : varchar(255)
position_requester : varchar(255)
date_of_request : datetime
approval_status : varchar(50)
location_region : int(11)
location_district : int(11)
location_site : int(11)
department : int(11)
division : int(11)
takenby_firstname : varchar(50)
takenby_lastname : varchar(50)
takenby_phone : varchar(255)
firstname_approved : varchar(255)
lastname_approved : varchar(255)
position_approved : varchar(255)
comment_approve : varchar(255)
date_of_approval : date
firstname_estate : varchar(255)
lastname_estate : varchar(255)
position_estate : varchar(255)
comment_estate : varchar(255)
is_owner : int(11)
updated_date : date
end_date : date

inventory_app1 computers_transfer
id : int(11)
reference_No : varchar(255)
serial_number : varchar(255)
computer_name : varchar(15)
employee_username_from : varchar(255)
employee_lastname_from : varchar(255)
employee_firstname_from : varchar(255)
employee_username_to : varchar(255)
period : varchar(255)
reason : varchar(255)
firstname_requester : varchar(255)
lastname_requester : varchar(255)
position_requester : varchar(255)
approval_status : varchar(50)
date_of_request : datetime
firstname_approved : varchar(255)
lastname_approved : varchar(255)
position_approved : varchar(255)
comment_approve : varchar(255)
date_of_approval : date
firstname_estate : varchar(255)
lastname_estate : varchar(255)
position_estate : varchar(255)
comment_estate : varchar(255)
is_owner : int(11)
updated_date : date
end_date : date

inventory_app1 computers
id : int(11)
equipment_type : varchar(255)
make_id : int(11)
model_id : int(11)
serial_number : varchar(255)
ComputerName : varchar(255)
LoggedInUser : varchar(255)
IPAddress : varchar(255)
LastBootUpTime : varchar(255)
OSName : varchar(255)
OSVersion : varchar(255)
owner_username : varchar(255)
RAM_GB : double
Storage : longtext
CPU : longtext
InstalledApps : longtext
created_at : timestamp
updated_at : timestamp
Uptime : varchar(255)

inventory_app1 users
id : bigint(20) unsigned
username : varchar(255)
lastname : varchar(255)
firstname : varchar(255)
sex : varchar(255)
phone_number : varchar(255)
email : varchar(255)
department_id : int(11)
division_id : int(11)
status : varchar(255)
privileges : varchar(255)
job_title : varchar(255)
email_verified_at : timestamp
password : varchar(255)
remember_token : varchar(100)
created_at : timestamp
updated_at : timestamp

inventory_app1 departments
department_id : int(11)
department_name : varchar(255)
created_at : datetime

inventory_app1 divisions
division_id : int(11)
department_id : int(11)
division_name : varchar(255)
creation_date : datetime

inventory_app1 computers_make
make_id : int(11)
make_name : varchar(255)
created_at : datetime

inventory_app1 computers_model
model_id : int(11)
make_id : int(11)
model_name : varchar(255)
purchase_date : date
created_at : timestamp

inventory_app1 computers_disposed
id : int(11)
reference_No : varchar(50)
serial_number : varchar(50)
reason : text
disposed_firstname : varchar(50)
disposed_lastname : varchar(50)
disposed_position : varchar(255)
date_of_disposal : datetime

inventory_app1 computers_maintenance
id : int(11)
reference_No : varchar(255)
serial_number : varchar(255)
owner_username : varchar(255)
entrance_date : datetime
lastname_requester : varchar(255)
firstname_requester : varchar(255)
position_requester : varchar(255)
request_status : varchar(255)
plainte : varchar(255)
diagnosis : varchar(255)
solution : varchar(255)
starting_time : date
end_time : date
decision_of_equipment : varchar(255)
reason_for_decision : varchar(255)
intervener_firstname : varchar(255)
intervener_lastname : varchar(255)
intervener_position : varchar(255)
intervener_email : varchar(50)
comment_responsible : varchar(255)
decision_responsible : varchar(255)
responsible_firstname : varchar(255)
responsible_lastname : varchar(255)
position_responsible : varchar(255)
takenby_firstname : varchar(255)
takenby_lastname : varchar(255)
date_of_request : datetime
decision_date : timestamp

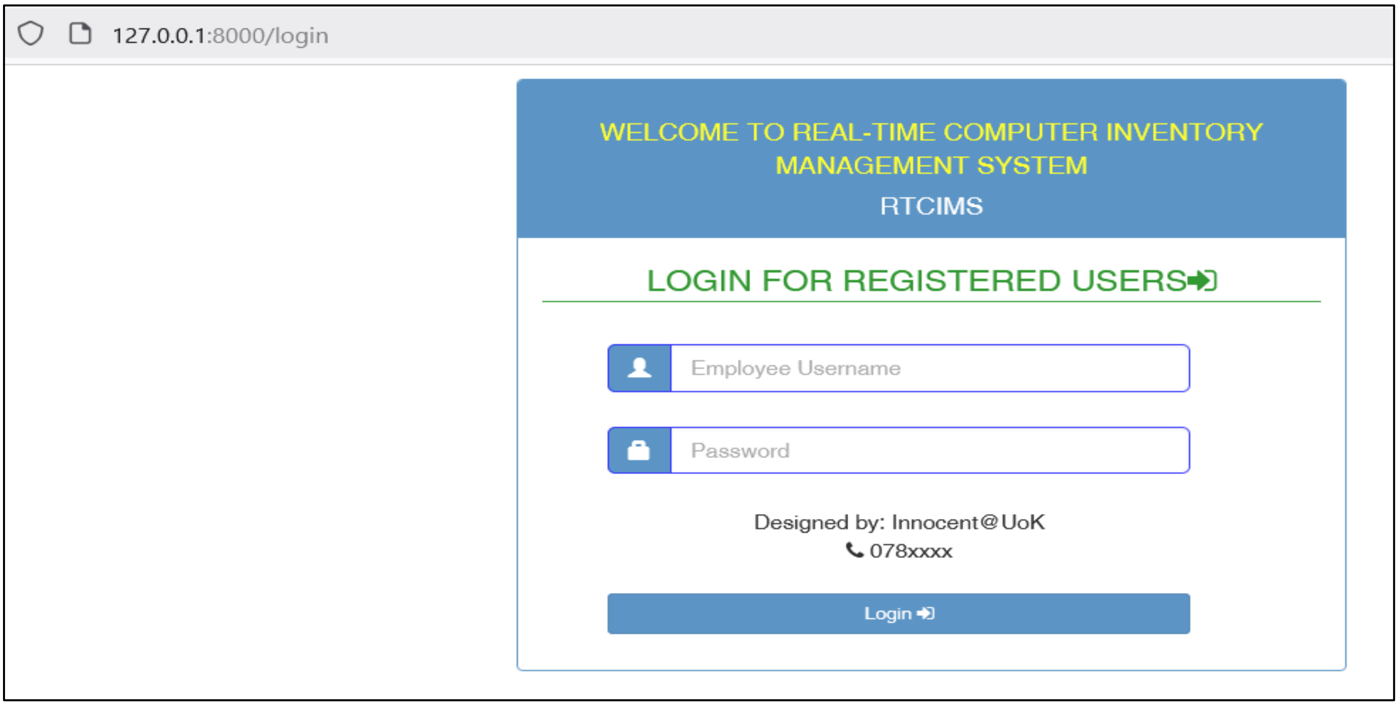
Fig 14 Database Diagram

➢ *Login Page*

A login page is a website or webpage that greets users to log in with credentials such as a username/email and password to gain access to an application, website, or system. It is an entry point for secured resources where authenticated users only can move ahead.

The following figure show the login page as the entry point of real-time computer inventory management system.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1009



127.0.0.1:8000/login

WELCOME TO REAL-TIME COMPUTER INVENTORY MANAGEMENT SYSTEM

RTCIMS

LOGIN FOR REGISTERED USERS

Employee Username

Password

Designed by: Innocent@UoK

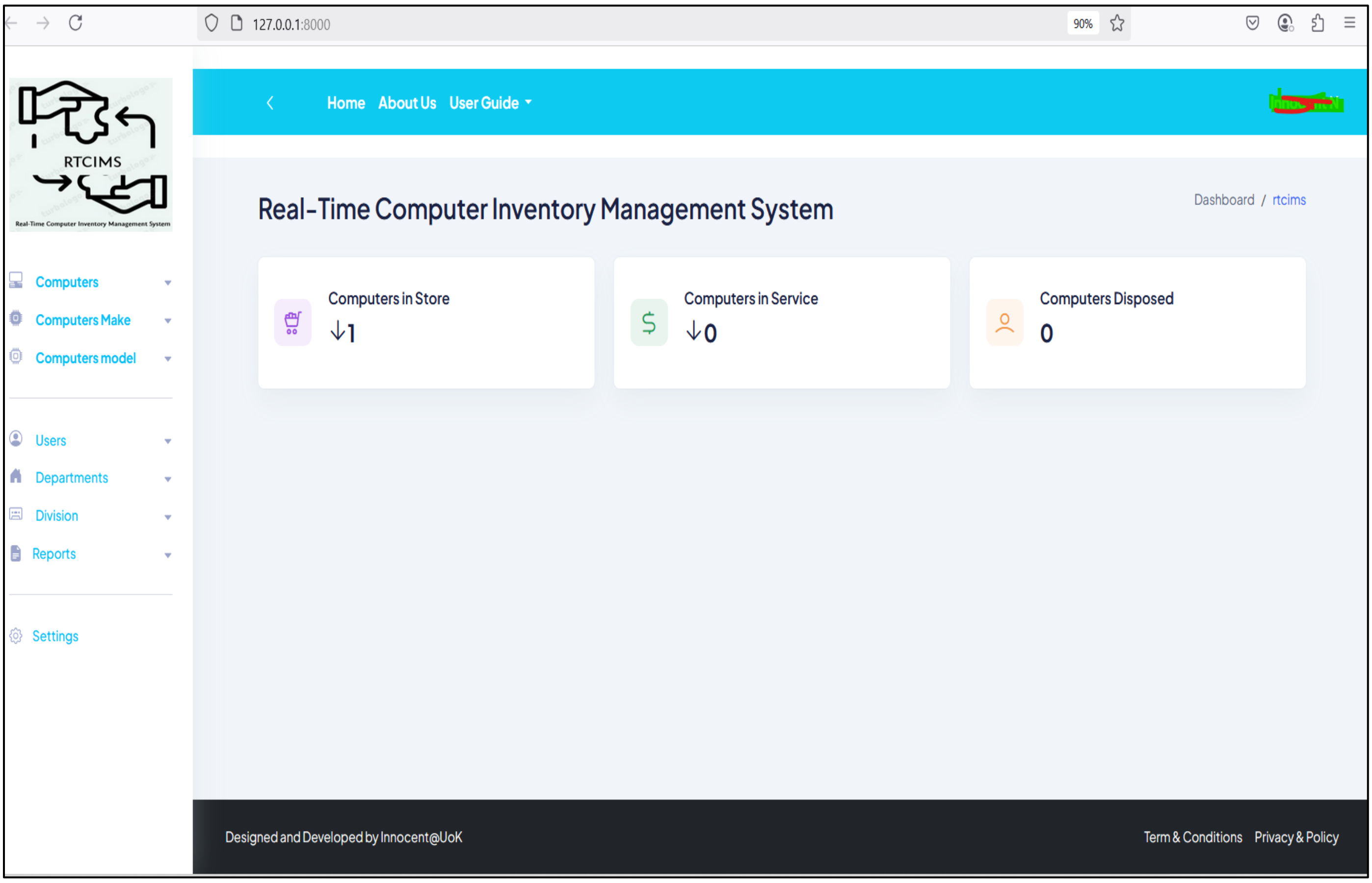
☎ 078xxxx

Login

Fig 15 Login Page

➢ *Landing Page/Dashboard*

A dashboard page is a graphical interface that presents key system data and functions in a centralized view, allowing users to monitor and interact with live updates efficiently. It typically includes charts, tables, alerts, and buttons to summarize and access various system features. Dashboards enhance decision-making and are widely used in IT asset management and other business systems.



127.0.0.1:8000

Real-Time Computer Inventory Management System

Computers in Store ↓1

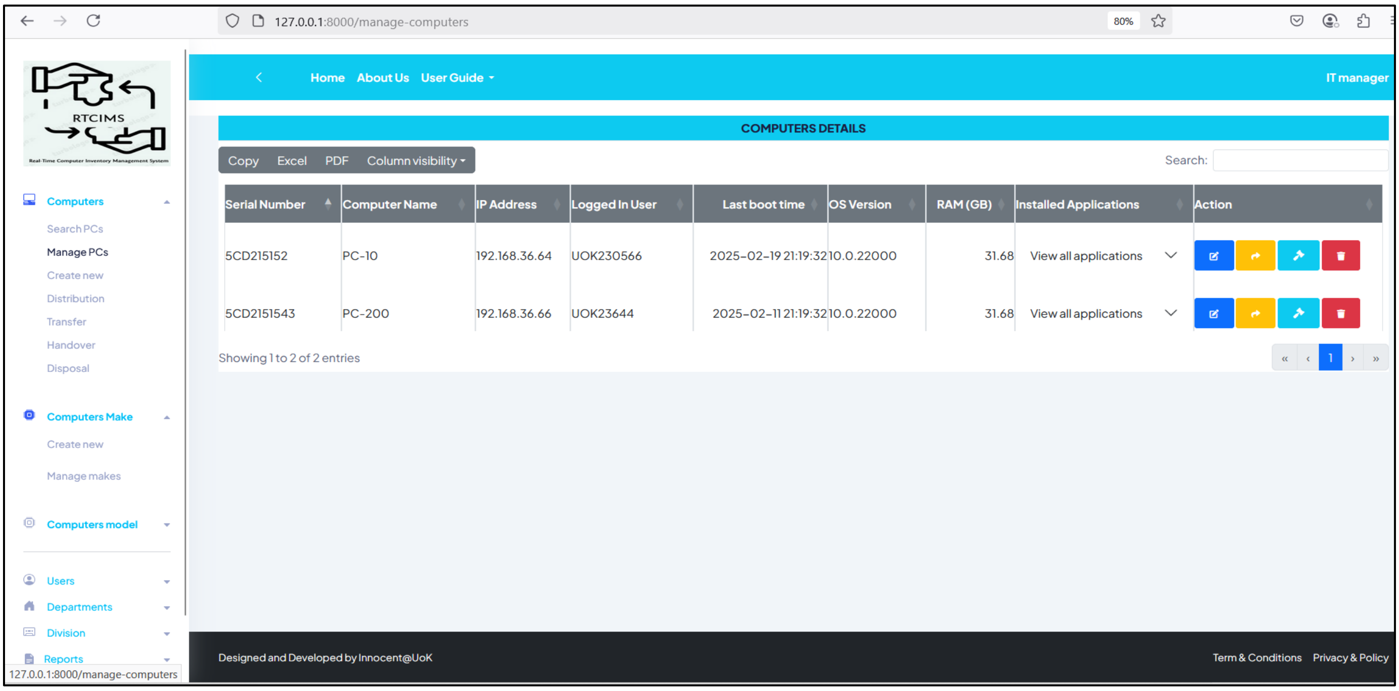
Computers in Service ↓0

Computers Disposed 0

Designed and Developed by Innocent@UoK

Fig 16 Dashboard Page

➢ *Computer Management Portal*



COMPUTERS DETAILS

Serial Number	Computer Name	IP Address	Logged In User	Last boot time	OS Version	RAM (GB)	Installed Applications	Action
5CD215152	PC-10	192.168.36.64	UOK230566	2025-02-19 21:19:32	10.0.22000	31.68	View all applications	
5CD2151543	PC-200	192.168.36.66	UOK23644	2025-02-11 21:19:32	10.0.22000	31.68	View all applications	

Showing 1 to 2 of 2 entries

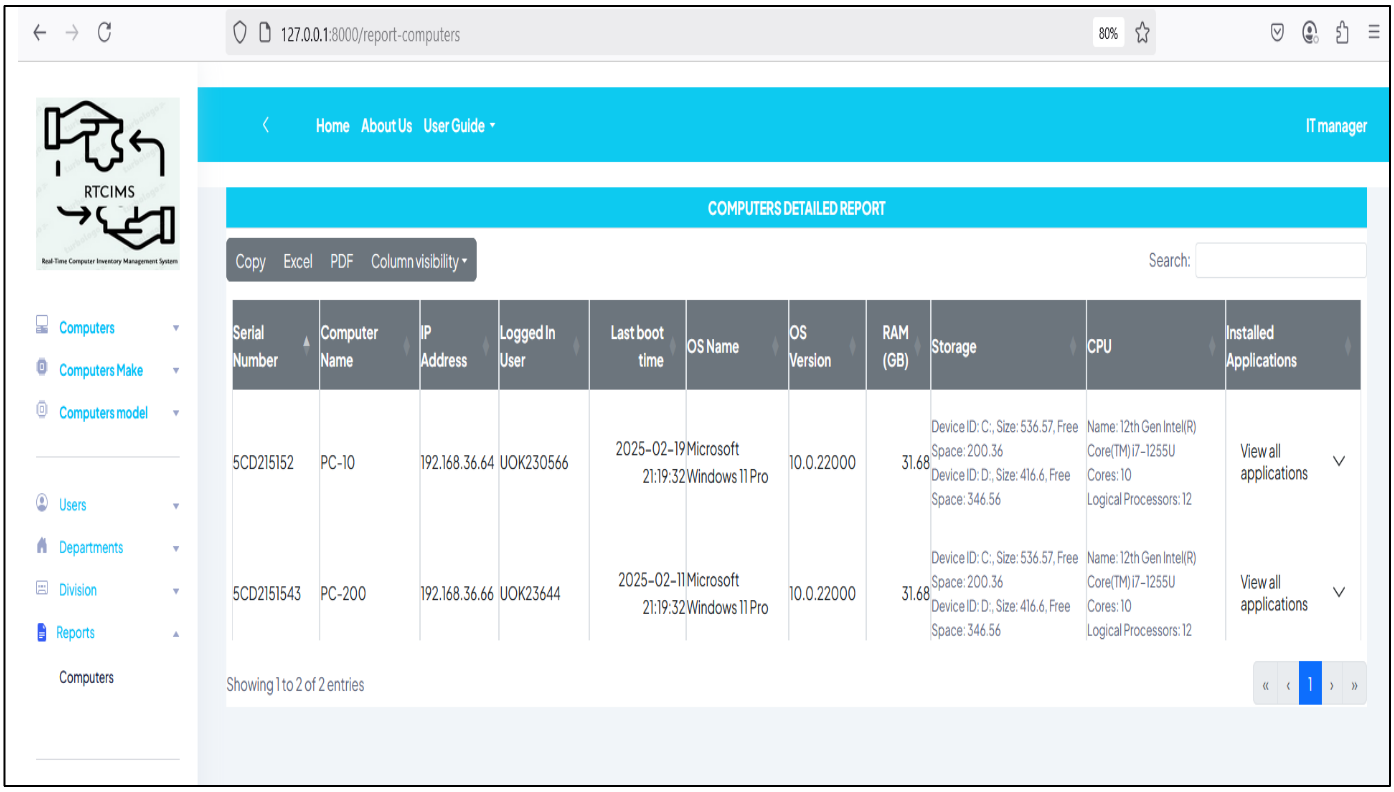
Designed and Developed by Innocent@UoK

Term & Conditions Privacy & Policy

Fig 17 Computer Management Page

➢ *Computer report portal*

On this portal, an IT manager can print a report in different format (Excel, CSV, PDF, …). He can also view all hardware and software details and take decision accordingly.



COMPUTERS DETAILED REPORT

Serial Number	Computer Name	IP Address	Logged In User	Last boot time	OS Name	OS Version	RAM (GB)	Storage	CPU	Installed Applications
5CD215152	PC-10	192.168.36.64	UOK230566	2025-02-19 21:19:32	Microsoft Windows 11 Pro	10.0.22000	31.68	Device ID: C:, Size: 536.57, Free Space: 200.36 Device ID: D:, Size: 416.6, Free Space: 346.56	Name: 12th Gen Intel(R) Core(TM) i7-1255U Cores: 10 Logical Processors: 12	View all applications
5CD2151543	PC-200	192.168.36.66	UOK23644	2025-02-11 21:19:32	Microsoft Windows 11 Pro	10.0.22000	31.68	Device ID: C:, Size: 536.57, Free Space: 200.36 Device ID: D:, Size: 416.6, Free Space: 346.56	Name: 12th Gen Intel(R) Core(TM) i7-1255U Cores: 10 Logical Processors: 12	View all applications

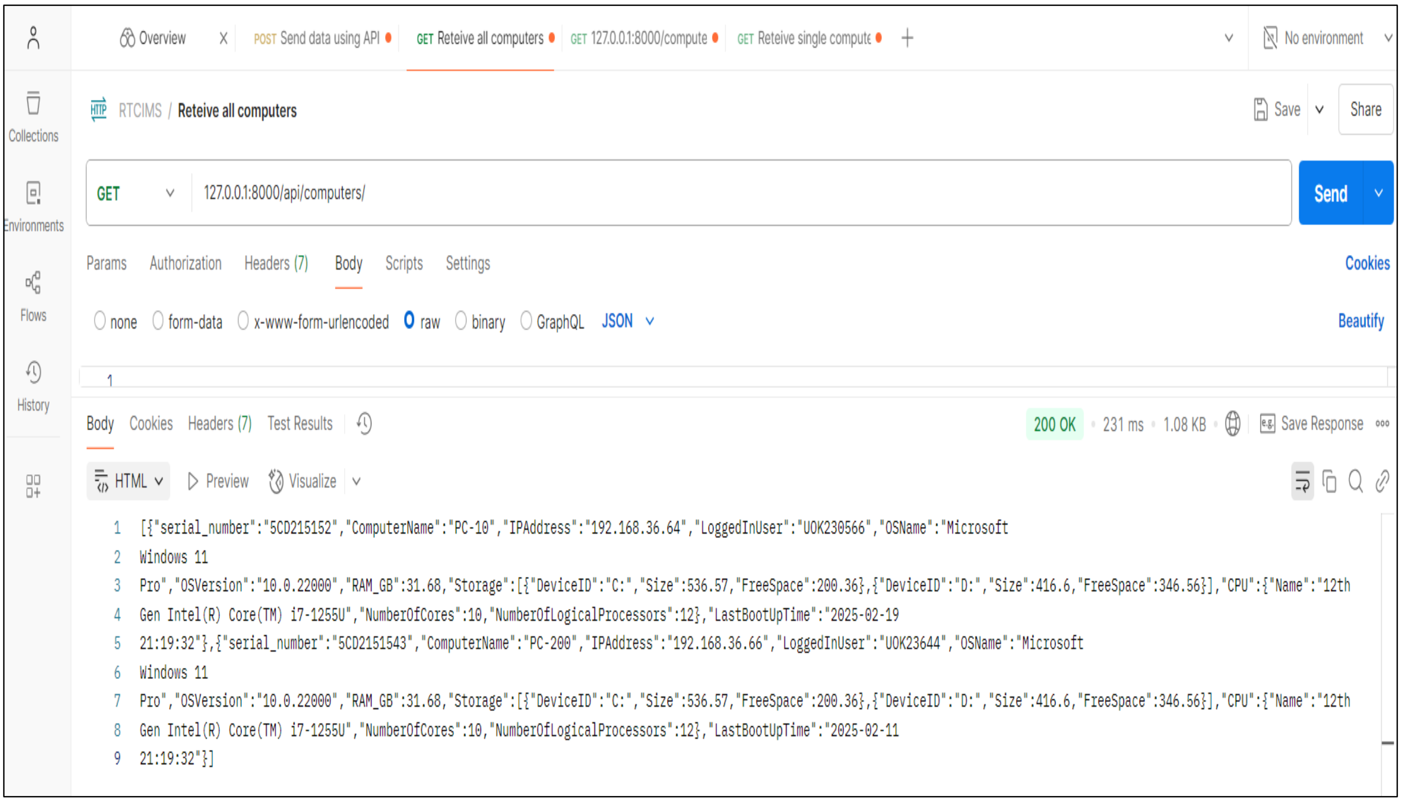
Showing 1 to 2 of 2 entries

Fig 18 Computer Report

➢ *Integration with Existing System*

As the purpose of this research is the design and implement the enhanced real-time computer inventory management system, we have developed API endpoint using Laravel so that existing systems can connect and retrieve computer real-time hardware and software information. The following are the screen shoots showing the API testing for retrieval of computer information using Postman as API testing tool.

- *Request all Computers*



GET 127.0.0.1:8000/api/computers/

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

200 OK · 231 ms · 1.08 KB

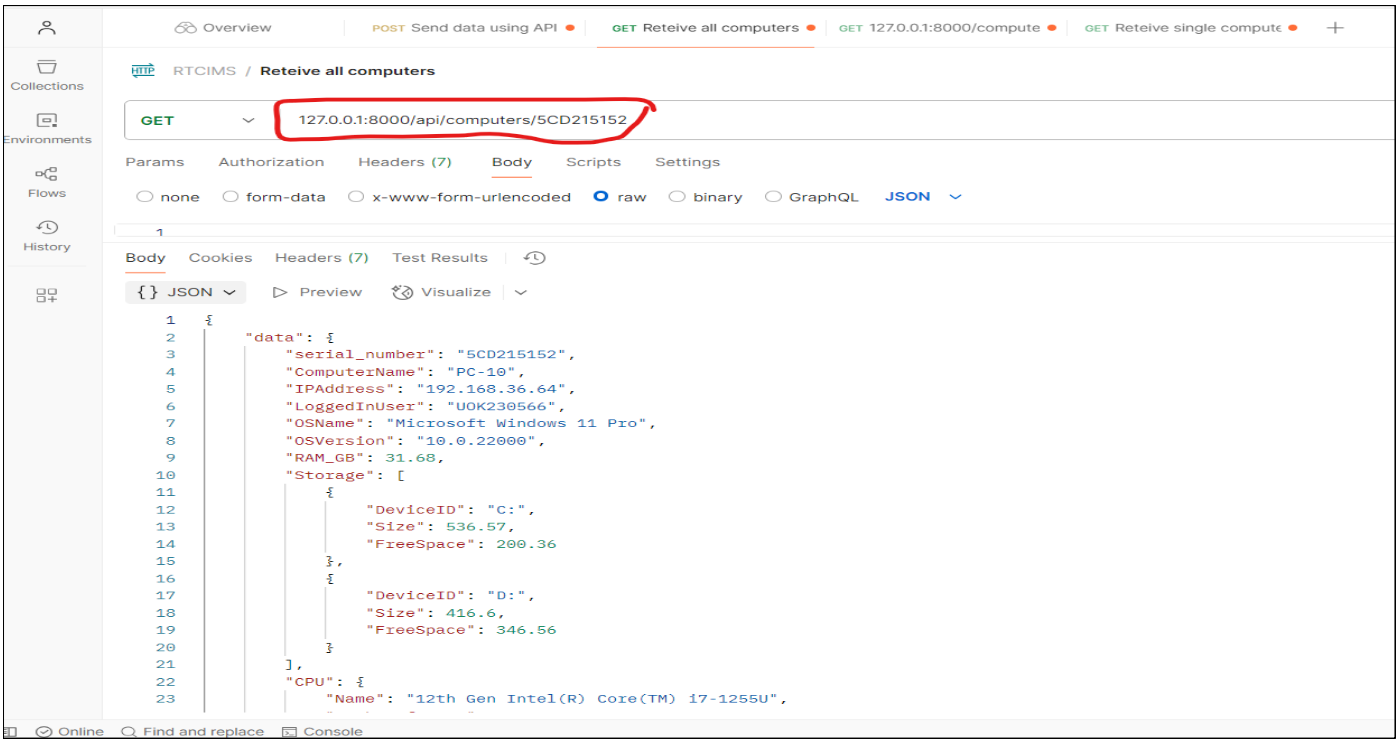
HTML

```
[{"serial_number":"5CD215152","ComputerName":"PC-10","IPAddress":"192.168.36.64","LoggedInUser":"UOK230566","OSName":"Microsoft
Windows 11
Pro","OSVersion":"10.0.22000","RAM_GB":31.68,"Storage":[{"DeviceID":"C:","Size":536.57,"FreeSpace":200.36},{"DeviceID":"D:","Size":416.6,"FreeSpace":346.56}],"CPU":{"Name":"12th
Gen Intel(R) Core(TM) i7-1255U","NumberOfCores":10,"NumberOfLogicalProcessors":12},"LastBootUpTime":"2025-02-19
21:19:32"},{"serial_number":"5CD2151543","ComputerName":"PC-200","IPAddress":"192.168.36.66","LoggedInUser":"UOK23644","OSName":"Microsoft
Windows 11
Pro","OSVersion":"10.0.22000","RAM_GB":31.68,"Storage":[{"DeviceID":"C:","Size":536.57,"FreeSpace":200.36},{"DeviceID":"D:","Size":416.6,"FreeSpace":346.56}],"CPU":{"Name":"12th
Gen Intel(R) Core(TM) i7-1255U","NumberOfCores":10,"NumberOfLogicalProcessors":12},"LastBootUpTime":"2025-02-11
21:19:32"}]
```

Fig 19 API Endpoint for Retrieving Many Records

- *Request for Single Record:*

A user from another system can request a single record/ computer. By using a serial number the API can return the associated information from this real-time system. This can be used in time of some personal audit of IT equipment.



GET 127.0.0.1:8000/api/computers/5CD215152

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

{} JSON

```
{
    "data": {
        "serial_number": "5CD215152",
        "ComputerName": "PC-10",
        "IPAddress": "192.168.36.64",
        "LoggedInUser": "UOK230566",
        "OSName": "Microsoft Windows 11 Pro",
        "OSVersion": "10.0.22000",
        "RAM_GB": 31.68,
        "Storage": [
            {
                "DeviceID": "C:",
                "Size": 536.57,
                "FreeSpace": 200.36
            },
            {
                "DeviceID": "D:",
                "Size": 416.6,
                "FreeSpace": 346.56
            }
        ],
        "CPU": {
            "Name": "12th Gen Intel(R) Core(TM) i7-1255U",
```

Fig 20 API Endpoint for Single Record

➢ *Technologies Used and System Testing*

The implementation of the Real-Time Computer Inventory Management System (RTCIMS) utilized a range of modern development tools and technologies. **Visual Studio Code** was selected as the primary code editor due to its versatility, debugging tools, Git integration, and support for various programming languages. **MySQL** served as the relational database for structured data storage, known for its reliability, scalability, and compatibility with Laravel.

For local development and testing, **XAMPP** was used as a complete package that bundles Apache, MySQL, and PHP, streamlining web server setup. The system was built on **Laravel**, a powerful PHP framework following the MVC architecture, offering robust features like routing, templating, authentication, and database migrations.

The user interface was crafted using **HTML5** and **CSS3**, which provide semantic structuring and responsive design capabilities. These were complemented by **Bootstrap**, a front-end toolkit offering ready-to-use components and mobile-first design. **JavaScript** was integrated to handle dynamic interactions, while **jQuery** simplified DOM manipulation, animations, and event handling.

To enable asynchronous data updates without refreshing pages, **AJAX** was employed, improving the real-time responsiveness of the system. For hardware and software data collection, **PowerShell** scripts were developed to automatically retrieve system details like CPU, memory, and installed programs, then transmit the information to the Laravel backend via APIs.

In terms of **system testing**, thorough evaluations were conducted to ensure functionality, performance, and security. Functional testing validated key features such as asset registration and data synchronization, while non-functional testing assessed performance under load and scalability. Security testing was implemented to protect asset data against unauthorized access. With AI-assisted anomaly detection and continuous testing practices from Agile and DevOps, the system was refined to maintain accuracy and efficiency in real-time asset tracking.

VI. CONCLUSION

The findings of the present study indicate that real-time automated IT asset management is a crucial element to enhance operational efficiency, security, and decision-making. The system implemented provides centralized, automated, and scalable IT asset management capabilities through PowerShell script-based automated data collection, Laravel-based API-functionality, and MySQL for database-based data structuring.

By offering seamless integration with existing systems through RESTful APIs and Webhooks, the system literally bridges the legacy-new IT inventory solution gap. In addition, real-time tracking minimizes errors by humans, maximizes asset utilization, and enhances regulatory compliance by maintaining updated asset records.

In short, the system is an automation of IT asset management, reduces manual effort, and ensures correct asset tracking. The proposed methodology effectively addresses scalability, automation, and system integration challenges in modern IT infrastructures.

RECOMMENDATION

I recommend big Institutions including Rwanda Revenue Authority which was the case study on this study to start using this proposed real-time computer inventory management system due to its automation, scalability, easily integrable with other systems and it produces reliable and accurate, real-time reports.

REFERENCES

[1]. Andy , D., Dipankar, D., Paul , D., & Brenna , S. (2023). *Maximizing IT Asset Performance with Real-Time Management Systems.* Deloitte.

[2]. 19770-1:2017, I. (2017). *Information technology — IT asset management . .*

[3]. Ali, R., & Ahmed, S. (2021). Integrating Modern Frameworks for IT Asset Management. *International Journal of IT Systems*, 45-59. Retrieved 08 03, 2024

[4]. Bank, A. D. (2022). *Digital Transformation Strategy for Africa (2020–2030).* Retrieved 10 20, 2024, from www.afdb.org

[5]. Barney, J. (1991). *Firm resources and sustained competitive advantage* (Vol. 17(1)). doi:https://doi.org/10.1177/014920639101700108

[6]. Braun, V., & Clerke, V. (2006). *Using thematic analysis in psychology.* doi:http://dx.doi.org/10.1191/1478088706qp063oa

[7]. Brinkmann, S. (2014). *Unstructured and Semi-Structured Interviewing.* doi:https://doi.org/10.1093/oxfordhb/9780199811755.013.030

[8]. Brown, A., & Green, T. (2019). Evolution of Asset Management Technology. 45-59.

[9]. Buttazo, G. C. (2011). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications.* Springer.

[10]. C.Laudon, K., & P.Laudon, J. (2020). *Management Information Systems: Managing the Digital Firm.* Pearson. Retrieved from https://repository.dinus.ac.id/docs/ajar/Kenneth_C.Laudon,Jane_P_.Laudon_-_Management_Information_Sysrem_13th_Edition_.pdf

[11]. Chen, R., Lee, S., & Kang, Y. (2022). Automation IT Processes with PowerShell. *Journal of Autmation in Systems*. Retrieved from https://doi.org/10.1016/j.jaits.2022.03.001

[12]. Codd, E. (1970). A relational model of data for large shared data banks. Communication of Acm.

[13]. Creswell, J., & Plano Clark, V. L. (2019). *Designing and conducting mixed methods research (3rd ed.).* SAGE Publications. doi:https://doi.org/10.1177/1937586719832223

[14]. Davis, M., & Gracia, R. (2023). Scalability in IT Asset Management System. *Jounal of Information Technology*, 35(2), 87-101.

[15]. Davis, R. (2021). Reducing Downtime with Proactive Asset Management. *Maintenance Techology Journal*, 18(3), 90-102.

[16]. Gartner. (2020). *IT Asset Management Key Initiative Overview*. Retrieved 2024, from Gartner: https://www.gartner.com/en/documents/2519015

[17]. Gartner. (2024). . Retrieved from https://www.gartner.com: https://www.gartner.com/en/information-technology/insights/cybersecurity

[18]. Guest, G., Bunce, A., & Johnson , L. (2006). How Many Interviews Are Enough . doi:https://doi.org/10.1177/1525822X05279903

[19]. Guest, G., Bunce, A., & Johnson, L. (2006). How Many Interviews Are Enough. *Sage Publications*, 59-82. doi:https://doi.org/10.1177/1525822X05279903

[20]. Hagaman, A., & Wutich, A. (2017). How Many Interviews Are Enough to Identify Metathemes in Multisited and Cross-cultural Research? Another Perspective on Guest, Bunce, and Johnson's (2006) Landmark Study. *Thousand Oaks, CA: Sage., 29*, 23-41. doi:https://doi.org/10.1177/1525822X16640447

[21]. Hussain, A., & Awan, S. A. (2021). Leveraging Automation for IT Asset Management. *Journal of Advanced IT Research*, 45-67.

[22]. IBM. (2024). *https://www.ibm.com/topics/data-loss-prevention*. Retrieved from https://www.ibm.com.

[23]. Joseph, L., & Jenkins, A. (2019). Automated IT Asset Management: Tools and Techniques. *Jounal of Information Technology Management*, 45-60.

[24]. Kallio, H., Pietila, A.-M., Martin, J., & Mari Kangasniemi, D. (2016). *Systematic methodological review: developing a framework for a qualitative semi-structured interview guide* (Vol. 72). Journal of Advanced Nursing. doi:https://doi.org/10.1111/jan.13031

[25]. Kim, H., & Lee, J. (2022). RFID-based Asset Management in Hospital. Healthcare Management Rewiew. 28(2), 132-135.

[26]. KPMG. (2023). *Navigating-the-digital-frontier-asset-intensive-organisations.pdf.* Retrieved from https://assets.kpmg.com/content/dam/kpmg/au/pdf/2023/navigating-the-digital-frontier-asset-intensive-organisations.pdf

[27]. Lee, C., & Park, S. (2020). RFID-based Real time asset Tracking in Healthcare. 26(3),233-245.

[28]. Malik, P., Zhang, Q., & Lee, J. (2023). Scalability and Security in Relational Databases for Real-Time System. Database Management Quarterly. 78-90.

[29]. Patel, R., & Nguyen, T. (2022). User Experience and Satisfaction in IT Asset Management Systems. *Journal of User Experience*.

[30]. Patricia, L. (. (2014). *The Oxford Handbook of Qualitative Research.* Oxford University Press. doi:https://doi.org/10.1093/oxfordhb/9780199811755.001.0001

[31]. Patton, & Michael , Q. (2002). *Qualitative research and evaluation methods.* Thousand Oaks, Calif. : Sage Publications.

[32]. Patton, M. Q. (n.d.).

[33]. RRA. (2024, 11 10). Retrieved from RRA Website: https://www.rra.gov.rw/en/about-us/rra-profile#:~:text=Mission%20and%20Vision%20statement&text=%E2%80%9CTo%20be%20a%20model%20revenue%20administration%20optimally%20financing%20national%20needs.%E2%80%9D&text=%E2%80%9CTo%20efficiently%20mobilize%20revenue

[34]. RRA. (2024). *Mission and Vision statement*. Retrieved from www.rra.gov.rw: https://www.rra.gov.rw/en/details?tx_news_pi1%5Baction%5D=detail&tx_news_pi1%5Bcontroller%5D=News&tx_news_pi1%5Bnews%5D=1062&cHash=b69f99b9f1809e20a67532ee4a5ed2f8

[35]. Shavah, V., & Rao, N. (2020). Modern Approaches in Inventory Systems Using Real-Time Data. *IT Systems Journal*, 29-42.

[36]. Smith, P., & Jones, M. (2021). Automating IT Asset Discovery for Improved IT Governance. *International Journal of IT Management*, 8(2), 34-50.

[37]. Wang, X., Patel, N., & Singh, S. (2022). Addressing Scalability in Real-Time Inventory Systems. *Journal of IT Scalability*, 123-137.