DSR of sdlc lab

Table of

Aim and Objective

The project aims to create a DSR (Design Specification Report) of the SDLC (Software Development Life Cycle) Lab. The objective is to access data on different electronic components in the lab.

Technical Details

The project involves collecting data, understanding the components, creating an algorithm, designing a flowchart, storing data in an Excel file, converting it to a CSV file, and developing a program to extract the data.

Actions Taken

The actions taken include data collection, understanding the components, creating an algorithm, designing a flowchart, storing data in an Excel file, converting it to a CSV file, and developing a program to extract the data.

Data Taken

The data taken includes the details of different electronic components in the SDLC Lab, which have been stored in an Excel file and converted to a CSV file for further processing.

AIM and OBJECTIVES

1. **Project targets to create a DSR of SDLC Lab**
2. To access data of different electronics components of the Lab

Actions Taken

1 Data Collection

The first step was to collect the relevant data from the SDLC lab. This involved gathering information about the different electronic components available in the lab.

2 Understanding

After collecting the data, the team worked on understanding the information and identifying the key details that would be needed for the project.

3 Algorithm Development

With the data and understanding in place, the team then developed an algorithm to process the information and generate the desired output.

4 Flowchart Creation

To visualize the algorithm, the team created a flowchart to map out the various steps and processes involved.

5 Data Storage

The collected data was then stored in an Excel file for further processing and analysis.

6 Data Conversion

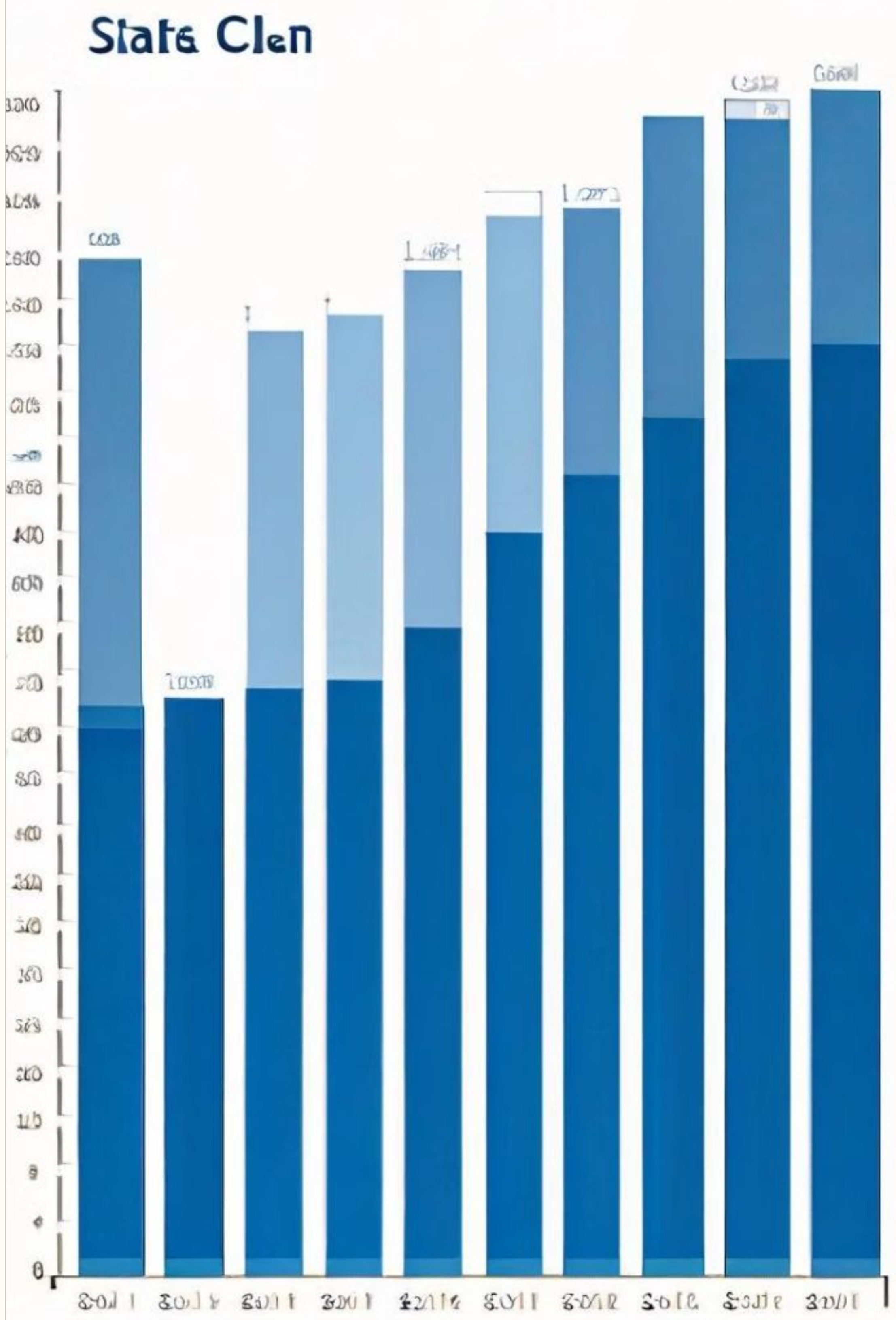
To make the data more accessible and usable, the Excel file was then converted into a CSV format.

7 Program and Output

Finally, the team developed a program to extract and display the relevant information from the CSV file, generating the desired output for the project.

DATA taken

We have taken data from the SDLC lab, stored it in an Excel file, and then converted it to a CSV file. This data has been extracted and used in the program.



Access Key

1. **EED_SDLCuser** for USER
2. **EED_SDLCadmin** for ADMIN
3. Different access keys used for accessing different components

BASIC WORKING

Login

The basic working of the system starts with the user logging in. The user can log in as either an admin or a regular user.

User

The regular user can read the component data stored in the system. They have access to view the information about the different components in the lab.

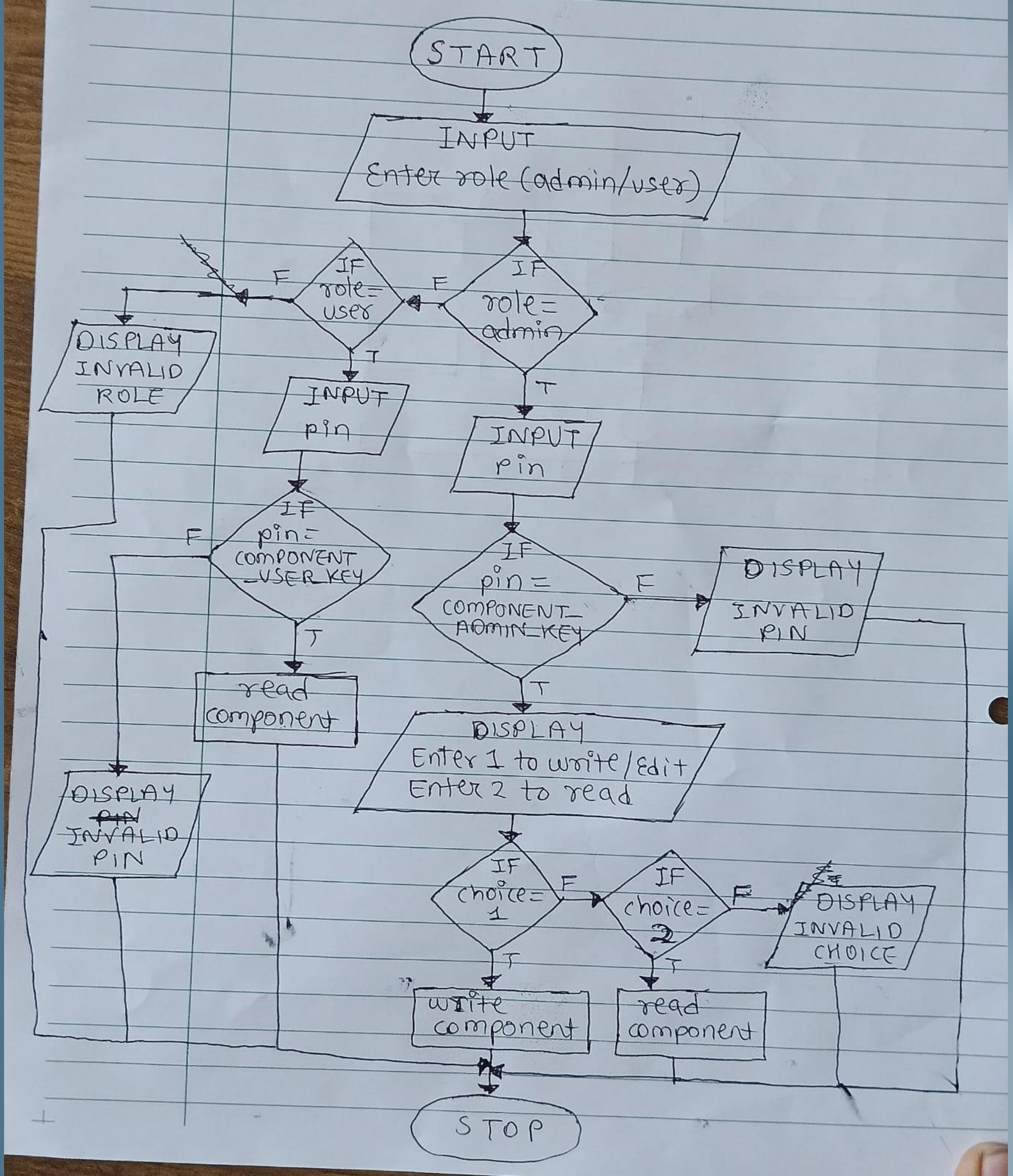
Admin

The admin user has additional privileges beyond the regular user. They can not only read the component data, but also edit and update the information as needed.

Read

Both the user and admin roles can read the component data stored in the system. They can view the details of the different components in the lab.

Flowchart


START
INPUT
Enter role (admin/user)
IF role= admin
F
T
IF role= user
F
T
DISPLAY INVALID ROLE
INPUT pin
INPUT pin
IF pin= COMPONENT_USER_KEY
F
T
IF pin= COMPONENT_ADMIN_KEY
F
T
DISPLAY INVALID PIN
read component
DISPLAY Enter 1 to write/Edit Enter 2 to read
DISPLAY ~~PIN~~ INVALID PIN
IF choice= 1
F
T
IF choice= 2
F
T
DISPLAY INVALID CHOICE
write component
read component
STOP

Code Structure

Component Class: The Component class represents a component and contains member variables for the component's name, value, quantity, and key. It also includes a **display_Component()** function to display the component's information. **Global Variables:** The code includes two global variables **COMPONENT_ADMIN_KEY** and **COMPONENT_USER_KEY** that store the access keys for the admin and user roles, respectively.

write_component() Function: This function allows the admin role to write new component data to the CSV file. It prompts the user to enter the number of components to add and then prompts for the name, value, and quantity of each component. The component data is then written to the CSV file.

Code Structure

The code structure includes several key functions that represent the admin and user roles, as well as the main entry point of the program. The **ComponentAdmin()** function handles the admin role, prompting the user to enter a PIN and checking if it matches the admin access key. If the PIN is correct, the user is then prompted to choose between writing/editing or reading component data. Depending on the choice, either the **write_component()** or **read_components()** function is called.
The **ComponentUser()** function represents the user role. It prompts the user to enter a PIN and checks if it matches the user access key. If the PIN is correct, the **read_components()** function is called to display component data.

The **main()** function is the entry point of the program. It prompts the user to enter a role (admin or user) and calls the corresponding role function based on the input.

Overall, the code structure is designed to handle both admin and user functionality, with clear separation of concerns and appropriate access control mechanisms in place.