جامعة برج العرب التكنولوجية

BORG AL ARAB TECHNOLOGICAL UNIVERSITY

Faculty of Industrial and Energy Technology

Information Technology Program

Smart Printing System (SPS)

Graduation Project Spring Semester Academic Year 2024/2025

By AITP Team:

Zeyad Rabea Abd El-Hamid	Maryam Eid Abdelsalam	Mariam Ibrahim Mohamed
Hager Ragab El-Said Madian	Malk Saif Alden Attia	Mariam Mohamed Mubarak
Seif Eldeen Ehab Mohamed	Mariam Medhat Omar Gewely	Hager Taha Hassan Mohamed
Nada Mohammed Ragab	Wafaa Abdul Raouf Mohammed	Osama El-Sayed Ali
Manar Ashraf Mohamed	Mohammed Hamada Mohaseb	Abdulrahman Ahmed Attia
Mohmed Hossam Abdelfatah	Mohamed Abdelhamed El- Sayed	Manar Mahmoud El-Sayed
Abdelrahman Mohamed Said	Huda Hamdy Ibrahim	Karim Ahmed Mohamed
Abd El-Hamid Mohammed	Rahma Saleh Ramadan	Shrouk Hesham Mohamed
Abdulrahman Khamis Abdo	Rahma El-Shahat Nour	Aya Mokhtar Eid
Mahmoud Arafa Abdel Halim	Basma Abd al Rasoul	Abdulrahman Muhammad Salim
Abdulrahman Awad Daif Allah	Abdulrahman Muhammad Yusuf	Shahd Muhammad Rajab Turki
Shihab al-Din Muhammad	Hanin Mohammed Murad Mohammed	

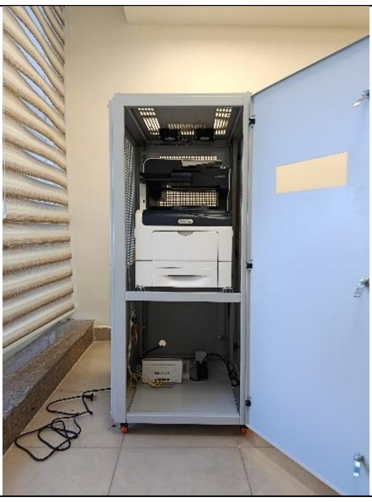
Under Supervision of:

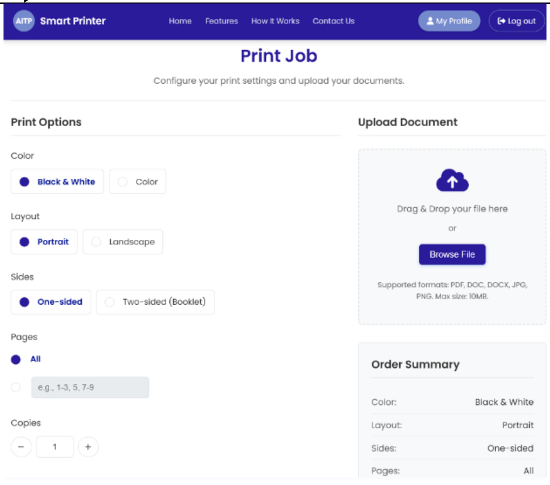
[Assoc. Prof. Osama El-Nahas - Eng.Rania Helal]

كلية تكنولوجيا الصناعة والطاقة
FACULTY OF INDUSTRIAL AND ENERGY TECHNOLOGY

جامعة برج العرب التكنولوجية
BORG AL ARAB TECHNOLOGICAL UNIVERSITY

Smart Printing System (SPS)




AITP Smart Printer
Home · Features · How It Works · Contact Us · My Profile · Log out

Print Job

Configure your print settings and upload your documents.

Print Options

Color: [✓ Black & White] Color

Layout: [✓ Portrait] Landscape

Sides: [✓ One-sided] Two-sided (Booklet)

Pages: [✓ All] e.g., 1-3, 5, 7-9

Copies: 1

Upload Document

Drag & Drop your file here

or

Browse File

Supported formats: PDF, DOC, DOCX, JPG, PNG. Max size: 10MB.

Order Summary

Color:	Black & White
Layout:	Portrait
Sides:	One-sided
Pages:	All

Graduation Project Spring Semester 2024/2025

Submitted by AITP team members

Serial	Student Name	Student ID
1	Zeyad Rabea Abd El-Hamid	2320241
2	Maryam Eid Abdelsalam	2320603
3	Mariam Ibrahim Mohamed	2320598
4	Hager Ragab El-Said Madian	2320707
5	Malk Saif Alden Attia	2320630
6	Mariam Mohamed Mubarak	2320604
7	Seif Eldeen Ehab Mohamed	2320269
8	Mariam Medhat Omar Gewely	2320605
9	Hager Taha Hassan Mohamed	2320712
10	Nada Mohammed Ragab	2320677
11	Wafaa Abdul Raouf Mohammed	2320728
12	Osama El-Sayed Ali	2320091
13	Manar Ashraf Mohamed	2320633
14	Mohammed Hamada Mohaseb	2320499

15	Abdulrahman Ahmed Attia	2023312
16	Mohmed Hossam Abdelfatah	2320498
17	Mohamed Abdelhamed El- Sayed	2320523
18	Manar Mahmoud El-Sayed	2320636
19	Abdelrahman Mohamed Said	2320337
20	Huda Hamdy Ibrahim	2320721
21	Karim Ahmed Mohamed	2320450
22	Abd El-Hamid Mohammed Abd El-Hamid	2320310
23	Rahma Saleh Ramadan	2320222
24	Shrouk Hesham Mohamed	2320280
25	Abdulrahman Khamis Abdo	2320326
26	Rahma El-Shahat Nour	2320220
27	Aya Mokhtar Eid	2320155
28	Mahmoud Arafa Abdel Halim	2320583
29	Basma Abd al Rasoul	2320160
30	Abdulrahman Muhammad Salim Khalil	2320338
31	Abdulrahman Awad Daif Allah Awad	2320334
32	Abdulrahman Muhammad Yusuf Abdul Mawjoud	2320344
33	Shahd Muhammad Rajab Turki	2320296
34	Shihab al-Din Muhammad Ahmad Hashim	2320286
35	Hanin Mohammed Murad Mohammed	2320210

Under Supervision of:

[Assoc. Prof. Osama El-Nahas – Eng.Rania Helal]

ABSTRACT

Traditional printing services in academic institutions often face significant challenges such as long queues, manual intervention, inefficient payment methods, and the absence of centralized management.

The Smart Printing System (SPS) is an innovative, self-service solution designed to address these problems by delivering a fully automated and digitally managed printing environment tailored for university campuses and similar settings.

SPS allows users to securely upload their documents through an intuitive web-based interface, customize printing options such as paper size, color mode, and number of copies, and complete payment using a coin-based or internal wallet system.

Users can top up their virtual balance via electronic transfers, which can then be used for print transactions.

At the core of the system is a Raspberry Pi device that acts as the main controller, interfacing between the website and the printer using the CUPS (Common UNIX Printing System). The system architecture also includes a MySQL database for managing user data, file queues, and transaction logs, along with a RESTful API and real-time communication via WebSocket for live status updates and notifications.

Key advantages of SPS include reducing student wait times, eliminating the need for dedicated printing staff, enabling 24/7 availability, and supporting the university's digital transformation. The system’s modular and scalable design allows for future enhancements, such as mobile app integration, NFC authentication, or advanced analytics.

Ultimately, SPS provides a modern, secure, and user-friendly alternative to traditional printing services, making it a valuable asset to smart campus ecosystems and encouraging independent, efficient, and environmentally conscious printing behavior among students.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our supervisor, Prof. Osama Elnahas, for his invaluable guidance, continuous support, and insightful feedback throughout every phase of this project. His expertise has significantly contributed to shaping our vision and achieving our objectives.

We are also deeply thankful to Eng. Rania Helal, our academic advisor, for her encouragement, dedication, and technical support, which greatly enriched the development process.

Our heartfelt thanks go to the faculty and staff of the Faculty of Industrial and Energy Technology for equipping us with the knowledge, resources, and learning environment necessary to complete this project successfully.

We would also like to extend special appreciation to our families and friends for their unwavering support and motivation during our journey.

Last but not least, we would like to thank all team members for their dedication, collaboration, and team spirit. This project would not have been possible without each one’s contribution and shared commitment.

LIST OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Full Term
API	Application Programming Interface
CSS	Cascading Style Sheets
DB	Database
DBMS	Database Management System
ESP	Espressif (Microcontroller Manufacturer)
ESP8266	Wi-Fi Module based on Espressif Chip
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
JS	JavaScript
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LCD	Liquid Crystal Display
LAN	Local Area Network
MAC	Media Access Control (Address)
PCB	Printed Circuit Board
PHP	Hypertext Preprocessor
PSU	Power Supply Unit
QR	Quick Response (Code)
RPI	Raspberry Pi
RAM	Random Access Memory
RJ45	Registered Jack 45 (Ethernet Connector)
ROM	Read-Only Memory
SPS	Smart Printing System
SQL	Structured Query Language
SSID	Service Set Identifier (Wi-Fi Name)
TCP/IP	Transmission Control Protocol / Internet Protocol
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
UX	User Experience
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

Chapter 1
Case Study/ Problem Statement

In many academic institutions, students rely on shared printing services for their daily academic needs. However, traditional printing setups present several inefficiencies. Long queues, manual handling by staff, lack of real-time monitoring, and limited service hours are among the common issues faced by students and administrators alike.

Additionally, these systems often depend on outdated payment methods such as cash-only transactions or require the presence of supervisors to operate the printers. This leads to delays, increased operational costs, and inconvenience for users.

Furthermore, there is no centralized system that tracks user transactions, balances, and print history, which limits administrative oversight and data analytics capabilities.

In light of these challenges, there is a clear need for an intelligent, self-service printing system that streamlines the process, reduces dependency on human intervention, supports modern payment methods, and enhances accessibility. The Smart Printing System (SPS) addresses these problems by offering a fully automated, secure, and user-friendly solution tailored for educational environments**.**

Chapter 2
System Objectives and Scope

The Smart Printing System (SPS) is developed to provide a modern, self-service, and user-friendly printing solution tailored for academic institutions. It addresses common issues such as long queues, manual handling, and inefficient payment processes.

Objectives:

- Eliminate waiting lines and enable students to print documents independently.
- Introduce a virtual coin-based wallet system to facilitate quick, cashless transactions.
- Automatically deduct printing costs from the user's balance upon job completion.
- Provide a dedicated admin dashboard for system supervision and monitoring.
- Allow users to upload PDF files, choose print settings (color, copies, etc.), and track job status in real-time.
- Improve operational efficiency and reduce human intervention in printing services.
- Encourage environmentally conscious printing through a controlled and accountable system.

The SPS covers:

- A secure web interface for students to log in, upload files, and configure print jobs.
- A Raspberry Pi device running CUPS to manage communication with the physical printer.
- Integration with a MySQL database to manage users, transactions, print jobs, and balance logs.
- A virtual wallet system using internal coins that users can recharge manually via admin.
- Real-time communication using RESTful APIs and WebSocket for updates and notifications.
- An administrative panel that allows the supervisor to monitor activities, manage balances, and view logs.

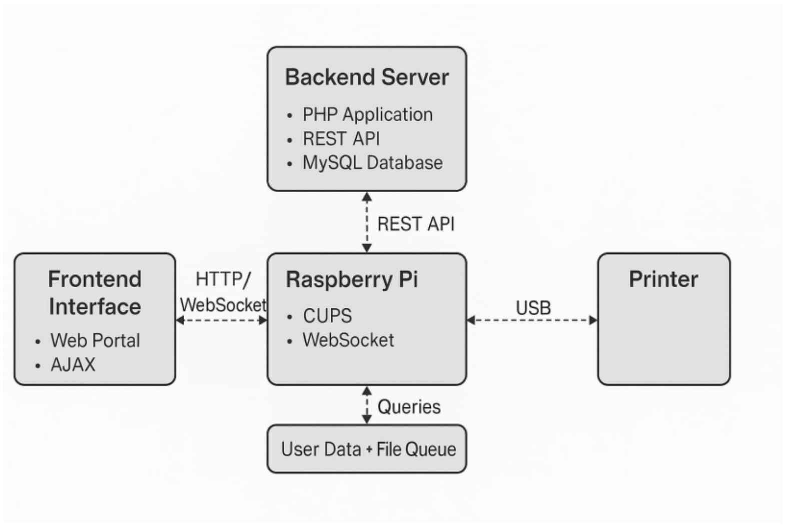
Out of Scope:

- Online or automated electronic recharge of the virtual wallet.
- Mobile application or NFC-based authentication.
- Advanced analytics or printing behavior reports.

System Architecture Overview

The Smart Printing System (SPS) architecture is modular and distributed across several key components:

- Front-End Web Interface: Allows users to register, log in, upload documents, configure print settings, and view balance and status.
- Back-End Server (PHP): Handles authentication, session management, file processing, cost calculation, and virtual wallet transactions.
- MySQL Database: Stores user credentials, uploaded files, printer job queue, transaction history, and admin settings.
- Raspberry Pi (RPI): Acts as a hardware bridge between the server and the printer. It fetches print jobs and sends them to the printer via CUPS.
- Printer (Xerox VersaLink C405): Receives and executes jobs through the CUPS protocol.



Visual representation of the SPS architecture, illustrating how system components interact through REST API, WebSocket, and USB communication channels.

Chapter 3
Methodology

3.0. Development Process Overview

The development of the Smart Printer web application followed a structured, iterative approach that included several key phases:

1. Requirement Analysis: We identified the needs of both regular users and system administrators, including key functionalities such as print job submission, virtual credit handling, and print monitoring.

2. System Design: Wireframes and user flow diagrams were created to visualize the user interface and backend logic. The database schema and entity relationships were carefully planned to support all operations.

3. Front-End Development: A responsive and user-friendly interface was built using modern technologies, with a focus on accessibility and clean design.

4. Back-End Development: Core logic was implemented using PHP and MySQL, including session management, file handling, cost calculation, and notification systems.

5. Integration: Front-end and back-end were connected using JavaScript and AJAX to ensure dynamic user interaction and real-time feedback.

6. Testing and Validation: Comprehensive testing was carried out to validate system functionality, including login processes, file uploads, cost deductions, and notification delivery.

7. Hardware Integration: Once the web application was fully tested and functional, it was integrated with a physical printer via a Raspberry Pi (RPI) to enable real-time communication and automatic job execution.

8. Physical Enclosure Design: After confirming the success of the full system—both software and hardware—a custom external casing was designed and fabricated to house the printer. This enclosure provides protection and visual appeal, making the device suitable for demonstration and deployment.

9. Deployment and Documentation: The final system was documented and prepared for presentation as a complete smart printing solution.

3.1. Front-end Development

For visual clarification of the components and steps described in this section, please refer to the corresponding figures in the appendices.

The front-end of the application was designed to be responsive, accessible, and visually clean across various screen sizes and devices. Emphasis was placed on user experience (UX) and interface consistency.

Technologies and Tools Used:

- HTML5, CSS3, and JavaScript (ES6) for structure, styling, and interactivity.
- Bootstrap 5 for responsive layout and reusable components.
- Font Awesome for visual icons.
- Google Fonts for enhanced typography.
- AJAX for asynchronous communication with the back-end.
- Responsive Web Design (RWD) techniques for cross-device compatibility.

Key Front-End Features:

- **Home Page:** Explains system usage and highlights project features.
- **Login Page**: Secure login with validation for both users and administrators.
- **User Dashboard:** Displays user profile, coin balance, upload history, and notifications.
- **Print Options Page:** Allows document upload and selection of print settings (e.g., color, sides, orientation, number of pages and copies).
- **Admin Dashboard:** Enables administrators to manage users, update print prices, view system reports, manage notifications, and monitor printer status.

The design ensures simplicity and professionalism, incorporating best UI/UX practices such as consistent layout, clear call-to-action buttons, and smooth user navigation.

3.2. Back-end Development

The back-end was developed using PHP to handle all logic, data validation, file management, and interactions with the database.

Technologies and Tools Used:

- PHP 8.x for server-side scripting.
- MySQL for structured data storage and relational database management.
- phpMyAdmin for database administration.
- Composer as a dependency manager.
- Smalot/pdfparser PHP library for counting PDF pages automatically.
- JavaScript (ES6) and AJAX for dynamic client-server interaction.

Core Functionalities:

- **Authentication System:** Handles secure login sessions for users and admins.
- **File Handling:** Users can upload PDF documents with specified print settings.
- **Cost Calculation:** The system automatically calculates the cost based on user options (e.g., number of pages, copies, color).
- **Page Count Extraction:** The smalot/pdfparser library analyzes uploaded PDF files to determine the total number of pages.
- **Virtual Coin Management**: Deducts the correct balance from users after confirming the print job.
- **Admin Management:** Administrators can adjust print prices, credit users, send notifications, and track daily usage.
- **Printer Notifications:** The system logs incoming status messages from the printer (e.g., low paper or error alerts).

Database Design

*The database **diagram** is provided in the appendix for reference.*

A MySQL database was created to support and organize all operations, ensuring data consistency, performance, and scalability. The following main tables were implemented:

- **users:** Stores user account details and balances.
- **printer_jobs:** Tracks each print job with details such as file name, print settings, timestamps, and job status.
- **transaction:** Logs all credit additions, deductions, and transaction histories.
- **notification:** Stores system notifications sent to users.
- **printer_status:** Monitors real-time printer condition updates.
- **printer_notification:** Logs alerts and messages from the physical printer device.
- **price_settings:** Contains price configurations based on print options.
- **daily_reports:** Automatically generated summaries of usage and transactions per day.
- **admin_logs:** Records administrative actions for auditing purposes.

Integration and Communication

AJAX and JavaScript were used to connect the front-end and back-end effectively. This allows real-time operations such as:

- Live updates of cost based on selected print options.
- Immediate feedback on file upload and print status.
- Interactive dashboards without needing full-page reloads.

This seamless integration ensures a smooth and fast user experience and enhances overall system responsiveness.

Security Considerations

To ensure safe and secure operations, the system implements several layers of protection:

Session Management: All user and admin sessions are secured with PHP session tokens and timeouts to prevent hijacking.

File Upload Validation: Only PDF files are accepted. The system checks MIME types and limits file sizes to prevent malicious uploads.

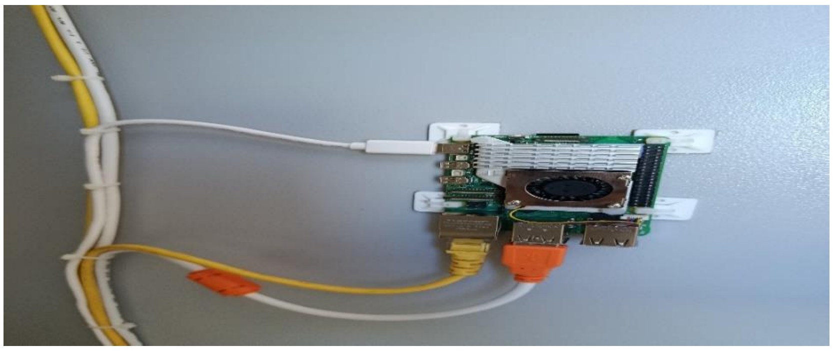
Database Access Control: Database users have limited privileges; the Raspberry Pi uses a restricted read/write user.

Input Sanitization: All user inputs are validated and sanitized using PHP built-in functions to prevent SQL injection and XSS attacks.

Admin Access Restriction: Admin panel is protected with an additional login layer, and all actions are logged for audit purposes.

Future improvements can include HTTPS encryption and two-factor authentication (2FA) for added security.

3.3. Raspberry Pi Integration



Introduction

The Raspberry Pi 5 functions as the primary hardware controller within the Smart Printing System. It facilitates seamless communication between the website backend and the physical printer (Xerox VersaLink C405) using CUPS, while executing Python scripts to automate print tasks. This section provides a comprehensive explanation of the Raspberry Pi's setup and configuration.

System Setup

- Operating System
  Installed Raspberry Pi OS on a 32GB SD card using Raspberry Pi Imager.
  Connected the device to the local Wi-Fi network.

- Enabling Remote Access
  Enabled SSH via /boot directory and VNC via Pi Configuration.
  Accessed remotely using PuTTY and VNC Viewer.

- Software Installation
  Installed CUPS with sudo apt install cups
  Created Python venv for project scripts

Printer Configuration

- Adding the Printer
  Accessed CUPS at http://localhost:631
  Added Xerox VersaLink C405 printer
  Verified printer drivers and test print
- Printing Capabilities
  Checked supported duplex options using:
  ```
  lpoptions -p Printer_Name -l
  ```
  Restarted CUPS to apply changes

Communication Methods Used in the System

At the initial stages of system development, the WebSocket protocol was adopted to enable real-time communication between the Raspberry Pi and the backend server. However, due to technical and financial challenges, it was later replaced with a more efficient and scalable solution—Firebase Firestore. The following sections outline the evolution of the communication methods used.

RESTful API

RESTful API was first implemented to facilitate communication between the Raspberry Pi and the backend. The Raspberry Pi periodically polled the server to check for new print jobs. While this method was simple, it introduced response delays and increased resource consumption due to repetitive requests. As a result, it was deemed inadequate for real-time operations and was eventually phased out.

WebSocket Protocol

To overcome the limitations of RESTful API, the WebSocket protocol was introduced. It enabled bi-directional, real-time data exchange between the server and Raspberry Pi. A WebSocket server was set up using PHP or Node.js, while the Raspberry Pi used a Python WebSocket client to establish the connection. When a new print job was added, the server would immediately notify the Raspberry Pi to execute the task.

Although this approach significantly reduced latency and improved communication efficiency, it presented challenges such as complex setup requirements, the need for a dedicated server, and increased costs—especially with scalability and SSL configuration. Therefore, the system transitioned to a more flexible cloud-based solution.

Firebase Firestore (Current Method)

Firebase Firestore, a real-time cloud database by Google, was adopted as the current communication method. A Firestore collection named 'orders' was created to store print job information, including file URLs, copy counts, and print settings.

The Raspberry Pi leverages the on_snapshot function to listen for real-time updates. Once a new print job is detected, the device retrieves and processes the task instantly. This solution integrates seamlessly using the firebase_admin library and does not require a dedicated backend server. Authentication and security are handled through a JSON credential file.

Thanks to its flexibility, cost-effectiveness, and scalability, Firebase Firestore is considered the final and optimal choice.

Real-Time Connectivity

The primary goal of implementing real-time connectivity was to ensure immediate execution of print commands. Initially, this was achieved using WebSocket, and later enhanced through Firebase Firestore. The Raspberry Pi now responds to changes in the Firestore database in real-time and executes printing operations promptly.

This transition contributed to better system performance, minimized delays, and reduced computational load.

Raspberry Pi Setup and Script Execution

System Configuration

The Raspberry Pi OS was installed on an SD card. Wi-Fi was configured with a static IP address. The CUPS print server was installed using the command:

```
sudo apt install cups
```

The Xerox VersaLink C405 printer was added through the CUPS interface at http://localhost:631. Required Python libraries including firebase_admin and requests were also installed.

Firebase Configuration

A Firebase service account credential file (JSON format) was generated and stored in: /home/pi/smart_printer/fireprint/test.json

The Firestore database was initialized within the script using the firebase_admin library, allowing secure access and interaction.

Script Operation

The smart_printer.py script was developed to handle the following operations:

- Logging: Error logs are saved to /tmp/errors.log.
- Firestore Listening: The script listens to the 'orders' collection using on_snapshot.
- File Download: Files are downloaded using the requests library.
- Print Execution: The lp command is used to print with custom settings (copies, colors, etc.).
- Error Handling: The script retries failed operations up to three times with a 5-second interval between attempts.

Print Job Execution

Test print jobs were manually inserted into the Firestore database to verify script functionality. Logs were monitored using the command:

```
tail -f /tmp/errors.log
```

Common issues such as duplex printing settings or access permissions were identified and resolved to improve stability.

Suggested Alternatives

- Programming Language: PHP can be replaced with Node.js or Django based on system requirements.
- Database: MySQL can be substituted with MongoDB for non-relational data management.
- Communication Method: WebSocket with Node.js remains a viable option but requires additional infrastructure and cost.

Technical Challenges

- Ensuring seamless integration between Firebase, CUPS, and Raspberry Pi.
- Managing real-time execution of print jobs with minimal delay.
- Controlling potential service costs related to Paymob and cloud subscriptions.

Script Architecture Overview

The smart_printer.py script consists of the following components:

- Imports: firebase_admin, requests, subprocess
- Logging Setup: Logs errors to /tmp/errors.log
- Firebase Initialization: Authenticates and connects to Firestore
- Retry Mechanism: Attempts operations up to 3 times with delay
- Functions:
  • wakeup_printer – Activates the printer
  • print_pdf – Downloads and prints the document using lp
  • on_snapshot – Listens to Firestore changes and triggers execution
- Main Loop: Keeps the script running continuously

Print Job Error Handling & Retry Logic

To ensure reliability, the Raspberry Pi script is equipped with basic error-handling features:

If a print job fails (e.g., due to printer offline or file not found), the job status is updated as "error" in the database.

A retry mechanism attempts to resend the job after 5 seconds, up to 3 retries per file.

Failed jobs are logged with timestamp and reason in a dedicated log file (/home/pi/logs/smart_printer.log).

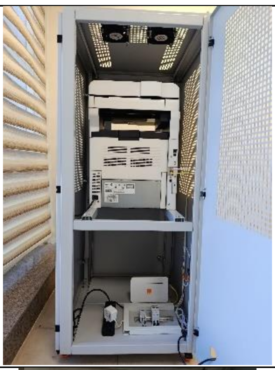
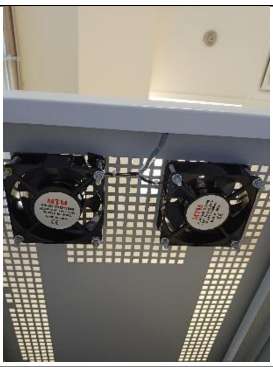
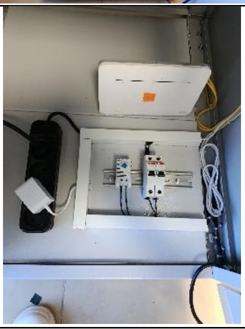
Administrators can review failed jobs and requeue them manually from the dashboard if necessary.

Conclusion

The Raspberry Pi component operates reliably as a bridge between the backend and the printer, allowing real-time printing. The system can be expanded to support multiple printers or cloud-based operation in the future.

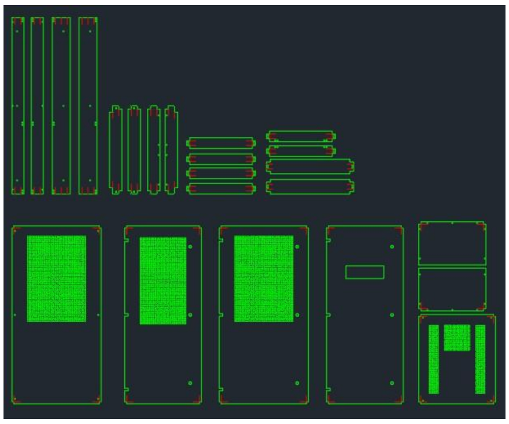
3.4. Physical Security Frame

For visual clarification of the components and steps described in this section, please refer to the corresponding figures in the appendices.

1. Initial Design on CAD Software

The first step was designing the frame using a 3D design software (like SolidWorks or AutoCAD). This design precisely determined the frame's dimensions, including length, width, and height, as well as the locations of the ventilation openings.



Large ventilation openings were designed on the sides, top, and bottom to ensure efficient cooling for the printer during operation.

The positions of the two top panels for mounting the printer were specified, along with the locations for the two fans (one for exhaust and one for air intake).

Frame Dimensions

The final metal enclosure was constructed with the following dimensions:

Component	Dimension (cm)
Total Height	140 cm
Width (outer)	60 cm
Depth	70 cm
Top Panel Thickness	1.5 mm galvanized steel
Ventilation Holes	10 cm x 10 cm grid
Fan Diameter	12 cm

These dimensions were carefully chosen to support the printer's size and allow sufficient air flow.

2. Material Analysis Using Specialized Software

After completing the design, the drawings were transferred to a specialized software (such as material analysis tools) to calculate the required amount of sheets and the number of pieces.

This software helped determine the number of galvanized steel sheets needed and allowed us to cut them accurately to minimize waste.

We ensured that each piece of the frame was cut to the exact dimensions before starting the manufacturing process.

3. Frame Manufacturing

We used heavy-duty galvanized steel to ensure the frame is strong, capable of supporting the printer's weight, and resistant to rust.

The sheets were cut to the specified dimensions using CNC machines for precision.

After cutting, ventilation holes were created on the sheets using a punching machine.

The sheets were assembled using screws and bolts, and welding was applied at certain points to enhance durability.

4. Installation of the Ventilation System

Two fans were installed on the frame's roof: an exhaust fan to remove hot air from inside and an intake fan to bring in cool air.
A thermostat was installed inside the frame to measure the temperature and control the fans' operation. If the temperature exceeds a certain threshold, the fans automatically turn on.

5. Setting Up the Electrical Circuit

An integrated electrical circuit was set up inside the frame. We used a power strip to connect all electrical components to a single point.
A circuit breaker was installed to protect the components from short circuits or overcurrent.
A single power output was extended from the box to connect the entire frame to one power source.
A switch was added to manage internet distribution to the printer and Raspberry Pi.

6. Installation of the Printer and Electronic Components

The printer was mounted on the two panels at the top of the frame, ensuring it was securely fixed to prevent movement during operation.
The Raspberry Pi was connected to the printer to control the printing process via the internet.
All electrical and internet cables were connected to the switch and power strip.

7. Testing and Operation

After completing the installation, we conducted a comprehensive test of the frame. We operated the printer to ensure it printed correctly without any issues.
The ventilation system and thermostat were tested, confirming that the fans operated properly when the temperature increased.
We verified that the electrical circuit was functioning safely and that the switch distributed the internet as required.

8. Packaging and Preparation for Use

After testing, the frame was cleaned and prepared for use.
Warning labels (such as "Beware of Electrical Wires") were added to the frame, and simple operating instructions were provided.

Chapter 4
Cost Estimation

Item	Quantity	Unit Price	Total Price
Xerox c 405 laser printer	1	-	13750
Raspberry pi 5 8 GB	1	-	6000
Rpi 5 power supply	1	-	1500
Rpi 5 cooler	1	-	450
Micro SD card 32 GB	1	-	380
Protection frame	1	-	10000
cooling fans	2	250	500
miniature	2	250	500
thermostat	1	-	400
Electrical wires and others	-	-	110
Frame wheels	4	40	160
Access point	1	-	450
power strip	1	-	250
printer cable	1	-	40
Ethernet cable 10 m	1	-	75
HDMI cable	1	-	75
Micro HDMI adapter	1	-	30
Vega to HDMI adapter	1	-	100
Shipping & Installation	-	-	2500

Total Cost	37270 جنيهًا

Chapter 5
Findings & Future Work

Experimental Results

During the development and testing phases, the Smart Printing System (SPS) was evaluated for functionality, reliability, and user experience. The system successfully handled file uploads, page count analysis using the smalot/pdfparser PHP library, and document printing via the Raspberry Pi using CUPS. Test cases included various file types, print settings, and multi-user scenarios. The interface was intuitive for users, and the admin dashboard provided real-time updates and control. The payment deduction mechanism worked smoothly, with virtual credits automatically subtracted upon successful printing.

System Testing Summary

During the testing phase, the system was evaluated across the following scenarios:

Test Case	Expected Result	Outcome	Status
User uploads valid PDF	File processed and cost shown	Success	Passed
Insufficient coin balance	Block print, show error message	Error shown	Passed
Print job processed by Raspberry Pi	Printer receives file, prints	Printed	Passed
File with password protection	Reject upload, notify user	Rejected	Passed
Network loss during print	Retry logic engaged	Recovered	Passed
Print duplex and color modes	As per selected options	Verified Passed	Passed

The system achieved a test pass rate of 100% across critical scenarios, ensuring high reliability in live usage.

Overall, the system operated with high efficiency and accuracy.

Conclusion

SPS provides a modern, efficient, and autonomous printing environment tailored to academic needs. It reduces operational burdens, enhances service availability, and empowers students with control over their printing tasks. The use of virtual currency and an integrated admin panel further elevate the system’s functionality.

Future Work

Future plans include the development of a mobile application that replicates the web interface, providing users with flexible printing access anytime and anywhere. Additionally, we plan to incorporate electronic payment gateways such as Paymob, allowing students to securely top up their virtual balance using bank cards or digital wallets. These enhancements will increase user convenience, system scalability, and integration with smart campus initiatives.

REFERENCES

This project was primarily developed through the practical efforts, programming experience, and problem-solving skills of the team members. While no academic research papers or textbooks were directly cited during the implementation, several official documentation sources and open-source libraries were consulted to ensure technical accuracy and successful integration of system components. The following references were instrumental in guiding the development process:

[1] Raspberry Pi Documentation – https://www.raspberrypi.com/documentation/

[2] CUPS – Common UNIX Printing System – https://www.cups.org/

[3] Bootstrap 5 Framework – https://getbootstrap.com/

[4] PHP Manual – https://www.php.net/

[5] Smalot PDF Parser Library (GitHub) – https://github.com/smalot/pdfparser

الملخص العربي

تواجه خدمات الطباعة التقليدية في المؤسسات الأكاديمية العديد من التحديات، من بينها الطوابير الطويلة، والتدخل اليدوي المستمر، وطرق الدفع غير الفعّالة، وغياب منظومة إدارة مركزية.

ومن هذا المنطلق، تم تطوير نظام الطباعة الذكي (SPS) كحل مبتكر ذاتي الخدمة، يهدف إلى توفير بيئة طباعة رقمية مؤتمتة بالكامل، مصممة خصيصًا لتلبية احتياجات الجامعات والبيئات التعليمية المشابهة.

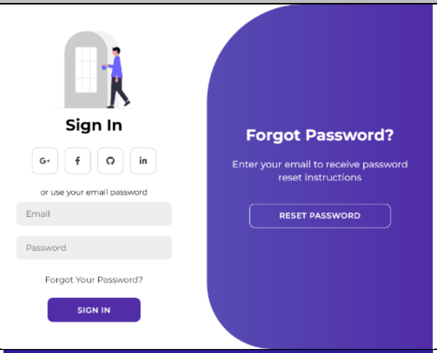
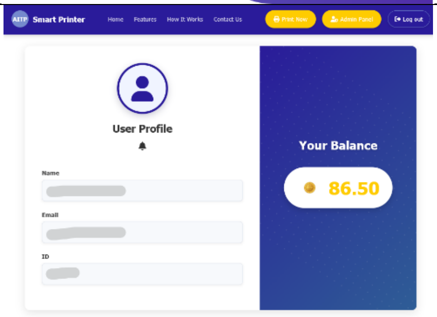
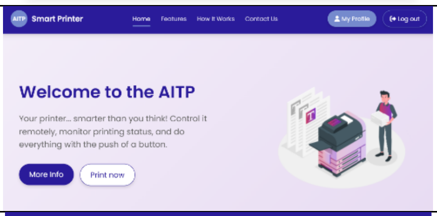
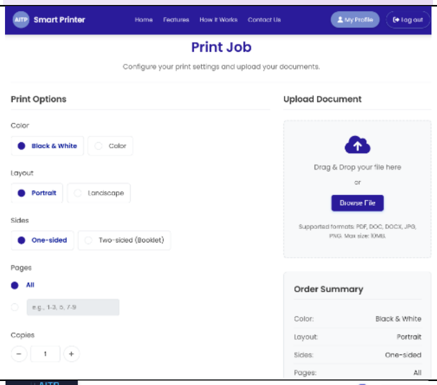
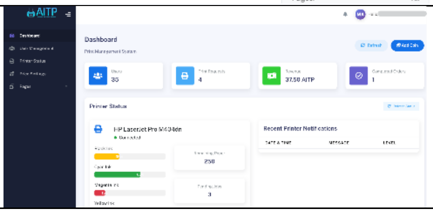
يُتيح النظام للمستخدمين رفع مستنداتهم عبر واجهة ويب سهلة وآمنة، مع إمكانية تخصيص خيارات الطباعة مثل حجم الورق، وضع الألوان، وعدد النسخ، وإتمام عملية الدفع إما باستخدام العملات المعدنية أو المحفظة الداخلية الافتراضية، والتي يمكن شحنها عن طريق التحويلات الإلكترونية.

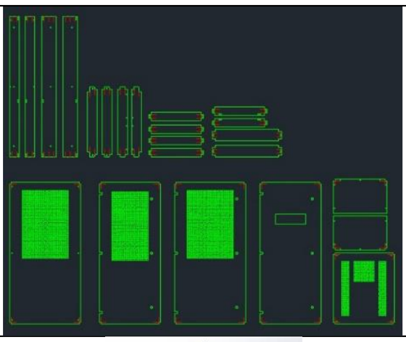
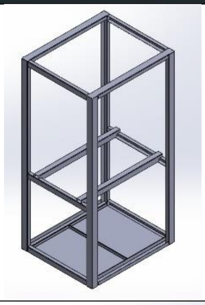
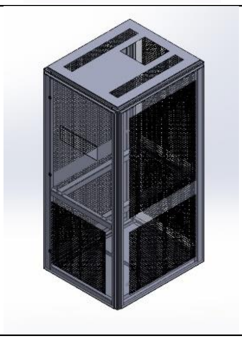
ويعتمد النظام في بنيته على جهاز Raspberry Pi كوحدة تحكم رئيسية تتولى التنسيق بين الموقع والطابعة باستخدام نظام CUPS، بالإضافة إلى قاعدة بيانات MySQL لإدارة بيانات المستخدمين وطوابير الملفات وسجلات المعاملات، إلى جانب واجهة برمجية RESTful واتصال فوري عبر WebSocket لتحديثات الحالة والتنبيهات اللحظية.

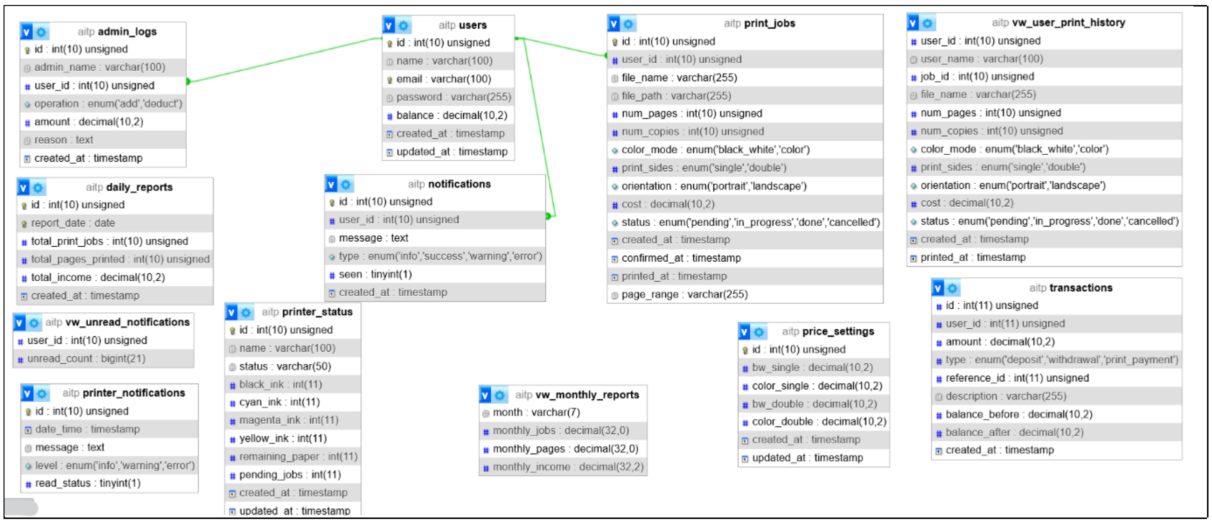
من أبرز مزايا هذا النظام: تقليل أوقات الانتظار، إلغاء الحاجة إلى موظفي طباعة مخصصين، إتاحة الخدمة على مدار الساعة، ودعم التحول الرقمي في الجامعة. كما يتميز النظام بهيكله المرن والقابل للتطوير، مما يسمح بإضافة مزايا مستقبلية مثل تطبيقات الهاتف المحمول، المصادقة عبر NFC، أو تحليلات متقدمة.

يقدم نظام الطباعة الذكي تجربة طباعة حديثة وآمنة وسهلة الاستخدام، مما يجعله إضافة قيّمة إلى بيئات الحرم الجامعي الذكي، ويساهم في تعزيز الاستقلالية والكفاءة والسلوك البيئي المسؤول لدى الطلاب.

Appendices

No.	Section	Description	Figure
1	Front-End	Sign In Page	 Sign In or use your email password Email Password Forgot Your Password? SIGN IN Forgot Password? Enter your email to receive password reset instructions RESET PASSWORD
2	Front-End	User Page	 Smart Printer Home Features How It Works Contact Us User Profile Name Email ID Your Balance 86.50
3	Front-End	Home Page	 Smart Printer Home Features How It Works Contact Us My Profile Log out Welcome to the AITP Your printer... smarter than you think! Control it remotely, monitor printing status, and do everything with the push of a button. More Info Print now
4	Front-End	Print Options Page	 Print Job Configure your print settings and upload your documents. Print Options Color: ● Black & White ○ Color Layout: ● Portrait ○ Landscape Sides: ● One-sided ○ Two-sided (Booklet) Pages: ● All ○ e.g., 1-3, 5, 7-9 Copies: − 1 + Upload Document Drag & Drop your file here or Browse File Supported formats: PDF, DOC, DOCX, JPG, PNG. Max size: 10MB Order Summary Color: Black & White Layout: Portrait Sides: One-sided Pages: All
5	Front-End	Admin Dashboard Page	 AITP Dashboard Printer Status HP LaserJet Pro M404dn Recent Printer Notifications 37.50 AITP

6	Hardware	Distribution of Metal Sheets Used in Frame Construction	
7	Hardware	Internal Structure with Printer Support Panels	
8	Hardware	Final Frame with Ventilation Openings	



```
aitp admin_logs
id : int(10) unsigned
admin_name : varchar(100)
user_id : int(10) unsigned
operation : enum('add','deduct')
amount : decimal(10,2)
reason : text
created_at : timestamp

aitp daily_reports
id : int(10) unsigned
report_date : date
total_print_jobs : int(10) unsigned
total_pages_printed : int(10) unsigned
total_income : decimal(10,2)
created_at : timestamp

aitp vw_unread_notifications
user_id : int(10) unsigned
unread_count : bigint(21)

aitp printer_notifications
id : int(10) unsigned
date_time : timestamp
message : text
level : enum('info','warning','error')
read_status : tinyint(1)

aitp users
id : int(10) unsigned
name : varchar(100)
email : varchar(100)
password : varchar(255)
balance : decimal(10,2)
created_at : timestamp
updated_at : timestamp

aitp notifications
id : int(10) unsigned
user_id : int(10) unsigned
message : text
type : enum('info','success','warning','error')
seen : tinyint(1)
created_at : timestamp

aitp printer_status
id : int(10) unsigned
name : varchar(100)
status : varchar(50)
black_ink : int(11)
cyan_ink : int(11)
magenta_ink : int(11)
yellow_ink : int(11)
remaining_paper : int(11)
pending_jobs : int(11)
created_at : timestamp
updated_at : timestamp

aitp vw_monthly_reports
month : varchar(7)
monthly_jobs : decimal(32,0)
monthly_pages : decimal(32,0)
monthly_income : decimal(32,2)

aitp print_jobs
id : int(10) unsigned
user_id : int(10) unsigned
file_name : varchar(255)
file_path : varchar(255)
num_pages : int(10) unsigned
num_copies : int(10) unsigned
color_mode : enum('black_white','color')
print_sides : enum('single','double')
orientation : enum('portrait','landscape')
cost : decimal(10,2)
status : enum('pending','in_progress','done','cancelled')
created_at : timestamp
confirmed_at : timestamp
printed_at : timestamp
page_range : varchar(255)

aitp price_settings
id : int(10) unsigned
bw_single : decimal(10,2)
color_single : decimal(10,2)
bw_double : decimal(10,2)
color_double : decimal(10,2)
created_at : timestamp
updated_at : timestamp

aitp vw_user_print_history
user_id : int(10) unsigned
user_name : varchar(100)
job_id : int(10) unsigned
file_name : varchar(255)
num_pages : int(10) unsigned
num_copies : int(10) unsigned
color_mode : enum('black_white','color')
print_sides : enum('single','double')
orientation : enum('portrait','landscape')
cost : decimal(10,2)
status : enum('pending','in_progress','done','cancelled')
created_at : timestamp
printed_at : timestamp

aitp transactions
id : int(11) unsigned
user_id : int(11) unsigned
amount : decimal(10,2)
type : enum('deposit','withdrawal','print_payment')
reference_id : int(11) unsigned
description : varchar(255)
balance_before : decimal(10,2)
balance_after : decimal(10,2)
created_at : timestamp
```

The database diagram