



CERTIFICATE GENERATOR USING PYTHON TKINTER



CS3711 - SUMMER INTERNSHIP

Submitted By

RAMKUMAR R

(732521104040)

in partial fulfilment for the award of the degree
of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SHREE VENKATESHWARA HI-TECH ENGINEERING COLLEGE

GOBI-638455

BONAFIDE CERTIFICATE

Certified that this Summer Internship and Project report **“CERTIFICATE GENERATOR USING PYTHON TKINTER”** is the bonafide work of **“RAMKUMAR R (732521104040)”** who carried out the Summer Internship and Project work under my supervision.

SIGNATURE

Dr.T.SENTHILPRAKASH,M.E,Ph.D.,
Professor and Head
Department of Computer Science and
Engineering
Shree Venkateshwara Hi-Tech
Engineering College,
Gobichettipallayam-638455
Erode(D.T),TN.

SIGNATURE

J. VIJAYASHANKAR MCA., MBA.,
ME., PHD,
Managing Director
The Global Service , 139,3rd Floor
Golden Castle Building,
Above PSR Silks,
Sathy Main Road
Erode(D.T),TN

CERTIFICATE GENERATOR USING PYTHON TKINTER

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and Deep regards to our beloved **Thiru.P.VENKATACHALAM**, Chairman, **Thiru.K.C.KARUPANAN**, Secretary, **Thiru. G.P.KETTIMUTHU**, Joint Secretary, and the Management of Shree Venkateshwara Hi-Tech Engineering College who provided all the facilities with the state of art of infrastructure.

I express my earnest sense of gratitude to my beloved Principal **Dr.P.THANGAVEL, M.E., MBA, Ph.D.**, Shree Venkateshwara Hi-Tech Engineering College whose immense help has been a candle in the darkness with regards to the Summer Internship project work.

I'm very grateful to **Dr.T.SENTHIL PRAKASH ME., PhD, Professor and Head , Department of Computer Science and Engineering** , for the aspiring suggestion , invaluable constructive criticism and friendly advice during the Summer Internship and Project period

I would like to express my profound thanks to my project coordinator **Mr.J.VIJAYASANKAR MCA.,MBA.,ME.,PHD, managing director, THE GLOBAL SERVICE** thoughtful words helped us in completing my Summer Internship and Project successfully.

Also, I thank all the Teaching and non teaching staff, who have patiently provided us assistance in this Summer Internship and Project. I also thank to my classmates for their encouragement and help during the course of the Summer Internship and Project.

ABSTRACT

This Summer Internship and Project aims to develop a Certificate Generator using Python Tkinter library to create a user-friendly Graphical User Interface (GUI). The primary goal of this application is to automate the process of generating certificates for various purposes, such as courses, events, or workshops. Users can input essential certificate details, including the recipient's name, course title, date of issue, and the instructor's or organization's name.

The Summer Internship and Project leverages several Python libraries to accomplish this task. Tkinter is utilized to design the input form, allowing users to enter the required details in a simple, easy-to-navigate layout. The Pillow library (PIL) is used to work with image-based certificate templates, enabling the application to position the text accurately on the template. The program allows users to upload a pre-designed certificate template, add the dynamic text to it, and save the output as an image file (e.g., PNG or JPEG). If PDF generation is required, the ReportLab library is integrated to create certificates in PDF format, offering flexibility in output choices.

This tool simplifies certificate generation, eliminating the need for manual editing of templates in graphic design software for each new certificate. It ensures consistency and reduces errors, which can occur when manually creating certificates in large batches. It is especially beneficial for educational institutions, businesses, and event organizers that regularly issue certificates.

Moreover, the Summer Internship and Project demonstrates the integration of Python for real-world applications with GUI design and automation. By combining Tkinter for GUI, Pillow for image manipulation, and ReportLab for PDF generation, the application offers a practical solution to streamline certificate creation. This Summer Internship and Project showcases Python's versatility in building functional tools and provides an efficient method for generating certificates accurately.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	5
	LIST OF ABBREVIATIONS	10
	LIST OF FIGURES	11
1	INTRODUCTION	12
	1.1 INTRODUCTION	12
2	LITERATURE SURVEY	13
	2.1 CERTIFICATE GENERATING TOOL	13
	2.2 PYTHON LIBRARIES FOR GUI	13
	2.3 PYTHON LIBRARIES FOR IMAGE	14
	AND PDF PROCESSING	
	2.4 AUTOMATED WORKFLOW INTEGRATION	14
3	SYSTEM ANALYSIS	15
	3.1 EXISTING SYSTEM	15
	3.1.1 DISADVANTAGES	16
	3.2 PROPOSED SYSTEM	16
	3.2.1 ADVANTAGES	17

4	SYSTEM SPECIFICATION	19
	4.1 SYSTEM REQUIREMENTS	19
	4.1.1 SOFTWARE REQUIREMENTS	19
	4.1.2 HARDWARE REQUIREMENTS	19
5	SYSTEM ENVIRONMENTS	20
	5.1 ALGORITHM	20
6	SYSTEM IMPLEMENTATION	22
	6.1 MODULE DESCRIPTION	22
	6.1.1 USER INTERFACE	22
	6.1.2 DATA HANDLING	22
	6.1.3 CERTIFICATE GENERATION	22
	6.1.4 FILE MANAGEMENT	23
	6.1.5 ERROR HANDLING	23
	6.2 DIAGRAM	24
	6.2.1 SYSTEM FLOW DIAGRAM	24
	6.2.2 DATA FLOW DIAGRAM	25
7	SYSTEM DESIGN	26
	7.1 DETAILED DESIGN	26
	7.1.1 HARDWARE DESIGN	26

	7.1.2 SOFTWARE DESIGN	27
8	SYSTEM STUDY	28
	8.1 FEASIBILITY STUDY	28
	8.1.1 ECONOMICAL FEASIBILITY	28
	8.1.2 TECHNICAL FEASIBILITY	29
	8.1.3 OPERATIONAL FEASIBILITY	29
9	SYSTEM TESTING	31
	9.1 TESTING	31
	9.2 TYPES OF TEST	31
	9.2.1 UNIT TESTING	31
	9.2.2 INTEGRATION TESTING	31
	9.2.3 FUNCTIONAL TESTING	32
	9.2.4 ACCEPTANCE TESTING	32
	9.2.5 WHITE BOX TESTING	33
	9.2.6 BLACK BOX TESTING	33
10	CONCLUSION AND FUTURE ENHANCEMENT	34
	10.1 CONCLUSION	34
	10.2 FUTURE ENHANCEMENT	34

11	APPENDICES	36
	11.1SAMPLE CODE	36
	11.2 SCREENSHOT	39
	11.2.1 CERTIFICATE GENERATE PAGE	39
	11.2.2 OUTPUT IMAGE	40
	11.3 REFERENCES	41

LIST OF ABBREVIATIONS

ACRONOMY

ABBREVIATIONS

GUI	Graphical User Interface
PIL	Python Imaging Library
PNG	Portable Network Graphics
JPEG	Joint Photographic Experts Groups
PDF	Portable Document Format
Tk	Toolkit
API	Application Programming Interface

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
6.2.1	System Flow Diagram	24
6.2.2	Data Flow Diagram	25
11.2.1	Certificate Generate Page	39
11.2.2	Output Image	40

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Certificate Generator Using Python Tkinter is a desktop application designed to streamline the creation of personalized certificates using the Tkinter library for its Graphical User Interface (GUI). In educational institutions, professional events, or online courses, generating certificates is a common yet repetitive task. Manually creating certificates for each participant can be time-consuming and error-prone. This Summer Internship and Project aims to automate the certificate generation process, allowing users to quickly input details like the recipient's name, course or event title, date, and the instructor's or organization's name.

By leveraging libraries such as Pillow (for image manipulation) and ReportLab (for PDF generation), this tool offers flexibility in terms of certificate format and layout. Users can select from pre-designed certificate templates and generate high-quality certificates in either image or PDF formats with dynamic text placement. This ensures uniformity in certificate design while significantly reducing the manual effort involved in creating multiple certificates.

The Summer Internship and Project highlights Python's versatility for real-world applications, combining GUI development, automation, and file handling. This tool is particularly useful for organizations that frequently issue certificates, such as schools, training centers, and event organizers, making the certificate creation process more efficient, accurate, and scalable.

CHAPTER 2

LITERATURE SURVEY

A literature survey for a Certificate Generator Using Python Tkinter explores various existing tools, libraries, and methodologies relevant to certificate generation, GUI design, and automation. Here is a review of pertinent literature and tools with authors:

2.1.Certificate Generation Tools:

Certificate Maker Tools: Tools like Adobe Spark and Canva are popular for designing customizable certificates. These web-based tools offer ease of use but generally lack automation for bulk certificate generation. (Adobe Systems Inc., 2023; Canva, 2023)

Automated Certificate Generation: Platforms such as DocuSign and Certifier provide automated certificate creation, typically integrated with online systems. While powerful, these tools may not always offer the customization needed for specific applications. (DocuSign, 2023; Certifier, 2023)

2.2.Python Libraries for GUI:

Tkinter: Tkinter is the standard GUI library for Python, providing a straightforward method to create windows, dialogs, and forms. Its simplicity and direct integration with Python make it an excellent choice for desktop applications (Grayson, 2013).

PyQt and Kivy: Alternatives like PyQt and Kivy offer more advanced features and modern UI components. PyQt is known for its comprehensive set of tools but

can be more complex to use (Summerfield, 2008). Kivy is a more recent library designed for multi-touch applications and mobile apps (Kivy.org, 2023).

2.3. Python Libraries for Image and PDF Processing:

Pillow (PIL): Pillow, the updated fork of the Python Imaging Library (PIL), is extensively used for image processing tasks in Python. It allows for editing images and adding text overlays (Lutz, 2013).

ReportLab: ReportLab is a robust library for generating PDFs in Python, supporting complex document creation with text and graphics. It is well-regarded for its flexibility in producing high-quality PDF documents (Kraak, 2009).

2.4. Automated Workflow Integration:

Python Automation: Python's capability to automate repetitive tasks through scripting is well-documented. Automation of tasks like certificate generation can be efficiently managed using Python scripts that interface with GUI elements and handle batch processes (Beazley, 2009).

CHAPTER 3

SYSTEM ANALYSIS

3.1.EXISTING SYSTEM

Several systems currently address certificate generation, each with different levels of automation and customization. Canva is a popular online design platform that offers various certificate templates. Users can manually input information and export certificates as images or PDFs. While Canva is easy to use and visually flexible, it lacks automated bulk generation and system integration, which limits its utility for large-scale certificate issuance (Canva,).Adobe Spark also enables certificate creation through customizable templates but requires manual entry for each certificate, making it less practical for high-volume needs .

In contrast, platforms like DocuSign and Certifier provide more automation. DocuSign offers automated certificate generation within its digital signature and document management services, integrating with various systems to simplify certificate issuance. However, its primary focus is on digital signatures rather than extensive design customization (DocuSign, 2023). Certifier is designed for the automated issuance of digital certificates and badges, integrating with learning management systems (LMS) to handle large volumes based on preset criteria. Despite its bulk issuance capabilities, Certifier may offer limited design customization compared to specialized design tools (Certifier).

These existing systems reveal the need for a solution that blends ease of use, customization, and automation, which the Python Certificate Generator Summer Internship and Project aims to deliver by allowing users to efficiently create personalized certificates.

3.1.1 DISADVANTAGES

1. Lack of Automation (Canva, Adobe Spark):

- Requires manual input for each certificate, making it time-consuming and inefficient for bulk generation.

2. Limited Integration (Canva, Adobe Spark):

- Does not integrate easily with other systems for automated data input, reducing scalability for larger organizations.

3. Design Customization Limitations (DocuSign, Certifier):

- Focuses primarily on document management and digital signatures, offering limited flexibility for detailed certificate design.

4. Restricted Bulk Customization (Certifier):

- Although it supports bulk issuance, customization options for certificate templates are limited compared to dedicated design tools.

5. Cost Constraints:

- Many of these tools, particularly DocuSign and Certifier, can be expensive, especially for long-term use or large-scale organizations.

6. Limited Offline Functionality:

- Most platforms are web-based, requiring an internet connection, which may not be ideal for all environments.

3.2. PROPOSED SYSTEM

The proposed system, a Python Certificate Generator using Tkinter, addresses the limitations of existing certificate generation tools by combining ease of use, extensive customization, and automation. This desktop application provides a graphical user interface (GUI) built with Tkinter, allowing users to input details such

as the recipient's name, course title, date, and instructor's name through an intuitive form. The system utilizes the Pillow library to manipulate and overlay text on certificate templates, which can be pre-designed and uploaded by the user. This enables the generation of certificates in image formats like PNG or JPEG, or in PDF format using the ReportLab library, providing flexibility in output choices.

The application is designed to automate the certificate creation process, reducing manual effort and ensuring consistency across multiple certificates. By integrating these tools, the proposed system offers a streamlined, efficient solution for generating personalized certificates quickly and accurately, overcoming the drawbacks of current systems such as limited automation and design flexibility. This approach not only enhances the efficiency of certificate issuance but also provides users with the ability to customize certificates to meet specific requirements, making it ideal for educational institutions, event organizers, and other organizations that need to issue certificates regularly.

3.2.1 ADVANTAGES

1. **Automation:** Automatically generates certificates in bulk, reducing manual effort and saving time.
2. **Customization:** Allows users to customize certificate templates with specific details like names, course titles, and dates.
3. **User-Friendly GUI:** Features an intuitive, easy-to-use interface built with Tkinter for smooth user interaction.
4. **Multiple Output Formats:** Supports both image (PNG, JPEG) and PDF formats for certificate output, providing flexibility.
5. **Consistency:** Ensures uniformity and accuracy across all certificates by using

pre-designed templates.

6. **Integration with Libraries:** Utilizes Pillow for image manipulation and ReportLab for PDF generation, offering advanced design and formatting options.
7. **Cost-Effective:** As an open-source Python-based solution, it is free to use and modify, avoiding expensive licensing fees.
8. **Scalability:** Can handle the generation of a large number of certificates, making it suitable for institutions and events with many participants.
9. **Cross-Platform Compatibility:** As a Python application, it works on multiple platforms, including Windows, macOS, and Linux.

CHAPTER 4

SYSTEM SPECIFICATION

4.1.SYSTEM REQUIREMENTS

4.1.1 SOFTWARE REQUIREMENTS

- Operating System : Windows 10/11, macOS, or Linux (Ubuntu).
- Platform : GUI application
- Tool : Integrated Development Environment (IDE)
- Front End : Tkinter , Python-version 3.12.3
- Back End : Mysql - version 8.0

4.1.2 HARDWARE REQUIREMENTS

- Processor : Intel Core processor
- Hard Disk : 500 GB or more
- RAM : 8 GB or above

CHAPTER 5

SYSTEM ENVIRONMENT

5.1.ALGORITHM

The Python Certificate Generator using Python Tkinter project primarily involves basic programming logic rather than complex algorithms. However, several steps and procedures (which can be considered algorithms) are used to manage the certificate generation process. These include:

Text Placement Algorithm:

The application needs to determine where to place text (e.g., recipient's name, course name, date) on the certificate template. This typically involves calculating coordinates (x, y positions) for the text based on the dimensions of the certificate template, ensuring the text is centered or positioned correctly.

Input Validation Algorithm:

Basic input validation is required to ensure the fields (name, course, etc.) are filled in correctly. For instance, ensuring that all fields are completed before allowing the user to generate the certificate, or preventing invalid characters from being entered.

Image Manipulation Algorithm (using Pillow):

After input is provided, the Pillow library handles the image processing. This involves opening a pre-designed template image, overlaying the user's input onto specific areas of the image (using calculated coordinates), and saving the modified image as a new certificate file.

PDF Generation Algorithm (using ReportLab):

If PDF output is required, ReportLab is used to create a PDF document. This involves calculating where to place text and any other graphical elements in the document.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1.MODULE DESCRIPTION

6.1.1 User Interface Module:

- **Purpose:** Provides the graphical user interface (GUI) for user interactions.
- **Components:** Created using Tkinter, this module includes input forms for entering certificate details such as recipient's name, course title, date, and instructor's name. It also features buttons for submitting data and selecting certificate formats.
- **Functionality:** Collects user inputs, validates them for completeness and correctness, and triggers the certificate generation process upon user request.

6.1.2 Data Handling Module:

- **Purpose:** Manages data retrieval and storage.
- **Components:** Interfaces with MySQL to store and retrieve certificate templates and user data. It handles database connections and performs CRUD (Create, Read, Update, Delete) operations as required.
- **Functionality:** Retrieves the appropriate certificate template based on user selection, stores user input data, and ensures data integrity throughout the process.

6.1.3 Certificate Generation Module:

- **Purpose:** Handles the creation of the certificate.
- **Components:** Utilizes the Pillow library for image manipulation.

- **Functionality:** Applies user-provided details to the selected certificate template. This module ensures that text is accurately placed and formatted according to the template's design. It generates and saves the final certificate in the chosen format (image or PDF).

6.1.4 File Management Module:

- **Purpose** : Manages file storage and retrieval.
- **Components** : Handles saving and accessing generated certificates.
- **Functionality:** Saves the generated certificate to the user's specified directory or prompts for a save location. Manages file naming conventions and ensures proper file handling.

6.1.5 Error Handling Module:

- **Purpose:** Provides robust error management and user feedback
- **Components:** Includes exception handling mechanisms to address potential issues during user input, data retrieval, template processing, and file saving.
- **Functionality:** Captures and reports errors to the user, allowing for corrective actions and ensuring a smooth user experience.

6.2 DIAGRAM

6.2.1 SYSTEM FLOW DIAGRAM

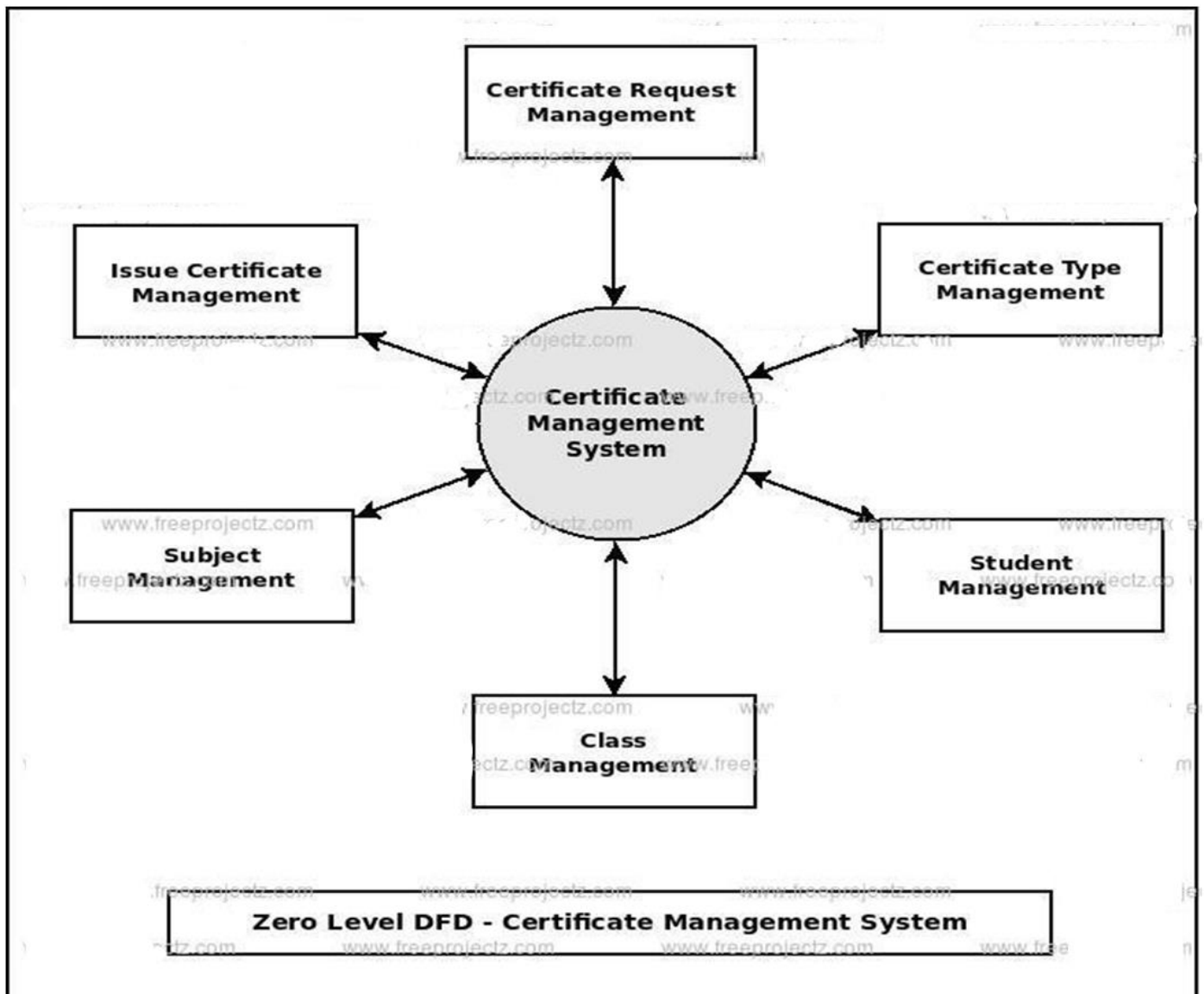


Fig.No.6.2.1.SYSTEM FLOW DIAGRAM

6.2.2 DATA FLOW DIAGRAM

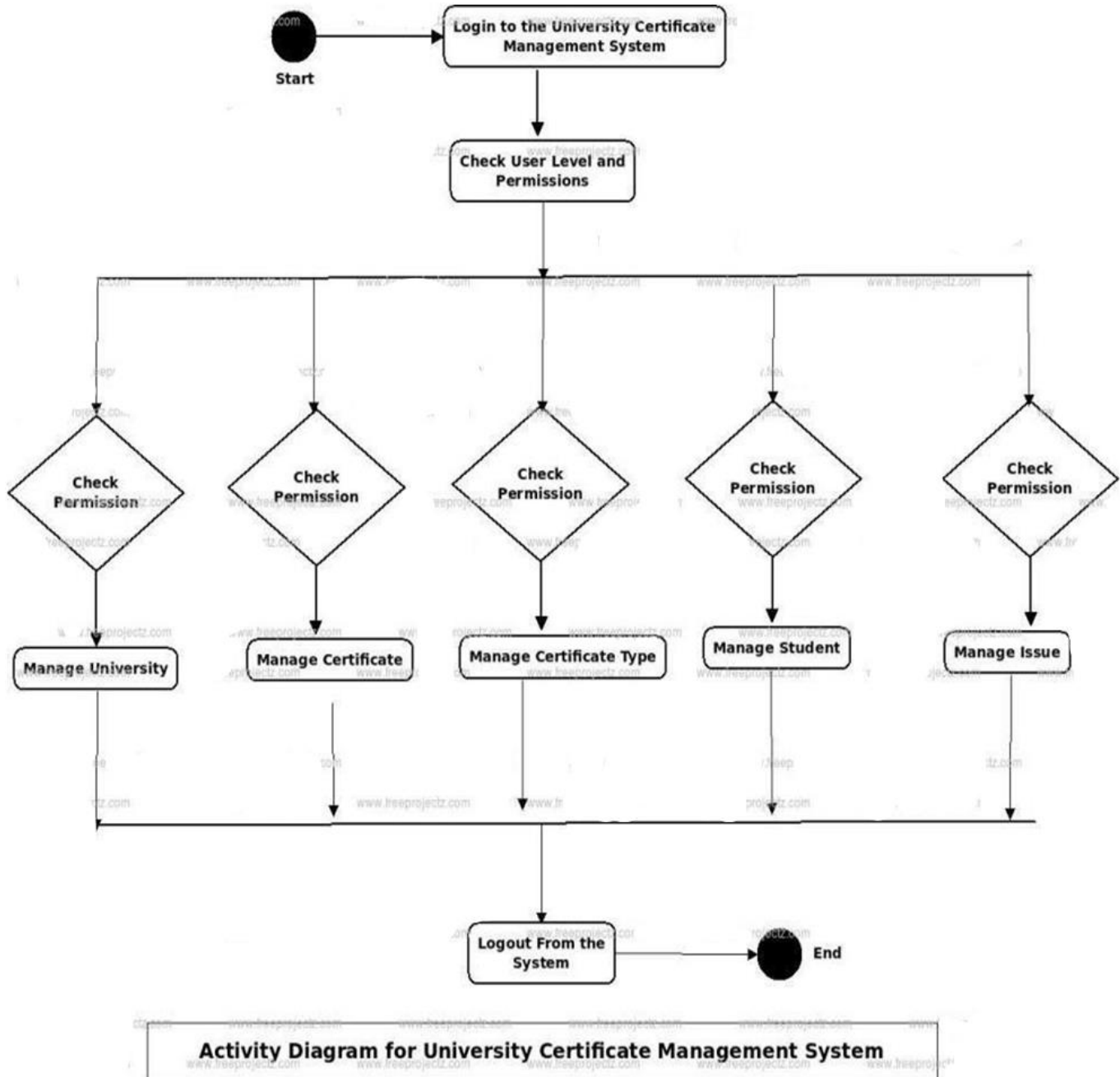


Fig.No.6.2.2. DATA FLOW DIAGRAM

CHAPTER 7

SYSTEM DESIGN

7.1.DETAILED DESIGN

The detailed design of the system outlines the workflow and structure of each module. The user interface, built with Tkinter, allows users to input necessary details and select a template. The data handling module, connected to MySQL, manages the storage and retrieval of user data and templates. For certificate generation, the system uses Pillow to manipulate image templates, overlaying text on designated areas, and ReportLab for PDF creation. The file management module handles saving certificates in user-specified formats and directories. Error handling is integrated throughout the system to ensure that issues like incorrect inputs or file errors are properly managed, enhancing the reliability of the application.

7.1.1 HARDWARE DESIGN

The hardware design ensures that the application runs efficiently on standard systems. The minimum hardware requirements include a Intel core processor to handle multiple operations and certificate generation tasks smoothly. At least 8 GB of RAM is needed for effective multitasking, particularly when generating multiple certificates or handling large template files. Additionally, 500 GB of storage is recommended to accommodate the application, database, and stored certificates. The hardware design ensures the system remains responsive and functional, even on mid-range machines.

7.1.2 SOFTWARE DESIGN

The software design leverages a combination of Python libraries and frameworks. Tkinter is used for the front end, providing a simple yet effective GUI for users. MySQL serves as the backend, offering a structured way to manage and store data. The Pillow library handles image-based certificates, allowing text to be added dynamically to templates, while ReportLab generates certificates in PDF format. The system is developed using an Integrated Development Environment (IDE) such as PyCharm or Visual Studio Code, ensuring an efficient development and debugging process. The software design emphasizes flexibility, ease of use, and scalability for future enhancements like batch processing or role-based access.

CHAPTER 8

SYSTEM STUDY

8.1.FEASIBILITY STUDY

Before the development of the Python Certificate Generator Summer Internship and Project, a feasibility study is essential to evaluate its practicality from multiple perspectives: technical, operational, and economic. This ensures that the Summer Internship and Project is viable and that resources invested will lead to the successful deployment of the system.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ OPERATIONALFEASIBILITY

8.1.1 ECONOMICAL FEASIBILITY

Economic feasibility assesses the cost-effectiveness of the Summer Internship and Projectby comparing the expected costs to the anticipated benefits. The costs associated with the development of the system include software development, hardware procurement, database setup, and ongoing maintenance. Since the system utilizes open-source technologies like Python and MySQL, software costs are minimized. The primary expenses involve hardware upgrades (if necessary), development time, and training for users. In contrast, the benefits include improved efficiency, reduced paperwork, streamlined processes.The automation of tasks like admission processing, fee management, and attendance tracking will save

significant time and resources. Additionally, the system will reduce errors associated with manual data handling, leading to more accurate and reliable records. The return on investment (ROI) is expected to be positive, as the long-term savings and improvements in institutional operations will outweigh the initial costs.

8.1.2 TECHNICAL FEASIBILITY

The Summer Internship and Project is built on reliable and well-supported technologies, including Python, Tkinter, Pillow, MySQL, and ReportLab. Python is a versatile and widely used programming language, making it easy to find resources, libraries, and support. Tkinter provides a straightforward and effective way to create Graphical User Interfaces (GUIs), ensuring the application is user-friendly. The integration of Pillow for image manipulation and ReportLab for PDF generation ensures the system can handle diverse output formats. Moreover, MySQL is a robust choice for data storage, offering scalability and security. The hardware requirements are modest, with the system capable of running smoothly on a machine with a dual-core processor, 4 GB of RAM, and 500 GB of storage, making the technical feasibility strong.

8.1.3 OPERATIONAL FEASIBILITY

The system is designed to be user-friendly, allowing individuals with minimal technical skills to generate certificates effortlessly. Tkinter provides an intuitive interface for data input and template selection, streamlining the certificate generation process. Automation of certificate generation significantly reduces manual work, saving time and ensuring accuracy. This makes the system highly practical for educational institutions, event organizers, and businesses that frequently issue

certificates. The application's modular design allows for easy updates and future improvements, ensuring long-term usability and scalability.

CHAPTER 9

SYSTEM TESTING

9.1 TESTING

Testing is a crucial phase in the development of the Certificate Generator Using Python Tkinter Summer Internship and Project to ensure the system functions as expected and meets the Summer Internship and Project's requirements. The testing process will involve various types of tests to evaluate the system's functionality, performance, and usability. Below is a breakdown of the testing methodology for the Summer Internship and Project:

9.2 TYPES OF TESTING

9.2.1 UNIT TESTING

- **Objective:** To verify the accuracy of individual components or functions, ensuring they work as expected in isolation.
- **Focus:** Each function or class, such as certificate data input, text overlay on templates, and file-saving functionality, will be tested independently. For example, the certificate generation process using the Pillow library will be unit tested to ensure the text is correctly overlaid on the chosen template.
- **Example:** Verifying that a name entered in the Tkinter GUI appears in the correct position on the certificate template.

9.2.2 INTEGRATION TESTING

- **Objective:** To test the interaction between different system modules, ensuring that they work harmoniously when combined.

- **Focus:** Testing the integration between the Tkinter GUI, MySQL database, and certificate generation engine. This involves ensuring that user data entered into the GUI is correctly passed to the database, retrieved for certificate generation, and saved in the appropriate format.
- **Example:** Testing the process where a user inputs data, selects a template from the database, and generates the certificate as a PDF or PNG file.

9.2.3 FUNCTIONAL TESTING

- **Objective:** To confirm that the system functions according to the specified requirements and produces the expected results.
- **Focus:** All key features will be tested, including the user input, template selection, certificate preview, and file-saving functions. Functional tests will ensure the system behaves as intended and that the user can generate certificates without errors.
- **Example:** Generating certificates using different templates and formats (PNG, PDF) and ensuring the correct data appears on each certificate.

9.2.4 ACCEPTANCE TESTING

- **Objective:** To ensure the final product meets the end-user's expectations and is ready for deployment.
- **Focus:** End-users will test the system in a real-world scenario, verifying that all features function correctly and the system meets the Summer Internship and Project requirements.
- **Example:** Test users will generate certificates using the system to ensure it behaves as expected and provides the required output.

9.2.5 WHITE BOX TESTING

White Box Testing (or clear box testing) involves testing the internal structures or workings of an application, as opposed to its functionality (black box testing). Testers with knowledge of the internal code conduct this testing to ensure that all paths, branches, and loops within the code are functioning correctly. For instance, the tester might check whether all possible conditions in a decision-making code structure are properly evaluated and handled.

9.2.6 BLACK BOX TESTING

Black Box Testing focuses on testing the functionality of the system without knowing its internal code structure. The tester only considers the inputs and outputs. For this Summer Internship and Project, black box testing would involve inputting data into various forms (such as the student enquiry form) and verifying that the correct outputs (e.g., data stored in the database, confirmation messages) are produced.

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

10.1 CONCLUSION

The Certificate Generator Using Python Summer Internship and Project, developed using Tkinter, offers an efficient and user-friendly solution for automating the creation of customized certificates. By leveraging Python's versatility, the Summer Internship and Project successfully integrates a graphical user interface, database connectivity with MySQL, and certificate generation capabilities through libraries like Pillow and ReportLab. This system significantly reduces the time and manual effort required for certificate generation, making it ideal for educational institutions, event organizers, and businesses. The Summer Internship and Project is technically feasible, operationally efficient, and economically viable, ensuring that it meets the demands of users with minimal resource investment. The automation of certificate creation improves accuracy, ensures consistency, and eliminates human errors associated with manual methods. The modular design of the system also allows for future scalability and easy maintenance.

10.2 FUTURE ENHACEMENT

In the future, the Python Certificate Generator can be expanded with additional features to enhance its functionality and user experience:

- a. Batch Certificate Generation
- b. Cloud Integration
- c. Customizable Templates

- d. Role-Based Access Control
- e. Multilingual Support
- f. Web-Based Version

These enhancements would not only make the system more robust and flexible but also cater to a wider range of users, ensuring long-term sustainability and growth of the Summer Internship and Project.

CHAPTER 11

APPENDICES

11.1 SAMPLE CODE

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageDraw, ImageFont, ImageTk

def generate_certificate():
    name = entry_name.get()
    course_title = entry_course.get()
    duration = entry_duration.get()
    signature = entry_signature.get()

    if not name or not course_title or not duration or not signature:
        messagebox.showerror("Input Error", "All fields are required!")
        return
    try:
        template_path = "certificate_template.png"
        certificate = Image.open(template_path)
        draw = ImageDraw.Draw(certificate)

        font_path = "calibri.ttf"
        font_large = ImageFont.truetype(font_path, 40)
        font_medium = ImageFont.truetype(font_path, 30)
```

```

font_small = ImageFont.truetype(font_path, 20)

cert_width, cert_height = certificate.size

draw.text((cert_width // 2, 250), f"This is to certify that",
font=font_medium, fill="black", anchor="ms")

draw.text((cert_width // 2, 300), name, font=font_medium, fill="black",
anchor="ms")

draw.text((cert_width // 2, 350), f"has successfully completed the
course", font=font_medium, fill="black", anchor="ms")

draw.text((cert_width // 2, 400), course_title, font=font_medium,
fill="black", anchor="ms")

draw.text((cert_width // 2, 450), f"Duration: {duration}",
font=font_medium, fill="black", anchor="ms")

draw.text((cert_width - 200, cert_height - 100), signature,
font=font_small, fill="black", anchor="ms")

draw.text((cert_width - 200, cert_height - 70), "Signature",
font=font_medium, fill="black", anchor="ms")

save_path = filedialog.asksaveasfilename(defaultextension=".png",
filetypes=[("PNG files", "*.png")])

if save_path:
    certificate.save(save_path)

    messagebox.showinfo("Success", "Certificate generated
successfully!")

except Exception as e:

```

```
        messagebox.showerror("Error", str(e))
root = tk.Tk()
root.title("Certificate Generator")
tk.Label(root, text="Name:").grid(row=0, column=0, padx=10, pady=10)
entry_name = tk.Entry(root, width=30)
entry_name.grid(row=0, column=1, padx=10, pady=10)

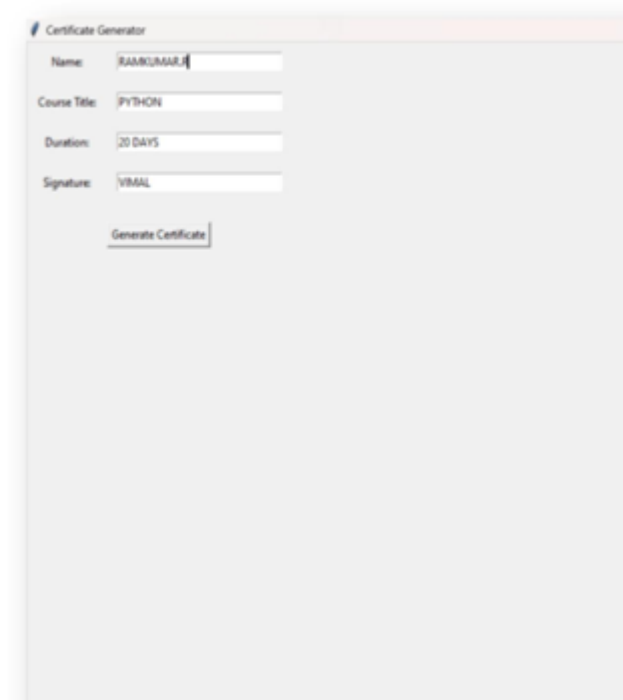
tk.Label(root, text="Course Title:").grid(row=1, column=0, padx=10,
pady=10)
entry_course = tk.Entry(root, width=30)
entry_course.grid(row=1, column=1, padx=10, pady=10)

tk.Label(root, text="Duration:").grid(row=2, column=0, padx=10, pady=10)
entry_duration = tk.Entry(root, width=30)
entry_duration.grid(row=2, column=1, padx=10, pady=10)

tk.Label(root, text="Signature:").grid(row=3, column=0, padx=10, pady=10)
entry_signature = tk.Entry(root, width=30)
entry_signature.grid(row=3, column=1, padx=10, pady=10)
generate_button = tk.Button(root, text="Generate Certificate",
command=generate_certificate)
generate_button.grid(row=4, column=0, columnspan=2, pady=20)
root.mainloop()
```

11.2 SCREENSHOTS

11.2.1 CERTIFICATE GENERATOR PAGE



The screenshot displays a web application titled "Certificate Generator". It features a form with the following fields and values:

- Name: RAMKUMARU
- Course Title: PYTHON
- Duration: 20 DAYS
- Signature: VMAL

Below the input fields is a button labeled "Generate Certificate".

Fig.No.11.2.1. CERTIFICATE GENERATOR PAGE

11.2.2 OUTPUT IMAGE



Fig.No.11.2.2. OUTPUT IMAGE

11.3 REFERENCES

1. Python Documentation:

- Python Software Foundation. (n.d.). Python 3 documentation. Retrieved from <https://docs.python.org/3/>

2. Tkinter Documentation:

- Python Software Foundation. (n.d.). Tkinter documentation. Retrieved from <https://docs.python.org/3/library/tk.html>

3. Pillow Library Documentation:

- Pillow Developers. (n.d.). Pillow (PIL Fork) Documentation. Retrieved from <https://pillow.readthedocs.io/en/stable/>

4. ReportLab Documentation:

- ReportLab. (n.d.). ReportLab User Guide. Retrieved from <https://www.reportlab.com/docs/reportlab-userguide.pdf>

5. MySQL Documentation:

- Oracle Corporation. (n.d.). MySQL Documentation. Retrieved from <https://dev.mysql.com/doc/>

6. MySQL Connector/Python Documentation:

- MySQL. (n.d.). MySQL Connector/Python Developer Guide. Retrieved from <https://dev.mysql.com/doc/connector-python/en/>

7. General Certificate Design Principles:

- Heumann, B. (2021). Certificate Design: A Guide to Creating Effective Certificates. Retrieved from <https://www.certificatedesign.com>

8. Software Development Best Practices:

- Sommerville, I. (2016). Software Engineering (10th ed.). Boston: Addison-Wesley.