CerGen

An E-Certificate Generator and Sender

A

Project Report

Submitted to

MATS UNIVERSITY

AARANG, RAIPUR (C.G.), INDIA

in partial fulfillment for the award of the Degree

of

Bachelor of Technology

in

Computer Science & Engineering

by

Siddhant Yogesh Totade
MU19BTCSE006

Under the Guidance of

Ms. Poonam Gupta

Department of Computer Science & Engineering
School of Engineering & IT
MATS University, Aarang, Raipur (C.G.), India

Session : Jan - Jun 2023

DECLARATION BY THE CANDIDATE

I am writing this declaration to confirm that I, **Siddhant Yogesh Totade**, am the sole author of the major project titled **CerGen**, submitted as a part of my academic program **Major Project** at **MATS University**. I confirm that I have carried out this project under the supervision of **Ms. Poonam Gupta**, and I am grateful for your guidance and support throughout the process.

I declare that the work presented in this project is my original work, and all sources of information have been duly acknowledged. I have used the citation and referencing styles as prescribed by the university guidelines, and I have not engaged in any form of plagiarism or academic misconduct. Furthermore, I confirm that this project has not been submitted for assessment previously in any other program or institution.

I also acknowledge that the final decision regarding the acceptance of this project lies with the university, and I understand that any false statements made in this declaration may lead to disciplinary action.

Once again, I would like to express my gratitude for your guidance and support, which has been invaluable throughout the project. I have thoroughly enjoyed working on this project and look forward to implementing the knowledge and skills acquired in my future endeavors.

Sincerely,

(Signature of the Candidate)

Siddhant Yogesh Totade
MU19BTCSE006
School of Engineering & IT,
MATS University Aarang, Raipur (C.G.)

CERTIFICATE BY THE SUPERVISOR

This is to certify that **Siddhant Yogesh Totade,** bearing **ID : MU19BTCSE006** & **Enrollment No. : 191309** has successfully completed the major project under my supervision. The project titled **CerGen** was completed during the academic year **2023** in partial fulfillment of the requirements for the **Bachelor of Technology.**

Throughout the project, **Siddhant Yogesh Totade** demonstrated a high level of dedication, professionalism, and expertise. Their commitment to the project was evident in the quality of work produced, which was of a very high standard. They consistently demonstrated an ability to work independently while seeking guidance and feedback when necessary.

Based on my observations and evaluation, I am confident that **Siddhant Yogesh Totade** has the knowledge, skills, and abilities required to excel in their future academic and professional endeavors. I highly recommend him for any future academic or professional endeavors.

Signed,

(Signature of the Supervisor)	**(Signature of the HOD)**
Ms. Poonam Gupta	**Dr. Meesala Sudhir Kumar**
Assistant Professor, Dept. of CSE,	**H.O.D, Dept. of CSE,**
School of Engineering and IT,	**School of Engineering and IT,**
MATS University, Aarang, Raipur	**MATS University, Aarang, Raipur**

Forwarded to MATS University, Raipur

(Signature of Principal)

Dr. Abhishek Kumar Jain
School of Engineering & IT,
MATS University, Aarang, Raipur

CERTIFICATE BY THE EXAMINERS

This is to certify that **Siddhant Yogesh Totade** has successfully completed the major project titled **CerGen** under our examination. The project was completed during the academic year **2023** in partial fulfillment of the requirements for the **Bachelor of Technology.**

We, the undersigned examiners, have evaluated the project report and the project presentation and are pleased to confirm that **Siddhant Yogesh Totade** has demonstrated an exceptional understanding of the project topic. Their project report was comprehensive, well-structured, and exhibited a high degree of originality and creativity. The project also included a well-executed implementation plan that demonstrated a practical application of the project's objectives.

Throughout the project, **Siddhant Yogesh Totade** exhibited excellent research skills and a deep understanding of the theoretical and practical aspects of the project topic. They demonstrated a high level of proficiency in the use of research methodologies and data analysis techniques. The project presentation was clear, concise, and well-structured, and the demonstration of the project implementation was executed to a high standard.

Based on our observations and evaluation, we are confident that **Siddhant Yogesh Totade** has the knowledge, skills, and abilities required to excel in their future academic and professional endeavors. We highly recommend him for any future academic or professional endeavors.

Signed,

____________________	____________________
(Internal Examiner)	**(External Examiner)**
Date : ____________	**Date : ____________**

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to everyone who has supported me throughout the completion of my major project titled **CerGen**, which was undertaken during the academic year **2023** in partial fulfillment of the requirements for the **Bachelor of Technology** in the **Computer Science & Engineering Department**.

First and foremost, I would like to thank my project supervisor **Ms. Poonam Gupta** for their invaluable guidance, support, and encouragement throughout the project. Their knowledge, expertise, and insights were instrumental in shaping the direction and outcome of the project. I am grateful for their patience, time, and dedication in providing me with constructive feedback and suggestions.

I would also like to express my appreciation to the faculty members of the **Computer Science & Engineering** department who have imparted their knowledge, skills, and experience to me throughout my academic journey. Their guidance, mentorship, and encouragement have played a crucial role in shaping my academic and professional development.

My heartfelt gratitude goes out to my family, friends, and colleagues who have provided me with their unwavering support, encouragement, and motivation throughout this journey. Their love, understanding, and support have been a constant source of strength and inspiration.

Last but not least, I would like to acknowledge the contribution of all the individuals who have directly or indirectly helped me in completing this project. Your support and assistance were invaluable and greatly appreciated.

Thank you all for your support, guidance, and encouragement.

Siddhant Yogesh Totade
MU19BTCSE006

ABSTRACT

This project aims to design and develop an e-certificate generator system using a combination of React, Django, Material UI, Tailwind CSS, and Python. The primary objective of the system is to automate the process of generating e-certificates for various events, workshops, and training programs.

The project is divided into two main components: the frontend and the backend. The frontend component is built using React, Material UI, and Tailwind CSS. React is used as the primary framework for developing the user interface of the system. Material UI provides pre-built UI components that make the development process faster and more efficient. Tailwind CSS is used to style the components and provides a robust set of CSS utility classes.

The backend component of the system is built using Django and Python. Django is used as the primary framework for developing the backend of the system. It provides a secure and scalable environment for developing web applications. Python is used as the primary programming language for developing the backend logic of the system. It provides a wide range of libraries and frameworks that make the development process faster and more efficient.

The system allows the administrator to create various templates for e-certificates with dynamic placeholders that can be populated with the participant's details. Once the administrator creates the templates, the system allows them to upload a list of participants' details. The system then automatically generates e-certificates for each participant and sends them via email.

The system is designed to be highly customizable and configurable. The administrator can customize the templates, add new placeholders, and modify the system's behavior to meet their specific requirements.

In conclusion, the e-certificate generator system developed using React, Django, Material UI, Tailwind CSS, and Python is an efficient and effective solution for automating the process of generating e-certificates. The system is highly customizable, scalable, and easy to use, making it a valuable asset for any organization that conducts events, workshops, and training programs.

Keywords - *React, Django, Material UI, Tailwind CSS, Backend, Frontend, E-Certificate*

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

● HTML	Hyper Text Markup Language
● CSS	Cascading Style Sheets
● JS	JavaScript
● PHP	Hypertext Preprocessor
● JSX	JavaScript XML
● XML	Extensible Markup Language
● API	Application Programming Interface
● ERD	Entity Relationship Diagram
● DFD	Data Flow Diagram
● E - Certificate	Electronic Certificate
● CRUD	Create, Read, Update, Delete
● QA	Quality Assurance
● UI	User Interface
● CSRF	Cross Site Request Forgery
● CORS	Cross Origin Resource Sharing
● XSS	Cross Site Scripting
● ORM	Object Relational Mapper
● SQL	Structured Query Language
● URL	Uniform Resource Locator

CHAPTER-1

INTRODUCTION

INTRODUCTION

1.1 Introduction

1.1.1 Introduction of App

Introducing my app that revolutionizes the process of generating e-certificates! My app is designed to streamline the certificate creation process through automation, saving you time and effort. With our app, you can quickly and easily generate e-certificates for your events, courses, and workshops with just a few clicks.

The purpose of our app is to simplify the process of creating e-certificates, which can be a time-consuming and tedious task. With our automation technology, you can eliminate the need for manual data entry and formatting, reducing errors and increasing efficiency. Our app is perfect for event organizers, trainers, educators, and anyone else who needs to generate certificates on a regular basis.

My app is user-friendly and easy to navigate, making it accessible to anyone, regardless of their technical expertise. Plus, with customizable templates and branding options, you can personalize your certificates to match your organization's style and branding.

Save time and increase efficiency with this e-certificate generation app.

1.1.2 Introduction of Certificate

A certificate is an official document that recognizes and validates a person's accomplishments or achievements in a particular field or domain. Certificates are often awarded after completing a course or training program, and they serve as proof of the individual's knowledge and skills in that area. Certificates can be of various types and have different benefits, depending on the purpose and nature of the certificate.

A certificate can help you stand out in a competitive job market, and it can demonstrate your expertise in a particular field. This can lead to better job opportunities and higher salaries.

Earning a certificate can be a great way to enhance your knowledge and skills in a particular subject area. It can also provide a sense of personal accomplishment and satisfaction.

1.1.3 Introduction of E-Certificate

An e-certificate is a digital version of a traditional paper certificate that is generated and stored electronically. The use of e-certificates is becoming increasingly popular, especially in today's digital age. E-certificates offer many benefits over traditional hard copy certificates, making them a popular choice for many organizations and individuals.
E-certificates can be easily accessed and shared online, making them convenient for recipients to receive and store. They can be accessed from anywhere at any time, eliminating the need for physical storage and transport.

E-certificates are highly secure, as they are generated and stored electronically, making it difficult for them to be tampered with or duplicated. This helps to prevent fraud and ensures the authenticity of the certificate.

1.2 Introduction of Web

The World Wide Web (WWW) has transformed the way we communicate, access information, and conduct business. It is an interconnected system of documents and resources that are accessed through the Internet using a web browser. The web has evolved significantly since its inception in the 1990s, and it continues to be a rapidly developing field with new technologies, frameworks, and applications emerging all the time. As such, the web is a vital component of the digital economy, enabling individuals and organizations to create, share, and consume content and services on a global scale. In this major project, we will explore various aspects of web development, including web design, client-side scripting, server-side scripting, and database integration, among others. We will also examine the role of web standards, accessibility, and usability in building effective and inclusive web experiences. Through our exploration of web development, we aim to gain a deep understanding of the technologies, tools, and practices that underpin the modern web and equip ourselves with the skills and knowledge needed to create high-quality web applications.

1.3 Introduction of Web Technologies

Web technologies refer to the various software tools, programming languages, and frameworks that are used to create and maintain web applications. These technologies include but are not limited to HTML, CSS, JavaScript, server-side scripting languages like PHP, Python, and Ruby, and databases like MySQL and MongoDB.
Web technologies have evolved significantly since the early days of the web, and they continue to evolve at a rapid pace. Today, modern web applications are highly interactive and

dynamic, providing rich user experiences that were once only possible in desktop applications.

1.4 Problem Definition

A normal paper certificate can have several drawbacks when compared to an e-certificate:

- Physical copies can be lost or damaged: Physical certificates can be misplaced, lost, or damaged due to various reasons, such as fires, floods, or accidents. This can lead to inconvenience and additional costs for the certificate holder to obtain a replacement copy.

- Difficult to verify: Physical certificates can be difficult to verify, especially if the certificate holder is in a different location than the organization issuing the certificate. This can result in delays and additional costs for the certificate holder, as well as potential fraud issues.

- Time-consuming and expensive: Issuing and delivering physical certificates can be a time-consuming and expensive process, involving printing, mailing, and handling fees. This can lead to delays in receiving the certificate and additional costs for the certificate holder.

- Limited access: Physical certificates are only accessible to the person holding the physical copy, making it difficult to share and verify the information with others.

1.5 Objectives of Project

The main objectives of an e-certificate generator are:

- Efficiency: The primary objective of an e-certificate generator is to create a streamlined and efficient process for generating certificates. This can save time and resources for both the issuer and the certificate holder.

- Accuracy: Another important objective of an e-certificate generator is to ensure the accuracy and reliability of the certificates being generated. This includes verifying the identity of the certificate holder, the authenticity of the information being presented, and the compliance with any relevant regulations or standards.

- Security: E-certificate generators must also prioritize the security of the certificates being generated, protecting against any potential fraud or tampering. This includes implementing secure authentication methods and using encryption technologies to protect the data.

- Accessibility: E-certificate generators should also strive to make the certificates accessible to a wide range of users, including those with disabilities or limited technical abilities. This can include providing multiple formats or interfaces for accessing the certificates.

- Customization: Finally, e-certificate generators should provide options for customization, such as allowing the issuer to add their logo or branding to the certificate, or allowing the certificate holder to customize certain fields or details. This can help make the certificates more meaningful and valuable to the recipients.

1.6 Scope of Project

The scope of an e-certificate generator is wide-ranging and can include the following:

- Education and Training: E-certificate generators can be used by educational institutions, training centers, and online learning platforms to issue certificates of completion, diplomas, and other educational credentials.

- Professional Development: E-certificate generators can also be used by professional associations and organizations to issue certifications and other professional development credentials to their members.

- Corporate Training: E-certificate generators can be used by corporations to issue certificates of completion for employee training and development programs.

- Event Management: E-certificate generators can be used by event organizers to issue certificates of participation, attendance, and achievement to attendees and speakers.

- Government Agencies: E-certificate generators can be used by government agencies to issue licenses, permits, and other official documents to individuals and businesses.

- Healthcare: E-certificate generators can be used by healthcare organizations to issue certificates of completion and other credentials to medical professionals who have completed continuing education courses or other training programs.

- Non-Profit Organizations: E-certificate generators can be used by non-profit organizations to issue certificates of appreciation, recognition, and achievement to volunteers, donors, and supporters.

1.7 Expected Outcomes

The expected outcomes of an e-certificate generator are:

- Increased Efficiency: E-certificate generators can significantly reduce the time and effort required to issue certificates. This can lead to increased productivity and cost savings for organizations.

- Improved Accuracy: E-certificate generators can reduce errors and inaccuracies associated with manual certificate issuance. This can improve the reliability and credibility of the certificates being issued.

- Enhanced Security: E-certificate generators can provide greater security and protection against fraud, tampering, and counterfeiting. This can increase the trust and confidence in the certificates being issued.

- Improved Accessibility: E-certificate generators can make certificates more accessible to a wider range of users. This can help organizations to reach a larger audience and provide greater value to certificate holders.

- Customization: E-certificate generators can provide greater customization options, allowing organizations to tailor certificates to their specific needs and branding.

- Better Tracking: E-certificate generators can enable organizations to track the issuance and usage of certificates more effectively. This can provide valuable insights into the effectiveness of training and development programs, as well as help organizations to monitor compliance with relevant regulations and standards.

1.8 Organization of Project Report

1.8.1 Introduction

CerGen - A Certificate Generator and Sender is a web application which serves the certificates with the help of automation. This application is capable of generating the certificates of participation as well as merit for the participants. CerGen has the functionality in which you can choose your own templates and generate the certificates.

Creating a certificate manually is a time taking process and the paper or the hard copies of the certificates can damage easily. That's where the e-certificates came which takes less time to generate and safer and easier to send.

CerGen takes the data from the participants via google form and generates and sends certificates to the participants who are eligible for it.

CerGen uses django as backend with react as frontend. For designing CerGen uses material ui with tailwind css.

1.8.2 Literature Review

The e-certificate generator is designed in python which is capable enough to generate certificates within minutes. The API's of this project is built on django which is a framework of python.

This type of generator is also available in google slides but the slides can only generate one certificate type at once. This application generates a lot of certificates of participation as well as a certificate of merit at the same time.

CHAPTER - 2

FEASIBILITY STUDY AND PROPOSED WORK

FEASIBILITY STUDY AND PROPOSED WORK

2.1 Software Engineering Model

2.1.1 Requirements Gathering

First, gather the requirements for the e-certificate generator. Determine what type of certificates will be generated and what information will need to be included on the certificates. Also, consider the expected number of users, how they will access the system, and how the certificates will be distributed.

Second, gather the requirements for the application. Determine what types of technologies are required to build such applications.

2.1.2 Design

Next, design the system architecture and user interface. Decide on the programming language and tools to be used, and create a detailed plan for implementation.

2.1.3 Implementation

Develop the software according to the design plan. This may involve creating a database to store user information, setting up an interface for users to input data, and creating algorithms to generate certificates.

2.1.4 Testing

Conduct thorough testing of the software to ensure that it works as intended. This may include unit testing, integration testing, and system testing. Identify and fix any bugs or issues.

2.1.5 Deployment

Deploy the e-certificate generator to the production environment. This may involve setting up servers and configuring the software for optimal performance.

2.1.6 Maintenance

Monitor the system for issues and address any bugs or errors that arise. Provide ongoing support and updates to ensure that the system remains functional and secure.

2.2 Cost Estimation

2.2.1 Development Cost

2.2.1.1 E-Certificate

The initial cost of developing an e-certificate system can vary widely depending on the complexity of the software, the number of developers involved, and the programming languages and tools used. The development cost can range from a few thousand rupee to tens of thousands of rupee.

2.2.1.2 Plain Certificate

The cost of designing and printing a paper certificate can vary depending on the design complexity, quality of the paper used, and printing method. This cost can range from a few rupee to several rupee per certificate.

A normal certificate paper cost starts from ₹15 and up to ₹45 depending upon the quality of paper. Lesser the price, lesser the quality.

Also, the cost of printing is involved which includes the ink of printing and printer itself.

2.2.2 Infrastructure and Mailing Cost

2.2.2.1 E-Certificate

E-certificates are generated and distributed through an online system, which requires server infrastructure, bandwidth, and storage. The infrastructure cost will depend on the number of certificates generated, the amount of traffic on the system, and the storage requirements. The infrastructure cost can range from a few hundred rupee to several thousand rupee per year.

2.2.2.2 Paper Certificate

Once the paper certificates are printed, they need to be mailed or shipped to the recipients. This cost includes the postage fees and any additional costs associated with packaging and handling. This cost can range from a few cents to several dollars per certificate.

2.2.3 Distribution and Storage Cost

2.2.3.1 E-Certificate

Since e-certificates are delivered digitally, there are no costs associated with printing, mailing, or shipping. The cost of distribution is minimal, as it requires only an email or download link.

2.2.3.2 Paper Certificate

Paper certificates need to be stored in a safe place, which can be costly if the certificates need to be stored for a long time. Storage costs can include the cost of storage boxes, filing cabinets, or even off-site storage facilities.

2.3 Past Related Work

2.3.1 Academic Research

E-certificates are electronic certificates that are generated and distributed digitally, typically via email or download links. They are becoming increasingly popular in major projects as a more efficient and cost-effective alternative to traditional paper certificates. My findings suggest that e-certificates offer several advantages over paper certificates, including reduced costs, increased speed and convenience, and enhanced security. However, there are also several challenges associated with e-certificates, including concerns about data privacy and security, and the need to ensure that the e-certificate system is accessible and user-friendly for all stakeholders. We conclude by discussing the implications of our findings for the design and implementation of e-certificate systems in major projects.

In conferences, seminars, training programs, and workshops, certificates are often awarded to participants to acknowledge their participation and achievements. Traditionally, these certificates have been printed on paper and distributed by mail or in person. However, with the increasing availability and affordability of digital technology, e-certificates are becoming a popular alternative to paper certificates. E-certificates are electronic certificates that are generated and distributed digitally, typically via email or download links. This paper explores the advantages and challenges of e-certificates in major projects, based on a review of existing literature and case studies of e-certificate implementations.

2.3.1.1 Advantages

E-certificates offer several advantages over traditional paper certificates. First, they are more cost-effective, as there are no costs associated with printing, mailing, or shipping. Second, they are faster and more convenient, as they can be generated and distributed instantly. Third, they offer enhanced security, as they can be encrypted and protected from tampering. Fourth, they are more environmentally friendly, as they reduce paper waste and carbon emissions.

2.3.1.2 Challenges

To address the challenges associated with e-certificates, it is important to design and implement e-certificate systems that prioritize security, accessibility, and user-friendliness. This may involve using encryption and other security measures to protect personal data, providing multiple options for accessing and downloading e-certificates, and offering user support and training to ensure that stakeholders can use the e-certificate system effectively. Additionally, it may be helpful to provide incentives for stakeholders to use e-certificates, such as offering discounts or other benefits to participants who opt for e-certificates over paper certificates.

2.3.1.3 Conclusion

E-certificates are a valuable alternative to traditional paper certificates, but implementing an e-certificate system can be challenging. Our web-based e-certificate generator provides an efficient and secure solution for generating e-certificates. The system is scalable, customizable, and user-friendly, making it suitable for use in a variety of academic and professional settings.

2.3.2 Development Research

The system was developed using modern web technologies, including HTML, CSS, JavaScript, Django, React, Material UI, Tailwind Css and was designed to be scalable, secure, and user-friendly. The e-certificate generator is built on python and the API's were in django. The database used by this application is dbSqlite which is the default database of django but it can be modifiable to any other sql database like MySQL, PostgreSql etc.

2.3.2.1 About Technologies

Django - Django is a free and open-source web framework for building web applications using the Python programming language. It is designed to help developers build high-quality web applications quickly and efficiently, by providing a robust set of tools and features out of the box.

React JS - React is a popular open-source JavaScript library used for building user interfaces (UIs). It was developed by Facebook and is now maintained by both Facebook and a community of developers. React is known for its ability to create complex UIs using a component-based architecture.

Material UI - Material UI is a popular open-source React component library that provides pre-designed UI components based on Google's Material Design system. Material Design is a design language developed by Google that emphasizes a clean and modern look, and Material UI makes it easy for developers to implement this design system in their React applications.

Tailwind CSS - Tailwind CSS is a popular open-source CSS framework that provides a utility-first approach to building user interfaces. It is designed to make it easy for developers to quickly create responsive and customizable UIs using pre-built CSS classes.

2.3.2.2 - Why these Technologies

Django

Some of the key features of Django include:

- Object-Relational Mapping (ORM): Django provides an ORM layer that allows developers to interact with databases using Python objects, making it easy to work with data without needing to write SQL queries.

- URL routing: Django provides a powerful URL routing system that allows developers to map URLs to views, making it easy to create clean and understandable URLs for their web applications.

- Template engine: Django's template engine allows developers to create dynamic and reusable templates for their web applications, making it easy to build complex user interfaces.

- Built-in security features: Django includes built-in security features such as cross-site scripting (XSS) protection, cross-site request forgery (CSRF) protection, and user authentication and authorization.

- Administration interface: Django provides a built-in administration interface that allows developers to manage the data in their applications without needing to write any code.

- Scalability: Django is designed to be scalable and can handle large amounts of traffic and data.

React JS

React works by breaking down a UI into smaller, reusable components. Each component is responsible for rendering a specific part of the UI, and can be combined with other components to create a complete UI. This approach makes it easy to manage large and complex UIs, as developers can work on each component separately and then combine them to create the final UI.

Some of the key features of React include:

- One of the key features of React is its use of a virtual DOM (Document Object Model). The virtual DOM is a lightweight representation of the actual DOM, and React uses it to optimize updates to the UI. When a component's state changes, React updates the virtual DOM instead of the actual DOM, and then compares the two to determine the minimum number of changes needed to update the actual DOM. This makes React's updates to the UI fast and efficient.

- Another key feature of React is its use of JSX, a syntax extension that allows developers to write HTML-like code within JavaScript. This makes it easy to write and manage UI code, as everything is contained within a single file. JSX is then transpiled into regular JavaScript for use by the browser.

- React has a large and active community, and there are many third-party libraries and tools available for use with React. This makes it easy to extend and customize React to fit the needs of a specific project.

Material UI

Material UI provides a wide range of customizable UI components, such as buttons, form inputs, cards, and navigation elements. These components are designed to be easy to use and customize, with many options for changing colors, sizes, and other properties. Material UI also provides a theming system that allows developers to easily customize the look and feel of their applications.

Some of the key features of React include:

- One of the key benefits of Material UI is that it simplifies the design process for developers. Instead of spending time designing and coding UI components from scratch, developers can use Material UI's pre-designed components to quickly build high-quality UIs. This can save time and effort, while also ensuring consistency and adherence to the Material Design system.

- Material UI is also well-documented, with extensive API documentation and examples available on its website. Additionally, the library has a large and active community, with many third-party plugins and extensions available to enhance its functionality.

Tailwind CSS

The utility-first approach of Tailwind CSS means that instead of defining custom CSS styles for each element, developers can apply pre-built CSS classes directly to the HTML elements. These classes provide a wide range of styling options, such as margin, padding, typography, and colors. This makes it easy to create complex layouts and styles quickly and efficiently.

Some of the key features of Tailwind Css include:

- Tailwind CSS also provides a highly customizable and extendable design system, allowing developers to define their own custom styles and utilities. This makes it easy to create a unique look and feel for each project, while also ensuring consistency and maintainability.

- Another key benefit of Tailwind CSS is its focus on responsive design. It provides a wide range of responsive classes that allow developers to create layouts that adapt to different screen sizes and devices. This can save time and effort compared to writing custom media queries and CSS for each breakpoint.

- Tailwind CSS is also highly optimized for performance, with a small file size and optimized styles that reduce the need for additional CSS code. This can help to improve the load times and overall performance of web applications.

2.3.3 Market Survey

The purpose of this survey is to gather feedback from potential users of an e-certificate generator, which is being developed as a major project. We would like to understand the needs and preferences of individuals and organizations who require the generation and distribution of digital certificates.

Questions have been asked to the people:

1. Have you ever participated in any contest before ?

    a. Yes
    b. No

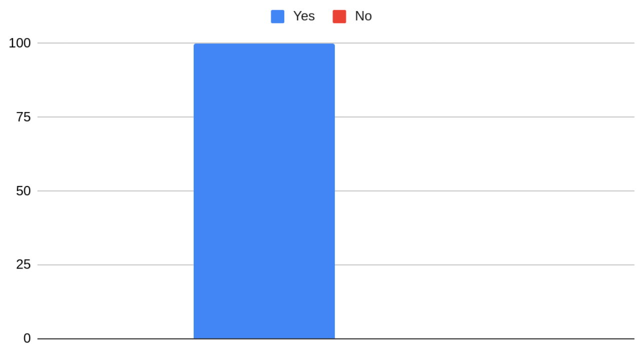


Fig. 2.3.2 : MS - 1

2. Has the organizer provided you with a certificate for participating in the contest ?

    a. Yes
    b. No

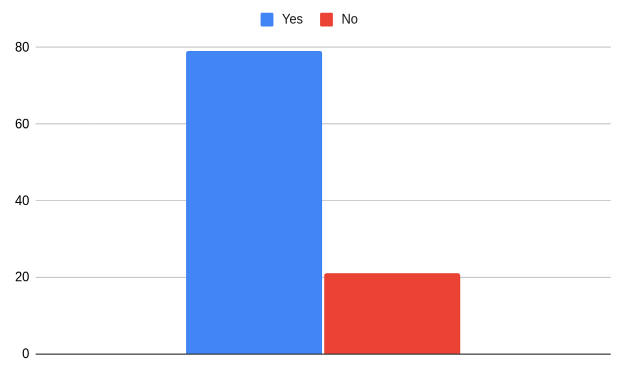


Fig. 2.3.2 : MS - 2

3. What is the type of certificate you have received ?

    a. Soft Copy / E - Certificate / Digital Certificate
    b. Hard Copy / Physical Copy

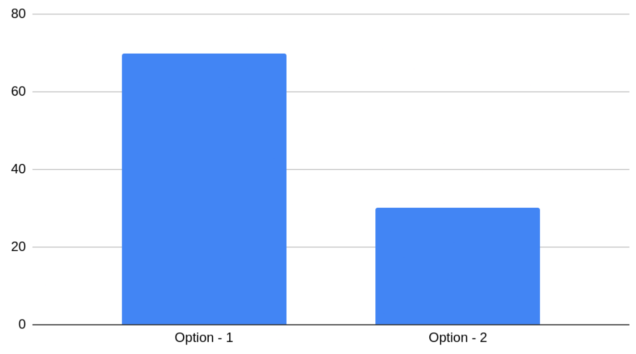


Fig. - 2.3.2 : MS - 3

4. Do you think that the certificate you have received is manually generated or automatically generated ?

    a. Automatically
    b. Manually

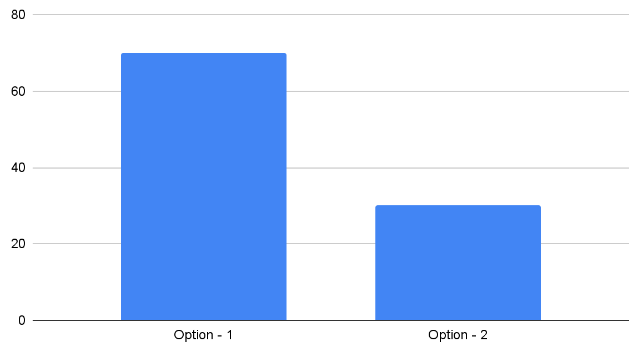


Fig. 2.3.2 : MS - 4

5. If you have got the digital certificate then how was it provided to you ?

   a. Via link which contains only your certificate
   b. Via link which contains all of the participants certificate
   c. Via email which contains only your certificate with or without option - 1
   d. Via email which contains all of the participants certificate with or without option - 2
   e. I received a hard copy

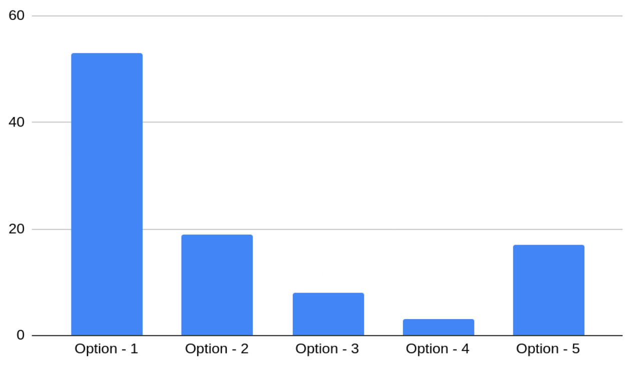


Fig. 2.3.2 : MS - 5

6. What is the format of the digital certificate you have got ?

   a. PDF - Portable Document Format
   b. Image - JPG / JPEG
   c. I received a hard copy

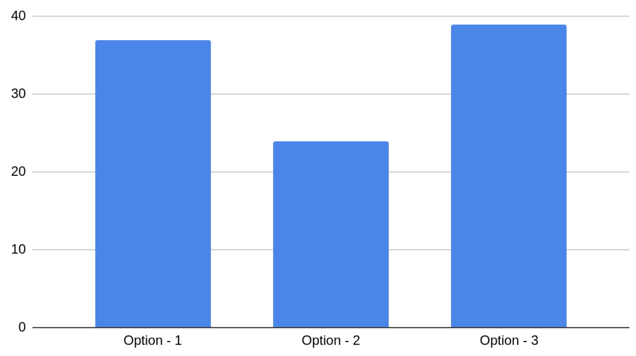


Fig. 2.3.2 : MS - 6

7. In the hard copy certificate you have received, the fields like Name, Event Name, Position etc. are handwritten or generated with the help of any software

    a. Looks like handwritten
    b. Looks like it is generated with the help of software
    c. I received a hard copy

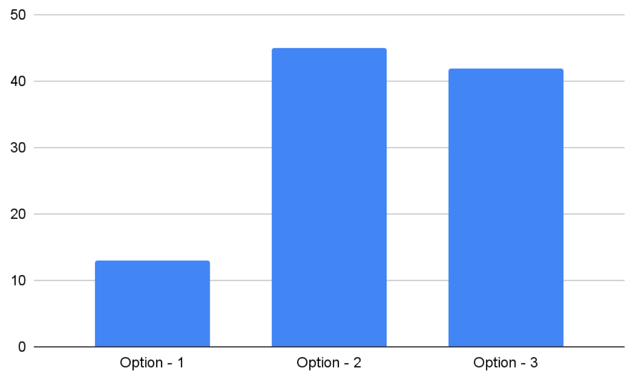


Fig. 2.3.2 : MS - 7

8. According to you which method is more efficient.

    a. Generating hard copy certificates for participants
    b. Generating soft copy certificate for participants

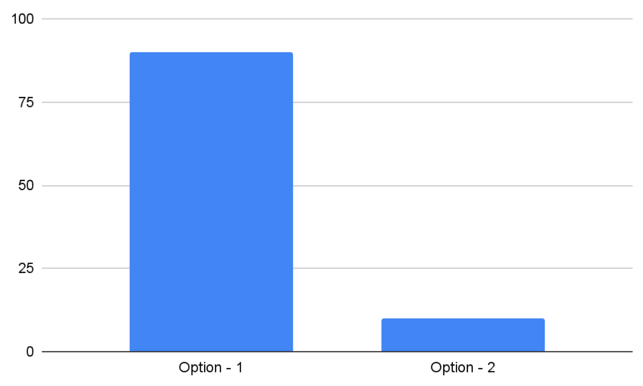


Fig. 2.3.2 : MS - 8

9. Have you ever tried an application which automatically generates the certificates and sends them to the participants' email automatically ?

   a. Yes
   b. No

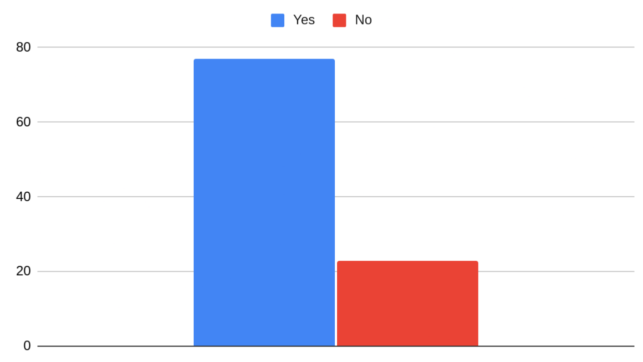


Fig. 2.3.2 : MS - 9

10. Name of the application that you tried or used for certificate generation and sending ? If you don't just write 'No'.

    a. ______________________________

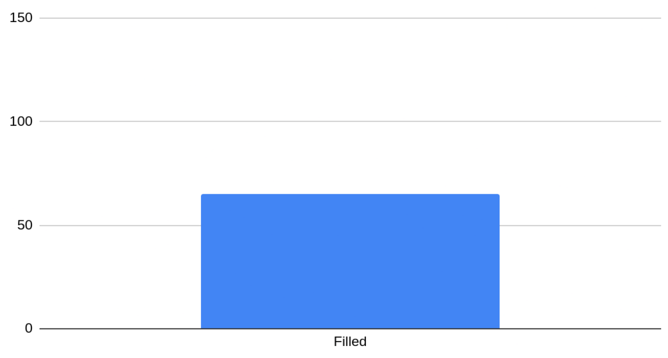


Fig. 2.3.2 : MS - 10

11. Pause for a while and read the below description carefully.

    Python is a programming language used in the field of application designing, data science, machine learning, game modeling etc. Python provides you automation and efficiency which reduces your daily tasks.

    Have you ever heard of or work with python ?

    a. Yes
    b. No
    c. Maybe

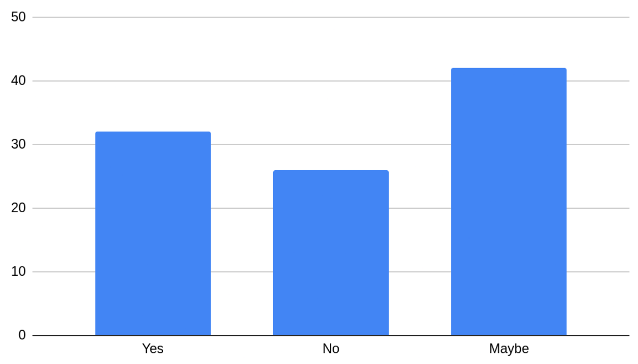


Fig. 2.3.2 : MS - 11

12. Pause for a while and read the below description carefully.

    CerGen is a web application which generates the certificates for events / contest's and sends them to every participant which has participated in the contest automatically. The CerGen application is built using Django and OpenCV which are the python technologies.

a. I understand the above description
b. No, I don't

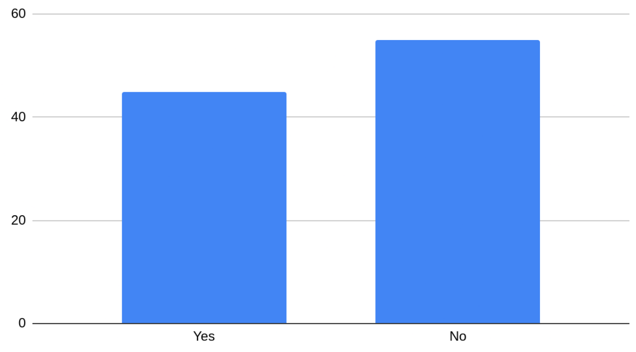


Fig. 2.3.2 : MS - 12

2.4 Proposed Work

The proposed project aims to develop an e-certificate generator that can be used to create and distribute certificates online. The e-certificate generator will be a web-based application that will automate the process of certificate generation, making it easier and more efficient for organizations to issue certificates to their employees, participants, and stakeholders.

The objectives of this project are as follows:

- To develop a user-friendly web-based e-certificate generator that can be used by organizations to create and distribute certificates online.

- To automate the certificate generation process to save time and increase efficiency.

- To enable organizations to customize their certificate templates with their logo, branding, and other details.

- To ensure the security and confidentiality of certificate data.

The proposed e-certificate generator will be a valuable tool for organizations to automate their certificate generation process. The application will provide a user-friendly interface, customized certificate templates, and secure handling of certificate data. By developing this application, we hope to increase efficiency, reduce manual efforts, and provide a better user experience to our clients

CHAPTER-3

METHODOLOGIES

METHODOLOGIES

3.1 Frontend and Backend

Frontend and backend are two essential components of a software application. The frontend is the visible part of the application that the user interacts with. It includes the user interface, which is responsible for displaying the content, graphics, and other visual elements of the application. The frontend is built using technologies such as HTML, CSS, and JavaScript, which enable developers to create dynamic and responsive user interfaces that can run on various devices and browsers.

On the other hand, the backend is the part of the application that operates behind the scenes and is responsible for the business logic and data processing. The backend includes the server-side programming, database management, and APIs that enable the frontend to interact with the data and perform operations on it. The backend is usually built using programming languages such as Python, Java, or PHP, and frameworks such as Django or Flask, which provide a structure for building scalable and robust applications.

CerGen is a powerful tool built on the Django backend that enables organizations to automate the process of generating and distributing certificates online. This web application is designed to help organizations save time and resources while providing a better user experience to their employees, participants, and stakeholders.

CerGen is built using React as the frontend framework. React is a popular JavaScript library for building user interfaces. Material UI and Tailwind CSS are used for UI components and styling. Material UI provides a rich set of pre-built components while Tailwind CSS provides a utility-first CSS framework for rapid UI development.

The E-Certificate Generator Web Application has the following features:

- User-Friendly Interface: The application has a user-friendly interface that is easy to navigate, enabling users to create and customize their certificates quickly.

- Customizable Certificate Templates: The application enables organizations to customize their certificate templates with their branding, logo, and other details. This feature allows organizations to create certificates that are tailored to their specific needs.

- Automated Certificate Generation: The application automates the certificate generation process, saving time and reducing manual effort. Once the certificates are created, they can be downloaded, printed, or emailed directly from the application.

3.2 Algorithms

The E-Certificate Generator Web Application uses an algorithm to automate the process of generating and distributing certificates online. The algorithm follows a set of predefined steps to generate certificates based on the input data provided by the user.

Firstly, the algorithm validates the input data provided by the user, including the certificate template, participant details, and any customizations. Next, it uses this data to populate the fields of the certificate template and generates the certificate. The algorithm then saves the generated certificate to a database and provides the user with options to download, print, or email the certificate directly from the application.

The algorithm also ensures the security and confidentiality of the certificate data by using QR technology. It is designed to handle a large number of certificate requests simultaneously and is optimized for performance and speed.

The E-Certificate Generator Web Application uses a robust and efficient algorithm to automate the process of generating and distributing certificates online.

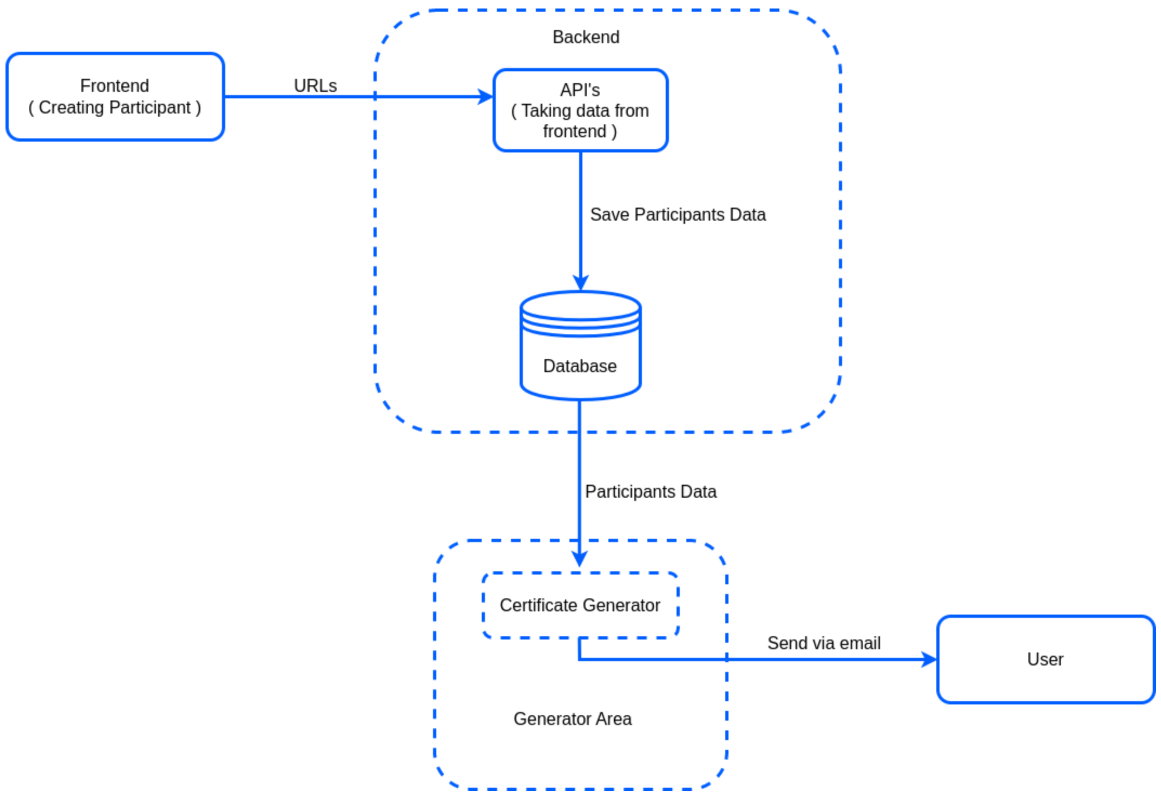


Fig. 3.2 - Basic Flow Diagram

3.2.1 Frontend Data Upload

This application uses a form to save the data of participants from the frontend. This form includes all the necessary fields which populates the data via url to api and that data will be stored in the backend.

The required fields are:

- Event_Id *
- Participant_Name *
- Participant_Id *
- Participant_Email *
- Participant_Phone *
- Participant_Certificate_Status *
- Participant_Certificate_Id *
- Participant_Certificate_Sent_Status

These fields are necessary and to be filled while uploading the participant data to the backend.

3.2.2 Storing Data

After getting all the required data from the frontend the application automation uploads the data to the database and generates a response which says “Participant Created Successfully”.

This data will be reflected in the page and will be able to perform further operations of e-certificate generation.

The generator automatically fetches the data from the database and puts it in the certificate template dynamically. This operation will flow in the loop for many participants.

3.3 Flow Chart

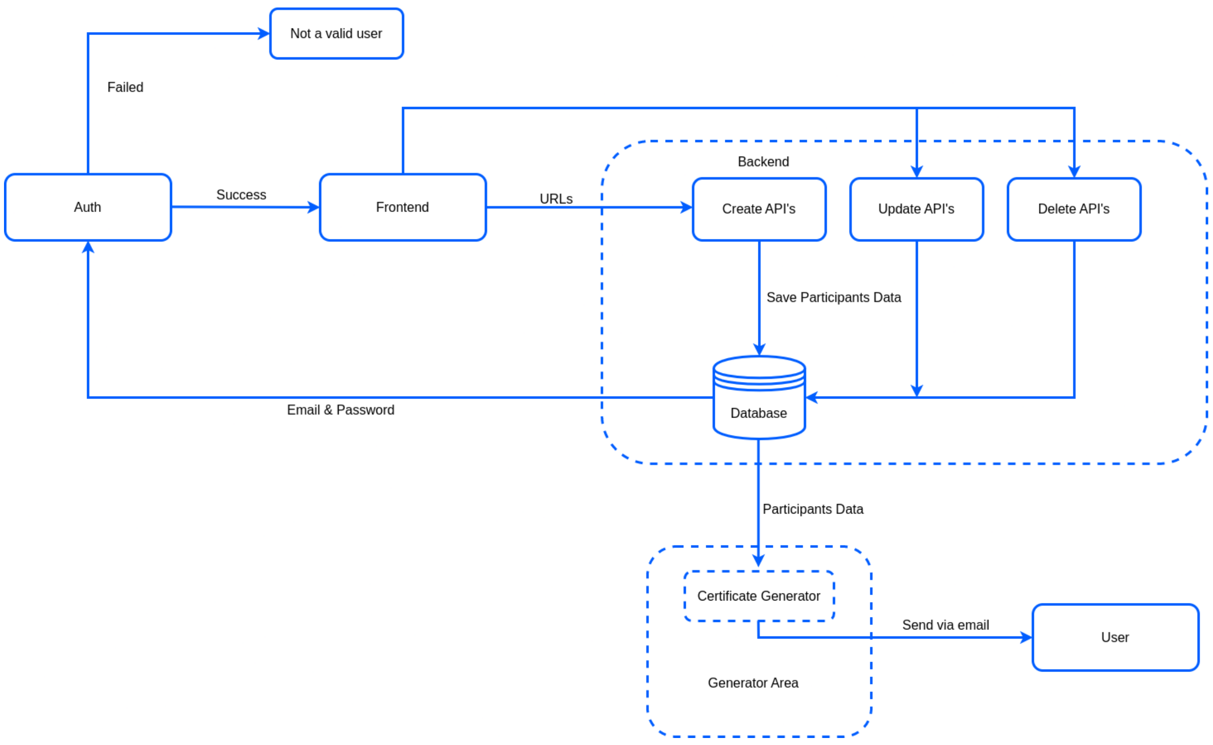


Fig. 3.3 : App Flow Diagram

The Fig. - 3.4 : App Flow Diagram shows the flow of the diagram from the initial stage to the generation of the certificate. This includes all the parts of the application.

The very first part is authentication. If the person is admin then only he/she will be able to login otherwise not.

The next part is the application where the admin can create events as well as upload participants and perform CRUD operations on events and participants.

The third part is the certificate generator which fetches the participants data from the database and generates the certificates.

3.4 Data Flow Diagram

A data flow diagram (DFD) is a visual representation of how data flows through a system. It is a useful tool for understanding the functions and processes of a system and can be used in the development of software or information systems.

This is a high-level overview of the entire system, showing the inputs, outputs, and major processes involved. This is a more detailed diagram that breaks down the major processes into smaller, more specific processes. In addition to the processes and inputs/outputs, it is important to identify any data stores that are involved in the system. These can be physical or virtual locations where data is stored.

3.4.1 Authentication DFD

Authentication is required while creating a web app. This application uses admin based authentication which means only admins are able to login and access it.

Admin-based authentication is an essential component of many major projects. This type of authentication ensures that only authorized users with administrative privileges are able to access and manage sensitive information or critical functions of the system. Admin-based authentication typically involves a login process that requires the user to enter a username and password. Once authenticated, the user is granted access to the administrative features of the system, such as managing user accounts, modifying system settings, and accessing sensitive data. Proper implementation of admin-based authentication is crucial to the security and integrity of the system, and can help to prevent unauthorized access or malicious activities. It is important to carefully design and test the admin-based authentication system to ensure that it is robust and secure, and that it meets the needs of the project and its stakeholders.

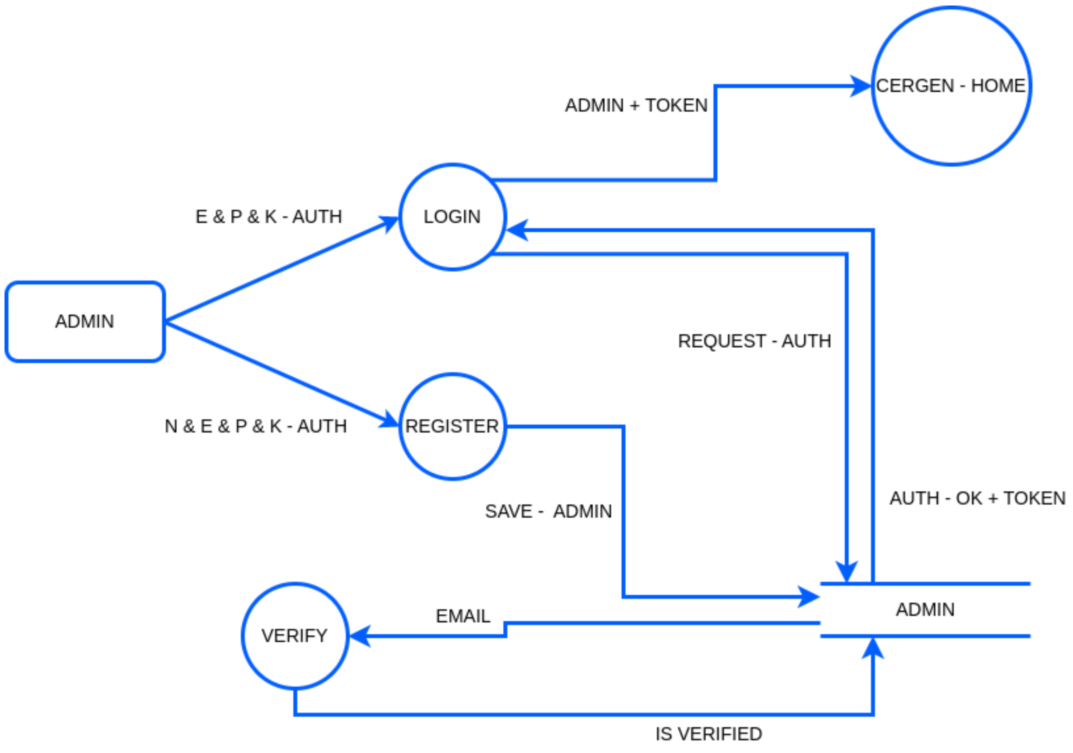


Fig. 3.4.1 : Authentication DFD

The Fig. - 3.4.1 shows the authentication of the application.

At first the admin tries to login via email and password through the admin panel and if the authentication is successful the application redirects the person to the homepage of the application. If the provided information is false then it shows an error which says “Email and Password doesn't exist”.

Next the admin has to register himself/herself by simply filling the required fields which are name, email, password and confirm password. After submitting the form the person received an email of verification which contains the link. After clicking on the link the person is verified and becomes admin.

Now the person is verified and able to authenticate.

3.4.2 Application DFD - 1

An application DFD (Data Flow Diagram) for an e-certificate generator system would show the flow of data between various components of the system.

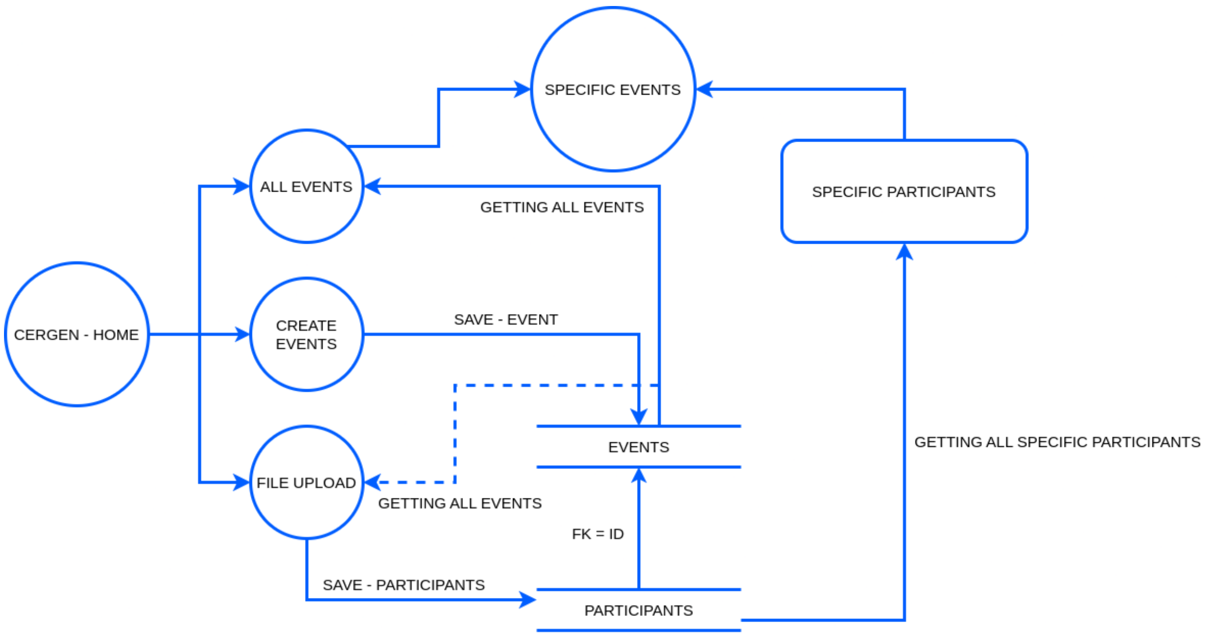


Fig. 3.4.2 : Application DFD - 1

The Fig. 3.4.2 shows about the application flow part - 1.

After logging in, the admin will face the homepage of the application. Here the admin can view events, create events and upload participant files. Through this the admin is able to view the participants of the events.

3.4.3 Application DFD - 2

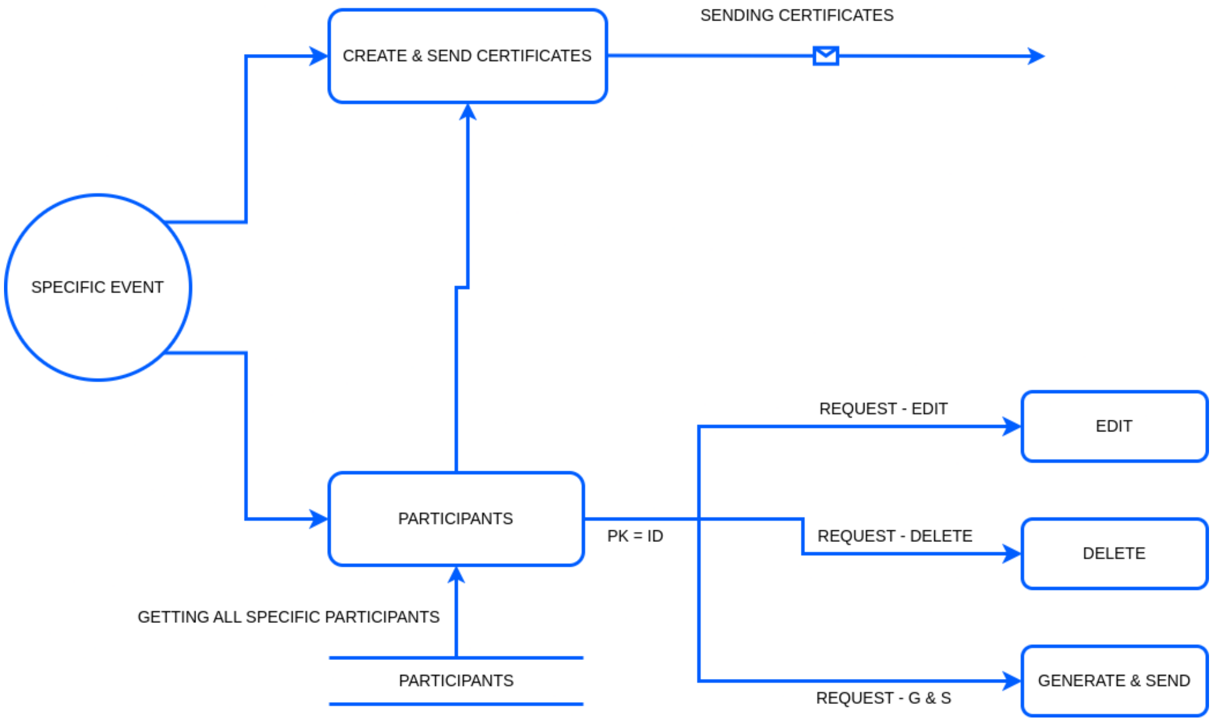


Fig. 3.4.3 : Application DFD - 2

The Fig. - 3.4.3 shows about the application flow part - 2.

This DFD shows the application specific event page which has all the participants as well as many functionalities like creating participants individually, uploading student images and creating event albums. Also if the admin will be able to modify these fields as well as delete these fields.

This page contains the certificate generation for all participants as well as individually. Once you click on the “Issue and Send Certificates” button, it will automatically generate the certificates for all participants and if the admin wants to generate the certificate for an individual participant then the admin has to click on the send certificate icon and the certificate will be sended to the participant automatically.

With the certificate the application sends a text message which says “Thank you for participating in the event/contest. Your e-certificate will be delivered to you via email. Please check your email”.

3.5 Entity Relationship Diagram

An entity-relationship diagram (ERD) is a graphical representation of the relationships between entities (objects or concepts) in a database. The ERD represents the database's structure and can help in designing, organizing, and understanding the relationships between different entities in a system.

Here are some of the components of an ERD:

- Entities : Entities are objects or concepts that are represented in a database. Examples of entities might include customers, products, employees, or orders.

- Attributes : Attributes are the characteristics or properties of an entity. For example, a customer entity might have attributes such as name, address, phone number, and email address.

- Relationships : Relationships describe how entities are related to each other. For example, a customer entity might be related to an order entity through a "placed by" relationship.

- Cardinality : Cardinality describes the number of instances of one entity that can be related to another entity. For example, a customer entity might be able to place many orders, but an order entity can only be placed by one customer.

- Keys : Keys are unique identifiers for each entity instance. For example, a customer entity might have a unique customer ID number.

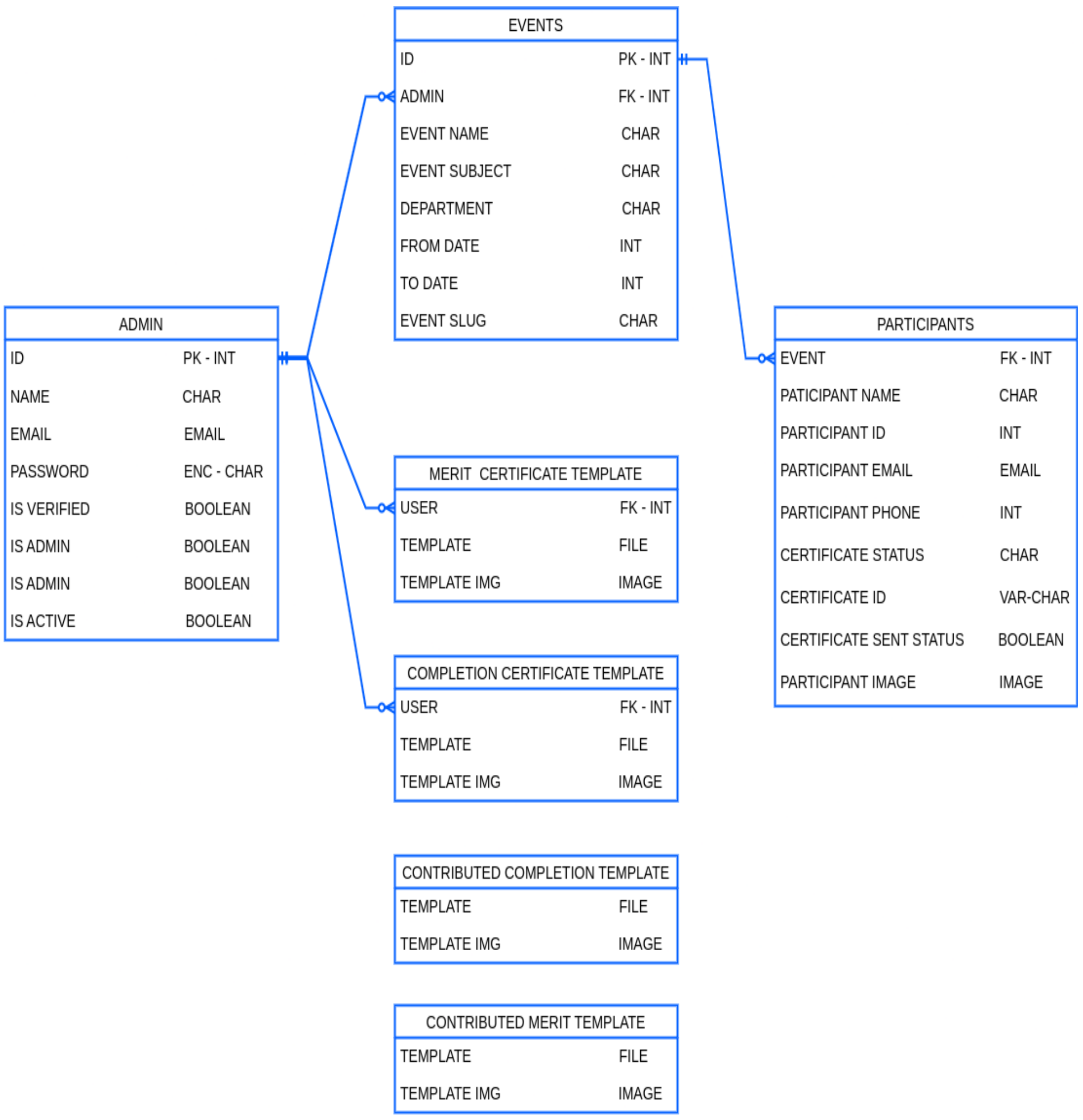


Fig. 3.5 : Entity Relationship Diagram

The Fig. 3.5 shows the entity relationship diagram of this application. This shows the relationship between the tables as well as the attributes.

The application consists of seven tables in a single database. The seven tables are listed below:

- Admin : This table contains all the attributes related to the admin.
- Event : This table contains all the attributes related to the events.

- Participants : This table contains all the attributes related to the participants

- Merit Certificate Template : This table contains all the attributes related to the merit certificate template

- Completion Certificate Template : This table contains all the attributes related to the completion certificate template

- Contributed Completion Template : his table contains all the attributes related to the contributed completion template

- Contributed Merit Template : This table contains all the attributes related to the contributed merit template

The contributed completion template and contributed merit template do not have any connections with the other tables because the data it contains is for all the admins which are available in the network.

3.6 Use Case Diagram

A use case diagram is a graphical representation of the interactions between a system or a software application and its users or external systems. It is one of the Unified Modeling Language (UML) diagrams used to capture the functional requirements of a system and to illustrate the various ways in which users interact with it.

In a use case diagram, use cases are depicted as ovals, and actors (users, external systems or devices) are represented as stick figures. The use cases show the actions or services that the system provides to the actors, while the actors show the roles played by the users or external systems in the interaction with the system.

Use case diagrams can be used to:

- Identify and define the requirements of the system or software application

- Provide a high-level overview of the system's functionality and how it interacts with its environment
- Identify the actors involved in the system and their roles in the interaction

- Show the relationship between the use cases and actors

- Serve as a basis for testing and validating the system's functionality.

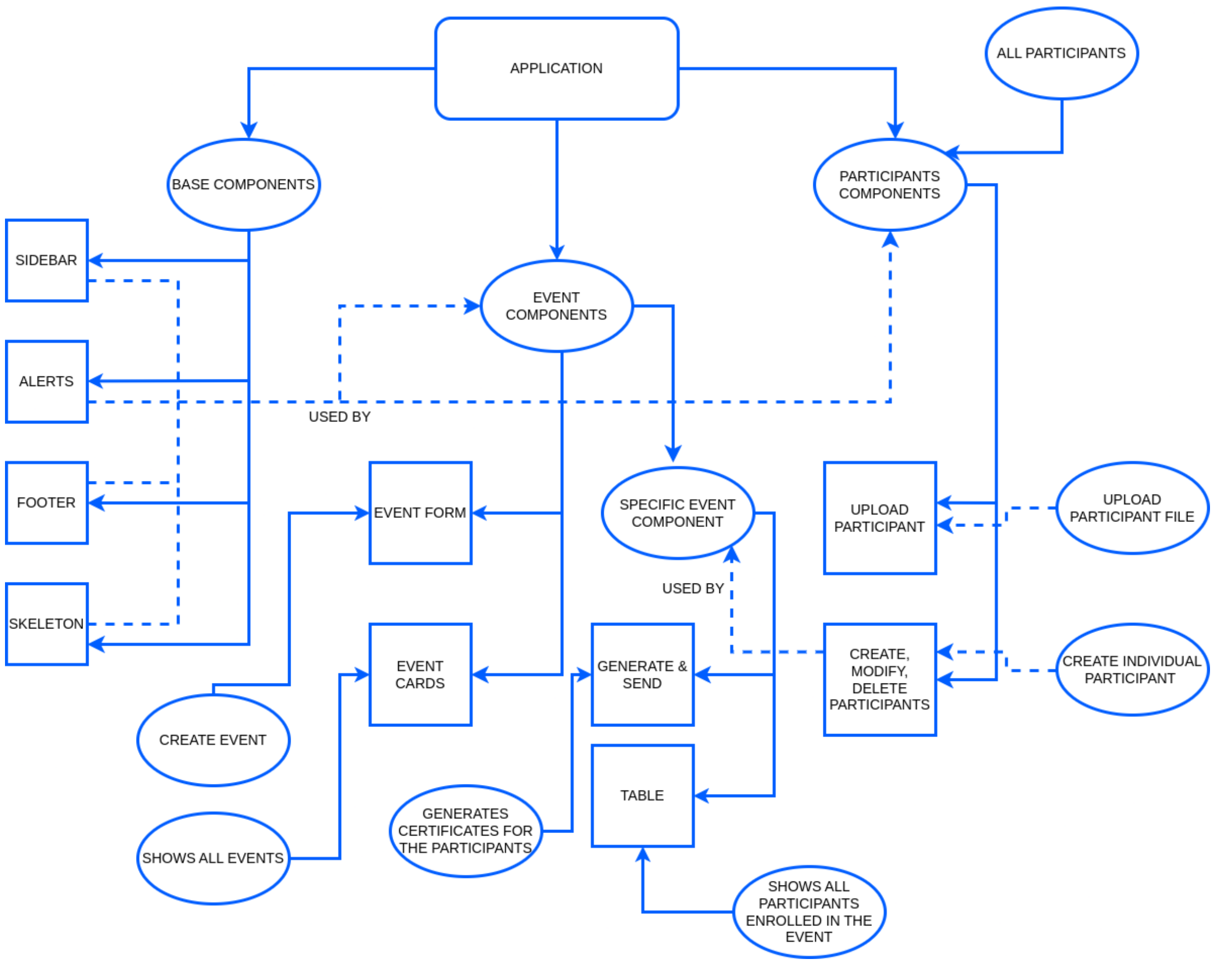


Fig. 3.6 : Use Case Diagram

The Fig. - 3.6 shows the use case diagram of the application.

This diagram consists of all the components used in this application with their work. The main application is divided into three components listed below :

- Base Component : The base component consists of all the child components which are reusable in the application
- Event Component : The event component consists of all the event cards with the event details through which the admin is able to navigate to the specific events.
- Specific Event Component : The specific event component consists of all the participants with their details and be able to perform operations on a single participant. This component also has certificate generation for all the participants.
- Participants Components : The participant component uploads the participants excel file with their details and all the participants data will show on the specific event page.

CHAPTER - 4

IMPLEMENTATION

IMPLEMENTATION

4.1 Implementation of Modules

Implementation refers to the process of putting the project plan into action. It involves executing the tasks and activities necessary to create and deliver the final product or service. Implementation typically follows the planning phase of a project, during which the project team develops a detailed project scope, timeline, and budget.

The implementation of modules in a major project involves the process of designing and developing individual components or units of the project that can be tested, integrated, and deployed as part of the overall system. Modules are typically created to handle specific functions or features of the project and are designed to be interchangeable, making it easier to modify or update the system as needed.

The implementation of modules is an important part of the overall project development process because it allows for greater flexibility and scalability. By breaking the project down into smaller, more manageable pieces, the development team can focus on specific areas of functionality, which can reduce the risk of errors and increase the efficiency of the development process.

4.1.1 Implementation of Backend

4.1.1.1 Implementation of Models

In Django, a model is a Python class that represents a database table. Models define the structure of the data that will be stored in the database and provide an easy way to interact with that data using Python code.

Each attribute of a model represents a field in the database table, and each instance of the model represents a row in that table. Django provides a range of built-in field types, such as CharField for strings, IntegerField for integers, and DateTimeField for dates and times, as well as the ability to create custom fields.

Models also provide a range of built-in methods and tools for working with the data in the database. For example, models can be used to query the database to retrieve specific data, filter and sort data, and create or update records.

Django also includes an Object-Relational Mapping (ORM) system, which allows developers to interact with the database using Python code instead of SQL. This makes it easy to work with the database and avoids the need to write complex SQL queries.

- **FILE NAME** - models.py

  This file contains all the tables used in the application. All the tables are interrelated with each other and store different entries.

```
from django.db import models
from django.contrib.auth.models import
BaseUserManager, AbstractBaseUser
from django.utils.text import slugify
from django.utils.translation import gettext
as _
import os
import string
import random

# Create your models here.

def generate_random_string():
str =
"".join(random.choices(string.ascii_lowercase,
k=10))
return str

def convert_to_img(file_name):

ppt_to_image_command = f'unoconv -f jpg
certificate-data/{file_name}'
os.system(ppt_to_image_command)
```

```
img_file_name =
os.path.splitext(str(file_name))[0]

return f"{img_file_name}.jpg"

class UserManager(BaseUserManager):
def create_user(self, email, name, tc,
password=None, password2=None):
if not email:
raise ValueError("Admin must have an email
address")

user = self.model(
email=self.normalize_email(email),
name=name,
tc=tc
)

user.set_password(password)
user.save(using=self._db)
return user

def create_superuser(self, email, name, tc,
password=None):
user = self.create_user(
email,
password=password,
name=name,
tc=tc,
)

user.is_admin = True
user.save(using=self._db)
return user
```

```
class User(AbstractBaseUser):
email = models.EmailField(
verbose_name="Email Address",
max_length=255,
unique=True
)

name = models.CharField(max_length=200)
tc = models.BooleanField()
is_active = models.BooleanField(default=True)
is_admin = models.BooleanField(default=False)
created_at =
models.DateTimeField(auto_now_add=True)
updated_at =
models.DateTimeField(auto_now=True)

objects = UserManager()

USERNAME_FIELD = 'email'
REQUIRED_FIELDS = ['name', 'tc']

def __str__(self):
return str(self.id)

def has_perm(self, perm, obj=None):
return self.is_admin

def has_module_perms(self, app_label):
return True

@property
def is_staff(self):
return self.is_admin
```

```
class SendersCredentials(models.Model):
user = models.ForeignKey(User,
on_delete=models.CASCADE)
senders_email = models.EmailField(null=True,
blank=True)
senders_password =
models.CharField(max_length=255, null=True,
blank=True)
senders_phone =
models.CharField(max_length=13, null=True,
blank=True)

def __str__(self):
return str(self.user)

class Event(models.Model):
id = models.AutoField(primary_key=True)
user = models.ForeignKey(User,
on_delete=models.CASCADE)
event_name = models.CharField(max_length=20,
null=True, blank=True)
subject = models.CharField(max_length=250,
null=True, blank=True)
event_department =
models.CharField(max_length=20, null=True,
blank=True)
from_date = models.DateField()
to_date = models.DateField()
event_year = models.IntegerField(blank=True,
null=True)
slug = models.SlugField(null=True, blank=True)
certificates_file = models.FileField(
upload_to='pdf-certificates-files', default="")
```

```
certificate_file_name = models.CharField(
max_length=255, null=True, blank=True)

def save(self, *args, **kwargs):
self.slug = slugify(self.event_name) +
generate_random_string()
return super().save(*args, **kwargs)

def __str__(self):
return str(self.id)

class EventFile(models.Model):
event_name = models.ForeignKey(Event,
on_delete=models.CASCADE)
xlsx_file = models.FileField(
upload_to='certificates/csv_files/', null=True,
blank=True)

class Participant(models.Model):
event = models.ForeignKey(Event,
on_delete=models.CASCADE)
participant_name =
models.CharField(max_length=50, null=True,
blank=True)
participant_id =
models.CharField(max_length=50, null=True,
blank=True)
email = models.EmailField(max_length=254,
null=True, blank=True)
phone = models.CharField(max_length=13,
blank=True)
certificate_status =
models.CharField(max_length=10, null=True,
blank=True)
```

```
certificate_id =
models.CharField(max_length=50, null=True,
blank=True)
certificate_sent_status =
models.BooleanField(default=False)
participant_image = models.ImageField(
upload_to='participant-image', null=True,
blank=True)

class
CompletionCertificateTemplate(models.Model):
user = models.ForeignKey(User,
on_delete=models.CASCADE, default=None)
template = models.FileField(
upload_to='completion-certificate-templates/')
template_img = models.ImageField(null=True,
blank=True)

def save(self, *args, **kwargs):
self.template_img =
convert_to_img(self.template.name)
return super().save(*args, **kwargs)

def __str__(self):
return str(self.id)

class MeritCertificateTemplate(models.Model):
user = models.ForeignKey(User,
on_delete=models.CASCADE, default=None)
template = models.FileField(
upload_to='merit-certificate-templates/')
template_img = models.ImageField(null=True,
blank=True)
```

```
def save(self, *args, **kwargs):
self.template_img =
convert_to_img(self.template)
return super().save(*args, **kwargs)

def __str__(self):
return str(self.id)

class
ContributedCompletionCertificates(models.Model)
:
template = models.FileField(
upload_to='contributed-completion-certificate-t
emplates/')
template_img = models.ImageField(null=True,
blank=True)

def save(self, *args, **kwargs):
self.template_img =
convert_to_img(self.template)
return super().save(*args, **kwargs)

def __str__(self):
return str(self.id)

class
ContributedMeritCertificates(models.Model):
template = models.FileField(
upload_to='contributed-merit-certificate-templa
tes/')
template_img = models.ImageField(null=True,
blank=True)

def save(self, *args, **kwargs):
```

```
self.template_img =
convert_to_img(self.template)
return super().save(*args, **kwargs)

def __str__(self):
return str(self.id)

class ParticipantAlbum(models.Model):
event = models.ForeignKey(Event,
on_delete=models.CASCADE, default=None)
image_album = models.ImageField(
upload_to='image-album/', null=True,
blank=True)
```

4.1.1.2 Implementation of Views

In Django, views are Python functions or classes that handle incoming HTTP requests and return HTTP responses. A view function is responsible for processing the user's request and producing an appropriate response, whether it's rendering an HTML template, returning a JSON object, or redirecting the user to a different page.

In Django, views are defined in the views.py file within an app. A view function is typically defined with the def keyword and takes a request object as its first argument. The request object contains information about the user's request, such as the HTTP method, URL, and any data submitted in the request.

- **FILE NAME** - views.py

  This file contains all the api's used in the application. All the api's perform different operations.

```
from django.http import JsonResponse
from django.contrib.auth import authenticate
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from rest_framework.permissions import
IsAuthenticated
from rest_framework.decorators import
permission_classes
from rest_framework.permissions import
IsAuthenticated
from rest_framework_simplejwt.tokens import
RefreshToken
from rest_framework.authentication import *
from .models import *
from .serializers import *
from .resources import *
from .helpers import *
from itertools import islice
from collections import OrderedDict
import openpyxl
import random

# Create your views here.

# Generate access and refresh tokens for users
def get_tokens_for_user(user):
refresh = RefreshToken.for_user(user)
return {
'refresh': str(refresh),
'access': str(refresh.access_token)
}

# User registration view
class UserRegistrationView(APIView):
renderer_classes = [UserRenderer]
```

```
def post(self, request, format=None):
serializer =
UserRegistrationSerializer(data=request.data)
serializer.is_valid(raise_exception=True)
user = serializer.save()
token = get_tokens_for_user(user)
return Response({'token': token, 'msg': 'Admin
registered successfully'},
status=status.HTTP_201_CREATED)

# User login view
class UserLoginView(APIView):
renderer_classes = [UserRenderer]

def post(self, request, format=None):
serializer =
UserLoginSerializer(data=request.data)
serializer.is_valid(raise_exception=True)
email = serializer.data.get('email')
password = serializer.data.get('password')
user = authenticate(email=email,
password=password)
if user is not None:
token = get_tokens_for_user(user)
return Response({'token': token, 'msg': 'Login
success'}, status=status.HTTP_200_OK)
else:
return Response({'errors':
{'non_fields_errors': ['Email or Password is
not valid']}},
status=status.HTTP_404_NOT_FOUND)

# User profile view
class UserProfileView(APIView):
```

```
renderer_classes = [UserRenderer]
permission_classes = [IsAuthenticated]
def get(self, request, format=None):
serializer =
UserProfileSerializer(request.user)
return Response(serializer.data,
status=status.HTTP_200_OK)

# User change password view
class UserChangePasswordView(APIView):
renderer_classes = [UserRenderer]
permission_classes = [IsAuthenticated]

def post(self, request, format=None):
serializer = UserChangePasswordSerializer(
data=request.data, context={'user':
request.user})
serializer.is_valid(raise_exception=True)
return Response({'msg': 'Password changed
successfully'}, status=status.HTTP_200_OK)

# User send password-reset link view
class SendPasswordResetEmailView(APIView):
renderer_classes = [UserRenderer]

def post(self, request, format=None):
serializer =
UserSendPasswordResetEmailSerializer(data=requ
est.data)
serializer.is_valid(raise_exception=True)
return Response({'msg': 'Password reset link
has been sent on your e-mail'},
status=status.HTTP_200_OK)
```

```
# User password-reset view
class UserPasswordResetView(APIView):
renderer_classes = [UserRenderer]

def post(self, request, uid, token,
format=None):
serializer = UserPasswordResetSerializer(
data=request.data, context={'uid': uid,
'token': token})

serializer.is_valid(raise_exception=True)
return Response({'msg': 'Password reset
successfully'}, status=status.HTTP_200_OK)

# Getting single events by slug
@permission_classes((IsAuthenticated,))
def get_event_by_slug(request, slug):
event = Event.objects.filter(slug=slug)
if event:
event_data = EventSerializer(event, many=True)
return JsonResponse(event_data.data,
safe=False)
return JsonResponse("No Data", safe=False)

# Generating UID
def generate_uid(stu_id, eve_name, eve_dept,
eve_date):
random_num = random.randint(1000, 9999)
certificate_id =
str(stu_id)+str(eve_name)+str(eve_dept) + \
str(eve_date).replace("-", "")+str(random_num)
return certificate_id

# Getting all events view
```

```
class EventsOperations(APIView):
renderer_classes = [UserRenderer]
permission_classes = [IsAuthenticated]
def get(self, request):
all_events =
reversed(Event.objects.filter(user=request.user
))
if all_events:
event_serializer_data =
EventSerializer(all_events, many=True)
return
JsonResponse(event_serializer_data.data,
safe=False)
return JsonResponse("No event data",
safe=False)

def post(self, request):
event_serialized_data =
EventSerializer(data=request.data)
if event_serialized_data.is_valid():
event_serialized_data.save()
return JsonResponse("Event added
successfully", safe=False)
return JsonResponse("Failed to add event",
safe=False)

def put(self, request, pk):
event_id = Event.objects.get(id=pk)
event_serialized_data = EventSerializer(
instance=event_id, data=request.data,
partial=True)

if event_serialized_data.is_valid():
event_serialized_data.save()
```

```
return JsonResponse("Event updated
successfully", safe=False)
return JsonResponse("Failed to update event",
safe=False)

# Getting data from xlsx and uploading to
database
class UploadParticipant(APIView):
def get(self, request):
all_participants = Participant.objects.all()

if all_participants:
participant_serializer =
ParticipantSerializer(
all_participants, many=True)
return
JsonResponse(participant_serializer.data,
safe=False)
return JsonResponse("No participant data",
safe=False)

def post(self, request):
event_id = request.data['event_id']
participant_file =
request.data['participants_file']

wb = openpyxl.load_workbook(participant_file)
work_sheet = wb['Form Responses 1']

excel_data = list()
for row in islice(work_sheet.values, 1,
work_sheet.max_row):
data = OrderedDict()
data['id'] = row[0]
```

```
data['Full_Name'] = row[1]
data['Participant_Id'] = row[2]
data['Email'] = row[3]
data['Phone'] = row[4]
data['Certificate_Status'] = row[5]
excel_data.append(data)

eve_id = Event.objects.filter(id=event_id)
event_new_id = Event.objects.get(id=event_id)

event_name = ''
event_dept = ''
event_date = ''

for eve in eve_id:
event_name = eve.event_name
event_dept = eve.event_department
event_date = eve.from_date

event_name_words = event_name.split()
event_name_chars_list = [word[0] for word in
event_name_words]
event_name_chars_string =
"".join(event_name_chars_list)

for data in excel_data:
participant_name = data['Full_Name']
participant_id = data['Participant_Id']
email = data['Email']
phone = data['Phone']
if "+91" in str(phone):
phone = data['Phone']
else:
phone = "+91"+str(data['Phone'])
```

```
certificate_status = data['Certificate_Status']
certificate_id =
generate_uid(data['Participant_Id'],
event_name_chars_string,
event_dept, event_date)
Event.id = Participant.objects.create(
event=event_new_id,
participant_name=participant_name,
participant_id=participant_id, email=email,
phone=phone,
certificate_status=certificate_status,
certificate_id=certificate_id)

return JsonResponse("Participants uploaded
successfully", safe=False)

def delete(self, request, pk):
participant_by_slug =
Participant.objects.get(pk=pk)
participant_by_slug.delete()
return JsonResponse("Participant deleted
successfully", safe=False)

# Uploading each participant from xlsx file
class UploadEachParticipant(APIView):
def post(self, request):
participant_serialized_data =
ParticipantSerializer(data=request.data)

if participant_serialized_data.is_valid():
participant_serialized_data.save()
return JsonResponse("Participant added
successfully", safe=False)
```

```
return JsonResponse("Failed to add
participant", safe=False)

def put(self, request, pk):
participant_by_id =
Participant.objects.get(pk=pk)
participant_serialized_data =
ParticipantSerializer(
instance=participant_by_id, data=request.data,
partial=True)

if participant_serialized_data.is_valid():
participant_serialized_data.save()
return JsonResponse("Participant updated
successfully", safe=False)
return JsonResponse("Failed to update
participant", safe=False)

# Upload participant image
class UploadParticipantImage(APIView):
def patch(self, request, pk):
participant_img =
request.FILES['participant_image']
image = Participant.objects.get(id=pk)
image.participant_image = participant_img
image.save()
return JsonResponse("Image uploaded
successfully", safe=False)

# Filtering events by slug
@ permission_classes((IsAuthenticated,))
class FilteredEvent(APIView):
def get(self, request, slug):
event = Event.objects.get(slug=slug)
```

```
participants =
reversed(Participant.objects.filter(event=event
))

if participants:
participants =
ParticipantSerializer(participants, many=True)
return JsonResponse(participants.data,
safe=False)
return JsonResponse("0", safe=False)

def delete(self, request, slug):
event_by_slug = Event.objects.get(slug=slug)
event_by_slug.delete()
return JsonResponse("Event deleted
successfully", safe=False)

# Uploading completion templates
class UploadCompletionTemplate(APIView):
def get(self, request):
image_file =
CompletionCertificateTemplate.objects.filter(
user=request.user)

if image_file:
image_serializer =
CompletionCertificateSerializer(
image_file, many=True)
return JsonResponse(image_serializer.data,
safe=False)
return JsonResponse("Failed to get images",
safe=False)

def post(self, request):
```

```
file = request.FILES['pptx_file']
contribute = request.data['contribute']

if contribute == "true":
ContributedCompletionCertificates.objects.creat
e(
template=file).save()
else:
user = request.user.id
user_id = User.objects.get(id=user)
CompletionCertificateTemplate.objects.create(
user=user_id, template=file).save()
return JsonResponse("Completion template
uploaded successfully", safe=False)

# Uploading merit templates
class UploadMeritTemplate(APIView):
def get(self, request):
image_file =
MeritCertificateTemplate.objects.filter(
user=request.user)

if image_file:
image_serializer = MeritCertificateSerializer(
image_file, many=True)
return JsonResponse(image_serializer.data,
safe=False)
return JsonResponse("Failed to get images",
safe=False)

def post(self, request):
file = request.FILES['pptx_file']
contribute = request.data['contribute']
```

```
if contribute == "true":
ContributedMeritCertificates.objects.create(tem
plate=file).save()
else:
user = request.user.id
user_id = User.objects.get(id=user)
MeritCertificateTemplate.objects.create(
user=user_id, template=file).save()
return JsonResponse("Merit template uploaded
successfully", safe=False)

# Upload contribute completion templates
class ContributeCompletion(APIView):
def get(self, request):
contribute_img =
ContributedCompletionCertificates.objects.all()

if contribute_img:
contribute_img_serializers =
ContributeCompletionCertificateSerializer(
contribute_img, many=True)
return
JsonResponse(contribute_img_serializers.data,
safe=False)
return JsonResponse("Failed to get images",
safe=False)

# Upload contribute merit certificate
class ContributeMerit(APIView):
def get(self, request):
contribute_img =
ContributedMeritCertificates.objects.all()

if contribute_img:
```

```
contribute_img_serializers =
ContributeMeritCertificateSerializer(
contribute_img, many=True)
return
JsonResponse(contribute_img_serializers.data,
safe=False)
return JsonResponse("Failed to get images",
safe=False)

# Upload and get participant image album
class ParticipantImageAlbum(APIView):
def get(self, request, slug):
image_album_slug =
Event.objects.get(slug=slug)
album_images =
ParticipantAlbum.objects.filter(event=image_alb
um_slug)

if album_images:
album_image_serializer = ImageAlbumSerializer(
album_images, many=True)
return
JsonResponse(album_image_serializer.data,
safe=False)
return JsonResponse("Failed to get images",
safe=False)
def post(self, request, slug):
album_images =
request.FILES.getlist('album_images')
event = Event.objects.get(slug=slug)

for img in album_images:
ParticipantAlbum.objects.create(
event=event, image_album=img)
```

```
return JsonResponse("Image uploaded
successfully", safe=False)
```

4.1.1.3 Implementation of Urls

In Django, URLs (Uniform Resource Locators) are used to map incoming requests from a web browser or other client to the appropriate view function that can handle the request. URLs provide a way to identify and locate resources on the web, such as web pages, images, or data.

- **File Name** - urls.py

  This file contains all the urls and correspondingly connected views from views.py file used in the application. All the urls perform different operations and routing.

  ```
  from .views import *
  from django.urls import path
  from .certificate_generator import *
  from .ppt_2_image_preview import *

  urlpatterns = [
  path('register/',
  UserRegistrationView.as_view(),
  name='register'),
  path('login/', UserLoginView.as_view(),
  name='login'),
  path('profile/', UserProfileView.as_view(),
  name='profile'),
  path('change-password/',
  UserChangePasswordView.as_view(),
  name='change_password'),
  path('reset-password/',
  SendPasswordResetEmailView.as_view(),
  name='send_reset_email_password'),
  ```

```
path('reset-password/<uid>/<token>/',
UserPasswordResetView.as_view(),
name='reset_password'),

path("all-events/",
EventsOperations.as_view()),
path("all-events/<int:pk>",
EventsOperations.as_view()),
path("event/<slug:slug>",
FilteredEvent.as_view()),
path("event-details/<slug:slug>",
get_event_by_slug),
path("create-participant/",
UploadEachParticipant.as_view()),
path("update-participant/<int:pk>",
UploadEachParticipant.as_view()),
path("delete-participant/<int:pk>",
UploadParticipant.as_view()),
path("upload-participants/",
UploadParticipant.as_view()),
path("upload-participant-image/<int:pk>",
UploadParticipantImage.as_view()),
path("upload-event-album/<slug:slug>",
ParticipantImageAlbum.as_view()),
path("generate-certificate/<slug:slug>",
GenerateCertificate.as_view()),
path("generate-certificate/<slug:slug>/<int:pk>
",
GenerateCertificateById.as_view()),
path("preview-certificate/",
Ppt2Image.as_view()),
path("upload-completion-template/",
UploadCompletionTemplate.as_view()),
path("upload-merit-template/",
UploadMeritTemplate.as_view()),
path("contribute-completion-template/",
ContributeCompletion.as_view()),
path("contribute-merit-template/",
ContributeMerit.as_view()),
]
```

4.1.1.4 Implementation of Serializers

In Django, a serializer is a tool that allows you to convert complex data types, such as Django model instances or Python data structures, into a format that can be easily rendered into JSON, XML, or other content types. Serializers are used to facilitate the transmission and storage of data, and are especially useful when building web APIs that need to communicate with external systems or applications.

Django comes with a built-in serialization framework that makes it easy to create serializers for your models and other data types. The framework provides a set of serializer classes that you can use to convert your data to and from various content types.

- **File Name** - serializers.py

  This file contains all the serializers related to the models which are in models.py file and correspondingly connected with the views in the views.py file used in the application. All the serializers perform different operations.

```
from django.contrib.auth.models import User
from django.utils.http import
urlsafe_base64_decode, urlsafe_base64_encode
from django.contrib.auth.tokens import
PasswordResetTokenGenerator
from django.utils.encoding import smart_str,
force_bytes, DjangoUnicodeDecodeError
from rest_framework import serializers
from rest_framework import serializers
from xml.dom import ValidationErr
from .models import *
from .renderers import *

# User registration serializer
class
UserRegistrationSerializer(serializers.ModelSe
rializer):
password2 = serializers.CharField(
style={'input_type': 'password'},
write_only=True
)

class Meta:
model = User
fields = ['email', 'name', 'password',
'password2', 'tc']
extra_kwargs = {
'password': {'write_only': True}
}

def validate(self, attrs):
password = attrs.get('password')
password2 = attrs.get('password2')

if password != password2:
raise serializers.ValidationError(
"Password and Confirm Password is not
matching")
return attrs

def create(self, validated_data):
```

```
return
User.objects.create_user(**validated_data)

# User login serializer
class
UserLoginSerializer(serializers.ModelSerialize
r):
email = serializers.EmailField(max_length=255)

class Meta:
model = User
fields = ['email', 'password']

# User profile serializer
class
UserProfileSerializer(serializers.ModelSerializ
er):
class Meta:
model = User
fields = ['id', 'email', 'name']

# User change password
class
UserChangePasswordSerializer(serializers.Seria
lizer):
password = serializers.CharField(
max_length=255, style={'input_type':
'password'}, write_only=True)
password2 = serializers.CharField(
max_length=255, style={'input_type':
'password2'}, write_only=True)

class Meta:
model = User
```

```
fields = ['password', 'password2']

def validate(self, attrs):
password = attrs.get('password')
password2 = attrs.get('password2')
user = self.context.get('user')

if password != password2:
raise serializers.ValidationError(
"Password and Confirm Password is not
matching")

user.set_password(password)
user.save()
return attrs

# User reset password email
class
UserSendPasswordResetEmailSerializer(serialize
rs.Serializer):
email = serializers.EmailField(max_length=255)

class Meta:
fields = ['email']

def validate(self, attrs):
email = attrs.get('email')

if User.object.filter(email=email).exists():
user = User.objects.get(email=email)
uid =
urlsafe_base64_encode(force_bytes(user.id))
token =
PasswordResetTokenGenerator().make_token(user)
```

```
link =
f'http://localhost:3000/api/user/reset-passwor
d/{uid}/{token}'
body = f'Click on the link to reset password -
{link}'
data = {
'email_subject': 'Reset your password',
'email_body': body,
'to_email': user.email
}
return attrs
else:
raise serializers.ValidationError("You are not
a registered admin")

# User reset password
class
UserPasswordResetSerializer(serializers.Serial
izer):
password = serializers.CharField(
max_length=255, style={'input_type':
'password'}, write_only=True)
password2 = serializers.CharField(
max_length=255, style={'input_type':
'password2'}, write_only=True)

class Meta:
model = User
fields = ['password', 'password2']

def validate(self, attrs):
try:
password = attrs.get('password')
password2 = attrs.get('password2')
```

```
uid = self.context.get('uid')
token = self.context.get('token')

if password != password2:
raise serializers.ValidationError("Password
does not matching")

id = smart_str(urlsafe_base64_decode(uid))
user = User.objects.get(id=id)

if not
PasswordResetTokenGenerator().check_token(user
, token):
raise ValidationErr("Token is not valid")

user.set_password(password)
user.save()
return attrs
except DjangoUnicodeDecodeError as identifier:
PasswordResetTokenGenerator().check_token(user
, token)
raise ValidationErr("Token is not valid")

# Event serializer
class
EventSerializer(serializers.ModelSerializer):
class Meta:
model = Event
fields = '__all__'

def create(self, validated_data):
event =
Event.objects.create(user=validated_data['user
'], event_name=validated_data['event_name'],
subject=validated_data['subject'],
event_department=validated_data['event_departm
ent'],
from_date=validated_data['from_date'],
to_date=validated_data['to_date'],
event_year=validated_data['event_year'])
event.save()
return event

# Participant serializer
class
ParticipantSerializer(serializers.ModelSeriali
zer):
class Meta:
model = Participant
fields = '__all__'

# Participant image serializer
class
ParticipantImageSerializer(serializers.ModelSe
rializer):
class Meta:
model = Participant
fields = ['participant_image']

def update(self, validated_data):
participant_img = Participant.objects.update(
student_image=validated_data['participant_imag
e'])
participant_img.save()
return participant_img

# Completion certificate serializer
```

```
class
CompletionCertificateSerializer(serializers.Mod
elSerializer):
class Meta:
model = CompletionCertificateTemplate
fields = ['template_img']

# Merit Certificate serializer
class
MeritCertificateSerializer(serializers.ModelSer
ializer):
class Meta:
model = MeritCertificateTemplate
fields = ['template_img']

#  Contribution completion certificate
serializer
class
ContributeCompletionCertificateSerializer(seria
lizers.ModelSerializer):
class Meta:
model = ContributedCompletionCertificates
fields = ['template_img']

#  Contribution merit certificate serializer
class
ContributeMeritCertificateSerializer(serializer
s.ModelSerializer):
class Meta:
model = ContributedMeritCertificates
fields = ['template_img']

#  Image album serializer
```

```
class
ImageAlbumSerializer(serializers.ModelSerializ
er):
class Meta:
model = ParticipantAlbum
fields = '__all__'

def create(self, validated_data):
album_image = ParticipantAlbum.objects.create(
event=validated_data['event'],
image_album=validated_data['album_images'])
album_image.save()
return album_image
```

4.1.2 Implementation of Generator and Sender

The E-Certificate generator generates the e-certificate for the participants based on the input data provided by the admin/user. With the help of python and python's library "**python ppt text replacer**" the generator achieved its admin desire.

- **File Name** - certificate_generator.py

  This file contains the generation program of e-certificate. This will fetch the data from the database and based on the input data it generates the certificates. All the serializers perform different operations. This file also contains the sending of an e-certificate program via email and also has the text message program.

```
import glob
import os
from .models import *
from django.http import JsonResponse
from django.core.mail import EmailMessage
from rest_framework.views import APIView
from django.conf import settings
from collections import OrderedDict
from python_pptx_text_replacer import TextReplacer
import tqdm
import qrcode
from pptx import Presentation
from PIL import Image
import smtplib
import ssl
from twilio.rest import Client
from decouple import config
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

def send_message(participant_name, phone):
account_sid = config("TWILIO_SID")
auth_token = config("TWILIO_AUTH_TOKEN")
client = Client(account_sid, auth_token)
client.messages.create(
body=f"Dear {participant_name},
thankyou for participating in the event/contest.
Your certificate will be delivered to you via
e-mail. Check
your email.",
from_="+15855951968",
to=phone
)
return "SENT"

def send_mail(subject, body, email_to,
certificate_file):
email_from = settings.EMAIL_HOST_USER
```

```
password = settings.EMAIL_HOST_PASSWORD
try:
message = MIMEMultipart()
message['From'] = email_from
message['To'] = email_to
message['Subject'] = subject
message['Bcc'] = email_to

message.attach(MIMEText(body, 'plain'))
file = certificate_file

with open(file, 'rb') as attachment:
part = MIMEBase('application', 'octet-stream')
part.set_payload(attachment.read())

encoders.encode_base64(part)

part.add_header('Content-Disposition',
f"attachment;filename={file.replace('
./cert_gen_sen_app_backend/certificate_data/particip
ants-certificates/',
'')}")

message.attach(part)
text = message.as_string()

context = ssl.create_default_context()
with smtplib.SMTP_SSL('smtp.gmail.com', 465,
context=context) as server:
server.login(email_from, password)
server.sendmail(email_from, email_to, text)

return "SENT"
except Exception as e:
```

```
print(e)

# Sending mail to each participant
def sendMail(subject, message, email_to,
certificate_file):
try:
email_form = settings.EMAIL_HOST_USER
certificate = EmailMessage(subject, message,
email_form, [email_to])
certificate.attach_file(certificate_file)
certificate.send()
return "SENT"
except Exception as e:
print(e)

# Certificate directory cleaner
def cleanUp():
participant_files =
"../app/cert_gen_sen_app_backend/certificate_data/pa
rticipants-certificates"
participant_filelist =
glob.glob(os.path.join(participant_files, "*"))
for f in participant_filelist:
os.remove(f)

merit_files =
"../app/cert_gen_sen_app_backend/certificate_data/me
rit-certificates"
merit_filelist = glob.glob(os.path.join(merit_files,
"*"))
for f in merit_filelist:
os.remove(f)

# Placing QR code in PPT
```

```
def place_qrcode(pptx_path, qrcode_path,
replace_str):
pptx_file = Presentation(pptx_path)

for slide in pptx_file.slides:
for shape in slide.shapes:
if shape.has_text_frame:
if shape.text.find(replace_str) != -1:
slide.shapes.add_picture(
qrcode_path, shape.left, shape.top)
pptx_file.save(pptx_path)


# Set size of QR code
def resize_qrcode(qrcode_path):
img = Image.open(qrcode_path)
width = 100
height = 100
resize = img.resize((width, height), Image.NEAREST)
resize.save(qrcode_path)

# Generate QR code
def generate_qrcode(stu_name, stu_id, cert_id,
eve_name, eve_department, eve_date):
qr_code = qrcode.QRCode(
version=5, box_size=2, border=1)
qr_code.add_data(
f"Participant Name - {stu_name}\nParticipant Id -
{stu_id}\nEvent Name - {eve_name}\nEvent
Department - {eve_department}\nEvent Date -
{eve_date}\nCertificate Id - {cert_id}")

type(qr_code)
qr_code.make(fit=True)
```

```
img = qr_code.make_image(fill_color="black",
back_color="white")
img.save(
f'./cert_gen_sen_app_backend/certificate_data/qr-cod
e/{cert_id} - {stu_name}.png')

# resize_qrcode(
#
f'./cert_gen_sen_app_backend/certificate_data/qr-cod
e/{cert_id} - {stu_name}.png')

return
f'./cert_gen_sen_app_backend/certificate_data/qr-cod
e/{cert_id} - {stu_name}.png'

# Generate certificates for all participants type
def generate_participant_certificate(stu_name,
cert_id, qrcode_path, completion_certificate_path):
replacer =
TextReplacer(f'../app{completion_certificate_path}',
slides="", tables=True, charts=True,
textframes=True)

replacer.replace_text(
[("{{StudentName}}", stu_name), ("{{UID}}",
cert_id)])

replacer.write_presentation_to_file(
f'./cert_gen_sen_app_backend/certificate_data/ppt-ce
rtificates/{cert_id} - {stu_name}.pptx')

pptx_path =
f'./cert_gen_sen_app_backend/certificate_data/ppt-ce
rtificates/{cert_id} - {stu_name}.pptx'
```

```
place_qrcode(pptx_path, qrcode_path, "{{QR}}")

path =
'./cert_gen_sen_app_backend/certificate_data/ppt-cer
tificates'
ext = 'pptx'

files = [f for f in glob.glob(
path + "/**/*.{}".format(ext), recursive=True)]

for f in tqdm.tqdm(files):
command = "unoconv -f pdf \"{}\"".format(f)
move_file = "mv
./cert_gen_sen_app_backend/certificate_data/ppt-cert
ificates/*.pdf
./cert_gen_sen_app_backend/certificate_data/particip
ants-certificates/"
os.system(command)
os.system(move_file)

return
f'./cert_gen_sen_app_backend/certificate_data/partic
ipants-certificates/{cert_id} - {stu_name}.pdf'

# Generate certificates for merit participants
def generate_merit_certificate(stu_name, cert_id,
rank, qrcode_path, merit_certificate_path):
replacer =
TextReplacer(f'../app{merit_certificate_path}',
slides="", tables=True, charts=True,
textframes=True)

if (rank == "1"):
```

```
replacer.replace_text(
[("{{StudentName}}", stu_name), ("{{UID}}",
cert_id), ("{{POS}}", f"{rank} st")])
elif (rank == "2"):
replacer.replace_text(
[("{{StudentName}}", stu_name), ("{{UID}}",
cert_id), ("{{POS}}", f"{rank} nd")])
elif (rank == "3"):
replacer.replace_text(
[("{{StudentName}}", stu_name), ("{{UID}}",
cert_id), ("{{POS}}", f"{rank} rd")])

replacer.write_presentation_to_file(
f'./cert_gen_sen_app_backend/certificate_data/ppt-ce
rtificates/{cert_id} - {stu_name}.pptx')

pptx_path =
f'./cert_gen_sen_app_backend/certificate_data/ppt-ce
rtificates/{cert_id} - {stu_name}.pptx'

place_qrcode(pptx_path, qrcode_path, "{{QR}}")

path =
'./cert_gen_sen_app_backend/certificate_data/ppt-cer
tificates'
ext = 'pptx'

files = [f for f in glob.glob(
path + "/**/*.{}".format(ext), recursive=True)]

for f in tqdm.tqdm(files):
command = "unoconv -f pdf \"{}\"".format(f)
```

```
move_file = "mv
./cert_gen_sen_app_backend/certificate_data/ppt-cert
ificates/*.pdf
./cert_gen_sen_app_backend/certificate_data/merit-ce
rtificates/"
os.system(command)
os.system(move_file)

return
f'./cert_gen_sen_app_backend/certificate_data/merit-
certificates/{cert_id} - {stu_name}.pdf'

# Generate certificates for all
class GenerateCertificate(APIView):
def post(self, request, slug):
obj_event = Event.objects.get(slug=slug)
event = Event.objects.filter(slug=slug)
participants =
Participant.objects.filter(event=obj_event)

completion_certificate_path =
request.data['completion']
merit_certificate_path = request.data['merit']

stu_data = []
eve_data = OrderedDict()

for stu in participants:
if stu.certificate_sent_status == False:
if stu.certificate_status == 'T' or
stu.certificate_status == '1' or
stu.certificate_status == '2' or
stu.certificate_status == '3':
```

```
stu_data.append(dict(id=stu.id,
participant_name=stu.participant_name,
participant_id=stu.participant_id, email=stu.email,
phone=stu.phone,
certificate_status=stu.certificate_status,
certificate_id=stu.certificate_id))

for eve in event:
eve_data['event_name'] = eve.event_name
eve_data['event_department'] = eve.event_department
eve_data['from_date'] =
eve.from_date.strftime('%d-%m-%Y')

for stu in stu_data:
if stu['certificate_status'] == "1" or
stu['certificate_status'] == "2" or
stu['certificate_status'] == "3":
qrcode_path = generate_qrcode(
stu["participant_name"], stu["participant_id"],
stu["certificate_id"], eve_data["event_name"],
eve_data["event_department"],
eve_data["from_date"])
certificate_path = generate_merit_certificate(
stu['participant_name'], stu['certificate_id'],
stu['certificate_status'], qrcode_path,
merit_certificate_path)
send_certificate = send_mail("Certificate of
Participation",
"Thank you for participating in the Event/Contest",
stu["email"], certificate_path)

if send_certificate == "SENT":
Participant.objects.filter(
id=stu['id']).update(certificate_sent_status=True)
```

```
else:
qrcode_path = generate_qrcode(
stu["participant_name"], stu["participant_id"],
stu["certificate_id"], eve_data["event_name"],
eve_data["event_department"],
eve_data["from_date"])
certificate_path = generate_participant_certificate(
stu["participant_name"], stu["certificate_id"],
qrcode_path, completion_certificate_path)
send_message(stu["participant_name"], stu["phone"])
send_certificate = send_mail("Certificate of
Participation",
"Thank you for participating in the Event/Contest",
stu["email"], certificate_path)

if send_certificate == "SENT":
Participant.objects.filter(
id=stu['id']).update(certificate_sent_status=True)

return JsonResponse("Certificate generated and sent
successfully", safe=False)
# return JsonResponse("Some problem occured while
sending", safe=False)

# Generate certificates for specific participant
class GenerateCertificateById(APIView):
def post(self, request, slug, pk):
participant = Participant.objects.filter(id=pk)
event = Event.objects.filter(slug=slug)

stu_data = OrderedDict()
eve_data = OrderedDict()
```

```
completion_certificate_path =
request.data['completion']
merit_certificate_path = request.data['merit']

for stu in participant:
stu_data['participant_name'] = stu.participant_name
stu_data['participant_id'] = stu.participant_id
stu_data['email'] = stu.email
stu_data['phone'] = stu.phone
stu_data['certificate_status'] =
stu.certificate_status
stu_data['certificate_id'] = stu.certificate_id

for eve in event:
eve_data['event_name'] = eve.event_name
eve_data['event_department'] = eve.event_department
eve_data['from_date'] =
eve.from_date.strftime('%d-%m-%Y')

if stu_data["certificate_status"] == 'F' or
stu_data["certificate_status"] == None:
return JsonResponse("This participant is not
eligible for certificate", safe=False)
elif stu_data["certificate_status"] == '1' or
stu_data["certificate_status"] == '2' or
stu_data["certificate_status"] == '3':
certificate_path = generate_merit_certificate(
stu_data["name"], stu_data["certificate_id"],
stu_data["certificate_status"], qrcode_path,
merit_certificate_path)
else:
qrcode_path = generate_qrcode(
stu_data["participant_name"],
stu_data["participant_id"],
stu_data["certificate_id"],
eve_data["event_name"],
eve_data["event_department"],
eve_data["from_date"])
certificate_path = generate_participant_certificate(
stu_data["participant_name"],
stu_data["certificate_id"], qrcode_path,
completion_certificate_path)
send_message(stu_data["participant_name"],
stu_data["phone"])
send_certificate = send_mail("Certificate of
Participation",
"Thank you for participating in the Event/Contest",
stu_data["email"],
certificate_path)

if send_certificate == "SENT":
Participant.objects.filter(
id=pk).update(certificate_sent_status=True)
return JsonResponse("Certificate generated and sent
successfully", safe=False)

return JsonResponse("Some problem occured while
sending", safe=False)
```

CHAPTER - 5

EXPERIMENTAL RESULTS

EXPERIMENTAL RESULTS

5.1 Project Testing

Project testing is a critical aspect of any major project, as it helps to ensure that the project meets the necessary requirements and functions as intended. Testing can also help to identify potential issues or bugs that need to be addressed before the project is deployed or released to end-users.

There are several types of testing that can be conducted during the project development lifecycle, including unit testing, integration testing, system testing, and acceptance testing.

Unit testing involves testing individual components or modules of the project to ensure that they function as intended. This type of testing is typically conducted by developers using specialized testing tools or frameworks.

Integration testing involves testing how different modules or components of the project work together. This type of testing is important to ensure that the project can integrate with other systems or technologies as needed.

System testing involves testing the entire system or application to ensure that it meets the necessary requirements and functions as intended. This type of testing is typically conducted by quality assurance (QA) teams and may involve a range of testing techniques, including functional testing, performance testing, and security testing.

Acceptance testing involves testing the project with end-users to ensure that it meets their needs and expectations. This type of testing can be conducted through user acceptance testing (UAT) sessions, surveys, or other feedback mechanisms.

5.1.1 Testing of Authentication

The authentication system is built on django. The "**djoser**" library is used in the authentication system. This system is based on the django rest framework and provides JWT authentication. This authentication system provides access token and refresh token to the user so that the user can easily be able to connect with the backend and access the backend resources. This token has a time limit which is configured by the admin/user.

The authentication system is working fine and will be able to create as well as login an admin/user.

5.1.2 Testing of Backend

The backend is designed in python django. All the api’s are written in django and will operate CRUD operations successfully. The api’s are divided into several categories which are listed below :

- Event API’s - The event api's perform several different operations on events like CRUD operations.
- Participant API’s - The event api's perform several different operations on events like CRUD operations.
- Generator API - This api has subdivided into two types listed below :
    - Certificates Generator for all participants - This will generate certificates for all the participants.
    - Certificate Generator for a single participant - This will generate a certificate for a single participant.

  The generator api is working fine and can be able to generate as many certificates as you want.
- Sender API - This api is for sending the certificates for the participants after generating the certificates and will be able to send as many certificates as the admin/user wants.

The backend is working on its full potential and can be used at production level.

5.1.4 Testing of Frontend

The frontend is designed in react js, material ui and tailwind css. The frontend is capable of performing different activities based on the user requirements. The frontend is also divided into several categories which are listed below :

- Event - This will help the admin/user to perform CRUD operations using an interactive view.
- Participant - This will help the admin/user to perform CRUD operations using an interactive view.

- Generation of Certificates - This will help the admin/user to generate the certificates using a single click.

5.1.5 Testing of Generator

The generator is built on python and can be able to generate as many certificates as the admin/user wants. The generator are of two types listed below :

- Generator for all participants - This will generate certificates for all the participants present in the event/contest.

- Generator for a single participant - This will generate a certificate for a single participant.

The testing of the generator is successful and will be able to be used in production as well.

5.1.6 Testing of Sender

The sender is also capable of sending the certificates to the participants in a single click. The sender needs a sender’s email and password through which the admin/user will be able to send the generated certificates. The sender is also of several types listed below

- Certificate sender for participants - This will send the certificate to all the participants presented in the event.

- Certificate sender for merit - This will send the certificate to all the merit participants.

The only difference between the sender of participants and the sender of merit is the message which is going to be delivered as a body of email.

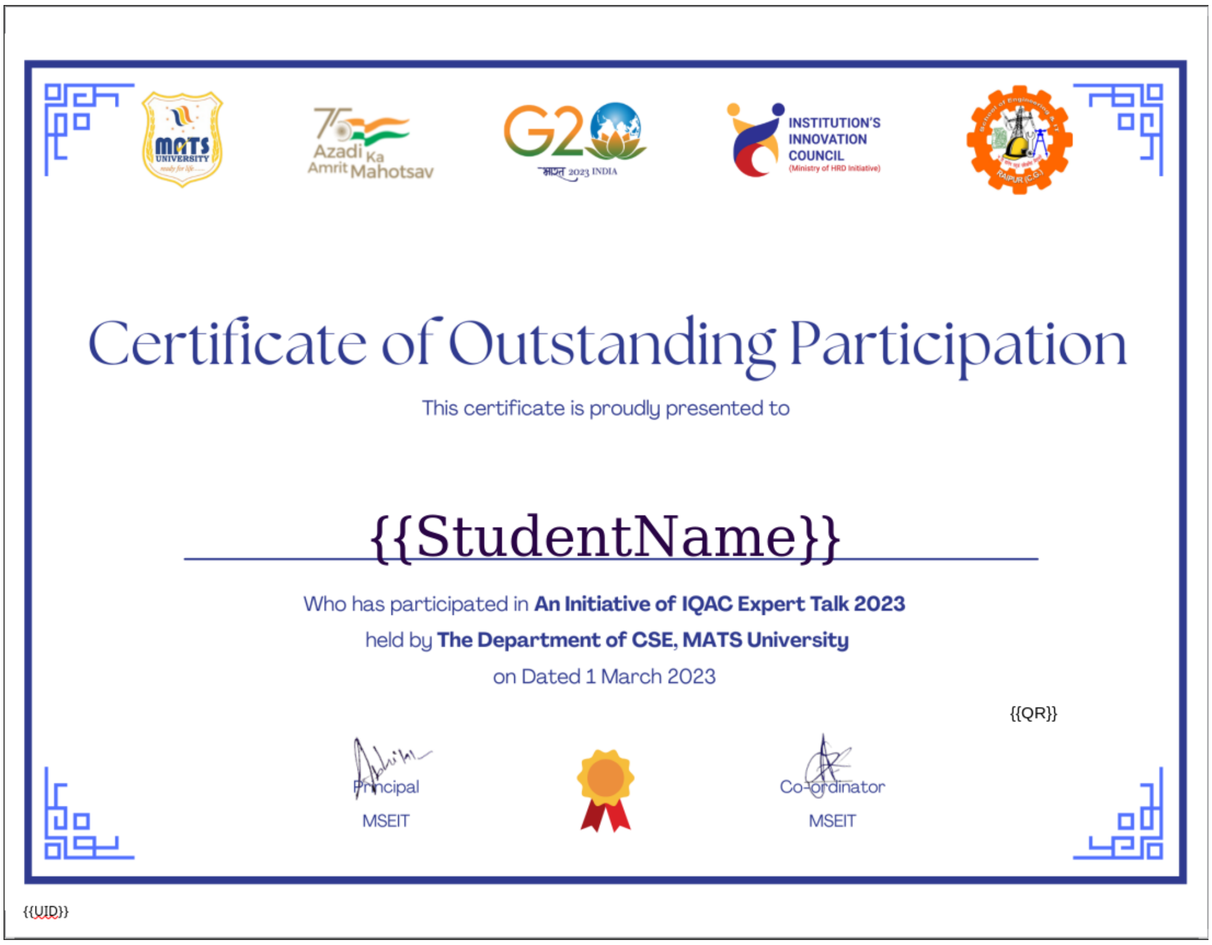
There is also a text message sender which sends text messages to the user's mobile phone number.

5.2 Experimental Output

Experimental output in a major project is the data or results collected from experiments conducted as part of the project. It can include statistical summaries, graphs, tables, and other visualizations of the data collected. It can also include reports and summaries of the findings and conclusions drawn from the experiments.

The output includes the certificate with the name of the participant, position of the participant, uid of the participant as well as the QR code which holds the details of the participant.

Below is the result of the application :



Certificate of Outstanding Participation

This certificate is proudly presented to

{{StudentName}}

Who has participated in **An Initiative of IQAC Expert Talk 2023**
held by **The Department of CSE, MATS University**
on Dated 1 March 2023

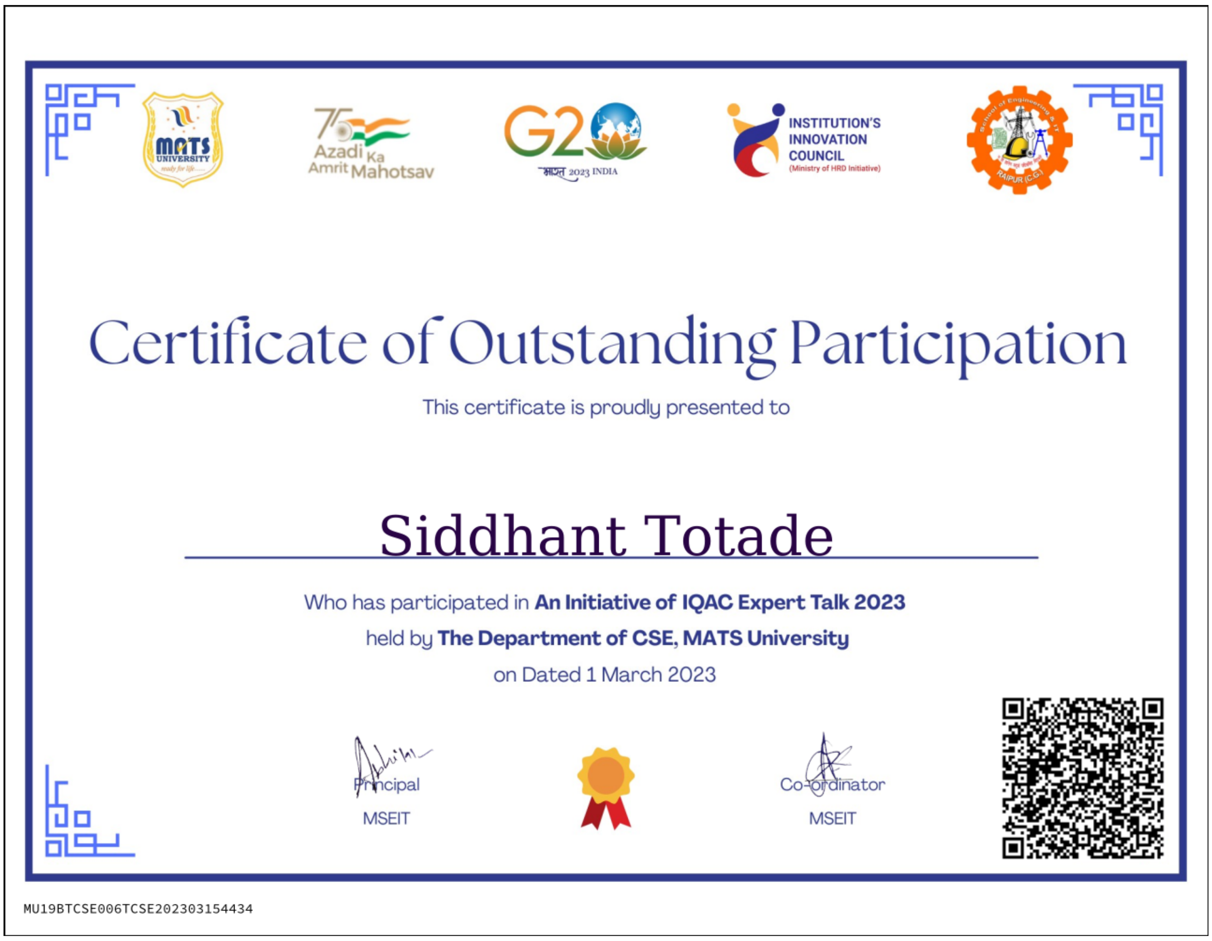
{{QR}}

Principal
MSEIT

Co-ordinator
MSEIT

{{UID}}

Fig. 5.2 - Before Generation of Certificate



INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

Certificate of Outstanding Participation

This certificate is proudly presented to

Siddhant Totade

Who has participated in **An Initiative of IQAC Expert Talk 2023** held by **The Department of CSE, MATS University** on Dated 1 March 2023

[illegible]
Principal
MSEIT

[illegible]
Co-ordinator
MSEIT

MU19BTCSE006TCSE202303154434

Fig. 5.2 - After Generation of Certificate

CHAPTER - 6

CONCLUSION & FUTURE SCOPE

CONCLUSION & FUTURE SCOPE

6.1 Conclusion

The e-Certificate Generator project is an effective way of creating customized e-Certificates for different types of events. It is a user-friendly and cost-effective project that can help organizations save a lot of money and time when creating e-Certificates. The project also includes several features that allow users to easily customize their e-Certificates according to their needs. The e-Certificate Generator project has been tested and proven to be reliable and efficient.

In conclusion, the development of an e-certificate generator for a major project has several advantages. First and foremost, it saves time and resources by automating the certificate creation process. Instead of manually designing and printing certificates, the system can generate them in a matter of seconds, making it much more efficient.

Furthermore, the use of an e-certificate generator ensures the accuracy and consistency of the certificates. With a standardized template, all certificates will have the same format and information, reducing the risk of errors or inconsistencies. This also makes it easier for recipients to verify the authenticity of their certificate, as they can compare it to others and ensure that it matches the template.

In addition, an e-certificate generator allows for greater flexibility and customization. The system can be designed to accommodate different certificate types, such as certificates of completion, participation, or achievement. It can also be tailored to include specific information about the recipient or the event, such as the date, location, or name of the organizer.

Overall, the development of an e-certificate generator is a valuable tool for any organization that frequently issues certificates. It streamlines the process, ensures accuracy and consistency, and provides greater flexibility and customization options. With the increasing trend towards digitalization and automation, an e-certificate generator is a step towards a more efficient and effective certificate issuance process.

6.2 Future Enhancement

There are several potential future enhancements that could be made to an e-certificate generator:

- Integration with a database: An e-certificate generator could be integrated with a database that stores recipient information, event details, and other relevant information. This would enable the system to automatically populate certificates with accurate information, reducing the risk of errors and saving time.

- Customizable templates: While an e-certificate generator can provide standard templates, offering more customizable templates will give users the ability to personalize the design of the certificates. Users could select from a range of designs, fonts, colors, and images to create a more unique certificate.

- Incorporation of multimedia elements: An e-certificate generator could include multimedia elements such as video, audio, or animations, providing a more interactive and engaging experience for the recipient.

- Integration with digital signature technology: Incorporating digital signature technology into the e-certificate generator could enable recipients to verify the authenticity of their certificate, providing greater security and trustworthiness.

- Mobile compatibility: Making the e-certificate generator compatible with mobile devices would allow for greater convenience and accessibility, enabling users to generate and share certificates on-the-go.

- Multilingual support: Adding support for multiple languages would make the e-certificate generator more accessible to a wider range of users, especially in countries with diverse linguistic backgrounds.

- Integration with social media platforms: Integrating the e-certificate generator with social media platforms could enable users to share their certificates directly on their profiles, increasing visibility and promoting the event or organization that issued the certificate.

REFERENCES

REFERENCES

For creating the application's frontend and backend I use the following tech stack.

Django	https://docs.djangoproject.com/en/4.0/
React JS	https://react.dev/
Material UI	https://mui.com/
Tailwind Css	https://tailwindcss.com/

Ref. Table - 1

For debugging and tips & tricks I use stack overflow and youtube.

StackOverflow	https://stackoverflow.com/
YouTube	https://www.youtube.com/

Ref. Table - 2