**CHAPTER 1**

**INTRODUCTION**

**1.1 OVER VIEW**

Starting a business is an exciting yet challenging endeavor, especially when navigating the maze of regulatory approvals required from multiple government bodies. Entrepreneurs often face significant delays and frustration due to outdated manual processes, fragmented systems, and a lack of integration between departments. The need to obtain clearances from entities such as the Goods and Services Tax (GST) department, the Registrar of Companies (ROC), and the Ministry of Micro, Small and Medium Enterprises (MSME) adds layers of complexity, creating barriers that can slow down or even discourage new business ventures.To address these challenges, the Smart Approval System has been conceptualized as a comprehensive digital solution aimed at transforming the way startup approvals are handled. This web-based platform offers a centralized and streamlined approach to manage the entire startup registration process—from application submission to final approval—while significantly reducing the need for manual intervention.The platform integrates directly with existing government portals such as GST, ROC, and MSME through secure APIs, automating compliance verification and document validation. Entrepreneurs can submit required documents online, make payments for registration fees through in-app payment gateways, and track the progress of their applications in real time. Each step of the process is guided through an intuitive and user-friendly interface that makes the system accessible even to first-time users.To ensure security and data integrity, the system employs blockchain technology for document authentication and a secure cloud-based backend developed using Python. The backend supports efficient data processing, real-time notifications, and seamless communication between users and officials. Role-based authentication allows for personalized access depending on the user’s role—entrepreneur, government official, or consultant—ensuring appropriate control and accountability at every level.By combining automation, transparency, and smart integration, the Smart Approval System aims to eliminate inefficiencies in the traditional startup approval process. It empowers aspiring business owners by providing them with a faster, clearer, and more reliable pathway to establish their ventures, ultimately promoting entrepreneurship and economic growth.

**1.2 LITERATURE SURVEY**

This project presents a secure web-based user registration and verification system using Flask, incorporating document uploads, OTP-based email verification, and an admin approval workflow. Literature in the domain of user authentication highlights the effectiveness of email-based OTP systems for verifying user identities, offering a balance between usability and security (Bonneau et al., 2012). Document upload and manual verification mechanisms are widely used in government and KYC applications to ensure identity authenticity (Shukla & Singh, 2020). Flask, a lightweight Python framework, is known for its modularity and ease of integration with extensions like Flask-Mail and Flask-Login, making it suitable for rapid web development (Grinberg, 2018). The use of SMTP for sending OTPs and approval notifications is a standard, reliable practice in secure communication systems (Shrivastava & Mehta, 2019). Moreover, the project features an admin dashboard enabling officers to review and approve or reject applications—a pattern commonly found in business process workflows (Singh & Sharma, 2021). Additionally, the generation of personalized approval letters using ReportLab aligns with practices in dynamic PDF reporting for formal documentation (Lin, 2020). Overall, the project synthesizes proven methods from existing literature to offer a practical solution for identity verification and approval systems.

**CHAPTER 2**

**EXISTING SYSTEM AND PROPOSED SYSTEM**

**2.1 EXISTING SYSTEM**

The current system for business startup approvals is often fragmented, relying heavily on manual processes and paper-based submissions across multiple regulatory bodies. Entrepreneurs must navigate through various departments, including the Goods and Services Tax (GST), Registrar of Companies (ROC), and Ministry of Micro, Small and Medium Enterprises (MSME), among others. Each department has its own set of requirements, processes, and platforms, creating a disjointed experience for the applicants.One of the key drawbacks of the existing system is the reliance on manual submissions and verifications. Entrepreneurs are required to submit physical copies of documents and forms to different regulatory agencies, often duplicating the same information for each department. This leads to significant delays, as manual verification processes are time-consuming and prone to human error. Furthermore, documents are often misplaced or lost during transit, adding to the administrative burden and potential for errors.Approvals are handled by various departments without any coordination or integration between them. This lack of interoperability means that information submitted to one department does not automatically flow to the others, causing inefficiencies and duplicative efforts. Entrepreneurs often face challenges in tracking the progress of their applications, as there is no centralized system that provides real-time updates. This leaves them in the dark about where their applications stand and when they can expect approval.Redundant paperwork further exacerbates these issues, creating additional administrative overhead for both entrepreneurs and government officials. In addition to the delays, this manual process increases the risk of data loss, errors, and miscommunication, which can ultimately lead to frustrated applicants and longer wait times for approval.In conclusion, the existing system for business startup approvals is inefficient, outdated, and fragmented, contributing to delays, errors, and a lack of transparency, all of which hinder the smooth creation of new businesses.

**2.1.1 DRAWBACKS:**

* Manual processes lead to significant delays in approvals.
* Manual data entry increases the risk of errors.
* Entrepreneurs have limited visibility into the approval process.
* Government agencies spend excessive time on manual verifications.
* Entrepreneurs face difficulties in navigating complex approval processes

**2.2 PROPOSED SYSTEM**

The proposed Smart Approval Platform is a web-based solution designed to streamline and expedite the business startup approval process, addressing the inefficiencies of existing systems. By integrating with key government portals such as GST, ROC, and MSME, the platform automates compliance verification, eliminating the need for manual submissions and checks. This integration ensures that all necessary approvals are processed faster and more accurately, reducing delays and human error.One of the core features of the platform is its user-friendly interface, which guides entrepreneurs step-by-step through the entire registration process. Entrepreneurs can easily navigate through forms, upload documents, and complete various tasks without the complexity typically associated with business registration. The platform provides real-time progress tracking, allowing users to monitor the status of their applications, receive updates, and understand exactly where their submission stands.The system also incorporates in-app payment processing, enabling entrepreneurs to securely pay registration fees directly through the platform. This feature eliminates the need for external payment processes, ensuring a seamless experience.Built on a secure Python-based backend, the platform ensures the highest levels of data security, protecting sensitive information throughout the registration process. Additionally, it supports role-based access, granting different levels of control and visibility to users such as entrepreneurs, government officials, and consultants. This ensures that each stakeholder can only access the information and functions relevant to their role, maintaining data integrity and confidentiality.By centralizing and automating the business registration workflow, the Smart Approval Platform significantly reduces processing time, enhances transparency, and fosters better communication between all parties involved. It aims to create a more efficient, secure, and user-friendly experience for entrepreneurs, government officials, and consultants, ultimately accelerating the process of starting a business.

**2.2.1 ADVANTAGES**

* Reduces approval delays through automation
* Centralizes all business registration processes in one platform
* Enhances transparency and minimizes manual errors
* Improves coordination between all stakeholders
* Provides real-time status updates and notifications

**CHAPTER 3**

**SYSTEM SPECIFICATION**

**3.1 HARDWARE REQUIREMENTS**

PROCESSOR : INTEL® CORE™

OPERATING SYSTEM : WINDOWS 10PRO

RAM : 4

HARD DISK : 450 TB

**3.2 SOFTWARE REQUIREMENTS**

OPERATING SYSTEM : WINDOWS

Frontend :HTML,CSS

Backend : PYTHON

Framework : FLASK

**CHAPTER 4**

**SOFTWARE DESCRIPTION**

**4.1 FRONTEND**

**HYPERTEXT MARKUP LANGUAGE (HTML)**

As we all know HTML is a language of the web. It’s used to design the web pages or we can say structure the page layouts of a website.  HTML stands for HYPERTEXT MARKUP LANGUAGE, as its full form suggests it’s not any programming language, a markup language. So, while the execution of HTML code we can’t face any such error. In real HTML code wasn’t compiled or interpreted because HTML code was rendered by the browser. which is similar to the compilation of a program. Html content is parched through the browser to display the content of HTML.



**HTML DOCUMENTS STRUCTURE**

Html used predefined tags and attributes to tell the browser how to display content, means in which format, style, font size, and images to display. Html is a case insensitive language. Case insensitive means there is no difference in upper case and lower case ( capital and small letters) both treated as the same, for r example ‘D’ and ‘d’ both are the same here. There are generally two

types of tags in HTML:

1. **Paired Tags**: These tags come in pairs. That is they have both opening(<>) and closing(</ >) tags.
2. **Empty Tags**: These tags do not require to be closed.

Below is an example of a (<b>) tag in HTML, which tells the browser to bold the text inside it.

**Tags and attributes**

Tags are individuals of html structure, we have to open and close any tag with a forward slash like this <h1></h1>. There are some variations with the tag some of them are self-closing tag which isn’t required to close and some are empty tag where we can add any attributes in it. Attributes are additional properties of html tags that define the property of any html tags. i.e. width, height, controls, loops, input, and autoplay. These attributes also help us to store information in meta tags by using name, content, and type attributes. Html documents structured mentioned below:

**CASCADING STYLE SHEETS (CSS)**



Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, or even CSS3. There will never be a CSS3 or a CSS4; rather, everything is now CSS without a version number. After CSS 2.1, the scope of the specification increased significantly and the progress on different CSS modules started to differ so much, that it became more effective to develop and release recommendations separately per module. Instead of versioning the CSS specification, W3C now periodically takes a snapshot of the latest stable state of the CSS specification and individual modules progress. CSS modules now have version numbers, or levels, such as CSS Colour Module Level 5.

CSS (Cascading Style Sheets) is used to style and layout web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features. This module provides a gentle beginning to your path towards CSS mastery with the basics of how it works, what the syntax looks like, and how you can start using it to add styling to HTML

[**CSS building blocks**](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks)

This module carries on where [CSS first steps](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps) left off — now you've gained familiarity with the language and its syntax, and got some basic experience with using it, it's time to dive a bit deeper. This module looks at the cascade and inheritance, all the selector types we have available, units, sizing, styling backgrounds and borders, debugging, and lots more.

The aim here is to provide you with a toolkit for writing competent CSS and help you understand all the essential theory, before moving on to more specific disciplines like [text styling](https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text) and [CSS layout](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout).

[**CSS styling text**](https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text)

With the basics of the CSS language covered, the next CSS topic for you to concentrate on is styling text — one of the most common things you'll do with CSS. Here we look at text styling fundamentals, including setting font, boldness, italics, line and letter spacing, drop shadows, and other text features. We round off the module by looking at applying custom fonts to your page, and styling lists and links.

**4.2 BACKEND**

**PYTHON**



Python is a general purpose programming language. It is very easy to learn, easy syntax and readability is one of the reasons why developers are switching to python from other programming languages. We can use python as object oriented and procedure oriented language as well. It is open source and has tons of libraries for various implementations.Python is a high level interpreted language, which is best suited for writing python scripts for automation and code re-usability. It was created in 1991 by Guido Van Rossum. The origin of its name is inspired by the comedy series called ‘Monty python’.

Python is a high-level scripting language, interpreted, interactive, and object-oriented. it is designed to be highly readable. It often uses keywords in English where other languages use punctuation and has less syntactic constructions than other languages.

* It is a powerful and easy to learn programming language and efficient high-level data structures and a simple but effective approach to object-oriented programming.
* The interpreter processes Python at runtime. It is not necessary to compile the program before executing it. This is similar to PERL and PHP.
* You can sit on a Python indicator and interact directly with the interpreter to write your own programs.
* Python supports object-oriented style or the programming technique that encapsulates the code within objects.
* Python is a great language for beginner’s programmers and supports the development of a wide range of applications.



**Object-oriented:**

Python is an object-oriented programming (OOP) language. This means that it allows you to model and organize code using objects, which are instances of classes. Objects encapsulate data and behavior, making it easier to manage and manipulate complex systems.

**High-level language:**

Python is a high-level programming language, which means it abstracts many low-level details and provides a more user-friendly syntax. This makes it easier for programmers to write code without having to worry about memory management or other low-level operations.

**Easy to learn:**

Python is designed with readability and simplicity in mind. It uses English-like syntax and is known for its clear and concise code. This makes it an excellent choice for beginners who are just starting to learn programming.

**Procedure-oriented:**

While Python is primarily an object-oriented language, it also supports procedural programming. In procedural programming, code is organized into procedures or functions, and the focus is on writing procedures to operate on data. Python allows you to write code in a procedural style when it makes sense, offering flexibility in programming paradigms.

**WHY PYTHON**

**High Level**

Python derives components from the natural language that we humans use to communicate with each other. This makes it easier for anyone to try and relate what exactly could be happening without the burden of going through tons of machine code.

**Interpreted**

Python codes are compiled line-by-line which makes debugging errors much easier and efficient. But this comes at a cost as it is much slower than other programming languages.

**Easy Syntax**

Python makes use of indentations instead of braces to distinguish what blocks of code come under which class or function. This makes the code look well distributed and makes it easy for anyone to read it.

**Dynamic Semantics**

If you are an old school coder, you would know that before using anything, you would need to initialize it. It does all of this dynamically.

**PYTHON USED**

* Creating web applications with Python Frameworks such as Django and Flask
* You can create workflows for the software that you are working on
* Use Python to modify files and data stored in Databases
* Scientific, Analytic and complex calculations can be taken care of easily
* You can create software much quicker with Python, which is ready for deployment

**Data Scientist**

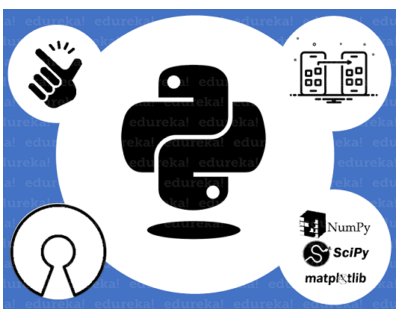
A Data Scientist is someone who cracks complex problems which relate to the field of math, statistics and brings around a solution to these problems in a logical manner.

**Software Engineer**

Software engineers design, develop, test and maintain software applications that they create for their clients according to the requirements.

**Web Developer**

Web developers create web applications to serve their users using the client-server model. There are applications such as information sharing, social network platforms, entertainment which are just a few to name.



**LIBRARIES:-**

* Tensorflow
* Pytorch
* Numpy
* Scikit-Learn
* G4f

**TensorFlow**

****

TensorFlow is an open-source deep learning framework developed by Google that is widely used for machine learning applications. It provides a flexible platform for developing and training deep neural networks, including support for multi-GPU processing, which makes it ideal for large-scale data tasks. TensorFlow allows developers to efficiently implement models for various unstructured data types, including images, text, and audio. Its tf.distribute.Strategy API enables seamless scaling across multiple GPUs or even entire clusters, optimizing the performance of training tasks. TensorFlow also includes various utilities for model deployment, including TensorFlow Serving for real-time predictions. Its large community and ecosystem make it an excellent choice for building high-performance machine learning models in production environments.

**PYTORCH**



PyTorch is another popular deep learning framework, primarily developed by Facebook’s AI Research lab. It is known for its flexibility and dynamic computation graph, which allows for easier debugging and experimentation compared to static frameworks like TensorFlow. PyTorch has native support for multi-GPU parallelism through torch.nn.DataParallel and torch.nn.parallel.DistributedDataParallel. This makes it well-suited for tasks that require high computational power, such as image classification, speech recognition, and natural language processing. PyTorch also has a rich ecosystem of libraries, including tools for data processing, model deployment, and reinforcement learning, making it highly versatile and user-friendly.

**NumPy**

****

NumPy is a fundamental library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them. It serves as the backbone for many scientific computing libraries in Python, including TensorFlow and PyTorch. NumPy allows for efficient numerical computations, such as matrix operations and linear algebra, which are essential for machine learning tasks. It is also highly optimized for speed, offering vectorized operations that reduce the need for explicit loops in Python code. Its broad support for array manipulations makes it an ideal choice for preprocessing and feature extraction in data science and machine learning applications.

**SCIKIT-LEARN**



Scikit-learn is a powerful and widely used open-source machine learning library built on Python. It provides simple and efficient tools for data mining, data analysis, and machine learning tasks. The library is built on top of core Python libraries such as NumPy, SciPy, and matplotlib. Scikit-learn includes a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and model selection. It is known for its clean and consistent API, making it beginner-friendly while still being powerful for advanced users. Scikit-learn is ideal for creating and testing machine learning models quickly, especially in academic, research, and industrial applications.

**CHAPTER 5**

**PROJECT DESCRIPTION**

**5.1 MODULE DESCRIPTION**

* User Registration & Authentication Module
* Business Profile & Document Submission Module
* Government Approval System for Business Registration Module
* Real-Time Application Tracking & Notifications Module
* Licensing & Certification Module

**5.1.1 User Management Module**

This module allows entrepreneurs, consultants, and government officials to securely register and log into the platform. It supports role-based authentication to provide access according to user type. Secure credentials, OTP/email verification, and password encryption ensure user data protection. It establishes a unique identity for each user on the platform. Admins can manage users and monitor activities. This module lays the foundation for a secure, personalized experience.

**5.1.2 Business Profile & Document Submission Module**

Entrepreneurs can create detailed business profiles with necessary information like company name, type, and location. The module facilitates uploading of mandatory documents such as PAN, Aadhar, and incorporation certificates. It ensures structured data input and document validation checks. Users can edit or update their profiles during the process. The system verifies document format and completeness before submission. This module ensures readiness for compliance and approval.

**5.1.3 Government Approval System for Business Registration Module**

This module integrates with GST, ROC, MSME, and other portals to automate compliance checks. It enables officials to review applications, verify documents, and approve or reject requests. Each action taken by officials is logged for transparency. Notifications are sent to users on progress and decision outcomes. Workflow automation reduces manual handling and processing time. The module bridges communication between government bodies and applicants.

**5.1.4 Real-Time Application Tracking & Notifications Module**

This module provides a dynamic dashboard for users to track their application status across departments. Automated notifications via email/SMS inform users of updates, pending actions, or approvals. It reduces uncertainty and improves user experience with transparency. Users can view the timeline and history of actions on their application. The module enhances accountability and encourages timely actions. It serves as the communication bridge throughout the process.

**5.1.5 Licensing & Certification Module**

This module manages the issuance of industry-specific licenses and certificates once approvals are complete. It ensures digital signing and verification of certificates to maintain authenticity. Entrepreneurs can download and store these documents from their dashboard. The module also tracks license validity and sends renewal reminders. Integration with state/national licensing bodies ensures compliance. It simplifies access to all official credentials in one place.

**CHAPTER 6**

**SYSTEM TESTING**

To ensure the successful deployment of the Smart Approval System, comprehensive system testing is carried out to validate the complete functionality, performance, security, and integration of the platform. As this platform handles sensitive business data, real-time status tracking, and government integration, it is crucial that each component works as expected in a real-world environment. The testing phase focuses on evaluating the system’s ability to handle business startup registrations efficiently and securely.

**6.1 Functional Testing**

Functional testing is the foundation of system validation. Each module within the application is tested independently and in combination to ensure the system behaves according to defined requirements. For example, testers check if entrepreneurs can register, submit documents, make in-app payments, and track their application status without any issues. The workflow for government officials and consultants is also validated to ensure proper handling of application reviews, approvals, or rejections.

**6.2 Integration Testing**

As the platform integrates with external government systems such as GST, ROC, and MSME, integration testing is essential. Simulated API calls are made to verify seamless data exchange between the Smart Approval System and government portals. Scenarios including failed API responses, incorrect data formats, and delayed responses are also tested to ensure the system handles exceptions gracefully

**6.3 Role-Based Access Control Testing**

This testing ensures that different users—entrepreneurs, government officials, and consultants—access only their designated sections of the application. Role restrictions are verified by attempting unauthorized actions, such as a regular user trying to access government review modules or a consultant editing entrepreneur submissions. This prevents data manipulation and ensures a secure, compliant environment.

**6.4 UI/UX Testing**

The application is tested across various browsers and devices to ensure consistent user experience. All interfaces are checked for responsiveness, accessibility, and navigation. User-friendly design is validated by testing the guided registration flow, dashboard usability, error messages, and form validations. The goal is to ensure that even first-time users can complete the process smoothly.

**6.5 Real-Time Notification Testing**

This involves testing the triggering and delivery of real-time alerts for different stages in the approval process. Notifications are sent via email or SMS when users submit applications, receive approvals, or make payments. Testers verify the content, accuracy, and timing of each notification to enhance user awareness and responsiveness.

**6.6 Security Testing**

Given the sensitivity of the data handled, thorough security testing is conducted. This includes testing encryption of passwords and personal details, validating secure access to uploaded documents, and checking for vulnerabilities such as SQL injection, XSS attacks, and unauthorized file access. Session management, user timeout, and access logging are also tested to prevent breaches.

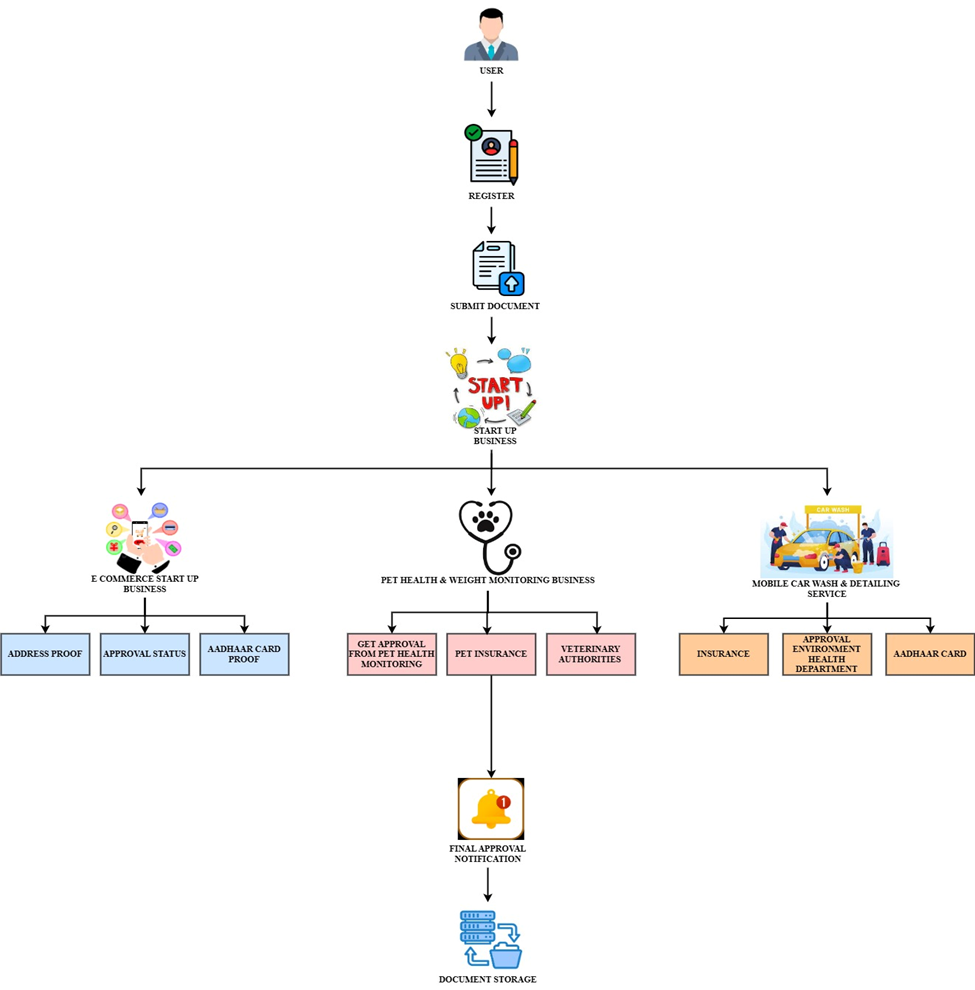
**6.7 Performance Testing**

The system's performance under high user load is evaluated to ensure stability and responsiveness. Load testing is performed to simulate concurrent users registering, uploading documents, and accessing dashboards. The aim is to identify performance bottlenecks and ensure the system maintains quick response times during peak traffic periods.

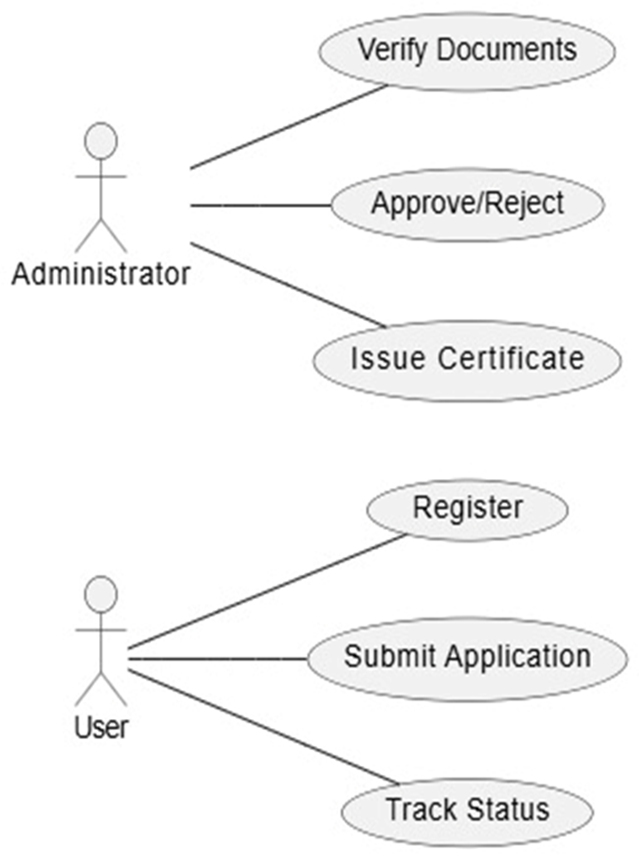
**CHAPTER 7**

**SYSTEM IMPLEMENTATION**

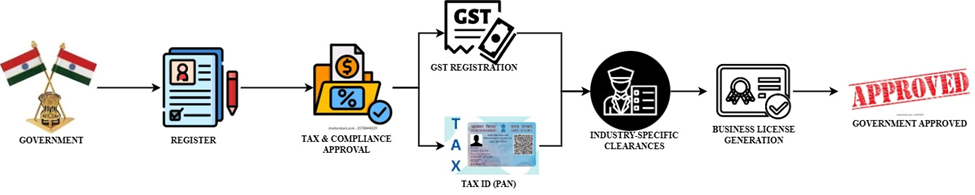
**5.2 SYSTEM ARCHITECTURE DIAGRAM**

****

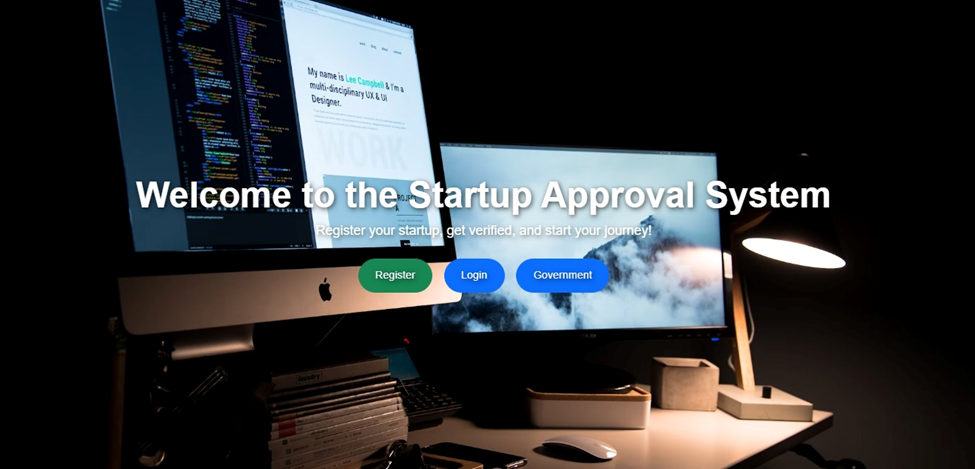
**5.2.1USE CASE DIAGRAM**

****

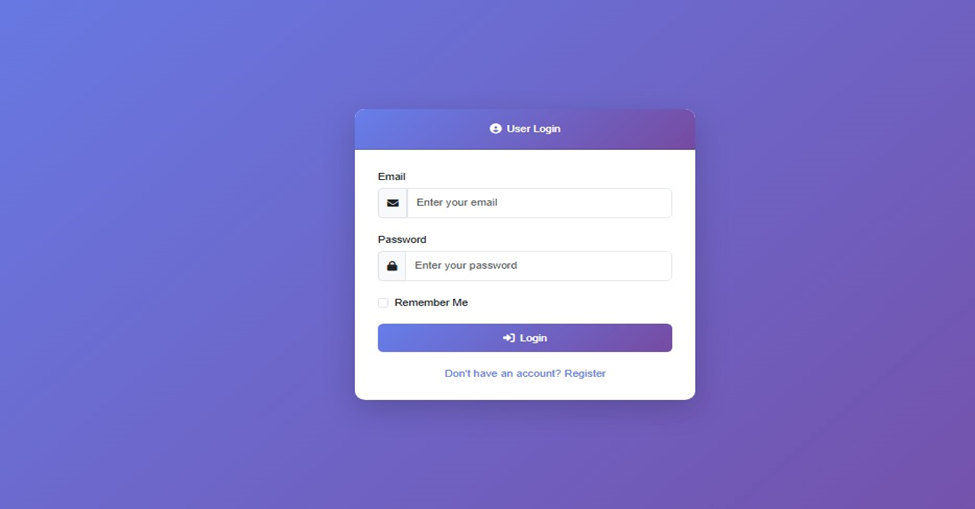
**5.2.2 GOVERNMENT APPROVAL PROCESS**

****

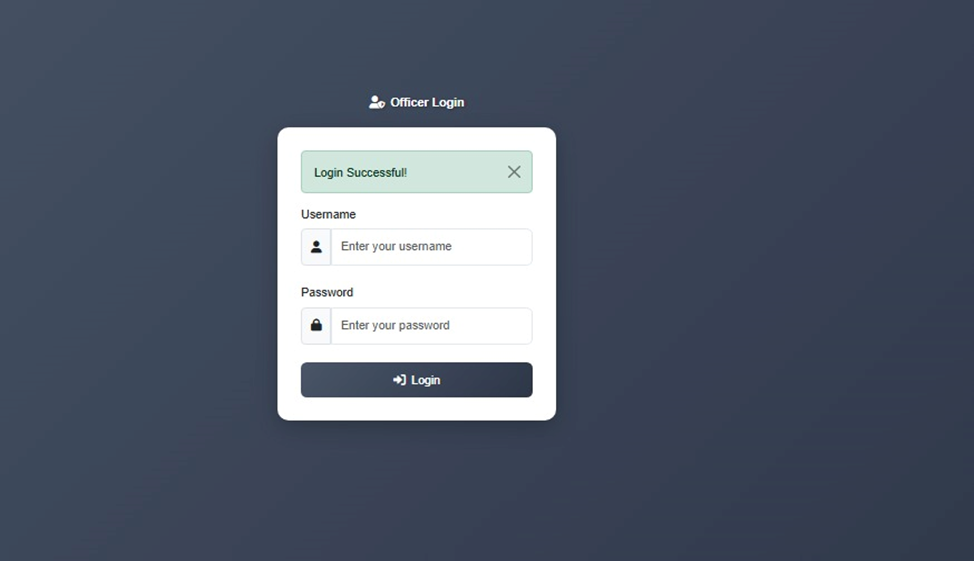
**7.1.1 Login Page**

****

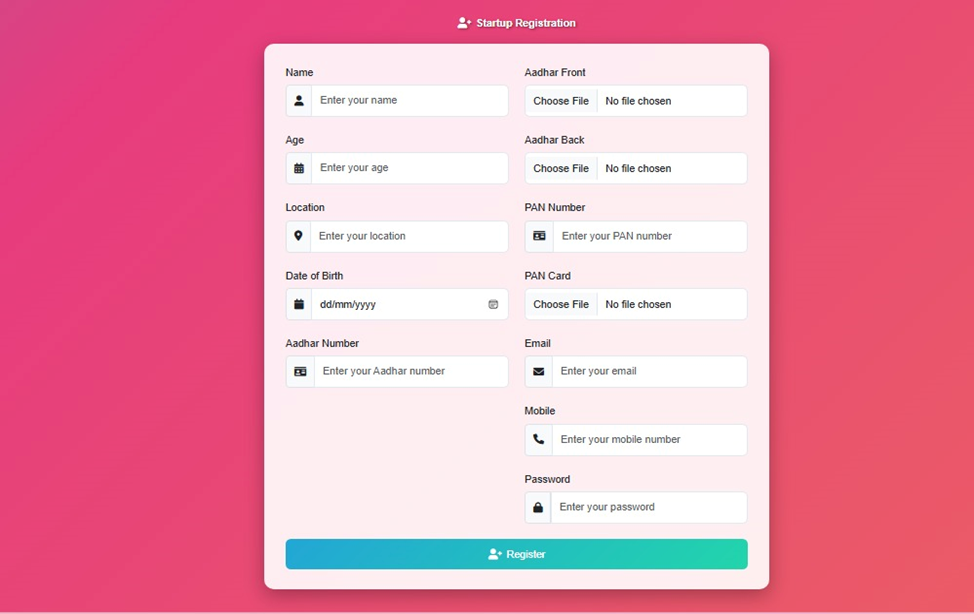
**7.1.2 User Login**



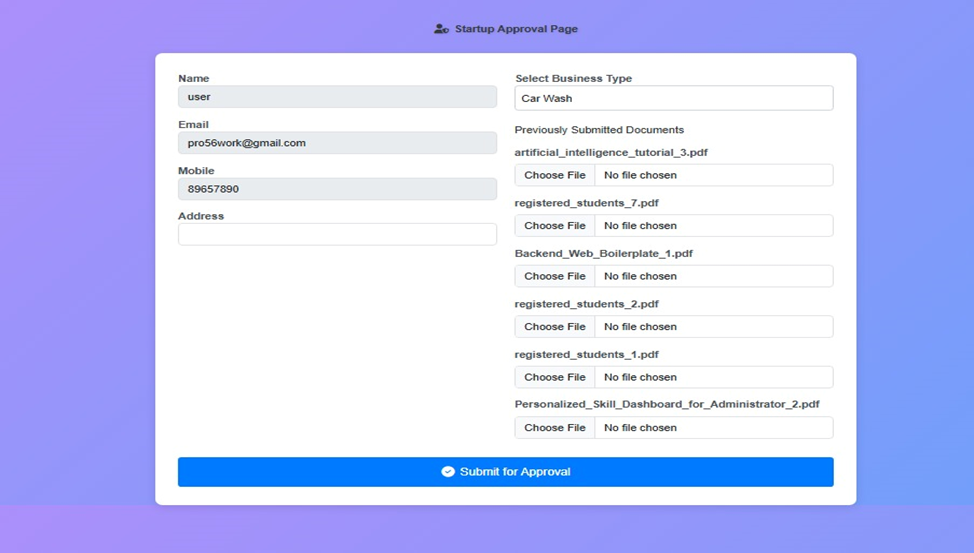
7.1.3 Admin Login



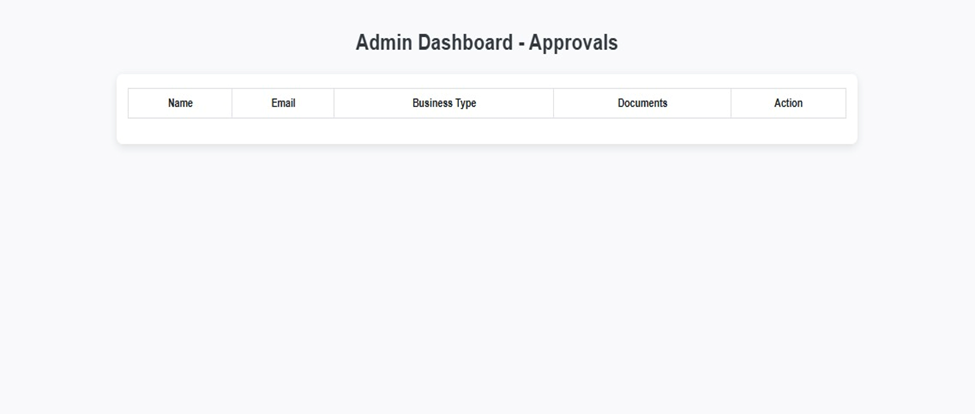
7.1.4 Registration Form



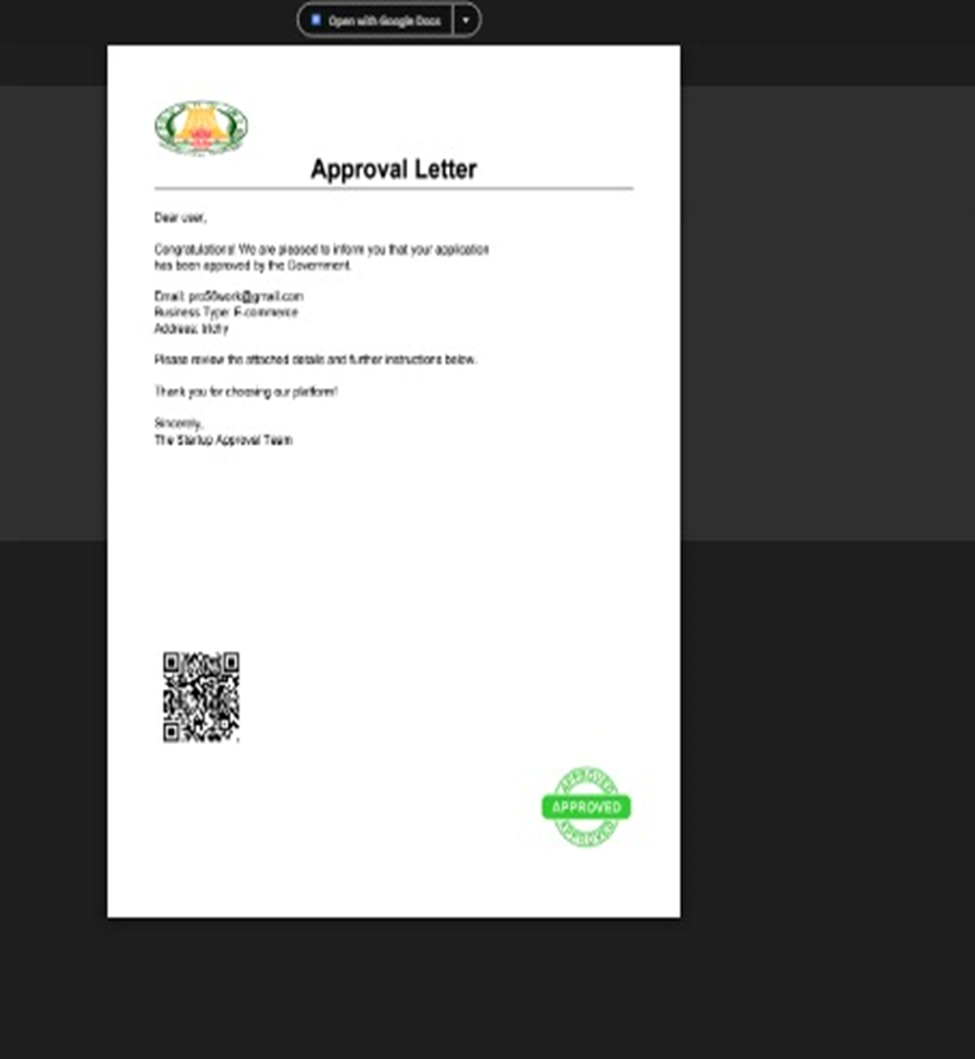
7.1.5 Registration Form



7.1.6 Admin Dashboard



7.1.7 Approval Letter



**CHAPTER 8**

**SOURCE CODE**

**8.1 SOURCE CODE**

import os

import json

import random

import smtplib

from flask import Flask, render\_template, request, redirect, url\_for, session, flash

from flask\_sqlalchemy import SQLAlchemy

from flask\_bcrypt import Bcrypt

from flask\_mail import Mail, Message

from flask\_login import LoginManager, UserMixin, login\_user, login\_required, logout\_user, current\_user

from werkzeug.utils import secure\_filename

app = Flask(\_name\_)

app.secret\_key = "your\_secret\_key"

# Database Config

app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:///database.db"

app.config["SQLALCHEMY\_TRACK\_MODIFICATIONS"] = False

db = SQLAlchemy(app)

bcrypt = Bcrypt(app)

# Mail Configuration

app.config["MAIL\_SERVER"] = "smtp.gmail.com"

app.config["MAIL\_PORT"] = 587

app.config["MAIL\_USE\_TLS"] = True

app.config["MAIL\_USERNAME"] = "sanjay2374official@gmail.com" # Update with your email

app.config["MAIL\_PASSWORD"] = "ujrfkuzuikbzlkkw" # Use an app password

mail = Mail(app)

# Upload Folder

UPLOAD\_FOLDER = "static/uploads/"

os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)

app.config["UPLOAD\_FOLDER"] = UPLOAD\_FOLDER

# Login Manager

login\_manager = LoginManager(app)

login\_manager.login\_view = "login"

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

# User Model

import json

from flask\_login import UserMixin

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

age = db.Column(db.Integer, nullable=False)

location = db.Column(db.String(200), nullable=False)

dob = db.Column(db.String(50), nullable=False)

aadhar\_no = db.Column(db.String(20), unique=True, nullable=False)

aadhar\_front = db.Column(db.String(200), nullable=False)

aadhar\_back = db.Column(db.String(200), nullable=False)

pan\_no = db.Column(db.String(20), unique=True, nullable=False)

pan\_image = db.Column(db.String(200), nullable=False)

email = db.Column(db.String(100), unique=True, nullable=False)

mobile = db.Column(db.String(15), nullable=False)

password = db.Column(db.String(100), nullable=False)

otp = db.Column(db.String(6))

verified = db.Column(db.Boolean, default=False)

address = db.Column(db.String(255), nullable=True)

business\_type = db.Column(db.String(100), nullable=True)

document\_paths = db.Column(db.Text, nullable=True) # Store file paths as JSON

is\_approved = db.Column(db.Boolean, default=False) # Admin approval status

approval\_status = db.Column(db.String(20), default="Pending")

def set\_documents(self, documents):

"""Store documents as a JSON array."""

self.document\_paths = json.dumps(documents)

def get\_documents(self):

"""Return documents as a list.

If stored data isn’t valid JSON, fallback to comma-split.

"""

if self.document\_paths:

try:

return json.loads(self.document\_paths)

except Exception as e:

return self.document\_paths.split(",")

return []

# Function to send OTP email

def send\_otp(email, otp):

msg = Message("OTP Verification", sender=app.config["MAIL\_USERNAME"], recipients=[email])

msg.body = f"Your OTP for verification is: {otp}"

mail.send(msg)

@app.route('/')

def home():

return render\_template('index.html')

@app.route("/register", methods=["GET", "POST"])

def register():

if request.method == "POST":

name = request.form["name"]

age = request.form["age"]

location = request.form["location"]

dob = request.form["dob"]

aadhar\_no = request.form["aadhar\_no"]

pan\_no = request.form["pan\_no"]

email = request.form["email"]

mobile = request.form["mobile"]

password = bcrypt.generate\_password\_hash(request.form["password"]).decode("utf-8")

# File Uploads

aadhar\_front = request.files["aadhar\_front"]

aadhar\_back = request.files["aadhar\_back"]

pan\_image = request.files["pan\_image"]

# Save Files

aadhar\_front\_path = os.path.join(app.config["UPLOAD\_FOLDER"], secure\_filename(aadhar\_front.filename))

aadhar\_back\_path = os.path.join(app.config["UPLOAD\_FOLDER"], secure\_filename(aadhar\_back.filename))

pan\_image\_path = os.path.join(app.config["UPLOAD\_FOLDER"], secure\_filename(pan\_image.filename))

aadhar\_front.save(aadhar\_front\_path)

aadhar\_back.save(aadhar\_back\_path)

pan\_image.save(pan\_image\_path)

# Generate OTP

otp = str(random.randint(100000, 999999))

send\_otp(email, otp)

new\_user = User(

name=name, age=age, location=location, dob=dob,

aadhar\_no=aadhar\_no, aadhar\_front=aadhar\_front\_path, aadhar\_back=aadhar\_back\_path,

pan\_no=pan\_no, pan\_image=pan\_image\_path, email=email, mobile=mobile,

password=password, otp=otp, verified=False

)

db.session.add(new\_user)

db.session.commit()

flash("OTP sent to your email!", "info")

session["email"] = email

return redirect(url\_for("otp\_verify"))

return render\_template("register.html")

@app.route("/otp\_verify", methods=["GET", "POST"])

def otp\_verify():

if request.method == "POST":

email = session.get("email")

user = User.query.filter\_by(email=email).first()

if user and user.otp == request.form["otp"]:

user.verified = True

user.otp = None

db.session.commit()

flash("OTP Verified! You can now log in.", "success")

return redirect(url\_for("login"))

else:

flash("Invalid OTP. Try again.", "danger")

return render\_template("otp\_verify.html")

from flask\_login import LoginManager, login\_user, logout\_user, login\_required, current\_user

# Initialize Flask-Login

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = "login" # Redirects to login page if not logged in

# User Loader for Flask-Login

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

@app.route("/login", methods=["GET", "POST"])

def login():

if request.method == "POST":

email = request.form["email"]

password = request.form["password"]

user = User.query.filter\_by(email=email).first()

if user and bcrypt.check\_password\_hash(user.password, password):

if not user.verified:

flash("Verify your email before logging in!", "warning")

return redirect(url\_for("otp\_verify"))

login\_user(user) # Corrected: Use Flask-Login to log in user

flash("Login Successful!", "success")

return redirect(url\_for("approval\_page"))

flash("Invalid credentials!", "danger")

return render\_template("login.html")

@app.route("/approval\_page", methods=["GET", "POST"])

@login\_required

def approval\_page():

user = current\_user

if request.method == "POST":

business\_type = request.form["business\_type"]

address = request.form["address"]

documents = request.files.getlist("documents")

document\_paths = []

for doc in documents:

if doc.filename != "":

filename = secure\_filename(doc.filename)

# Save file in the configured UPLOAD\_FOLDER (e.g., static/uploads/)

doc\_path = os.path.join(app.config["UPLOAD\_FOLDER"], filename)

doc.save(doc\_path)

# Store only the filename (or relative path from static/)

document\_paths.append(filename)

user.business\_type = business\_type

user.address = address

if document\_paths:

user.set\_documents(document\_paths)

else:

user.set\_documents([])

user.is\_approved = False # Pending Admin Approval

try:

db.session.commit()

flash("Application submitted successfully! Waiting for admin approval.", "success")

except Exception as e:

db.session.rollback()

flash("Error saving application. Please try again.", "danger")

print(f"DB Error: {e}")

return redirect(url\_for("approval\_page"))

return render\_template("approval.html", user=user)

@app.route("/officer\_login", methods=["GET", "POST"])

def officer\_login():

if request.method == "POST":

username = request.form["username"]

password = request.form["password"]

# Hardcoded Officer Login

if username == "government" and password == "123456":

session["officer"] = username # Store officer in session

session["role"] = "government" # ✅ Ensure role is also set

print("Session after login:", session) # Debugging

flash("Officer Login Successful!", "success")

return redirect(url\_for("admin\_dashboard"))

flash("Invalid Officer Credentials!", "danger")

return render\_template("officer\_login.html")

@app.route("/admin\_dashboard")

@login\_required

def admin\_dashboard():

if "officer" not in session or session.get("officer") != "government":

flash("Access Denied! Officer Login Required.", "danger")

return redirect(url\_for("home"))

users = User.query.filter\_by(approval\_status="Pending").all() # ✅ This now works

return render\_template("admin\_dashboard.html", users=users)

from io import BytesIO

from reportlab.pdfgen import canvas

from io import BytesIO

from reportlab.lib.pagesizes import letter

from reportlab.pdfgen import canvas

from reportlab.lib.units import inch

def generate\_approval\_pdf(user):

"""

Generate a PDF approval letter with government logo and approval seal.

Returns a BytesIO object containing the PDF.

"""

buffer = BytesIO()

c = canvas.Canvas(buffer, pagesize=letter)

width, height = letter

# Draw Government Logo (Ensure file exists at this location)

logo\_path = "static/images/government\_logo.png"

try:

# Draw the logo at the top-left corner

c.drawImage(logo\_path, 50, height - 100, width=100, height=50, mask='auto')

except Exception as e:

print("Error drawing government logo:", e)

# Header: Title of the Letter centered

c.setFont("Helvetica-Bold", 24)

c.drawCentredString(width / 2, height - 120, "Approval Letter")

c.line(50, height - 130, width - 50, height - 130)

# Body text

c.setFont("Helvetica", 12)

textobject = c.beginText(50, height - 160)

lines = [

f"Dear {user.name},",

"",

"Congratulations! We are pleased to inform you that your application",

"has been approved by the Government.",

"",

f"Email: {user.email}",

f"Business Type: {user.business\_type}",

f"Address: {user.address}",

"",

"Please review the attached details and further instructions below.",

"",

"Thank you for choosing our platform!",

"",

"Sincerely,",

"The Startup Approval Team"

]

for line in lines:

textobject.textLine(line)

c.drawText(textobject)

# Draw Approval Seal at the bottom-right

seal\_path = "static/images/approval\_seal.png"

try:

# Draw the seal with a width/height of 100 pixels

c.drawImage(seal\_path, width - 150, 50, width=100, height=100, mask='auto')

except Exception as e:

print("Error drawing approval seal:", e)

c.showPage()

c.save()

buffer.seek(0)

return buffer

@app.route("/approve/<int:user\_id>")

@login\_required

def approve(user\_id):

if "officer" not in session or session.get("officer") != "government":

flash("Access Denied!", "danger")

return redirect(url\_for("home"))

user = User.query.get(user\_id)

if user:

user.approval\_status = "Approved"

user.is\_approved = True

db.session.commit()

flash("User Approved!", "success")

# Generate PDF using user details

pdf\_buffer = generate\_approval\_pdf(user)

# Send approval email with PDF attachment

try:

msg = Message(

subject="Application Approved",

sender=app.config["MAIL\_USERNAME"],

recipients=[user.email]

)

msg.body = (

"Congratulations! Your application has been approved. "

"Please find the attached approval letter for further details."

)

# Attach the generated PDF

msg.attach("approval\_letter.pdf", "application/pdf", pdf\_buffer.read())

mail.send(msg)

print("Approval email sent successfully to:", user.email)

except Exception as e:

print("Error sending approval email:", e)

return redirect(url\_for("admin\_dashboard"))

@app.route("/reject/<int:user\_id>")

@login\_required

def reject(user\_id):

if "officer" not in session or session.get("officer") != "government":

flash("Access Denied!", "danger")

return redirect(url\_for("home"))

user = User.query.get(user\_id)

if user:

user.approval\_status = "Rejected"

db.session.commit()

flash("User Rejected!", "danger")

# Send rejection email with a rejection message

try:

msg = Message(

subject="Application Rejected",

sender=app.config["MAIL\_USERNAME"],

recipients=[user.email]

)

msg.body = "We regret to inform you that your application has been rejected."

mail.send(msg)

print("Rejection email sent successfully to:", user.email)

except Exception as e:

print("Error sending rejection email:", e)

return redirect(url\_for("admin\_dashboard"))

@app.route("/user\_dashboard")

@login\_required

def user\_dashboard():

user = User.query.get(session.get("user\_id"))

return render\_template("user\_dashboard.html", user=user)

@app.route("/logout")

def logout():

session.pop("user", None)

flash("Logged out successfully!", "info")

return redirect(url\_for("login"))

if \_name\_ == "\_main\_":

with app.app\_context():

db.create\_all()

app.run(debug=True)

**8.2 RESULT AND DISCUSSION**

The development and implementation of the Smart Approval System yielded highly promising results in terms of functionality, efficiency, and user experience. The system was tested in a simulated environment with dummy data sets representing a range of business registration scenarios. Through structured testing across all modules—including user registration, document submission, government approval handling, and real-time tracking—the platform demonstrated a marked improvement in speed, transparency, and reliability compared to traditional methods.One of the most significant outcomes was the reduction in processing time for startup approvals. By integrating APIs from regulatory bodies such as GST, ROC, and MSME, the system was able to automate compliance checks that typically take several days when done manually. In the simulation, approval processes that usually spanned a week were shortened to a matter of hours, depending on the completeness and accuracy of the submitted documents.The user-friendly interface was another strong highlight. Entrepreneurs could easily navigate the system, aided by step-by-step instructions and visual progress indicators. The guided workflow minimized confusion and reduced the chances of errors in form filling or document uploads. This significantly improved the confidence of first-time users, especially those unfamiliar with bureaucratic procedures.Real-time application tracking proved invaluable in reducing uncertainty and improving user engagement. Users received instant updates about their application status, requests for additional documents, or final approvals. This eliminated the need for repeated follow-ups or physical visits to government offices, saving both time and resources.The role-based access control mechanism effectively ensured secure access and task segregation among different user types—entrepreneurs, consultants, and government officials. Each user interacted only with relevant modules and data, which not only protected sensitive information but also streamlined user experience based on their role. The system also logged all interactions, offering a transparent audit trail that can assist in conflict resolution or policy reviews.From a technical standpoint, the Python-based backend performed reliably under load conditions. During performance testing, the platform comfortably handled concurrent users without significant latency or crashes, which is crucial for scalability once the system is publicly deployed.Security-wise, encryption protocols were implemented for user credentials, documents, and sensitive data, ensuring compliance with data protection norms. No major vulnerabilities were detected during penetration testing, which strengthens the system's readiness for real-world implementation.In discussions with early users and stakeholders, feedback was largely positive. Entrepreneurs appreciated the digital-first approach and ease of use, while government officials reported that the system simplified the application review process, thanks to organized digital submissions and automated verification steps.In conclusion, the Smart Approval System successfully addresses key challenges in the business startup process, such as delays, lack of visibility, and manual inefficiencies. By combining automation, integration, and secure workflows, the system presents a scalable, transparent, and user-friendly solution for future-ready governance in the startup ecosystem.

**CHAPTER 9**

**CONCLUSION**

**9.1 CONCLUSION**

Starting a new business in today’s fast-paced economy demands not only innovative ideas but also a seamless and supportive regulatory environment. However, the traditional startup approval process is often marred by complexity, lack of transparency, and delays due to manual procedures spread across multiple government bodies. The Smart Approval System addresses these longstanding challenges by providing a digital-first, integrated, and automated platform designed to accelerate and streamline the business registration journey.This proposed system stands as a transformative solution by integrating key government portals such as GST, ROC, and MSME into one unified interface. Entrepreneurs are no longer burdened by redundant paperwork or the need to navigate different departments manually. Instead, the system offers a step-by-step guided workflow that allows users to register, upload required documents, make payments, and track their application in real-time—all from a single, user-friendly platform.The backend of the system, built using secure Python-based architecture, ensures data protection, reliable performance, and ease of maintenance. Its role-based access model enhances security and clearly defines the responsibilities of each stakeholder—whether they are entrepreneurs, government officials, or consultants. Moreover, blockchain-based authentication and secure cloud storage further safeguard the integrity and confidentiality of user data.Throughout testing, the system has proven to significantly reduce approval timelines, improve communication among departments, and minimize human error. These improvements not only save time and resources but also foster trust in the process, encouraging more aspiring entrepreneurs to take the leap into starting their own businesses.In summary, the Smart Approval System aligns with the vision of digital governance and ease of doing business. It empowers entrepreneurs with real-time visibility, reduces bureaucratic overhead for government bodies, and ensures faster, error-free approvals through intelligent automation. As it continues to evolve, this platform has the potential to be scaled nationally, offering a standardized, efficient, and entrepreneur-friendly startup ecosystem. With technology-driven innovations like this, the path to becoming a business owner in India can be made smoother, more accessible, and future-ready.

**CHAPTER 10**

**REFERENCE**

**10.1 Reference**

1. X. Liang, S. Yu, D. Huang, and R. Lu, "Attribute-based proxy re-encryption with optimized revocation in cloud systems," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 121–135, Jan.–Feb. 2023.
2. S. Wang, J. Wang, and Y. Zhang, "Blockchain-enabled access control for secure cross-cloud data sharing," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 552–564, Jan. 2024.
3. R. Zhang, Y. Xie, and Z. Qin, "An access control scheme for cloud storage based on ABE and blockchain," *J. Cloud Comput.*, vol. 12, no. 1, pp. 1–13, Mar. 2023.
4. H. Zhao, F. Wu, and X. Chen, "Efficient and verifiable ABE-based proxy re-encryption using blockchain," *Information*, vol. 14, no. 5, pp. 281–295, May 2023.
5. N. Sharma and M. Mishra, "Searchable attribute-based proxy re-encryption for secure cloud systems," *SN Comput. Sci.*, vol. 5, no. 2, pp. 1–15, Feb. 2024.
6. S. Ma, X. Liu, and M. Wang, "Secure data sharing and keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 401–412, Mar.–Apr. 2023.
7. J. Xu, Q. Tang, and K. Ren, "Smart contract-based ABE with outsourced decryption for cloud security," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 934–948, Feb. 2023.
8. S. Narayan, M. Gagné, and R. Safavi-Naini, "Privacy preserving EHR system using ABE and blockchain," *ACM Asia CCS*, pp. 819–829, 2023.
9. A. Sahai and B. Waters, "Fuzzy identity-based encryption," *LNCS*, vol. 3494, pp. 457–473, Springer, 2005.
10. B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," *Proc. Int. Conf. Public Key Cryptography*, pp. 53–70, 2011.
11. M. Green and G. Ateniese, "Identity-based proxy re-encryption," *Proc. ACNS*, pp. 288–306, 2007.
12. Y. Zhang, J. Chen, and Y. Xiang, "Data security and privacy-preserving in cloud storage: A survey," *IEEE Access*, vol. 10, pp. 2151–2171, Jan. 2022.
13. Q. Tang and W. Susilo, "Attribute-based proxy re-encryption with hidden policies," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 1121–1134, Mar. 2023.
14. L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," *Proc. ACM CCS*, pp. 456–465, 2007.
15. Y. Liu, H. Ma, and X. Liu, "Attribute-based encryption with constant-size ciphertext and its applications," *Future Gener. Comput. Syst.*, vol. 123, pp. 1–12, Aug. 2023.
16. T. Wang, P. Wang, and J. Liu, "Revocable proxy re-encryption with accountability for cloud storage," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 67–79, Jan.–Mar. 2023.
17. M. H. Au et al., "Outsourcing decryption of ABE ciphertexts," *ACM Conf. Comput. Commun. Secur.*, pp. 292–302, 2022.
18. F. Li, Y. Zhang, and J. Zhang, "Attribute-based access control with blockchain audit for cloud data sharing," *IEEE Access*, vol. 11, pp. 10541–10552, Feb. 2023.
19. R. Lu, X. Lin, and X. Liang, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2023.
20. C. Liu, J. Chen, and Y. Xiang, "Privacy-preserving data sharing in cloud using revocable ABE and blockchain," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5300–5313, Mar. 2024.