



AI DRIVEN MEDICAL FUNDRAISING VERIFICATION SYSTEM TO DETECT AND PREVENT FRADULENT TREATMENT REQUEST

A Project Report

Submitted by

HARISH KUMAR .N (210221104011)

SARAVANAN.S (210221104029)

SRINATH.M (210221104034)

SUDHAN SANTHOSRAJ.J (210221104035)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

APOLLO ENGINEERING COLLEGE

ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2025

BONAFIDE CERTIFICATE

Certified that this project report “**AI DRIVEN MEDICAL FUNDRAISING VERIFICATION SYSTEM TO DETECT AND PREVENT THE FRADULENT TREATMENT REQUEST**” is the bonafide work of “**HARISH KUMAR (210221104011), SARAVANAN (210221104029), SRINATH (21221104034), SUDHAN SANTHOSRAJ (210221104035)**” who carried out the project work under my supervision.

Supervisor

Mrs. N. Krishnaveni, M.E.,

Assistant Professor

Department of CSE

Apollo Engineering College

Kancheepuram - 602 105

Head of the department

Mrs. R. Sudha, M.Tech.,

HOD and Assistant Professor

Department of CSE

Apollo Engineering College

Kancheepuram - 602 105

Submitted to Project and Viva Examination held on

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We express our profound gratitude to our Chairman, Vice Chairman and Secretary of our college, for providing us the necessary facilities to carry out the project work.

We would like to express my sincere thanks to **Dr. R. Manoharan, M.Tech., Ph.D.**, Principal of our college for their constant support and encouragement towards completion of this project.

We like to express our heartfelt thanks to **Mrs. R. Sudha, M.Tech.**, Head of the department, Professor and internal guide, Department of Computer Science and Engineering and project coordinator for help in completing the project.

We would like to express my heartfelt thanks to the department staff and family members for their continuous support and encouragement.

ABSTRACT

Medical Fund refers to financial assistance provided to individuals or families in need of support for medical treatments, surgeries, or emergencies. Such initiatives often rely on crowdfunding platforms, social media campaigns, or charitable organizations to raise funds. However, the rise of Medical Fund Fraud has become a significant challenge, where fraudsters fabricate treatment documents or bills to solicit donations deceitfully, undermining the trust of donors and affecting genuine beneficiaries. Existing fraud detection systems are often manual or semi-automated, requiring human verification of submitted documents. These processes are time-consuming, error-prone, and struggle to identify sophisticated fraudulent attempts. The lack of comprehensive and automated mechanisms further exacerbates the issue, leading to donor skepticism and reduced willingness to contribute. This project offers an AI-driven approach to detect and block fraudulent medical fund requests. It incorporates advanced YOLOv8 for detecting text regions in uploaded treatment bills and PaddleOCR for extracting and recognizing the text. The extracted information—such as hospital names, patient details, and treatment costs—is verified against a trusted hospital dataset using the Fuzzy Matching Algorithm, which measures the similarity between extracted text and stored records to identify discrepancies effectively. By automating text detection, recognition, and pattern matching, this system ensures accurate verification of medical fund requests, safeguarding donor contributions and fostering trust in medical crowdfunding efforts.

TABLE OF CONTENTS

S. NO	TITLE	PAGE NO
	ABSTRACT	1
1	INTRODUCTION	5
	1.1. Overview	5
	1.2. Problem Statement	6
	1.3. Deep Learning	7
	1.4. Aim and Objective	9
	1.5. Scope of the Project	9
2	LITERATURE SURVEY	10
3	SYSTEM ANALYSIS	13
	3.1. Existing System	13
	3.1.1. Disadvantage	15
	3.2. Proposed System	15
	3.2.1. Advantages	16
4	SYSTEM REQUIREMENT	17
	4.1. Hardware Requirement	17
	4.2. Software Requirement	17
	4.3. Software Description	17
5	SYSTEM DESIGN	25
	5.1. System Architecture Overview	25
	5.2. User Interaction Flow	25
6	SYSTEM ARCHITECTURE	27
	6.1. System Architecture	27
	6.2. Data Flow Diagram	28
	6.4. ER Diagram	30
7	MODULES DESCRIPTION	31
	7.1. System Implementation	31
	7.2. System Description	32
	7.3. System Flow	33
	7.4. Modules Description	34

8	SYSTEM TESTING	41
	8.1. Testing Strategies	41
	8.2. Test Cases	42
	8.3. Test Report	44
9	CONCLUSION	46
10	FUTURE ENHANCEMENT	47
	APPENDIX	48
	A. SOURCE CODE	48
	B. SCREENSHOTS	66
	BIBLIOGRAPHY	74
	Journal reference	74
	Book reference	75
	Web reference	76

LIST OF FIGURES

S.NO	FIGURES	PAGE NO
6.1	Architecture Diagram	27
6.2.1	Data Flow Diagram Level 0	28
6.2.2	Data flow Diagram Level 1	28
6.2.3	Data flow diagram Level 2	29
6.3	ER Diagram	30

LIST OF ABBREVIATION

S.NO	ABBREVIATION	EXPANSION
1	OCR	Optical Character Recognition
2	ML	Machine Learning
3	AI	Artificial Intelligence
4	UI	User Interface
5	UX	User Experience
6	DB	Database
7	API	Application Programming Interface
8	ID	Identity Document
9	SMS	Short Message Service
10	OTP	One-Time Password

CHAPTER 1

INTRODUCTION

1.1. OVERVIEW

Medical fundraising is the process of raising financial support for individuals who need funds for medical treatments, surgeries, or ongoing healthcare expenses. It is commonly done through crowdfunding platforms, charitable organizations, NGOs, and community-driven efforts. People create campaigns, share their medical conditions, and request donations from the public, friends, family, or corporate sponsors.



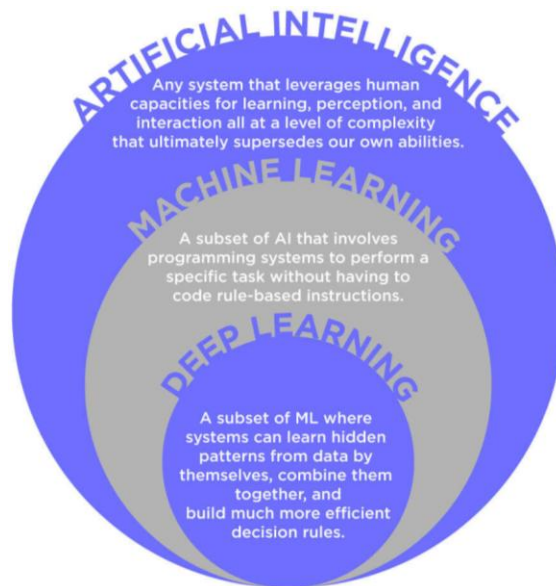
With the rise of online fundraising platforms, individuals can share their medical fund requests through social media, websites, and donation portals. However, the lack of proper verification mechanisms has led to fraudulent activities where scammers create fake medical bills to exploit donors. Hence, advanced fraud detection systems using AI and pattern-matching algorithms are essential to ensure transparency and authenticity in medical fundraising.

1.2. PROBLEMS DEFINITION

The growing trend of medical crowdfunding has been marred by an increase in fraudulent fund requests, where scammers manipulate or fabricate medical treatment bills to deceive potential donors. This issue not only results in financial losses for well-meaning donors but also undermines the integrity of legitimate fundraising efforts. Current systems for verifying medical treatment requests are often manual, time-consuming, and prone to errors, making it challenging to accurately detect fraudulent claims. The lack of an automated, reliable mechanism for fraud detection further exacerbates this problem, leaving donors vulnerable to exploitation. There is an urgent need for an AI-based solution that can automatically verify the authenticity of medical fund requests, detect fraudulent documents, and ensure that donations reach the deserving individuals who genuinely require financial assistance for medical treatment. The consequences of such fraudulent activities are severe and multifaceted. Financially, well-intentioned donors are tricked into diverting their resources away from genuinely needy individuals, resulting in wasted funds that could have otherwise provided life-saving treatment. Ethically and socially, these scams erode public trust in crowdfunding platforms and create a climate of skepticism, making it harder for legitimate patients to gain the support they desperately need. Moreover, repeated incidents of fraud diminish the overall reputation of medical crowdfunding, discouraging future participation and contributing to widespread donor fatigue.

1.3. DEEP LEARNING

Deep learning reflects a subset of machine learning and artificial intelligence. Deep learning is machine learning that draws inspiration from the human brain structure. Its algorithms tend to analyse data within a logical system continuously. This helps deep learning construct patterns that relate to the human brain essential in making conclusions and decisions.

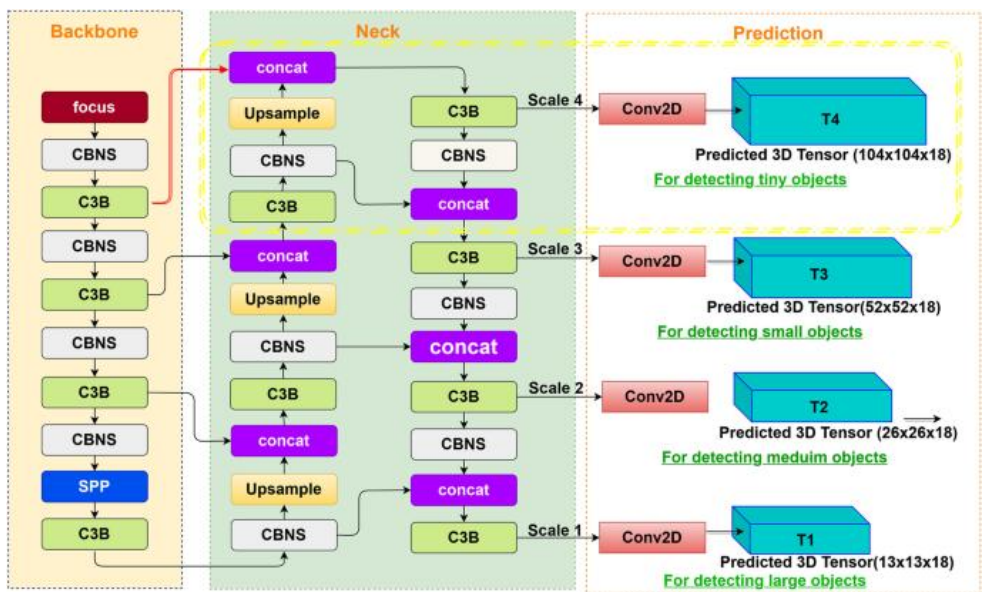


A deep learning model normally refers to a neural network, typically with more than two layers. Deep learning is usually used for computationally dense tasks like computer vision (images) and natural language processing. In technical terms, deep learning uses something called "neural networks," which are inspired by the human brain. These networks consist of layers of interconnected nodes that process information. The more layers, the "deeper" the network, allowing it to learn more complex features and perform more sophisticated tasks.

- **Handling unstructured data:** Models trained on structured data can easily learn from unstructured data, which reduces time and resources in standardizing data sets.
- **Handling large data:** Due to the introduction of graphics processing units (GPUs), deep learning models can process large amounts of data with lightning speed.
- **High Accuracy:** Deep learning models provide the most accurate results in computer visions, natural language processing (NLP), and audio processing.
- **Pattern Recognition:** Most models require machine learning engineer intervention, but deep learning models can detect all kinds of patterns automatically.

1.3.1. YOLOv8

YOLOv8 represents a significant milestone in computer vision, designed to enhance the efficiency and precision of object detection. Its underlying principle simplifies the process by processing an entire image through a neural network known as a convolutional neural network (CNN) in a single pass, eliminating the need for laboriously inspecting each part of the image individually. This streamlined approach, often referred to as “You Only Look Once,” leads to impressive gains in both speed and efficiency.



At its core, YOLOv8 follows a straightforward set of principles. It all starts with an input image, which can be of varying sizes, making it a versatile tool for a wide range of applications. The image then undergoes a process of feature extraction within a neural network. This network, carefully engineered and trained, excels at identifying and highlighting the most relevant features within the image, akin to a detective piecing together crucial clues at a crime scene. YOLOv8 stands out by not just detecting objects at a single scale but by simultaneously spotting them at different scales, ensuring it can identify objects both large and small effectively.

Anchor-Free Detection

YOLOv8 switched to anchor-free detection to improve generalization. The problem with anchor-based detection is that predefined anchor boxes reduce the learning speed for custom datasets. With anchor-free detection, the model directly predicts an object’s mid-point and reduces the number of bounding box predictions. This helps speed up Non-max Suppression (NMS) – a pre-processing step that discards incorrect predictions.

1.4. AIM AND OBJECTIVE

Aim

To develop an AI-powered system for detecting and preventing fraudulent medical fund requests by verifying treatment documents using advanced text detection, recognition, and pattern-matching techniques.

Objectives

- To automate the verification of medical fund requests using AI-based fraud detection.
- To integrate YOLOv8 for detecting text in treatment bills and PaddleOCR for text recognition.
- To apply Fuzzy Matching Algorithm to compare extracted text with authentic hospital records.
- To enhance donor trust by ensuring that only genuine medical fund requests are approved.
- To minimize fraudulent activities in medical crowdfunding by providing a reliable and efficient verification system.

1.5. SCOPE OF THE PROJECT

This project aims to develop a comprehensive AI-driven system for detecting fraudulent medical fund requests. The scope of the project includes:

- **User Roles:** The system will support two main user roles—patients (requesting medical help) and donors (providing financial assistance).
- **User Registration and Login:** Patients and donors will register and log in to the platform securely, providing necessary details for identification.
- **Medical Request Submission:** Patients can submit medical treatment requests by uploading documents, including treatment bills and related proofs.
- **Fraud Detection:** The system will utilize YOLOv8 for text detection in images, PaddleOCR for text recognition, and Fuzzywuzzy for comparing extracted data against verified hospital datasets to detect fraudulent requests.
- **Request Verification:** Only verified, legitimate requests will be visible to donors, while fraudulent ones will be blocked in real-time.
- **Donation and Payment Processing:** Donors will be able to view approved medical fund requests and contribute funds through integrated payment gateways like Razorpay.

CHAPTER 2

LITERATURE SURVEY

2.1. IHSAN: A Secure and Transparent Crowdfunding Platform Leveraging Comprehensive Decentralized Technologies

(Omar Khalid Alia; Duaa Mohammad Suleiman, 2024)

Methodology: This study introduces IHSAN, a decentralized crowdfunding platform that leverages blockchain technology to address issues of trust, transparency, and accountability in traditional crowdfunding platforms. The research integrates BigchainDB, a decentralized database technology, to enhance scalability and query efficiency while maintaining security.

Algorithms: The platform utilizes blockchain for decentralized control and immutable transactions, ensuring transparent fund allocation.

Findings: IHSAN significantly reduces platform fees to as low as 1%, compared to the 8% and 2.9% fees charged by traditional crowdfunding platforms. It also achieves faster transaction times (around 2 minutes) compared to 2–5 days or even up to 14 days in conventional platforms. Additionally, the platform provides unrestricted global access, increasing donor confidence and project success rates.

2.2. How Do Project Updates Influence Fundraising on Online Medical Crowdfunding Platforms? Examining the Dynamics of Content Updates

(Yi Wu; Miao Zhang; Yi Shen, 2024)

Methodology: This study analyzes 2334 medical crowdfunding projects on a leading Chinese platform from January 2020 to July 2021. It draws on Aristotle's persuasion theory to examine how different types of project updates influence fundraising success.

Algorithms: The study categorizes project updates into three types: credibility appeals, rational appeals, and emotional appeals, assessing their impact on fundraising using statistical analysis.

Findings: All three types of appeals positively impact fundraising. However, the effectiveness of credibility appeals decreases over time, while rational and emotional appeals become more influential. The study contributes to crowdfunding literature by highlighting the evolving role of project updates in fundraising success.

2.3. A Blockchain-Based Crowdsourcing Loan Platform for Funding Higher Education in Developing Countries

(Kwame Omono Asamoah; Adjei Peter Darko, 2023)

Methodology: This research proposes a blockchain-based decentralized loan platform to help students in developing countries access higher education funding through crowdsourced investments. The platform connects students with investors via registered financial institutions.

Algorithms: The platform leverages blockchain for security, smart contracts for loan agreements, and a decentralized ledger for transparency in financial transactions.

Findings: The proposed system reduces government financial burdens while allowing investors to earn interest on student loans. It also ensures secure transactions and increased accessibility to education funding, thereby helping students complete their studies and harness their potential.

2.4. Crowdfunding Fraud Prevention using Smart Contracts

(Akshay Kumar; Jhanvi Lamba, 2023)

Methodology: This study explores the integration of blockchain and smart contracts to enhance the security of crowdfunding platforms. It highlights the role of decentralized ledgers in improving trust, transaction efficiency, and fraud prevention.

Algorithms: The platform employs smart contracts for automated fund disbursement and blockchain technology for secure, transparent transactions.

Findings: Smart contracts enhance transaction security, improve liquidity, and reduce fraud risks. The study emphasizes that decentralized crowdfunding increases project visibility and benefits all stakeholders, including project creators, backers, and platform administrators.

2.5. Influence of Narrative Strategies on Fundraising Outcome: An Exploratory Study of Online Medical Crowdfunding

(Lu Zheng; Lihui Jiang, 2022)

Methodology: This research examines how different narrative styles in online medical crowdfunding campaigns influence fundraising success. It uses a large dataset from a widely used Chinese platform and applies natural language processing (NLP) techniques.

Algorithms: NLP techniques analyze the textual content of fundraising campaigns to classify narrative styles into optimism, moral mobilization, and financial burden emphasis.

Findings: Campaigns with optimistic narratives achieve higher fundraising success, while those focusing on financial burden or moral mobilization have a negative impact. The study suggests that well-crafted storytelling plays a crucial role in increasing donor engagement.

2.6. The Influence of Social Percolation in Improving Fundraising Strategies of Charity Organizations

(Emi Trepçi; Rainer Hasenauer, 2018)

Methodology: This study investigates how social network factors, particularly word-of-mouth influence, affect donation decisions. It uses surveys and computer simulations to analyze social percolation effects on donation behavior.

Algorithms: The study employs social percolation theory to model donation trends and predict the spread of fundraising initiatives.

Findings: Donations are more likely to occur under social pressure than in private settings. The simulation models provide insights into the effectiveness of different fundraising strategies and help charity organizations improve their donation outreach.

CHAPTER 3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

Traditional methods for verifying medical fund requests have relied on manual processes, third-party validation, and basic rule-based checks. While these methods have been used for years, they are slow, inefficient, and prone to errors. Fraudsters have found ways to bypass these verification steps, leading to a rise in medical fund fraud. Below are some of the commonly used traditional methods and their limitations.

- **Manual Document Verification**

In this method, hospitals, crowdfunding platforms, or donors manually review medical bills, prescriptions, and other supporting documents submitted by patients. They check for inconsistencies, formatting errors, and any signs of forgery. However, this process is highly **time-consuming**, relies on human expertise, and is prone to errors, especially with sophisticated fraudulent documents.

- **Phone or Email Verification**

Donors or fundraising platforms contact hospitals or doctors via phone or email to verify if a particular patient is undergoing treatment. While this method helps confirm details, it depends on the **responsiveness of hospitals** and can lead to delays in fund approvals. Additionally, fraudsters may provide fake contact details to deceive verifiers.

- **Third-Party Trust Networks**

Some crowdfunding platforms rely on NGOs, hospitals, or medical institutions to validate medical fund requests. These trusted third parties verify patient details before approving funds. While this method adds a layer of security, it lacks scalability and cannot handle a large number of requests efficiently. It also depends on the availability and cooperation of third-party organizations.

- **Pattern-Based Rule Engines**

Basic fraud detection systems use predefined rules to detect suspicious activities. For example, the system may flag requests with similar medical bills, repetitive content, or unusual donation patterns. However, fraudsters can easily modify their tactics to bypass these rules, making this approach less effective against advanced forgery techniques.

- **Community-Based Reporting**

Some platforms allow users or donors to report fraudulent medical fund requests. If multiple reports are received, the request undergoes additional verification or is removed. However, this method relies on user awareness and does not prevent fraud before funds are collected. Many fraudulent campaigns may go unnoticed if no one reports them in time.

Existing Rule-Based Algorithms

Rule-based algorithms are traditional approaches used in fraud detection systems, relying on predefined conditions to identify fraudulent activities. These methods operate on static rules that classify transactions or documents as genuine or fraudulent based on specified criteria. However, while effective in simple cases, rule-based approaches struggle with evolving fraud techniques and sophisticated forgeries. Below are some common rule-based algorithms used in fraud detection, along with their descriptions and limitations:

- **Regular Expression (Regex) Matching**

Regular Expression Matching is widely used to detect predefined patterns in medical invoices, receipts, and supporting documents. It checks for specific structures such as hospital names, patient details, and date formats.

Example: A rule might validate if a date follows the format DD-MM-YYYY or if a hospital name matches an approved list.

- **Keyword-Based Filtering**

This technique scans medical fund requests for specific keywords like "emergency," "ICU," "surgery," or "cancer treatment." If a request lacks these critical terms, it may be flagged as suspicious.

Example: A donation request for medical treatment should contain essential keywords like "hospital," "prescription," or "doctor." If these words are absent, the system raises a red flag.

- **Basic Pattern Matching**

Pattern Matching compares uploaded medical bills with a database of verified invoices to detect inconsistencies in structure, format, or content.

Example: If a fraudulent bill has an incorrect hospital logo, altered font styles, or mismatched invoice numbers, the system identifies the discrepancies.

3.1.1. Disadvantages

- Time-consuming manual verification delays fund approvals
- Prone to human errors, allowing fraudulent documents to go undetected
- Inefficient for handling large volumes of fund requests
- Easily exploitable as fraudsters manipulate documents to bypass verification
- Delayed fund disbursement causes hardship for genuine patients
- Rigid and static system unable to adapt to evolving fraud techniques
- High error rate leading to false positives and false negatives
- Simple modifications can bypass detection mechanisms
- Requires frequent manual updates, limiting scalability
- Struggles with complex data like images and handwritten documents

3.2. PROPOSED SYSTEM

The proposed system aims to enhance the detection and prevention of fraudulent medical fund requests by integrating AI-driven technologies. It automates the verification process, ensuring accuracy and efficiency while minimizing human intervention.

- **AI-Based Fraud Detection**

The system employs YOLOv8 for detecting text regions in medical bills and PaddleOCR for extracting textual information such as hospital names, patient details, and treatment costs. These extracted details are then analyzed to identify potential discrepancies.

- **Pattern Matching for Verification**

To ensure authenticity, the system utilizes the Fuzzy Matching Algorithm, which compares extracted text with a trusted hospital dataset. This method effectively measures similarity and detects inconsistencies in treatment details, preventing fraudulent fund requests.

- **Automated Document Processing**

Unlike traditional manual verification methods, the proposed system automates document processing, significantly reducing the time required for fraud detection. It eliminates human errors and ensures consistency in identifying fake medical fund requests.

- **Secure and Transparent Donation Process**

The system enhances donor confidence by providing a transparent verification process. Only verified medical fund requests are displayed to potential donors, ensuring that contributions reach genuine beneficiaries.

3.2.1. Advantages

- Effectively identifies and blocks fake medical fund requests using AI-driven verification
- Eliminates manual document checking, reducing errors and improving efficiency
- Ensures transparency, encouraging more donors to contribute without fear of fraud
- Quickly analyzes and verifies medical bills, preventing fraudulent requests instantly
- Protects both donors and genuine beneficiaries by verifying fund requests before approval
- Can be expanded to support multiple hospitals, crowdfunding platforms, and NGOs
- Reduces the need for manual fraud detection teams, saving operational costs
- Simplifies the donation and verification process for both patients and donors

CHAPTER 4

SYSTEM SPECIFICATION

4.1. HARDWARE SPECIFICATIONS

- **Processor** : Intel Core i3 or higher
- **RAM** : 4GB or more
- **Storage** : 100GB available disk space
- **Internet** : Stable connection for cloud hosting and data processing

4.2. SOFTWARE SPECIFICATIONS

- **Operating System** : Windows 10/11
- **Programming** : Python 3.x, JavaScript, HTML5, CSS3
- **Frameworks** : Flask
- **Libraries** : PaddleOCR, YOLOv8, FuzzyWuzzy
- **Image Processing** : OpenCV, NumPy, Pandas
- **Payment Gateway** : Razorpay API
- **Database** : MySQL
- **Web Server** : WampServer

4.3. SOFTWARE DESCRIPTION

4.3.1. PYTHON 3.7.4

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.



Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working

in Web Development Domain. Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

Tensor Flow

Tensor Flow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and gives developers the ability to easily build and deploy ML-powered applications.



Tensor Flow provides a collection of workflows with intuitive, high-level APIs for both beginners and experts to create machine learning models in numerous languages. Developers have the option to deploy models on a number of platforms such as on servers, in the cloud, on mobile and edge devices, in browsers, and on many other JavaScript platforms. This enables developers to go from model building and training to deployment much more easily.

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform Tensor Flow. It was developed with a focus on enabling fast experimentation.



Simple. Flexible. Powerful.

- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc. This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine.

Pandas

pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.



Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

The development of pandas introduced into Python many comparable features of working with Data frames that were established in the R programming language. The panda's library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Data frames.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.



NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.



Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Scikit Learn

scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.



Scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Pillow

Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors.



Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new images, etc.

OpenCV

OpenCV is an open-source library for the computer vision. It provides the facility to the machine to recognize the faces or objects.



In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and video.

4.3.2. MySQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.



MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

4.3.3. WAMPSEVER

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your database.



WAMPSEVER is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn't require a payment or subscription.

WampServer also supports the configuration of virtual hosts, enabling users to create and run multiple local websites simultaneously under different domain names. Its comprehensive log access and error-reporting features are invaluable for debugging and troubleshooting during the development process. Additionally, WampServer's quick installation process and seamless integration of components make it an ideal solution for developers seeking a robust local development platform. Whether used for learning, developing new applications, managing local WordPress or CMS installations, or testing before deployment, WampServer provides a stable,

efficient, and flexible environment. Its ability to work offline ensures complete control and safety over web development projects, reducing the risk of unintended changes on live servers. Overall, WampServer plays a crucial role in the workflow of web developers by offering a complete package to design, develop, test, and manage web applications with ease and efficiency directly from a Windows machine.

4.3.4. BOOTSTRAP 4

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.



It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers -Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project.

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

4.3.5. FLASK

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.



Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python. Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

CHAPTER 5

SYSTEM DESIGN

5.1. SYSTEM ARCHITECTURE OVERVIEW

The Medical Fund Verification System follows a modular, layered architecture designed to ensure scalability, security, and efficiency in fraud detection and fund processing. The architecture comprises four main layers: The User Interface Layer, the Application Layer, the Processing and Intelligence Layer, and the Data and Storage Layer.

- **User Interface Layer:** This layer provides access points for different users—patients, donors, and administrators—through a responsive web interface. It supports secure login, fund request submission, donor payments, and notification viewing.
- **Application Layer:** This acts as the control hub, managing user requests, routing them to appropriate modules, and handling business logic. It coordinates between UI, processing, and data services.
- **Processing and Intelligence Layer:** Core functionalities like YOLOv8-based document image detection, OCR for text extraction, machine learning models for fraud classification, and trust score evaluation reside here. It processes uploaded documents to detect anomalies such as fake bills, manipulated entries, or forged stamps.
- **Data and Storage Layer:** This layer handles secure storage of user information, medical documents, transaction logs, and training datasets. It uses a relational database with encrypted access control to protect sensitive data.

The layered architecture promotes separation of concerns, making the system easier to maintain, scale, and secure while enabling real-time, intelligent decision-making in the verification process.

5.2. USER INTERACTION FLOW

The user interaction flow outlines how different users engage with the system from initial login to fund disbursement. It ensures a seamless and secure process for patients, donors, and administrators:

1. User Registration and Login:

- Patients, donors, and admins register and log in through dedicated portals.
- Authentication ensures secure access based on user roles.

2. Fund Request Submission (Patient):

- Patients upload required medical documents such as bills, prescriptions, and hospital proofs.
- The system performs YOLOv8 object detection and OCR to scan and extract document details.

3. **Fraud Detection and Trust Score Generation:**

- Uploaded documents are analyzed using ML algorithms to detect manipulation, duplication, or forgery.
- A trust score is generated to indicate document authenticity.

4. **Admin Verification and Approval:**

- Admins review flagged results and trust scores.
- Verified and legitimate requests are approved for funding.

5. **Donor Interaction:**

- Donors log in to view verified fund requests.
- They can donate securely via integrated payment gateways.
- Donors receive real-time **notifications** and can track the utilization of their contributions.

6. **Notification and Confirmation:**

- Patients are notified upon fund disbursement.
- Donors receive confirmation of successful transactions and impact details.

7. **Data Logging and Record Maintenance:**

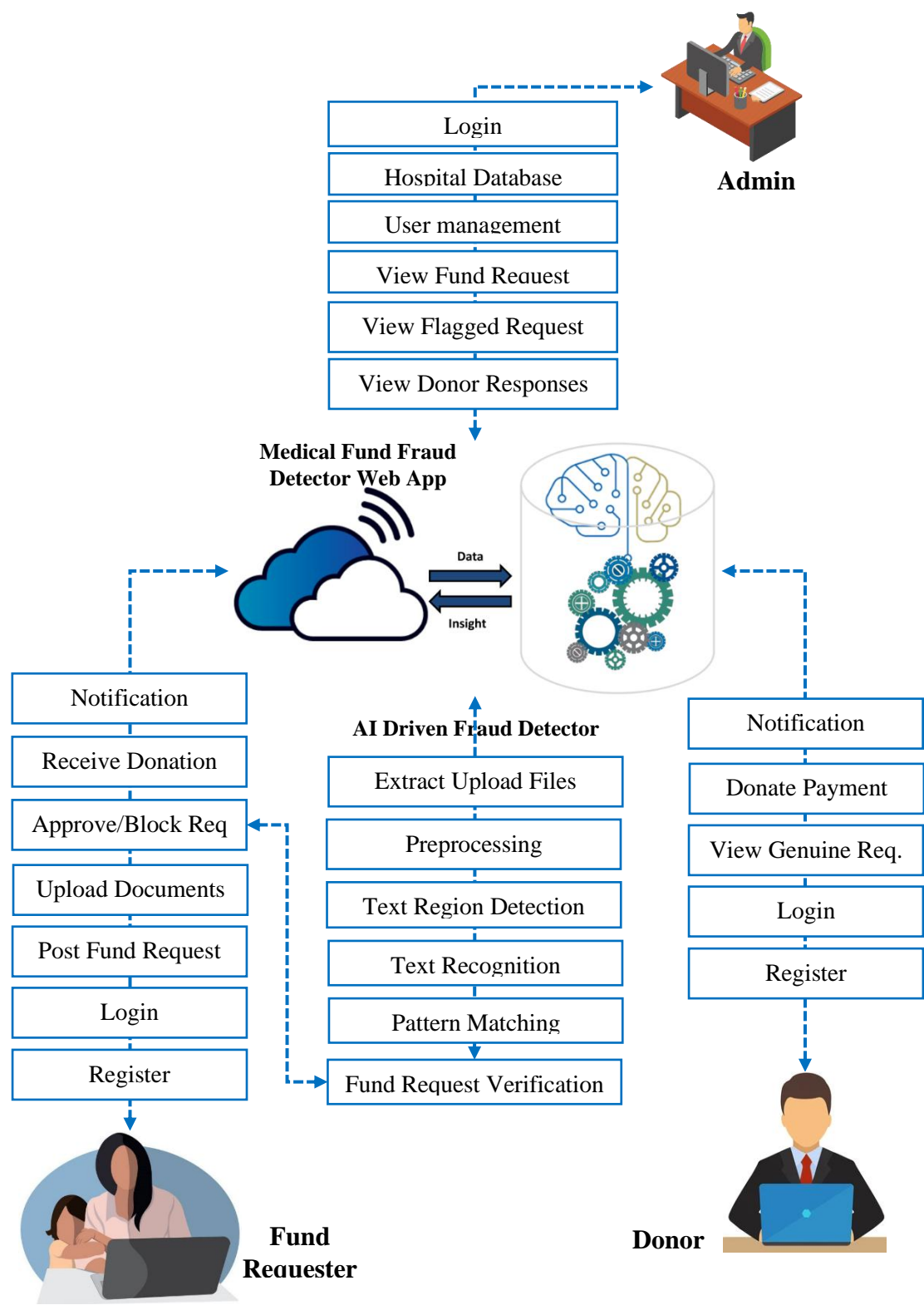
- All activities and decisions are logged in the system for transparency and auditability.
- Documents and transactions are stored securely for future reference.

This structured flow enhances user experience, reduces manual effort, increases security, and builds trust in the medical fund distribution process.

CHAPTER 6

SYSTEM ARCHITECTURE

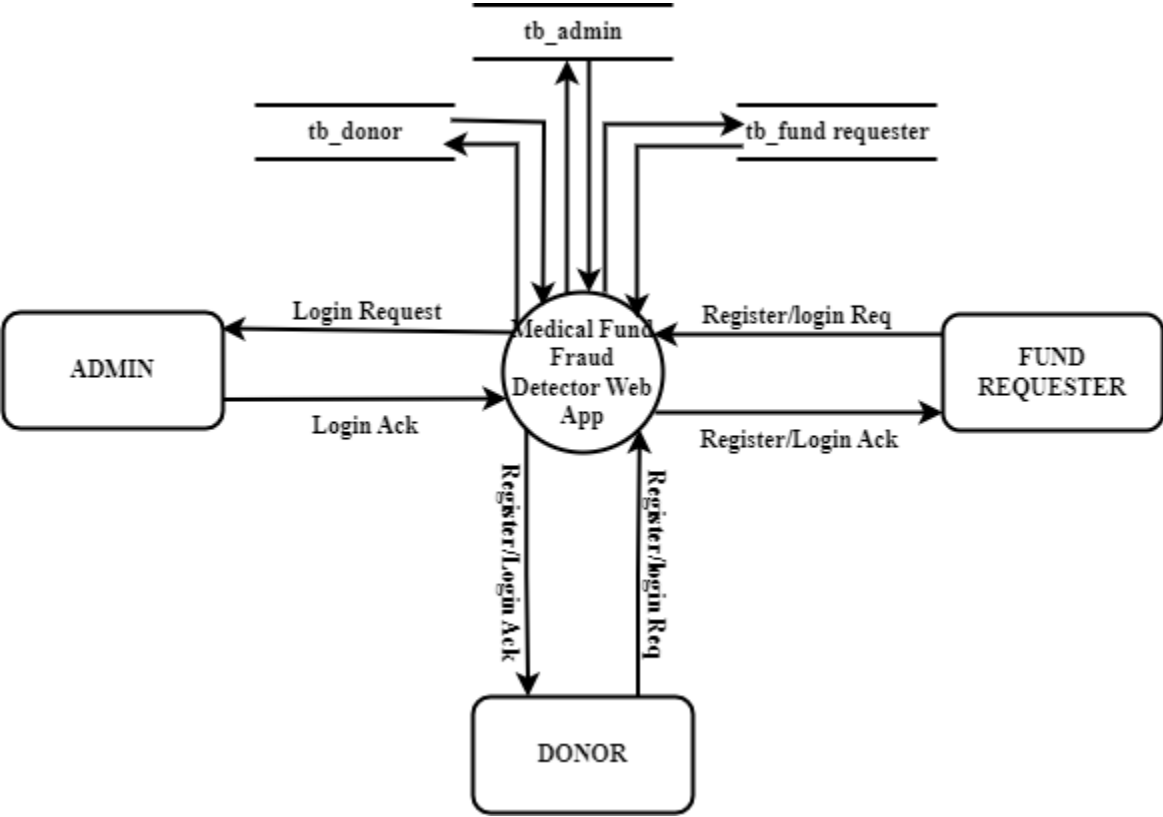
6.1. SYSTEM ARCHITECTURE



4.1. System Implementation

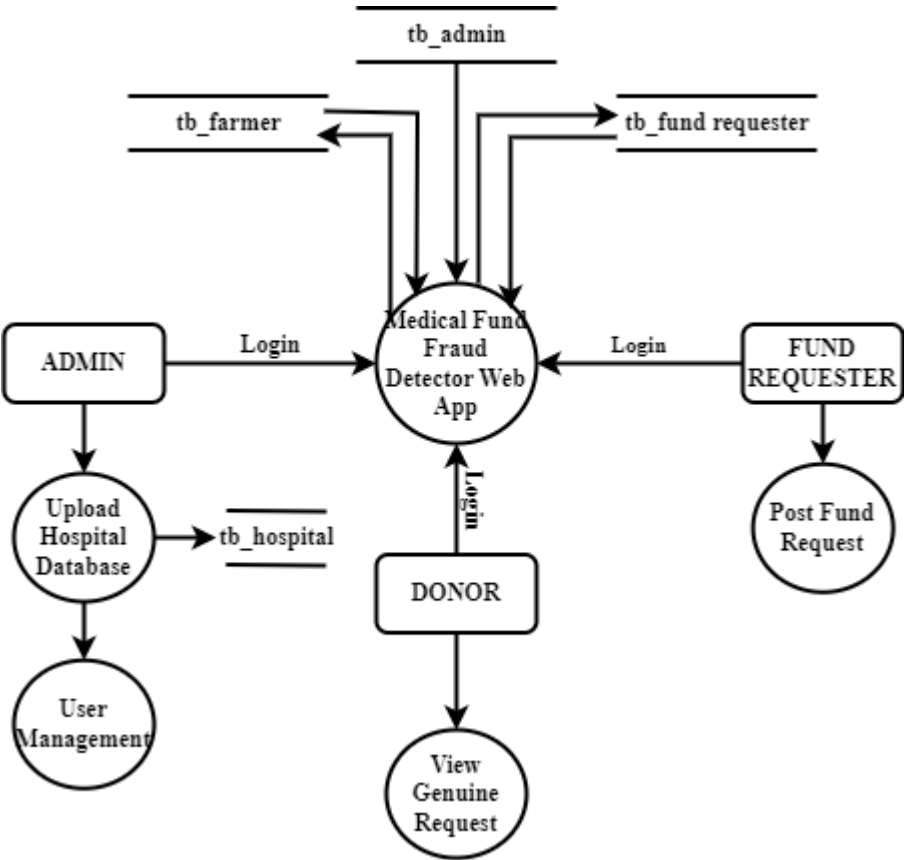
6.2. DATA FLOW DIAGRAM

LEVEL 0



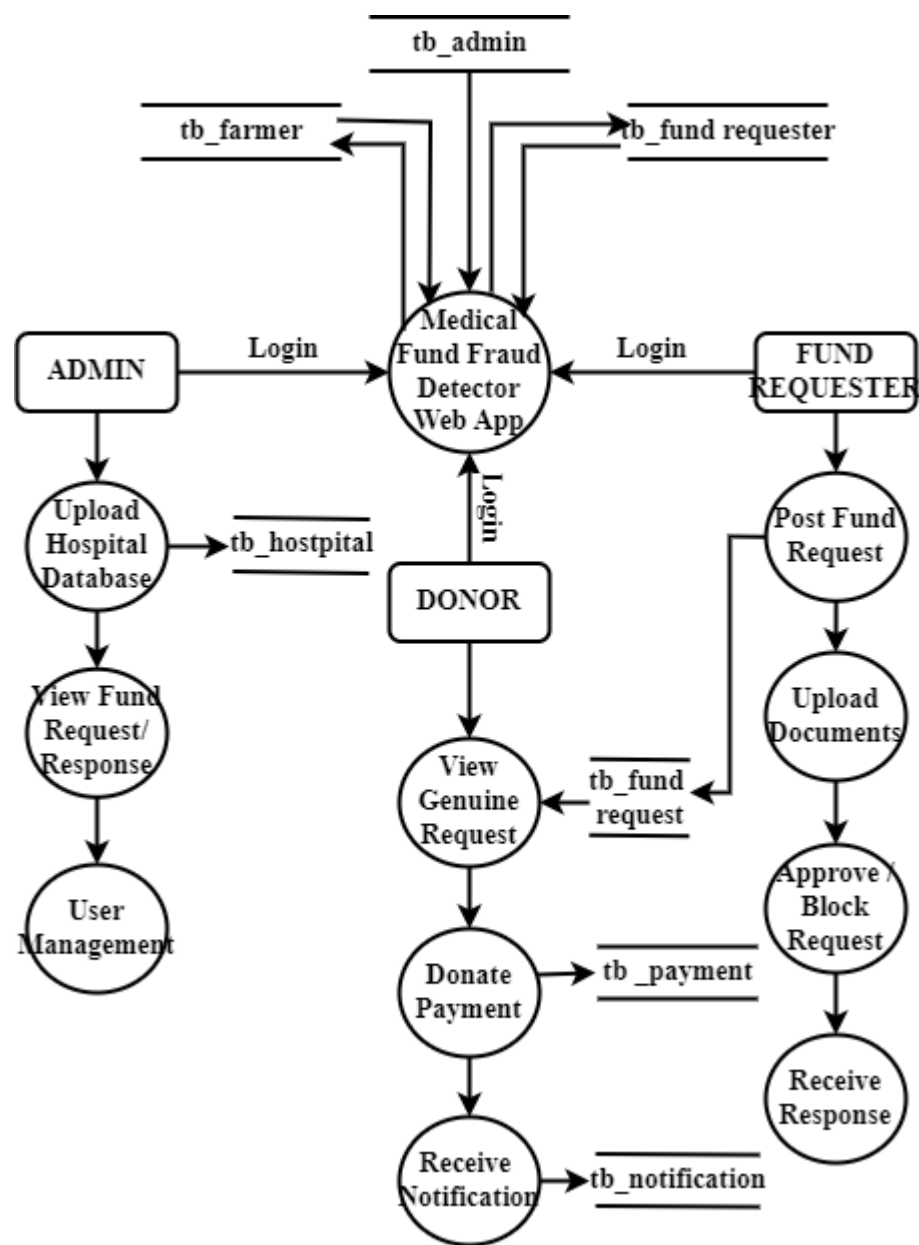
6.2.1. Data Flow Diagram Level 0

LEVEL 1



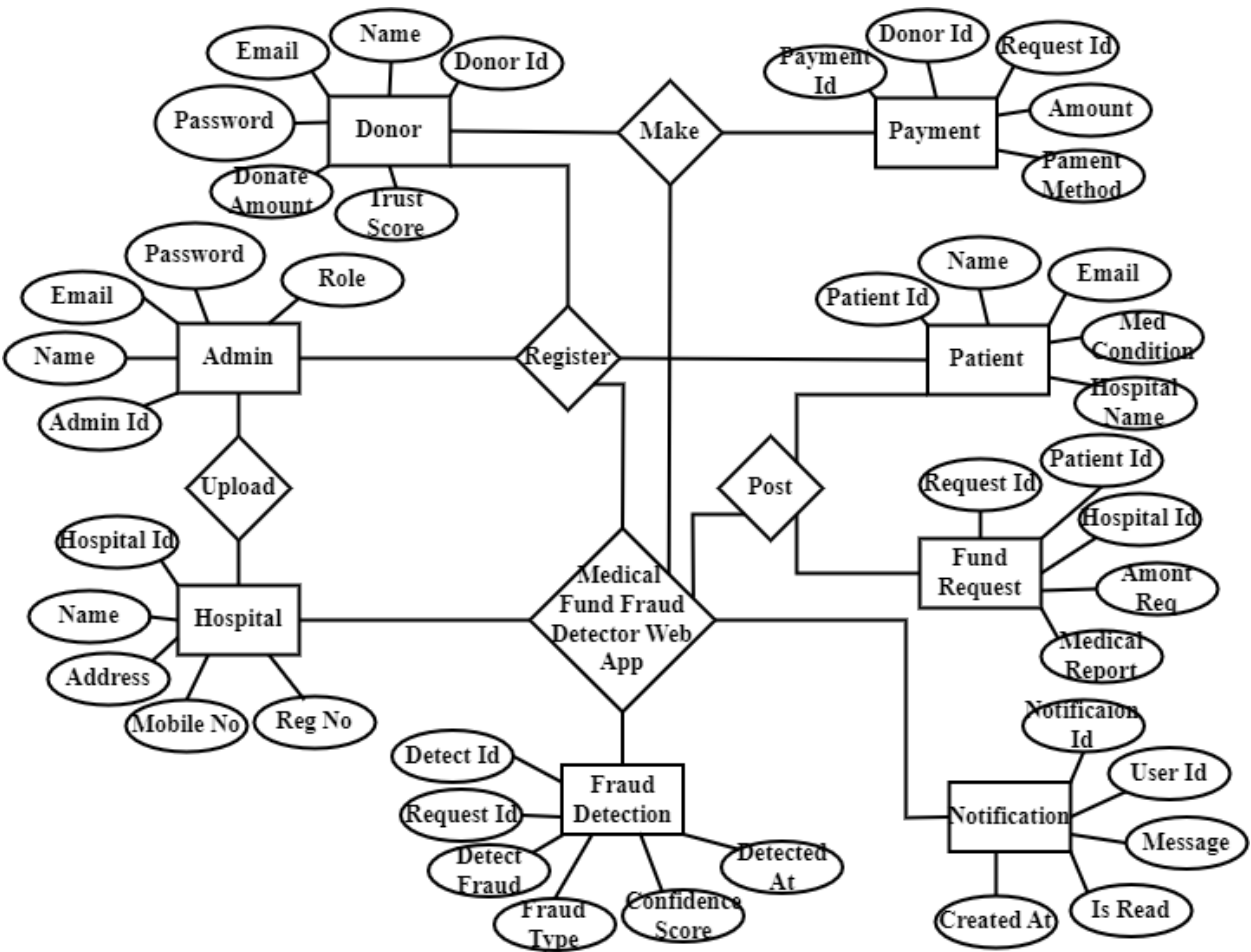
6.2.2. Data Flow Diagram Level 1

LEVEL 2



6.2.3. Data Flow Diagram Level 3

6.3. ER DIAGRAM



6.3. ER Diagram

CHAPTER 7

MODULES DESCRIPTION

7.1. SYSTEM IMPLEMENTATION

The system implementation phase brings together all the components of the Medical Fund Verification System into a cohesive, functional platform. It includes backend logic, frontend development, database integration, machine learning deployment, and secure payment processing.

1. Backend Development

The backend is implemented using Python with Flask, which manages routing, session control, and API handling. Flask provides a lightweight and scalable structure to support modular design and RESTful communication between components.

2. Frontend Design

The user interface is created using HTML, CSS, JavaScript, and Bootstrap. This ensures a responsive and user-friendly design for different stakeholders—patients, donors, and administrators. The UI allows easy uploading of documents, viewing of trust scores, and fund request tracking.

3. Database Integration

MySQL is used to store and manage user data, document records, fund requests, trust scores, and transaction history. The database schema is designed to ensure data normalization, relational consistency, and easy retrieval for validation and reporting.

4. Document Verification Modules

The system incorporates YOLOv8 for object detection, which identifies key visual elements such as hospital logos or stamps. OCR (Optical Character Recognition) is used to extract textual content from submitted medical documents. Both modules work together to detect signs of forgery, duplication, or tampering.

5. Machine Learning-Based Fraud Detection

A trained machine learning model analyzes features from OCR and YOLO results to detect patterns associated with fraudulent submissions. It assigns a Trust Score to each request, which assists in verifying authenticity before funds are released.

6. Role-Based User Access

The platform includes secure login modules for patients, donors, and administrators. Each user type has specific permissions—for example, patients can upload requests, donors can view verified cases, and admins can manage approvals and flag suspicious cases.

7. Payment Gateway Integration

For real-time donations, the system integrates a secure payment gateway. It allows donors to contribute directly to verified cases while maintaining transaction records and sending notifications to recipients and admins.

8. Notifications and Alerts

The implementation includes a notification module that informs users of status updates, fraud detection results, and donor contributions. This ensures transparency and timely action throughout the process.

9. Logging and Auditing

Every activity, including uploads, verifications, fund releases, and user actions, is logged for audit purposes. This improves accountability, aids in tracking misuse, and helps in system maintenance.

This structured implementation provides a secure, efficient, and reliable platform for preventing medical fund fraud and ensuring fair distribution of financial aid. Let me know if you'd like the diagrams or code samples for these modules.

7.2. SYSTEM DESCRIPTION

The Medical Fund Fraud Detection System is designed to ensure transparency and security in medical fund requests by detecting fraudulent activities. This project leverages advanced technologies, including Python, Flask, MySQL, Wampserver, TensorFlow, Pandas, Scikit-Learn, Matplotlib, NumPy, Seaborn, Pillow, OpenCV, and Bootstrap, to analyze and verify fund requests. The system allows patients to submit medical fund requests, donors to contribute securely, and administrators to monitor and review transactions. A core component of this system is the fraud detection module, which utilizes YOLOv8 for text region detection and OCR for text recognition to analyze uploaded medical documents. The extracted data is cross-verified with a hospital database to identify inconsistencies using pattern-matching techniques and fuzzy logic algorithms. Requests are classified as valid, suspicious, or fraudulent, ensuring only genuine cases receive donations. Additionally, the system integrates a secure payment gateway, real-time notifications, and an admin review feature for flagged requests. This approach minimizes fraudulent

transactions, enhances trust between donors and recipients, and streamlines the medical funding process with automated verification and real-time fraud detection.

7.3. SYSTEM FLOW

1. User Registration & Login

The process begins with users (patients, donors, and administrators) registering on the platform. Each user is authenticated through a secure login system that ensures role-based access to system features.

2. Fund Request Submission

Patients can submit medical fund requests by uploading necessary medical documents, including hospital bills, prescriptions, discharge summaries, and ID proof. The system collects this data and stores it securely in the database.

3. Document Preprocessing

Uploaded documents undergo preprocessing steps such as resizing, denoising, and formatting to enhance the accuracy of the OCR and YOLOv8 object detection modules.

4. Forgery Detection using YOLOv8

The YOLOv8 object detection model analyzes the uploaded documents to identify fake stamps, edited texts, or any signs of tampering. The model flags suspicious regions that indicate document forgery.

5. Text Extraction using OCR

Optical Character Recognition (OCR) is applied to extract relevant information from medical documents such as names, dates, hospital names, diagnosis, and bill amounts.

6. Verification using Machine Learning

Extracted data is validated using trained machine learning models that detect manipulation patterns and duplicate entries in the database. The system uses anomaly detection and comparison algorithms to assess document authenticity.

7. Trust Score Generation

Based on the verification results, the system generates a **Trust Score** for each fund request. This score reflects the document authenticity and is used by donors and administrators to make funding decisions.

8. Donor Review & Payment

Verified requests with high trust scores are presented to donors. The system offers a secure payment gateway for processing medical fund donations. Real-time alerts and transaction confirmations are generated.

9. Admin Dashboard & Monitoring

Administrators monitor all activity through a centralized dashboard, manage flagged cases, and generate reports on request status, fraud cases, and fund utilization.

10. Notification System

Patients, donors, and admins receive real-time SMS/email/app notifications regarding status updates such as document verification results, funding approval, or rejection.

7.4. MODULES DESCRIPTION

1. Medical Fund Fraud Detector Web App

The Medical Fund Fraud Detection System is developed to ensure transparency and authenticity in medical fund requests by leveraging Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning techniques. This project is built using Python, Flask, MySQL, WampServer, TensorFlow, Pandas, Scikit-Learn, Matplotlib, NumPy, Seaborn, Pillow, OpenCV, and Bootstrap, making it a robust web-based application. The system integrates fraud detection models with document verification to identify and prevent fraudulent medical fund requests. Flask serves as the backend framework, handling requests, authentication, and communication with the database. MySQL is used to store user information, fund request details, and fraud verification results, while WampServer provides a local environment for testing and database management. TensorFlow, Scikit-Learn, and OpenCV power the fraud detection system by processing medical documents and identifying manipulated elements such as forged signatures, tampered invoices, and duplicate requests. The YOLOv8 object detection model plays a crucial role in detecting fraud patterns in medical bills and hospital stamps. The frontend is designed using Bootstrap, ensuring a responsive and user-friendly interface. Patients can submit fund requests, upload supporting documents, and track verification results. Administrators oversee fraud detection, manage users, and train the AI model. Donors can browse verified fund requests and contribute with confidence. The project also integrates text recognition, pattern matching, and deep learning algorithms to enhance fraud detection accuracy. By combining AI-driven fraud analysis with hospital database verification, the system ensures that only genuine medical cases receive financial aid, minimizing scams and improving transparency in medical fundraising.

2. End User

The End User Module is designed to facilitate interaction between different stakeholders of the system, ensuring seamless and secure fund transactions while preventing fraudulent activities. It consists of three primary user roles: Admin, Patient/Fund Requester, and Fund Donor, each having distinct functionalities.

2.1 Admin

The Admin plays a crucial role in managing the system by overseeing fund requests, fraud detection, and user management. The Admin Dashboard allows administrators to log in securely and perform various tasks such as training the fraud detection model, adding or removing users, and verifying fund requests. The model training process involves feeding the system with datasets of fraudulent and genuine medical documents to enhance accuracy in fraud detection. Admins can review flagged fund requests that the AI system detects as potentially fraudulent and take necessary actions. Additionally, they have access to detailed fund request and response records, ensuring transparency and accountability in financial transactions.

2.2 Patient/Fund Requester

The Patient or Fund Requester is an individual seeking financial assistance for medical treatment. They begin by registering and logging into the system. Once authenticated, they can post a fund request by providing details about their medical condition, required treatment, and uploading relevant medical documents such as bills, prescriptions, and hospital reports. After submission, the system analyzes and verifies the request using fraud detection techniques, and the requester can view the verification results. If the request is approved and donors contribute to the cause, the patient receives payment from the donor directly, ensuring a secure and transparent process.

2.3 Fund Donor

The Fund Donor is an individual or organization willing to provide financial assistance to verified medical fund requests. After registering and logging in, the donor gains access to a list of genuine and verified fund requests, ensuring that their contributions go to legitimate cases. The donor can browse patient requests, review supporting documents, and make an informed decision before donating securely through the system. This ensures a trustworthy donation process, preventing fraudulent transactions and encouraging more donors to participate.

This module establishes a secure and transparent platform where fund requests are authenticated, and donations reach the intended recipients without manipulation or fraud.

3. Hospital Database Integrator

This module is designed to establish a secure and efficient connection between the system and hospital databases. Its primary function is to validate medical fund requests by cross-referencing submitted medical documents, prescriptions, and hospital bills with the respective healthcare providers' records. This module enhances the accuracy and credibility of the verification process, reducing fraudulent claims. The integration process begins by fetching hospital data from authorized healthcare institutions, including details such as patient records, treatment history, hospital registration, and billing information. When a patient submits a fund request, the system automatically retrieves relevant details from the linked hospital database to verify the authenticity of the provided documents. This verification includes checking hospital stamps, doctor signatures, billing authenticity, and prescribed treatments. This module supports multiple hospital databases through secure APIs, encrypted data transfer protocols, and authentication mechanisms. It also implements real-time validation, where discrepancies between submitted documents and hospital records trigger alerts for further review by the admin. Additionally, this module is designed to comply with data privacy regulations, ensuring that sensitive patient information is handled securely and only used for verification purposes.

4. Medical Fund Request

The Medical Fund Request module serves as the core functionality for patients seeking financial assistance for medical treatments. This module enables patients (fund requesters) to submit requests for funding by providing necessary details, including personal information, medical condition, hospital details, treatment cost, and supporting documents such as medical prescriptions, hospital bills, and test reports. Upon submission, the system processes the request through multiple verification layers. The uploaded documents undergo preprocessing, text extraction, and fraud detection mechanisms to ensure authenticity. The module integrates with the Hospital Database Integrator to cross-verify the submitted medical records with the respective hospitals, checking for duplicate requests, forged documents, and manipulated information. Each fund request is assigned a unique reference ID and categorized based on urgency, severity of the medical condition, and required treatment cost. The module also provides real-time tracking, allowing patients to view the status of their requests, whether it is under review, approved, flagged for verification, or rejected. If the request is flagged as suspicious, the system escalates it to the Admin for manual inspection. Once verified, the request is made available

to registered fund donors who can review the case details and contribute financially. If a donation is made, the module ensures secure payment processing and updates the request status accordingly. This module ensures transparency, security, and efficiency, streamlining the process of receiving medical financial aid while mitigating fraudulent activities.

5. Fraud Detection

The Fraud Detection module is a critical component designed to ensure the authenticity of medical fund requests by analyzing submitted documents and identifying fraudulent activities. It employs a combination of deep learning, Optical Character Recognition (OCR), and pattern matching techniques to detect anomalies such as forged medical bills, fake hospital stamps, manipulated prescriptions, and duplicate requests. The module systematically processes uploaded documents through three key submodules:

5.1. Text Region Detection

This step involves identifying the areas within an uploaded document that contain textual information. Using YOLOv8, a powerful object detection model, the system scans and detects text regions in medical bills, prescriptions, and hospital documents. YOLOv8 is trained to recognize specific document attributes such as hospital names, patient details, payment sections, and diagnostic information. By isolating these text regions, the system ensures that only relevant portions of the document are processed for further analysis.

5.2. Text Recognition

Once text regions are detected, the system extracts the textual content using Optical Character Recognition (OCR) techniques, primarily leveraging Tesseract OCR. This step converts printed and handwritten text into machine-readable format, allowing for further processing. The extracted text is preprocessed to remove noise, distortions, and irregular font styles, ensuring higher accuracy in subsequent verification processes. This phase enables the system to retrieve critical data such as patient names, medical conditions, treatment costs, and hospital details for fraud analysis.

5.3. Pattern Matching

The extracted text is then analyzed to detect inconsistencies and fraudulent patterns. This is achieved through pattern matching techniques, where the text is compared against a verified hospital database to identify anomalies. The system employs fuzzy logic algorithms to detect inconsistencies such as:

- Fake or manipulated medical reports where hospital names, patient details, or treatment costs do not match official records.
- Duplicate fund requests submitted under different names but with identical medical details.
- Mismatched hospital stamps and signatures that deviate from authentic hospital documentation.

By combining machine learning, OCR, and database verification, the Fraud Detection module enhances the reliability of the system, ensuring that only genuine medical cases receive financial aid while fraudulent requests are flagged for review.

6. Fund Request Approve /Decline

The Fund Request Verification module is responsible for assessing the authenticity of submitted medical fund requests before they are approved for donor contributions. This module integrates an intelligent decision-making system that categorizes each request based on its credibility, allowing for efficient fraud prevention and transparency. The Decision System is the core of this module, automatically classifying fund requests into three categories: "Valid," "Suspicious," or "Fraud." The classification is based on various factors, including document authenticity, consistency in patient details, verification against the hospital database, and fraud detection algorithms. If a request is deemed valid, it is immediately made available for potential donors. If marked suspicious, it undergoes further review, while requests flagged as fraudulent are rejected from the system. To ensure a thorough verification process, the module includes a Manual Review Option for administrators. This feature allows system admins to manually inspect flagged fund requests, cross-check submitted documents, and make informed decisions regarding approval or rejection. Admins can override automated classifications in cases where the system might have incorrectly flagged a request. Another key feature of this module is the Trust Score Generation, which assigns a fraud probability score to each request. Using machine learning algorithms, the system evaluates factors such as document consistency, past request history, hospital validation checks, and detected anomalies to generate a score. The trust score helps admins and donors assess the reliability of a request before making a donation.

7. Donor Payment Processing

The Donor Payment Processing module is responsible for handling financial transactions between donors and verified fund requesters in a secure and transparent manner. This module ensures that donations reach genuine beneficiaries while maintaining a seamless and fraud-resistant transaction system. A key component of this module is Secure Payment Gateway Integration, which ensures that all financial transactions are conducted safely. The system integrates trusted payment gateways to facilitate donations using multiple payment methods, such as credit/debit cards, online banking, and digital wallets. Advanced encryption techniques and multi-layer authentication mechanisms are implemented to protect donor information from unauthorized access and cyber threats. The module also features Transaction Tracking, which records and logs every donation transaction in a secure database. Each transaction is linked to the respective donor, patient, and fund request ID, ensuring full traceability. Donors can access their donation history, check the status of their contributions, and receive automated receipts for tax or record-keeping purposes. Admins can also monitor transaction logs to detect any irregularities or potential fraud attempts.

To address fraudulent fund requests, the module includes a Refund Mechanism that handles reimbursement procedures in cases where a donation was made to a fraudulently flagged request. If a request is later identified as fraudulent, the system notifies the donor and initiates a refund process. The refund can be automatically processed or manually reviewed by admins, depending on the case. This mechanism builds trust among donors by assuring them that their contributions are safeguarded from misuse.

8. Notification Module

The Notification Module is designed to provide real-time updates and alerts to all users involved in the fund request and donation process. This module ensures that administrators, fund requesters (patients), and donors are kept informed about important events, such as fund request approvals, rejections, flagged fraud cases, and donation payments. A core feature of this module is Real-time Alerts, which instantly notify users of critical updates. When a fund request is submitted, processed, or reviewed, the system sends immediate notifications to both the fund requester and the admin. Donors also receive alerts when their donations are successfully processed or when a refund is issued in cases of fraudulent fund requests.

To maximize communication efficiency, the Email & SMS Integration feature ensures that users receive updates across multiple communication channels. The system automatically generates and sends email confirmations, SMS notifications, and in-app alerts based on the user's preferences. For example, a patient will be notified when their request is approved, a donor will receive a confirmation of their payment, and an admin will be alerted when a suspicious request is flagged.

CHAPTER 8

SYSTEM TESTING

8.1. TESTING STRATEGIES

- **Unit Testing**

Each core module, including fraud detection algorithms, OCR-based document verification, secure payment processing, user authentication, and notification services, is tested in isolation. This ensures that individual functionalities work correctly and any defects are identified and resolved early in development.

- **Integration Testing**

Various modules, such as fundraising request management, fraud detection, document verification, database interactions, role-based access control, and real-time notifications, are tested together. The goal is to validate seamless communication between different components and ensure data consistency across the system.

- **System Testing**

The entire system is evaluated under real-world conditions to verify that all modules function correctly together. This includes testing end-to-end user workflows, multiple fundraising scenarios, large data handling, and overall system behavior under normal and peak loads to ensure reliability.

- **User Acceptance Testing (UAT)**

The system is tested by real users, including patients, donors, and administrators, to validate that the platform meets expectations. Key areas assessed include intuitive navigation, fraud alerts, document verification accuracy, donation tracking, and seamless communication between users.

- **Performance Testing**

The system undergoes extensive testing to assess response time, search efficiency, server load capacity, database performance, and scalability. It ensures that the platform remains efficient even with high traffic, large volumes of fundraising requests, and multiple concurrent users. Performance testing plays a vital role in ensuring software quality, user satisfaction, and business continuity, as slow or unresponsive applications can lead to poor user engagement, revenue loss, and reputational damage. Therefore, integrating performance testing early and consistently throughout the development process is a best practice for building robust, high-performing software systems.

- **Security Testing**

To safeguard sensitive data, security testing checks for vulnerabilities in payment processing, access controls, fraud detection algorithms, encryption methods, and protection against cyber threats like SQL injection and cross-site scripting (XSS). This type of testing involves evaluating the system's ability to protect data, maintain functionality under attack, and ensure that access controls are properly enforced. It encompasses several types, including **vulnerability scanning, penetration testing, risk assessment, security auditing, ethical hacking, and posture assessment**. Security testing focuses on areas such as authentication mechanisms, session management, data encryption, input validation, and secure configuration of servers and databases.

- **Regression Testing**

Each time the system is updated with new fraud detection enhancements, UI modifications, or additional security layers, regression testing ensures that existing functionalities continue to operate correctly. This prevents new updates from introducing bugs or breaking core features.

8.2. TEST CASES

1. Test Case ID: TC001

- **Input:** User enters valid registration details (username, email, password).
- **Expected Result:** System registers the user successfully.
- **Actual Result:** User registration successful.
- **Status:** Pass

2. Test Case ID: TC002

- **Input:** User logs in with correct credentials.
- **Expected Result:** System grants access to the user dashboard.
- **Actual Result:** User successfully logged in; dashboard accessible.
- **Status:** Pass

3. Test Case ID: TC003

- **Input:** Admin logs in with correct credentials.
- **Expected Result:** System allows access to the admin interface.
- **Actual Result:** Admin successfully logged in; admin dashboard accessible.
- **Status:** Pass

4. Test Case ID: TC004

- **Input:** User uploads a valid medical bill image.
- **Expected Result:** System accepts the file and stores it for processing.
- **Actual Result:** File uploaded and stored successfully.
- **Status:** Pass

5. Test Case ID: TC005

- **Input:** User uploads an unsupported file format.
- **Expected Result:** System rejects the file and displays an error message.
- **Actual Result:** Error message displayed: "Invalid file format."
- **Status:** Pass

6. Test Case ID: TC006

- **Input:** User uploads a clear medical bill image.
- **Expected Result:** System extracts text successfully.
- **Actual Result:** Text extracted without errors.
- **Status:** Pass

7. Test Case ID: TC007

- **Input:** User uploads an original hospital bill.
- **Expected Result:** System marks the bill as genuine.
- **Actual Result:** Bill detected as genuine.
- **Status:** Pass

8. Test Case ID: TC008

- **Input:** User uploads a tampered or forged medical bill.
- **Expected Result:** System detects and flags the bill as fraudulent.
- **Actual Result:** Fraud detected, request flagged.
- **Status:** Pass

9. Test Case ID: TC09

- **Input:** System compares extracted text with hospital records.
- **Expected Result:** System finds a match or reports a mismatch.
- **Actual Result:** Correct matching/mismatch report generated.
- **Status:** Pass

10. Test Case ID: TC010

- **Input:** Admin verifies and approves a valid fund request.
- **Expected Result:** Request is marked as approved.
- **Actual Result:** Request successfully approved.
- **Status:** Pass

8.3. Test Report

Introduction

The project is designed to detect fraudulent medical fund requests by analyzing uploaded hospital bills using Optical Character Recognition (OCR) and pattern-matching techniques. The system automates fund request verification and ensures donors contribute only to genuine cases. This report provides an overview of the testing process, including test objectives, scope, environment, and conclusions.

Test Objective

The primary objective of testing is to validate the system's functionalities, ensuring accuracy, security, and reliability. The tests focus on:

- User authentication (Registration & Login)
- Medical bill upload and text extraction
- Fraud detection using pattern matching
- Fund request approval/rejection process
- Secure donor transactions
- Performance under high loads

Test Scope

The test covers all critical system functionalities, including:

- **Functional Testing:** Validates core features such as user authentication, bill uploads, fraud detection, and donations.
- **Security Testing:** Ensures protection against cyber threats such as SQL injection and unauthorized access.
- **Performance Testing:** Tests system stability and response times under heavy loads.
- **Usability Testing:** Evaluates user experience and ease of navigation.

Test Environment

- **Operating System:** Windows 10/Linux
- **Backend:** Python (Flask), MySQL
- **Frontend:** HTML, CSS, JavaScript (Bootstrap)

- **Database:** MySQL 5.7
- **Testing Tools:** Selenium (for UI automation), JMeter (for load testing), Postman (for API testing)

Test Conclusion

The testing phase successfully validated the system's functionalities. All major test cases were executed, and the system met the expected outcomes. The fraud detection algorithm effectively identified fraudulent cases, ensuring accuracy in fund verification. The system performed well under various conditions without major slowdowns. Minor UI-related issues were identified and resolved. A comprehensive set of test cases was designed to cover all major functionalities of the application, including user authentication, document submission, fund request processing, fraud detection, and administrative verification workflows. The successful completion of the testing phase has significantly strengthened confidence in the system's readiness for deployment and real-world use, ensuring that it will reliably assist in verifying fund requests and protecting donors from fraudulent campaigns.

CHAPTER 9

CONCLUSION

Medical fund fraud is a growing issue where individuals or organizations submit fake, manipulated, or duplicate medical documents to unlawfully claim financial aid. This fraudulent activity results in misuse of resources, financial losses, and delays in assistance for genuine patients in need. Traditional systems for fund distribution often rely on manual verification, which is time-consuming, prone to human error, and lacks real-time fraud detection mechanisms. These existing methods fail to efficiently identify forged documents or repetitive claims, leading to ineffective fund allocation and donor distrust. To overcome these challenges, our project introduces an intelligent, automated Medical Fund Verification System that integrates YOLOv8 object detection, Optical Character Recognition (OCR), machine learning algorithms, and secure payment gateways. The system effectively detects forged bills, manipulated documents, and fake hospital stamps, ensuring that only genuine requests receive funding. Key features include automated fraud detection, a trust score system, a real-time notification module, and a secure donor payment processing system. The project not only enhances accuracy and efficiency in fund verification but also reduces fraud risks and improves transparency in medical fund distribution. With its robust fraud detection capabilities, this system increases donor confidence, ensures fair fund allocation, and streamlines the financial aid process, making it a reliable and impactful solution for combating medical fund fraud.

CHAPTER 10

FUTURE ENHANCEMENT

- **Blockchain Integration for Security**

Implementing blockchain technology will ensure tamper-proof record-keeping for medical documents and transactions. Smart contracts can automate fund disbursement, ensuring that payments are released only after proper verification, reducing human errors and fraudulent claims.

- **Mobile Application for Accessibility**

Developing a mobile version of the system will allow patients, donors, and administrators to access and manage fund requests from anywhere. Real-time notifications, instant fund verification, and fraud alerts can improve usability and engagement.

- **Integration with Government and Insurance Databases**

To enhance verification accuracy, the system can be integrated with government healthcare schemes and insurance company databases. This will allow real-time validation of medical records and reduce fraudulent claims.

APPENDIX

A. SOURCE CODE

Packages

```
from flask import Flask, render_template, Response, redirect, request, session,
abort, url_for, jsonify
import os
import time
import datetime
from random import randint
import cv2
import PIL.Image
from PIL import Image
import imagehash
import hashlib
import PyPDF2
pip install PyMuPDF
import fitz
import docx
from docx import Document
from diff_match_patch import diff_match_patch
from pdf2docx import parse
from typing import Tuple
from PIL import Image, ImageDraw, ImageFont
import textwrap
from spire.doc import *
from spire.doc.common import *
import pytesseract
from skimage.metrics import structural_similarity
from fuzzywuzzy import fuzz
from flask import send_file
from werkzeug.utils import secure_filename
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import mysql.connector
```

Database Connection

```
mydb = mysql.connector.connect(  
host="localhost",  
user="root",  
passwd="",  
charset="utf8",  
database="medical_fundraising"
```

Login

```
def login():  
msg=""  
if request.method=='POST':  
uname=request.form['uname']  
pwd=request.form['pass']  
cursor = mydb.cursor()  
cursor.execute('SELECT * FROM mf_user WHERE uname = %s AND pass =  
%s', (uname, pwd))  
account = cursor.fetchone()  
if account:  
session['username'] = uname  
return redirect(url_for('userhome'))  
else:  
msg = 'Incorrect username/password!'
```

User Registration

```
def reg_user():  
msg=""  
if request.method=='POST':  
name=request.form['name']  
mobile=request.form['mobile']  
email=request.form['email']  
address=request.form['address']  
city=request.form['city']  
acc_name=request.form['acc_name']  
bank_name=request.form['bank_name']  
account=request.form['account']  
branch=request.form['branch']  
gpay_number=request.form['gpay_number']
```

```

uname=request.form['uname']
pass1=request.form['pass']
now = datetime.datetime.now()
rdate=now.strftime("%d-%m-%Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT count(*) FROM mf_user where uname=%s",(uname,
cnt = mycursor.fetchone()[0]
if cnt==0:
mycursor.execute("SELECT max(id)+1 FROM mf_user")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO
mf_user(id,name,mobile,email,address,city,acc_name,bank_name,account,branch,
gpay_number,uname,pass) VALUES (%s, %s, %s, %s, %s, %s,
%s,%s,%s,%s,%s,%s,%s,%s)"
val =
(maxid,name,mobile,email,address,city,acc_name,bank_name,account,branch,gpay
_number,uname,pass1)
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
msg='success'
else:
msg="fail"

```

Post Medical Bill Information

```

def add_post():
msg=""
pid=request.args.get("pid")
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM mf_user where uname=%s",(uname,))
data = mycursor.fetchone()
mycursor.execute("SELECT * FROM mf_treatment")
data1 = mycursor.fetchall()
mycursor.execute("SELECT * FROM mf_status")
data2 = mycursor.fetchall()

```

```

now = datetime.datetime.now()
rdate=now.strftime("%d-%m-%Y")
if request.method=='POST':
    pat_name=request.form['pat_name']
    gender=request.form['gender']
    dob=request.form['dob']
    address=request.form['address']
    city=request.form['city']
    aadhar=request.form['aadhar']
    hospital=request.form['hospital']
    location=request.form['location']
    hos_city=request.form['hos_city']
    patient_id=request.form['patient_id']
    treatment=request.form['treatment']
    hospital_status=request.form['hospital_status']
    req_amount=request.form['req_amount']
    treatment1=""
    if treatment=="other":
        treatment1=request.form['oth_treatment']
        mycursor.execute("SELECT max(id)+1 FROM mf_treatment")
        maxid2 = mycursor.fetchone()[0]
        if maxid2 is None:
            maxid2=1
        sql2 = "INSERT INTO mf_treatment(id,treatment) VALUES (%s,%s)"
        val2 = (maxid2,treatment1)
        mycursor.execute(sql2, val2)
        mydb.commit()
    else:
        treatment1=treatment
        hos_status=""
        if hospital_status=="other":
            hos_status=request.form['oth_status']
            mycursor.execute("SELECT max(id)+1 FROM mf_status")
            maxid3 = mycursor.fetchone()[0]
            if maxid3 is None:
                maxid3=1

```

```

sql3 = "INSERT INTO mf_status(id,status) VALUES (%s,%s)"
val3 = (maxid3,hos_status)
mycursor.execute(sql3, val3)
mydb.commit()
else:
    hos_status=hospital_status
    mycursor.execute("SELECT max(id)+1 FROM mf_post")
    maxid = mycursor.fetchone()[0]
    if maxid is None:
        maxid=1
    ds=dob.split("-")
    dob1=ds[2]+"-"+ds[1]+"-"+ds[0]
    mycursor.execute("SELECT * FROM mf_patient_data")
    d2 = mycursor.fetchall()
    idd=0
    for dd in d2:
        if aadhar==dd[6] and hospital==dd[7]:
            if pat_name==dd[1] or patient_id==dd[10]:
                idd=dd[0]
            break
    else:
        idd=0
    print("idd")
    print(idd)
    if idd>0:
        mycursor.execute("SELECT * FROM mf_patient_data where id=%s",(idd,))
        d3 = mycursor.fetchone()
        bamount=d3[13]
        bill=int(bamount)
        ramount=int(req_amount)
        if ramount<=bill:
            sql = "INSERT INTO
mf_post(id,uname,pat_name,gender,dob,address,city,aadhar,hospital,location,hos_
city,patient_id,treatment,hospital_status,req_amount,req_date,req_status)
VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

```

```

val =
(maxid,uname,pat_name,gender,dob1,address,city,aadhar,hospital,location,hos_cit
y,patient_id,treatment1,hos_status,req_amount,rdate,'0')
mycursor.execute(sql, val)
mydb.commit()
pid=str(maxid)
mycursor.execute("SELECT count(*) FROM mf_files where post_id=%s",(pid,))
cnt = mycursor.fetchone()[0]
if cnt>0:
mycursor.execute("update mf_files set post_id=%s,status=1 where
status=0",(pid,))
mydb.commit()
msg="ok"
else:
msg="nofile"

```

Verification

```

def verify1():
act=request.args.get("act")
pid=request.args.get("pid")
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM mf_user where uname=%s",(uname,))
data = mycursor.fetchone()
return render_template('web/verify1.html',msg=msg,act=act,data=data,pid=pid)
def convert_pdf2docx(input_file: str, output_file: str, pages: Tuple = None):
if pages:
pages = [int(i) for i in list(pages) if i.isnumeric()]
result = parse(pdf_file=input_file,
docx_with_path=output_file, pages=pages)
summary = {
"File": input_file, "Pages": str(pages), "Output File": output_file
return result
def word_to_img(wfile,fid):
document = Document()
document.LoadFromFile("static/upload/"+wfile)
# Or load a Word DOC file
#document.LoadFromFile("Sample.doc")

```



```

image_streams = document.SaveImageToStreams(ImageType.Bitmap)
i = 1
# Save each image stream to a PNG file
for image in image_streams:
    image_name = "m"+str(fid)+"_"+str(i) + ".png"
    with open("static/upload/"+image_name,'wb') as image_file:
        image_file.write(image.ToArray())
    i += 1
# Close the document
document.Close()
return i

def extract_text_from_docx(file_path):
    doc = docx.Document(file_path)
    full_text = ""
    for para in doc.paragraphs:
        full_text += para.text + "\n"
    return full_text

def extract_text_from_pdf(file_path):
    doc = fitz.open(file_path)
    full_text = ""
    for page in doc:
        full_text += page.get_text()
    return full_text

#YoloV8 - Object Detection
def object_detect():
    ap = argparse.ArgumentParser()
    ap.add_argument('-i', '--image', required=True,
        help = 'path to input image')
    ap.add_argument('-c', '--config', required=True,
        help = 'path to yolo config file')
    ap.add_argument('-w', '--weights', required=True,
        help = 'path to yolo pre-trained weights')
    ap.add_argument('-cl', '--classes', required=True,
        help = 'path to text file containing class names')
    args = ap.parse_args()
    def get_output_layers(net):

```

```

layer_names = net.getLayerNames()
try: output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
except:
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
return output_layers

def draw_prediction(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
label = str(classes[class_id])
color = COLORS[class_id]
cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)
cv2.putText(img, label, (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color, 2)

image = cv2.imread(args.image)
Width = image.shape[1]
Height = image.shape[0]
scale = 0.00392
classes = None
with open(args.classes, 'r') as f:
classes = [line.strip() for line in f.readlines()]
COLORS = np.random.uniform(0, 255, size=(len(classes), 3))
net = cv2.dnn.readNet(args.weights, args.config)
blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)
net.setInput(blob)
outs = net.forward(get_output_layers(net))
class_ids = []
confidences = []
boxes = []
conf_threshold = 0.5
nms_threshold = 0.4
for out in outs:
for detection in out:
scores = detection[5:]
class_id = np.argmax(scores)
confidence = scores[class_id]
if confidence > 0.5:
center_x = int(detection[0] * Width)
center_y = int(detection[1] * Height)

```

```

w = int(detection[2] * Width)
h = int(detection[3] * Height)
x = center_x - w / 2
y = center_y - h / 2
class_ids.append(class_id)
confidences.append(float(confidence))
boxes.append([x, y, w, h])
indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
for i in indices:
    try:
        box = boxes[i]
    except:
        i = i[0]
        box = boxes[i]
        x = box[0]
        y = box[1]
        w = box[2]
        h = box[3]
        draw_prediction(image, class_ids[i], confidences[i], round(x), round(y),
            round(x+w), round(y+h))
#PaddleOCR
def PaddleOCR():
    params = parse_args(mMain=False)
    params.__dict__.update(**kwargs)
    assert ( params.ocr_version in SUPPORT_OCR_MODEL_VERSION
), "ocr_version must in { }, but get {}".format(
SUPPORT_OCR_MODEL_VERSION, params.ocr_version
    params.use_gpu = check_gpu(params.use_gpu)
    if not params.show_log:
        logger.setLevel(logging.INFO)
    self.use_angle_cls = params.use_angle_cls
    lang, det_lang = parse_lang(params.lang)
    # init model dir
    det_model_config = get_model_config("OCR", params.ocr_version, "det",
det_lang)
    params.det_model_dir, det_url = confirm_model_dir_url(

```

```

params.det_model_dir,
os.path.join(BASE_DIR, "whl", "det", det_lang),
det_model_config["url"],
rec_model_config = get_model_config("OCR", params.ocr_version, "rec", lang)
params.rec_model_dir, rec_url = confirm_model_dir_url(
params.rec_model_dir,
os.path.join(BASE_DIR, "whl", "rec", lang),
rec_model_config["url"],
cls_model_config = get_model_config("OCR", params.ocr_version, "cls", "ch")
params.cls_model_dir, cls_url = confirm_model_dir_url(
params.cls_model_dir,
os.path.join(BASE_DIR, "whl", "cls"),
cls_model_config["url"],
if params.ocr_version in ["PP-OCRv3", "PP-OCRv4"]:
params.rec_image_shape = "3, 48, 320"
else: params.rec_image_shape = "3, 32, 320"
if kwargs.get("rec_image_shape") is not None:
params.rec_image_shape = kwargs.get("rec_image_shape")
# download model if using paddle infer
if not params.use_onnx:
maybe_download(params.det_model_dir, det_url)
maybe_download(params.rec_model_dir, rec_url)
maybe_download(params.cls_model_dir, cls_url)
if params.det_algorithm not in SUPPORT_DET_MODEL:
logger.error("det_algorithm must in {}".format(SUPPORT_DET_MODEL))
sys.exit(0)
if params.rec_algorithm not in SUPPORT_REC_MODEL:
logger.error("rec_algorithm must in {}".format(SUPPORT_REC_MODEL))
sys.exit(0)
if params.rec_char_dict_path is None:
params.rec_char_dict_path = str(
Path(__file__).parent / rec_model_config["dict_path"])
logger.debug(params)
# init det_model and rec_model
super().__init__(params)
self.page_num = params.page_num
img = preprocess_image(img)

```

```

dt_boxes, elapse = self.text_detector(img)
if dt_boxes.size == 0:
    ocr_res.append(None)
    continue
tmp_res = [box.tolist() for box in dt_boxes]
ocr_res.append(tmp_res)
return ocr_res
else:
    ocr_res = []
    cls_res = []
    for img in imgs:
        if not isinstance(img, list):
            img = preprocess_image(img)
            img = [img]
        if self.use_angle_cls and cls:
            img, cls_res_tmp, elapse = self.text_classifier(img)
            if not rec:
                cls_res.append(cls_res_tmp)
            rec_res, elapse = self.text_recognizer(img)
            ocr_res.append(rec_res)
        if not rec:
            return cls_res
    return ocr_res
def PPStructure():
    params = parse_args(mMain=False)
    params.__dict__.update(**kwargs)
    assert ( params.structure_version in
SUPPORT_STRUCTURE_MODEL_VERSION
), "structure_version must in {}, but get {}".format(
SUPPORT_STRUCTURE_MODEL_VERSION, params.structure_version
    params.use_gpu = check_gpu(params.use_gpu)
    params.mode = "structure"
    if not params.show_log:
        logger.setLevel(logging.INFO)
    lang, det_lang = parse_lang(params.lang)
    if lang == "ch":

```

```

table_lang = "ch"
else:
table_lang = "en"
if params.structure_version == "PP-Structure":
params.merge_no_span_structure = False
# init model dir
det_model_config = get_model_config("OCR", params.ocr_version, "det",
det_lang)
params.det_model_dir, det_url = confirm_model_dir_url(
params.det_model_dir,
os.path.join(BASE_DIR, "whl", "det", det_lang),
det_model_config["url"],
rec_model_config = get_model_config("OCR", params.ocr_version, "rec", lang)
params.rec_model_dir, rec_url = confirm_model_dir_url(
params.rec_model_dir,
os.path.join(BASE_DIR, "whl", "rec", lang),
rec_model_config["url"],
table_model_config = get_model_config(
"STRUCTURE", params.structure_version, "table", table_lang
params.table_model_dir, table_url = confirm_model_dir_url(
params.table_model_dir,
os.path.join(BASE_DIR, "whl", "table"),
table_model_config["url"],
layout_model_config = get_model_config(
"STRUCTURE", params.structure_version, "layout", lang
params.layout_model_dir, layout_url = confirm_model_dir_url(
params.layout_model_dir,
os.path.join(BASE_DIR, "whl", "layout"),
layout_model_config["url"],
formula_model_config = get_model_config(
"STRUCTURE", params.structure_version, "formula", lang
params.formula_model_dir, formula_url = confirm_model_dir_url(
params.formula_model_dir,
os.path.join(BASE_DIR, "whl", "formula"),
formula_model_config["url"],
if not params.use_onnx:

```

```

maybe_download(params.det_model_dir, det_url)
maybe_download(params.rec_model_dir, rec_url)
maybe_download(params.table_model_dir, table_url)
maybe_download(params.layout_model_dir, layout_url)
maybe_download(params.formula_model_dir, formula_url)
if params.rec_char_dict_path is None:
    params.rec_char_dict_path = str(
        Path(__file__).parent / rec_model_config["dict_path"]
    )
if params.table_char_dict_path is None:
    params.table_char_dict_path = str(
        Path(__file__).parent / table_model_config["dict_path"]
    )
if params.layout_dict_path is None:
    params.layout_dict_path = str(
        Path(__file__).parent / layout_model_config["dict_path"]
    )
if params.formula_char_dict_path is None:
    params.formula_char_dict_path = str(
        Path(__file__).parent / formula_model_config["dict_path"]
    )
logger.debug(params)
super().__init__(params)
#Fuzzywuzzy
def test_Fuzzywuzzy():
    strings = ["atlanta braves", "Cães danados",
               "//// Mets $$$", "Ça va?"]
    for string in strings:
        proc_string =
        StringProcessor.replace_non_letters_non_numbers_with_whitespace(string)
        regex = re.compile(r"(?ui)[\W]")
        for expr in regex.finditer(proc_string):
            self.assertEqual(expr.group(), " ")
    def test_dont_condense_whitespace(self):
        s1 = "new york mets - atlanta braves"
        s2 = "new york mets atlanta braves"
        p1 = StringProcessor.replace_non_letters_non_numbers_with_whitespace(s1)
        p2 = StringProcessor.replace_non_letters_non_numbers_with_whitespace(s2)
        self.assertNotEqual(p1, p2)
    def setUp(self):

```

```

self.s1 = "new"
self.s1a = "mets"
self.s2 = "new YORK mets"
self.s3 = "the wonderful new york mets"
self.s4 = "new york mets vs atlanta braves"
self.s5 = "atlanta braves vs new york mets"
self.s6 = "new york mets - atlanta braves"
self.mixed_strings = [
    "Lorem Ipsum is simply dummy text of the printing and typesetting industry.",
    "C'est la vie",
    "Ça va?",
    "Cães danados",
    "\xacCamarões assados",
    "a\xac\u1234\u20ac\u00008000",
    "\u00C1"
]
def tearDown(self):
    pass
def test_asciidammit(self):
    for s in self.mixed_strings:
        utils.asciidammit(s)
def test_asciionly(self):
    for s in self.mixed_strings:
        # ascii only only runs on strings
        s = utils.asciidammit(s)
        utils.asciionly(s)
def test_fullProcess(self):
    for s in self.mixed_strings:
        utils.full_process(s)
def test_fullProcessForceAscii(self):
    for s in self.mixed_strings:
        utils.full_process(s, force_ascii=True)
self.cirque_strings = [
    "cirque du soleil - zarkana - las vegas",
    "cirque du soleil ",
    "cirque du soleil las vegas",
    "zarkana las vegas",

```



```

"las vegas cirque du soleil at the bellagio",
"zarakana - cirque du soleil - bellagio"
self.baseball_strings = [
    "new york mets vs chicago cubs",
    "chicago cubs vs chicago white sox",
    "philladelphia phillies vs atlanta braves",
    "braves vs mets",
    def tearDown(self):
        pass
    def testEqual(self):
        self.assertEqual(fuzz.ratio(self.s1, self.s1a), 100)
        self.assertEqual(fuzz.ratio(self.s8, self.s8a), 100)
        self.assertEqual(fuzz.ratio(self.s9, self.s9a), 100)
    def testCaseInsensitive(self):
        self.assertNotEqual(fuzz.ratio(self.s1, self.s2), 100)
        self.assertEqual(fuzz.ratio(utils.full_process(self.s1), utils.full_process(self.s2)),
            100)
    def testPartialRatio(self):
        self.assertEqual(fuzz.partial_ratio(self.s1, self.s3), 100)
    def testTokenSortRatio(self):
        self.assertEqual(fuzz.token_sort_ratio(self.s1, self.s1a), 100)
    def testPartialTokenSortRatio(self):
        self.assertEqual(fuzz.partial_token_sort_ratio(self.s1, self.s1a), 100)
        self.assertEqual(fuzz.partial_token_sort_ratio(self.s4, self.s5), 100)
        self.assertEqual(fuzz.partial_token_sort_ratio(self.s8, self.s8a, full_process=False),
            100)
        self.assertEqual(fuzz.partial_token_sort_ratio(self.s9, self.s9a, full_process=True),
            100)
        self.assertEqual(fuzz.partial_token_sort_ratio(self.s9, self.s9a, full_process=False),
            100)
        self.assertEqual(fuzz.partial_token_sort_ratio(self.s10, self.s10a,
            full_process=False), 50)
    def testTokenSetRatio(self):
        self.assertEqual(fuzz.token_set_ratio(self.s4, self.s5), 100)
        self.assertEqual(fuzz.token_set_ratio(self.s8, self.s8a, full_process=False), 100)
        self.assertEqual(fuzz.token_set_ratio(self.s9, self.s9a, full_process=True), 100)

```

```

self.assertEqual(fuzz.token_set_ratio(self.s9, self.s9a, full_process=False), 100)
self.assertEqual(fuzz.token_set_ratio(self.s10, self.s10a, full_process=False), 50)
def testPartialTokenSetRatio(self):
self.assertEqual(fuzz.partial_token_set_ratio(self.s4, self.s7), 100)
def testQuickRatioEqual(self):
self.assertEqual(fuzz.QRatio(self.s1, self.s1a), 100)
def testQuickRatioCaseInsensitive(self):
self.assertEqual(fuzz.QRatio(self.s1, self.s2), 100)
def testQuickRatioNotEqual(self):
self.assertNotEqual(fuzz.QRatio(self.s1, self.s3), 100)
def testWRatioEqual(self):
self.assertEqual(fuzz.WRatio(self.s1, self.s1a), 100)
def testWRatioCaseInsensitive(self):
self.assertEqual(fuzz.WRatio(self.s1, self.s2), 100)
def testWRatioPartialMatch(self):
self.assertEqual(fuzz.WRatio(self.s1, self.s3), 90)
def testWRatioMisorderedMatch(self):
# misordered full matches are scaled by .95
self.assertEqual(fuzz.WRatio(self.s4, self.s5), 95)
def testWRatioUnicode(self):
self.assertEqual(fuzz.WRatio(unicode(self.s1), unicode(self.s1a)), 100)
def testQRatioUnicode(self):
self.assertEqual(fuzz.WRatio(unicode(self.s1), unicode(self.s1a)), 100)
def testEmptyStringsScore100(self):
self.assertEqual(fuzz.ratio("", ""), 100)
self.assertEqual(fuzz.partial_ratio("", ""), 100)
def testIssueSeven(self):
s1 = "HSINCHUANG"
s2 = "SINJHUAN"
s3 = "LSINJHUANG DISTRIC"
s4 = "SINJHUANG DISTRICT"
self.assertTrue(fuzz.partial_ratio(s1, s2) > 75)
self.assertTrue(fuzz.partial_ratio(s1, s3) > 75)
self.assertTrue(fuzz.partial_ratio(s1, s4) > 75)
def testQRatioUnicodeString(self):
s1 = "\u00C1"

```

```

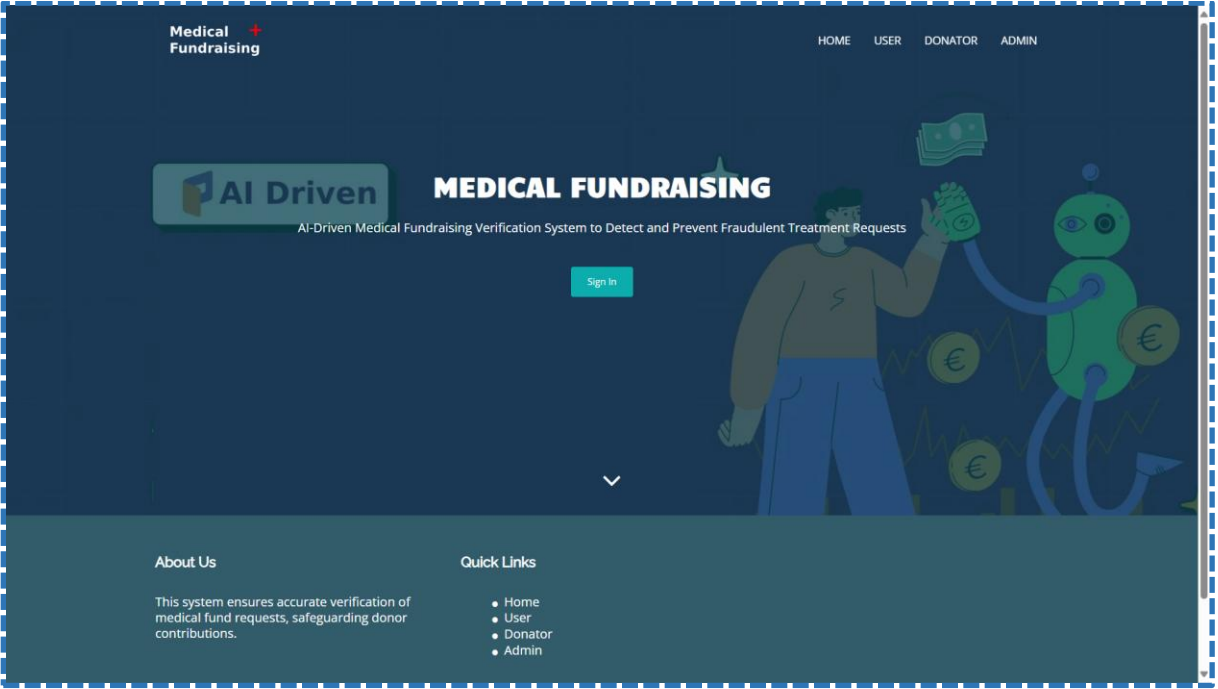
s2 = "ABCD"
score = fuzz.QRatio(s1, s2)
self.assertEqual(0, score)
s1 = "\u043f\u0441\u0438\u0445\u043e\u043b\u043e\u0433"
s2 =
\u043f\u0441\u0438\u0445\u043e\u0442\u0435\u0440\u0430\u043f\u0435\u0432\
u0442"
score = fuzz.QRatio(s1, s2, force_ascii=False)
self.assertNotEqual(0, score)
s1 = "\u6211\u4e86\u89e3\u6570\u5b66"
s2 = "\u6211\u5b66\u6570\u5b66"
score = fuzz.QRatio(s1, s2, force_ascii=False)
self.assertNotEqual(0, score)
def testQratioForceAscii(self):
s1 = "ABCD\u00C1"
s2 = "ABCD"
score = fuzz.QRatio(s1, s2, force_ascii=True)
self.assertEqual(score, 100)
score = fuzz.QRatio(s1, s2, force_ascii=False)
self.assertLess(score, 100)
def testQRatioForceAscii(self):
s1 = "ABCD\u00C1"
s2 = "ABCD"
score = fuzz.WRatio(s1, s2, force_ascii=True)
self.assertEqual(score, 100)
score = fuzz.WRatio(s1, s2, force_ascii=False)
self.assertLess(score, 100)
def testTokenSetForceAscii(self):
s1 = "ABCD\u00C1 HELP\u00C1"
s2 = "ABCD HELP"
score = fuzz._token_set(s1, s2, force_ascii=True)
self.assertEqual(score, 100)
score = fuzz._token_set(s1, s2, force_ascii=False)
self.assertLess(score, 100)
def testTokenSortForceAscii(self):
s1 = "ABCD\u00C1 HELP\u00C1"

```

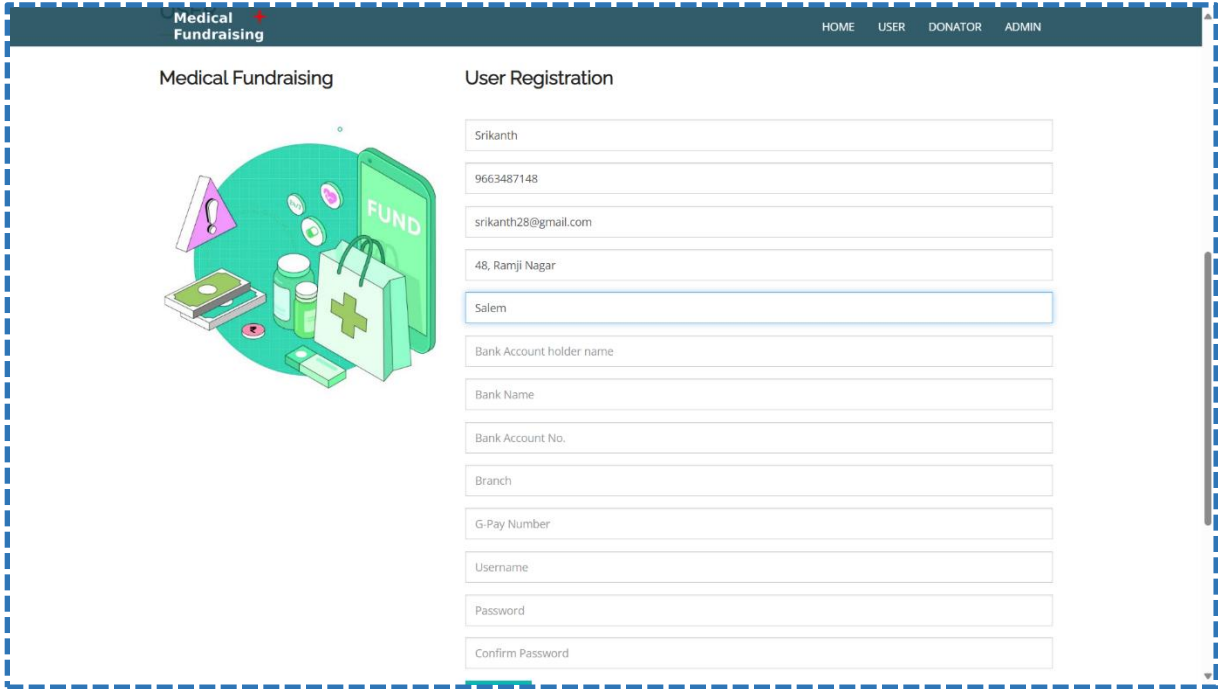
```
s2 = "ABCD HELP"
score = fuzz._token_sort(s1, s2, force_ascii=True)
self.assertEqual(score, 100)
score = fuzz._token_sort(s1, s2, force_ascii=False)
self.assertLess(score, 100)
```

B. SCREENSHOT

Landing Page :



User Registration Page :




Donor Signup Page:

Medical Fundraising

HOME USER DONATOR ADMIN

DONATOR

Medical Fundraising



Donator - Sign Up here

Vishnu

887562494

vishnu124@gmail.com

Madurai

Service organized in India Health Fund

vishnu

Sign In

New User - Sign Up


Fund Requester Signup Page :

Medical Fundraising

HOME USER DONATOR ADMIN

USER

Medical Fundraising



User's Medical Fund Requests

srikanth

Sign In

New User - Sign Up

About Us

Quick Links

Fund Request Portal:


User

Dashboard

Posts

Fund Received

Logout



Srikanth

Mob: 9663487148

Email: srikanth28@gmail.com

📍 48, Ramji Nagar, Salem

f

🐦

in

@

Send Your Post

Patient Name

Sivakumar

Gender

Male

Date of Birth

12-10-1970

Patient's Address

48, Ramji Nagar

Patient's City

Salem

Aadhar No.

255514330271

Hospital Name

Apollo

Hospital Location

Palpannai

Hospital City

Trichy

Patient ID

AP240341

Treatment for

Heart operation

Hospitalization Status

Discharged from hospital

Total Cost Required (Rs.)

₹5000

Attachments

Medical Bill/Patient Report Attachment:

Medical Fundraising - Personal - Microsoft Edge

localhost:5000/add_attach

Attachment

Medical Report/Bill (Image/PDF/Document)

Select the File

Choose File

No file chosen

Describe the File

Add

Attachments

#	Details	File	Action
1	Patient Report	f1D4r57003d.png	Delete
2	Medical Bill	f2dc377004.png	Delete

Attachment

Medical Report/Bill (Image/PDF/Document)

Select the File

Choose File

No file chosen

Describe the File

Add

Attachments

1. f3bill3.docx

Medical Bill

Evaluation Warning: The document was created with Spire.Doc for Python.

NTC Hospitals

Address: 187, Thatthaneri Main Road,
Near ESI Hospital
Vaihiyanathapuram,
Madurai - 625018.

Phone No.: 63856 02664
Email ID: ntchospitals@gmail.com

Hospital Bill Book

Bill

Patient Name: Murugan S

Patient ID: NT240412

Address: No.4, 5th Cross, KK Nagar, Trichy

Phone No.: 9861174681

Email ID: murugan55@gmail.com

Bill No. 24015

Admit Date: 10-10-2024

Admit Till Date: 21-10-2024

69

Fund Request Verification:

User

Dashboard

Posts

Fund Received

Logout

Dashboard

Send New Post

Information from Server

Patient Name	Murugan.S
Patient ID	NT240411
Gender	Male
Date of Birth	14-08-1972
Address	No.4, 5th Cross, KK Nagar, Trichy
Hospital	NTC Hospitals, Vaithiyanathapuram, Madurai
Treatment	Neuro surgery
Bill Amount(Rs.)	102200.0

Posted Information

Patient Name	Murugan.S
Patient ID	NT240411
Gender	Male
Date of Birth	14-08-1972
Address	No.4, 5th Cross, KK Nagar, Trichy
Hospital	NTC Hospitals, Vaithiyanathapuram, Madurai
Treatment	Neuro surgery
Bill Amount(Rs.)	102200.0

NTC Hospitals

Address


#87, Thathaneri Main Road,
Near ESI Hospital
Vaithiyanathapuram,
Madurai - 625018

Phone No.

63656 02664

Email ID

ntc.hospitals@gmail.com



Hospital Bill Book

NTC Hospitals

Address


#87, Thathaneri Main Road,
Near ESI Hospital
Vaithiyanathapuram,
Madurai - 625018

Phone No.

63656 02664

Email ID

ntc.hospitals@gmail.com



Hospital Bill Book

User

Dashboard

Posts

Fund Received

Logout

Evaluation Warning: The document was created with Spire Doc for Python.

NTC Hospitals

Address: 187, Thathaneil Main Road,
Near ESI Hospital
Vaiyyandilapattanam,
Madurai - 625018

Phone No: 63856 02664
Email ID: ntc hospitals@gmail.com

Hospital Bill Book

Bill
Patient Name: Murugan.S
Patient ID: NT240411
Address: No.4, 5th Cross, KK Nagar, Trichy
Phone No: 9881174681
Email ID: murugan5@gmail.com

Bill No: 24015
Admit Date: 10-10-2024
Admit Till Date: 21-10-2024

Sl.No.	Description	Quantity	Price/Unit	GST (%)	Amount
1	Registration Charges	1	50	12%	₹ 56.00
2	Room Rent	1	3600	12%	₹ 4032.00
3	Consultant Charges	2	7600	12%	₹ 8512.00
4	OT Charges	2	8000	12%	₹ 8960.00
5	Medicines	15	72000	12%	₹ 80640.00
Sub Total					₹ 476.40

Amount In Words

Discount	₹ 1000.00
Final Amount	₹ 110296.00
Amount Paid	₹ 102200.00
Balance	₹ 00.00

Declaration

Client's Signature

Business Signature

Thanks for business with us!!! Please visit us again!!!

NTC HOSPITALS

NTC INSTITUTE OF MEDICAL, DENTISTRY, NURSING

Extracted Text

Evaluation
Warning:
The
document

Evaluation Warning: The document was created with Spire Doc for Python.

NTC Hospitals

Address: 187, Thathaneil Main Road,
Near ESI Hospital
Vaiyyandilapattanam,
Madurai - 625018

Phone No: 63856 02664
Email ID: ntc hospitals@gmail.com

Hospital Bill Book

Bill
Patient Name: Murugan.S
Patient ID: NT240412
Address: No.4, 5th Cross, KK Nagar, Trichy
Phone No: 9881174681
Email ID: murugan5@gmail.com

Bill No: 24015
Admit Date: 10-10-2024
Admit Till Date: 21-10-2024

Sl.No.	Description	Quantity	Price/Unit	GST (%)	Amount
1	Registration Charges	1	50	12%	₹ 56.00
2	Room Rent	1	3600	12%	₹ 4032.00
3	Consultant Charges	2	7600	12%	₹ 8512.00
4	OT Charges	2	8000	12%	₹ 8960.00
5	Medicines	15	72000	12%	₹ 80640.00
Sub Total					₹ 476.40

Amount In Words

Discount	₹ 1000.00
Final Amount	₹ 110296.00
Amount Paid	₹ 102200.00
Balance	₹ 00.00

Declaration

Client's Signature

Business Signature

Thanks for business with us!!! Please visit us again!!!

NTC HOSPITALS

NTC INSTITUTE OF MEDICAL, DENTISTRY, NURSING

Extracted Text

Name:
Murugan.S
Bill
No.
24015

Sample Output 1:

User

Dashboard

Posts

Fund Received

Logout

1	Registration Charges	1	50	12%	₹ 56.00
2	Room Rent	1	3600	12%	₹ 4032.00
3	Consultant Charges	2	7600	12%	₹ 8512.00
4	OT Charges	2	8000	12%	₹ 8960.00
5	Medicines	15	72000	12%	₹ 80640.00
Sub Total					₹ 476.40

Amount In Words

Discount	₹ 1000.00
Final Amount	₹ 110296.00
Amount Paid	₹ 102200.00
Balance	₹ 00.00

Declaration

Client's Signature

Business Signature

Thanks for business with us!!! Please visit us again!!!

NTC HOSPITALS

NTC INSTITUTE OF MEDICAL, DENTISTRY, NURSING

Extracted Text

Name:
Murugan.S
Bill
No.
24015.
Patient
ID:
NT240411
Address:
No.4,
5th
Cross,
KK

1	Registration Charges	1	50	12%	₹ 56.00
2	Room Rent	1	3600	12%	₹ 4032.00
3	Consultant Charges	2	7600	12%	₹ 8512.00
4	OT Charges	2	8000	12%	₹ 8960.00
5	Medicines	15	72000	12%	₹ 80640.00
Sub Total					₹ 476.40

Amount In Words

Discount	₹ 1000.00
Final Amount	₹ 110296.00
Amount Paid	₹ 102200.00
Balance	₹ 00.00

Declaration

Client's Signature

Business Signature

Thanks for business with us!!! Please visit us again!!!

NTC HOSPITALS

NTC INSTITUTE OF MEDICAL, DENTISTRY, NURSING

Extracted Text

Name:
Murugan.S
Bill
No.
24015.
Patient
ID:
NT240412
Address:
No.4,
5th
Cross,
KK

This Information is Fake!!

71

Sample Output 2:

User

Dashboard

Posts

Fund Received

Logout

1	Registration Charges	1	50	12%	₹ 56.00
2	Room Rent	1	3600	12%	₹ 4032.00
3	Consultant Charges	2	7500	12%	₹ 8512.00
4	OT Charges	2	8000	12%	₹ 8960.00
5	Medicines	15	72000	12%	₹ 80640.00
			Sub Total		₹ 478.40
			Discount		₹ 1000.00
			Grand Amount		₹ 112204.00
			Amount Paid		₹ 112200.00
			Balance		₹ 00.00

Declaration

Client's Signature

Business Signature

Thanks for business with us!!! Please visit us again!!!

NTC HOSPITALS

NTC INSTITUTE OF MEDICAL SCIENCES (PUNE)

 | | | | | | | |---|----------------------|----|--------------|-----|-------------| | 1 | Registration Charges | 1 | 50 | 12% | ₹ 56.00 | | 2 | Room Rent | 1 | 3600 | 12% | ₹ 4032.00 | | 3 | Consultant Charges | 2 | 7500 | 12% | ₹ 8512.00 | | 4 | OT Charges | 2 | 8000 | 12% | ₹ 8960.00 | | 5 | Medicines | 15 | 72000 | 12% | ₹ 80640.00 | | | | | Sub Total | | ₹ 478.40 | | | | | Discount | | ₹ 1000.00 | | | | | Grand Amount | | ₹ 112204.00 | | | | | Amount Paid | | ₹ 112200.00 | | | | | Balance | | ₹ 00.00 | Declaration Client's Signature Business Signature Thanks for business with us!!! Please visit us again!!! NTC HOSPITALS NTC INSTITUTE OF MEDICAL SCIENCES (PUNE) |

Extracted Text

Name:
Murugan.S
Bill
No.
24015.
Patient
ID:
NT240411
Address:
No.4,
5"
Cross,
KK

Extracted Text

Name:
Murugan.S
Bill
No.
24015.
Patient
ID:
NT240411
Address:
No.4,
5"
Cross,
KK

This Information is Real!!

Donor Signup Page :

Medical Fundraising

HOME USER DONATOR ADMIN

AI Driven

DONATOR

Medical Fundraising

Fund Donator



ravi

Sign In

New Donator - Sign Up

72

Donor Portal :

Fund Donator

Dashboard

Posts

Fund Donated

Logout

Dashboard

Posts

Fund Donator

Ravindran

Mob: 9645781223

Email: ravi23@gmail.com

Salem

f

New Posts

Post by srikanth08-01-2025

Patient Name: Murugan.S

Gender: Male

Date of Birth: 14-08-1972

Aadhar No.: 232451417838

Address: No.4, 5th Cross, KK Nagar, Trichy

Hospital: 14-08-1972, Vaithiyanathapuram, Madurai

Patient ID: NT240411

Treatment for Neuro surgery

Status: Discharged from hospital

Medical Bill: Rs.102200.0

Medical Bill Attachment / Donate Fund

Medical Fundraising

Fund Donator

Dashboard

Posts

Fund Donated

Logout

Dashboard

Posts

Fund Donator

Ravindran

Mob: 9645781223

Email: ravi23@gmail.com

Salem

f

Fund Donation

Fund Amount (Rs.)50000

Transaction ID536214401824

Payment

Medical Fundraising

BIBLIOGRAPHY

JOURNAL REFERENCES

- [1] Sahu, S., & Nayak, R. (2020). “Medical Fraud Detection using Machine Learning Techniques.” *Journal of Healthcare Engineering*, 2020. DOI: 10.1155/2020/3281564
- [2] Koo, D., & Jeong, S. (2020). “Deep Learning for Medical Fraud Detection.” *Computers in Biology and Medicine*, 123, 103894. DOI: 10.1016/j.combiomed.2020.103894
- [3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). “You Only Look Once: Unified, Real-Time Object Detection.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. DOI: 10.1109/CVPR.2016.91
- [4] Vijayalakshmi, A., & Rajendran, S. (2019). “Fuzzy Matching Algorithm for Fraudulent Data Detection in Healthcare Systems.” *International Journal of Engineering and Advanced Technology*, 8(6), 198–203. DOI: 10.35940/ijeat.F8325.088619
- [5] Deng, Y., & Liu, L. (2021). “PaddleOCR: An Open-Source Optical Character Recognition (OCR) Toolkit.” *arXiv preprint arXiv:2104.01932*. DOI: 10.48550/arXiv.2104.01932
- [6] Jha, A., & Verma, S. (2020). “Blockchain-Based Transparent Fundraising for Medical Applications.” *Future Generation Computer Systems*, 108, 791–800. DOI: 10.1016/j.future.2020.03.001
- [7] Kshetri, N. (2018). “Blockchain and Healthcare Fraud Detection: An Overview.” *Computers, Privacy, and Security Issues in Healthcare*, 1, 19–34. DOI: 10.1007/978-3-319-77627-1_2
- [8] Rid, A., & Laskowski, A. (2016). “Ethical Issues in Crowdfunding for Medical Expenses.” *JAMA Internal Medicine*, 176(5), 681–686. DOI: 10.1001/jamainternmed.2016.1087
- [9] Lalitha Syama Sundari, C., Harika, G., Mahi Durga Lakshmi, G., Pavaneeth, J., & Bhargav Ram, G. (2024). *Fraudulent Health Insurance Claims Detection Using Machine Learning*. *International Journal of Novel Research and Development (IJNRD)*, 9(3).
- [10] Taherdoost, H. (2023). *Privacy and Security of Blockchain in Healthcare: Applications, Challenges, and Future Perspectives*. *Sci*, 5(4), 41.

- [11] Young, M. J., & Scheinberg, E. (2017). *The Rise of Crowdfunding for Medical Care: Promises and Perils*. JAMA, 317(16), 1623–1624.
- [12] Rawte, V., & Anuradha, G. (2020). *Study of Fuzzy Expert Systems Towards Prediction and Detection of Fraud in Health Insurance*. Procedia Computer Science, 167, 1821–1830.
- [13] Saleh, H., et al. (2019). *Blockchain Based Trusted Charity Fund-Raising*. International Journal of Soft Computing and Engineering (IJSCE), 10(1).
- [14] Young, M. J., & Scheinberg, E. (2017). *The Rise of Crowdfunding for Medical Care: Promises and Perils*. JAMA, 317(16), 1623–1624.
- [15] Bourque, F. (2016). *Fund My Treatment!: A Call for Ethics-Focused Social Science Research into the Rise of Crowdfunding for Medical Care*. Social Science & Medicine, 169, 27–30.
- [16] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788.

BOOK REFERENCES

1. HTML5, CSS3, JavaScript, Bootstrap – Duckett, J. "HTML & CSS: Design and Build Websites", Wiley.
2. Python – Lutz, M. "Learning Python", O'Reilly Media.
3. Flask – Grinberg, M. "Flask Web Development: Developing Web Applications with Python", O'Reilly Media.
4. MySQL – DuBois, P. "MySQL Cookbook", O'Reilly Media.
5. WampServer – Babin, C. "Apache, MySQL, and PHP Web Development All-in-One Desk Reference", Wiley.
6. OpenCV – Kaehler, A., & Bradski, G. "Learning OpenCV 4: Computer Vision with Python", O'Reilly Media.
7. NumPy, Pandas – VanderPlas, J. "Python Data Science Handbook", O'Reilly Media.

WEB LINK REFERENCES

1. HTML5, CSS3, JavaScript, Bootstrap – <https://developer.mozilla.org/en-US/docs/Web>
2. Python – <https://docs.python.org/3/>
3. Flask – <https://flask.palletsprojects.com/en/latest/>
4. MySQL – <https://dev.mysql.com/doc/>
5. WampServer – <https://www.wampserver.com/en/>
6. OpenCV – <https://docs.opencv.org/master/>
7. NumPy – <https://numpy.org/doc/stable/>
8. Pandas – <https://pandas.pydata.org/docs/>