

ANUDIP FOUNDATION

Project Title

“Contact Management System Using Django”

By

Name	Enrollment No
Kshitija Jadhav	AF0481826

Under Guidance

Of

Rajshri Thete

ABSTRACT

A **Contact Management System** (CMS) is a software application designed to store and manages information about individuals or organizations, typically including names, phone numbers, email addresses, and other relevant details. The purpose of such a system is to streamline the process of organizing and retrieving contact information efficiently.

This project focuses on developing a Contact Management System using Python, which provides users with the ability to add, search, update, and delete contact records. The system uses a simple text-based user interface (UI) that interacts with the user via the command line. The system's back-end utilizes Python's built-in file handling capabilities to store contact information in a CSV (Comma-Separated Values) format, ensuring ease of access and portability.

This Contact Management System aims to improve organization, accessibility, and ease of use for managing contact information, making it an ideal tool for personal or small business use.

Additionally, it offers a foundation for expanding functionality, such as integrating databases or developing a graphical user interface (GUI) for enhanced user experience.

ACKNOWLEDGEMENT

The project “**Contact Management System**” is the Project work carried out by

Name	Enrollment No
1.Kshitija Sunil Jadhav	AF0481826

Under the Guidance.

Rajshri Thete

We are thankful to my project guide for guiding me to complete the Project.

His suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

TOPIC OF THE PROJECT	Page No
1. Title of the Project	6-6
2. Introduction/Objective	7-9
2.1. Introduction	7-7
2.2. Objective	7-8
3. System Analysis	9-29
3.1. Problem Definition	9-9
3.2. Preliminary Investigation	9-11
3.3. Feasibility Study	11-15
3.4. Project Planning	15-16
3.5. Project Scheduling	16-17
3.6. Software Requirement Specification	18-19
3.9 Functional Requirements	19-22
3.10 Software Engineering Paradigm	23-29
4. System Design	30-43
4.1. Modularization Details	30-31
4.2. Database Design	31-34
4.3. Procedural Design	35-35
4.4. Flowchart Design	36-40
4.5. Outputs of the Report	41-43
4.6. List Of Reports	44-44
5. Coding	45-54
5.1. Complete Project Coding	45-53

5.4.	<i>Parameters Passing</i>	54-54
6.	<i>Testing</i>	55-60
6.1.	<i>Testing Strategies</i>	55-56
6.2.	<i>Conduction Test Cases</i>	57-60
7.	<i>System Security Measure</i>	61-63
7.1.	<i>Security Strategies</i>	61-61
7.2.	<i>Interface Security</i>	61-62
7.3.	<i>Database Security</i>	62-63
8.	<i>Future Application of the Project</i>	64
9.	<i>Bibliography</i>	66



TITLE OF THE PROJECT

**“CONTACT MANAGEMENT
SYSTEM USING DJANGO”**

INTRODUCION AND OBJECTIVE OF THE PROJECT

A **Contact Management System** is a basic yet essential application used to store and manage contact details like names, phone numbers, email addresses, and addresses in an organized manner. Developing such a system in Python provides an excellent opportunity to practice core programming concepts such as file handling, data structures (like lists and dictionaries), functions, and user input handling. Python's simplicity and readability make it an ideal language for beginners to build such systems while learning about data organization and CRUD (Create, Read, Update, Delete) operations.

The purpose of this Contact Management System is to allow users to add new contacts, search for existing ones, update contact information, delete entries, and display all saved contacts. Each operation can be accessed through a simple text-based menu, making the system user-friendly and easy to navigate. Data can be stored temporarily in memory or persistently in a text file, CSV file, or database, depending on the version of the system. This ensures that contact details are not lost when the program is closed. The system is flexible and can be expanded by adding features like sorting contacts, importing/exporting files, or even building a graphical user interface using Tkinter or PyQt. For students and beginners, this project is a great starting point to understand real-world applications of Python programming and improve their problem-solving skills. Additionally, it lays a strong foundation for building more advanced applications by emphasizing modular code, reusability, and data management techniques. Overall, a Contact Management System in Python is not only useful but also a highly educational project that demonstrates how to implement a practical and functional software solution using basic to intermediate programming skills.

OBJECTIVES:

The primary objective of developing a Contact Management System using Python is to create a simple, efficient, and user-friendly application that enables users to store and manage their contact information in an organized way. This system aims to provide essential functionalities such as adding, viewing, updating, searching, and deleting contact details, thereby making the management of personal or professional contacts easier and more effective. One of the key goals is to ensure data accuracy and quick access to information through a structured interface. By implementing this system, users can avoid manual record-keeping and reduce the chances of losing important contact information. From a learning perspective, the project also aims to strengthen the developer's understanding of core Python concepts, including file handling, functions, loops, conditionals, and data structures like lists and dictionaries. It also introduces the practice of writing modular and reusable code.

key objectives of the Contact Management System in **points**:

- ☐ To develop a simple and efficient application for storing and managing contact information.
- ☐ To allow users to perform CRUD operations — Create, Read, Update, and Delete contacts easily.
- ☐ To practice and apply core Python programming concepts like file handling, loops, functions, and data structures.
- ☐ To ensure user-friendly interaction through a simple menu-driven interface.
- ☐ To maintain data accuracy and easy retrieval of contact information.
- ☐ To provide data persistence by saving contacts in files (like text or CSV) or databases.
- ☐ To improve problem-solving, logical thinking, and coding skills in real-world project development.
- ☐ To create a practical project that can be expanded into a larger contact management or CRM system.
- ☐ To help beginners gain confidence in software development using Python.



Problem faced by an individual due to lack of security are as follows:

- Information of the users will be no more personal, if adequate security measures are not taken to protect these data.
- The data so derived can be put to various numbers of misuses ranging from publishing them on the web to using them in an improper way.
- Another major problem that rises is the ability of the system to handle concurrent accesses to the database, as multiple users will be handling the system simultaneously.
- In other word, the use of computer and internet will gradually come to an end due to lack of security because security is like statistics. The extent of data it conceals is vital.

□ **PRELIMINARY INVESTIGATION:**

Purpose – The purpose of a Contact Management System is to efficiently store, organize, and manage contact information such as names, phone numbers, email addresses, and other relevant details. It helps users keep all their contacts in one place, making it easy to add, search, update, or delete records as needed. Such systems improve productivity by reducing the time spent manually managing contact lists and ensuring that important information is easily accessible and up to date. They are commonly used by individuals, businesses, and organizations to maintain clear and organized communication records.

Benefits-

The purpose of a Contact Management System is to efficiently store, organize, and manage contact information such as names, phone numbers, email addresses, and other relevant details. It helps users keep all their contacts in one place, making it easy to add, search, update, or delete records as needed. Such systems improve productivity by reducing the time spent

manually managing contact lists and ensuring that important information is easily accessible and up to date. They are commonly used by individuals, businesses, and organizations to maintain clear and organized communication records.

Proposed System – The proposed “ **Contact Management System**” is designed to be more entertaining and easier to use than traditional or existing systems. Unlike older methods such as handwritten notes, spread sheets, or complex software, this system offers a clean, modern, and interactive interface that makes managing contacts enjoyable and hassle-free. It allows users to quickly add, edit, search, and delete contacts through simple clicks and easy-to-understand forms, reducing the chances of confusion or errors.

One of the key improvements in this system is its visually appealing layout, which provides a pleasant user experience. The organized structure ensures that users can view all contact details at a glance, with clearly labelled sections and user-friendly buttons. This eliminates the need to scroll through cluttered data or use complicated shortcuts. Features like instant search and quick-edit options make the process smooth and efficient. Overall, the proposed system is more efficient, attractive, and enjoyable than the present systems. It saves time, reduces stress, and improves accuracy, all while offering a simple and fun way to manage important contact information. Whether used for personal or professional purposes, this system makes contact management an easy task for everyone, even those with little technical knowledge. It transforms a usually boring activity into a smooth and satisfying experience.

Technical Specifications :

- **Frontend:** HTML, CSS, Bootstrap (for responsive UI)
- **Backend:** Django (Python framework)
- **Database:** MySQL
- **Other Tools:** JavaScript, jQuery for interactivity

□ REQUIREMENT SPECIFICATIONS

The requirements that are to be satisfied from the proposed system are:

- **To ensure Reliability:** System performs its projected functions over a time.
- **To ensure Good Organization:** System performs a particular purpose.
- **To ensure Correctness:** Meets system specifications and user objectives.
- **To ensure Usability:** It is easy to understand and operate the system.
- **To ensure Maintainability:** System errors are detected and removed.
- **To ensure Expandability:** Simplicity of mounting the exiting database.
- **To ensure Portability:** Simplicity of transporting a program from one hardware configuration to another.
- **To ensure Accurateness:** To ensure accuracy in input, editing, computations, and productivity.
- **To ensure Error Easiness:** Error detection and correction vs. error prevention.
- **To ensure Communication:** Ensure effective inputs and outputs of the system are compatible with the existing system.

□ FEASIBILITY STUDY

A feasibility study is an evaluation of a proposal designed to determine the difficulty in carrying out a designated task. Generally, a feasibility study precedes technical development and project implementation. In other words, a feasibility study is an evaluation or analysis of the potential impact of a proposed project

We can also say that a feasibility study is a detailed analysis of a company and its operations that is conducted in order to predict the results of a

specific future course of action. Small business owners may find it helpful to conduct a feasibility study whenever they anticipate making an important strategic decision. For example, a company might perform a feasibility study to evaluate a proposed change in location, the acquisition of another company, a purchase of major equipment or a new computer system, the introduction of a new product or service, or the hiring of additional employees. In such situations, a feasibility study can help a small business's managers understand the impact of any major changes they might contemplate.

The feasibility study is the important step in any software development process. This is because it makes analysis of different aspects like cost required for developing and executing the system, the time required for each phase of the system and so on. If these important factors are not analyzed then definitely it would have impact on the organization and the development and the system would be a total failure.

The feasibility study provides one with objective information to evaluate existing services and strengths. Through feasibility study we gain an understanding of the competition and marketplace indicators.

Steps in Conducting a Feasibility Study:

- ❖ The main objective of a feasibility study is to determine whether a certain plan of action is feasible—that is, whether or not it will work, and whether or not it is worth doing economically. Although the core of the study is dedicated to showing the outcomes of specific actions, it should begin with an evaluation of the entire operation. For example, a good feasibility study would review a company's strengths and weaknesses, its position in the marketplace, and its financial

situation. It would also include information on a company's major competitors, primary customers, and any relevant industry trends. This sort of overview provides small business owners and managers with an objective view of the company's current situation and opportunities. By providing information on consumer needs and how best to meet them, a feasibility study can also lead to new ideas for strategic changes.

- ❖ The second part of a good feasibility study should focus on the proposed plan of action and provide a detailed estimate of its costs and benefits. In some cases, a feasibility study may lead management to determine that the company could achieve the same benefits through easier or cheaper means. For example, it may be possible to improve a manual filing system rather than purchase an expensive new computerized database. If the proposed project is determined to be both feasible and desirable, the information provided in the feasibility study can prove valuable in implementation. It can be used to develop a strategic plan for the project, translating general ideas into measurable goals.

Advantages of making Feasibility study:

- ✓ There are many advantages of making feasibility study some of which are summarized below:
- ✓ This study being made as the initial step of software development life cycle has all the analysis part in it which helps in analyzing the system requirements completely. .
- ✓ Helps in identifying the risk factors involved in developing and deploying the system..
- ✓ The feasibility study helps in planning for risk analysis.
- ✓ Feasibility study helps in making cost/benefit analysis which helps the organization and system to run efficiently.

- ✓ Feasibility study helps in making plans for training developers for implementing the system.
- ✓ So a feasibility study is a report which could be used by the senior or top persons in the organization. This is because based on the report the organization decides about cost estimation, funding and other important decisions which is very essential for an organization to run profitably and for the system to run stable.

Different Types of Feasibility:

In the conduct of feasibility study, the analyst will usually consider seven distinct but inter-related types of feasibility.

The different types of feasibility studies are as follows:

- **Technical feasibility study:** It is used to determine the requirements of technologies for the current system.
- **Schedule feasibility study:** It is used to determine the time factor related to the current system.
- **Social feasibility study:** It is used to determine whether a proposed project will be acceptable to the people or not. This determination typically examines the probability of the project being accepted by the group directly affected by the proposed system change.
- **Legal feasibility study:** It is used to determine the legal scrutiny of the current system.
- **Marketing feasibility study:** It is used to determine the single and multidimensional market forces that affect the current system. There are four types of marketing feasibility study, which are as follows:

1) Economic feasibility

- 2) **Legal feasibility**
- 3) **Operational feasibility**
- 4) **Schedule feasibility**
- 5) **Is it technically feasible or not?**

The project is made with the help of a PC is easily available.

Secondly, the software used for encryption and decryption purpose is a easily available.

Finally, the other software used to develop this project can easily interface with each other and also compatible with Windows 2000, Windows XP, Windows Server 2003, or Windows NT 4.0 operating system.

6) **Is it technically feasible or not?**

This website is very user –friendly. Any user who knows simple English language can use this website. The user does not need to be familiar with .net framework or SQL server to use this website.

□ **PROJECT PLANNING AND SCHEDULING:**

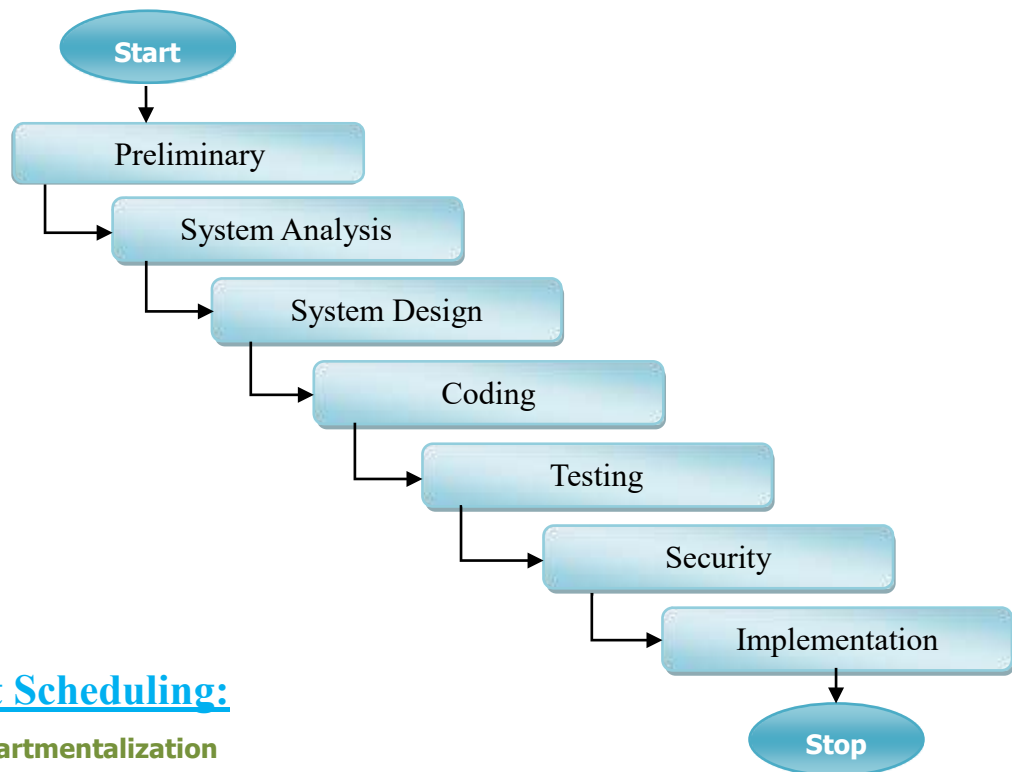
PROJECT PLANNING

■ Project Planning

■ **Phase To Cover**

1. Preliminary Investigation
2. System Analysis
3. System Design
4. Coding
5. Security
6. Testing
7. Implementation

■
Acti
ty
Diagr
am:



Project Scheduling:

■ **Compartmentalization**

Contact Management System	
Preliminary Investigation	Identification of Need
	Requirement Specifications
	Feasibility Study
System Analysis	Project Planning & Scheduling
	SRS Preparations
	Software Engineering Paradigm
System Design	Module Description
	Database Design
	Procedural Design
	User Interface Design
Coding	Error free Coding
	Debugging
Testing	Unit Testing
	Integration Testing
	System Testing
Security	Application Security
	Database Security
Implementation	Implementation
	Documentation

□ SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Requirements analysis is done in order to understand the problem, which is to be solved. That is very important activity for the development of database system. The requirements and the collection analysis phase produce both data requirements and functional requirements. The data requirements are used as a source of database design. The data requirements should be specified in as detailed and complete form as possible. In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of user defined operations that will be applied to the database (retrievals and updates). The functional requirements are used as a source of application software design.

Contact Management System :

The Data-requirements are given as follows

User Module:

User Registration / Login

User can: **Register** with the system by :

Full Name

Username (for login)

Password

Email Address (optional)

Phone Number (optional)

Manage Contacts: User can **add new contacts** by entering

Contact Name

Phone Number

Email Address

Address (optional)

User can **view a list of all contact** Contacts displayed with key details (name, phone, email)

User can search contacts by:

Name

Phone number

Email

User can **edit existing contact details**

User can **delete contacts**

System Features

1. Data Security

User data (credentials and contacts) are securely stored

Passwords are stored using hashing (recommended)

2. Navigation and Usability

Clean and simple interface

Clear buttons for add/edit/delete/search actions

Functional Requirements

The Contact Management System allows users to register and log in using a unique username and a secure password. Upon successful authentication, users can access the main features of the system. A key functionality is the ability to manage personal contacts. Users can add new contacts by entering details such as name, phone number, email address, physical address, and optional notes or tags. The system ensures that all required fields, such as the contact name and phone number, are properly filled before saving a contact. Once contacts are saved, users can view a list of all their contacts in a structured format. The system enables users to search for specific contacts

using name, phone number, or email address. Additionally, contacts can be selected to view detailed information, edited to update their information, or deleted from the system.

To enhance usability, the system supports filtering of contacts based on tags or other attributes (if implemented). Account management features include the ability for users to update their profile details and change their password securely. Security is maintained by encrypting or hashing stored passwords and ending user sessions after logout.

SOFTWARE AND HARDWARE REQUIREMENTS

User Interface:

The user interface of the Contact Management System is designed to be **simple, intuitive, and user-friendly**, ensuring smooth navigation and usability for all users. The system uses **HTML and CSS** for layout and styling.

Software Constraints

The system will run under any Operating System having HTML, CSS, MySQL Server, Python, Django Framework

Hardware Constraints :

Processor: Dual-core 1.6 GHz or higher

RAM: 4 GB

Storage: 500 MB of free disk space

Display: Minimum 1024x768 screen resolution

Software Requirements:

Following Software will be used for Development testing of the proposed project :

Windows 98SE, Windows 2000, Windows XP, Windows 10

-Microsoft internet explorer or web browser.

MySQL Server

-Microsoft Office 2010,11

The following software would be required for running the proposed project after implementation on target machine.

Python 3.x (Recommended: Python 3.9 or higher)

Code Editor or IDE: Visual Studio Code (VS Code)

Frontend : Html & CSS

Framework / Libraries :

Flask or Django (if using a web-based interface)

Web Browser : Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions for testing)

FUNCTIONAL REQUIREMENTS

User Registration/Login – Users can register themselves by providing basic details like name, username, password, phone number, and email. Once registered, they can log in to access the system and manage their contacts securely. Login sessions ensure user-specific contact management and data protection.

Dashboard Overview – After logging in, users are directed to a dashboard that gives them access to all key functionalities. This includes viewing the total number of contacts, quick access to add new contacts, and search options for fast navigation..

Add Contact – This section allows users to add new contact details by entering the contact's name, phone number, email address, address, and additional notes or tags. It ensures that required fields are validated before saving.

View/Search Contacts – Users can view a list of all their saved contacts in a table or card format. A search bar allows filtering of contacts by name, phone number, or email for quick access. This section enhances user experience and saves time when managing large lists.

Edit/Delete Contact – Users can update the details of any existing contact through the edit option. If a contact is no longer needed, it can be permanently removed from the system using the delete function.

Contact Details View – Clicking on any contact shows detailed information in an expanded view, including optional data like address and notes. This helps users review full information before taking any action like calling or editing.

SOFTWARE ENGINEERING PARADIGM

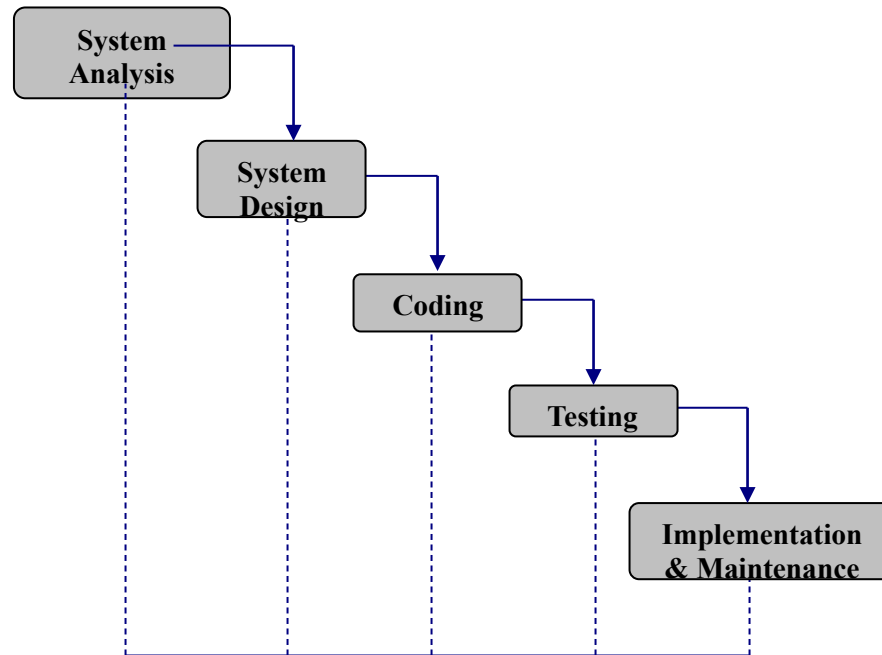
The conventional and mostly used software engineering paradigm till now – the Waterfall Model is applied on this project. But according to project complexity and nature slight deviation from the proposed original waterfall model is taken into consideration. The three characteristics of Waterfall Model as linear, rigid, and monolithic are diverted here.

The reasons and assumptions are explained below:

The linearity of Waterfall Model is based on the assumption that software development may proceed linearly from analysis down to coding. But, in this project feedback loops are applied to the immediately preceding stages. Another underlying assumption of the Waterfall Model is phase rigidity – i.e., that the results of each phase are frozen before proceeding to the next phase. But, this characteristic is not strictly maintained throughout the project. Therefore overlapping of two or more phases is allowed according to needs and judgments.

Finally, the Waterfall Model is monolithic in the sense that all planning is oriented to single delivery date. But since this project has deadline imposed by the organization, planning is done into successive phases. This project is the term-end project and obviously need to submit as soon as possible calendar date. With such considerations is selected the Waterfall model as the project development paradigm.

Water-fall Model with feedback mechanism for the proposed application as Software Engineering paradigm:



ANALYSIS DIAGRAMS

DFD (Data Flow Diagram): In the late 1970s *data-flow diagrams* (DFDs) were introduced and popularized for structured analysis and design (Game and Sarson 1979). DFDs show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage.

A **data flow diagram** (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.

Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the

business and continues by analyzing each of the functional areas of interest. This analysis can be carried out to precisely the level of detail require.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

Additionally, a **DFD** can be utilized to visualize data processing or a structured design. A DFD illustrates technical or business processes with the help of the external data stored, the data flowing from a process to another , and the results.

A designer usually draws a context-level DFD showing the relationship between the entities inside and outside of a system as one single step. This basic DFD can be then disintegrated to a lower level diagram demonstrating smaller steps exhibiting details of the system that is being modeled. Numerous levels may be required to explain a complicated system.

Process

The process shape represents a task that handles data within the application. The task may process the data or perform an action based on the data.

Multiple Processes

The multiple process shape is used to present a collection of sub processes. The multiple processes can be broken down into its sub processes in another DFD.

External Entity

The external entity shape is used to represent any entity outside the application that interacts with the application via an entry point.

Data Flow

The data flow shape represents data movement within the application. The direction of the data movement is represented by the arrow.

Data Store

The data store shape is used to represent locations where data is stored. Data stores do not modify the data, they only store data.

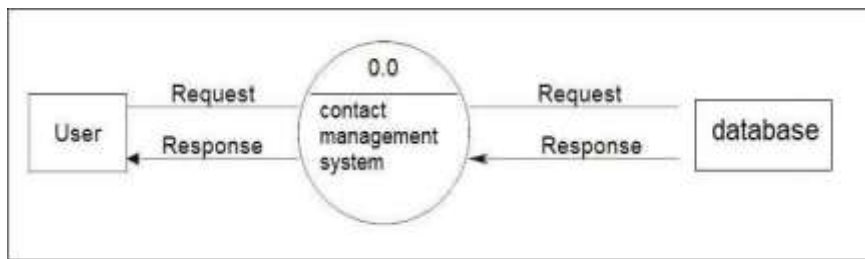
Privilege Boundary

The privilege boundary shape is used to represent the change of privilege levels as the data flows through the application.

Examples of Data Flow Diagrams - These examples demonstrate how to draw data flow diagram.

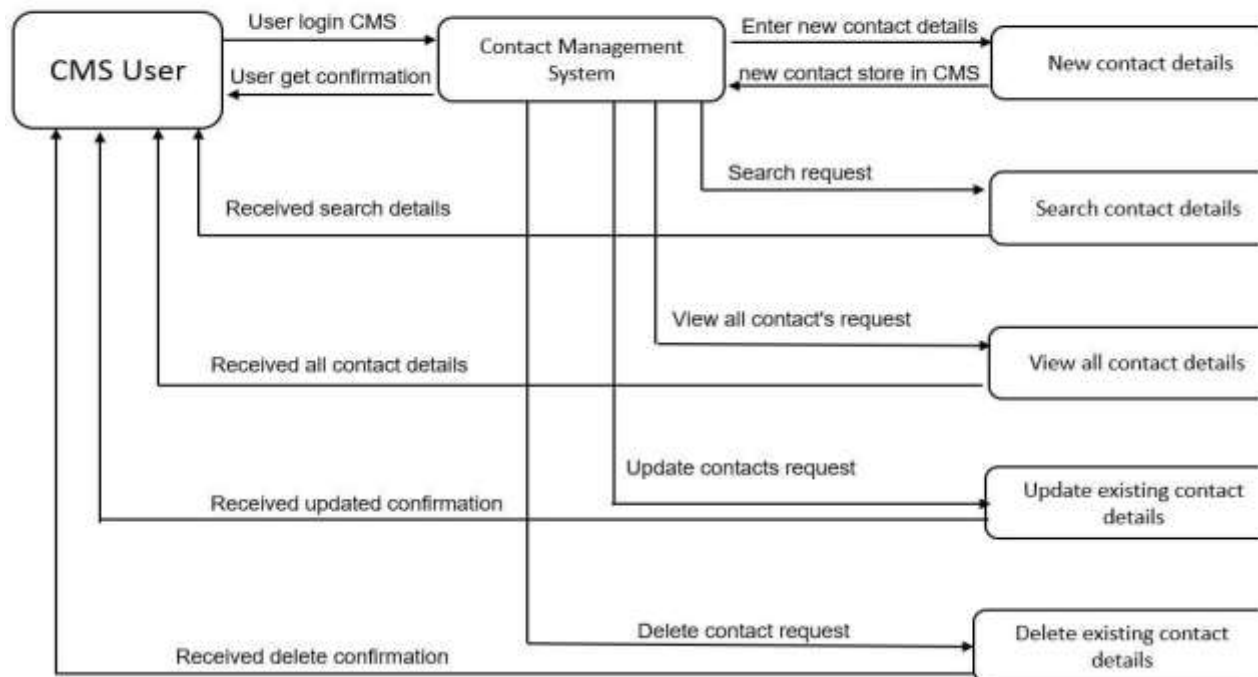
Before it was eventually replaced, a copy machine suffered frequent paper jams and became a notorious troublemaker. Often, a problem could be cleared by simply opening and closing the access panel. Someone observed the situation and flowcharted the troubleshooting procedure used by most people.

CONTEXT LEVEL DIAGRAM



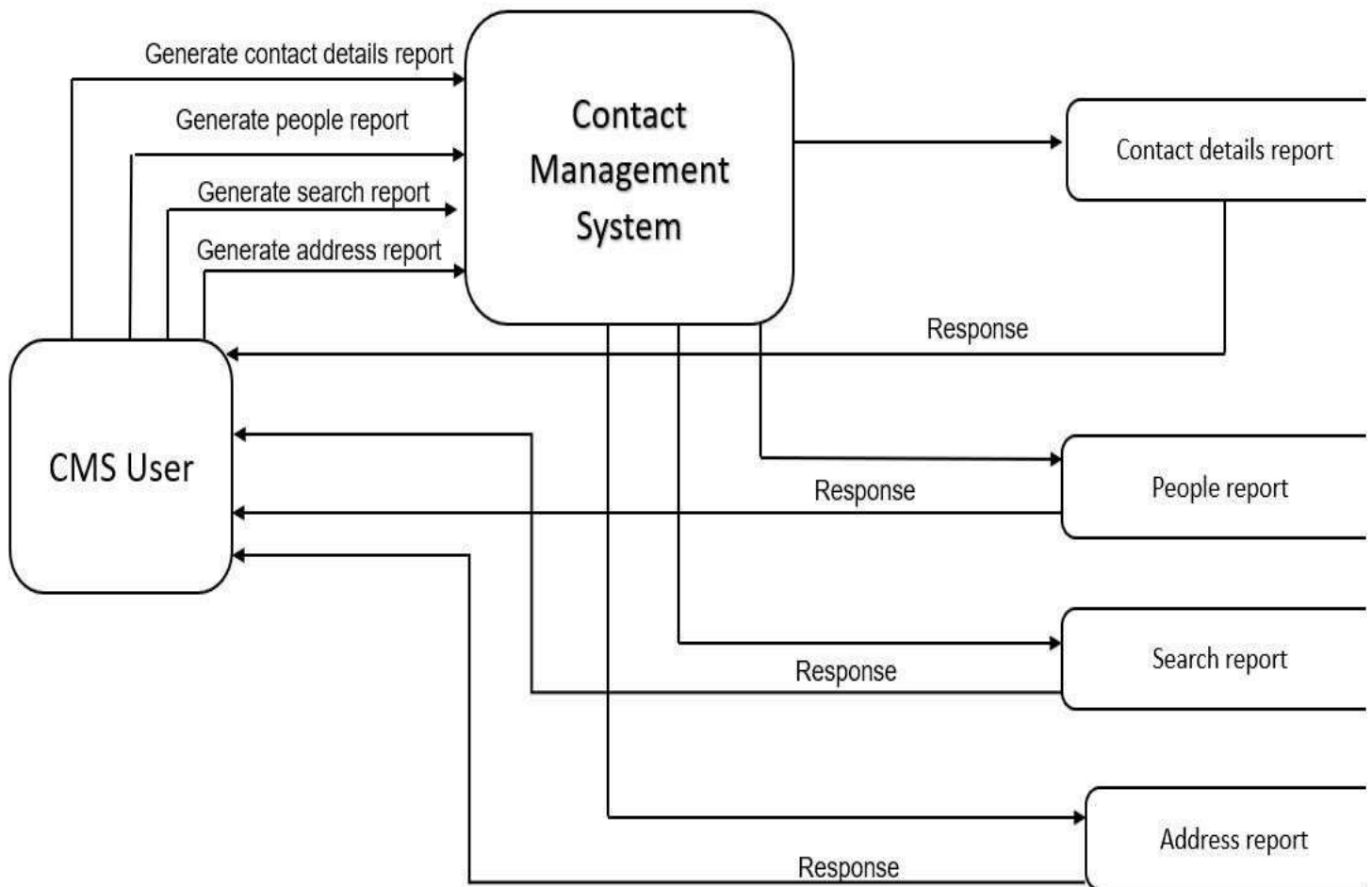
Generat 1st Level Data Flow Diagram

LEVEL -1 : DFD

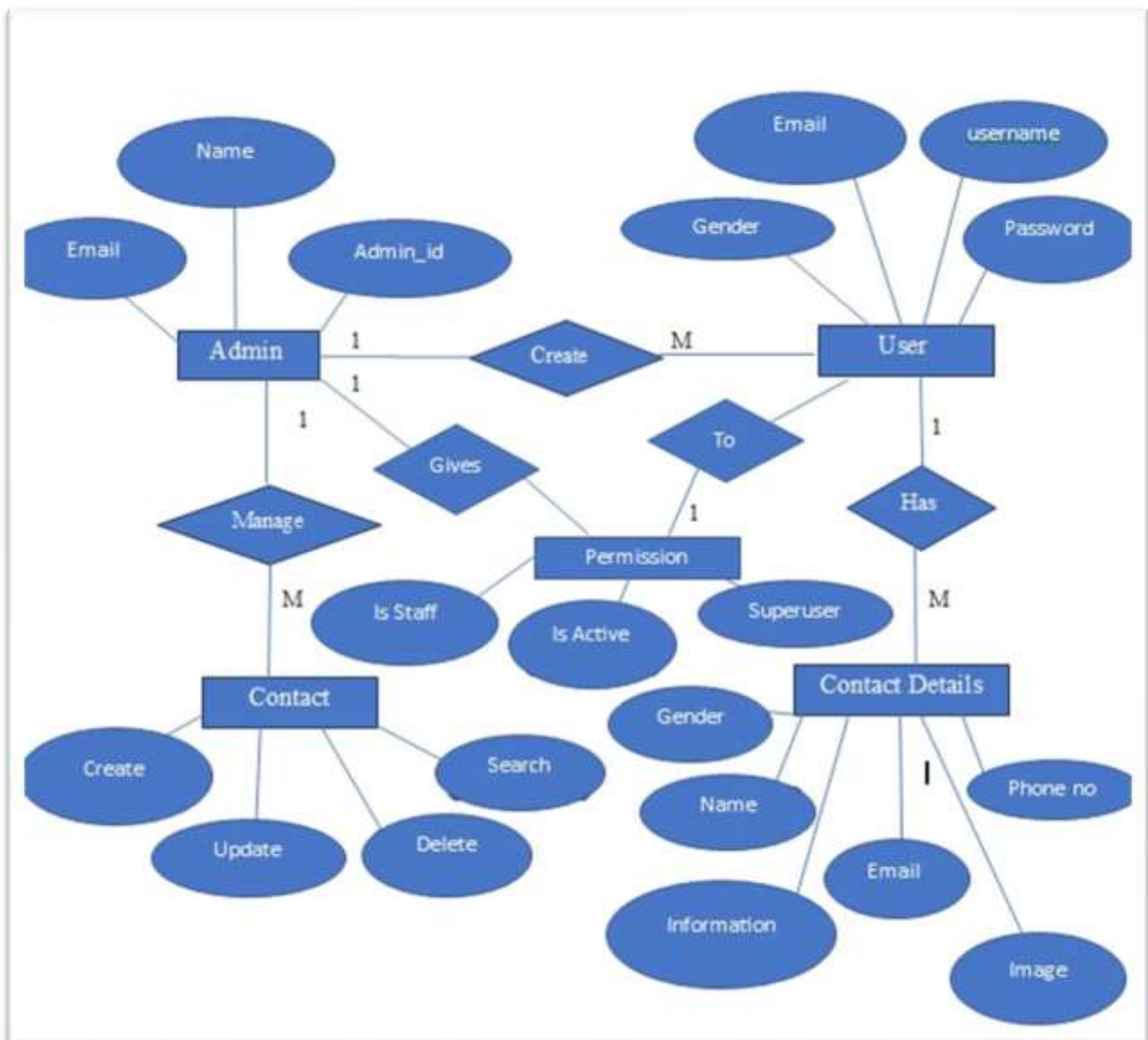


2nd Level Data Flow Diagram

LEVEL -2 : DFD



Entity Relationship Diagram (ERD):





□ MODULES AND THEIR DESCRIPTION

The various modules used in the project are as follows:

MODULES AND THEIR DESCRIPTION

User Registration/Login – Users can register themselves by providing basic details like name, username, password, phone number, and email. Once registered, they can log in to access the system and manage their contacts securely. Login sessions ensure user-specific contact management and data protection.

Dashboard Overview – After logging in, users are directed to a dashboard that gives them access to all key functionalities. This includes viewing the total number of contacts, quick access to add new contacts, and search options for fast navigation..

Add Contact – This section allows users to add new contact details by entering the contact's name, phone number, email address, address, and additional notes or tags. It ensures that required fields are validated before saving.

View/Search Contacts –. Users can view a list of all their saved contacts in a table or card format. A search bar allows filtering of contacts by name, phone number, or email for quick access. This section enhances user experience and saves time when managing large lists.

Edit/Delete Contact – Users can update the details of any existing contact through the edit option. If a contact is no longer needed, it can be permanently removed from the system using the delete function.

Contact Details View – Clicking on any contact shows detailed information in an expanded view, including optional data like address.

□ DATA STRUCTURE OF ALL MODULES:

Table Name: app_contact

```
mysql> desc app_contact;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20) | NO   |     | NULL    |               |
| email | varchar(100) | NO   |     | NULL    |               |
| phone | int   | NO   |     | NULL    |               |
| info  | varchar(30) | NO   |     | NULL    |               |
| gender | varchar(50) | NO   |     | NULL    |               |
| image | varchar(100) | NO   |     | NULL    |               |
| date_added | datetime(6) | NO   |     | NULL    |               |
| manager_id | int | NO   | MUL | NULL    |               |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.06 sec)
```

Table Name: **auth_group**

```
mysql> desc auth_group;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| name  | varchar(150) | NO   | UNI | NULL    |               |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

Table Name: auth_group_permissions

```
mysql> desc auth_group_permissions;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
group_id	int	NO	MUL	NULL	
permission_id	int	NO	MUL	NULL	

3 rows in set (0.01 sec)

Table Name: auth_permission

```
mysql> desc auth_permission;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
content_type_id	int	NO	MUL	NULL	
codename	varchar(100)	NO		NULL	

4 rows in set (0.01 sec)

Table Name : auth_user

```
mysql> desc auth_user;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
password	varchar(128)	NO		NULL	
last_login	datetime(6)	YES		NULL	
is_superuser	tinyint(1)	NO		NULL	
username	varchar(150)	NO	UNI	NULL	
first_name	varchar(150)	NO		NULL	
last_name	varchar(150)	NO		NULL	
email	varchar(254)	NO		NULL	
is_staff	tinyint(1)	NO		NULL	
is_active	tinyint(1)	NO		NULL	
date_joined	datetime(6)	NO		NULL	

11 rows in set (0.01 sec)

Table Name : auth_user_groups

```
mysql> desc auth_user_groups;
+-----+-----+-----+-----+-----+-----+
| Field      | Type | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int  | NO   | PRI | NULL    | auto_increment |
| user_id    | int  | NO   | MUL | NULL    |                |
| group_id   | int  | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Table Name :auth_user_permissions

```
mysql> desc auth_user_user_permissions;
+-----+-----+-----+-----+-----+-----+
| Field          | Type | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id             | int  | NO   | PRI | NULL    | auto_increment |
| user_id        | int  | NO   | MUL | NULL    |                |
| permission_id  | int  | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Table Name :django_admin_log

```
mysql> desc django_admin_log;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id             | int           | NO   | PRI | NULL    | auto_increment |
| action_time    | datetime(6)   | NO   |     | NULL    |                |
| object_id      | longtext      | YES  |     | NULL    |                |
| object_repr    | varchar(200)  | NO   |     | NULL    |                |
| action_flag    | smallint unsigned | NO   |     | NULL    |                |
| change_message | longtext      | NO   |     | NULL    |                |
| content_type_id | int           | YES  | MUL | NULL    |                |
| user_id        | int           | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```


Table Name : django_content_type

```
mysql> desc django_content_type;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int           | NO   | PRI | NULL    | auto_increment |
| app_label  | varchar(100)  | NO   | MUL | NULL    |                |
| model      | varchar(100)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Table Name : django_migrations



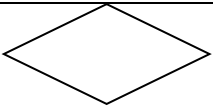


```
mysql> desc django_migrations;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int           | NO   | PRI | NULL    | auto_increment |
| app        | varchar(255)  | NO   |     | NULL    |                |
| name       | varchar(255)  | NO   |     | NULL    |                |
| applied    | datetime(6)   | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

Table Name :django_session

```
mysql> desc django_session;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| session_key    | varchar(40)   | NO   | PRI | NULL    |                |
| session_data   | longtext      | NO   |     | NULL    |                |
| expire_date    | datetime(6)   | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

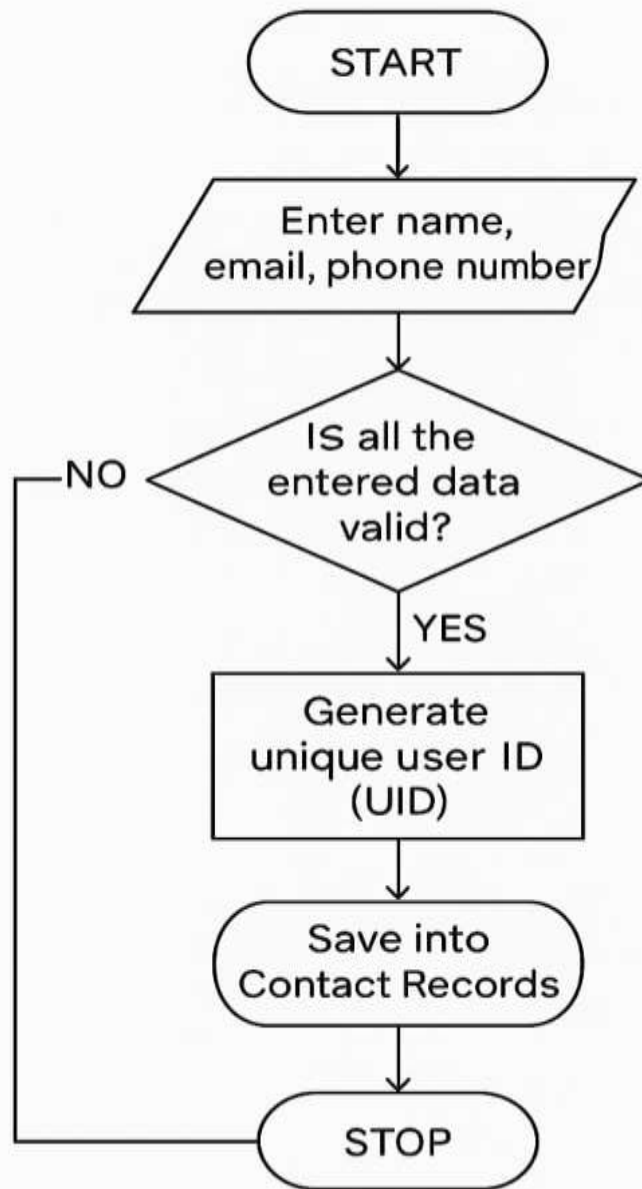
□ PROEDURAL DESIGN:

PROCESS LOGIC (FLOWCHART) OF EACH MODULE

<u>Notations Used For FLOW CHART</u>		
NOTATIONS		USED FOR
	Known as TERMINATOR	Used for START / STOP
	Known as DATA	Used for ACCEPTING INPUTS FROM USER AND also DISPLAYING THE ERROR MESSAGES GENERATED BY THE SYSTEM
	Known as DECISION	Used for DECISION MAKING
	Known as DISPLAY	Used for OUTPUT GENERATED BY THE SYSTEM
	Known as MAGNETIC DISK	Used for STORING INPUTS GIVEN TO THE SYSTEM

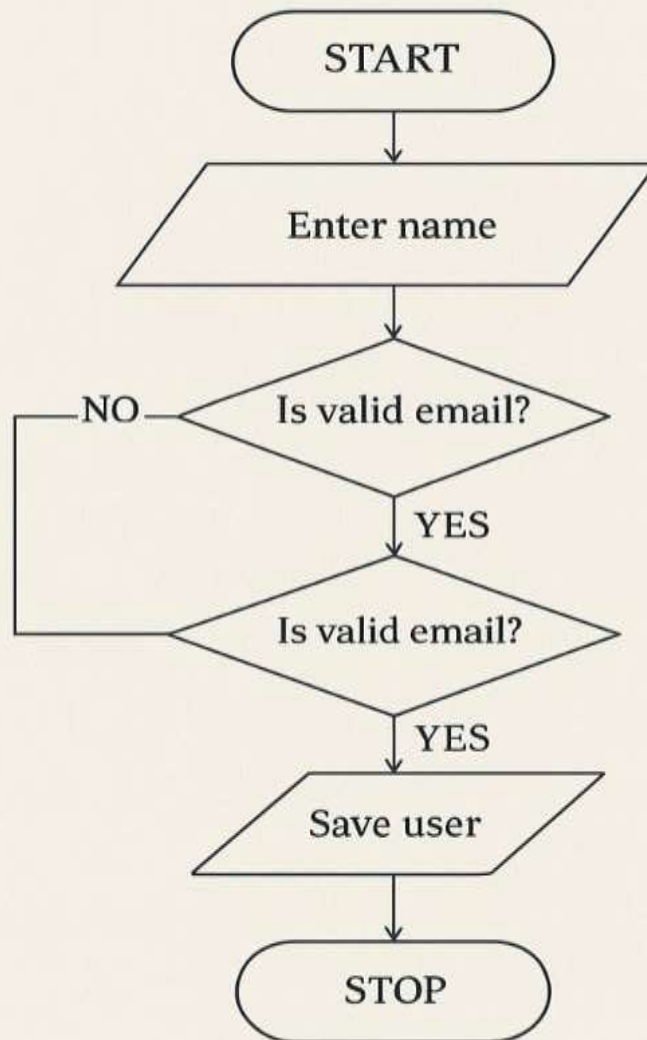
Flow chart for “Auth User”

Flow chart for "Auth User — Contact Management System

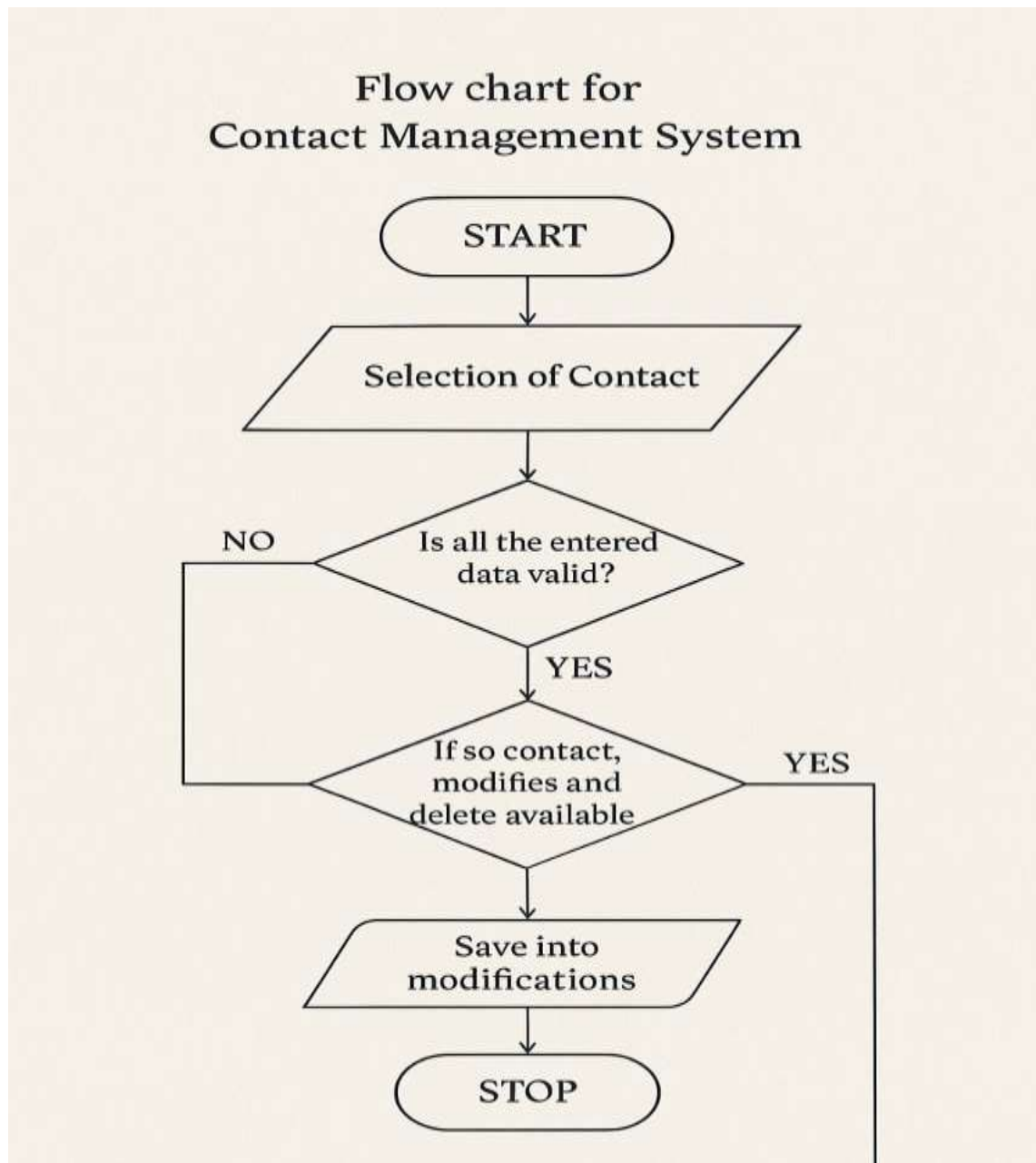


Flow chart for "User Registration"

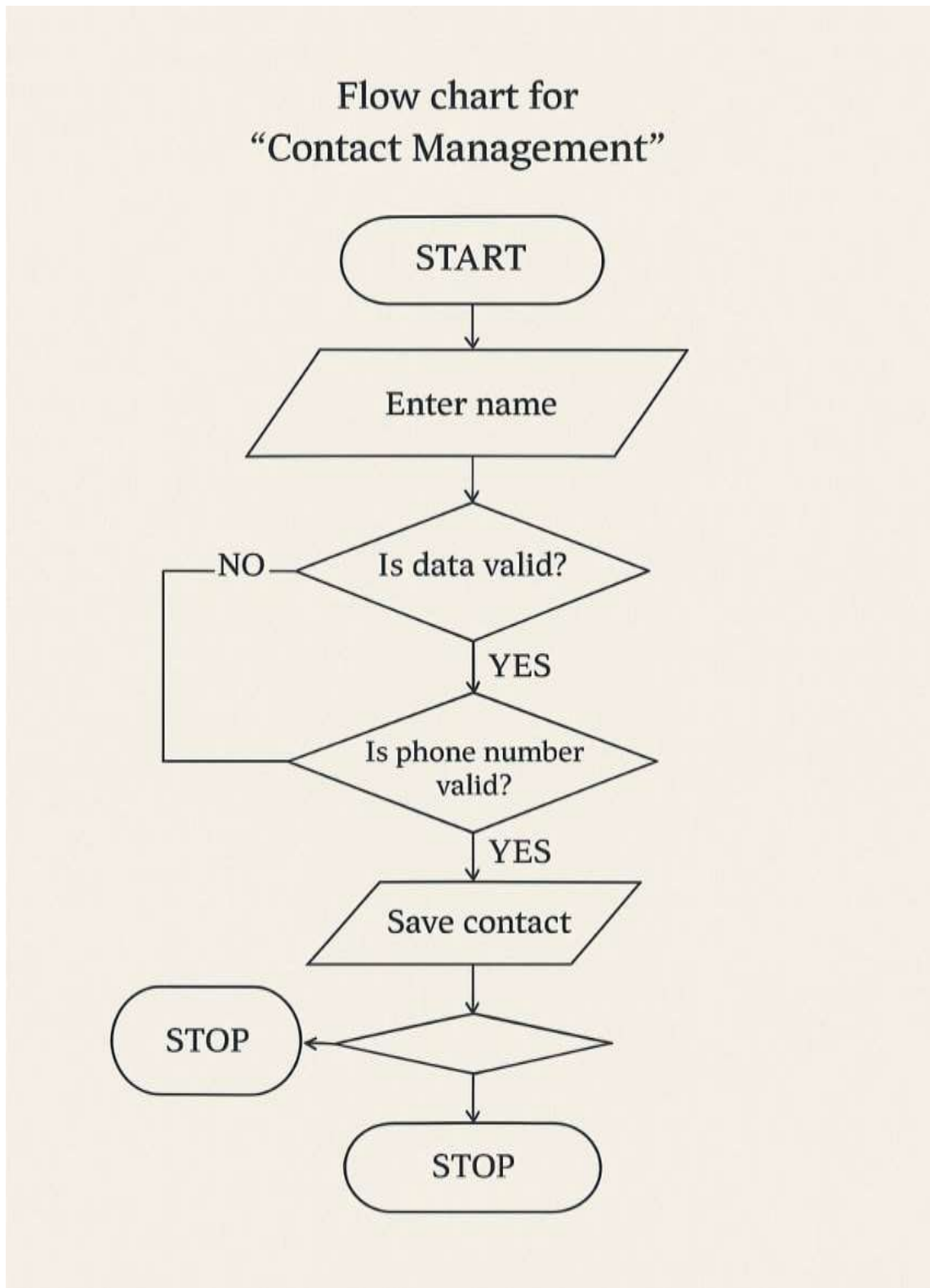
Flow chart for “User Registration”



Flow chart for “Selection Of Contact”

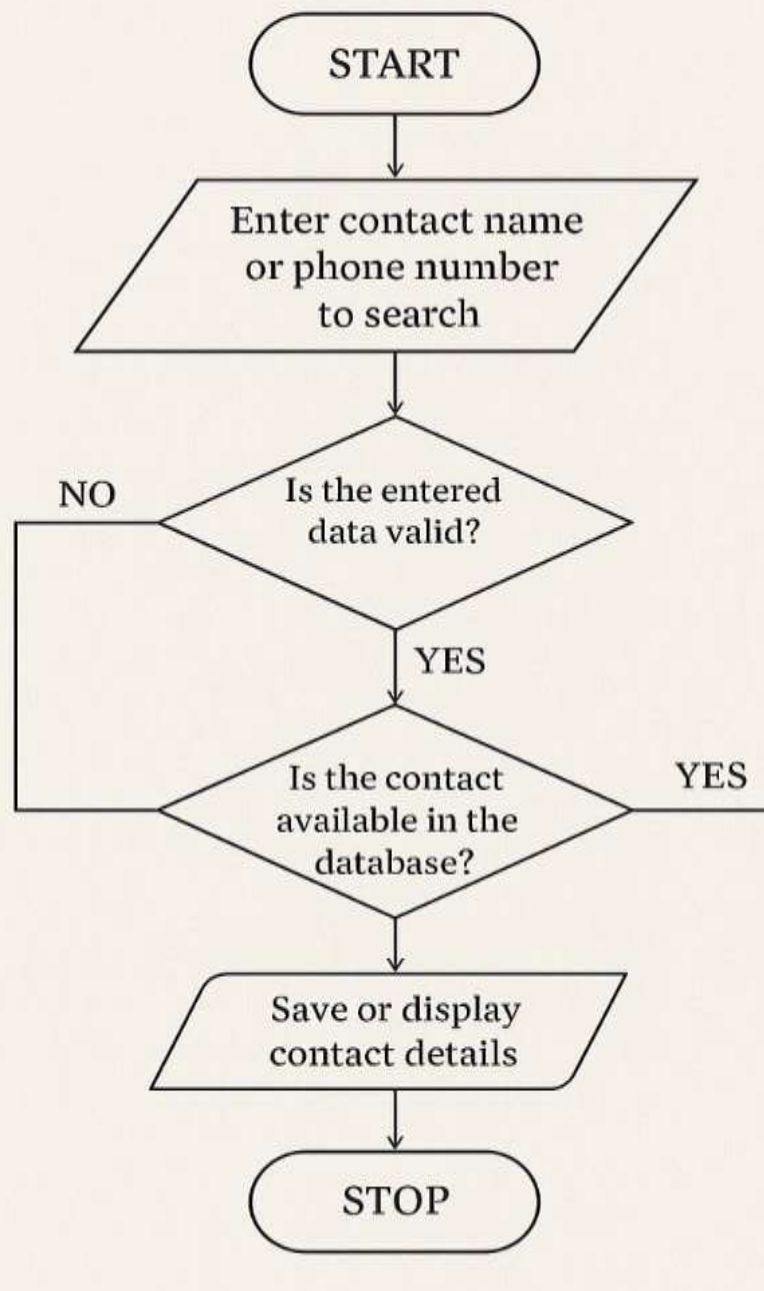


Flow chart for “Valid Data ”



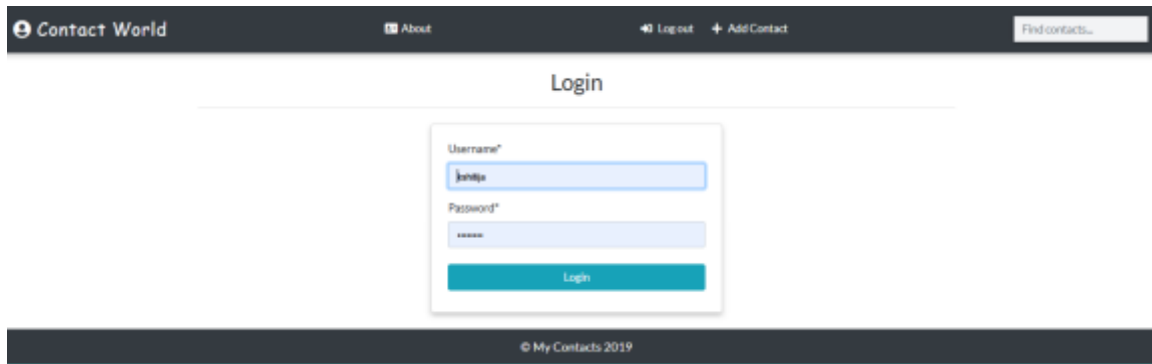
1. Flow chart for “Save and Display Contacts”

Flow chart for ‘Select Contact – Contact Management System



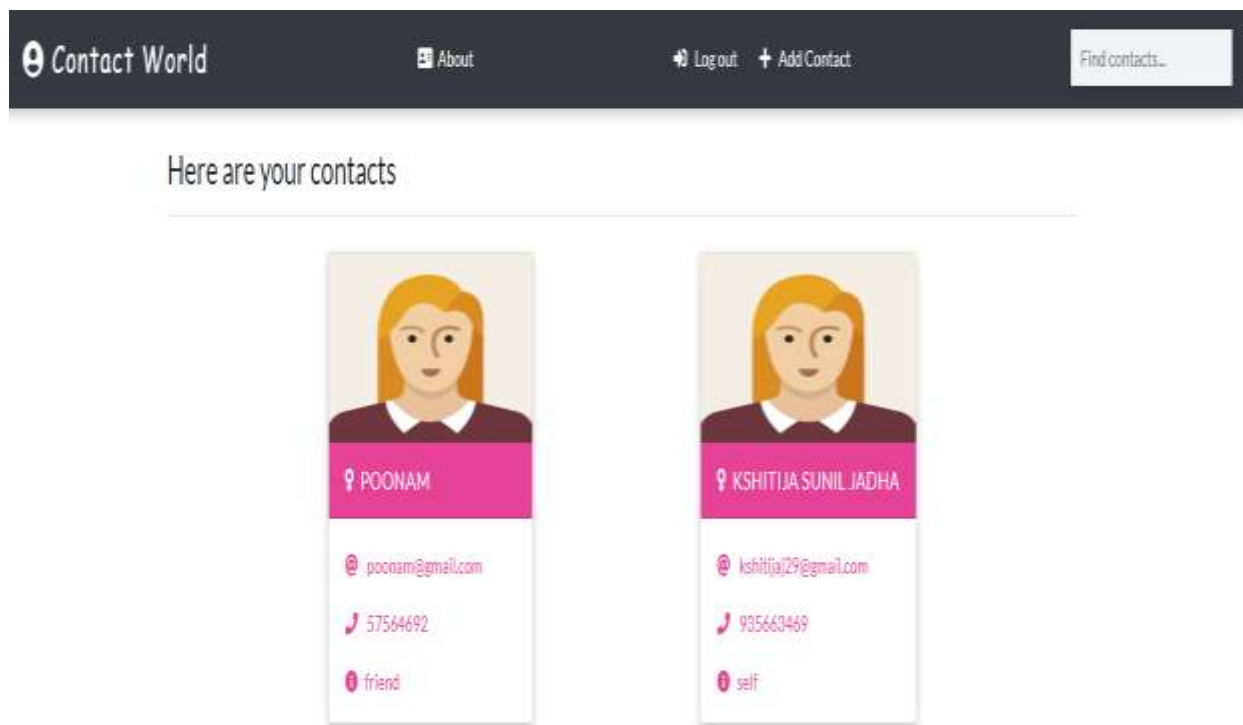
OUTPUT SCREENS

1. Login Page



The screenshot shows the login page of a web application. At the top is a dark navigation bar with the text "Contact World" on the left, "About" in the center, and "Log out" and "Add Contact" on the right. A search bar with the placeholder "Find contacts..." is on the far right. Below the navigation bar, the word "Login" is centered. Underneath, there is a white login form with two input fields: "Username*" containing the text "kshitija" and "Password*" containing six asterisks. A teal "Login" button is at the bottom of the form. At the very bottom of the page is a dark footer bar with the text "© My Contacts 2019".

2. Home Page



3. About



Contact World About Logout + Add Contact Find contacts...

Welcome to Contact World - Your Smart Contact Management Solution

Effortless Contact Management Made Simple

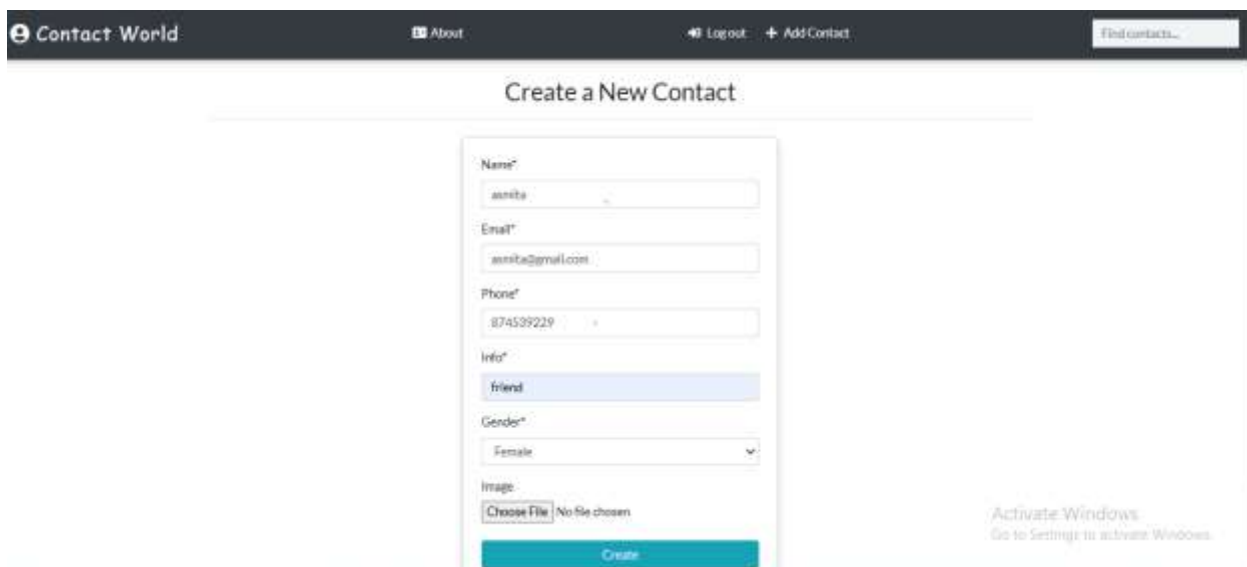
Contact World is a modern web application built with Django that helps individuals and teams organize their personal or business contacts with ease.

Store essential information like names, phone numbers, email addresses, and notes all in one place. Our platform allows you to search, update, and manage contacts securely and efficiently.

Designed for professionals, freelancers, and small businesses looking to streamline communication and keep relationships organized.

Explore App

4. Add Contact



Contact World About Logout + Add Contact Find contacts...

Create a New Contact

Name*
anita

Email*
anita@gmail.com

Phone*
874539229

Info*
friend

Gender*
Female

Image:
Choose File No file chosen

Create

Activate Windows
Go to Settings to activate Windows.

5. App Data Add Users:

The screenshot shows the 'Add contact' form in the 'Contacts Project' application. The left sidebar contains a search bar and two main sections: 'APP' with a 'Contacts' link and an '+ Add' button, and 'AUTHENTICATION AND AUTHORIZATION' with a 'Users' link and an '+ Add' button. The main form area is titled 'Add contact' and includes the following fields: 'Manager' (a dropdown menu showing 'admin_12' with edit and add icons), 'Name' (text input with 'admin'), 'Email' (text input with 'admin@gmail.com'), 'Phone' (text input with '0907399'), 'Info' (text input with 'friend'), 'Gender' (dropdown menu with 'Female'), 'Image' (button labeled 'Choose File' and text 'No file chosen'), and 'Date added' (date and time pickers showing '2020-05-24' and '12:51:21'). A note below the date pickers states 'Note: You are 0.5 hours behind server time.' At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'. An 'Activate Windows' watermark is visible in the bottom right corner.

AUTHENTICATION AND AUTHORIZATION OF USER :

The screenshot shows the 'Add user' form in the 'Contacts Project' application. The left sidebar is similar to the previous screenshot, but the 'Users' link under 'AUTHENTICATION AND AUTHORIZATION' is highlighted. The main form area is titled 'Add user' and includes the following fields: 'Username' (text input with 'admin', with a note 'Requires 128 characters or fewer. Letters, digits and @, ., +, =, _ are allowed'), 'Password-based authentication' (radio buttons for 'Enabled' and 'Disabled', with a note 'Whether the user will be able to authenticate using a password or not. If disabled, they may still be able to authenticate using other methods such as Single Sign-On or LDAP'), 'Password' (password input field with a note 'Your password must be 128 bits longer than your other personal information. Your password must contain at least 11 characters. Your password can't be a commonly used password. Your password can't be an email address.'), and 'Password confirmation' (password input field with a note 'Enter the same password as before, for verification.'). At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

LIST OF REPORTS THAT ARE LIKELY TO BE GENERATED

Following types of reports can be generated from the system:

1. Category-wise / Group-wise List of Contacts

View and manage contacts grouped by categories such as **Family, Friends, Work, Clients, Emergency**, etc.

Similar to genre-wise movie listings, this helps in organizing and filtering your contact list easily.

2. Organization-wise and Department-wise Contact Listings

Generate reports showing contacts from different **companies (brands)** and their **departments (branches)** such as *HR, Sales, Support*.

Useful in corporate environments where contacts belong to different teams or business units.

3. Daily / Weekly / Monthly / Yearly Contact Activity Trends

- Analyze how often new contacts are added or how frequently interactions (calls/messages/emails) are made over various timeframes.
- Helps track periods of high or low contact activity, similar to movie ticket booking trends

4. Payment-Mode Wise Contact Contributions (for business use)

- If your system tracks financial transactions (e.g., invoices or client payments), show stats by **Cash, Card, UPI, Bank Transfer**, etc.
- Equivalent to payment mode stats in a movie booking system.

5. Location-wise Contact Distribution or Engagement

- Show where most of your contacts are located or where you interact the most (e.g., based on city or region).
- Equivalent to viewer traffic reports in different locations.

CODING

Sample Code:

1.HTML File

Login.html

```
{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block title %}
Login
{% endblock title %}

{% block content %}
<div class="container">
  <h2 class="heading font-weight-light text-center">Login</h2>
  <hr>
</div>
<div class="container w-25 card p-4 mt-4">
  <form action="" method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form|crispy }}
    <input type="submit" value="Login" class="btn btn-info w-100 text-white text-center my-2">
  </form>
</div>
{% endblock content %}
```

Sign Up

```
{% endblock title %}

{% block content %}
<div class="container">
  <h2 class="heading font-weight-light text-center">Sign Up</h2>
  <hr>
</div>
<div class="container w-25 card p-4 mt-4">
```

```

<form action="" method="POST" enctype="multipart/form-data">
  {% csrf_token %}
  {{ form|crispy }}
  <input type="submit" value="Signup" class="btn btn-info w-100 text-white text-center my-2">
</form>
</div>
{% endblock content %}

```

About.html

```

{% extends 'base.html' %}
{% load static %}

{% block title %} About Contact World {% endblock title %}
{% extends 'base.html' %}
{% load static %}
{% block title %} About Contact World {% endblock title %}
{% block content %}
<div class="container my-4">
  <h4 class="text-info">Welcome to Contact World - Your Smart Contact
  Management Solution</h4>
</div>

<div class="row justify-content-center">
  <div class="col-md-6">
    <h2 class="featurette-heading text-success">
      Effortless Contact Management <span class="text-muted">Made
      Simple</span>
    </h2>
    <p class="lead mt-4">
      Contact World is a modern web application built with Django that helps
      individuals and teams organize their personal or business contacts with ease.
    </p>
    <p class="lead">
      Store essential information like names, phone numbers, email addresses, and
      notes all in one place. Our platform allows you to search, update, and manage
      contacts securely and efficiently.
    </p>

```

```

    <p class="lead">
      Designed for professionals, freelancers, and small businesses looking to
      streamline communication and keep relationships organized.
    </p>
  </div>
  <div class="col-md-4">
    
    <div class="my-3">
      <a href="{% url 'home' %}"><button class="btn btn-outline-dark"
type="button" name="button">Explore App</button></a>
    </div>
  </div>
</div>
{% endblock %}

```

Index.html

```

{% extends 'base.html' %}
{% load static %}
{% block title%}
My Contacts
{% endblock title %}

{% block content %}
<!-- Main Section -->
<div class="container">
  <h3 class="heading font-weight-light">Here are your contacts</h3>
  <hr />
</div>
<div class="container main-part">
  {% if contacts %}
  {% for contact in contacts %}
  {% include 'partials/_card.html' %}
  {% endfor %}

```


Django Files:

admin.py

```
from django.contrib import admin
from .models import Contact
from django.contrib.auth.models import Group
from import_export.admin import ImportExportModelAdmin

# Register your models here.
class ContactAdmin(ImportExportModelAdmin): # 'id', it will create unique id for
every object.
    list_display = ('id','name','gender','email','info','phone') # It will display all the
columns.
    list_display_links = ('id','name')
    list_editable = ('info',) # It will give you the editing opportunity
    list_per_page = 10 # It will show the item you give.Like, you gave 10.That
means you will can see the 10 items of all the items.
    search_fields = ('name','gender','email','info','phone') # It will create a search
filed
    list_filter = ('gender','date_added') # It will allow you to filtering each item.

admin.site.register(Contact, ContactAdmin) # It will register these items in admin
panel
admin.site.unregister (Group) # It will remove the Group section from
Authentication and authorization in admin panel.
```

Models.py

```
from django.db import models
from django.utils.timezone import datetime
from django.contrib.auth.models import User

# Create your models here.
class Contact(models.Model):
    manager = models.ForeignKey(User, on_delete=models.CASCADE,
default=None) # When the user is deleted it will clear the contact
    name = models.CharField(max_length=20)
    email = models.EmailField(max_length=100)
```



```

phone = models.IntegerField()
info = models.CharField(max_length=30)
gender = models.CharField(max_length=50, choices=(
    ('male','Male'),
    ('female','Female')
))
image = models.ImageField(upload_to='images/', blank=True)
date_added = models.DateTimeField(default=datetime.now)

def __str__(self):
    return self.name

class Meta:
    ordering = ['-id']  ## Ordering the display items by id

```

urls.py

```

from django.urls import path
from . import views

urlpatterns = [
    # path("", views.home, name = 'home'),
    path("", views.HomePageView.as_view(), name = 'home'),
    # path('detail/<int:id>/', views.detail, name='detail'),
    path('detail/<int:pk>/', views.ContactDetailView.as_view(), name="detail"),
    path('search/', views.search, name = "search"),
    path('contacts/create', views.ContactCreateView.as_view(), name = "create"),
    path('contacts/update/<int:pk>', views.ContactUpdateView.as_view(), name = "update"),
    path('contacts/delete/<int:pk>', views.ContactDeleteView.as_view(), name = "delete"),
    path('signup/', views.SignUpView.as_view(), name = "signup"),
    path('about/', views.about, name='about'),
]

```

Views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Contact
from django.views.generic import ListView, DetailView
from django.db.models import Q # Q for multiple search
from django.views.generic.edit import CreateView, UpdateView, DeleteView
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.mixins import LoginRequiredMixin # It will making views to
require login to acces.
from django.contrib.auth.decorators import login_required # It will making search to require
login to acces.
from django.urls import reverse_lazy
from django.contrib import messages

# Create your views here.
# def home(request):
#     context = {'contacts': Contact.objects.all()}
#     return render(request, 'index.html', context)

# def detail(request, id):
#     context = {'contact': get_object_or_404(Contact, pk=id)}
#     return render(request, 'detail.html', context)

####--##--#### Home view Start ####--##--####
# By using LoginRequiredMixin, You must have to login first to enter the website.
class HomePageView(LoginRequiredMixin, ListView):
    template_name = 'index.html'
    model = Contact
    context_object_name = 'contacts'

    def get_queryset(self): # Showing contacts of logged in user only. By using this function.
        contacts = super().get_queryset()
        return contacts.filter(manager = self.request.user)
####--##--#### Home view End ####--##--####
```

```
###--##--### Contact Details View Start ###--##--###
```

```
class ContactDetailView(LoginRequiredMixin, DetailView):
```

```
    template_name = 'detail.html'
```

```
    model = Contact
```

```
    context_object_name = 'contact'
```

```
###--##--### Contact Details View End ###--##--###
```

```
###--##--### Search Start ###--##--###
```

```
@login_required # By using @login_required, user can't search anything without login
```

```
def search(request):
```

```
    if request.GET:
```

```
        search_term = request.GET['search_term']
```

```
        search_results = Contact.objects.filter(
```

```
            Q(name__icontains=search_term) |
```

```
            Q(email__icontains=search_term) |
```

```
            Q(info__icontains=search_term) |
```

```
            Q(phone__iexact=search_term) # when iexact is used, we need to search by exact value
```

```
        )
```

```
        context = { 'search_term':search_term, 'contacts':search_results.filter(manager=request.user)
```

```
        } # By using filter, only logged in user can search their contacts.
```

```
        return render(request, 'search.html', context)
```

```
    else:
```

```
        return redirect('home')
```

```
###--##--### Search End ###--##--###
```

```
###--##--### Contact Create View Start ###--##--###
```

```
class ContactCreateView(LoginRequiredMixin, CreateView):
```

```
    model = Contact
```

```
    template_name = 'create.html'
```

```
    fields = ['name','email','phone','info','gender','image']
```

```
    # success_url = '/'
```

```
    def form_valid(self,form):
```

```
        instance = form.save(commit=False)
```

```
        instance.manager = self.request.user
```

```
        instance.save()
```

```
        messages.success(self.request, 'Your contact has been successfully created!')
```

```
        return redirect('home')
```

```
###--##--### Contact Create View End ###--##--###
```

```
###--##--### Contact Update View Start ###--##--###
```

```
class ContactUpdateView(LoginRequiredMixin, UpdateView):  
    model = Contact  
    template_name = 'update.html'  
    fields = ['name','email','phone','info','gender','image']  
    # success_url = '/'  
    def form_valid(self,form):
```

Manage.py

```
def main():  
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'contacts.settings')  
    try:  
        from django.core.management import execute_from_command_line  
    except ImportError as exc:  
        raise ImportError(  
            "Couldn't import Django. Are you sure it's installed and "  
            "available on your PYTHONPATH environment variable? Did you "  
            "forget to activate a virtual environment?"  
        ) from exc  
    execute_from_command_line(sys.argv)  
  
if __name__ == '__main__':  
    main()
```

□ PARAMETER PASSING

Function Name	Description
<u>mysql_close</u>	Close MySQL connection
<u>mysql_connect</u>	Open a connection to a MySQL Server
<u>mysql_create_db</u>	Create a MySQL database
<u>mysql_db_name</u>	Retrieves database name from the call to <u>mysql_list_dbs</u>
<u>mysql_db_query</u>	Selects a database and executes a query on it
<u>mysql_errno</u>	Returns the numerical value of the error message from previous MySQL operation
<u>mysql_error</u>	Returns the text of the error message from previous MySQL operation
<u>mysql_fetch_array</u>	Fetch a result row as an associative array, a numeric array, or both
<u>mysql_fetch_assoc</u>	Fetch a result row as an associative array



Testing Strategy Overview:

Definition of Testing:

- Testing is a process to provide stakeholders with information about the quality of a software product or service.
- It offers an independent view of the software to help stakeholders understand risks during implementation.
- It involves executing a program/application to detect software bugs.

Purpose of Software Testing:

- To validate and verify that a software program:
 1. Meets business and technical requirements.
 2. Works as expected.
 3. Can be implemented with consistent results.

When Testing is Done:

- Testing can be done at any point in the development process.
- Typically, most testing is done after requirements are defined and coding is complete.

Testing in Development Models:

- **Traditional models:** Testing happens after the development phase.
- **Agile and modern models:** Emphasize **test-driven development** where developers do more testing during coding before formal testers are involved.

Limitations of Testing:

- Testing can't find every defect.
- It acts as a **criticism or comparison**, checking if the software behaves correctly based on defined expectations (called oracles).

Testing methods

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

In unit testing, the smallest unit of the software, i.e. modules, is tested. This test focuses on the internal process logic and data structure within the boundaries of a component.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is also called **component testing**.

UNIT TESTING

Test Case # 1.

Create a New Contact

Name*

Kshitija Sunil Jadha

Email*

Phone*

9356634692

Info*

self

Gender*

Female

Image

Choose File

No file chosen

Create

! Please fill out this field.

Create a New Contact

Name*

Kshitija Sunil Jadha

Email*

| .

Phone*



Please fill out this field.

9356634692

Info*

self

Gender*

Female



Image

Choose File No file chosen






Create

Test Case#2

Change contact

poonam

Please correct the error below.

Manager:	kshitija	  
Name:	poonam	
Email:	poonam@gmail.com	
Phone:	59086	
Info:	This field is required. <div></div>	
Gender:	Female	
Image:	<div>Choose File</div> No file chosen	
Date added:	Date: 2025-05-14 Today  Time: 13:40:15 Now 	

Test Case #3

Email Address Field Validation

Create a New Contact

Name*

Kshitija Sunil Jadha

Email*

kshitijaj29gmail.com

9356634692

Info*

self

Gender*

Female

Image

Choose File

No file chosen

Create

Please include an '@' in the email address. 'kshitijaj29gmail.com' is missing an '@'.



This software itself being a security project provides some of the best software security measures. In this website a user cannot login and use the features and facilities provided by the website without creating user account. Instead of that every time the user wants to login he has to give his user name and password which adds to the security of the project. Moreover, all the data that are saved using this software are encrypted i.e. no unauthorized user cannot access it.

Security Issues of ISO 74982:

➤ Authentication:

It is any process by which a system verifies the identity of a user who wishes to access the resource stored in system.

➤ Access Control:

Access control is ability to permit or deny the use of a particular resource by particular entity.

➤ Confidentiality:

The website we will develop will achieve data security through Encryption/Decryption concept.

➤ Data Integrity:

With the help of this website we can handle data accessing, webpage bugs can be deleted and improved later on.

➤ Non-repudiation:

❖ This website will be user specific i.e. only some user can access specific commands buttons.

❖ The website will use security policy which specifies who is authorized to do the task.

Data Integrity, Non-repudiation

The security is the most important factor of any project. Here the main concern is that the project is based on Online that means the security part should be very high in order to protect the system. The security system maintains the data integrity of the whole system. The security system is

maintained by allowing the user to enter a login id and password to login to the system.

1. Site Administrator: *Site Administrator is the actually Super user of this site that means s/he can only access the all features provided by the site. Administrator also can manipulate the database. S/he also can view the all reports generated by the System.*

INTERFACE SECURITY

The site Administrator is only the super user of this website. S/he can only access all the features of the site and also can view all system generated reports. As a other user can just only view the site as a normal site viewer. Can only get their own profile.

Therefore, for Site Administrator there is Login Profile procedure. Admin user can access the site by providing valid login profile through the following login Page:

Login Page

The screenshot shows a web form titled "Add user". Below the title, it says "After you've created a user, you'll be able to edit more user options." A red-bordered box contains the message "Please correct the error below:". The form has three main sections: "Username:" with a text input containing "kajal" and a note "Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only."; "Password-based authentication:" with radio buttons for "Enabled" (selected) and "Disabled", and a note "Whether the user will be able to authenticate using a password or not. If disabled, they may still be able to authenticate using other methods, such as Single Sign-On or LDAP."; and "Password:" with a text input containing "*****" and a note "Your password can't be too similar to your other personal information. Your password must contain at least 8 characters. Your password can't be a commonly used password. Your password can't be entirely numeric." Below the password field is a "Password confirmation:" section with a text input and a note "The two password fields didn't match. Enter the same password as before, for verification." At the bottom, there are three buttons: "SAVE", "Save and add another", and "Save and continue editing". In the bottom right corner, there is a watermark that says "Activate Windows Go to Settings to activate Windows."

DATABASE SECURITY

Database is the storage device. That means the data is getting stored in the database server. Thus it is very important to secure the data residing in the database. Thus every database server provides a login procedure. This technique is similar to application level security. The entity which is willing to use the database has to

supply in a login ID and password to access and manipulate the data in the database. Every database server provides a default login ID and password to access the data. Apart from this the database administrator can also create other database user.

In this project the default login ID and password has been used. Since the database server used in this project is **MYSQL** thus the default login ID and password is:

Login id: root

password: security

Hostname: localhost

Database Connectivity

```
import mysql.connector

conn = mysql.connector.connect(

    host="localhost",

    user="root",

    password="", # Use your password if not blank

    database="contact_db" # Replace with your actual database name

)

# Create a cursor object to execute SQL commands

cursor = conn.cursor()

# Check connection

if conn.is_connected():

    print("Connected to MySQL database successfully.")

else:

    print("Failed to connect.")
```



Future Scope

The project named as “**Contact Management System**” considers a web application; this has great potential in organizations and businesses that deal with a large number of contacts. Many firms still manage their contact data manually or via spreadsheets. With this system, they can be offered a web-based solution customized to their requirements.

Today, managing contacts through digital systems is becoming the norm. With the evolution of software technology, manual data entry, duplication, and loss of information are becoming issues of the past. People now prefer quick, reliable, and secure ways to handle contact data—whether for clients, vendors, or internal use.

This project is designed to meet the requirements of efficient contact management. It has been developed using **Python with Django** as the backend framework and **MySQL** as the database, keeping in mind the user-friendly specifications of the system.

In our project: with this contact management solution, organizations can ensure better data organization, improved communication, and enhanced productivity. The relationship between employees and customers is made more efficient by ensuring accurate and up-to-date contact information.

With this platform we developed, we aim to reduce mismanagement of contact data, improve data access, increase user satisfaction, and minimize manual workload. We believe that we have met our development goals and are confident in the usability of the system.

Further Enhancements

- ➤ Integration of **User Role Management**, allowing different levels of access such as Admin, Manager, and Regular User.
- ➤ Adding **Search and Filter features** using Django QuerySets and Ajax for real-time filtering.

- ➤ Providing options to **export data** in CSV/PDF formats for reporting and analysis.
- ➤ **SMS/Email notification integration** to alert users when contact data is updated or added.
- ➤ **Data Backup & Restore Module** to avoid accidental data loss and ensure security.



Henceforth, we have reached the end of our project where we came across various different sections of it. For putting up this whole project, many references have been taken. Proper guidance and helpful documentation have contributed significantly to enhance the project development process.

- **Django Documentation** – Official documentation for Django web framework. Retrieved from: <https://docs.djangoproject.com>
- **Instamojo API Documentation** – Reference for integrating payment gateway using Instamojo APIs. Retrieved from: <https://www.instamojo.com/developers/>
- **Bootstrap Documentation** – Official guide for Bootstrap front-end framework. Retrieved from: <https://getbootstrap.com/docs/>
- **Stack Overflow** – Community-driven programming solutions and technical discussions. Retrieved from: <https://stackoverflow.com>
- **HTML & CSS (W3Schools)** – Tutorials and examples used for front-end development.
<https://www.w3schools.com>

REFERENCES

- [Django Official Website](#) – Backend framework used.
- [Bootstrap Official Website](#) – Front-end styling and layout.
- [Instamojo Developer Portal](#) – Payment gateway integration.
- [MySQL Official Website](#) – Relational database used for data storage.