INDIRA GANDHI NATIONAL OPEN UNIVERSITY

PROJECT REPORT
ON

TAABIR: The Crowdfunding Website
By

Adhyatamjot Singh
Enrollment Number: 2105339859

Under Guidance of
Dr. Neeta

Submitted to the School of Computer and Information Sciences
In partial fulfilment of the requirements
For the degree of
Bachelor of Computer Applications

Indira Gandhi National Open University
Maidan Garhi

Approval letter

Registration page

Guide documen

s

INDIRA GANDHI NATIONAL OPEN UNIVERSITY

SYNOPSIS
ON

TAABIR: The Crowdfunding Website
By

Adhyatamjot Singh
Enrollment Number: 2105339859

Under Guidance of
Dr. Neeta

Submitted to the School of Computer and Information Sciences
In partial fulfilment of the requirements
For the degree of
Bachelor of Computer Applications

Indira Gandhi National Open University
Maidan Garhi

PROJECT TITLE

TAABIR

The Crowdfunding Website

INDEX

INTRODUCTION AND OBJECTIVE

INTRODUCTION

Taabir is a crowdfunding platform that enables non-profit organizations and individuals to raise funds for causes that matter to them. Our goal is to create a world where everyone can make a positive impact by giving back to society. Through Taabir, we empower individuals to support the causes they care about, and help non-profits reach their fundraising goals. Together, we can make a real difference in the world.

OBJECTIVE

The objective of Taabir is to connect people and organizations who share a passion for social good, and empower them to make a positive impact in the world. By offering a simple, secure, and user-friendly platform for crowdfunding, we aim to democratize philanthropy and provide a new avenue for individuals to give back to the causes they care about. At Taabir, we believe that every donation, no matter how small, can make a real difference. We strive to create a world where generosity and compassion are the norm, and where every person can make a meaningful impact.

.

PROJECT CATEGORY

This application is developed using three tier architecture of Java and J2EE Technologies, HTML, CSS, JavaScript, Angular is used for front end designing & development and MySQL, Java Servlets are used as a back end or database. The connection between the front-end and the back-end is established by using JDBC-ODBC Bridge for MySQL Database.
Some style sheets and scripting technologies are used to enhance the dynamics of the project.

Frontend Stack: HTML, CSS, JavaScript with TypeScript and Angular
Backend Stack: MySQL, Java J2EE, Java Servlets, JDBC, JSP.

HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements: -

- Intel Core i3-1315U with 8 threads and 10MB L3 cache.
- 4GB RAM.
- 80 GB Hard Disk Drive.
- 3.5" – 1.44 MB Diskette Drive.
- 52X CD-ROM Drive.
- Intel® Pro 10/100+ LAN Card..
- The computer should have a proper backup and recovery facility.

Software Requirements: -

- Node JS (Runtime for Angular)
- Angular CLI (Angular CLI tool)
- MySQL (Database Software)
- Java JDK (to run Java)
- Eclipse IDE (To code Java Servlets)
- VS Code IDE (To code Frontend, mainly HTML, CSS and JavaScript with Angular)
- Browser (Chrome / Firefox :- To test the Application)

REQUIREMENTS SPECIFICATIONS PROJECT PLANNING & SCHEDULING

SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements, and may include a set of use cases that describe interactions the users will have with the software.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

- ## Software Development Process Model
  The incremental model is a software development process that breaks down a project into smaller, more manageable chunks, or increments. Each increment builds upon the previous one, adding functionality and features in a step-by-step manner. The process involves multiple phases: Requirements gathering and analysis, design, implementation, testing, and evaluation.

1. Introduction
Purpose
The primary goal of this Software Requirements Specification (SRS) document is to provide a comprehensive outline of the Taabir crowdfunding platform's functional and non-functional requirements.
Scope
This SRS focuses on the development of Taabir, a web-based crowdfunding platform for medical and NGO projects, built using Angular, Java Servlets (J2EE), and MySQL database.

Definitions, Acronyms, and Abbreviations

- Taabir: Crowdfunding platform
- J2EE: Java 2 Enterprise Edition
- MySQL: Open-source relational database management system
- SRS: Software Requirements Specification

2. Overall Description

Product Perspective

Taabir is an independent crowdfunding platform that connects project creators with potential supporters for medical and NGO causes.

Product Functions

Taabir enables users to create and manage projects, discover and support campaigns, and facilitate secure financial transactions between parties.

User Characteristics

Taabir serves individuals and organizations seeking financial support for medical and NGO projects, as well as contributors looking to fund these initiatives.

Constraints

Taabir must comply with relevant crowdfunding regulations and integrate with third-party payment gateways for secure transactions.

Assumptions and Dependencies

Users are expected to have compatible devices and internet connectivity, as well as a basic understanding of crowdfunding concepts.

3. Specific Requirements

Functional Requirements

- User Registration: Users should be able to create and manage their accounts.
- Project Creation: Users should be able to create and publish crowdfunding projects.
- Project Browsing: Users should be able to browse and filter available projects.
- Financial Transactions: Users should be able to contribute funds to projects securely.
- Communication: Users should be able to communicate with project creators and other supporters.

Non-Functional Requirements

- Performance: Taabir should load within 3 seconds and support up to 1000 concurrent users.
- Security: User data and transactions should be secure and encrypted.
- Usability: Taabir should have a user-friendly interface and intuitive navigation.

MODULES & TIME ESTIMATION

Modules and Processes

1. Actor: Admin

   a. **Module 1: Admin Login**
   - **Input-** Email, Password
   - **Process -** Login form submit
   - **Output**: Success / Failure

   b. **Module 2: Verify Fundraising Requests**
   - **Input-** Verified or Rejected
   - **Process -** Verify option
   - **Output**: Success / Failure

   c. **Module 3: Verify / Manage Users**
   - **Input-** Verified or Rejected
   - **Process -** Verify option
   - **Output**: Success / Failure

   d. **Module 4: Verify Payment Source**
   - **Input-** Verified or Rejected
   - **Process -** Verify option
   - **Output**: Success / Failure

   e. **Module 5: Create FAQs**

   - **Input-** FAQ title, FAQ Description, FAQ active
   - **Process -** FAQ form submit
   - **Output**: Success / Failure

2. Actor: User

   a. **Module 1: Registration**
   - **Input-** Name, email, mobile number
   - **Process -** Registration Form
   - **Output**: Success / Failure

   b. **Module 2: Login**

- **Input-** Email, Password
- **Process -** Login Form
- **Output**: Success / Failure

   c. **Module 3: Create Fundraising Requests**
- **Input-** Amount, reason,
- **Process -** Fundraising request Form
- **Output**: Success / Failure

   d. **Module 4: Fundraise contribution**
- **Input-** Amount, Misc info
- **Process -** Fund Raising Contribute form
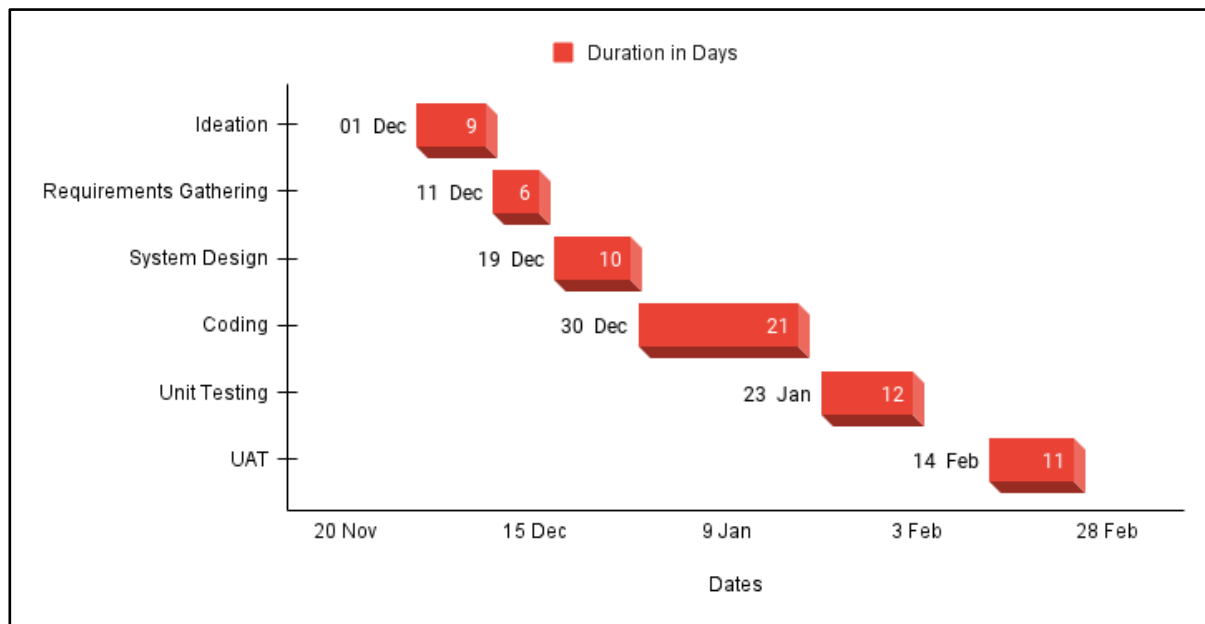- **Output**: Success / Failure

3. <u>Actor: Guest</u>

   a. **Module 1: Guest Fundraise contribution**
- **Input-** Amount to contribute, name, PAN Card
- **Process -** Contribute to Funds Form
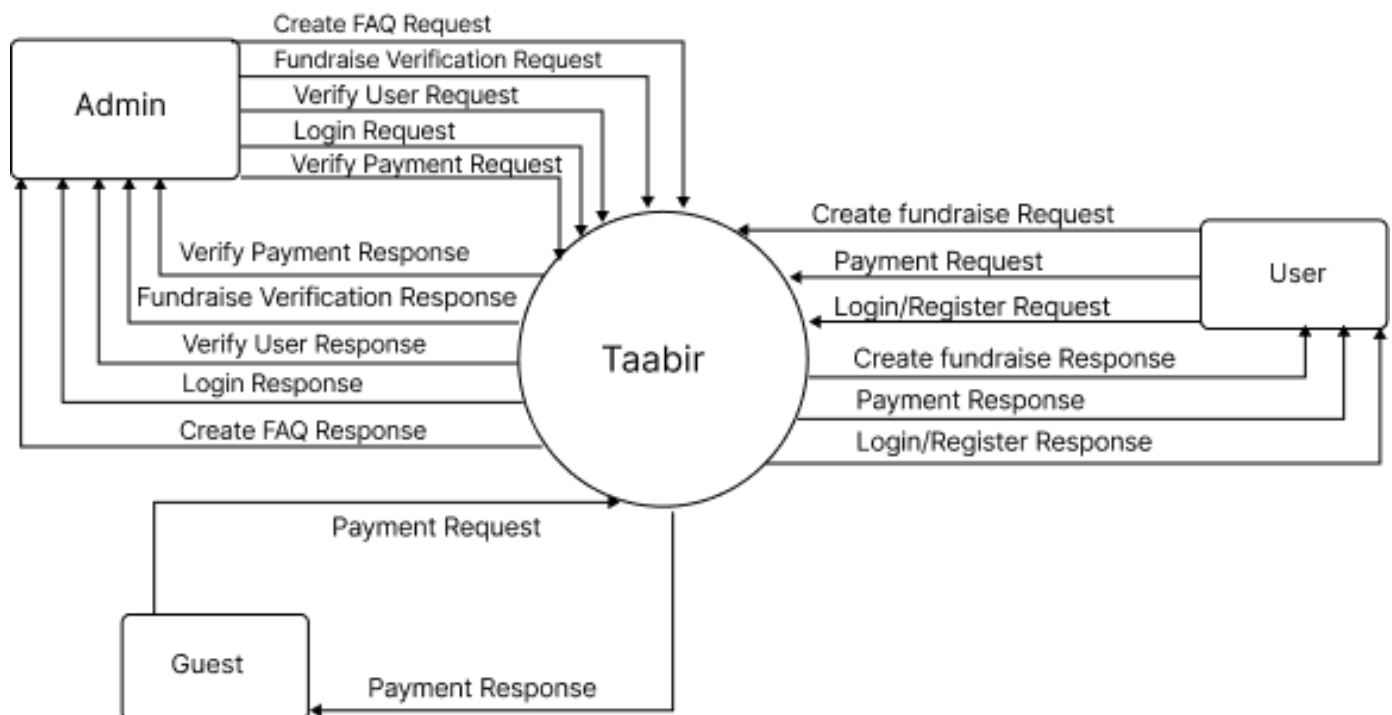- **Output**: Success / Failure

Time Estimation: Gantt Chart

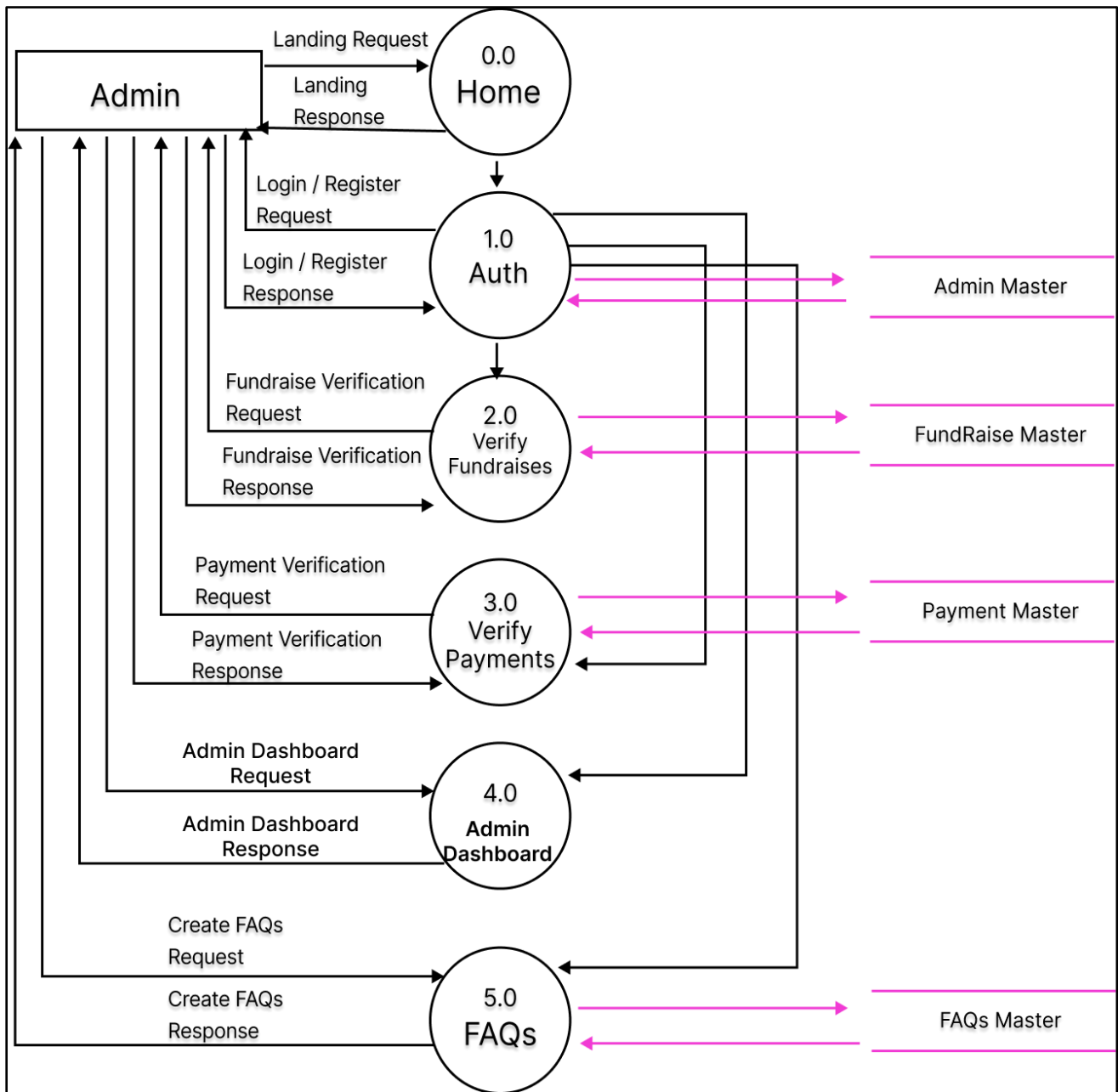- ## <u>Gantt Chart</u>
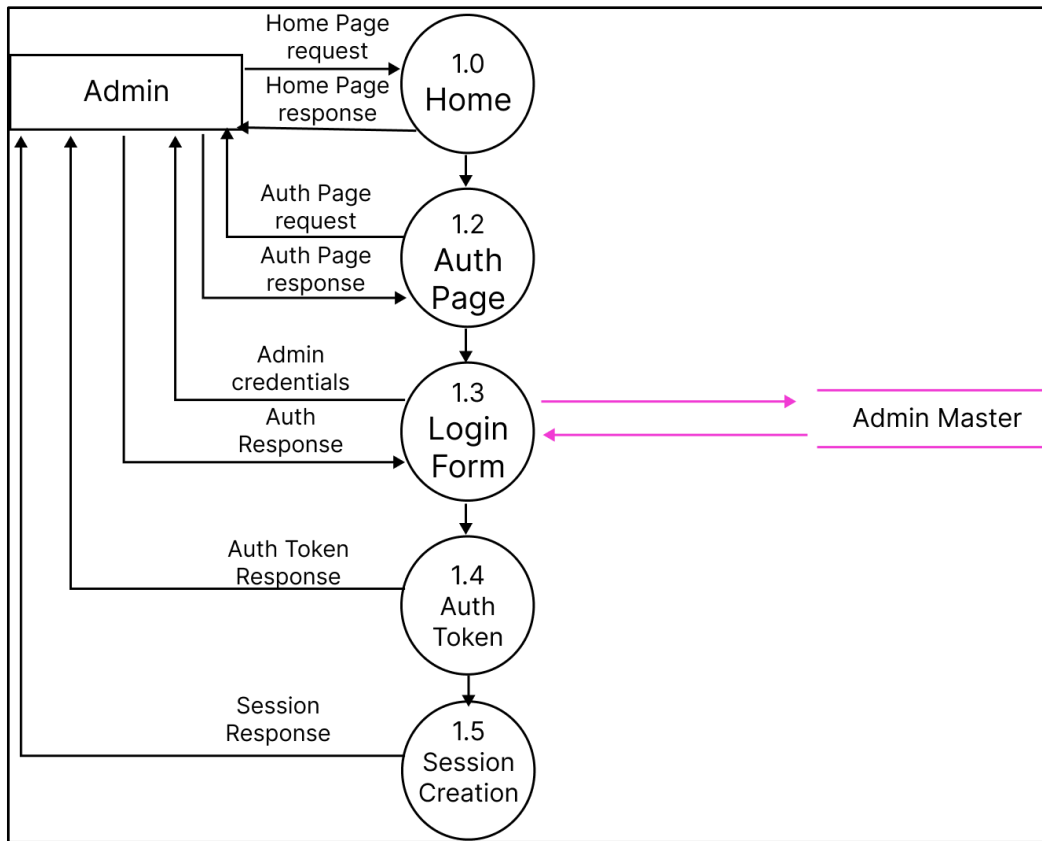

Gantt Chart

DATA FLOW DIAGRAM
1. **Level 0 DFD**

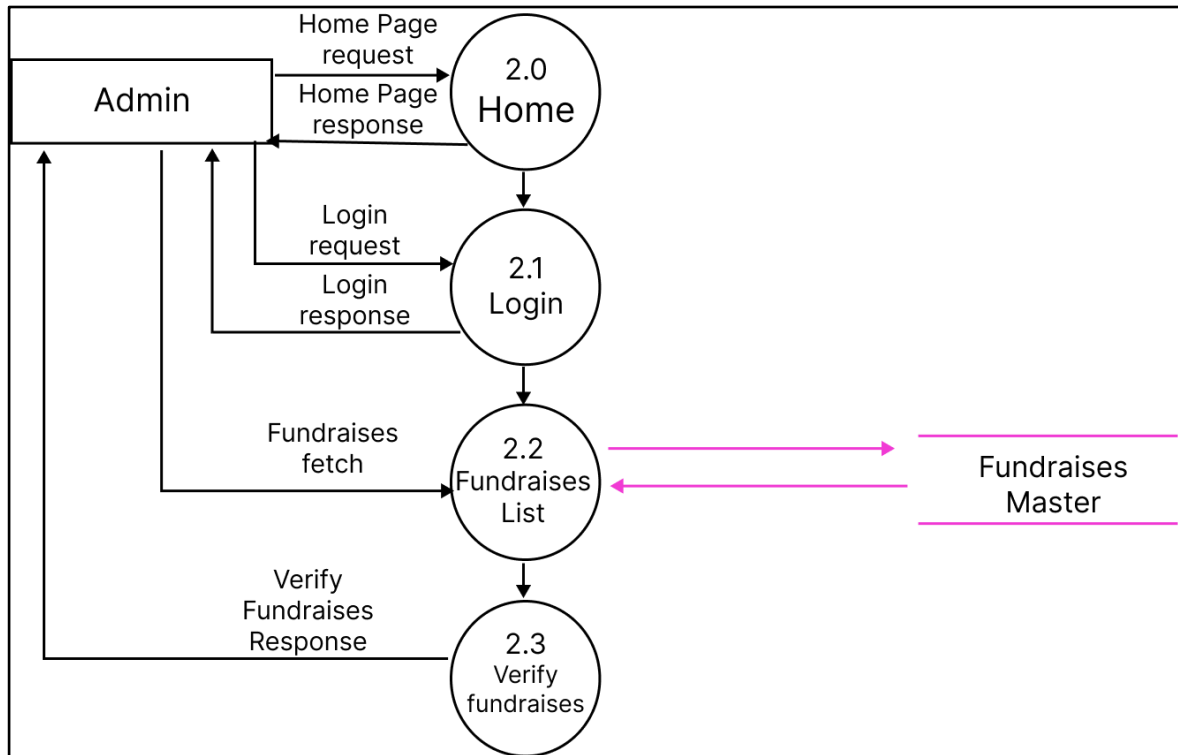*Level 0 DFD : Admin , User and Guest*

2. **Admin DFDs**
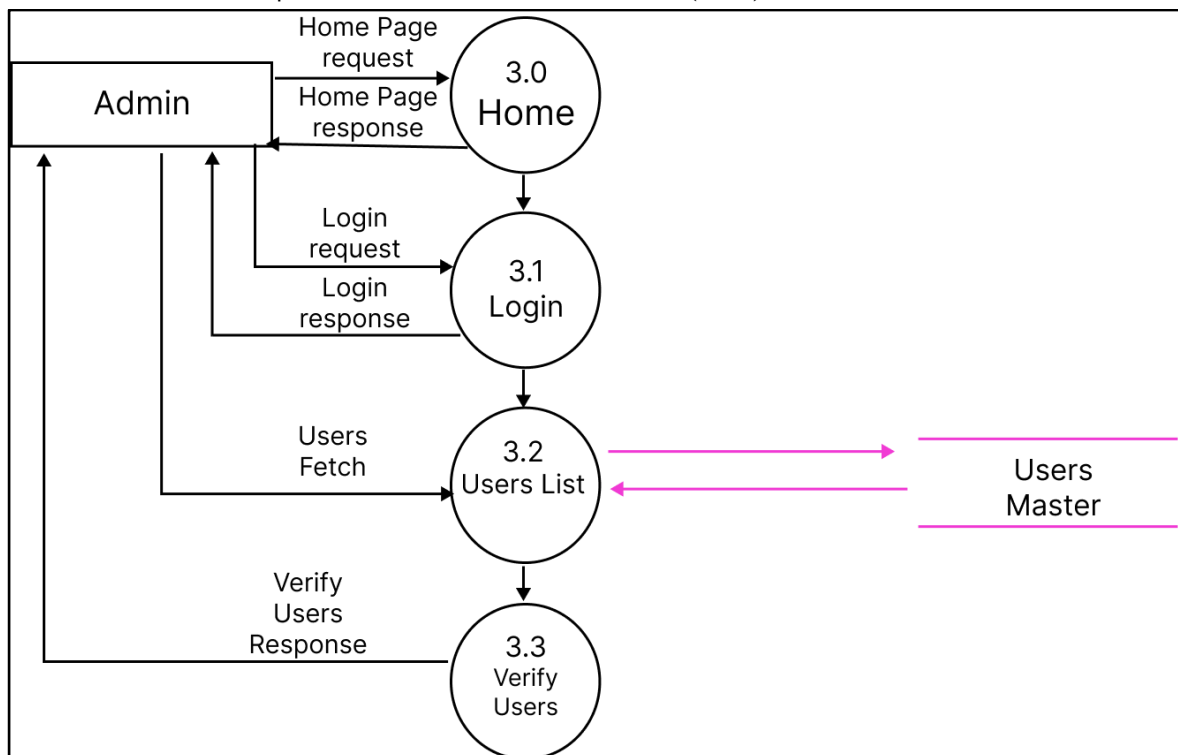   a. **Level 1 DFD**

Level 2 DFD | Admin auth Module(1.0)

b. **Level 2 DFD | Verification of Fundraises (2.0)**

c. **Level 2 DFD | Verification of Users Module(3.0)**

d. **Level 2 DFD | Verification of Payments Module(4.0)**
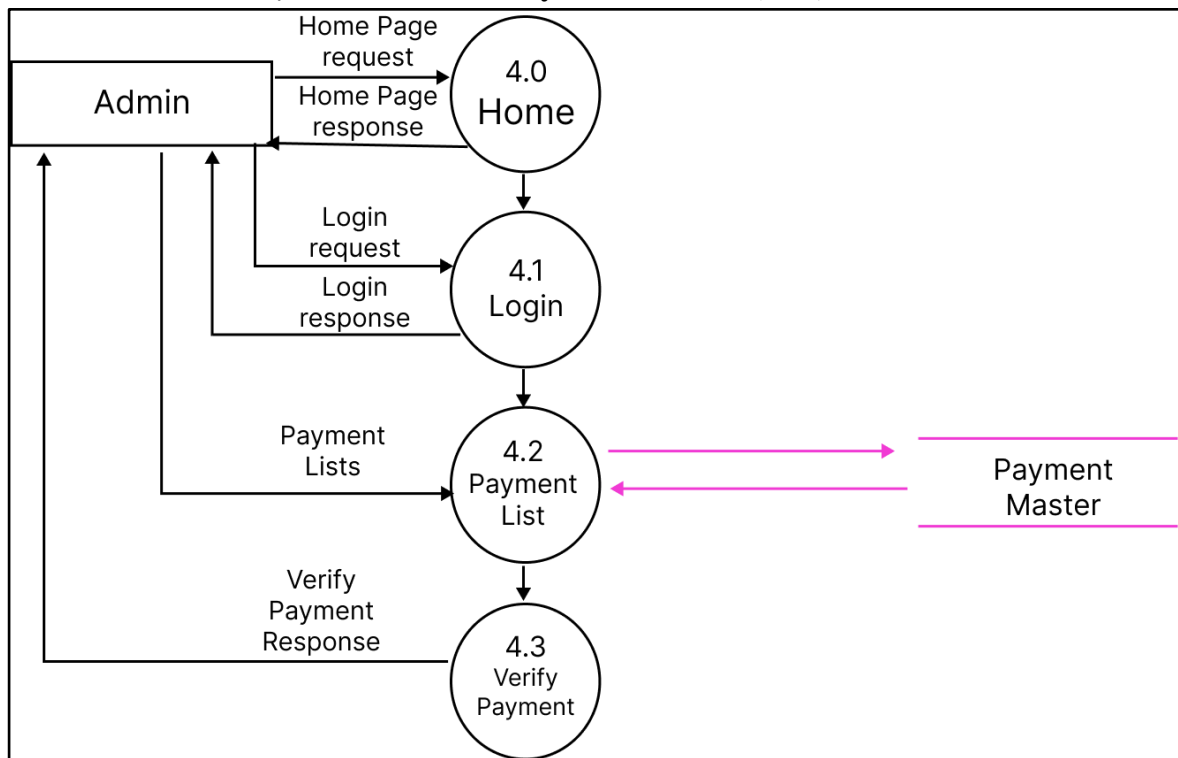
e. **Level 2 DFD | Management of FAQs (5.0)**
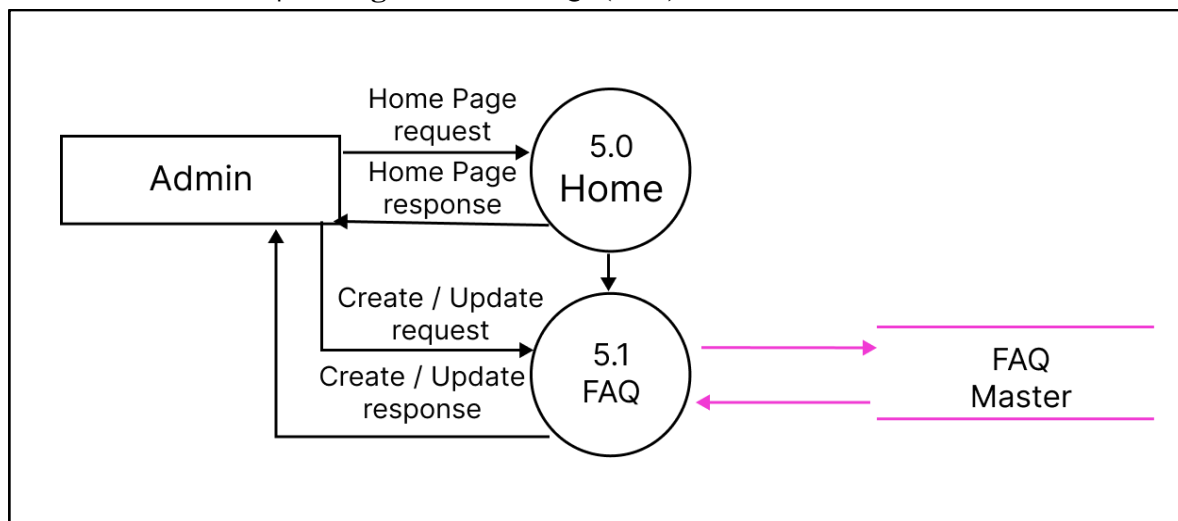
3. **Users DFDs**
   a. **Level 1 DFD**

   b. **Level 2 DFD: User Auth (Login / Register) Module (1.0)**

c. **Level 2 DFD: User Module Fundraiser payment**
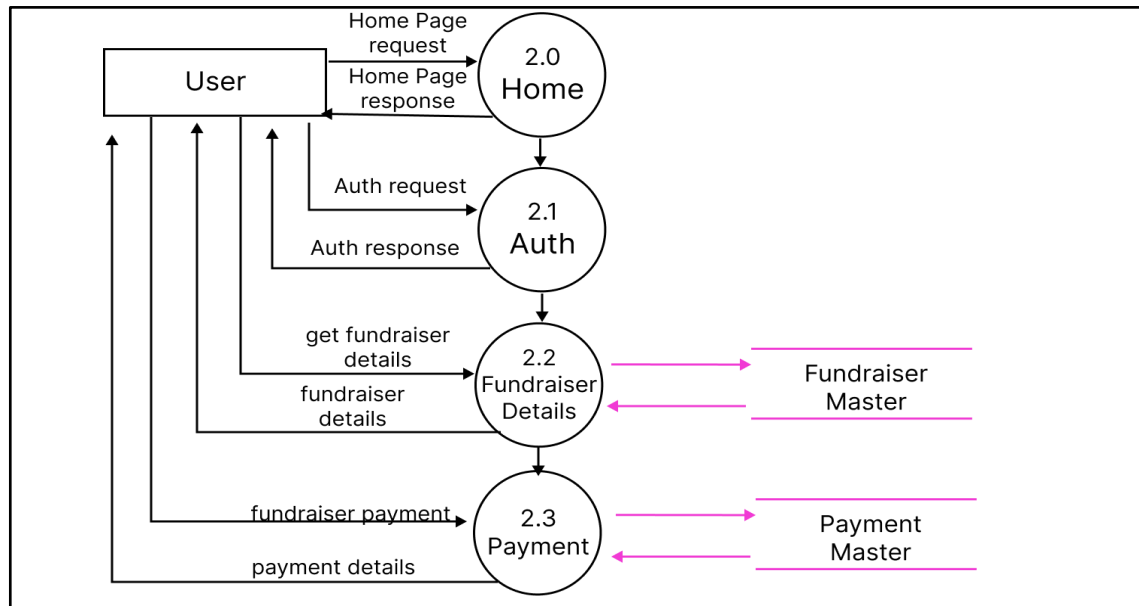
d. **Level 2 DFD: User Fundraiser Creation**
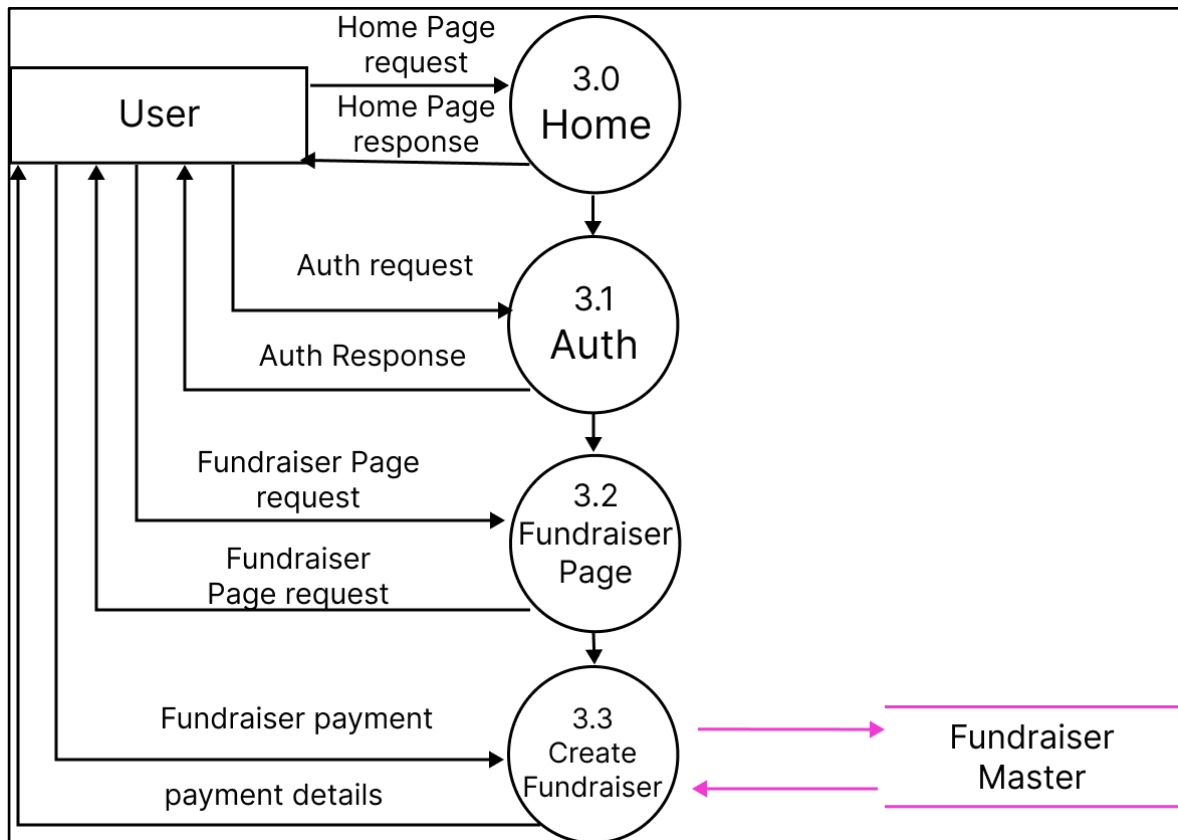
4.

   a. **Level 1 DFD : Guest**
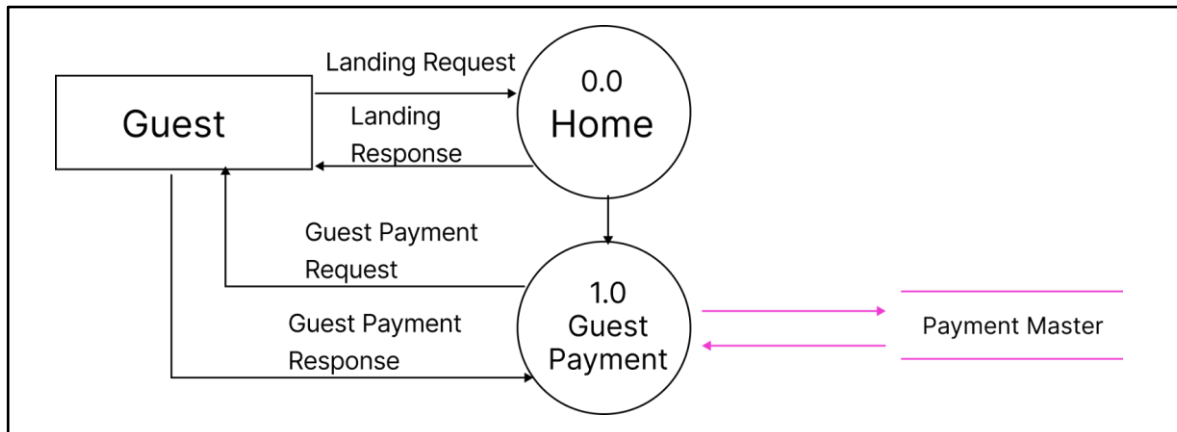
   b. **Level 2 DFD Guest Module for fundraising payment**
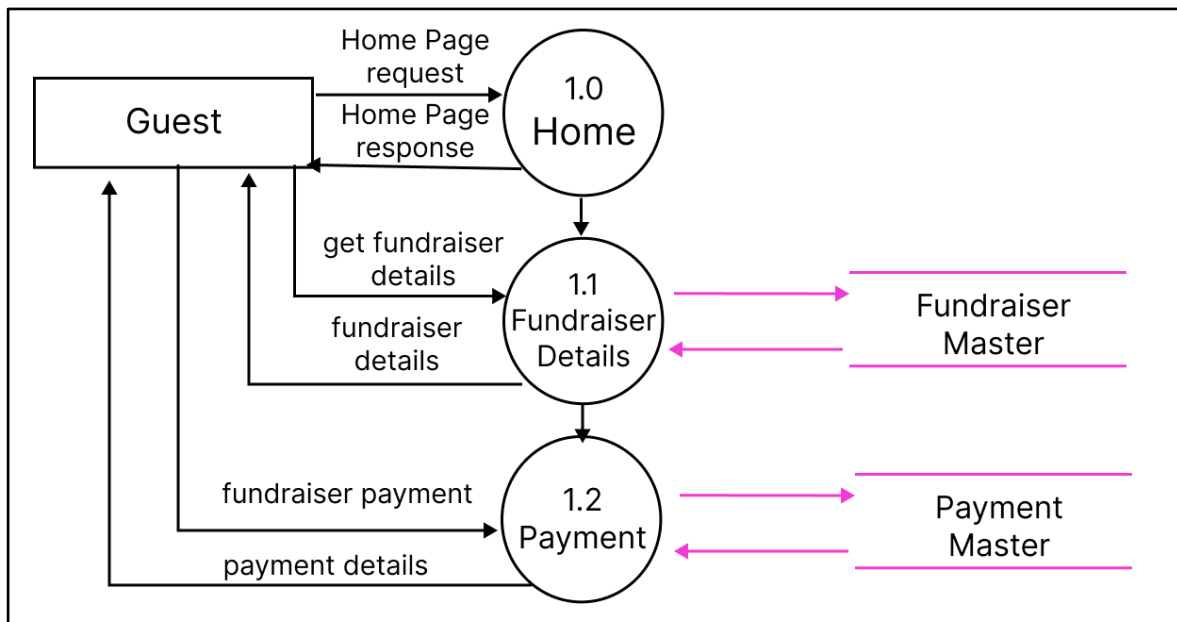
ER DIAGRAM
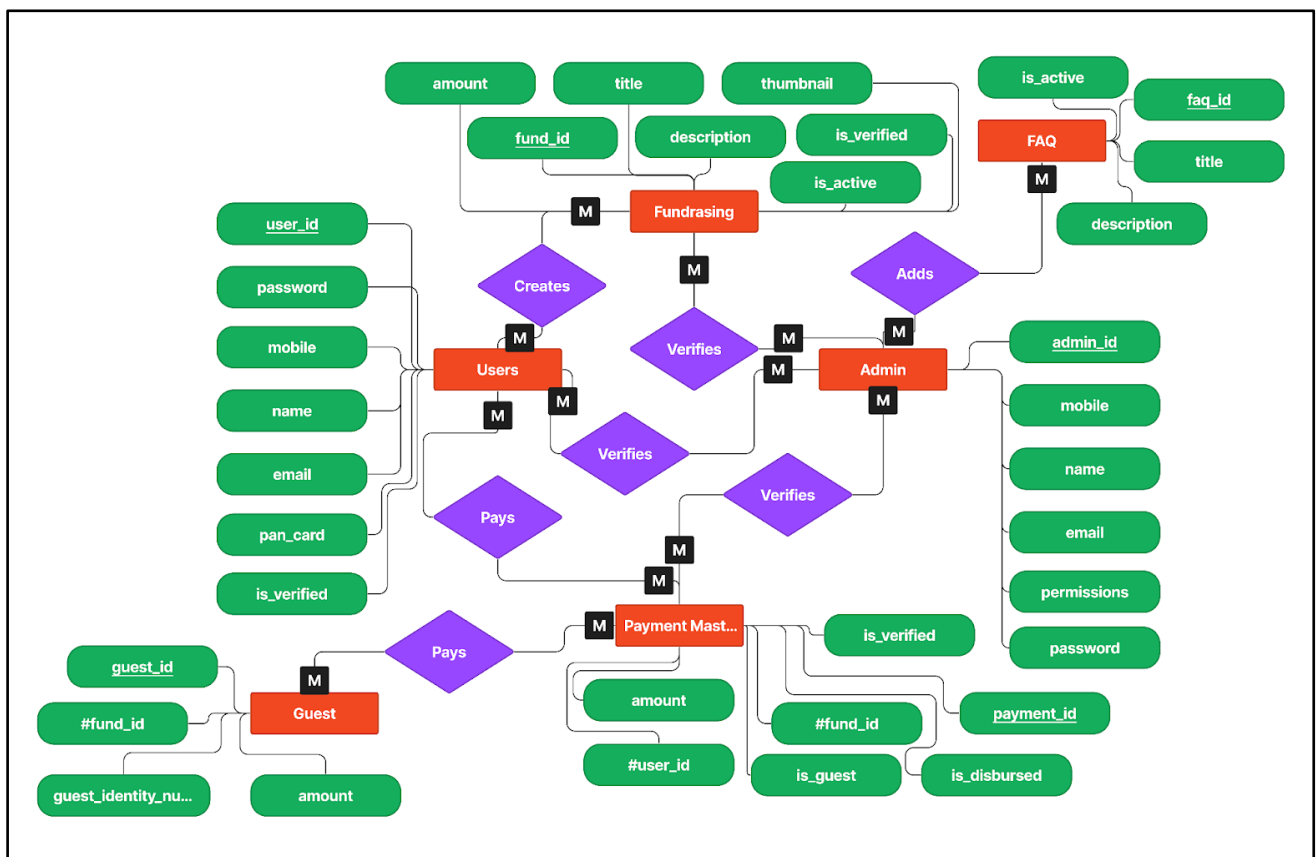
DATABASE DESIGN

1. Users Table

Field Name	Type	Constraints
user_id	int	Primary Key
name	text	-
email	text	Unique
mobile	Big int	Unique
pan_card	text	-
password	text	-
is_verified	bit	-

2. Admin Table
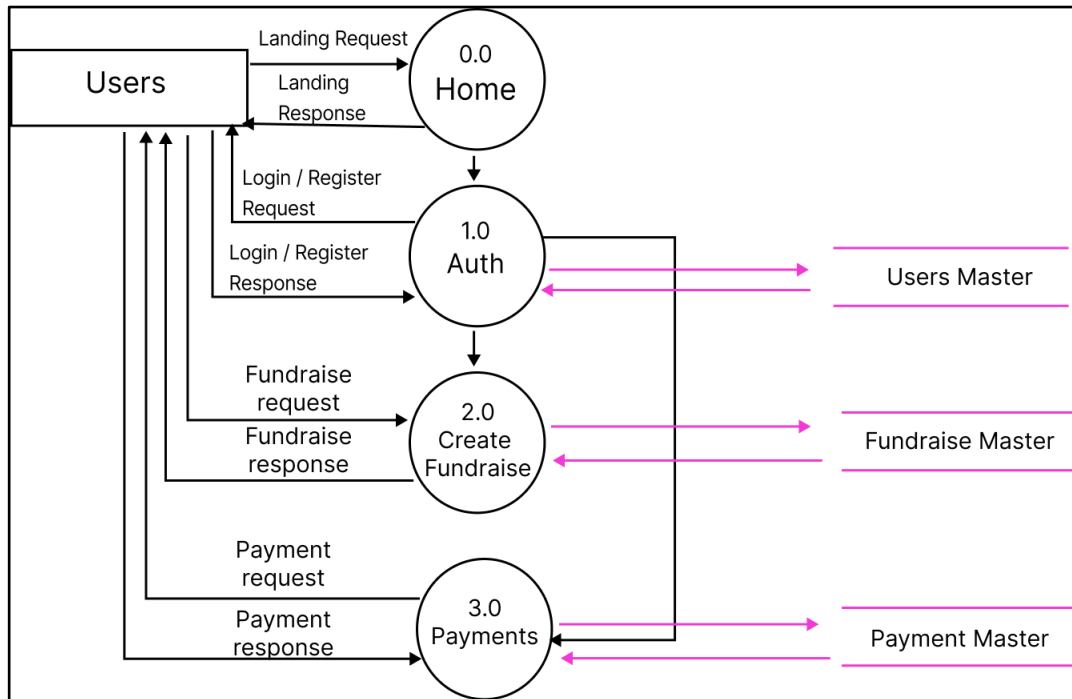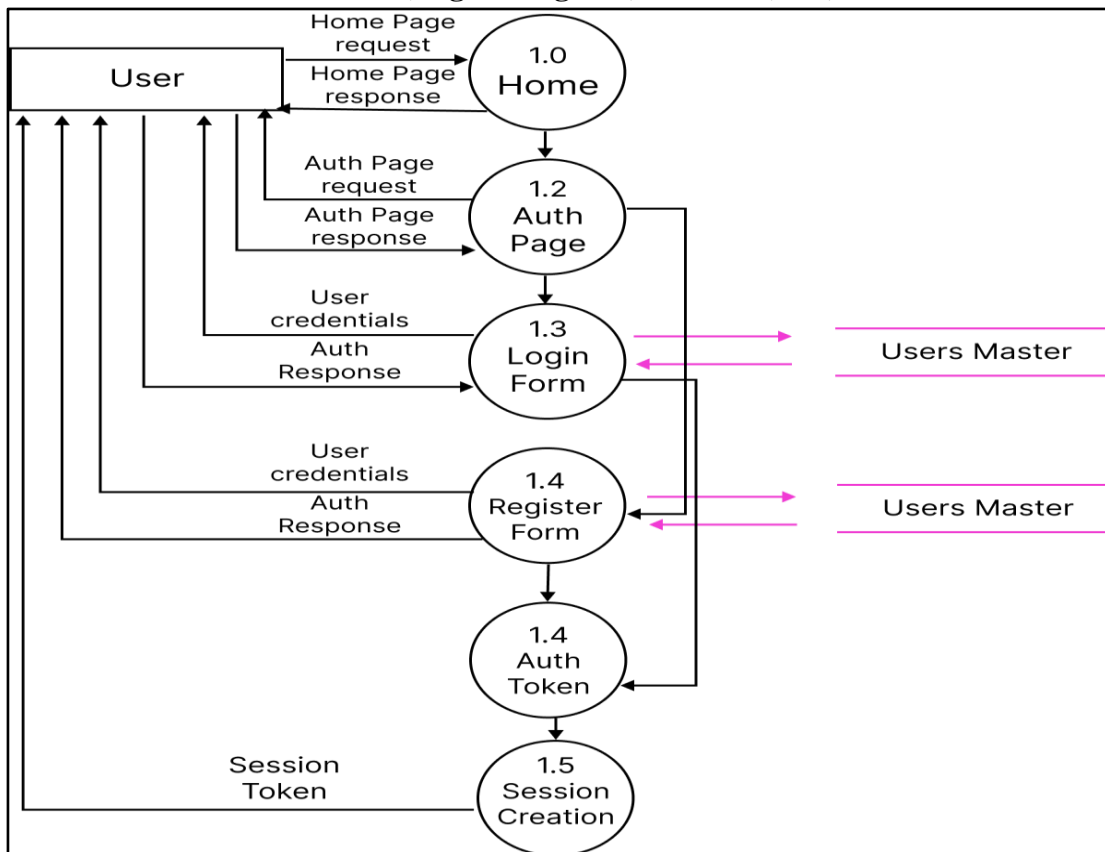
Field Name	Type	Constraints
admin_id	int	Primary Key
name	text	-
email	text	Unique
mobile_number	Big int	Unique
permissions	text	-
password	text	-

3. Payment Table

Field Name	Type	Constraints
payment_id	int	Primary Key
amount	text	-
user_id	int	foreign key
fund_id	int	foreign key
is_guest	bit	-
is_verified	bit	-

4. Fundraising Request

Field Name	Type	Constraints
fund_id	int	Primary Key
amount	text	-
user_id	text	Unique, foreign key
is_verified	bit	-
is_active	bit	-
title	text	-
description	text	-
is_guest	bit	-

5. Guest

Field Name	Type	Constraints
guest_id	int	Primary Key
amount	int	-
guest_reference_number	text	Unique
is_active	bit	

6. FAQ

Field Name	Type	Constraints
faq_id	int	Primary Key
title	text	-
description	text	Unique
is_active	bit	

TESTING

○ Objective

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet undiscovered error.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

● Unit Testing

The purpose of the coding and unit testing phase of software development is to translate the software design into source code.Each component of the design is implemented as a program module.The end-product of this phase is a set of program modules that have been individually tested.To enable the engineers to write good quality programs, every software development organization normally formulates its own coding standards that suits itself.

● Integration & System Testing

Integration of different modules is undertaken once they have been coded and uni tested.During the integration and syste3m testing phase, the modules are integrated in a planned manner.The different modules making up a software product are almost never integrated in one shot.Integration is normally carried out incrementally over a number of steps.DURING each integration step, the partially integrated system is tested and a set of previously planned modules are added to it.Finally, when all the modules have been successfully integrated and tested, system testing is carried out.

FUTURE SCOPE

Here are some future trends that can be opted in our product.

- <u>Internationalisation</u>: Expanding the platform to support fundraising for causes in other countries.
- <u>Mobile app development</u>: Building a mobile app to make the platform more accessible to donors and fundraisers on the go.
- <u>Crowdlending</u>: Adding a lending component to the platform, where donors can lend money to borrowers with a clear repayment plan, allowing for a more sustainable form of giving.
- <u>Impact tracking</u>: Adding features that allow users to track the impact of their donations in real-time.
- <u>Community building</u>: Building a community around the platform, where users can connect with each other and discuss causes that they care about.
- <u>Advanced analytics</u>: Developing advanced analytics features to provide users with insights on the impact of their donations and the performance of their campaigns.

BIBLIOGRAPHY

- **Books**
  - BCS-053 Web Programming
  - MCSL-016 Internet Concepts and Web Design
  - MCS-024 Object Oriented Technology and Java Programming

- **Websites**
  - https://egyankosh.ac.in
  - https://swayam.gov.in/

INDIRA GANDHI NATIONAL OPEN UNIVERSITY

PROJECT REPORT
ON

TAABIR: The Crowdfunding Website
By

Adhyatamjot Singh
Enrollment Number : 2105339859

Under Guidance of
Dr. Neeta

Submitted to the School of Computer and Information Sciences
In partial fulfilment of the requirements
For the degree of
Bachelor of Computer Applications

Indira Gandhi National Open University
Maidan Garhi

PROJECT TITLE

TAABIR
The Crowdfunding Website

INDEX

INTRODUCTION AND OBJECTIVE

INTRODUCTION

Taabir is a crowdfunding platform that enables non-profit organizations and individuals to raise funds for causes that matter to them. Our goal is to create a world where everyone can make a positive impact by giving back to society. Through Taabir, we empower individuals to support the causes they care about, and help non-profits reach their fundraising goals. Together, we can make a real difference in the world.

OBJECTIVE

The objective of Taabir is to connect people and organizations who share a passion for social good, and empower them to make a positive impact in the world. By offering a simple, secure, and user-friendly platform for crowdfunding, we aim to democratize philanthropy and provide a new avenue for individuals to give back to the causes they care about. At Taabir, we believe that every donation, no matter how small, can make a real difference. We strive to create a world where generosity and compassion are the norm, and where every person has the opportunity to make a meaningful impact.

PROJECT CATEGORY

This application is developed using three tier architecture of Java and J2EE Technologies, Figma is used for front end designing but JSP is used as tools and techniques and MySQL is used as a back end or database. The connection between the front-end and the back-end is established by using JDBC-ODBC Bridge.

Some style sheets and scripting technologies are used to enhance the dynamics of the Project.

Frontend Stack: HTML, CSS, JavaScript with TypeScript and Angular
Backend Stack: MySQL, Java J2EE, Java Servlets, JDBC, JSP.

HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements: -

- Intel Core i3-1315U with 8 threads and 10MB L3 cache.
- 4GB RAM.
- 80 GB Hard Disk Drive.
- 3.5" – 1.44 MB Diskette Drive.
- 52X CD-ROM Drive.
- Intel® Pro 10/100+ LAN Card..
- The computer should have a proper backup and recovery facility.

Software Requirements: -

- Node JS (Runtime for Angular)
- Angular CLI (Angular CLI tool)
- MySQL (Database Software)
- Java JDK (to run Java)
- Eclipse IDE (To code Java Servlets)
- VS Code IDE (To code Frontend, HTML, CSS and JavaScript with Angular)
- Browser (Chrome / Firefox :- To evaluate the Application)

SYSTEM ANALYSIS

"**System analysis**" is a problem-solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. According to the Merriam-Webster dictionary, systems analysis is "the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way". Analysis and synthesis, as scientific methods, always go hand in hand; they complement one another. Every synthesis is built upon the results of a preceding analysis, and every analysis requires a subsequent synthesis to verify and correct its results.

This field is closely related to requirements analysis or operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than she might otherwise have made.

The principal objective of the systems-analysis phase is the specification of what the system needs to do to meet the requirements of end users. In the systems-design phase such specifications are converted to a hierarchy of charts that define the data required and the processes to be carried out on the data so that they can be expressed as instructions of a System program. Many information systems are implemented with generic software, rather than with such custom-built.

REQUIREMENTS SPECIFICATIONS PROJECT PLANNING & SCHEDULING

SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements, and may include a set of use cases that describe interactions the users will have with the software.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

- ## Software Development Process Model

  The incremental model is a software development process that breaks down a project into smaller, more manageable chunks, or increments. Each increment builds upon the previous one, adding functionality and features in a step-by-step manner. The process involves multiple phases: Requirements gathering and analysis, design, implementation, testing, and evaluation.

1. Introduction

Purpose
The primary goal of this Software Requirements Specification (SRS) document is to provide a comprehensive outline of the Taabir crowdfunding platform's functional and non-functional requirements.

Scope
This SRS focuses on the development of Taabir, a web-based crowdfunding platform for medical and NGO projects, built using Angular, Java Servlets (J2EE), and MySQL database.

Definitions, Acronyms, and Abbreviations

- Taabir: Crowdfunding platform
- J2EE: Java 2 Enterprise Edition
- MySQL: Open-source relational database management system

● SRS: Software Requirements Specification

2. Overall Description

Product Perspective

Taabir is an independent crowdfunding platform that connects project creators with potential supporters for medical and NGO causes.

Product Functions

Taabir enables users to create and manage projects, discover and support campaigns, and facilitate secure financial transactions between parties.

User Characteristics

Taabir serves individuals and organizations seeking financial support for medical and NGO projects, as well as contributors looking to fund these initiatives.

Constraints

Taabir must comply with relevant crowdfunding regulations and integrate with third-party payment gateways for secure transactions.

Assumptions and Dependencies

Users are expected to have compatible devices and internet connectivity, as well as a basic understanding of crowdfunding concepts.

3. Specific Requirements

Functional Requirements

● User Registration: Users should be able to create and manage their accounts.
● Project Creation: Users should be able to create and publish crowdfunding projects.
● Project Browsing: Users should be able to browse and filter available projects.
● Financial Transactions: Users should be able to contribute funds to projects securely.
● Communication: Users should be able to communicate with project creators and other supporters.

Non-Functional Requirements

● Performance: Taabir should load within 3 seconds and support up to 1000 concurrent users.
● Security: User data and transactions should be secure and encrypted.
● Usability: Taabir should have a user-friendly interface and intuitive navigation.

MODULES & TIME ESTIMATION

Modules and Processes

4. <u>Actor: Admin</u>

    a. **Module 1: Admin Login**
    ● **Input-** Email, Password
    ● **Process -** Login form submit
    ● **Output**: Success / Failure

    b. **Module 2: Verify Fundraising Requests**
    ● **Input-** Verified or Rejected
    ● **Process -** Verify option
    ● **Output**: Success / Failure

    c. **Module 3: Verify / Manage Users**
    ● **Input-** Verified or Rejected
    ● **Process -** Verify option
    ● **Output**: Success / Failure

    d. **Module 4: Verify Payment Source**
    ● **Input-** Verified or Rejected
    ● **Process -** Verify option
    ● **Output**: Success / Failure

    e. **Module 5: Create FAQs**
    ● **Input-** FAQ title, FAQ Description, FAQ active
    ● **Process -** FAQ form submit
    ● **Output**: Success / Failure

5. <u>Actor: User</u>

    a. **Module 1: Registration**
    ● **Input-** Name, email, mobile number
    ● **Process -** Registration Form
    ● **Output**: Success / Failure

    b. **Module 2: Login**
    ● **Input-** Email, Password

- **Process -** Login Form
- **Output**: Success / Failure

c. **Module 3: Create Fundraising Requests**
- **Input-** Amount, reason,
- **Process -** Fundraising request Form
- **Output**: Success / Failure

d. **Module 4: Fundraise contribution**
- **Input-** Amount, Misc info
- **Process -** Fund Raising Contribute form
- **Output**: Success / Failure

6. <u>Actor: Guest</u>

a. Module 1: Guest Fundraise contribution
- **Input-** Amount to contribute, name, PAN Card
- **Process -** Contribute to Funds Form
- **Output**: Success / Failure

Time Estimation: Gantt Chart
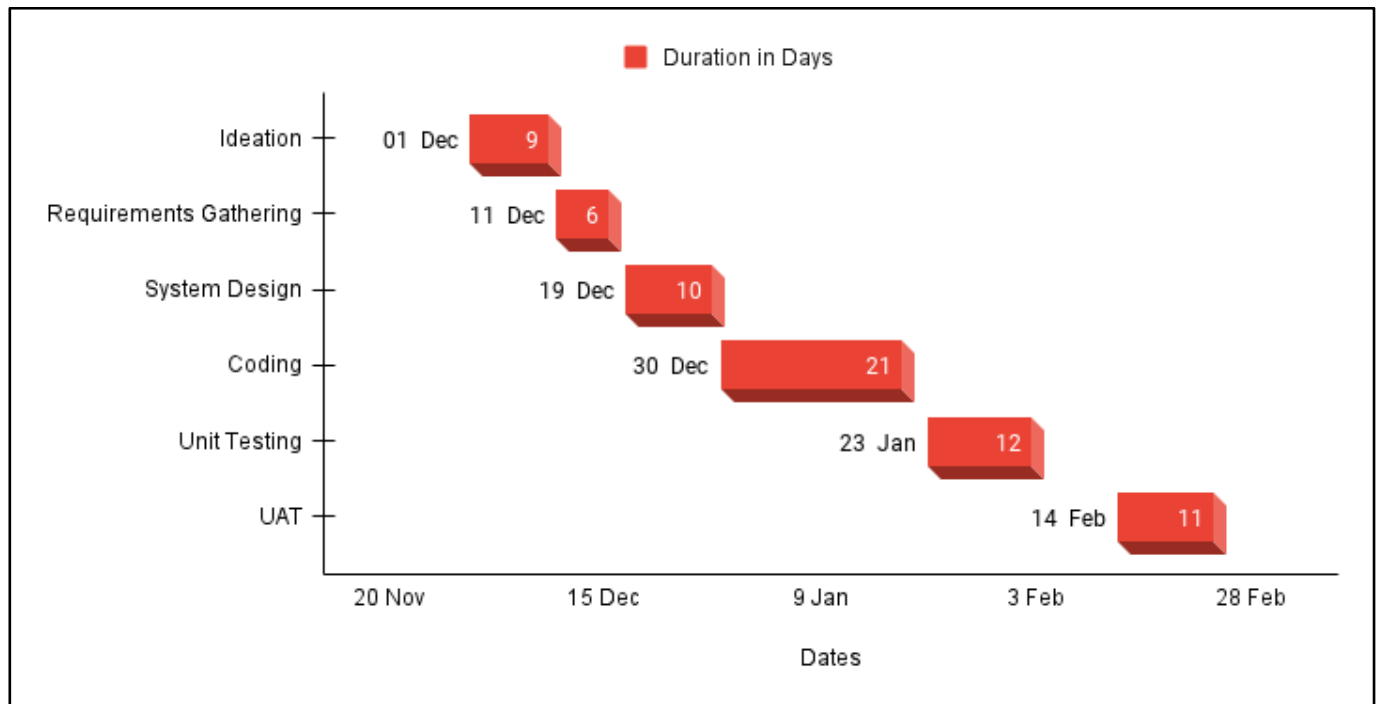
- Gantt Chart

Gantt Chart

DATA FLOW DIAGRAM

Data flow diagram is used to depict exactly the process of data transformation. It visualises the process of data transformation, relations and data processing among various inputs and outputs and internal/external entities of the system.

Elements of the DFD:-

External Entity   Process   Data Store   Data Flow
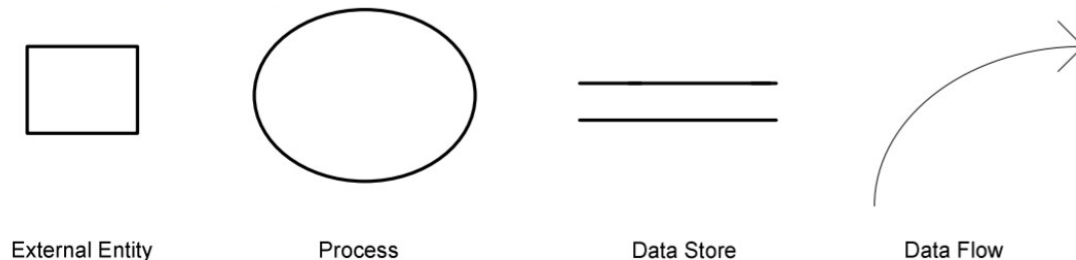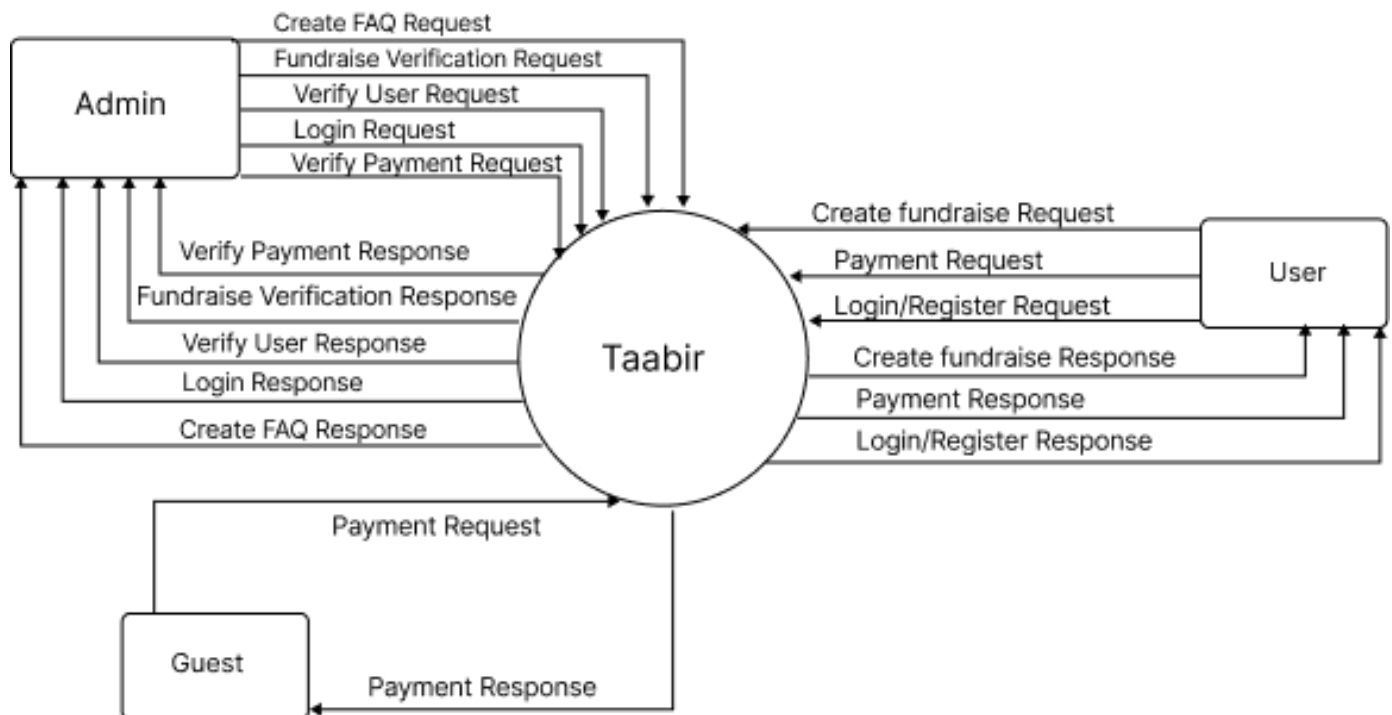
1. Level 0 DFD

Level 0 DFD : Admin , User and Guest

2. Admin DFDs
Level 1 DFD
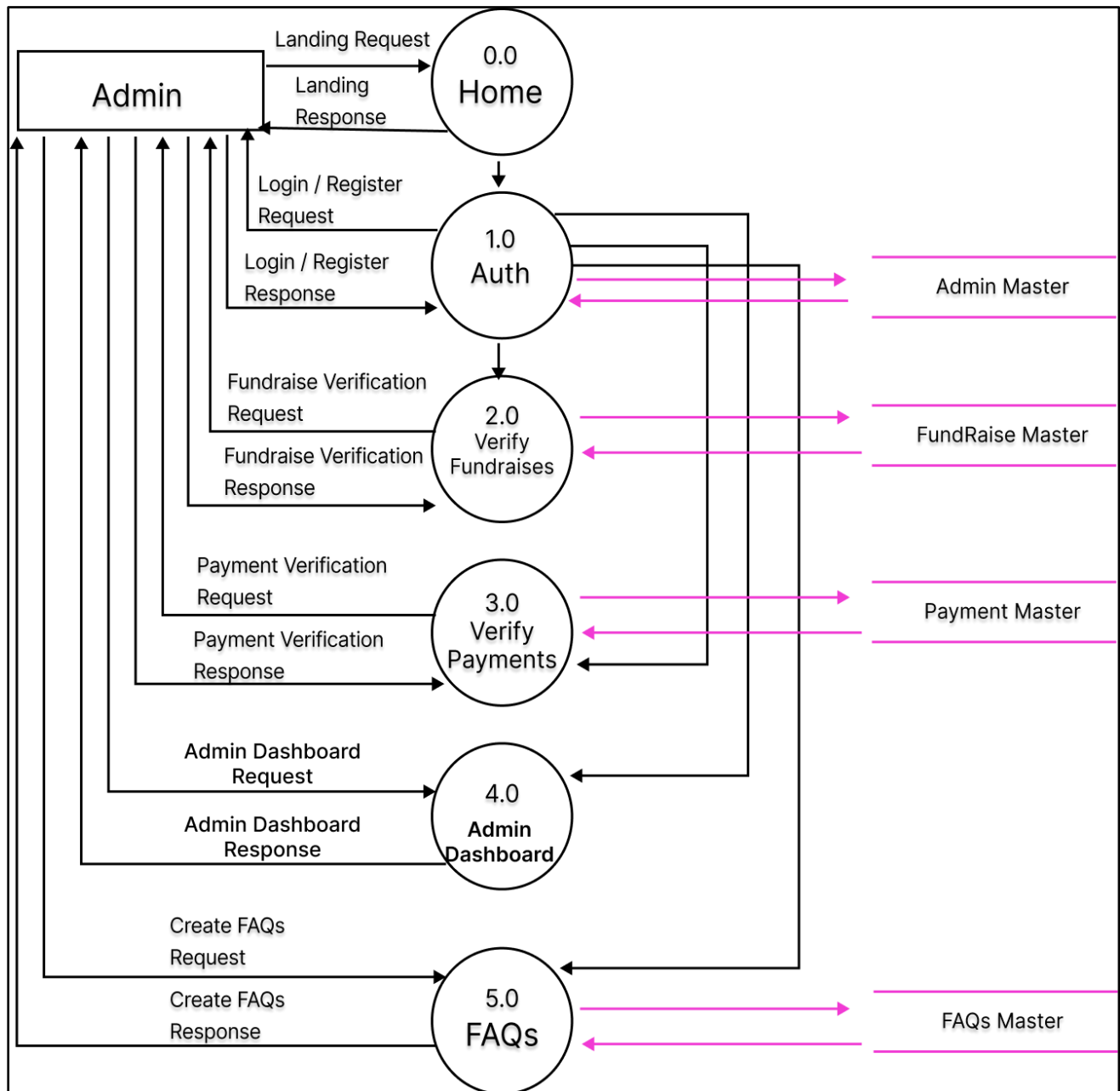
Level 1 DFD : Admin

Level 2 DFD

Level 2 DFD | Admin auth Module(1.0)
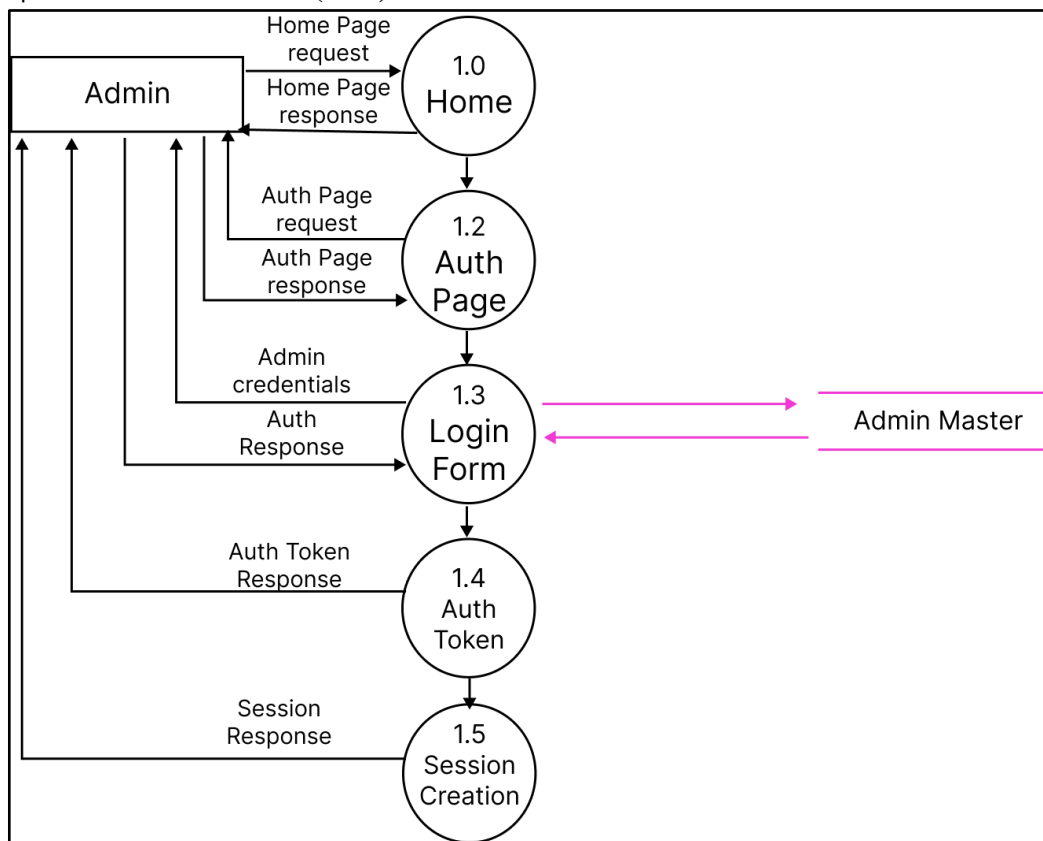
Level 2 DFD : Admin Login and Session Creation

Level 2 DFD | Verification of Fundraises (2.0)
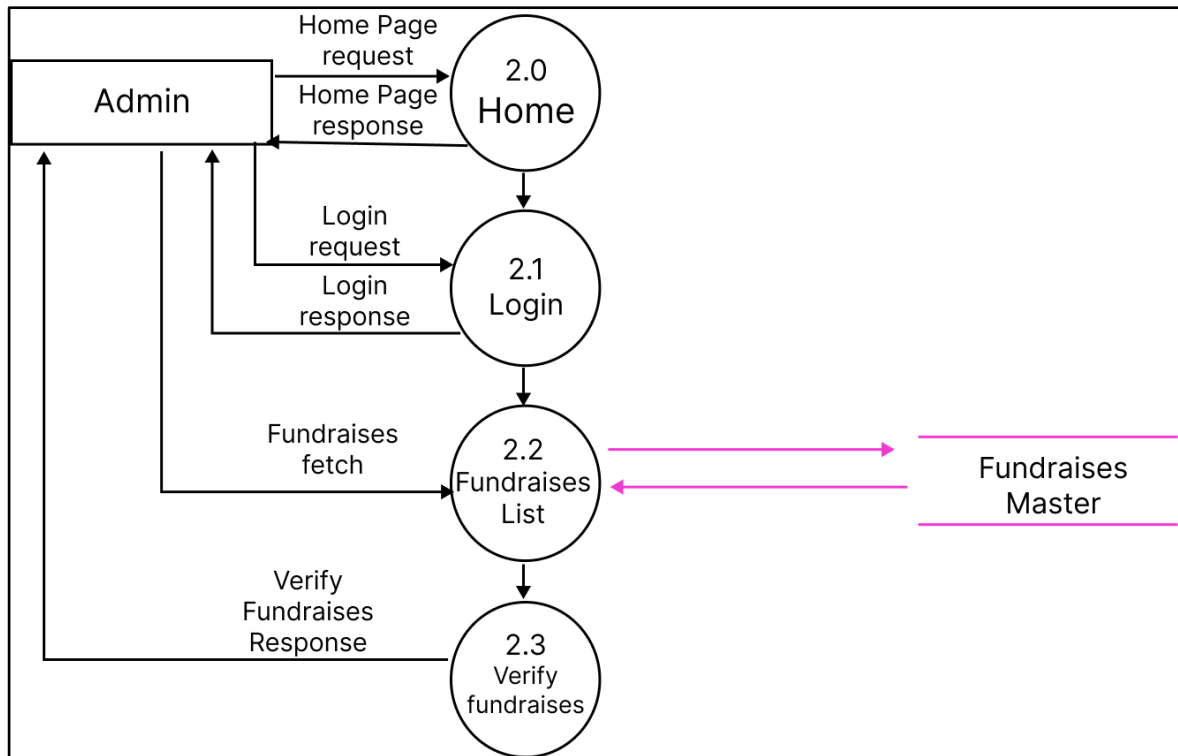
Level 2 DFD : Fundraises verification By Admin

Level 2 DFD | Verification of Users Module(3.0)
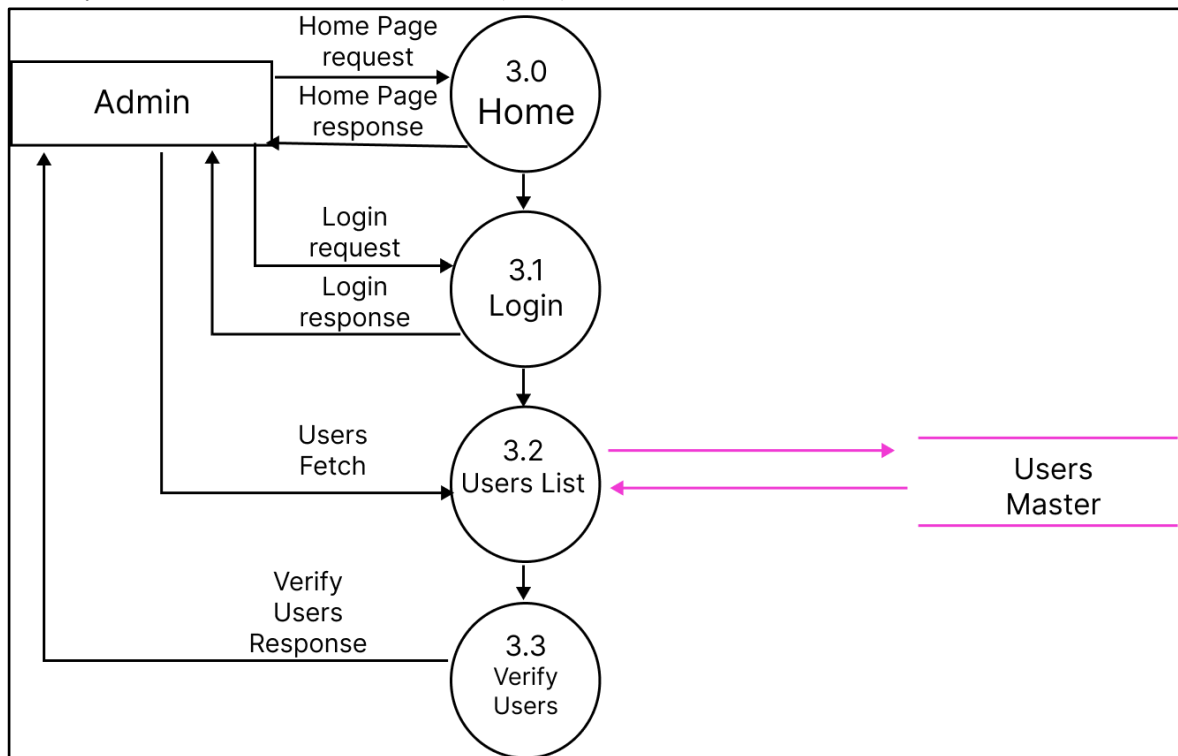
Level 2 DFD : Verify Users by Admin

Level 2 DFD | Verification of Payments Module(4.0)
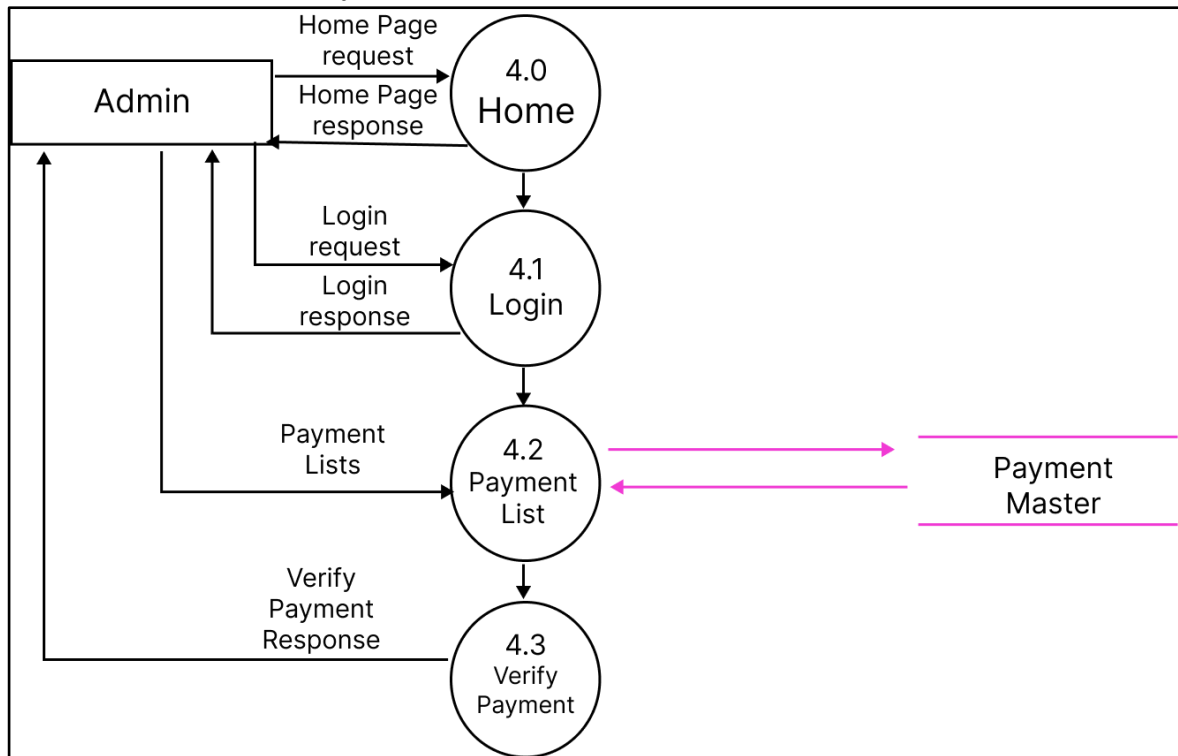
Level 2 DFD : Admin Payment Verification

Level 2 DFD | Management of FAQs (5.0)
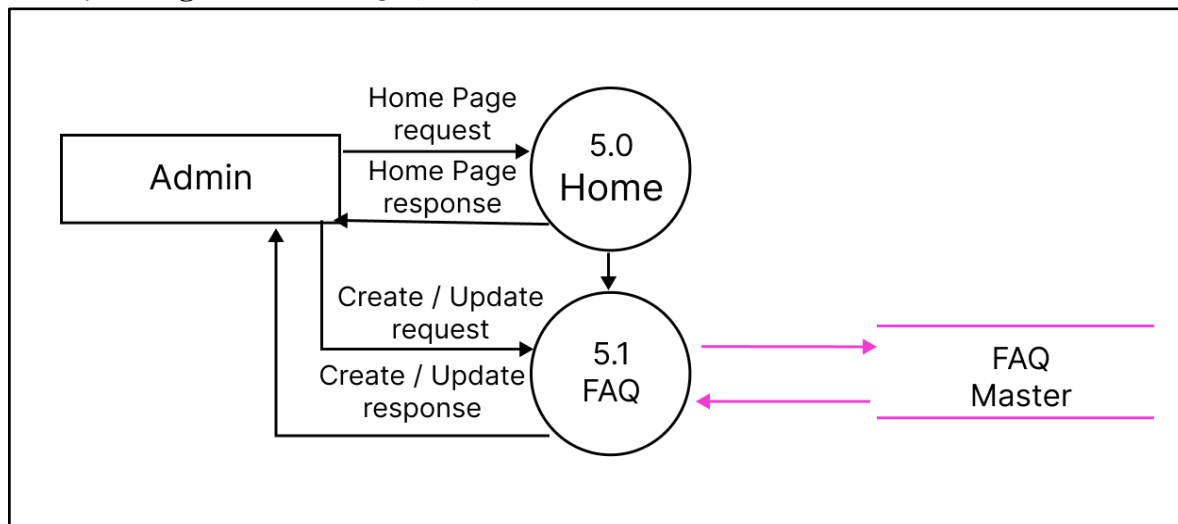
Level 2 DFD : Admin FAQs management for frontend display

2. Users DFDs

Level 1 DFD
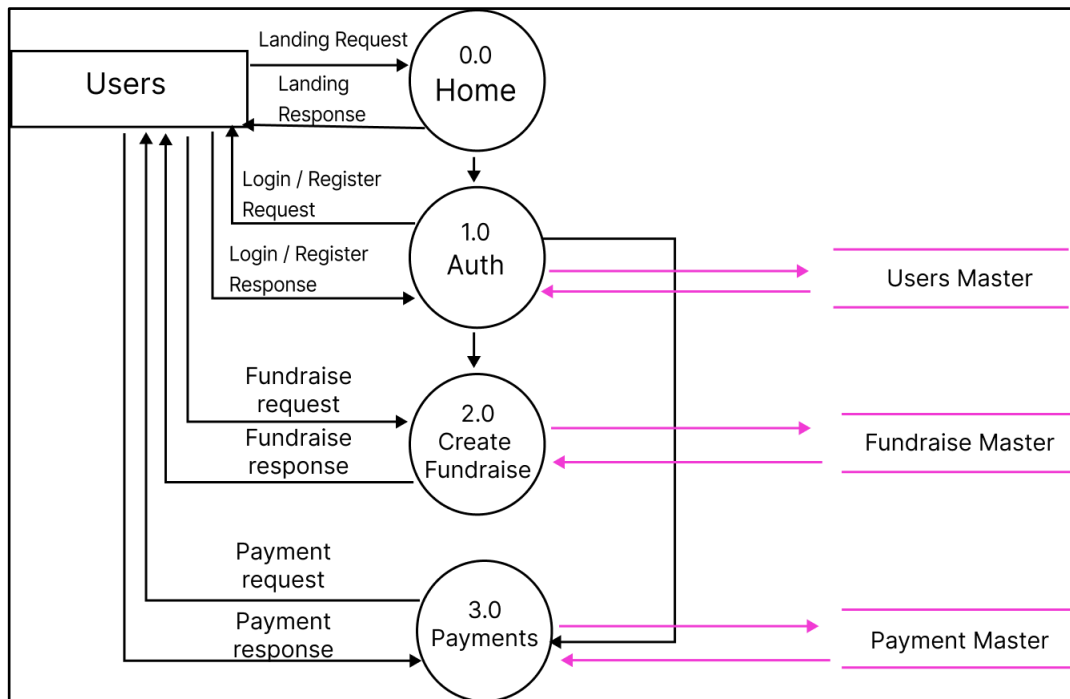
Level 1 DFD : User Module

Level 2 DFD: User Auth (Login / Register) Module (1.0)
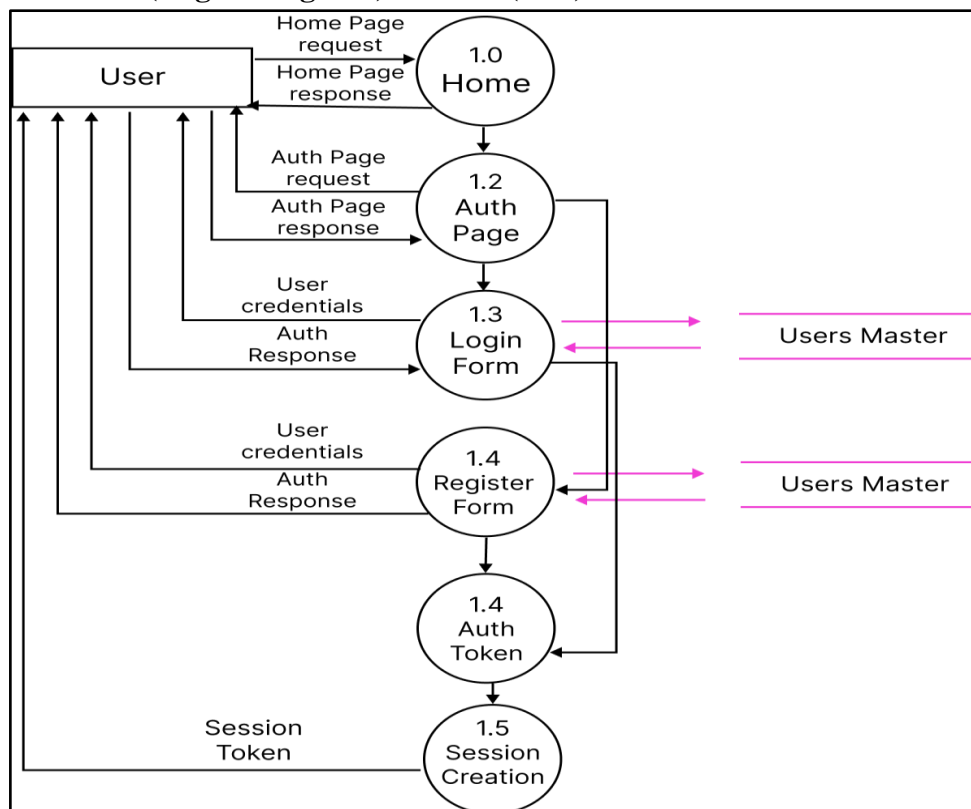
Level 2 DFD : User Register / Login

Level 2 DFD: User Module Fundraiser payment
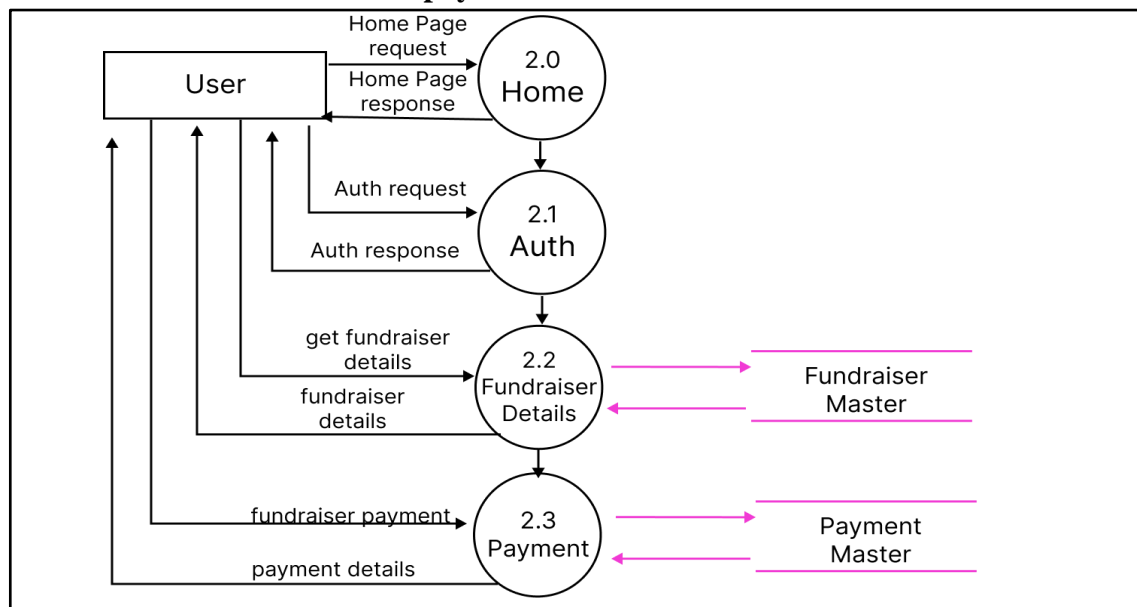
Level 2 DFD : User Fundraiser contribution

Level 2 DFD: User Fundraiser Creation
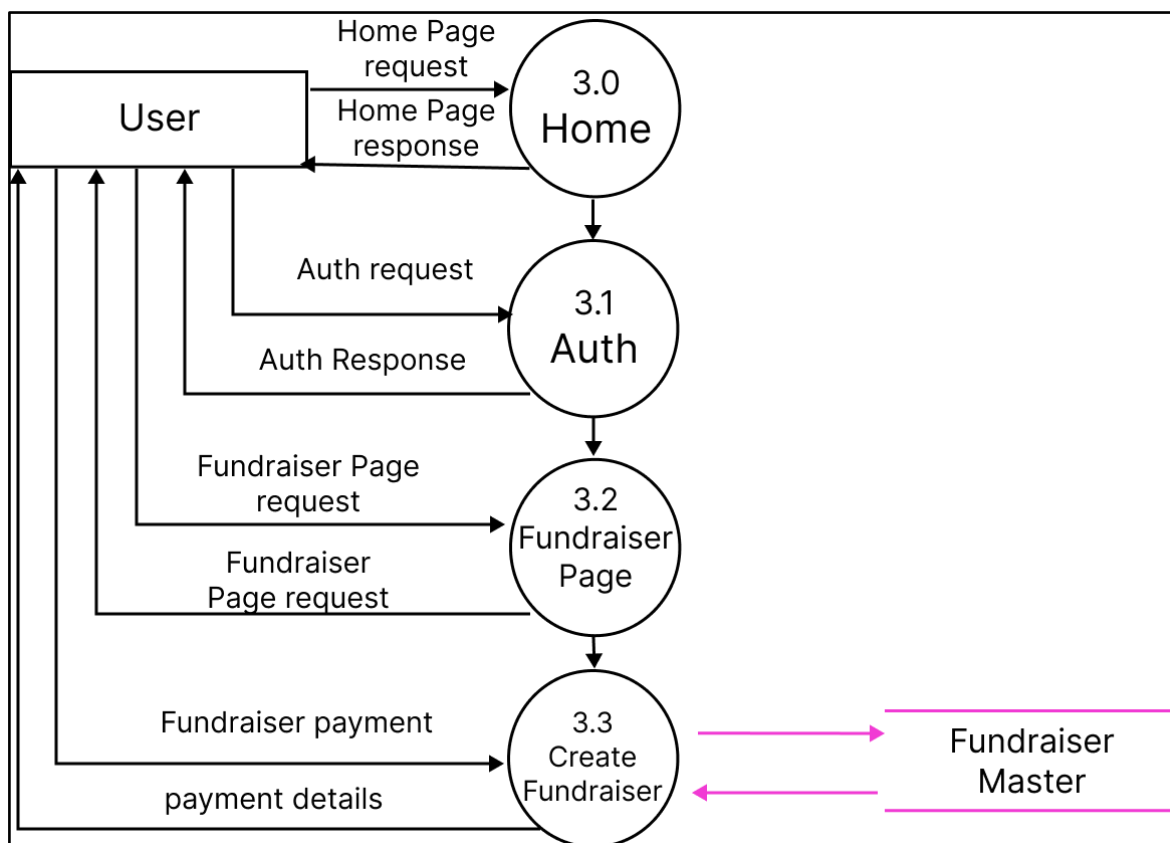
Level 2 DFD : User Fundraiser creation

3. Guest DFDs

Level 1 DFD : Guest
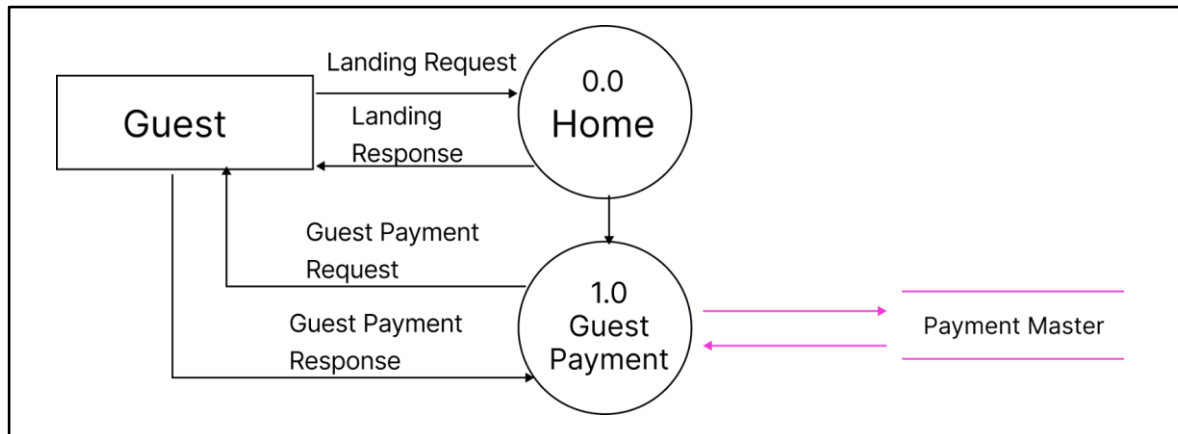
Level 1 DFD : Guest Module

Level 2 DFD Guest Module for fundraising payment
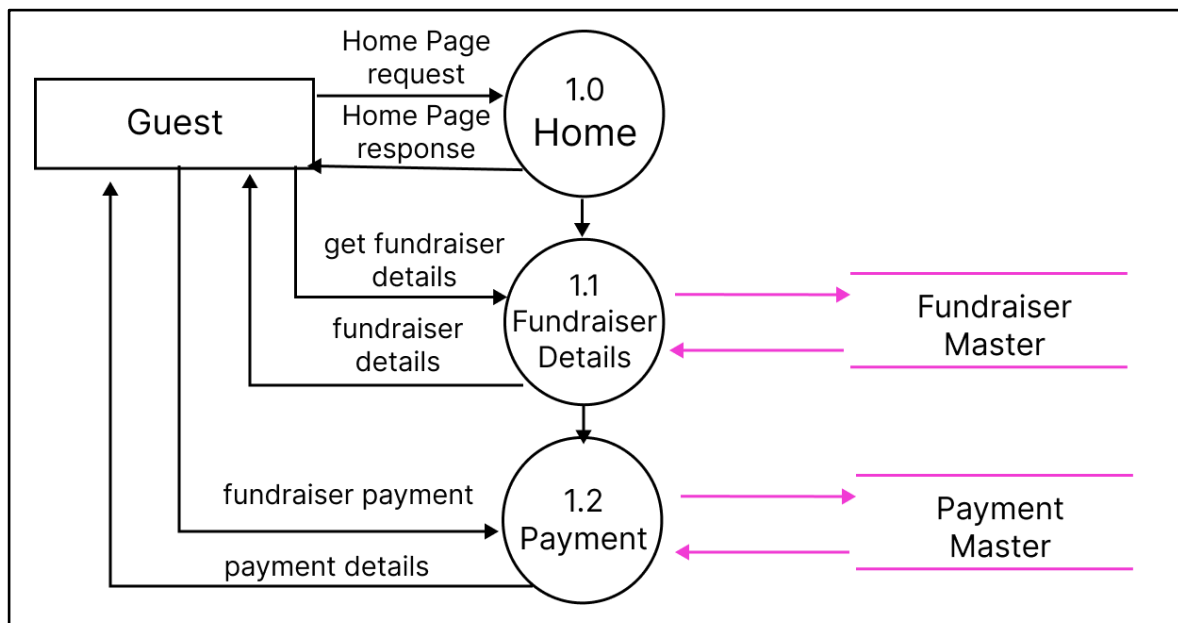
Level 2 DFD : Guest / Anonymous Payment

USE CASE DIAGRAM (UML)

A Use Case Diagram is a vital tool in system design, it provides a visual representation of how users interact with a system. It serves as a blueprint for understanding the functional requirements of a system from a user's perspective, aiding in the communication between stakeholders and guiding the development process.
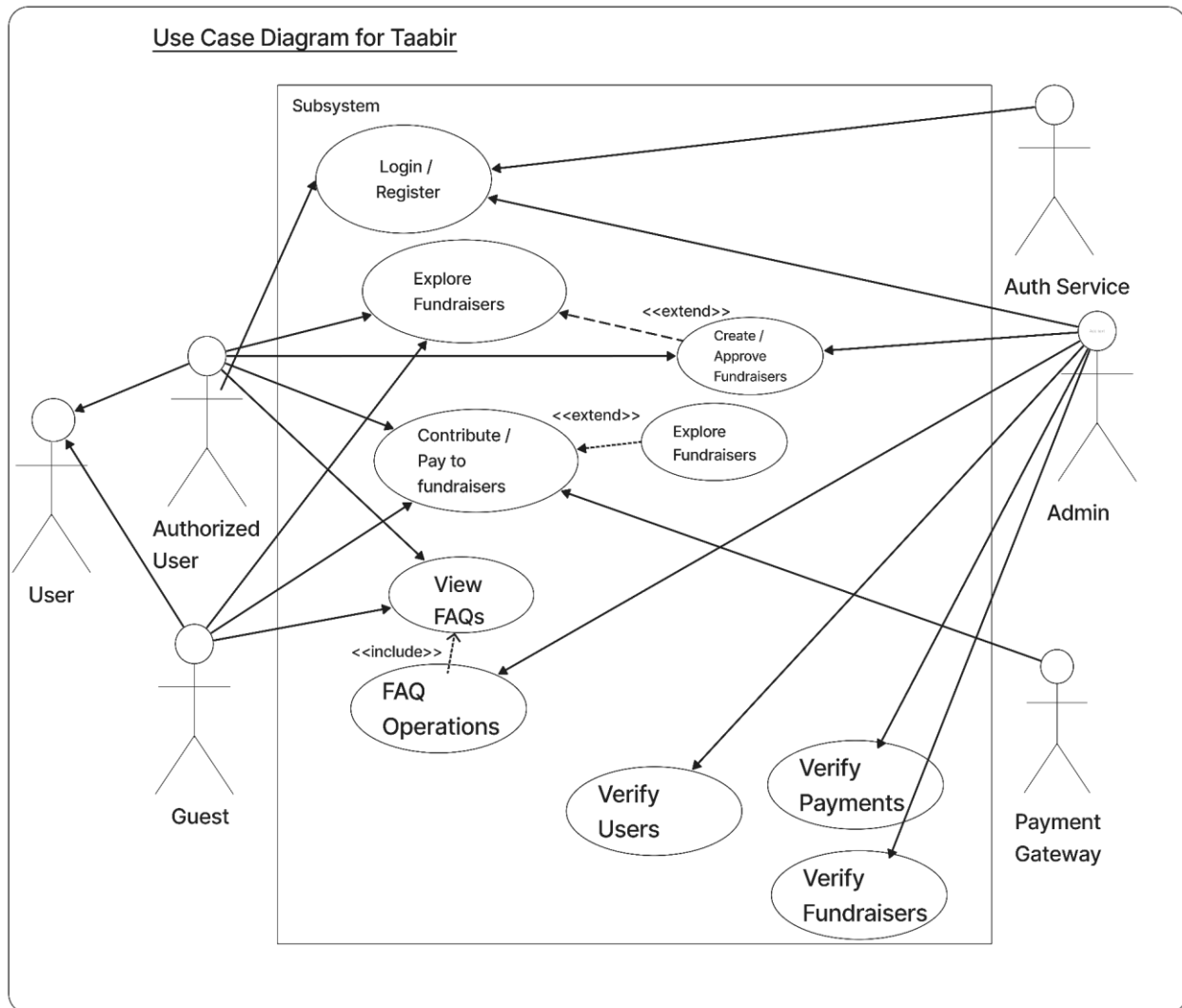
Note: All the cases here depict the functionality of the project

Actors:

- Admin = Related to Admin Module
- User = Authorised and Guest, wil do tasks like creating the fundraisers and contributing to them
- Payment Gateway = will take payment and provide payment details
- Auth Service = Will check for the user33333 if it is logged in or not.

ER DIAGRAM

An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualise business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterise them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of the relationships.

PROPERTIES OF E-R DIAGRAM:-

1) ENTITY :-
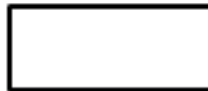
2) RELATIONSHIP :-
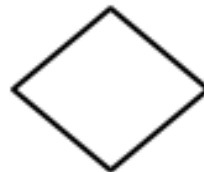
3) ATTRIBUTE :-

ER DIAGRAM (Taabir)
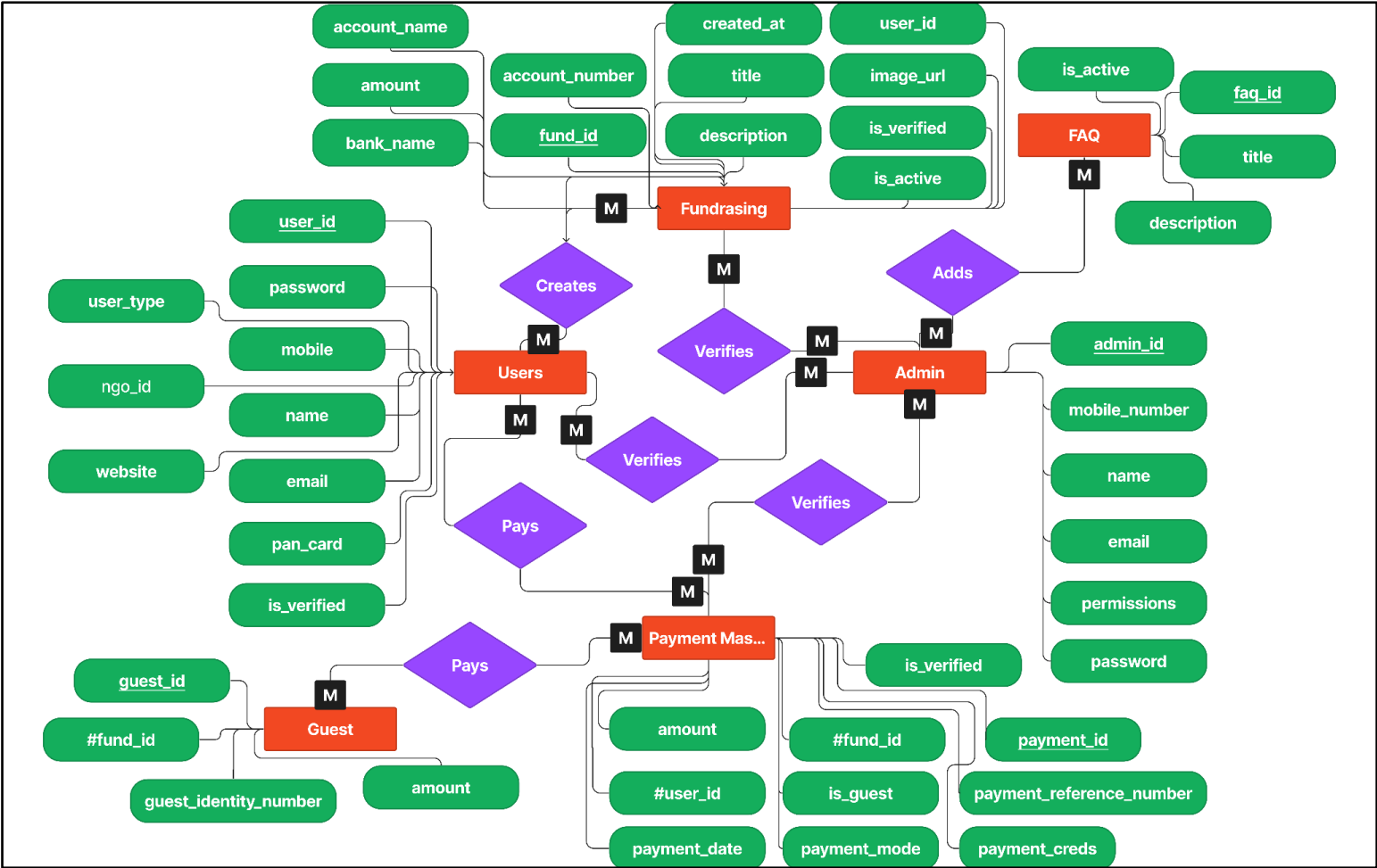
DATABASE DESIGN

1. **users** (Users Table)

Field Name	Type	Constraints	Description
user_id	int	Primary Key	Unique ID for users table
name	text	-	Name of the user
email	text	Unique	Email of the user
mobile	Big int	Unique	Mobile of the user
pan_card	text	-	PAN Card of the user
password	text	-	Password for user account
is_verified	bit	Default (0)	Whether user is verified or not
user_type	int	-	User Type (1= Individual, 2 = NGO)
ngo_id	varchar (50)	Default(NULL)	NGO Darpan Id (If user_type is NGO)
website	varchar(200)	Default(NULL)	NGO Website(If user_type is NGO)

2. **admin** (Admin Table)

Field Name	Type	Constraints	Description
admin_id	int	Primary Key	Unique ID for admin table
name	text	-	Admin name
email	text	-	Admin email
mobile_number	Big int	-	Admin mobile
permissions	text	-	Admin permissions
password	text	-	Admin Password

3. **payments**(Payment Table)

Field Name	Type	Constraints	Description
payment_id	int	Primary Key	Unique Id for payment table
amount	text	-	Payment amount
user_id	int	foreign key	Payment user id
fund_id	int	foreign key	Payment fundraiser id
is_guest	bit	-	Whether it a guest payment or not
is_verified	bit	Default (0)	Is Payment verified
payment_reference_number	varchar(255)	-	Payment reference id for each payment
payment_mode	int	-	1 = UPI, 2 = Card, 3 = Wallet
payment_creds	text	-	Encoded Payment Details like UPI ID,
payment_date	datetime	Default (Current timestamp)	Payment done date

4. **fundraises** (Fundraising Request)

Field Name	Type	Constraints	Description
fund_id	int	Primary Key	Unique ID for fundraiser
amount	foat	-	Amount of fundraiser
user_id	int	Unique, foreign key	User Id of Fundraiser creator
is_verified	bit	Default (0)	Verified by Admin
is_active	bit	Default (0)	If it is completed / active
title	text	-	Main Title of fundraiser
description	text	-	Description of fundraiser
image_url	text	-	Main Thumbnail of fundraiser
account_name	varchar(255)	-	Account Name of creator
account_number	varchar(20)	-	Account Number of creator
bank_name	varchar(200)	-	Bank Name of creator
created_at	date	Default (current_timestamp)	Date on which Fundraiser is created

5. **guests** (Guests)

Field Name	Type	Constraints	Description
guest_id	int	Primary Key	Guest Unique ID
amount	int	-	Amount guest paid
guest_reference_number	text	Unique	Payment reference number
fund_id	int	Foreign key	Fundraiser ID that is contributed by guest
is_active	bit	Default (0)	Whether guest payment is active

6. FAQ

Field Name	Type	Constraints	Description
faq_id	int	Primary Key	Unique Id for every FAQ
title	text	-	Title of FAQ
description	text	Unique	FAQ description
is_active	bit	Default (0)	Is Active to be visible to user

DATA INTEGRITY AND CONSTRAINTS

Database Code

-- Database: `taabir`

-- Create Database taabir

CREATE DATABASE IF NOT EXISTS taabir;

-- Use Taabir as main database

USE taabir;

--

-- Table structure for table `admin`

--

```sql
CREATE TABLE `admin` (
 `admin_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 `name` text NOT NULL,
 `email` text NOT NULL,
 `mobile_number` bigint(20) NOT NULL,
 `permissions` text NOT NULL,
 `password` text NOT NULL
);
```

-- Table structure for table `faq`

```
CREATE TABLE `faq` (
 `faq_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 `title` text NOT NULL,
 `description` text NOT NULL,
 `is_active` tinyint(4) NOT NULL DEFAULT 0
);
```

--

-- Table structure for table `fundraises`

--

```
CREATE TABLE `fundraises` (
 `fund_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 `amount` float NOT NULL,
 `user_id` int(11) NOT NULL,
 `is_verified` tinyint(4) NOT NULL DEFAULT 0,
 `is_active` tinyint(4) NOT NULL DEFAULT 0,
 `title` text NOT NULL,
 `description` text NOT NULL,
 `image_url` text NOT NULL,
 `account_name` varchar(255) DEFAULT NULL,
 `account_number` varchar(20) DEFAULT NULL,
 `bank_name` varchar(200) DEFAULT NULL,
 `created_at` date NOT NULL DEFAULT current_timestamp());
```

--

-- Table structure for table `guests`

--

```sql
CREATE TABLE `guests` (
 `guest_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 `amount` float NOT NULL,
 `guest_reference_number` varchar(255) NOT NULL,
 `fund_id` int(11) NOT NULL,
 `is_active` tinyint(4) NOT NULL DEFAULT 0
);
```

--

-- Table structure for table `payments`

--

```sql
CREATE TABLE `payments` (
 `payment_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 `amount` float NOT NULL,
 `user_id` int(11) NOT NULL,
 `fund_id` int(11) NOT NULL,
 `is_guest` tinyint(4) NOT NULL,
 `is_verified` tinyint(4) NOT NULL DEFAULT 0,
 `payment_reference_number` varchar(255) NOT NULL,
 `payment_mode` int(11) NOT NULL,
```

```sql
  `payment_creds` text NOT NULL,

  `payment_date` datetime NOT NULL DEFAULT current_timestamp()

);
```

```sql
--

-- Table structure for table `users`

--

CREATE TABLE `users` (

  `user_id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,

  `name` text NOT NULL,

  `email` text NOT NULL,

  `mobile` bigint(20) NOT NULL,

  `pan_card` text NOT NULL,

  `password` text NOT NULL,

  `is_verified` tinyint(4) NOT NULL DEFAULT 0,

  `user_type` int,
  `ngo_id` varchar(50),
  `website` varchar(200);

);
```

CODING

→ In Angular instead of javascript, HTML, CSS in one file, Angular uses a modular approach, with separate files for components, services, modules, and stylesheets. It also utilises TypeScript, which provides stricter type checking and OOP features. Coding in Angular involves using declarative templates (HTML+Angular expressions) and components with TypeScript files. Vanilla JS, HTML, and CSS, on the other hand, use plain HTML, CSS, and JavaScript without any special syntax. The structure is more basic and often involves loading multiple scripts in a single HTML file.

→ Using **Angular 17** and **Tailwind CSS 3.0**

Frontend Folder Structure

A folder structure is a directory file tree that is used to show where each file of a folder is located.
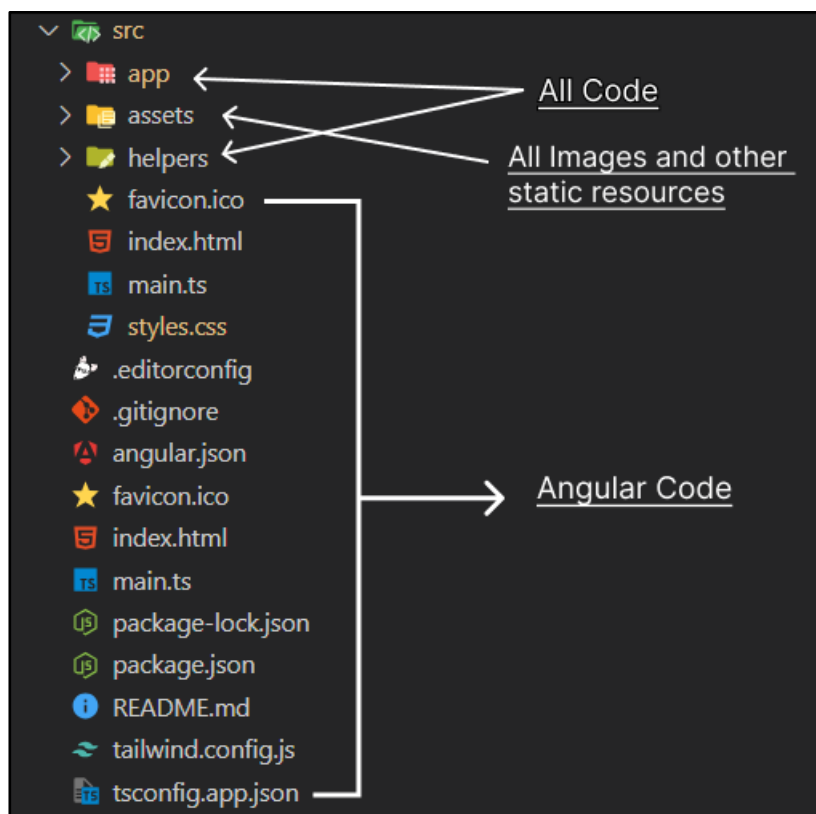
Folder structure for all HTML, CSS and Angular Files

Frontend Code

1. Main Files

● **Main index.html (src\index.html)**

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Taabir | The Crowd Funding App</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

● **Main TypeScript File (src\main.ts)**

```typescript
import { bootstrapApplication } from '@angular/platform-browser';
import { appConfig } from './app/app.config';
import { AppComponent } from './app/app.component';

bootstrapApplication(AppComponent, appConfig)
  .catch((err) => console.error(err));
```

● **Main Style CSS File (src\style.css)**

```css
/* You can add global styles to this file, and also import other style files */

@tailwind base;
@tailwind components;
@tailwind utilities;

.glass-bg{
  @apply bg-gray-500 rounded-md bg-clip-padding backdrop-filter backdrop-blur-sm bg-opacity-20
    shadow text-white;
}

.button-group {
```

```
      @apply flex space-x-2 justify-center gap-3;
   }

   [type=checkbox]{
      @apply rounded outline-none ring-0 shadow-none;
   }

   :disabled {
      @apply opacity-50 cursor-not-allowed;
   }
```

2. Home Page

- **cta.component.html (src\app\pages\home\cta\cta.component.html)**

```
<!-- Part of Home Page  -->
<section class="text-gray-950 body-font">
  <div class="container mx-auto flex px-5 py-24 md:flex-row flex-col items-center">
    <div class="lg:flex-grow md:w-1/2 lg:pr-24 md:pr-16 flex flex-col md:items-start md:text-left mb-16
      md:mb-0 items-center text-center">
      <h1 class="title-font sm:text-4xl text-3xl mb-4 font-medium text-gray-900 uppercase">Step out as
      the one
        <br class="hidden lg:inline-block"><span class=" font-bold ">to the needy</span>
      </h1>
      <p class="mb-8 leading-relaxed text-lg my-6">Our team is committed to building a safe and secure
      platform that facilitates transparent, impactful giving. We are dedicated to supporting our users and
      ensuring that they have the tools they need to succeed in their fundraising efforts.
        .</p>
      <div class="flex justify-center">
        <a routerLink="explore" class="inline-flex text-white bg-rose-500 border-0 py-2 px-6
      focus:outline-none hover:bg-rose-600 rounded text-lg">Explore Now</a>
        <a routerLink="contact-us" class="ml-4 inline-flex text-gray-700 bg-gray-100 border-0 py-2 px-6
      focus:outline-none hover:bg-gray-200 rounded text-lg">Contact Us</a>
      </div>
    </div>
    <div class="lg:max-w-lg lg:w-full md:w-1/2 w-5/6">
      <img class="object-cover object-center rounded" alt="hero" src="assets/img/home/cta.svg">
    </div>
  </div>
</section>
```

- **cta.component.ts (src\app\pages\home\cta\cta.component.ts)**

```typescript
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';
@Component({
 selector: 'app-cta',
 standalone: true,
 imports: [
  RouterModule ],
 templateUrl: './cta.component.html',
 styles: ''
})
export class CtaComponent {}
```

- **feature.component.html (src\app\pages\home\feature\feature.component.html)**

```html
<section class="text-gray-950 body-font"><!-- part of home page  -->
 <div class="container px-5 py-24 mx-auto">
  <div class="flex items-center lg:w-3/5 mx-auto border-b pb-10 mb-10 border-gray-200 sm:flex-row
   flex-col">
  <div
   class="sm:w-32 sm:h-32 h-20 w-20 sm:mr-10 inline-flex items-center justify-center rounded-full
  bg-rose-100 text-rose-500 flex-shrink-0 p-4">
   <svg data-slot="icon" fill="none" stroke-width="1.5" stroke="currentColor" viewBox="0 0 24 24"
  xmlns="http://www.w3.org/2000/svg" aria-hidden="true">
    <path stroke-linecap="round" stroke-linejoin="round" d="M11.48 3.499a.562.562 0 0 1 1.04
  0l2.125 5.111a.563.563 0 0 0 .475.345l5.518.442c.499.04.701.663.321.988l-4.204 3.602a.563.563
  0 0 0-.182.557l1.285 5.385a.562.562 0 0 1-.84.61l-4.725-2.885a.562.562 0 0 0-.586 0L6.982
  20.54a.562.562 0 0 1-.84-.611l1.285-5.386a.562.562 0 0 0-.182-.557l-4.204-3.602a.562.562 0 0 1
  .321-.988l5.518-.442a.563.563 0 0 0 .475-.345L11.48 3.5Z"></path>
   </svg>
  </div>
  <div class="flex-grow sm:text-left text-center mt-6 sm:mt-0">
   <h2 class="text-gray-900 text-lg title-font font-medium mb-2">Shooting Stars</h2>
   <p class="leading-relaxed text-base">We aim and help other to aim for higher things so that no
  one has to suffer.</p>
   <a class="mt-3 text-rose-500 inline-flex items-center" routerLink="/about-us">Learn More
    <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-
  width="2"
     class="w-4 h-4 ml-2" viewBox="0 0 24 24">
     <path d="M5 12h14M12 5l7 7-7 7"></path>
```

```
        </svg>
      </a>
    </div>
  </div>
  <div class="flex items-center lg:w--3/5 mx-auto border-b pb-10 mb-10 border-gray-200 sm:flex-row
    flex-col">
    <div class="flex-grow sm:text-left text-center mt-6 sm:mt-0">
      <h2 class="text-gray-900 text-lg title-font font-medium mb-2">The Catalyzer</h2>
      <p class="leading-relaxed text-base">We cut the harrassment of the loan sharks and So called
      financers by providing you total no feecharged crowd funding.</p>
      <a class="mt-3 text-rose-500 inline-flex items-center" routerLink="/about-us">Learn More
        <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-
        width="2"
          class="w-4 h-4 ml-2" viewBox="0 0 24 24">
          <path d="M5 12h14M12 5l7 7-7 7"></path>
        </svg>
      </a>
    </div>


    <div
      class="sm:w-32 sm:order-none order-first sm:h-32 h-20 w-20 sm:ml-10 inline-flex items-center
      justify-center rounded-full bg-rose-100 text-rose-500 flex-shrink-0">
      <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-
      width="2"
        class="sm:w-16 sm:h-16 w-10 h-10" viewBox="0 0 24 24">
        <circle cx="6" cy="6" r="3"></circle>
        <circle cx="6" cy="18" r="3"></circle>
        <path d="M20 4L8.12 15.88M14.47 14.48L20 20M8.12 8.12L12 12"></path>
      </svg>
    </div>
  </div>
  <div class="flex items-center lg:w--3/5 mx-auto sm:flex-row flex-col">
    <div
      class="sm:w-32 sm:h-32 h-20 w-20 sm:mr-10 inline-flex items-center justify-center rounded-full
      bg-rose-100 text-rose-500 flex-shrink-0 p-6">
      <svg data-slot="icon" fill="none" stroke-width="1.5" stroke="currentColor" viewBox="0 0 24 24"
      xmlns="http://www.w3.org/2000/svg" aria-hidden="true">
        <path stroke-linecap="round" stroke-linejoin="round" d="M3.98 8.223A10.477 10.477 0 0 0
        1.934 12C3.226 16.338 7.244 19.5 12 19.5c.993 0 1.953-.138 2.863-.395M6.228 6.228A10.451
        10.451 0 0 1 12 4.5c4.756 0 8.773 3.162 10.065 7.498a10.522 10.522 0 0 1-4.293 5.774M6.228
```

```
6.228 3 3m3.228 3.228 3.65 3.65m7.894 7.894L21 21m-3.228-3.228-3.65-3.65m0 0a3 3 0 1 0-
4.243-4.243m4.242 4.242L9.88 9.88"></path>
      </svg>
    </div>
    <div class="flex-grow sm:text-left text-center mt-6 sm:mt-0">
      <h2 class="text-gray-900 text-lg title-font font-medium mb-2">The James Bond</h2>
      <p class="leading-relaxed text-base">We take your security seriously will never let your data out
anywhere.</p>
      <a class="mt-3 text-rose-500 inline-flex items-center" routerLink="/about-us">Learn More
        <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-
width="2"
          class="w-4 h-4 ml-2" viewBox="0 0 24 24">
          <path d="M5 12h14M12 5l7 7-7 7"></path>
        </svg>
      </a>
    </div>
  </div>
 </div>
</section>
```

- **feature.component.ts (src\app\pages\home\feature\feature.component.ts)**

```
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-feature',
  standalone: true,
  imports: [RouterModule],
  templateUrl: './feature.component.html',
  styles: ''
})
export class FeatureComponent {}
```

- **Mini-about.component.html (Src\app\pages\home\mini-about\mini-about.component.html)**

```
<!-- Part of home page -->
< section class="bg-rose-500 text-white px-5 md:px-20 py-8">
   <h2 class="text-center text-3xl uppercase mb-7">Who We Are?</h2>
```

```html
<p class="px-10 text-center text-lg">Taabir is on a mission to revolutionize crowdfunding by
connecting people who want to make a difference. Together, they empower individuals and
organizations by providing an accessible and innovative platform where dreams become reality.
Through empathy, passion, and collaboration, Taabir transform challenges into successes, shaping a
brighter future for countless lives. Join Taabir on their journey to change the world, one campaign
at a time..</p>
</section>
<section class="text-gray-950 body-font px-5 md:px-20 py-8 bg-gray-100">
    <div class="lg:w-2/3 flex flex-col sm:flex-row sm:items-center items-start mx-auto">
      <h1 class="flex-grow sm:pr-16 text-2xl font-medium title-font text-gray-900">Join our
community, fund your dreams, and embark on life-changing journeys together.</h1>
      <button (click)="router.navigateByUrl('auth/login')"
        class="flex-shrink-0 text-white bg-rose-500 border-0 py-2 px-8 focus:outline-none hover:bg-
rose-600 rounded text-lg mt-10 sm:mt-0">Join Now</button>
    </div>
</section>
```

- **Mini-about.component.html (Src\app\pages\home\mini-about\mini-about.component.html)**

```typescript
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-mini-about',
  standalone: true,
  imports: [],
  templateUrl: './mini-about.component.html',
  styles: "
})
export class MiniAboutComponent {
  constructor(protected router: Router){}
}
```

- **Home.component.html (Src\app\pages\home\home.component.html)**

```html
<!-- Included 'base-header.component.html' -->
<app-base-header></app-base-header>
  <!-- Included 'cta.component.html' -->
  <app-cta />

  <!-- Included 'mini-about.component.html' -->
```

```html
<app-mini-about />

<!-- Included 'feature.component.html' -->
<app-feature />

<!-- Included 'base-footer.component.html' -->
<app-base-footer />
```

- **home.component.ts (src\app\pages\home\home.component.ts)**

```typescript
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { CtaComponent } from './cta/cta.component';
import { FeatureComponent } from './feature/feature.component';
import { MiniAboutComponent } from './mini-about/mini-about.component';

@Component({
  selector: 'app-home',
  standalone: true,
  imports: [
    RouterModule,
    BaseHeaderComponent,
    BaseFooterComponent,
    CtaComponent,
    FeatureComponent,
    MiniAboutComponent],
  templateUrl: './home.component.html',
  styles: ''
})
export class HomeComponent {}
```

3. About Page

- **faq.component.html (src\app\pages\about\faq\faq.component.html)**

```
<!-- Part of About page –>
@if (faqs$ | async) {
<h2 class="my-8 text-3xl underline font-bold text-center">Frequently Asked Questions</h2>
<div class="m-2 space-y-2 px-5">
@for (faq of faqs$ | async; track $index) {
   @if (faq.active) {
     <div class="group flex flex-col gap-2 rounded-lg bg-rose-500 p-5 text-white" [tabindex]="$index
   + 1">
        <div class="flex cursor-pointer items-center justify-between">
          <span class="font-semibold text-lg capitalize"> {{faq.title}} </span>
          <img src="https://upload.wikimedia.org/wikipedia/commons/9/96/Chevron-icon-drop-down-
   menu-WHITE.png"
             class="h-2 w-3 transition-all duration-500 group-focus:-rotate-180" />
        </div>
        <div
          class="invisible h-auto max-h-0 items-center opacity-0 transition-all group-focus:visible
   group-focus:max-h-screen group-focus:opacity-100 group-focus:duration-1000">
          {{faq.description}}
        </div>
     </div>
     }
   }
</div>
}
```

- **faq.component.ts (src\app\pages\about\faq\faq.component.ts)**

```typescript
import { HttpClientModule } from '@angular/common/http';
import { Component } from '@angular/core';
import { FaqService } from '../../../services/faq.service';
import { AsyncPipe } from '@angular/common';

@Component({
  selector: 'app-faq',
  standalone: true,
  imports: [
   HttpClientModule,
   AsyncPipe
  ],
  providers: [FaqService],
```

```
    templateUrl: './faq.component.html',
    styles: ''
})
export class FaqComponent {
  faqs$ = this.faqService.getFaq();
  constructor(private faqService: FaqService) { }
}
```

- **About.component.html (Src\app\pages\about\about.component.html)**

```
<!-- Included base-header.component.html -->
<app-base-header />
<div class="h-screen mt-24 bg-gray-50">
    <div class="flex px-6 gap-10 flex-wrap md:flex-nowrap items-center h-full">
        <div class="flex flex-col items-start justify-center h-full">
            <h1 class="text-4xl font-bold text-gray-800 text-left">About Us</h1>
            <p class="text-gray-700 mt-4 text-md w-11/12">Welcome to Taabir, a pioneering
crowdfunding platform dedicated to transforming lives and fueling dreams. We connect
compassionate individuals, groups, and organizations worldwide, empowering them to create
meaningful change through collective action.
        Founded in 2022, Taabir has already helped countless individuals raise funds for various
causes, including medical emergencies, education, disaster relief, and innovative projects. Our
mission is to build a global community that uplifts those in need and cultivates a culture of empathy
and generosity.
        At Taabir, we believe in the power of unity and collaboration. Our platform provides the tools
and resources for people to come together and turn ideas into reality, creating a brighter future for
all. We are committed to fostering a safe, transparent, and inclusive environment that encourages
continuous growth and positive change.
        Join Taabir and become part of a movement that celebrates the potential of every person to
make a difference. With your help, we can turn dreams into reality and build a better, more
compassionate world for everyone..</p>
        </div>
        <img src="assets/img/home/about.jpg" class="w-[30vw] rounded shadow" />
    </div>
</div>

<h2 class="text-3xl font-bold text-gray-800 text-center pt-4">Why Us?</h2>
<!-- Included feature.component.html -->
<app-feature />

<div class="mb-10">
    <!-- Included faq.component.html -->
```

```
    <app-faq />
</div>
<!-- Included base-footer.component.html -->
<app-base-footer />
```

- **about.component.ts (src\app\pages\about\about.component.ts)**

```
import { Component } from '@angular/core';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { RouterModule } from '@angular/router';
import { FaqComponent } from './faq/faq.component';
import { FeatureComponent } from '../home/feature/feature.component';

@Component({
  selector: 'app-about',
  standalone: true,
  imports: [
    BaseHeaderComponent,
    BaseFooterComponent,
    RouterModule,
    FaqComponent,
    FeatureComponent
  ],
  templateUrl: './about.component.html',
  styles: ''
})
export class AboutComponent {}
```

4. Contact Page

- **Contact.component.html (src\app\pages\contact\contact.component.html)**

```
<!-- included base-header.component.html -->
<app-base-header class="relative z-30" />

<section
  class=" body-font relative contact-bg z-0 h-[78vh] flex justify-center gap-10 items-center py-5 mt-28
    md:px-48">
  <div
    class="w-2/3 bg-white rounded-lg p-8 flex flex-col md:ml-auto mt-10 md:mt-0 h-[50vh] relative
    z-10 shadow-md mx-auto mb-10">
    <h2 class="text-3xl font-semibold">Contact Us</h2>
    <p class="leading-relaxed mb-10">You can contact us through the following means:</p>
```

```html
<ul class="text-lg h-2/3 flex flex-col gap-4 justify-start list-disc pl-5">
  <li>
    Email: taabir.project{{'@'}}proton.me
  </li>
  <li>
    Phone: +91 9912345678
  </li>
  <li>
    Address: IAEA HOUSE, 17-B, Ring Rd, IP Estate, New Delhi, Delhi, 110002
  </li>
</ul>
</div>
</section>
<!-- included base-footer.component.html -->
<app-base-footer />
```

- **contact.component.ts (src\app\pages\contact\contact.component.ts)**

```typescript
import { Component } from '@angular/core';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { TubButttonComponent } from '../../ui/tub-buttton/tub-buttton.component';

@Component({
  selector: 'app-contact',
  standalone: true,
  imports: [
    BaseHeaderComponent,
    BaseFooterComponent,
    TubButttonComponent
  ],
  templateUrl: './contact.component.html',
  styleUrl: './contact.component.css'
})
export class ContactComponent {}
```

- **Contact.component.css (src\app\pages\contact\contact.component.css)**

```css
.contact-bg {
  @apply bg-gradient-to-br from-rose-500 to-rose-800 bg-cover bg-no-repeat;
}
input:focus, textarea:focus{
```

```
  @apply outline-0 ring-0 shadow-none;
}
```

5. Explore Page

- **Explore.component.html (src\app\pages\explore\explore.component.html)**

```html
<!-- including base-header.component.html -->
<app-base-header />
<section class="text-gray-950 body-font bg-rose-100 min-h-screen mt-16 pt-20">
  <h2 class="text-center uppercase text-4xl font-semibold mt--5">
    Explore Fundraisers
  </h2>
  <p class="text-center">Contribute to someone and make their lives easier</p>
  <div class="container px-5 py-24 mx-auto ">
    <div class="flex flex-wrap -m-4 gap-6 justify-center">
      @if (fundraises.length > 0) {
      @for (fund of fundraises; track $index) {
      <div class="lg:w-1/4 md:w-1/2 bg-gray-50 w-full shadow rounded">
        <a class="block relative h-48 rounded overflow-hidden">
          <img [alt]="fund.title" class="object-cover object-center w-full h-full block"
            [src]="fund.imageUrl">
        </a>
        <div class="p-4 mt-4 flex justify-between items-center">
          <div>
            <h2 class="text-gray-900 title-font text-lg font-medium">{{fund.title}}</h2>
            <p class="mt-1">{{fund.amount | currency:'INR':true}}</p>
          </div>
          <a [routerLink]="'/fundraiser/' + btoa(fund.fundId)">
            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
  stroke-width="1.5"
              stroke="currentColor" class="w-12 h-12 cursor-pointer transition-all
  hover:stroke-rose-600">
              <path stroke-linecap="round" stroke-linejoin="round"
                d="m12.75 15 3-3m0 0-3-3m3 3h-7.5M21 12a9 9 0 1 1-18 0 9 9 0 0 1 18 0Z"
  />
            </svg>
          </a>
        </div>
      </div>
```

```
      }}
    </div>
  </div>
</section>
<!-- including base-footer.component.html -->
<app-base-footer />
```

- **explore.component.ts (src\app\pages\explore\explore.component.ts)**

```
import { Component, OnInit } from '@angular/core';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { FundRaiser } from '../../../assets/data/fundraises';
import { PaymentService } from '../../services/payment.service';
import { HttpClientModule } from '@angular/common/http';
import { CurrencyPipe } from '@angular/common';
import { RouterModule } from '@angular/router';
import { FundraisesService } from '../../services/fundraises.service';

@Component({
  selector: 'app-explore',
  standalone: true,
  imports: [
    BaseHeaderComponent,
    BaseFooterComponent,
    HttpClientModule,
    CurrencyPipe,
    RouterModule
  ],
  providers: [PaymentService],
  templateUrl: './explore.component.html',
  styles: ''
})
export class ExploreComponent implements OnInit{

  btoa = (id: number) => btoa(id.toString()); // Shorthand to convert string to BASE64 encode
  fundraises: FundRaiser[] = []; // Fundraiser array
  constructor(private fundService: FundraisesService ) {}
  ngOnInit(): void {
    this.fundService.getFundraises().subscribe(fundraisers => {
     this.fundraises = fundraisers.filter(f => f.isActive);
    });
```

```
    }
}
```

6. Single Fundraiser Page

● **Single-fundraiser.component.html (src\app\pages\fundraiser\single-fundraiser.component.html)**

```
@if(fundraiser){
<app-base-header />
<section class="text-gray-950 bg-slate-100 body-font overflow-hidden">
  <div class="container px-5 py-24 mx-auto">
    <div class="lg:w-4/5 mx-auto flex flex-wrap">
      <div class="lg:w-1/2 w-full lg:pr-10 lg:py-6 mb-6 lg:mb-0">
        <button (click)="back()"
          class="flex items-center justify-center w-1/2 py-2 text-sm text-gray-700 transition-colors
duration-20 mb-5 gap-x-2 sm:w-auto dark:hover:bg-gray-800 dark:bg-gray-900 hover:bg-gray-100
dark:text-gray-200 dark:border-gray-700">
          <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-
width="1.5"
            stroke="currentColor" class="w-5 h-5 rtl:rotate-180">
            <path stroke-linecap="round" stroke-linejoin="round"
              d="M6.75 15.75L3 12m0 0l3.75-3.75M3 12h18" />
          </svg>
          <span>Go back</span>
        </button>
        <h2 class="text-sm title-font text-gray-500 tracking-widest uppercase">Help the one in need
now!</h2>
        <h1 class="text-gray-900 text-3xl title-font font-medium mb-4">{{fundraiser.title}}</h1>
        <p class="leading-relaxed mb-4">
          {{fundraiser.description}}
        </p>
        <div class="flex border-t border-gray-200 py-2">
          <span class="text-gray-500">Total Amount</span>
          <span class="ml-auto text-gray-900">{{fundraiser.amount | currency:'INR':true}}</span>
        </div>
        <div class="flex border-t border-gray-200 py-2">
          <span class="text-gray-500">Amount Needed</span>
          <span class="ml-auto text-gray-900">{{fundraiser.amount - paidAmount |
currency:'INR':true}}</span>
        </div>
        <div class="flex border-t border-b mb-6 border-gray-200 py-2">
          <span class="text-gray-500">Active</span>
          <span class="ml-auto text-gray-900">{{fundraiser.isActive ? '&check;' :
'&cross;'}}</span>
```

```
            </div>
            @if (fundraiser.isActive && fundraiser.userId !== userId) {
            <div class="flex justify-between items-center">
                <label for="" class="w-1/2">
                    <small class="text-xs font-semibold">Enter the Amount you wish to contribute.</small>
                    <input min="1" [max]="fundraiser.amount - paidAmount" class="title-font font-medium
    text-lg text-gray-900 w-full" type="number"
                        (input)="setAmountToPay($event)" [value]="amountToPay" [disabled]="userId > 0
    && user && user.isVerified == false" />
                    @if (userId > 0 && user && user.isVerified == false) {
                    <small class="bg-amber-500 text-white p-1 rounded">
                        <i class="py-0.5 px-2.5 text-amber-500 bg-white rounded-full">i</i>
                        Thank for joining us,we are going through some formalities you will be notified
    soon!
                    </small>
                    }
                </label>
                <span>
                    <p class="text-right">{{amountToPay | currency:'INR':true}}</p>
                    <!-- include 'pg.component.html' -->
                    <app-pg [amount]="amountToPay"[fundId]="fundraiser.fundId" (paid)="reload()" />
                </span>
            </div>}
            @else if(fundraiser.isActive && fundraiser.userId !== userId){
                <p class="bg-green-700 text-white p-3 rounded">
                    <i class="py-0.5 px-2.5 text-green-700 bg-white rounded-full">i</i>
                    This is a fundraiser created by you, So you can't Contribute to it.</p>
            }
            @else {
                <p class="bg-green-700 text-white p-3 rounded">
                    <i class="py-0.5 px-2.5 text-green-700 bg-white rounded-full">i</i>
                    As of now this fundraiser is not taking any contributions.</p>
            }
        </div>
        <img [alt]="fundraiser.title" class="lg:w-1/2 w-full lg:h-auto h-64 object-cover object-center
    rounded"
            [src]="fundraiser.imageUrl">
        </div>
    </div>
</section>
<app-base-footer />}
```

- **single-fundraiser.component.ts (src\app\pages\fundraiser\single-fundraiser.component.ts)**

```typescript
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router, RouterModule } from '@angular/router';
import { PgComponent } from '../../ui/pg/pg.component';
import { PaymentService } from '../../services/payment.service';
import { FundRaiser } from '../../../assets/data/fundraises';
import { CurrencyPipe } from '@angular/common';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { FundraisesService } from '../../services/fundraises.service';
import { User } from '../../../assets/data/users';
import { AdminService } from '../../services/admin.service';
import { switchMap } from 'rxjs';

@Component({
  selector: 'app-fundraiser',
  standalone: true,
  imports: [RouterModule, PgComponent, CurrencyPipe, BaseHeaderComponent,
    BaseFooterComponent],
  providers: [PaymentService],
  templateUrl: './single-fundraiser.component.html',
  styles: ''
})
export class SingleFundraiserComponent implements OnInit {

  toggleModal: boolean = false;
  constructor(private router: Router,
        private route: ActivatedRoute,
        private fundService: FundraisesService,
        private userService: AdminService
      ) { }
  fundraiser!: FundRaiser;
  amountToPay: number = 0;
  paidAmount = 0;
  user?: User;
  userId = +(localStorage.getItem('auth-id') ?? -1);
  ngOnInit(): void {

    this.route.paramMap.subscribe(param => {
      if (param.has('fid')) {
        let fid = +atob(param.get('fid') as string);
```

```
    this.fundService.getFundraises().pipe(switchMap(fundraiser => {
      if (fundraiser.length)
        {this.fundraiser = fundraiser.find(f => f.fundId == +fid)!;}
      return this.fundService.getAmountNeededForFundRaiser(this.fundraiser.fundId);
    })).subscribe(res => this.paidAmount = res.paidAmount as number);
    }
    else {
     this.router.navigate(['not-found']);
    }
  });

  this.userService.getUsers().subscribe(res => {
   this.user = res.find(user => user.userId == this.userId);
  });
 }


 setAmountToPay(event: Event) {
  let target = (event.target as HTMLInputElement);
  if (+target.value > +target.max) {
   target.value = target.max;
  }
  else if (+target.value < +target.min) {
   target.value = target.min;
  }
  this.amountToPay = +target.value;
 }
 back() {
  window.history.back();
 }
 reload(){
   location.reload();
 }
}
```

7. Payment Gateway Page

- **Pg.component.html (src\app\ui\pg\pg.component.html)**

```
<button [disabled]="amount <= 0" class="text-white bg-rose-500 border-0 py-2 px-6 focus:outline-
    none hover:bg-rose-600 rounded text-lg"
   (click)="myDialog.showModal()">Pay Now</button>
<dialog #myDialog class="w-2/3 mx-auto rounded shadow-white p-5">
```

```
@if (isLogged === 0) {
<div class="w-full mb-5 h-10">
    <span class="float-right text-2xl">x</span>
</div>}

@if (isLogged !== 0) {

@if (paymentDone) {
<div class="grid place-items-center gap-3">
    <img class="w-60" src="https://uxwing.com/wp-content/themes/uxwing/download/checkmark-
  cross/done-icon.png"
       alt="">
    <p>Thanks a lot for Contributng for a greater cause</p>
</div>
}
@else {


<div class="flex my-5 justify-between items-center">
    <div>
       <h3 class="text-2xl">🟧Taabir Pay</h3>
       <p class="text-gray-500">Taabir Pay is the easiest way to pay.</p>
    </div>
    <div class=" w-1/3">
       <h3 class="text-2xl text-rose-500 font-semibold">₹{{amount + amount * (18 /100)}} </h3>
       <p class="text-gray-500">(₹{{amount}} + 18% GST).</p>
    </div>
</div>

<div class="pay-form">
    <div class="w-full flex gap-4">
       <button [class]="{'tabs': true, 'active': paymentType == 1}" (click)="paymentType =
  1;">UPI</button>
       <button [class]="{'tabs': true, 'active': paymentType == 2}" (click)="paymentType =
  2;">Card</button>
       <button [class]="{'tabs': true, 'active': paymentType == 3}" (click)="paymentType =
  3;">Wallet</button>
    </div>
    <form [formGroup]="payForm" class="w-full h-[50vh] rounded m-2 border-4 border-gray-800 p-
  3 flex items-center justify-center">
       <div [class]="{'hidden': paymentType != 1}">
          <div class="upi">
```

```html
          <div class="qr-code flex items-center flex-col border-r-2 pr-5">
            <img src="assets/img/demo-qr.png" alt="" class="w-32 border-2 border-rose-600
rounded">
            <p class="w-32 text-center">Scan this QR code to Pay</p>
          </div>
          <div class="form-group h-full my-auto">
            <label for="upi" class="block">Or Enter your UPI ID to Pay: </label>
            <input [formControl]="payForm.controls.forUpi.controls.upiId" type="text"
class="rounded outline-transparent focus:border-rose-600" id="upi"
                placeholder="Enter UPI: example@bank">
          </div>
        </div>
      </div>
      <div [class]="{'hidden': paymentType != 2}">
        <div class="form-group mt-2">
          <label for="cardNumber" class="block">Card Number</label>
          <input [formControl]="payForm.controls.forCard.controls.cardNumber" type="text"
class="form-control rounded" id="cardNumber" placeholder="Enter Card Number">
        </div>
        <div class="form-group mt-2">
          <label for="expiryDate" class="block">Expiry Date</label>
          <input [formControl]="payForm.controls.forCard.controls.expiryDate" [min]="today"
type="date" class="form-control rounded" id="expiryDate" placeholder="Enter Expiry Date">
        </div>
        <div class="form-group mt-2">
          <label for="cvv" class="block">CVV</label>
          <input [formControl]="payForm.controls.forCard.controls.cvv" maxlength="3"
minlength="3" type="password" class="form-control rounded" id="cvv" placeholder="Enter
CVV">
        </div>
      </div>
      <div [class]="{'hidden': paymentType != 3}">
        <div class="grid grid-cols-2 gap-2">
          <label class="wallet" for="wallet1">
            <img class="my-5"
              src="https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-
media/paytm-icon.png"
              alt="">
            <span>Paytm</span>
            <input type="radio" [formControl]="payForm.controls.forWallet.controls.walletName"
value="paytm" name="wallet" id="wallet1">
          </label>
```

```html
        <label class="wallet" for="wallet2">
          <img src="https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-media/phonepe-icon.png"
            alt="">
          <span>Phonepe</span>
          <input type="radio" [formControl]="payForm.controls.forWallet.controls.walletName"
   value="phonepe" name="wallet" id="wallet2">
        </label>
        <label class="wallet" for="wallet3">
          <img src="https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-media/google-pay-icon.png"
            alt="">
          <span>GPay</span>
          <input type="radio" [formControl]="payForm.controls.forWallet.controls.walletName"
   value="google-pay" name="wallet" id="wallet3">
        </label>
        <label class="wallet" for="wallet4">
          <img class="my-5"
            src="https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-media/amazon-pay-icon.png"
            alt="">
          <span>Amazon Pay</span>
          <input type="radio" [formControl]="payForm.controls.forWallet.controls.walletName"
   value="amazon-pay" name="wallet" id="wallet4">
        </label>
      </div>
    </div>
  </form>
</div>
<div class="button-group">
  <button (click)="doPayment()"
    class="w-32 text-center text-white bg-rose-500 border-0 py-2 px-6 focus:outline-none hover:bg-rose-600 rounded text-lg">
    Pay
  </button>
  <button (click)="myDialog.close()"
    class="w-32 text-center text-white bg-gray-700 border-0 py-2 px-6 focus:outline-none hover:bg-gray-600 rounded text-lg">
    Close
  </button>
</div> }
}
```

```
@else{
  <div class="grid place-items-center gap-10">
    <h2 class="text-2xl font-semibold">It seems like you are not logged In</h2>
    <p class="flex gap-3">
      <button (click)="router.navigateByUrl('auth/login')" class="inline-flex text-white bg-rose-500
border-0 py-2 px-6 focus:outline-none hover:bg-rose-600 rounded text-lg">
        Login Now
      </button> <button (click)="isLogged = 2; isGuestPayment = true;" class="inline-flex text-
white bg-gray-800 border-0 py-2 px-6 focus:outline-none hover:bg-gray-600 rounded text-lg">
        Pay as Guest
      </button>

    </p>
  </div>
}
</dialog>
```

- **pg.component.ts (src\app\ui\pg\pg.component.ts)**

```typescript
import { Component, EventEmitter, Input, Output, ViewChild } from '@angular/core';
import { PaymentService } from '../../services/payment.service';
import { isLoggedIn } from '../../../helpers/auth';
import { Router, RouterModule } from '@angular/router';
import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';
import { Payment, PaymentMode } from '../../../assets/data/fundraises';
import { v4 as uuidv4 } from 'uuid';

@Component({
  selector: 'app-pg',
  standalone: true,
  imports: [RouterModule, ReactiveFormsModule],
  providers: [
    PaymentService,
  ],
  templateUrl: './pg.component.html',
  styleUrl: './pg.component.css'
})
export class PgComponent {

  @Input() amount = 0;
  @Input() fundId!: number;
  @Output() paid = new EventEmitter<boolean>();
  @ViewChild('myDialog') myDialog!: any;

  paymentType = PaymentMode.UPI;
  paymentDone = false;
  isLogged = isLoggedIn() ? 1 : 0;
  isGuestPayment = false;
  today = (new Date()).toISOString().split('T')[0];
  payForm = this.fb.group({
    forUpi: this.fb.group({
      upiId: ['', Validators.required]
    }),
    forCard: this.fb.group({
      cardNumber: ['', Validators.required],
      expiryDate: ['', Validators.required],
      cvv: ['', [Validators.required, Validators.pattern(/\d{3}/)]]
    }),
    forWallet: this.fb.group({
```

```
    walletName: [", Validators.required]
  }),
});

constructor(private payService: PaymentService, protected router: Router, private fb: FormBuilder) {
  }

doPayment() {

  let isUpi = this.paymentType == PaymentMode.UPI && this.payForm.controls.forUpi.invalid;
  let isCard = this.paymentType == PaymentMode.Card && this.payForm.controls.forCard.invalid;
  let isWallet = this.paymentType == PaymentMode.Wallet &&
    this.payForm.controls.forWallet.invalid;

  if (isUpi || isCard || isWallet) {
    alert('Fill all the details');
    return;
  }

  let paymentDetails = ";

  switch (this.paymentType) {
    case PaymentMode.UPI:
      paymentDetails = JSON.stringify(this.payForm.controls.forUpi.value);
      break;
    case PaymentMode.Card:
      paymentDetails = JSON.stringify(this.payForm.controls.forCard.value);
      break;
    case PaymentMode.Wallet:
      paymentDetails = JSON.stringify(this.payForm.controls.forWallet.value);
      break;
    default:
      break;
  }

  let payment: Payment = {
    paymentMode: this.paymentType,
    isGuest: this.isGuestPayment,
    isVerified: false,
    amount: this.amount,
    fundId: this.fundId,
    userId: (this.isLogged ? +localStorage.getItem('auth-id')! : -1),
```

```
      paymentReferenceNumber: uuidv4(),
      paymentCreds: btoa(paymentDetails),
      paymentId: -1,
    };

    this.payService.pay(payment).subscribe(
      res => {
        this.paymentDone = true;
        setTimeout(() => {
          this.isLogged = isLoggedIn() ? 1 : 0;
          this.paid.emit(true);
          this.myDialog.nativeElement.close();
        }, 2000);
      }
    );
  }

}
```

● **Pg.component.css (Src\app\ui\pg\pg.component.css)**

```css
.button-group{
   @apply flex gap-4 items-center justify-center mt-5;
}

.pay-form
{
   @apply flex flex-col gap-4 items-center justify-center mt-5;
}

.tabs{
   @apply w-1/3 bg-gray-800 text-white rounded p-3;
}

.tabs.active{
   @apply bg-rose-600 shadow;
}

.upi{
   @apply flex gap-3
}

input[name="wallet"]{
```

```css
  @apply invisible;
}

label.wallet {
  @apply grid place-items-center border-2 rounded w-32 h-32 p-2;
}

label.wallet:has(input[type=radio]:checked){
  border-color: var(--primary);
}

label img {
  @apply w-20 object-cover;
}

button[disabled]{
  opacity: 0.6;
  pointer-events: none;
}
```

8. Login Page

- **Login.component.html (src\app\auth\login\login.component.html)**

```html
<app-base-header />
<div class="flex min-h-screen flex-col justify-center px-6 py-12 lg:px-8 bg mt-20">
  <div class="bg-white h-full m-auto p-8 rounded shadow w-1/2">
   <div class="sm:mx-auto sm:w-full sm:max-w-sm">
    <h2 class="mt-10 text-center text-2xl font-bold leading-9 tracking-tight text-gray-950
    underline">Sign in to your Account
    </h2>
   </div>

   <div class="mt-10 sm:mx-auto sm:w-full sm:max-w-sm">
    <form class="space-y-6" [formGroup]="loginForm" (ngSubmit)="submit()">
     <div>
      <label for="email" class="block text-sm font-medium leading-6 text-gray-950">Email
    address</label>
      <div class="mt-2">
       <input id="email" formControlName="email" type="email"  required
        class="block w-full rounded-md py-1.5 text-gray-950 shadow-sm bg-transparent
    placeholder:text-gray-950 sm:text-sm sm:leading-6 border border-gray-800">
```

```html
      </div>
    </div>

    <div>
      <div class="flex items-center justify-between">
        <label for="password" class="block text-sm font-medium leading-6 text-gray-
950">Password</label>
      </div>
      <div class="mt-2">
        <input id="password" formControlName="password" type="password"  required
          class="block w-full rounded-md border py-1.5 text-gray-950 shadow-sm border-gray-800
placeholder:text-gray-950sm:text-sm sm:leading-6 bg-transparent">
      </div>
    </div>

    <div>
      <button type="submit"
        class="flex w-full justify-center rounded-md text-white hover:opacity-70 px-3 py-1.5 text-sm
font-semibold leading-6 bg-rose-600 hover:shadow">Sign in</button>
        <p class="mt-3 text-gray-950 text-center">Don't have an Account yet <a
routerLink="/auth/register" class="text-rose-600 font-semibold">Register Now!</a></p>
      </div>
    </form>

  </div>
 </div>
</div>
<app-base-footer />
```

● **login.component.ts (src\app\auth\login\login.component.ts)**

```typescript
import { Component } from '@angular/core';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { AuthService } from '../../services/auth.service';
import { Router, RouterModule } from '@angular/router';

@Component({
 selector: 'app-login',
 standalone: true,
 imports: [
  ReactiveFormsModule,
```

```typescript
    BaseHeaderComponent,
    BaseFooterComponent,
    RouterModule
  ],
  providers: [],
  templateUrl: './login.component.html',
  styleUrl: './login.component.css'
})
export class LoginComponent {

  loginForm = this.fb.group({
    email: this.fb.control('', [Validators.required, Validators.email]),
    password: this.fb.control('', [Validators.required, Validators.minLength(8)])
  });

  constructor(private fb: FormBuilder, private authService: AuthService, private route: Router) { }
  submit() {
    if (this.loginForm.invalid) {
      alert('Fill all the details');
    }
    else {
      let login = {
        email: this.loginForm.value.email!,
        password: this.loginForm.value.password!
      };
      this.authService.login(login).subscribe(
        res => {
          alert(res.message);
          this.route.navigateByUrl('/');
        },
        err => {
          alert(err.error?.message ?? "Technical Issue!");
        }
      );
    }
  }
}
```

- **Login.component.css (src\app\auth\login\login.component.css)**

```css
.bg {
    @apply bg-gradient-to-tr from-rose-500 to-rose-800 text-white rounded;
}
```

9. Register Page

- **Register.component.html (src\app\auth\register\register.component.html)**

```html
<app-base-header />
<div class="flex min-h-screen flex-col justify-center px-6 py-12 lg:px-8 bg mt-20">
 <div class="bg-white h-full m-auto p-8 rounded shadow w-1/2">
  <div class="sm:mx-auto sm:w-full sm:max-w-sm">
   <h2 class="mt-10 text-center text-2xl font-bold leading-9 tracking-tight text-gray-950
   underline">Sign in to your Account
   </h2>
  </div>

  <div class="mt-10 sm:mx-auto sm:w-full sm:max-w-sm">
   <form class="space-y-6" [formGroup]="registerForm" (ngSubmit)="submit()">
    <div>
     <label for="name" class="block text-sm font-medium leading-6 text-gray-950">Full
   Name</label>
     <div class="mt-2">
      <input id="name" placeholder="Rahul Kathak" formControlName="name" type="text" required
       class="block w-full rounded-md py-1.5 text-gray-950 shadow-sm bg-transparent
   placeholder:text-gray-950 sm:text-sm sm:leading-6 border border-gray-800">
     </div>
    </div>
    <div>
     <label for="email" class="block text-sm font-medium leading-6 text-gray-950">Email
   address</label>
     <div class="mt-2">
      <input id="email" placeholder="example@mail.con" formControlName="email" type="email"
   required
       class="block w-full rounded-md py-1.5 text-gray-950 shadow-sm bg-transparent
   placeholder:text-gray-950 sm:text-sm sm:leading-6 border border-gray-800">
     </div>
    </div>
    <div>
     <label for="mobile" class="block text-sm font-medium leading-6 text-gray-950">Mobile
   Number</label>
     <div class="mt-2">
      <input id="mobile" maxlength="10" formControlName="mobile" type="tel" required
   placeholder="e.g. 8012345678"
       pattern="[0-9]{10}"
```

```
      class="block w-full rounded-md py-1.5 text-gray-950 shadow-sm bg-transparent
placeholder:text-gray-950 sm:text-sm sm:leading-6 border border-gray-800">
    </div>
  </div>

  <div>
    <label for="userType" class="block text-sm font-medium leading-6 text-gray-950">User
Type</label>
    <div class="mt-2">
      <select id="userType" formControlName="userType" required
(change)="toggleUserType($any($event).target)"
      class="block w-full rounded-md py-1.5 text-gray-950 shadow-sm bg-transparent sm:text-sm
sm:leading-6 border border-gray-800">
        <option [value]="null">Select Options</option>
        <option [value]="1">Individual</option>
        <option [value]="2">NGO</option>
      </select>
    </div>
  </div>
  <!-- Open fields 'ngo darpan id' and 'ngo website' if user is of NGO type -->
  @if(isNGO){

  <div>
    <div class="flex items-center justify-between">
      <label for="ngoId" class="block text-sm font-medium leading-6 text-gray-950">NGO Darpan
ID</label>
      <div class="text-sm">
      </div>
    </div>
    <div class="mt-2">
      <input id="ngoId" placeholder="AB/2010/0001234" formControlName="ngoId" type="text"
required
      class="block w-full rounded-md border py-1.5 text-gray-950 shadow-sm border-gray-800
placeholder:text-gray-950sm:text-sm sm:leading-6 bg-transparent">
    </div>
  </div>

  <div>
    <div class="flex items-center justify-between">
      <label for="website" class="block text-sm font-medium leading-6 text-gray-950">NGO
Website</label>
      <div class="text-sm">
```

```
        </div>
      </div>
    <div class="mt-2">
      <input id="website" placeholder="www.ngo.com" formControlName="website" type="url"
required
        class="block w-full rounded-md border py-1.5 text-gray-950 shadow-sm border-gray-800
placeholder:text-gray-950sm:text-sm sm:leading-6 bg-transparent">
    </div>
  </div>
  }

  <div>
    <div class="flex items-center justify-between">
      <label for="pan" class="block text-sm font-medium leading-6 text-gray-950">PAN
card</label>
      <div class="text-sm">
      </div>
    </div>
    <div class="mt-2">
      <input id="pan" placeholder="AABBC1111H" formControlName="panCard" type="text"
required
        class="block w-full rounded-md border py-1.5 text-gray-950 shadow-sm border-gray-800
placeholder:text-gray-950sm:text-sm sm:leading-6 bg-transparent">
    </div>
  </div>
  <div>
    <div class="flex items-center justify-between">
      <label for="password" class="block text-sm font-medium leading-6 text-gray-
950">Password</label>
      <div class="text-sm">
      </div>
    </div>
    <div class="mt-2">
      <input id="password" placeholder="********" formControlName="password"
type="password" required
        class="block w-full rounded-md border py-1.5 text-gray-950 shadow-sm border-gray-800
placeholder:text-gray-950sm:text-sm sm:leading-6 bg-transparent">
    </div>
  </div>

  <div>
    <button type="submit"
```

```
        class="flex w-full justify-center rounded-md text-white hover:opacity-70 px-3 py-1.5 text-sm
    font-semibold leading-6 bg-rose-600 hover:shadow">Sign
        Up</button>
      <p class="mt-3 text-gray-950 text-center">Don't have an Account yet <a
    routerLink="/auth/regiser"
          class="text-rose-600 font-semibold">Register Now!</a></p>
      </div>
      </form>


    </div>
    </div>
</div>
<app-base-footer />
```

- **Register.component.ts (src\app\auth\register\register.component.ts)**

```
import { Component } from '@angular/core';
import { Router, RouterModule } from '@angular/router';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';
import { AuthService } from '../../services/auth.service';
import { UserType } from '../../../assets/data/users';

@Component({
  selector: 'app-register',
  standalone: true,
  imports: [
    RouterModule,
    BaseHeaderComponent,
    BaseFooterComponent,
    ReactiveFormsModule
  ],
  providers: [AuthService],
  templateUrl: './register.component.html',
  styleUrl: './register.component.css'
})
export class RegisterComponent {

  isNGO = false;
  registerForm = this.fb.group({
    name: this.fb.control('', [Validators.required]),
```

```typescript
      email: this.fb.control(", [Validators.required, Validators.email]),
      mobile: this.fb.control(", [Validators.required]),
      panCard: this.fb.control(", [Validators.required, Validators.pattern(/[A-Z]{5}\d{4}[A-Z]/)]),
      password: this.fb.control(", [Validators.required, Validators.minLength(8)]),
      userType: this.fb.control<number|null>(null, [Validators.required]),
      ngoId: this.fb.control("),
      website: this.fb.control("),
    });

  constructor(private fb: FormBuilder, private authService: AuthService, private route: Router) { }

  submit() {

    if (this.registerForm.invalid) {
      Object.entries(this.registerForm.controls)
        .filter(r => r[1].invalid).forEach(con => console.log(con));
      alert('Fill all the details');
    }
    else {
      let login = {
        email: this.registerForm.value.email!,
        password: this.registerForm.value.password!,
        name: this.registerForm.value.name!,
        mobile: this.registerForm.value.mobile!,
        pan_card: this.registerForm.value.panCard!,
        user_type: +this.registerForm.value.userType!,
        ngo_id: this.registerForm.value.ngoId ?? null,
        website: this.registerForm.value.website ?? null
      };

      this.authService.register(login).subscribe(
        res => {
          alert(res.message);
          this.route.navigateByUrl('/');
        },
        err => {
          alert(err.error?.message || 'There is some issue, please try again');
        }
      );
    }
  }
```

```
toggleUserType(target: HTMLSelectElement){
 let arrays = ['ngoId', 'website'];

 // If option selected is NGO add the two fields else hide them
 if(+target.value == UserType.NGO){
  const urlPattern = /^(http:\/\/www\.|https:\/\/www\.|http:\/\/|https:\/\/)?[a-z0-9]+([\-\.]{1}[a-z0-9]+)*\.[a-z]{2,5}(:[0-9]{1,5})?(\/.*)?$/i;
  this.registerForm.get('website')!.setValidators([Validators.required, Validators.pattern(urlPattern)]);
  this.registerForm.get('ngoId')!.setValidators([Validators.required, Validators.minLength(6)]);
  this.isNGO = true;
 }
 else{
  arrays.forEach(element => {
   this.registerForm.get(element)!.setValidators(null);
  });
  this.isNGO = false;
 }
 this.registerForm.updateValueAndValidity();
}

}
```

● **Register.component.css (src\app\auth\register\register.component.css)**

```
.bg {
   @apply bg-gradient-to-tr from-rose-500 to-rose-800 text-white rounded;
}
```

10.   My Fundraisers Page

● **My-fundraisers.component.html (src\app\pages\fundraisers\my-fundraisers.component.html)**

```
<!-- Included base-header.component.html -->
<app-base-header />
<div class="h-screen mt-32 px-10 py-12">
   <div class="flex justify-between">
      <h2 class="text-3xl font-semibold uppercase">Your Fundraisers</h2>
      <!-- Included create-fundraiser.component.html -->
      <app-create-fundraiser [disabled]="(userId > 0 && user !== undefined && user.isVerified ==
      false)" [openModal]="editFund !== undefined" [fundraiser]="editFund"
      (closeModal)="checkClosedModal($event)" />
   </div>
```

```html
@if (fundraises$ | async; as fundraises) {
@if (fundraises.length) {
  <table class="funds-table">
    <thead class="bg-rose-500 text-white">
      <tr>
        <th>Title</th>
        <th>Amount</th>
        <th>Created At</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      @for (fundraiser of fundraises; track $index) {
      <tr>
        <td>{{fundraiser.title}}</td>
        <td>₹ {{fundraiser.amount}}</td>
        <td>{{fundraiser.createdAt | date}}</td>
        <td>
          <button class="bg-blue-500 text-white px-4 py-2 rounded mr-1"
(click)="edit(fundraiser)">Edit</button>
          <button class="bg-red-500 text-white px-4 py-2 rounded"
(click)="delete(fundraiser)">Delete</button>
        </td>
      </tr>
      }
    </tbody>
  </table>
}
@else {
<h2 class="text-2xl text-center text-gray-500 mt-10">No Fundraisers Found</h2>
}
}
</div>
<!-- Included base-footer.component.html -->
<app-base-footer />
```

- **my-fundraisers.component.ts (Src\app\pages\fundraisers\my-fundraisers.component.ts)**

```typescript
import { Component, OnInit } from '@angular/core';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { CreateFundraiserComponent } from '../create-fundraiser/create-fundraiser.component';
import { FundraisesService } from '../../services/fundraises.service';
```

```typescript
import { AsyncPipe, DatePipe } from '@angular/common';
import { FundRaiser } from '../../../assets/data/fundraises';
import { User } from '../../../assets/data/users';
import { AdminService } from '../../services/admin.service';

@Component({
  selector: 'app-fundraisers',
  standalone: true,
  imports: [
    BaseHeaderComponent,
    BaseFooterComponent,
    CreateFundraiserComponent,
    AsyncPipe,
    DatePipe
  ],
  providers: [],
  templateUrl: './my-fundraisers.component.html',
  styleUrl: './my-fundraisers.component.css'
})
export class MyFundraisersComponent implements OnInit{
  user_id = +localStorage.getItem('auth-id')!;
  fundraises$ = this.fundraiseService.getFundraises(this.user_id);
  user?: User;
  userId = +(localStorage.getItem('auth-id') ?? -1);
  editFund?: FundRaiser;

  constructor(private fundraiseService: FundraisesService, private userService: AdminService) { }

  ngOnInit(): void {
    this.userService.getUsers().subscribe(res => {
      this.user = res.find(user => user.userId == this.userId);
    });
  }

  checkClosedModal($event: any) {
    if ($event) {
      this.fundraises$ = this.fundraiseService.getFundraises(this.user_id);
    }
  }

  edit(fund: FundRaiser) {
    this.editFund = fund;
```

```
  }

  delete(fund: FundRaiser) {
    this.fundraiseService.deleteFundraise(fund.fundId).subscribe(_ =>
      this.fundraises$ = this.fundraiseService.getFundraises(this.user_id)
    );
  }
}
```

- **My-fundraisers.component.css (src\app\pages\fundraisers\my-fundraisers.component.css)**

```
.funds-table{
   /* Using tailwind '@apply' directive to add classes in a single class */
   @apply w-full text-left border m-8 mx-auto rounded shadow overflow-hidden;
}
td, th{
   padding: 4px 7px;
   border-right: 2px solid #e2e8f0;
}
```

11. Create Fundraisers Page

- **Create-fundraiser.component.html (src\app\pages\create-fundraiser\create-fundraiser.component.html)**

```html
<button (click)="open()" [disabled]="disabled"
   class=" text-white bg-rose-500 border-0 py-2 px-6 focus:outline-none hover:bg-rose-600 rounded
     text-lg float-right">
   Create Fundraiser
</button>

<dialog #myDialog class="w-2/3 mx-auto rounded shadow-white">
   <div class="header border-b">
     <div>
       <h2 class="text-2xl font-semibold uppercase">Create a Fundraiser</h2>
       <span class="text-3xl font-bold cursor-pointer" (click)="close()">&Cross;</span>
     </div>
   </div>
   <div class="body">
     <form [formGroup]="fundForm">
       @if (pageNumber == 1) {
```

```
<div class="flex flex-wrap gap-4 mx-auto justify-center py-5">
  <div class="form-group">
    <label for="title">Title</label>
    <input type="text" formControlName="title" id="title" placeholder="title" class="form-control" />
  </div>
  <div class="form-group">
    <label for="amount">Amount</label>
    <input type="number" id="amount" formControlName="amount" class="form-control" placeholder="Amount">
  </div>
  <div class="form-group description">
    <label for="description">Description</label>
    <textarea formControlName="description" id="description" class="form-control"
      placeholder="Description"></textarea>
  </div>
  <div class="form-group description">
    <label for="thumbnail">
      Thumb Nail / Fundraiser Image
    </label>
    <input accept=".png, .jpg, .jpeg" type="file" [required]="imageFile === ''" id="thumbnail" class="border"
      (change)="setThumbNail($event)"/>
    @if (fundraiser !== undefined && fundraiser.imageUrl === imageFile) {
      <span>Selected Earlier: <a [href]="imageFile" class="text-rose-600 font-medium" target="_blank">Click Here!</a></span>
    }
  </div>
</div>
<div class="button-group">
  <button type="button" (click)="next()"
    class=" text-white bg-rose-500 border-0 py-2 px-6 focus:outline-none hover:bg-rose-600 rounded text-lg w-56">
    Next
  </button>
</div> }
@else {
<div class="flex flex-wrap gap-4 mx-auto justify-center">
  <div class="form-group">
    <label for="accountName">Account Holder Name</label>
    <input type="text" id="accountName" formControlName="accountName" placeholder="Account Name">
```

```html
        </div>
        <div class="form-group">
          <label for="accountNumber">Account Holder Number</label>
          <input type="text" id="accountNumber" formControlName="accountNumber"
    placeholder="Account Name">
        </div>
        <div class="form-group">
          <label for="bankName">Bank Name</label>
          <select id="bankName" formControlName="bankName">
            <option value="">Select</option>
            @for (bank of (banks$ | async) | keyvalue; track $index) {
            <option [value]="bank.value">{{bank.key}}</option>
            }
          </select>

        </div>
      </div>

      <div class="button-group">
        <button (click)="pageNumber = 1"
          class=" text-white bg-gray-800 border-0 py-2 px-6 focus:outline-none hover:bg-gray-600
    rounded text-lg w-56">
          Previous
        </button>
        <button (click)="next(true)"
          class=" text-white bg-rose-500 border-0 py-2 px-6 focus:outline-none hover:bg-rose-600
    rounded text-lg w-56">
          {{ pageNumber == 1 ? 'Next' : 'Submit'}}
        </button>
      </div>
      }
    </form>
  </div>
</dialog>
```

- **Create-fundraiser.component.ts (src\app\pages\create-fundraiser\create-fundraiser.component.ts)**

```typescript
import { Component, ViewChild, Input, Output, EventEmitter, SimpleChanges, OnChanges, OnInit }
    from '@angular/core';
import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';
import { FundraisesService } from '../../services/fundraises.service';
import { AsyncPipe, KeyValuePipe } from '@angular/common';
import { FundRaiser } from '../../../assets/data/fundraises';
import { User } from '../../../assets/data/users';
import { AdminService } from '../../services/admin.service';

@Component({
  selector: 'app-create-fundraiser',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    AsyncPipe,
    KeyValuePipe
  ],
  providers: [],
  templateUrl: './create-fundraiser.component.html',
  styleUrl: './create-fundraiser.component.css'
})
export class CreateFundraiserComponent implements OnChanges, OnInit {

  @Input() openModal = false;
  @Input() fundraiser?: FundRaiser;
  @Input() disabled = false;
  @Output() closeModal = new EventEmitter<any>();

  @ViewChild('myDialog') myDialog!: { nativeElement: HTMLDialogElement };

  pageNumber = 1;
  imageFile = '';

  fundForm = this.fb.group({
    fundId: this.fb.nonNullable.control(-1),
    title: ['', [Validators.required, Validators.pattern(/[A-Za-z\s]{3,50}/)]],
    description: ['', [Validators.required]],
    amount: [0, [Validators.required, Validators.min(100)]],
    accountName: ['', [Validators.required, Validators.pattern(/[A-Za-z\s]{3,50}/)]],
    accountNumber: ['', [Validators.required, Validators.pattern(/[0-9]{10,10}/)]],
    bankName: ['', [Validators.required]],
```

```typescript
    });

  banks$ = this.fundService.getBanks();

  userId = +(localStorage.getItem('auth-id') ?? -1);
  user?: User;

  constructor(private fb: FormBuilder, private fundService: FundraisesService, private userService:
    AdminService) { }

  ngOnInit(): void {

    this.userService.getUsers().subscribe(res => {
     this.user = res.find(user => user.userId == this.userId);
    });
  }
  ngOnChanges(changes: SimpleChanges): void {
    if (changes['fundraiser']?.currentValue !== undefined) {
      this.fundraiser = changes['fundraiser'].currentValue as FundRaiser;
      this.fundForm.patchValue({
        fundId: this.fundraiser.fundId,
        title: this.fundraiser.title,
        description: this.fundraiser.description,
        amount: this.fundraiser.amount,
        accountName: this.fundraiser.accountName,
        accountNumber: this.fundraiser.accountNumber,
        bankName: this.fundraiser.bankName,
      });
      this.imageFile = this.fundraiser.imageUrl;
      this.open();
    }
  }
  open() {
    this.myDialog.nativeElement.showModal();
  }

  close() {
    this.myDialog.nativeElement.close();
  }

  setThumbNail(event: Event) {
    let target = event.target as HTMLInputElement;
```

```
    let file = target.files?.item(0);
    if (file) {
     this.fundService.uploadFile(file).subscribe((res: any) => {
      this.imageFile = res.url;
     });
    }
   }
  next(last = false) {
   if (last) {
    if (this.fundForm.valid && this.imageFile !== '') {
     this.fundService.addFundraiser(this.fundForm.value, this.imageFile).subscribe((res) => {
      this.closeModal.emit(res);
      this.fundForm.reset();
      this.close();
     });
    }
    else {
     alert("Enter correct Details!");
    }
   } else {
    if ([
     this.fundForm.get('title')?.invalid,
     this.fundForm.get('description')?.invalid,
     this.fundForm.get('amount')?.invalid].includes(true) ||
     this.imageFile === '') {
     alert("Fill details please!");
     return;
    }
    this.pageNumber++;
   }
  }
}
```

- **Create-fundraiser.component.css (Src\app\pages\create-fundraiser\create-fundraiser.component.css)**

```css
.header{
   @apply bg-rose-500 text-white;
}

.header > div{
   @apply w-full flex items-center justify-between px-5 py-3;
}
```

```css
.body{
  @apply px-5 py--3;
}

.btn-grp{
  @apply flex gap-5 justify-center mt-5;
}

.form-group{
  @apply w-5/12 flex flex-col gap-2 mb-5;
}

.form-group.description{
  @apply w-10/12;
}

.form-group input, .form-group textarea{
  @apply rounded focus-within:shadow ring-0 outline-0 shadow-transparent ;
}

input:focus, textarea:focus{
  @apply border-2 border-rose-500;
}
```

12. My Payments Page

- **My-payments.component.html (src\app\pages\my-payments\my-payments.component.html)**

```html
<app-base-header />
<div class="h-screen w-full pt-20">
  <div class="border-b border-gray-200 shadow p-5">
    <table class="divide-y divide-gray-300 mt-12 w-full">
      <thead class="bg-rose-500 text-left text-lg text-white">
        <tr>
          <th class="px-6 py-4 text-sm ">
            ID
          </th>
          <th class="px-6 py-4 text-sm ">
            Fundraise Name
          </th>
          <th class="px-6 py-4 text-sm ">
```

```html
            Amount
          </th>
          <th class="px-6 py-4 text-sm ">
            Payment Date
          </th>
          <th class="px-6 py-4 text-sm ">
            Status
          </th>
        </tr>
      </thead>
      <tbody class="bg-white divide-y divide-gray-300">
        @for (payment of payments$ | async; track $index) {
        <tr class="whitespace-nowrap">
          <td class="px-6 py-4 text-sm ">
            {{$index + 1}}
          </td>
          <td class="px-6 py-4">
            <div class="text-sm text-gray-900">
              <a routerLink="{{getFundLink(getFundRaise(payment.fundId).fundId)}}"
  class="hover:text-rose-500">
                {{ getFundRaise(payment.fundId).title }}
                <tub-icon icon="external-link" class="h-4 w-4 inline-block ml-1" />
              </a>
            </div>
          </td>
          <td class="px-6 py-4 text-sm ">
            {{ payment.amount }}
          </td>
          <td class="px-6 py-4 text-sm ">
            {{ payment.paymentDate | date: 'dd/MM/yyyy'}}
          </td>
          <td class="px-6 py-4">
            <span [class]="'rounded p-2 text-sm' + (payment.isVerified ? ' text-green-900 bg-green-300' : ' text-red-200 bg-red-700')">
              {{ payment.isVerified ? 'Verified' : 'Pending' }}
            </span>
          </td>
        </tr>}
      </tbody>
    </table>
  </div>
</div>
```

```
<app-base-footer />
```

● **My-payments.component.html (src\app\pages\my-payments\my-payments.component.ts)**

```typescript
import { Component,OnInit } from '@angular/core';
import { BaseHeaderComponent } from '../../partials/base-header/base-header.component';
import { BaseFooterComponent } from '../../partials/base-footer/base-footer.component';
import { FundraisesService } from '../../services/fundraises.service';
import { FundRaiser } from '../../../assets/data/fundraises';
import { AsyncPipe, DatePipe } from '@angular/common';
import { RouterModule } from '@angular/router';
import { FeatherIconsComponent } from '../../ui/tub-icon/feather-icons.component';
import { PaymentService } from '../../services/payment.service';
import { map } from 'rxjs';

@Component({
  selector: 'app-my-payments',
  standalone: true,
  imports: [
    BaseHeaderComponent,
    BaseFooterComponent,
    AsyncPipe,
    DatePipe,
    RouterModule,
    FeatherIconsComponent
  ],
  providers: [],
  templateUrl: './my-payments.component.html',
  styles: ''
})
export class MyPaymentsComponent implements OnInit{
  constructor(private payService: PaymentService, private fundServce: FundraisesService) { }

  myUserId = localStorage.getItem('auth-id') ? +localStorage.getItem('auth-id')! : undefined;
  payments$ = this.payService.getPayments(this.myUserId);

  fundRaises: FundRaiser[] = [];

  btoa = btoa;

  ngOnInit(): void {
    this.fundServce.getFundraises().subscribe((res) => this.fundRaises = res);
```

```
  }


  getFundRaise(id: number) {
   return this.fundRaises.find(f => f.fundId === id)!;
  }

  getFundLink(id: number) {
   return `/fundraiser/${btoa(id.toString())}`;
  }
}
```

13. Base Header (Header for all main pages)

● **Base-header.component.html (src\app\partials\base-header\base-header.component.html)**

```
<header class="text-gray-950 body-font bg-rose-50 fixed top-0 w-screen z-50">
  <div class="container mx-auto flex flex-wrap p-5 flex-col md:flex-row items-center">
    <a class="flex title-font font-medium items-center text-gray-900 mb-4 md:mb-0 w-20 md:w-40"
      [routerLink]="['/']">
      <img src="/assets/img/main-logo.png" alt="">
    </a>
    <nav
      class="md:mr-auto md:ml-4 md:py-1 md:pl-4 md:border-l md:border-gray-400   flex flex-wrap
  items-center text-base justify-center">
      <a class="mr-5 hover:text-gray-900" [routerLink]="['/']" routerLinkActive="active"
        [routerLinkActiveOptions]="{exact: true}">Home</a>
      <a class="mr-5 hover:text-gray-900" [routerLink]="['/about-us']"
  routerLinkActive="active">About</a>
      <a class="mr-5 hover:text-gray-900" [routerLink]="['/explore']"
  routerLinkActive="active">Explore</a>
      <a class="mr-5 hover:text-gray-900" [routerLink]="['/contact-us']"
  routerLinkActive="active">Contact</a>
    </nav>

    @if (unauthorized) {
    <tub-buttton icon="lock" text="Login" (click)="router.navigateByUrl('/auth/login')" />
    }
    @else if (isAdmin) {
    <tub-buttton icon="user" text="Account" (click)="router.navigateByUrl('account')" />
    }
    @else {
```

```html
<!-- profile dropdown with tailwind css using group focus functionality -->
@if (data) {
  <button class="group relative">
    <div class="flex items-center cursor-pointer gap-1">
      <img class="w-11 h-11 rounded-full" src="{{data.avatar}}" alt="avatar">
      <p class="text-xl">{{data.name.split(' ')[0]}}</p>
      <span class="text-3xl"> &triangledown;</span>
    </div>
    <div class="z-10 hidden group-focus-within:block absolute mt-2 w-48 bg-white rounded-lg shadow-xl -bottom-22 -left-5 text-left">
      <a class="block px-4 py-2 text-gray-800 hover:bg-gray-200 border-b" [routerLink]="['/my-fundraisers']">My Fundraisers</a>
      <a class="block px-4 py-2 text-gray-800 hover:bg-gray-200 border-b" [routerLink]="['/my-payments']">My Payments</a>
      <a routerLink="#" class="block px-4 py-2 text-gray-800 hover:bg-gray-200" (click)="logOut()">Logout</a>
    </div>
  </button>
}

}
  </div>
</header>
```

● **Base-header.component.html (src\app\partials\base-header\base-header.component.ts)**

```typescript
import { Component, OnInit } from '@angular/core';
import { Router, RouterModule } from '@angular/router';
import { TubButttonComponent } from '../../ui/tub-buttton/tub-buttton.component';
import { isAdmin, isLoggedIn, logOut } from '../../../helpers/auth';
import { AuthService } from '../../services/auth.service';
import { HttpClientModule } from '@angular/common/http';

@Component({
  selector: 'app-base-header',
  standalone: true,
  imports: [RouterModule, TubButttonComponent, HttpClientModule],
  providers: [AuthService],
  templateUrl: './base-header.component.html',
```

```
  styleUrl: './base-header.component.css'
})
export class BaseHeaderComponent implements OnInit {
 unauthorized = !isLoggedIn();
 isAdmin = isAdmin();
 logOut = logOut;
 data: { name: string, email: string, avatar: string } = { name: ', email: ', avatar: ' };

 constructor(protected router: Router, private authService: AuthService) { }

 ngOnInit(): void {
  if (!this.unauthorized && !this.isAdmin) {
   this.authService.getProfile().subscribe((data: { name: string, email: string }) => {
    this.data.name = data.name;
    this.data.email = data.email;
    this.data.avatar = 'https://ui-avatars.com/api/?background=random&name=' + data.name.split('
   ').join('+');
   });
  }
 }
}
```

- **Base-header.component.css (src\app\partials\base-header\base-header.component.css)**

```
.active{
   border-bottom: 2px solid #f88;
}

a {
   @apply text-xl;
}
```

14. Base Footer (Footer for all main pages)

- **Base-footer.component.html (src\app\partials\base-footer\base-footer.component.html)**

```
<footer>
   Copyright &copy; Taabir {{year}}
</footer>
```

- **Base-footer.component.ts (src\app\partials\base-footer\base-footer.component.ts)**

```
import { Component } from '@angular/core';
```

```
@Component({
  selector: 'app-base-footer',
  standalone: true,
  imports: [],
  templateUrl: './base-footer.component.html',
  styleUrl: './base-footer.component.css'
})
export class BaseFooterComponent {

  year = new Date().getFullYear();
}
```

- **src\app\partials\base-footer\base-footer.component.css**

```
footer{
    position: relative;
    width: 100%;
    font-weight: 600;
    text-align: center;
    padding: 0.5% 2%;
    background-color: #eee;
}
.top{
    top: 85vh;
}
}
```

15. Admin Dashboard (When Admin is Logged In)

- **color-pallete.ts (src\app\account\home\inner\stats\color-pallete.ts)**

```
// these variabes are for dynamic color for Admin Dashboard as tailind doesnt allow
// dynamic strings for class

export const borderColors = {
    red: 'border-red-200',
    green: 'border-green-200',
    blue: 'border-blue-200',
    yellow: 'border-yellow-200',
};

export const shadowColors = {
    red: 'bg-red-100',
    green: 'bg-green-100',
    blue: 'bg-blue-100',
    yellow: 'bg-yellow-100',
```

```
};

export const textColors = {
  red: 'text-red-600',
  green: 'text-green-600',
  blue: 'text-blue-600',
  yellow: 'text-yellow-600',
};
```

- **Stats.component.html (src\app\account\home\inner\stats\stats.component.html)**

```
<article [class]="borderColor">
  <div class="flex items-center justify-between">
    <div>
      <p [class]="labelColor">{{label}}</p>

      <p class="text-3xl font-medium text-gray-900">
        @if(type === statusType.MONEY){
          {{prefixForValue}}{{value}}
        }
        @else {
          {{value}}{{prefixForValue}}
        }
      </p>
    </div>

    <span [class]="shadowColor">
      {{icon}}
    </span>
  </div>
</article>
```

- **Stats.component.ts (src\app\account\home\inner\stats\stats.component.ts)**

```
import { Component, Input } from '@angular/core';
import { borderColors, shadowColors, textColors } from './color-pallete';

export enum StatsType {
  MONEY = 'money',
  TIME = 'time',
  PERCENTAGE = 'percentage',
```

```
  NUMBER = 'number'
}

@Component({
  selector: 'app-stats',
  standalone: true,
  imports: [],
  templateUrl: './stats.component.html',
  styles: ''
})
export class StatsComponent {

  @Input() type: StatsType = StatsType.NUMBER;
  @Input() value: number = 10;
  @Input() label: string = 'Label';
  @Input() icon: string = '🏵';
  @Input() color: string = 'sky';

  bColor = `border-${this.color}-200`;
  statusType = StatsType;

  get prefixForValue() : string {
    switch(this.type) {
      case StatsType.MONEY: return '₹';
      case StatsType.TIME: return 'Hrs.';
      case StatsType.PERCENTAGE: return '%';
      case StatsType.NUMBER: return '+';
    }
  }
  get borderColor() : string {
    return `rounded-lg border-4  bg-white p-6 ${borderColors[(this.color as any).toLowerCase() as keyof
      typeof borderColors]}`;
  }
  get shadowColor() : string {
    return `rounded-full w-24 h-24 grid place-items-center text-4xl ${shadowColors[(this.color as
      any).toLowerCase() as keyof typeof shadowColors]} ${textColors[(this.color as
      any).toLowerCase() as keyof typeof textColors]}`
  }
  get labelColor() : string {
    return `text-md ${textColors[(this.color as any).toLowerCase() as keyof typeof textColors]} `
  }
}
```

- **home.component.html (src\app\account\home\home.component.html)**

```html
@if(totalAmountPaid && totalFundsRaiserCreated && totalUsers)
{
<!-- Main Dasboard Page-->
  <h3 class="text-4xl uppercase font-semibold text-center mt-8 mb-8 underline">Data So Far</h3>
  <div class="grid grid-cols-2 place-items-center gap-5 w-full">
    <app-stats class="w-[60vw] md:w-[30vw]" [value]="totalUsers" [type]="statusType.NUMBER"
  color="yellow" label="Total Users Joined" icon="👤" />
    <app-stats class="w-[60vw] md:w-[30vw]" [value]="totalFundsRaiserCreated"
  [type]="statusType.NUMBER" color="blue" label="Total Fundraises created" icon="🤩" />
    <app-stats class="w-[60vw] md:w-[30vw]" [value]="totalPayments"
  [type]="statusType.NUMBER" color="green" label="Total Payments Initiated" icon="💳" />
    <app-stats class="w-[60vw] md:w-[30vw]" [value]="totalAmountPaid"
  [type]="statusType.MONEY" color="red" label="Total Contributions" icon="💰" />
  </div>
}
```

- **home.component.ts (src\app\account\home\home.component.ts)**

```typescript
import { Component, OnInit } from '@angular/core';
import { StatsComponent, StatsType } from './inner/stats/stats.component';
import { AdminService } from '../../services/admin.service';
import { tap, zip } from 'rxjs';

@Component({
 selector: 'app-home',
 standalone: true,
 imports: [
  StatsComponent
 ],
 templateUrl: './home.component.html',
 styles: "
})
export class HomeComponent implements OnInit{

 statusType = StatsType;
 constructor(private service: AdminService){}

 totalAmountPaid = 0;
```

```
totalPayments = 0;
totalUsers = 0;
totalFundsRaiserCreated = 0;

ngOnInit(): void {
  // All APIs to call at once and get values in all variables
  zip(this.service.getPayments(), this.service.getUsers(), this.service.getFunds()).pipe(
    tap(result => {
      this.totalAmountPaid = result[0].reduce((previousValue, paymentObject) => previousValue +
paymentObject.amount, 0);
      this.totalPayments = result[0].length;
      this.totalUsers = result[1].length;
      this.totalFundsRaiserCreated = result[2].length;
    })
  ).subscribe();
}
}
```

16. **Admin Fundraisers Verify Page(When Admin is Logged In)**

- **Admin-funds.component.html (src\app\account\pages\admin-funds\admin-funds.component.html)**

```html
<div class="h-screen mt-32 px-10 py-12">
  <div class="text-center">
    <h2 class="text-3xl font-semibold uppercase underline">All Fundraisers</h2>
  </div>
  @if (fundraises$ | async; as fundraises) {
  @if (fundraises.length) {
    <table class="funds-table">
      <thead class="bg-rose-500 text-white">
        <tr>
          <th>ID</th>
          <th>Title</th>
          <th>Amount</th>
          <th>Created By</th>
          <th>Created At</th>
          <th>Active</th>
        </tr>
      </thead>
      <tbody>
        @for (fundraiser of fundraises; track $index) {
        <tr>
```

```html
            <td>{{$index + 1}}</td>
            <td>{{fundraiser.title}}</td>
            <td>₹ {{fundraiser.amount}}</td>
            <td>{{getUser(fundraiser.fundId).name}}</td>
            <td>{{fundraiser.createdAt | date:'dd-MM-YYYY' }}</td>
            <td>
               <input type="checkbox" [checked]="fundraiser.isActive"
                  (change)="toggleFundActive(fundraiser, $any($event).target.checked)">
            </td>
          </tr>
          }
        </tbody>
      </table>
    }
    @else {
    <h2 class="text-2xl text-center text-gray-500 mt-10">No Fundraisers Found</h2>
    }
    }
</div>
```

- **admin-funds.component.ts (src\app\account\pages\admin-funds\admin-funds.component.ts)**

```typescript
import { Component } from '@angular/core';
import { AdminService } from '../../../services/admin.service';
import { AsyncPipe, DatePipe } from '@angular/common';
import { User } from '../../../../assets/data/users';
import { FundRaiser } from '../../../../assets/data/fundraises';
import { switchMap } from 'rxjs';

@Component({
  selector: 'app-admin-funds',
  standalone: true,
  imports: [AsyncPipe,
    DatePipe],
  templateUrl: './admin-funds.component.html',
  styleUrl: './admin-funds.component.css'
})
export class AdminFundsComponent {

  fundraises$ = this.service.getFunds();
  users: User[] = [];
```

```
  constructor(protected service: AdminService) { }

  ngOnInit(): void {
    this.service.getUsers().subscribe((res) => this.users = res as User[]);
  }

  getUser(id: number) {
    return this.users.find(u => u.userId === id) || this.users[0];
  }
  toggleFundActive(fund: FundRaiser, active: any) {
    this.service.toggleFundActive(fund, active as boolean).pipe(switchMap(() =>
      this.service.getUsers())).subscribe(
      res => this.users = res
    );
  }

}
```

- **Admin-funds.component.css (src\app\account\pages\admin-funds\admin-funds.component.css)**

```
.funds-table{
    @apply w-full text-left border border-rose-500 m-8 mx-auto rounded-md overflow-hidden;
}
td, th{
    padding: 4px 7px;
}

td {
    @apply bg-rose-50;
}
```

17. **Admin Payments Verify (When Admin is Logged In to verify payments)**
- **Payments.component.html (src\app\account\pages\payments\payments.component.html)**

```
<div class="h-screen w-full pt-20">
    <div class="rounded overflow-hidden p-5">
        <div class="mb-8 text-2xl uppercase font-semibold text-center">All Payments</div>
        <table class="divide-y divide-gray-300 mt-12 w-full shadow">
```

```html
<thead class="bg-rose-500 text-left text-lg text-white">
  <tr>
    <th class="px-6 py-4 text-sm ">
      ID
    </th>
    <th class="px-6 py-4 text-sm ">
      Fundraise Name
    </th>
    <th class="px-6 py-4 text-sm ">
      Paid By
    </th>
    <th class="px-6 py-4 text-sm ">
      Amount
    </th>
    <th class="px-6 py-4 text-sm ">
      Payment Date
    </th>
    <th class="px-6 py-4 text-sm ">
      Status
    </th>
  </tr>
</thead>
<tbody class="bg-white divide-y divide-gray-300">
  @for (payment of payments$ | async; track $index) {
  <tr class="whitespace-nowrap">
    <td class="px-6 py-4 text-sm ">
      {{$index + 1}}
    </td>
    <td class="px-6 py-4">
      <div class="text-sm text-gray-900">
          {{ getFundRaise(payment.fundId).title }}
      </div>
    </td>
    <td class="px-6 py-4">
      <div class="text-sm text-gray-900">
          {{ payment.isGuest ? 'Guest' :  getUser(payment.userId).name }}
      </div>
    </td>
    <td class="px-6 py-4 text-sm ">
      {{ payment.amount }}
    </td>
    <td class="px-6 py-4 text-sm ">
```

```
                    {{ payment.paymentDate | date: 'dd/MM/yyyy'}}
               </td>
               <td class="px-6 py-4">
                  <input type="checkbox" [checked]="payment.isVerified"
    (change)="togglePayment(payment, $any($event).target.checked)">
               </td>
           </tr>}
        </tbody>
     </table>
   </div>
</div>
```

- **payments.component.ts (src\app\account\pages\payments\payments.component.ts)**

```
import { AsyncPipe, DatePipe } from '@angular/common';
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';
import { FeatherIconsComponent } from '../../../ui/tub-icon/feather-icons.component';
import { FundRaiser, Payment } from '../../../../assets/data/fundraises';
import { User } from '../../../../assets/data/users';
import { AdminService } from '../../../services/admin.service';

@Component({
  selector: 'app-payments',
  standalone: true,
  imports: [AsyncPipe,
    DatePipe,
    RouterModule,
    FeatherIconsComponent],
  templateUrl: './payments.component.html',
})
export class PaymentsComponent {

  constructor(private service: AdminService) { }
  payments$ = this.service.getPayments();
  fundRaises: FundRaiser[] = [];
  users: User[] = [];
  btoa = btoa;
  ngOnInit(): void {
    this.service.getFunds().subscribe((res) => this.fundRaises = res);
    this.service.getUsers().subscribe((res) => this.users = res);
```

```
  }

  getFundRaise(id: number) {
   return this.fundRaises.find(f => f.fundId === id)!;
  }

  getUser(id: number) {
   return this.users.find(u => u.userId === id)!;
  }

  togglePayment(pay: Payment, checked: boolean) {
   this.service.togglePayActive(pay, checked).subscribe(
    () => this.payments$ = this.service.getPayments()
   );
  }
}
```

18. **Admin Users Verify (When Admin is Logged In to verify users)**

● **users.component.html (src\app\account\pages\users\users.component.html)**

```html
<div class="m-10">
  <h1 class="text-3xl font-bold text-gray-900">Users</h1>

  <table class="divide-y divide-gray-300 mt-12 w-full">
    <thead class="bg-rose-500 text-left text-lg text-white">
      <tr>
        <th class="px-6 py-4 text-sm ">
          ID
        </th>
        <th class="px-6 py-4 text-sm ">
          User Name
        </th>
        <th class="px-6 py-4 text-sm ">
          Email
        </th>
        <th class="px-6 py-4 text-sm ">
          User Type
        </th>
```

```html
      <th class="px-6 py-4 text-sm ">
        Mobile
      </th>
      <th class="px-6 py-4 text-sm ">
        PAN Card
      </th>
      <th class="px-6 py-4 text-sm ">
        Website
      </th>
      <th class="px-6 py-4 text-sm ">
        Active
      </th>
    </tr>
  </thead>
  <tbody class="bg-white divide-y divide-gray-300">
    @for (user of users$ | async; track $index) {
    <tr class="whitespace-nowrap">
      <td class="px-6 py-4 text-sm ">
        {{$index + 1}}
      </td>
      <td class="px-6 py-4 text-sm ">
        {{ user.name }}
      </td>
      <td class="px-6 py-4 text-sm ">
        {{ user.email}}
      </td>
      <td class="px-6 py-4 text-sm ">
        {{ user.userType == 2 ? "NGO" : "Individual"}}
      </td>
      <td class="px-6 py-4">
        {{user.mobile}}
      </td>
      <td class="px-6 py-4">
        {{user.panCard.replaceAll(user.panCard.substring(2, 8), '******')}}
      </td>
      <td class="px-6 py-4">
        @if (user.website) {
          <a class="text-rose-500 hover:underline" [href]="user.website.includes('https') ?
user.website: 'https://' + user.website" target="_blank">{{user.website}}</a>
        }
        @else {
          <span>Not Applicable</span>
```

```html
          }
        </td>
        <td>
          <input type="checkbox" [checked]="user.isVerified"
            (change)="toggleUserActive(user, $any($event).target.checked)">
        </td>
      </tr>}
    </tbody>
  </table>
</div>
```

- **users.component.ts (src\app\account\pages\users\users.component.ts)**

```typescript
import { Component } from '@angular/core';
import { AdminService } from '../../../services/admin.service';
import { AsyncPipe } from '@angular/common';
import { User } from '../../../../assets/data/users';

@Component({
  selector: 'app-users',
  standalone: true,
  imports: [
    AsyncPipe
  ],
  templateUrl: './users.component.html',
})
export class UsersComponent {

  users$ = this.adminService.getUsers();
  constructor(private adminService: AdminService) { }

  toggleUserActive(user: User, active: boolean) {
    this.adminService.toggleUserActive(user, active).subscribe(
      _ => {
        this.users$ = this.adminService.getUsers();
      }
    );
  }

}
```

19. Admin FAQ Creation (When Admin is Logged In)

- **Admin-faqs.component.html (src\app\account\pages\admin-faqs\admin-faqs.component.html)**

```html
<div class=" h-screen mt-5 px-10 py-12">
  <div class="flex justify-between">
    <h2 class="text-3xl font-semibold uppercase mb-5 ">Frequently Asked Questiond</h2>
    <button (click)="open()"
      class=" text-white bg-rose-500 border-0 py-2 px-6 focus:outline-none hover:bg-rose-600
  rounded text-lg float-right">
      Create FAQ
    </button>
  </div>

  <dialog #myDialog class="w-1/3 mx-auto rounded shadow-white">
    <div class="header border-b">
      <div>
        <h2 class="text-2xl font-semibold uppercase">Create a FAQ</h2>
        <span class="text-3xl font-bold cursor-pointer" (click)="close()">&Cross;</span>
      </div>
    </div>
    <div class="body">
      <form [formGroup]="faqForm">
        <div class="flex flex-col gap-4 mx-auto justify-center py-5">
          <div class="form-group w-full">
            <label for="title">Title</label>
            <input type="text" formControlName="title" id="title" placeholder="title"
              class="form-control" />
          </div>
          <div class="form-group description">
            <label for="description">Description</label>
            <textarea formControlName="description" id="description" class="form-control"
              placeholder="Description"></textarea>
          </div>
          <div class="flex gap-2 items-center">
            <input type="checkbox" formControlName="active" id="active" class="form-control"
  />
            <label for="active">Active</label>
          </div>
        </div>
        <div class="button-group">
          <button type="submit" (click)="save()"
```

```
                  class=" text-white bg-rose-500 border-0 py-2 px-6 focus:outline-none hover:bg-rose-600
          rounded text-lg w-56">
                      {{faqForm.controls.id.value! > 0 ? 'Save' : 'Add'}}
                </button>
            </div>
          </form>
      </div>
  </dialog>


  @if (faqs$ | async; as faqs) {
      @if (faqs.length) {
          <table class="funds-table w-full mx-auto mt-10">
              <thead class="bg-rose-500 text-white">
                  <tr>
                      <th>ID</th>
                      <th>Question</th>
                      <th>Answer</th>
                      <th>Actions</th>
                      <th>Active</th>
                  </tr>
              </thead>
              <tbody>
                  @for (faq of faqs; track $index) {
                  <tr>
                      <td>{{$index + 1}}</td>
                      <td>{{faq.title}}</td>
                      <td>{{faq.description}}</td>
                      <td>
                          <button class="bg-blue-500 text-white px-4 py-2 rounded mr-1"
  (click)="editFaq(faq)">Edit</button>
                      </td>
                      <td>
                          <input type="checkbox" [checked]="faq.active"
                              (change)="toggleActive(faq, $any($event).target.checked)">
                      </td>
                  </tr>
                  }
              </tbody>
          </table>
      }
      @else {
```

```html
        <h2 class="text-2xl text-center text-gray-500 mt-10">No FAQs Found</h2>
      }
    }
</div>
```

- **admin-faqs.component.ts (src\app\account\pages\admin-faqs\admin-faqs.component.ts)**

```typescript
import { AsyncPipe, DatePipe } from '@angular/common';
import { Component, ViewChild } from '@angular/core';
import { FAQ,  FaqService } from '../../../services/faq.service';
import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';

@Component({
  selector: 'app-admin-faqs',
  standalone: true,
  imports: [ AsyncPipe,
    ReactiveFormsModule],
  templateUrl: './admin-faqs.component.html',
  styleUrl: './admin-faqs.component.css'
})
export class AdminFaqsComponent {


  faqs$ = this.service.getFaq();

  faqForm = this.fb.group({
    id: this.fb.control(-1),
    title: this.fb.control('', Validators.required),
    description: this.fb.control('', Validators.required),
    active: this.fb.control(false)
  });

  @ViewChild('myDialog') myDialog!: { nativeElement: HTMLDialogElement };
  constructor(protected service: FaqService, private fb: FormBuilder) {}

  save() {
    this.service.saveFaq({...this.faqForm.value} as FAQ).subscribe(
      res => {
        this.faqs$ = this.service.getFaq();
        this.faqForm.reset();
        this.close();
```

```
      }
    );
  }

  open() {
    this.myDialog.nativeElement.showModal();
  }

  editFaq(faq: FAQ) {

    this.faqForm.controls.id.setValue(faq.id);
    this.faqForm.controls.active.setValue(faq.active);
    this.faqForm.controls.title.setValue(faq.title);
    this.faqForm.controls.description.setValue(faq.description);
    this.myDialog.nativeElement.showModal();
  }

  close() {
    this.myDialog.nativeElement.close();
  }

  toggleActive(faq: FAQ, checked: boolean){
    this.service.saveFaq({...faq, active: checked} as FAQ).subscribe(
      res => {
        this.faqs$ = this.service.getFaq();
        this.faqForm.reset();
        this.close();
      }
    );
  }

}
```

● **Admin-faqs.component.css (src\app\account\pages\admin-faqs\admin-faqs.component.css)**

```
td, th {
  @apply p-3 ;
}
th {
  @apply text-left;
}

td{
```

```css
  @apply bg-gray-100 border-b;
}
.header{
  @apply bg-rose-500 text-white;

}

.header > div{
  @apply w-full flex items-center justify-between px-5 py-3;
}
.body{
  @apply px-5 py-3;
}

.btn-grp{
  @apply flex gap-5 justify-center mt-5;
}

.form-group{
  @apply flex flex-col gap-2 mb-5;
}

.form-group input, .form-group textarea{
  @apply rounded focus-within:shadow ring-0 outline-0 shadow-transparent ;
}

input:focus, textarea:focus{
  @apply border-2 border-rose-500;
}
```

20. Admin Sidebar (Admin Dashboard)

● **Sidebar.component.html (src\app\account\sidebar\sidebar.component.html)**

```html
<div id="sidebar" class="group w-full h-full p-4">

  <ul class="list-reset mt-10">
    <li class="my-2 md:my-0">
      <a routerLink="/account"
        class="flex items-center py-1 md:py-3 pl-1 align-middle text-gray-950 no-underline
   hover:text-rose-400">
        🏠<span class="link-text group-hover:inline-block ">Home</span>
```

```
          </a>
        </li>
      <li class="my-2 md:my-0 ">
        <a routerLink="/account/faq"
           class="block py-1 md:py-3 pl-1 align-middle text-gray-950 no-underline hover:text-rose-
    400">
          !?<span class="link-text group-hover:inline-block ">FAQs</span>
        </a>
      </li>
      <li class="my-2 md:my-0 ">
        <a routerLink="/account/fundraisers"
           class="block py-1 md:py-3 pl-1 align-middle text-gray-950 no-underline hover:text-rose-
    400">
          🙌<span class="link-text group-hover:inline-block ">fundraisers</span>
        </a>
      </li>
      <li class="my-2 md:my-0">
        <a routerLink="/account/users"
           class="block py-1 md:py-3 pl-1 align-middle text-gray-950 no-underline hover:text-rose-
    400">
          👤<span class="link-text group-hover:inline-block ">Users</span>
        </a>
      </li>
      <li class="my-2 md:my-0">
        <a routerLink="/account/payments"
           class="block py-1 md:py-3 pl-1 align-middle text-gray-950 no-underline hover:text-rose-
    400">
          🤑<span class="link-text group-hover:inline-block ">Payments</span>
        </a>
      </li>
    </ul>

</div>
```

- **sidebar.component.ts (src\app\account\sidebar\sidebar.component.ts)**

```
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-sidebar',
  standalone: true,
```

```
  imports: [RouterModule],
  templateUrl: './sidebar.component.html',
  styleUrl: './sidebar.component.css'
})
export class SidebarComponent {

}
```

- **sidebar.component.css (src\app\account\sidebar\sidebar.component.css)**

```
.link-text {
   @apply hidden ml-3 pb-1 md:pb-0 text-sm;
}

li a{
   @apply text-gray-300 hover:text-rose-300;
}

.link-text {
   @apply pb-1 md:pb-0 text-sm;
}
```

21. Admin Main Page (Main blok of Admin module)

- **Main.component.html (src\app\account\main\main.component.html)**

```
<div class="dashboard">
<!-- Included admin sidebar module -->
   <app-sidebar></app-sidebar>

   <section class="main-container">
      <!-- include dashboard header -->
      <app-dashboard-header></app-dashboard-header>
      <div class="full-container">
         <!-- include Routes handling module -->
         <router-outlet ></router-outlet>
      </div>
   </section>
</div>
```

- **Main.component.ts (src\app\account\main\main.component.ts)**

```typescript
import { Component } from '@angular/core';
import { SidebarComponent } from '../sidebar/sidebar.component';
import { DashboardHeaderComponent } from '../dashboard-header/dashboard-header.component';
import { RouterModule } from '@angular/router';
import { PgComponent } from '../../ui/pg/pg.component';

@Component({
  selector: 'app-main',
  standalone: true,
  imports: [
    SidebarComponent,
    DashboardHeaderComponent,
    RouterModule,
    PgComponent
  ],
  templateUrl: './main.component.html',
  styleUrl: './main.component.css'
})
export class MainComponent {}
```

- **Main.component.css (src\app\account\main\main.component.css)**

```css
.dashboard{
   @apply h-screen w-full flex;
}

app-sidebar{
   @apply h-screen w--16 hover:w-48 transition-all bg-slate-950 text-white flex shadow;
}

.main-container{
   @apply h-screen w-full overflow-y-hidden;
}

.full-container{
   height: calc(100vh - 10rem);
   overflow-y: scroll;
   overflow-x: hidden;
}
```

22. Admin Dashboard Header (When Admin is Logged In)

- **Dashboard-header.component.html (src\app\account\dashboard-header\dashboard-header.component.html)**

```html
<div class="h-28 lg:h-28 w-full flex flex-wrap">
  <nav id="header1"
    class="bg-slate-800 text-white flex justify-between w-auto flex-1 border-b- order-1 lg:order-2 h-full px-5">
    <img src="../../../assets/img/logo_dark.png" class="w-32 white-image object-cover" alt="">
    <div class="flex h-full justify-between items-center">

      <!--Menu-->

      <div class="flex relative pr-6 items-center gap-10">

        <div class="relative text-lg">
          <span class="hidden md:inline-block">🤚 Hi, Admin </span>
        </div>
        <tub-buttton text="Log Out" icon="log-out" (click)="logout()" />
      </div>
    </div>

  </nav>
</div>
```

- **dashboard-header.component.ts (src\app\account\dashboard-header\dashboard-header.component.ts)**

```typescript
import { Component } from '@angular/core';
import { FeatherIconsComponent } from '../../ui/tub-icon/feather-icons.component';
import { TubButttonComponent } from '../../ui/tub-buttton/tub-buttton.component';
import { logOut } from '../../../helpers/auth';

@Component({
  selector: 'app-dashboard-header',
  standalone: true,
  imports: [
    FeatherIconsComponent,
    TubButttonComponent
  ],
  templateUrl: './dashboard-header.component.html',
})
```

```
export class DashboardHeaderComponent {

  logout = logOut;
}
```

23. Services (Angular Files to hit HTTP request)

- **admin.service.ts (src\app\services\admin.service.ts)**

```
import { HttpClient } from '@angular/common/http';
import { Injectable, inject } from '@angular/core';
import { User } from '../../assets/data/users';
import { tap } from 'rxjs';
import { FundRaiser, Payment } from '../../assets/data/fundraises';
import { FundraisesService } from './fundraises.service';
import { baseUrl } from '../../helpers/url-helper';
import { toSnakeCaseBody } from '../../helpers/case-helper';

@Injectable({
 providedIn: 'root'
})
export class AdminService {

  constructor(private http: HttpClient) { }

  fundService = inject(FundraisesService);

  getUsers() {
   return this.http.get<User[]>(baseUrl('users'));
  }

  getPayments() {
   return this.http.get<Payment[]>(baseUrl('payment'));
  }

  getFunds(){
   return this.fundService.getFundraises();
  }


  toggleUserActive(user: User, active: boolean) {
    user.isVerified = active;
    return this.http.post<User>(baseUrl('users'), user);
```

```
    }
  toggleFundActive(fund: FundRaiser, active: boolean) {
   return this.fundService.addFundraiser({
     title: fund.title,
     description: fund.description,
     amount: fund.amount,
     accountName: fund.accountName,
     accountNumber: fund.accountNumber,
     bankName: fund.bankName,
     active: active,
     fundId: fund.fundId,
   }, fund.imageUrl);
  }

  togglePayActive(pay: Payment, active: boolean){
   pay.isVerified = active;
   return this.http.post<Payment>(baseUrl('payment'), toSnakeCaseBody(pay));
  }

}
```

● **auth.service.ts (src\app\services\auth.service.ts)**

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable, tap } from 'rxjs';
import { baseUrl } from '../../helpers/url-helper';

@Injectable({
 providedIn: 'root'
})
export class AuthService {
  constructor(private http: HttpClient) { }

  login(value: Partial<{ email: string; password: string; }>): Observable<{ message: string, token:
    string, isAdmin?: boolean, userId?: number }> {
   return this.http.post<{ message: string, token: string, isAdmin?: boolean
     }>('http://localhost:8080/taabir/login', value).pipe(
     tap((res) => {
      localStorage.setItem('auth-token', res.token);
      if (res.isAdmin) {
       localStorage.setItem('role', 'admin');
       } else {
```

```
        localStorage.setItem('auth-id', res.userId!.toString());
      }

    })
  );
}

register(value: any): Observable<{ message: string, token: string, user_id?: number }> {
  return this.http.post<{ message: string, token: string }>('http://localhost:8080/taabir/register',
    value).pipe(
    tap((res) => {
      if (res.token) {
        localStorage.setItem('auth-token', res.token);
        localStorage.setItem('auth-id', res.user_id!.toString());
      }
    })
  );
}

getProfile(): Observable<{ name: string, email: string }> {
  return this.http.put<{ name: string, email: string }>(baseUrl('users'), {
    user_id: localStorage.getItem('auth-id') ? +localStorage.getItem('auth-id')! : -1
  });
}
}
}
```

- **faq.service.ts (src\app\services\faq.service.ts)**

```
import { UploadClient, uploadFile } from '@uploadcare/upload-client'
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { from, map, of } from 'rxjs';
import { FundRaiser } from '../../assets/data/fundraises';
import { baseUrl } from '../../helpers/url-helper';

@Injectable({
 providedIn: 'root'
})
export class FundraisesService {

  constructor(private http: HttpClient) { }
```

```typescript
addFundraiser(value: Partial<{ fundId?: number, title: string | null; description: string | null; amount:
    number | null; accountName: string | null; accountNumber: string | null; bankName: string | null;
    active?: boolean}>, imageFile: string) {
  let fundraise: FundRaiser = {
    fundId: value?.fundId ?? -1,
    title: value.title!,
    description: value.description!,
    amount: value.amount!,
    accountName: value.accountName!,
    accountNumber: value.accountNumber!,
    bankName: value.bankName!,
    createdAt: new Date().toISOString().split('T')[0],
    imageUrl: imageFile!,
    isActive: true,
    userId: +localStorage.getItem('auth-id')!,
  };
  return this.http.post<FundRaiser[]>(baseUrl('funds'), fundraise);
}
getFundraises(user_id?: number) {
  let request = this.http.get<FundRaiser[]>(baseUrl('funds'));
  return user_id ? request.pipe(map(data => data.filter(d => d.userId! === user_id))) : request;
}

deleteFundraise(id: number) {
  return this.http.delete(baseUrl('funds'), {
    params: {
      fund_id: id
    }
  });
}

getBanks() {
  return this.http.get('/assets/data/banks.json');
}

uploadFile(file: File | Blob) {
  const client = new UploadClient({ publicKey: 'e9c5890593531058b67f', store: 'auto' });
  return from(client.uploadFile(file)).pipe(map((res) => ({ url: `https://ucarecdn.com/${res.uuid}/`
    })));
}

getAmountNeededForFundRaiser(fundId: number){
```

```
    return this.http.put<{paidAmount: number}>(baseUrl('funds'), {
      fund_id: fundId
    });
  }
}
```

- **payment.service.ts (src\app\services\payment.service.ts)**

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Payment } from '../../assets/data/fundraises';
import { baseUrl } from '../../helpers/url-helper';
import { toSnakeCaseBody, toSnakeCaseRequest } from '../../helpers/case-helper';
import { map } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class PaymentService {
  constructor(private http: HttpClient) { }

  pay(pay: Payment) {
    return this.http.post(baseUrl('payment'), toSnakeCaseBody(pay));
  }

  getPayments(user_id?: number) {
    let payments = this.http.get<Payment[]>(baseUrl('payment'));
    return user_id ? payments.pipe(map(res => res.filter(r => r.userId == user_id))) : payments;
  }
}
```

- **fundraises.service.ts (src\app\services\fundraises.service.ts)**

```
import { UploadClient, uploadFile } from '@uploadcare/upload-client'
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { from, map, of } from 'rxjs';
import { FundRaiser } from '../../assets/data/fundraises';
import { baseUrl } from '../../helpers/url-helper';

@Injectable({
  providedIn: 'root'
})
export class FundraisesService {
```

```typescript
constructor(private http: HttpClient) { }
addFundraiser(value: Partial<{ fundId?: number, title: string | null; description: string | null; amount:
  number | null; accountName: string | null; accountNumber: string | null; bankName: string | null;
  active?: boolean}>, imageFile: string) {
 let fundraise: FundRaiser = {
   fundId: value?.fundId ?? -1,
   title: value.title!,
   description: value.description!,
   amount: value.amount!,
   accountName: value.accountName!,
   accountNumber: value.accountNumber!,
   bankName: value.bankName!,
   createdAt: new Date().toISOString().split('T')[0],
   imageUrl: imageFile!,
   isActive: true,
   userId: +localStorage.getItem('auth-id')!,
 };
 return this.http.post<FundRaiser[]>(baseUrl('funds'), fundraise);
}
getFundraises(user_id?: number) {
 let request = this.http.get<FundRaiser[]>(baseUrl('funds'));
 return user_id ? request.pipe(map(data => data.filter(d => d.userId! === user_id))) : request;
}

deleteFundraise(id: number) {
 return this.http.delete(baseUrl('funds'), {
   params: {
     fund_id: id
   }
 });
}

getBanks() {
 return this.http.get('/assets/data/banks.json');
}

uploadFile(file: File | Blob) {
 const client = new UploadClient({ publicKey: 'e9c5890593531058b67f', store: 'auto' });
 return from(client.uploadFile(file)).pipe(map((res) => ({ url: `https://ucarecdn.com/${res.uuid}/`
   })));
}
```

```
getAmountNeededForFundRaiser(fundId: number){
  return this.http.put<{paidAmount: number}>(baseUrl('funds'), {
    fund_id: fundId
  });
  }
}
```

24. Guards (Files to protect routes)

- **admin.guard.ts (src\app\guard\admin.guard.ts)**

```
import { CanActivateFn, Router } from '@angular/router';
import { isAdmin, isLoggedIn } from '../../helpers/auth';
import { inject } from '@angular/core';

// Guard to check if user is admin
export const adminGuard: CanActivateFn = (route, state) => {

  let router = inject(Router);

  if(isAdmin()){
    return true;
  }
  else{
    router.navigate(['not-found']);
    return false;
  }
};

// Guard to check if user is logged in
export const loggedInGuard: CanActivateFn = (route, state) => {

  let router = inject(Router);

  if(isLoggedIn()){
    return true;
  }
  else{
    router.navigate(['not-found']);
    return false;
  }
};
```

```
// Guard to check if user is logged out
export const loggedOutGuard: CanActivateFn = (route, state) => {
 let router = inject(Router);
 if (isLoggedIn()) {
  router.navigate(['/']);
  return false;
 }
 else{
  return true;
 }
}
```

25. Route Files (For frontend Angular)

● **app.routes.ts (src\app\app.routes.ts)**

```
import { Routes } from '@angular/router';
import { HomeComponent } from './pages/home/home.component';
import { NotFoundComponent } from './pages/not-found/not-found.component';
import { MainComponent as AccountComponent } from './account/main/main.component';
import { ExploreComponent } from './pages/explore/explore.component';
import { SingleFundraiserComponent } from './pages/fundraiser/single-fundraiser.component';
import { ContactComponent } from './pages/contact/contact.component';
import { AboutComponent } from './pages/about/about.component';
import { MyFundraisersComponent } from './pages/fundraisers/my-fundraisers.component';
import { MyPaymentsComponent } from './pages/my-payments/my-payments.component';
import { adminGuard, loggedInGuard } from './guard/admin.guard';

export const routes: Routes = [
   { path: '', pathMatch: 'full', component: HomeComponent },
   { path: 'explore', pathMatch: 'full', component: ExploreComponent },
   { path: 'contact-us', pathMatch: 'full', component: ContactComponent },
   { path: 'about-us', pathMatch: 'full', component: AboutComponent },
   { path: 'my-fundraisers', pathMatch: 'full', component: MyFundraisersComponent , canActivate:
     [loggedInGuard]},
   { path: 'fundraiser/:fid', pathMatch: 'full', component: SingleFundraiserComponent },
   { path: 'my-payments', pathMatch: 'full', component: MyPaymentsComponent, canActivate:
     [loggedInGuard]},
   { path: 'auth', pathMatch: 'prefix', loadChildren: () => import('./auth/auth.module').then(m =>
     m.AuthModule) },
```

```ts
  { path: 'account', component: AccountComponent, pathMatch: 'prefix', loadChildren: () =>
    import('./account/account.module').then(m => m.AccountModule), canActivate: [adminGuard] },
  { path: '**', pathMatch: 'full', component: NotFoundComponent },
];
```

- **auth-routing.module.ts (src\app\auth\auth-routing.module.ts)**

```ts
import { RouterModule, Routes } from "@angular/router";
import { RegisterComponent } from "./register/register.component";
import { LoginComponent } from "./login/login.component";
import { NgModule } from "@angular/core";
import { loggedOutGuard } from "../guard/admin.guard";

const routes: Routes = [
  {
    path: 'login',
    component: LoginComponent,
    canActivate: [loggedOutGuard]
  },
  {
    path: '',
    pathMatch: 'prefix',
    redirectTo: 'login'
  },
  {
    path: 'register',
    component: RegisterComponent,
    canActivate: [loggedOutGuard]
  },

];

@NgModule({
  imports: [
    LoginComponent,
    RegisterComponent,
    RouterModule.forChild(routes)],
  exports: [
    LoginComponent,
    RegisterComponent
  ],
})
```

```
export class AuthRoutingModule {
}
```

- **account.routes.ts (src\app\account\account.routes.ts)**

```
import { Routes } from "@angular/router";
import { HomeComponent as AccountHomeComponent } from "./home/home.component";
import { PaymentsComponent } from "./pages/payments/payments.component";
import { UsersComponent } from "./pages/users/users.component";
import { AdminFaqsComponent } from "./pages/admin-faqs/admin-faqs.component";
import { AdminFundsComponent } from "./pages/admin-funds/admin-funds.component";

export const routes: Routes = [
   { path: '', component: AccountHomeComponent },
   { path: 'payments', component: PaymentsComponent},
   { path: 'faq', component: AdminFaqsComponent},
   { path: 'users', component: UsersComponent},
   { path: 'fundraisers', component: AdminFundsComponent},
];
```

26. Helpers & Data Models (Required for extra functionality)

- **auth.ts (src\helpers\auth.ts)**

```
export function isLoggedIn() {
   return localStorage.getItem('auth-token') !== null;
}

export function isAdmin() {
   if (isLoggedIn()) {
      return localStorage.getItem('role') !== null; }
   else
      return false;
}

export function logOut() {
   localStorage.removeItem('auth-token');
   localStorage.removeItem('role');
   localStorage.removeItem('auth-id');
   location.href = '/';
}
```

- **case-helper.ts (src\helpers\case-helper.ts)**

```typescript
import { HttpRequest } from '@angular/common/http';

export function toSnakeCaseRequest(req: HttpRequest<any>): HttpRequest<any> {

  if (!req.body) {
    return req;
  }

  const snakeCaseBody = convertKeysToSnakeCase(req.body);

  return req.clone({ body: snakeCaseBody });
}

export function toSnakeCaseBody(req: any): any {

  if (!req) {
    return req;
  }

  const snakeCaseBody = convertKeysToSnakeCase(req);

  return snakeCaseBody;
}

function convertKeysToSnakeCase(obj: any): any {
  if (typeof obj !== 'object' || obj === null) {
    return obj;
  }
  return Array.isArray(obj)
    ? obj.map(convertKeysToSnakeCase) // Recursively convert array elements
    : Object.entries(obj).reduce(
      (acc, [key, value]) => ({
        ...acc,
        [key.replace(/\B([A-Z])/, '_$1').toLowerCase()]: convertKeysToSnakeCase(
          value
        ),
      }),
      {}
    );
}
```

```ts
export function convertToCamelCaseResponse(obj: any): any {
  if (typeof obj !== 'object' || obj === null) {
    return obj;
  }

  return Array.isArray(obj)
    ? obj.map(convertToCamelCaseResponse) // Recursively convert array elements
    : Object.entries(obj).reduce(
      (acc, [key, value]) => ({
        ...acc,
        [key.replace(/([_][a-z])/g, (_, char) => char.toUpperCase()).replaceAll('_', '')]:
        convertToCamelCaseResponse(
          value
        ),
      }),
      {}
    );
}
```

- **url-helper.ts (src\helpers\url-helper.ts)**

```ts
export const baseUrl = (uri?: string) => `http://localhost:8080/taabir/${uri ?? ''}`;
```

- **fundraises.ts (src\app\data\fundraises.ts)**

```ts
export enum PaymentMode {
  UPI = 1,
  Card = 2,
  Wallet = 3,
}
export interface Payment {
  paymentId: number;
  fundId: number;
  userId: number;
  paymentMode: number;
  isGuest: boolean;
  isVerified: boolean;
  paymentReferenceNumber: string
  paymentCreds: string;
```

```typescript
    amount: number;
    paymentDate?: string;
}

export interface FundRaiser {
    fundId: number;
    title: string;
    description: string;
    imageUrl: string;
    amount: number;
    isActive: boolean;
    createdAt: string;
    accountName: string;
    accountNumber: string;
    bankName: string;
    userId: number;
}
```

- **users.ts (src\app\data\users.ts)**

```typescript
export enum UserType{
    INDIVIDUAL = 1,
    NGO = 2
}

export function userTypeValue(user: UserType): string{
    switch(user){
        case UserType.INDIVIDUAL:
            return 'Individual';
        case UserType.NGO:
            return 'NGO';
        default:
            return '';
    }
}

export interface User {
    userId: number;
    name: string;
    email: string;
    mobile: string;
    permissions?: string;
    panCard: string;
```

```
password?: string;
userType?: UserType;
ngoId?: string;
website?: string;
isVerified: boolean;
}
```

27. Extra (Custom Components)

● **Tub-button.component.html (src\app\ui\tub-buttton\tub-buttton.component.html)**

```html
<button
  [class]="'btn ' + btnType + extraClasses">
  @if (icon !== undefined) {
    <tub-icon [icon]="icon"></tub-icon>
  }
  {{text}}</button>
```

● **Tub-button.component.ts (src\app\ui\tub-buttton\tub-buttton.component.ts)**

```typescript
import { Component, Input } from '@angular/core';
import { FeatherIconsComponent } from '../tub-icon/feather-icons.component';

type ButtonTheme = 'primary' | 'light';
type ButtonType = 'submit' | 'button';

@Component({
  selector: 'tub-buttton',
  standalone: true,
  imports: [
    FeatherIconsComponent
  ],
  templateUrl: './tub-buttton.component.html',
  styleUrl: './tub-buttton.component.css'
})
export class TubButttonComponent {

  @Input() type: ButtonType = 'button';
  @Input() theme: ButtonTheme = 'primary';
  @Input() text: string = 'Click';
  @Input() icon?: string;
```

```
  @Input() class?: string;
  @Input() navigate?: string;

  get extraClasses(): string {
   return this.class ? ` ${this.class}` : '';
  }

  get btnType(): string {
   return this.theme === 'primary'? 'btn-primary' : 'btn-light';
  }
}
```

- **Tub-button.component.css (src\app\ui\tub-buttton\tub-buttton.component.css)**

```
.btn {
   @apply text-lg rounded transition-all hover:shadow hover:shadow-rose-100 flex items-center justify-
    center gap-1 px-4 py-2;
}

.btn-primary{
   @apply bg-primary-100 hover:bg-primary-50 text-white;
}

.btn-light{
   @apply bg-transparent text-rose-900 hover:text-rose-700;
}
```

- **feather-icons.component.html (src\app\ui\tub-icon\feather-icons.component.html)**

```
    <i class="{{class}}" [attr.data-feather]="icon"></i>
```

- **feather-icons.component.ts (src\app\ui\tub-icon\feather-icons.component.ts)**

```
import { Component, Input } from '@angular/core';
import * as feather from 'feather-icons';

@Component({
 selector: 'tub-icon',
 standalone: true,
```

```
  templateUrl: './feather-icons.component.html',
})

export class FeatherIconsComponent {

  @Input() icon: string | any;
  @Input() class: string | any;
  @Input() height: number = 16;
  @Input() width: number = 16;


  ngAfterViewInit() {
   feather.replace({
     height: this.height,
     width: this.width,
   });
  }
}
```

Backend Folder Structure

A folder structure is a directory file tree that is used to show where each file of a folder is located.
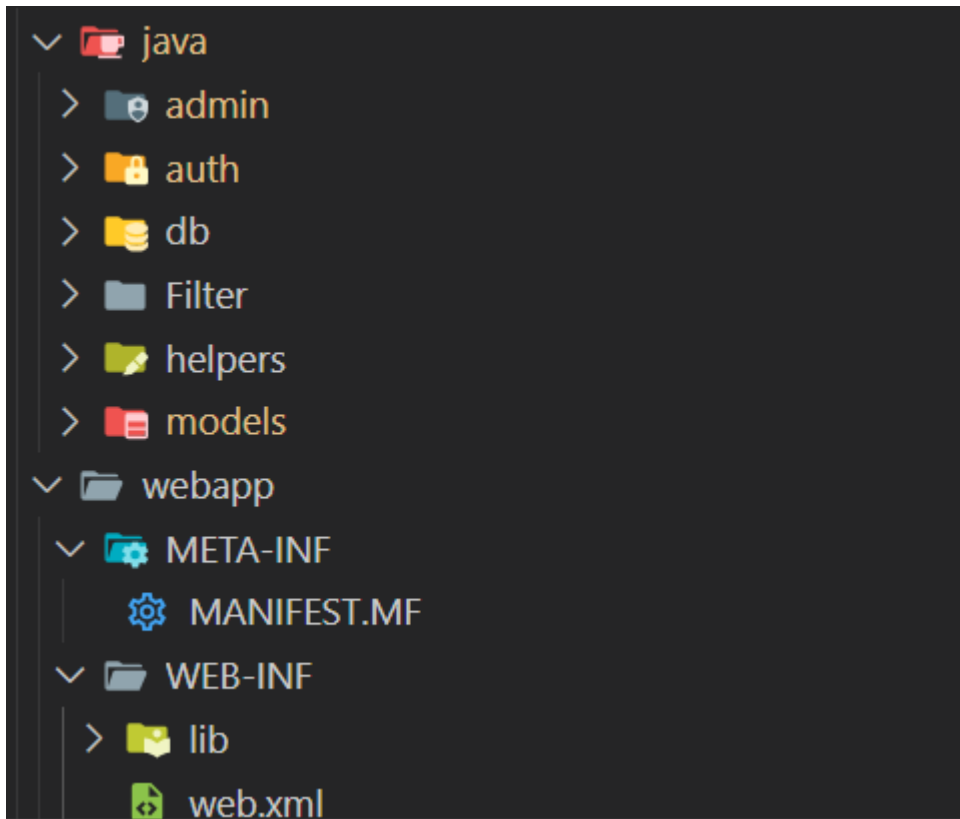
Folder structure for all Java Servlets Files

Backend Code

1. **Servlets**

    ○ **Users.java (api\java\admin\Users.java)**

    package admin;

    import jakarta.servlet.ServletException;
    import jakarta.servlet.annotation.WebServlet;
    import jakarta.servlet.http.HttpServlet;
    import jakarta.servlet.http.HttpServletRequest;
    import jakarta.servlet.http.HttpServletResponse;
    import models.PaymentModel;
    import models.UsersModel;

    import java.io.IOException;
    import java.io.PrintWriter;

```java
import java.sql.SQLException;

import org.json.simple.JSONObject;

import helpers.JsonHelper;

/**
 * Servlet implementation class Users
 */
public class Users extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Users() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            out.print(UsersModel.fetchAll());
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        UsersModel user = UsersModel.fromJSON(JsonHelper.getJSON(request));
        JSONObject jsonRes = new JSONObject();
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        try {
            boolean done = user.verifyUser();

            if (done) {
```

```java
            jsonRes.put("success", true);
            out.println(jsonRes);

        } else {
            jsonRes.put("success", false);
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            out.println(jsonRes);
        }
    } catch (Exception e) {

    }
}

protected void doPut(HttpServletRequest request, HttpServletResponse response) throws
  IOException {
    JSONObject req = JsonHelper.getJSON(request);
    int user_id = JsonHelper.getInt(req, "user_id");
    PrintWriter out = response.getWriter();

    JSONObject obj = new JSONObject();
    response.setContentType("application/json");

    try {
        obj = UsersModel.getProfile(user_id);
        if (obj != null) {
            obj.put("success", true);
            out.println(obj);
        } else {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            out.println(obj);
        }
    } catch (ClassNotFoundException | SQLException e) {
        obj.put("success", false);
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        out.println(obj);
    }

  }
}
```

○ **Payment.java (api\java\admin\Payment.java)**

```java
package admin;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import models.FaqModel;
import models.FundRaiserModel;
import models.PaymentModel;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;

import org.json.simple.JSONObject;

import helpers.JsonHelper;

public class Payment extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Payment() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            out.print(PaymentModel.fetchAll());
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```java
    throws ServletException, IOException {
  PaymentModel pm = PaymentModel.fromJSON(JsonHelper.getJSON(request));
  response.setContentType("application/json");
  PrintWriter out = response.getWriter();
  try {
    JSONObject jsonRes = new JSONObject();

    boolean done = false;
    if (pm.paymentId == -1) {
      HashMap<String, Integer> prePaidAmount =
  PaymentModel.getFundPaymentDetails(pm.fundId);
      if(prePaidAmount.get("paid_amount") + pm.amount ==
  prePaidAmount.get("total_amount")) {
        FundRaiserModel.complete(pm.fundId);
      }
      done = pm.insert();

    } else {
      done = pm.verifyPayment();
    }
    if (done) {
      jsonRes.put("success", true);
      out.println(jsonRes);

    } else {
      jsonRes.put("success", false);
      response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
      out.println(jsonRes);
    }
  } catch (ClassNotFoundException e) {
    e.printStackTrace();
  } catch (SQLException e) {
    e.printStackTrace();
  }
}

}
```

○ **Funds.java (api\java\admin\Funds.java)**

```java
package admin;

import jakarta.security.auth.message.callback.PrivateKeyCallback.Request;
```

```java
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import models.FundRaiserModel;
import models.PaymentModel;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;

import org.json.simple.JSONObject;

import helpers.JsonHelper;

public class Funds extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Funds() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            out.print(FundRaiserModel.fetchAll());
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        FundRaiserModel fund = FundRaiserModel.fromJSON(JsonHelper.getJSON(request));
        JSONObject jsonRes = new JSONObject();
        response.setContentType("application/json");
```

```java
        PrintWriter out = response.getWriter();
        try {
            boolean done = false;
            if (fund.fundId == -1) {
                done = fund.insert();
            } else {
                done = fund.update();
            }

            if (done) {
                jsonRes.put("success", true);
                out.println(jsonRes);

            } else {
                jsonRes.put("success", false);
                response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                out.println(jsonRes);
            }
        } catch (Exception e) {
            jsonRes.put("success", false);
            response.setStatus(HttpServletResponse.SC_BAD_GATEWAY);
            out.println(jsonRes);
        }
    }

    @Override
    protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws
      ServletException, IOException {
        int fundId = Integer.parseInt(req.getParameter("fund_id"));
        resp.setContentType("application/json");
        PrintWriter out = resp.getWriter();

        JSONObject json = new JSONObject();
        try {
            json.put("success", FundRaiserModel.delete(fundId));
        } catch (ClassNotFoundException | SQLException e) {
            // TODO Auto-generated catch block
            json.put("success", false);
            e.printStackTrace();
        }
        finally {
            out.print(json);
```

```java
        }
    }

    @Override
    protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws
      ServletException, IOException {
        int fundId = JsonHelper.getInt(JsonHelper.getJSON(req), "fund_id");
        resp.setContentType("application/json");
        PrintWriter out = resp.getWriter();
        try {
            JSONObject obj = new JSONObject(PaymentModel.getFundPaymentDetails(fundId));
            out.println(obj);
        } catch (ClassNotFoundException | SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
}
```

○ **Faq.java (api\java\admin\Faq.java)**

```java
package admin;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import models.FaqModel;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;

import org.json.simple.JSONObject;

import helpers.EncodeHelper;
import helpers.JsonHelper;

public class Faq extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

```java
/**
 * @see HttpServlet#HttpServlet()
 */
public Faq() {
    super();
    // TODO Auto-generated constructor stub
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("application/json");
    PrintWriter out = response.getWriter();

    try {
        out.print(FaqModel.fetchAll());
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
    }
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    FaqModel fq = FaqModel.fromJSON(JsonHelper.getJSON(request));
    JSONObject jsonRes = new JSONObject();
    response.setContentType("application/json");
    PrintWriter out = response.getWriter();
    try {
        boolean done = false;
        if (fq.faqId == -1) {
            done = fq.insert();
        } else {
            done = fq.update();
        }
        if (done) {
            jsonRes.put("success", true);
            out.println(jsonRes);

        } else {
            jsonRes.put("success", false);
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            out.println(jsonRes);
```

```
        }
    } catch (ClassNotFoundException | SQLException e) {
        jsonRes.put("success", false);
        response.setStatus(HttpServletResponse.SC_BAD_GATEWAY);
        out.println(jsonRes);
    }
  }
}
```

○ **Login.java (api\java\auth\Login.java)**

```
package auth;

import helpers.EncodeHelper;
import helpers.JsonHelper;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import models.Auth;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.UUID;

import org.json.simple.JSONObject;

public class Login extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Login() {
        super();
        // TODO Auto-generated constructor stub
    }

    @SuppressWarnings("unchecked")
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        JSONObject json = JsonHelper.getJSON(request);
        String pass;
        JSONObject jsonRes = new JSONObject();
```

```java
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        try {
          String user = JsonHelper.getString(json, "email");
          pass = JsonHelper.getString(json, "password");
          int id = Auth.loginExists(user, pass);
          if (id > 0) {
            jsonRes.put("success", true);
            jsonRes.put("message", "Login Successful");
            jsonRes.put("token", EncodeHelper.bin2hex(user));
            jsonRes.put("user_id", id);
            out.println(jsonRes);
          } else if (Auth.isAdmin(user, pass)) {
            jsonRes.put("success", true);
            jsonRes.put("message", "Login Successful");
            jsonRes.put("isAdmin", true);
            jsonRes.put("token", EncodeHelper.bin2hex(user));
            out.println(jsonRes);
          } else {
            jsonRes.put("success", false);
            jsonRes.put("message", "Login Failed!");
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            out.println(jsonRes);
          }
        } catch (Exception e) {
          jsonRes.put("success", false);
          jsonRes.put("message", "Server Error");
          response.setStatus(HttpServletResponse.SC_BAD_GATEWAY);
          out.println(jsonRes);
        }

    }

}
```

- ○ **Register.java (api\java\auth\Register.java)**

```java
package auth;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
```

```java
import models.Auth;
import models.UsersModel;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;

import org.json.simple.JSONObject;

import helpers.EncodeHelper;
import helpers.JsonHelper;

public class Register extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Register() {
        super();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        JSONObject json = JsonHelper.getJSON(request);
        JSONObject jsonRes = new JSONObject();

        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            UsersModel um = UsersModel.fromJSON(json);
            int id = um.insert();
            if (id > 0) {
                jsonRes.put("success", true);
                jsonRes.put("message", "Account Created Successfully!");
                jsonRes.put("token", EncodeHelper.bin2hex(um.email));
                jsonRes.put("user_id", id);
                out.println(jsonRes);
            } else {
                jsonRes.put("success", false);
                response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                out.println(jsonRes);
            }
        } catch (ClassNotFoundException | SQLException e) {
```

```java
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

2. **Database File (MySQL)**

   ○ **DatabaseConnection.java (api\java\db\DatabaseConnection.java)**

```java
package db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// This class can be used to initialize the database connection
public class DatabaseConnection {
    public static Connection initializeDatabase()
        throws SQLException, ClassNotFoundException
    {
        // Initialize all the information regarding
        // Database Connection
        String dbDriver = "com.mysql.jdbc.Driver";
        String dbURL = "jdbc:mysql://localhost:3306/";
        // Database name to access
        String dbName = "taabir";
        String dbUsername = "root";
        String dbPassword = "";

        Class.forName(dbDriver);
        Connection con = DriverManager.getConnection(dbURL + dbName,
                            dbUsername,
                            dbPassword);
        return con;
    }
}
```

3. **HTTP Filter**

   ○ **CorsFilter.java (api\java\Filter\CorsFilter.java)**

```java
package Filter;

import jakarta.servlet.Filter;

import jakarta.servlet.http.HttpFilter;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import jakarta.servlet.FilterChain;
import jakarta.servlet.FilterConfig;
import jakarta.servlet.ServletException;
import jakarta.servlet.ServletRequest;
import jakarta.servlet.ServletResponse;

public class CorsFilter extends HttpFilter implements Filter {

    /**
     * @see HttpFilter#HttpFilter()
     */
    public CorsFilter() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Filter#destroy()
     */
    public void destroy() {
        // TODO Auto-generated method stub
    }

    /**
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain
      chain) throws IOException, ServletException {

        HttpServletResponse res = (HttpServletResponse) response;
          res.setHeader("Access-Control-Allow-Origin", "*");
          res.setHeader("Access-Control-Allow-Methods", "POST, GET, PUT, OPTIONS,
    DELETE, PATCH");
```

```java
        res.setHeader("Access-Control-Allow-Headers", "Content-Type, X-Requested-
        With");

        chain.doFilter(request, response);
    }

    /**
     * @see Filter#init(FilterConfig)
     */
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }

}
```

4. **Helpers (Helper classes that are used to Assist in functionality)**

   ○ **EncodeHelper.java (java\helpers\EncodeHelper.java)**

```java
package helpers;
public class EncodeHelper {
    public static String bin2hex(String input) {
    // Convert string to hexadecimal encoded string
        StringBuilder hexString = new StringBuilder();
        for (char c : input.toCharArray()) {
            hexString.append(String.format("%02x", (int) c));
        }
        return hexString.toString();
    }

    public static String hex2bin(String hex) {
    // Convert hexadecimal encoded string back to normal string
        StringBuilder textString = new StringBuilder();
        for (int i = 0; i < hex.length(); i += 2) {
            String hexChar = hex.substring(i, i + 2);
            int decimal = Integer.parseInt(hexChar, 16);
            char character = (char) decimal;
            textString.append(character);
        }

        // Print the text string
        return textString.toString();

    }
```

}

○ **JsonHelper.java (java\helpers\JsonHelper.java)**

```java
package helpers;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Optional;

import jakarta.servlet.http.HttpServletRequest;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;


public class JsonHelper {
    public static JSONObject getJSON(HttpServletRequest request) {
        // Get request body as a string
        // This class is used in All the Servlet classes to fetch the JSON request as HttpServlet
      Class only supports FormBody
        try {
            BufferedReader reader = new BufferedReader(new
        InputStreamReader(request.getInputStream()));
            StringBuilder sb = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
            String jsonData = sb.toString();

            // Parse JSON data
            JSONParser parser = new JSONParser();
            JSONObject jsonObject = (JSONObject) parser.parse(jsonData);

            return jsonObject;

        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
```

```java
public static String getJSON(HttpServletRequest request, String key) {
    // Get request body as a string from a key
    try {
        BufferedReader reader = new BufferedReader(new
    InputStreamReader(request.getInputStream()));
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            sb.append(line);
        }
        String jsonData = sb.toString();

        // Parse JSON data
        JSONParser parser = new JSONParser();
        JSONObject jsonObject = (JSONObject) parser.parse(jsonData);



        return (String) jsonObject.get(key);
    } catch (Exception e) {
        e.printStackTrace();
        return "";
    }

    // Create and send response as needed...
}

public static String getString(JSONObject o, String key) {
    // Get String Value from JSON
    if (o.get(key) != null)
        return o.get(key).toString();
    else
        return "";
}

public static long getLong(JSONObject o, String key) {
    // Get Long Value from JSON

    if (o.get(key) != null)
        return Long.parseLong(o.get(key).toString());
    else
        return 0;
```

```java
    }

    public static int getInt(JSONObject o, String key) {
      // Get Integer Value from JSON

      if (o.get(key) != null)
        return Integer.parseInt(o.get(key).toString());
      else
        return 0;
    }

    public static float getFloat(JSONObject o, String key) {
      // Get Float Value from JSON

      if (o.get(key) != null)
        return Float.parseFloat(o.get(key).toString());
      else
        return 0;
    }

    public static boolean getBool(JSONObject o, String key) {
      // Get Boolean Value from JSON

      if (o.get(key) != null)
        return Boolean.parseBoolean(o.get(key).toString());
      else
        return false;
    }

}
```

5. **Models (All Classes used in Servlets for Database Operations)**
   - ○ **Auth.java (java\models\Auth.java)**

```java
package models;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import db.DatabaseConnection;
```

```java
public class Auth {

    public static int loginExists(String username, String password) throws SQLException,
    ClassNotFoundException {
        Connection con = DatabaseConnection.initializeDatabase();
        String sql = "SELECT user_id FROM users WHERE email = ? AND password = ?";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, username);
        pstmt.setString(2, password);
        ResultSet rs = pstmt.executeQuery();
        if (rs.next())
            {
            return rs.getInt(1);}
        else
            {return -1;}
    }

    public static boolean isAdmin(String email, String password) throws
    ClassNotFoundException, SQLException {
        Connection con = DatabaseConnection.initializeDatabase();
        String sql = "SELECT * FROM admin WHERE email = ? AND password = ?";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, email);
        pstmt.setString(2, password);

        ResultSet rs = pstmt.executeQuery();
        return rs.next();
    }
}
```

○ **FaqModel.java (java\models\FaqModel.java)**

```java
package models;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
```

```java
import db.DatabaseConnection;
import helpers.JsonHelper;

public class FaqModel {
    public int faqId;
    public String title, description;
    public boolean isActive;

    public static FaqModel fromJSON(JSONObject obj) {

        // Get JSON Object from HttpServletRequest Object
        FaqModel fq = new FaqModel();
        fq.faqId = JsonHelper.getInt(obj, "id");
        fq.title = JsonHelper.getString(obj, "title");
        fq.description = JsonHelper.getString(obj, "description");
        fq.isActive = JsonHelper.getBool(obj, "active");

        return fq;
    }

    public JSONObject toJSON() {
        // Convert FAQ Data JSOn Object
        JSONObject js = new JSONObject();
        js.put("faq_id", this.faqId);
        js.put("title", this.title);
        js.put("description", this.description);
        js.put("active", this.isActive);

        return js;
    }

    public boolean insert() throws SQLException, ClassNotFoundException {
        // Insert FAQ Data
        Connection con = DatabaseConnection.initializeDatabase();
        String sql = "INSERT INTO faq (title, description, is_active) VALUES (?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, this.title);
        pstmt.setString(2, this.description);
        pstmt.setBoolean(3, this.isActive);

        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
```

```java
    }

    public boolean update() throws SQLException, ClassNotFoundException {
        // Update FAQ Data

        Connection con = DatabaseConnection.initializeDatabase();
        String sql = "UPDATE faq SET title = ?, description = ? , is_active = ? WHERE faq_id =
    ?";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, this.title);
        pstmt.setString(2, this.description);
        pstmt.setBoolean(3, this.isActive);
        pstmt.setInt(4, this.faqId);

        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
    }

    public static JSONArray fetchAll() throws SQLException, ClassNotFoundException {
        // Fetch All FAQs Data and send it in Response

        Connection con = DatabaseConnection.initializeDatabase();
        JSONArray json = new JSONArray();
        ArrayList<FaqModel> faqs = new ArrayList<>();

        String query = "SELECT * FROM faq";
        PreparedStatement pstmt = con.prepareStatement(query);

        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            FaqModel fq = new FaqModel();

            fq.faqId = rs.getInt("faq_id");
            fq.title = rs.getString("title");
            fq.description = rs.getString("description");
            fq.isActive = rs.getBoolean("is_active");

            faqs.add(fq);
        }
        for (FaqModel fq : faqs) {
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("id", fq.faqId);
```

```java
        jsonObj.put("title", fq.title);
        jsonObj.put("active", fq.isActive);
        jsonObj.put("description", fq.description);

        json.add(jsonObj);
      }
      return json;
    }
  }
```

○ **FundRaiserModel.java (java\models\FundRaiserModel.java)**

```java
package models;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import com.mysql.cj.protocol.Resultset;

import db.DatabaseConnection;
import helpers.JsonHelper;

public class FundRaiserModel {

  public int fundId, userId;
  public float amount;
  public boolean isActive;
  public String title, description, accountName, accountNumber, bankName, imageUrl,
    createdAt;

  public static FundRaiserModel fromJSON(JSONObject obj) {
    // Get JSON Object from HttpServletRequest Object

    FundRaiserModel fund = new FundRaiserModel();
```

```java
        fund.fundId = JsonHelper.getInt(obj, "fund_id");
        fund.userId = JsonHelper.getInt(obj, "user_id");
        fund.amount = JsonHelper.getFloat(obj, "amount");
        fund.isActive = JsonHelper.getBool(obj, "is_active");
        fund.title = JsonHelper.getString(obj, "title");
        fund.description = JsonHelper.getString(obj, "description");
        fund.accountNumber = JsonHelper.getString(obj, "account_number");
        fund.bankName = JsonHelper.getString(obj, "bank_name");
        fund.accountName = JsonHelper.getString(obj, "account_name");
        fund.imageUrl = JsonHelper.getString(obj, "image_url");

        return fund;
    }

    public JSONObject toJSON() {
        // Convert Fundraiser Data JSON Object

        JSONObject js = new JSONObject();
        js.put("fund_id", this.fundId);
        js.put("description", this.description);
        js.put("amount", this.amount);
        js.put("is_active", this.isActive);
        js.put("title", this.title);
        js.put("user_id", this.userId);
        js.put("account_name", this.accountName);
        js.put("bank_name", this.bankName);
        js.put("account_number", this.accountNumber);
        js.put("image_url", this.imageUrl);
        js.put("createdAt", this.createdAt);

        return js;
    }

    public boolean insert() throws SQLException, ClassNotFoundException {
        // Insert Fundraiser Data

        Connection con = DatabaseConnection.initializeDatabase();
        String sql = "INSERT INTO fundraises (amount, is_active, title, description, user_id,
     bank_name, account_name, account_number, image_url) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setFloat(1, this.amount);
        pstmt.setBoolean(2, true);
```

```java
        pstmt.setString(3, this.title);
        pstmt.setString(4, this.description);
        pstmt.setInt(5, this.userId);
        pstmt.setString(6, this.bankName);
        pstmt.setString(7, this.accountName);
        pstmt.setString(8, this.accountNumber);
        pstmt.setString(9, this.imageUrl);

        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
    }

    public boolean update() throws SQLException, ClassNotFoundException {
        // Update Fundraiser Data

        Connection con = DatabaseConnection.initializeDatabase();
        if (this.userId != 0) {
            String sql = "UPDATE fundraises set user_id=?, amount=?, is_active=?, title=?,
description=?, user_id=?, bank_name=?, account_name=?, account_number=?, image_url=?
where fund_id=?";
            PreparedStatement pstmt = con.prepareStatement(sql);
            pstmt.setInt(11, this.fundId);
            pstmt.setInt(1, this.userId);
            pstmt.setFloat(2, this.amount);
            pstmt.setBoolean(3, this.isActive);
            pstmt.setString(4, this.title);
            pstmt.setString(5, this.description);
            pstmt.setInt(6, this.userId);
            pstmt.setString(7, this.bankName);
            pstmt.setString(8, this.accountName);
            pstmt.setString(9, this.accountNumber);
            pstmt.setString(10, this.imageUrl);
            int affectedRows = pstmt.executeUpdate();
            return affectedRows > 0;
        } else {
            String sql = "UPDATE fundraises set is_active=1, is_verified=1 where fund_id=?";
            PreparedStatement pstmt = con.prepareStatement(sql);
            pstmt.setInt(1, this.fundId);
            int affectedRows = pstmt.executeUpdate();
            return affectedRows > 0;
        }
    }
```

```java
public static JSONArray fetchAll() throws SQLException, ClassNotFoundException {
    // Fetch All Fundraisers Data and send it in Response

    Connection con = DatabaseConnection.initializeDatabase();
    JSONArray json = new JSONArray();

    String query = "SELECT * FROM fundraises";
    PreparedStatement pstmt = con.prepareStatement(query);

    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {
        FundRaiserModel fund = new FundRaiserModel();

        fund.fundId = rs.getInt("fund_id");
        fund.userId = rs.getInt("user_id");
        fund.amount = rs.getFloat("amount");
        fund.isActive = rs.getBoolean("is_active");
        fund.title = rs.getString("title");
        fund.description = rs.getString("description");
        fund.accountName = rs.getString("account_name");
        fund.accountNumber = rs.getString("account_number");
        fund.bankName = rs.getString("bank_name");
        fund.imageUrl = rs.getString("image_url");
        fund.createdAt = rs.getString("created_at");

        json.add(fund.toJSON());
    }

    return json;
}

public static boolean delete(int fundId) throws SQLException, ClassNotFoundException {
    String sql = "DELETE FROM fundraises where fund_id = ?";

    Connection con = DatabaseConnection.initializeDatabase();
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setInt(1, fundId);

    return ps.executeUpdate() > 0;
}
```

```java
    public static void complete(int fundId) throws SQLException, ClassNotFoundException {
        String sql = "UPDATE fundraises set is_active = 0 where fund_id = ?";

        Connection con = DatabaseConnection.initializeDatabase();
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, fundId);
        ps.executeUpdate();
    }
}
```

○ **GuestModel.java (java\models\GuestModel.java)**

```java
package models;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import db.DatabaseConnection;

public class GuestModel {

    public int guestId, fundId;
    public String guestReferenceNumber;
    public boolean isActive;
    public float amount;

    public boolean insert() throws SQLException, ClassNotFoundException {
        // Insert Data into database
        Connection con = DatabaseConnection.initializeDatabase();

        String sql = "INSERT INTO guests (amount, guest_reference_number, is_active, fund_id)
    VALUES (?, ?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setFloat(1, this.amount);
        pstmt.setString(2, this.guestReferenceNumber);
        pstmt.setBoolean(3, true);
        pstmt.setInt(4, this.fundId);
        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
    }

}
```

- ○ **PaymentModel.java (java\models\PaymentModel.java)**

```java
package models;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import db.DatabaseConnection;
import helpers.JsonHelper;

public class PaymentModel {
    public int paymentId, fundId, userId, paymentMode;
    boolean isGuest, isVerified;
    public float amount;
    String paymentDate, paymentReferenceNumber, paymentCreds;

    public static PaymentModel fromJSON(JSONObject obj) {

        // Create PaymentModel Object from JSON Object
        PaymentModel pm = new PaymentModel();
        pm.paymentId = JsonHelper.getInt(obj, "payment_id");
        pm.fundId = JsonHelper.getInt(obj, "fund_id");
        pm.userId = JsonHelper.getInt(obj, "user_id");
        pm.paymentMode = JsonHelper.getInt(obj, "payment_mode");

        pm.isGuest = JsonHelper.getBool(obj, "is_guest");
        pm.isVerified = JsonHelper.getBool(obj, "is_verified");

        pm.paymentReferenceNumber = JsonHelper.getString(obj, "payment_referencenumber");
        pm.paymentCreds = JsonHelper.getString(obj, "payment_creds");

        pm.amount = JsonHelper.getFloat(obj, "amount");

        return pm;
    }
```

```java
public JSONObject toJSON() {
    // Create JSON Object from PaymentModel Object

    JSONObject js = new JSONObject();
    js.put("payment_id", this.paymentId);
    js.put("amount", this.amount);
    js.put("fund_id", this.fundId);
    js.put("user_id", this.userId);
    js.put("payment_mode", this.paymentMode);
    js.put("is_guest", this.isGuest);
    js.put("is_verified", this.isVerified);
    js.put("payment_reference_number", this.paymentReferenceNumber);
    js.put("payment_creds", this.paymentCreds);
    js.put("payment_date", this.paymentDate);

    return js;
}

public boolean insert() throws SQLException, ClassNotFoundException {
    // Insert PaymentModel data into database

    Connection con = DatabaseConnection.initializeDatabase();
    String sql = "INSERT INTO PAYMENTS (fund_Id, user_id, payment_mode, is_guest,
 payment_reference_number, amount, payment_creds) VALUES (?, ?, ?, ?, ?, ?, ?)";

    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setInt(1, this.fundId);
    pstmt.setInt(2, this.userId);
    pstmt.setInt(3, this.paymentMode);
    pstmt.setBoolean(4, this.isGuest);
    pstmt.setString(5, this.paymentReferenceNumber);
    pstmt.setFloat(6, this.amount);
    pstmt.setString(7, this.paymentCreds);

    if (this.isGuest) {
        GuestModel gm = new GuestModel();
        gm.amount = this.amount;
        gm.guestReferenceNumber = this.paymentReferenceNumber;
        gm.fundId = this.fundId;
        gm.insert();
    }
```

```java
        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
    }

    public boolean verifyPayment() throws SQLException, ClassNotFoundException {
        // Verify payment by toggling active column
        Connection con = DatabaseConnection.initializeDatabase();

        String sql = "UPDATE PAYMENTS set is_verified=? where payment_id=?";

        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setBoolean(1, this.isVerified);
        pstmt.setInt(2, this.paymentId);

        int affectedRows = pstmt.executeUpdate();
        return affectedRows > 0;
    }

    public static JSONArray fetchAll() throws SQLException, ClassNotFoundException {
        // Read All PaymentModel data from database
        Connection con = DatabaseConnection.initializeDatabase();
        JSONArray json = new JSONArray();
        ArrayList<PaymentModel> payments = new ArrayList<>();

        String query = "SELECT * FROM payments";
        PreparedStatement pstmt = con.prepareStatement(query);

        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            PaymentModel pm = new PaymentModel();

            pm.paymentId = rs.getInt("payment_id");
            pm.amount = rs.getFloat("amount");
            pm.userId = rs.getInt("user_id");
            pm.fundId = rs.getInt("fund_id");
            pm.isGuest = rs.getBoolean("is_guest");
            pm.isVerified = rs.getBoolean("is_verified");
            pm.paymentReferenceNumber = rs.getString("payment_reference_number");
            pm.paymentMode = rs.getInt("payment_mode");
            pm.paymentCreds = rs.getString("payment_creds");
            pm.paymentDate = rs.getString("payment_date");
```

```java
        payments.add(pm);
      }

      for (PaymentModel pm : payments) {
        JSONObject jsonObj = pm.toJSON();

        json.add(jsonObj);
      }
      return json;
   }

   public static HashMap<String, Integer> getFundPaymentDetails(int fundId) throws
     SQLException, ClassNotFoundException {
      // Get all payments recieved for a particular fundraiser
      Connection con = DatabaseConnection.initializeDatabase();
      HashMap<String, Integer> paidMap = new HashMap<String, Integer>();
      paidMap.put("paid_amount", 0);

      PreparedStatement funstmt = con.prepareStatement("SELECT amount FROM fundraises
     WHERE fund_id = ?;");
      funstmt.setInt(1, fundId);

      PreparedStatement pstmt = con.prepareStatement("SELECT SUM(amount) AS
     total_payment FROM payments WHERE fund_id = ?;");
      pstmt.setInt(1, fundId);

      ResultSet rs = pstmt.executeQuery();
      ResultSet rs2 = funstmt.executeQuery();
      if(rs.next()) {
        paidMap.put("paid_amount", Math.round(rs.getFloat(1)));
      }
      rs2.next();
      paidMap.put("total_amount", Math.round(rs2.getFloat(1)));
      return paidMap;
   }
}
```

  ○ **UsersModel.java (java\models\UsersModel.java)**

```java
package models;

import java.sql.Connection;
```

```java
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import db.DatabaseConnection;
import helpers.JsonHelper;

public class UsersModel {
    public int userId, userType;
    public String name, email, panCard, password, ngoId, website;
    public long mobile;
    public boolean isVerified;

    public static UsersModel fromJSON(JSONObject obj) {
        UsersModel user = new UsersModel();
        user.userId = JsonHelper.getInt(obj, "user_id");
        user.name = JsonHelper.getString(obj, "name");
        user.email = JsonHelper.getString(obj, "email");
        user.panCard = JsonHelper.getString(obj, "pan_card");
        user.password = JsonHelper.getString(obj, "password");
        user.mobile = JsonHelper.getLong(obj, "mobile");
        user.isVerified = JsonHelper.getBool(obj, "is_verified");
        user.userType = JsonHelper.getInt(obj, "user_type");
        user.ngoId = JsonHelper.getString(obj, "ngo_id");
        user.website = JsonHelper.getString(obj, "website");

        return user;
    }

    public JSONObject toJSON() {
        JSONObject js = new JSONObject();
        js.put("user_id", this.userId);
        js.put("name", this.name);
        js.put("email", this.email);
        js.put("pan_card", this.panCard);
        js.put("password", this.password);
        js.put("mobile", this.mobile);
        js.put("is_verified", this.isVerified);
```

```java
      js.put("user_type", this.userType);
      js.put("ngo_id", this.ngoId);
      js.put("website", this.website);

      return js;
   }

   public int insert() throws SQLException, ClassNotFoundException {
      Connection con = DatabaseConnection.initializeDatabase();
      String sql = "INSERT INTO users (name, email, pan_card, password, mobile, user_type,
   ngo_id, website) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
      PreparedStatement pstmt = con.prepareStatement(sql,
   Statement.RETURN_GENERATED_KEYS);
      pstmt.setString(1, this.name);
      pstmt.setString(2, this.email);
      pstmt.setString(3, this.panCard);
      pstmt.setString(4, this.password);
      pstmt.setLong(5, this.mobile);
      pstmt.setInt(6, this.userType);

      if(this.userType == 2) {
         pstmt.setString(7, this.ngoId);
         pstmt.setString(8, this.website);
      }
      else {
         pstmt.setString(7, null);
         pstmt.setString(8, null);
      }
      int affectedRows = pstmt.executeUpdate();
      try (ResultSet generatedKeys = pstmt.getGeneratedKeys()) {
         if (generatedKeys.next()) {
            affectedRows = generatedKeys.getInt(1);
         }
      }
      return affectedRows;
   }

   public boolean verifyUser() throws SQLException, ClassNotFoundException {
      Connection con = DatabaseConnection.initializeDatabase();

      String sql = "UPDATE USERS set is_verified=? where user_id=?";
```

```java
      PreparedStatement pstmt = con.prepareStatement(sql);
      pstmt.setBoolean(1, this.isVerified);
      pstmt.setInt(2, this.userId);

      int affectedRows = pstmt.executeUpdate();
      return affectedRows > 0;
  }

  public static JSONArray fetchAll() throws SQLException, ClassNotFoundException {
      Connection con = DatabaseConnection.initializeDatabase();
      JSONArray json = new JSONArray();

      String query = "SELECT * FROM users";
      PreparedStatement pstmt = con.prepareStatement(query);

      ResultSet rs = pstmt.executeQuery();
      while (rs.next()) {
        UsersModel user = new UsersModel();

        user.userId = rs.getInt("user_id");
        user.name = rs.getString("name");
        user.email = rs.getString("email");
        user.panCard = rs.getString("pan_card");
        user.password = rs.getString("password");
        user.mobile = rs.getLong("mobile");
        user.isVerified = rs.getBoolean("is_verified");
        user.userType = rs.getInt("user_type");
        user.ngoId = rs.getString("ngo_id");
        user.website = rs.getString("website");

        json.add(user.toJSON());
      }

      return json;
  }

  public static JSONObject getProfile(int id) throws SQLException, ClassNotFoundException
   {
      Connection con = DatabaseConnection.initializeDatabase();
      String query = "SELECT name, email FROM users where user_id = ?";
      PreparedStatement pstmt = con.prepareStatement(query);
      pstmt.setInt(1, id);
```

```java
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                JSONObject obj = new JSONObject();
                obj.put("name", rs.getString("name"));
                obj.put("email", rs.getString("email"));
                return obj;
            }
            return null;
        }
    }
```

6. Web.XML (Main file that routes all Java servlets)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="https://jakarta.ee/xml/ns/jakartaee" xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
    https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd" id="WebApp_ID" version="6.0">
  <display-name>Taabir</display-name>
  <welcome-file-list>
   <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
   <description></description>
   <display-name>Login</display-name>
   <servlet-name>Login</servlet-name>
   <servlet-class>auth.Login</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>Login</servlet-name>
   <url-pattern>/login</url-pattern>
  </servlet-mapping>
  <servlet>
   <description></description>
   <display-name>Register</display-name>
   <servlet-name>Register</servlet-name>
   <servlet-class>auth.Register</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>Register</servlet-name>
   <url-pattern>/register</url-pattern>
  </servlet-mapping>
```

```xml
<servlet>
  <description></description>
  <display-name>Faq</display-name>
  <servlet-name>Faq</servlet-name>
  <servlet-class>admin.Faq</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Faq</servlet-name>
  <url-pattern>/faq</url-pattern>
</servlet-mapping>
<filter>
  <display-name>CorsFilter</display-name>
  <filter-name>CorsFilter</filter-name>
  <filter-class>Filter.CorsFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>CorsFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<servlet>
  <description></description>
  <display-name>Payment</display-name>
  <servlet-name>Payment</servlet-name>
  <servlet-class>admin.Payment</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Payment</servlet-name>
  <url-pattern>/payment</url-pattern>
</servlet-mapping>
<servlet>
  <description></description>
  <display-name>Users</display-name>
  <servlet-name>Users</servlet-name>
  <servlet-class>admin.Users</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Users</servlet-name>
  <url-pattern>/users</url-pattern>
</servlet-mapping>
<servlet>
  <description></description>
  <display-name>Funds</display-name>
```

```
   <servlet-name>Funds</servlet-name>
   <servlet-class>admin.Funds</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>Funds</servlet-name>
   <url-pattern>/funds</url-pattern>
  </servlet-mapping>
</web-app>
```

7. Index.jsp (Main JSP file)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Taabir Backend</title>
</head>
<body>

   <!--  Option to redirect to Angular App -->
   <a href="http://localhost:7777" target="_blank" rel="noopener noreferrer">
     Click Here to go to Angular Frontend
   </a>
</body>
</html>
```

USER INTERFACE DESIGN

1. Home Screen
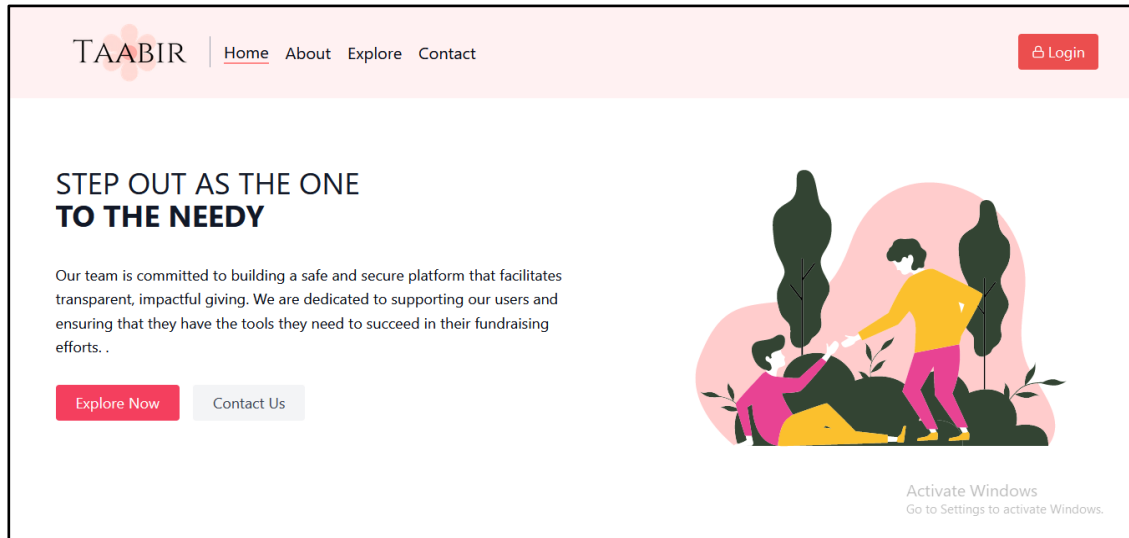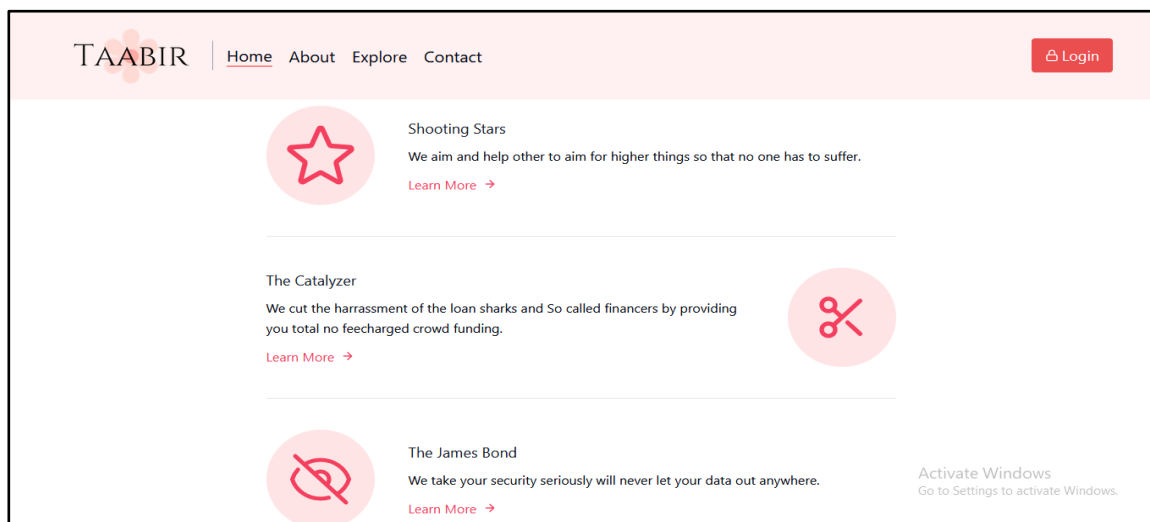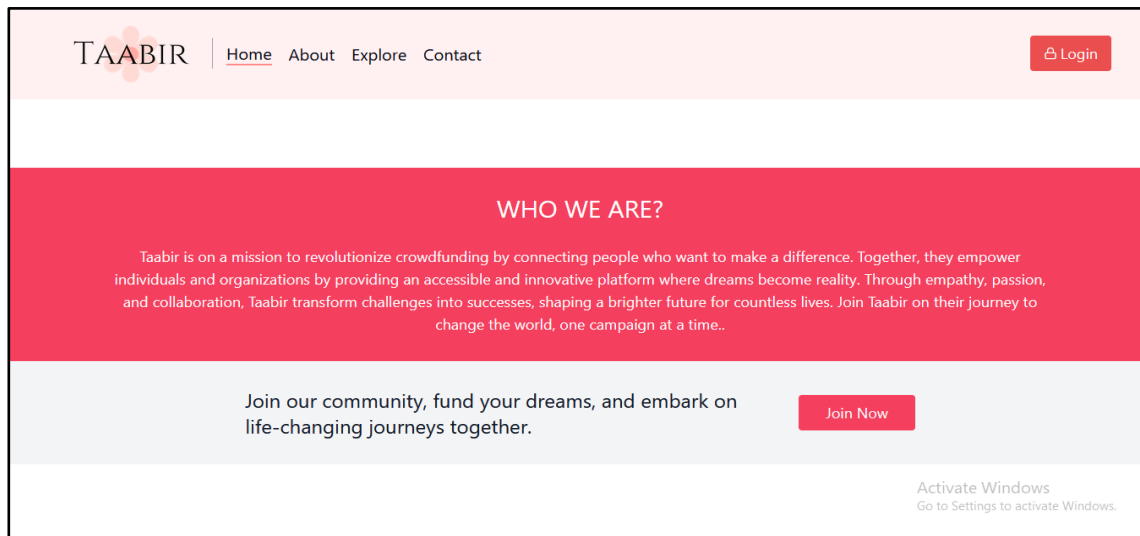
2. About Screen

About Us

Welcome to Taabir, a pioneering crowdfunding platform dedicated to transforming lives and fueling dreams. We connect compassionate individuals, groups, and organizations worldwide, empowering them to create meaningful change through collective action. Founded in 2022, Taabir has already helped countless individuals raise funds for various causes, including medical emergencies, education, disaster relief, and innovative projects. Our mission is to build a global community that uplifts those in need and cultivates a culture of empathy and generosity. At Taabir, we believe in the power of unity and collaboration. Our platform provides the tools and resources for people to come together and turn ideas into reality, creating a brighter future for all. We are committed to fostering a safe, transparent, and inclusive environment that encourages continuous growth and positive change. Join Taabir and become part of a movement that celebrates the potential of every person to make a difference. With your help, we can turn dreams into reality and build a better, more compassionate world for everyone..

Why Us?

Shooting Stars

We aim and help other to aim for higher things so that no one has to suffer.

Learn More →

The Catalyzer

We cut the harrassment of the loan sharks and So called financers by providing you total no feecharged crowd funding.

Learn More →

The James Bond

We take your security seriously will never let your data out anywhere.

Learn More →

3. Explore Screen
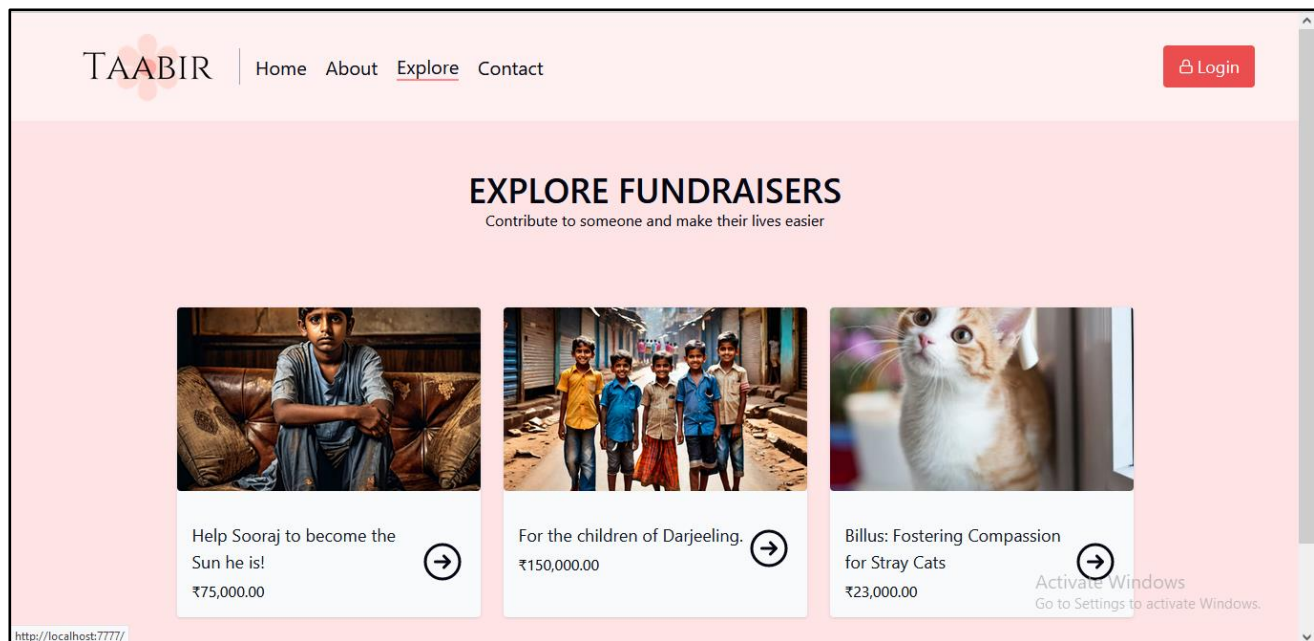
4. Single Fundraiser Screen
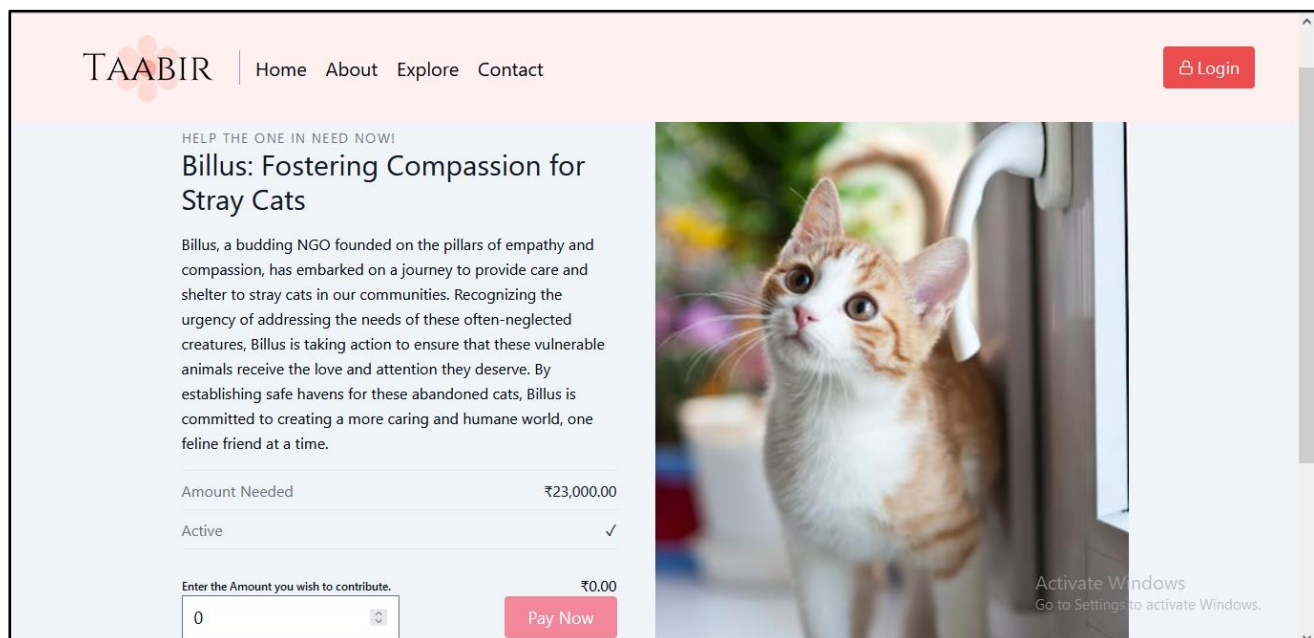
5. Payment Gateway Screen

6. Contact Screen

Contact Us
You can contact us through the following means:

- Email: taabir.project@proton.me
- Phone: +91 9912345678
- Address: IAEA HOUSE, 17-B, Ring Rd, IP Estate, New Delhi, Delhi, 110002

Copyright © Taabir 2024

7. Login Screen

TAABIR

Home About Explore Contact
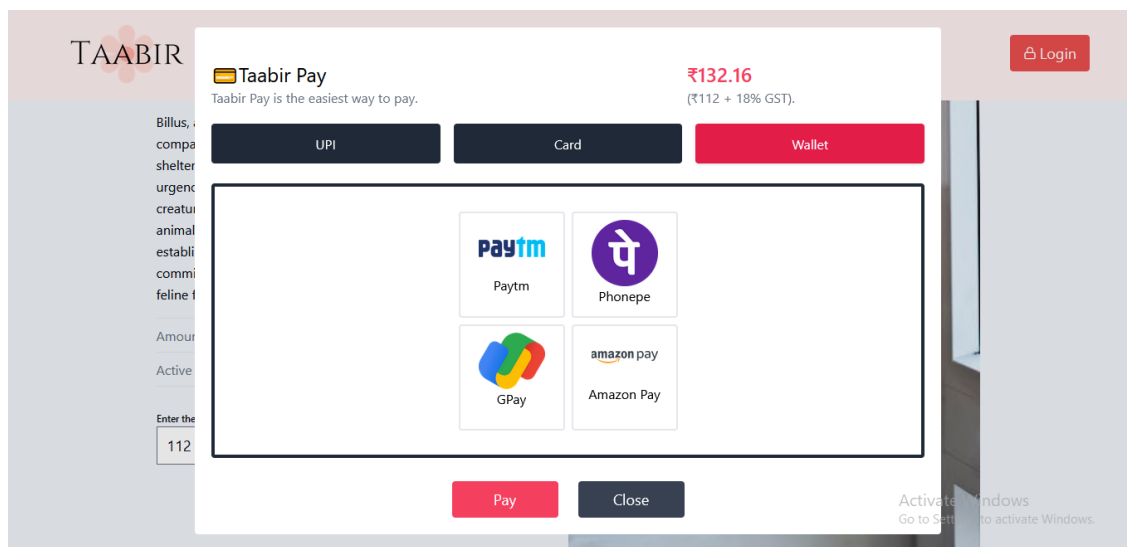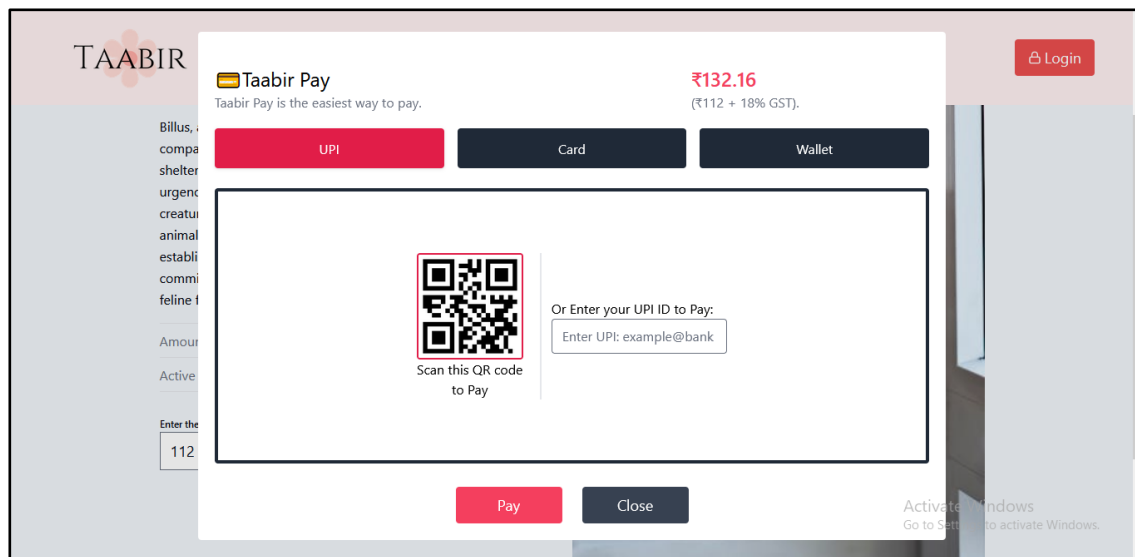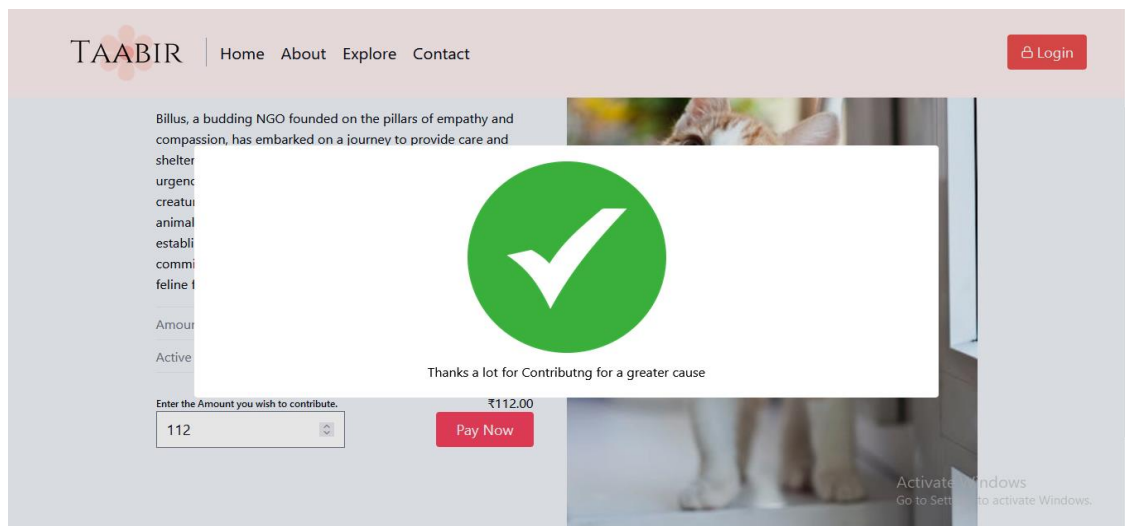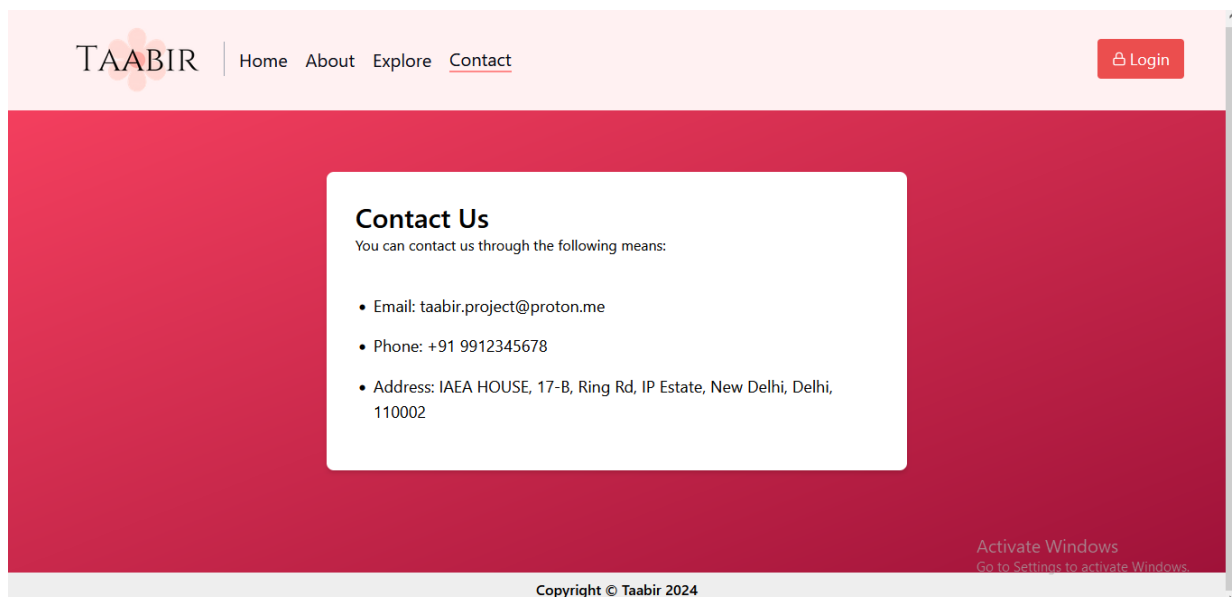
🔒 Login

Sign in to your Account

Email address

Password   Forgot password?

Sign in

Don't have an Account yet **Register Now!**

Activate Windows
Go to Settings to activate Windows.

8. Register Screen

Register Page Screens

9. My Fundraisers Screen

Figure | This screen shows all fundraisers created by the user

10.   Create Fundraiser Screen

First

Block of create fundraiser where we enter details like title, description thumbnail etc.

11. My Payments Screen

12. Admin Home Screen

Admin Dashboard where we can see all stats related to payment users etc.

13. Admin Fundraisers Verify Screen

ALL FUNDRAISERS

ID	Title	Amount	Created By	Created At	Active
1	Pee	₹ 203	noddy	18-05-2024	☐
2	Meenakshi Needs Your Attention!	₹ 52000	hello	12-05-2024	☐
3	Help Sooraj to become the Sun he is!	₹ 75000	Pepe Master	13-05-2024	☑
4	For the children of Darjeeling.	₹ 150000	Aloo Nas	18-05-2024	☑
5	Billus: Fostering Compassion for Stray Cats	₹ 23000	noddy	19-05-2024	☑
6	demo	₹ 12002	user five	26-05-2024	☑

Activate Windows
Go to Settings to activate Windows.

14. Admin Payments Verify Screen

ALL PAYMENTS

ID	Fundraise Name	Paid By	Amount	Payment Date	Status
1	Meenakshi Needs Your Attention!	Aloo Nas	5555	19/05/2024	☑
2	For the children of Darjeeling.	Guest	234	19/05/2024	☐
3	For the children of Darjeeling.	Guest	123	19/05/2024	☑
4	For the children of Darjeeling.	Guest	123	19/05/2024	☐
5	For the children of Darjeeling.	Guest	1500	19/05/2024	☑

Activate Windows
Go to Settings to activate Windows.

15. Admin Users Verify Screen

TAABIR

👋 Hi, Admin   [→ Log Out]

- 🏠 Home
- ⁉️ FAQs
- 🧑‍🤝‍🧑 fundraisers
- 👤 Users
- 🧩 Payments

Users

ID	User Name	Email	User Type	Mobile	PAN Card	Website	Active
1	noddy	hi.noddydev@gmail.com	Individual	7777777777	AB******8J	Not Applicable	☑️
2	hello	hello@byom.fe	Individual	8899889988	AB******7H	Not Applicable	☑️
3	Pepe Master	pepe@bom.bom	Individual	9898983983	AB******8J	Not Applicable	☑️
4	Aloo Nas	aloo11@gmail.com	Individual	9898983983	AB******7J	Not Applicable	☑️
5	user five	user567@byom.de	Individual	8989121233	AV******0C	Not Applicable	☐
6	aoo ui	info@ujik.com	Individual	9988998899	NM******8J	Not Applicable	☐

Activate Windows
Go to Settings to activate Windows.

16.   Admin FAQs Screen

All FAQs Screen - Where FAQs can be created and Edited and Toggled

17.   Create FAQ Screen

TEST CASES

Test Id	Test Scenario	Test Steps	Expected Results	Actual Result	Pass / Fail
T1	Open Home Page	Go to http://localhost:7777	On going on URL, home page will be displayed	As Expected	Pass
T2	Open About Page	Go to http://localhost:7777/about-us	On going on URL, about page will be displayed	As Expected	Pass
T3	Open Contact Page	Go to http://localhost:7777/contact-us	On going on URL, contact page will be displayed	As Expected	Pass
T4	Open Explore Page	Go to http://localhost:7777/explore	On going on URL, explore page will be displayed	As Expected	Pass
T5	Open Login Page	Go to http://localhost:7777/auth/login	On going on URL, login page will be displayed	As Expected	Pass
T6	Open Register Page	Go to http://localhost:7777/auth/register	On going on URL, register page will be displayed	As expected	Pass

Test Id	Test Scenario	Test Steps	Expected Results	Actual Result	Pass / Fail
T7	User Login	1. Go to http://localhost:7777/auth/login 2. Enter email and password	On entering the login credentials, user will be logged in	As expected	Pass
T8	Admin Login	1. Go to http://localhost:7777/auth/login 2. Enter email and password	On entering the login credentials, admin will be logged in	As expected	Pass
T9	User Registration	1. Go to http://localhost:7777/auth/register 2. Enter all details like name, email, mobile, pan card etc.	On entering details user will have access to authorized reources	As expected	Pass
T10	User Logout	Click on logout button	User should be logged out	As expected	Pass

Test Id	Test Scenario	Test Steps	Expected Results	Actual Result	Pass / Fail
T11	Open a Fundraiser	1. Go to http://localhost:7777/explore 2. Select any fundraiser	On going on fundraiser the page will be displayed	As Expected	Pass
T12	Pay to fundraiser	1. Go to http://localhost:7777/explore 2. Select any fundraiser 3. Enter the amount and click on pay button	After following the steps you would be able to pay for a fundraiser	As Expected	Pass
T13	Admin Dashboard	Go to http://localhost:7777/account	On going on URL, Admin Dashboard page will be displayed, if you are an admin	As Expected	Pass
T14	Admin All Users Page	Go to http://localhost:7777/account/users	On going on URL, Admin User Display page will be displayed, if you are an admin	As Expected	Pass
T15	Admin all funds page	Go to http://localhost:7777/account/fundraisers	On going on URL, Admin Fundraisers display page will be displayed, if you are an admin	As Expected	Pass
T16	Admin all payments page	Go to: http://localhost:7777/account/payments	On going on URL, Admin all payments display page will be displayed, if you are an admin	As expected	Pass
T17	Admin FAQ Page	Go to http://localhost:7777/account/faq	On going on URL, Admin all FAQs display page will be displayed, if you are an admin	As expected	Pass

FUTURE SCOPE

Here are some future trends that can be opted in our product.

- <u>Internationalisation</u>: Expanding the platform to support fundraising for causes in other countries.
- <u>Mobile app development</u>: Building a mobile app to make the platform more accessible to donors and fundraisers on the go.
- <u>Crowdlending</u>: Adding a lending component to the platform, where donors can lend money to borrowers with a clear repayment plan, allowing for a more sustainable form of giving.
- <u>Impact tracking</u>: Adding features that allow users to track the impact of their donations in real-time.
- <u>Community building</u>: Building a community around the platform, where users can connect with each other and discuss causes that they care about.
- <u>Advanced analytics</u>: Developing advanced analytics features to provide users with insights on the impact of their donations and the performance of their campaigns.

BIBLIOGRAPHY

- **Books**
  - BCS-053 Web Programming
  - MCSL-016 Internet Concepts and Web Design
  - MCS-024 Object Oriented Technology and Java Programming

- **Websites**
  - https://egyankosh.ac.in
  - https://swayam.gov.in/