A PROJECT ON

Tiffin Delivery and Management System

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE COURSE OF DIPLOMA IN ADVANCED COMPUTING FROM CDAC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY

Hinjawadi

SUBMITTED BY:

Anuj Nemanwar,

Omkar Mirgane,

Rohan Dhobale,

Rajat Selukar,

Tarun Chandu Kottamasu

UNDER THE GUIDENCE OF:

Mrs. Pooja Jaiswal

Member

Sunbeam Institute of Information Technology, Pune

ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Yogesh Kolhe (Course Coordinator, SIIT ,Pune) .

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Anuj Nemanwar,

Omkar Mirgane,

Rohan Dhobale,

Rajat Selukar,

Tarun Chandu Kottamasu

PG-DAC

SIIT Pune

A PROJECT ON

"Child Adoption System"

SUBMITTED IN
PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE COURSE OF
DIPLOMA IN ADVANCED COMPUTING FROM CDAC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY
Hinjawadi

SUBMITTED BY:

Anuj Nemanwar,

Omkar Mirgane,

Rohan Dhobale,

Rajat Selukar,

Tarun Chandu Kottamasu

UNDER THE GUIDENCE OF:
Mrs. Pooja Jaiswal
Faculty Member
Sunbeam Institute of Information Technology, PUNE.

CERTIFICATE

This is to certify that the project work under the title 'Web Portal for Student and teacher' is done by Anuj Nemanwar, Omkar Mirgane, Rohan Dhobale, Rajat Selukar, Tarun Chandu Kottamasu in partial fulfillment of the requirement for award of Diploma in Advanced Computing Course.

Mr. Yogesh Kolhe

Project Guide

Course Co-Coordinator

Date: 16-07-2024

PROJECT REPORT: CHILD HOME MANAGEMENT SYSTEM

Table of Contents

1. Introduction

1.1 Project Overview

The **Child Home Management System** is a web-based application designed to manage child homes, adoption requests, parents, and employees efficiently. It provides an interactive platform for administrators, parents, employees, and child home managers to interact and manage data seamlessly.

1.2 Objectives

- To streamline the management of child homes and adoption requests.
- To provide an admin dashboard for managing parents, employees, and children.
- To enable secure authentication and authorization for different user roles.
- To ensure efficient CRUD operations for requests, children, and parents.
- To improve efficiency in record-keeping and request management.

1.3 Scope of the Project

This project covers the management of:

- Child homes and available children for adoption.
- Adoption request processing and approvals.
- Parent registration and verification.
- Employee management for child homes.
- Secure user authentication and role-based access.

2. System Architecture

The system follows a **client-server** architecture where the **React.js** frontend interacts with the **Spring Boot** backend via REST APIs. The backend communicates with a relational database (MySQL) to store user and system data.

2.1 Technology Stack

Frontend (React.js)

- React.js with Hooks
- React Router for navigation
- Axios for API communication
- Bootstrap for styling
- React Context API for authentication
- React Toastify for notifications

Backend (Spring Boot)

- Spring Boot with Spring MVC
- Spring Data JPA (Hibernate) for database management
- Spring Security with JWT for authentication
- RESTful APIs for communication
- Lombok for reducing boilerplate code
- MySQL as the database

2.2 Architectural Explanation

The **Child Home Management System** follows a **three-tier architecture**, which consists of:

1. **Presentation Layer (Frontend - React.js)**
   - Handles user interactions and displays UI components.
   - Communicates with the backend using REST APIs.
   - Implements authentication using JWT tokens.
   - Uses React Context API for state management.
2. **Business Logic Layer (Backend - Spring Boot)**
   - Exposes REST APIs to the frontend.
   - Processes business logic such as request approvals, user authentication, and role-based access control.
   - Uses Spring Security for authentication and authorization.
   - Implements service classes to manage application logic.
3. **Data Layer (Database - MySQL)**
   - Stores all persistent data, including users, requests, child homes, and employee records.
   - Uses Hibernate and JPA for database interaction.
   - Implements entity relationships like One-to-Many and Many-to-One.

Workflow

1. **User Login:**
    - Parents, employees, and admins log in through the frontend.
    - The request is sent to the backend API, where Spring Security validates credentials.
    - On success, a JWT token is returned to the frontend.
2. **Data Processing:**
    - Admins manage child homes, adoption requests, and user roles.
    - Child Home Managers register children and assign employees.
    - Parents submit adoption requests.
    - Employees handle adoption processing.
3. **Database Communication:**
    - The backend interacts with MySQL to store and retrieve data.
    - ORM (Hibernate/JPA) handles efficient data operations.
    - Queries are optimized for performance.

2.3 Use Case

Roles:

1. **Admin** – Manages users, child homes, and adoption requests.
2. **Child Home Manager** – Manages child home details, employees, and children.
3. **Parent** – Views child homes, requests adoption, and books slots.
4. **Employee** – Handles child operations and assists in adoption processes.

Use Cases:

- **Admin:**
  - Manage child homes
  - Manage parents and employees
  - Approve/reject adoption requests
  - View reports and analytics
- **Child Home Manager:**
  - Register children for adoption
  - Assign employees to child homes
  - Manage adoption requests

- **Parent:**
    - Register/Login
    - View child homes and available children
    - Submit adoption requests
    - Book an adoption slot
- **Employee:**
    - View assigned child home details
    - Assist in the adoption process
    - Handle child-related tasks

3. Functionalities

3.1 User Roles and Features

Admin

- Manage child homes (add, update, delete)
- Manage parents and employees
- View and approve adoption requests

Child Home Manager

- Manage child home details
- Register children for adoption
- Assign social workers (employees) to child homes
- Manage adoption requests and feedback

Parent

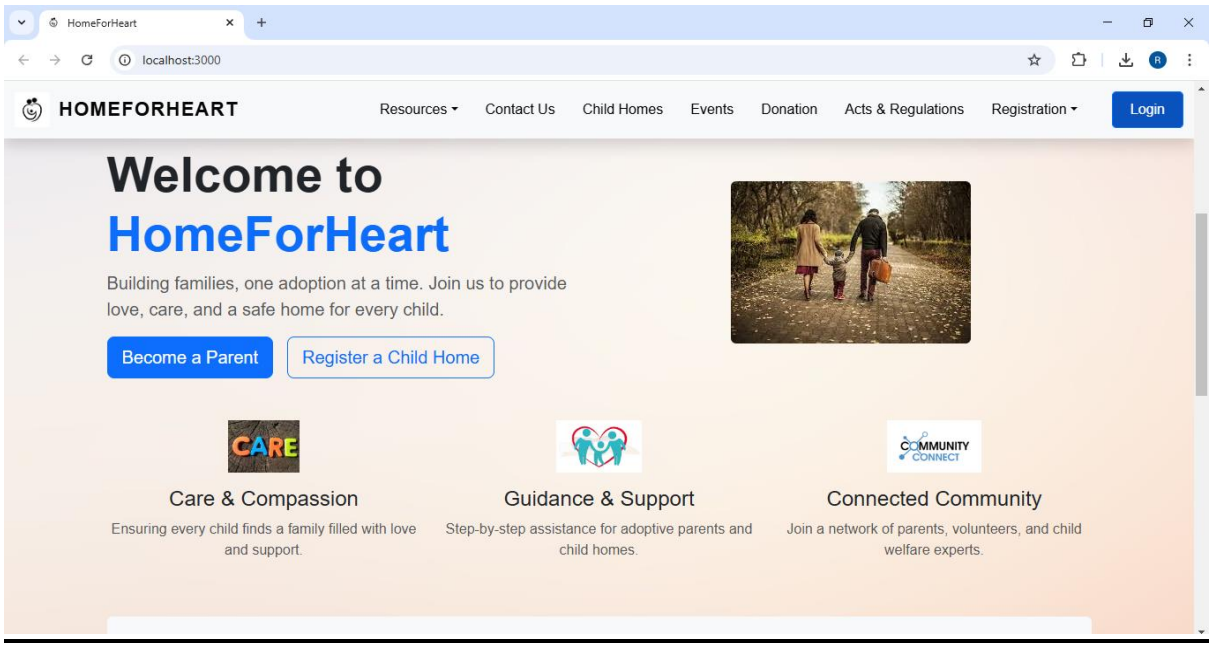
- Register and log in
- View child home details
- Request adoption of a child
- Book an adoption slot

Employee

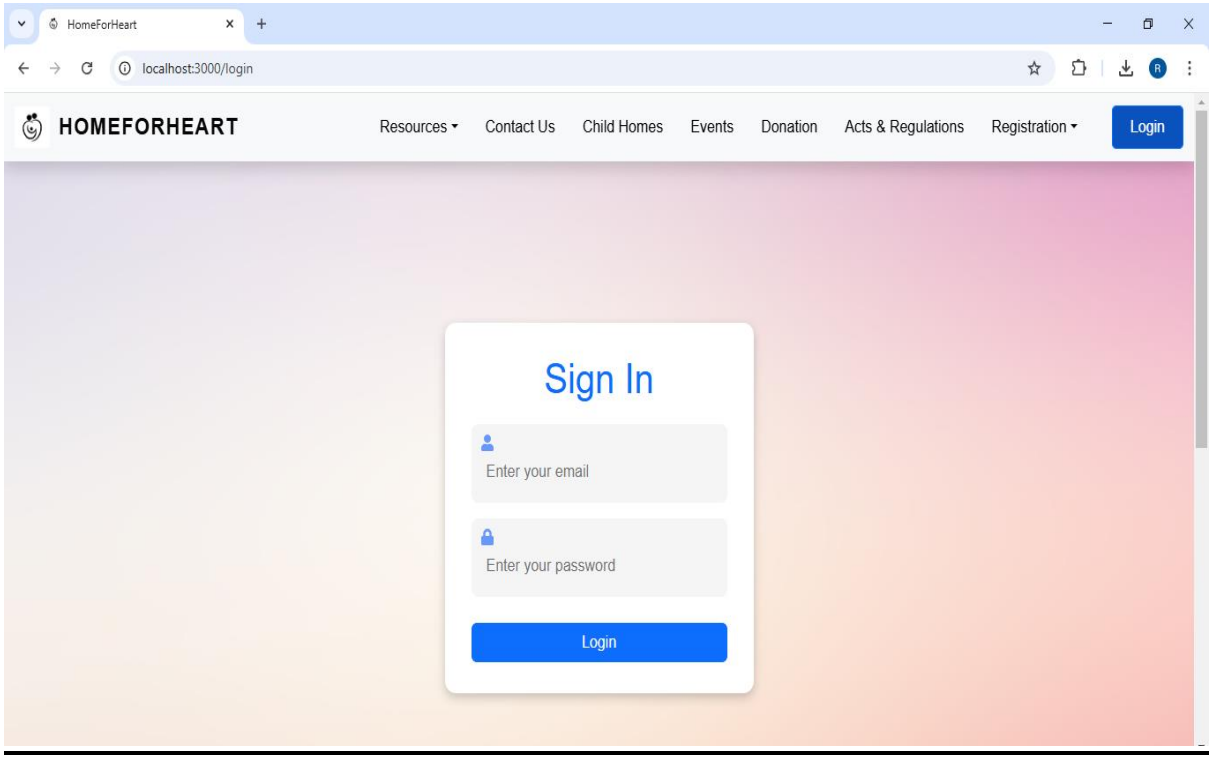
- View assigned child home details
- Handle child-related operations
- Assist with adoption process

3.2 User Interface Screens

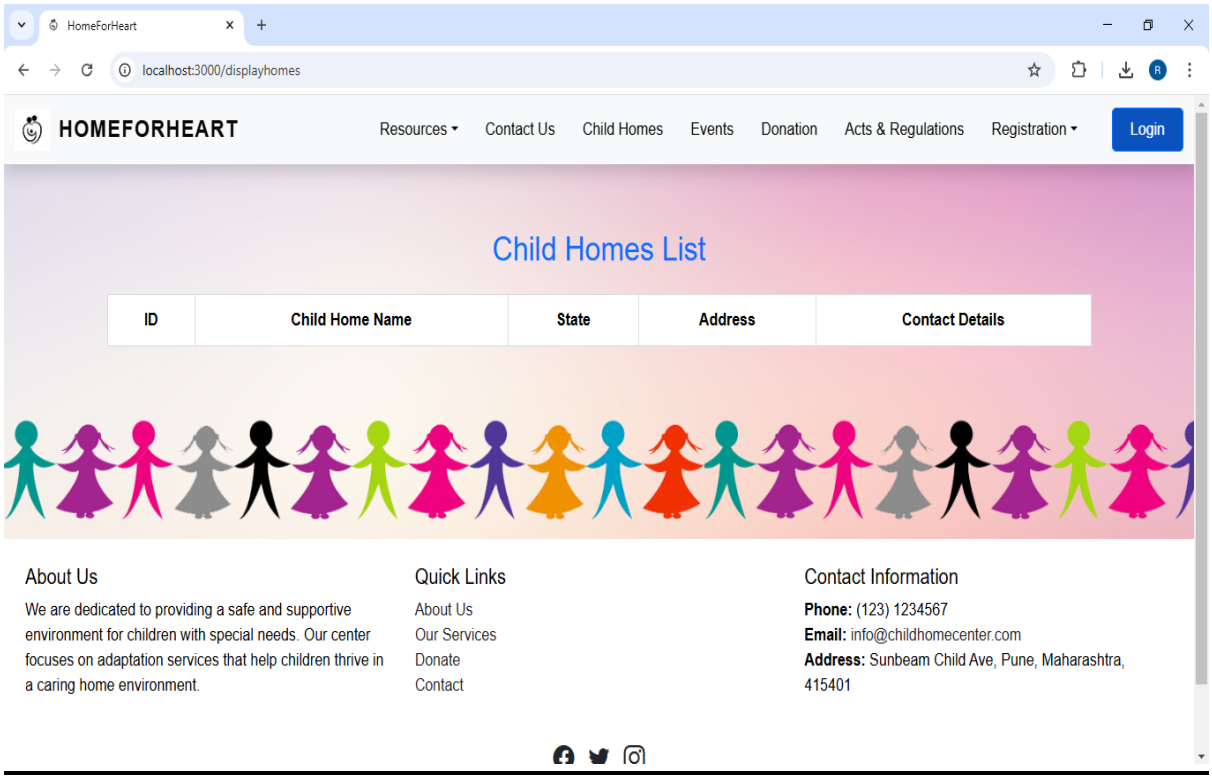
- **Home Page**



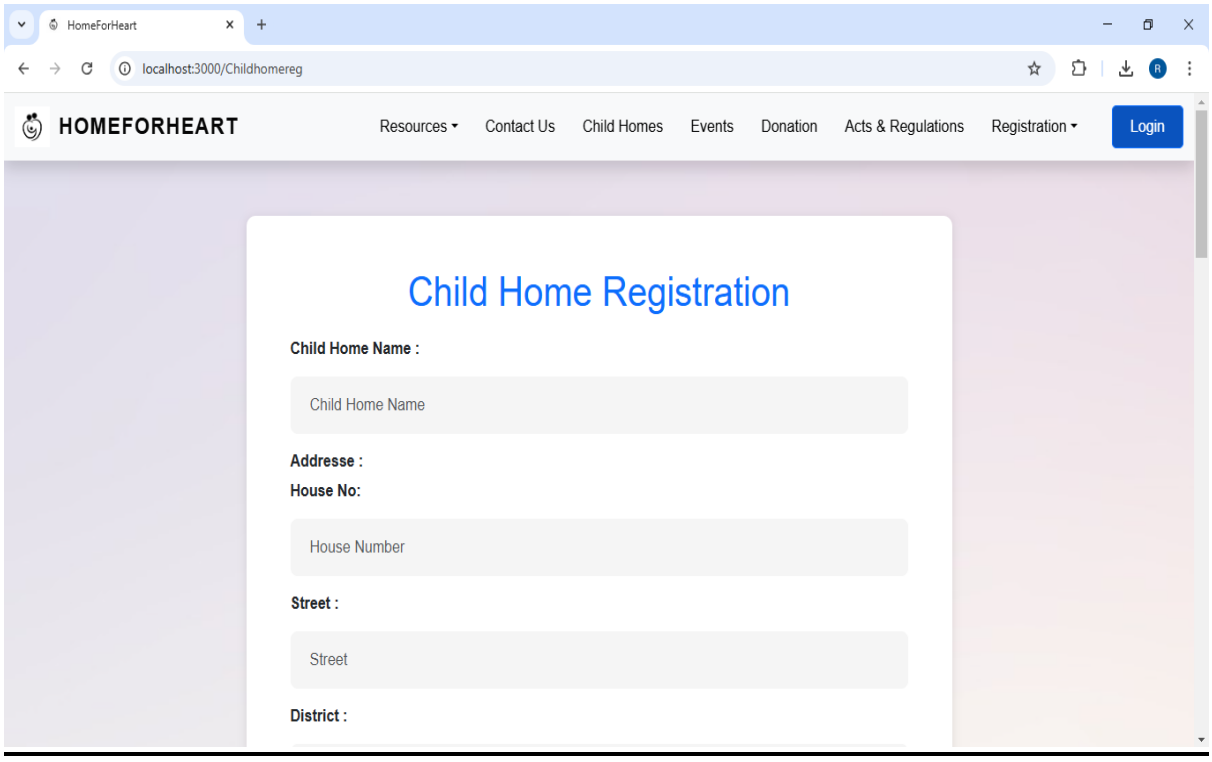
- **Login Page**



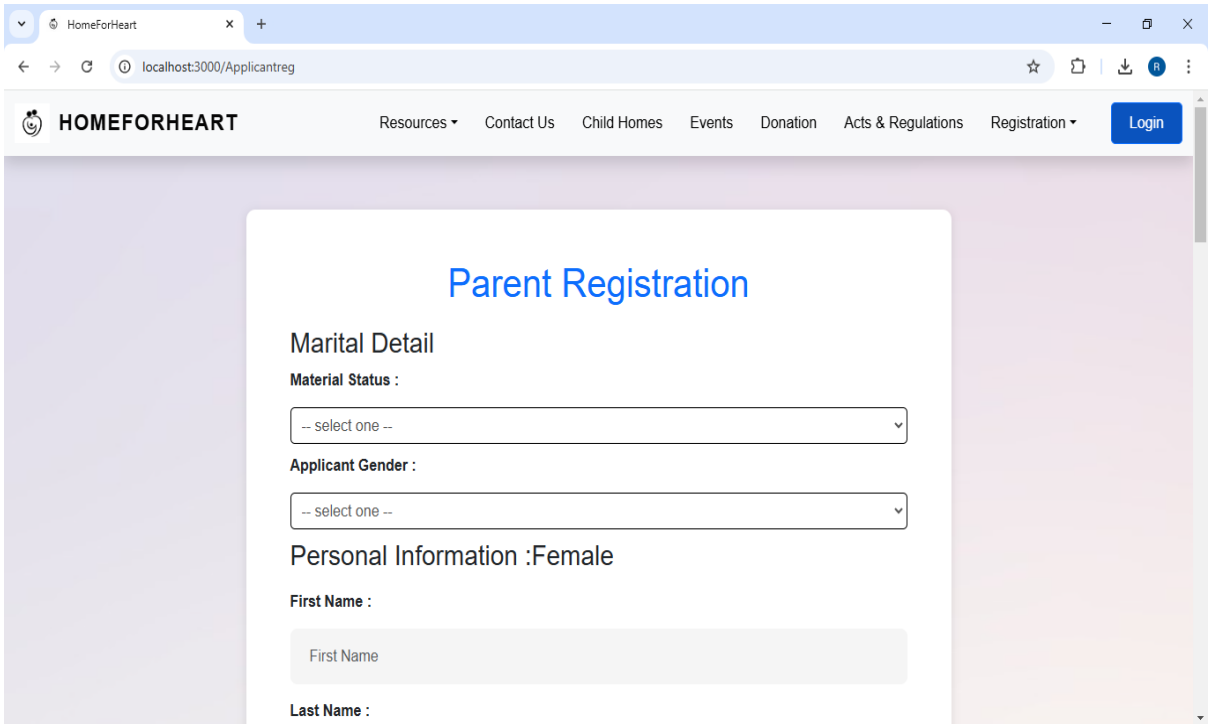
- **Child Homes List Page**



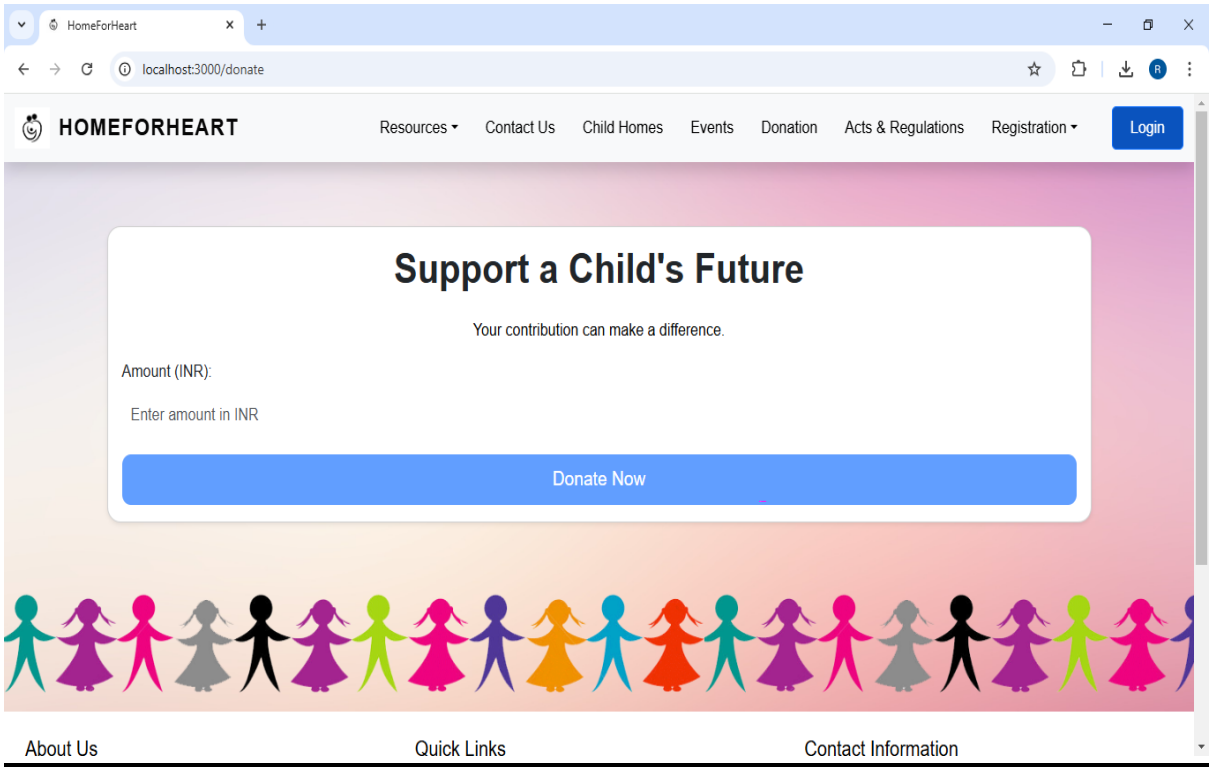
- **Child Home Registration Page**



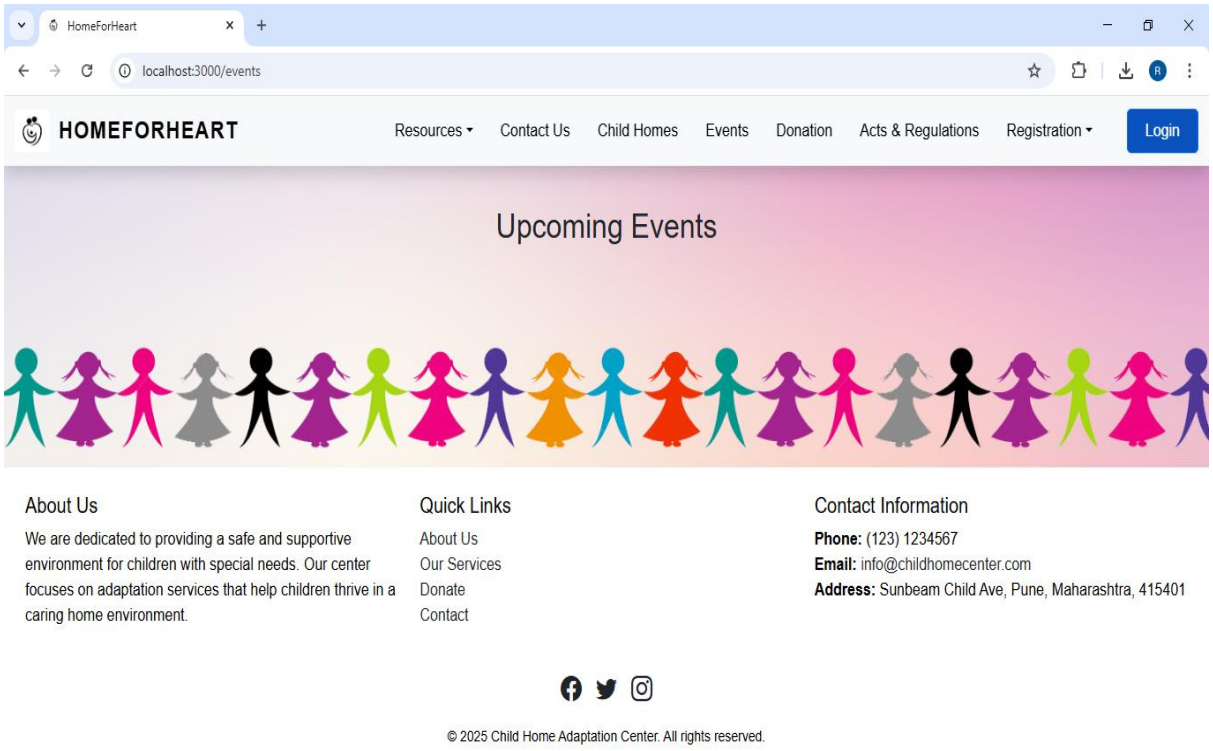
- **Parent Registration Page**



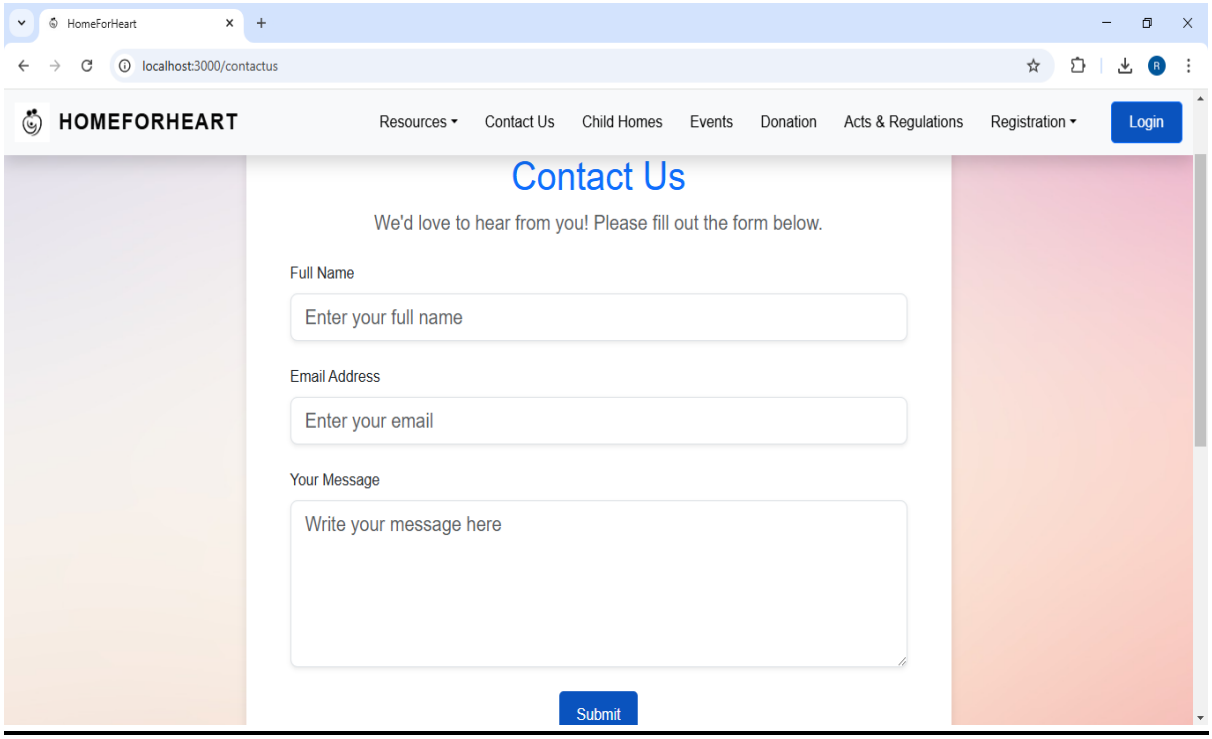
- **Donation Page**



- **Events Page**



- **Contact Us Page**



4. Database Design

4.1 Tables and Entities

1. User Table

Column Name	Data Type	Description
`user_id`	INT (PK)	Unique identifier for each user
`first_name`	VARCHAR(50)	User's first name
`last_name`	VARCHAR(50)	User's last name
`email`	VARCHAR(100)	User's email address
`password`	VARCHAR(255)	Hashed password
`role`	ENUM('ADMIN', 'PARENT', 'EMPLOYEE', 'CHILDHOME_MANAGER')	User role
`created_at`	TIMESTAMP	Timestamp when user was created

2. ChildHome Table

Column Name	Data Type	Description
`childhome_id`	INT (PK)	Unique identifier for child home
`name`	VARCHAR(100)	Name of the child home
`location`	VARCHAR(255)	Address of child home
`capacity`	INT	Maximum capacity of children
`manager_id`	INT (FK)	User ID of child home manager

3. Child Table

Column Name	Data Type	Description
`child_id`	INT (PK)	Unique identifier for child
`name`	VARCHAR(100)	Child's name
`age`	INT	Age of the child
`gender`	ENUM('Male', 'Female', 'Other')	Gender of the child
`status`	ENUM('Available', 'Adopted', 'Pending')	Adoption status
`childhome_id`	INT (FK)	Child Home where the child is registered

4. Parent Table

Column Name	Data Type	Description
`parent_id`	INT (PK)	Unique identifier for parent
`user_id`	INT (FK)	Associated user ID from User table
`occupation`	VARCHAR(100)	Parent's occupation
`income`	DECIMAL(10,2)	Annual income
`verified`	BOOLEAN	Indicates if the parent is verified

5. Adoption Request Table

Column Name	Data Type	Description
`request_id`	INT (PK)	Unique identifier for adoption request
`parent_id`	INT (FK)	Parent requesting adoption
`child_id`	INT (FK)	Child being adopted
`status`	ENUM('Pending', 'Approved', 'Rejected')	Status of the request
`date_submitted`	TIMESTAMP	Date the request was submitted

6. Employee Table

Column Name	Data Type	Description
`employee_id`	INT (PK)	Unique identifier for employee
`user_id`	INT (FK)	Associated user ID from User table
`role`	VARCHAR(50)	Role of employee (Social Worker, Caretaker, etc.)
`childhome_id`	INT (FK)	Associated child home

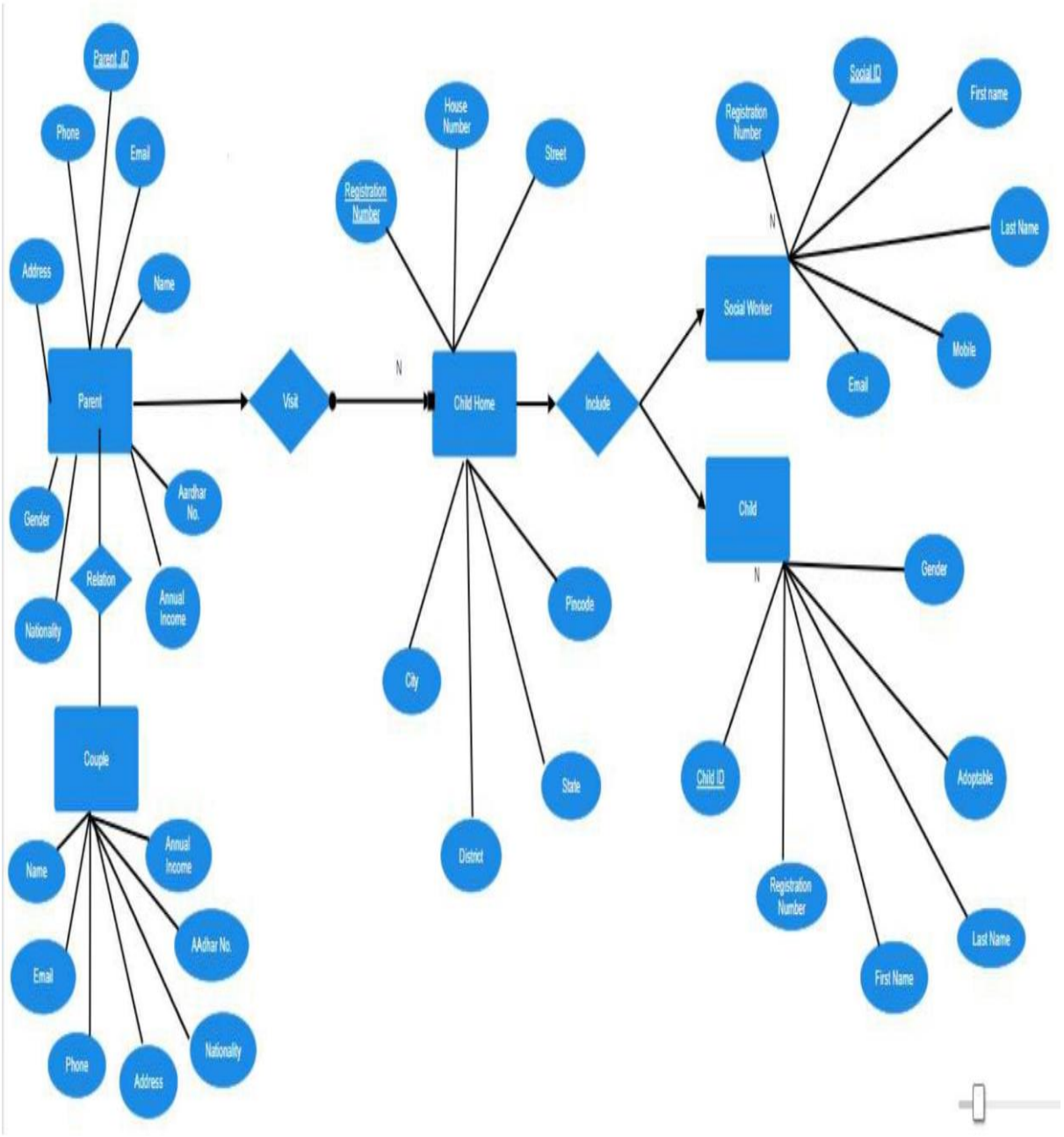
7. Events Table

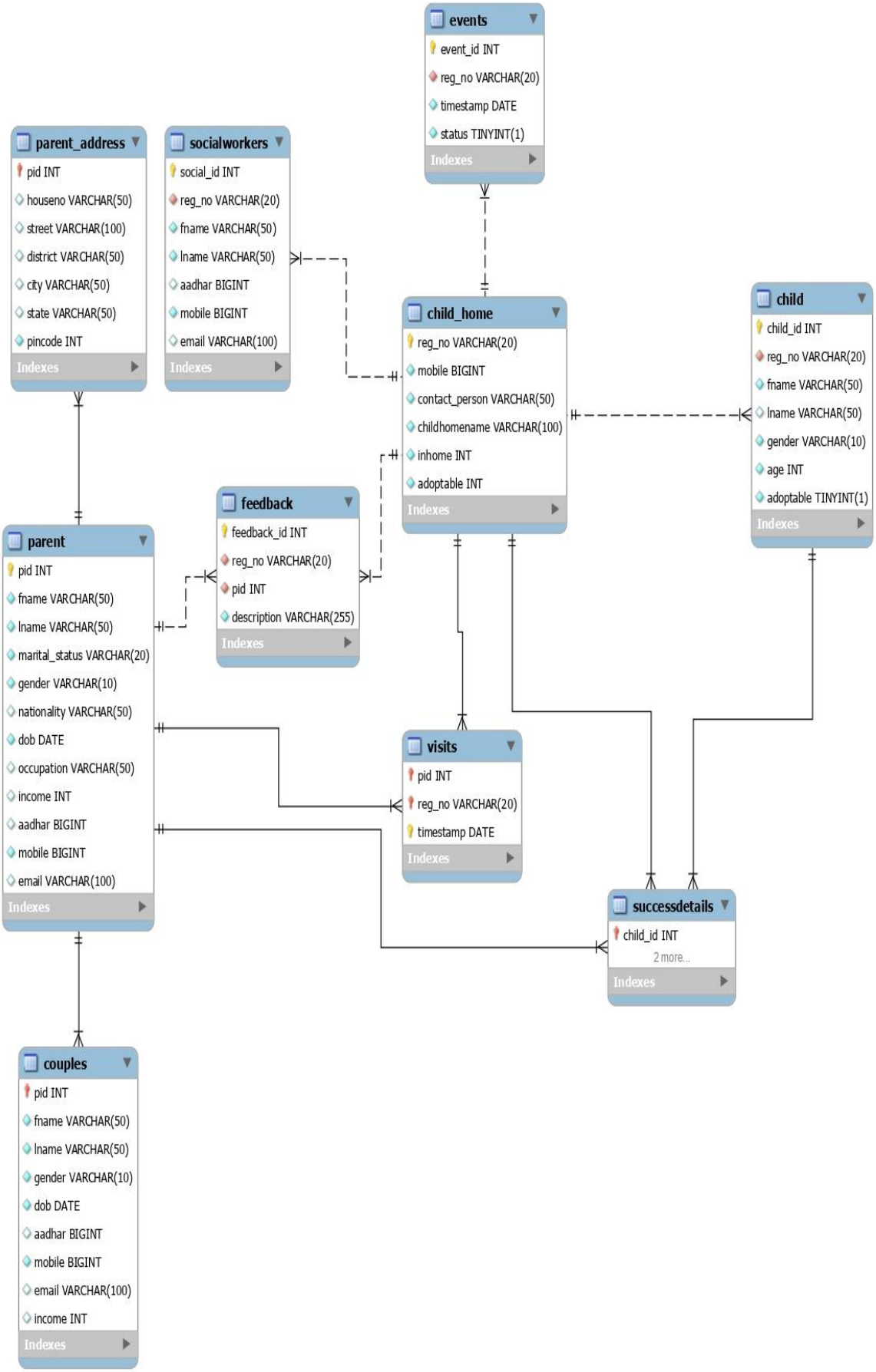
Column Name	Data Type	Description
`event_id`	INT (PK)	Unique identifier for an event
`name`	VARCHAR(100)	Name of the event
`date`	DATE	Date of the event
`description`	TEXT	Event description
`childhome_id`	INT (FK)	Associated child home

4.2 Entities and Relationships:

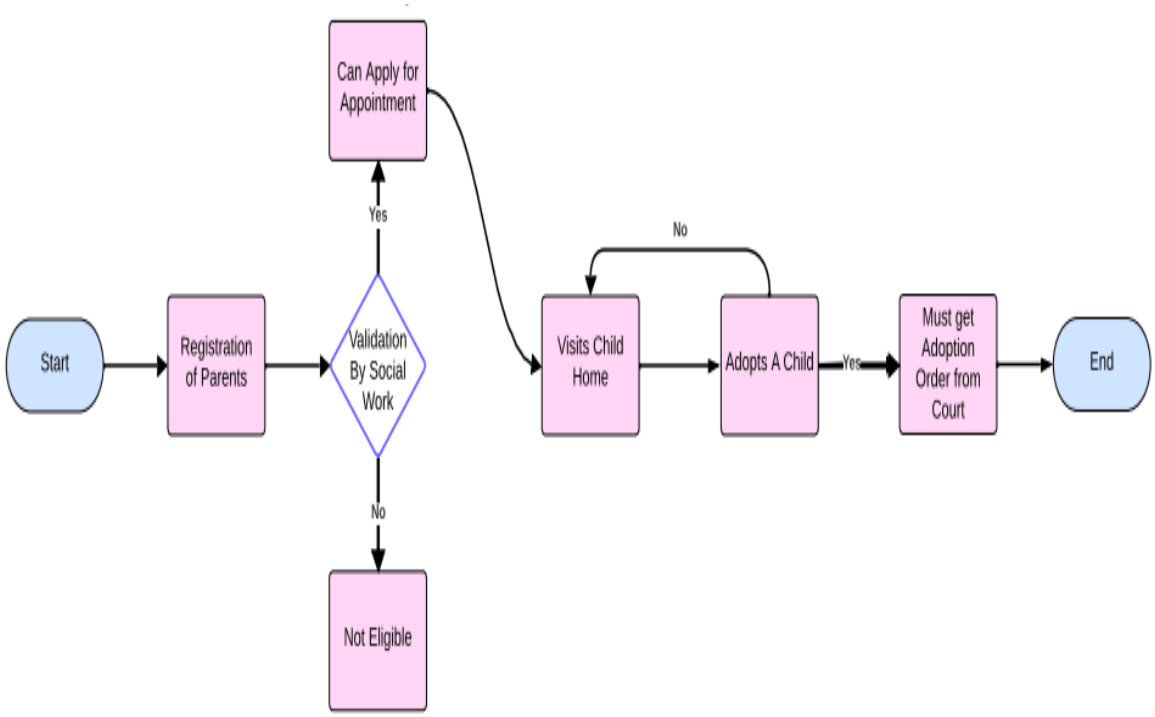
1. **User**
    - Attributes: UserID, FirstName, LastName, Email, Role, Password, Status
    - Relationships:
        - One-to-Many with ChildHome
        - One-to-One with Parent
        - One-to-One with Employee
2. **ChildHome**
    - Attributes: ChildHomeID, Name, Location, Capacity, ManagerID
    - Relationships:
        - One-to-Many with Child
        - One-to-Many with Employee
        - One-to-Many with Request
        - One-to-Many with Events
3. **Child**
    - Attributes: ChildID, Name, Age, Gender, Status
    - Relationships:
        - Many-to-One with ChildHome
        - One-to-One with Request

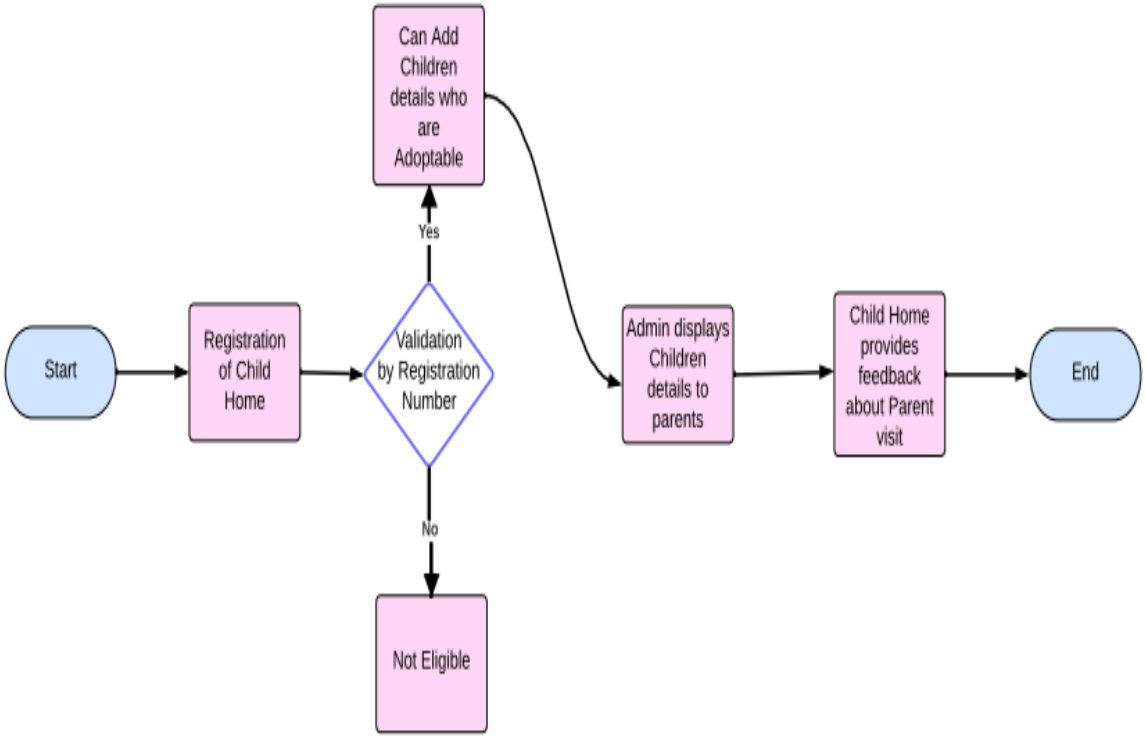
4. **Parent**
    - Attributes: ParentID, Name, Occupation, Income, Verified
    - Relationships:
        - One-to-One with User
        - One-to-Many with Request
5. **Request**
    - Attributes: RequestID, Status, Date, Feedback
    - Relationships:
        - Many-to-One with ChildHome
        - One-to-One with Child
        - One-to-One with Parent
6. **Employee**
    - Attributes: EmployeeID, Name, Role
    - Relationships:
        - One-to-One with User
        - Many-to-One with ChildHome
7. **Events**
    - Attributes: EventID, Name, Date, Description
    - Relationships:
        - Many-to-One with ChildHome

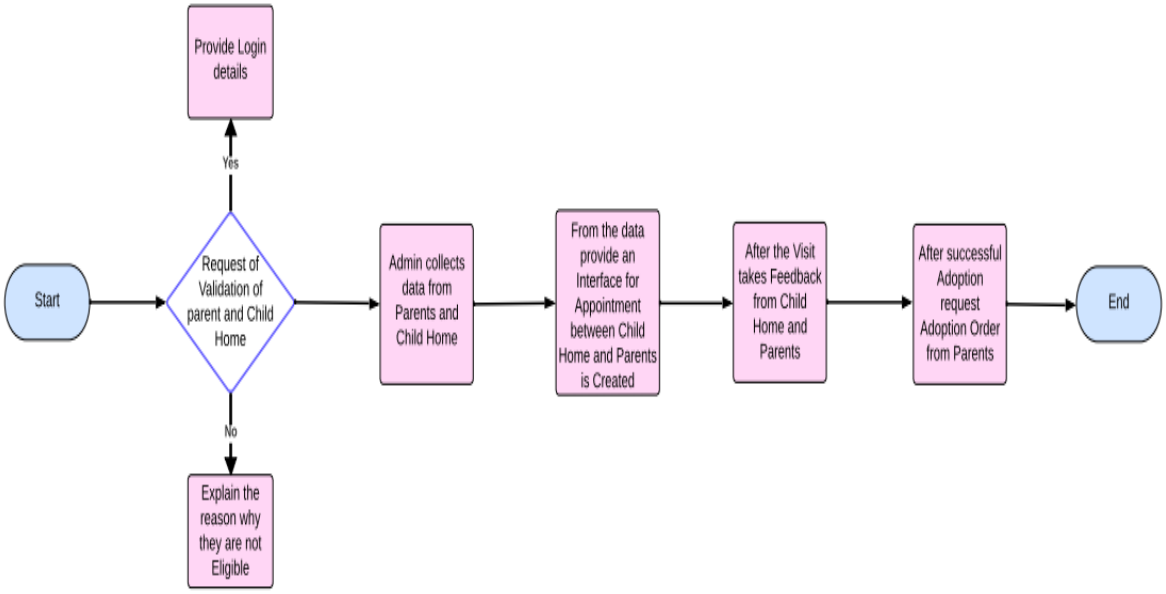

Parent_ID
Phone
Email
Address
Name
Parent
Gender
Aardhar No.
Relation
Nationality
Annual Income
Visit
N
House Number
Registration Number
Street
Child Home
City
Pincode
State
District
Include
Social Worker
Registration Number
Social ID
First name
Last Name
Mobile
Email
N
Child
Gender
Child ID
Adoptable
Registrafon Number
Last Name
First Name
Couple
Name
Annual Income
AAdhar No.
Email
Phone
Address
Nationality


events
event_id INT
reg_no VARCHAR(20)
timestamp DATE
status TINYINT(1)
Indexes
parent_address
pid INT
houseno VARCHAR(50)
street VARCHAR(100)
district VARCHAR(50)
city VARCHAR(50)
state VARCHAR(50)
pincode INT
Indexes
socialworkers
social_id INT
reg_no VARCHAR(20)
fname VARCHAR(50)
lname VARCHAR(50)
aadhar BIGINT
mobile BIGINT
email VARCHAR(100)
Indexes
child_home
reg_no VARCHAR(20)
mobile BIGINT
contact_person VARCHAR(50)
childhomename VARCHAR(100)
inhome INT
adoptable INT
Indexes
child
child_id INT
reg_no VARCHAR(20)
fname VARCHAR(50)
lname VARCHAR(50)
gender VARCHAR(10)
age INT
adoptable TINYINT(1)
Indexes
feedback
feedback_id INT
reg_no VARCHAR(20)
pid INT
description VARCHAR(255)
Indexes
parent
pid INT
fname VARCHAR(50)
lname VARCHAR(50)
marital_status VARCHAR(20)
gender VARCHAR(10)
nationality VARCHAR(50)
dob DATE
occupation VARCHAR(50)
income INT
aadhar BIGINT
mobile BIGINT
email VARCHAR(100)
Indexes
visits
pid INT
reg_no VARCHAR(20)
timestamp DATE
Indexes
successdetails
child_id INT
2 more...
Indexes
couples
pid INT
fname VARCHAR(50)
lname VARCHAR(50)
gender VARCHAR(10)
dob DATE
aadhar BIGINT
mobile BIGINT
email VARCHAR(100)
income INT
Indexes

4.3 Data Flow


Can Apply for Appointment
Yes
No
Start
Registration of Parents
Validation By Social Work
Visits Child Home
Adopts A Child
Yes
Must get Adoption Order from Court
End
No
Not Eligible


Can Add Children details who are Adoptable
Yes
Start
Registration of Child Home
Validation by Registration Number
Admin displays Children details to parents
Child Home provides feedback about Parent visit
End
No
Not Eligible


Provide Login details
Yes
Start
Request of Validation of parent and Child Home
No
Explain the reason why they are not Eligible
Admin collects data from Parents and Child Home
From the data provide an Interface for Appointment between Child Home and Parents is Created
After the Visit takes Feedback from Child Home and Parents
After successful Adoption request Adoption Order from Parents
End

Key Components of the Data Flow

1. **External Entities**
    - Admin
    - Child Home Manager
    - Parents
    - Employees
2. **Processes**
    - User Authentication & Role Assignment
    - Child Registration & Management
    - Adoption Request Handling
    - Parent Registration & Verification
    - Event Management
3. **Data Stores**
    - Users Database
    - Children Database
    - Adoption Requests Database
    - Events & Feedback Database

5. API Endpoints

5.1 Authentication APIs

HTTP Method	Endpoint	Description
POST	/login	User login and JWT token generation
POST	/register	Register a new user (Admin, Parent, Employee, etc.)
POST	/logout	Logout user and invalidate session

5.2 Admin APIs

HTTP Method	Endpoint	Description
DELETE	/admin/deletechildhome/{id}	Delete a child home
DELETE	/admin/deleteparent/{id}	Delete a parent
GET	/admin	Fetch all requests
GET	/admin/feedback/{id}	Get feedback for a request
GET	/admin/child	Fetch all child homes
GET	/admin/parent	Fetch all parents

5.3 Child Home Manager APIs

HTTP Method	Endpoint	Description
POST	/childhome/addchild	Add a new child
GET	/childhome/allEventsDetails	Get all events details
POST	/childhome/addevents	Add an event
POST	/childhome/addemployee	Add a new employee (social worker)
POST	/childhome/addrequest	Add a new adoption request
GET	/childhome/getchildhomedetails/{id}	Get details of a specific child home
POST	/childhome/updatechildhome/{id}	Update child home details
POST	/childhome/updaterequest/{id}	Update request status
GET	/childhome/allChildhomedetails	Get all child home details
GET	/childhome/childhomecount	Get count of child homes
GET	/childhome/employeecount	Get count of employees
GET	/childhome/getchilds/{id}	Get all children in a specific child home

5.4 Parent APIs

HTTP Method	Endpoint	Description
POST	/parent/bookSlot/{id}	Book a slot for adoption
GET	/childhomes	Get all child homes available
GET	/profile/{id}	Get parent profile details
PUT	/profile/{id}	Update parent profile details

5.5 Employee APIs

HTTP Method	Endpoint	Description
`POST`	`/employee/updateemployee`	Update employee details
`GET`	`/employee/getemployeedetails/{id}`	Get employee details
`GET`	`/employee/getrequestdetails`	Get all requests assigned to an employee
`POST`	`/employee/updaterequest/{id}`	Update request status

6. Security and Authentication

Security is a critical aspect of the **Child Home Management System**, ensuring that sensitive data related to **users**, **children**, **adoption requests**, **and child homes** remains protected. The system implements multiple security mechanisms, including **JWT authentication**, **role-based access control (RBAC)**, **encryption**, **and API security**.

6.1 Authentication Mechanism

The system uses **JWT (JSON Web Token)** for authentication.

- **Process Flow:**
    1. The user logs in using their **email** and **password**.
    2. The backend validates the credentials using **Spring Security**.
    3. If valid, the backend generates a **JWT token** and sends it to the frontend.
    4. The frontend stores the JWT token in **local storage/session storage**.
    5. For every API request, the token is sent in the **Authorization header**.
    6. The backend verifies the JWT token before processing the request.

JWT Token Structure:

- **Header:** Algorithm (HS256), Token Type
- **Payload:** User ID, Role, Expiry
- **Signature:** Encrypted using a secret key

6.2 Role-Based Access Control (RBAC)

The system implements **RBAC** to ensure that different user roles have restricted access to APIs.

Role	Accessible Features
Admin	Manage child homes, employees, and requests
Child Home Manager	Register children, manage employees, and adoption requests
Parent	View child homes, submit adoption requests
Employee	View assigned child homes, assist in adoption

- **Admin can access all APIs**
- **Child Home Managers can only access child home-related endpoints**
- **Parents can only request adoptions**
- **Employees have restricted access to assigned homes only**

6.3 Password Security

- Passwords are **hashed** using **BCrypt** before storing them in the database.
- During login, the password is verified against the **hashed password**.

6.4 API Security

- **JWT Authentication**: All API endpoints are secured using JWT.
- **CORS (Cross-Origin Resource Sharing)**: Configured to allow only trusted domains to access APIs.
- **Rate Limiting**: Prevents **brute-force attacks** by limiting login attempts.
- **SQL Injection Prevention**: Uses **prepared statements** in queries.

6.5 Data Encryption & Protection

- **Sensitive data** like passwords and personal information are **encrypted** using AES-256.
- **HTTPS (SSL/TLS)** is enforced to protect data transmission.
- **Logging & Monitoring** is enabled to track **unauthorized access attempts**.

6.6 Logout and Session Management

- Users can **log out** by invalidating the JWT token.
- Tokens **expire automatically** after a predefined time (e.g., **24 hours**).
- **Refresh Tokens** can be used to generate new tokens securely.

7. Performance Optimization

- Implemented caching using Redis.
- Optimized API calls to reduce load times.
- Used pagination for large datasets.

8. Challenges & Solutions

- **Managing authentication for different roles** – Solved using JWT and Spring Security.
- **Efficient state management in React** – Used React Context API to manage authentication.
- **Database relations & constraints** – Implemented entity relationships using Hibernate.

9. Future Enhancements

- Implementing **real-time notifications** using WebSockets.
- Enhancing **UI/UX** with Material-UI.

10. Conclusion

The Child Home Management System simplifies child adoption and child home management through a structured web-based solution. The combination of **React.js** for the frontend and **Spring Boot** for the backend ensures a **scalable, secure, and efficient** application.

11. REFERENCES

6.1. Spring Boot Documentation
URL: https://spring.io/projects/spring-boot

6.2. React.js Documentation
URL: https://reactjs.org/docs/getting-started.html

6.3. Redux Documentation
URL: https://redux.js.org

6.4. Java Programming Language
URL: https://www.oracle.com/java/

6.5. MySQL Workbench Documentation
URL: https://dev.mysql.com/doc/workbench/en/

6.6. Spring Boot with React and Redux
URL: https://www.baeldung.com/spring-boot-react-and-redux

6.7. Java Persistence API (JPA) Documentation
URL: https://www.eclipse.org/eclipselink/documentation/2.7/

6.8. Swagger Documentation for Spring Boot
URL: https://springdoc.org/

6.9. MDN Web Docs
URL: https://developer.mozilla.org/

6.10. React Redux Integration Guide
URL: https://react-redux.js.org/