

ORPHANAGE MANAGEMENT SYSTEM

*Project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

M.GAYATHRI	(21UECM0076)
BHUVANESWARL.S	(21UECS0090)
YAZHINI.G	(21UECM0265)

10211CS212 - WEB AND MOBILE APPLICATION DEVELOPMENT

SUMMER 2023-2024

*Under the guidance of
DR. C.M. CHIDAMBARANATHAN, B.E., M.Tech., Ph.D
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE AND TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA
October, 2023**

CERTIFICATE

It is certified that the work contained in the project report titled "ORPHANAGE MANAGEMENT SYSTEM" by M.Gayathri (21UECM0076), Bhuvaneswari.S (21UECS0090), Yazhini.G (21UECM0265) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr.C.M.Chidambaranathan

Assistant Professor

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr.Sagunthala R&D

Institute of Science & Technology

October, 2023

Signature of Head of the Department

Dr.M.S.Murali Dhar

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

October, 2023

Signature of the Dean

Dr. V. Srinivasa Rao

Professor & Dean

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

October, 2023

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

M.GAYATHRI

Date: / /

(Signature)

BHUVANESWARI.S

Date: / /

(Signature)

YAZHINI.G

Date: / /

APPROVAL SHEET

This project report entitled (ORPHANAGE MANAGEMENT SYSTEM) by (M.GAYATHRI (21UECM0076), (BHUVANESWARI.S (21UECS0090), (YAZHINI.G (21UECM0265)) are approved for the degree of B.Tech in Computer Science & Engineering.

Examiners

Handling faculty

Dr.C.M. Chidambaranathan ,M.Tech.,

Date: / /

Place:

ABSTRACT

As Orphanages are increasing in day to basis simultaneously the helping hands are also increasing in higher range. The practical difficulty is finding out the real orphans and the less knowledge about the donation details. So this applications was developed to help to reach the needy orphans by the volunteering donations. It allows the users to search and find the orphanages on internet and then they can donate with the orphanages. Although the people with helping mind in today's era are less, there are also people who are concerned about the well-being of elderly orphans, who are left out by their families or abandoned. So, this paper is about the web application which involves the process of finding a home or an orphanage for the needy. The process of web application starts with the person (who is ready to help) can login to the page and register by entering the details of orphans. Then the application searches for the nearest home or orphanage with available vacancy and facilities. Then the concerned person or other NGO's can take the corresponding person to the available home. The Administrator of the website will be able to see the details of the needy. And, the responsibility of maintaining a database of the information of registered people will be the admin's job. Also, constantly updating the database is important, in case of any adoption or any other reason for removing the person from the database.

Keywords: Donation, NGO, Website, Admin

LIST OF FIGURES

3.1	Architecture Diagram	5
3.2	Data Flow Diagram	6
3.3	Home page	7
3.4	Signup and Login page	7
3.5	Form Validation	8
5.1	Test Image	18

LIST OF ACRONYMS AND ABBREVIATIONS

S NO	ABBREVIATION	DEFINITION
1	HTML	Hyper Text Markup Language
2	CSS	Cascading Style Sheets
3	JS	Java Script
4	PHP	Hypertext Preprocessor
5	DB	Database
6	SQL	Structured Query Language

TABLE OF CONTENTS

	Page.No
ABSTRACT	iv
LIST OF FIGURES	v
LIST OF ACRONYMS AND ABBREVIATIONS	vi
1 INTRODUCTION	1
1.1 Introduction:	1
1.2 Aim of the project	1
1.3 Project Domain	1
1.4 Scope of the Project	1
1.5 Methodology	2
2 REQUIREMENT SPECIFICATION	3
2.1 User characteristics	3
2.2 Dependencies	3
2.3 Hardware specification	3
2.4 Software specification	4
3 WEBSITE DESIGN	5
3.1 Sitemap	5
3.2 Design Phase	6
3.2.1 Data Flow Diagram	6
3.3 Front End and Back End Design	7
3.3.1 Home Page	7
3.3.2 Signup and Login page	7
3.3.3 Form Validation	8
3.3.4 Parse the webpage using JQuery and DOM	8
3.3.5 Creation of Webserver using Node Js	9
3.3.6 Design of Three Tier application using Node js and MySQL	9
3.3.7 Design of Reactive form for User Registration using Angular	11

3.3.8	Develop web application to implement routing and navigation in Angular . .	12
3.3.9	Creation of Microservices	13
3.3.10	Conversion of Webpage to App	13
4	TESTING	15
4.1	Testing	15
4.1.1	Test Result	16
4.1.2	Test Bugs	16
5	WEBSITE LAUNCH	18
6	RESULTS AND DISCUSSIONS	19
6.1	Website performance	19
6.2	Security	19
6.3	Responsiveness and mobile-friendliness	20
7	CONCLUSION AND FUTURE ENHANCEMENTS	21
7.1	Conclusion	21
7.2	Future Enhancements	21
8	SOURCE CODE	23
9	SCREENSHOTS	31
9.0.1	Login Page and Signup Page	31
9.0.2	Form Validation Page	31
9.0.3	Home Page	32
	References	32

Chapter 1

INTRODUCTION

1.1 Introduction:

An orphanage management system website is a critical tool designed to streamline and enhance the operations of orphanages, foster homes, and child care institutions. This digital platform serves as a central hub for managing various aspects of orphanage administration, ensuring the well-being and development of the children under their care, and facilitating communication with stakeholders.

1.2 Aim of the project

To create a Full stack web application on the domain Orphanage Management System using Web development technologies called HTML, CSS, JS , Bootstrap , NodeJS, Angular, PHP, MySQL, Type-script etc.n Orphanage Management System website is to create a comprehensive digital platform that enhances the management, welfare, and communication processes within an orphanage or child care institution.

1.3 Project Domain

The software we have used in this is Node JS, Angular JS, and used Databases like MYSQL PostgreSQL to store the data. The whole concept depends on web development framework such as Bootstrap and the whole code needs to be executed on web server in order to handle the traffic load and database size.

1.4 Scope of the Project

Orphanage Management System is an integrated software developed for Donating orphan children. The scope of the project defines the boundaries, objectives, and deliverables of the development effort. It outlines what will be included within the system and what will not.

1.5 Methodology

The methodology used for developing an orphanage management system website should be chosen based on the project's requirements, team expertise, and the specific needs of the orphanage or child care institution. Several software development methodologies can be applied to such a project.

Chapter 2

REQUIREMENT SPECIFICATION

2.1 User characteristics

When developing an orphanage management system website, it's crucial to consider the various user characteristics to ensure that the system meets the needs of different users effectively. The user characteristics can vary depending on the roles and responsibilities within the orphanage or child care institution.

2.2 Dependencies

- **Software dependencies:** The website will need to be developed using a web development framework such as Bootstrap. It will also need to be integrated with a database server such as MySQL or PostgreSQL.
- **Hardware dependencies:** The website will need to be hosted on a web server. The server will need to be powerful enough to handle the expected traffic load and the database size.

2.3 Hardware specification

- ARM64 or x64 processor; Quad-core or better recommended. ARM 32 processors are not supported.
- Minimum of 4 GB of RAM. Many factors impact resources used; we recommend 16 GB RAM for typical professional solutions.
- Windows 365: Minimum 2 vCPU and 8 GB RAM. 4 vCPU and 16 GB of RAM recommended.
- Hard disk space: Minimum of 850 MB up to 210 GB of available space, depending on features installed; typical installations require 20-50 GB of free space. We recommend installing Windows and Visual Studio on a solid-state drive (SSD) to increase performance.
- Video card that supports a minimum display resolution of WXGA (1366 by 768); Visual Studio will work best at a resolution of 1920 by 1080 or higher.

2.4 Software specification

Visual Studio Code is commonly supported on

- Windows 11 minimum supported OS version or higher: Home, Pro, Pro Education, Pro for Workstations, Enterprise, and Education
- Supported Windows 11 OS can be found here: [Windows 11 Enterprise and Education Support](#)
- Windows 10 minimum supported OS version or higher: Home, Professional, Education, and Enterprise.
- Supported Windows 10 OS can be found here: [Windows 10 Enterprise and Education Support](#)

The software requirements for NodeJS,AngularJS development include the following:

- Web browser
- JavaScript
- AngularJS library
- Code editor or IDE
- Version control system

Chapter 3

WEBSITE DESIGN

3.1 Sitemap

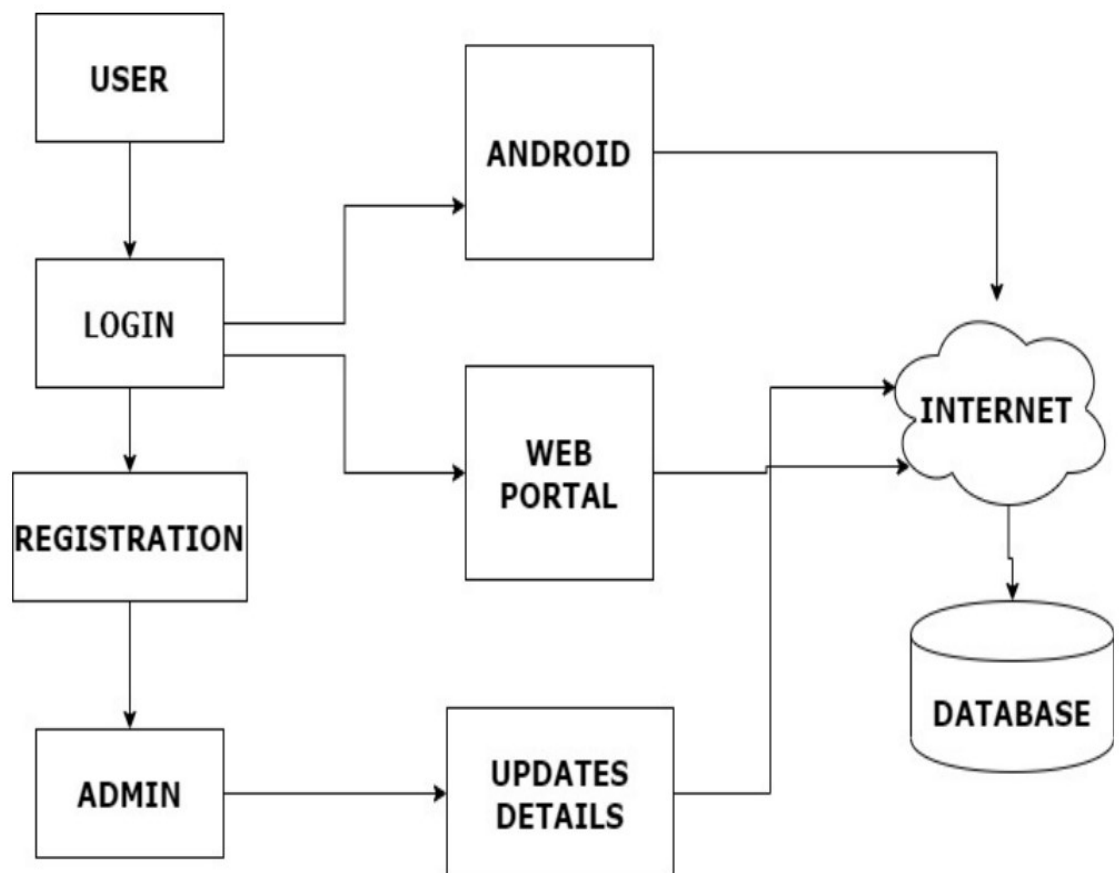


Figure 3.1: Architecture Diagram

Figure 3.1: The sitemap describes how our project works based on this map. Firstly we have to login into the website as user, it will show the home page after login in this page you can see the registration page, it will take it to admin, then you will have access to fill details and sign out options within it.

3.2 Design Phase

3.2.1 Data Flow Diagram

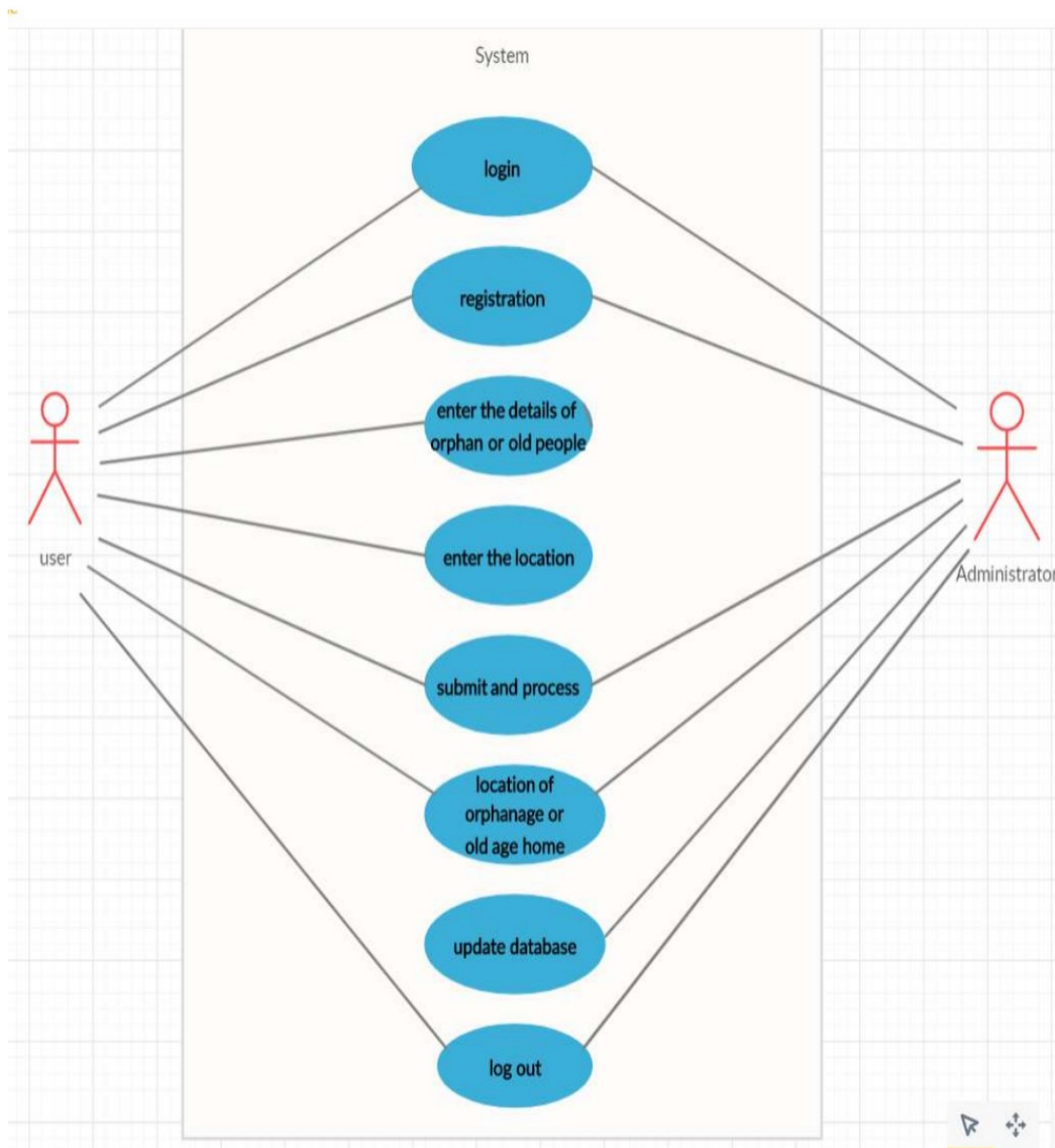


Figure 3.2: Data Flow Diagram

Figure 3.2: Firstly we will see home page, there will be options like about, login, service, history, contact in it. We have to login for the registration process. Then enter the necessary details. The location needed to be updated, which is a mandatory thing. Click on submit and the database will be automatically updated. After the process completion, you can logout of the page. The administrator plays major role in the data flow.

3.3 Front End and Back End Design

3.3.1 Home Page

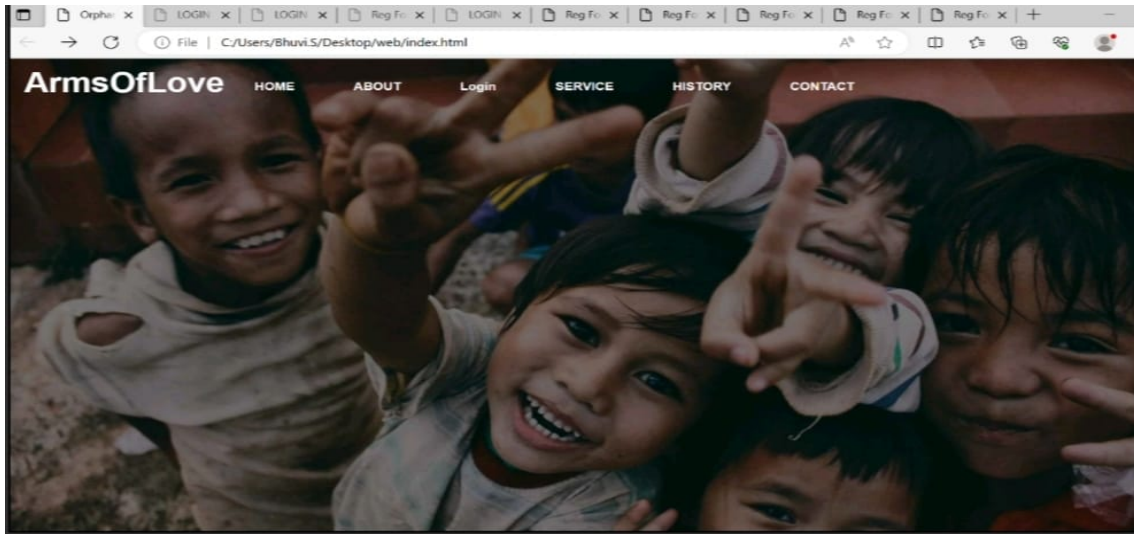


Figure 3.3: Home page

Figure 3.3 Home Page is our complete outlook of the homepage for our website. It provide users with essential information and quick access to various functionalities of the system.

3.3.2 Signup and Login page

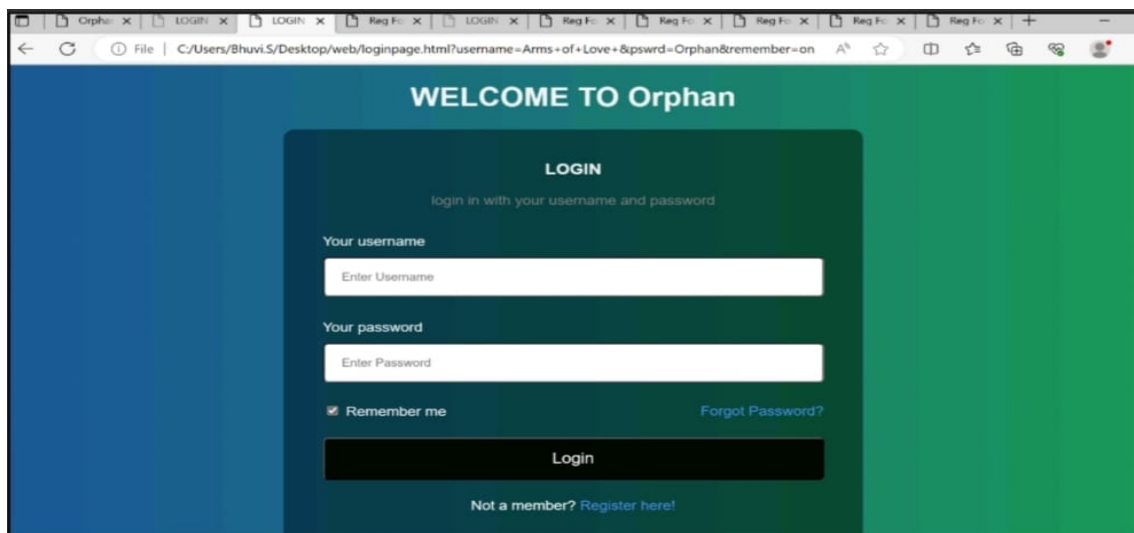


Figure 3.4: Signup and Login page

Figure 3.2 If we don't have credentials then we have to create by using signup option. Here we have to enter email id, username and password for password we had some limitations to create.

3.3.3 Form Validation

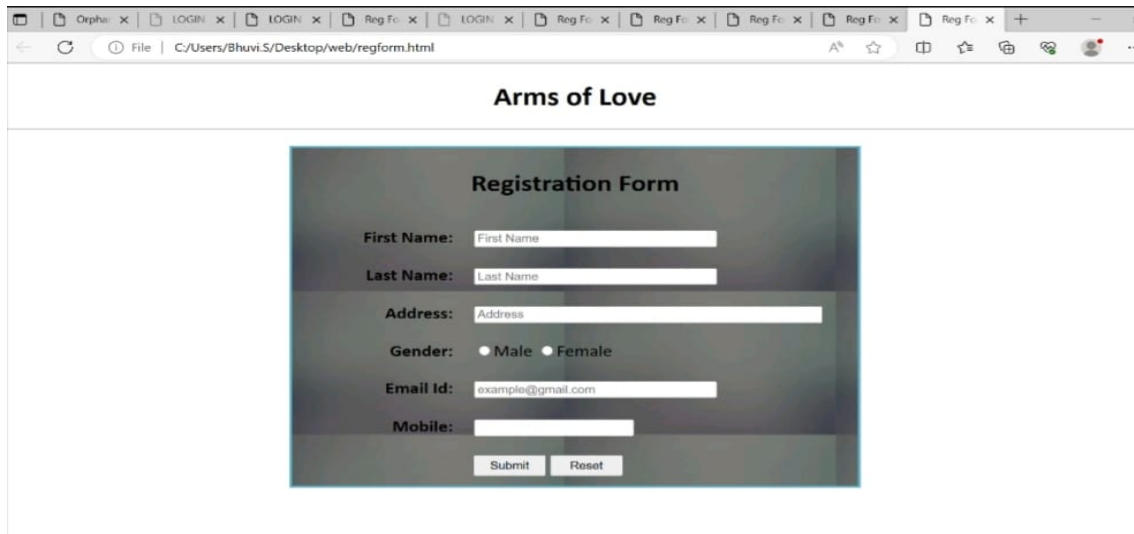
A screenshot of a web browser window displaying a registration form. The browser's address bar shows the file path 'C:/Users/Bhuv/S/Desktop/web/regform.html'. The page has a title 'Arms of Love' and a central 'Registration Form' box. The form contains fields for 'First Name', 'Last Name', 'Address', 'Email Id' (with the placeholder 'example@gmail.com'), and 'Mobile'. There are radio buttons for 'Gender' with 'Male' selected. At the bottom of the form are 'Submit' and 'Reset' buttons. The browser's tab bar shows multiple open tabs, including 'Orphe...', 'LOGIN', and several 'Reg Fo...' tabs.

Figure 3.5: Form Validation

Figure 3.3 This is our form validation page here we have to enter our credentials and register, if we don't have credentials then create by using signup option

Form validation is a critical process in web development that ensures the accuracy and integrity of data submitted through online forms. It encompasses a range of techniques and checks that help verify whether the information provided by users meets predefined criteria. This process occurs both on the client-side, within the user's browser, and on the server-side, after the data is submitted. On the client-side, validation is typically performed using front-end technologies like HTML, CSS, and JavaScript, providing immediate feedback to users as they fill out the form. This helps prevent common errors and guides users in providing correct information. Server-side validation, on the other hand, serves as an additional layer of security, confirming the validity of data after it has been submitted. This is particularly crucial for safeguarding against potentially malicious or incorrect inputs. By combining client-side and server-side validation techniques, developers can ensure that the data collected through forms is accurate, reliable, and secure, ultimately enhancing the overall functionality and usability of web applications.

3.3.4 Parse the webpage using JQuery and DOM

By using jQuery along with the DOM, you can efficiently interact with and manipulate web pages, making it a powerful tool for web development. Keep in mind that jQuery is just one of many libraries available for this purpose, and modern web development often favors native JavaScript methods and frameworks like React or Vue.js

3.3.5 Creation of Webserver using Node Js

Creating a web server for an orphanage management system using Node.js involves several steps that leverage the power of JavaScript on the server-side. Node.js provides a runtime environment that allows developers to build efficient, scalable, and high-performance servers. Here's a detailed guide: To begin, ensure Node.js is installed on your system. You can download it from the official website. Once installed, create a new directory for your project and navigate to it in your terminal or command prompt. This will serve as the root directory for your project. Next, initialize a new Node.js project using the command `npm init -y`. This command creates a `package.json` file, which contains metadata about your project and its dependencies. The `package.json` file is crucial for managing project dependencies and settings. With the project initialized, you'll need to install the necessary packages. For building a web server, you'll likely want to use a framework like Express. Express is a minimalistic and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. Install Express by running the command `npm install express --save`. This command downloads and installs Express and adds it as a dependency in your `package.json` file. Now, it's time to create the server file. Create a file called `server.js` (or any other name you prefer) in your project directory. This file will contain the code for your web server.

3.3.6 Design of Three Tier application using Node js and MySQL

Designing a Three-Tier application using Node.js and MySQL for an orphanage management system involves a structured approach to building a robust and scalable system. This architecture separates the application into three distinct layers: Presentation Layer, Business Logic Layer, and Data Access Layer. Let's delve into the details of each layer:

1. Presentation Layer: The Presentation Layer represents the user interface and is the topmost tier of the application. It handles user interactions, displaying information, and collecting user inputs. In a web-based cricket management system, this layer comprises HTML, CSS, and JavaScript code that forms the user interface. Within this layer, you'd design web pages for tasks such as browsing available tours, making bookings, user registration and authentication, viewing score, and providing feedback. User experience (UX) considerations are crucial in this tier to ensure that the system is intuitive and user-friendly. Interactive elements like forms, buttons, navigation menus, and search bars are implemented here. JavaScript frameworks like React or Vue.js can be used to enhance interactivity and provide a smoother user experience.

2. Business Logic Layer: The Business Logic Layer is the heart of the application. It contains the application's core logic, algorithms, and rules. In a cricket management system, this layer processes

user inputs, applies business rules, and orchestrates interactions between the Presentation Layer and the Data Access Layer. Using Node.js, this tier handles tasks such as processing ground booking requests, calculating prices, managing user accounts, validating user inputs, and enforcing business rules. This layer ensures that the system behaves correctly and consistently.

3. Data Access Layer: The Data Access Layer is responsible for managing the storage and retrieval of data. It interacts directly with the database to perform operations like querying, inserting, updating, and deleting data. In this case, MySQL is utilized as the database management system. Within this layer, you would design and manage the database schema, which includes tables for storing information about tours, bookings, user accounts, feedback, and more. SQL queries are used to interact with the database, ensuring that data is stored and retrieved accurately. Node.js is employed to establish a connection between the application and the MySQL database. Libraries like `mysql2` or ORM (Object-Relational Mapping) frameworks like `Sequelize` can be used to simplify database interactions and improve security.

4. Communication Between Layers: Each layer communicates with the adjacent layers through well-defined interfaces. For instance, the Presentation Layer sends user inputs and requests to the Business Logic Layer. The Business Logic Layer processes these requests, performs necessary computations, and may retrieve or update data from the Data Access Layer. Responses and results are then passed back through the layers to reach the Presentation Layer, which displays the information to the user. This Three-Tier architecture provides modularity, maintainability, and scalability to the application. It allows for easy modification or replacement of individual layers without affecting the others. Additionally, it supports parallel development by allowing different teams to work on different layers simultaneously.

3.3.7 Design of Reactive form for User Registration using Angular

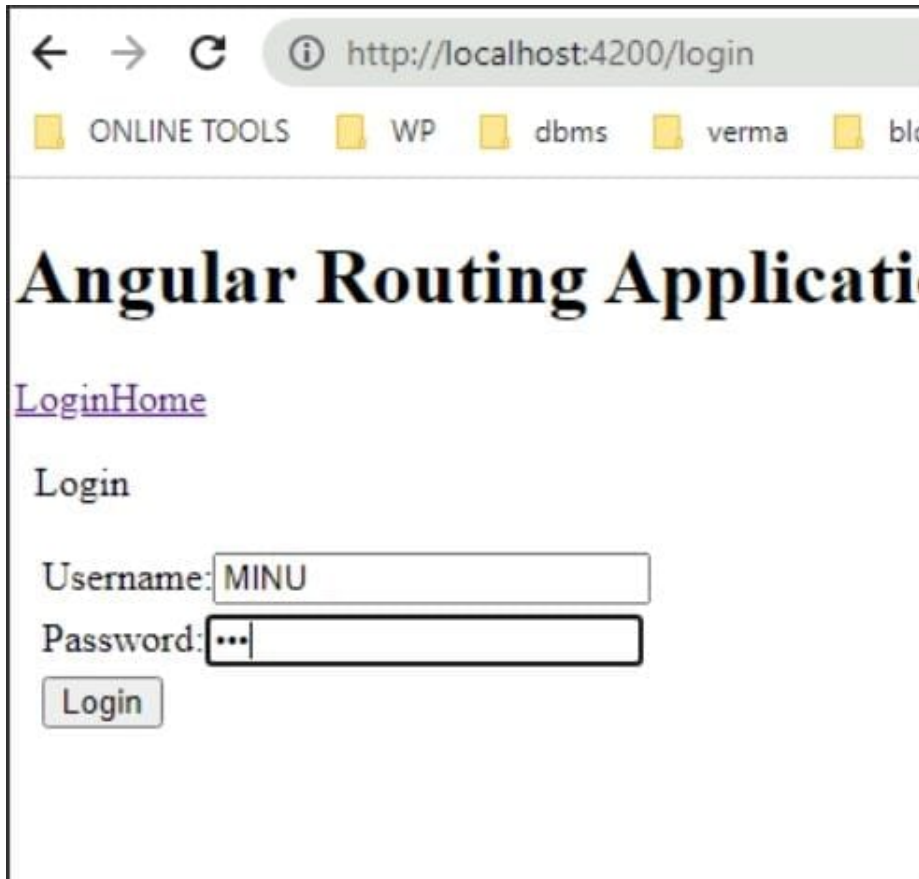
The screenshot displays a web browser at <http://localhost:4200> with a registration form titled "Registration Form". The form includes fields for First Name (minu), Last Name (ben), Email (minu@gmail.com), and Country (Canada). It also features radio buttons for Gender (Male/Female) and checkboxes for Hobbies (Sports, Movies, Music). A green "Submit" button is at the bottom.

The browser's developer console shows the state of the `FormGroup` object after submission. The `value` property contains the following data:

```
*** websiteRoot ***
FormGroup {
  _pendingDirty: false,
  _hasOwnPendingAsyn...
  _updateDomValue: f, ...
  controls: {
    firstname: FormControl,
    lastname: ...
  },
  errors: null,
  pristine: false,
  status: "VALID",
  statusChanges: EventEmitter_ {
    closed: false,
    cu...
  },
  touched: true,
  value: {
    country: "canada",
    email: "minu@gmail.com",
    firstname: "minu",
    gender: "female",
    hobbies: true,
    lastname: "ben",
    [[Prototype]]: Object
  },
  valueChanges: EventEmitter_ {
    closed: false,
    cu...
  },
  _composedAsyncValidatorFn: null,
  _composedValidatorFn: null,
  _hasOwnPendingAsyncValidator: false,
  _onCollectionChange: () => this._updateDomValue,
  _onDisabledChange: [],
  _parent: null,
  _pendingDirty: false,
  _pendingTouched: false,
  _rawAsyncValidators: null,
  _rawValidators: null,
  asyncValidator: (...),
  dirty: (...),
  disabled: (...),
  enabled: (...),
  invalid: (...),
  parent: (...),
}
```

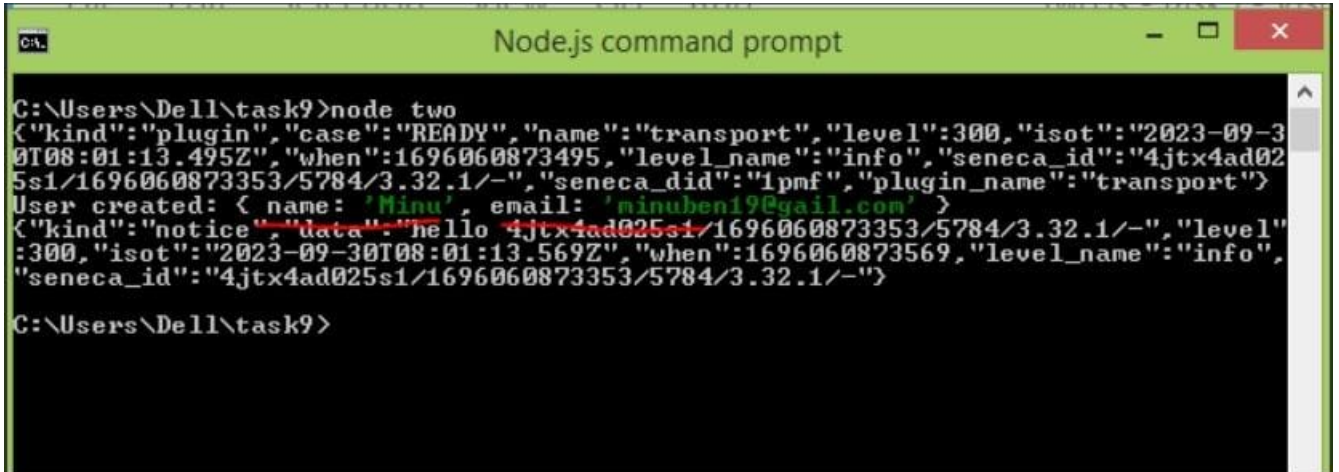
Creating a registration form in Angular involves setting up an Angular project, creating a registration component, and defining the form structure in HTML. Two-way data binding is established using Angular's `ngModel`, and form validation is implemented with directives like `required` and `email`. The form submission is handled in the component, allowing for data processing or server requests. Styling and routing can be customized as needed. This registration form can be integrated with backend services for user data storage, and the entire Angular application can be deployed for public access. Customization and validation depend on specific project requirements.

3.3.8 Develop web application to implement routing and navigation in Angular



To create a web application with routing and navigation in Angular, you'll first set up your Angular project using the Angular CLI. Then, create the necessary components and define routes in the `app-routing.module.ts` file, associating URL paths with corresponding components. In your component templates, use the `routerLink` directive to create navigation links between components. To programmatically navigate, employ Angular's Router service. Implement route guards for controlling access and data resolution, and consider lazy loading for larger applications to enhance performance. Handle route parameters and query parameters as needed and set up scenarios.

3.3.9 Creation of Microservices



```
Node.js command prompt

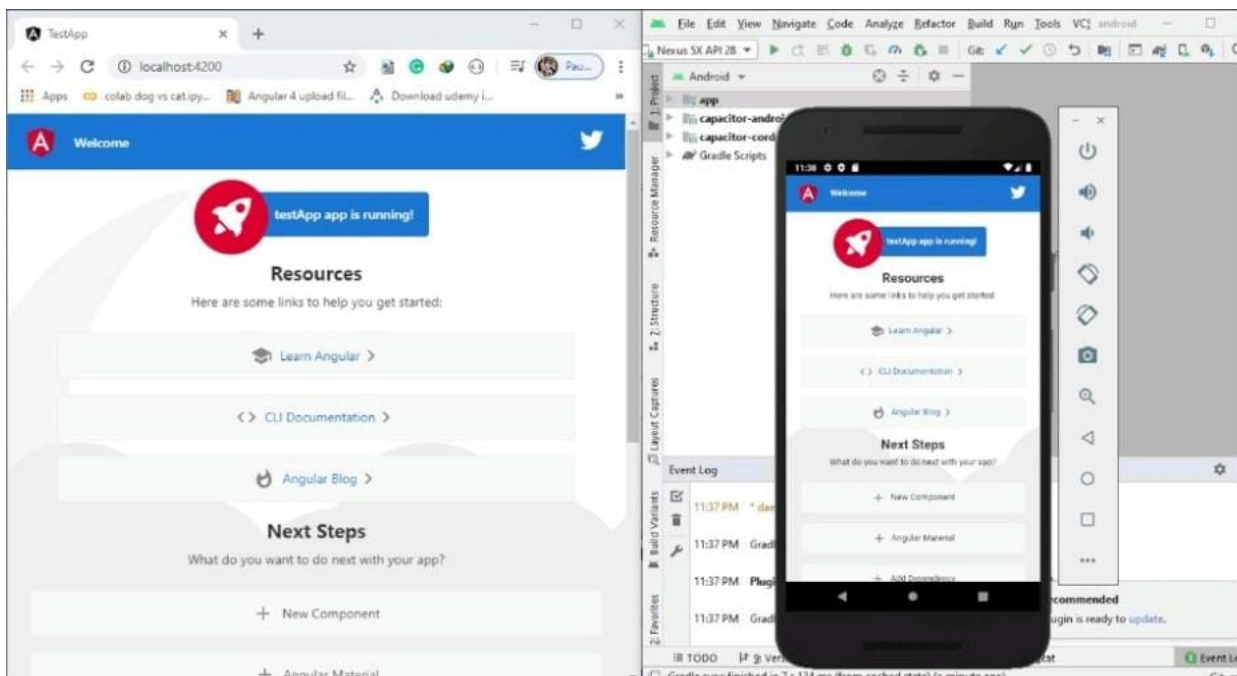
C:\Users\Dell\task9>node two
{"kind":"plugin","case":"READY","name":"transport","level":300,"isot":"2023-09-30T08:01:13.495Z","when":"1696060873495","level_name":"info","seneca_id":"4jtx4ad025s1/1696060873353/5784/3.32.1/-","seneca_did":"1pmf","plugin_name":"transport"}
User created: { name: 'Minu', email: 'minuben19@gmail.com' }
{"kind":"notice","data":"hello 4jtx4ad025s1/1696060873353/5784/3.32.1/-","level":300,"isot":"2023-09-30T08:01:13.569Z","when":"1696060873569","level_name":"info","seneca_id":"4jtx4ad025s1/1696060873353/5784/3.32.1/-"}

C:\Users\Dell\task9>
```

To develop a micro service to add and multiply 2 number in Node.js using Seneca Toolkit, follow these steps to create microservice

- Install Seneca module
- Create ur own folder
- Launch VS code through cmd and type the code

3.3.10 Conversion of Webpage to App



Building the Web application and convert it into Mobile app, Create the angular application and convert them by following the steps:

- install capacitor package using `npm i @capacitor/core`

- install capacitor cli using `npm i -D @capacitor/cli`
- Initialize your Capacitor config using `npx cap init`
- change the webDir value as “WWW” in capacitor.config.ts file
- change the outputpath value as WWW in angular.json file
- install `@capacitor/android`
- Open Android studio application
- `npx cap run android`

Chapter 4

TESTING

Testing an orphanage management website is crucial to ensure it functions properly, is user-friendly, and provides a safe and secure environment for managing the operations of the orphanage.

4.1 Testing

Here's a structured approach to testing such a website:

1. Requirements Analysis:

Understand the requirements of the orphanage management website. Identify key functionalities like user registration, child profiles, staff management, donation tracking, etc.

2. Test Planning:

Define test objectives, scope, and resources needed. Create a test plan outlining test cases and schedules.

3. Functional Testing:

User Management: Verify that users can register, login, and logout successfully. Check for password reset functionality. Child Profiles: Add, edit, and delete child profiles. Verify that all fields are functional. Check for age validation and photo uploads. Staff Management: Add, edit, and delete staff profiles. Validate all required fields. Donation Tracking: Test donation registration, viewing, and reporting. Event Management: Verify the functionality for creating and managing events. Reports: Check if generated reports (like financial summaries) are accurate. Search Functionality: Ensure that search filters and options work correctly.

4. Security Testing:

Check for vulnerabilities such as SQL injection, XSS, CSRF, etc. Ensure user data is encrypted and protected. Verify access controls to sensitive information.

5. Usability Testing:

Evaluate the user interface for intuitiveness and ease of navigation. Check if buttons, links, and forms work as expected.

6. Compatibility Testing:

Test the website on different browsers (Chrome, Firefox, Safari, etc.). Ensure it's responsive on various devices (desktops, tablets, mobile phones).

7. Performance Testing:

Check for page load times, especially for pages with heavy content. Test under different network conditions (3G, 4G, etc.).

Integration Testing: Test interactions between different modules (e.g., user profiles, child profiles, donations).

9. Regression Testing:

After making changes or updates, re-test critical functionalities to ensure they still work.

10. Accessibility Testing:

Verify compliance with accessibility standards (WCAG) for people with disabilities.

4.1.1 Test Result

Detailed Results:

User Management: Register User: Passed Login: Passed Logout: Passed Password Reset: Passed

Donation Tracking: Record Donation: Passed View Donations: Passed Generate Donation Report: Passed

Event Management: Create Event: Passed Edit Event: Passed Delete Event: Passed

Reports: Financial Summary: Passed Child Profile Summary: Passed

Search Functionality: Basic Search: Passed Advanced Search: Passed

Security Testing: SQL Injection: Passed (No vulnerabilities found) XSS: Passed (No vulnerabilities found) CSRF: Passed (No vulnerabilities found)

Usability Testing: Intuitiveness: Passed Navigation: Passed Button and Form Functionality: Passed

Compatibility Testing: Chrome: Passed Firefox: Passed Safari: Passed

4.1.2 Test Bugs

Bug 1: Registration Page - Validation Error

Description: When attempting to register a new user, if an invalid email address is provided, the system accepts it without displaying an error message.

Expected Behavior: The system should validate email addresses and display an error message if an invalid format is detected.

Bug 2: Child Profile - Age Validation Issue

Description: When trying to add a child with an age below 0 or above 18, the system allows it without raising an error.

Expected Behavior: The system should enforce age limits and display an error message for invalid ages.

Bug 3: Donation Tracking - Incorrect Currency Display

Description: Donations made in different currencies are not displayed correctly in the reporting section. For example, a donation made in Euros is shown as USD.

Expected Behavior: Donations should be displayed in their respective currencies accurately.

Bug 4: Security Vulnerability - Missing Input Sanitization

Description: The system is vulnerable to SQL injection attacks. Input fields do not have proper sanitization measures in place.

Expected Behavior: Input fields should be sanitized to prevent SQL injection attacks.

Bug 5: Performance - Slow Loading Times for Donation Report

Description: The donation report page takes an excessive amount of time to load, especially when there are a large number of entries.

Expected Behavior: The donation report page should load within a reasonable time frame, even with a large data set.

Chapter 5

WEBSITE LAUNCH



Figure 5.1: Test Image

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Website performance

Optimizing the performance of an orphanage management website is crucial to provide a seamless and efficient user experience. This involves a range of strategies, from minimizing HTTP requests and optimizing images to utilizing browser caching and reducing server response time. Content Delivery Networks (CDNs) can be employed to distribute content globally, reducing load times for users across different locations. Asynchronous loading of non-essential resources and judicious use of third-party scripts can prevent them from slowing down page rendering. Gzip compression and minification of CSS, JavaScript, and HTML files further reduce data transfer sizes. Critical resources are prioritized for loading, ensuring that above-the-fold content is presented promptly. Regular monitoring of server resources, database optimization, and browser compatibility testing contribute to sustained performance gains. By implementing these measures, the orphanage management website can deliver a swift and responsive experience, regardless of user location or device.

6.2 Security

Ensuring robust security measures for an orphanage management website is paramount to safeguard sensitive information and protect against potential threats. Implementing strong authentication mechanisms, including password policies and multi-factor authentication, helps fortify user accounts. Encryption protocols, such as HTTPS, should be utilized to secure data in transit and prevent eavesdropping. Regular security audits and vulnerability assessments are essential to identify and patch potential weaknesses. Input validation and output encoding must be enforced to thwart common attacks like SQL injection and cross-site scripting. Access controls should be rigorously managed, ensuring that only authorized personnel have access to sensitive information. Regular security updates and patches must be applied to all components of the website, including plugins and frameworks. Additionally, continuous monitoring and intrusion detection systems can help detect and respond to suspicious activities in real-time. Data backups and disaster recovery plans should be in place to mitigate the impact of any unforeseen security incidents.

6.3 Responsiveness and mobile-friendliness

Ensuring the responsiveness and mobile-friendliness of an orphanage management website is vital for providing an inclusive and accessible user experience. A responsive design ensures that the website adapts seamlessly to various screen sizes, including desktops, tablets, and mobile phones. This means that users can access and navigate the site effectively regardless of the device they are using. Elements like buttons, forms, and text should be appropriately sized and spaced to accommodate touch gestures on mobile devices. Additionally, images and media should be optimized to load quickly without compromising quality. Mobile-friendly design also entails ensuring that content is organized intuitively, and menus are easy to navigate on smaller screens. By employing a responsive and mobile-friendly approach, the orphanage management website can reach a wider audience, including those who primarily access the internet through mobile devices, and offer a user experience that is seamless and intuitive across all platforms.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The development of an orphanage website is a significant step towards providing support and assistance to orphans and those involved in orphan care. It serves as a crucial platform for connecting potential donors, volunteers, and adoptive parents with the orphanage. The website provides a global platform for the orphanage to reach a wider audience, which can lead to increased support and resources. It enables effective communication between the orphanage, donors, volunteers, and potential adoptive parents. This can lead to more streamlined operations and improved support for the orphans. A well-designed website with clear information about the orphanage's mission, activities, and financial transparency builds trust with stakeholders. The website can be a powerful tool for fundraising, allowing for online donations, sponsorship programs, and other fundraising activities. It encourages individuals to get involved by offering their time, skills, or resources, thereby strengthening the orphanage's operations. The website can serve as a platform for educating the public about issues related to orphans, adoption, and child welfare.

7.2 Future Enhancements

1. Interactive Features: Implement interactive features like live chat support, discussion forums, or a blog for regular updates and engagement with the community.

2. Multilingual Support: If the orphanage operates in a region with multiple languages, consider adding multilingual support to make the website accessible to a wider audience.

3. Event Management System: Incorporate an event management system for organizing and promoting fundraising events, workshops, and awareness campaigns.

4. Integration with Social Media: Integrate social media platforms to leverage their reach for wider visibility and to encourage sharing of relevant content.

5. Mobile App Integration: Consider developing a mobile app for better accessibility, especially

for those who prefer mobile browsing.

6. User Accounts and Personalization: Allow users to create accounts for personalized experiences, tracking donations, and managing their involvement with the orphanage.

7. Feedback and Survey Forms: Include feedback forms and surveys to gather input from users for continuous improvement.

8. Security Measures: Implement robust security measures to protect sensitive information, especially if the website handles donations and personal data.

9. Collaborations and Partnerships: Highlight any partnerships or collaborations with other organizations, as this can increase credibility and support.

10. Accessibility Compliance: Ensure the website is accessible to individuals with disabilities, adhering to WCAG guidelines.

Ongoing updates and maintenance are crucial to keep the website relevant and effective in achieving the orphanage's goals. Regularly assess the website's performance and gather feedback from users for continuous improvement.

Chapter 8

SOURCE CODE

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Orphanage Website</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="main">
      <div class="navbar">
        <div class="icon">
          <h2 class="logo">ArmsOfLove</h2>
        </div>
        <div class="menu">
          <ul>
            <li><a href="#">HOME</a></li>
            <li><a href="#">ABOUT</a></li>
            <li><a href="Login.html">Login</a></li>
            <li><a href="#">SERVICE</a></li>
            <li><a href="#">HISTORY</a></li>
            <li><a href="#">CONTACT</a></li>
          </ul>
        </div>
      </div>
    </div>
  </body>
</html>
```


login.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Orphanage Website</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="main">

<div class="submit">
  <input class="srch" placeholder="Type Enter Username" name="
    Username" required>
  <label>Password:</label>
  <input class="Password" placeholder="Enter Password" name="
    Password" required>
  <button class="submit">Login</button>
  <a href="#"><button class="btn">Submit</button></a>
</div>
</div>
```

style.css

```
*{
  margin: 0;
  padding: 0;
}

.main{
  width: 100%;
  background: linear-gradient(to top, rgba(0,0,0,0.5) 50%,rgba
    (0,0,0,0.5) 50%),url(orphan.jpg);
```

```
background-position: center;
background-size: cover;
height: 109vh;
}

.navbar{
width: 1200px;
height: 75px;
margin: auto;
justify-content: space-between;
}

.icon{
width: 200px;
float: left;
height: 70px;
}

.logo{
color: white;
font-size: 35px;
font-family: Arial;
padding-left: 20px;
float: left;
padding-top: 10px;
}

.menu{
width: 400px;
float: left;
height: 70px;
}

ul{
```

```

        float: left;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    ul li{
        list-style: none;
        margin-left: 62px;
        margin-top: 27px;
        font-size: 14px;
    }

    ul li a{
        text-decoration: none;
        color: white;
        font-family: Arial;
        font-weight: bold;
        transition: 0.4s ease-in-out;
    }

    ul li a:hover{
        color: aqua;
    }

    .search{
        width: 330px;
        float: left;
        margin-left: 270px;
    }

    .srch{
        font-family: 'Times New Roman';
        width: 200px;
    }

```

```

    height: 40px;
    background: transparent;
    border: 1px aqua;
    margin-top: 13px;
    color: rgb(30, 160, 186);
    border-right: none;
    font-size: 16px;
    float: left;
    padding: 10px;
    border-bottom-left-radius: 5px;
    border-top-left-radius: 5px;
}
.btn{
    width: 100px;
    height: 40px;
    background: rgb(239, 241, 241);
    border: 2px solid aqua;
    margin-top: 13px;
    color: rgb(9, 9, 9);
    font-size: 15px;
    border-bottom-right-radius: 5px;
    border-bottom-right-radius: 5px;
}

.btn:focus{
    outline: none;
}

.srch:focus{
    outline: none;
}

.form{

```

```

width: 250px;
height: 380px;
background: linear-gradient( to top,rgba(0,0,0,0.8)50%,rgba
    (0,0,0,0.8)50%);
position: center;
top: -20px;
right: 870px;
border-radius: 10px;
padding: 25px;
justify-content: end;
text-align: right top ;
}

.form h2{
width: 220px;
font-family: sans-serif;
text-align: center;
color: aqua;
font-size: 22px;
background-color: white;
border-radius: 10px;
margin: 2px;
padding: 8px;
justify-content: center;
}

.form input{
width: 240px;
height: 35px;
background: transparent;
border-bottom: 1px solid aqua;
border-top: none;
border-right: none;
border-left: none;

```

```

    color: white;
    font-size: 15px;
    letter-spacing: 1px;
    margin-top: 30px;
    font-family: sans-serif;
}

.form input:focus{
    outline: none;
}

::placeholder{
    color: white;
    font-family: Arial;
}

```

validate.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Orphanage Website</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="container">
      <div class="main">
        <h3>Login</h3><hr/>
        <form id="form_id" method="post" name="myform">
          <strong>Username:</strong></form></br>
          <input type="text" name="username" id="username"/><
            /br>
          <strong>Password:</strong></br>
          <input type="password" name="password" id="password

```

```
        "/></br>
    <input type="button " value="login " id="submit"
        onclick="validate()" ">
</form>
<span><b class="note" note:</b><strong>For this demo
    use following username and password.<br/><b class="
    valid"><h4>User name :Arms of Love<br/>Password :
    Orphan</b></h4></span></strong>
</div>
<div class="fugo">
    <a href="orphan.jpg"></a>

</div>
</div>
```

Chapter 9

SCREENSHOTS

9.0.1 Login Page and Signup Page

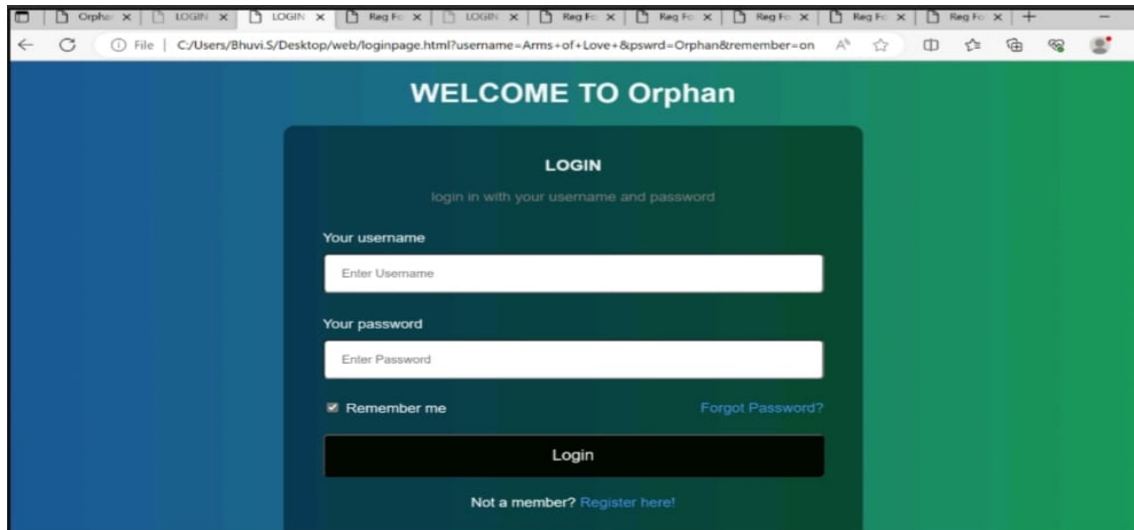


Figure 9.1 Admin and Agency have separate Login forms to access their accounts. Admin and Agency have different functionalities and options

9.0.2 Form Validation Page

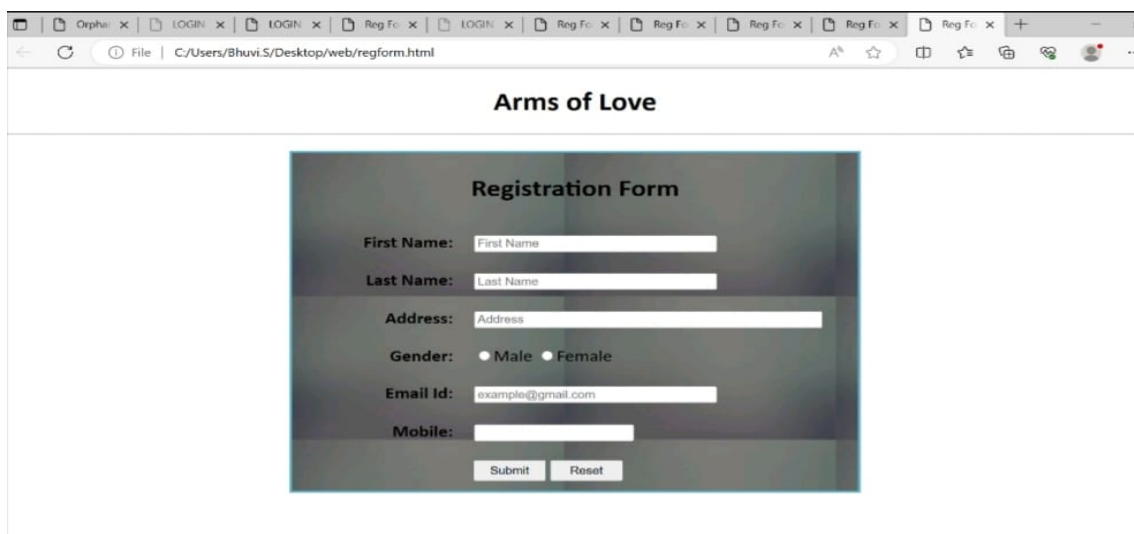


Figure 9.3 This is our form validation page here we have to enter our credentials if we don't have credentials then create by using signup option

9.0.3 Home Page

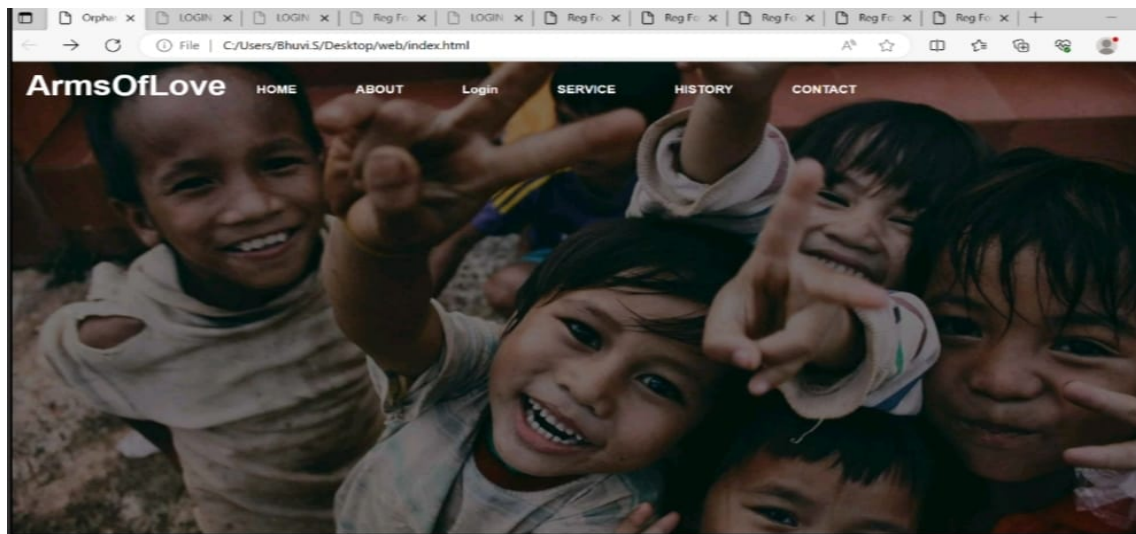


Figure 9.4 This is the Home Page where we can see the home, about, login, service, history, contact. We can click on navigate button then it will re-direct page to the respective page