**An online Laundry Provider Platform: A Case Study Of Fua Hub**

**SYSTEM DOCUMENTATION**

**MUTURI MERCY WANJIKU**

**DSE-01-0078/2022**

**A SYSTEM DOCUMENTATION SUBMITTED TO THE DEPARTMENT OF ICT, MEDIA AND ENGINEERING IN PARTIAL FULFILMENT FOR THE AWARD OF A DIPLOMA IN INFORMATION TECHNLOGY BY ZETECH UNIVERSITY**

**Dec, 2024**

**DECLARATION**

I hereby certify that content of this report was not derived from the internet, AI assistants, other students or students from previous semesters of Zetech University and or any other university and it only includes work done solely with input from the unit instructor.

|  |  |  |
| --- | --- | --- |
| Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Supervisor Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**ACKNOWLEDGMENT**

My appreciation goes to Mr Gitonga Munyi for guiding me during the semester and the useful advice and recommendations for change where necessary where I sought individual direction from him. I would also want to thank Chris Mwiti for helping me develop my database. My friends Mike, Steve for helping me out through consultation .

Contents

[CHAPTER ONE 11](#_Toc184022198)

[SYSTEM OVERVIEW 11](#_Toc184022199)

[1**.**1Statement of Problem 11](#_Toc184022200)

[1.2 **System Justification** 11](#_Toc184022202)

[1.3 System Objectives 11](#_Toc184022204)

[1.3.1 General Objectives 11](#_Toc184022205)

[1.3.2Specific Objectives 12](#_Toc184022206)

[CHAPTER TWO: FRONT END 13](#_Toc184022207)

[2.1 Introduction to User Interface 13](#_Toc184022208)

[2.2 Login Page 13](#_Toc184022209)

[2.2.2 User Dashboard 14](#_Toc184022210)

[2.4 Conclusion to Interfaces 20](#_Toc184022211)

[CHAPTER THREE: BACK END 21](#_Toc184022212)

[(DATABASE AND LOGIC) 21](#_Toc184022213)

[3.1 Introduction 21](#_Toc184022214)

[3.2 Database Implementation 21](#_Toc184022215)

[3.2.1 Setting up SQL Database 22](#_Toc184022216)

[1. Installing MySQL 22](#_Toc184022217)

[ Download and Install MySQL: 22](#_Toc184022218)

[i. Go to the MySQL website. 22](#_Toc184022219)

[ii. Download the MySQL Community Server installer and follow the setup wizard. 22](#_Toc184022220)

[iii. During installation, choose "Developer Default" and configure a strong root password. 22](#_Toc184022221)

[ Create a New Database: 22](#_Toc184022222)

[I. Open the MySQL Workbench. 22](#_Toc184022223)

[II. Connect to your local server using the root credentials. 22](#_Toc184022224)

[III. Run the following query to create the database: 22](#_Toc184022225)

[sql 22](#_Toc184022226)

[Copy code 22](#_Toc184022227)

[CREATE DATABASE FuaHub; 22](#_Toc184022228)

[ Set Up Tables: 22](#_Toc184022229)

[o Use SQL scripts to create required tables like users, service\_requests, and service\_providers. 22](#_Toc184022230)

[2. Installing MySQL Connector in PyCharm 22](#_Toc184022231)

[ Install Connector Package: 22](#_Toc184022232)

[o Open PyCharm, navigate to File > Settings > Project: YourProjectName > Python Interpreter. 22](#_Toc184022233)

[o Click the "+" icon, search for mysql-connector-python, and install it. 22](#_Toc184022234)

[ Verify Installation: 22](#_Toc184022235)

[o Run the following Python script to confirm the connector works: 22](#_Toc184022236)

[python 22](#_Toc184022237)

[import mysql.connector 22](#_Toc184022238)

[print("MySQL Connector Installed Successfully!") 22](#_Toc184022239)

[3. Configuring the Database in PyCharm 22](#_Toc184022240)

[ Establish a Database Connection: 22](#_Toc184022241)

[o Add the following Python code to connect to your MySQL database: 22](#_Toc184022242)

[ Test Database Connection: 23](#_Toc184022254)

[o Run the script to verify a successful connection to the FuaHub database. 23](#_Toc184022255)

[ Integrate Queries: 23](#_Toc184022256)

[o Use SQL queries within Python scripts to interact with the database for user registration, service requests, and provider matching. 23](#_Toc184022257)

[3.3 Logic 26](#_Toc184022258)

[3.3.2 **Managing Users** 27](#_Toc184022260)

[CHAPTER FOUR: SYSTEM QUALITY TESTS 31](#_Toc184022261)

[4.1 Introduction 31](#_Toc184022262)

[4.2 User Entry and Validation Test (Feature Test) 31](#_Toc184022263)

[4.3 Account Details Validation and Auto-Save Test (Feature Test) 31](#_Toc184022264)

[4.4 Account creation Logic Test (Feature Test) 32](#_Toc184022265)

[4.5 Database Integration Test: Save and Retrieve Events 32](#_Toc184022266)

[4.6 Delete Functionality Test (Feature and DB Test) 33](#_Toc184022267)

[CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS 34](#_Toc184022268)

[Conclusion 34](#_Toc184022269)

[Recommendation 34](#_Toc184022270)

[APPENDICES 35](#_Toc184022271)

[Appendix I: Gantt Chart 35](#_Toc184022272)

[Appendix II: Resource Requirement and Economic Feasibility Assessment 35](#_Toc184022273)

[REFERENCES 36](#_Toc184022274)

**ABSTRACT**

This study aims at exploring how the development of an application that’s going to connect users with knock-one mama fuas who iron and wash cloth manually will solve the problem of inadequate access to efficient service providers through word of mouth which could be appalling. The purpose is the creation of a simple search and booking system for mama fuas based on certain parameters like qualifications, working hours, and preferable location. Literature studies were also done with available service platforms evaluated, user feedback collected, and potential user as well as service providers consulted to establish requirements and areas of concern. To achieve the above objectives, the following sources of data were utilized: industry reports, user feedback, and competitor analysis to understand their needs in the market. The divisions of the paper include problem statement, objectives, system design and the implementation plan. The structured approach used in this case guarantees a consolidated solution on improving the availability and quality of services to users and mama fuas.

**ABBREVIATIONS AND ACRONYMS**

MF- Mama fua

MFs-Mama fua’s

ID-Identification Number

**DEFINITION OF KEY TERMS**

**Clients-** people who give jobs to mama fuas

**Service providers-** This are the personnel who provide services

**Location-** where the client lives or within the proximity

**Description-** how the task will be handled

**Booked-** when the service provider is occupied

**Not booked-** when the service provider is not occupied

# CHAPTER ONE

# SYSTEM OVERVIEW

## 1**.**1Statement of Problem

## The current process of accessing manual laundry services, provided by "Mama Fua" workers, relies heavily on informal, word-of-mouth referrals. This system is inefficient, as clients face challenges in finding available and reliable providers, leading to inconsistent service quality and unmet needs. Additionally, service providers often experience exploitation, lack of proper recognition, and limited access to clients beyond their local circles. These inefficiencies result in dissatisfaction and hinder the growth of the manual laundry service industry. To address these issues, a structured, digital solution is necessary to bridge the gap between clients and service providers effectively and transparently.

## 1.2 **System Justification**

## The proposed online platform for "Mama Fua" services aims to address existing gaps by creating a digital marketplace that connects clients with verified service providers. This system simplifies service access, allowing clients to search for providers based on location, availability, and expertise. Features such as detailed profiles, ratings, and secure booking ensure reliability and satisfaction. For service providers, the platform offers opportunities to expand their client base, secure consistent work, and receive fair treatment. By digitizing this informal sector, the system promotes trust, efficiency, and professionalism, ultimately benefiting both clients and workers while enhancing the broader gig economy.

## 1.3 System Objectives

### General Objectives

The overall objective of this project is to develop an online laundry service platform.

### 1.3.2Specific Objectives

* 1. To develop an interface where a client can get information about the service provider and get in touch with them.
  2. To develop a front end interface where a service provider can create an account
  3. To develop a back end where the admin can verify an account and update its status to indicate whether it is pending, active, suspended or deleted.
  4. **Functional Requirements**
  5. **Bookings table**

|  |  |  |
| --- | --- | --- |
| **Column name** | **Data type** | **constraint** |
| **id** | **INT** | **PRIMARY KEY** |
| **Us**er-id | **INT** | **FOREIGN KEY** |
| **Quantity** | **INT** | **Foreign key** |
| **Service id** | **INT** | **NOT NULL** |
| **Total price** | **DECIMAL** | **NOT NULL** |
| **Status** | **VARCHAR** | **DEFAULT** |
| **Created at** | **TIMESTAMP** |  |

* 1. **Users Table**

|  |  |  |
| --- | --- | --- |
| **User** | **User Activities** | **Features** |
| User Registration | Enable users (clients and service providers) to create accounts. | Clients: Name, email, location.  - Providers: Name, contact, experience, profile photo, specialties. |
| Profile Management | Allow clients and providers to manage their profiles. | Update personal details.  - Display skills, experience, availability. |
| Service Search | Facilitate clients to search for "Mama Fua" providers. | Search by location, availability, specialization, and ratings |
| Booking System | Enable clients to book services directly via the platform. | - Real-time booking with provider’s schedule.  - Immediate or scheduled bookings |
| Validation & Verification | Verify service providers’ identities and qualifications. | ID verification, background checks.  - Status updates (active, pending, suspended). |

# CHAPTER TWO: FRONT END

## 2.1 Introduction to User Interface

This chapter focuses on different pages and screens of the system the user interacts with in the system.

## 2.2 Login Page

This page will be used by every user to access the system.



*Below is a html code from the login page that was generated by the code above*

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<meta name="viewport" content="width=device-width, initial-scale=1.0">*

*<title>Login - Fua Hub</title>*

*<link rel="stylesheet" href="styles.css"> <!-- Link your CSS file -->*

*</head>*

*<body>*

*<header>*

*<h1>Fua Hub</h1>*

*<nav>*

*<a href="index.html">Home</a>*

*<a href="about.html">About Us</a>*

*<a href="account.html">Account</a>*

*<a href="contact.html">Contact Us</a>*

*<a href="services.html">Services</a>*

*<a href="signup.html">Sign Up</a>*

*</nav>*

*</header>*

*<main>*

*<h2>Login</h2>*

*<form action="login.php" method="POST">*

*<label for="email">Email:</label>*

*<input type="email" id="email" name="email" placeholder="Enter your email" required>*

*<label for="password">Password:</label>*

*<input type="password" id="password" name="password" placeholder="Enter your password" required>*

*<button type="submit">Login</button>*

*</form>*

*<p>Don't have an account? <a href="signup.html">Sign up here</a>.</p>*

*</main>*

*<footer>*

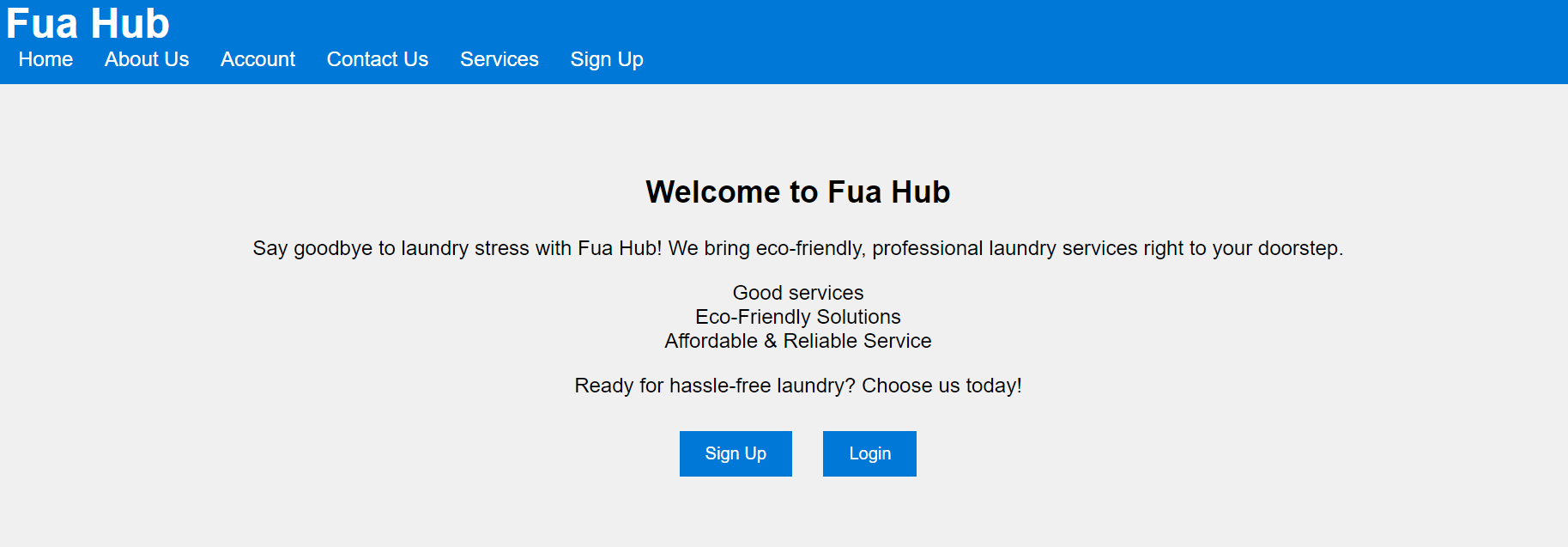
*<p>&copy; 2024 Fua Hub. All rights reserved.</p>*

*</footer>*

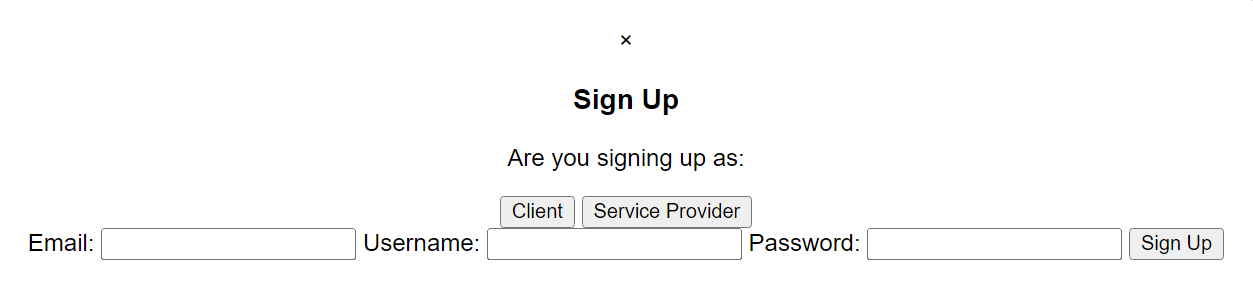
*</body>*

*</html>*

### 2.2.2 User Dashboard

The **Fua Hub Dashboard** provides a user-friendly interface featuring a centralized summary of key metrics, recent activities, and shortcuts for quick navigation to essential features. Its intuitive design ensures seamless access to real-time updates and streamlined management of ongoing tasks. 

*Fig 2.2.2-1 UserDashboard – Users Section*



*The user sign up to be able to access the services on the website tha is wherether they want to be a client or a service provider*

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<meta name="viewport" content="width=device-width, initial-scale=1.0">*

*<title>Sign Up</title>*

*<style>*

*.success {*

*color: green;*

*font-weight: bold;*

*margin-top: 10px;*

*}*

*.error {*

*color: red;*

*font-weight: bold;*

*margin-top: 10px;*

*}*

*</style>*

*</head>*

*<body>*

*<h1>Sign Up</h1>*

*<form action="register.php" method="POST">*

*<input type="text" name="username" placeholder="Username" required>*

*<input type="email" name="email" placeholder="Email" required>*

*<input type="password" name="password" placeholder="Password" required>*

*<select name="role">*

*<option value="client">Client</option>*

*<option value="service\_provider">Service Provider</option>*

*</select>*

*<button type="submit">Sign Up</button>*

*</form>*

*<!-- Success or Error Messages -->*

*<div id="message"></div>*

*<script>*

*// Get URL parameters*

*const urlParams = new URLSearchParams(window.location.search);*

*const success = urlParams.get('success');*

*const error = urlParams.get('error');*

*const messageDiv = document.getElementById('message');*

*if (success) {*

*messageDiv.innerHTML = '<p class="success">Successfully signed up!</p>';*

*} else if (error === 'email\_exists') {*

*messageDiv.innerHTML = '<p class="error">Email already exists. Please use a different email.</p>';*

*} else if (error === 'database\_error') {*

*messageDiv.innerHTML = '<p class="error">An error occurred. Please try again later.</p>';*

*}*

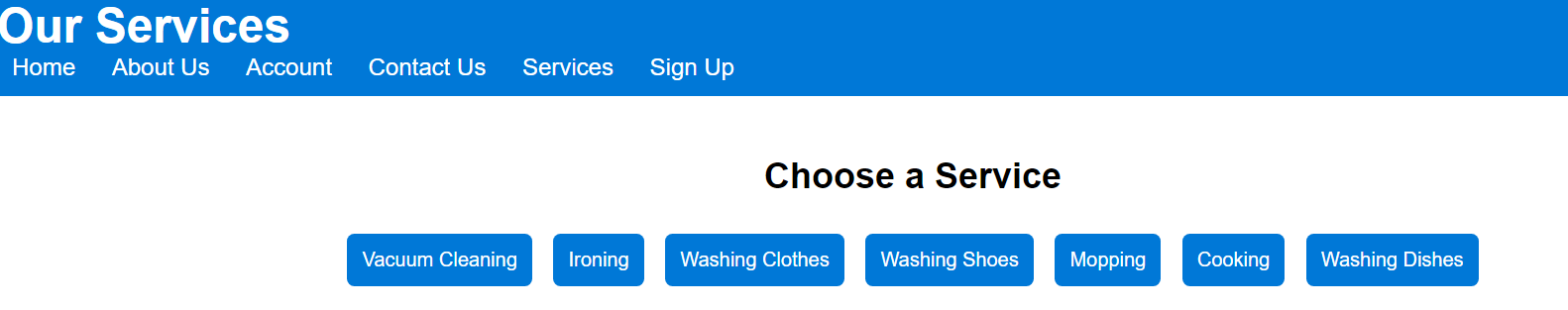
*</script>*

*</body>*

*</html>*

*Sign Up Dashboard*

Services dashboard



*A php code on how the total cost is calculated*

<?php

include 'db\_config.php';

$sql = "SELECT \* FROM services";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

    while ($row = $result->fetch\_assoc()) {

        echo "<div>";

        echo "<h3>" . $row['name'] . "</h3>";

        echo "<p>" . $row['description'] . "</p>";

        echo "<p>Price: Ksh. " . $row['base\_price'] . "</p>";

        echo "</div>";

    }

} else {

    echo "No services available.";

}

$conn->close();

?>

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<meta name="viewport" content="width=device-width, initial-scale=1.0">*

*<title>Services - Fua Hub</title>*

*<link rel="stylesheet" href="styles.css">*

*</head>*

*<body>*

*<header>*

*<h1>Our Services</h1>*

*<nav>*

*<a href="index.html">Home</a>*

*<a href="about.html">About Us</a>*

*<a href="account.html">Account</a>*

*<a href="contact.html">Contact Us</a>*

*<a href="services.html">Services</a>*

*<a href="signup.html">Sign Up</a>*

*</nav>*

*</header>*

*<main>*

*<section class="services">*

*<h2>Choose a Service</h2>*

*<div id="serviceButtons">*

*<button class="service-btn" data-service="vacuum">Vacuum Cleaning</button>*

*<button class="service-btn" data-service="ironing">Ironing</button>*

*<button class="service-btn" data-service="washing\_clothes">Washing Clothes</button>*

*<button class="service-btn" data-service="washing\_shoes">Washing Shoes</button>*

*<button class="service-btn" data-service="mopping">Mopping</button>*

*<button class="service-btn" data-service="cooking">Cooking</button>*

*<button class="service-btn" data-service="washing\_dishes">Washing Dishes</button>*

*</div>*

*<div id="serviceDetails" class="hidden">*

*<h3 id="serviceTitle"></h3>*

*<form id="serviceForm">*

*<div id="options"></div>*

*<div class="quantity">*

*<button type="button" id="subtractBtn">-</button>*

*<input type="number" id="quantity" name="quantity" value="1" min="1">*

*<button type="button" id="addBtn">+</button>*

*</div>*

*<p>Total Cost: Ksh. <span id="totalCost">0</span></p>*

*<button type="button" id="calculateBtn">Calculate Price</button>*

*<button type="submit">Book Service</button>*

*</form>*

*</div>*

*</section>*

*</main>*

*<footer>*

*<p>© 2024 Fua Hub. All rights reserved.</p>*

*</footer>*

*<script src="services.js"></script>*

*</body>*

*</html>*

*Fig 2.3-2 Services Dashboard. html Code*

## 2.4 Conclusion to Interfaces

As earlier stated the login page focus on simplicity, accessibility, and efficiency, ensuring a seamless experience for all users. By implementing a role-specific dashboard structure and intuitive navigation, the system caters to the unique needs of coordinators, teachers, and students while maintaining a cohesive design.

# CHAPTER THREE: BACK END

# (DATABASE AND LOGIC)

## 3.1 Introduction

This chapter focuses on the implementation of database (for data storage and retrieval) and back-end code (ensuring interactivity of the system)

## 3.2 Database Implementation

database implementation consists of three main tables: **users**, **service\_requests**, and **service\_providers**.

This relational structure enables clients to submit service requests, which are matched with available providers based on service type and status. The system supports signup/login functionality, service booking, and payment integration, creating an efficient platform for both clients and providers.

### 3.2.1 Setting up SQL Database

### 1. Installing MySQL

### Download and Install MySQL:

### Go to the [MySQL website](https://dev.mysql.com/downloads/installer/).

### Download the MySQL Community Server installer and follow the setup wizard.

### During installation, choose "Developer Default" and configure a strong root password.

### Create a New Database:

### Open the MySQL Workbench.

### Connect to your local server using the root credentials.

### Run the following query to create the database:

### sql

### Copy code

### CREATE DATABASE FuaHub;

### Set Up Tables:

### Use SQL scripts to create required tables like users, service\_requests, and service\_providers.

### 2. Installing MySQL Connector in PyCharm

### Install Connector Package:

### Open PyCharm, navigate to File > Settings > Project: YourProjectName > Python Interpreter.

### Click the "+" icon, search for mysql-connector-python, and install it.

### Verify Installation:

### Run the following Python script to confirm the connector works:

### python

### import mysql.connector

### print("MySQL Connector Installed Successfully!")

### 3. Configuring the Database in PyCharm

### Establish a Database Connection:

### Add the following Python code to connect to your MySQL database:

### python

### Copy code

### import mysql.connector

### conn = mysql.connector.connect(

### host="localhost",

### user="root",

### password="your\_password",

### database="FuaHub"

### )

### cursor = conn.cursor()

### print("Database Connected Successfully!")

### Test Database Connection:

### Run the script to verify a successful connection to the FuaHub database.

### Integrate Queries:

### Use SQL queries within Python scripts to interact with the database for user registration, service requests, and provider matching.

**Creating and Testing the Tables**

**Step 1: Data Normalization**

To ensure efficient data organization, reduce redundancy, and improve database performance, the tables in the system were normalized using the standard normalization forms:

**1NF (First Normal Form)**

Ensures that each column contains atomic values (indivisible data) and each record is unique.

* + The users table was designed with columns like user\_id, username, email, and password. Each column holds only one value per record.
  + Repeating groups (e.g., multiple emails for a single user) were eliminated.

**2NF (Second Normal Form)**

* Definition: Builds on 1NF by removing partial dependencies, ensuring that all non-primary key attributes are fully dependent on the primary key.
* Implementation:
  + Service details like service\_name, price, and description were moved to a separate services table, linked to service\_requests via foreign keys.
  + This prevents redundancy in storing the same service details for multiple requests.

**3NF (Third Normal Form)**

Ensures no transitive dependency, meaning non-primary key attributes depend only on the primary key.

* + Service provider availability (e.g., status and availability) was placed in the service\_providers table, ensuring no unrelated attributes were linked to a service request.

**Testing the Tables**

* Data Insertion: Tested inserting records into each table, ensuring relationships between tables (e.g., foreign keys) were correctly enforced.
* Data Retrieval: Verified data could be queried efficiently using JOIN statements.
* Updates and Deletions: Tested updates (e.g., changing a service provider’s status) and deletions (e.g., removing a service request) to ensure data integrity.
* Edge Cases: Tested scenarios like duplicate user emails and invalid service IDs to confirm constraints and error handling were functioning as intended.

*Final Tables and Relationships*

***Users Table***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Column Name*** | ***Data Type*** | ***Constraints*** | ***Description*** |
| *id* | *INT* | *PRIMARY KEY, AUTO\_INCREMENT* | *Unique user Id* |
| *username* | *VARCHAR(50)* | *NOT NULL* | *Users name* |
| *email* | *VARCHAR(100)* | *NOT NULL* | *User email address* |
| *Password* | *VARCHAR(255)* | *NOT NULL* | *Hashed password* |
| *role* |  | *NOT NULL* | *role* |
| *Create\_at* | *TIMESTAMP* | *DEFAULT CURRENT\_TIMESTAMP* |  |

*Table 3.2.2-1 Users Table*

*Step 3: Relationships*

*Fig 3.2.2-1 Entity Relationship Diagram*

*Step 4: Creating Tables in PyCharm*

The following steps were taken to create the tables in Pycharm’s Database

1. Access Database Tool Window:
   * Go to *View > Tool Windows > Database.*
   * Click the **+** icon and select *Data Source > MySQL.*
   * Enter the connection details and test the connection.
2. Open SQL Console:
   * Right-click the database node (FuaHubDB) and select *New > Console.*
3. Run SQL Commands:
   * Copy and paste the SQL commands for table creation into the console.
   * Execute them by clicking the sign upbutton.
   * The console will show Query executed successfully for each table created.

Then, the following SQL command codes were written to set up the tables to the DB in Pycharm:

-- Create the database

**CREATE DATABASE FuaHub**;

-- Use the FuaHub database

USE FuaHub;

-- Create the users table

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

role ENUM('client', 'service\_provider') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Create the services table

CREATE TABLE services (

service\_id INT AUTO\_INCREMENT PRIMARY KEY,

service\_name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL

);

-- Create the service\_requests table

CREATE TABLE service\_requests (

request\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

service\_id INT,

request\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('pending', 'completed', 'cancelled') DEFAULT 'pending',

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (service\_id) REFERENCES services(service\_id)

)

-- Create the service\_providers table

CREATE TABLE service\_providers (

provider\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

service\_id INT,

status ENUM('Available', 'Unavailable') DEFAULT 'Available',

price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (service\_id) REFERENCES services(service\_id)

);

*Fig 3.2.2-2 SQL Tables Created*

## 3.3 Logic

This is the backend code that makes the system interactive e.g. data can be correctly saved, and retrieved, buttons are working correctly, inputs and outputs are possible, etc.

The following are samples of logic for different features and functionalities of the system

**3.3.1 Login Functionality**

### The login functionality is designed to securely authenticate users based on their email and password. With proper input validation, session management, and security measures like password hashing, the login system ensures that only legitimate users can access restricted areas of the website. By incorporating good practices, such as preventing SQL injection and handling errors gracefully, the system can maintain both functionality and security

If the details are correct and no empty field, a success message will appear depending on where the program is leading you, whether to the time management dashboard (if it’s a teacher or prefect logging in) or to the coordinator’s dashboard (if it’s the coordinator logging in). In the case below, it’s a teacher who’s successfully logging in:

### 3.3.2 **Managing Users**

The logic for the above is shown below, as extracted from the *UserDashboard .html* file

Managing users effectively is crucial for maintaining the integrity and functionality of your platform. User management can be done by implementing roles, such as 'client' and 'service provider,' to differentiate between the types of users and customize their experiences accordingly. This can be achieved through role-based access control (RBAC), where specific features and permissions are granted based on the user’s role. Administrators should have the ability to manage user accounts, including actions like activating or deactivating accounts, updating user information, and handling user complaints. A robust user management system also involves ensuring secure authentication, such as using hashed passwords, enforcing strong password policies, and incorporating email verification or multi-factor authentication. Regular audits of user activity and data access can help ensure that the platform remains secure and that users’ personal data is protected in compliance with privacy regulations. Additionally, offering a self-service portal for users to update their profiles or request support can improve user satisfaction and streamline the management process.

<?php

session\_start();

include 'db\_config.php';

// Ensure the user is logged in

if (!isset($\_SESSION['user\_id'])) {

    header("Location: login.html");

    exit();

}

$user\_id = $\_SESSION['user\_id'];

// Fetch user details if needed

$sql = "SELECT username, email, role, status FROM users WHERE id = ?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("i", $user\_id);

$stmt->execute();

$stmt->bind\_result($username, $email, $role, $status);

$stmt->fetch();

$stmt->close();

$conn->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Account - Fua Hub</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Fua Hub Account</h1>

        <nav>

            <a href="index.html">Home</a>

            <a href="about.html">About Us</a>

            <a href="services.html">Services</a>

            <a href="contact.html">Contact Us</a>

            <a href="logout.php">Logout</a>

        </nav>

    </header>

    <main>

        <h2>Welcome, <?php echo htmlspecialchars($username); ?>!</h2>

        <p>Email: <?php echo htmlspecialchars($email); ?></p>

        <p>Role: <?php echo ucfirst($role); ?></p>

        <p>Status: <?php echo ucfirst($status); ?></p>

        <a href="account\_status.php">Change Account Status</a>

        <div class="account-buttons">

            <button onclick="location.href='inquire\_service.php'">Customer - Inquire for Service</button>

            <button onclick="location.href='search\_jobs.php'">Mama Fua - Looking for a Job</button>

        </div>

    </main>

    <footer>

        <p>&copy; 2024 Fua Hub. All rights reserved.</p>

    </footer>

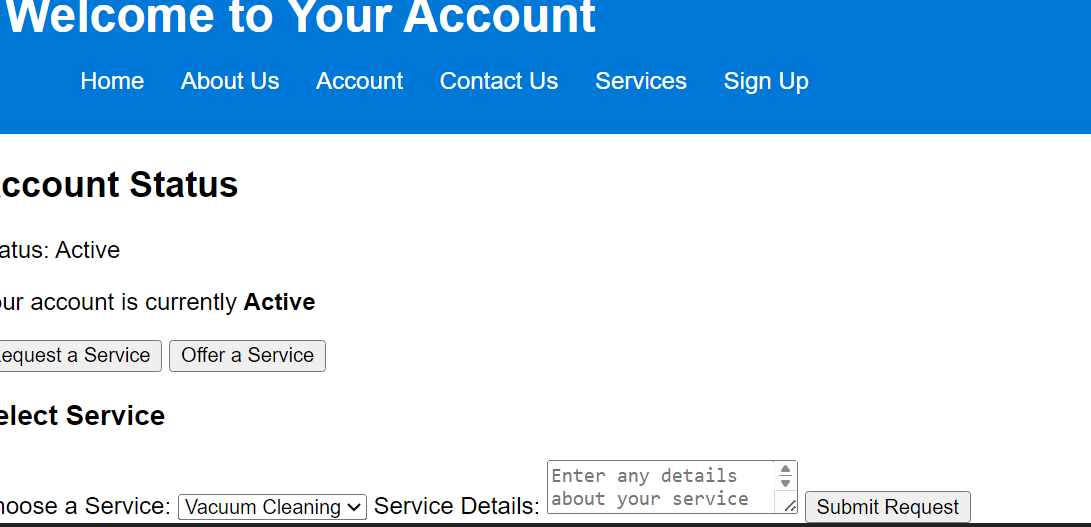
</body>

</html>

.*php file of account status*

*Fig 3.3.2-1 Managing User Details Logic Code*

The following is the error message when user attempts to delete a user entry:



*Fig 3.3.2-2 User Details Deletion Confirmation*

The EDIT button toggles between EDIT and SAVE. If you try to type into the next cell when the previous is empty, it shows an empty field error message as shown below. Also as seen in the

*Fig 3.3.2-2 Empty Cells Error Message and Saved Vs Editable Entries*

# CHAPTER FOUR: SYSTEM QUALITY TESTS

## 4.1 Introduction

To ensure that everything is working as expected, the following feature tests and DB integration tests were conducted:

## 4.2 User Entry and Validation Test (Feature Test)

This test was performed to confirm if the program coordinator can add a new user, validate the inputs, and prevent submission if any input is invalid.

*Steps taken to conduct this test:*

1. Open the Users Dashboard.
2. Click on "Add User" to create a new row.
3. Fill in the details (Name, password, email address)
4. Leave one field empty or use invalid data (e.g., email address).

*Expected Output:*

*Correct Output:*

* + All fields are valid: The row becomes read-only, and the details are saved to the array or database.
  + Duplicate or invalid Staff/Student Number: An error message like "Staff/Student Number must be unique!" prevents proceeding.
  + Empty fields: Error message "Field cannot be empty!".

*Incorrect Output:*

* + Allows empty fields or invalid/duplicate entries to be saved.
  + Does not provide appropriate error messages.

## Account Details Validation and Auto-Save Test (Feature Test)

This test was conducted to ascertain if the user dashboard: validates account details (Client or service provider name, email ,password)and auto-saves entries upon valid input.

*Steps taken to conduct this test:*

1. Open the Account Dashboard.

* Key in user details

1. Leave one field empty (e.g., Client Name) or select an invalid time (e.g.,John).

*Expected Output:*

*Correct Output:*

* + Valid details: Data is saved automatically to the database (User details).
  + Invalid or empty fields: Error message "Field cannot be empty or invalid!" prevents proceeding.
  + Entry row becomes read-only after valid data is saved.

*Incorrect Output:*

* + Allows saving of empty or invalid fields (e.g., hour 99).
  + No error message for invalid input.

## 4.4 Account creation Logic Test (Feature Test)

This test was conducted to confirm if the system correctly rings the bell and displays the countdown interface when the current time matches an event time.

*Steps taken in conducting account creationt:*

1.Enter username, email address and password

2.wait for account activation

Expected Output:

*Correct Output:*

* *The user logs in successfully*

*Incorrect Output:*

* + The account creation appears inactive.

## 4.5 Database Integration Test: Save and Retrieve Events

This test was conducted to confirm if time details are saved and retrieved correctly from the database (user and account tables).

*Incorrect Output:*

* Data is not saved in the database or is missing upon reload.
* Incorrect days or times are stored in the database.

## 4.6 Delete Functionality Test (Feature and DB Test)

This test was conducted to confirm if deleting a user removes it from both the interface and the databse.

*Expected Output:*

*Correct Output:*

* + The User account is deleted

Returns no rows

The user is active

*Incorrect Output:*

|  |
| --- |
| Invalid or empty information saved; no error messages. |
| No access to services without an account. |

In summary, the above tests and their outputs are as follows:

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Output** | **Incorrect Output** |
| User Entry Validation | Valid inputs saved; invalid inputs blocked with error messages. | Allows invalid/empty entries to save. |
| User Details Validation | Details auto-saved; errors block progression. | Invalid or empty information saved; no error messages. |
| Sign up logicLogic | User must sign up to have an account | No access to services without an account. |
| Database Save/Retrieve | Service provider information | Data missing or incorrect after reload. |

*Table 4.6 Summary of Tests*

# CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

## Conclusion

This project successfully implements a dynamic and user-centric web platform, Fua Hub, which connects clients with service providers for various household services, including cleaning, washing, ironing, and more. By integrating a functional login system, service request management, and a database structure, the website ensures that users can efficiently sign up, log in, and request services based on their needs.The system also adheres to best practices in web development, including SQL injection prevention, secure password handling, and session management. With the ability to handle bookings, manage user roles, and integrate payments, Fua Hub is a robust solution for providing hassle-free, eco-friendly, and professional services directly to clients' doorsteps.In conclusion, this project offers a comprehensive and scalable solution for household service management, and with further enhancements, such as adding real-time chat support, reviews, and ratings, it has the potential to provide even more value to users and service providers alike.

## Recommendation

The project demonstrates a robust and scalable approach to providing on-demand laundry and service provider solutions through an integrated platform. The website's architecture, which includes features like user authentication, service request management, and provider matching, aligns well with the goal of connecting clients with service providers. By implementing security features such as password hashing and session management, the project ensures data privacy and protects user accounts from unauthorized access. However, improvements could include further enhancing the user interface for a smoother experience, adding a more advanced payment integration (such as Stripe or PayPal), and incorporating real-time availability tracking for service providers. Additionally, the database could be optimized further with additional indexing for faster query processing as the platform grows. Overall, the system is poised for real-world deployment, offering a convenient and reliable service for both clients and providers.

# APPENDICES

## Appendix I: Gantt Chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **September** | | | | **October** | | | | **November** | | | **Dec** |
| **Activity** | **Week 1**  **(2nd-6th Sep)** | **Week 2**  **(9th-13th Sep)** | **Week 3**  **(16th-20th Sep)** | **Week 4**  **(23rd-27th Sep)** | **Week 5**  **(30th – 4th Oct)** | **Week 6**  **(7th – 11th Oct )** | **Week 7**  **(14th-18th Oct)** | **Week 8**  **(21st-25th Oct)** | **Week 9**  **(28th-1st Nov)** | **Week 10**  **(4th-8th Nov)** | **Week 11**  **(8th – 15th Nov)** | **Week 12**  **(2nd Dec)** |
| 1. Lecturer’s Introduction to Fundamental Concepts of a Software Project |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Development of a Workplan |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Developing U.I. |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Developing DB |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Logic implementation with tests |  |  |  |  |  |  |  |  |  |  |  |  |
| 1. Panel presentation |  |  |  |  |  |  |  |  |  |  |  |  |

## Appendix II: Resource Requirement and Economic Feasibility Assessment

|  |  |  |
| --- | --- | --- |
| **Resource** | **Purpose** | **Cost (Ksh)** |
| *Hardware resources:* | | |
| 1. Laptop / PC | * Develop project (front end, back end, testing) * Presentation | * Already available |
| *Software resources:* | | |
| 1. Python 3.12 | * For writing simple Python scripts | * Opensource |
| 1. Pycharm | * Front end implementation * Logic code writing and testing * Inhouse DB implementation * Running the project files | * Opensource |
| 1. MySQL | * Setting up an SQL logic environment on the computer | * Opensource |
| *Labour:* | | |
| * Development | | * Free |
| **TOTAL** | | **0.00** |

# REFERENCES

Aspose. (n.d.). *Image merge tool*. Retrieved from <https://products.aspose.app/imaging/image-merge>

CloudConvert. (n.d.). *File conversion*. Retrieved from <https://cloudconvert.com/>

Dev MySQL. (n.d.). *MySQL installer downloads*. Retrieved from <https://dev.mysql.com/downloads/installer/>

Dev MySQL. (n.d.). *MySQL tutorial excerpt*. Retrieved from <https://dev.mysql.com/doc/mysql-tutorial-excerpt/8.0/en/>

Docs Python. (n.d.). *Python documentation*. Retrieved from <https://docs.python.org/3/>

Docs Python. (n.d.). *Winsound module*. Retrieved from <https://docs.python.org/3/library/winsound.html>

FreeConvert. (n.d.). *Convert video to MP3*. Retrieved from <https://www.freeconvert.com/convert/video-to-mp3>