Community-Based Local Services Platform

CS346 - Software Engineering Lab Assignment 3 - Group 1A

Technical Documentation

Introduction

Welcome to the guide for the Community Service Management System (CSMS) software. This document provides essential insights for developers, administrators, and users involved in implementing and maintaining CSMS.

CSMS follows a client-server architecture, with Visual Basic Windows Form powering the frontend and MariaDB SQL supporting the backend. Whether you're developing, administrating, or using CSMS, this guide equips you with the knowledge to navigate and optimize the system effectively.

Database Schema

Users Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
userID	INT		False	True
userName	NVARCHAR	50	False	False
userType	NVARCHAR	50	False	False
email	NVARCHAR	255	False	False
userPhoto	VARBINARY	MAX	True	False
password	NVARCHAR	255	False	False
profileCompleted	BIT		False	False
twoFactorAuth	BIT		False	False

Service Providers Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
serviceProviderID	INT		False	True
userID	INT		False	False
serviceProviderPhotos	VARBINARY	MAX	True	False
serviceProviderName	NVARCHAR	255	False	False
serviceProviderEmail	NVARCHAR	255	False	False
serviceProviderDescription	NVARCHAR	MAX	True	False

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
rating	DECIMAL	3, 2	True	False
experienceYears	INT		True	False
minimumNoticeHours	INT		True	False
registrationStatus	NVARCHAR	50	True	False

Contact Details Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
contactID	INT		False	True
userID	INT		False	False
email	NVARCHAR	255	True	False
location	NVARCHAR	255	True	False
mobileNumber	NVARCHAR	10	True	False
socialMedia	NVARCHAR	MAX	True	False
address	NVARCHAR	255	True	False

Bank Details of Service Providers Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
bankDetailID	INT		False	True
serviceProviderID	INT		False	False
accountHolderName	NVARCHAR	255	True	False
accountNumber	NVARCHAR	50	True	False
bankName	NVARCHAR	255	True	False
branchName	NVARCHAR	255	True	False
ifscCode	NVARCHAR	20	True	False

Service Areas Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
areaID	INT		False	True
location	NVARCHAR	255	True	False

Service Types Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
serviceID	INT		False	True
serviceTypeName	NVARCHAR	255	False	False

Services Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
serviceID	INT		False	True

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
serviceProviderID	INT		False	False
serviceTypeID	INT		False	False
serviceName	NVARCHAR	255	True	False
price	DECIMAL	10, 2	True	False
serviceDescription	NVARCHAR	700	True	False
completionTime	INT		True	False
areaID	INT		True	False
servicePhoto	LONGBLOB		True	False
flagbit	BIT	1	True	False

Work Hours Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
hourID	INT		False	True
serviceProviderID	INT		False	False
startTime	TIME		True	False
endTime	TIME		True	False

Service Area Timeslots Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
areaTimeslotID	INT		False	True
serviceProviderID	INT		False	False
serviceID	INT		False	False
areaID	INT		False	False
serviceTypeID	INT		False	False
startTime	TIME		True	False
timeslotDate	DATE		True	False

Appointments Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
appointmentID	INT		False	True
serviceProviderID	INT		False	False
customerID	INT		False	False
areaTimeslotID	INT		False	False
appointmentStatus	NVARCHAR	50	True	False
bookingAdvance	DECIMAL	10, 2	True	False
serviceID	INT		False	False

Payments Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
paymentID	INT		False	True
appointmentID	INT		False	False
amount	DECIMAL	10, 2	True	False
paymentDateTime	DATETIME		True	False
paymentType	NVARCHAR	50	True	False
paymentStatus	NVARCHAR	50	True	False

Reviews Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
reviewID	INT		False	True
appointmentID	INT		False	False
rating	DECIMAL	3, 2	True	False
reviewText	NVARCHAR	MAX	True	False
reviewDate	DATETIME		True	False
givenForID	INT		True	False
givenByID	INT		True	False

Messages Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
messageID	INT		False	True
appointmentID	INT		False	False
senderID	INT		False	False
receiverID	INT		False	False
messageText	NVARCHAR	MAX	True	False
sentDateTime	DATETIME		True	False
isRead	BIT		True	False

Address Queries Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
queryID	INT		False	True
userID	INT		False	False
description	NVARCHAR	MAX	True	False
queryDate	DATETIME		True	False
status	NVARCHAR	50	True	False
resolutionDate	DATETIME		True	False
reply	VARCHAR	MAX	True	False
type	VARCHAR	MAX	False	False

Software Feedback Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
feedbackID	INT		False	True
userID	INT		False	False
feedbackText	NVARCHAR	MAX	True	False
feedbackDate	DATETIME		True	False

Notifications Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
notificationID	INT		False	True
notificationMessage	NVARCHAR	MAX	True	False
userID	INT		True	False
notificationDateTime	DATETIME		True	False

OTPs Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
otpID	INT		False	True
appointmentID	INT		False	False
otpCode	NVARCHAR	10	True	False
otpExpiration	DATETIME		True	False

Cancelled Appointments Table

COLUMN_NAME	DATA_TYPE	max_length	is_nullable	is_identity
cancellationID	INT		False	True
cancellationTime	DATETIME		True	False
refundAmount	DECIMAL	10, 2	True	False
appointmentID	INT		False	False

Development Environment

- Platform: Microsoft Visual Studio 2022
- Frontend : Visual Basic Windows Form
- Backend : MariaDB SQL

Code Structure

Admin

Admin page :

1. Get Queries list:

- **Description**: Populates all the queries information
- **Effect**: Query data loaded from AddressQueries table
- **SQL Query**

```
SELECT * FROM AddressQueries;
```

2. Get Registration Requests

- **Description**: Populates SPs whose registrationStatus is Pending
- **Effect**: SP data loaded from the serviceproviders table
- **SQL Query**

```
SELECT * FROM serviceproviders WHERE registrationStatus = 'Pending';
```

3. Get SP List:

- **Description**: Populates Service providers information whose registrationStatus is Accepted
- **Effect**: SP data loaded from the serviceproviders table
- **SQL Query**:

```
SELECT * FROM serviceproviders WHERE registrationStatus = 'Accepted';
```

4. Update SP registration status

- **Description**: Clicking on Approve button changes the registrationStatus of SP from Pending to Accepted, Reject button changes the registrationStatus of SP from Pending to Rejected
- **Effect**: SP data updated in the serviceproviders table
- **SQL Query**:

```
UPDATE serviceproviders SET registrationStatus = 'Accepted' WHERE servicePro
```

4. Resolve/Delete query

- **Description**: Clicking on Resolve sends the reply to the query raiser and Delete button deletes the query.
- **Effect**: Query data updated/deleted in addressQueries table
- **SQL Query**:

```
UPDATE AddressQueries SET status = 'Resolved', resolutionDate = CURRENT_TIME
DELETE FROM AddressQueries WHERE queryID = @Query_ID;
```

Customer

Home page :

1. Get Service Types

- **Description**: Populate service types in the filter section of the page
- **Effect**: Service types data loaded from serviceTypes table
- **SQL Query**:

```
SELECT * FROM serviceTypes;
```

2. Get Locations`

- **Description**: Populate serviceareas in the filter section of the page
- **Effect**: Service areas data loaded from serviceAreas table.
- **SQL Query**:

```
SELECT * FROM serviceAreas;
```

3. Search for SPs`

- **Description**: Search for SPs based on filters such has servie type, location, price and rating and populate the data into services list.
- **Effect**: Services data loaded from services table.
- **SQL Query**:

```
SELECT *
FROM services
WHERE serviceTypeID = @serviceTypeID
AND areaID = @areaID
AND price <= @maxPrice
AND rating >= @minRating;
```

4. Get Service details`

- **Description**: Populate service information into the UI
- **Effect**: Services data loaded from service table
- **SQL Query**:

```
SELECT *
FROM services
WHERE serviceID = @serviceID;
```

5. Get Reviews`

- **Description**: Populate reviews of a particular service provider.
- **Effect**: Reviews data loaded from reviews table.
- **SQL Query**:

```
SELECT *
FROM reviews
WHERE appointmentID IN (
  SELECT appointmentID
  FROM appointments
  WHERE serviceProviderID = @serviceProviderID
);
```

6. Initiate Booking and Payment`

- **Description**: Selecting SP, service type, location, date, and time slot shows the total price and advance amount required, and clicking on proceed to pay leads to the payment gateway for advance payment to confirm the booking.
- **Effect**: Insert data into appointments and payments table.
- **SQL Query**:

```
INSERT INTO appointments (serviceProviderID, customerID, areaTimeslotID, app
VALUES (@serviceProviderID, @customerID, @areaTimeslotID, 'Pending', @bookin


INSERT INTO payments (appointmentID, amount, paymentDateTime, paymentType, p
VALUES (LAST_INSERT_ID(), @bookingAdvance, NOW(), 'Advance', 'Pending');
```

Appointments Page

1. Get appointments:

- **Description**:Populate appointments information in the appointments list UI.
- **Effect**: Appointments data loaded from appointments table
- **SQL Query**:

```
SELECT * FROM appointments where customerID = @userID;
```

2. Raise Query:

- **Description**: On clicking the query button, the customer can raise query regarding any appointment by selecting the query type (service, timing, payment) and query description.
- **Effect**: Insert query record into addressQueries table with status as Pending.
- **SQL Query**:

```
INSERT INTO addressQueries (userID, query, queryDate, status)
VALUES (@userID, @queryDescription, NOW(), 'Pending');
```

3. View Appointment & OTP flow:

- **Description**:Shows details of a particular appointment. For Scheduled appointments, display the OTP and the remaining balance to be paid, also a chat box to communicate with the service provider. Also has the option to reschedule and cancel the appointment.
- **Effect**: Appointments data loaded from the appointments table.
- **SQL Query**:

```
SELECT a.appointmentID,sa.timeslotDate, sa.startTime, sp.serviceProviderName
FROM appointments a
JOIN serviceAreaTimeslots sa ON a.areaTimeslotID = sa.areaTimeslotID
JOIN serviceproviders sp ON a.serviceProviderID = sp.serviceProviderID
JOIN users u ON sp.userID = u.userID
JOIN serviceTypes st ON sa.serviceTypeID = st.serviceID
LEFT JOIN OTPs o ON a.appointmentID = o.appointmentID
WHERE a.customerID = <user_id>
AND a.appointmentStatus = 'Scheduled';
```

Service Provider

Home Page

1. Get information about the Service Provider

- **Description**: Fetch the name, rating, start and closing time, location and experience years of the service provider.
- **Effect**: Service Providers information loaded from Service Provider's Table.
- **SQL Query**:

```
SELECT serviceProviderName, rating, startTime, endTime, experienceYears
FROM serviceproviders
WHERE serviceProviderID = <service_provider_id>;
```

2. Get Reviews

- **Description**: Populate reviews of a particular service provider.
- **Effect**: Reviews data loaded from reviews table.
- **SQL Query**:

```
SELECT rating, reviewText, reviewDate
FROM reviews
WHERE givenForID = <service_provider_id>;
```

3. Get services

- **Description**: Populate services of a particular service provider.
- **Effect**: Services loaded from services table.
- **SQL Query**:

```
SELECT serviceName, price, serviceDescription, completionTime
FROM services
WHERE serviceProviderID = <service_provider_id>;
```

4. Delete and Edit services

- **Description**: Allows the service provider to edit or delete the information about a service.

- **Effect**: Data changed in the services table.
- **SQL Query**:

```
DELETE FROM services
WHERE serviceID = <service_id>;
UPDATE Services
SET serviceName = <new_service_name>, price = <new_price>, serviceDescriptio
WHERE serviceID = <service_id>;
```

5. Edit Profile

- **Description**: Allows the service provider to change its information.
- **Effect**: Data changed in the service provider table.
- **SQL Query**:

```
UPDATE serviceProviders
SET serviceProviderName = <new_name>, serviceProviderEmail = <new_email>, se
WHERE serviceProviderID = <service_provider_id>;
```

Appointments Page

1. Get Appoinments

- **Description**: Populate appointments information in the appointments list UI
- **Effect**: Appointments data loaded from appointments table
- **SQL Query**:

```
SELECT *
FROM appointments
WHERE serviceProviderID = <service_provider_id>;
```

2. Raise query

- **Description**: On clicking the query button, the customer can raise query regarding any appointment by selecting the query type (service, timing, payment) and query description.
- **Effect**: Insert query record into addressQueries table with status as Pending.
- **SQL Query**:

```
INSERT INTO AddressQueries (userID, query, queryDate, status)
VALUES (<user_id>, '<query_description>', CURRENT_TIMESTAMP, 'Pending');
```

3. View Appointment & OTP flow :-

- **Description**: Allows the service provider to change its information.
- **Effect**: Data changed in the service provider table.
- **SQL Query**:

```
SELECT a.appointmentID,sa.timeslotDate, sa.startTime, sp.serviceProviderName
FROM appointments a
JOIN serviceAreaTimeslots sa ON a.areaTimeslotID = sa.areaTimeslotID
JOIN serviceproviders sp ON a.serviceProviderID = sp.serviceProviderID
JOIN users u ON sp.userID = u.userID
JOIN serviceTypes st ON sa.serviceTypeID = st.serviceID
LEFT JOIN OTPs o ON a.appointmentID = o.appointmentID
WHERE sp.serviceProviderID = <service_provider_id>
AND a.appointmentStatus = 'Scheduled';
```

```
Start coding or generate with AI.
```

The integration of our academic section management system application, developed by a team of seven individuals, involved a systematic approach to merging distinct modules and functionalities into a cohesive final release version. Each team member was assigned specific responsibilities, ensuring a smooth integration process. We meticulously coordinated the integration steps, meticulously validating each component's compatibility and functionality within the integrated system.