Spiritual Deval Developer Documentation

Development environment setup:

Download VScode:

https://code.visualstudio.com/download

Download the newest version of Microsoft .NET for your platform:

https://dotnet.microsoft.com/en-us/download/dotnet/8.0

Ensure .NET 8 SDK is installed. You can check with the command:

dotnet --version

Install the following extensions in VScode:

C# Dev Kit, C#, Blazor snippets

Clone the Spiritual Deval repo:

git clone https://github.com/Sanmeet-EWU/cscd-488-490-project-spiritual-deval

Navigate to the repo directory

Navigate to the SpiritualDevalApp directory:

cd SpiritualDevalApp

Change the connection string in “appsettings.json” to match your local or AWS hosted MySQL instance.

Then run the command:

```
dotnet run
```

This will begin hosting the application locally at http://localhost:5164

For more information on deploying the project to an AWS EC2 instance and setting up the RDS MySQL database, refer to the Developer Instructions document.

Codebase overview and structure:

SpiritualDevalApp/

SpiritualDevalApp/Program.cs - The entry point that configures services, middleware, and starts the app.

SpiritualDevalApp/appsettings.json – A configuration file in a Blazor project that stores settings like the connection string, which defines how the app connects to the MySQL database.

Components/Layout

MainLayout.razor - Defines the primary layout structure.

MainLayout.razor.css - Styling for MainLayout.

NavMenu.razor - Navigation bar component.

NavMenu.razor.css - Styling for NavMenu.

Components/Pages/Account

AccessDenied.razor - Displays a message when a normal user tries to access an Admin page.

Login.razor - User login page.

Logout.razor - User logout page.

Components/Pages

About.razor - About page for the application.

-Should be updated for a more personalized about message.

Admin.razor - Admin dashboard.

Admin.razor.css - Styling for Admin.

Contact.razor - Contact information page.

-Needs be updated with temple's contact info.

CreateAccount.razor - Account creation form.

-Needs feature to verify email before account creation is confirmed.

Donation.razor - Razorpay donation page.

-Needs to be replaced with Donation.razor from 'KevinF-Branch' when Razorpay api keys are obtained.

Error.razor - Error handling page.

EventCreate.razor - Page for creating new events.

-Should implement feature to upload photo corresponding to the event.

Home.razor - Home page.

Mail.razor - Send email to volunteers. Contains email and gmail app password you must change to activate subscription feature. Refer to the Developer Instructions for more detail.

MyAccount.razor - View your account info/change your password.

PaymentComponent.razor - (on KevinF-Branch) Razorpay component for taking donation.

PromoteAdmin.razor - Admin promotion functionality.

Register.razor - Event page where you may register for an event.

Volunteer.razor - Volunteer info page.

VolunteerRegister.razor - Event-Volunteer page where you may sign up to volunteer (input your skills) and/or subscribe for email updates about the event.

_Imports.razor - Common Razor directives.

App.razor - Root application component.

Routes.razor - Route definitions for navigation.

Data

AppDbContext.cs - Entity Framework DbContext for managing database interactions. This is where all the database schema is defined.

Data/Payment (on KevinF-Branch)

PaymentController.cs - Provides endpoints for payment actions, creates orders.

RazorpayService.cs - Handles Razorpay order creation logic and integrates with Razorpay's API.

Models/Entities

Event.cs - Represents event details (corresponds to table in DB ‘events’).

- **Events**: related to eventregistrations through the primary key ‘event_id’ which corresponds to ‘event_id’ in that table. It also has columns ‘EventTitle’, ‘EventDescription’, ‘EventDate’, ‘EventLocation’, ‘MaxParticipants’, ‘VolunteersRequired’, and ‘SkillsRequired’.

EventRegistration.cs - Tracks registrations for events (corresponds to table in DB 'eventregistrations').

- **EventRegistrations**: has a 'user_id' as well as a 'event_id' column which are foreign keys corresponding to 'UserAccounts' and 'events' tables respectively. It also has columns 'event_registration_id' and 'registration_date'.

*User.cs - Core user data model.

UserAccount.cs - Manages account-specific details (corresponds to table in DB 'useraccounts').

- **Useraccounts**: related to EventRegistrations through the primary key 'id' which corresponds to 'user_id' in that table. Also, in this table are stored the user 'password's which are hashed for security. It also contains columns 'user_name', 'role', 'email', 'first_name', and 'last_name'.

VolunteerRegistration.cs - Tracks volunteer sign-ups (corresponds to table in DB 'VolunteerRegistrations').

- **VolunteerRegistrations**: contains 'user_id' and 'event_id' which are foreign keys corresponding to 'UserAccounts' and 'events' tables respectively. It also contains columns 'volunteer_registration_id', 'registration_date', and 'skills_provided'.

Models/ViewModels

LoginViewModel.cs - Handles login-specific data for UI binding.

wwwroot

app.css - Main style sheet.

razorpay.js - Handles Razorpay checkout process.

wwwroot/bootstrap

bootstrap.min.css - Minified Bootstrap CSS.

bootstrap.min.css.map - Bootstrap source map for debugging.

API documentation:

Razorpay: https://razorpay.com/docs/api/

Files Regarding Razorpay:

- PaymentComponent.razor
- Razorpay.js
- PaymentController.cs
- RazorpayService.cs

Testing guidelines:

Inside the SpiritualDevalApp directory, install xunit packages:

dotnet add package xunit

dotnet add package xunit.runner.visualstudio

Navigate to SpiritualDevalApp.Tests

Ensure that xunit versions in the project and test project are the same

In the same directory, run the tests:

dotnet test

The SpiritualDevalApp.Tests environment should now be ready for test development. Due to time constraints, testing code was not able to be written for the Blazor components used. We performed tests for the websites features manually, documented cases and outcomes, and verified the output. The guidelines for our testing process consisted of collecting data for each component test with the following format:

Test Case	**Description**	**Expected Outcome**	**Actual Outcome**	**Status**
Component				

Refer to the 'Testing and Evaluation' section of our **Final Project Report** document for the test results and process.

Known bugs and future enhancements:

Bugs:

- Home.razor and Volunteer.razor pages have 2 scroll bars.
- Volunteer.razor ‘Get in Touch’ button takes you to CreateAccount.razor even if you are logged in. If you try to log in again, you get an error.

Future Enhancements:

- Finish Implementation of payment portal
- Finish Implementation of admin payment statistics viewing
- Implement feature to upload images when creating events
- Add a ‘share’ button on events
- Add comments under events
- Make a slideshow for the home page showing off the temples culture