Spiritual Deval Deployment Instructions

Forking the GitHub Repository

To have your own version of the Spiritual Deval project to deploy, follow these steps to fork the repository:

1. Go to the Repository

Spiritual Deval GitHub Repo

2. Fork the Repository

• Click the Fork button (top right corner).

• This creates a copy under your own GitHub account.

3. Clone Your Fork (Main Branch)

On your local machine, open a terminal and run:

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

git checkout main

After checking out main, the project can be ran locally as-is once you change the SpiritualDevalApp /appsettings.json file's connection string to the database to match your setup. Further instructions are detailed later in this document.

Setup AWS RDS MySQL Database

Creating the Database (RDS MySQL)

1. Log into AWS and navigate to Aurora and RDS.

2. Click Databases → Create Database.

3. Select Standard Create.

4. Choose MySQL as the engine.

5. Leave the default engine version.

6. Select Production, Dev/Test, or Free Tier template.

7. Keep default availability and durability settings.

8. Set a DB cluster identifier.

9. Configure Master Username & Password.

10. Leave Instance Configuration as default.

11. Enable public access.

12. Click Create Database.

Setup Gmail Volunteer Updates

1. Go to Gmail and create a new Google account.

2. Verify the account and sign in.

3. Open Google Account Security Settings:

https://myaccount.google.com/security

4. Enable 2-Step Verification:

• Scroll to “Signing in to Google”.

• Click “2-Step Verification”.

• Follow the setup using your phone number.

5. Generate an App Password:

• Go to https://myaccount.google.com/apppasswords.

• Select “Mail” as the app and “Other” for the device.

• Name it “Blazor Mail” and click Generate.

• Copy the 16-character password provided.

6. Open the SpiritualDevalApp project.

7. Navigate to SpiritualDevalApp/Components/Pages/Mail.razor file.

8. Locate this line:

```
smtp.Credentials = new
System.Net.NetworkCredential("spiritualdeval@gmail.com", "your-app-password-
here");
```

9. Replace "your-app-password-here" with the generated 16-digit app password.

10. Save the file

Setting Up the EC2 Instance

1. Search EC2 in AWS.

2. Click Launch Instance.

3. Set Server Name under “Names and Tags”.

4. Select Ubuntu as the OS.

5. Create a new key pair and save it securely.

6. Create a Security Group and allow:

• SSH, HTTP, HTTPS traffic from the internet.

7. Click launch instance and wait for its status to be running before proceeding.

8. In the instances tab, click on your instance. Find the security tab, and click on the security group of your instance. Find “inbound rules” and click “edit inbound rules”. Add a rule with Custom TCP, port range 5000, CIDR block 0.0.0.0/0 , then click “save rules”.

9. On your local machine, navigate to the directory of your keypair and update key permissions (replace mypair.pem with the name of your keypair file):

```
chmod 400 mypair.pem
```

10. In a different terminal window, SSH into the EC2 instance using the public IP from AWS and install .NET. Use the file path of your keypair pem file. In the example, the path is “~/Downloads/mypair.pem”. (public IP can be found by clicking on your instance and viewing the instance summary):

```
ssh -i ~/Downloads/mypair.pem ec2-user@<EC2_PUBLIC_IP>

wget https://packages.microsoft.com/config/ubuntu/24.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb

sudo dpkg -i packages-microsoft-prod.deb

sudo apt update

sudo apt install -y dotnet-sdk-8.0 aspnetcore-runtime-8.0
```

11. On your local machine, find the appsettings.json file inside the SpiritualDevalApp folder. Retrieve your database master username and password by going to the Aurora and RDS dashboard, clicking on your database, and going to configurations.

Currently, the connection string inside appsettings.json looks like:

"ConnectionStrings": {
    "DefaultConnection": "Server=database-1.c3okwcawi0tj.us-east-2.rds.amazonaws.com;Database=spiritualdeval;User=admin;Password=<nw0|TbU)f#v>V~NcQVV!kmEO2N*;Port=3306;"
  }

You will need to edit the string to follow this format, but with your RDS endpoint, database name, username, and password:

"ConnectionStrings": {

  "DefaultConnection": "Server=RDS_ENDPOINT;Database=DB_NAME;User=USERNAME;Password=PASSWORD;Port=3306;"

}

Once edited, save the appsettings.json file.

12. On your local machine, apply migrations:

```
cd SpiritualDevalApp
dotnet ef migrations add initial
dotnet ef database update
```

13. On your local machine, deploy your Blazor app:

```
dotnet publish -c Release -o out
```

14. On your local machine, transfer files. Use the file path of your keypair pem file. In the example, the path is “~/Downloads/mypair.pem”. Also use your EC2 public IP. This may take a couple minutes.

```
scp -i ~/Downloads/mypair.pem -r out/*
ubuntu@<EC2_PUBLIC_IP>:/home/ubuntu/app
```

Running the App in EC2

1. In the window where you are SSH'd into EC2, navigate to the app folder:

```
cd app
```

2. Start the application:

```
dotnet SpiritualDevalApp.dll --urls http://0.0.0.0:5000
```

3. To run in the background:

```
nohup dotnet SpiritualDevalApp.dll --urls "http://0.0.0.0:5000" > output.log
2>&1 &
```

4. Verify process:

```
ps aux | grep dotnet
```

5. Stop the process if needed:

```
pkill -f SpiritualDevalApp.dll
```

Automating Deployment with Systemd

1. Inside EC2, create a systemd service:

```
sudo nano /etc/systemd/system/spiritualdeval.service
```

2. Add the following configuration and save the file:

```
[Unit]
Description=Spiritual Deval Blazor Server
After=network.target

[Service]
WorkingDirectory=/home/ubuntu/app
ExecStart=/usr/bin/dotnet /home/ubuntu/app/SpiritualDevalApp.dll --urls
"http://0.0.0.0:5000"
Restart=always
User=ubuntu
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

3. Reload and enable the service:

```
sudo systemctl daemon-reload
sudo systemctl enable spiritualdeval
sudo systemctl start spiritualdeval
```

4. Check service status:

```
sudo systemctl status spiritualdeval
```

5. The site should now be accessible at http://<public_ip>:5000

6. You can stop the service from running with the command:

```
sudo systemctl stop spiritualdeval
```

Running the App Locally

If you want to run the app locally on your machine to develop or test changes:

1. Navigate to the project directory

```
cd SpiritualDevalApp
```

2. Run the project

```
dotnet build
```

```
dotnet watch
```

3. Use “ctrl+c” to end the run

Updating the Deployment

If updates are made to the project and need to be deployed on the EC2 instance, follow these steps:

1. Navigate to the SpiritualDevalApp directory

```
cd SpiritualDevalApp
```

2. Publish the project:

```
dotnet publish -c Release -o out
```

3. Transfer files. Use the file path of your keypair pem file. In the example, the path is "~/Downloads/mypair.pem":

```
scp -i ~/Downloads/mypair.pem -r out/*
ubuntu@<EC2_PUBLIC_IP>:/home/ubuntu/app
```

4. In a separate window, SSH into EC2 restart the app:

```
sudo systemctl restart spiritualdeval
```

If you did not follow the "Automating Deployment with Systemd" section, then just launch the application again with:

```
dotnet SpiritualDevalApp.dll --urls http://0.0.0.0:5000
```

Backup & Recovery

Restoring a Backup (AWS RDS Snapshot)

1. Go to AWS RDS Console → Snapshots.

2. Select a snapshot and click Restore Snapshot.

3. Configure:

• DB Instance Identifier (e.g., database-restore).

• VPC & Security Groups (same as original DB).

4. Click Restore DB Instance and wait for status to change to available.

Performance and Security Checks

EC2 Performance & Security

Performance Monitoring

1. Go to AWS EC2 Console → Instances.

2. Select your EC2 instance.

3. Click Monitoring:

• CPU Utilization (High = possible overload).

• Network Traffic (Unexpected spikes = possible attack).

• Disk Activity (Slow I/O = bottleneck).

Security Checks

1. Go to AWS EC2 Console → Security Groups.

2. Click Inbound Rules:

• Port 22 (SSH): Only allow your authorized IP.

3. Check Outbound Rules (should allow all traffic).

RDS MySQL Performance & Security

Performance Monitoring

1. Go to AWS RDS Console → Databases.

2. Select your RDS instance.

3. Click Monitoring:

• CPU Utilization (Spikes = inefficient queries).

• Database Connections (Too many = performance issues).

• Read/Write Latency (High = slow query performance).

Security Checks

1. Go to AWS RDS Console → Security.

2. Ensure:

• Public Access = Enabled.

• SSL Enabled = On.

Connect MySQL Workbench to AWS MySQL RDS

You may want to access your MySQL database through an editor application like MySQL Workbench to view all tables or make manual adjustments. Below are instructions on how to connect the database to MySQL Workbench.

1. Get RDS Endpoint & Credentials

2. Go to AWS RDS Console → Click your MySQL RDS instance.

3. Copy the Endpoint (e.g., your-db-instance.abcdefg123.us-east-2.rds.amazonaws.com).

4. Note your Master Username and Password (from when you created the instance).

5. Allow Your IP in RDS Security Group

6. Go to EC2 Console → Security Groups.

7. Find the Security Group attached to your RDS instance.

8. Click Edit Inbound Rules → Add Rule:

• Type: MySQL/Aurora

• Port: 3306

• Source: Your IP (My IP)

9. Click Save Rules.

10. Connect in MySQL Workbench

11. Open MySQL Workbench → Click + (New Connection).

12. Set Up Connection:

• Connection Name: *Your Choice*

• Hostname: *Your RDS Endpoint*

• Port: 3306

• Username: *Your RDS Master Username*

13. Click Store in Vault → Enter your Password.

14. Click Test Connection → If successful, click OK → Connect.

You are now connected to your MySQL RDS database in MySQL Workbench.

Domain Configuration

To have your domain point to your EC2 instance, you must obtain an elastic IP and change your domain DNS settings to point to your instance.

1. Go to AWS EC2 Console → Click Elastic IPs (under "Network & Security").

2. Click Allocate Elastic IP Address → Click Allocate.

3. Select the new Elastic IP → Click Actions → Associate Elastic IP Address.

4. Choose:

• Instance: Select your EC2 instance.

• Private IP: Leave default.

5. Click Associate.

6. Copy your Elastic IP (e.g., 3.145.81.18).

Update GoDaddy DNS to Point to EC2

8. Log in to GoDaddy.

9. Go to My Products → Find your domain → Click DNS.

10. Edit the A Record:

• Host: @

• Points to: *Your Elastic IP*

• TTL: Default (or 600 sec).

11. Save changes.

Now, your domain (yourdomain.com) will point to your EC2 instance.

Allow HTTP Traffic in AWS EC2 Security Group

12. Go to AWS EC2 Console → Security Groups.

13. Find your instance's Security Group.

14. Click Edit Inbound Rules → Click Add Rule.

• Type: HTTP

• Port: 80

• Source: 0.0.0.0/0

15. Click Save Rules.

Now we must run the Blazor app. If you followed the "Automating deployment with Systemd" steps, you must edit a line in the service we created before. If you did not follow those steps, run the app normally like in the "Running The App in EC2" section:

16. Inside EC2, run the command:

```
sudo nano /etc/systemd/system/spiritualdeval.service
```

17. Replace the following line:

```
ExecStart=/usr/bin/dotnet /home/ubuntu/app/SpiritualDevalApp.dll --urls
"http://0.0.0.0:5000"
```

With:

```
ExecStart=/usr/bin/dotnet /home/ubuntu/app/SpiritualDevalApp.dll --urls
"http://0.0.0.0:80"
```

18. Reload systemd:

```
sudo systemctl daemon-reload
```

19. Start the service:

```
sudo systemctl start spiritualdeval
```

20. Enable the service to run on boot:

```
sudo systemctl enable spiritualdeval
```

21. Check service status:

```
sudo systemctl status spiritualdeval
```

22. Open a browser and go to:

http://yourdomain.com

Your Blazor app should now be live at your domain.