Mohamed Othman

Professor Bradford

Cse 3200

2/13/2024

Lab2 Part 1

GitHub Link for startup code: <https://github.com/M0e0thman/Cse3200/tree/92df543298a78ec37b5a56af5631df6bd61ad9cb/LAB2>

The Model-View-Controller (MVC) pattern is a software design pattern that separates an application into three interconnected components: Model, View, and Controller. Each component has a specific role in managing and processing data, user interface, and user input. Here's an architecture diagram and a bullet point list highlighting the three instances of MVC for the described functionalities:

Specifics to our project:

1. Questions/Next-Questions Functionality/Viewing the Questions:

Model:

1. Manages question data.

2. Holds question information.

3. Communicates with the Controller for data updates.

View:

1. Displays questions to the user.

2. Manages the user interface for interacting with questions.

3. Receives updates from the Model and sends user input to the Controller.

Controller:

1. Manages data flow between Model and View.

2. Updates the Model based on user input.

3. Ensures the View reflects the current state of the data.

2. Score/Score-Increments/Viewing the Scores:

Model:

1. Manages score data.

2. Holds score information.

3. Communicates with the Controller for data updates.

View:

1. Displays scores to the user.

2. Manages the user interface for viewing scores.

3. Receives updates from the Model and sends user input to the Controller.

Controller:

1. Manages data flow between Model and View.

2. Updates the Model based on user input.

3. Ensures the View reflects the current state of the data.

3. User-Identity/Set-Get-User-ID/View the User-Identity:

Model:

1. Manages user identity data.

2. Holds user identity information.

3. Communicates with the Controller for data updates.

View:

1. Displays user identity information to the user.

2. Manages the user interface for managing user identity.

3. Receives updates from the Model and sends user input to the Controller.

Controller:

1. Manages data flow between Model and View.

2. Updates the Model based on user input.

3. Ensures the View reflects the current state of the data.

**View**

Viewing the Question

Viewing the scores

Viewing the user identity

in

**Database**

**Controller**

Next-Questions Functionality

Score-Increments

Set-Get-User-ID

**Model**

User-Identity

Score

Questions