Design Document

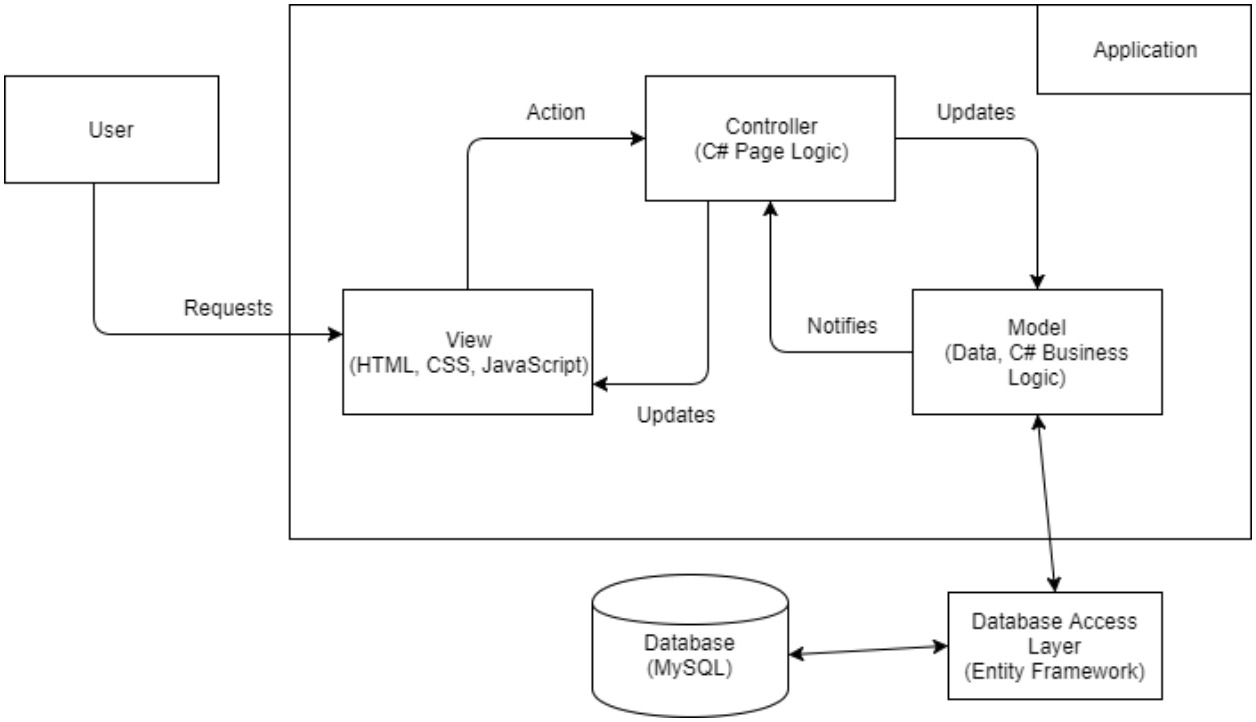
CSCE 361 Spring 2018
Group 7: CodePyramid

1. Introduction

The purpose of this document is to illustrate the architectural design and database relationships of the CodePyramid website. This document contains architectural diagrams and database table relationship diagrams for the CodePyramid system. This document serves its audience of software engineers in the implementation and maintenance of the CodePyramid project.

2. Architecture

2.1 Introduction



The high level architectural model of our system will be the Model-View-Controller pattern. We have chosen to use this pattern because it helps to separate domain objects in the model and presentation objects that are seen on screen such that we can have objects in the model completely self contained and work without reference to the presentation. The view piece of the model is what is shown to the user in the browser. The controller part of the model handles incoming requests, user input and interactions, and executes appropriate application logic. The model module represents the data in the application. This data can be read and written to via the data access layer, which will use the Entity Framework. The MySQL database will store permanent information.

2.2 Modules

2.2.1 Views

The view module is the component that the user will see and interact with. All information for the user will be structured using HTML and CSS, and all interactive elements will be handled using JavaScript.

All interactions with the view by the user will be sent to the controller, which may update the view, depending upon the actions the user performed.

2.2.2 Controller

The controller is the component that handles user interaction, works with the model, and ultimately selects a view to render that displays the User Interface. The controller handles user interaction by updating the view depending on the actions that the user takes on the page. For example, a user could click a link on the page to another page on the site, which would prompt the controller to take action. The Controller also works with the Model by using data in the model to populate necessary aspects of the view. The Controller can also make changes to the Model depending on actions taken on the View

2.2.3 Model

The Model is the part of the application that implements the logic for the application's data domain. This business logic, which in our case will be logic that controls grades for quizzes and completion of lessons, will be stored here and used on appropriate data. The data that the Model uses is pulled from the Database through the Database Access Layer. Changes to data, for example a grade change for a student's quiz, will be stored in the Database by the Model through the Database Access Layer. Data held in the Model will also be used by the Controller to perform tasks for presenting Views.

2.2.4 Database Access Layer

This module is responsible for selecting and storing data between the Database module and the Model module . All access to and from the Database must go through this module. This layer is also responsible for converting database data into C# objects that can be stored in the Model for further use by the Controller. The primary internal functions of this module involve querying tables in the database to access required information and storing this information into C# objects. It will also have functions that will permanently store any changes made to these objects by the other layers into the database. This layer will be implemented with the help of the Entity Framework.

2.2.5 Database

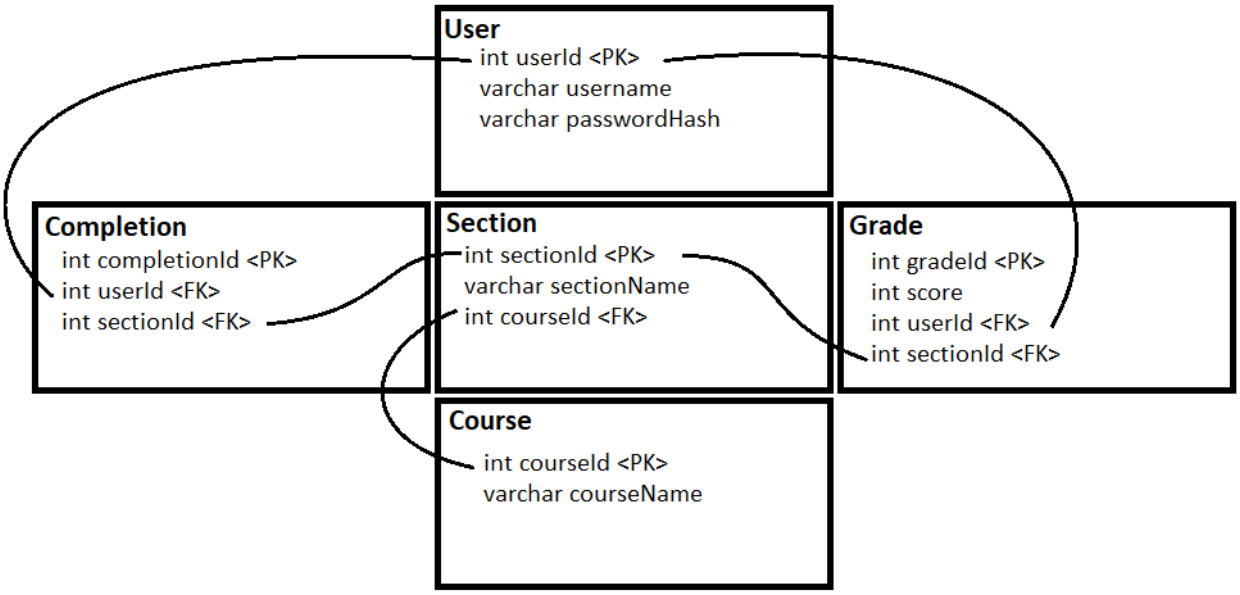
The database is responsible for keeping track of all persistent information in the system, as well as defining all relationships between the aforementioned data. The system will use an SQL database that will be implemented using MySQL. See Section 3.1 for more information regarding database information and relationships

3. Class Diagrams

3.1 Data Table Classes

The system will be using an SQL database to hold all the persistent data necessary. This includes information on users, courses, sections and the relationships between them. The figure below shows each table in the database with its respective columns as well as the relationships between the different tables.

3.1.1 Schema



3.1.2 Schema Information

The data in the database shall be organized in the following tables and columns.

User: This table holds all data for a single user of the system including information such as their username and password (hashed form).

Course: This table will represent the different courses, or programming languages, taught by CodePyramid. Each course has a name, such as HTML, CSS, or Javascript.

Section: This table will represent the different sections within each course. Each section will have a name, such as HTML Basics or HTML Quiz One.

Completion: This table represents the relationship between a user and a lesson section. If a user has completed an instructional section, an entry will be added to the Completion table. Each completion will have the relevant student and section.

Grade: This table represents the relationship between a user and an assessment section. If a user has taken a section that is a quiz, an entry will be added to the Grade table. Each grade will have the relevant student and section, as well as the score the student received.

3.2 Class Information

Classes will be implemented through C# in the model module. These classes will closely follow and replicate the schema illustrated in the database schema, and will receive attributes from the data access layer. The controller module will use these classes to update the view module, displaying any relevant class information to the user.