README For “Sched” App

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Email | Student # | Duties |
| Luke Morrison | lukemorrison@cmail.carleton.ca |  | * Design with MVC * Consistent refactoring to achieve design and structure * Initial sql setup * Handled all ajax * CSS for the look |
| Volodymyr Sharovar |  |  | * Configured Scheduling Algorithms using OO * Did basic initial UI elements * Coordinate backend |
| Matthew Crooke |  |  | * Java Client * Couple bugs |

Here is the crude output of a tree command to demonstrate the general structure.

+--**-controller**

| +---Pages

| | \---Validation

| \---schedule

+---**model**

+---styles

\---**view**

\---pages

+---JavaClient

| +---.settings

| +---bin

| \---src

Basically, how it works is that we have a view-model controller schema where we define the backedn modelling to be completely separated from any other component of the app. The controller is used as an interface between PHP and HTML that displays the content and interactions with the user using forms, buttons and a menu.

Install.php is the file which initially creates the table and populates the database. N

Index.html takes the main content and passes the control over Pagecontrol.js which dynamically set ups the html page using ajax calls and files from the view/Pages folder.

View/pages contains the set of high level pages that the user will be using to navigate their entire experience. To expand, if there was a feature (like an academic audit) it would be put inside its own modularized structure.

Also, Pagecontrol.js is divided into 2 sections, view and backend which are depicted as thius

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HTML VIEW CONTENT

….

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PHP BACKEND

….

These 2 sets of operations describe the functionality and is necessary for the functionality of the app to avoid having a bad user experience by interrupting and refreshing the page.

It should also be mentioned that this hierarchical structure would be maintained with AJAX by sectioning off HTML and PHP AJAX calls first into separate files, then their own modules – i.e. php backend for everything until they get to CUPORTAL. And another for CUPORTAL.

5. Provide a brief description of your folders/files. If your project includes several

folders, give the content of each folder. If you had separate scripts to implement

each task in the project, then provide the mapping tasks to html/php files.

6. Provide the instructions to execute when deploying your application to offer the

service to all programs of the Faculty of Engineering. What is the format of data

to provide to your application (for each academic program, for the prerequisite

trees, for the complementary studies electives, for the basic science electives, for

the (breadth) engineering electives ? What files in your project will need to be

modified to include all Engineering programs ? Briefly, describe the changes that

must be done in the code ?

7. Explain how you implement the prerequisite dealing with 3rd year status or 4th

yea status in Engineering and how does your code checks whether a student has

or not the proper status. Explain how your software processes the perquisites that

can be taken concurrently with the course (eg. ECOR 4995 must taken at the

same time as the 4th year project; STAT 2605 and STAT 3502 are prerequisite to

SYSC 4602, but the STAT can be taken concurrently to SYSC 4602). Explain how

your application deals with program transfer : For instance, SYSC 2006 requires

either SYSC 1005 or ECOR 1606. Software Eng. requires SYSC 1005, while

Communications Eng. requires ECOR 1606. A student starting in Software Eng.

may switch to Communications Eng, or vice versa. How does your software

determine if the student can take or not SYSC 2006 ?