INTRODUCTION

Today, the web is a repository of information which changes over time. In the days of web 1.0, servers would simply display static information to visitors (clients) without any more interaction than that. However, in the world of 2.0 where users are the world's major content generators, it is necessary for any non-trivial website to be able to store, display, receive and react to data on a grand scale through server-side processors and databases.

PROJECT OUTLINE

Create a server-side web application which makes use of a database to act as an interactive and data-driven administration system for a doctor's office.

Background and Context:

The website will act as an administration portal for employees at the doctor's office. A doctor's office is a building with many rooms and doctors where patients can come for treatment. Make sure to construct your user interface to support data entry and functionality by focusing on navigation and validation.

What you will do:

You must construct a multi-page web application in PHP with a backing MySQL database.

The application must include:

- Multiple pages
- A functional navigation system
- A database consisting of multiple tables

Your database must store data about:

- The doctors, who each have a name, a specialisation and a photograph.

- The rooms, which each have a floor, a number, and a name. Each doctor can be assigned multiple rooms, but each room has only one doctor assigned to it.
- The patients, which each have a name, a medical aid number, and a phone number.
- The appointments. Each appointment is between a patient and a doctor at a specific date and time. Each appointment must start on the hour, every hour, from 09:00 to 16:00 each day.

The functionality of the site must include:
- Viewing all the doctors.
- Searching for a patient by medical aid number.
- Viewing all appointments a doctor has and all the rooms that doctor is assigned.
- Viewing all appointments a patient has.
- Adding new patients.
- Assigning rooms to doctors.
- Booking new appointments.

Validation must include:
- Ensuring all fields in a form are completed before the form is submitted.
- Ensuring that a doctor cannot have an appointment booked at a time when he already has another appointment.
- Prompting the user in such a way as to aid error avoidance and correction.

The functionality above should be tested using a PHPUnit test suite. Download the PHPUnit dependency with composer.

In addition to the website, you will also produce a rationale document detailing your design decisions and reasoning. You will present the website and rationale deliverables to your lecturer at the end of the term.

What you will be given:

You will receive images of the doctors.

PROCESS

This section of the brief contains helpful guidance and information concerning your process. You will be technically guided through each phase of the project.

Phase 1: Planning

Website planning

Design and plan the website. Consider the layout and navigation system so that users can easily use the website and its functionality. Record all assumptions, decisions, and reasoning in your rationale.

Database planning

Plan out your database by deciding what tables you need, what fields those tables should have, and how the tables are related to each other. Record all assumptions, decisions, and reasoning in your rationale.

Form planning

The website includes a number of different forms. Design your forms to aid the user by preventing invalid inputs when possible, providing validation and feedback about errors, and displaying output in a clear and meaningful way. Record all assumptions, decisions, and reasoning in your rationale.

Phase 2: Implementation Phase

Website implementation

Create the pages based on the designs from your planning phase. Make sure that the navigation is functional. Record the changes you made and what you learned during this phase in your rationale.

Database Implementation

Create and populate the database following the designs from your planning phase. Record the changes you made and what you learned during this phase in your rationale. Additionally, add a Crow's foot ERD of your database as an appendix to your rationale.

Form implementation

Implement the forms based on the designs from your planning phase. Ensure that the forms are interactive and supply the user with appropriate information and error messages to aid data entry.

Phase 3: Testing phase

Automated Unit Testing

Ensure that your functionality is working as expected by creating a PHPUnit test suite and ensuring all tests are GREEN.

Manual Testing

Ensure that your database is being read and written to correctly by triggering form events ad checking for changes in your database.

Validation and Feedback

Make sure that the order page communicates information to the user effectively. Any input errors must be considered and communicated, and users should be guided through the form entry process.

Quality Control

Make sure the pages on your website are consistent, follow your designs, display all the information that they should, and support all the functionality that they should.

Phase 4: Final Phase

Documentation

Go back to your rationale document and polish it. This document should include all your assumptions, decisions, and reasoning.

Presentation

You must present your deliverables to your lecturer and the class.

DELIVERABLES

1

Website files

The project folder including all your PHP and CSS files as well as your `composer.json` manifest file. Do not include your `vendor` folder.

2

Rationale Document

This document should be delivered as a PDF. It must include all your assumptions, design decisions, planning and reasoning.

In addition to the soft copy you submit to the Google Classroom, you must also submit a printed hard copy before presenting your project with an attached and signed plagiarism form.

3

Screenshots and Recordings

You must supply 4 large PNG screenshots of your project's output. Do not include screenshots of code. These files should be named `xxxxxx_IDV203_S1.png`, `xxxxxx_IDV203_S2.png`, `xxxxxx_IDV203_S3.png`, and `xxxxxx_IDV203_S4.png` where `xxxxxx` is your student number.

You must supply an AVI screen capture of you interacting with your application. The video should be no shorter than 60 seconds and no longer than 3 minutes. This file should be named `xxxxxx_IDV203.avi` where `xxxxxx` is your student number.

These files are going to be used for display and advertising purposes, so they should demonstrate the best elements of your project without being too technical. They should also include your name to help identify you while on display.

These files should not be included in your Google Classroom archive folder submission. Instead, you must add these files to the `IDV_SUB/203/` folder on the desktop of the lecturer's machine in A2 during week 9.

SUBMISSION

- NO LATE SUBMISSIONS WILL BE ACCEPTED
- Submission and presentation will take place in Week 9
- Projects should be digitally submitted on Google Classroom
- All deliverables must be placed in a single zipped archive
- The zipped folder must follow the naming convention **`xxxxxx_IDV203.zip`** where **`xxxxxx`** is your student number

OUTCOME

By completing this project, learners will have learned to:

- Set up a local XAMP stack
- Create and access a database
- Create server-side PHP which produces client-side HTML
- Manipulate a database through SQL
- Write automated unit tests with PHPUnit
- Manage dependencies with composer
- Document a website implementation
- Professionally present projects
- Manage their time

ASSESSMENT

Learners will be assessed on their:

- Data analysis abilities
- Planning
- Problem solving skills
- Decision making abilities
- Tool use
- Technical skill
- Technical understanding
- Correctness in implementation
- Usage considerations
- Device and hardware considerations
- Initiative and brief expansion
- Layout, communication and aesthetics
- Documentation
- Professionalism

**Please see the attached marksheet for a detailed breakdown of the assessment weightings.*

SCHEDULE

WEEK	DATE	CLASS WORK	MILESTONES
1 (L1)	2017-07-10 to 2017-07-14	• Briefing • XAMPP Setup • Database fundamentals	• Planning Phase
2 (L2)	2017-07-17 to 2017-07-21	• SQL • Joins • Table Relationships	• Planning Phase • **Homework Assignment Due**
3 (L3)	2017-07-24 to 2017-07-28	• PHP Syntax • Fragments and Includes	• Planning Phase • **Homework Assignment Due**
4 (L4)	2017-07-31 to 2017-08-04	• Database Connections • PHP Data Objects (PDO)	• Implementation Phase • **Homework Assignment Due**
5	2017-08-07 to 2017-08-11	• Contact Week	• Contact Week
6 (L5)	2017-08-14 to 2017-08-18	• Superglobals • HTTP Verbs	• Implementation Phase • **Progress Milestone**
7 (L6)	2017-08-21 to 2017-08-25	• Server-side validation • PHPUnit	• Testing Phase • **Progress Milestone**
8 (L7)	2017-08-28 to 2017-09-01	• Constructing a rationale	• Final Phase • **Progress Milestone**
9	2017-09-04 to 2017-09-08	• Submission, demonstration, and assessment	