|  |
| --- |
| **Make a new document using the naming convention HWFinal-MUid.docx (example: HWFinal-johnsok9.docx).**  **Objective**: The objective of this exercise is to:  Use all the skills learned in the entire course.  **Submit**: screenshots, your word document, html files and links to your web page(s)  You **may** discuss the concepts with your fellow students  You **may** **not** show or share code with your fellow students, except your partner  **You may not show or share code with internet sources**  You **may** discuss this with your instructor or TA. |

**NOTE: This assignment includes your mandatory presentation to the Professor and submission to Canvas. Any assignments that are not presented will receive a zero for the entire assignment. This must be working on your OpenStack instance.**

**The programming part of the final project is a five-page website: an introduction (landing) page, two personal information pages, an AI front-end page, and a history page. These pages are described below. You are responsible for any sections that are left out (such as the actual layout of the page).**

# Part #1: Create a GitLab project

*Estimated time: .5 – 2 hours*

**Exercise:**

* Gitlab
  + **Follow the lab assignment for this part**
  + Create your final project in GitLab (one person only)
    - Share this with your partner
  + Clone it into your ~ubuntu directory
  + Note: you and your partner will work together. Make sure you both have access to one OpenStack server and the git repository (so you can show this later)
* Project
  + Make certain docker, apache, database, phpliteadmin, and password protection are all working

# Part #2: Landing Pages and Structure

Assignment: Vanna Smart Database Frontend Application

Estimated time: 20–40 hours

Develop a responsive web application using Docker, HTML, CSS, Bootstrap, JavaScript, jQuery, AJAX, Vanna, ChatGPT API, and external PHP REST services. The front end designed by students will communicate user queries to the Vanna backend, display responses, and log transactions via your own PHP REST server.

A diagram of a cloud server

AI-generated content may be incorrect.

# Pages Required

## Landing Page

* Engaging introduction and clear overview of your smart database frontend application.
* Brief explanation of the app's capabilities and intended use-case scenarios.
* General team information and easy navigation to other sections.

## Team Member Pages (two individual pages)

* Professional-style personal photo.
* Clear description of individual skills, interests, and relevant technical competencies.
* Detailed overview of personal contributions to the project.
* Reflections or insights gained from working on the project.
* Links to personal GitLab/GitHub or other relevant professional profiles (optional, encouraged).

## Database Query Frontend Page (Student-Created Frontend Interface)

* User-friendly interface for receiving user inputs as natural language queries.
* Clearly separates student frontend from Vanna backend.
* Uses AJAX to forward user inputs to Docker-based Vanna backend and displays processed responses.
* Logs transaction metadata to the PHP REST backend interfaces.
* Note: Before it can work, the internal RAG system must be trained on the MySQL structure and queries. This will be an in-class discussion for those teams on possible methodologies.

## Transaction History Page

* Comprehensive overview of past user interactions and database queries.
* Interactive and sortable table for transaction details.
* Fetches historical data from the PHP REST backend.

# Technical Clarification on System Interaction

Your team-built front end must act as the primary interface for user interactions. User queries should be captured and sent to your Docker-based Vanna backend, which processes these natural language inputs into SQL queries, executes them on the provided MySQL database, and returns structured responses. The front end then displays these responses clearly to the user. Simultaneously, the frontend captures transaction metadata and stores it via AJAX requests sent to the student-built PHP REST backend interface for historical logging.

# Technical Requirements

* Implement a responsive design with Bootstrap, ensuring functionality across various screen sizes and devices.
* Use external JavaScript and CSS files exclusively for better organization and maintainability.
* Employ AJAX calls (using jQuery with .done() and .fail() callbacks) exclusively for interacting with backend APIs.
* Ensure graceful degradation, clearly demonstrating adjustments in layout and usability when resizing from desktop to mobile views.

### Database Design (SQLite)

* **SQLite**
  + **Tables:**
  + transactions (log user interactions, API requests, and related details)
* MySQL
  + Given

### API Integration

* **ChatGPT:**
  + Content and image generation based on user prompts.
* **Vanna**
  + Loaded from Docker
* **PHP REST API (OpenStack):**
  + Database interactions including logging, retrieval, and detailed transaction records.

**Requirements**

Note: Everything you write must use the following (from the exam)

* Concepts
  + Html
  + CSS
  + JavaScript
  + Bootstrap (must be responsive)
  + jQuery
  + Ajax
* Style
  + Good looking using CSS
  + **All CSS, ajax, JavaScript must be external files**
  + Every page (including the menu) must include pictures that integrate well into the web page
* Location
  + All code MUST be working on your OpenStack instance HTTP server
  + All code must be in git
* Ajax
  + **You are to write all code using the jQuery Ajax routines as done in the lecture, labs, and homework. Some examples you see will use other methods, notably the fetch command. You may use those as an example to review, but you MUST use the $.ajax with done and fail calls for credit.**

The menu must provide the following items.

* Menu of pages you are providing (with a mechanism to switch pages): **You are to research and decide on your own mechanism** to provide the front-end menu of pages.
  + This was not covered in class. However, you have already seen websites for research that include them.
  + **Do NOT just add buttons or links to the top of a page (hint: nav), you must use a navigation element (navbar)**

### Helpful Resources

* [Bootstrap Documentation](https://www.getbootstrap.com/)
* [Chart.js Documentation](https://www.chartjs.org/)
* [OpenStack Documentation](https://www.openstack.org/)
* [SQLite Documentation](https://www.sqlite.org/)
* [ChatGPT API](https://platform.openai.com/docs/api-reference)
* [Vanna Documentation](https://www.vanna.ai/)

# Part #5: Each person must write their own Word Document

*Estimated time: 45 minutes (note: a 2 minute word document will be graded appropriately)*

**Exercise:**

* Make sure it includes your name, class, assignment etc…
* Explain (relating to this assignment and the entire course) in **full paragraphs** with **full sentences** for **full credit**.
  + what worked
  + what didn’t
  + how long it took
  + What you liked
  + What you didn’t

# SUBMIT:

* Word Document
* **Mandatory Presentation of working system in zoom session**
* Upload all code
* Git all code
* Paste the URL of your menu page into the submission as a comment