**Phoenicia University Advising System: Replication Rulebook**

**1. Overview**

The Phoenicia University Advising System is a web application built with Streamlit and Python that enables academic advisors to:

* Manage student course eligibility.
* Advise students by selecting “Advised” and “Optional” courses.
* Generate downloadable advising reports (both individual and comprehensive full student views).
* Sync key data (courses table, progress report, advising selections, and full student view) with Google Drive for persistence and cross-session collaboration.

**2. System Architecture**

**2.1 Components**

* **Front-End (Streamlit App):**
  + *app.py:* Main entry point that orchestrates the application.
  + *eligibility\_view.py:* Module for advising a single student.
  + *full\_student\_view.py:* Module for displaying a comprehensive view of all students and their course statuses.
  + *data\_upload.py:* Module handling file uploads for the courses table, progress report, and advising selections.
  + *google\_drive.py:* Module for interacting with the Google Drive API (upload, download, sync).
  + *reporting.py:* Module for generating and formatting Excel reports.
  + *utils.py:* Collection of utility functions (eligibility checking, logging, etc.).
* **Back-End:**
  + **Google Drive:** Used for data synchronization (files are stored in a specified folder).
* **Logging:**
  + All critical events and errors are logged to a file (app.log) using Python’s logging module.

**3. File Structure**

graphql

Copy

phoenix\_advising\_system/

│

├── app.py # Main application file.

├── data\_upload.py # Handles data file uploads.

├── eligibility\_view.py # Student advising view.

├── full\_student\_view.py # Full student view and reporting.

├── google\_drive.py # Google Drive API interactions.

├── reporting.py # Excel report generation.

├── utils.py # Utility functions and logging.

├── requirements.txt # Python dependencies.

├── .streamlit/

│ └── secrets.toml # Secrets for local testing.

├── pu\_logo.png # University logo.

├── app.log # Log file for system events.

└── README.md # Project documentation.

**4. Installation and Setup**

**4.1 Prerequisites**

* **Python 3.7 or higher**
* **Streamlit**
* **Pandas**
* **Openpyxl**
* **Google API Libraries:**
  + google-auth
  + google-auth-oauthlib
  + google-api-python-client
* **(Optional) PyDrive** if using it for Drive operations

**4.2 Installation Steps**

1. **Clone the Repository:**

bash

Copy

git clone https://github.com/yourusername/phoenix\_advising\_system.git

cd phoenix\_advising\_system

1. **Install Dependencies:**

Ensure you have a requirements.txt file (sample below) and run:

bash

Copy

pip install -r requirements.txt

**Example requirements.txt:**

nginx

Copy

streamlit

pandas

openpyxl

google-auth

google-auth-oauthlib

google-api-python-client

1. **Set Up Secrets:**

For local testing, create a .streamlit folder in your project directory and add a secrets.toml file with the following content:

toml

Copy

[google]

client\_id = "YOUR\_GOOGLE\_CLIENT\_ID"

client\_secret = "YOUR\_GOOGLE\_CLIENT\_SECRET"

refresh\_token = "YOUR\_REFRESH\_TOKEN"

folder\_id = "119WepIeS5SuG1sKMuPhYezikGi3OKOJ8"

Replace placeholders with your actual Google API credentials and the provided Google Drive folder ID.

1. **Obtain a Refresh Token:**

Use the provided get\_refresh\_token.py script to generate a refresh token:

bash

Copy

python get\_refresh\_token.py

Copy the printed refresh token into your secrets.toml.

**5. Data Requirements**

**5.1 Courses Table**

* **File Name:** courses\_table.xlsx
* **Required Columns:**
  + Course Code
  + Type (e.g., "Required", "Intensive")
  + Prerequisite
  + Concurrent
  + Corequisite
  + Offered (e.g., "Yes"/"No")
  + Credits (numerical value)

**5.2 Progress Report**

* **File Name:** progress\_report.xlsx
* **Sheets:**
  + Required Courses
  + Intensive Courses
* **Required Columns:** Must include student details such as:
  + ID
  + NAME
  + # of Credits Completed
  + (Optionally) # Registered

**5.3 Advising Selections**

* **File Name:** advising\_selections.xlsx
* **Required Columns:**
  + ID
  + Advised (comma-separated course codes)
  + Optional (comma-separated course codes)
  + Note (advisor’s note)

**5.4 Full Student View**

* The Full Student View is generated dynamically by the app and includes columns like:
  + ID
  + NAME
  + # of Credits Completed
  + Standing
  + Advising Status
  + One column per course (status values: c, a, na, ne)

**6. Running the Application**

1. **Run Locally:**

bash

Copy

streamlit run app.py

1. **Using the Application:**
   * **Data Upload:** Use the sidebar to upload the Courses Table, Progress Report, and Advising Selections.
   * **Advising:** Use the Student Eligibility View to advise students by selecting courses.
   * **Full Student View:** View the complete student report with all course statuses.
   * **Report Downloads:** Download individual, full, or all advised student reports.
   * **Google Drive Sync:** The app automatically syncs changes to Google Drive. You can also manually trigger synchronization via sidebar buttons.

**7. Google Drive Synchronization**

* The system synchronizes the following files with a Google Drive folder (ID: 119WepIeS5SuG1sKMuPhYezikGi3OKOJ8):
  + Courses Table (courses\_table.xlsx)
  + Progress Report (progress\_report.xlsx)
  + Advising Selections (advising\_selections.xlsx)
  + Full Student View (full\_view\_table.xlsx)
  + Full Advising Report (Full\_Advising\_Report.xlsx)
  + All Advised Students Reports (All\_Advised\_Students.xlsx)
* **Synchronization Process:**
  + Files are saved locally and then uploaded/updated on Google Drive using the functions in google\_drive.py.
  + Google API credentials are securely loaded from st.secrets.

**8. Logging**

* All important events and errors are logged to app.log.
* Logging functions (log\_info and log\_error) are defined in utils.py.

**9. Deployment**

**Deploying on Streamlit Cloud**

1. **Push Code to GitHub:**  
   Ensure your repository contains all the files described in this rulebook.
2. **Connect Repository to Streamlit Cloud:**
   * Log in to Streamlit Cloud.
   * Create a new app by connecting to your GitHub repository.
   * Set up the necessary environment (Streamlit will install dependencies from requirements.txt).
3. **Configure Secrets:**
   * In Streamlit Cloud, go to your app's settings and add your Google API credentials under the "Secrets" section using the format described in Section 4.
4. **Deploy and Test:**
   * Once deployed, test all functionalities (data upload, advising, report downloads, and synchronization).

**10. Troubleshooting**

* **Secrets Not Found:**  
  Verify that secrets.toml exists in .streamlit/ for local testing or that secrets are configured in Streamlit Cloud.
* **Google Drive Sync Issues:**  
  Ensure your Google API credentials are correct and that the folder ID is accurate. Check app.log for error details.
* **Data Upload Errors:**  
  Confirm that uploaded Excel files contain the required columns and follow the specified format.
* **Log File:**  
  Check app.log for detailed logging information to help diagnose issues.

**11. Replication Summary**

To replicate the system:

1. Clone the repository.
2. Install dependencies using pip install -r requirements.txt.
3. Set up your .streamlit/secrets.toml (for local testing) or configure secrets in Streamlit Cloud.
4. Ensure your data files are formatted as described.
5. Run the app using streamlit run app.py.
6. Follow the in-app instructions to upload data, advise students, and sync data with Google Drive.

**12. License**

This project is licensed under the MIT License. See the LICENSE file for details.

Feel free to customize this rulebook further to match any additional internal guidelines or specific setup details for your environment.

**Credit and Calculation Rules**

**1. Student Credit Calculation**

**1.1. Total Credits Completed**

* **Definition:**  
  The system calculates a student’s "Total Credits Completed" by summing two fields (if available):
  + **# of Credits Completed:** This value comes directly from the progress report (usually from the "Required Courses" sheet).
  + **# Registered (Optional):** If present, this value represents the credits the student is currently registered for.  
    **Formula:**

nginx

Copy

Total Credits Completed = (# of Credits Completed) + (# Registered)

If the "# Registered" field is not available or is empty, then the total is simply the "# of Credits Completed."

**1.2. Student Standing Determination**

* **Rules:**  
  The system determines a student's academic standing based on the total number of credits:
  + **Senior:** If the total credits are 60 or more.
  + **Junior:** If the total credits are between 30 and 59 (inclusive).
  + **Freshman/Sophomore:** If the total credits are less than 30.

**Function Used:**  
The get\_student\_standing function implements this logic:

python

Copy

def get\_student\_standing(credits\_completed):

if credits\_completed >= 60:

return 'Senior'

elif credits\_completed >= 30:

return 'Junior'

else:

return 'Freshman/Sophomore'

**2. Advising and Report Calculations**

**2.1. Course Status**

Each course for a student is assigned one of the following statuses:

* **c (Completed):** The student has already completed the course.
* **a (Advised):** The course has been explicitly recommended (advised) for the student.
* **o (Optional):** The course has been marked as optional (sometimes merged with "Eligible not chosen" as "na").
* **na (Eligible not chosen):** The student meets the eligibility requirements for the course but did not choose it as an advised course.
* **ne (Not Eligible):** The student does not meet the eligibility requirements for the course.

**2.2. Calculation of Course Status**

* **Completed Check:**  
  The system checks if the course is marked with a "c" in the student's record.
* **Eligibility Check:**  
  The system uses the check\_eligibility function to verify if the student meets prerequisites, concurrent requirements, and corequisites for a course.
  + If a course is neither completed nor selected as advised/optional but the student meets eligibility criteria, it is flagged as "na" (Eligible not chosen).
  + Otherwise, if the student doesn’t meet the criteria, the course is flagged as "ne" (Not Eligible).

**2.3. Credits Advised and Credits Optional**

* **Calculation:**
  + **Credits Advised:** Sum the Credits values from the Courses Table for all courses that are in the student's "Advised" selection.
  + **Credits Optional:** Sum the Credits values for all courses that are in the student's "Optional" selection.
* These values are used to give advisors a quick snapshot of the academic load recommended and optionally suggested for each student.

**3. Data Aggregation for Reports**

**3.1. Full Student View Report**

* The full student view table displays key student information along with a column for each course. Each cell in a course column contains the course status (e.g., c, a, na, or ne).
* **Additional Calculated Fields:**
  + **Total Credits Completed:** Calculated as described above.
  + **Standing:** Derived from total credits.
  + **Advising Status:** Determined based on whether advising selections (advised/optional courses) exist for the student.

**3.2. Summary Sheet in Full Advising Report**

* The summary sheet aggregates course statuses across all students. For each course, it provides the total number of students with each status:
  + **Completed (c)**
  + **Advised (a)**
  + **Eligible not chosen (na)**
  + **Not Eligible (ne)**
* This aggregation helps identify trends (for example, courses that are frequently advised or those where many students do not meet eligibility).

**4. Consistency and Replication Guidelines**

* **Uniform Data Input:**  
  Ensure that all Excel files (Courses Table, Progress Report, Advising Selections) adhere to the required column names and formats as specified.
* **Data Synchronization:**  
  All changes made to the advising selections or progress data should be reflected in the full student view table and synchronized to Google Drive. The system uses the calculated values from the uploaded files to ensure that all reports are based on the latest data.
* **Testing:**  
  Always test the calculation logic with sample data to confirm that credits and standings are computed accurately before deploying or updating the system.