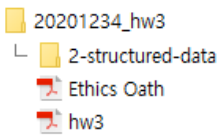
EE477 Database and Big Data Systems, Spring 2021
HW3*

Due date: May 14, 2021 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homeworks. Your submission should be one gzipped tar file whose name is `YourStudentID_hw3.tar.gz`. For example, if your student ID is 20211234, and it is for homework #3, please name the file as `20211234_hw3.tar.gz`. You can also use these extensions: tar, gz, zip, tar.zip. Do not use other options not mentioned here.



Please make the file for submission as follows. First of all, make a directory named `YourStudentID_hw3`. Then put all your files in the directory: one folder that contains all the source codes (`2-structured-data`), one PDF file (`hw3.pdf`) and an Ethics Oath pdf file. Finally, compress `YourStudentID_hw3` directory as specified above. Also, do not include Korean letters in any file name or directory name when you submit. **If you violate any of the file name format or extension format, we will deduct 1 point of total score per mistake.**

Submitting writeup: Prepare answers to the homework questions into a single PDF file. You can use the following template. Please write as succinctly as possible.

Ethics Oath: For every homework submission, please fill out and submit the **PDF** version of this document that pledges your honor that you did not violate any ethics rules required by this course and KAIST. You can either scan a printed version into a PDF file or make the Word document into a PDF file after filling it out. Please sign on the document and submit it along with your other files.

Discussions with other people are permitted and encouraged. However, when the time comes to write your solution, such discussions (except with course staff members) are no longer appropriate: you must write down your own solutions independently. If you received any help, you must specify on the top of your written homework any individuals from whom you received help, and the nature of the help that you received. *Do not, under any circumstances, copy another person's solution.* We check all submissions for plagiarism and take any violations seriously.

*Material adapted from Google Cloud Manual and Simon Fraser University CMPT354.

Ever wondered how you can use your database in the real world?
In this homework, you will implement a simple Web application based on a Cloud SQL database. Instead of implementing an application from scratch, you will start with an existing sample code (called Bookshelf) and extend it with simple functionalities (search and rate books). The instructions for setting up the Bookshelf application are detailed in Sections 1 to 5. The functionalities to implement are described in Section 6. We will give full credit if you can run the Bookshelf application and implement the functionalities. We will also give extra credit for more creative solutions.

We note that you should carefully check **whether each step needs to run on your local computer or on the Google GCP console**.

Throughout this homework, you will use multiple technologies: Google Cloud SQL (for the database), Google App Engine (for hosting the Web application), Flask (for the Web application), and SQLAlchemy (for communicating with the database). Have fun!

1 Initial setup

Follow this tutorial to setup the Bookshelf app:
https://cloud.google.com/python/getting-started/tutorial-app

We also provide more detailed steps below, in order to minimize trial and error:

- Create new Google Cloud App Engine on your GCP project. Select region as 'asia-northeast3.' Go to App Engine
- Start to use APIs in your GCP project : Cloud Datastore, Cloud Pub/Sub, Google Cloud Storage JSON, Cloud SQL Admin, Google+ API.

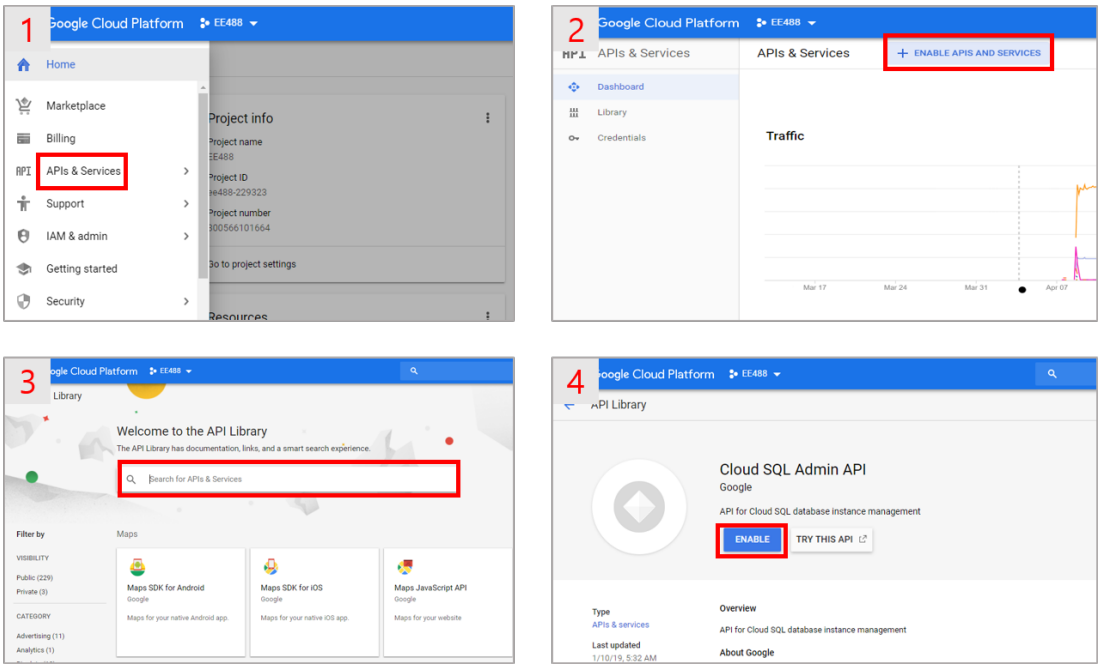


Figure 1. An example for enabling APIs.

- Install Google SDK on your local computer for developing your application. The installation guide is described in the link:
  https://cloud.google.com/sdk/docs/downloads-versioned-archives.
  If you use `WINDOWS`, you should add the `google-cloud-sdk/bin` directory to the `PATH variable`. After initializing the SDK, get the local user credentials for authentication with the GCP service.
  Also, verify that the project associated with SDK is correct.

  ```
  gcloud auth application-default login
  gcloud config list
  ```

- Unzip *2-structured-data.zip* that provided via KLMS. We will use the codes under this directory for this homework.

2 Cloud SQL Connection

Follow this tutorial to setup the Cloud SQL connection:
https://cloud.google.com/python/getting-started/using-cloud-sql

Again, we also provide more detailed steps below to minimize trial and error.

- Install SQL proxy on your local computer. The Cloud SQL proxy is used to connect to the Cloud SQL instance when running locally. The installation guide is described in the link: https://cloud.google.com/python/getting-started/using-cloud-sql
  Follow 'Connecting from an external application using the proxy' - step 2 and 6

- Use the Cloud SDK to run the following command in your **local computer**:

  ```
  gcloud sql instances describe [YOUR_INSTANCE_NAME]
  ```

  We will use the same SQL instance that you created in HW0. You can find your instance name in the Instance Connection Name in SQL page. The format of Instance Connection Name is `[PROJECT_NAME]:[REGION_NAME]:[INSTANCE_NAME]`.

- Create new database `bookshelf` in the **Google GCP console**.

  ```
  gcloud sql connect [YOUR_INSTANCE_NAME] --user=root
  CREATE DATABASE bookshelf;
  ```

  If you see an error saying 'Mysql client not found,' install MySQL client as described in the link: https://cloud.google.com/sql/docs/mysql/connect-admin-ip

- Change configurations in `config.py` and `app.yaml` under *2-structured-data/*

  In the `config.py` :
  `DATA_BACKEND = 'cloudsql'`
  `PROJECT_ID = [YOUR_PROJECT_ID]`
  `CLOUDSQL_USER = [YOUR_USER_ID]` (For simplicity, you can use the root user.)
  `CLOUDSQL_PASSWORD = [YOUR_CORRESPONDING_PASSWORD]`
  `CLOUDSQL_DATABASE = 'bookshelf'`

`CLOUDSQL_CONNECTION_NAME = [YOUR_CONNECTION_NAME]` (The connection name should be in the format 'project:region:cloudsql-instance'.)

In the `app.yaml` :

`cloud_sql_instances : [YOUR_CONNECTION_NAME]`
It should be in the format 'project:region:cloudsql-instance'. **Uncomment** this line.

3 Test Your Code Using Cloud SQL Proxy

Before deploying your Web application (Section 4), it is a good practice to test it locally. Proxy can help you quickly check your developing process on the Web. If you want to get more information about the Cloud SQL proxy, please go to the link: https://cloud.google.com/sql/docs/mysql/sql-proxy

- Before running the program, you should turn on the Cloud SQL proxy that you've installed in the section 2. **Keep the Cloud SQL proxy running** the entire time you test your app locally.

  For `LINUX/MACOS`,

  ```
  ./cloud_sql_proxy -instances="[YOUR_INSTANCE_CONNECTION_NAME]"=tcp:3306
  ```

  For `WINDOWS`,

  ```
  cloud_sql_proxy.exe -instances="[YOUR_INSTANCE_CONNECTION_NAME]"=tcp:3306
  ```

  Here, if the tcp number `3306` is already used by another network, you should use a different number (say, 3307). In that case, you should change the tcp number in the variable `LOCAL_SQLALCHEMY_DATABASE_URI` of the `config.py`.

- **Open a new terminal** and go to the directory where you stored *2-structured-data*. (i) Now, install Anaconda which enables to use virtual environments. You can skip this step if your local computer already has installed Anaconda. (ii) Next, enter the following commands to create a virtual environment with Python 3.6.13 (you can use any environment name in `[ENV_NAME]`) and install dependencies:

  For `LINUX/MACOS`,

  ```
  conda create -n [ENV_NAME] python=3.6.13 anaconda
  conda activate [ENV_NAME]
  pip install -r requirements.txt
  ```

  For `WINDOWS`,

  ```
  conda create -n [env_name] python=3.6.13 anaconda
  conda activate [env_name]
  pip install -r requirements.txt
  ```

- Enter the following command to create a table in the database `bookshelf`.
  For `LINUX/MACOS`,

  ```
  python bookshelf/model_cloudsql.py
  ```

  For `WINDOWS`,

  ```
  python bookshelf\model_cloudsql.py
  ```

- Now, you can test the sample codes in your local machine. First, run the `main.py`.

  ```
  python main.py
  ```

  Then you can check the website at `http://localhost:8080`.

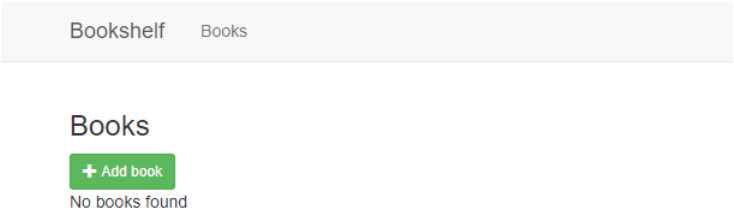


Figure 2. The initial Bookshelf Web application should look like this.

4 Deploy Web Application Using App Engine

Now you are ready to deploy your Web application.

- Deploy your Web application to the Google Cloud App Engine. Run the following command in your local computer.

  ```
  gcloud app deploy
  ```

  This command can take more than 10 minutes.
  Note This command may produce some errors on the first try. If you got an error, please wait for a minute and re-try. If the error still occurs on the second try, you have to debug the previous steps.

- If an error saying that `UNKNOWN: Error Response: [4] DEADLINE_EXCEEDED` occurs, this is due to the overall request times out. Increase the timeout using following gcloud config command:

  ```
  gcloud config set app/cloud_build_timeout [TIMEOUT_SECONDS]
  ```

  The original timeout is set at 10 minutes. You can use 1000 as `TIMEOUT_SECONDS` for example.

- Check your website at `https://[PROJECT_ID].[REGION_ID].r.appspot.com`
  You can check your own website address in your App Engine dashboard (Figure 3).

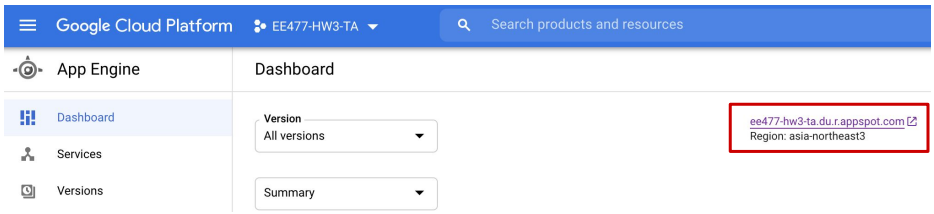


Figure 3. Website address.

- Deploying your application using App Engine each time when you modify your code will take take some time and also it may cost you a lot. Therefore, we recommend you to use local machine for test as described in section 3, then deploy your final version using App Engine.

- `[Important]` When you do not use your application, please **DISABLE** the application in **App Engine Settings** (Figure 4). If you have deployed your application multiple times, please **delete old versions** in the App Engine (Figure 5).

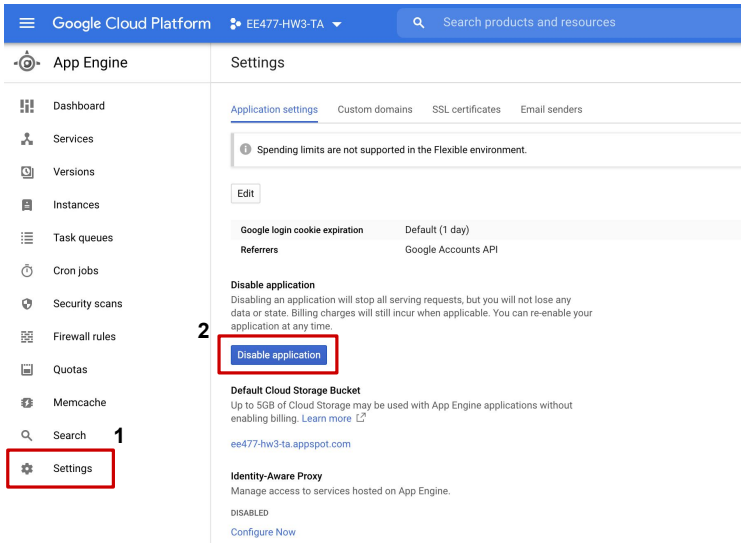


Figure 4. How to disable the application in GCP.

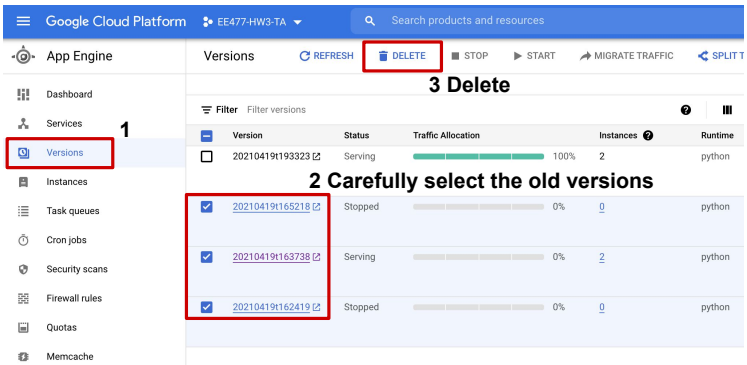


Figure 5. How to delete the old versions in GCP.

5 Understand Flask and SQLAlchemy

Within the Bookshelf application, you should get familiar with the following tools:

- Flask is a microframework for Python based Web developing.
  http://flask.pocoo.org/docs/1.0/
- SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
  https://www.sqlalchemy.org/
- You may find the html examples helpful to understand Flask and SQLAlchemy.

6 Requirements

- [**Task 1**] (40 points) Demonstrate that you can successfully deploy the Bookshelf application.
- [**Task 2**] (20 points) Enable a user to rate books (say with 1 to 5 stars.)
- [**Task 3**] (40 points) Enable a user to search for books (say by title and year.)

  Task 2 and 3 are open-ended, and you may use any database and UI designs you want for the implementation. Also for simplicity, you may assume that you are the only user of the system.
- [**Extra credit**] (10 points) If you implement something more creative or sophisticated than the above (e.g., implementing ratings by multiple users), we will give up to 10 more points.
- Describe all the above results with detailed screenshots in a report and name it hw3.pdf. If the screenshots are not clear, we will ask you to demo the Web application in person during office hours.
- Also submit all your source code under the directory: *2-structured-data/*.