Case Study Subject – University Administration system

Feasibility Study / Requirement –

Create a Client-Server Architecture which help to develop Web application from scratch using Python Flask, Django and Databases like MySQL and MongoDB.

Introduction

University Administration system is one of the most common and the first application implemented in any higher educational organization.

In University a large amount of data is processed, and the results are used in running an organization. The University management system project maintains the list of colleges, list of student details as per collage and their different streams. It also maintains the examination and the result department with a proper menu system (web form).

Project Category

Category of this project will be "Client-Server Web application which help to develop and improve the user interface and user experience of web app".

Project Prerequisites

Need some prior knowledge about -

- What types of Python code distribution exist?
- Why building a web application can be a good choice?
- What a web application is
- How content gets delivered over the Internet
- What web hosting means
- Which hosting providers exist and which one to use
- Client-server architecture

Project Hardware/Software

- Operation System – Windows / Open Source (Linux / Ubuntu / CentOS)
- Programming Language – Python , Flask, Django
- Database – MySQL, MongoDB
- Containers – Kubernetes, Dockers
- Cloud – AWS / Azure / GCP
- Tools – Pip, Third-party library, microservices, APIs

Project Design -

Distribute Your Python Code - Traditionally, there are four approaches you can use to distribute your code so that others can work with your programs:

1. Python library / packages
2. Standalone program
3. Python web application
4. Python REST APIs

Overall Design describe applications attributes, methods/functionality and relationship.

Overall problem-solving Approach.

1. Use appropriate user defined classes like University, College, Student, Marksheet etc. to store entries
2. Use appropriate data structures like list, tuple, set, dictionaries.
3. Design UI for Add/update/delete/serialize operations for University Administration system
4. Modularize application wherever possible by creating REST API's for resource based operations (CRUD)

Object Oriented class Design

1. Correct Data abstraction and data hiding.
2. Correct encapsulation and interfaces.
3. Single responsibility principle.

Version Control and CICD

1. Set up a GIT repository(use public GIT) for version control
2. Automate the Build and Deployment process using CICD (eg: AWS or Azure or GCP CICD services)

Common Coding practices to be used

1. Class/method level comments.
2. Separate implementation files for classes.
3. Meaningful class / method / object names
4. Error / exception handling / validation (ex : Check file / database connection open)

Attributes of Application - University Administration system

University Administration system maintains detailed records of all the Colleges and Students as well as and the Examination and the Result department.

The University data file should contain following information:

University Name, University Code, list of collages etc.

Details of College information

In this section of project keeps the record of college id, college name, college location, college running the stream, and list of students.

Details of Student formation

In this section of project keeps the record of student id, student name, student address, father's name, contact number, degree stream, std code, marksheet details.

Details of Marksheet formation

In this section of project keeps the records of subject Name, subject Marks, percentage, and result status (Pass / Fail), Examination class, Examination year.

Implementation and expected correct output

Improve the User Interface of Your Web Application

To improve the user interface and user experience of your web app, you'll need to work with languages other than Python, namely front-end languages such as HTML, CSS, and JavaScript. This case study avoids going into these as much as possible, to remain focused on using Python.

However, if you want to add an input box to your web app, then you'll need to use *some* HTML.

Collect User Input

Start by creating a <form> element on your landing page.

- <form> Element
  - Input Box
  - Submit Button
  - Receive User Input
  - Escape User Input
  - Process User Input

- Important Functions/Methods and Their Purposes -
  1. Create a University Administration system **web App Form** which allow to perform following actions.
     i. Add the last year academic details of student which will be helpful for current year fresh admission.
     ii. Admission process for current year on the basics of last year academic report. If result status has pass, then allow for next year else do the admission for same class.
        a. Note – Passing percentage is 40%

iii. Search Operations –
    1. Search student details by collage and academic year.
    2. Search student details by stream and academic year
    3. Find the percentage of specific student after exam.
    4. Find the marks of specific student after exam.

iv. Count Operation –
    1. Total collages under University
    2. Total number of students in university.
    3. Total number of students per collage.
    4. Total streams in University
    5. Total streams in collage
    6. Total number of students for specific stream as per collage name.
    7. Total number of students for specific stream in University.

v. Update Operation –
    1. Update Collage Name and Collage Code
    2. Update Collage Streams
    3. Update Student details as per collage name and year.

vi. Delete Operation –
    1. Delete Collage Name and Collage Code
    2. Delete Collage Streams
    3. Delete Student details as per collage name and year

vii. Fill Exam form
    1. This Button perform the Fill Exam Form operation on college information from College object and student information from Student object. Which will be initialize the student academic subject for Exam result.

viii. Appear for Exam
    1. This Button to perform the Appear for Exam operation for All colleges on college information from College object and student information from Student object. Which will be initialize the student academic Score/Marks as per the given subject for Exam result.
        i. Note – use rand() function to initialize marks, marks range between 0 to 100.

ix. File handling operation is an important part of any web application.
    1. create student.csv, student.xls, student.json, collage.csv, collage.xls, collage.json, university.csv, university.xls, and university.json format.
    2. Perform read/write operation with operations with university, collage, and student objects and store details in above files.

x. Database Operation - Python can be used in database applications. create a database record set in following DB
    1. MySQL (Structure database)

2. MongoDB (Structure, semi-structure, and unstructured database)
   i. Note – perform DDL and DML operations with university, collage, and student table(database)/ document.

xi. **Deploy** your code to Cloud (AWS/ Azure / Google App Engine)
   i. Choose a Hosting Provider: Using cloud will give you a good start in learning about deploying Python code to the web as it strikes a balance between abstracting away complexity and allowing you to customize the setup.
   ii. When choosing a web hosting provider, you need to confirm that it supports running Python code. Many of them cost money, but this case study will stick with a free option that's professional and highly scalable yet still reasonable to set up:eg: Google App Engine.
   iii. Note: Google App Engine enforces daily quotas for each application. If your web application exceeds these quotas, then Google will start billing you. If you're a new Google Cloud customer, then you can get a promotional free credit when signing up.
   iv. Google App Engine is part of the Google Cloud Platform (GCP), which is run by Google and represents one of the big cloud providers, along with Microsoft Azure and Amazon Web Services (AWS).

2. Create a Docker Image and Upload It to Docker Hub.
   a) Package the University Administration system web application into a Docker image. containerize this application, push it to a Docker image registry, and then run the containerized application.
   b) Join a Container to the Network Namespace of Another Container.
   c) Modifying the University Administration system Web App to Connect to a Backend Datastore
   d) Create Services that will act as connectors to map the external requests to the destination pods so that we can access the pods externally without entering the cluster.
   e) Increase the number of replicas running the application by updating the replicas field of a Deployment spec.(scale a Kubernetes application up and down.)

Testing

Testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies the specified requirements. Normally, most of the testing and debugging is done after the system has been implemented. A large percentage of errors are discovered during testing originates in the requirement and design phase. The techniques that have been proposed for unit testing include the following:

Path testing: Each possible path from input to output is traversed once.

Branch testing: Each path must be traversed at least once.

Functional testing: Each, functional decomposition is tested at least once.

Special value testing: Testing for all values assumed to cause problems.

All the modules that have been developed before and tested individually are put together (integrated) in this phase and tested as a whole system. After doing this in the project will be almost ready to be delivered.

***** Implement at least 2** testing methods as mentioned above in your application

Subjective Assignment Weightage:

*Design	25%
• Use of Web application and Object-Oriented Concepts • Class & object relationships, databases, problem solving approach • Overall design of the application • Use of REST API's	
Implementation	**55%**
• Implementation of specific points(functional requirements) expected in the web application • Valid output without any crash (session timeout) • No compilation errors (less overall weightage) • Use of GIT Version Control and CI CD	
Language Coding Practices	**10%**
• Class/method level comments. • Separate implementation files for classes. • Meaningful class / method / object names • Error / exception handling / validation (ex : Check file / database connection open)	
Miscellaneous	**10%**
Total	**100**