Project Epsilon

System Design Document

Team member utorid’s:
venturo4, fungcore, siniat, gandhihr, hameed10, liweiyu, louiskob

Table of Contents

CRC Cards

App

Class Name: app	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Starts the application, ties all the classes together, renders the HTML.	**Collaborators**: Every class.

DAO

Class Name: DAO	
Parent Class: N/A **Subclasses**: DAOComapny, DAOCompanyTag, DAOIndustry, DAORequest, DAORole, DAORStatus, DAOTag, DAOTeam, DAOType, DAOUser	
Responsibilities: Handles generic CRUD queries with the db.	**Collaborators**: None

Request (Deprecated)

Class Name: Request	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its status, knows its requesting user, knows its requested team, knows when it was created and when it was updated. Has a method to turn it into a str.	**Collaborators**: User, Company, RStatus

Team

Class Name: Team	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its user and their role. Has a method to turn it into a str.	**Collaborators**: User, Company, Role

User

Class Name: User	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its role, its type, its name, contact information, and description. Has a method to turn it into a str.	**Collaborators**: Role, Type

Company

Class Name: Company	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its name, description, create date and industry. Has a method to turn it into a str.	**Collaborators**: Industry

Industry

Class Name: Industry	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its name. Has a method to turn it into a str.	**Collaborators**: None

Tag

Class Name: Tag	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its name and industry. Has a method to turn it into a str.	**Collaborators**: Industry

CompanyTag

Class Name: CompanyTag	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its company and corresponding tag. Has a method to turn it into a str.	**Collaborators**: Company, Tag

Service

Class Name: Service	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its corresponding user and service type. Knows its title, description, price and link. Has a method to turn it into str.	**Collaborators**: ServiceTypes, User

TeamCode

Class Name: TeamCode	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its corresponding team and generated code. Has a method to turn it into str.	**Collaborators**: Teams

JobPosting

Class Name: JobPosting	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has an identifier, knows its posting Company, knows its title, knows its description, knows when it was created and knows whether it's an active posting. Has a method to turn it into a str.	**Collaborators**: Company

JobApplication

Class Name: JobApplication	
Parent Class: N/A **Subclasses**: ApplicantDetail	
Responsibilities: Has an identifier, knows its status, knows its user (applicant), knows its job posting, knows when it was created. Has a method to turn it into a str.	**Collaborators**: JobPosting, User, RStatus

RStatus

Class Name: RStatus (Enum)	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has names (accepted, rejected, pending) and values (1, 2, 3). Has a method to turn it into a str.	**Collaborators**: None.

ServiceType

Class Name: ServiceType (Enum)	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has names (Product Development, Accounting and Bookkeeping, Legal, Marketing, Sales and CRM) and values (1, 2, 3, 4, 5). Has a method to turn it into a str.	**Collaborators**: None.

Role

Class Name: Role (Enum)	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has names (team owner, team admin, team member) and values (1, 2, 3). Has a method to turn it into a str.	**Collaborators**: None

Type

Class Name: Type (Enum)	
Parent Class: N/A **Subclasses**: N/A	
Responsibilities: Has names (startup user, service provider, admin) and values (1, 2, 3). Has a method to turn it into a str.	**Collaborators**: None

ApplicantDetail

Class Name: ApplicantDetail	
Parent Class: JobApplication **Subclasses:** N/A	
Responsibilities: Has an identifier, knows its status, knows its user (applicant) details such as name, contact and description, knows its job posting including job title and job description, knows when it was created. Has a method to turn it into a str.	**Collaborators**: Job Posting, User, RStatus

DAOCompany

Class Name: DAOCompany	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Company table in the db.	**Collaborators**: Company

DAOCompanyTag

Class Name: DAOCompanyTag	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the CompanyTag table in the db.	**Collaborators**: CompanyTag

DAOIndustry

Class Name: DAOIndustry	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Industry table in the db.	**Collaborators**: Industry

DAORequest

Class Name: DAORequest	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Request table in the db.	**Collaborators**: Request

DAORole

Class Name: DAORole	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Role table in the db.	**Collaborators**: Role

DAORStatus

Class Name: DAORStatus	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the RStatus table in the db.	**Collaborators**: RStatus

DAOService

Class Name: DAOService	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Services table in the db.	**Collaborators**: ServiceType, User

DAOServiceType

Class Name: DAOServiceType	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the ServiceTypes table in the db.	**Collaborators**: None

DAOTeam

Class Name: DAOTeam	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Team table in the db.	**Collaborators**: Team

DAOType

Class Name: DAOType	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Type table in the db.	**Collaborators**: Type

DAOUser

Class Name: DAOUser	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the User table in the db.	**Collaborators**: User

DAOTeamCode

Class Name: DAOUser	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the TeamCode table in the db.	**Collaborators**: TeamCode

DAOTag

Class Name: DAOTag	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the Tag table in the db.	**Collaborators**: Tag

DAOJobApplication

Class Name: DAOJobApplication	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the JobApplication table in the db.	**Collaborators**: JobApplication, ApplicantDetail

DAOJobPosting

Class Name: DAOJobPosting	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the JobPosting table in the db.	**Collaborators**: JobPosting

DAOProfilePic

Class Name: DAOProfilePic	
Parent Class: DAO **Subclasses**: N/A	
Responsibilities: Handles CRUD queries related to the ProfilePic table in the db.	**Collaborators**: N/A

FormIncompleteError

Class Name: FormIncompleteError(Exception)
Parent Class: Exception

Subclasses: N/A	
Responsibilities: Has a message of the error. Has a method to turn it into a str.	**Collaborators**: None.

InputInvalidError

Class Name: InputInvalidError(Exception)	
Parent Class: Exception **Subclasses**: N/A	
Responsibilities: Has a message of the error and name of object causing the error. Has a method to turn it into a str.	**Collaborators**: None.

ObjectExistsError

Class Name: ObjectExistsError(Exception)	
Parent Class: Exception **Subclasses**: N/A	
Responsibilities: Has a message of the error and name of object causing the error. Has a method to turn it into a str.	**Collaborators**: None.

ObjectNotExistsError

Class Name: ObjectNotExistsError(Exception)	
Parent Class: Exception **Subclasses**: N/A	
Responsibilities: Has a message of the error and name of object causing the error. Has a method to turn it into a str.	**Collaborators**: None.

Software Architecture

Description

Project Epsilon is a Startup Marketplace application in current development. The architecture of this project is a **Three-Tier Architecture**. This means the application will have a

presentation tier, a logic or application tier, and a data tier.[1] All of these are running in their own infrastructure. By choosing this architecture, the team is allowed to run the development of these tiers independently and avoid conflict.

Presentation Tier

For the front end of the application, we will be using the **Flask** framework (Jinja2 template) along with **bootstrap**, to assist in styling the webpage and displaying/hiding elements depending on the type of user and the role of the user.

Application Tier

The application tier houses the project logic and API. We will use the **Flask** framework in Python code. Flask offers the flexibility we need as it is compatible with a lot of other frameworks such as Docker. Allowing for design changes in the future makes it easier to scale this project according to any new requirements that might come in. Flask is also independent from the front end framework and the database management systems, allowing us to choose the most convenient for us.

Data Tier

The chosen database management system is **MySQL**. We are connecting the application and data tiers through the flask-mysqldb[2] library. We chose a relational DMS because the team has had more exposure to this type of database and it appears the most fit for our database schema as we want the data to be more structured.

---

[1] Education, I. C. (2021, April 5). *Three-Tier Architecture*. IBM. https://www.ibm.com/cloud/learn/three-tier-architecture

[2] Flask-MySQLdb. (n.d.). *Welcome to Flask-MySQLdb's documentation! — Flask-MySQLdb 0.2.0 documentation*. Retrieved June 10, 2021, from https://flask-mysqldb.readthedocs.io/en/latest/

Diagram

The following is the System Architecture Diagram of Project Epsilon. It is a three tier architecture as cited above (https://www.ibm.com/cloud/learn/three-tier-architecture).