Project Epsilon

System Design Document

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

Table of Contents

Table of Contents **1**

CRC Cards **2**

App 2
DAO 2
Request 2
Team 2
User 3
Company 3
Industry 3
Tag 3
CompanyTag 4
RStatus 4
Role 4
Type 4
DAOCompany 5
DAOCompanyTag 5
DAOIndustry 5
DAORequest 5
DAORole 5
DAORStatus 6
DAOTag 6
DAOTeam 6
DAOType 6
DAOUser 6
FormIncompleteError 7
InputInvalidError 7
ObjectExistsError 7

Software Architecture **8**

Description 8
Presentation Tier 8
Application Tier 8
Data Tier 8
Diagram 9

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

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

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.

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

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

DAOTag

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

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

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 **React** framework. React is a javascript library that helps build the graphic user interface of web applications. It is lightweight and scalable[2] and even though it does not have excellent documentation, the community feeds online offer sufficient support.

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[3] 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] Lvova, E. (2021, April 5). *React vs. Vue in 2021: Best JavaScript Framework*. Dzone. https://dzone.com/articles/react-vs-vue-in-2021-best-javascript-framework

[3] 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).