STACK OVERFLOW FOR STUDENTS

Name - Vaibhav Jain Net ID - vj2075
Name - Sharath Katta Sridhar Net ID - sk8671

PART 1 - DETAILS

a) High-level description of your application

Stack Overflow for Students is a dedicated question-and-answer online forum for professional and enthusiast programming students belonging to a particular organization (University). It features questions and answers on a wide range of topics in computer programming. It is created to be a more open alternative to earlier question-and-answer websites such as Experts-Exchange. The website serves as a platform for users to login as an administrator, check all the active users, questions posted by them with their tags and tag categories. Admin can also search questions based on the question body, tags or tag categories.

Moreover, admin searches for a user by its first name or email and get all the activities being performed by it like question posted, answers posted, comments posted on questions and comments posted on answers. Stack Overflow helps people find the answers they need when they need them. It's a Q&A platform to ask questions, learn, and share technical knowledge by the user.

b) Explicitly list all entity sets, relationship sets, and business rules in your application; as you describe the business rules, you should describe all key and participation constraints, -

ER	Student posts Question
Business Rule	A Student may post 0, 1, or multiple questions. Each question is posted by exactly one student.
Restriction	The restriction that each question is posted by exactly one student means that this is a key constraint
ER	Student posts Answer
Business Rule	A Student may post 0, 1, or multiple answers. Each answer is posted by exactly one Student.

Restriction	The restriction that each answer is posted by exactly one student means that this is a key constraint
ER	Question contains Answers
Business Rule	A question may have 0, 1, or multiple answers. Each answer addresses exactly one question.
Restriction	The restriction that each answer addresses exactly one question means that this is a key constraint
ER	Question contains Tags
Business Rule	A question may have 0, 1, or multiple tags. Each tag is associated with some question.
Restriction	The restriction that each tag is associated with some (read “at least one”) question means that this is a participation(total) constraint.
ER	Tags are associated with Tag categories
Business Rule	A tag may save 0, 1, or multiple tag categories. Each tag category can be associated with 0, 1, or multiple tags.
Restriction	The restriction that each tag is associated with 0 or more tag categories means this is an example of partial participation constraint
ER	Student posts comments
Business Rule	A student may post 0, 1, or multiple comments. Each comment is posted by exactly one student
Restriction	The restriction that each comment is posted by exactly one student means this is a key constraint.
ER	Question has comments
Business Rule	A question may have 0, 1, or multiple comments. Each comment is associated with only one question

Restriction	The restriction that each comment is associated with only one question, means that this is a key constraint
ER	Answer has comments
Business Rule	An answer may have 0, 1, or multiple comments. Each comment is associated with only one answer
Restriction	The restriction that each comment is associated with only one answer, means that this is a key constraint
ER	Organization consists of Schools
Business Rule	An Organization may have 1, or multiple Schools. Each School is associated with only one organization
Restriction	The restriction that each school is associated with only one organization, means this is a key constraint. Secondly, the restriction that each organization may have 1 or more schools, means that this is a total participation constraint
ER	School has Departments
Business Rule	A department may have 1, or multiple schools. Each school is associated with exactly one department
Restriction	The restriction that each school is associated with exactly one department means that this is a key constraint. Secondly, the restriction that each school may have 1 or more departments, means that this is a total participation constraint
ER	Department is made up of students
Business Rule	A Department may have 1, or multiple students. Each student can be associated with 1 or multiple departments.
Restriction	The restriction that each student can be associated with 1 or more departments , means this is a total participation constraint.

	Secondly, the restriction that each department may have 1 or more students, means that this is a total participation constraint
ER	Student lives in locations
Business Rule	A student can live in 0,1 or multiple locations. Each location is associated with exactly one student
Restriction	The restriction that each location is associated with only one student , means this is a key constraint.

Entity Sets
Organizations (org_id, org_name)
Schools(school_id, school_name,org_id)
Departments(dept_id, dept_name,school_id)
Locations(location_id, address_line1,address_line2,city,state, zipcode,country)
Students(sid, first_name, last_name, email,password,degree,dept_id,graduation_year,location,dob)
Tag Categories(tc_id, catgegory_name)
Tags(tid, tag_name,tc_id)
Questions(qid, title, body, posted_by,created_at, updated_at)
Question Tags(qt_id, qid, tag_id)
Answers(aid, body, qid, posted_by)
Comments(comment_id, body, posted_at, created_at, updated_at)

Relationship Sets
Organization “consists” Schools
Schools “has” Departments
Departments “is made up of” Students

Student “lives in” Location
Student “posts” Question
Student “posts” Answer
Relationship Sets
Student “posts” Comments
Question “contains” Answers
Question “has” Comments
Answer “has” Comments
Question “contains” Tags
Tag “associates with” Tag Categories

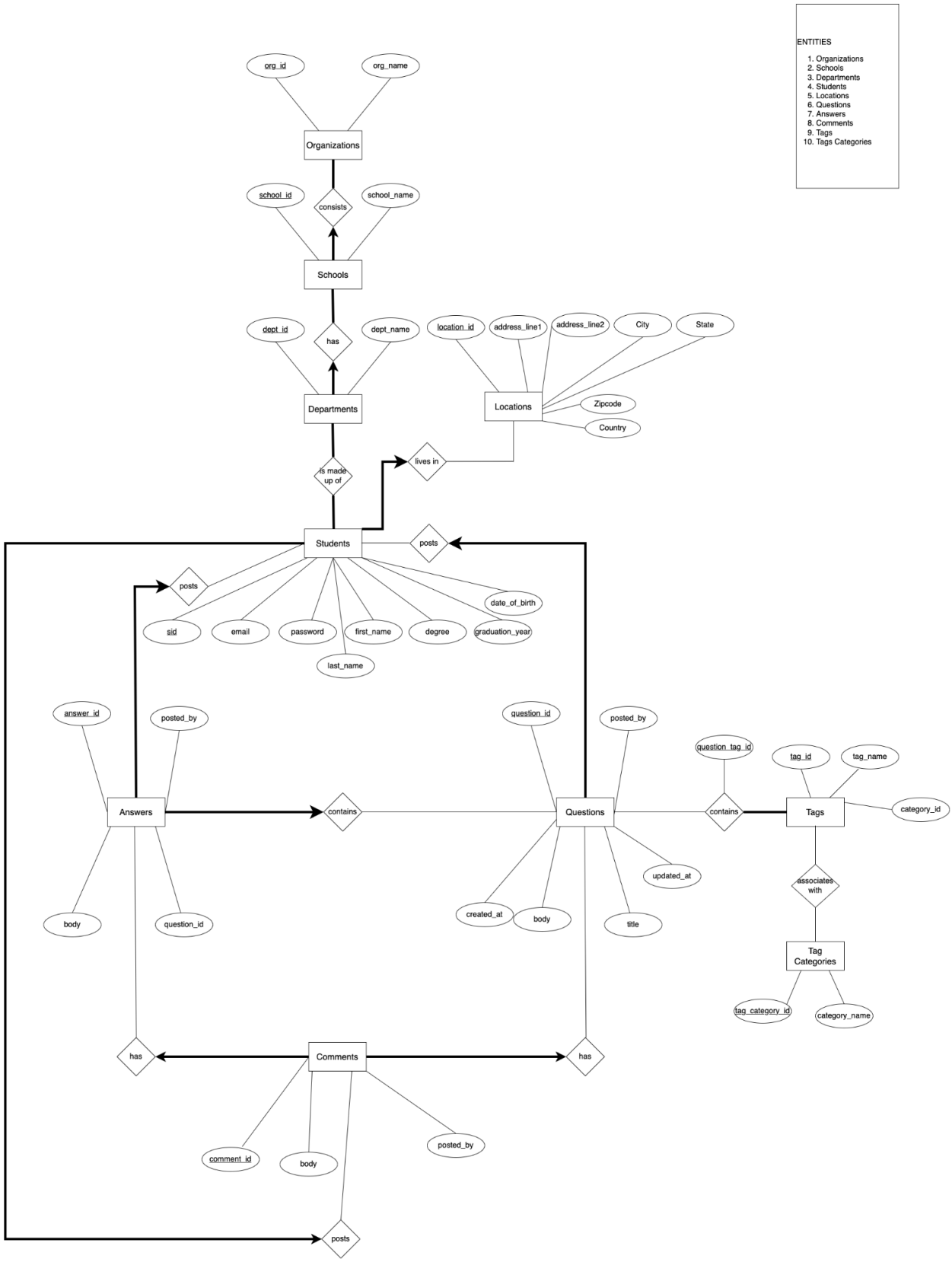
c) Explain how you plan to acquire data for your application,

We acquire dummy data for our application from different sources. Also we are giving admin the right to add more questions on to the portal. The admin is going to login to our application and while interacting with it, the data is being generated and stored in the database so that other users can also interact with the application as described above.

d) Discuss how the user will interact with your database - what kinds of questions they will be able to ask.

- The user can login to the application.
- The user can post answers to multiple questions
- The user can search for a fellow user by its first name or email.
- The user can see all the activities being performed by its fellow users like what and how many questions it asked, answers answers, commented on questions or answers.
- The user can search for a question and see its corresponding details, like who posted that question, tags, tag categories, answers and different types of comments.
- The user can search questions based on the tags associated with the question and answer.
- The user can also browse questions and answers by particular tag categories
- The user can see all its details available in the profile page, details like location, department, school, university, year of graduation, etc.

ER DIAGRAM


ENTITIES
1. Organizations
2. Schools
3. Departments
4. Students
5. Locations
6. Questions
7. Answers
8. Comments
9. Tags
10. Tags Categories
org_id
org_name
Organizations
school_id
consists
school_name
Schools
dept_id
has
dept_name
location_id
address_line1
address_line2
City
State
Departments
Locations
Zipcode
Country
is made up of
lives in
Students
posts
posts
sid
email
password
first_name
degree
graduation_year
date_of_birth
last_name
answer_id
posted_by
Answers
contains
Questions
question_id
posted_by
question_tag_id
tag_id
tag_name
contains
Tags
category_id
associates with
Tag Categories
tag_category_id
category_name
body
question_id
created_at
body
title
updated_at
has
Comments
has
comment_id
body
posted_by
posts

schema.sql

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp" WITH SCHEMA public;

-- Table: public.organizations

DROP TABLE IF EXISTS public.organizations;

CREATE TABLE IF NOT EXISTS public.organizations
(
    org_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    org_name character(128) COLLATE pg_catalog."default" NOT NULL UNIQUE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.organizations
--     OWNER to postgres;




-- Table: public.schools

DROP TABLE IF EXISTS public.schools;

CREATE TABLE IF NOT EXISTS public.schools
(
    school_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    school_name character(128) NOT NULL UNIQUE,
    org_id uuid NOT NULL,
    FOREIGN KEY (org_id)
        REFERENCES public.organizations (org_id)
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.schools
--     OWNER to postgres;



-- Table: public.departments
```

```
DROP TABLE IF EXISTS public.departments;

CREATE TABLE IF NOT EXISTS public.departments
(
    dept_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    dept_name character(128) NOT NULL,
    school_id uuid NOT NULL,
    FOREIGN KEY (school_id)
        REFERENCES public.schools (school_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.departments
--     OWNER to postgres;


-- Table: public.locations

-- DROP TABLE IF EXISTS public.locations;

CREATE TABLE IF NOT EXISTS public.locations
(
    location_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    address_line1 character(64) NOT NULL,
    address_line2 character(64),
    city character(32) NOT NULL,
    state character(32) NOT NULL,
    zipcode integer NOT NULL,
    country character(32) NOT NULL
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.locations
--     OWNER to postgres;




-- Table: public.students

DROP TABLE IF EXISTS public.students;

CREATE TABLE IF NOT EXISTS public.students
(
    sid uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
```

```
    first_name character(32) NOT NULL,
    last_name character(32) NOT NULL,
    email character(64) NOT NULL UNIQUE,
    password character(64) NOT NULL,
    degree character(128),
    dept_id uuid NOT NULL,
    graduation_year bigint,
    location uuid NOT NULL,
    dob date,
    FOREIGN KEY (dept_id)
        REFERENCES public.departments (dept_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (location)
        REFERENCES public.locations (location_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.students
--     OWNER to postgres;




-- Table: public.tag_categories

DROP TABLE IF EXISTS public.tag_categories;

CREATE TABLE IF NOT EXISTS public.tag_categories
(
    tc_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    category_name character(32) NOT NULL UNIQUE
);


-- Table: public.tags

DROP TABLE IF EXISTS public.tags;

CREATE TABLE IF NOT EXISTS public.tags
(
    tid uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    tag_name character(32) NOT NULL UNIQUE,
    tc_id uuid NOT NULL,
    FOREIGN KEY (tc_id)
        REFERENCES public.tag_categories (tc_id)
```

```
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.tags
--     OWNER to postgres;


-- Table: public.questions
-- Table: public.questions

DROP TABLE IF EXISTS public.questions;

CREATE TABLE IF NOT EXISTS public.questions
(
    qid uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    body text NOT NULL,
    title character(64) NOT NULL UNIQUE,
    posted_by uuid NOT NULL,
    created_at timestamp with time zone NOT NULL DEFAULT now(),
    updated_by timestamp with time zone NOT NULL DEFAULT now(),
    FOREIGN KEY (posted_by)
        REFERENCES public.students (sid)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.questions
--     OWNER to postgres;



-- Table: public.answers

DROP TABLE IF EXISTS public.answers;

CREATE TABLE IF NOT EXISTS public.answers
(
    aid uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    body text COLLATE pg_catalog."default" NOT NULL,
    qid uuid NOT NULL,
    posted_by uuid NOT NULL,
    FOREIGN KEY (posted_by)
        REFERENCES public.students (sid)
        ON UPDATE CASCADE
```

```
        ON DELETE CASCADE,
    FOREIGN KEY (qid)
        REFERENCES public.questions (qid)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.answers
--     OWNER to postgres;


-- Table: public.tags_questions

DROP TABLE IF EXISTS public.tags_questions;

CREATE TABLE IF NOT EXISTS public.tags_questions
(
    qid uuid NOT NULL,
    tag_id uuid NOT NULL,
    qt_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    FOREIGN KEY (qid)
        REFERENCES public.questions (qid)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (tag_id)
        REFERENCES public.tags (tid)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.tags_questions
--     OWNER to postgres;


-- Table: public.comments_questions

DROP TABLE IF EXISTS public.comments_questions;

CREATE TABLE IF NOT EXISTS public.comments_questions
(
    comment_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    qid uuid NOT NULL,
    body character(256) NOT NULL UNIQUE,
    posted_by uuid NOT NULL,
    created_at timestamp with time zone NOT NULL DEFAULT now(),
```

```
    updated_at timestamp with time zone NOT NULL DEFAULT now(),
    FOREIGN KEY (posted_by)
        REFERENCES public.students (sid)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (qid)
        REFERENCES public.questions (qid)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.comments_questions
--     OWNER to postgres;


-- Table: public.comments_answers

-- DROP TABLE IF EXISTS public.comments_answers;

CREATE TABLE IF NOT EXISTS public.comments_answers
(
    comment_id uuid NOT NULL DEFAULT uuid_generate_v4() PRIMARY KEY,
    aid uuid NOT NULL,
    body character(256) NOT NULL UNIQUE,
    posted_by uuid NOT NULL,
    FOREIGN KEY (aid)
        REFERENCES public.answers (aid)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (posted_by)
        REFERENCES public.students (sid)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

-- TABLESPACE pg_default;

-- ALTER TABLE IF EXISTS public.comments_answers
--     OWNER to postgres;
```