****

**CS 457/CS 557 – Database Software Design**

*An Voting Website*

1. **Contribution of each member of your group**

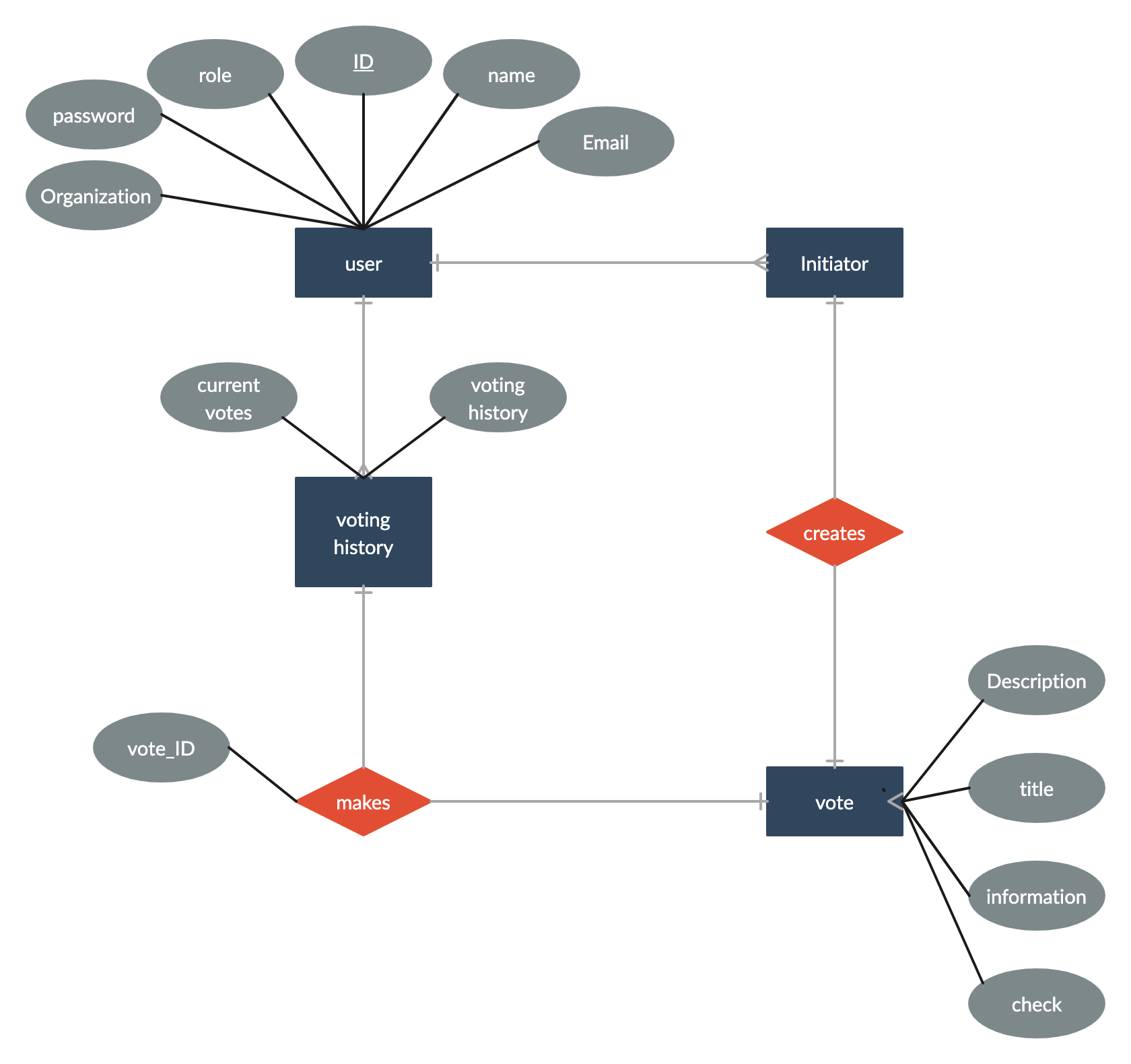
|  |  |
| --- | --- |
| Full name | Contribution |
| Sarvenaz Daryaei | system design, HTML, content collection, SignIn Page, Report. |
| Reza Ghasemi | system design, CSS, Bootstrap, UI, SignUp Page, Add account. |
| Masoumeh Goudarzi | system design, javascript, EJS, Voting Pages, new voting post. |
| Yao Lai | system design, Node, Express, React, Voting Pages, edit voting post, deploy in the cloud. |

1. **Project description**

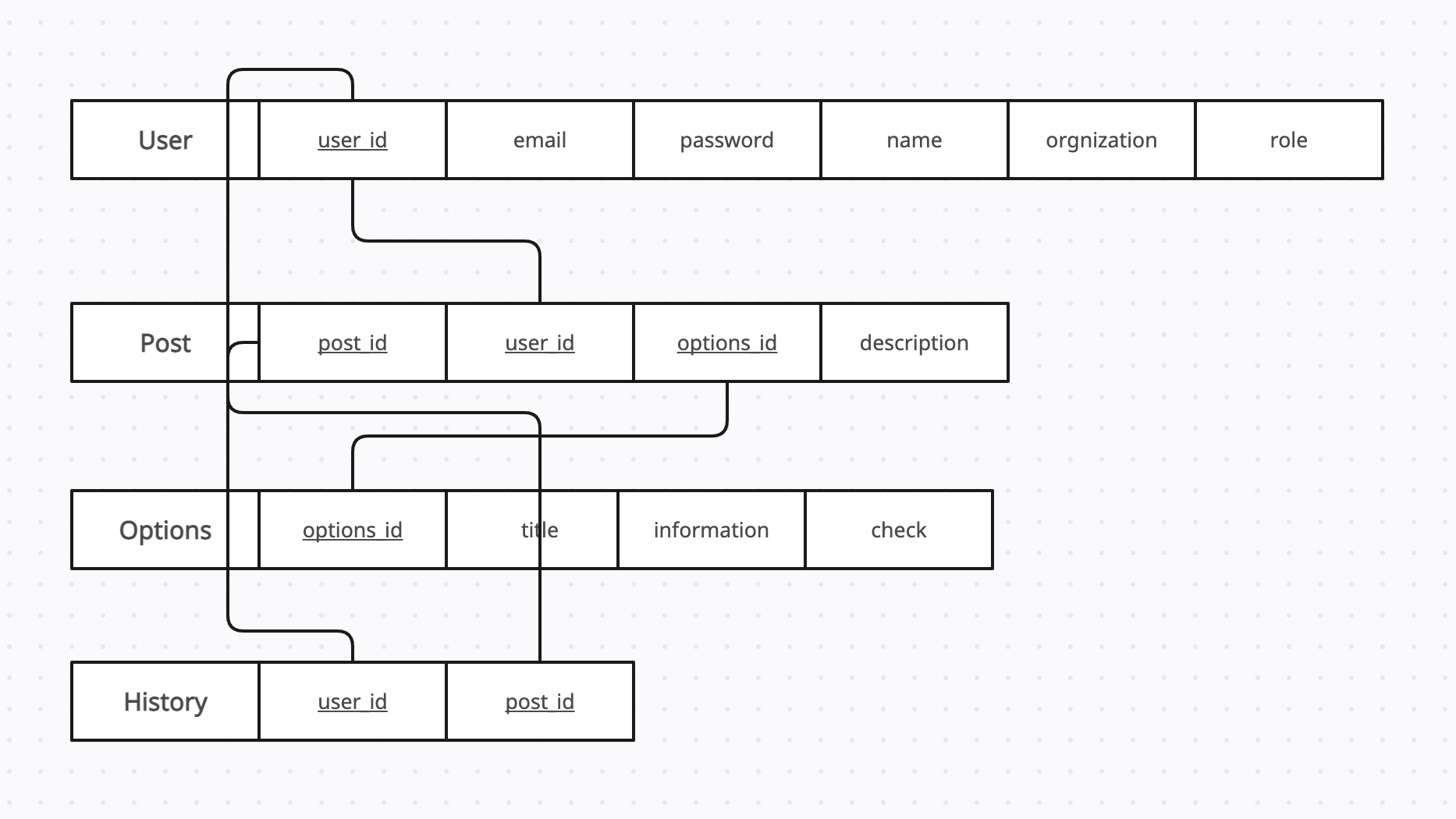
* Give a description of your project:
  + A common task in any organization is to collect structured information from a group of people. And voting is an collective decision making process happen frequently in any organization. The voting website is trying to easy the process of voting.
  + Account management: the website has signIn, signUp pages can let the user access their account and participate in the voting of their organization.
  + Voting: the initiator can send a new voting post to the participants. They can edit the voting, delete the voting, view the voting results.
* Explain the different stages to realize your project.
  + system design: in theis stage, group members discuss what the project will look like, what functionality should we implement.
  + UI design, One of the import part is the user inteface, we first draw some sketch, then use boostrap make it more official. Collect iamges an icons to decorate the website.
  + After now the functions and UI of the voting system, we now know what informations we should collect and manages. We began to design the database. Drawing ER, EER, Relation model and build the database.
  + We have the statistic website and the database, we need to interactive with the database, so we use Node, express, ejs, javascript to collect information user input and send it to the database, after insert, update, delete, search operations, return the information back to the website.
  + Modify and testing: there are some bugs and new functions we want to add during the development, so we continue testing and modifying our web project.
  + Deploy in the coloud service: the project can only run in the local computer before, we want to other users can access our voting service through the internet. So we deploy our project in the cloud and everyone can access it.

1. **Modeling Scheme**

**ER:**

****

**Relational Model:**



1. **SQL queries**

**Create tables:**

CREATE TABLE user (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

name VARCHAR(255) NOT NULL,

organization VARCHAR(255),

role ENUM('admin', 'voter') NOT NULL

);

CREATE TABLE post (

post\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

description TEXT NOT NULL,

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE options (

options\_id INT AUTO\_INCREMENT PRIMARY KEY,

post\_id INT NOT NULL,

title VARCHAR(255) NOT NULL,

information TEXT,

check BOOLEAN DEFAULT false,

FOREIGN KEY (post\_id) REFERENCES post(post\_id)

);

CREATE TABLE history (

user\_id INT NOT NULL,

post\_id INT NOT NULL,

PRIMARY KEY (user\_id, post\_id),

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

FOREIGN KEY (post\_id) REFERENCES post(post\_id)

);

**Insert:**

1. Insert a new user:

INSERT INTO user (email, password, name, organization, role) VALUES ('john@example.com', 'password', 'John Smith', 'ACME Inc.', 'voter');

1. Insert a new post:

INSERT INTO post (user\_id, description) VALUES (1, 'Which is your favorite color?');

INSERT INTO options (post\_id, title, information) VALUES (1, 'Red', 'A bright, warm color');

INSERT INTO options (post\_id, title, information) VALUES (1, 'Green', 'A cool, refreshing color');

INSERT INTO options (post\_id, title, information) VALUES (1, 'Blue', 'A calming color');

1. Add a user's vote to the history table:

INSERT INTO history (user\_id, post\_id) VALUES (1, 1);

1. Add a new option to a post:

INSERT INTO options (post\_id, title, information) VALUES (1, 'Yellow', 'A sunny, cheerful color');

**Update:**

1. Update a user's name:

UPDATE user SET name = 'Jane Doe' WHERE user\_id = 1;

1. Update a post’s description:

UPADATE post SET description = ‘Which is your favorite fruit?’ WHERE post\_id = 1;

1. Mark an option as checked:

UPDATE options SET check = true WHERE options\_id = 1;

1. Change a user's role:

UPDATE user SET role = 'admin' WHERE user\_id = 1;

**Delete:**

1. Delete a user:

DELETE FROM user WHERE user\_id = 1;

1. Delete a post and its options:

DELETE FROM post WHERE post\_id = 1;

DELETE FROM options WHERE post\_id = 1;

1. Remove a vote from the history table:

DELETE FROM history WHERE user\_id = 1 AND post\_id = 1;

1. Delete an option from a post:

DELETE FROM options WHERE options\_id = 1;

**Join Queries:**

1. Get all posts with their options:

SELECT post.post\_id, post.description, options.title, options.information

FROM post

JOIN options ON post.post\_id = options.post\_id;

1. Get all users who have voted on a post:

SELECT user.name, user.email

FROM user

JOIN history ON user.user\_id = history.user\_id

WHERE history.post\_id = 1;

1. Get all posts with their options and the number of votes for each option:

SELECT post.post\_id, post.description, options.title, options.information, COUNT(history.user\_id) AS votes

FROM post

JOIN options ON post.post\_id = options.post\_id

LEFT JOIN history ON options.options\_id = history.options\_id

GROUP BY options.options\_id;

1. Get all posts and their options, with the name of the user who created the post:

SELECT post.post\_id, post.description, options.title, options.information, user.name

FROM post

JOIN options ON post.post\_id = options.post\_id

JOIN user ON post.user\_id = user.user\_id;

**Why important**: Because simply that is how we can know what user participate in what voring and what did they vote for. without join, we cannot know the relaitonship between the user and the posting vote.

1. **Implementation**

Describe the tools you used: using vscode to developing the whole website. postsql as databse software. Deploy in cyclic.sh. Coupling: We tried to use data coupling because the modules are communicating by data and they are a kind of data couples and it doesn’t have tramp data.

1. **Screenshots of the user interface of your final project**

You can always check the website through the following link instead of install in your local computer: <https://nice-lime-goldfish-vest.cyclic.app,> if you have problem connecting to the link, don’t hesitate to contact me through email: [ylai20@ubishops.ca,](mailto:ylai20@ubishops.ca,) sometimes it can be updating the project. The last version of the source code, you can check in the github link below: <https://github.com/Cosmosly/LetsVote.>

Below we provide some screenshots of our UI:

|  |  |
| --- | --- |
|  | Home Page |

|  |  |
| --- | --- |
|  | SQL:   1. Insert new account   INSERT INTO user (email, password, name, organization, role) VALUES ('newuser@example.com', 'password', 'New User', 'New Organization', 'voter');   1. Find if email already exist   SELECT \* FROM user WHERE email = 'email@example.com'; |

|  |  |
| --- | --- |
|  | SQL:  Check password and email :  SELECT \* FROM user WHERE email = 'email@example.com' AND password = 'password'; |

|  |  |
| --- | --- |
|  | SQL:  create new vote  INSERT INTO post (user\_id, description, title) VALUES (1, 'Description of the vote', 'Title of the vote'); |

|  |  |
| --- | --- |
|  | SQL:  view all the vote:  SELECT post\_id, title, description FROM post WHERE user\_id = 1; |

|  |  |
| --- | --- |
|  | SQL:  add new vote options:  INSERT INTO options (post\_id, title, information, checked) VALUES (1, 'Option title', 'Option information', false);  delete vote options:  DELETE FROM options WHERE option\_id = 1; |

1. **Conclusions and future works**

Security, right now we don’t apply any security methods in our database. In the future we want to increase the level of our database. 1. Database Encrption. 2. Salting and hashing the password.

Send the vote to the people you selected. This part is very important, unfortunately, don’t have enough time to finished. Very continue to improve our project and update in the github.

Scaling, right now we just consider we have few users. When one day we have millions of users, we definetly need to consider the response time of the database. Relational database like postsql have clear logic, but when it comes to large scale, it might be pretty slow. Like twitter using mongoDB instead, we are considering using non-relational database when we need to scale up the size of our database.

Future changes will update in the website, you can always check in the link below: <https://nice-lime-goldfish-vest.cyclic.app>