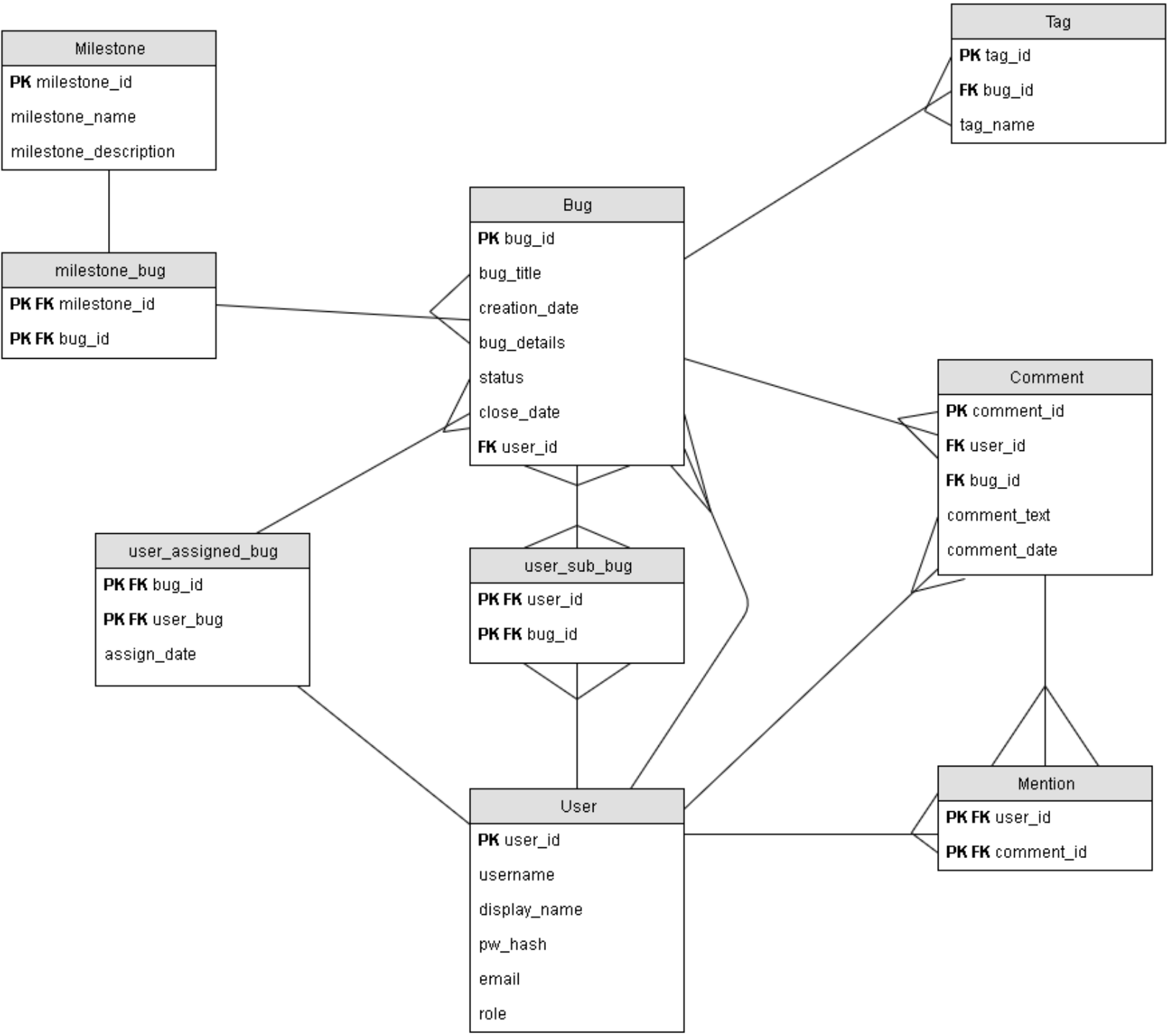
Group: RR3

Dennis Bruce
Gilbert Amador
Michael Pritchett

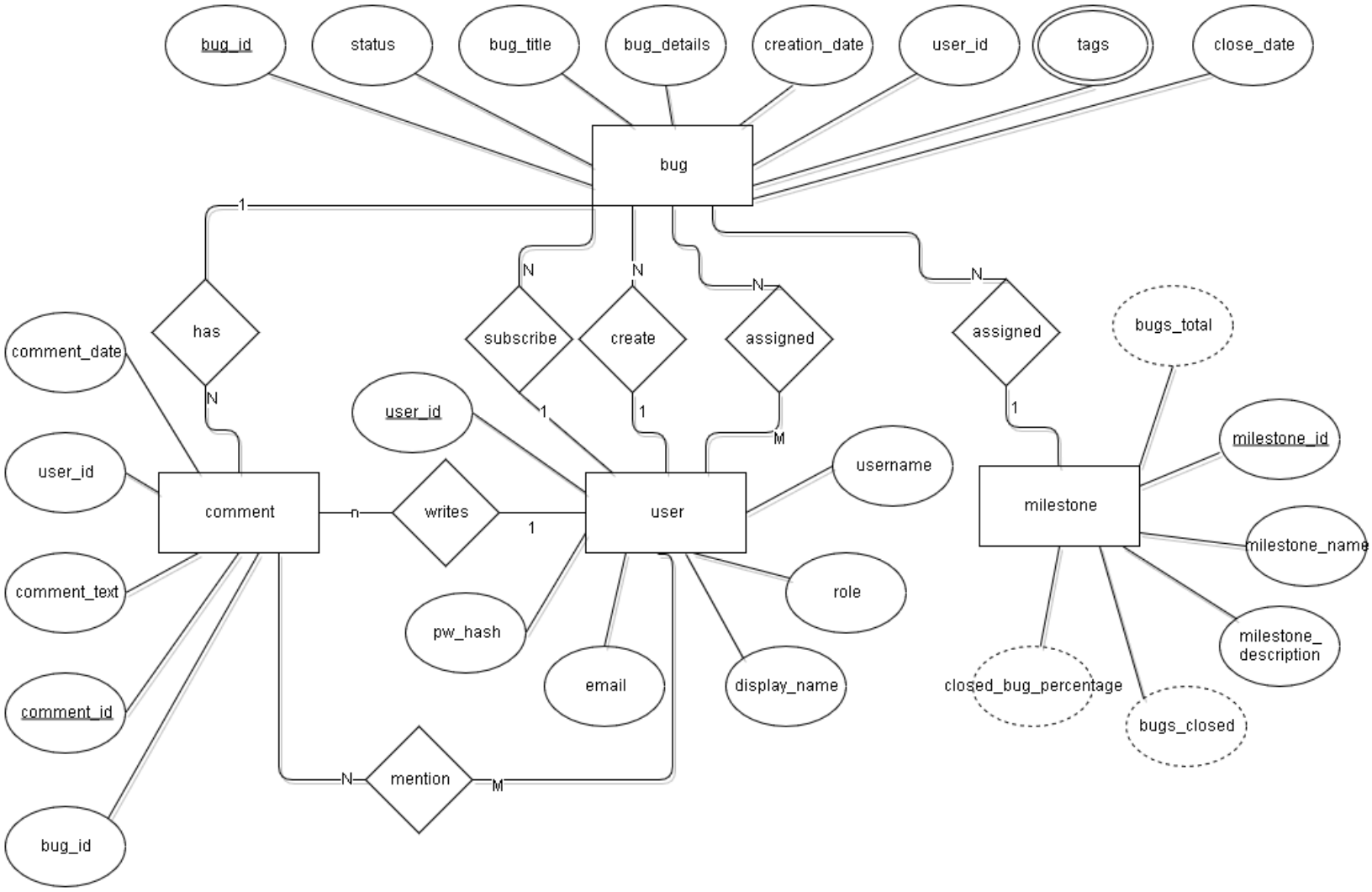
Instructions

To grade our project, you should click through all the links to try to use the basic functions and test the results to the database.

Physical Model



Entity-Relationship Diagram



Database DLL

```
create TABLE bug (
    bug_id integer not null default
nextval('bug_bug_id_seq'::regclass),
    bug_title varchar not null,
    creator_id integer not null,
    creation_date timestamp not null default now(),
    bug_details varchar default ''::character varying,
    status stat not null default 'open'::stat,
    close_date timestamp,
    PRIMARY KEY (bug_id),
    FOREIGN KEY (creator_id) REFERENCES "user" (user_id)
);
CREATE UNIQUE INDEX bug_pkey ON bug (bug_id);
CREATE UNIQUE INDEX bug_bug_title_key ON bug (bug_title)create
TABLE comment (
    comment_id integer not null default
nextval('comment_comment_id_seq'::regclass),
    author_id integer not null,
    bug_id integer not null,
    comment_text varchar not null,
    comment_date timestamp not null default now(),
    PRIMARY KEY (comment_id),
    FOREIGN KEY (author_id) REFERENCES "user" (user_id),
    FOREIGN KEY (bug_id) REFERENCES bug (bug_id)
);
CREATE UNIQUE INDEX comment_pkey ON comment (comment_id)create
TABLE mention (
    user_id integer not null,
    comment_id integer not null,
    PRIMARY KEY (user_id, comment_id),
    FOREIGN KEY (user_id) REFERENCES "user" (user_id),
    FOREIGN KEY (comment_id) REFERENCES comment (comment_id)
);
CREATE UNIQUE INDEX mention_pkey ON mention (user_id,
comment_id)create TABLE milestone (
    milestone_id integer not null default
nextval('milestone_milestone_id_seq'::regclass),
    milestone_name varchar not null,
    milestone_description varchar not null default ''::character
varying,
    PRIMARY KEY (milestone_id)
);
CREATE UNIQUE INDEX milestone_pkey ON milestone (milestone_id);
```

```
CREATE UNIQUE INDEX milestone_milestone_name_key ON milestone
(milestone_name)create TABLE tag (
    tag_id integer not null default
nextval('tag_tag_id_seq'::regclass),
    bug_id integer not null,
    tag_name varchar not null,
    PRIMARY KEY (tag_id),
    FOREIGN KEY (bug_id) REFERENCES bug (bug_id)
);
CREATE UNIQUE INDEX tag_pkey ON tag (tag_id)create TABLE "user"
(
    user_id integer not null default
nextval('user_user_id_seq'::regclass),
    username varchar not null,
    display_name varchar not null,
    pw_hash varchar not null,
    email varchar not null,
    role user_role not null default 'user'::user_role,
    PRIMARY KEY (user_id)
);
CREATE UNIQUE INDEX user_pkey ON "user" (user_id);
CREATE UNIQUE INDEX user_username_key ON "user" (username);
CREATE UNIQUE INDEX user_display_name_key ON "user"
(display_name);
CREATE UNIQUE INDEX user_email_key ON "user" (email)create TABLE
user_assigned_bug (
    user_id integer not null,
    bug_id integer not null,
    assign_date timestamp not null default now(),
    PRIMARY KEY (user_id, bug_id),
    FOREIGN KEY (user_id) REFERENCES "user" (user_id),
    FOREIGN KEY (bug_id) REFERENCES bug (bug_id)
);
CREATE UNIQUE INDEX user_assigned_bug_pkey ON user_assigned_bug
(user_id, bug_id)create TABLE user_sub_bug (
    user_id integer not null,
    bug_id integer not null,
    PRIMARY KEY (user_id, bug_id),
    FOREIGN KEY (user_id) REFERENCES "user" (user_id),
    FOREIGN KEY (bug_id) REFERENCES bug (bug_id)
);
CREATE UNIQUE INDEX user_sub_bug_pkey ON user_sub_bug (user_id,
bug_id);
```

Page Descriptions

PAGE	URL	DESCRIPTION	STATUS
Default	/	a user that isn't logged in can browse bugs and look at comments	finished
User splash page	/home/<user_id>	like the default page but takes user to user specific links	finished
Registration	/reg	Uses the user table, making a list of all the people that have registered.	finished
Login	/login	Uses the user table. Is where the users login so that they are able to make more bugs and comment on existing bugs.	finished
User profile	/user/<int:user_id>	The profile page of the user. Uses the user and bug table, so that people who visit the profile page can see what bugs they are a part of.	finished
Bug description	/bug/<int:bug_id>	Uses the bug, mention, comment, tag, and subscription tables.	finished
Bug user	/bug/<int:bug_id>/<int:user_id>	same as bug description but has subscribe link for user	finished
Bug Comments	/bug/<int:bug_id>/comments/<int:user_id>	page that has comments for bugs	finished
Subscribe	/bug/<int:bug_id>/<int:user_id>/<string:subscribed>	subscribes or unsubscribes a user from a bug	finished
Milestone list	/milestone	list of all the milestones	finished
Milestone	/milestone/<int:mile_id>	Uses the bug and milestone tables. Shows all the bugs associated with a certain milestone.	finished
Browse	/browse/<string:browse>/<string:filter>/	Uses the bug and user tables. Let's a user search for a specific bug or browse through the list of bugs.	finished
Search	/search/<string:text>	Lets user search for a string in a bug title or description	finished
User browse	/user_browse/<int:user_id>/<string:browse>/<string:status>/	lets user search bugs related to them	finished
Bug creation	/newbug	Uses the bug and user tables. Allows a user to create a new bug.	finished
Newsfeed	/newsfeed/user/<int:user_id>	Uses the bug, user, and subscribe tables. Is a place that shows all the bugs that someone is working on and what they may be subscribed to when they first happen.	finished

Mile-Stone Feature

The mile-stone feature will allow the users to track the progress of a bug. A mile-stone will be assigned a bug, meaning the mile-stone table will have a relationship with the bug table and a mile-stone can have multiple bugs. Mile-stone will look at its bugs' statuses to derive the number of closed bugs. A progress percentage can be derived by the number of closed bugs divided by the total number of bugs.