

## SW Engineering CSC648/848 Summer 2019

### “The Better City”

Team: 04

#### Team Members:

##### Local Team

Maithri Chullikana House: Team Leader      Email: [mchullikanahouse@mail.sfsu.edu](mailto:mchullikanahouse@mail.sfsu.edu)

Soham Harshadbhai Prajapati: Front-End Leader

Patreck Benjamin Raymore: Back-End Leader

Bahar Moattar: GitHub Master, Document Editor

Ziping Huang: Back-End Developer

Aashutosh Bajgain: Back-End Developer

Fengze You: Front-End Developer

Zeyuan Cai: Fron-End Developer

Milestone: 2

Date: 07/28/2019

History Table: Revised Version Per Instructor's Feedback

## 1. Data Definitions V2

**Users:** There are 3 categories of users for this site-Unregistered Users, Registered Users and Admins. We won't be storing any data of the unregistered users. The 'Users' entity will be used to store the information of registered users and admins. For registration, each user must input a Name, valid email, and password.

1. username: The Name input by user during registration will be stored as a string.
2. Id: In users' entity, User\_Id is the unique string that is used as a username Id. It is primary key and will be stored as an int.
3. Password: Password will be stored as an unconvertable encrypted string. Only the user will know what the password is.
4. Email: The email in the user's entity will be stored as a string. This email is required for the user to log into the website
5. Role: This will differentiate whether you log in as an admin or regular user. Since we do not need registration for admin, the 'role' will give us access to choose as an admin or a user.
6. issues: They will be a foreign key used to track the issues reported by the user.

**Issues:** The Issues entity will be used to store details of each issue reported by users. Uploading photos for issues is optional. The location is mandatory field to submit issues, it can be inputted manually or gathered via geo-location. The status of the issue can be toggled between three states: Open, Work in progress, and Resolved. However, the state of the status can only be altered by a user with the role "admin". Furthermore, each issue must have a description in order to be submitted. The time stamp will be used to keep track of how long each issue has been open.

1. Issue\_Id: Primary key for issues so that each issue is reported are unique. This will be stored as an int.
2. Title: It will store the title of the issue as string
3. Photo: For the reporting issue entity, the photo can be uploaded to the posts, it will give the user as well as the admins a better understanding of what is the status of that issue. Path to the photo will be stored as string.
4. Location: The location shows where the issue has happened. It will be stored as string.
5. Status\_id (e.g. Open/Work in progress/Resolved): This is a foreign key that shows the status of the issues. This will help the admins to do their job more efficiently.
6. Description: User will give a brief summary of the issue's situation. It will be stored in string.
7. User\_Id: This is the foreign key that will show who reported the issue that matches the User\_Id in users table.
8. Timestamp: The timestamp will show when the user reported the issue. It will give users and admins a better idea of when that issue happened.
9. Category\_id: This is the foreign key that will show the category of the issue. It will give the admin and user a better understanding of what that issue is about.

**Category:** The Category entity will store all the different category of the issues, so when the user selects category, it will show the issues reported in that category.

1. Category\_Id: It is a primary key and will be stored as an integer. Each id will be linked to one specific category.
2. Title: It will have a title for each category so user and admin can select it for reporting issues. It will be stored as string.

**Comments:** The comments entity will store details of each comment that is posted by users on a particular issue.

1. Comment\_Id: This is to identify each comment; it will be stored as an integer. It is a primary key.
2. Comment: This stores comments from users in string format. It will show what other users thought about the posts.
3. TimeStamp: It will show the time when the user commented on the issue.
4. Issue\_id: It is a foreign key. It shows which comments are matched with which issues.
5. User\_Id: It is a foreign key. It shows which comments are matched with the user.

**Status:** The status entity will store the details of status\_id and corresponding status.

1. Status\_id: It is a primary key that will store an integer id for a status
2. Status: There are three statuses:
  - Open
  - In progress
  - Resolved

## 2. Functional Requirements V2

### **Priority 1**

#### Unregistered Users:

1. Users shall be able to register an account.  
Name, Email, and password are mandatory for a user to register an account.
2. Users shall be able to view issues.
3. Users shall be able to browse through the issues.
7. Users shall be able to search reported issues.
13. Users shall be able to filter issues by categories, and status.  
User can choose issues that belong to a certain category or status.

#### Registered Users:

- 8.+ Unregistered user's functions.
9. Users shall be able to log into the website.
10. Users shall be able to report environmental issues.

#### Admins:

18. Admin shall be able to publish a report.
19. Admin shall be able to change the status of a report.
  - 19.1. When the issue is being worked on, Admin shall change the status of that issue to “Work in progress”.
  - 19.2. Once an issue is resolved, Admin shall change the status of issue to “resolved”.
20. Admin shall be able to log in, by using the organization email and password.
22. Admin shall be able to manage the user database.  
e.g. Admin shall be able to remove duplicate or suspicious accounts.

### **Priority 2**

#### Unregistered Users:

4. Users shall be able to see the current status of any reported issue.

#### Registered Users:

14. Users shall be able to rate reports based on their priority (5 is the highest).  
Ex: If an issue needs immediate attention, user can rate it 5.

#### Admins:

23. Admin shall be able to edit/delete reported issues.  
e.g. Admin shall be able to delete duplicate or irrelevant reports.

### **Priority 3**

#### **Unregistered Users:**

6. Users shall be able to see the ratings of a report.

#### **Registered Users:**

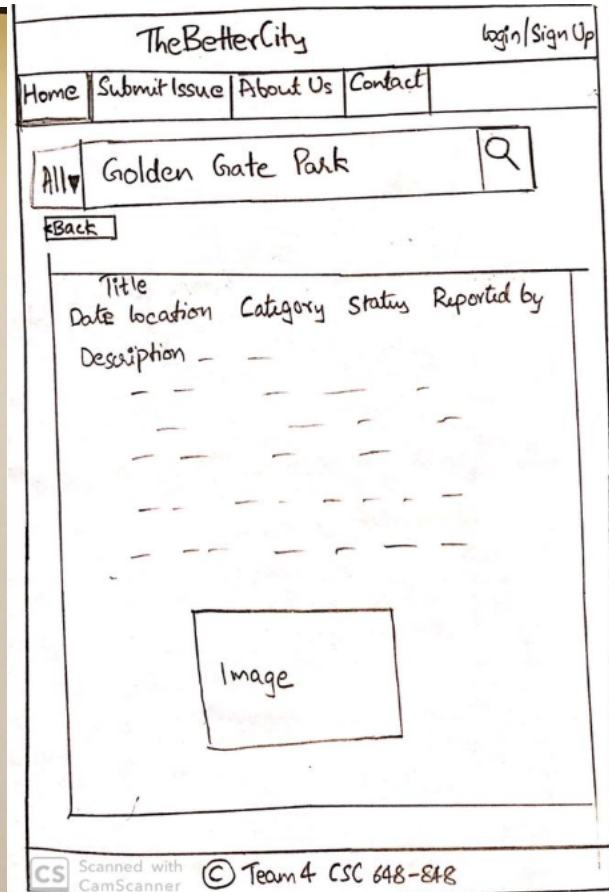
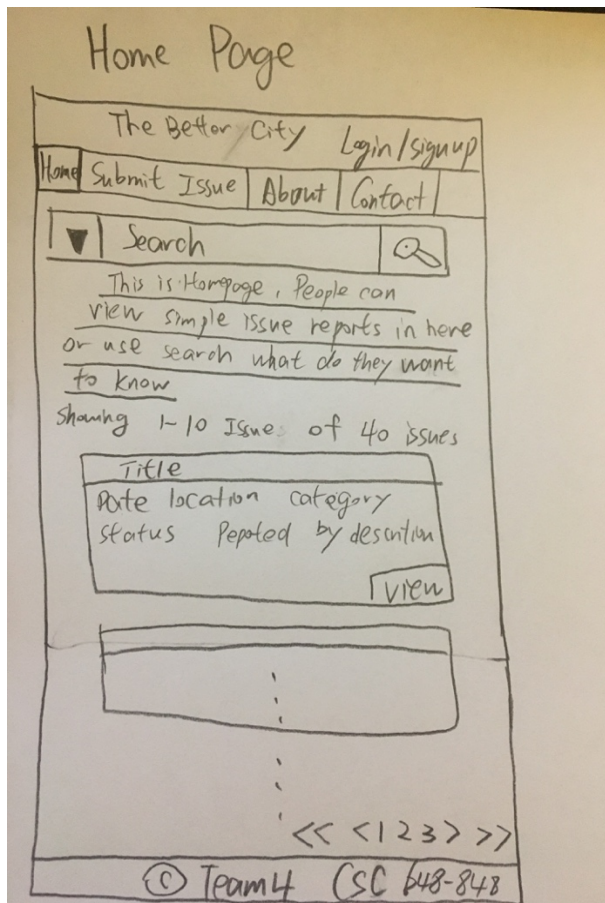
11. Users shall be able to edit the reported issues.
12. Users shall be able to post comments on the reports
15. Users shall be able to geotag.
16. Users shall be able to upload a photo to their report to show more details  
This will be optional, and it will give more details about the issue.
17. Users shall be able to upload a video related to their reports  
This will be optional, and it will give more details about the issue.

#### **Admins:**

21. Admin shall be able to assign tasks to their staff to get the issues resolved

### 3. UI Mockups and Storyboards (high level only)

1. Emily goes to the website and checks for environmental issues.



2. Emily is an unregistered user; she wants to submit an issue to our website. She fills in all the mandatory fields to report issue and clicks on submit. She is prompted to log in or sign up to the website. Since she doesn't have an account, she signs up to submit the already filled up issue.

Submit Issue Page

The Better City login/signup

Home Submit Issue About Contact

This page is submit Issue  
People can submit their issue  
In here

Title of Issue \*

Location \*

Category \*

Images

Description

Submit

© Team 4 CS 648-848

Submit Issue Page

The Better City login/signup

Home Submit Issue About Contact

This page is submit Issue  
People can submit their issue  
In here

Title of Issue \*

Location \*

Category \*

Images

Description

Submit

© Team 4 CS 648-848

Please log in to  
submit the report  
Don't have an account?  
Sign up!

Sign up Pop up

Name \*

Email \*

Password \*

Confirm Password \*

Sign Up


\* mandatory fields

3. Jessica is a registered user. She found an environmental issue at a park she visited. She wants to submit the issue on our site. She logs into our site and fills in all the mandatory fields to submit the issue.

### Home Page

The Better City Login/signup

Home Submit Issue About Contact

Search 

This is Homepage, People can view simple issue reports in here or use search what do they want to know

Showing 1-10 Issues of 40 issues

Title
Date location category status Reported by description
<input type="button" value="view"/>

<< < 1 2 3 > >>

© Team 4 CSC 648-848

### log in Pop Up

Email

Password

[Reset / Forget Password](#)

[New User? Sign Up](#)

### Submit Issue Page


The Better City Login/signup

Home Submit Issue About Contact

This page is submit Issue  
People can submit their issue in here

Title of Issue \*

Location \*



Category \*

▼

Images

Description

© Team 4 CSC 648-848



4. David, a system administrator logs into our website to check if there is any new open issues submitted on our website.

Home Page

The Better City Login/signup

Home Submit Issue About Contact

▼ Search

This is Homepage, People can view simple issue reports in here or use search what do they want to know

Showing 1-10 Issues of 40 issues

Title
Date location category status Reported by description
view

<< < 1 2 3 > >>

© Team 4 CSC 648-848

log in Pop Up

X

Email

Password

log in

Reset / Forget Password

New User? Sign Up

Home Page

The Better City Login/signup

Home Submit Issue About Contact

▼ Search

Status

- ☒ open
- ☐ Work In Progress
- ☐ closed

Category

- ☒ oil Spill
- ☒ Trash Overflow
- ☐ ...

This is Homepage, People can view simple issue reports in here or use search what do they want to know

Showing 1-10 Issues of 40 issues

Title
Date location category status Reported by description
view

<< < 1 2 3 > >>

© Team 4 CSC 648-848

#### 4. High level Architecture, Database Organization

DB Tables:

1. **Entities:** Users

**Attributes:**

- Id (primary key)
- username
- Password
- Email
- issues
- Role

2. **Entities:** Issues

**Attributes:**

- Issue\_Id (primary key)
- tittle
- Photo
- Location
- Description
- TimeStamp
- user\_id (foreign key)
- status\_id (foreign key)
- category\_id (foreign key)

3. **Entities:** Category

**Attributes:**

- Category\_Id (primary key)
- category

4. **Entities:** Comments

**Attributes:**

- id (primary key)
- comment
- TimeStamp
- issue\_id (foreign key)
- user\_id (foreign key)

5. **Entities:** Status

**Attributes:**

- status\_Id (primary key)
- status

**Media storage:**

All images will be stored by saving the file path to the image in the data field labeled “photo”. Images will be stored outside of the Document Root, so that they are safely outside the scope of being directly accessible

**Search/filter architecture and implementation:**

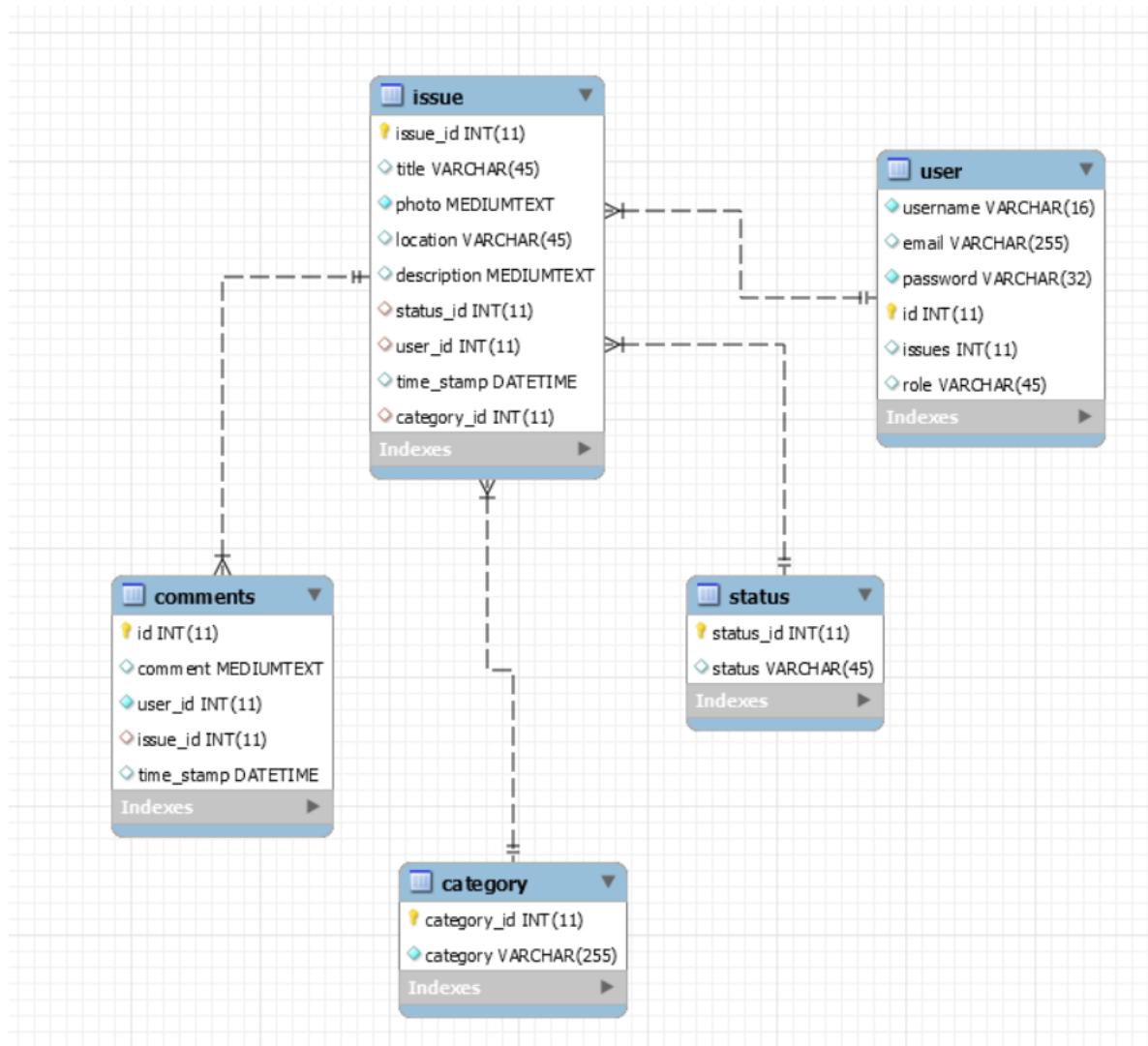
Issues can be filtered based upon status, and category. As for search, it will be implemented using % like.

**APIs:**

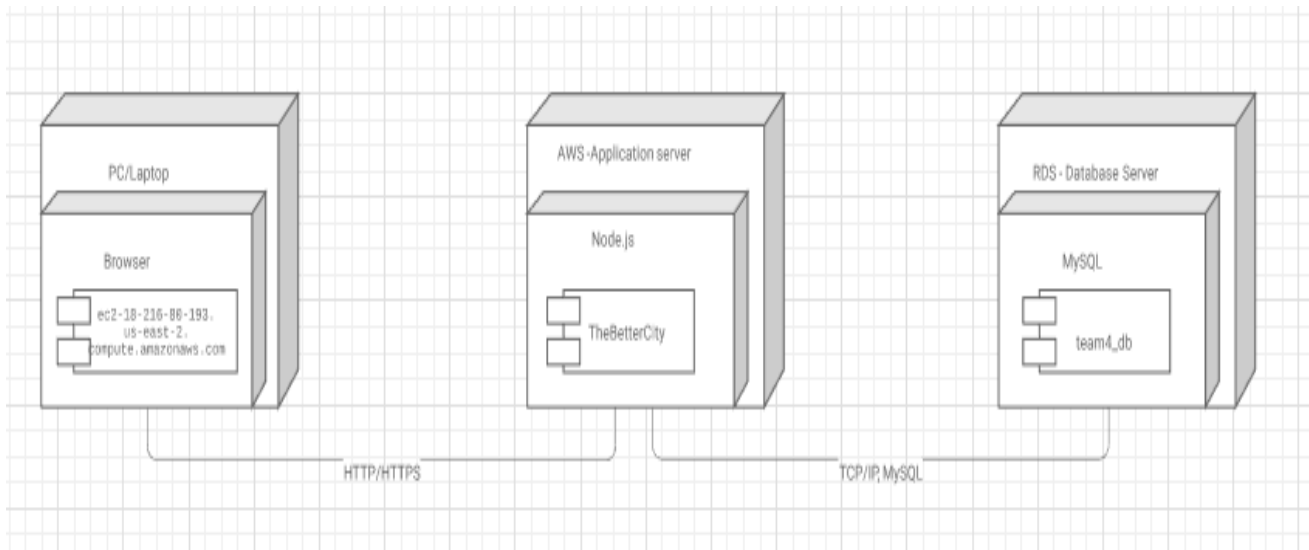
- Google map

## 5. High Level UML Diagrams

### a. High-level UML class diagrams



b. *UML Component and deployment diagrams*



## 6 .Identify *actual* key risks for your project at this time

- Skills risks:

In our team, everybody has different kinds of experiences in different areas. Some of us do not have much experience in teamwork but have more skills in the languages we are using for this project, the servers, and connections. Some others do not have many experiences in the back end or coding for this kind of project, but they are good in templating and designing in the front end.

We are trying to use the shared resources to learn and improve our skills. We are posting different videos, web site, and tutorials links that help us improve some skills and learn more about the area that we are focusing on. We are asking many questions to each other within the group, other group members, and the instructor to address our issues and find a way to resolve it.

- Schedule risks:

We all have different classes with different milestone due dates and other projects to do. Not staying on the schedule and not watching the due dates make it harder for the leader and other team members to be on track and make sure everything will be done on time.

We tried to manage each part differently. The front and back end have separate leaders with separate tasks to do, so they can focus on their own area. The leader will try to reach out to each one of them to follow up with the work that's supposed to be done on the schedule. We set different due dates for each part of the milestone to make sure everything will be done on time.

## 7. Project management

Milestone 2 has two separate sections to submit. So we started with the first section which was about the documentation. First, we all talked about it in the class to make sure everyone is on the same page and have the same understanding of what is this milestone about. This is an important part to do in a team work. If we don't have the same understanding, we will not be able to improve our work, give suggestions for edits and revisions, and have a better version to submit. In order to provide the best ideas and improve our work, we all try to help each other to understand the needs for this milestone and do the best in it. Then we start grabbing some ideas for each section from all the team members. How they are going to look like, how long they are supposed to be, how detailed they can be, and how much time for each of them we need to finish.

We started using Trello to assign tasks from milestone 1. Now we made a new board for milestone 2. The leader made sure to have a "card" for each section and assign them to each team member based on the area of their work. Some parts need all the team members to get involved with and help to get the best result, and some parts only need one or two person from the front or back end to finish. After assigning all the tasks, based on the amount of work and time each task needs, we set a due date for it and every team member commits to finish their tasks by that date.

We never stop talking about the process and how we are doing on each part in Slack. We ask each other questions and make sure we are on the right path of finishing our tasks. Every day, the leader asks for an update on how far we have made it to finish our work and gives us some feedback. We set up a google doc page to add our works in there, so the editor can easily have access to each part that's done, and the leader can keep track of how far we have made it through. When each part is finished, the editor starts reviewing, gives some feedback and asks for the edits to improve it. if there is anything that needs more revision, the leader will do another review and sets a new deadline to turn in the revised version. At the end, we all review each other's works and help in all the parts and all the tasks to make sure we turn in the best result.