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
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: 07/12/2019

History Table: First Version Submitted for Review

1. Data Definitions V2

Users: The 'Users' entity will be used to store the information of all individuals who logged into the site and wish to post an Issue. For registration, each user must input a Name, valid email, and password. However, the user is not required to upload a photograph.
1. Name: The Name input by user during registration will be stored as a string.
2. User_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 users entity will be stored as a string. This email is required for the user to log into the website
5. Photo: The Display photo, if uploaded by the user will be stored as a string. It is an optional field
6. 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.

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. Photo(s): 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. It will be stored as string.
3. Location: The location shows where the issue has happened. It will be stored as string.
4. Status (e.g. Open/Work in progress/Resolved): Shows the status of the issues. This will help the admins to do their job more efficiently. It will be stored in string.
5. Description: User will give a brief summary of the issue's situation. It will be stored in string.
6. User_Id: This is the foreign key that will show who reported the issue that matches the user_id in users table.
7. 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.
8. Category: The user will be able to select 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.

For 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. Body: This stores comments from users in string format. It will show what other users thought about the posts.
3. TimeStamp: 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.

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.
   Issues can be viewed without registering/logging in to the website.
3. Users shall be able to browse through the issues.

3.1 User shall be able to navigate to pages clicking on any of the displayed page numbers.

3.2 User shall be able to navigate to the previous page and next page.

3.3 User shall be able to navigate to first page and last page.
4. Users shall be able to choose any park or environment they want to see the report for.
7. Users shall be able to search reported issues.

Registered Users:
8.+ Unregistered user's functions.
9. Users shall be able to log into the website.
   Username and password are needed to log into the website.
10. Users shall be able to report environmental issues.
    Users shall report issue only when they are logged in.

Admins:
18. Admin shall be able to publish a report.
20. Admin shall be able to log in, by using the organization email and password.
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".

Priority 2

Unregistered Users:
 4. Users shall be able to see the current status of any reported issue.
 13. Users shall be able to filter issues by categories, ratings, and status.
     User can choose issues that belong to a certain category or status or has certain ratings.

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.
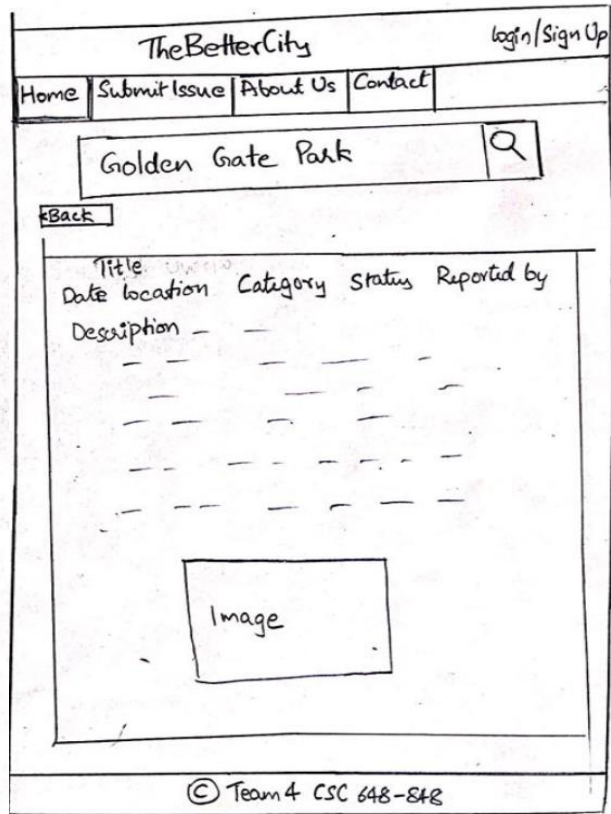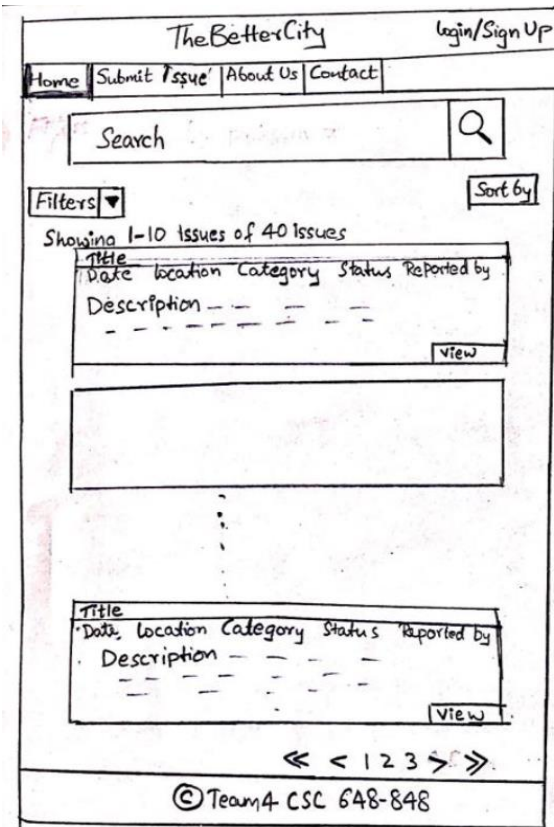
Registered Users:
    11. Users shall be able to edit the reported issues.
    12. Users shall be able to post comments on the reports
    15. Users shall be able to geotag.
    16. Users shall be able to upload a photo to their report to show more details
       This will be optional, and it will give 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
    22. Admin shall be able to manage the user database.
       e.g. Admin shall be able to remove duplicate or suspicious accounts.

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.

TheBetterCity — Login/Sign Up

Home | Submit Issue | About Us | Contact

Title of Issue *
Location ⊙ *
Category ▼
Images | Select
Description *

Submit   *mandatory fields

© Team 4 CSC 648-848

TheBetterCity — Login/Sign Up

Home | Submit Issue | About Us | Contact

Title of Issue *
Location ⊙ *
Category ▼
Images
Descrip

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

Submit   *mandatory fields

© Team 4 CSC 648-848

Sign Up: Pop Up

Name *
Email *
Photo | Select
Password *
Confirm Password *

* mandatory fields

Sign Up

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.

log in PopUp

The BetterCity — login/Sign Up

Home | Submit Issue | About Us | Contact

Search

Filters ▼   Sort by

Showing 1-10 Issues of 40 Issues

Title
Date   location   Category   Status   Reported by
Description — —   — —   — —

View

Title
Date   location   Category   Status   Reported by
Description — —   — —   — —

View

« < 1 2 3 > »

© Team4 CSC 648-848

Email

Password

log in

Reset / Forgot Password

Sign Up

The BetterCity — ⊗ User123 ▼

Home | Submit Issue | About Us | Contact

Title of Issue   *

Location ⊙ *

Category ▼

Images   Select

Description   *

Submit   * mandatory fields

© 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.

<u>4. High level Architecture, Database Organization</u>

DB Tables:
1. **Entities:** Users
   Attributes:
   - User_Id (primary key)
   - Name
   - Password
   - Email
   - Photo
   - Role
2. **Entities:** Issues
   Attributes:
   - Issue_Id(primary key)
   - Photo
   - Location
   - Status
   - Description
   - TimeStamp
   - user_id(foreign key)

3. **Entities:** Category
   Attributes:
   - Category_Id (primary key)
   - Tittle

4. **Entities:** Comments
   Attributes:
   - comment_Id (primary key)
   - body
   - TimeStamp
   - issue_id (foreign key)
   - user_id (foregin key)

Media storage:

All images will be stored by saving the file path to the image in the data field labeled "photo".

Search/filter architecture and implementation:

Issues can be filtered based upon, time of creation, park location, creator of issue. As for search, it will be implemented using % like.

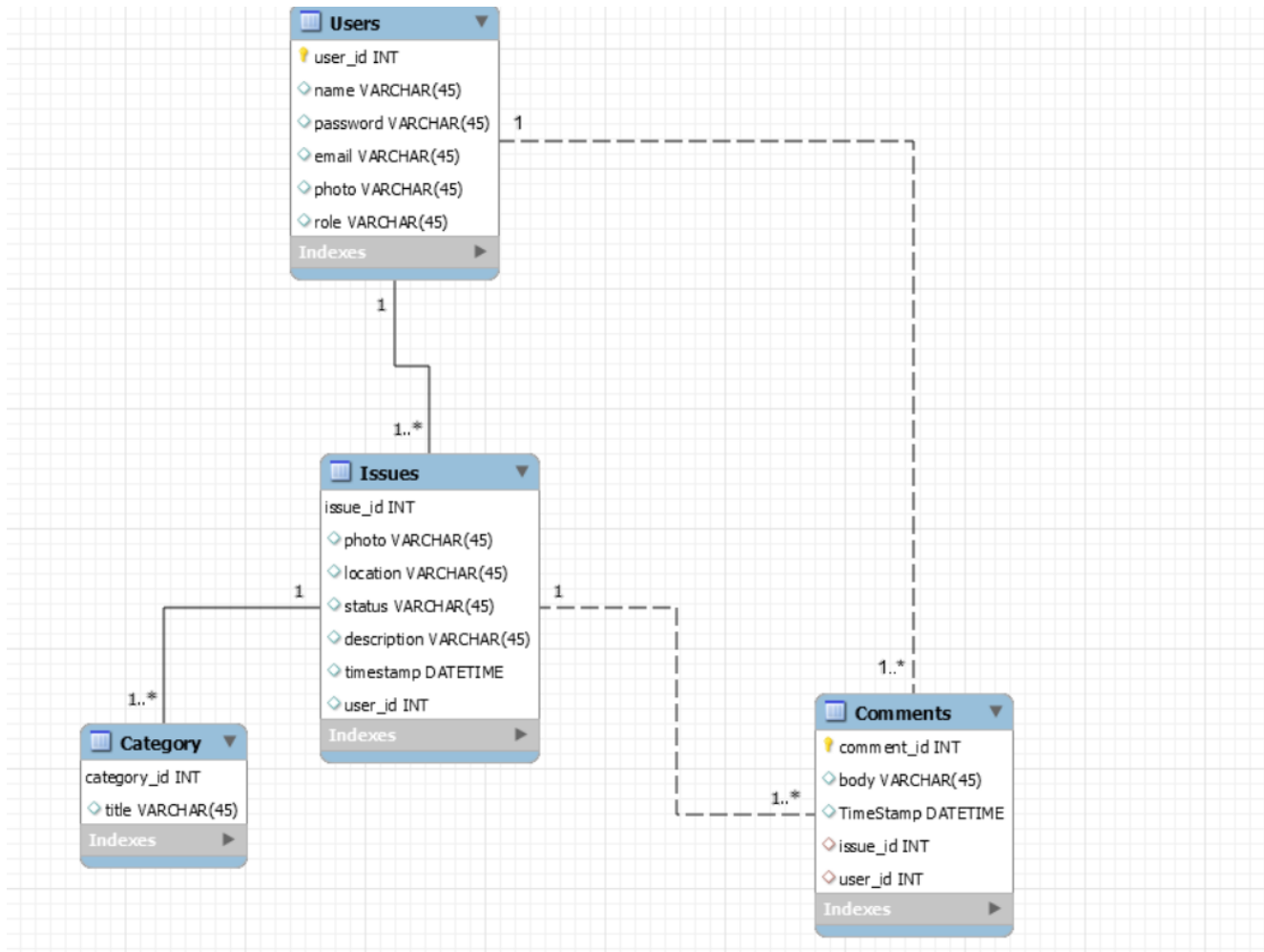**APIs:**
- Google map

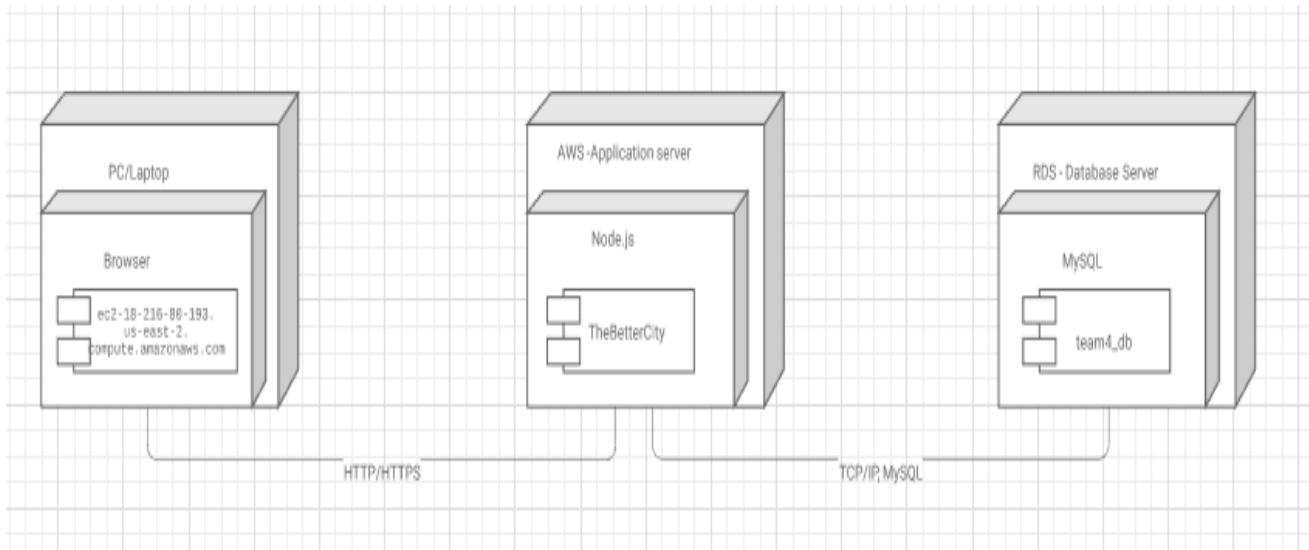**Significant non-trivial algorithm:**
No non-trivial algorithms will be used.

5. High Level UML Diagrams

a. *High-level UML class diagrams*

b. *UML Component and deployment diagrams*

<u>6 .Identify **_actual_** key risks for your project at this time</u>

- 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.

<u>7. Project management</u>

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.