Spill

SW Engineering CSC648/848

Section 01 Spring 2018

Team 06

Milestone 1

2/28/2018

Revision History:

Peter Mutch (peter2mutch@gmail.com), Satjit Bola, Alaric Gonzales, Lorraine Goveas, Albert Fernandez Saucedo, Sandhu Harpreet

Executive Summary

Reporting environmental issues is hard to do, and finding out about them is even harder! The EPA manages big issues, but small neighborhood concerns often slip under their radar. City- and county-level websites are often poorly designed and hard to navigate, and it can be difficult for private citizens to be sure their concerns are being addressed.

Spill aims to change that. Spill is a website where users share information about environmental issues. Users can report a new issue, check up on old ones, or find issues specific to an area, whether close to them or far away. Whether you're a corporate whistleblower, concerned parent, or local official, Spill is the place for your environmental concern.

Spill is made by a team of six students from San Francisco State University studying computer science, as part of Computer Science 648: Software Engineering under Dragutin Petkovic and Anthony Souza. For more information, check out http://sfsuteam06spilldemo.dnsd.info/.

Use Cases

1. Report an issue

- **Who**:
  - Registered User
- **Skill Level**:
  - Basic
- **Process**:
  - Gollum is a neighborhood dad who is concerned about broken glass he saw on the street corner by his favorite park. He opens Spill and tries to **post** a photo he took earlier that day, and finds he must create an **account** to do so. He creates an account and posts the **photo**, marks its **location**, and writes a short **comment** about it. Later, after the city cleans up the corner, he'll visit his post and mark it as **resolved**.

2. Look for new issues in any location

- **Who**:
  - Unregistered / registered User
- **Skill Level**:
  - Basic
- **Process**:
  - Arwen, who is concerned about the environment and wants to know what new challenges it is facing, wants to find out what the most recent **issues** are, no matter where. She opens the website and, instead of issues in a specific location, chooses to see **new** issues that have been reported to Spill, no matter where they are.

3. Check an issue for updates

- **Who**:
    - Unregister/Register User
- **Skill Level**:
    - Basic
- **Process**:
    - Gollum wants to check in on the **report** he made in our first use case. He opens the app, finds the report he made, and checks to see if it has been **updated**. He finds the incident hasn't been **resolved** but is **marked** for cleanup next week. Satisfied, he decides to go back to the park next weekend.

---

4. Resolve issues

- **Who**:
    - City Employee
- **Process**:
    - Galadriel, a municipal employee, opens the website and sees all of the issues that have been reported in the **city** where she works. She leaves a **comment** on one **report** to ask a question about the **issue**. She sees a different **report** about another **issue** that has already been dealt with, and marks it as **resolved**.

---

5. View issues by category

- **Who**:
    - Guest User
- **Process**:
    - Aragorn is writing a research paper about the environmental impacts of pollution, and wants to see all of the pollution incidents that have been reported. Instead of viewing all recent issues or all issues specific to a location, he instead views reports by **category**, and can further refine his view of reports if needed.

---

6. Upvotes existing issue

- **Who**:
  - Registered User
- **Skill**:
  - Basic
- **Process**:
  - Frodo understands that not all **reported issues** are significant, and wants to show which issues are most important to him. He logs into his **account**, finds an **issue** about a chemical spill on his street, and **upvotes** it to show that he wants this to get more attention. This will now show that more users are being affected by that **issue** and **prioritize** it in some views of all issues.

---

7. Removes post from website

- **Who**:
  - Admin
- **Process**:
  - Samwise sees an inappropriate or inaccurate **post** that has made its way onto the website. As an **administrator**, it is his responsibility to keep the website safe for all ages while still serving valid content. Using the **admin module**, he locates the **post**, removes it from the website, and an email is sent to the **user** to alert him/her of the events that just transpired.

Data Definitions

User: Any person who uses the website

Unregistered User: A person who utilizes the website but has not created an account. This user can only access the website as “read-only”. This type of user has the lowest level of privileges.

Registered User: A person who utilizes the website and has created an account. This user is granted can post a new issue, edit issues they have posted, delete their old posts, and do everything an unregistered user can.

Admin: A registered user with special permissions geared toward keeping the website running as intended and removing inappropriate content. Site admins are also responsible for tracking and removing/merging multiple posts about the same issue.

Post: A user report of an issue. Posts can include a location, date, the reporter’s username, description of the issue, and/or images or video.

Status: This refers to the state at which an issue is currently. An issue can be verified, in progress, planned, or resolved.

Verified: In this state, the post has been approved by an as an accurate representation of a real-world issue.

In Progress: In this state, the post has been accepted by an admin and is displayed on the main website.

Planned: In this state, the issue is planned for cleanup or resolution but is not being actively treated.

Resolved: In this state, the issue has been resolved by the appropriate authorities/facilities. The post is changed from in progress to resolved by an admin or the user who originally submitted it.

Locked: In this state, only admins can make changes to the post.

Agency: The establishment, company, or branch Government in charge of resolving or fixing the issue in question

Comment: A text message a registered user adds to a post

User Profile: Username and email. Username is displayed visibly to everyone, email is only visible to admins.

User Panel: See the user’s posts, up-votes, and edit profile

Admin Panel: The resource used by administrators to help mediate content submitted by users on the website

User Record: A record in the database that stores a registered user’s name and e-mail address

Issue Record: A record in the database that stores an issue's location, status, and short description of the issue.

List of Functional Requirements

1. **Users** shall be able to search for **incidents** by location.
2. **Users** shall be able to view **incident** reports.
3. **Registered Users** shall be able to **post** a report of an **incident**.
4. **Registered Users** shall be able to **comment** on **unlocked posts**.
5. **Registered Users** shall be able to mark the **status** of their post as **resolved**.
6. **Registered Users** shall be able to mark the location of an **incident** they are posting.
7. **Registered Users** shall be able to **upvote** a **post**.
8. **Agency** user shall have access to reports they are responsible for resolving.
9. **Agency** user shall be able to **comment** on **issue record** under their authority.
10. **Agency** user shall be able to update the **status** of an **incident** under their authority.
11. **Admin** shall be able to **lock** posts.
12. **Admin** shall be able to **remove** posts.
13. **Admin** shal be able to view user's **email addresses**.

List of Nonfunctional Requirements

1. Spill shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0
2. The website shall be optimized for standard desktop and laptop browser.
3. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed
4. Data shall be stored in Mongo DB
5. Application shall be media rich (at minimum contain images and maps)
6. No more than 50 concurrent users shall be accessing the application at any time
7. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
8. The language used shall be English.
9. Application shall be very easy to use and intuitive.
10. Google analytics shall be added
11. No e-mail clients shall be allowed
12. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated.
13. Best practices shall be used to ensure the website is secure.
14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
15. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project, Spring 2018. For Demonstration Only” at the top of the WWW page.
16. The website shall be protected by SSL encryption
17. The database shall require authentication to do any and all writes

Competitive Analysis

Feature	US EPA	Ireland EPA	Broward.org	Spill (Future Product)
Text Search	+	+	+	**+**
Interactive Map	+	+	+	**++**
Online Complaint Form	+	+	+	**+**
reCaptcha/check	+	+	+	**+**
Image Uploading	-	-	+	**+**
+ : feature exists	**++**: Superior	**-**: feature does exist		

Spill will be competitive in the current market by providing a superior interactive map and user image uploading. Spill's interactive map shall allow for local, regional, or national views as desired, and filter results according to user queries. Spill's user focus will allow users to view issues most relevant to them. Spill's voting mechanic will allow users to prioritize incidents which they believe are relevant to other users, which will enable Agencies to take proper actions accordingly. Spill's image uploading functionality will provide additional information, allowing Agencies and users to have a clearer understanding of the incident.

High-Level System Architecture

1. Spill will be developed using the MERN software stack. Data will be stored in Mongo DB. The backend wil be written in Node.js using the Express.js framework. The front end will be written in React.JS.
2. Spill will be hosted in Google Compute engine and use nginx as the server.
3. Spill will be optimized for Google Chrome and Mozilla Firefox browsers, the most recent version and one version previous.
4. The source code for Spill will be hosted on a GitHub repository and developed independently, collaboratively and simultaneously by all Team 06 developers.
5. Travis CI will be used to ensure continuous integration and help prevent build-breaking bugs being pushed to production.
6. Map integrations will be handled by utilizing the Google Maps API.

Team:

Team Lead: Peter Mutch (peter2mutch@gmail.com)

Backend Lead: Satjit Bolas

Frontend Lead: Alaric Gonzales

Lorraine Goveas, Albert Fernandez Saucedo, Sandhu Harpreet

Checklist:

Team found a time slot to meet outside of the class: **DONE**

Github master chosen: **DONE** (Team Lead: Peter Mutch)

Team decided and agreed together on using the listed SW tools and deployment server: **DONE**

Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on it: **DONE**

Team Lead ensured that all team members read the final M1 and agree/ understand it before submission: **DONE**