Service

Scrumbags Project Proposal

Anthony Ciminello | Nicholas Petty | Joshua Rodriguez-Santiago

Raquel Rosa | Gavin Wolf

Executive Summary

We will be creating a web application called Service, which will allow people to notify property owners of places and things that need to be serviced. The application's goal will be to streamline and crowd-source building maintenance. We feel that a great deal of the spaces we inhabit and pass through in our daily lives are in need of care. With Service, ordinary people will be able to reach out to property managers, and let them know where work is needed. Our users will be the general public, and anyone who claims ownership of or responsibility for a property.

The application will stand out in its simplicity. Anyone can join as a regular user and submit "service requests," which are the primary component of the system. A service request is a short note that includes a picture and the location of something that needs to be fixed. The "property manager" is special user that take responsibility for a building or location. Property managers are required to submit verification documents before they can claim a property, but once verified, they can respond to service requests. Again keeping options simple, the response can either be agreeing to fix the problem, or not. All users will be able to view all service requests and resolutions.

Service connects us through a better environment.

Competitive Analysis

Service is not alone in what it provides, and there are similar products available. However, we can bring a better experience to our users by combining and streamlining the processes on

these other sites and bringing them into a single application. The following table shows how we compare to similar products and their websites.

Product	*Submission form*	*User accounts*	*Images*	*Categories*
Service	Yes	User, property manager	Yes	No
	Website under construction			
City of Hialeah	Yes	No	No	Yes
	http://www.hialeahfl.gov/index.php?option=com_content&view=article&id=107&Itemid=113&lang=en			
FixMyStreet	Yes	Yes	Yes	Yes
	https://www.fixmystreet.com			
SeeClickFix	Yes	User, property manager	Yes	No
	http://en.seeclickfix.com			
FAU safety hazard reporting form	Yes	No	No	No
	http://www.fau.edu/facilities/ehs/safety/Hazard-Report-Form.php			
Penn State facilities reporting	No, email	No	No	No
	http://www.met.psu.edu/browse-by-audience/faculty-staff/report-a-facilities-issue			

Looking at the city and university problem reporting sites above, it's clear that a centralized, well-organized reporting system is a major opportunity for Service. FixMyStreet and SeeClickFix are the most direct competition for Service, offering practically identical reporting systems. In this case, we will leverage a simpler user experience to make our application more appealing.

Overview and Scenarios

Service will be used by the general public and property managers. Normal users create accounts, create repair requests, and track their requests. Property managers create accounts and respond to repair requests for their properties. Additionally, an administrator will moderate requests and verify property managers.

The first scenario any user will encounter is account creation. The application's home page will have options to create an account or log in. An account is required to use the system, so we must keep the creation as simple as possible. The normal user will provide an email address, a user name, and a password. To become a property manager, the same information is required, along with the name or location of a property and document that verifies ownership of or responsibility for that property. One the user is verified, either by email confirmation or document review, they are able to log in.

The second major action a Service user will perform is log in. Like any log in system, the user will enter their user name or email address along with their password. Successful entry will direct them to their personal homepage.

Service's primary use is creating requests, which is what most users will spend the most time doing. To create a service request, the user clicks the "Create Service Request" button, and fills in a short form. The form will ask for a property or location, a picture, and a description of the problem. Once this information is provided, the user submits the form and the request is created.

Both regular users and property managers can view service requests. In this way, it's less likely that repeated reports of the same problem are created. For property managers, two options are available; they can respond to the request and provide an estimated delivery date, or reject the request. After completing their work, the property manager closes the service request. The state of service request can be open, in progress, rejected, or closed, and this is status is displayed for all users.

The site administrator will have the ability to remove inappropriate or unnecessary service requests and revoke user accounts for system abuse. To do this, their view of the service request list will include an option to delete the request, and they will have access to a list of all site users, with the ability to delete accounts. Administrators are also responsible for verifying property managers' ownership documents, which are viewed and then approved or rejected.

The last major task any user will complete is logging out of Service. This simply requires clicking the logout button, which closes their session and brings them back to the website's home page.

Functional Requirements

1. Website
    a. Login page
    b. Account creation page
        i. Ownership documentation form
    c. Service request list view
        i. Create service request
        ii. Respond to service request
        iii. Logout

    d. Administration
        i. Delete request button
        ii. Account list with delete option
2. User database
    a. Email address
    b. User name
    c. Password
    d. Property manager status
        i. Properties
    e. Administrator status
3. Service request database
    a. Location
    b. Description
    c. Picture
    d. Status

Non-functional Requirements

1. Mobile responsive website provided by WebRatio templates. No action takes longer than 5 seconds. All actions must look and perform the same regardless of access method.
2. 100% uptime. Availability for at least 1,000 concurrent users provided by WebRatio hosting.
3. Security and session management provided by WebRatio development and deployment system.
4. Database for at least 10,000 users and 10,000 repair requests provided by WebRatio hosting and university license.

System Development Infrastructure

1. WebRatio: integrated development environment, web hosting, page templates, database, training materials.
2. Cameo Enterprise Architecture: use case and business process modelling.
3. GitHub: source code version control and code sharing.
4. SourceTree: Git repository management interface.
5. Trello: task management system.
6. Circuit: team collaboration communication system.
7. MS Word: documentation creation.
8. MS Excel: test set management.
9. Chrome, Firefox, Safari: web browsers for web site viewing and usage.

Team

- Product Owner: Nick Petty
- Scrum Master: Anthony Ciminello
- Back-end Developer: Joshua Rodriguez-Santiago
- Front-end Developer: Raquel Rosa
- UI/UX Designer: Gavin Wolf

Deliverables

1. Proposal: October 24
2. BPMN Modeling
3. Test Set Document
4. Final Presentation