

Project Design

K.S.D.A - Neighborhood Connect

By: Kaite Pelton, Shannon Hagmaier, Dan Pilsbury, and Ashton Robinson

Overview - A

Currently, there is no web-based solution that allows communication between citizens and public officials in Boston over issues that arise in their respective communities. The existing system allows for requests from citizens to be made, but there is one key issue that it does not address—the status of each request is not known after it has been submitted. This is a problem for not only the person who submitted the request but also for the community at large who may be impacted by the issue at hand. In order to solve this problem, we will use public communication channels to create online communities for viewing, interacting, and creating requests that will be hosted on our site, Neighborhood Connect. Neighborhood Connect is designed to allow the people of Boston to voice requests for any issues that they encounter on a day-to-day basis, track the status and associated commentary from the community about their requests, and view other issues that other members of the public have brought to light.

We want to create a centralized platform in order to make it easier for all citizens to have their voices heard quickly and publicly, so that not only the intended public official is notified, the community is as well. To solve this issue, our app will have a public forum for people to post requests, as well as display all requests that have been made. We will have the option for users to view requests by location as well as by unique tags decided by the creator. All users will also be able to vote and comment on current issues to show support or opposition. In order to decrease the time it takes for issues to get resolved, city officials will be granted admin access to communicate through these forums. They will be able to resolve issues as well as query non-admin users to find out more information about certain requests.

Conceptual Design - D

Concepts:

1. **Name:** Account

Purpose: store a user's information

State:

Privilege: User -> Member or Official
Name, password: User -> one String
Auth: one username -> one password
Requests: User -> set Requests
Comments: User -> set Comment
Votes: User -> set Vote

Actions:

create(u: Username, p: password)
delete(u, auth)
edit(u, auth, u')
getRequests(u)
getComments(u)

getVotes(u)

Operational Principle:

After create(u: User) => u in app.Users

2. **Name:** Member

Purpose: user that is member of the community and not a city official

State:

Account: User -> one Account

Actions:

create(u: username, p: password)

New Member()

Operational Principle:

After create() => User.privilege = Member

3. **Name:** Official

Purpose: display distinction as a city official with certain privileges

State:

Account: User -> one Account

Actions:

create(u, p, auth)

If auth verified by city:

New Official()

Operational Principle:

After create() => User.privilege = Official

4. **Name:** Request

Purpose: publicly submit a service request

State:

Title: Request -> one String

Description: Request -> one String

Author: Request -> one User

Location: Request -> one Latitude, one Longitude

Tags: Request -> set Tag

Type: Request -> one Issue or Proposal

Votes: Request -> set Upvote, set Downvote

Priority: Request -> set Priority Rank

Comments: Request -> set Comment

Actions:

create(t: title, d: description, auth, location*, tags, type)

New Request()

edit(r: Request, r')

delete(r: Request)

getByTag(t: Tag)

Return t.items

Operational Principle:

create() and not delete() => request in feed;

getByTag(t) returns requests where t in request.tags

5. **Name:** Comment

Purpose: allow discussion of a request

State:

Reply: Comment -> one Item
Content: Comment -> one String
Author: Comment -> one User

Actions:

create(i: Item, c: content, auth)
 New Comment()
 Comment in i.comments
edit(c: Comment, c')
delete(c: Comment)

Operational Principle:

After create() and not delete() =>
{ c in i.comments }

6. **Name:** Tag

Purpose: categorize an item

State:

Name: Tag -> one String
Items: Tag -> set Items

Actions:

create(n: Name)
 New Tag(n)
assign(t: Tag, i: Item)
 i in t.items
remove(t: Tag, i: Item)
 i not in t.items
delete(t: Tag)

Operational Principle:

After create(n) and assign(t, i) and not remove(t, i)
{ i in t.items }

7. **Name:** Upvote/Downvote

Purpose: track popularity of proposal

State:

Votes: Item -> set User
Count: Item -> one Nat

Actions:

upvote(i: Item, u: User)
 i.votes += u
downvote(i: Item, u: User)
 i.votes -= u

Operational Principle:

upvote()... unvote()
{ i.count is #upvote() - #downvote() }

8. **Name:** Priority Rank

Purpose: allow users to rank the priority of different requests

State:

set Rankings: {Low, Medium, High}
Rank: Item -> one Rankings

Actions:

assign_rank(p: Item, r: Rankings)
change_rank(p: Item, r: Rankings)
get_rank(p: Item)
remove(p: Item)

Operational Principle:

assign_rank(p: Item, r: Rankings); get_rank(p: Item) => r
change_rank(p: Item, r': Rankings); get_rank(p: Item) => r'
remove(p: Item)

9. **Name:** Session

Purpose: auth for extended interaction

State:

Name, password: User -> one String
Sessions: Client -> set User

Actions:

register(n: String, p: String)
login(n: String, p: String, c: Client)
logout(c: Client)
auth(c: Client)

Operational Principle:

register(n, p, u); login(n, p, c)
auth(c, u') => u' = u
logout(c); auth(c', u) => c' != c

10. **Name:** Map

Purpose: visualize the area requests are made and resolved

State:

Requests: location -> one Request

Actions:

displayRequests(i: Item)
For req in getRequests:
map.add(req.location)

Operational Principle:

After request is created with location =>
{request in map.requests}

11. **Name:** Resolved / Unresolved

Purpose: allow users to give feedback on whether an item is resolved

State:

Status: Item -> one Resolved or Unresolved

Author: Item -> one User

Resolved, unresolved: Item -> one Nat

Actions:

markResolved(i: Item)

l.resolved += 1

markUnresolved(i: item)

l.unresolved += 1

Operational Principle:

markResolved(), markUnresolved()

{i.resolved is number of markResolved}

{i.unresolved is number of markUnresolved}

12. **Name:** Following

Purpose: allow users to view specific tags

State:

Followers, following: User -> set Tags

Actions:

follow(u: User, t: Tag)

u.following.add(t)

t.followers.add(u)

unfollow(u: User, t: Tag)

u.following.remove(t)

t.followers.remove(u)

Operational Principle:

After follow(u, t) and not unfollow(u, t)

{t in u.following and u in t.followers}

Sketches - K

Here is a link to our [sketches](#). Please feel free to comment directly on the slides. We will also provide the sketches below for your convenience, although the slides include additional labeling and transitions.

Neighborhood Connect

Report

Sign In

Polar Express Hack !

MITHACK

Polar express hack all over MIT's Campus!

Pothole?

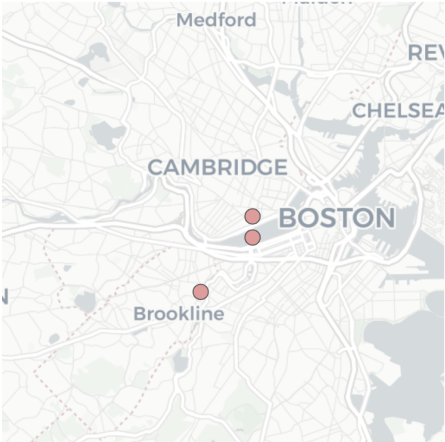
HARVARD

Pothole through Harvard Bridge !

Road Closures

BROOKLINE

...



Neighborhood Connect

Report

Sign In

Should we build another dome?

MIT

100 downvotes

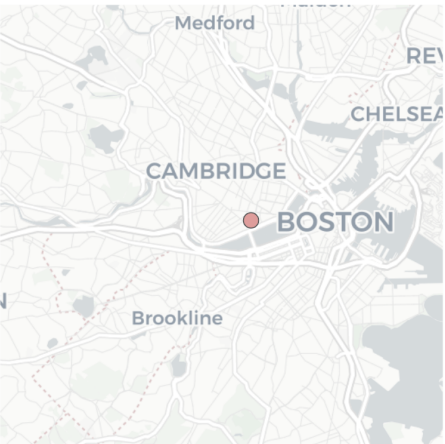
4,261 upvotes

Just think about it folks ... another dome ?

Downvote

Upvote

Comments:



Neighborhood Connect

Report

Sign In

Polar Express Hack !

MITHACK

Resolved

Unresolved

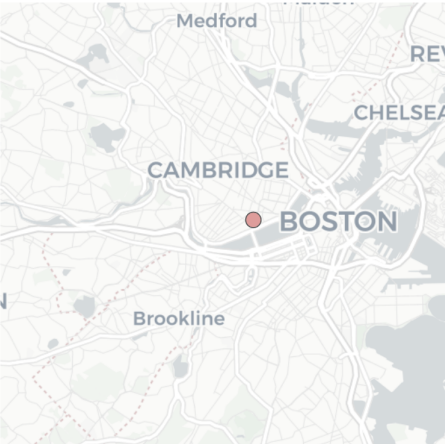
Polar express hack all over MIT's Campus!

Comments:

Oh my gosh was that a hack! #hacking culture

It will be taken down shortly.

Urgent: streamers outside lobby 10 must be taken down.



Sign In

Username

Password

Create Request

New to Neighborhood Connect? [Create an account](#)

New Request

Title

Resolved

Unresolved

Content

Tags

Create Request

Settings

Change Username

Change Password

Delete Account

Heuristics - S

1. Visibility of Status System

- The app always has the status of if a user is logged in or not in the upper right hand corner of the app, it will say sign in if not signed in or Hi "name" if a user is signed in

2. Match between system and the real world

- The app uses terms that are already used in these types of actions, for example creating requests, and the tags also are named with familiar issues to help users determine the type of request that they are submitting.

3. User control and freedom

- A user is allowed to edit their requests and delete a request if they need to. They are also able to change their username, password, or even delete their account if they no longer wish to be on the platform.

4. Consistency and standards

- We keep terms the same throughout, like tags and following. We also use conventional terms, like tags, that are already used on current platforms.

5. Error prevention

- Many of the actions that a user is given are only the ones that they are available to do, and there are further checks to make sure they have the correct authentication. For instance, when a user is not signed in they cannot edit any requests or access the following page.

6. Recognition rather than recall

- A user is able to look at all of the requests that are made at any time and go in to look at the tags that they follow. They also always have the option to create a request

7. Flexibility and efficiency of use

- I think we could add a shortcut to make it easier to create requests that are common, like there might be a lot of pothole requests so there could be a pothole template that could be chosen when creating a request

8. Aesthetic and minimalist design

- We only put necessary buttons on the interface and actions that are not available to the user, if they are not signed in for instance, are hidden on the ui.

9. Help users recognize, diagnose, and recover from errors

- When signing in or creating an account, if the password is incorrect or the username is already taken, a message is clearly shown to the user what the issue is so that they can correct it.

10. Help and documentation

- We have added a landing page so when a user first gets to the site it displays what the purpose of the app is and also some of the key concepts that are a part of it.

Design Commentary - S

Allowing Downvoting

We decided to allow users to upvote and downvote requests that are submitted. This allows users to easily share their opinions on ideas posted to share if they support this idea or not. The concept of upvoting and downvoting for Neighborhood Connect is different than for other social media sites since this gives the users the ability to give their input on community issues that may affect them and it is not directed towards the user who posted the request. We also made sure that users will only be able to upvote or downvote any request a maximum of once so there will be no skewing of the data.

Alternatives Considered: We thought about having only an upvote option but decided that would be restrictive towards users since if a user disagreed with a request there would be no easy way to communicate that. With the downvote option users have the option of showing their true opinion about it and also being indifferent by just not upvoting or downvoting.

Flexibility of Tags

A key concept for Neighborhood Connect is that requests can have a variety of tags associated with them to be able to easily filter or find similar issues. We decided to distinguish tags to be or of 2 types: Location and Descriptive. Since every request can only have a single location it is important that there not be conflicting location tags so we chose to have a location hierarchy system. There are different levels of specificity of locations so for instance a request can have location tags of Boston and 02119 but not Cambridge Boston 02119. Descriptive tags have the flexibility to be anything and requests can have as many descriptive tags as necessary. This allows any users to quickly see where an issue is taking place and what it is about without having to look at the entire post.

Alternatives Considered: We thought about having users be required to input the exact location of every request made, by inputting the lat/long, but we decided that this may deter people from actually posting to the site. We also thought it would be good to see requests at the different scales of cities, zip codes, and also exact location. We also thought about only having a set standard of descriptive tags for users to choose from but we thought it would be better to allow for more of a variety of tags that can fit a particular area's needs.

Mapping Requests

Mapping requests is one area we decided whose scope can be decreased if we run out of time. We want there to be a map displayed to users to showcase places where a user has made an issue request or proposal and areas where requests have been resolved. This can help the city track and see if any area has a disproportionate amount being made or areas that have sparse resolved requests. One drawback for the map would be it would only have the subset of points where a user has entered the specific location of a request being made. We decided on having this since we didn't want to require users to enter

a specific location if they didn't feel comfortable doing so. If we have time we will also make this a static map where users can filter based on tags.

Alternatives Considered: We thought about not having a map altogether since there would still be visibility of issues through the main feed. However, we decided it would be a benefit to both generic users and city officials to be able to visually see the requests.

Ethical Protocol Analysis - A

Imagined Scenario No. 1:

The city of boston releases a new application that allows community members to submit requests about relevant issues that require immediate assistance from public officials in hosted forums. The app is downloaded by more than 75 percent of the community, requests have an average response time of less than two weeks, and the community as well as the officials are happy with their new platform. Malicious users are continuing to be reported and expelled from the platform by the strong community that has been established. Admins are happy with the platform because they are able to focus their efforts into resolving requests that are most relevant to the community, and as a result, users are very satisfied with their experiences.

Imagined Scenario No. 2:

The city of boston releases a new application that allows community members to submit requests about relevant issues that require immediate assistance from public officials in hosted forums. The app is only downloaded by 25 percent of the community and there is a lack of engagement with the app. There are lots of overlapping requests, very few people are interacting with each request. Due to this lack of interaction, government officials that are assigned to respond to the requests of the users rarely follow through on requests in a timely manner. The site turns into a platform to slander government officials about the terrible jobs they are doing, and soon enough there are no more users interacting with our platform.

Stakeholders

The stakeholders of our site include but are not limited to: all citizens of Boston including temporary and permanent residents, public officials, construction crews, restaurants and bars, small and large businesses in the community, schools and universities.

Moral Values

Outcome Lense:

In what ways does what you're making turn out better or worse for your stakeholders?

Process Lense:

How did the process treat shareholders?

Structure Lense:

How are the outcomes distributed among different stakeholders? What are the differences in how different stakeholders were treated by the process?

Interaction Lense:

Are shareholders who are interacting with our platform being positively impacted because of their

contribution?

Value-Laden Design Decisions

Should we allow downvoting?

Upvoting Only

Values promoted: Outcome Lens--users can upvote posts most relevant to them so that requests are handled in a timely manner; Interaction Lens--users who upvote requests help lift relevant posts to the attention of the public officials attending to the matter.

Values demoted: Process lens--users and officials will not be able to differentiate malicious posts without the ability to express discontent with a post; Structure lens--bad actors aren't dealt with as easily.

Upvoting and Downvoting

Values promoted: Outcome Lens--users can upvote relevant posts and downvote malicious posts; Interaction Lens--users who upvote requests help lift relevant posts to the attention of the public officials attending to the matter as well as downvote to bring attention to malicious users of the site; Process lens--users and officials will be able to differentiate more easily between malicious posts

Values demoted: Structure lens--bad actors can now downvote others posts that may be relevant issues.

We have chosen to pick the second option--upvoting and downvoting. We feel that the need to express negative sentiment to requests places more emphasis on a strong community lead effort to deal with bad actors on our site, which is one of our main reasons to design this platform.

Should users be able to submit their requests anonymously and without geolocation?

Posting Anonymously without required Geolocation

Values Promoted: Structure Lens--all users are treated fairly and have the freedom to post anonymously or not; Process Lens--people are not pressured to share their names and locations to the public; Interaction Lens--others are still able to like and comment on various posts

Values Demoted: Outcome lens--officials will not be able to see where requests come from and have a more difficult time finding them

Posting Anonymously with required Geolocation

Values Promoted: Outcome lens--officials are able to see where requests come from and solve them without sacrificing anonymity of the user; Process Lens--people are not pressured to share their names to the public, simply the location of the corresponding issue; Interaction Lens--others are still able to like and comment on various posts

Values Demoted: Structure Lens--users who are posting about issues near their residences may still feel like they are exposing their personal information

Named Users Posting with Geolocation

Values Promoted: Outcome lens--officials are able to see where requests come from; Interaction Lens--others are still able to like and comment on named posts.

Values Demoted: Structure Lens--certain users do not want to sacrifice anonymity; Process Lens--people are pressured to share their names to the public;

We have decided to go with option one--posting anonymously without required geolocation. We feel that protecting the anonymity of the user is one of the most fundamental concepts when posting to online forums. Even though there may be an issue with the resolution of certain cases, we hope that admins will be able to handle these ambiguities through the seamless communication channels that we have built.

How do we establish priority of requests that need attention so that public officials know what to attend to

first?

Use Upvoting and Downvoting only

Values Promoted: Process Lens—users have a very simple and understandable way of interacting with requests; Interaction Lens—users are able to interact with posts they feel relevant;

Values Demoted: Structure Lens—users who have issues of high priority can go unnoticed to officials; Outcome lens—the most necessary requests may not be achieved in a timely fashion

Use a priority ranking along with Upvoting and Downvoting

Values Promoted: Interaction Lens—users are able to interact with posts they feel relevant in two different ways; Structure Lens—users with high priority requests can get the attention they deserve; Outcome Lens—officials will be able to clearly enumerate what issues are of higher priority than others

Values Demoted: Process Lens—users now have two ways to show their feelings about a post which may confuse them

For the MVP, We have decided to go with option two—use a priority ranking along with upvoting and downvoting. We feel that it is necessary to rank requests not only by community driven liking and disliking, but also by priority to give more information to public officials who are handling the requests. Even though this presents some ambiguities to the user at first, we hope that they will learn to use both of these features to maximize their engagement with the platform.

For the final, we have decided to go with option one as it simplifies the users lives to only have one way to show support or disinterest to a post.

How should we allow users to view all of the requests that are currently stored in our system?

Use a feed so that users can interact with all of the requests(ie commenting, etc.)

Values Promoted: Interaction Lens—users can interact with issues in their community; Process Lens—any user can have a voice in the community.

Values Demoted: Outcome Lens—this may place higher emphasis on the content of issues, but may neglect certain areas of the community since there is no map; Structure Lens—this may allow certain areas of the community that are more active to have a louder voice since there is no way to monitor that all areas of the community are heard on a map.

Use a feed as well as an interactive map to view all of the requests near any user

Values Promoted: Interaction Lens—users can interact with both the map and the feed to understand the relation between issues and location; Outcome Lens—users can make an impact on areas of the community they want improved; Process Lens—users can have a voice in any area of the community; Structure Lens—if an area of the community is not being responded to, it will be apparent on the map

Values Demoted: None

Use only a Map for Users to view requests on

Values Promoted: Interaction Lens—users can interact with issues with relation to their area in the community; Process Lens—users can have a voice in any area of the community; Structure Lens—if an area of the community is not being responded to, it will be apparent on the map

Values Demoted: Outcome Lens—this may place higher emphasis on the location of issues rather than the content of issues.

Addressing Suggested Changes:

We added a deployment link and DB-instructions. We included the option to create a request without signing in to encourage users to post requests anonymously. We apologize for the sign in issues, we have fixed this bug and thoroughly tested sign in and create user. We choose to include creating a request that doesn't have a location to encourage users to share all types of requests without worrying about sharing their location. We acknowledge the user-specified latitude and longitude are not user-friendly, we will incorporate a drag and drop map functionality when creating a request in the final product. We have addressed request specifications on the frontend and will be adding request middleware on the backend for the final product. We addressed all other comments within the code.