

SECE Geoloc v2.0

Advised By:

Prof. Henning Schulzerrine

&

Jan Janak

Riddhi N. Mehta

UNI : rnm2119

MS Computer Science

Columbia University

SECE Geoloc v2.0

Introduction

Components

1. Dashboard
2. Floor Plan Editor
3. Polygon Editor
 - 3.1 Floor Plan Polygons
 - 3.2 Google Maps Polygons
4. Smart Object Locator
5. Groups

Future Work

References

Introduction

Accurate geographic location of Smart Objects (down to centimeter accuracy) as well as the possibility to determine geospatial boundaries of real-world objects (buildings, floors, rooms, furniture) is important for intuitive naming of Smart Objects. Having the possibility to determine accurate location of Smart Objects both indoors and outdoors is important because it makes it easy to deploy such objects. Since Smart Objects are often attached to fixed physical devices being monitored or controlled, it is often sufficient to determine its exact location once--when it is being installed or configured. Having a database with geospatial data for real-world objects would then make it possible to refer to Smart Objects with names relative to those real-world objects, which is how humans tend to refer to objects in the real world ("the lamp in the living room", "the light switch in the garage"). Without this kind of information, we could only refer to Smart Objects by whatever unique identifiers they might have (a hostname, IP address, MAC address). If we have a database with accurate location information for Smart Objects, we can use that database to discover smart objects using geospatial queries (give me all Smart Objects near my current location). If we also have a database with geospatial boundaries for physical objects, we can give those Smart Objects relative names, using the physical objects as reference points ("lamp in living room", "all thermometers on 7th floor of CEPSR", "the light switch in IRT lab").

The goal of this work is to create a web-based User Interface (UI) that could be used to determine precise location for Smart Objects, as well as to build a database with geospatial boundaries for real-world objects. Specifically the UI should be easy and intuitive enough so that it can be used directly by users of the SECE system, without having to acquire specific knowledge about geospatial systems first.

The UI consists of the following components:

- Dashboard
- Floor Plan Editor
- Polygon Editor
- Smart Object Locator

The dashboard component is the first component seen by SECE users when they select the link to the geolocation UI. The dashboard provides an overview of all the geospatial objects, floor plans, and Smart Objects the user has created in the system so far. The Floorplan editor component allows the user to upload, and orient in the world floor plan images. The Polygon Editor components allows the user to draw polygon overlays over floor plans or Google Maps and save them in a database as geospatial boundaries for objects. The Smart Object Locator adds the possibility to pinpoint a precise location for Smart Objects using already created floor plans or Google Maps.

Components

This section describes the components of the new SECE geoloc UI which can support the various objects which can be added into the SECE system.

The technologies used to implement these complements include HTML(5), Javascript (including polygon drawing plugin), CSS, JQuery, Twitter bootstrap(styling library as well as JQuery plugins), MySQL, PHP(including ba-simple-php-proxy plugin).

The mysql database is used to store all the object entities as well smart objects into the system. When a new object is added into the system it is first stored in the obj_ids table of the mysql database by means of storing a unique id of the object. no two objects no matter of what type do not have the same id and therefore any type of object can be a parent of any other type object.

1. Dashboard

Dashboard is the first component of the Geolocation UI that the user sees

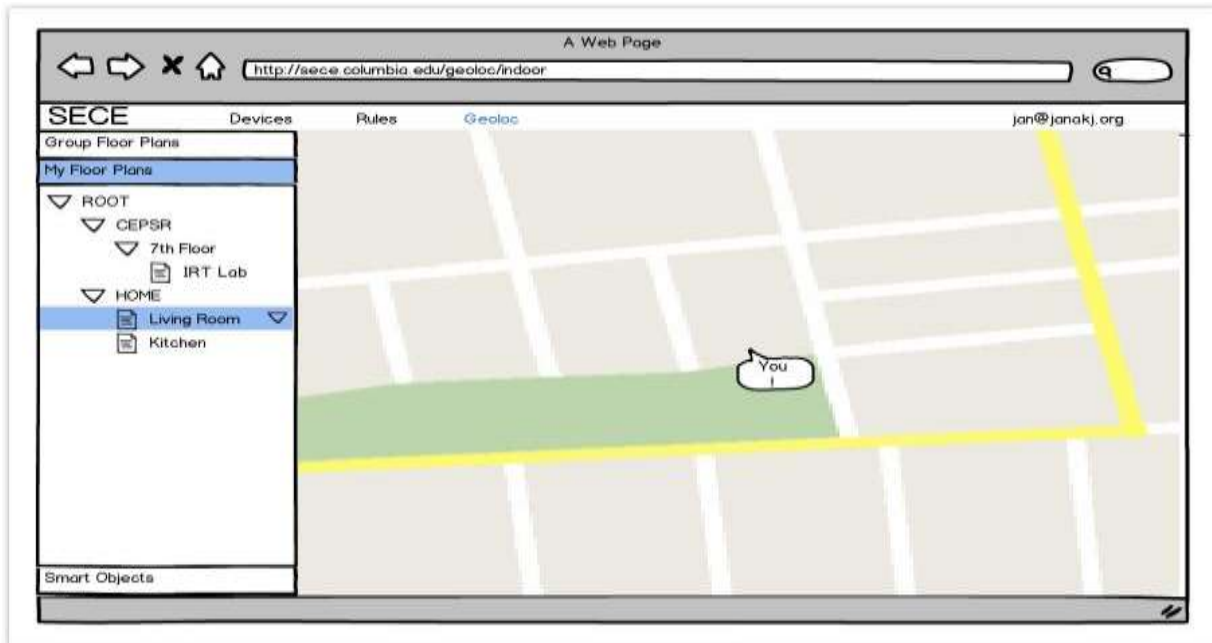


Fig 1.1. Dashboard initial proposal mockup with polygon, group and floorplans tree

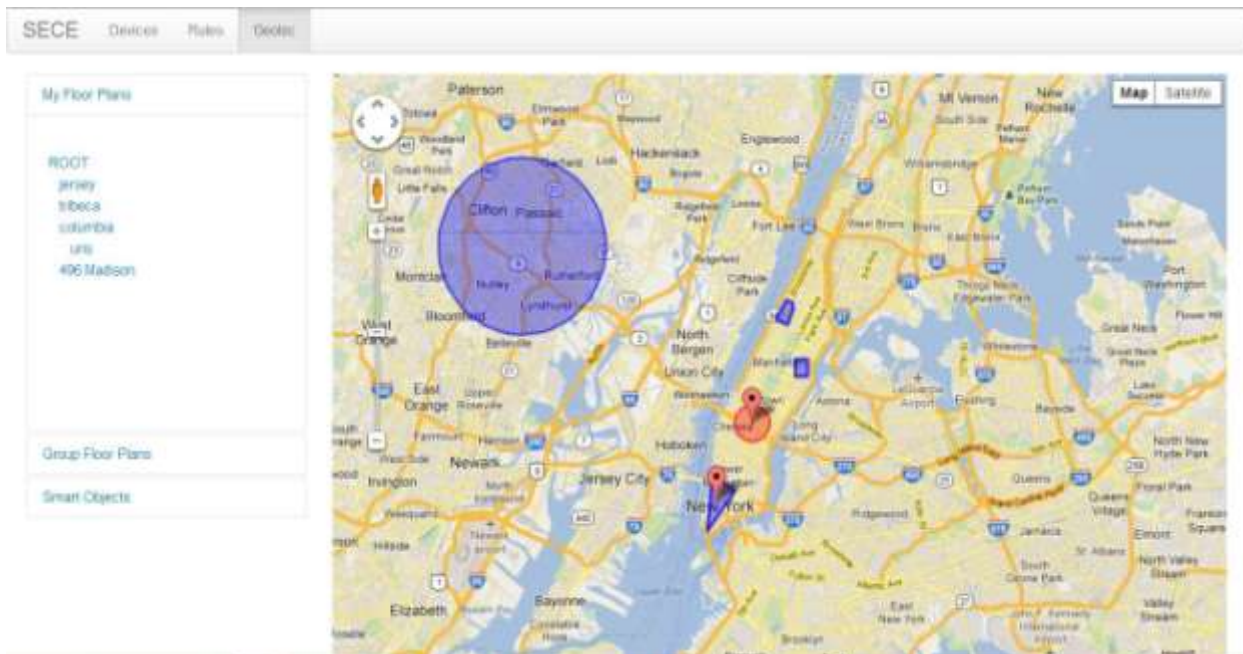


Fig 2.1 Dashboard in implementation with polygon, group and floorplans tree

The main navigation entity is the tree of all the objects. The tree always has the root. The root exists even if there are no floor plans, groups, or polygons. The user can add a new child element of a tree item by pulling up the context menu of the item.

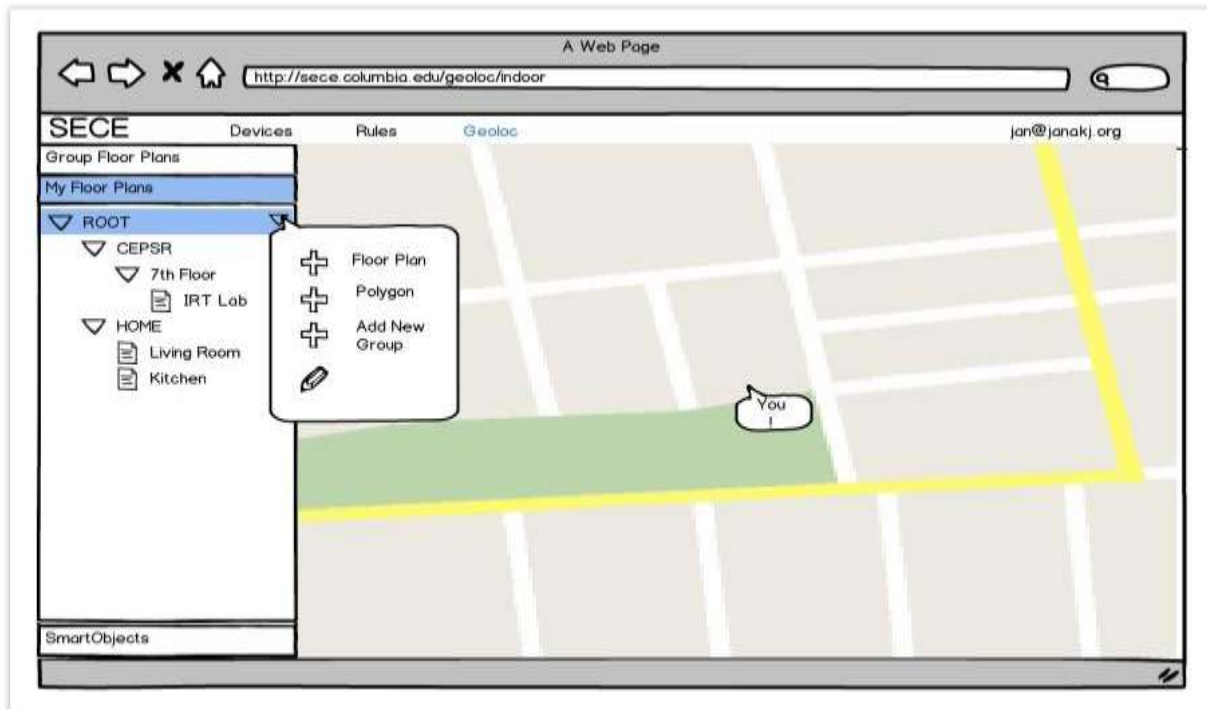


Fig 1.2. Dashboard initial mockup with context menu view

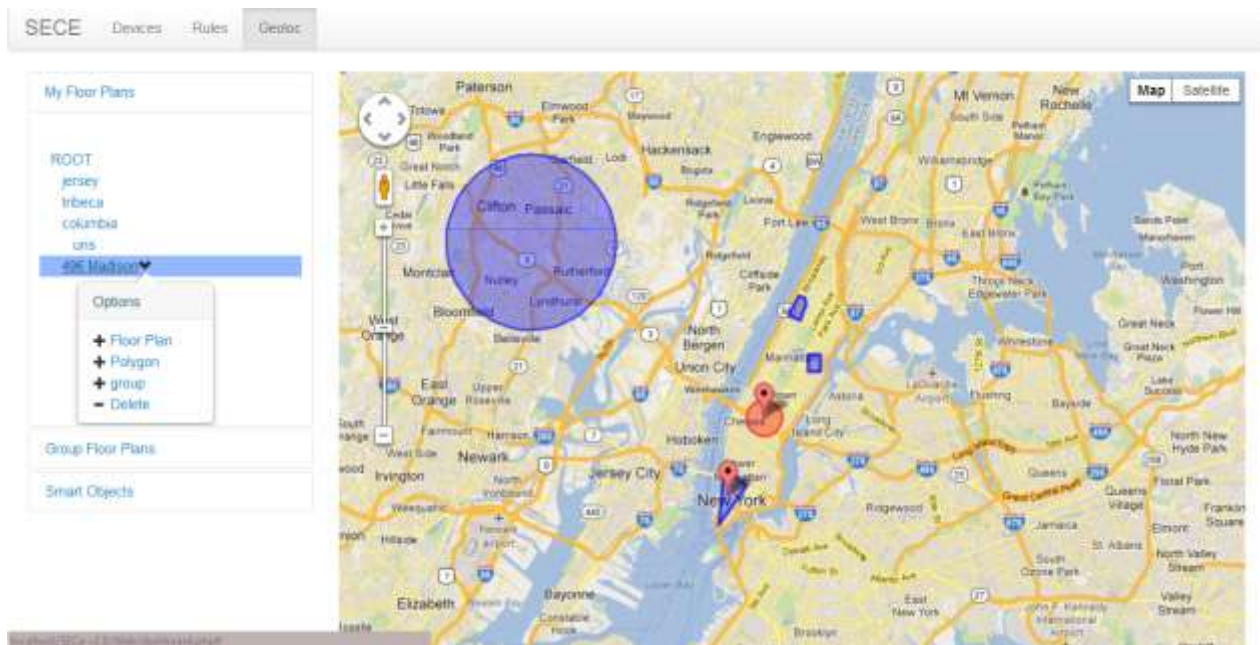


Fig 2.2 Dashboard with context menu in actual implementation

The context menu has items to:

- add a floor plan
- Add a polygon
- Add a group
- Move to trash (not present for ROOT since root always exists and cannot be deleted)

When the user clicks on the object link within the tree they are redirected to the respective edit object(i.e. polygon or floor plan) page.

When the user clicks on "add a floor plan", the "upload floor plan" UI will appear where the user can create a new floor plan and orient it using google maps view. The parent of that floor plan will already be pre-selected.

When the user click on "add a polygon", they will be taken to the polygon editor. There will be a polygon editor UI for Google Maps, and a separate one for floor plans. The system selects the appropriate polygon editor depending on the parent node type. When the user clicks on "add a group", a new item will appear in the tree and the user can edit it's name. On the left side there will be three tabs: Group Floor Plans, My Floor Plans, and Smart Objects. The user can click on any of them to unfold the contents. Group Floor Plans tab contains floor plans floor plans and polygons created by other users or obtained from public sources. The assumption here is that each user will have his/her own polygon database, but objects from other people's databases may be pulled into it, provided that the user has the necessary permissions to do so. My Floor Plans presents the contents of the user's geospatial database presented as a tree. The user can drag and drop items from the tree to reposition them within the tree. Each item in the tree will also have a context menu. The context menu contains items to add a new item in the tree as the direct descendant of the currently selected node, modify the node, or delete it (move to trash).

Smart Objects tabs presents a list of Smart Objects overlaid over the hierarchical polygon tree (My Floor Plans). Only the Smart Objects which the user configured in the Device Manager (another part of the SECE UI) and for which a precise location information is available are overlaid this way. Smart Objects for which precise location is not known may be shown in a flat list. Each element in the tree will have a context menu. The user can use the context menu to edit the Smart Objects' parameters or hide it from the Geoloc UI. When the user selects the edit option the UI will switch to the Smart Object Locator component.

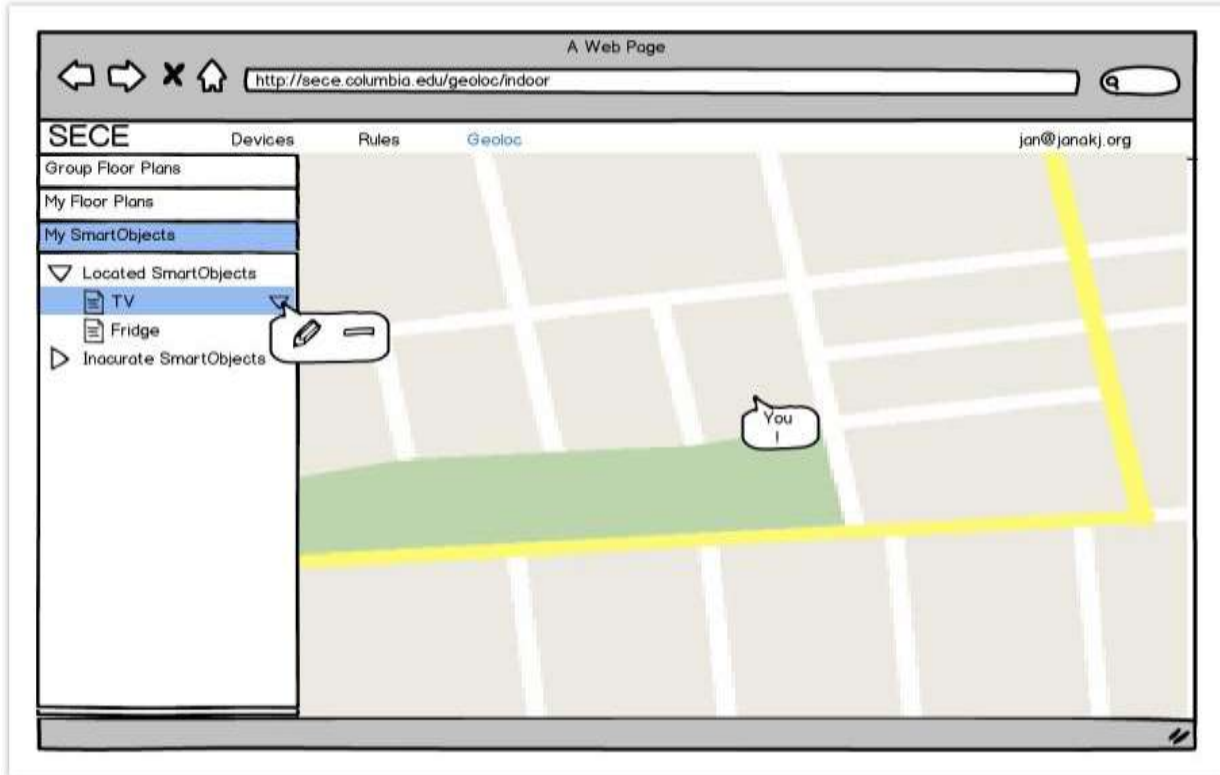


Fig 1.3. Dashboard initial proposal mockup with smartObjects accordion view

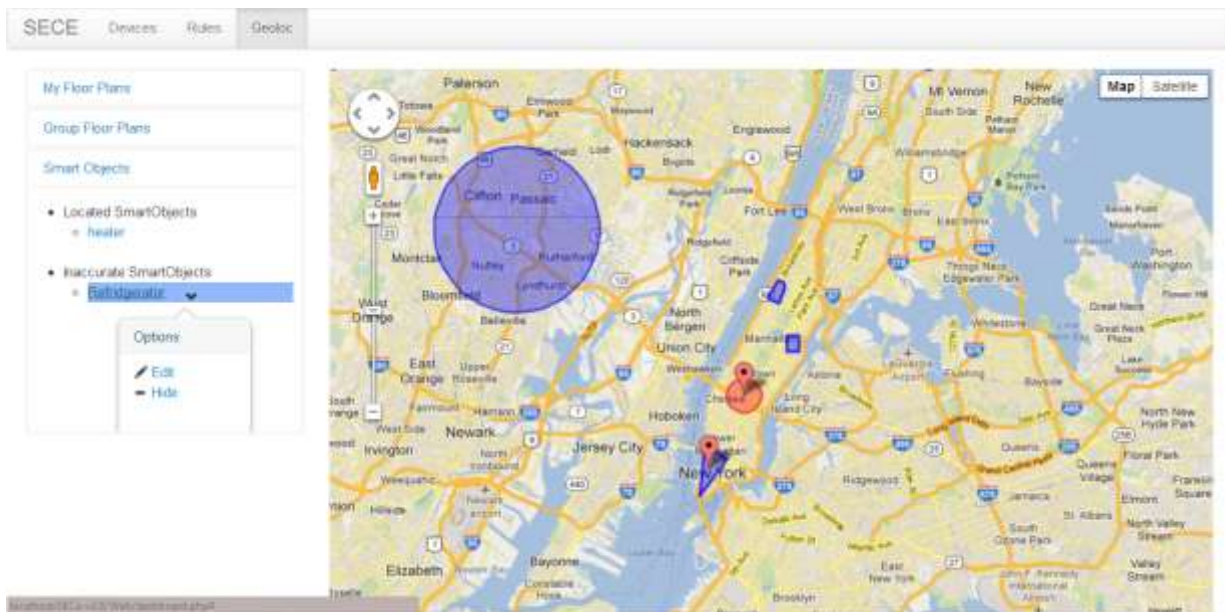


Fig 2.3 Smart Object tree with context menu in implementation

The right hand part of the Dashboard shows a Google Maps view with polygon overlays and Smart Objects markers.

All of these components within the dashboard are rendered through php which interacts with the mysql database to get the SECE objects which includes the groups, floor plans, polygons and smartObjects. All these objects are then converted into javascript objects and rendered onto the google maps view which is shown on the dashboard. The tree is displayed in the bootstrap accordion on the left side of the page, each of the components on which is highlighted as soon as the mouse hovers over it. And on clicking on the arrow which appears on hover the context menu as described above appears. The context menu is a bootstrap popup which is triggered on-click.

2. Floor Plan Editor

The Floorplan Editor component allows the user to upload new floor plans (in form of images) and orient them in the world using Google Maps. Floor Plan editor may not be the best name for this feature because some of the plans uploaded by the user may not be, in fact, floor plans. The user gets to the Floor Plan Editor from the Dashboard by selecting "Add new floorplan" context menu option on an item in the tree.

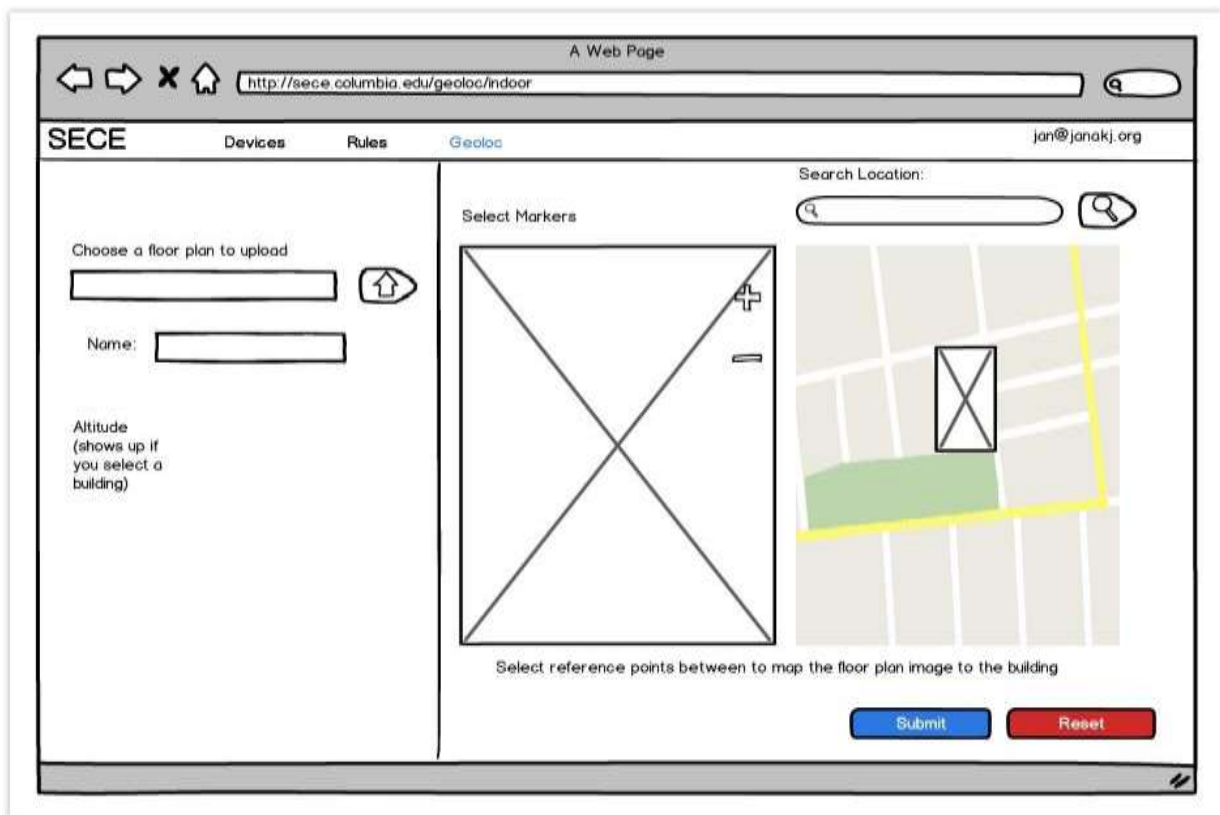


Fig 1.4. Adding Floorplan view in initial proposal mockup

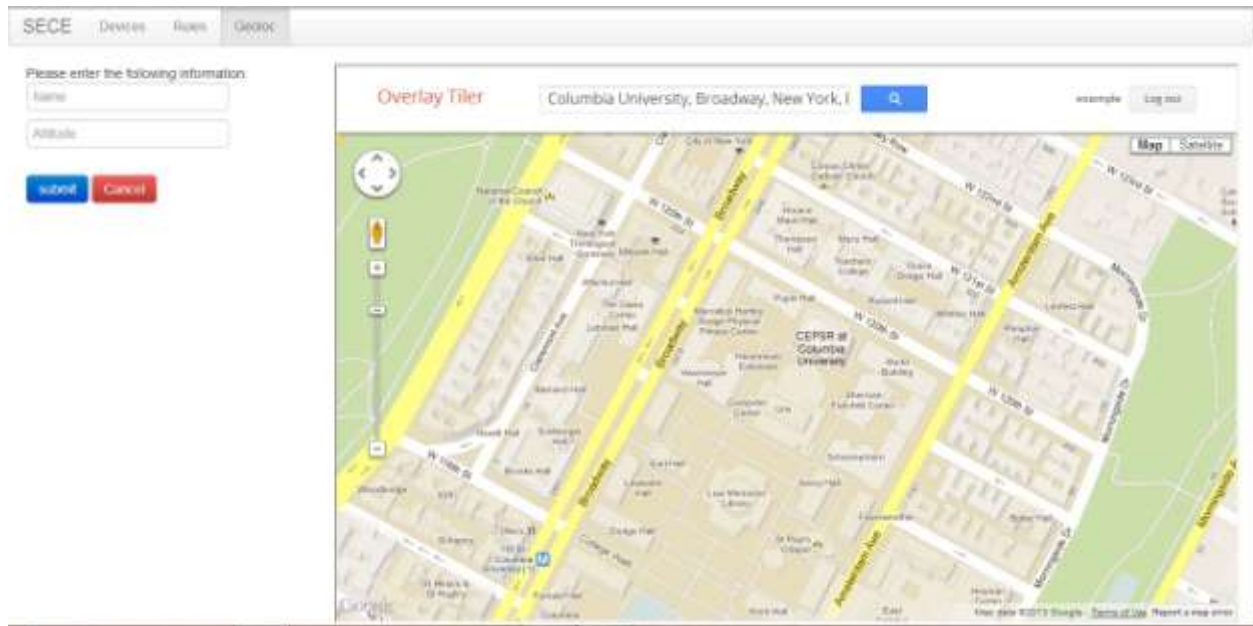


fig 2.4 Add floor plan in implementation

On the left there will be a simple form where the user can add the name, altitude for the floorplan and selected a floor plan image on his/her local computer or drag and drop an image onto the google maps area.

We will use some of the more recent features of Google Maps so there will be no need to provide reference points anymore. The search bar above the map view can be used to zoom in the map view into the appropriate place quickly. This new feature being the overlay-tiler component which has been made available as a go application. The floorplan editor therefore has a iframe on the right which has the search bar along with a google map onto which the user can drop the floorplan image. once the image is dropped it is uploaded into the irtlaptop server by the go application in the background while the image can be manipulated by moving the markers to fit the area on the map which the user intends. Once the user has done this and entered the additional information regarding the floor plan they can click the submit button.

After clicking the submit button the floor plan, along with all its attributes (location data, form data), is saved into the database and the user is taken back to the dashboard where the floor plan appears in the geospatial object tree. The floorplan and all the attributes are obtained by the geoloc application through a json file which contains all the details required by the application passed to the application by the overlay-tiler application. When the user selects the object on the tree they are redirected to edit an already existing floor plan, he/she is taken to the same UI component (this). Since the overlay-tiler is a separate component embedded into the geoloc system editing a floor plan is as good as creating a new floorplan and replacing the old one with the one which the user uploads through this edit interface.

There were a number of challenges faced in order to integrate and get information from the overlay-tiler application. This challenge faced is basically because the 2 applications run on different servers and

to send a direct call to the other application would cause a cross-site scripting error. To overcome this challenge all the communication between the 2 applications occurs through the php proxy. also the UI of geoloc currently uses an iframe and not a UI of its own to implement this since the go application overlay tiler is running on the port 8080 and the apache server on which we are running the UI cannot directly post to it due to cross site scripting restrictions.

3. Polygon Editor

The floor plan polygon editor is a component of the Geoloc UI where the user can draw polygon overlays over Google Maps or Floor plans in order to create a hierarchical database of geospatial objects. The user is taken to the Polygon Editor UI when he/she clicks on the corresponding item from the contextual menu in the Dashboard object tree.

The left side of the UI contains a simple form with information about the polygon being created or edited. The right pane shows either a Google Maps view or a Floor Plan, depending on the type of the parent object from the Dashboard object tree.

3.1 Floor Plan Polygons

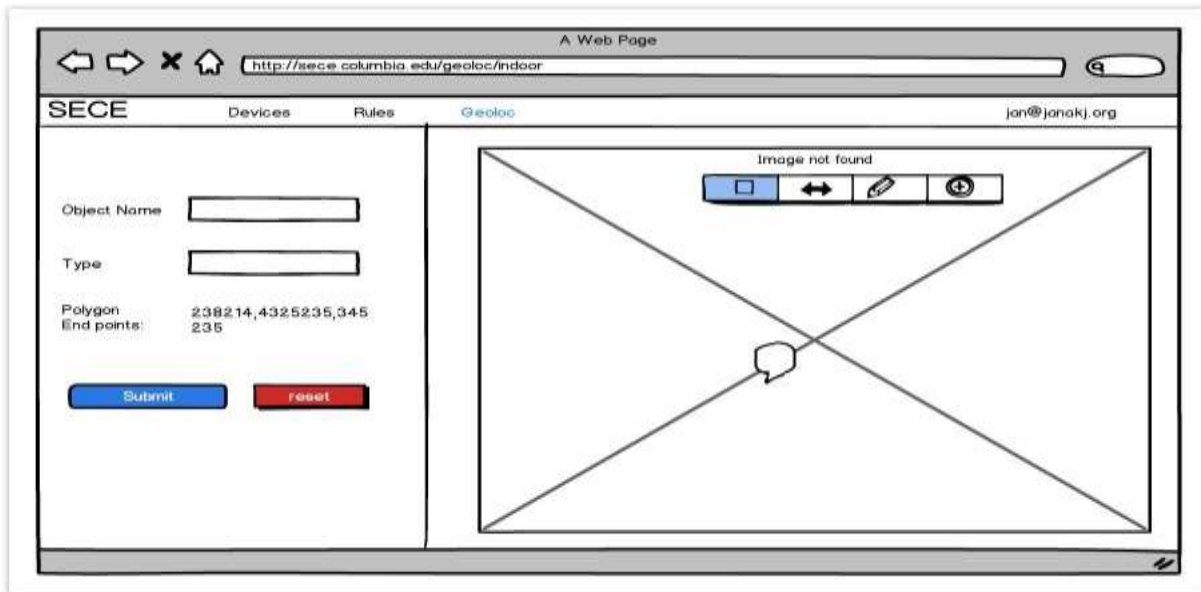


Fig 1.5. Add Floor plan polygons view in initial proposal mockup

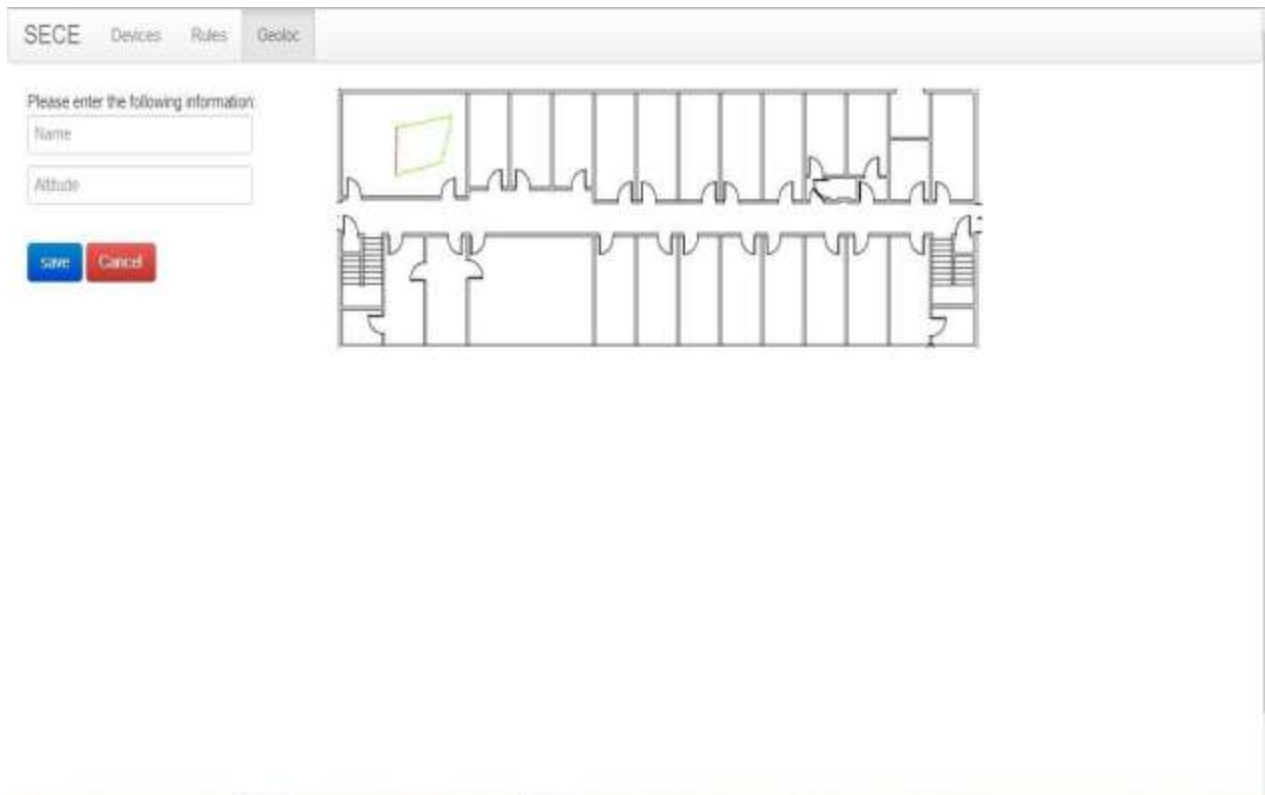


Fig 1.5 Add polygon to Floor plan in implementation

The right side of the screen then shows the floor plan covering the entire pane. the user can click on the floor plan image to indicate the vertices of the polygon to be added to the floor plan. When done the user clicks on submit to finish the process. At that point the polygon will be saved in the database and the user is taken back to the Dashboard. The user need not complete the polygon as the javascript plugin used to implement the polygon drawing on the floor plan ensures that every time the user selects more than 2 points on the plan they are stored and displayed as a closed polygon.

3.2 Google Maps Polygons

The right side of the screen shows a Google Maps view. The search box above the map view can be used to quickly reposition the map view. The user can draw polygon overlays over the maps view, using the usual set of drawing tools (circles, rectangles, multi-line polygons, etc). There should be a couple of attributes the user can set (line color, opacity) that will be saved into the database along with the polygon.

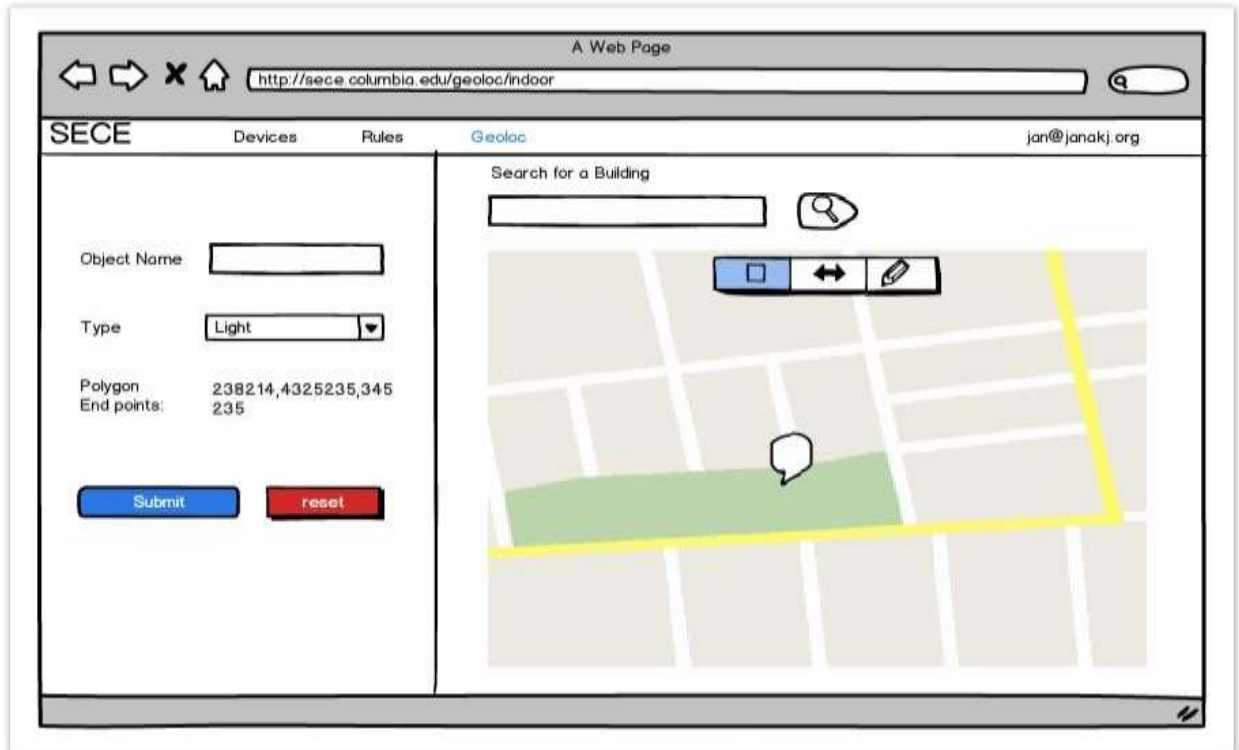


Fig 1.6. Add polygon to google maps view in initial proposal mockup

When done the user clicks on the submit button and is taken back to the Dashboard.

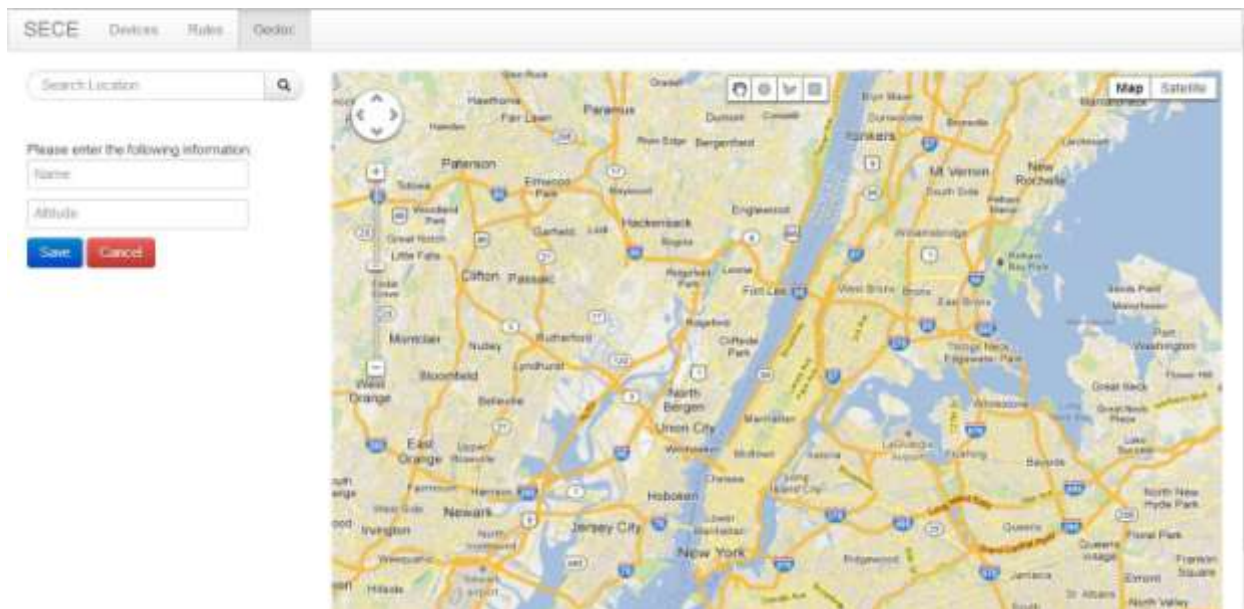


Fig 2.6 Add Map polygon view in implementation

4. Smart Object Locator

The purpose of this component is to give the user the option to select precise (down to cm accuracy) position for selected Smart Objects without the use of specialized tools (GPS, indoor location). This is mostly useful for stationary Smart Objects, such as those commonly used for home or office automation. The locator component works with a database of Smart Objects and their location information managed by another component of the SECE interface (the Device Manager).

The Smart Object location database is a collection of unique Smart Object IDs. For each Smart Object is contains the location information of that Smart Object.

The location information consists of:

- 1 Longitude
- 2 Latitude
- 3 Altitude
- 4 Accuracy
- 5 Altitude Accuracy
- 6 Timestamp
- 7 Location source (GPS, wifi, manual)

The Geoloc UI will select Smart Objects with accuracy level below some pre-configured threshold and let the user provide a more accurate location for that object using Google Maps and previously uploaded floor plans. When the user selects a Smart Object in the Dashboard, he/she is taken to the Smart Object Locator component.

The components consists of two panes:

Google Maps view on the left and (optional) floor plan view on the right.

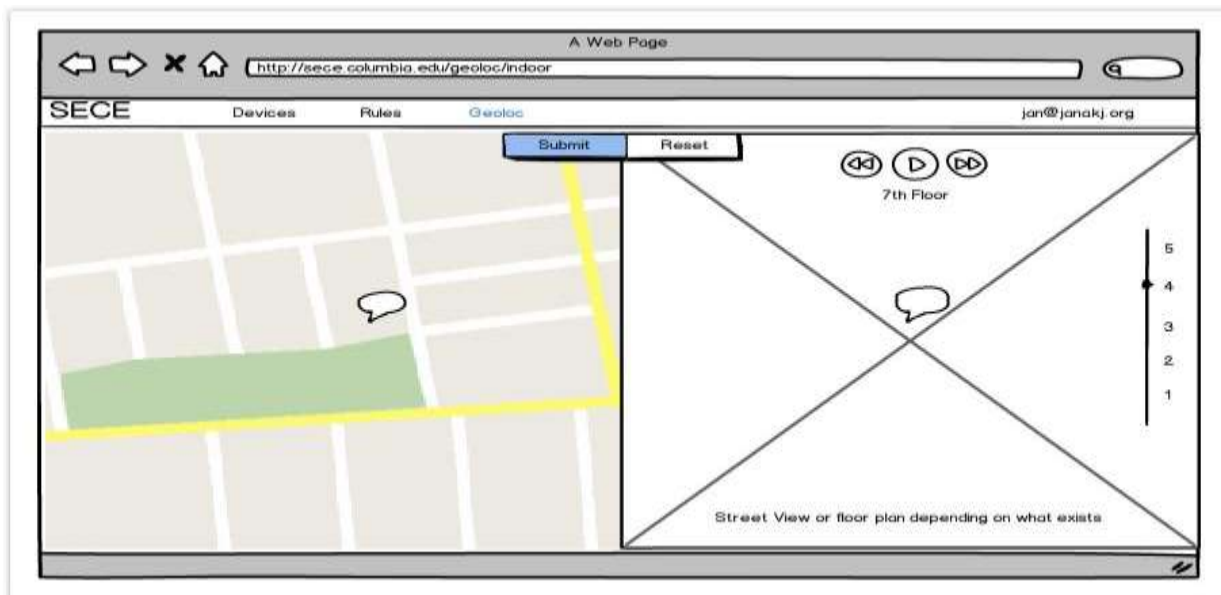


Fig 1.7. Edit SmartObject view in initial proposal mockup

The user can drag and drop the Smart Object's marker in the maps view to locate it. If the object is dropped onto a previously created floor plan, the floor plan will be shown on the right side. The object's marker will be shown on the floor plan too and the user can drag and drop it there, too. If multiple floor plans are available for the object's location (such as when the system contains floor plans for individual floors). If multiple floor plans are found then the smallest enclosing floor plan is shown first. The user can select the desired floor plan using navigation above the floor plan. If the selected floor plan has an altitude range, another navigation item shows on the right where the user can also select precise altitude for the smart object.

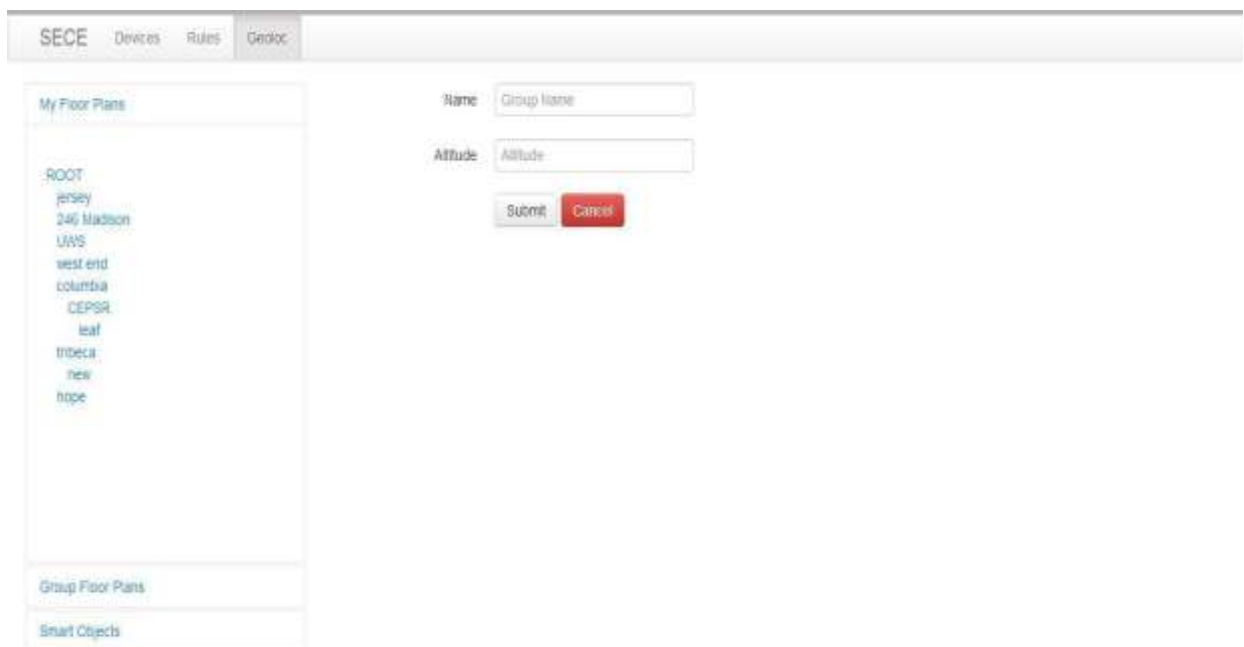
When done the user clicks on Submit button, the absolute location of the Smart Object is calculated and its database entry is updated, setting the source of the location data to manual. The user is then taken back to the Dashboard. The validation which recognises if the smartobject belongs to a floor plan is done through a mysql bounds query which can determine if a point exists inside a polygon geometry object.

Currently due to difficulties faced while coupling overlaytiler with the rest of the geoloc UI application the right hand part of the UI which can be used to locate a smartobject within a floor plan is not yet functioning and the user can only position the smartobject using either google maps or street-view.

5. Groups

The geoloc UI also provides the functionality to add groups into the system. This abstraction has been added so that incase the user wants to couple various objects entities without any location, floor-plan or even altitude attached to it. In order to be able to couple objects in this manner we have included the abstraction of groups into the system.

Groups also appear as an object in the object tree and the user can add new floor-plans, polygons and groups into this group by clicking at the respective links in the context menu. the group can also be deleted in this way. The most important example of a group is the ROOT which is the parent of all the objects of the system.



The screenshot displays the SECE Geoloc interface. At the top, there is a navigation bar with tabs for 'SECE', 'Devices', 'Rules', and 'Geoloc'. Below the navigation bar, the interface is divided into two main sections. On the left, there is a tree view titled 'My Floor Plans' which contains a list of objects: '.ROOT', 'jersey', '246 Madison', 'UWS', 'west end', 'columbia', 'CEPSR', 'leaf', 'tribeca', 'new', and 'hope'. Below this tree view, there are two additional sections: 'Group Floor Plans' and 'Smart Objects'. On the right side of the interface, there is a form for creating a new group. It includes two input fields: 'Name' with the placeholder text 'Group name' and 'Altitude' with the placeholder text 'Altitude'. Below these fields are two buttons: 'Submit' and 'Cancel'.

Future Work

Implementation

Due to difficulties faced in order to couple the Geoloc UI application with the overlay tiler application which was used to generate tiles of the floor plans to be overlaid on top of google maps, there are a few features which are not yet implemented in the system.

- 1 The file upload in the add floorplan feature occurs through drag and drop and not through the form on the left
- 2 Cannot adjust the location of a floor plan on a floor plan
- 3 Cannot precisely locate polygon added on to a floor plan

Other

- Can we get access to the database that Google maps already maintains?
- Can we also get access to polygons created by Google, and not just by users?
- Can we obtain the polygon for Old Town Square in Prague?

Consistency Checks:

- Do we need to ensure that the logical ordering of items in the tree is synchronized with their geo-based ordering?
- Is this something the user would expect?
- Maybe we could implement a button that would make it possible to re-arrange the tree so that the user can see either the logical ordering or the geo-based ordering of the tree
- The initial prototype will be without any consistency checks
- We may need to do A/B testing on a small group of users to see if this is required at all
- For some of the features in the UI A/B testing with a small group of users may be needed

References

- 1 [A Noob's Guide to Creating Google Maps Tiles from a Plain Old Image | Bryce Thomas](#)
- 2 [Overlays - Google Maps JavaScript API v3 — Google Developers](#)
- 3 [Map Types - Google Maps JavaScript API v3 — Google Developers](#)
- 4 [MySQL :: MySQL 5.0 Reference Manual :: 12.16.5.4 Functions for Testing Spatial Relations Between Geometric Objects](#)
- 5 [MySQL. Create polygon geometries](#)
- 6 [http - The Go Programming Language](#)
- 7 [How to use MySQL Spatial Extensions: Using Circular Area Selection](#)
- 8 [Creating a Store Locator with PHP, MySQL & Google Maps - Google Maps API — Google Developers](#)
- 9 [Writing Web Applications - The Go Programming Language](#)
- 10 [gis - See if lat / long falls within a polygon using mysql - Stack Overflow](#)
- 11 [Draw Cycle Overlay on google map - Stack Overflow](#)
- 12 [MySQL :: MySQL 5.0 Reference Manual :: 12.16.4.4 Populating Spatial Columns](#)
- 13 [PHP - unpacking Multipolygons from MySQL \(Page 1\) — Programming and Web Development — Forum](#)
- 14 [Draggable | jQuery UI](#)
- 15 [mysql - Traverse array as tree in php - Stack Overflow](#)
- 16 <http://irtlaptop8.cs.columbia.edu:8000>
- 17 <https://github.com/zsalzbank/SECE-Location-API>
- 18 <https://github.com/zsalzbank/SECE-Device-Manager>
- 19 <http://dev.mysql.com/doc/refman/5.1/en/functions-for-testing-spatial-relations-between-geometric-objects.html#relations-on-geometry-mbr>
- 20 <http://benalman.com/projects/php-simple-proxy/>
- 21 <https://github.com/riddhinm5/SECE>
- 22 http://brittonkerin.com/image_region_selector/irs_demo.html