# **CSC 431** Cadastre UI System Architecture Specification (SAS)

**Team 4: !bad**

|  |  |
| --- | --- |
| Amanda Davis | Team Member |
| Halle Fields | Team Member |
| Nathan Fox | Team Member |
| Devin Michaels | Team Member |
| Brandon Purvis | Team Member |
| Alex Voutsinas | Team Member |

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Authors | Change Comments |
| 0.0.1 | 03/18/2018 | Davis, Fields, Fox, Michaels, Purvis, Voutsinas | Created document and filled in initial information |
| 0.0.2 | 05/06/2018 | Davis, Fields, Fox, Michaels, Purvis, Voutsinas | Proofread document, final updates |

# Table of Contents

1. [System Analysis……………………………………………………………………………………………….](#_3dy6vkm) 4
   1. [System Overview](#_1t3h5sf)[……………………………………………………………………………………](#_3dy6vkm)4
   2. [System Diagram](#_4d34og8)[……………………………………………………………………………………](#_3dy6vkm).4
   3. [Actor Identification](#_2s8eyo1)[…………………………………………………………………………………](#_3dy6vkm).5
   4. [Design Rationale](#_17dp8vu)[……………………………………………………………………………………](#_3dy6vkm).5
      1. [Architectural Style](#_3rdcrjn)[…………………………………………………………………………](#_3dy6vkm)5
      2. [Design Patterns](#_26in1rg)[………………………………………………………](#_3dy6vkm)…………………....5
      3. [Framework](#_lnxbz9)[………………………………………………………………………………](#_3dy6vkm)..5
2. [Functional Design](#_35nkun2)[……………………………………………………………………………………………](#_3dy6vkm).7
   1. [Accepted Polygon](#_1ksv4uv)[…………………………………………………………………………………](#_3dy6vkm)... 7
   2. Rejected Polygon[……………………………………………………………………………………](#_3dy6vkm) 7
3. [Structural Design](#_44sinio)[……………………………………………………………………………………………](#_3dy6vkm)..8
   1. Class Diagram[………………………………………………………………………………………](#_3dy6vkm). 8

# Table of Figures

1. [System Diagram………………………………………………………………………………………………](#_3dy6vkm)4
2. [Accepted Polygon…………………………………………………………………………………………](#_35nkun2)….7
3. [Rejected Polygon…………………………………………………………………………………………](#_44sinio)…..8
4. Class Diagram………………………………………………………………………………………………....8

### System Analysis

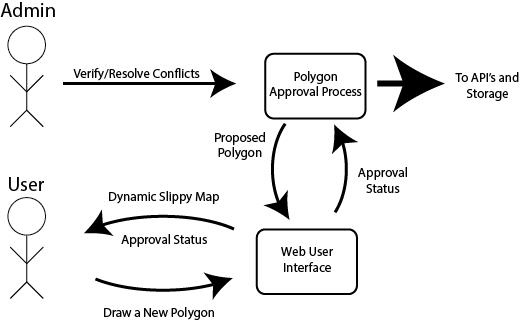
#### System Overview

Our system makes sure that citizens in Colombia can lay claim to property in a timely manner. We are responsible for the cadastre layer of the map, drawing the polygons that dictate property borders. Our team needs to use existing code which draws polygons, and we will take this code and find ways to have polygons snap into place to make choosing boundaries easier, since they will be aware of surrounding polygons. The system will also have a way to confirm or deny property boundaries based on existing boundary information. Overlapping polygons will be rejected and dealt with by an administrator, while polygons that do not overlap will be accepted. This data will then be exported to a database to be stored and used.

**Cross Cutting Requirements**

1. Polygon Drawing – The ability to draw polygons is used by several of the requirements for this project. Neighbor awareness, conflict detection, and almost every other requirement needs polygon drawing implemented before they can be addressed. It must be done before outputting polygon data and therefore is necessary to do before other teams can make use of our data.
2. Neighbor Awareness – Neighbor awareness is necessary for customizable snapping, because polygons can only really snap to a location if they are aware of other neighboring polygons. This step is necessary before final polygon data can be submitted and therefore must be done before other teams can make use of our data.
3. Conflict Detection/Resolution – This requirement is necessary to complete the review/approval process for polygons, as no polygon can be approved if it collides with one that is already established. That means this process is important to the Workflow Management team as they manage whether these polygons are approved or not.
4. Storing polygon data on external database – This requirement provides the output of our project, and thus intersects with the requirements of groups that make use of that output (Data, Search, Download of Public Facing Data, etc).
5. Slippy Tiled images – This requirement intersects with storing the polygon data because the data would have to scroll along with the map. Therefore it is useful to the back end teams that would use our output.

#### System Diagram



#### Actor Identification

* User
  + A user in our system has the ability to draw new polygons and give them attributes.
* System Administrator
  + A system administrator can verify the validity of newly added polygons and resolve conflicts in polygon overlapping.
* Application Server
  + Application server hosts the server-side code that runs the back-end of our web-based UI. In our case, we will be using Amazon Web Services(AWS) to host our layer.
* Web Server
  + The web server hosts our client-side code which contains all of the UI side, which is also going to be hosted by an AWS server.
* Database System
  + We will be interfacing with a database system as provided to us by the Back End and team of this project.

#### Design Rationale

##### Architectural Style

Our group will be utilizing a 3-tier architecture for our web-based system. For the front-end, we will be using a typical web-development set-up with a Javascript-based UI and HTML/CSS elements. For the server-side, we will be using a Java framework that interfaces with the API team from this project. Finally, we will be utilizing the databases provided to us by the Back-end team from this project.

##### Design Patterns

* Multiton
  + Used for our Polygon class
* Factory Method
  + Define interface for defining objects in our system but create subclasses for instantiation
* Facade
  + Provide an easy-to-use interface for user input to interact with the complex system running beneath the UI
* Chain of Responsibility
  + Used for the approval workflow of colliding polygons

##### Framework

Javascript Angular Framework - for front end

NodeJS Framework - for back end

GeoJSON Entity Framework - to work with user-submitted data to form GeoJSON packages of information to send to related groups in the project teams

Bootstrap/CSS Framework for HTML development -- Will be used to format our UI portions

### Functional Design

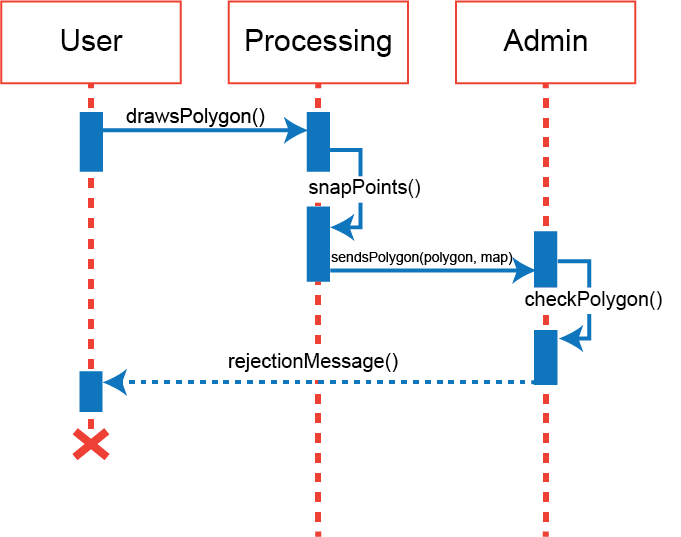
#### Accepted Polygon

### 

### 

#### 

#### Rejected Polygon



### Structural Design

#### 

#### Class Diagram

