**Image Tagging**

**Technologies used:**

Front end: HTML5, CSS3. Javscript

Frameworks: NodeJs, ReactJs

Database: MongoDB

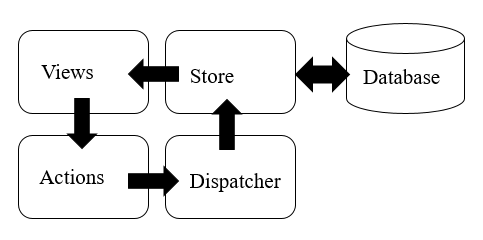
Web service: ExpressJs

**Architect Technologies:**

**Flux:**

Used by most websites like, Facebook, Netflix etc. Invented by Facebook during 2013.

*There is slight variation only store speaks to database however in actual actions speak to database*.



**Phase 1:**

We have agreed to implement a web based application for labelling image for three different types of cells. For this first phase I have considered and implemented the following scenarios.

1. Storing images in MongoDB
2. Web based application to label the cell
3. Delete the label if not valid
4. Retrieving an image.

I have installed mongoDB in my laptop, and as a user I will store different images to the database and I can give the path to store the image file (which is hardcoded as of now). Html page to show the image in web and labels. Each user can label the cells which they want by doubling on that cell. These labels are hardcoded (red, green, orange) as of now. If user does not want to keep the previous label on the cell he can remove just clicking it twice. For this phase I have done 2 separate programs for inserting and retrieving images from database and Labelling of cells in an image. Later point in time, we can merge these two projects to make it one single project.

***Chrome or firefox web browser, used mostly support png or jpeg images, so I have converted image from tiff to png format.***

**To be done**

1. Storing labelling information in the database. (needed)
2. REST APIs to get points and tag details on the image.
3. Process the image for similar cells in the image. (not possible with technical difficulties)

If these are fine, then I can do further enhancements to store the labels of an image in database with its label and positions in the image. To work full fledge we need to make an image gallery of different images. If a user selects a cell, we should be able to recognize other cells in the image. These no. of cells and its position should also be stored in the database.

**Phase 2:**

Created the html file as the deployable app in **NodeJs** framework. Updated the tags given by the user to the **MongoDB database.** External user can access the points by the REST APIs developed in expressJs. REST Apis can be used for reading the values from the mongoDB.

**Result:**

When you enter this URL you will get the points and tags entered by the user.

[**http://localhost:3000/users/getTagpoints**](http://localhost:3000/users/getTagpoints)

[{"\_id":"57018220c5a6824c15b66a97","tag":"red","positions":[{"x":378,"y":256},{"x":538,"y":241},{"x":456,"y":96},{"x":288,"y":101}]},{"\_id":"57018229c5a6824c15b66a98","tag":"orange","positions":[{"x":163,"y":258},{"x":310,"y":209},{"x":217,"y":161},{"x":344,"y":154}]},{"\_id":"5701822fc5a6824c15b66a99","tag":"green","positions":[{"x":671,"y":268},{"x":489,"y":324},{"x":528,"y":119}]},{"\_id":"5701823ac5a6824c15b66a9a","tag":"green","positions":[{"x":596,"y":303},{"x":474,"y":278},{"x":407,"y":159}]},{"\_id":"5701823fc5a6824c15b66a9b","tag":"red","positions":[{"x":175,"y":265},{"x":109,"y":165},{"x":195,"y":131}]},{"\_id":"57018249c5a6824c15b66a9c","tag":"green","positions":[{"x":302,"y":266},{"x":256,"y":302},{"x":372,"y":338},{"x":331,"y":224}]},{"\_id":"57018251c5a6824c15b66a9d","tag":"orange","positions":[]},{"\_id":"57018257c5a6824c15b66a9e","tag":"orange","positions":[{"x":535,"y":449},{"x":168,"y":401},{"x":221,"y":444}]},{"\_id":"5701825dc5a6824c15b66a9f","tag":"orange","positions":[{"x":265,"y":367},{"x":59,"y":361},{"x":151,"y":389}]}]

This points could be used in mat lab to get the exact tag location (I have seen in Mathworks, but needs to be checked with matlab experts).

**Phase 3:**

1. There are several end points in web servers to support CRUD operations in database mongoDB.

END points:

<http://localhost:3000/getTagPoints>

Returns all the Tags

<http://localhost:3000/addTagpoints>

Adds a new Tag with all positions

<http://localhost:3000/deleteATag>

Returns a Tag with id

<http://localhost:3000/updateATag>

Updates a Tag with new positions

<http://localhost:3000/removeAll>

Remove all tags from database.

***These end points works only after the server started.***

1. Tags can be removed by clicking each and a cross mark. If users has tagged a cell with a wrong a information, they can just delete it using 2 clicks.
2. Tags can be dragged to a correct location if it was misplaced. If a tag is placed in the wrong cell in the image, it can be placed to the correct placed to correct location by dragging it.

To conclude, in an image users can click on image in multiple places and a right click will be used to add the tags; they can drag the tag to the proper cells on the image; they can remove the tag; In every operation, tags will be read from the database, deleted in the database and updated in the database. Thus, we can retrieve any details from the database. There are many future enhancements to the website in many ways which can be taken as we proceed further.