# Watchtower Final Project Report
# Team Awesome

Members : Pranay Singh
Date : 29th April 2016

## Introduction

Watchtower is a Lab Workflow Management tool that simplifies the process of tracking and reporting observations about lab artefacts by assigning them a unique ID and then using the Github Version Source Control System to keep an updated copy in an organised project structure. The whole application is aided by a friendly user interface that is simple to use. The application consists of a Rest application that responds to a user interface and connects to a MongoDB database. The application is built using the MEAN stack.

## Architecture :-

1) **NodeJS Server :-** The application uses a NodeJS front end server to handle the front end and the business logic of the application. The reason to choose NodeJS was because of its non blocking code via asynchronous callback that register functions in the event loop that are triggered later when the preceding operation is completed. So instead of waiting for a specific piece of code to return an output node carries on. When the output is returned an event is triggered which causes the callback function to execute in the event loop. Due to this model, there are huge gains in concurrency due to which NodeJS can scale to high i/o rates and even can handle multiple concurrent requests with ease.

2) **ExpressJS :-** ExpressJS is used to handle the routing for all the calls to the server. ExpressJs was chosen due to high availability of supporting packages that make handling routes much easier. For example the authentication for Signups and Logins was made using the Passport model for Express. All the Get/Post requests from the front end are passed through the Express router before they are accepted by the core application.

3) **MongoDB :-** MongoDB was used as the data store for this application. This decision was made due the schema-less paradigm of MongoDB. Since this is still a test application I wasn’t sure of the Data models that I would be using to store data. Thus to provide me the availability to change models on the go a NO-SQL database was ideal. Thus Mongodb. I also used Mongoose as a shim over Mongodb. Mongoose is a lightweight wrapper around Mongodb that provides an easier way to create and use models as well as find and store data.

4) **Github :-** This project warranted the need of a reliable source version control system that could be used to track and store the observation data that would be coming in from multiple users on a daily basis. Since I have extensively used Github it seemed to be the best option for the job. I connected with Github using the Github API that can be found on their website. The API enabled me to create a folder/file in a repository and manipulate the data inside of it.

5) **UUID :-** In order to track lab artefacts I needed an ID generator that would have a very small repeatable percentage. Therefore I decided to go with UUID version 4, which created a unique ID composed of your MAC address and the current timestamp.

## Results

The final outcome of the project was that any lab researcher can now use Watchtower to replace his/her lab notebook. Since Watchtower provides a glue between observing and tracking information, it forms a far superior product compared to current lab notebooks. Because of the architecture choices and the ease of use Watchtower can handle heavy use as well as provide an easy enough interface so that researchers don't have to invest time into learning how to use it.

## Future work

This is only the first step in solving the lab tracking problem. This seems to be a million dollar problem but Watchtower is a step in the right direction. With adequate user feedback and and customer research additional features need to be added at which point the Github API may turn out to be a hindrance and a custom SVC solution may need to be developed, but at this point it provides a firm footing to enter into the lab and increase customer retention. A possible GUI is to display the files visually may also be a direction to explore but not at the cost of  impacting functionality. As it was found with Watchtower, developing a GUI before developing the functionality is clearly a wrong approach to take.