AUTOMATED DEPLOYMENT of STEM CENTER TUTOR WEBSITE

By

Rajesh Vallabineni

This project is submitted to the Gannon University graduate faculty in

Partial fulfillment for the degree Master of Science in Computer and Information Science.

Option: **Information Systems**

Approved:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Dcn. Stephen T. Frezza, Ph.D., P.S.E.M.* |  | *Theresa M. Vitolo, Ph.D.* |
| Advising Professor in Charge of Research |  | Committee Member |
|  |  |  |
|  |  |  |
| *Mei-Huei Tang*, Ph.D. |  | *Mei-Huei Tang*, Ph.D. |
| Committee Member |  | Chair, Computer and Information Science Department |
|  |  |  |

Gannon University

Erie, Pennsylvania 16541

February 2016

## Acknowledgements

I am sincerely thankful to my stakeholder Mr. Adam Nogaj for providing necessary guidance regarding project designing and development. I am also grateful to my Project Advisor and a great teacher Dr. Stephen T. Frezza for his support through-out the project. I would like to thank my team member Gopal Morampudi who dedicated his time and knowledge in the implementation of this project, and people who have willingly helped me out with their abilities.

Abstract

This document describes some of the aspects of the STEM Center Tutor Information gathering website Project. This project includes the exploration, database design, testing and deploying of STEM website project. A detailed description of the system is included along with key learning experiences.

# List of Figures

[Figure 1: Existing system 9](#_Toc442274939)

[Figure 2: Proposed system Diagram 11](file:///C:\Users\raaj\Desktop\RV%20Doc%20v4%20stf.docx#_Toc442274940)

[Figure 3: Web Application Architectural diagram 21](#_Toc442274941)

[Figure 4: ER-Diagram 23](#_Toc442274942)

[Figure 5: Jenkins fitting 25](#_Toc442274943)

[Figure 6 : Create a new repository 26](#_Toc442274944)

[Figure 7: Repository file system 27](#_Toc442274945)

[Figure 8 : Work space in local system 28](#_Toc442274946)

[Figure 9: Add files to repository 29](#_Toc442274947)

[Figure 10: Commit files to remote Github repository 30](#_Toc442274948)

[Figure 11: Creation of Jenkins job 31](#_Toc442274949)

[Figure 12: XAMPP Control Panel 32](#_Toc442274950)

[Figure 13 Execute Shell 34](#_Toc442274951)

[Figure 14 Execute Windows batch command 34](#_Toc442274952)

# List of Tables

[Table 1: Position statement 13](#_Toc442179261)

[Table 2: Benefits and Supporting Features 14](#_Toc442179262)

[Table 3 Build Time Trend 33](#_Toc442179263)

Contents

[**Acknowledgements** **2**](#_Toc442275562)

[Abstract 3](#_Toc442275563)

[List of Figures 4](#_Toc442275564)

[List of Tables 5](#_Toc442275565)

[1. Introduction 8](#_Toc442275566)

[1.1 Overview 8](#_Toc442275567)

[1.2 Key Stakeholder needs 9](#_Toc442275568)

[1.3 Product Perspective 10](#_Toc442275569)

[1.4 Continuous Integration 11](#_Toc442275570)

[1.5 Product Position Statement 12](#_Toc442275571)

[1.6 Summary of Capabilities 13](#_Toc442275572)

[1.7 Alternatives and Competition 14](#_Toc442275573)

[1.8 Project Management Plan 14](#_Toc442275574)

[1.9 Curriculum Scope 15](#_Toc442275575)

[2 Requirements 16](#_Toc442275576)

[2.1 Elicitation: 16](#_Toc442275577)

[2.2 Key Design and Implementation Constraints 16](#_Toc442275578)

[2.3 Operating Environment 16](#_Toc442275579)

[2.4 Interface Requirements 17](#_Toc442275580)

[2.4.1 User Interfaces 17](#_Toc442275581)

[2.4.2 Hardware Interfaces 17](#_Toc442275582)

[2.4.3 Software Interfaces 17](#_Toc442275583)

[2.5 Communications Interfaces 17](#_Toc442275584)

[2.6 Test automation framework 18](#_Toc442275585)

[Types of Test Automation Framework 19](#_Toc442275586)

[2.7 Why Jenkins? 19](#_Toc442275587)

[3 Design 21](#_Toc442275588)

[3.1 Introduction 21](#_Toc442275589)

[3.1.1 Software Context 21](#_Toc442275590)

[3.1.2 Architecture diagram 21](#_Toc442275591)

[3.2 Data Design 21](#_Toc442275592)

[3.2.1 Major Internal Software Data Structures 22](#_Toc442275593)

[3.2.2 Database Description 22](#_Toc442275594)

[3.3 Why MySQL? 22](#_Toc442275595)

[3.4 E-R Diagram 23](#_Toc442275596)

[4 Deployment and Hosting 25](#_Toc442275597)

[4.1 Project Delivery Infrastructure 25](#_Toc442275598)

[4.2 Source Code Management 25](#_Toc442275599)

[4.3 Continuous Integration tool 30](#_Toc442275600)

[4.1.1 Installation and configuration of Jenkins 30](#_Toc442275601)

[4.3.1 Create a job 31](#_Toc442275602)

[4.4 Deployment 32](#_Toc442275603)

[4.5 Build 33](#_Toc442275604)

[4.5 Summary 35](#_Toc442275605)

[5 Testing 36](#_Toc442275606)

[5.1 Type of Frame work: 36](#_Toc442275607)

[5.2 Unit Testing Framework: 36](#_Toc442275608)

[5.3 Exception Handling and Synchronization strategy: 36](#_Toc442275609)

[5.4 Build Tools: 36](#_Toc442275610)

[5.5 Page Object Design pattern: 37](#_Toc442275611)

[5.6 Execution flow of the test cases 38](#_Toc442275612)

[5.7 Execution process: 39](#_Toc442275613)

[5.8 Configured Allure Reporting tool for projects: 39](#_Toc442275614)

[5.9 Summary 39](#_Toc442275615)

[6 Conclusion 40](#_Toc442275616)

[References 41](#_Toc442275617)

[Appendix A: Glossary 42](#_Toc442275618)

[Appendix D: Design Models 43](#_Toc442275619)

[Appendix E: Testing Log and Summary Status 47](#_Toc442275620)

[Appendix F: Screen Captures 61](#_Toc442275621)

# Introduction

* 1. Overview

The *STEM Center Tutor Information Gathering* website project was developed to improve the productivity of managing tutors for a large STEM Center at Gannon University. It is a website intended to significantly improve the process of adding new tutors to the STEM Center and for managing and reporting of their contact hours with the students the tutors serve. The site supports an administrator with the responsibility of scheduling individual tutors, and scheduling of students with the scheduled tutors. This STEM website provides a web platform to facilitate these three types of end-users: STEM Center administrators, tutors and students who schedule appointments with tutors.

The stakeholder for this project was the Director of STEM Center at Gannon University. As managing changing bodies of STEM tutors and students is central to their operations, they were interested in getting a custom tutor website that would streamline the process of collecting, updating and utilizing schedule information about tutors and students who need tutors. The core of this project centers on facilitating on-line scheduling and tracking of Tutor’s available and used hours. The site allows STEM center tutors to identify scheduling preferences/requirements, course responsibilities, and other specific information. The STEM Center administrators work with 50-60 tutors per semester and schedule them weekly. These assignments must take into consideration multiple factors: their own course schedules, personal schedules, and preferences about total number of hours and when they prefer to hold them. Each Tutor has expertise in one or more of the various courses that Gannon University offers under its various departments. Stakeholder is interested in a website that would allow tutors to log In to update this information periodically, allow administrator to view or edit the information for use in scheduling and students to request for help in the courses offered.

Another workflow that could be automated is the identification of tutors, Students can request the site admin to become as a Tutor. Tutors have an option to cancel the booked appointments by raising a request to the admin. In such cases, email notifications are sent to students about the fixed appointment. STEM website provides all these functionalities with a friendly User-Interface and satisfying the needs of end-users i.e. Tutors and Students.

Users need to register as a Tutor or Student and login to use above functionalities. Without Logging in to the website, end users can contact the site admin for any queries via contact page.

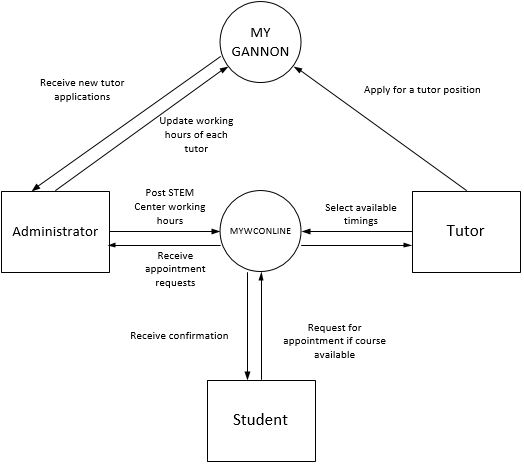


Figure : Existing system

Figure 1 shows the existing system. STEM center is currently using “MYWCONLINE” a third party website for Scheduling, Recordkeeping & Reporting for Academic Support Centers, with a subscription fees of $715/year or $65/month.

In-order to apply for a tutor position each student have to login to Gannon university website and fill a tutor/assistant application and mention their availability. Based on tutors availability administrator makes schedule in MYWCONLINE. Administrator has to track number of hours worked by tutor manually based on their availability. Tutor request for time off via email; students after entering into the “MYWCONLINE” portal can select the course from the drop down list and select tutors based on their availability.

* 1. Key Stakeholder needs

The Stakeholder of this project is targeting two types of users: One is Student and the other is Tutor.

Students would like to fix an appointment with a Tutor for the course they need. They should be able to view all the appointments fixed by them. Some students would also like to become Tutors, and this request must be submitted to the admin. As a user, they would be able to view and update their profile details along with changing password.

Tutors accept the schedules fixed by admin which are suitable to them. They confirm the slots in which they are available for the students to fix an appointment. They should be able to view all the appointments fixed with them. If any issue regarding appointment arises like cancellation of appointment, then they should request the admin for the needful by placing a request. As a user, they would be able to view and update their profile details along with changing password.

The stakeholder wants to fulfill these requirements in a web base application.

The existing “MYWCONLINE” portal used by the stakeholder have the following limitations which we would likely overcome in this new application:

* Student needs to select the course from a vast list of courses which are not appropriate to them. No filtering is carried out.
* It is difficult for admin to track number of hours worked by a tutor
* Tutors have to access another website in-order to apply for a tutor position
* Cancellation of appointments have to be done manually
  1. Product Perspective

The context of the product is user-centered. The application acts as an interface for interaction between admin and the users’ viz. Tutors and Students. The application is designed and developed based on the following process equivalents. The key context of the application is shown below:

The context shows how the users like tutors and students make their requests to the application which in turn sends to the admin. The Admin accepts/rejects the request based on his criteria and acknowledges the application. The application will show the status of the appointments and send the email notification to the users if required.

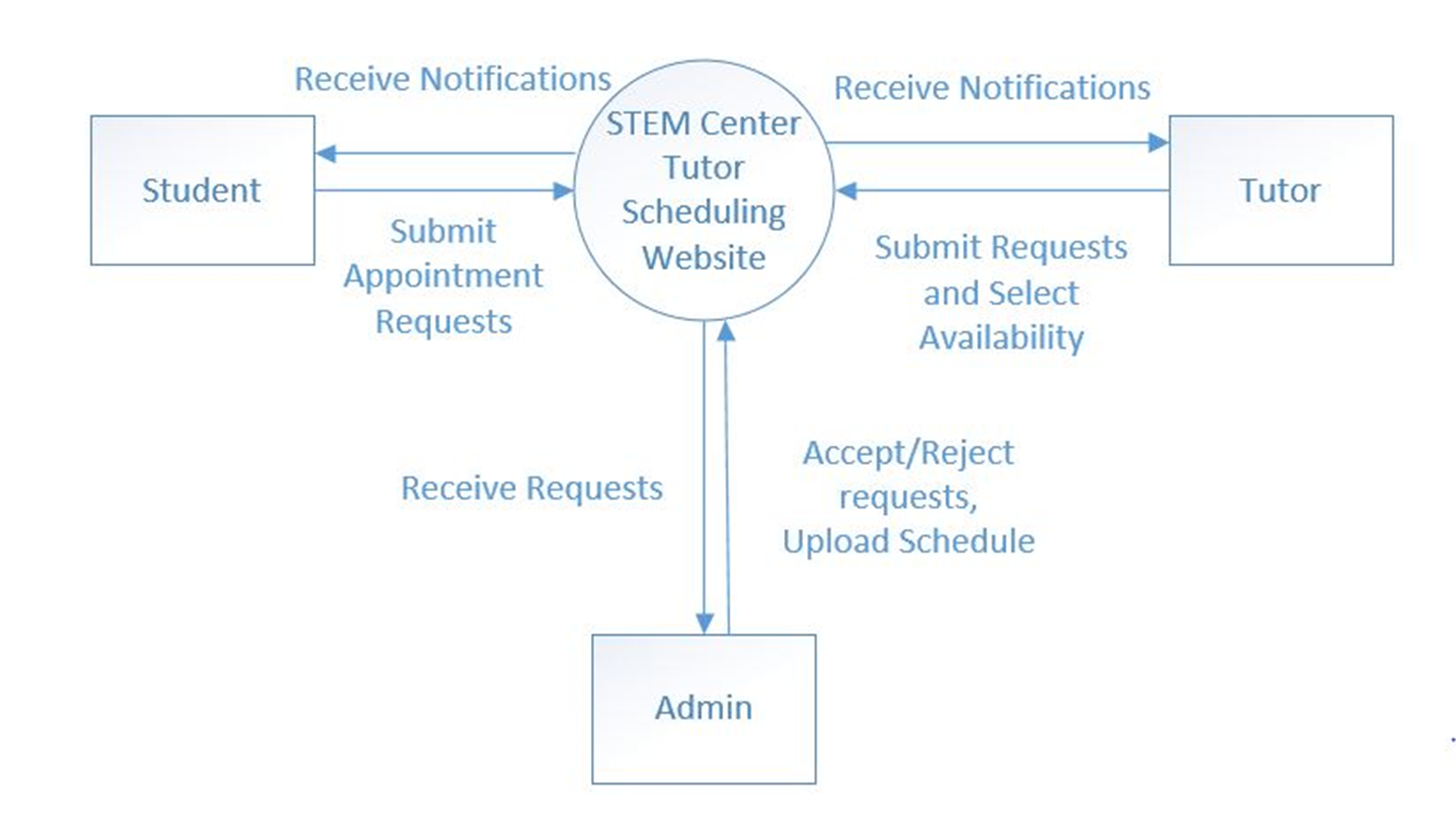


Figure : Proposed system Diagram

* 1. Continuous Integration

Continuous integration (5) is way to increase code quality without putting more burden on developing. We can handle tests and checks on a server and automatically reported back to the team. This process leaves the way for the more important process: continuous deployment.We have opted for continuous integration to make our testing and deployment automated.

Basically Continuous Integration is the practice of running your tests on a non-developer machine automatically every time someone pushes new code into the source repository. Continuous integration ensures that your software is built and tested regularly. It can help you to demonstrate that your software does what it claims to do, and that it does so correctly. It also helps you to rapidly release bug-fixes and more functional versions of your software.

CI is a way of ensuring that software is tested regularly. A continuous integration server automatically gets the current version of the code, rebuilds the software, and runs the tests. It then notifies the developers about the success or failure of the build and tests.

Continuous integration servers offer a number of advantages. They publish build and test results within a structured, web-based dashboard to make it easy to see the status of the build and tests, the successes, the failures and reasons for these. They can also present information on builds and tests in progress, and aggregate build-and-test runs from multiple developers. Continuous integration servers can also support various forms of notifications, for example, emails or RSS feeds. The continuous integration server runs on its own machine so the developer can continue to work on his own machine while the test are under way.

The most significant advantage of a continuous integration server is its ability to monitor changes to source code held under [revision control](http://en.wikipedia.org/wiki/Revision_control). If the code changes, then the server can automatically spawns a new build-and-test job. This means that the software is rebuilt and tested every time the code is changed. Typically, continuous integration servers will also allow build-and-test jobs to run at specific times.

This is why continuous integration helps your software to always be releasable: tests are run in response to changes to the code, and you are notified quickly when tests fails so that you can correct the reason for the failure  It is easier to fix a bug in something have written a few minutes ago, than something have written yesterday.

This has the tremendous advantage of always knowing if all tests work and getting fast feedback. The fast feedback is important so it can be always known right after someone broke the build what cause did that failed and how to revert it.  
  
If tests were run occasionally the problem is that a lot of code changes may have happened since the last time and it is rather hard to figure out which change introduced the problem. When it is run automatically on every push then it is always pretty obvious what and who introduced the problem.  
  
Built on top of Continuous Integration are Continuous Deployment/Delivery where after a successful test run your instantly and automatically release the latest version of your codebase. Makes deployment a non-issue and helps you speed up your development.

* 1. Product Position Statement

Table1 shows the position statement of the system. The development lead had perceived the problem of Administrator, tutors and students to complete all the process of the STEM Center Tutor management and illustrated the affects and results of the problem. We also listed the benefits of STEM Center Tutor scheduling application to solve this problem.

|  |  |
| --- | --- |
| Problem of | Adding new tutors to the STEM Center and for managing and reporting of their contact hours with the students the tutors serve. |
| Affects | * Stake holder * Students * Tutor |
| Results in | * Administrator have to access various web applications in order manage tutors availability which consumes man hours. * Tutors contact admin via email or directly, to update their availability * Students face problem to fix an appointment with appropriate tutor |
| Benefits of solution | * An administrator can log in at any time to add new tutors, accept requests from tutors, students and schedule them on availability. * Tutors will be able view/update the hours of their availability. * Students can request appointment with tutors on availability and contact admin if needed any course help |

Table 1: Position statement

* 1. Summary of Capabilities

Following table gives the summary of major benefits and features this STEM Web application provides:

|  |  |
| --- | --- |
| **Benefits** | **Supporting Features** |
| The Stakeholder adds new Department | Admin Console- Add Department |
| The Stakeholder adds new Course | Admin Console- Add Course |
| The Stakeholder accepts new Tutor requests | Admin Console- Tutor Requests |
| The Stakeholder schedules the timings for which appointments should be booked | Admin Console- Schedule |
| The Stakeholder addresses normal user | Admin Console- Enquiries |
| The Stakeholder adds students as tutors | Admin Console- Student as tutor |
| The Stakeholder addresses Tutors time off requests | Admin Console- Tutor Notifications |
| Tutor Schedules his availability | Tutor Console- Schedule |
| Tutor views all his appointments | Tutor Console- Appointments |
| Tutor request admin to modify appointment | Tutor Console- Request to admin |
| Student makes appointment request | Student Console- Make appointment |
| Student views his appointments | Student Console- Appointments |
| Student request to become tutor | Student Console- Become Tutor |
| Users can update their profile | Console- Profile |
| Users can change their password | Console- Change password |
| Users can send their enquiry | Home- Contact us |

Table 2: Benefits and Supporting Features

* 1. Alternatives and Competition

**Strengths**

* The Application robustly fixes the appointments and saves them in the database.
* Notifications will be sent to the users to their registered email.
* Easy to track no of hours worked by tutors
* Information available is consistent and clear
* Students can easily be able to apply for tutor position.

**Weaknesses**

* The application needs to be maintained as the number of Users increases.

**Security**

* This application cannot be used by unauthorized users due to login system incorporated.
* The application ensures no Data injection attacks happen and DB is maintained in robust way.
* Users cannot use admin features as the admin account is protected and handled separately.
  1. Project Management Plan

The project management plan was to the develop a robust system for the stakeholder by proving a better and time consuming service for the Users which was created and maintained by standard version control processes. This application development follows a well-defined and mature application development process based on use case analysis. This new STEM application is an alternative for the existing portal and makes the work easy and simple for the tutors and students. This work was divided among the two-person development team to enable parallel development and deployment of all three aspects (modules) of the project. Each of the two development leads had particular task to focus on to have the whole project divided, in order to reach the goal for each module in enough detail.

Gopal Morampudi was responsible for designing the user interaction, web interface and implementation of all the three modules Administrator, Tutor and Student.

Rajesh Vallabineni was responsible for Database design, automation testing and automated Deployment for all the three modules Administrator, Tutor and Student. Following the development plan, we were able to combine all the development work to get the project released to the web in its final version.

* 1. Curriculum Scope

STEM Online application project was developed incorporating all the phases of Software Development Life Cycle (SDLC). Each stage of the life cycle right from inception, requirements elicitation, implementation and acceptance, provided learning experiences in both the project manager and developer roles. This experience is really valuable capstone to the master’s program as it required and tested all of the skills of a software engineer.

The following curriculum learning goals were archived throughout this project

* Elicit, document, and analyze the requirements for software systems (GCIS504-

Requirements Engineering)

* Identify, plan, and manage the schedule and risks for the activities involved in software-based

Systems development (GCIS611-Software Project Management)

* Identify effective quality measures and organize activities to support software-based systems

Development (GCIS 515-Software Testing and Quality Assurance).

1. Requirements

The stem center tutor information gathering website project is a solution for limitations in existing system MYWCONLINE. It is comprised of three modules i.e., admin, tutor, student. The requirements, features and stakeholder needs for the project were elicited and then documented. Once documented, analysis was done to determine if they were clear and complete.

## Elicitation:

The requirements were elicited by the development team using two methods.

* Interviews
* Brainstorming

First, the project stakeholder was interviewed. Informal notes were kept and then used to develop the user classes and characteristics for a formal document. Later, in development process team met with stakeholder to know what kind of problems being faced with existing system, and understand more how this project will help admin, tutors and students in scheduling appointments, processing requests and track tutor working hours. These two techniques were particularly useful because they provided an environment where all stakeholder could be involved in decision-making and resulted in requirements documentation that was critical to the development of the rest of the project. Once the interviews and brainstorming sessions were complete and the notes were analyzed, a specific list of requirements and features were compiled.

## Key Design and Implementation Constraints

The application is developed using PHP programming in the front-end and MySQL Database in backend. The Application is developed in such a way that it accesses the STEM database from MySQL Server. The Database design was done as per the requirement of the application. If the application requires any modification in the relational structure of database, then the required changes should be done in the database.

## Operating Environment

As the Application is a Web-application it requires a compatible browser to open the URL of the application i.e. GUI to display the application. As the application contacts MySQL database to access database, a MySQL server should be present at a specific location/server. Commonly the Hosting environment could be a Windows/Linux server and Client interface will be a windows system.

## Interface Requirements

### User Interfaces

User Interface of the application is a browser compatible to open PHP pages and using HeidiSQL to access MySQL.

1. Hardware Interfaces

For Windows (XP/7):

* RAM 128MB
* Processor Pentium IV 2 GHz MHz or faster
* Disk Space 181MB

For Linux:

* RAM 1.5GB
* Processor Pentium IV 2 GHz MHz or faster
* Disk Space 181MB

1. Software Interfaces

* PHP
* JDK
* JRE
* Apache Tomcat
* MySQL 5.0
* Internet Explorer-7
* Google Chrome Browser
* Maven
* Github
* Gitbash
* Jenkins
* PSTools
* Xampp

## Communications Interfaces

Communication to the application happens through HTTP requests over TCP/IP protocol. All the communication protocols applicable to a webpage are applicable here. The MySQL server active port should be open and must be accessible through the browser. The application also uses SMTP to email notifications to the users, if any. We need to install latest versions of Java JDK and JRE and set their path in environmental variables. After installing apache maven we need to set the variable in environmental variables. Jenkins (CI)(2) runs on the port 8080 which is a default port and apache uses the same port to listen, here the conflict arises between the ports in order to run apache server we need to change the port of apache server and restart the server.

**Other Requirements**

* The Database developed should be updated and maintained regularly.
* Certain indexing should be done regularly to provide faster search options

## Test automation framework

Test automation framework is an application that allows to write a series of tests without worrying about the constraints or limitations of the underlying test tools. Test automation framework development is a multi-stage process. And passing through each stage involves multiple challenges to be addressed. Testing framework is always application independent that is it can be used with any application irrespective of the complications of application under test. The framework should be scalable and maintainable.

**Why framework**

A framework defines the way of doing things - a 'Single Standard' approach.

Framework Design - have to choose carefully. Framework design involves identifying requirements from multiple areas.

* Identification of necessary utility/components related to application functionalities
* Types of input data store to be communicated for data flow
* Communication between the utilities/components (for example: data check-point components communicating to the logger.
* Communication between the systems and utility/component development related to the same. (for
* Tool extending capabilities - Developing utilities/components for the validations not supported by the identified test automation tool, if any.

Advantage of Test Automation framework

* Reusability of code
* Maximum coverage
* Recovery scenario
* Low cost maintenance
* Minimal manual intervention
* Easy Reporting

### ****Types of Test Automation Framework****

Now that we have a basic idea of what is an Automation Framework, there are various types of automation frameworks available in the market.

There is a divergent range of Automation Frameworks available now days. These frameworks may differ from each other based on their support to different key factors to do automation like reusability, ease of maintenance etc.

**Few most popularly used Test Automation Frameworks are:**

1. Module Based Testing Framework
2. Library Architecture Testing Framework
3. Data Driven Testing Framework
4. Keyword Driven Testing Framework
5. Hybrid Testing Framework
6. Behavior Driven Development Framework

## Why Jenkins?

The reason behind choosing Jenkins (2) as a continuous integration tool is from the learning experience which we had in the previous semester in software project management course, where each group containing few students were asked to choose a continuous integration tool of their interest and present it in the class, our team has chosen Jenkins as our continuous integration tool to present. We did research and played with the CI tool and present it in the class, based on the research Jenkins tools has a user-friendly interface, highly customizable, lot of resources and tutorials and it is backed by a very strong and active open source community, which has developed hundreds of useful plugins.

Advantages of Jenkins:

* Jenkins is self-hosted

Jenkins does not offer a hosted solution, instead, the only way to use Jenkins is to install it locally. There are not many tools that could rival Jenkins locally and self-hosting also provides a safer location to store key environment variables.

* Free and open source

Jenkins is a free and open source continuous integration tool. Jenkins' source code is hosted on GitHub.

* Stable release line for users who want less changes

This is called the Jenkins Long-Term Support (LTS) version and helps to provide the most stable and the most assuring version of the Jenkins CI possible. About every three months a version which has been deemed the most reliable by the community is chosen. After this, it's branched, well-tested features are added if they are missing, it's then tested with the new features, bug fixes are done if necessary and then it's released as the official Jenkins LTS version.

Disadvantages of Jenkins:

* Outdated interface

The Jenkins interface seems outdated and clunky by today's standards. It does not follow good design principles, has no whitespace which makes the views feel crowded and confusing.

* High overhead

Unlike some of the simple and hosted alternatives, users need to host and setup Jenkins by themselves. This results in both a high initial setup time as well as time sunk into maintenance over a projects duration.

* Poor documentation

The most useful information found about Jenkins is on StackOverflow and various strange corners of the internet that careful Googling may sometimes find. The actual documentation for Jenkins is poorly organized and extremely vague and outdated. Documentation for plugins, even the plugins most heavily relied upon, is almost non-existent.

* 1. **Summary**

In this chapter we have discussed about the approaches that were followed for requirements gathering, about interface requirements, what is an automation framework, its advantages and types of automation framework available in market, why Jenkins was chosen as a continuous integration tool.

**Summary of this chapter goes here… What should be remembered to read the rest of the paper?**

1. Design

## 3.1 Introduction

Approval was needed to make the requirement finalization in order to proceed for design, coding, testing, deploying and finally implement STEM website for end users. The GUI of STEM website was made user friendly and compatible with all the operating systems (Windows, Mac, Linux…).

### Software Context

Our software attempts to quicken the process of a student finding a course within a department with a list of tutors with their schedule and tutor finding their preferable time slot among different time slots.

### Architecture diagram

The architecture diagram shows the three actors student, tutors and administrator accessing the web application via internet.

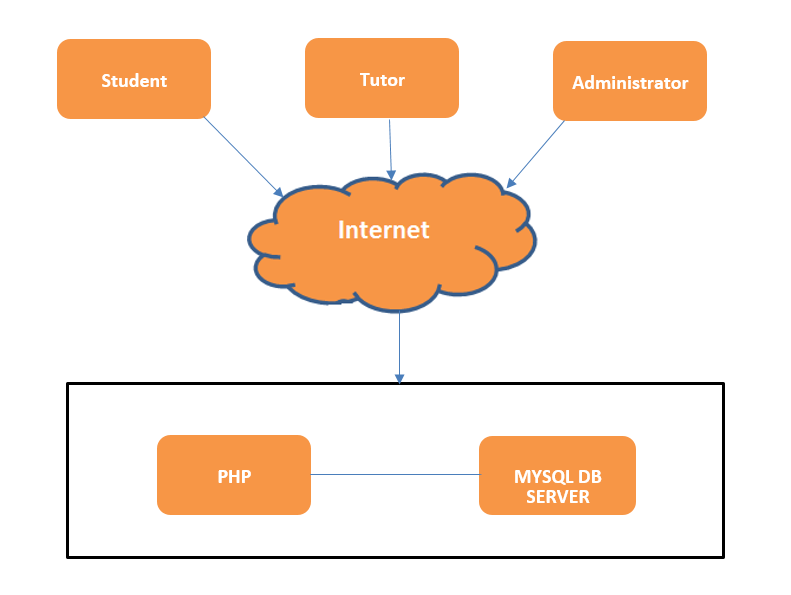


Figure 3: Web Application Architectural diagram

## Data Design

The most important consideration in designing the database is how the information would be used.

* Business activities that will use the database to perform were analyzed.
* Business rules that apply to these activities were also considered.
* Data wanted to maintain in the database, is stored in structured fashion
* Concepts of RDBMS were used during Data Design.
  + 1. Major Internal Software Data Structures

The logic of this application is coded in Scripting pages in PHP. A PHP page is used to extend the capabilities of servers that host applications accessed via a request-response programming model. For this application, PHP page technology defines HTTP-specific PHP pages

* + 1. Database Description
* Create and deploy a database for maintaining the courses of the student for a particular department, time schedule of the tutors and process data needed for supporting the STEM Application.
* Develop and deploy a database-driven website that measurably achieves the goals of the system.
* Design & management of the database
* Admin screens for managing the website *&* CRUD of data*.*

## Why MySQL?

In many ways the database has become a commodity. The products differ on different constraints price, performance, ease of database administration and functionality. Since our project is a web application, we have looked at two options MySQL (10) and PostgreSQL (13). Both MySQL and PostgreSQL are open source relational database management systems. PostgreSQL is highly programmable, and therefore extendible, with custom procedures that are called "stored procedures". These functions can be created to simplify the execution of repeated, complex and often required database operations.

Although this DBMS does not have the popularity of MySQL, there are many amazing third-party tools and libraries that are designed to make working with PostgreSQL simple, despite this database's powerful nature. Nowadays it is possible to get PostgreSQL as an application package through many operating-system's default package manager with ease. For simple operations, PostgreSQL might appear less performant than MySQL, it lacks behind the terms of popularity which might affect how easy it might be possible to get support.

MySQL is the most popular one of all the large-scale database servers. It is a feature rich, open-source product that powers a lot of web-sites and applications online. Getting started with MySQL is relatively easy and developers have access to a massive array of information regarding the database on the internet.

Given the popularity of the product, there are a lot of third-party applications, tools and integrated libraries which help greatly with many aspects of working with this RDBMS. MySQL's security features provide reliable protection for data-access in a simple way. Majority of websites can simply and easily work on MySQL which is flexible and scalable tool is easy to use and easy to manage.

## E-R Diagram

Following Database design script was used to create schemas based on data model.

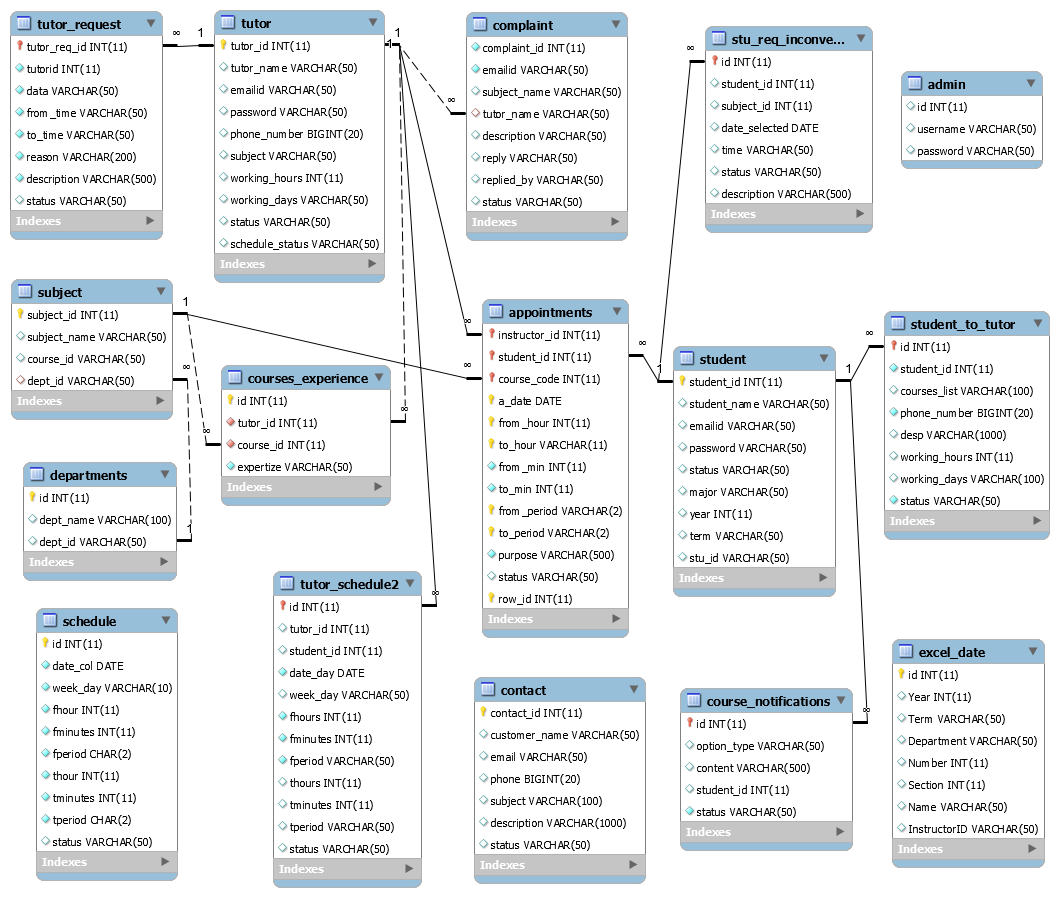


Figure 4: ER-Diagram

The database is the core of the Stem project. A detailed data model of the database was essential for creating a logical structure that could be used for the implementation of the code. Outlining the logical structure visually helped identify data that may otherwise have been missed. Figure 4 shows the data model of the Stem project which consists of the data grouped elements required for the system. It consists of data related to tutor, admin, student, departments, contact, schedule, subject, tutor request, complaint, student request, courses experience, appointments, student to tutor, tutor schedule, course notification and excel date.

* 1. **Summary**

Proper database design is the only way your application will be efficient, flexible, and easy to manage and maintain. An important aspect of database design is to use relationships between tables instead of throwing all your data into one long flat file. MySQL and PostgreSQL have been reviewed based on the requirements of our project, major difference between the two databases is PostgreSQL doesn’t support replication, whereas MySQL does. We picked MySQL as our database based on comprehensive application development it provides comprehensive support for every application development need, Management ease and support.

1. Deployment and Hosting

## Project Delivery Infrastructure

Project files are added to the Github (8) repository and it is integrated to Continuous Integration tool *Jenkins (2).* We are using *Jenkins* to deploy the project, a job in Jenkins is configured that when a commit is made in a Github (2) it will trigger Jenkins and deploy the file into a remote server.

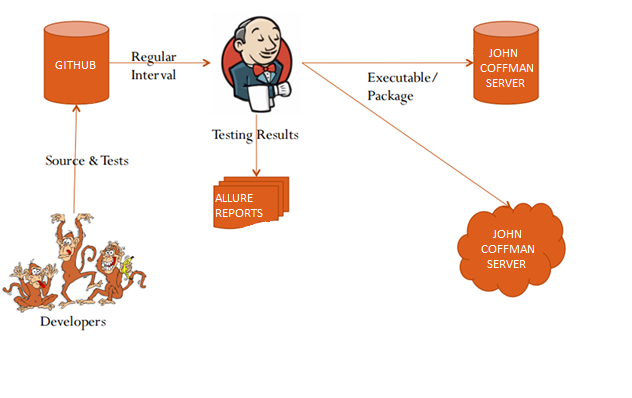


Figure : Jenkins fitting

## Source Code Management

The application is setup in source control management repository of Github to maintain version control and team collaboration. The lifecycle deployment was maintained in such a way that whenever a check-in happens in this repository, a job will be triggered through Jenkins (CI) to conditionally deploy the application on the hosting server.

Following image shows the application in Git repository.

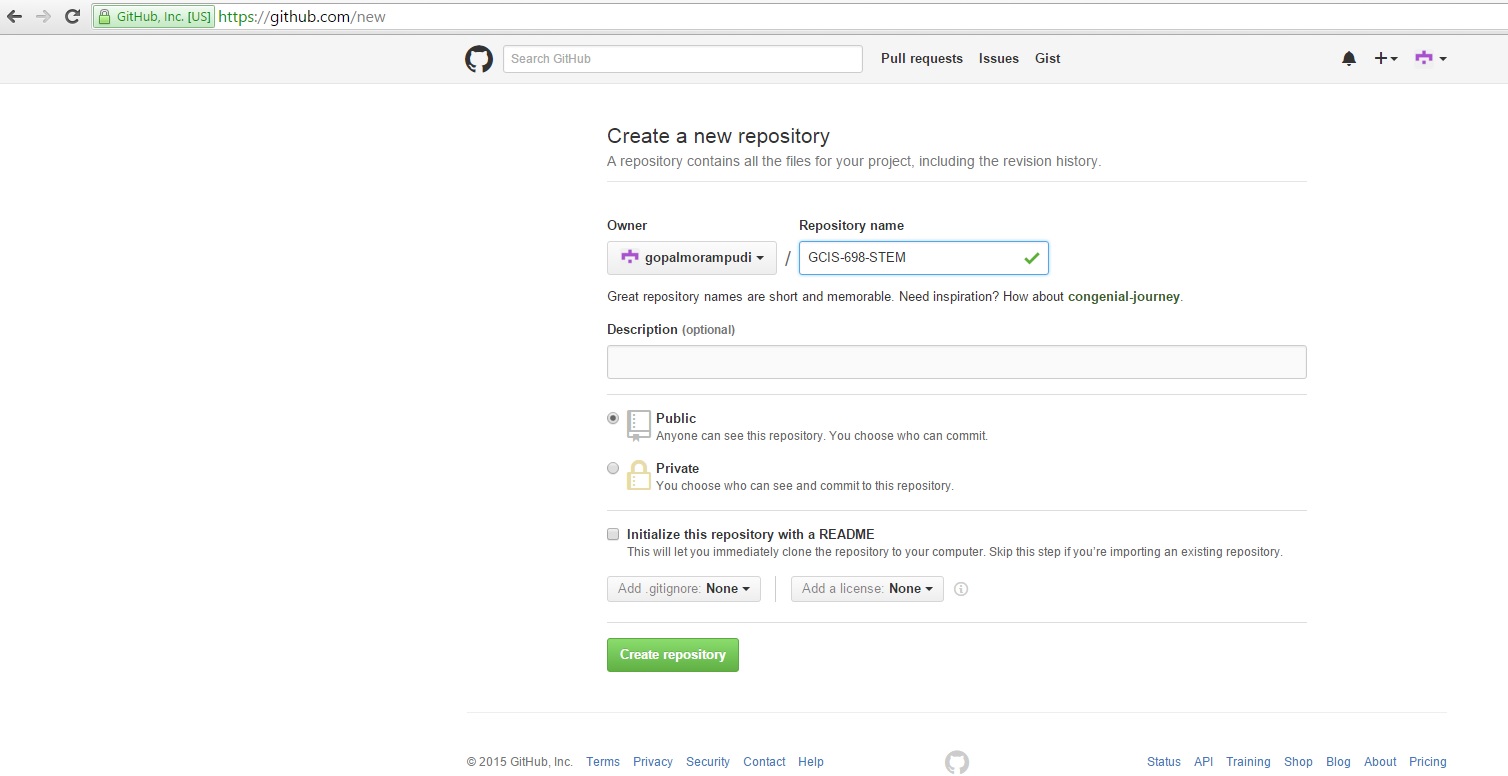


Figure 6 : Create a new repository

We need to sign-up and create a Github account In-order to create a repository and we named our repository as GCIS-698-STEM.

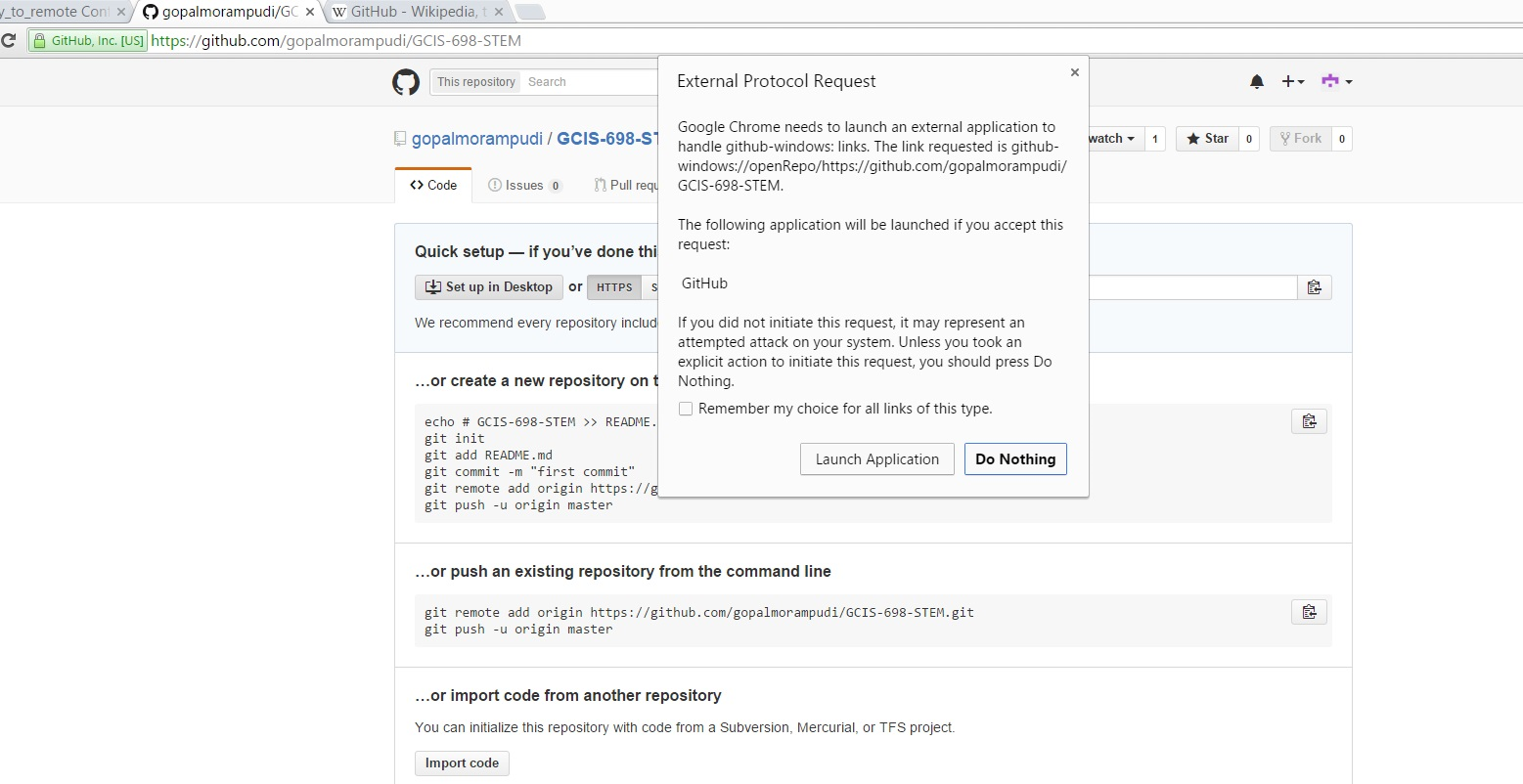


Figure : Repository file system

We need to create a local repository in our local machine in order to publish the code or commit the changes to that code. There is an option in the Github repository page to set up in desktop click it and it will launch a Github windows application as seen in the screen shot above.

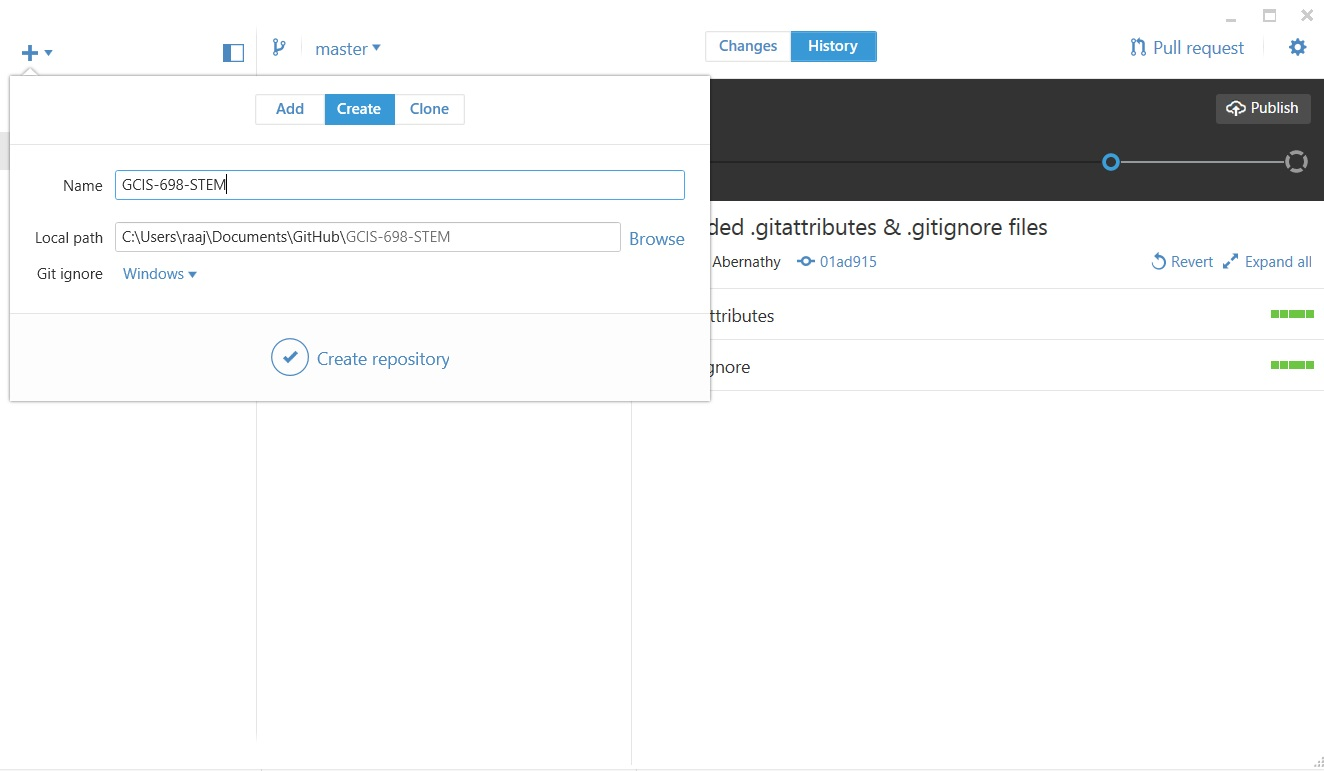


Figure 8 : Work space in local system

It will create a workspace in your local machine or we can use the clone option to create a clone of a repository from your Github account.

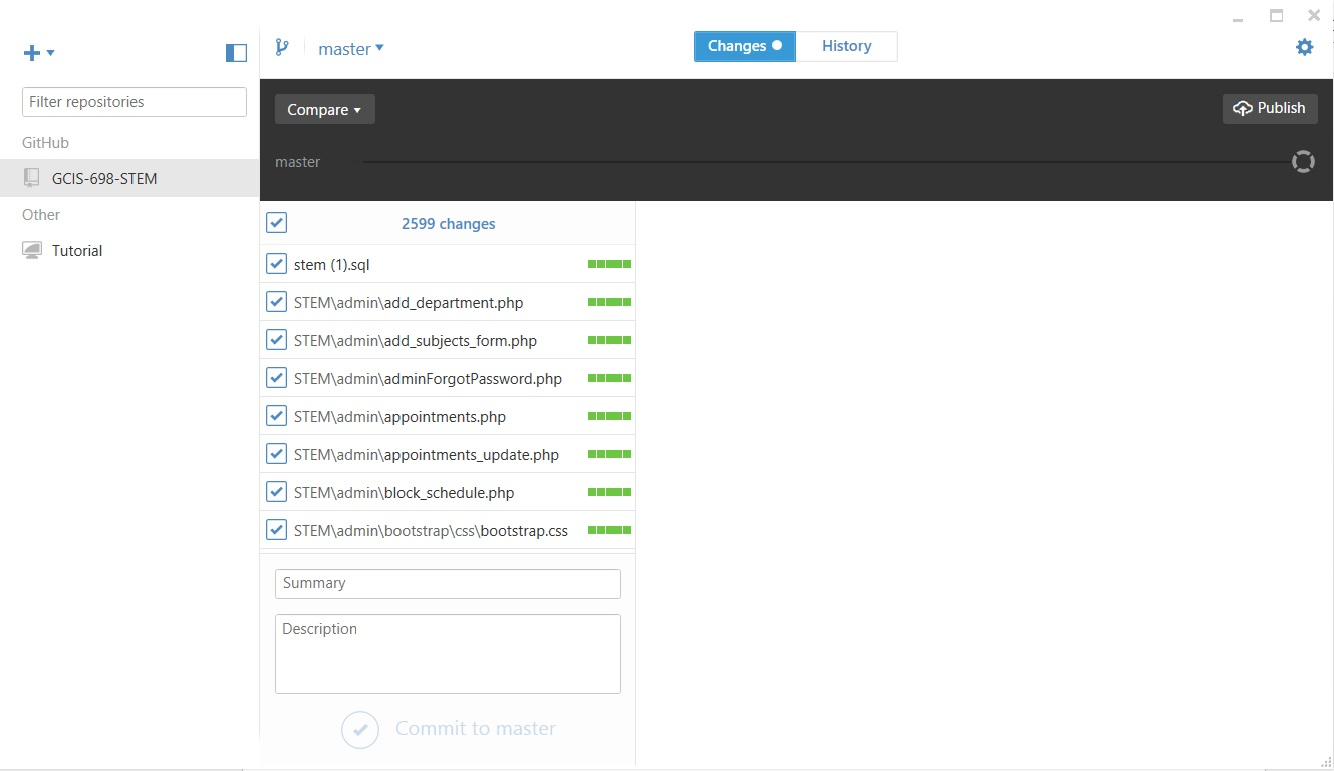


Figure 9: Add files to repository

Now drag and drop the program files in the repository and commit to master then publish master to your Github remote account.

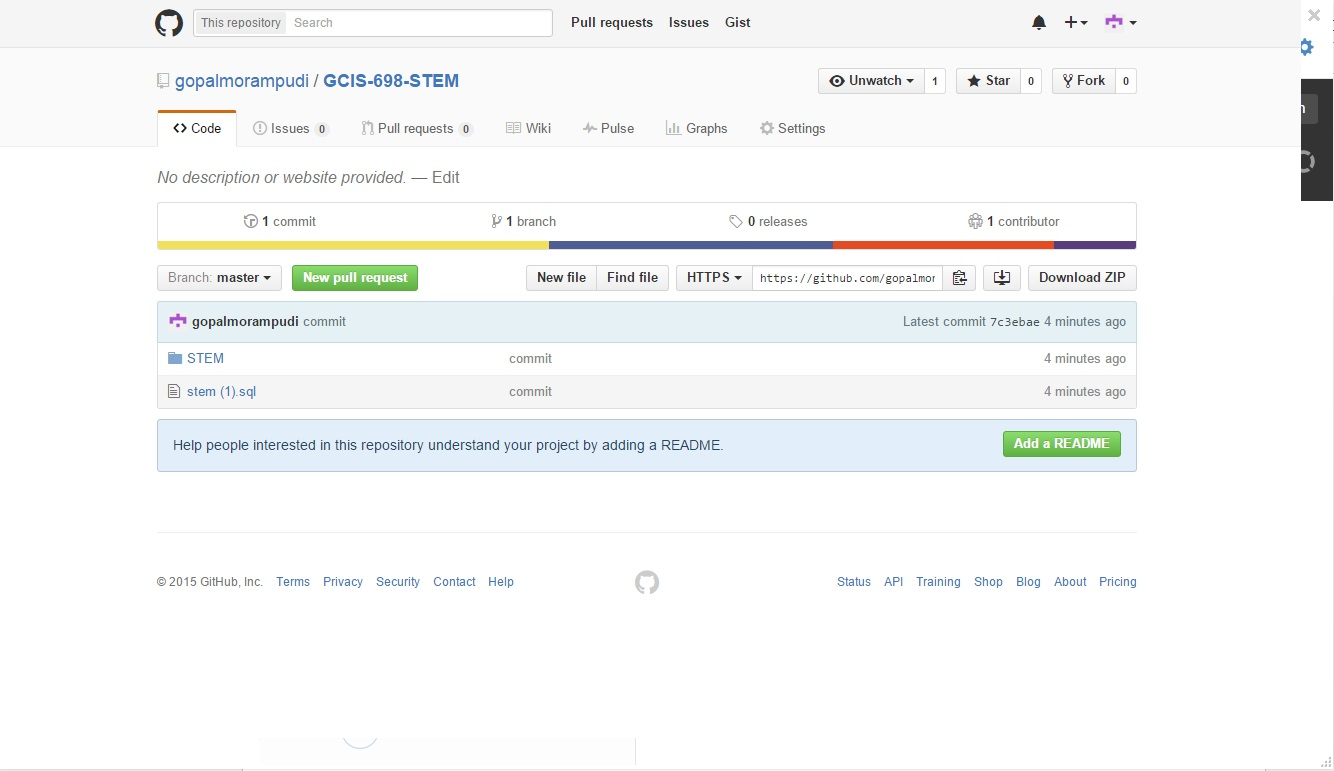


Figure : Commit files to remote Github repository

Now we can find the files in remote Github repository, we can edit the files or change the code in Github and publish it to the master and can be used for build and deployment

## Continuous Integration tool

To build and deploy the website application I am using Jenkins (CI) (5) a continuous integration tool

CI workflow of the application deployment was done with the help of Jenkins server. It has various plugins which contact with other tools like Git (9) and Maven (6) to deploy the application at the required location on the server. Jobs are configured in the Jenkins page and are linked with each other for continuous building.

* + 1. Installation and configuration of Jenkins

Download Jenkins [1] and choose windows as operating system and initiate the installation process from the downloaded file. In-order to run Jenkins successfully we need to install Tomcat server and latest version of Java Development kit and need to set the path in environment variables. After installation of Jenkins we need to install plugins, in-order to install plugins go to manage Jenkins and select manage plugins and choose the plugins to successfully run the job which contact with other tools like Git, Maven, PSTools.

We need to configure Jenkins, go to manage Jenkins and select configure system go to GIT and name it and give the path to Git (9) executable git.exe, which you can find in bin folder of your local Git installation folder and set the path to git.exe. For maven job got to Maven, maven installations and name it and choose select install automatically or you can select install from apache and version or can be installed from running commands, which it will install automatically while building a maven job. Similarly we can configure any kind of job based on the requirements to execute the job.

* + 1. Create a job

Select new item and it will ask you what kind of project you are going to use Jenkins for, I am going to select free-style project, Jenkins will build your project, combining any SCM with any build system, and this can

Even used for something other than software build. In-order to trigger a maven job choose a maven project

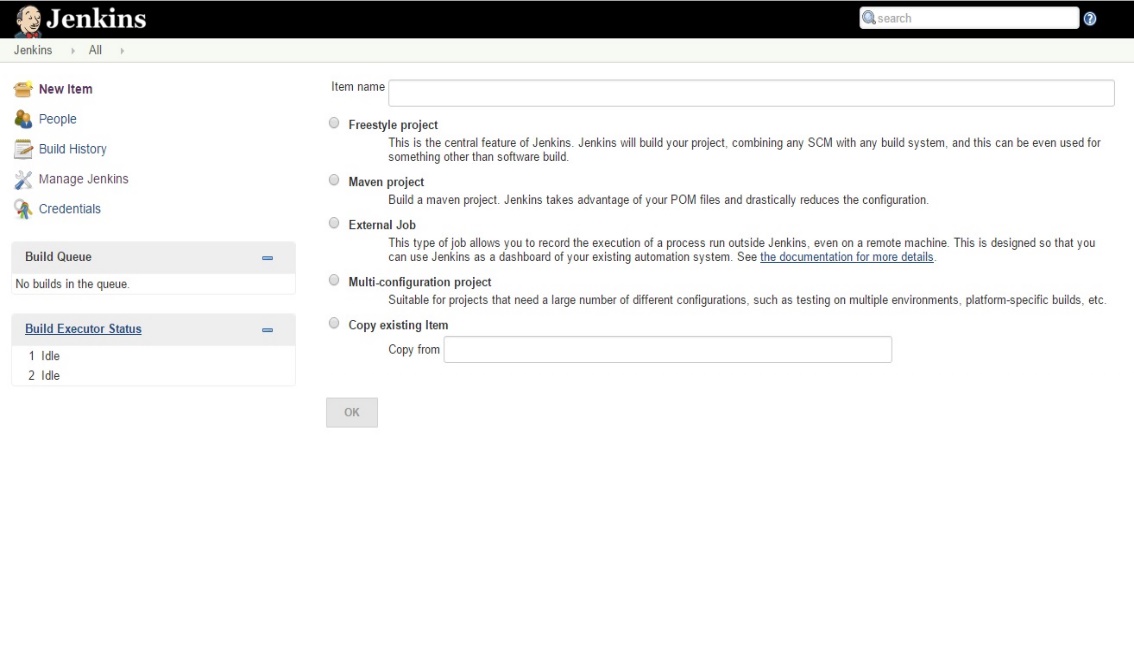
****For which, Jenkins takes advantage of POM files and drastically reduces the configuration. I have created a simple build job used a repository from my Github and triggered build when a change is pushed to Github and Poll SCM to every minute for which it will check and build for every minute.

Figure 11: Creation of Jenkins job

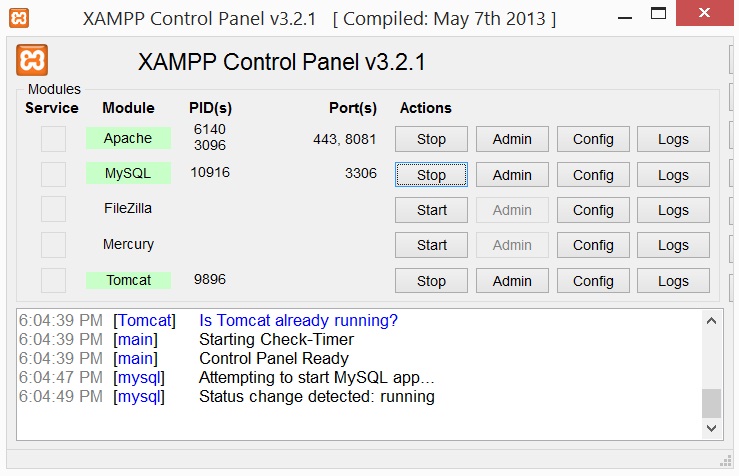


Figure 12: XAMPP Control Panel

## Deployment

The application after performing the necessary compilation should be hosted on to a web server. To install and maintain this server we make use of Xampp (7) tool which starts and stops the apache web server and MySQL server from its control panel console. Also we use another tool named Heidi SQL to view the database in MySQL server.

One goal of this project was to automate the whole process of deploying a job using automation tools on to a remote server. Whenever a new commit is made to Github where our project application resides, every minute Jenkins triggers the initial job thereby triggering other jobs too. This continuous flow provides a kind of automation to the project deployment. Care should be taken while migration of the project from one environment to another.

The deploying process has been tested by using these tools SSH, PSTOOLS, NETUSE.

SSH (4) is an acronym for secure shell it provides the best security when accessing the computer remotely.

To use SSH on windows, we have to download SSH client. One of the freely available clients is PuTTY.

After installing PuTTY we will be able to see a configuration window, where we can setup a new session, including the hostname to connect to and options for SSH port and what kind of protocol to be used and several other categories after entering the host name it will initiate connection to the remote server. This where the SSH comes in to play, common use of SSH is to transfer files, SSH is most commonly used for Linux machines.

PSTOOLS: PsTools (3) contains set of utilities to perform all sorts of administration tasks both locally, and on remote computers as well. PsExec is a light-weight telnet-replacement that lets you execute processes on other systems, below is the usage syntax for psexec.

psexec [\\computer[,computer2[,...] | @file]][-u user [-p psswd][-n s][-r servicename][-h][-l][-s|-e][-x][-i [session]][-c [-f|-v]][-w directory][-d][-<priority>][-a n,n,...] cmd [arguments]

Netuse:

Connects a computer to or disconnects a computer from a shared resource, or displays information about computer connections.

Syntax

net use [{DeviceName | \*}] [\\ComputerName\ShareName[\volume]] [{Password | \*}]] [/user: [DomainName\] UserName] [/user:[DottedDomainName\]UserName] [/user: [UserName@DottedDomainName] [/savecred] [/smartcard] [{/delete | /persistent:{yes | no}}]

Net use [DeviceName [/home [{Password | \*}] [/delete: {yes | no}]]

Net use [/persistent :{ yes | no}]

Net Use (1) command is used to deploy to an application server to map drives and copy from the workspace of Jenkins job to the htdocs of remote server from there we can be able to host the application. By using Netuse command we can connect to the remote server and while PSTOOLS and SSH tools have to install.

## Build

Jenkins can be used to perform the typical build server work, such as doing continuous/official/nightly builds, run tests, or perform some repetitive batch tasks. This is called "free-style software project" in Jenkins.

Setting up the project:

Go to Jenkins top page, select "New Job", then choose "Build a free-style software project". This job type consists of the following elements:

* Optional SCM, such as CVS or Subversion where your source code resides.
* Optional triggers to control when Jenkins will perform builds.
* some sort of build script that performs the build (ant, maven, shell script, batch file, etc.) where the real work happens
* Optional steps to collect information out of the build, such as archiving the artifacts and/or recording Javadoc and test results.
* Optional steps to notify other people/systems with the build result, such as sending e-mails, IMs, updating issue tracker, etc.

**Shell Scripts and Windows Batch Commands:**

If you're using a shell script to do your build, you can either put these environment variables directly into your shell scripts, or call them as parameters in your shell script. Below is an example how this can be done:



Figure Execute Shell

If you are executing a Windows Batch Command, the variables should be referenced using the %VARIABLE\_NAME% pattern. For example:

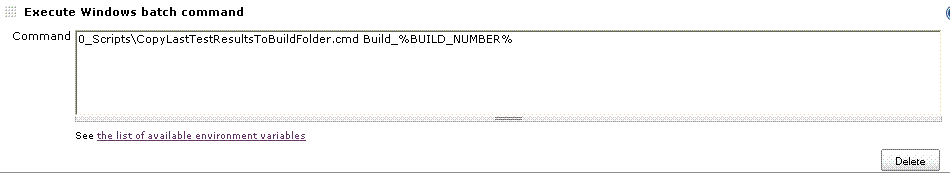


Figure Execute Windows batch command

**Configuring automatic builds:**

Builds in Jenkins can be triggered periodically (on a schedule, specified in configuration), or when source changes in the project have been detected, or they can be automatically triggered by requesting the URL:

<http://YOURHOST/jenkins/job/PROJECTNAME/build>

**Builds by e-mail (send mail):**

By having the root account of the system, the easiest to tweak /etc/aliases and add the following entry:

Jenkins-foo: "|/bin/wget -o /dev/null http://YOURHOST/jenkins/job/PROJECTNAME/build"

And then run "new aliases" command to let send mail know of the change. Whenever someone sends an e-mail to "jenkins-foo@yoursystem", this will trigger a new build.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Build 1** |  | 2.5 sec |  |
|  | **Build 2** |  | 5 sec |  |
|  | **Build 3** |  | 7.2 sec |  |
|  | **Build 4** |  | 29 sec |  |
|  | **Build 5** |  | 15 sec |  |
|  | **Build 6** |  | 23 min |  |
|  | **Build 7** |  | 2.9 sec |  |
|  | **Build 8** |  | 36 sec |  |

Table Build Time Trend

Table 3 shows the buld time trend for a sample job where each build that were triggered in jenkins

### Summary

In this chapter we have discussed about project delivery infrastructure consists of Github repository where source code is maintained, how to build a job, about automation testing and test reports discussed in section 5, Jenkins tools which is used for automation deployment along with tools used to execute the commands.

1. Testing

## Type of Frame work:

* Our Test automation frame work is Hybrid in nature, it is the combination of Modular and Data driven frameworks, which was entirely developed on JAVA.
* Our test automation framework developed in such a way that, we can run the developed automated selenium tests in all selenium supported browsers without changing the code.

## Unit Testing Framework:

Testing is the most powerful tool we know of to improve software quality. Tests reduce bugs, provide accurate documentation, and improve design.

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently tested for proper operation. Unit testing involves only characteristics that are more important to the performance of the unit under test, which will leaves a way to modify the source code without concerns which might the functioning of other units or the whole program. Unit testing is a time consuming process, well documentation must be maintained.

* We have been using TestNG (11) unit testing frame work for writing, running and evaluating the automated tests.
* TestNG is a testing framework where NG means next generation.

## Exception Handling and Synchronization strategy:

* Our automation frame work is having, strongly developed exception handling mechanism ensuring that the developed automated tests run smoothly at runtime.
* Our automation frame work contains better synchronization mechanism (timing) to deal with handling scenarios like AJAX and slow loadable components. We’ve used ‘Web Driver’ explicit waits, implicit waits, page load timeouts and developed java script functions (in java) to achieve synchronization in tests.

1. Build Tools:

* In our automation frame work, we have been using Maven (8) as build management tool we are using these tools for triggering the automation execution.

## Page Object Design pattern:

We are following the popular Page Object Design pattern to represent the application under test which would be very helpful for maintaining the developed automation tests.

Following are the main components of our Test automation Frame work:

**Driver manager**:

In order to execute the automated tests, Driver manager is the one which responsible running the tests in linear mode in the specified browser. It is responsible for creating the driver object and registers the created driver with the Web Driver Event tracker.

**Root Test:**

‘Root Test’ is the Parent class for the all the test classes. It is responsible for capturing the driver object from ‘Driver Manger’ and caters the driver object to the child test classes.

**Locators/Object repository:**

We have been using external Property files for storing the html element locators (XPATH/CSS selectors).Based on the AUT web pages we will create separate property files( with locators) for all the required pages. We will parse these files at the time of Page objects development.

This approach eliminates the hard coding of locators. In case if any locator on a page changes then simply we amend the corresponding locator in the corresponding property file so we no need to change the Java code.

**User Actions:**

All the user actions (Click, Type, Select ....etc.) that can be performed over on web page was developed be as part of User actions all these actions are perfectly synchronized and surrounded by strong exception handling mechanism.

**Web Driver Events tracker:**

This responsible for tracking the events (Click, Type, Select, Navigates to ....etc) generated by the ‘Web Driver’. It is also responsible for logging all the events in the log file; these things were very helpful while debugging the failed automation tests.

**Test results Reporters:**

For test results reporting purposes, we are using following 2 popular reporting tools: ReportNG and Allure. These two reporters display beautiful and easily understandable test results reports.

**Logger:**

We are using popular Log4J as logger for automation tests and we are using HTML pattern layout for logging file, the generated log files will be used for debugging failed tests

**Customized Test Listeners:**

We have developed customized TestNG listeners. These listeners are responsible for recording videos/taking screen shots (on failure) and attaching the recorded videos and screen shots to corresponding tests to the generated reports.

**Build tools:**

We are using Maven as Dependency, build management tool and it is also responsible for triggering the tests execution.

## Execution flow of the test cases

**Page Objects and corresponding Page functionalities:**

Initially we will develop the required Page objects for AUT (Application under Test). In these page objects we will write separate functions (java methods) for each and every functional flow that can be performed on the specific page.

These Page Objects will use the Locators component for identifying HTML elements on the web page

These page Objects functions (java methods) will consume the User Action component to perform the actions (Click, Type, Select ....etc.) on the web page.

**Test Classes with Test cases:**

We will create the Test classes, all these Test classes are Childs of RootTest and using TestNG unit testing framework, we create TestNG @Test method for each and every test Case

**STEMng.xml:**

**Parameters:**

Browser name (Chrome/Firefox/IE ...): We Use this parameter for running automated tests on Desired Web browsers (supported browsers by selenium).

**Test Classes /Test methods**:

Here we will list the test classes or methods which we need to run

**Listeners:**

We will include the customized test listeners and other required test listeners.

## Execution process:

1. STEMng.xml file contains the test suite with the list automation test classes and also it contains execution mode and browser name.

2. Build tools invokes the TestNG with the help of respective plugins (Maven uses sure fire plug-in)

3. TestNG, test runner Calls the RootTest. Based on the execution mode Parameter (present in STEMng.xml file) Root Test communicates with either Driver Manager. Based on the Browser name Parameter Driver Manager creates driver object and returns it to RootTest.

4. All the child test classes of RootTest will get the driver object from RootTest and all @Test method test cases will be executed by contacting corresponding page objects.

5. Using TestNG asserts results of test case (@Test method) will be evaluated

6. Logger logs desired things in the Log file and ‘Web Driver Event Tracker’ tracks the all Web Driver related events and logs the events in the file

7. After completion of test execution, Test Results Reporter will be called by Ant or Maven these reporters will gather the results and generates the consolidated reports.

## Configured Allure Reporting tool for projects:

I have configured allure test reports tool to our project

Besides ReportNG and Allure (12) provides clear reports to us. It will automatically generate charts (Bar, Pie charts) to us and it can show each test in timeline view.

Apart from these by using allure we can create self-documented tests and the test results and failure screenshots can be intertwined.

### Summary

In this chapter we have discussed about the whole testing process approached includes type of framework and how it was designed, page object design pattern, execution process and allure reporting tool which is a flexible light weight and multi-language test report tool generates charts automatically.

1. Conclusion

The main goal of this project is to design an efficient database and automate the testing and deployment process.

The web application has been successfully automated and deployed to a hosting server via Jenkins, where web application release should be a fast and repeatable process. Automation testing was successfully done

By automation framework TestNG and the Allure provides clear reports to us. It will automatically generate charts (Bar, Pie charts) to us and it can show each test in timeline view.

All of the objectives for the project were successfully fulfilled. Additionally, the learning experiences, team collaboration, and the project management were extraordinarily valuable lessons that can be used going forward.

Much was gained from working on the project: new tools such as Jenkins and automation framework, tools used to execute windows commands; project management was done with a real team.

As a whole, the success of this project can best be measured in all of the experiences that were had and knowledge gained.

**Measures of success**

* Automation Deployment without a programmer intervention for minimal edits.
* Cycle time- Time from deciding that you need to make a change to having it in production.

# References

|  |  |
| --- | --- |
| [1] | "netuse," [Online]. Available: https://technet.microsoft.com/en-us/library/bb490717.aspx. |
| [2] | [Online]. Available: https://jenkins-ci.org/. |
| [3] | "technet microsoft," 2015. [Online]. Available: https://technet.microsoft.com/en-us/sysinternals/psexec.aspx. |
| [4] | "SSH tutorial for windows," [Online]. Available: http://support.suso.com/supki/SSH\_Tutorial\_for\_Windows. |
| [5] | "CI-Jenkins," in *Jenkins: The Definitive Guide*, O`Reily Media Inc, july 12, 2011. |
| [6] | "Maven," 2015. [Online]. Available: https://maven.apache.org/what-is-maven.html. |
| [7] | "Apache Friends," [Online]. Available: https://www.apachefriends.org/index.html. |
| [8] | 2016. [Online]. Available: https://github.com/. |
| [9] | "GIT," [Online]. Available: http://www.git-scm.com/. |
| [10] | E. Sverdlov, "A BASIC MYSQL TUTORIAL," [Online]. Available: https://www.digitalocean.com/community/tutorials/a-basic-mysql-tutorial. |
| [11] | "How TestNG makes Selenium tests easier," [Online]. Available: http://www.guru99.com/all-about-testng-and-selenium.html. |
| [12] | "Allure reports," [Online]. Available: http://allure.qatools.ru/. |
| [13] | "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems," [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. |

Appendix A: Glossary

API : Application Programming Interface

: Set of routines, protocols, and tools for building software applications

CI : Continuous Integration

: Automation of different tasks

CIS : Computer and Information Science

: Department conferring graduate degree

CRUD : Create, Read, Update and Delete

: Operations on a record in a database

CSS : Cascaded Style Sheet

: A file used to specify the UI styles for the elements inside a web page

ER : Entity-Relation Model

: A data model for describing the data or information aspects of a business domain

GUI : Graphical User Interface

: A type of interface that allows users to interact with electronic devices through graphical icons and visual indicators

HTTP : Hyper Text Transfer Protocol

: A type of internet protocol

PHP : Personal Home Page/ Hypertext Preprocessor

: A server-side scripting language

MySQL : My Structured Query Language

: An open-source relational database management system (RDBMS)

RAM : Random Access Memory

: A form of computer data storage

RDBMS : Relational Database Management System

: A type of database management system

SCM : Source Control Management

: A program used to maintain files and documents, etc.

SDLC : Software Development Life Cycle

: Different stages/phases during development of software.

STEM : Science Technology Engineering and Math

SMTP : Simple Mail Transfer Protocol

: An Internet standard for electronic mail (email) transmission

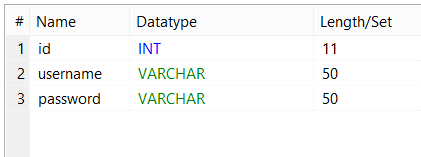
TCP/IP : Transmission Control Protocol / Internet Protocol

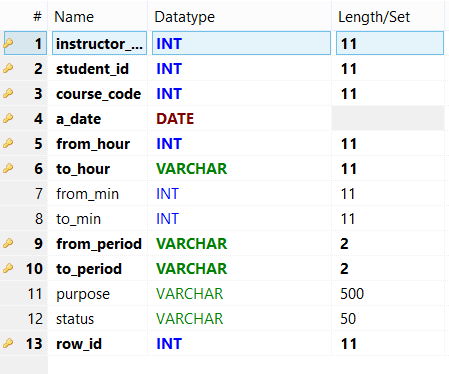
: Communication protocol

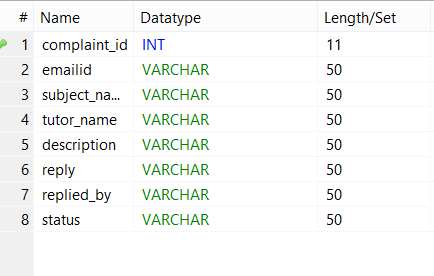
UI : User Interface

: It is the human interface with electronic devices

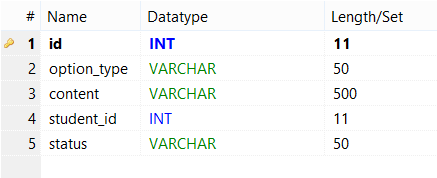
Appendix D: Design Models

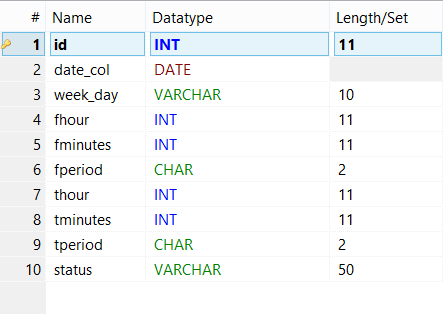
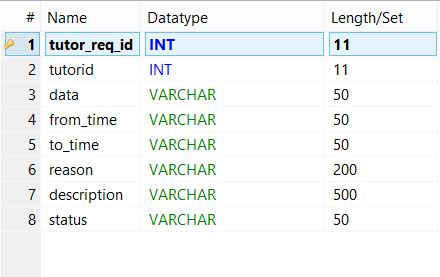
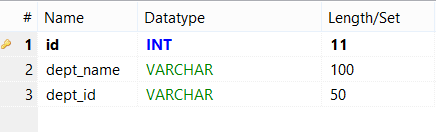
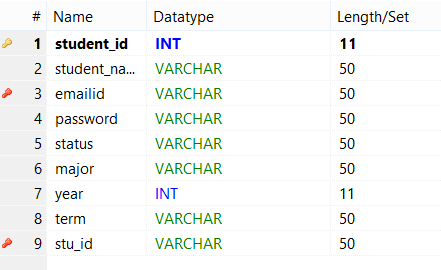
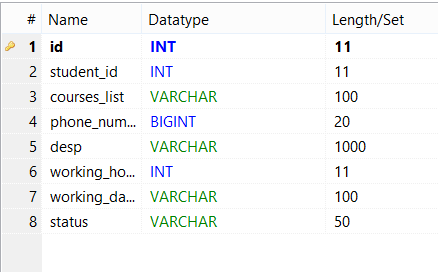


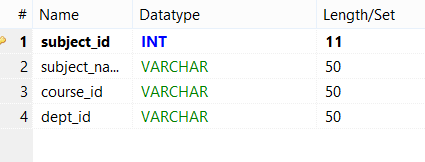
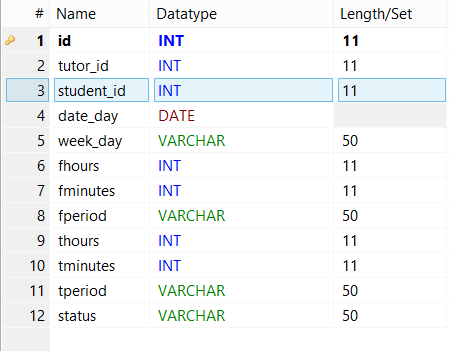


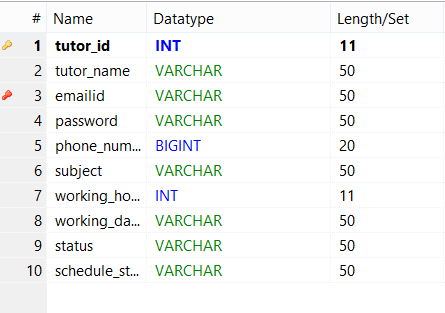
**

**









Appendix E: Testing Log and Summary Status

package com.diw.test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import ru.yandex.qatools.allure.annotations.Step;

import com.diw.main.RootTest;

import com.diw.page.AdminLoggedinPage;

import com.diw.page.AdminLoginPage;

import com.diw.page.StudentLogin;

public class AdminLoginPageTest extends RootTest {

private WebDriver driver;

private static String url = "http://localhost/STEM/admin";

@BeforeClass

public void setDriver(){

driver=super.getDriver();

driver.get(url);

driver.manage().deleteAllCookies();

}

@Parameters({ "emailadmin", "pwdadmin" })

@Step("Enter user name,pwd and click on sigin btn & verify the signin ")

@Test

public void checkAdminLogin(String email\_id, String pwd){

AdminLoginPage.doAdminLogin(driver, email\_id, pwd);

Assert.assertTrue(driver.getTitle().contains("Admin"));

}

@Parameters({ "courseid","coursename"})

@Step("Add course")

@Test(dependsOnMethods = {"checkAdminLogin"})

public void checkAddCourse(String courseid, String coursename){

AdminLoggedinPage.addCourse(driver, courseid,coursename );

Assert.assertTrue(driver.findElement(

By.xpath("//b[contains(text(),'All Courses List')]")).isDisplayed());

}

@Step("view course")

@Test(dependsOnMethods = {"checkAddCourse"})

public void checkViewCourse(){

AdminLoggedinPage.viewCourse(driver);

System.out.println("viewed courses");

}

@Step("View Tutors")

@Test(dependsOnMethods = {"checktutorrequest"})

public void viewtutorstest(){

AdminLoggedinPage.viewalltutors(driver);

System.out.println("view all tutors verified");

}

@Step("tutor request")

@Test(dependsOnMethods = {"checkViewCourse"})

public void checktutorrequest(){

AdminLoggedinPage.tutorRequest(driver);

System.out.println("tutor request verified");

}

@Parameters({ "deptID","deptName"})

@Step("add departments")

@Test(dependsOnMethods = {"checktutorrequest"})

public void addDepartmentsTest(String deptID, String deptName){

AdminLoggedinPage.addDepartment(driver, deptID, deptName);

System.out.println("Added department verified");

}

@Step("view department")

@Test(dependsOnMethods = {"addDepartmentsTest"})

public void checkviewdepartments(){

AdminLoggedinPage.viewDepartment(driver);

System.out.println("view departments verified");

}

@Step("Scheduled courses")

@Test(dependsOnMethods = {"makesceduletest"})

public void viewschedule(){

AdminLoggedinPage.viewschedule(driver);

System.out.println("viewed schedule");

}

@Step("make schedule")

@Test(dependsOnMethods = {"checkviewdepartments"})

public void makesceduletest(){

AdminLoggedinPage.makescedule(driver);

System.out.println("scheduled");

}

@Step("view students")

@Test(dependsOnMethods = {"viewschedule"})

public void viewallstudentstest(){

AdminLoggedinPage.viewallStudents(driver);

System.out.println("view all students verified");

}

@Step("tutor notifications")

@Test(dependsOnMethods = {"viewallstudentstest"})

public void tutorNotificationcheck(){

AdminLoggedinPage.tutorNotifications(driver);

System.out.println("tutor notifications verified");

}

@Step("Student As Tutor")

@Test(dependsOnMethods = {"tutorNotificationcheck"})

public void checkStudentAsTutor(){

AdminLoggedinPage.studentAsTutor(driver);

System.out.println("student request data verified");

}

@Step("New Student Notification")

@Test(dependsOnMethods = {"checkStudentAsTutor"})

public void checkNewStudentNotification(){

AdminLoggedinPage.newStudentNotifications(driver);

System.out.println("New Student Notifications verified");

}

@Step("Admin Logout")

@Test(dependsOnMethods = {"checkNewStudentNotification"})

public void adminLogoutTest(){

AdminLoggedinPage.adminLogout(driver);

System.out.println("Admin logout verified");

}

}

package com.diw.test;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import ru.yandex.qatools.allure.annotations.Step;

import com.diw.main.RootTest;

import com.diw.page.TutorLoginPage;

import com.diw.page.TutorProfile;

public class TutorLoginPageTest extends RootTest{

private WebDriver driver;

private static String url = "http://localhost/STEM/login.html";

@BeforeClass

public void setDriver(){

driver = super.getDriver();driver.get(url);

driver.manage().deleteAllCookies();

}

@Parameters({ "emailt", "pwdt" })

@Step("Tuorlogin Enter email, password")

@Test

public void checkTutorLogin(String email\_id, String pwd){

TutorLoginPage.doTutorLogin(driver, email\_id, pwd);

Assert.assertTrue(driver.getTitle().contains("Tutor"));

}

@Step("Tutor Appoinment")

@Test(dependsOnMethods = {"checkTutorLogin"})

public void checkAppointment(){

TutorProfile.makeAppointment(driver);

System.out.println("Tutor Appointment Scheduled");

}

@Step("View Appointments")

@Test(dependsOnMethods = {"checkAppointment"})

public void checkViewAppointment(){

TutorProfile.viewAppointments(driver);

System.out.println("Tutor View Appointments verified");

}

@Step("Tutor Logout")

@Test(dependsOnMethods = {"checkTutorLogin"})

public void checkTutorLogout(){

TutorProfile.doTutorLogout(driver);

//Assert.assertTrue(driver.getTitle().contains("Tutor|Login"));

}

}

package com.diw.test;

import org.openqa.selenium.WebDriver;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

import ru.yandex.qatools.allure.annotations.Step;

import com.diw.action.UserActions;

import com.diw.main.RootTest;

import com.diw.page.ContactPage;

import com.diw.page.CoursesPage;

import com.diw.page.StudentRegistration;

public class StudentRegistrationTest extends RootTest{

private WebDriver driver;

private static String url = "http://localhost/STEM/studentregister.php";

@BeforeClass

public void setDriver(){

driver = super.getDriver();

driver.get(url);

driver.manage().deleteAllCookies();

}

String[] a=new String[]{"mark","001","123","123","markb1213@gmail.com"};

@Step("StudentRegistration Testpage")

@Test(description="")

public void checkkmessageContact(){

//CoursesPage.clickOnStudentRegister(driver);

UserActions.waitForPageLoad(driver);

StudentRegistration.doStudentRegistration(driver, a);

}

}

package com.diw.test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import ru.yandex.qatools.allure.annotations.Step;

import com.diw.action.UserActions;

import com.diw.main.RootTest;

import com.diw.page.ContactPage;

import com.diw.page.CoursesPage;

import com.diw.page.StudentLogin;

import com.diw.page.StudentProfile;

public class StudentLoginTest extends RootTest{

private WebDriver driver;

private static String url = "http://localhost/STEM/login\_student.html";

@BeforeClass

public void setDriver(){

driver=super.getDriver();

driver.get(url);

driver.manage().deleteAllCookies();

}

@Parameters({ "email", "pwd" })

@Step("Enter user name,pwd and click on sigin btn & verify the signin ")

@Test

public void checkStudentLogin(String email\_id, String pwd){

StudentLogin.doStudentlogin(driver, email\_id, pwd);

System.out.println("Student Login succes");

Assert.assertTrue(driver.getTitle().contains("Student"));

}

@Parameters({"unavailable"})

@Step("Make Appointment")

@Test(dependsOnMethods = {"checkStudentLogin"})

public void checkMakeAppointment(String unavailable){

StudentProfile.makeAppointment(driver, unavailable);

System.out.println("Make Appointment verified");

}

@Step("Appointments")

@Test(dependsOnMethods = {"checkMakeAppointment"})

public void checkViewAppointments(){

StudentProfile.viewappointments(driver);

System.out.println("ViewAppointments Verified");

}

@Step("Click on Logout")

@Test(dependsOnMethods = {"checkViewAppointments"})

public void studentlogout()

{

StudentProfile.doLogout(driver);

System.out.println("Student Logout success");

Assert.assertTrue(driver.getTitle().contains("Student|Login"));

//Assert.assertEquals("Tutor|Login", driver.getTitle());

}

}

package com.diw.test;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

import ru.yandex.qatools.allure.annotations.Step;

import com.diw.action.UserActions;

import com.diw.main.RootTest;

import com.diw.page.ContactPage;

import com.diw.page.CoursesPage;

public class ContactPageTest extends RootTest {

private WebDriver driver;

private static String url = "http://localhost/STEM/contact.php";

@BeforeClass

public void setDriver(){

driver = super.getDriver();

driver.get(url);

driver.manage().deleteAllCookies();

}

String[] a=new String[]{"mark","markb1213@gmail.com","0123456789","subject","hi"};

@Step("Contact page")

@Test(description="")

public void checkkmessageContact(){

//CoursesPage.clickOnContact(driver);

UserActions.waitForPageLoad(driver);

ContactPage.messageContact(driver, a);

}

}

package com.diw.test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import ru.yandex.qatools.allure.annotations.Step;

import com.diw.main.RootTest;

import com.diw.page.AdminLoggedinPage;

import com.diw.page.AdminLoginPage;

import com.diw.page.StudentLogin;

public class AdminLoginPageTest extends RootTest {

private WebDriver driver;

private static String url = "http://localhost/STEM/admin";

@BeforeClass

public void setDriver(){

driver=super.getDriver();

driver.get(url);

driver.manage().deleteAllCookies();

}

@Parameters({ "emailadmin", "pwdadmin" })

@Step("Enter user name,pwd and click on sigin btn & verify the signin ")

@Test

public void checkAdminLogin(String email\_id, String pwd){

AdminLoginPage.doAdminLogin(driver, email\_id, pwd);

Assert.assertTrue(driver.getTitle().contains("Admin"));

}

@Parameters({ "courseid","coursename"})

@Step("Add course")

@Test(dependsOnMethods = {"checkAdminLogin"})

public void checkAddCourse(String courseid, String coursename){

AdminLoggedinPage.addCourse(driver, courseid,coursename );

Assert.assertTrue(driver.findElement(

By.xpath("//b[contains(text(),'All Courses List')]")).isDisplayed());

}

@Step("view course")

@Test(dependsOnMethods = {"checkAddCourse"})

public void checkViewCourse(){

AdminLoggedinPage.viewCourse(driver);

System.out.println("viewed courses");

}

@Step("View Tutors")

@Test(dependsOnMethods = {"checktutorrequest"})

public void viewtutorstest(){

AdminLoggedinPage.viewalltutors(driver);

System.out.println("view all tutors verified");

}

@Step("tutor request")

@Test(dependsOnMethods = {"checkViewCourse"})

public void checktutorrequest(){

AdminLoggedinPage.tutorRequest(driver);

System.out.println("tutor request verified");

}

@Parameters({ "deptID","deptName"})

@Step("add departments")

@Test(dependsOnMethods = {"checktutorrequest"})

public void addDepartmentsTest(String deptID, String deptName){

AdminLoggedinPage.addDepartment(driver, deptID, deptName);

System.out.println("Added department verified");

}

@Step("view department")

@Test(dependsOnMethods = {"addDepartmentsTest"})

public void checkviewdepartments(){

AdminLoggedinPage.viewDepartment(driver);

System.out.println("view departments verified");

}

@Step("Scheduled courses")

@Test(dependsOnMethods = {"makesceduletest"})

public void viewschedule(){

AdminLoggedinPage.viewschedule(driver);

System.out.println("viewed schedule");

}

@Step("make schedule")

@Test(dependsOnMethods = {"checkviewdepartments"})

public void makesceduletest(){

AdminLoggedinPage.makescedule(driver);

System.out.println("scheduled");

}

@Step("view students")

@Test(dependsOnMethods = {"viewschedule"})

public void viewallstudentstest(){

AdminLoggedinPage.viewallStudents(driver);

System.out.println("view all students verified");

}

@Step("tutor notifications")

@Test(dependsOnMethods = {"viewallstudentstest"})

public void tutorNotificationcheck(){

AdminLoggedinPage.tutorNotifications(driver);

System.out.println("tutor notifications verified");

}

@Step("Student As Tutor")

@Test(dependsOnMethods = {"tutorNotificationcheck"})

public void checkStudentAsTutor(){

AdminLoggedinPage.studentAsTutor(driver);

System.out.println("student request data verified");

}

@Step("New Student Notification")

@Test(dependsOnMethods = {"checkStudentAsTutor"})

public void checkNewStudentNotification(){

AdminLoggedinPage.newStudentNotifications(driver);

System.out.println("New Student Notifications verified");

}

@Step("Admin Logout")

@Test(dependsOnMethods = {"checkNewStudentNotification"})

public void adminLogoutTest(){

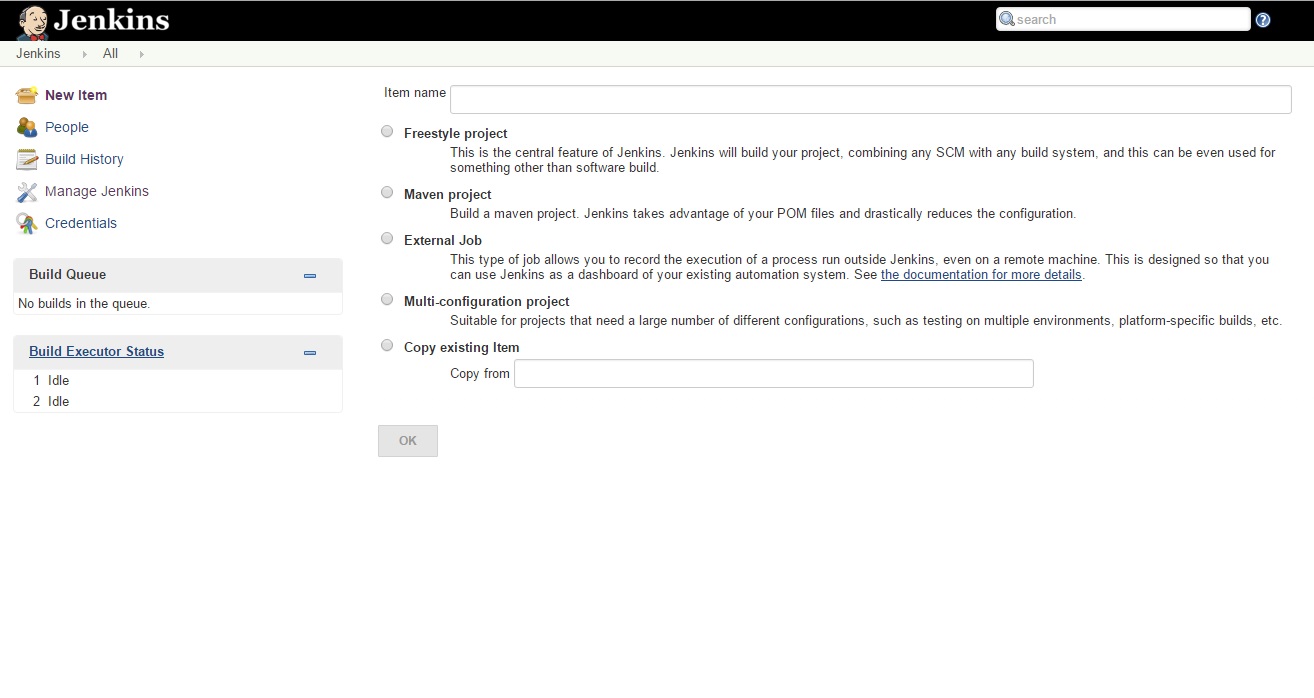
AdminLoggedinPage.adminLogout(driver);

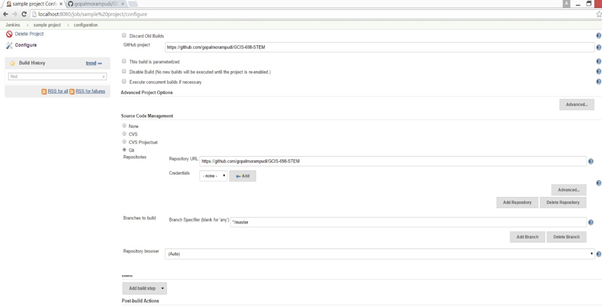
System.out.println("Admin logout verified");

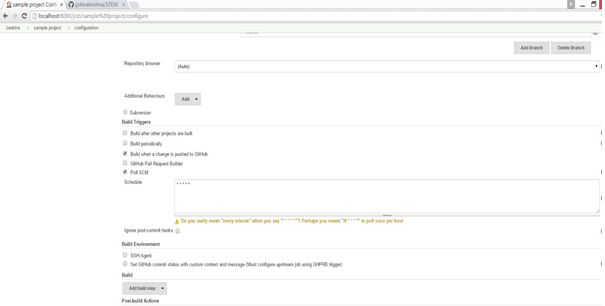
}

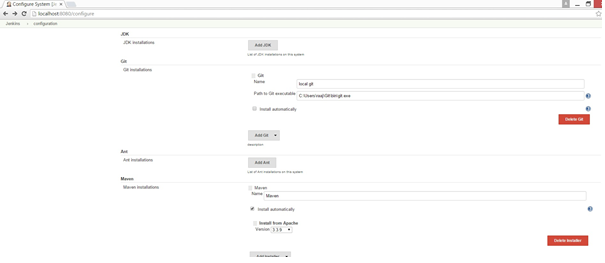
}

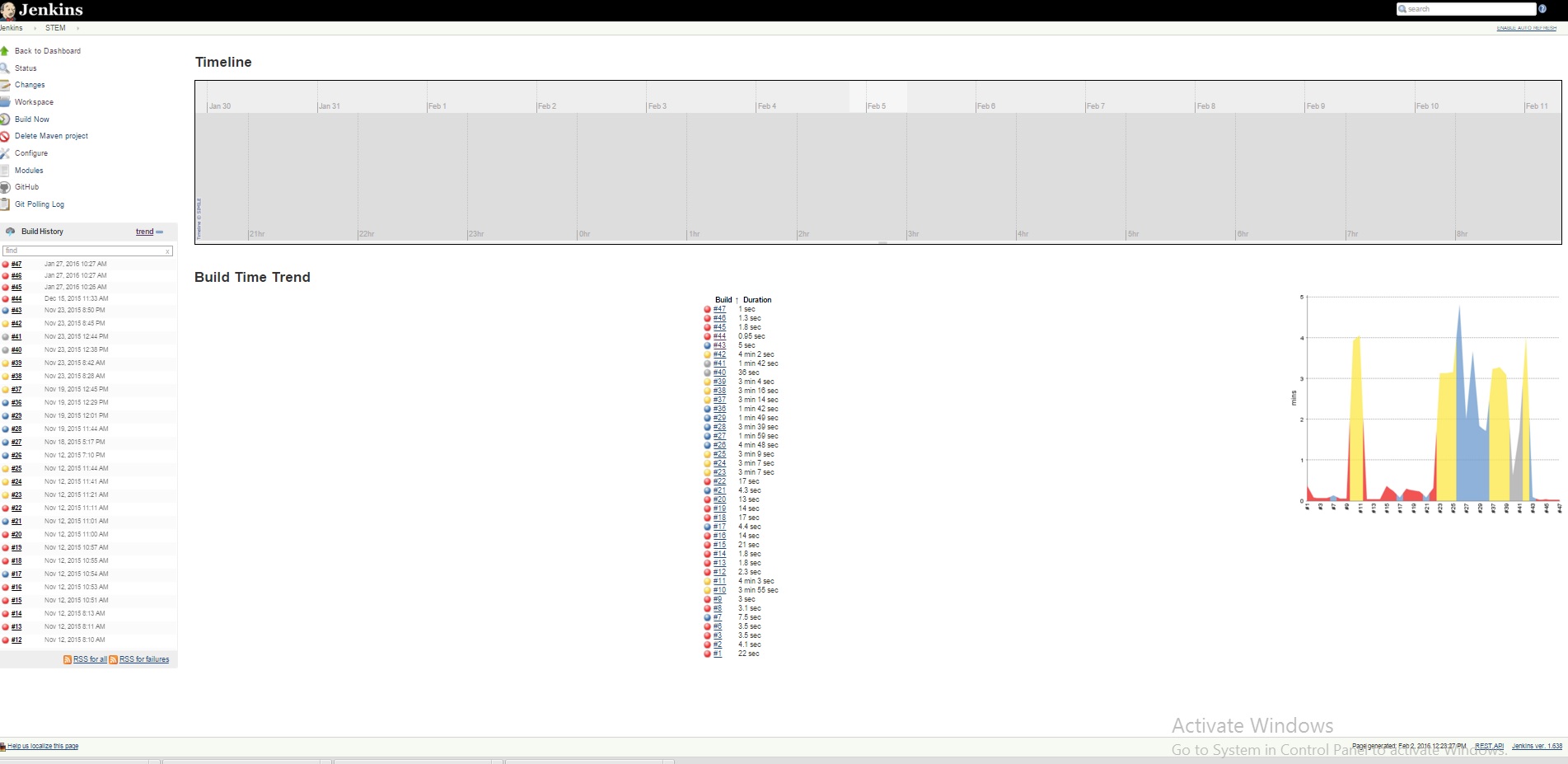
Appendix F: Screen Captures

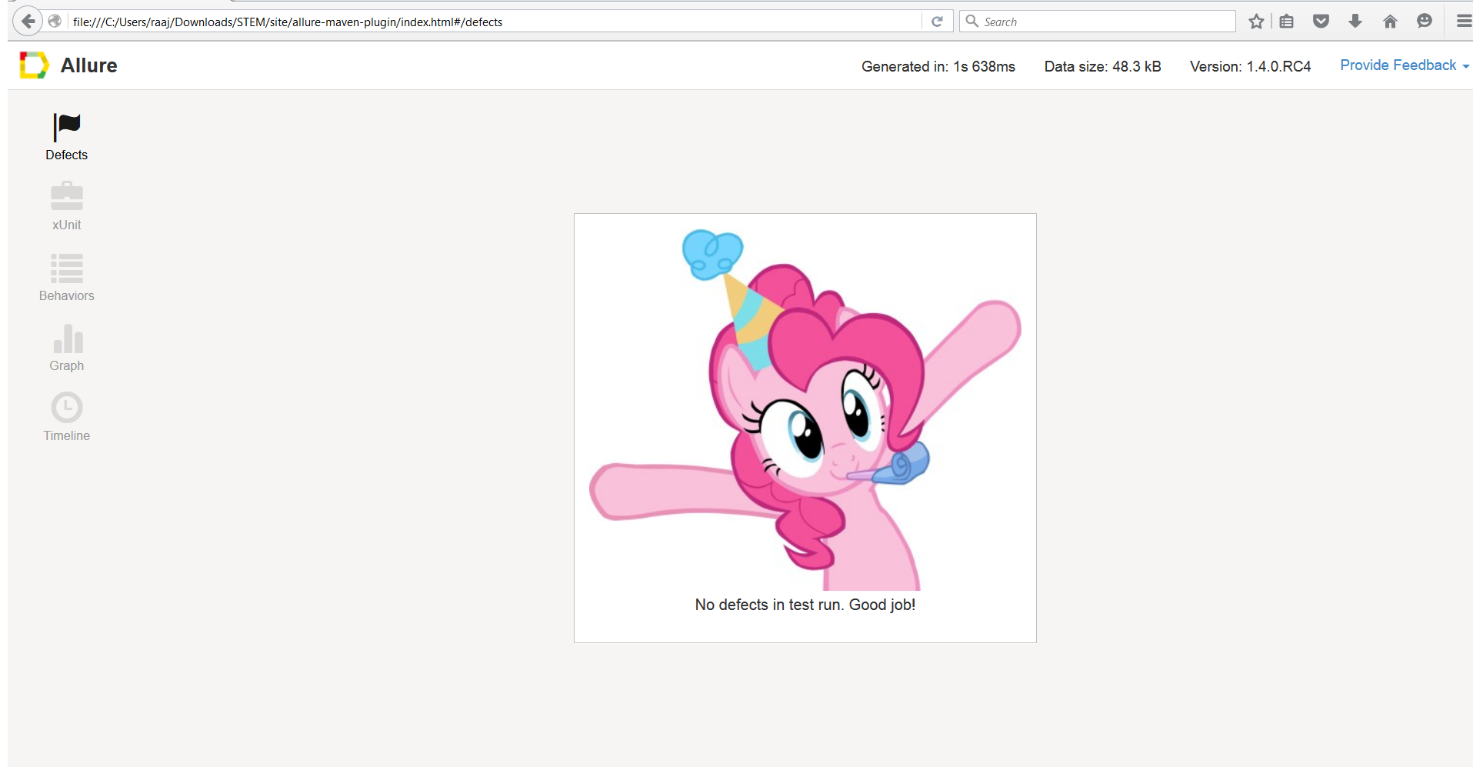


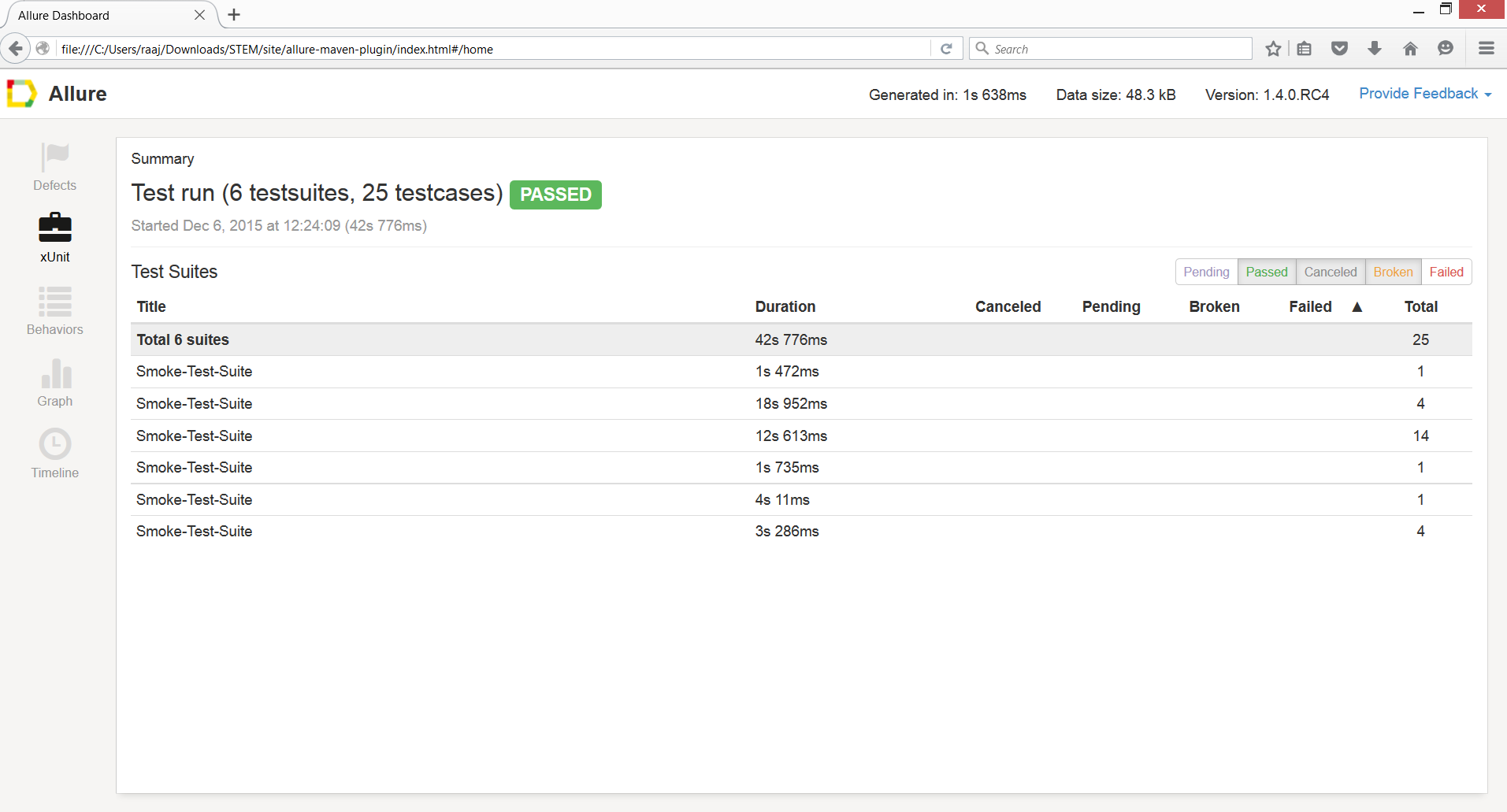


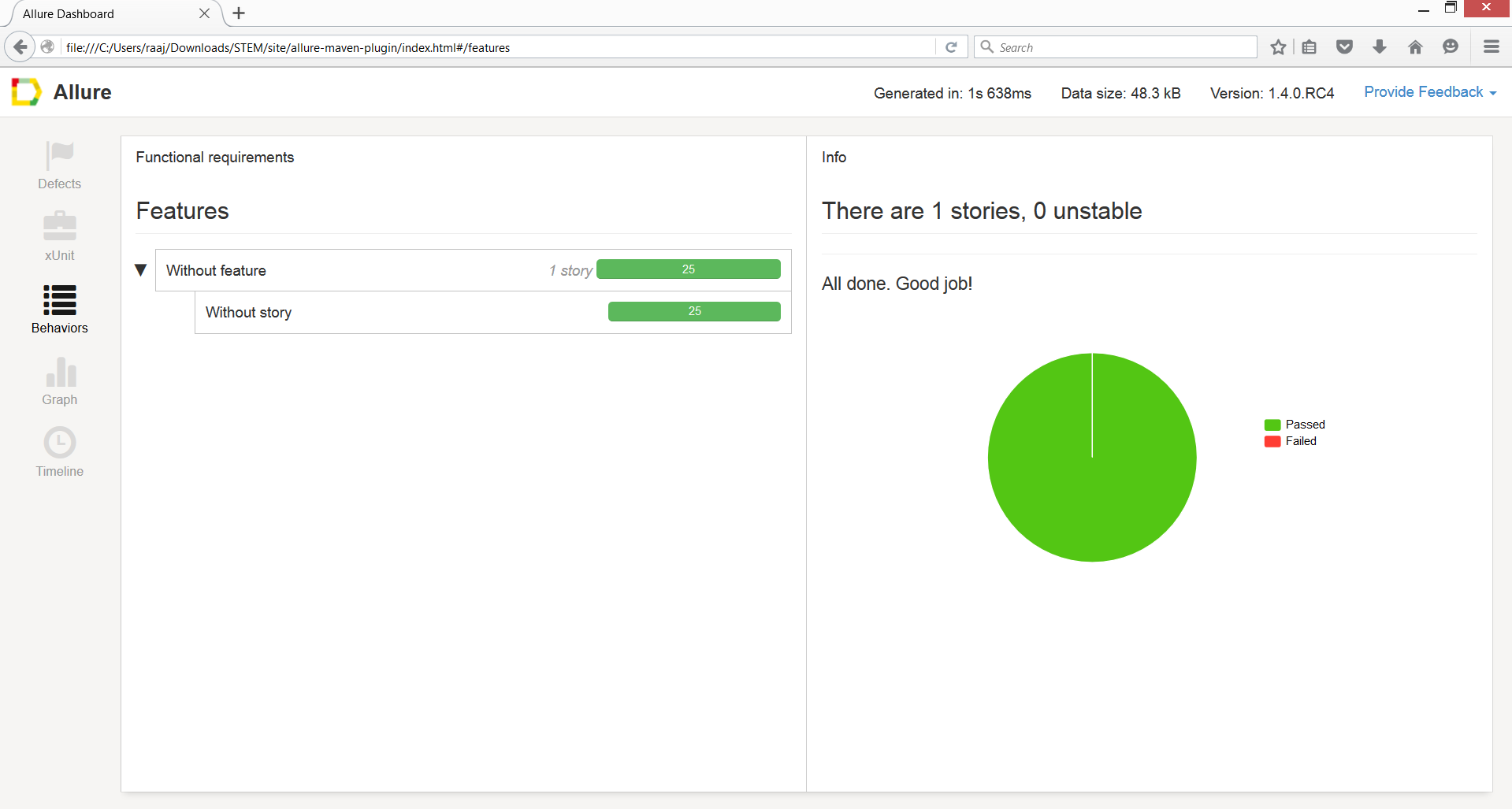


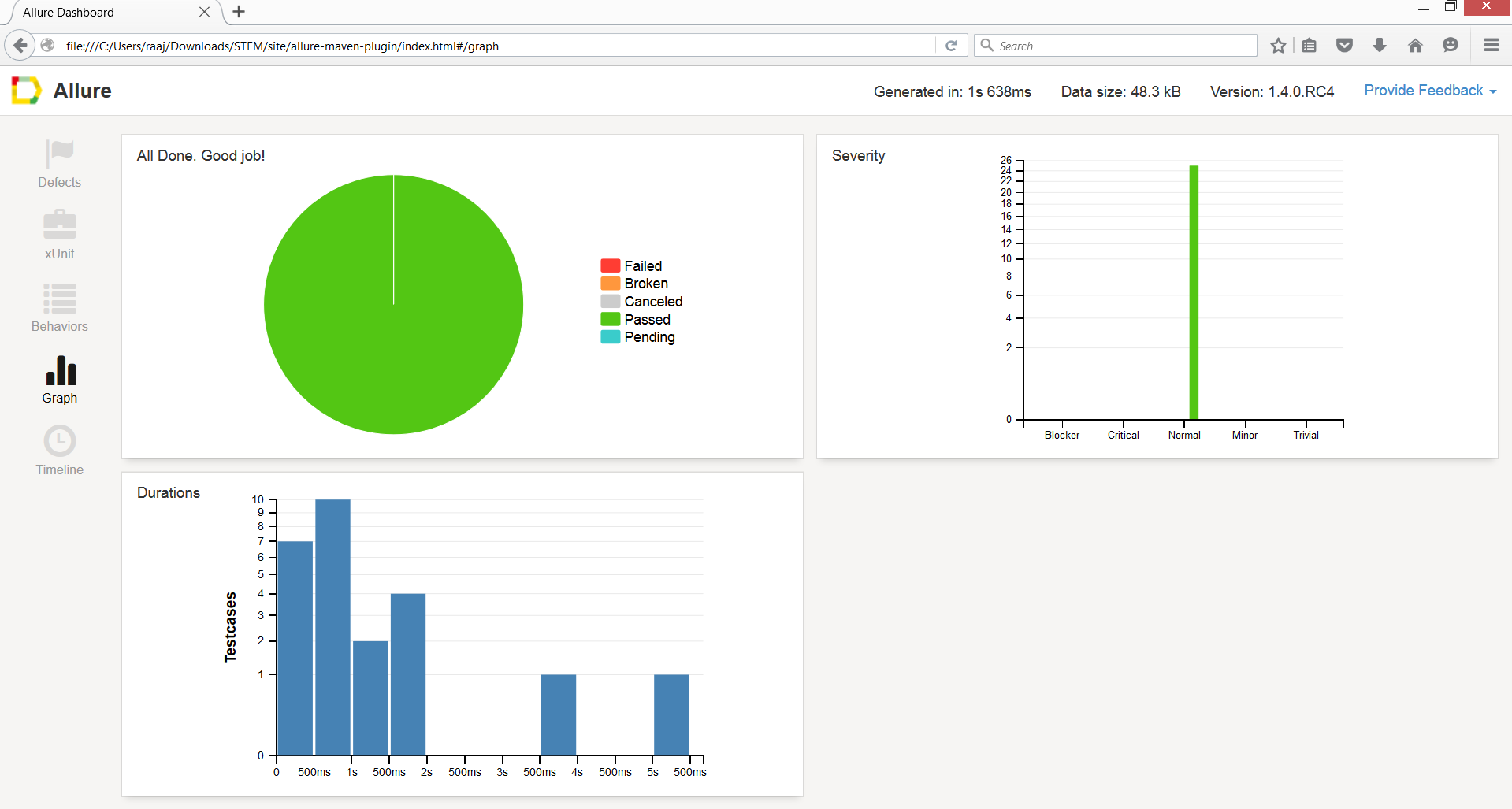


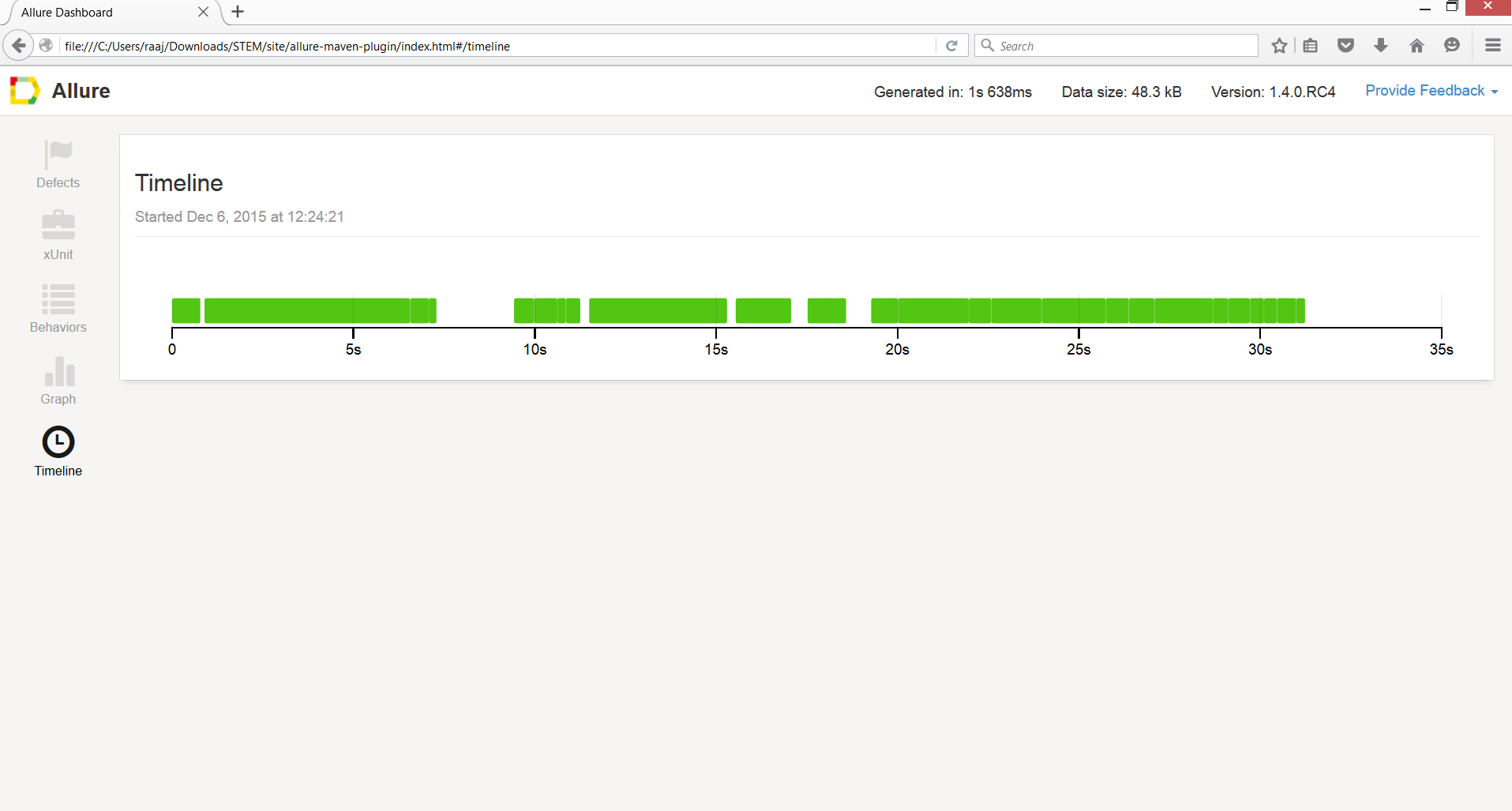












**Prerequisites:**

Jdk

Jre

Xampp

Jenkins

Git (Desktop)

NetBeans

**JDK & JRE**

 The major prerequisite is the Java SDK. We need to install the Java SDK, we should install it to a pathname without spaces,

Once Java is installed, must ensure that the commands from the Java SDK are in our PATH environment variable. Running, for example,

1. java -version

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

If we installed the Java Development Kit (JDK) we'll be setting the JAVA\_HOME environment variable. If we installed the Java Runtime Environment (JRE) we will follow the same steps but set the JRE \_HOME environment variable instead.

Set the JAVA\_HOME Variable

To set the JAVA\_HOME variable:

Find out where Java is installed. If we didn't change the path during installation, it will be something like this:   
C:\Program Files\Java\jdk1.8.0\_65

In Windows 7 right click **My Computer** and select **Properties** > **Advanced**.  
In Windows 8 go to **Control Panel** > **System** > **Advanced System Settings**.

Click the **Environment Variables** button.

Under **System Variables**, click **New.**

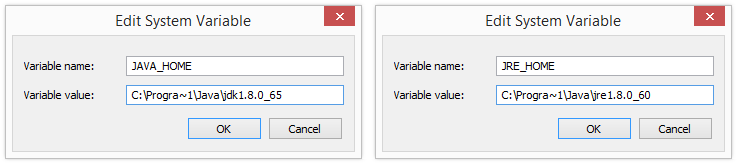
In the **Variable Name** field, enter:

JAVA\_HOME if we installed the JDK (Java Development Kit)  
or

JRE\_HOME if we installed the JRE (Java Runtime Environment)

In the **Variable Value** field, enter wer JDK or JRE installation path.

If the path contains spaces, use the shortened path name, for example C:\Progra~1\Java\jdk1.8.0\_65) 



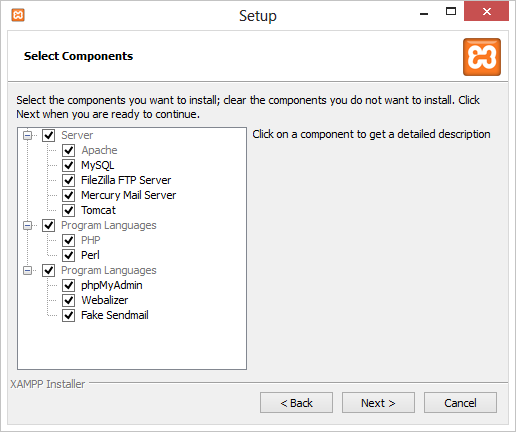
Click **OK**and **Apply Changes** as prompted.

We'll need to close any re-open any command windows that were open before we made these changes as there's no way to reload environment variables from an active command prompt. If the changes don't take effect even after reopening the command window, restart Windows.

**XAMPP installation wizard**

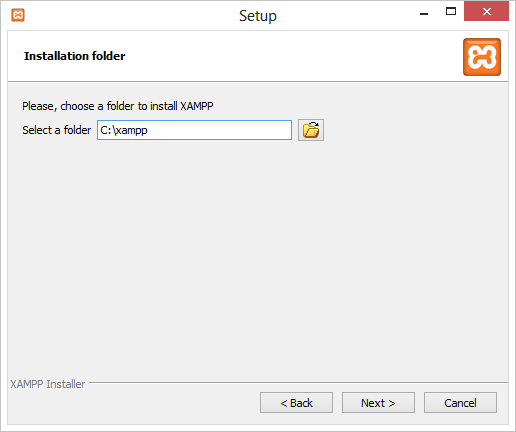
The XAMPP installation wizard is pretty straightforward. Using the **Next** button, we can navigate through all steps in the installation.

One of the first steps where input is required is the Select Components step. In here, we have to specify which components we want to install with XAMPP:

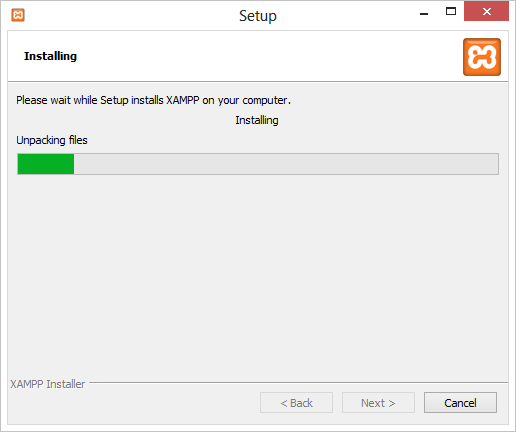


Depending on our plans, we can add/remove certain components here. To work with PHP, we typically require Apache, MySQL (if we want to work with a database) and PHP. Other components are utilities which may come in handy, such as the Fake Send mail which mimics a mail server that can be used during development.

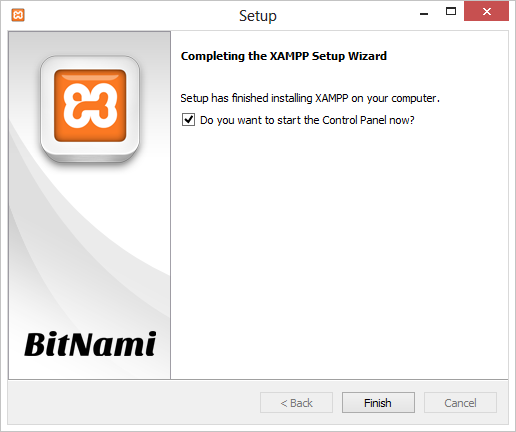
Next, we have to specify the installation folder. We will be using the *c:\xampp* folder.



After clicking the **Next** button again, all required components are extracted to the specified folder.



After extracting and installing all files, we can launch the XAMPP control panel. Let's keep the checkbox ticked and use the **Finish** button to complete installation.

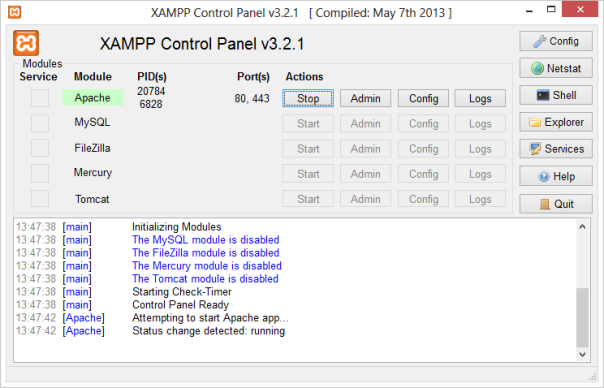


**XAMPP Control Panel**

After installation, we can use the XAMPP Control Panel to perform several actions such as starting / stopping servers and changing configurations.

**Starting and stopping servers**

To start a specific server, click the **Start** button next to them. To stop a running server, we can use the **Stop** button. Let's start the *Apache* server by clicking the **Start** button.



Depending on our system configuration, starting Apache may fail. If that is the case, verify that no other software is listening on TCP ports 80 and 443. Other web servers such as IIS may be using these ports, as well as Skype or other software. Close these other programs and try again, or modify the Apache configuration to listen on different TCP ports.

**Jenkins:**

On Windows it is good to run Jenkins as a service so it starts up automatically without requiring a user to log in. The easiest way is to run the Windows installer, linked from Jenkins' homepage. This also has the advantage of being easier to automate.

The manual way is to follow [Installing Jenkins as a Windows service](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+as+a+Windows+service).

Install Jenkins using the windows installer, we shouldn't need to do anything else here because the windows installer automatically runs Jenkins as a windows service.

First, we need to start Jenkins before installing it. This can be done from by running "java -jar Jenkins. War".

Now connect to Jenkins by going to the following URL http://<hostname>:8080/  
Once Jenkins is started this way, look for the "Install as Windows Service" link in the "Manage Jenkins"



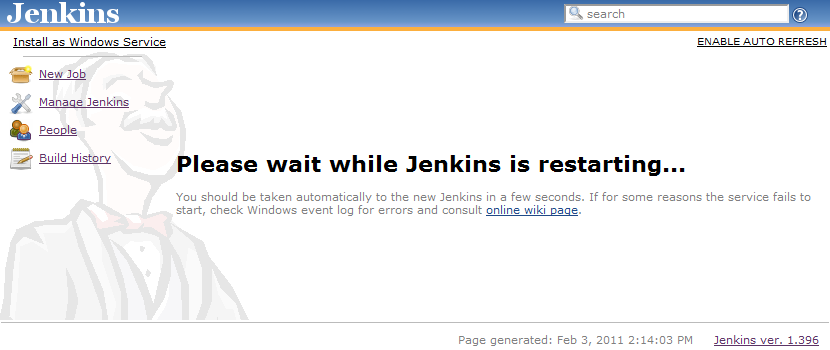
**NOTE**: The "Install as Windows Service" link indicated above will not appear if Jenkins is already running as a service on the machine. We can verifying that Jenkins is running as a service by launching the Services app from the Windows Start Menu and looking for a running service called "Jenkins" among the list of all windows services running on the machine.

Clicking this link shows us the installation screen: 

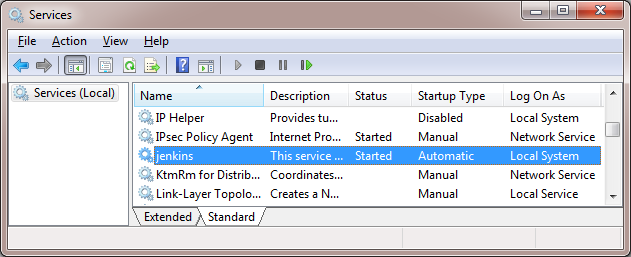
Choose the directory where Jenkins shall be installed (directory must already exist. If not create it before hand). This will become JENKINS\_HOME and used to store data files and programs alike.  
Upon successful completion of the installation, we should see a page asking we to restart Jenkins. 



This is to re-launch Jenkins as a newly installed Windows service. When we click yes, we'll be asked to wait until the restart completes: 



At this point we can use the service manager to confirm that Jenkins is running as a service. 



**How to install plugins**

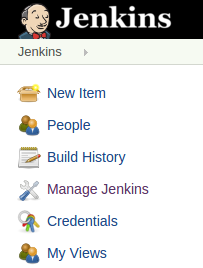
Plugin manager may not show the newest version of a given plugin if we're on an older Jenkins version and the new plugin version would be incompatible.

The simplest way is by going to installation's management screen and clicking ***Manage Plugins*** and go to the ***Available*** tab. We'll find the plugin we'd like to install, select the checkbox, and then either attempt to ***Install without restart* or *Download now and install after restart.*** The web interface will then download \*.hpi files from here. If we***install*** *without restart* the interface will show us progress and provide the results of the install.

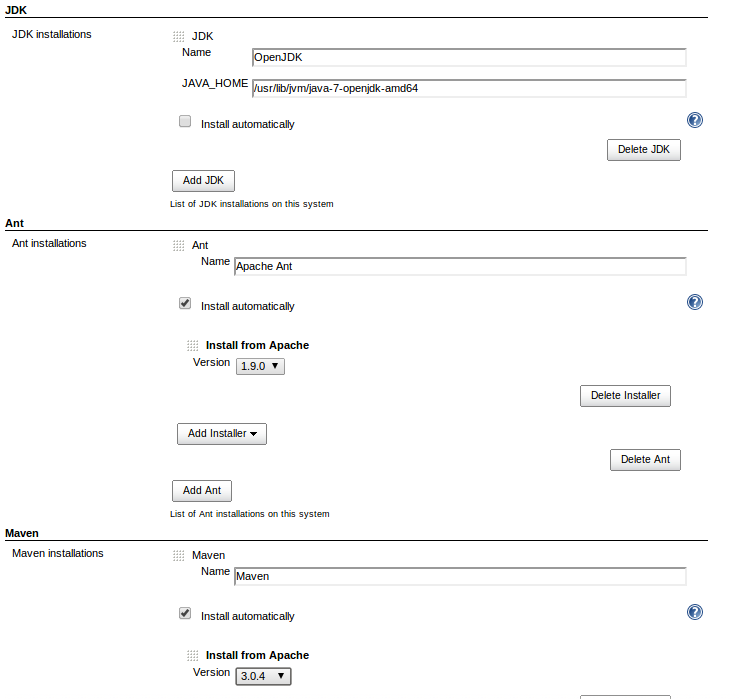
Sometimes when we install, we will notice that the list of available plugins is empty. If that is the case, from ***Advanced*** tab on the ***Manage Plugins*** page, click on ***Check now*** (button available in the bottom right of the page) to forcefully check for new updates. Once that is done, we should see the list of plugins.

**Configure Jenkins**

Before using Jenkins to build Java applications, we need to configure the location or it where wer JDK installation is. Select ***Manage Jenkins*** and afterwards ***Configure System***.



Enter the correct path to our JDK, Apache Ant and Maven and press the *Save* button below. Jenkins can also install these for our automatically.



**Jenkins Git Plugin**

The next thing we have to do is to get some source code on the machine. If we did not navigate away from the job page (http://localhost:8080/job/build/), click Configure link. We will notice that Git is not listed under Source Code Management at job configuration page. We have to install a plugin.

Go to Jenkins home page (it is http://localhost:8080/, to get there click Jenkins link in the top-left part of the page) and then click Manage Jenkins > Manage Plugins > Available. Check Git Plugin checkbox and click Install. Check Restart Jenkins when installation is complete and no jobs are running checkbox and wait for Jenkins to restart. This restarting Jenkins step is the strangest part. I am never sure when it has restarted.

Go back to job configuration page, and we will see Git listed under Source Code Management. Click it and enter URL of repository. I will use Git read only URL from project.

**Git**

Time to install Git. Download it from Git home page <https://git-scm.com> . On download page select Full installer for official Git for Windows. I did not want to install it in Program Files folder, so I have installed it in C:\Git.

Go to Jenkins configuration (Jenkins > Manage Jenkins) and click Configure System (Do not confuse it with job configuration at job > Configure.)

Under Path to Git executable we will see something like this: There's no such executable git.exe in PATH.... Replace git.exe with C:\Git\cmd\git.cmd and click button Save.

Do not replace git.exe with C:\Git\bin\git.exe. Cloning repositories via Git/SSH will not work. Do not ask. I have spent an entire day on that.

Go back to Configure System page, and if we did everything right, there should not be any error message under Path to Git executable. Leave the page open for 5-10 seconds. I have noticed that sometimes the error message does not appear immediately.

Go back to the job and build it again. It should work now. Finally. We will notice blue ball next to the newest build (instead of red ball for failing builds).

**Using Git, Jenkins and pre-build branch merging**

Continuous Integration tools such as Jenkins are useful on projects as they give users early indication that a particular codebase is 'unstable' - and that if a developer checks it out, there will be trouble ahead (they won't be able to work on their own code, because someone else has broken something).

Unfortunately, by the time the build completes, this is often too late (particularly if the build cycle time is very long), as a developer has updated their working copy to the latest, unstable code in the repository and has begun work.

This can lead to the code base remaining unstable as developers tread on each other’s toes steadily fixing one thing, but breaking something else.

Some environments (e.g. TeamCity) attempt to fix this by making commits into SVN only 'really' happen once they have been tested. These kinds of 'delayed-commits' are problematic, because local SCM tools assume that commits will be immediately available, which can confuse them. In many ways this mechanism is a hack to get around the fact that branch management in SVN is very heavyweight.

Fortunately, with GIT and Jenkins, we can achieve the same 'stable branches' with minimal effort.

Set up wer Jenkins project, and leave the 'branch' field in the Git SCM blank. This will cause Jenkins to consider any change on any branch for building.

Next, pick a particular branch name as the integration target in the ***'Advanced'*** section - (e.g. 'master', or 'stable'), and select ***'Merge before build'***.

Select 'Push GIT tags back to origin repository' from the post-build actions (this is required to update wer centralized Git repo with the results of the build).

Now, developers should never commit directly to wer integration branch (the 'master' or 'stable'). Instead, they should either use feature branches, or create new remote branches on commit push origin. We should also set up our GIT repository to only accept commits onto the integration branch from Jenkins. Commits should now be automatically merged with the integration branch (they will fail if they do not merge cleanly), and built. If the build succeeds, the result of the merge will be pushed back to the remote Git repository.

**Push notification from repository**

To minimize the delay between a push and a build, it is recommended to set up the post-receive hook in the repository to poke Jenkins when a new push occurs. To do this, add the following line in wer hooks/post-receive file, where <URL of the Git repository> is the fully qualified URL we use when cloning this repository.

This will scan all the jobs that:

* Have *Build Triggers > Poll SCM* enabled.  No polling *Schedule* is required.
* Are configured to build the repository at the specified URL
* Are configured to build the optionally specified branches or commit ID

For jobs that meet these conditions, polling will be immediately triggered.  If polling finds a change worthy of a build, a build will in turn be triggered.

This allows a script to remain the same when jobs come and go in Jenkins. Or if we have multiple repositories under a single repository host application, we can share a single post-receive hook script with all the repositories. Finally, this URL doesn't require authentication even for secured Jenkins, because the server doesn't directly use anything that the client is sending. It runs polling to verify that there is a change, before it actually starts a build.

**Jenkins Build Jobs**

Creating a new build job in Jenkins is simple: just click on the “New Job” menu item on the Jenkins dashboard. Jenkins supports several different types of build jobs, which are presented to we when we choose to create a new job

***Freestyle software project***

Freestyle build jobs are general-purpose build jobs, which provides a maximum of flexibility.

***Maven project***

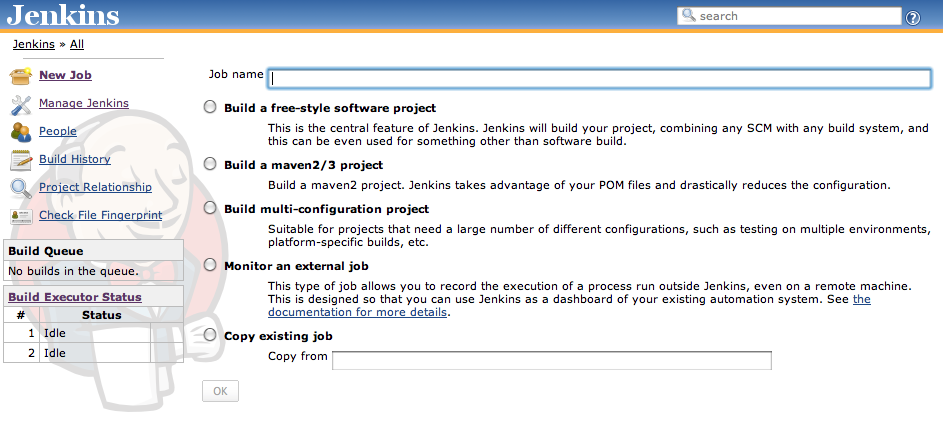
The “maven2/3 project” is a build job specially adapted to Maven projects. Jenkins understands Maven *pom* files and project structures, and can use the information gleaned from the *pom* file to reduce the work we need to do to set up wer project.

***Monitor an external job***

The “Monitor an external job” build job lets we keep an eye on non-interactive processes, such as cron jobs.

***Multiconfiguration job***

The “Multiconfiguration project” (also referred to as a “matrix project”) lets we run the same build job in many different configurations. This powerful feature can be useful for testing an application in many different environments, with different databases, or even on different build machines. We will be looking at how to configure Multiconfiguration build jobs later on in the book.



**Configuring Git to NetBeans (source: https://netbeans.org/kb/docs/ide/git.html)**

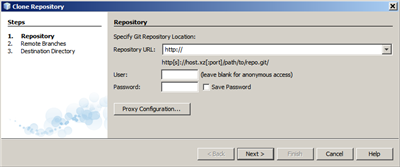
**Requirements**

|  |  |
| --- | --- |
| Software or Resource | Version Required |
| [NetBeans IDE](https://netbeans.org/downloads/index.html) | Version 7.4 or 8.0 |
| [Java Development Kit](http://www.oracle.com/technetwork/java/javase/downloads/index.html) (JDK) | Version 7 or 8 |

**Cloning a Git Repository**

To get a copy of an already existing Git repository, we need to clone it. Make sure we know the Git repository URL before starting the Clone Repository wizard in the IDE.

Choose Team > Git > Clone from the main menu. The Clone Repository wizard displays.



At the Repository page, specify the path to a Git repository location, user name and password (we can save them for the future if required).

(Optional) Click Proxy Configuration to display the Options dialog box and set the proxy server settings. Click OK when finished.

Click Next to switch to the next step of the wizard.

At the Remote Branches page, select the repository branch(es) to be fetched (downloaded) to local repository. Click Next.

At the Destination Directory page, specify the following:

In the Parent Directory field, the path to the directory intended for the cloned repository on local hard drive (alternatively, click the Browse button and navigate to the directory).  
The Parent Directory field is pre-filled with the path to the default NetBeans Projects directory where all NetBeans projects are stored.

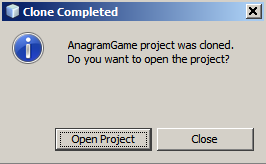
In the Clone Name field, the name of the local folder where the original project will be cloned to.  
By default Clone Name is filled out with the actual Git repository name.

In the Checkout Branch field, select the branch to be checked out into the working tree.

In the Remote Name field, the name that represents the original repository being cloned.  
Origin is the default alias of the repository being cloned. It is a recommended value.

Leave the Scan for NetBeans Projects after Clone checkbox selected to activate after-scanning right after the clone finishes. (The plugin searches for NetBeans projects in the cloned resources and offers to open the found projects.)

Click Finish.  
After a Git repository is cloned, the metadata .Git folder is created inside the folder we selected in the wizard.



Choose the desired option.

Adding Files to a Git Repository

To start tracking a new file and also to stage changes to an already tracked file in the Git repository, we need to add it to the repository.

When adding files to a Git repository, the IDE composes and saves snapshots of our project first in the Index. After we perform the commit, the IDE saves those snapshots in the HEAD. The IDE allows us to choose between the two workflows described in the following table.

|  |  |  |
| --- | --- | --- |
| Workflow Description | Explicitly add new or modified files to the Index and then commit only those that are staged in the Index to the HEAD | Skip adding new or modified files to the Index and commit the required files directly to the HEAD |
| Steps to Follow the Workflow | In the Projects window, right-click the file we want to add.  In the context menu, choose Git > Add. This adds the file contents to the Index before we commit it.  In the Projects window, right-click the file we want to commit.  In the Commit dialog box, select the Changes between HEAD and Index (  ) toggle button. This displays the list of files that are already staged.  Commit the file(s) as described in the [Committing Sources to a Repository](https://netbeans.org/kb/docs/ide/git.html#committing) section below. | In the Projects window, right-click the file we want to commit.  In the context menu, choose Git > Commit.  In the Commit dialog box, select the Select the Changes between Index and Working Tree () toggle button.  This displays the list of files that are not staged.  Commit the file(s) as described in the [Committing Sources to a Repository](https://netbeans.org/kb/docs/ide/git.html#committing) section below. |

Note: The [status](https://netbeans.org/kb/docs/ide/git.html#viewFileStatus) of the file in the HEAD displays in green to the left of the slash like shown in the following picture.

**Editing Files**

Once we have a Git versioned project opened in the IDE, we can begin making changes to sources. As with any project opened in NetBeans IDE, we can open files in the Source Editor by double-clicking on their nodes, as they appear in the IDE's windows (e.g., Projects (Ctrl-1), Files (Ctrl-2), Favorites (Ctrl-3) windows).

When working with source files in the IDE, there are various UI components at our disposal, which aid in both viewing and operating version control commands:

[Viewing Changes in the Source Editor](https://netbeans.org/kb/docs/ide/git.html#viewChanges)

[Viewing File Status Information](https://netbeans.org/kb/docs/ide/git.html#viewFileStatus)

[Reverting Changes](https://netbeans.org/kb/docs/ide/git.html#revert)

Viewing Changes in the Source Editor

When we open a versioned file in the IDE's Source Editor, we can view real-time changes occurring to the file as we modify it against the base version from the Git repository. As we work, the IDE uses color coding in the Source Editor's margins to convey the following information:

|  |  |
| --- | --- |
| **Blue** (       ) | Indicates lines that have been changed since the earlier revision. |
| **Green** (       ) | Indicates lines that have been added since the earlier revision. |
| **Red** (       ) | Indicates lines that have been removed since the earlier revision. |

**Viewing File Status Information**

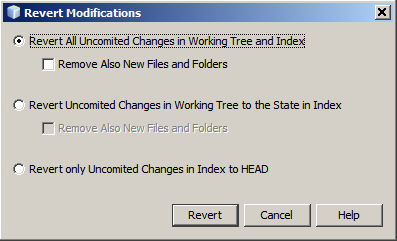
When we are working in the Projects (Ctrl-1), Files (Ctrl-2), Favorites (Ctrl-3), or Versioning views, the IDE provides several visual features that aid in viewing status information about project files. In the example below, notice how the badge color of the file name, and adjacent status label, all coincide with each other to provide we with a simple but effective way to keep track of versioning information on project files:

**Reverting Changes**

To throw away local changes made to selected files in project Working Tree and replace those files with the ones in the Index or HEAD:

Select a versioned file or folder (e.g. from the Projects, Files, or Favorites window).

Choose Team > Revert Modifications from the main menu.  
The Revert Modifications dialog box displays.



Specify additional options (e.g., Revert only Uncommitted Changes in Index to HEAD)

Click Revert.

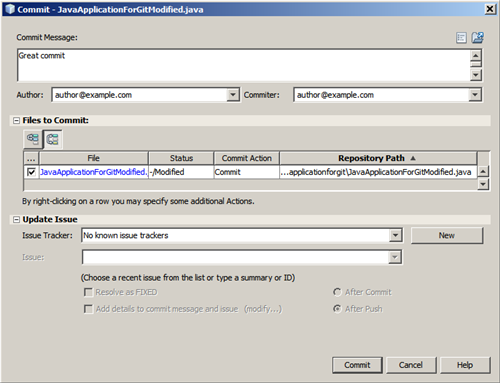
**Committing Sources to a Repository**

To commit files to the Git repository:

In the Projects window, right-click the file(s) we want to commit.

In the context menu, choose Git > Commit.

The Commit dialog box displays.



The Commit dialog box contains the following components:

Commit Message text area intended for describing the change being committed

Author and Committer drop-down lists that allow to differentiate between those who made the change and who physically committed the file if necessary.

Files to commit section that lists:

All files modified,

All files that have been deleted in the Working Tree (locally),

All new files (i.e. files that do not yet exist in the Git repository),

All files that we have renamed.

Two toggle buttons that switch the mode in which the actual commit is to be performed are available here:

|  |  |  |
| --- | --- | --- |
| UI Component | Name | Description |
|  | **Changes between HEAD and Index** | Displays a list of files that are staged. |
|  | **Changes between HEAD and Working Tree** | Displays a list of files that are either already staged or only modified/created and not staged yet. |

Note: To specify here whether to exclude individual files from the commit, either deselect the checkbox in the first column called Commit or right-click a file row in the Commit Action column and choose Exclude from commit from the pop-up menu. To display the Diff Viewer here, right-click a file row in the Commit Action column and choose Diff from the pop-up menu.

Update Issue section intended for tracking issues related to the change being committed.

Note: We need to install the JIRA or Subversion plugin to start tracking issues in the IDE.

Type in a commit message in the Commit Message text area. Alternatively, we can do any of the following:

Click the Recent Messages icon located in the upper right corner to view and select from a list of messages that we have previously used,

Click the Load Template icon located in the upper right corner to select a message template.

After specifying actions for individual files, click Commit.   
The IDE executes the commit and stores project file snapshots to the repository. The IDE's status bar, located in the bottom right of the interface, displays as the commit action takes place. Upon a successful commit, versioning badges disappear in the Projects, Files and Favorites windows, and the color coding of committed files returns to black.

**Working with Branches**

The IDE's Git support enables us to maintain different versions of an entire code base using branches.

When working with branches in the IDE, the following actions are supported:

[Creating](https://netbeans.org/kb/docs/ide/git.html#branchCreate)

[Checking out](https://netbeans.org/kb/docs/ide/git.html#branchCheckOut)

[Merging](https://netbeans.org/kb/docs/ide/git.html#branchMerge)

[Deleting](https://netbeans.org/kb/docs/ide/git.html#branchDelete)

**Creating a Branch**

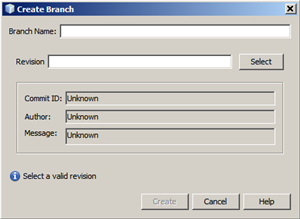
To create a local branch, if we want to work on a separate version of project file system for stabilization or experimentation purposes without disturbing the main trunk, complete the following steps:

In the Projects or Files window, choose a project or folder from the repository in which we want to create the branch.

In the main menu, choose Team > Branch/Tag > Create Branch.

Note: As an alternative right-click the versioned project or folder and choose Git > Branch/Tag > Create Branch in the popup menu.

The Create Branch dialog box displays.

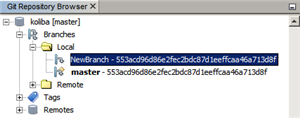


In the Branch Name field, enter the name of the branch being created.

Type a specific revision of the selected item by entering a commit ID, existing branch, or tag name in the Revision field or press Select to view the list of revisions maintained in the repository.

(Optional) In the Select Revision dialog box, expand Branches and choose the branch required, specify the commit ID in the adjacent list, and press Select.

Review the Commit ID, Author, and Message fields’ information specific to the revision being branched from and click Create.  
The branch is added to the Branches/Local folder of the Git repository.

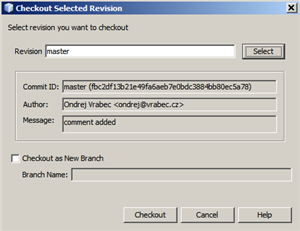


**Checking Out**

If we need to edit files on a branch that already exists, we can check out the branch to copy the files to project Working Tree.

To check out a revision, perform the following:

Choose Team > Checkout > Checkout Revision from the main menu.   
The Checkout Selected Revision dialog box displays.



Specify the revision required by entering a commit ID, existing branch, or tag name in the Revision field or press Select to view the list of revisions maintained in the repository.

Skip if we did not press Select in the previous step. In the Select Revision dialog box, expand Branches and choose the branch required, specify the commit ID in the adjacent list if required, and press Select.

Note: If the specified revision refers to a valid commit that is not marked with a branch name, our HEAD becomes detached and we are no longer on any branch.

Review the Commit ID, Author, Message fields information specific to the revision being checked out.

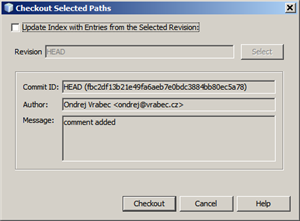
To create a new branch out of the checked out revision, choose the Checkout as New Branch option and enter the name in the Branch Name field.

Press Checkout to check out revision.  
Files in the Working Tree and in the Index are updated to match the version in the specified revision.

Note: If we want to switch project files to a branch that already exists (e.g., to a commit that is not at the top of one of project branches), we can use the Team > Git > Branch > Switch To Branch command, specify the branch in the Switch to Selected Branch dialog box, check it out as a new branch (optionally), and press Switch.

The IDE supports context-sensitive checkout of the file(s), folder(s), or project(s) currently selected in the IDE. To check out some files (not a branch) from the Index, complete the following steps:

Choose Team > Checkout > Checkout Files from the main menu.   
The Checkout Selected Paths dialog box displays.



Choose the Update Index with Entries from the Selected Revision option.  
If selected, the Index is updated with the state in the selected revision prior to the checkout itself (i.e., the selected files both in the Working Tree and Index are updated).

Specify the revision required by entering a commit ID, existing branch, or tag name in the Revision field or press Select to view the list of revisions maintained in the repository.

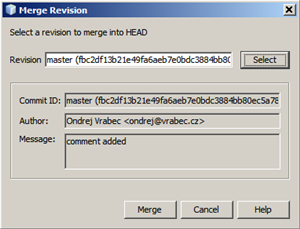
Skip if we did not press Select in the previous step. In the Select Revision dialog box, expand Branches and choose the branch required, specify the revision number in the adjacent list if required, and press Select.

Press Checkout to complete the checkout.

**Merging**

To port modifications from a repository revision to the Working Tree, do as follows:

Choose Team > Branch/Tag > Merge Revision from the main menu.  
The Merge Revision dialog box displays.



Specify the revision required by entering a commit ID, existing branch, or tag name in the Revision field or press Select to view the list of revisions maintained in the repository.

Skip if we did not press Select in the previous step. In the Select Revision dialog box, expand Branches and choose the branch required, specify the commit ID in the adjacent list if required, and press Select.

Press Merge.  
A three-way merge between the current branch, project Working Tree contents, and the specified branch is done.

Note: If a merge conflict occurs, the conflicting file is marked with [a red badge](https://netbeans.org/kb/docs/ide/git.html#badges) to indicate this.

Note: After merging we must still [commit](https://netbeans.org/kb/docs/ide/git.html#committing) the changes in order for them to be added to the HEAD.

**Deleting a Branch**

To delete an unnecessary local branch, complete the following steps:

Choose Team > Repository Browser from the main menu.

In the Git Repository Browser, choose the branch to be deleted.

Note: The branch must be inactive, i.e. not currently checked out into the Working Tree.

Right-click the selected branch and choose Delete Branch from the popup menu.

In the Delete Branch dialog box, press OK to confirm the branch deletion.  
The branch is removed from the local repository as well as the Git Repository Browser.

**Working with Remote Repositories**

When we work with other developers we need to share project work, which involves fetching, pushing, and pulling data to and from remote repositories hosted on the Internet or network.

[Fetching](https://netbeans.org/kb/docs/ide/git.html#fetch)

[Pulling](https://netbeans.org/kb/docs/ide/git.html#pull)

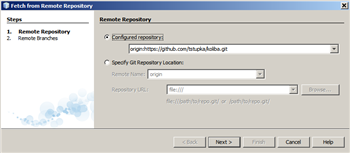
[Pushing](https://netbeans.org/kb/docs/ide/git.html#push)

**Fetching**

Fetching gets the changes from the original remote repository that we do not have yet. It never changes any of project local branches. Fetching gets all the branches from remote repositories, which we can merge into project branch or just inspect at any time.

To fetch the updates, do as follows:

Choose Team > Remote > Fetch.  
The Fetch from Remote Repository wizard displays.



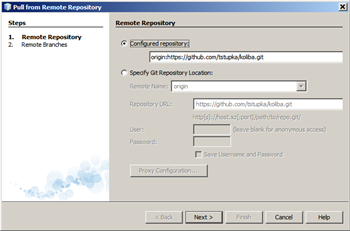
At the Remote Repository page of the wizard, select either the Configured repository (to use the path to the repository configured earlier) or Specify Git Repository Location option (to define the path to a remote repository that has not been accessed yet, its name, login, password, and proxy configuration if required) and click next.

At the Remote Branches page of the wizard, choose the branches to fetch changes from and click Finish.  
A local copy of a remote branch is created. The selected branch are updated in the Branches > Remote directory in the Git Repository Browser.  
Next the fetched updates can be merged into a local branch.

**Pulling**

When pulling some updates from a remote Git repository, the changes are fetched from it and merged into the current HEAD of our local repository.  
To perform pulling, complete the following steps:

Choose Team > Remote > Pull.  
The Pull from Remote Repository wizard displays.



At the Remote Repository page of the wizard, select either the Configured repository (to use the path to the repository configured earlier) or Specify Git Repository Location option (to define the path to a remote repository that has not been accessed yet, its name, and login and password if required) and click next.

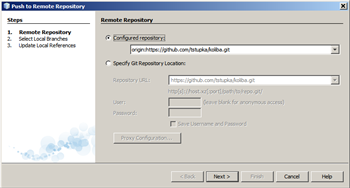
At the Remote Branches page of the wizard, choose the branches to be pulled changes from and click Finish.  
Project local repository is synchronized with the origin repository.

**Pushing**

To contribute changes from project local Git repository into a public Git repository, perform the following steps:

Note: When we are going to push project files commits into a repository and some other changes have been pushed in the meantime, we need to pull and merge those changes first.

Choose Team > Remote > Push.  
The Push to Remote Repository wizard displays.



At the Remote Repository page of the wizard, select either the Configured repository (to use the path to the repository configured earlier) or Specify Git Repository Location option (to define the path to a remote repository that has not been accessed yet, its name, and login and password if required) and click next.

At the Select Local Branches page, choose the branch (es) to push project files edits to and click next.

At the Update Local References page, choose the branch (es) to be updated in the Remotes directory of our local repository and click Finish.  
The specified remote repository branch is updated with the latest state of project local branch.