|  |  |
| --- | --- |
|  | **2016** |
|  |  |

|  |
| --- |
| **Interfacing Time tracking application wunderlist** |
|  |



Digital Engineering Team project

on

Time Tracking

Time tracking application that interfaces with the cloud application wunderlist.

|  |  |  |  |
| --- | --- | --- | --- |
| ***S.no*** | ***Team Members*** | ***Matrik.Nr*** | ***Role*** |
| *1.*  *2.* | *Venkata Valluri V*  *Balakrishna Pemmasani* | *217197*  *210049* | *Software Engineer*  *Test engineer* |
| *3.*  *4.* | *Ikram Ul Haq*  *Satya Sai Gokul* | 209372  206744 | *Software engineer*  *Operations Engineer* |

Table of Contents

[1 Motivation: 4](#_Toc457763024)

[2 Introduction 4](#_Toc457763025)

[3 State of the art:- 4](#_Toc457763026)

[3.1 Puppet: 4](#_Toc457763027)

[3.2 Installing Puppet on server: 5](#_Toc457763028)

[3.3 Apache & Tomcat 7](#_Toc457763031)

[4 Server Infrastructure 9](#_Toc457763034)

[4.1 Production.mrcc.ovgu.de 9](#_Toc457763035)

[4.2 testingcarrot.mrcc.ovgu.de: 10](#_Toc457763036)

[5 Continuous Integration and Continuous Delivery: 10](#_Toc457763037)

[5.1 Usage of CI & CD 10](#_Toc457763038)

[6 Jenkins 11](#_Toc457763039)

[7 GitHub 14](#_Toc457763041)

[8 Maven 14](#_Toc457763042)

[9 Scrum 15](#_Toc457763043)

[10 Diagrams notations 16](#_Toc457763044)

[11 Basic Architecture 16](#_Toc457763045)

[11.1 Database 17](#_Toc457763046)

[11.2 Timetracker 17](#_Toc457763047)

[11.3 Wunderlist 17](#_Toc457763048)

[12 Operations 17](#_Toc457763049)

[12.1 Create List 18](#_Toc457763050)

[12.2 Create Task 18](#_Toc457763051)

[12.3 Refresh/Synchronize 18](#_Toc457763052)

[12.4 Start, Pause and Finish 18](#_Toc457763053)

[13 Request Processing 19](#_Toc457763054)

[13.1 Step 1 19](#_Toc457763055)

[13.2 Step 2 19](#_Toc457763056)

[14 Refresh/Synchronize 19](#_Toc457763057)

[15 Start Task 20](#_Toc457763058)

[16 Pause Task 21](#_Toc457763059)

[17 Finish Task 22](#_Toc457763060)

[18 Unique Identification 23](#_Toc457763061)

[19 Mockito: 24](#_Toc457763062)

[19.1 Unit Testing: 24](#_Toc457763063)

[19.2 Steps for testing with mockito framework 25](#_Toc457763064)

[20 Black Box Testing: 26](#_Toc457763069)

[20.1 Selenium: 26](#_Toc457763070)

[20.2 Selenium Web Driver: 26](#_Toc457763071)

[20.3 Test Cases: 26](#_Toc457763072)

[21 Maven in Selenium: 29](#_Toc457763073)

[22 References: 29](#_Toc457763074)

**Abstract.**This paper addresses about the Time tracking application which interfaces with the cloud application wunderlist. Exemplary implementation includes development of application with continuous integration Jenkins and Maven tool, considered as a packaging and build tools that formalizes Java application dependencies.

**Keywords:** Time tracker, Cloud, wunderlist, Maven, Data abstraction, Security, Restful API, Web server.

1. Motivation:

The aim of this project is to develop a web application that helps user in hassle free management of his activities and also to track how much time is spent for working on a task, by considering wunderlist as an activity management application. Utilizing wunderlist Restful API to synchronize the tasks and lists from wunderlist into Time tracking application.

1. Introduction

Wunderlist is a cloud application for managing activities or tasks. It allows the user to organize tasks into lists in an intuitive web interface. Frameworks Java programming language is used for developing the application along with Apache tomcat which is an open-source web server and Java server pages(JSP) is used for developing web pages that support the dynamic content.

Further Technology stack consists of Maven that helps in packaging dependencies easily and has a standard directory structure. Git source code versioning tool is used to maintain records or changes to time tracker file over time so that our team members can recall specific versions later. Jenkins is one of the most popular tools used for automating a system, providing continuous Integration and continuous delivery. As we used Apache Maven type project, building of the project took place under Jenkins functionality by running automated tests.

For example, "Fit Bit" connected to the phone will send data to the server so that it helps stay motivated and improve the health of a person by tracking your activity, exercise, food, weight and sleep.

1. State of the art:-
   1. Puppet:

Puppet is an automation tool aid to configure a central server or master and agents or the clients synchronize themselves to it. Puppet helps to solve four major challenges .

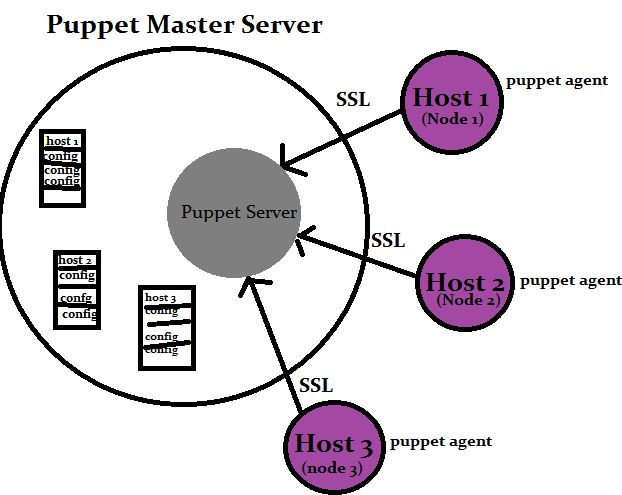
Deploying servers with base configurations

Remediating inconsistent configurations

Continuous deployment of applications

Automation all the above things.

Puppet uses a client server architecture. Server is called the puppet master and clients are called nodes. Puppet master is a Linux server. A puppet node can be linux, unix or windows and each node runs on a small program called puppet agent, master and node communicate securely via ssl. Puppet also verifies the configuration on a server for every specified time interval.



**Fig. 1.** how puppet works

Puppet supports many platforms, Unix and Linux are widely managed. Following are the platforms supported by puppet.

* Microsoft windows
* Debain
* Redhat...
* Facter helps puppet to get the all the details about the node to the puppet master.
  1. Installing Puppet on server:

Linux server is the operational platform and a master less puppet run is performed on the machine at the carrot.mrcc.ovgu.de address. The configuration and the package installation is done on this machine.

### Setting up puppet environment:**.**

Initially on puppet server we have downloaded a puppet package for Ubuntu 14.04.

wget http://apt.puppetlabs.com/puppetlabs-release-trusty.deb

dpkg -i /tmp/puppetlabs-release-trusty.deb

Above commands are used to download the package and then to install the package. Once we have the package an update operation is performed and then masterless puppet is installed. Below are the commands to install masterless puppet.

apt-get update

apt-get install puppet git-core

Puppet has been installed and we had done initial push to puppet repository in the directory etc/puppet , initialized Git repository here and to added everything to this directory by giving the commands

git init& git add.

### Puppet scripts:**.**

install\_puppet\_module.sh and run\_masterless\_puppet.sh command is passed to run the master less puppet under the opt/de\_scripts.

mastelesspuppet.jpg

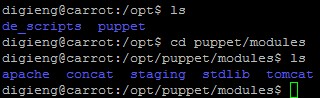
**Fig. 1.**

Installing Modules:-

Puppet modules are self contained bundles of code and data that are reusable and sharable. Module is a directory tree with a particular structure for instance basic structure of a module file is < MODULE NAME> followed by manifests, files, templates, lib, facts.d, examples, spec.

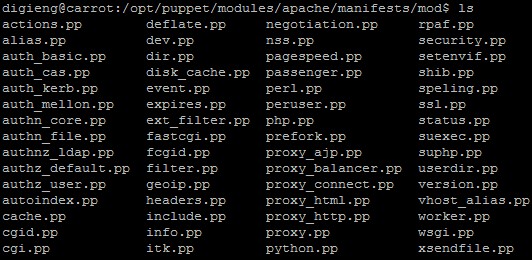
Apache, Concat, Staging, Stdlib and tomcat modules are placed under opt/puppet/modules.

Puppet modules are installed from forge.puppetlabs.com. Installing command for apache module on the master less puppet server is   
$ puppet module install puppetlabs-apache.



**Fig. 1.**

Each module consists of manifests i.e., language files for puppet and are named with .pp extension. "Site manifest" or "main manifest" is compiled first in a puppet as it is a single manifest. These manifests are kept under apache/manifests/mod.



**Fig. 1.**

Tomcat, concat, staging and stdlib modules are also installed under opt/puppet/modules.

* 1. Apache &Tomcat

To execute JEE specifications including Java servlets and to render web pages that has Java server Page coding Tomcat is used. The puppet module simplifies the task of creating configurations to manage Apache server in this infrastructure.

### Puppet install Apache.

To have Puppet install Apache with the default parameters, we declared the

[`apache`][] class:

puppet

class { 'apache': }

The Puppet module applies a default configuration for Ubuntu Operating system and will have a unique default configurations. The default [`apache`][] class sets up a virtual host on port 80.

puppet

apache::vhost { 'vhost.example.com':

port => '80',

docroot => '/var/www/vhost',

}

To obtain the functionality of the Apache on the server, run\_masterless\_puppet.sh command under opt/de\_script.

### Puppet install Tomcat**.**

The Puppet module applies a default configuration for Ubuntu Operating system and will have a unique default configurations. Puppet install on the server with the tomcat module is to install the Tomcat package from EPEL. A new tomcat version is used in this project.

command-- puppet module upgrade puppetlabs-stdlib

class { 'tomcat':

install\_from\_source => false,

}

class { 'epel': }

tomcat::instance{ 'default':

package\_name => 'tomcat',

}

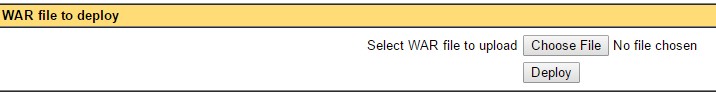
After installing the puppet modules to obtain the functionality of the tomcat on the server run\_masterless\_puppet.sh command under opt/de\_script.

Puppet manifest files are kept under the manifests/config/server

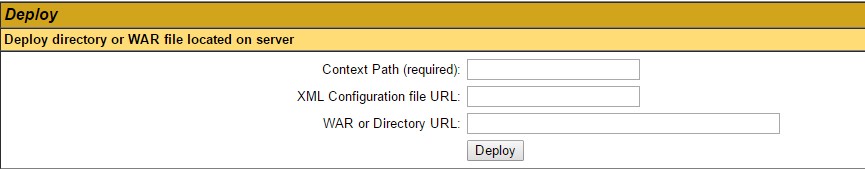
tomcat.jpg

1. Server Infrastructure
   1. Production.mrcc.ovgu.de

Production.mrcc.ovgu.de is the production server of the time tracking application. Developed code is placed in this tomcat web application manager. Files with .war extension are deployed directly under WAR file to deploy section. Once the war file is deployed , the functionality can be checked on the server directly, under their respective folders.

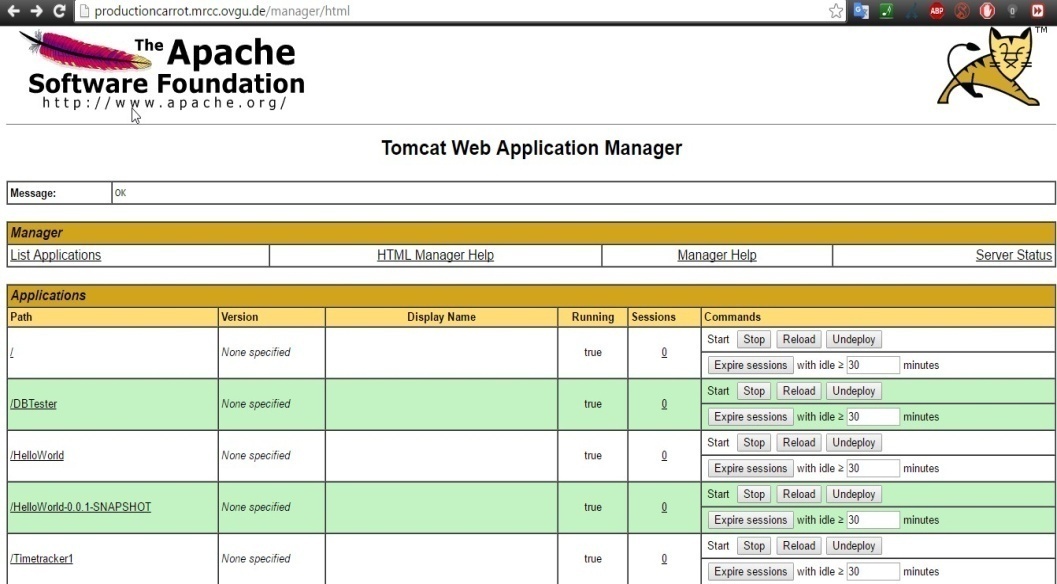
**Fig. 1.**select deploy after choosing the file

Files already present on the server can be directed directly to the Production server providing the URL of the WAR file or the Directory.



**Fig. 1.**Select deploy after providing the URL

Uploaded files are present directly on the home page of the production site.



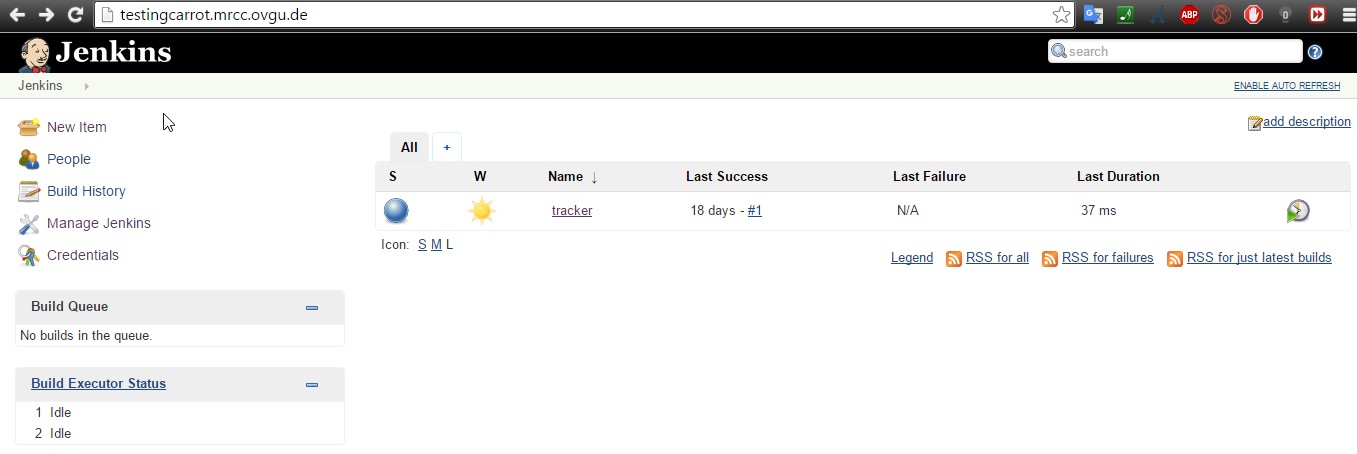
**Fig. 1.**production page

In the above picture DB Tester and the Timetracker1 are the .war files deployed on the Tomcat web application Manager.

* 1. testingcarrot.mrcc.ovgu.de:

Another instance of this VM is testingcarrot.mrcc.ovgu.de and is used for testing the time tracker application on server. Jenkins (ref. 4) is build on the testing server to run the continuous integration tests.

Project is imported directly from the GitHub repository. A build operation is performed after importing the project in to the server and build test results, the output can be seen in the console output window.



**Fig. 1.**testingcarrot.mrcc.ovgu.de dashboard

1. Continuous Integration and Continuous Delivery:

Continuous integration is a practice of merging all developer working copies several times a day, by validating each integration with an automated build.

Here we define continuous integration as having a build with unit tests that executes at every commit.

Continuous delivery is a software engineering approach in which teams produce software in short cycles ensuring that software can be reliably released at any time.By using continuous delivery in this project we aim to build, test and release software frequently by reducing the time and risk of delivery changes.

* 1. Usage of CI & CD

Maven gave basic support for build, version control and test cases. We defined the dependencies required for the time tracker application in pom.xml file. Here we used maven as a build tool for Jenkins. Jenkins which can build Apache maven projects by making calls to pom.xml file in our project, triggered new builds automatically when the new code is committed, unit tests are passed and then deployed it. A detailed usage of jenkins and maven in our project as a part of continuous integration and continuous delivery are explained further below.

1. Jenkins

Jenkins is a continuous integration tool and continuous delivery of the projects regardless of the platform we are working. It is an open source so that users can hold the continuous integrations and test their applications. We have installed Jenkins on the testingcarrot.mrcc.ovgu.de server to build the tests and deployment technologies. To get Jenkins on server following installation commands are performed on the Ubuntu server.

wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add

after giving the above command a message 'ok' is popped up and then below command is given again.

sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

An update command is performed before installing actual Jenkins command

sudo apt-get update

Following is the installation command of jenkins on the Ubuntu server

sudo apt-get install jenkins

Jenkins file is placed under etc/default as jenkins

etc jenkins.jpg

**Fig. 1.**Jenkins file

Problem:   
We had dealt a problem regarding with the port, as the port number of apache and Jenkins is same. To change the port number we opened the jenkins file under default file and in HTTP\_PORT section changes has been made from port 8080 to 8081.

port.jpg

**Fig. 1.**problem with port

To obtain the functionality of the Jenkins on the server we should run the run\_masterless\_puppet command under opt/de\_scripts.After running the above command Jenkins is perfectly installed on the server and you can see it by opening testingcarrot.mrcc.ovgu.de server.

To get the complete functionality of Jenkins, plugins should be imported, this can be done under manage plugins section by installing the available plugins. One important plugin is Git plugin. In this project we have

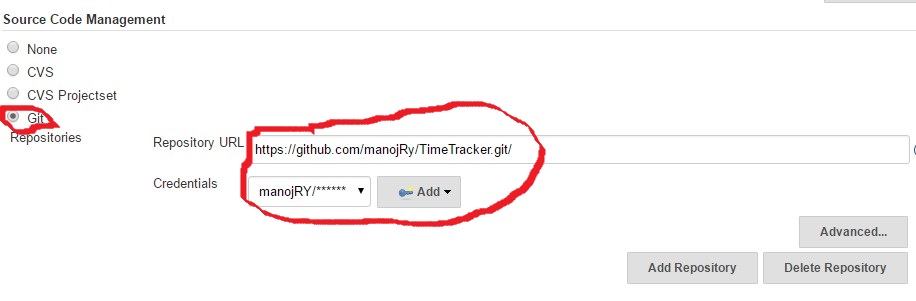
ported github plugin and git plugin such that we can import the project directly from Github version control system to Jenkins dashboard. After dealing with the plugins we have configured the jenkins under configure system.

Configuring the system includes the installation and providing the path to JDK, Git , Maven. JDK path is provided under the JDK category as /usr/lib/jvm/java-1.7.0-openjdk.. Followed by JDk, Git installations will be appeared and the path to the Git is given as var/lib/jenkins/plugins/git and then moving to next category CVS, the private key location is given as /var/lib/jenkins/.ssh/id\_rsa and the private key password is given for the access, save and apply the modified changes.



**Fig. 1.**Jenkins can be managed by the configuring the sections shown

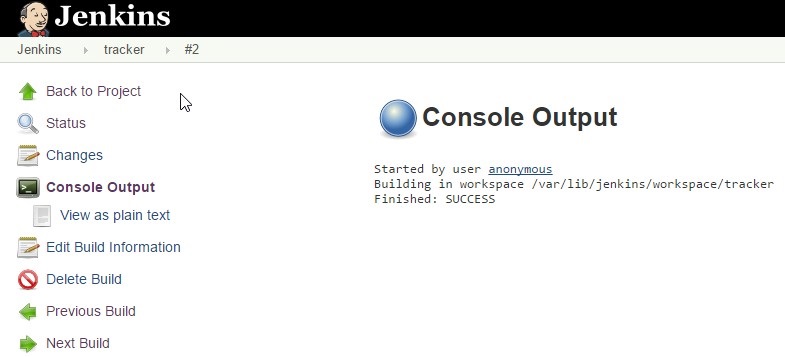
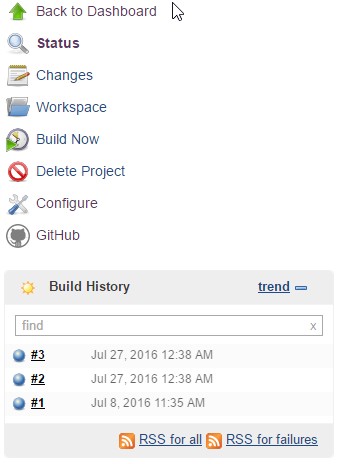
To build the time tracker project test cases we have clicked on new item and selected the free style project by giving the project name as time tracker. As timetracker project is ready we have configured it by providing the ssh key from our project in Github to Git subcategory under source code management, save and apply the modified changes.



**Fig. 1.** configuring Git

Building the project was the main task and we performed it by Build now option provided in the jenkins toolbar, after opening the project.

A build is scheduled and we can see that in Build history section below and the output is showed in console output section.

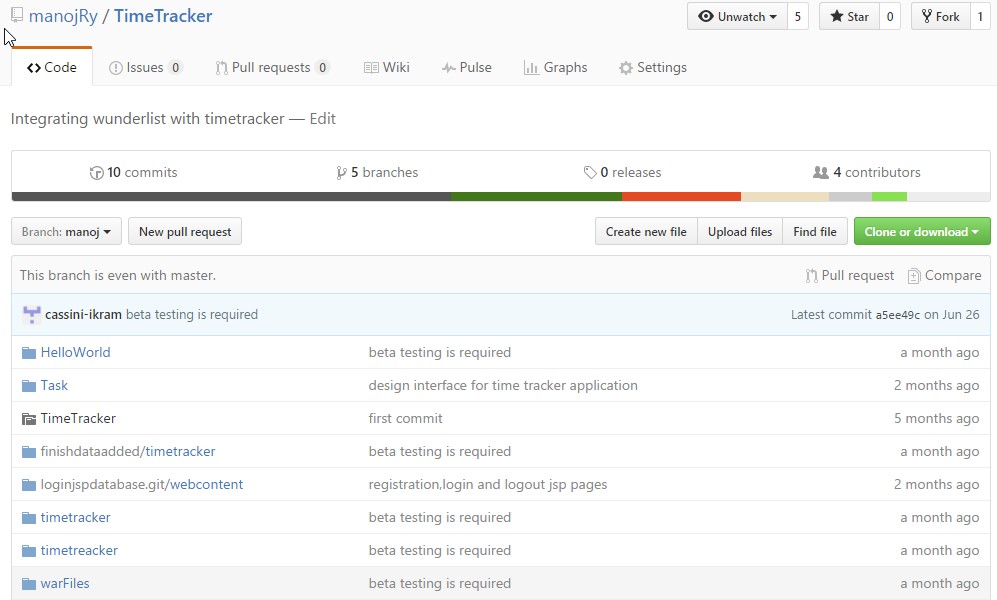


**Fig. 1.**Build now ,build history& console output showing the build as success

1. GitHub

GitHub helps the development team to work together and collaborate their code from anywhere under a single platform hosting as a version control.

For our developing team convenient we had used Github and created a repository as TimeTracker. Pull and Push requests were committed frequently as a part of working on the code in a daily basis.



**Fig. 1.** TimeTracker repository in Github

We have created 5 branches such that the developers and designers can perform their changes to the branch and then can apply the changes on the master. Initially we have cloned the repository to our local systems from command line and then the developed code is pushed to the Github repository if any changes has been applied.

1. Maven

Maven is a build automation tool. A complete build life cycle frame work is provided to developers by using Maven . Development team automates the project build infrastructure in less time by using standard directory layout and a default build life cycle. Project object model (POM) is an xml file, contains the project dependencies and plugins used while developing the project.

Further we have used Maven in our project as a build automation tool to achieve the dependent project build process.

Maven has been setup locally in developers work environment and code is developed by using Eclipse tool. The developed code is then converted as maven project and this can be done by making use of maven plugin available in Eclipse platform. A pom file is generated after converting the java project into maven project and the pom file consists of project dependencies.

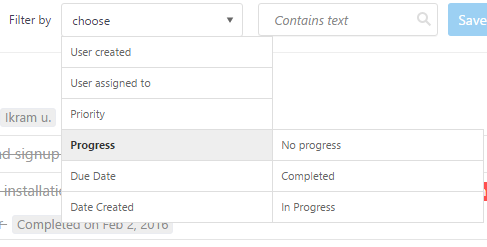


**Fig. 1.**Pom file of time tracker application

1. Scrum

According to scrum methodology work was divided into small pieces each and every task was made obvious to each team member. As we had decided that each sprint will be spanned on two weeks. After every two weeks, we used to plan our next sprint and review of previous sprint.

The tool which we had adopted, to support scrum process, was freedcamp.com. It’s a web application. Sprint backlog was maintained by adding tasks to freedcamp.com.



**Fig. 1.**scrum board

Scrum board functionality is available, by setting status of task. We can get tasks of particular phase that is “To Do”, “work in progress” and “Done” by filtering tasks on basis of progress.Scrum roles were also defined, each member who was working on this project was a team member. A scrum master, had a role to manage sprint backlog by adding new task to fulfill project requirements.

1. Diagrams notations

Rectangle represents process or task

Oval/circle connected to rectangle represents parameters

Oval represents operation

Arrow represents flow

Diamond represents condition

1. Basic Architecture

In basic architecture,there are three modules in Timetracking application.



**Fig. 1.**Time tracker application

* 1. Database

It will save data persistently into mysqldatabase, generated by wunderlist or timetracker application.

* 1. Timetracker

It is the main part of application. We can call it as controller of application. It has interface for users to interact with application. It controls the flow of data for synchronization of wunderlist with timetracking application.

* 1. Wunderlist

Timetracking application is synchronized with wunderlist by using wunderlistapi.

1. Operations

Prominent operations included in Time Tracking application are creating list, creating task, refreshing or synchronizing data and updating data in database.



**Fig. 1.**Operations

* 1. Create List

It will create new list in wunderlist application for same user by using wunderlistapi. After that it will update database.

* 1. Create Task

It will create task for selected list in wunderlist application by using wundrelist application. After that it will update database.

* 1. Refresh/Synchronize

It will get data from wunderlist and will compare it with data stored in database, changed and new data will be updated in database.

* 1. Start, Pause and Finish

It will update status and time information of task in database.

1. Request Processing

In Timetracking application requests from users, are handled by event handler. Event handler processes request in two steps.



**Fig. 1.**Request processing

* 1. Step 1

Request will be directed toward wunderlist by using wunderlistapi for getting particular data.

* 1. Step 2

Required data will be fetch from database by using DB Handler, for comparison. After comparing data, required updating will be done in database to data synchronized.

1. Refresh/Synchronize

It will fetch all lists from wunderlist for currently logged in user. At first it will get all completed tasks for each list. For each task it will check that task already exist in database or not. If not, then task will be added into database with finished status. If yes, then status of task in database will be checked that if status is finished or not. If finished, then no need to make any change. If not finished, then update status to finished and also total time will be calculated and updated in database.



**Fig. 1.**Refresh data

After processing completed tasks, it will get tasks which are not completed yet, from wunderlist for each list. For each task it will check that task already exist in database or not. If not, it will add task directly into database. If yes, then it will check that task is already started or not. If not, no need to do any kind of amendment. If yes, then it will calculate and update total time in database.

1. Start Task

For starting any task, “start task” event will be initiated. After initiating it will check that if task is already finished or not. If finished,then it’s not possible to start it again. If not finished, then it will check status of task that if it is active or not. If task will be active, then total time will be calculated and updated in database. If task will not be active, then current time will become start time of that particular task.



**Fig. 1.**Start Task

1. Pause Task

On initiating “pause task” event, it will check that if task is finished or not. If task is finished, then no need to make any change because it’s not allowed to pause the task which is already finished. If task is not finished it will check status of task. If task is active, then it will calculate total time for that task otherwise no need of calculation, just status of task will be updated.



**Fig. 1.**Pause Task

1. Finish Task

Processing of “finish task” event is almost similar to “pause task” event, it will check that if task is finished or not. If task is finished, then no need to make any change because it’s already finished. If task is not finished it will check status of task. If task is active, then it will calculate total time for that task otherwise no need of calculation, just status of task will be updated.



**Fig. 1.**Finish Task

1. Unique Identification

As this application is a web application. Any number of users, depending upon the limitation from server, can access this application at the same time. Also users don’t have to create account for timetracking application, only wunderlis account is enough for using this application. Figure a depicts the process of uniquely identifying the session for each user, so that particular tasks for particular users will be accessed.



**Fig. 1.**Unique identification

Timetracking application starts with “index.jsp” page, from this page user will be redirected towards wunderlist authentication process, with “client-id” and “redirect-url” parameters. User will provide credentials for wunderlist application. After successful authentication process, user will be redirected towards “timetracker.jsp” page of timetracking application, with “code” parameter. A request, to wunderlist, will be generated for getting “access-token” for user, with parameters “code”, “client-id” and “client-secret”. After successful processing by wunderlist, “access-token” will returned to “timetracker.jsp” page. That “access-token” will be used for uniquely identifying each session.

1. Mockito:

Mockito is a mocking framework helpful in creating mocks and spies in a simple and intuitive way, while at the same time providing great control of the whole process.

* 1. Unit Testing:

A unit test is a test related to a single responsibility of a single class, often referred to as the System Under Test (SUT). The purpose of unit tests is to verify that the code in an SUT works. A tested object usually talks to other objects known as collaborators. These collaborators need to be created so the tested object can be assigned to them in the test. To make unit testing simpler and allow control of all aspects of the execution context, it is useful to replace the real cooperating objects with their fake replacements called test doubles. They look like the originals, but do not have any dependencies to other objects. Test doubles can also be easily programmed with specific expectations, such as recording any interactions they've had. To make it clear for suppose in our time tracking application we need the database and wunderlist application access. So in order to be independent we should eliminate the dependencies i.e sometimes we come across a situation where we cannot have those dependencies at that time we use mock(dummy) objects and represent their behavior in those terms.

For this we use a lot of frameworks one among them is Mockito.

* 1. Steps for testing with mockito framework

1. Inject Mocks creates an instance of the class and injects the mocks

C:\Users\MANOJ 6\Downloads\13883931_10208224191216244_1386086079_n.png

1. that are created with the Mock. Here we created a instance for Tracker class and injects the mocks that are created with mock.

C:\Users\MANOJ 6\Downloads\13884399_10208224191256245_1008420628_n.png

1. Mock creates a mock. Here we created a mock object to be injected
2. Behavior or stubbing Mockito adds a functionality to a mock object using the methods when(). The following is the snippet

C:\Users\MANOJ 6\Downloads\13900813_10208224191296246_1702147363_n.png  
Here we've instructed Mockito to give a behaviour of data to the start method of timetrack and as a result, to return the value of 201672746 which is the date and time of the specific task i.e data. We have other functionality in mockito such as

### Verifying Behavior : . Mockito can ensure whether a mock method is being called with required arguments or not. It is done using the verify() method.

### Expectingcalls: Mockito provides a special check on the number of calls that can be made on a particular method.

### Exception handling:**.**

Mockito provides the capability to a mock to throw exceptions, so exception handling can be tested.

### Spying:**.**

Mockito provides option to create spy on real objects. When spy is called, then actual method of real object is called.

1. Black Box Testing:

Internal system design is not considered in this type of testing unlike white box testing. Tests are completely based on the requirements and functionality. There are several automation tools or third party services available in the market to do this black box testing. Here, we have used Selenium 2.53.0 version to check actual functional flow of our time tracking web application.

* 1. Selenium:

Selenium is an open source automated testing suite for web applications across different browsers and platforms. Selenium is not just a single tool but a suite of software’s, each catering to different testing needs of an organization. It has four components.

* Selenium Integrated Development Environment(IDE)
* Selenium Remote control(RC)
* Selenium Web Driver
* Selenium Grid
  1. **Selenium Web Driver:**

As we are developing time tracking web application, We have used the selenium web driver frame work to automate our tests. It checks the functionality of time tracking application not only on the local apache tomcat server but also on the actual server. We have initially created few test cases which are used to check how our web application responds in a procedural manner.

* 1. **Test Cases:**

A test case is a document which has sets of tests data, pre-conditions, expected results and post-conditions developed for a particular test scenario in order to verify the compliance against a specific requirement. We have created few test cases to check the functionality of Registration screen, Log in screen and different actions of time tracking application. We have included few of them below.

|  |
| --- |
| **Test Case ID**: TT\_01  **Test Priority (Low/Medium/High):** Medium  **Module Name:** Log In screen.  **Test Title:** Verify log in with valid user name and password. |

|  |
| --- |
| **Pre Conditions:**User has a valid user name and password. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Test Steps | Test Data | Expected Result | Actual Result | Status (Pass/Fail) |
| 1 | Navigate to given URL | http://localhost:8087/timetracker/index.jsp | User should be able to log in | User is navigated to | Pass |
| 2 | Provide a  valid user name | krishh |  | dashboard with |  |
| 3 | Provide valid password | 2580 |  | successful log in |  |
| 4 | Click on log in button |  |  |  |  |

|  |
| --- |
| **Post Conditions:**  User is validated with database and successfully login to account. The account session details are logged in database. |

|  |
| --- |
| **Test Case ID:** TT\_02  **Test Priority (Low/Medium/High):**High  **Module Name :** Time Tracking Application.  **Test Title:** Create a task and monitor it. |

|  |
| --- |
| **Pre Conditions:**User has a task to create . |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Steps | Test Data | Expected Result | Actual Result | Status (Pass/Fail) |  |
| 1 | Navigate to given URL | http://productioncarrot.mrcc.ovgu.de/timetracker/timeTracker.jsp?state=RANDOM&code=e6216e03560a3c39b352# | User should be able to create a task and monitor it. | User is  navigated to dashboard  and create a new task and start it, pause it after some time and finish it. | Pass |  |
| 2 | Provide a  valid task name | Movies |  |  |  |  |
| 3 | Start the task |  |  | . |  |  |
| 4 | Pause the task |  |  |  |  |  |
| 5 | Finish the task |  |  |  |  | |
| 6 | Refresh the task |  |  |  |  | |

|  |
| --- |
| **Post Conditions:** User is created some tasks using time tracking application and monitor them and these tasks are presented in the Wunderlist API. |

1. Maven in Selenium:

As Maven is building management tool, It helps us to manage our selenium project’s build easily. It manages our selenium test project’s build compilation, documentation and other project related tasks itself. We have been using maven to create right project structure and mange .jar files in the project’s build path etc..

Maven uses POM.xml configuration file which kept all our project configuration related information. For selenium, We need to provide selenium web driver version related information POM.xml file and then It will download all required .jar files automatically and store it in local repository called m2. Whenever we change version of selenium web driver then we need to simply modify version in POM.xml file only.

1. References:
2. [1]http://www.slashroot.in/sites/default/files/Working%20of%20Puppet%20Configuration%20Mangement%20tool.png