**A Test Driven Development approach for Student Test Evaluation   
System using Continuous Integration and Continuous Deployment   
Pipeline.**

# ABSTRACT

During the process of Software Application development, times may occur when there is lack of communication between Development, Testing and Operations teams. Due to this problem most of the development time is wasted. We are making a delivery pipeline for the continuous development, continuous integration and continuous deployment of an application (Student Evaluation System) with the help of a test driven development approach to save the time.

# TABLE OF CONTENT

Candidate’s Declaration ….……………………………………………………………...……2

Acknowledgement……………………………………………………………………………...3

Abstract…………………………………………………………………………………………4

Introduction……………………………………………………………………………………..6

Problem Statement …………………………………………………………………………….11

Literature Review…………………………………………………………………….………...11

Objective…………………………………………………………………………………….....16

Methodology……………………………………………………………………………….......16

Screenshots…………………………………………………………………….…………….....18

Results & Future Scope……………………………………………………………………...…28

References ……………..…………………………………………………………………..…..28

Annexure………………………………………………………………………………….……29

# INTRODUCTION

**Continuous integration** is a practice that encourages developers to integrate their code into a main branch of a shared repository early and often. Instead of building out features in isolation and integrating them at the end of a development cycle, code is integrated with the shared repository by each developer multiple times throughout the day. The idea is to minimize the cost of integration by making it an early consideration. Developers can discover conflicts at the boundaries between new and existing code early, while conflicts are still relatively easy to reconcile. Once the conflict is resolved, work can continue with confidence that the new code honors the requirements of the existing codebase.

When a software is developed by meticulously following the techniques of continuous integration the developers are supposed to merge or integrate their branches frequently, where the frequency can be several times a day. Continuous Integration encourages us to avoid the strategy of developing different modules and features of a project in isolation and to integrate the complete code at the end of the development cycle. The basic approach behind the strategy of Continuous Integration is to reduce the cost of integration of code by applying the process of integration at an early stage of development and that too quite frequent, so that all the issues which are suspected to occur in the project can be avoided and can be considered at an early stage of development. The conflicts which can occur between the boundaries of new and existing code, can be discovered by the developers at an early stage of development, as at that time it is relatively easy to reconcile. Once all the conflicts are resolved successfully, the developers can go on with the further development of the code with the complete confidence that the new code which they will be writing, will show no conflicts and will respect the existing code base. The Integration practices with respect to the Continuous Integration helps us ton automate the build process of the application and to automate the testing process. When a developer merges the code with the main branch or on the basis of the configuration mentioned by the user at the time of pushing the code into the Version Control System, an automates process is kicked off which initiates the build process of the code written the repository of the version control system. At the time of building the repository, the automated test suites are executed and automation testing is performed. If the test cases fails in that case the build also fails and the team is notified regarding the failure of the build, so that the team can work together to fix the build.

The main purpose of the Continuous Integration is to simplify the process of integration of different branches of the repository at an early stage of development to reduce the integration cost and to fix the defects as early as possible in the development cycle. It makes sure that the system we are trying to build is robust, automated and fast.

The process of Continuous Integration leads us to the process of Continuous Delivery. Using Continuous Integration the development team ensures that they can reduce the integration cost by integrating the code branches of the repository at an early stage of development, whereas in the process of Continuous Delivery, the development team tries to maintain a deployable code at the main branch i.e.; the production ready code. It ensures us that the code base is always in a deployable state , which makes the releasing of code as an unremarkable event, as the new releases can be brought to the production environment without any complicated coordination or the late stage testing. (LINKEDIN, 2018)

Continuous Delivery make the use of the deployment pipeline of which the process of Continuous Integration is an essential part. A deployment pipeline is an automated system that executes the test cases on the build increasingly and in a sequence of stages. Continuous Delivery picks up where the Continuous Integration leaves in the deployment pipeline. When the code is passed on for building and the test cases are performed on the code then the test cases are either passed or the test cases are failed which raises an alert to the development team to perform the fix operation on the build. This process occurs at the each stage of development. Once the build is passed then the build is deployed to the environment which mirrors the production environment as exact as possible. This deployment helps us to test the build, the deployment process and the environment of the production altogether. This pipeline ends when the build is deployed to the production environment finally. (Linkedin, n.d.)

**Continuous deployment** is an extension of continuous delivery that automatically deploys each build that passes the full test cycle. Instead of waiting for a human gatekeeper to decide what and when to deploy to production, a continuous deployment system deploys everything that has successfully traversed the deployment pipeline. Keep in mind that while new code is automatically deployed, techniques exist to activate new features at a later time or for a subset of users. Deploying automatically pushes features and fixes to customers quickly, encourages smaller changes with limited scope, and helps avoid confusion over what is currently deployed to production.

Continuous deployment also allows organizations to benefit from consistent early feedback. Features can immediately be made available to users and defects or unhelpful implementations can be caught early before the team devotes extensive effort in an unproductive direction. Getting fast feedback that a feature isn't helpful lets the team shift focus rather than sinking more energy into an area with minimal impact. The process of Continuous Integration leads us to the process of Continuous Delivery. Using Continuous Integration the development team ensures that they can reduce the integration cost by integrating the code branches of the repository at an early stage of development, whereas in the process of Continuous Delivery, the development team tries to maintain a deployable code at the main branch i.e.; the production ready code. It ensures us that the code base is always in a deployable state , which makes the releasing of code as an unremarkable event, as the new releases can be brought to the production environment without any complicated coordination or the late stage testing. (htt)

Continuous Delivery make the use of the deployment pipeline of which the process of Continuous Integration is an essential part. A deployment pipeline is an automated system that executes the test cases on the build increasingly and in a sequence of stages. Continuous Delivery picks up where the Continuous Integration leaves in the deployment pipeline. When the code is passed on for building and the test cases are performed on the code then the test cases are either passed or the test cases are failed which raises an alert to the development team to perform the fix operation on the build. This process occurs at the each stage of development. Once the build is passed then the build is deployed to the environment which mirrors the production environment as exact as possible. This deployment helps us to test the build, the deployment process and the environment of the production altogether. This pipeline ends when the build is deployed to the production environment finally.

The **Continuous delivery pipeline** is an automated set of processes that use tools to compile, test, and deploy code for new software features. The purpose of a continuous delivery (CD) pipeline is to create a continuous management and release setting where bugs, compatibility issues, and security breaches are identified and fixed as early as possible. Also called the “**Deployment pipeline**,” the CD pipeline is the vehicle that drives iterative software application development. It breaks down the strategy of continuous integration and continuous delivery into stages so development teams can gather results incrementally, at each stage of the build. Developers are then able to run tests and make other assessments without having to wait until the entire workflow is completed.

Continuous Delivery is an initial process for the Continuous Deployment which deals with the automation of the deployment process by removing the need for the human intervention for making decisions regarding choice of what and when to deploy to the production environment. It deploys everything that has successfully passed the delivery pipeline. Though the deployment system deploys everything which has passed the delivery pipepline, still we can implement new features for all the users or a subset of users of the application at any point of time later in the deployment process. It helps us to deploy the new features at the runtime for the customers quickly and removes all sort of confusions and bugs which are present in the current release of the system.

The Continuous Deployment can cause be a source of some of sort of tension for the organisations who are worried about redesigning the workflow of the automation system for the organization. The trade-off offered by the automated deployment is sometimes considered to be too dangerous for the payoff they provide.

Continuous Deployment always follows the best practises and helps us to implement the limited testing into the production as the code is already built and tested at the time of building the code in the environment similar to the production. Developers must take the complete responsibility of the code which they are about to deploy finally into the production environment and must ensure that their code is well designed and tested. This gives the development team the complete authority to decide that what and when to commit to the main branch and what and when to deploy to the production environment.

Feedback is must for any organization whether it is an IT organization or not. Continuous Deployment helps us to get the early feedbacks from the users on the application which is deployed. Features can be immidately made available to the users. This will help the team to catch most of the defects and exceptions into code at an early stage, before they tend to move the development into an uproductive direction. This is done because of the defects and bugs are not caught at an early stage then this might lead us to a bigger issue at the later stages in the development lifecycle of the application.

**Test Driven Development:**

Contrary to the conventional approach of software development where the code is developed first and then the testing is performed on it to check the legibility of the code that is written, the Test Driven Development asks us to divide the complete project into smaller stories or epics and write the test cases for a particular module against its requirements. Once the test cases are made, then the development process it initiated. Once the developed code passes all the test cases which have been written for the module, then the code which is developed is accepted. The testing of the developed code is automated with the help of the Continuous Integration engines which helps us to build the developed code and perform the testing on it in an environment which is similar to the production environment. This approach of software development helps the developer to design efficient code and to build confidence for the developed code.

American software engineer Kent Beck, who is credited with having developed or "rediscovered" the technique, stated in 2003 that TDD encourages simple designs and inspires confidence. (Beck, 2003 )

Unit tests are automated.

Code

Refactors

Unit tests

**Add Test:** Developer must have the clear understanding of the requirements and the specifications of features which he is supposed to develop the code for. To get a clear cut understanding of the requirement and specifications, the developer must be well versed with the use cases and the user stories and then decide that exceptional conditions. Once the requirements and the specifications are clear to the developer, then he can start writing the test cases for those features. These test cases in fact can be the improved version of the previously written test cases.

This development technique is actually different from the traditional unit testing approach as in this case we are focus on the requirement for writing the test cases rather developing the code.

**Run all the test and see if the new test fails:** After writing all the test cases execute those tests. As expected the test cases will fail as the code for which the test cases are written is not written yet.

**Write the code:** Once the test cases are written then start developing the code for the project. The code should be written now with the approach of passing all the test cases written for the particular feature. The test cases must be passed in elegant way. At this point of development, the code is just written in order to pass the test suites and the code must not be written beyond the functionality. The code will be improved at step 5.

**Run tests:** Once the code is developed, again the test suites must be executed to check that the code written is working perfectly and has not degraded the previously written code. The code written must not break the already written code and must work fine with it.

**Refactor code:** This is the phase when the code is refactored and arranged properly within the repository. The code has grown till now and it must be cleaned properly as we must try to ignore the duplicacy of the code. The code is moved and arranged to the location where the code belongs more logically. Object, class, module, variable and method names should clearly represent their current purpose and use, as extra functionality is added. As features are added, method bodies can get longer and other objects larger. They benefit from being split and their parts carefully named to improve readability and maintainability, which will be increasingly valuable later in the software lifecycle. Inheritance hierarchies may be rearranged to be more logical and helpful, and perhaps to benefit from recognized design patterns.

**Repeat:** Starting with another new test, the cycle is then repeated to push forward the functionality. The size of the steps should always be small, with as few as 1 to

10 edits between each test run. If new code does not rapidly satisfy a new test, or other tests fail unexpectedly, the programmer should undo or revert in preference to excessive debugging.

# PROBLEM STATEMENT

Existence of gap between the Development and Operations team during the course of Development of a Web Application.

# LITERATURE REVIEW

**Containerization:**

Containerization is a technology which enables enterprises to run suspicious applications in a different environment without influencing different procedures. Intended to increment enterprise security, they additionally streamline enterprise IT tasks. Containers are an answer for the issue of how to motivate programming to run dependably when moved starting with one figuring environment then onto the next. This could be from an engineer's workstation to a test environment, from an organizing environment into creation, and maybe from a physical machine in a server farm to a virtual machine in a private or open cloud. A holder comprises of a whole runtime environment: an application, in addition to all its conditions, libraries and different pairs, and setup records expected to run it, packaged into one bundle. By containerizing the application stage and its conditions, contrasts in OS disseminations and hidden infrastructure are disconnected away. A holder might be just many megabytes in size. Compartment technology isn't new; it has been incorporated with Linux as LXC for more than 10 years, and comparative working framework level virtualization has likewise been offered by FreeBSD correctional facilites, AIX Workload Segments and Solaris Containers. (htt5)

**Containerization vs Virtualization**

Containers give separated runtime environments to applications: the whole client space environment is solely displayed to the holder, and any progressions to it don't affect other containers' environments. To give this seclusion, a blend of OS-based components is utilized: Linux name spaces are utilized for disengagement and checking instrument. Document framework mounts characterize what records are open to the holder. cgroups characterize asset utilization of containers. Still all containers share a similar OS part which can understand memory impression efficiencies when indistinguishable libraries are utilized by different containers.

With framework virtualization, the hypervisor gives a full virtual machine to a visitor: the whole OS picture including the part is currently devoted to the virtual machine. CPU virtualization is utilized to furnish every visitor with a select perspective of a full framework environment, and these instruments likewise guarantee seclusion from different visitors. Hypervisor-based administration of virtual CPUs, memory and

I/O gadgets is utilized to characterize asset utilization of visitors. (htt6)

**Different types of Containerization tools:**

* **Docker:** Solomon Hykes started Docker in France as an internal project within dotCloud, a platform-as-a-service company, with initial contributions by other dotCloud engineers including Andrea Luzzardi and Francois-Xavier Bourlet. Docker is a computer program that performs operating-system-level virtualization, also known as "containerization". Docker is utilized to run programming bundles called "containers". Containers are secluded from one another and package their own application, apparatuses, libraries and arrangement documents; they can speak with one another through very much characterized channels. All containers are controlled by a solitary working framework part and are consequently more lightweight than virtual machines. Containers are made from "pictures" that determine their exact substance. Pictures are frequently made by consolidating and changing standard pictures downloaded from open archives.

Docker is a platform used to develop, ship and run applications. Docker provides isolation between running application and the infrastructure with the help of containers. Containers are light weight as they are run directly from within the host machine's kernel. Next, we have docker engine which has 3 major components: (doc)

1. A Server which is a daemon process.

2. A Rest API which is used as an interface between programs and the daemon process.

3. A Command line interface client.

Micro services are small autonomous services that work together to fulfill a business requirement. It is a method of developing software applications as a suite of small modular services, deployable in which services run unique process and communicates through a well-defined, lightweight mechanism to serve a business goal.

* **Kubernetes:** Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community**.** Kubernetes defines a set of building blocks ("primitives"), which collectively provide mechanisms that deploy, maintain, and scale applications based on CPU, memory or custom metrics. Kubernetes is loosely coupled and extensible to meet different workloads (htt9). This extensibility is provided in large part by the Kubernetes API, which is used by internal components as well as extensions and containers that run on Kubernetes. The basic scheduling unit in Kubernetes is a pod. It adds a higher level of abstraction by grouping containerized components. A pod consists of one or more containers that are guaranteed to be co-located on the host machine and can share resources. Each pod in Kubernetes is assigned a unique Pod IP address within the cluster, which allows applications to use ports without the risk of conflict. Within the pod, all containers can reference each other on localhost, but a container within one pod has no way of directly addressing another container within another pod; for that, it has to use the Pod IP Address. An application developer should never use the Pod IP Address though, to reference / invoke a capability in another pod, as Pod IP addresses are ephemeral - the specific pod that they are referencing may be assigned another Pod IP address on restart. Instead, they should use a reference to a Service, which holds a reference to the target pod at the specific Pod IP Address. (htt7)

**Travis:**

Travis CI introduces automated builds which makes it easier for project developers to develop their projects while testing it side by side. Whenever the developer pushes their code in their respective repository online then the builds are automatically triggered. Travis CI opens up the door for CI which means continuous integration which basically means to develop code and test it without much overhead borne by the user. This platform is free to use by all open source projects that are hosted upon GITHUB. To activate the automatic builds on the TRAVIS a configuration file called travis.yml has to be made which holds information about what branch to use for the pipeline.

**Java Spring Framework:**

Spring is a lightweight java framework. It can be thought of as a framework of frameworks because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems. The Spring framework comprises several modules such as IOC (Inversion of Control),

AOP (Aspect Oriented Programming), DAO (Data Access Object), Context,

ORM (Object Relation Model), WEB MVC etc. (htt8)

**Inversion Of Control (IOC) and Dependency Injection:** These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain. Let's understand this with the following code:

class Employee{

Address address;

Employee(){

address=new Address();

}

}

In such case, there is dependency between the Employee and Address (tight coupling).

In the Inversion of Control scenario, we do this something like this:

class Employee{

Address address;

Employee(Address address){

this.address=address;

}

}

Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment. In spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

**Advantage of Dependency Injection:**

* makes the code loosely coupled so easy to maintain.
* makes the code easy to test.

**JUnit:**

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit. Testing is the process of checking the functionality of an application to ensure it runs as per requirements. Unit testing comes into picture at the developers’ level; it is the testing of single entity (class or method). Unit testing plays a critical role in helping a software company deliver quality products to its customers. Unit testing can be done in two ways − manual testing and automated testing.

JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit.

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

Features of JUnit:

- JUnit is an open source framework, which is used for writing and running tests.

- Provides annotations to identify test methods.

- Provides assertions for testing expected results.

- Provides test runners for running tests.

- JUnit tests allow you to write codes faster, which increases quality.

- JUnit is elegantly simple. It is less complex and takes less time.

- JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.

- JUnit tests can be organized into test suites containing test cases and even other test suites.

- JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

**Version Controlling:**

Version controlling is a system which was specifically designed to record every changes that were made on a file at any instance of time, so that it could be reverted back to the earlier state if required. There are various types of version controlling systems:

1. Local Version Controlling System

In this system, changes to a files i.e. difference between two files also called patches are stored in a simple database. At the time of recreating file to any instance, the patches are added to the file to that point of time to recreate it. It was the 1st generation of version control systems.

1. Centralized Version Controlling System

This system was developed to help developers collaborate over other systems. A common server is there which holds the versioned files, users around the remote locations check’s out the files from that central server. It was the 2nd generation of version control systems.

1. Distributed Version Controlling System

In this system, Users checks out the whole repository from the server to their computer along with all previous commits. If any server failure occurs, then the systems that were collaborating with the servers can be used to restore back the server repositories. It was the 3rdgeneration of version control systems.

The first ever version controlling system was developed in 1972 at BELL LABS where UNIX was created, it was called SCCS. Only UNIX systems were able to use it at first and they worked only with source code files.

**GIT:**

GIT was developed by Linus Torvald in 2005. It was aimed to be a distributed version control system. It’s fast, efficient and has a branching system. Git stores the data by taking a series of snapshots of the miniature filesystem. Every time someone commits snapshots of what all files look like is taken and a reference to that particular snapshot is stored. This is what differentiate GIT with other version controlling systems. To maintain the integrity of data GIT uses checksum. Data stored is Checksummed and then cross referenced again by the checksum. Therefore GIT is able to detect if corruption on file occurs. GIT uses the SHA-1 hash function. Every thing in GIT is stored as hash values of the data instead of just file names.

In GIT there are three states for a file: modification, staging and commit.

* Modifications means the changes done to a file is not yet stored in database.
* Staging means the modified file has been marked to go for the next commit point.
* Commit is to safely store data in the database.

Git directory stores some metadata and objects corresponding to the repository. These objects and metadata are the ones that is copied when a repository is cloned on another system.

When checking out a repository, a working tree is formed where files are fetched from the compressed database of GIT and placed on our disk to modify it.

Staging area is basically a file itself, found in the GIT directory, whose purpose is to store information as to what goes in to the following commit.

Basic workflow of GIT is something like this:

1. Modify some file in the tree.
2. Selectively Stage the changes for making the commits.
3. Perform the commits, so that snapshots can be stored in the database.

GIT is accessible in many ways – command line, graphical user interface(desktop application as well as browser application).

GIT supports tagging feature as well, which grants it the ability to tag some parts of the file as important. Basically they are used for marking release points.

The feature which gives GIT it’s own unique standing is the branching and merging technique.

Git allows their user to have multiple branches of particular repository which are totally independent on each other. Creating, deleting and merging those branches are done in seconds. Because of this, developers can form two separated branches entirely dedicated for just testing and development phases.

# OBJECTIVE

Bridging the gap between Development, Testing and Operations team using Continuous Integration and Continuous Development pipeline.

# METHODOLOGY

**Module 1: Gather requirements for the web application.**

Requirements gathering is the most important phase of application design. It is said that a good start is half a job done. This phase is that start for the software development lifecycle. If this phase is not executed well then it will affect all the phases following it.

To gather the requirements for this project we have tried to communicate with the first year students of Btech Computer Science and asked them what kind of software do they need which can help them to get good grades. We have also circulated a google form to get their feedbacks and after performing such rigrous operations of requirement gathering we came to the conclusion that we can develop a website which help them to prepare for the online quiz and get good grades.

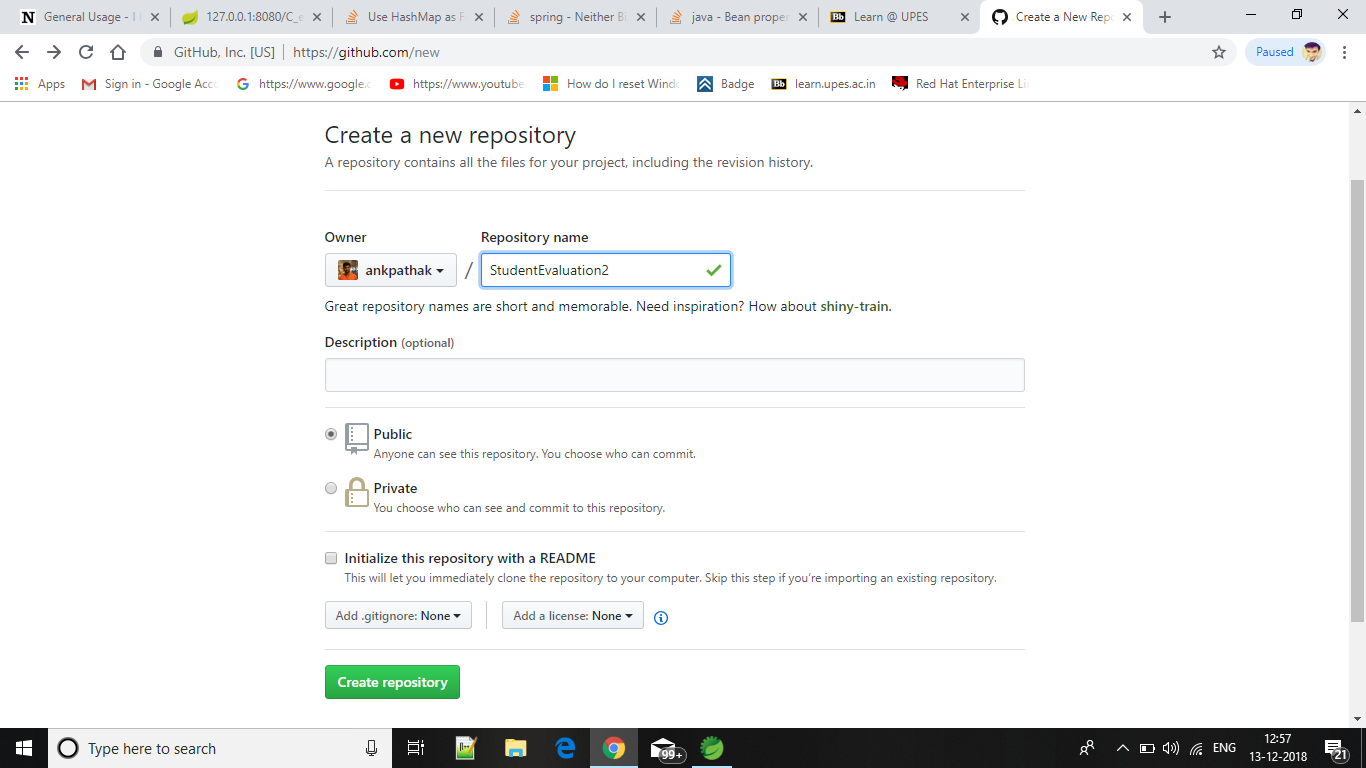
**Module 2: Setting up the Deployment pipeline**

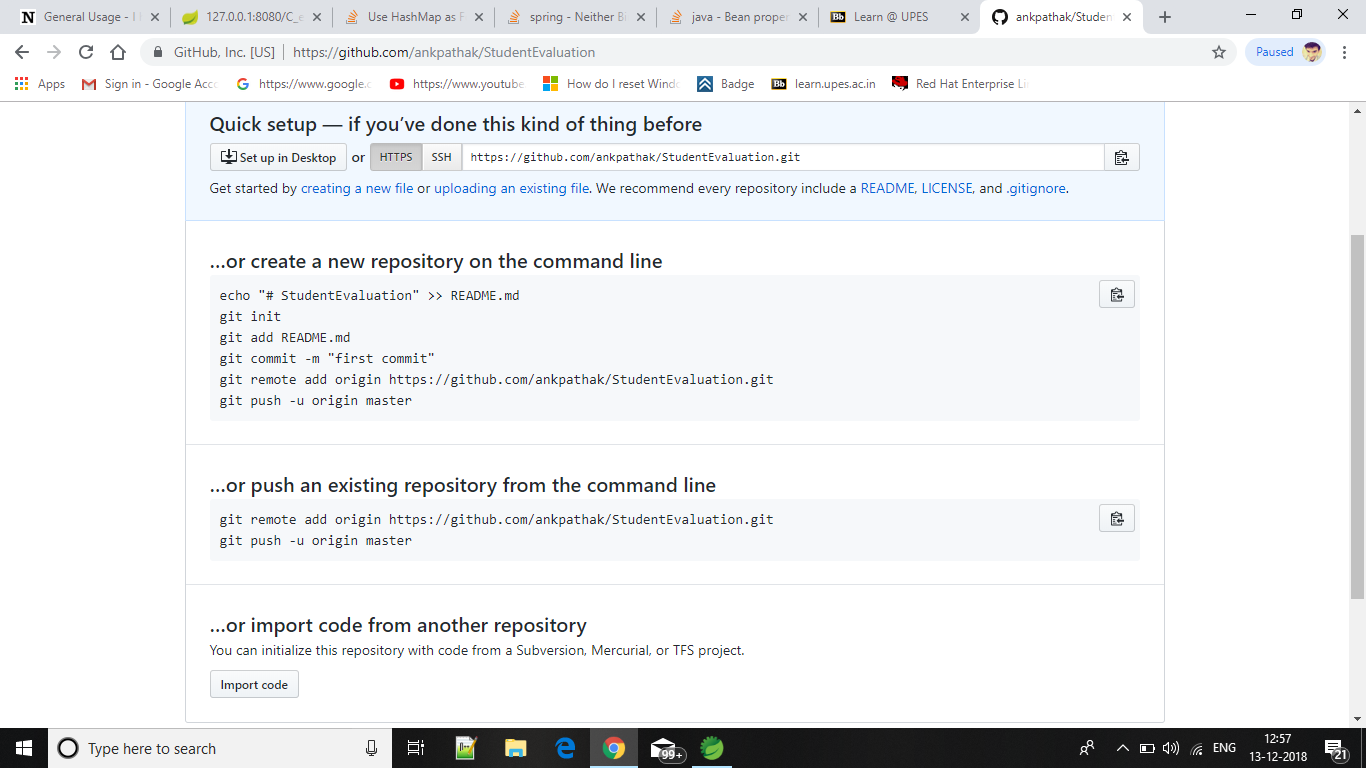
Deployment Pipeline is a path composing of the tools and software which are used during the development, testing and deployment phase of software development. The different tools can be Integrated Development Environment, Version Control System, Continuous Integration Engine, Cloud Deployment Platform, Code Building tool and Development Framework.

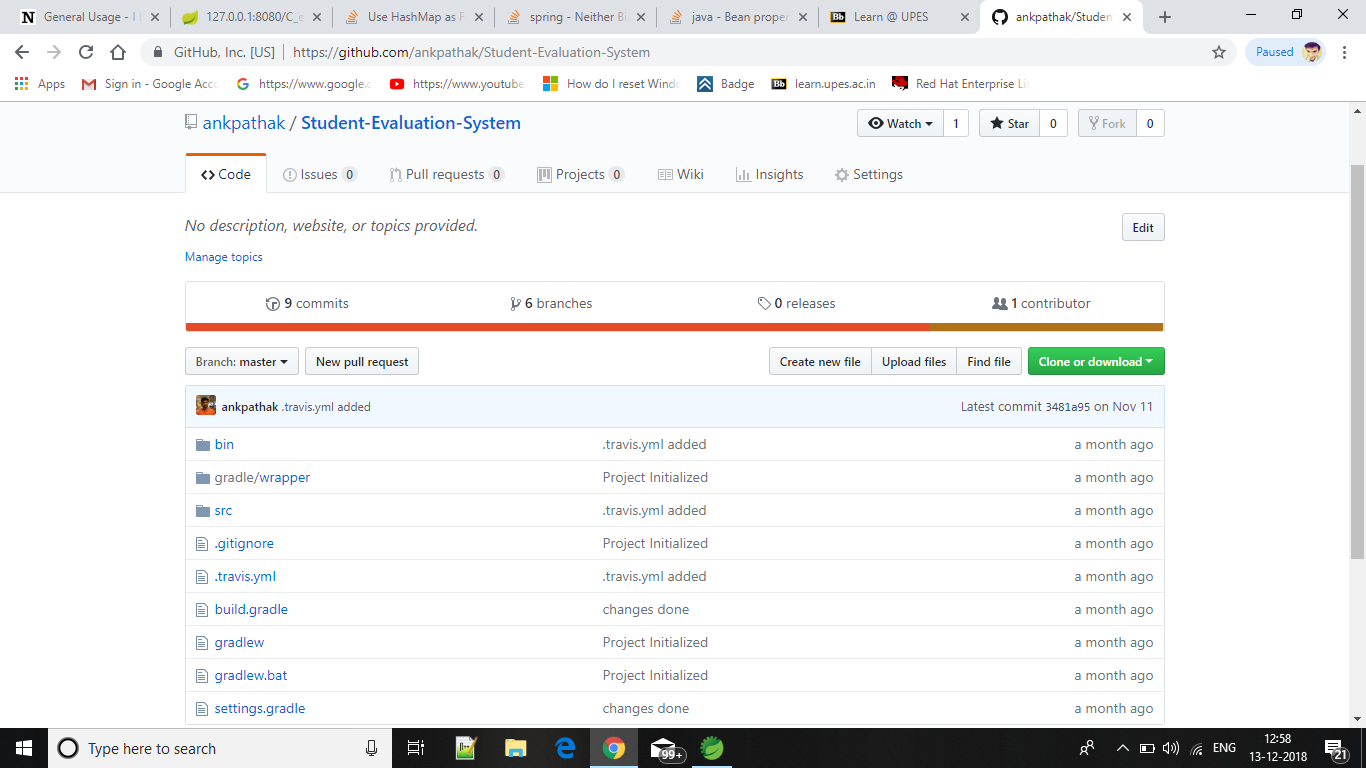
The tools which are used for setting up the Deployment Pipeline for our project are:

1. **Spring Tool Suite (Integrated Development Environment for Java based Spring applications)**
2. **Git, Github, Git Kraken (Version Control System)**
3. Github**:** Github Inc is a version control system, which is web hosted and provides us different functionalities like version controlling, bug tracking, feature requests, task management, and wikis for the project. Git is a client for the github. It is a distributed version control system for the systems and can be used for managing the code and its versions at the local system level. Git Kraken is nothing but a GUI based client for the git repositories clone to the local system and helps us in managing the different commits and branches of github, providing us a GUI based representation for the flow of code.
4. Setting up git repository:

The first and foremost step in setting up a deployment pipeline is to create a repository on the github and to clone it on our systems.







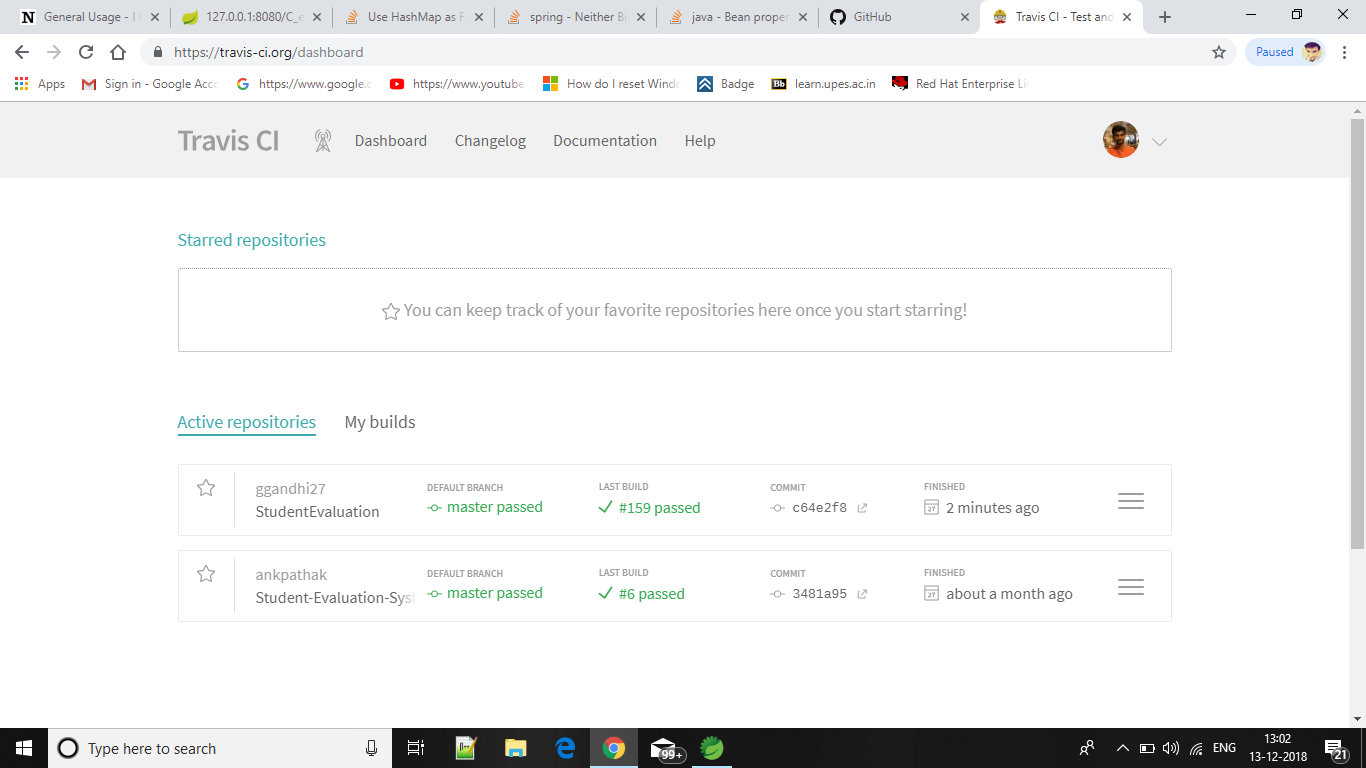
Our repository can be cloned to the system with the help of following GIT command:

*git clone* [*https://github.com/ggandhi27/StudentEvaluation.git*](https://github.com/ggandhi27/StudentEvaluation.git)

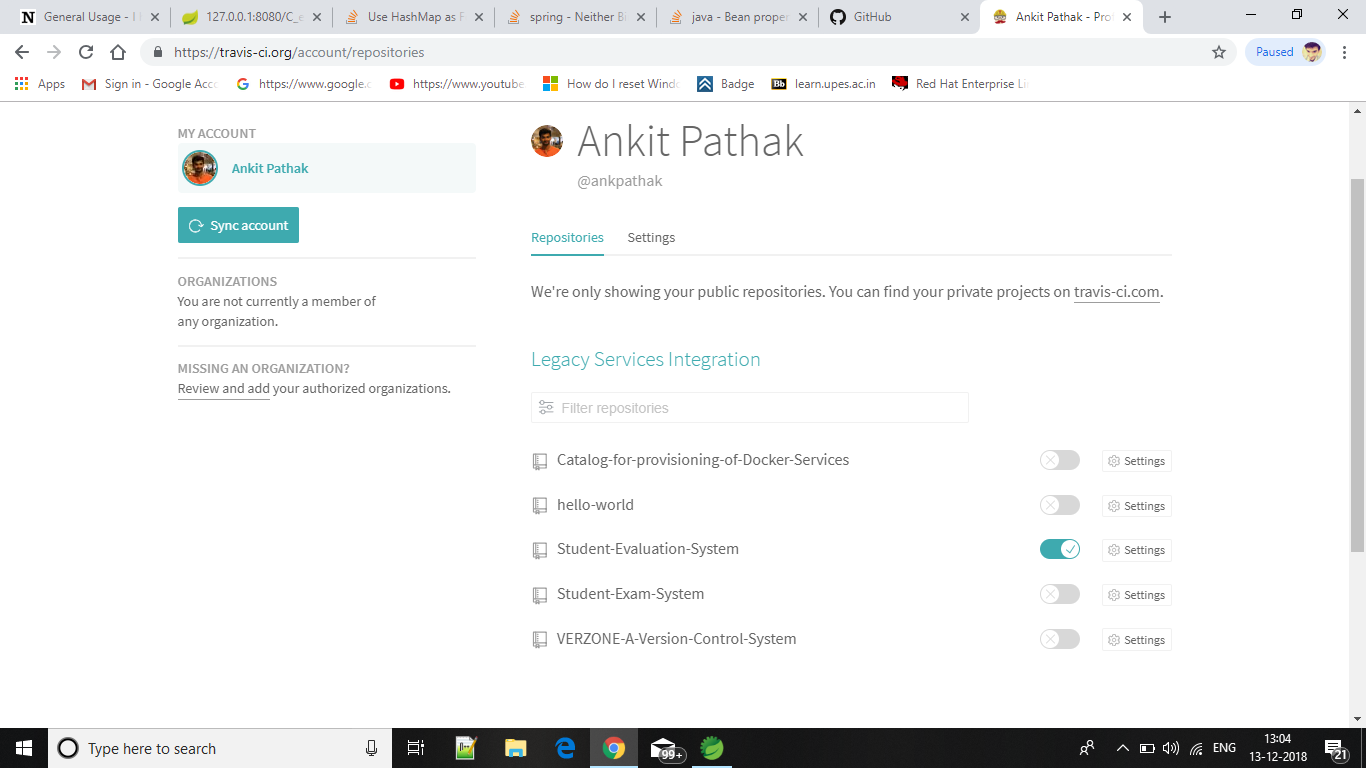
1. **Travis (Continuous Integration Engine)**

Travis is a free open source, hosted, distributed Continuous Integration tool which provides us the service of building and testing the code which is hosted on github. To initiate the building of the application on the Travis Engine, we have to add a configuration file named ".travis.yml" along with the code uploaded to the github repository.

After creating the github repository, we have to login to the Travis-ci.org website, and link our github account to Travis. One sign up on Travis with the help of their github account as well**.**

****

After linking the github account with the Travis account, go to the travis dashboard and activate the github repository for the Travis to build.



Once the repository is activated for performing the build, we have to create a file name ".travis.yml" in the git repository which we have created.

Just mention

"""

language: java

jdk: oraclejdk8

"""

In the .travis.yml file and save the file. The above two lines of code tells the Travis CI engine that the repository contains the code written in java programming language and the jdk (Java Development Kit) version which will be required for compiling and executing the code will be oraclejdk8. These two attributes for the file may vary according to the language and the compiler versions which will be required for that particular language suite.

Once the file is created and saved in the repository, create commit in the git repository and push the code to the github.

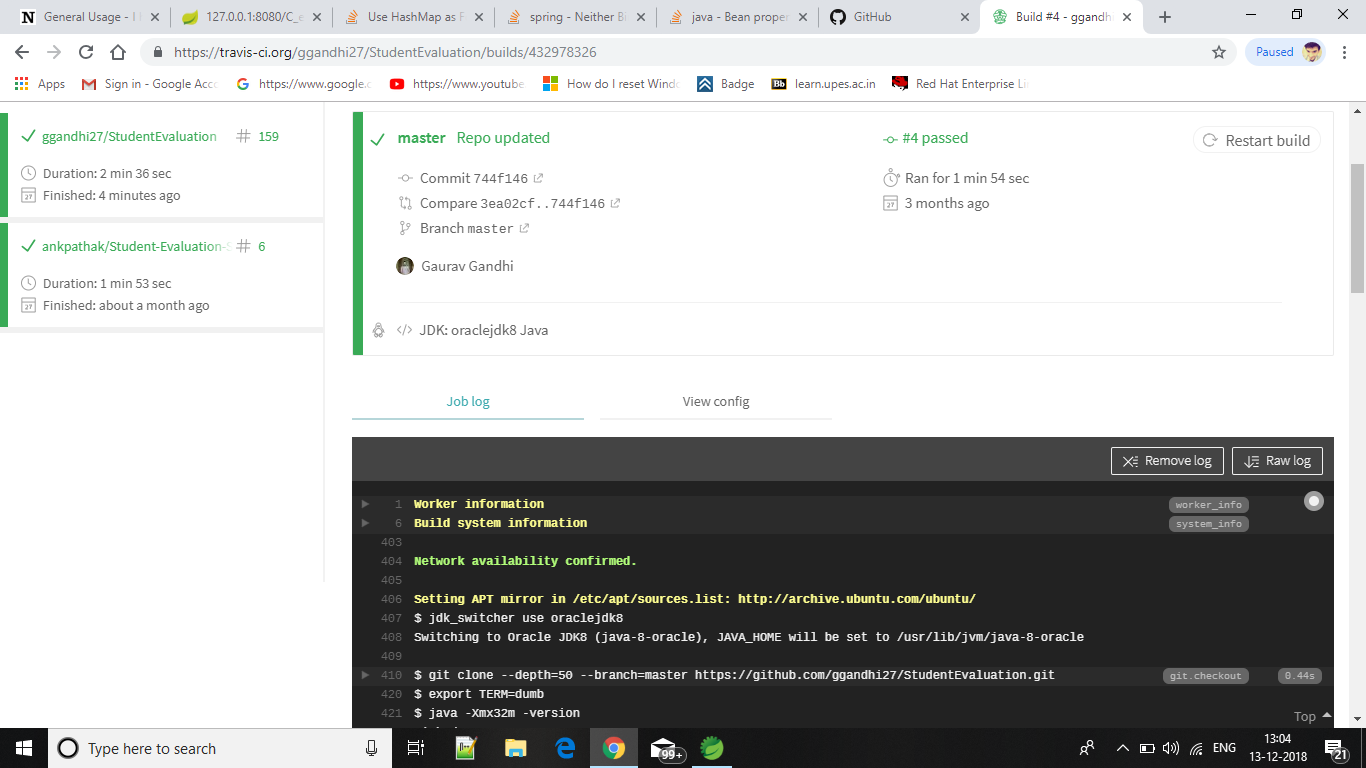
Commands for creating the commit and pushing the code to the git hub are.

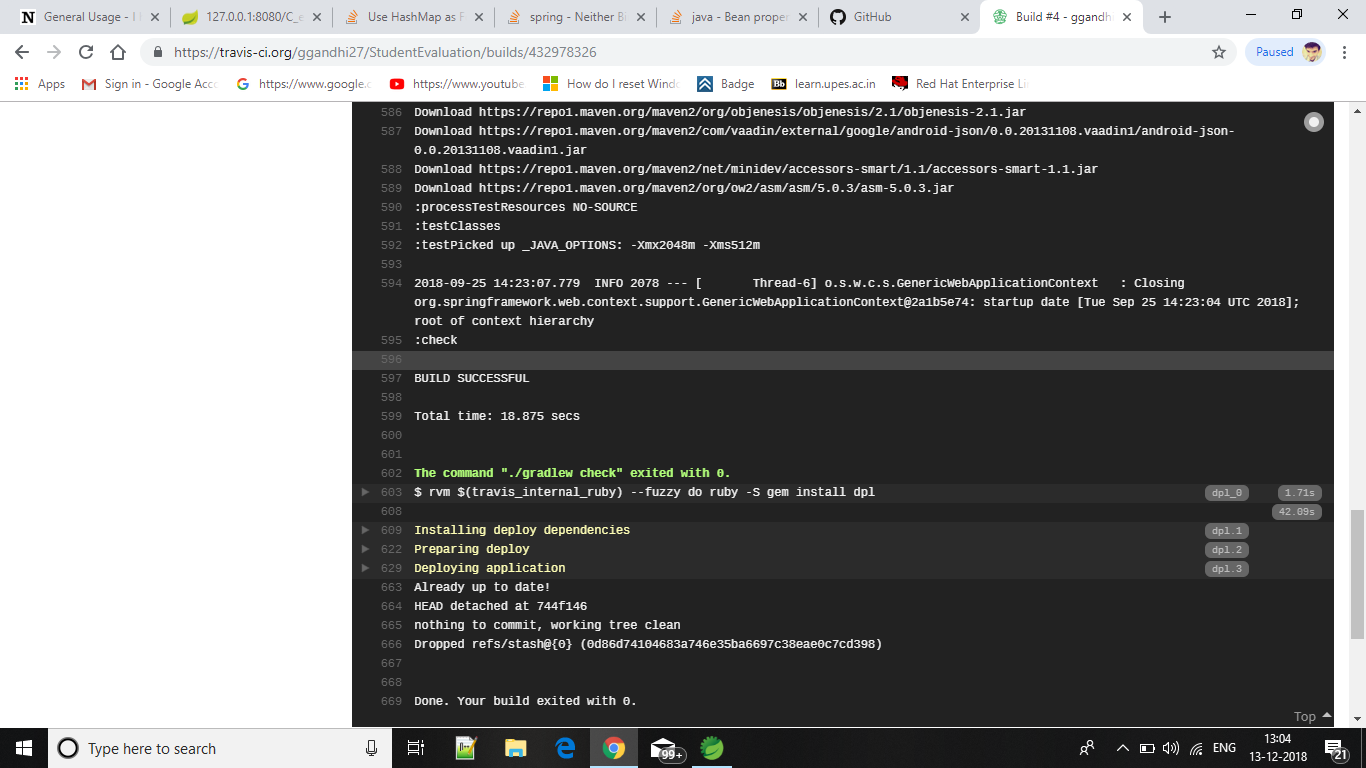
git add .travis.yml

git commit -m ".travis.yml configuration file for travis is added"

git push

When the git push command is executed, it will ask for the username and the password. Once the username and password is authenticated, the commit created will be pushed to the github and at the same time a build will be initiated in Travis CI corresponding to the code pushed**.**

****

****

1. **Heroku (Cloud Deployment Platform)**

Heroku is a cloud deployment platform which provides us free resources to some extent to deploy and test applications.

As we are going to use gradle as a build tool for our application, hence we have to follow the following steps for the deployment of a gradle application on Heroku.

- Create a heroku account.

- Install Heroku CLI on the system. (htt1)

- Install java jdk.

- Create a gradle based java application.

- Perform all these tasks with the git repository as the current working directory.

- $heroku login

- $heroku create

- $git push heroku master

- $heroku open

As we will execute the $heroku create command, it will create an application for us, which can be pushed to the heroku server by using the $ git push heroku master command.

Note the name of the application which is create by the heroku command line tool.

Now we have to deploy the code to the heroku application on the heroku cloud platform, from the travis CI. (htt2)

1. Generate a authorization token for the heroku with the help of following command. (htt3)
   1. $ heroku auth:token
2. Note the api which is generated by the heroku client.
3. Edit the .travis.yml file and append the following deployment code to it.

Deploy:

provider: heroku

app: "NAME OF THE APPLICATION CREATED"

api\_key:

secure: "YOUR ENCRYPTED API KEY"

Now save the file, commit the changes and push the commits to the github. Travis will automatically fetch the updated state of the repository from the github, and start performing the build operation on the code. Once the code is successfully built by Travis, it will deploy the code on the heroku application mentioned in the .travis.yml file.

**Module 3: Designing application according to the requirements gathered**

Our application will be a platform free for the students to test their skills on the programming languages like Java and C. The application will contain Multiple Choice type questions tests where the users of the application can anytime login into the application and can test their skills on the language of their own choice. It will be a web based application which will be accessible to the users of the application from anywhere over the internet from their web browser.

First of all the user has to get him registered on the application and once the user is registered in successfully he have to perform the login operation using the username and the password which he has chosen at the time of registering himself.

Once the user is successfully logged in to the system he can choose any of the languages available for which he has to give the test for. As soon as the user selects the language, a test will begin for him. There will be 25 questions which can be answered by the user.

On the user dashboard there will be a tab where the user can check his marks for the previous tests which were submitted by the user. User must get a log out button to successfully end the session which he has created once he logs into the system.

**Module 4: Generating Test Cases**

As we are using Test Driven Development approach for developing the application hence the test cases has to be generated prior to starting the development of the application. For generating the test cases for the application and to perform testing on the application we have used JUnit framework integrated with the gradle build tool. When the code is built in the continuous integration engine travis, at that time the test cases are executed. If any of the test cases fails to execute then the build fails and the developer must handle the issue which is causing the build to fail or else if the test cases passes, in that cases the code is built successfully and hence we can perform further operations in the development.

**Module 5: Development of Application**

To develop this application we have chosen Java as the programming language and the Java Spring Framework as the framework for the development of the application. The code is divided into three types of classes i.e. Controller, Service and Entity. Controller classes are responsible for accepting the request from the front end, Service classes contains the complete business logic for the application and Entity classes are the classes which contains the object from the database. The development of the application must go on in accordance with the test cases which are developed at the time of the Test Case Generation. As soon as a feature is developed, it must be committed and pushed to the github. Once the code is pushed to the github, Travis fetches the complete repository and performs build operation on the code. At the time of building, the developed code is tested for the test cases generated. If all the test cases executes successfully, in that case the build is performed successfully or else it fails. Developer must look after the issues which are causing the test suites to fail.

**Branching Strategy:**

Mainline branch strategy is the simplest yet most effective strategy for small to medium sized teams. The developers in the team constantly commit their work into a single, central branch—which is always in a deployment-ready state. In other words, the main branch for the project should only contain tested and quality work, and should never be broken.

Each unit of work (or sprint) may have an accordion effect on the number of branches. At first, the developers are all working on their own pieces, and the number of branches expands. Then, as each of the developers finishes his or her work and integrates it with the others’, the accordion compresses back down again. There are variations of this strategy that can be used to suit a particular team’s wishes.

In Git terms, each developer in the team will have its own local branch. For example if a team is working on a feature: Implement Environment Service and one of the tasks is Implement Env Serv API, then the developer can create a local branch from the master (for eg. env-serv-api). Remember that breaking down of tasks into a unit of work during Sprint/Release Planning becomes important where the Scrum Master/Product Owner makes sure that each unit of work is a self-contained code.

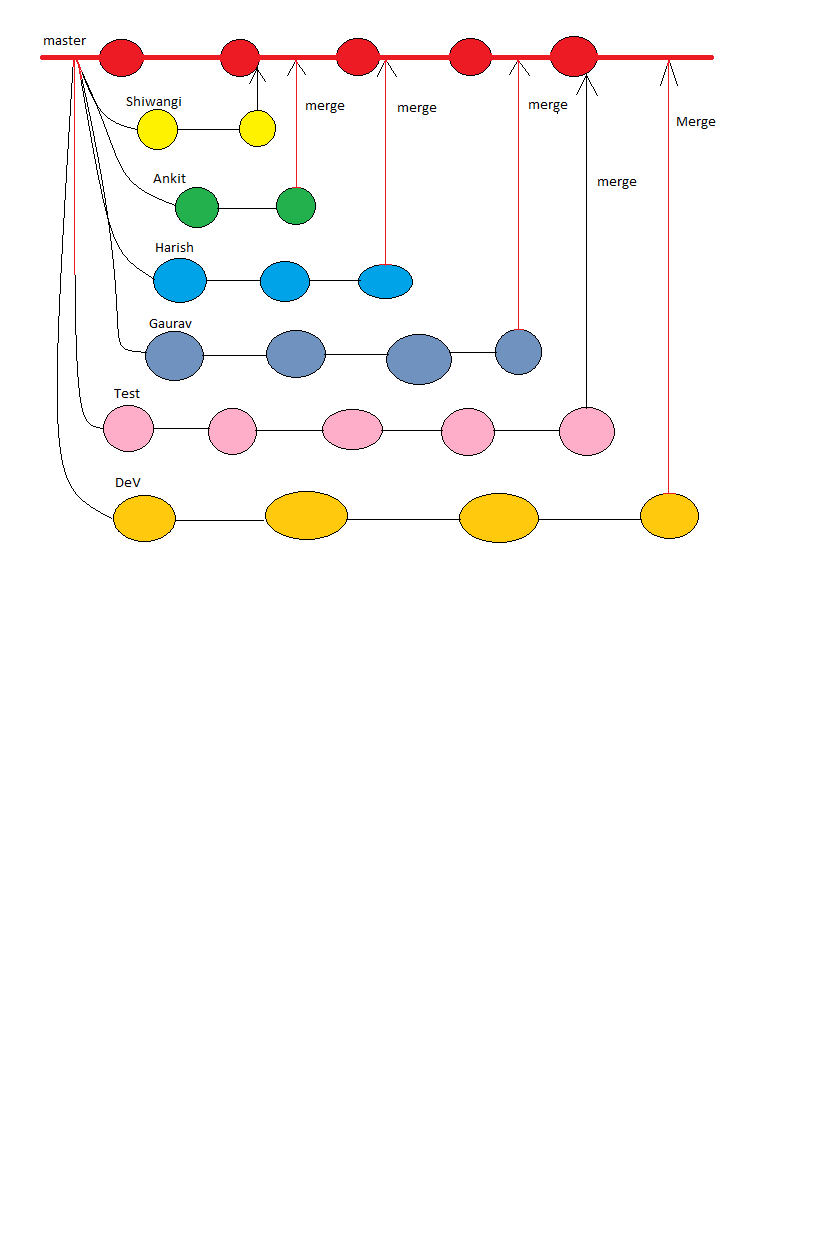
Thus, in figure below, Branch A, Branch B and Branch C corresponds to Git Local Branches on a developer workstation.

Keeping the branching strategy simple ensures that you do not spend unnecessary time doing manual merges.

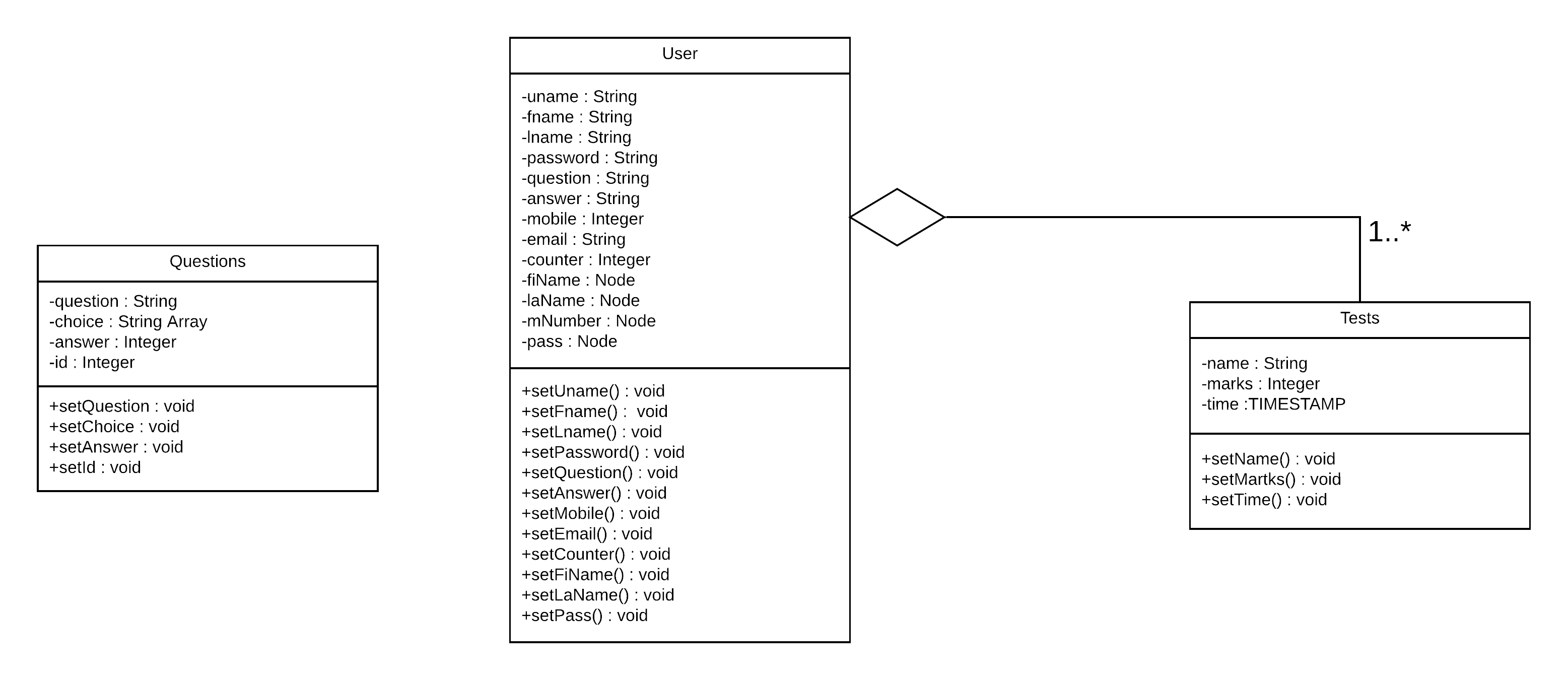
At scale, this approach of having a single working branch is used by teams working with automated build procedures.

**Advantages**

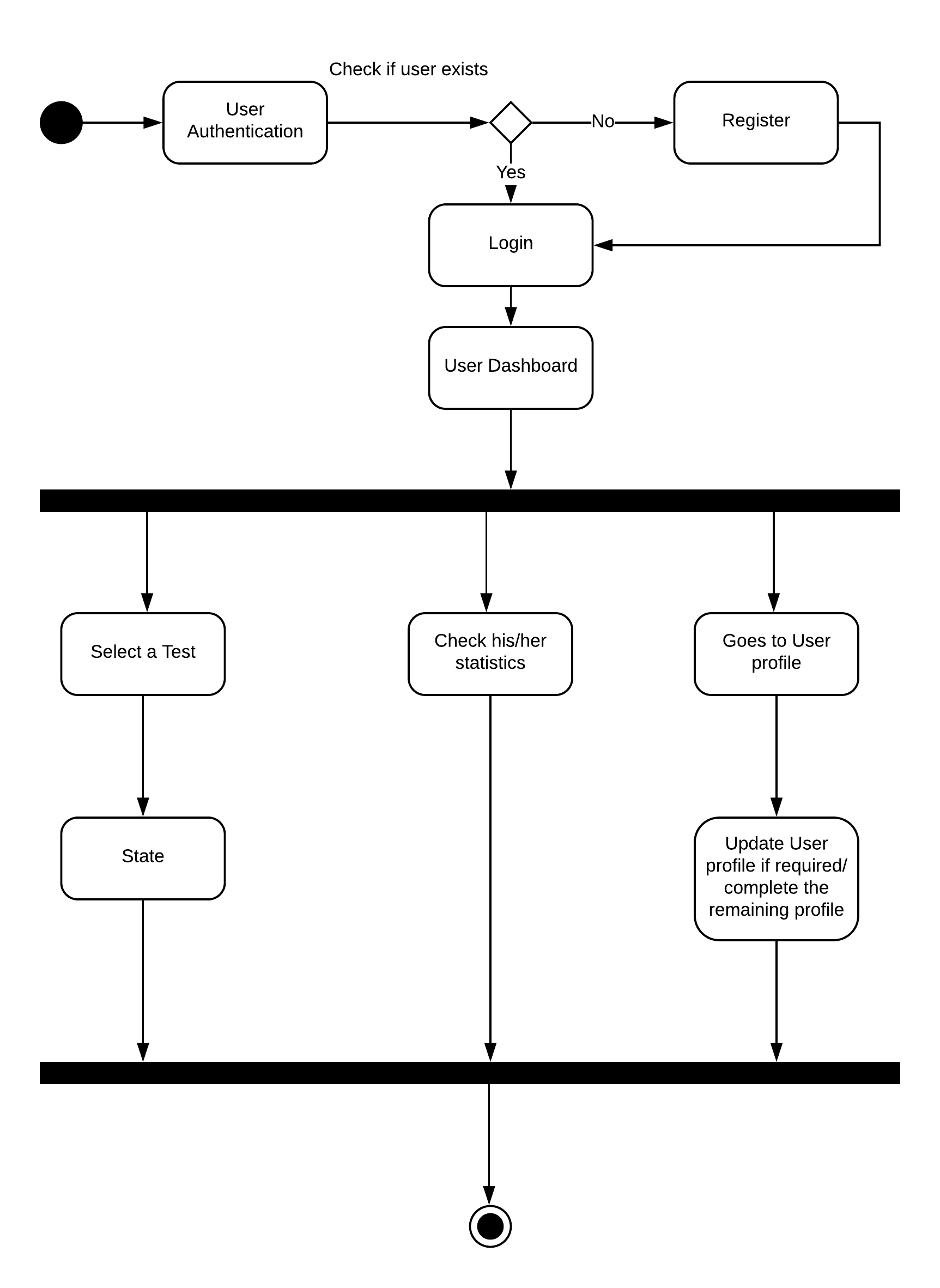
* There are not many branches across the project, which results in simplified workflows in the team as well as less confusion about where a change may have disappeared.
* Commits that are being made into the code base are relatively small and quick. If there is a problem, it should be relatively quick to undo the mistake.
* There are fewer emergency fixes, because any code that is saved into the main branch is ready to be deployed. Deployments can often be stressful for developers as they hold their breath while code goes live in production and wait to hear back from the code’s users. With tiny frequent updates, this procedure becomes practiced, and finally automated to the point where it should be almost invisible to the end user.



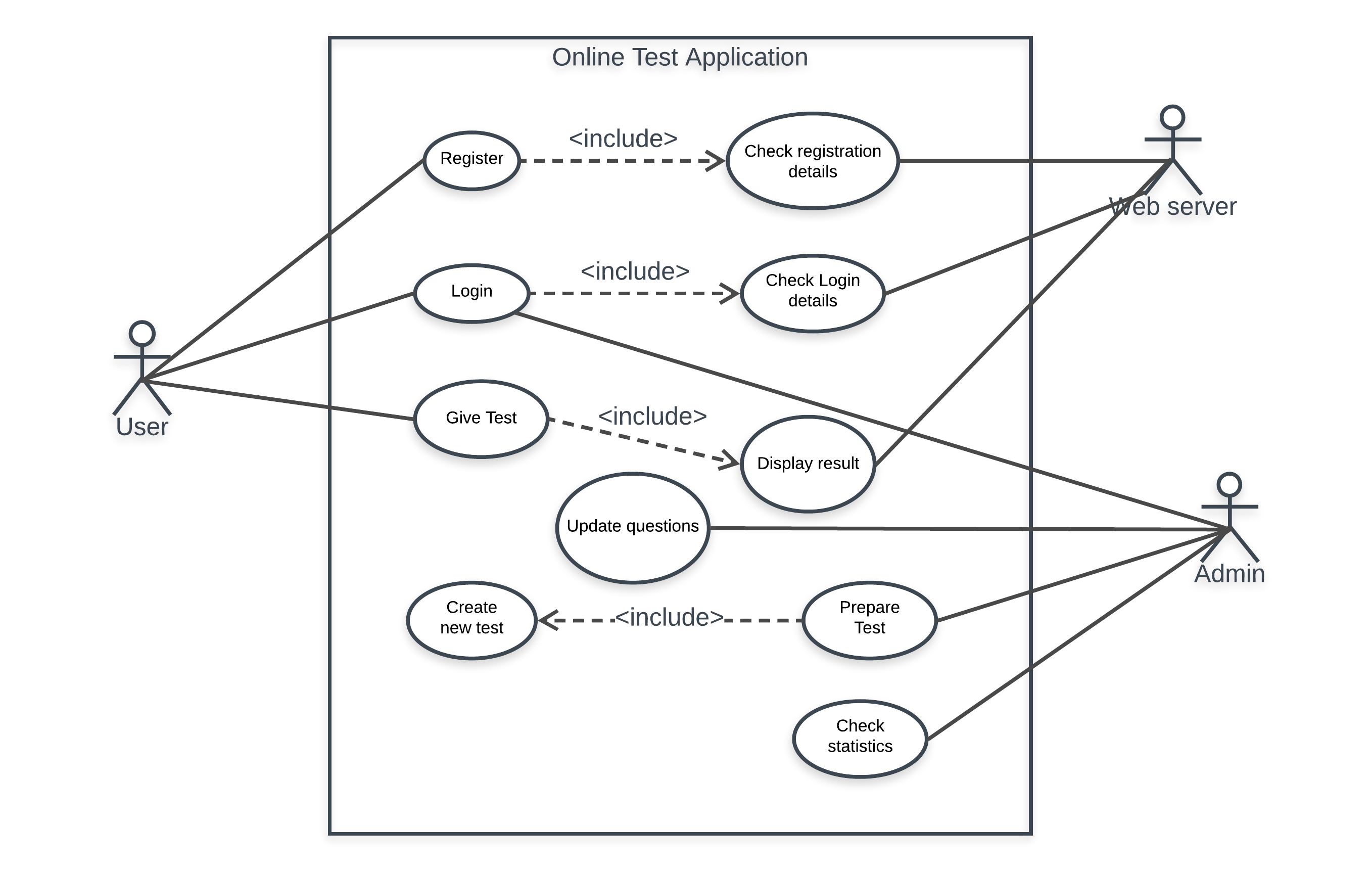
# CLASS DIAGRAM



# Activity Diagram



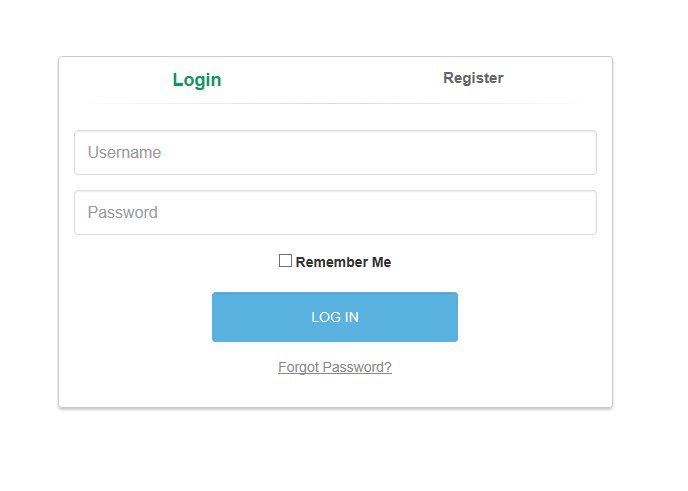
# Use Case



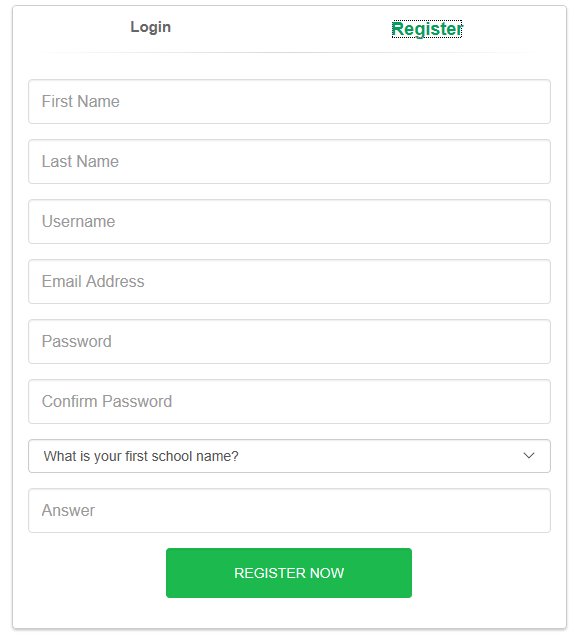
# Result and Future Scope

# Students will now be able to take tests online by creating an account and testing overhead for development and operations team will be reduced due to the setup of CI/CD pipeline.

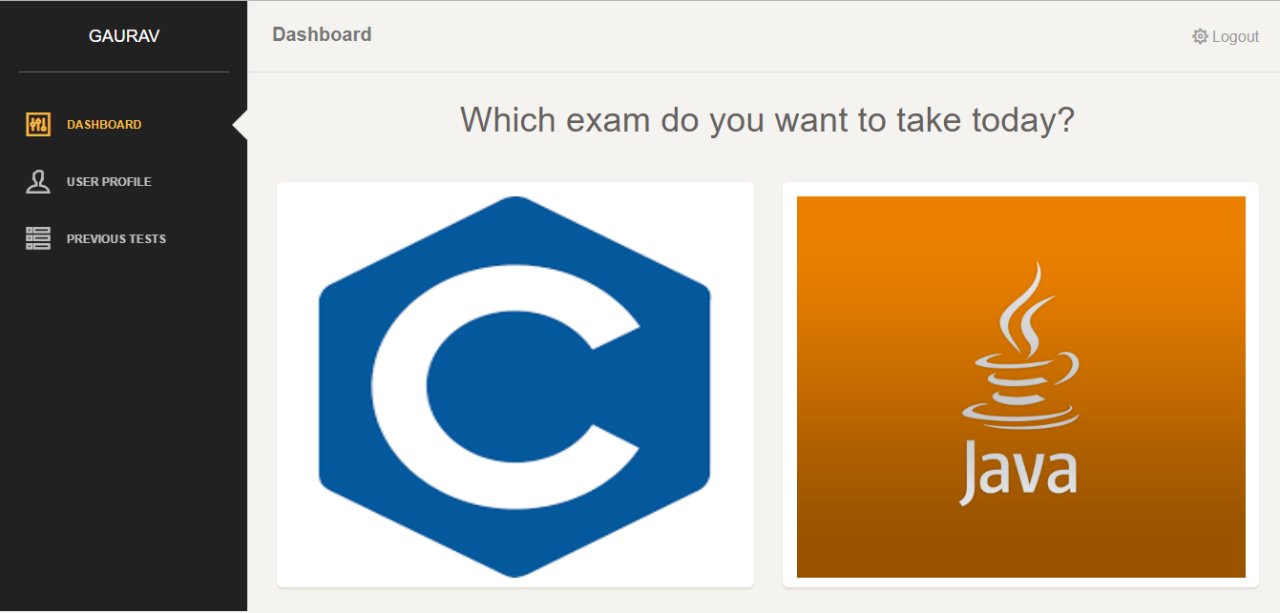
Login Page



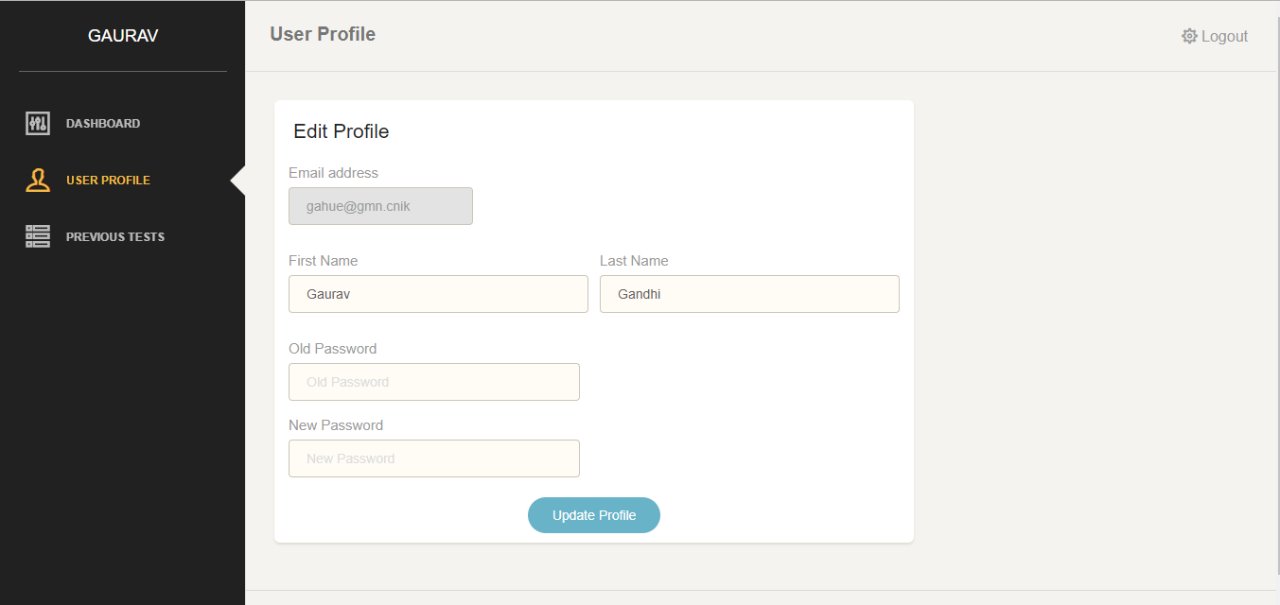
Registration Page



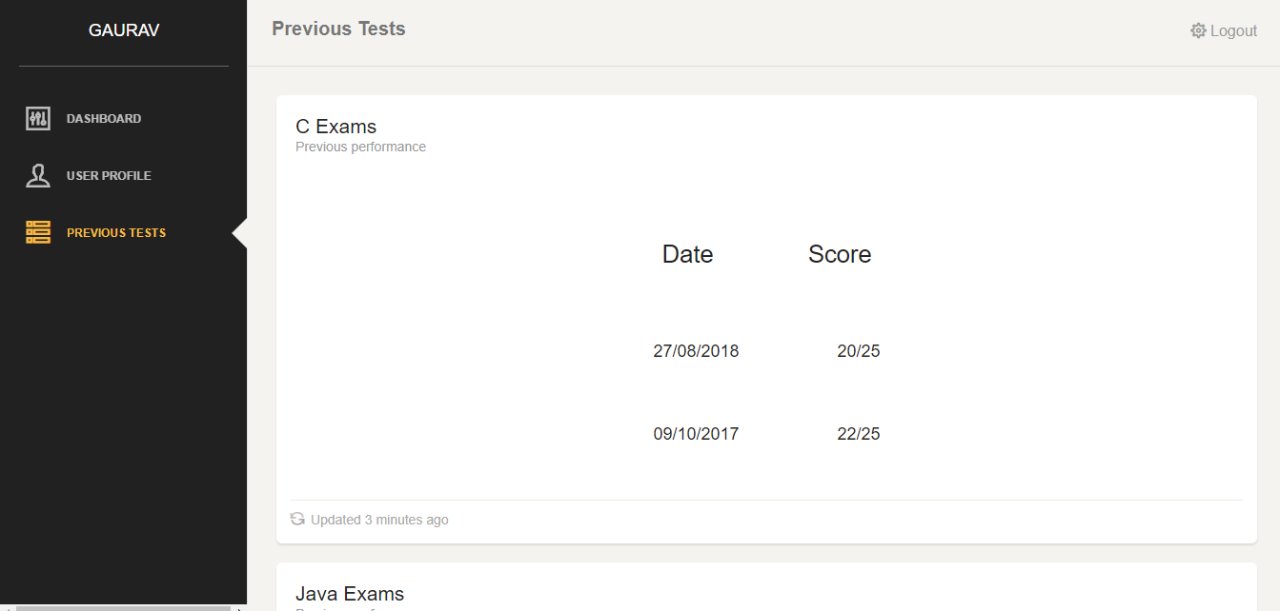
Dashboard



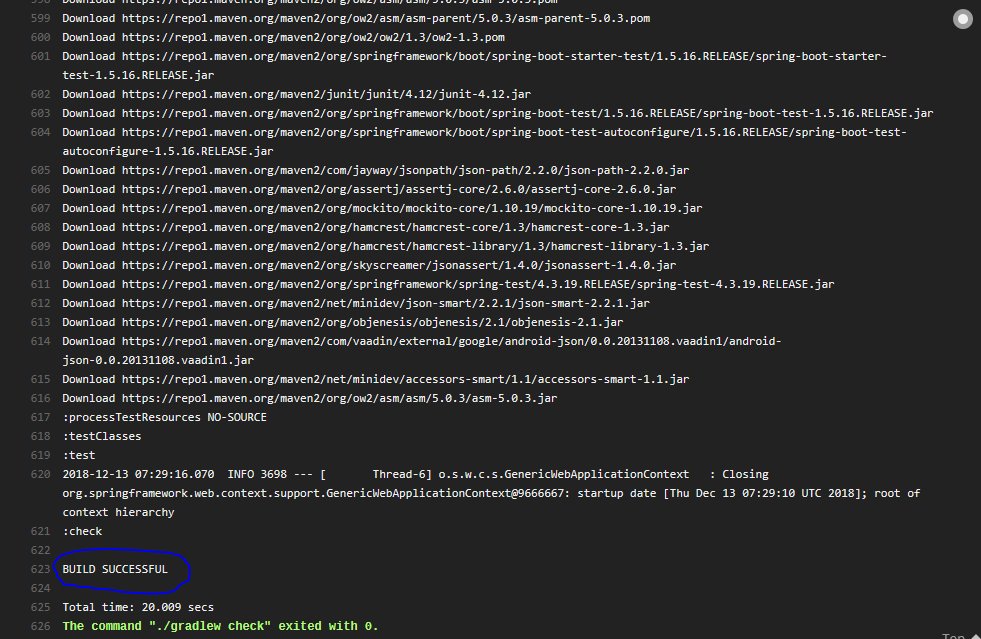
User profile



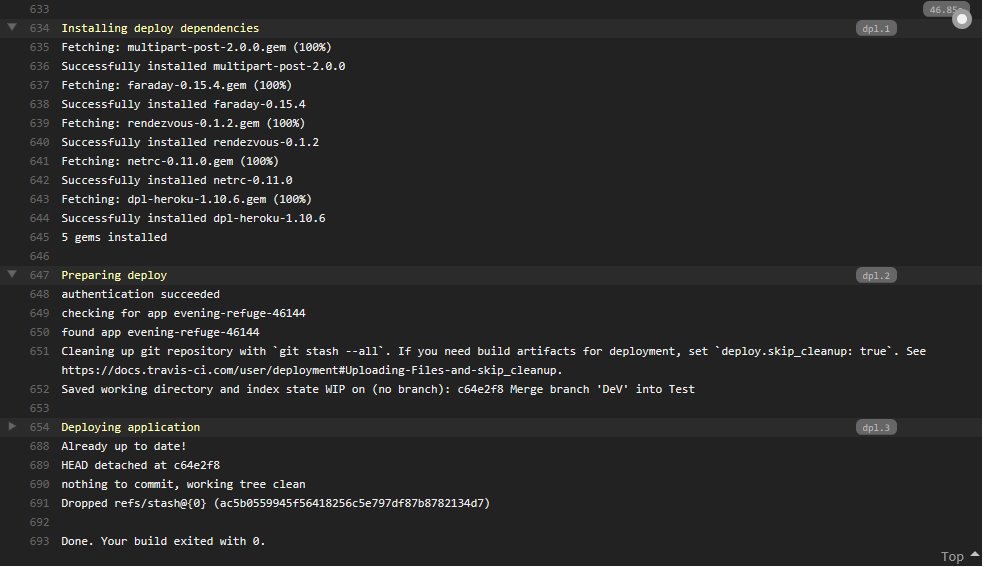
Previous test section



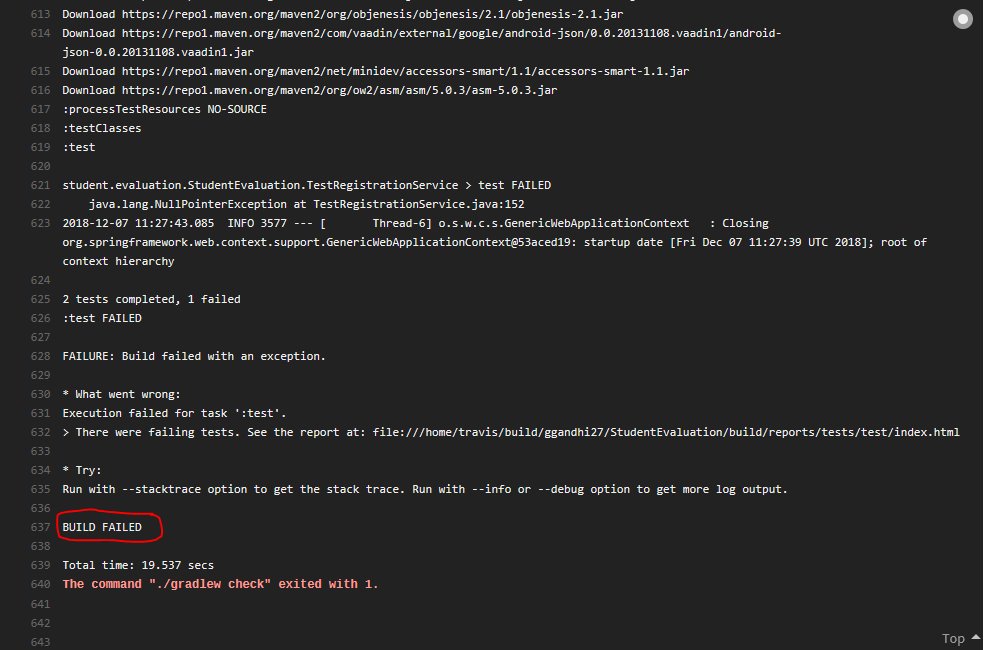
Successful build report



Successful deployment report



Unsuccessful build



# REFERENCES

(n.d.). Retrieved from https://www.linkedin.com/pulse/introduction-continuous-integration-delivery-rajesh-gurram

(n.d.). Retrieved from https://devcenter.heroku.com/articles/heroku-cli

(n.d.). Retrieved from https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku

(n.d.). Retrieved from https://docs.travis-ci.com/user/deployment/heroku/

(n.d.). Retrieved from https://www.cio.com/article/2924995/software/what-are-containers-and-why-do-you-need-them.html

(n.d.). Retrieved from https://containment.comodo.com/containerization

(n.d.). Retrieved from https://mainframedebate.com/2015/03/20/virtualization-vs-containerization/

(n.d.). Retrieved from docker.com

(n.d.). Retrieved from https://blog.powerupcloud.com/autoscaling-based-on-cpu-memory-in-kubernetes-part-ii-fe2e495bddd4

(n.d.). Retrieved from https://www.javatpoint.com/spring-tutorial

(n.d.). Retrieved from https://docs.bitnami.com/kubernetes/how-to/configure-autoscaling-custom-metrics/

(2018, 12 12). Retrieved from LINKEDIN: https://www.linkedin.com/pulse/introduction-continuous-integration-delivery-rajesh-gurram

Beck, K. (2003 ). *K. Test-Driven Development by Example.* Addison Wesley - Vaseem.

*Linkedin*. (n.d.). Retrieved from https://www.linkedin.com/pulse/introduction-continuous-integration-delivery-rajesh-gurram

# Annexure

1. LOGIN.HTML:

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org" xmlns="http://www.w3.org/1999/html">

<link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">

<script src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/js/bootstrap.min.js"></script>

<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>

<!------ Include the above in your HEAD tag ---------->

<style>

body {

padding-top: 90px;

}

.panel-login {

border-color: #ccc;

-webkit-box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.2);

-moz-box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.2);

box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.2);

}

.panel-login>.panel-heading {

color: #00415d;

background-color: #fff;

border-color: #fff;

text-align:center;

}

.panel-login>.panel-heading a{

text-decoration: none;

color: #666;

font-weight: bold;

font-size: 15px;

-webkit-transition: all 0.1s linear;

-moz-transition: all 0.1s linear;

transition: all 0.1s linear;

}

.panel-login>.panel-heading a.active{

color: #029f5b;

font-size: 18px;

}

.panel-login>.panel-heading hr{

margin-top: 10px;

margin-bottom: 0px;

clear: both;

border: 0;

height: 1px;

background-image: -webkit-linear-gradient(left,rgba(0, 0, 0, 0),rgba(0, 0, 0, 0.15),rgba(0, 0, 0, 0));

background-image: -moz-linear-gradient(left,rgba(0,0,0,0),rgba(0,0,0,0.15),rgba(0,0,0,0));

background-image: -ms-linear-gradient(left,rgba(0,0,0,0),rgba(0,0,0,0.15),rgba(0,0,0,0));

background-image: -o-linear-gradient(left,rgba(0,0,0,0),rgba(0,0,0,0.15),rgba(0,0,0,0));

}

.panel-login input[type="text"],.panel-login input[type="email"],.panel-login input[type="password"] {

height: 45px;

border: 1px solid #ddd;

font-size: 16px;

-webkit-transition: all 0.1s linear;

-moz-transition: all 0.1s linear;

transition: all 0.1s linear;

}

.panel-login input:hover,

.panel-login input:focus {

outline:none;

-webkit-box-shadow: none;

-moz-box-shadow: none;

box-shadow: none;

border-color: #ccc;

}

.btn-login {

background-color: #59B2E0;

outline: none;

color: #fff;

font-size: 14px;

height: auto;

font-weight: normal;

padding: 14px 0;

text-transform: uppercase;

border-color: #59B2E6;

}

.btn-login:hover,

.btn-login:focus {

color: #fff;

background-color: #53A3CD;

border-color: #53A3CD;

}

.forgot-password {

text-decoration: underline;

color: #888;

}

.forgot-password:hover,

.forgot-password:focus {

text-decoration: underline;

color: #666;

}

.btn-register {

background-color: #1CB94E;

outline: none;

color: #fff;

font-size: 14px;

height: auto;

font-weight: normal;

padding: 14px 0;

text-transform: uppercase;

border-color: #1CB94A;

}

.btn-register:hover,

.btn-register:focus {

color: #fff;

background-color: #1CA347;

border-color: #1CA347;

}

</style>

<script>

$(function() {

$('#login-form-link').click(function(e) {

$("#login-form").delay(100).fadeIn(100);

$("#register-form").fadeOut(100);

$('#register-form-link').removeClass('active');

$(this).addClass('active');

e.preventDefault();

});

$('#register-form-link').click(function(e) {

$("#register-form").delay(100).fadeIn(100);

$("#login-form").fadeOut(100);

$('#login-form-link').removeClass('active');

$(this).addClass('active');

e.preventDefault();

});

});

</script>

<div class="container">

<div class="row">

<div class="col-md-6 col-md-offset-3">

<div class="panel panel-login">

<div class="panel-heading">

<div class="row">

<div class="col-xs-6">

<a href="#" class="active" id="login-form-link">Login</a>

</div>

<div class="col-xs-6">

<a href="#" id="register-form-link">Register</a>

</div>

</div>

<hr>

</div>

<div class="panel-body">

<div class="row">

<div class="col-lg-12">

<form id="login-form" action="/Login" method="post" role="form" style="display: block;">

<div class="form-group">

<input type="text" name="username" id="username" tabindex="1" class="form-control" placeholder="Username" value="">

</div>

<div class="form-group">

<input type="password" name="password" id="password" tabindex="2" class="form-control" placeholder="Password">

</div>

<div class="form-group text-center">

<input type="checkbox" tabindex="3" class="" name="remember" id="remember">

<label for="remember"> Remember Me</label>

</div>

<div class="form-group">

<div class="row">

<div class="col-sm-6 col-sm-offset-3">

<input type="submit" name="login-submit" id="login-submit" tabindex="4" class="form-control btn btn-login" value="Log In">

</div>

</div>

</div>

<div class="form-group">

<div class="row">

<div class="col-lg-12">

<div class="text-center">

<a href="https://phpoll.com/recover" tabindex="5" class="forgot-password">Forgot Password?</a>

</div>

</div>

</div>

</div>

</form>

<form id="register-form" action="/register" method="post" role="form" style="display: none;">

<div class="form-group">

<input type="text" name="fname" id="fname" tabindex="2" class="form-control" placeholder="First Name">

</div>

<div class="form-group">

<input type="text" name="lname" id="lname" tabindex="2" class="form-control" placeholder="Last Name">

</div>

<div class="form-group">

<input type="text" name="uname" id="username" tabindex="2" class="form-control" placeholder="Username" value="">

</div>

<div class="form-group">

<input type="email" name="email" id="email" tabindex="2" class="form-control" placeholder="Email Address" value="">

</div>

<div class="form-group">

<input type="password" name="pass" id="password" tabindex="2" class="form-control" placeholder="Password">

</div>

<div class="form-group">

<input type="password" name="confirm-password" id="confirm-password" tabindex="2" class="form-control" placeholder="Confirm Password">

</div>

<div class="form-group">

<select tabindex="2" class="form-control" name="question">

<option value="What is your first school name?" tabindex="2" class="form-control">What is your first school name?</option>

<option value="Which city you were born in?" tabindex="2" class="form-control">Which city you were born in?</option>

<option value="What is the first name of your best friend?" tabindex="2" class="form-control">What is the first name of your best friend?</option>

</select>

</div>

<div class="form-group">

<input type="text" name="answer" id="answer" tabindex="2" class="form-control" placeholder="Answer">

</div>

<div class="form-group">

<div class="row">

<div class="col-sm-6 col-sm-offset-3">

<input type="submit" name="register-submit" id="register-submit" tabindex="4" class="form-control btn btn-register" value="Register Now">

</div>

</div>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</html>

1. Login.java

package student.evaluation.StudentEvaluation;

import java.io.File;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.SessionAttributes;

import org.springframework.web.servlet.ModelAndView;

import org.thymeleaf.templateresolver.ServletContextTemplateResolver;

import entity.User;

@Controller

@SessionAttributes("user")

public class Login {

@Autowired

private LoginService loginService;

@RequestMapping(value="/")

public String login() {

return "login";

}

@RequestMapping(value="/Login")

public ModelAndView checkLogin(@RequestParam(name="username")String username,

@RequestParam(name="password")String password) {

File file = new File("/tmp/"+username+".xml");

if (!file.exists()) {

return null;

}

else {

if(loginService.checkPassword(username,password)) {

ServletContextTemplateResolver templateResolver = new ServletContextTemplateResolver();

templateResolver.setPrefix("/templates/");

templateResolver.setSuffix(".html");

ModelAndView model=new ModelAndView("dashboard");

User user = loginService.getUserObject(username);

model.addObject("user", user);

return model;

}

else {

return null;

}

}

}

}

1. LoginService.Java

package student.evaluation.StudentEvaluation;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import org.springframework.stereotype.Service;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import entity.User;

@Service

public class LoginService {

private Document getDocument(String username) {

try {

File inputFile = new File("/tmp/"+username+".xml");

*// File inputFile = new File("e:\\New folder\\"+username+".xml");*

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(inputFile);

doc.getDocumentElement().normalize();

return doc;

}

catch(Exception e) {

return null;

}

}

public boolean checkPassword(String username,String password) {

File inputFile = new File("/tmp/"+username+".xml");

*// File inputFile = new File("e:\\New folder\\"+username+".xml");*

Document doc = this.getDocument(username);

if(doc==(null)) {

return false;

}

String pass;

NodeList nList = doc.getElementsByTagName("users");

Node node = nList.item(0);

Element element = (Element)node;

pass=element.getElementsByTagName("password").item(0).getTextContent();

if(pass.equals(password))

{

return true;

}

else {

return false;

}

}

public User getUserObject(String username) {

User user = new User();

Document doc = getDocument(username);

NodeList nList = doc.getElementsByTagName("users");

Node node = nList.item(0);

Element element = (Element)node;

user.setUname(element.getElementsByTagName("uname").item(0).getTextContent());

user.setPassword(element.getElementsByTagName("password").item(0).getTextContent());

user.setFname(element.getElementsByTagName("fname").item(0).getTextContent());

user.setLname(element.getElementsByTagName("lname").item(0).getTextContent());

user.setQuestion(element.getElementsByTagName("question").item(0).getTextContent());

user.setAnswer(element.getElementsByTagName("answer").item(0).getTextContent());

user.setEmail(element.getElementsByTagName("email").item(0).getTextContent());

return user;

}

}

1. LogoutController.Java

package student.evaluation.StudentEvaluation;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

@RequestMapping("/logout")

public class LogoutController {

@RequestMapping(method=RequestMethod.GET)

public String logout(HttpSession session) {

session.invalidate();

return "redirect:/";

}

}

1. Question.java

package student.evaluation.StudentEvaluation;

import entity.Questions;

import java.io.File;

import java.util.\*;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Attr;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

public class Question {

static Scanner sc = new Scanner(System.in);

static Scanner in = new Scanner(System.in);

*// private int count;*

private static int nchoice;

private static String quest;

private static String choice [] = new String[6];

private static int answer;

Element rootElement;

Element number;

Element question []= new Element[100];

Element choices[] = new Element[6];

Element ans;

Attr attr;

Attr attrtype[] = new Attr[6];

public void recurse(int counter) {

try {

DocumentBuilderFactory dbFactory =DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

*// root element*

rootElement = doc.createElement("questions");

doc.appendChild(rootElement);

for(int i=0; i<counter; i++) {

this.question();

this.choice();

this.ans();

*// Number element*

number = doc.createElement("number");

rootElement.appendChild(number);

*//setting id attribute*

attr = doc.createAttribute("id");

attr.setValue(""+(i+1));

number.setAttributeNode(attr);

*//Question element*

question[i] = doc.createElement("question");

number.appendChild(question[i]);

question[i].appendChild(doc.createTextNode(Question.quest));

for(int x = 0;x < Question.nchoice; x++) {

*//Choices elements*

choices[x] = doc.createElement("choice");

attrtype[x] = doc.createAttribute("id");

attrtype[x].setValue(""+(x+1));

choices[x].setAttributeNode(attrtype[x]);

choices[x].appendChild(doc.createTextNode(Question.choice[x]));

number.appendChild(choices[x]);

}

*//Answer Element*

ans = doc.createElement("ans");

number.appendChild(ans);

ans.appendChild(doc.createTextNode(""+Question.answer));

}

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File("src\\com\\javaxml\\c.xml"));

transformer.transform(source, result);

}

catch (Exception e) {

e.printStackTrace();

}

}

public ArrayList<Questions> fetchQuestion() {

ArrayList<Questions> arrayList = new ArrayList<Questions>();

try {

File inputFile = new File("src\\com\\javaxml\\c.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(inputFile);

doc.getDocumentElement().normalize();

NodeList numberList = doc.getElementsByTagName("number");

for (int temp = 0; temp < numberList.getLength(); temp++) {

Questions questions = new Questions();

Node numberTag = numberList.item(temp);

Element eElement = (Element) numberTag;

*//assign question*

questions.setQuestion(eElement.getElementsByTagName("question").item(0).getTextContent());

*//fetch choices*

int temp1 = eElement.getElementsByTagName("choice").getLength();

String choiceArray[] = new String[temp1];

for(int j = 0; j < temp1; j++) {

choiceArray[j] = eElement.getElementsByTagName("choice").item(j).getTextContent();

}

*//assign choices*

questions.setChoice(choiceArray);

*//assign answer*

questions.setAnswer(Integer.parseInt(eElement.getElementsByTagName("ans").item(0).getTextContent())-1);

*//assign id*

questions.setId(temp + 1);

arrayList.add(questions);

}

}

catch (Exception e) {

e.printStackTrace();

}

return arrayList;

}

public void question() {

System.out.print("Enter the Question:");

Question.quest = in.nextLine();

}

public void choice() {

System.out.print("Enter the number of choices:");

Question.nchoice = sc.nextInt();

for(int x = 0; x < Question.nchoice; x++) {

System.out.print("Enter your choice "+(x+1)+":");

Question.choice[x] = in.nextLine();

}

}

public void ans() {

System.out.print("What could be the answer:");

Question.answer = sc.nextInt();

}

}

1. RegisterServcice.java

package student.evaluation.StudentEvaluation;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.springframework.stereotype.Service;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import entity.User;

@Service

public class RegisterService {

public boolean register(User user) {

try {

File files = new File("/tmp/"+user.getUname()+".xml");

*// File files = new File("e:\\New folder\\"+user.getUname()+".xml");*

if (files.exists())

throw new Exception();

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

*// root element*

Element rootElement = doc.createElement("users");

doc.appendChild(rootElement);

*//fname element*

Element fname = doc.createElement("fname");

rootElement.appendChild(fname);

fname.appendChild(doc.createTextNode(user.getFname()));

*//lname element*

Element lname = doc.createElement("lname");

rootElement.appendChild(lname);

lname.appendChild(doc.createTextNode(user.getLname()));

*//username element*

Element uname = doc.createElement("uname");

rootElement.appendChild(uname);

uname.appendChild(doc.createTextNode(user.getUname()));

*// password element*

Element password = doc.createElement("password");

rootElement.appendChild(password);

password.appendChild(doc.createTextNode(user.getPassword()));

*// security element*

Element question = doc.createElement("question");

rootElement.appendChild(question);

question.appendChild(doc.createTextNode(user.getQuestion()));

*//securityAnswer element*

Element answer = doc.createElement("answer");

rootElement.appendChild(answer);

answer.appendChild(doc.createTextNode(user.getAnswer()));

*//mobile element*

Element mobile = doc.createElement("mobile");

rootElement.appendChild(mobile);

mobile.appendChild(doc.createTextNode(user.getMobile()));

*//email element*

Element email = doc.createElement("email");

rootElement.appendChild(email);

email.appendChild(doc.createTextNode(user.getEmail()));

*//tests child-parent element*

Element tests = doc.createElement("tests");

rootElement.appendChild(tests);

*//counter child element of tests*

Element counter = doc.createElement("counter");

tests.appendChild(counter);

counter.appendChild(doc.createTextNode(Integer.toString(user.getCounter())));

*// File files = new File("/UserDetails");*

*//*

*// if (!files.exists()) {*

*// System.out.println("Directory does not exists.");*

*// files.mkdir();*

*// System.out.println("Directory created.");*

*// }*

*//*

*// write the content into xml file*

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File("/tmp/"+user.getUname()+".xml"));

*// StreamResult result = new StreamResult(new File("e:\\New folder\\"+user.getUname()+".xml"));*

System.out.println("File created");

transformer.transform(source, result);

*// Output to console for testing*

StreamResult consoleResult = new StreamResult(System.out);

transformer.transform(source, consoleResult);

}

catch (Exception e) {

return false;

}

return true;

}

}

1. RegisterController.java

package student.evaluation.StudentEvaluation;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

import entity.User;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class RegisterController {

@Autowired

private RegisterService registerService;

@RequestMapping(value="/register", method = RequestMethod.POST)

public String register(@RequestParam(value="fname") String fname,

@RequestParam(value="lname") String lname,

@RequestParam(value="uname") String username,

@RequestParam(value="email") String email,

@RequestParam(value="pass")String password,

@RequestParam(value="question")String question,

@RequestParam(value="answer")String answer) {

File files = new File("/tmp/"+username+".xml");

if (files.exists())

return "Username already exists exists.";

User user = new User();

user.setUname(username);

user.setFname(fname);

user.setLname(lname);

user.setEmail(email);

user.setPassword(password);

user.setQuestion(question);

user.setAnswer(answer);

if(registerService.register(user))

{

return "redirect:/";

}

else {

return "registration failed";

}

*// return registerService.register(user);*

}

@RequestMapping(value="/read",method = RequestMethod.GET)

@ResponseBody

public String read(@RequestParam(value="file")String file)

{

try {

BufferedReader p = new BufferedReader(new FileReader("/tmp/"+file));

String word="",line;

while ((line = p.readLine()) != null) {

word = word +line;

}

return word;

}

catch(Exception e) {

return e.getMessage();

}

}

}

1. dashboard.html

<!doctype html >

<html lang="en" xmlns:th="http://www.thymeleaf.org" xmlns="http://www.w3.org/1999/html">

<head>

<meta charset="utf-8" />

<link rel="apple-touch-icon" sizes="76x76" th:href="@{assets/img/apple-icon.png}" />

<link rel="icon" type="image/png" sizes="96x96" th:href="@{assets/img/favicon.png}" />

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<title>User Dashboard</title>

<meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0' name='viewport' />

<meta name="viewport" content="width=device-width" />

<!-- Bootstrap core CSS -->

<link th:href="@{assets/css/bootstrap.min.css}" rel="stylesheet" />

<!-- Animation library for notifications -->

<link th:href="@{assets/css/animate.min.css}" rel="stylesheet" />

<!-- Paper Dashboard core CSS -->

<link th:href="@{assets/css/paper-dashboard.css}" rel="stylesheet" />

<!-- Fonts and icons -->

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/latest/css/font-awesome.min.css" rel="stylesheet" />

<link href='https://fonts.googleapis.com/css?family=Muli:400,300' rel='stylesheet' type='text/css' />

<link th:href="@{assets/css/themify-icons.css}" rel="stylesheet" />

</head>

<body>

<div class="wrapper">

<div class="sidebar" data-background-color="black" data-active-color="warning">

<!--

Tip 1: you can change the color of the sidebar's background using: data-background-color="white | black"

Tip 2: you can change the color of the active button using the data-active-color="primary | info | success | warning | danger"

-->

<div class="sidebar-wrapper">

<div class="logo">

<a class="simple-text" th:text="${user.getUname()}"></a>

</div>

<ul class="nav">

<li class="active">

<a th:href="@{/dash}">

<i class="ti-panel"></i>

<p>Dashboard</p>

</a>

</li>

<li>

<a th:href="@{/user}">

<i class="ti-user"></i>

<p>User Profile</p>

</a>

</li>

<li>

<a th:href="@{/ptests}">

<i class="ti-view-list-alt"></i>

<p>Previous Tests</p>

</a>

</li>

</ul>

</div>

</div>

<div class="main-panel">

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar bar1"></span>

<span class="icon-bar bar2"></span>

<span class="icon-bar bar3"></span>

</button>

<a class="navbar-brand" href="#">Dashboard</a>

</div>

<div class="collapse navbar-collapse">

<ul class="nav navbar-nav navbar-right">

<li>

<a th:href="@{/logout}">

<i class="ti-settings"></i>

<p>Logout</p>

</a>

</li>

</ul>

</div>

</div>

</nav>

<h2><center>Which exam do you want to take today?</center></h2>

<div class="content">

<div class="container-fluid">

<div class="row">

<div class="col-md-6">

<div class="card">

<div class="content">

<a th:href="@{/C\_exam}" ><img th:attr="src=@{/C.png}" alt="Smiley Face" height="400" width="470" /></a>

</div>

</div>

</div>

<div class="col-md-6">

<div class="card ">

<div class="content">

<a th:href="@{/J\_exam}"><img th:attr="src=@{/java.png}" alt="Sad Face" width="470" height="400" /></a>

</div>

</div>

</div>

</div>

</div>

</div>

<footer class="footer">

<div class="container-fluid">

<div class="copyright pull-right">

&copy; <script>document.write(new Date().getFullYear())</script>, made with <i class="fa fa-heart heart"></i> by our team.

</div>

</div>

</footer>

</div>

</div>

</body>

<!-- Core JS Files -->

<script th:src="@{assets/js/jquery.min.js}" type="text/javascript"></script>

<script th:src="@{assets/js/bootstrap.min.js}" type="text/javascript"></script>

<!-- Checkbox, Radio & Switch Plugins -->

<script th:src="@{assets/js/bootstrap-checkbox-radio.js}"></script>

<!-- Charts Plugin -->

<script th:src="@{assets/js/chartist.min.js}"></script>

<!-- Notifications Plugin -->

<script th:src="@{assets/js/bootstrap-notify.js}"></script>

<!-- Paper Dashboard Core javascript and methods for Demo purpose -->

<script th:src="@{assets/js/paper-dashboard.js}"></script>

<!-- Paper Dashboard DEMO methods, don't include it in your project! -->

<script th:src="@{assets/js/demo.js}"></script>

<script type="text/javascript">

$(document).ready(function(){

demo.initChartist();

$.notify({

icon: 'ti-gift',

message: "Welcome to <b>Your Dashboard</b> - A beautiful platform to test your skills."

},{

type: 'success',

timer: 4000

});

});

</script>

</html>

1. user.html

<!doctype html>

<html lang="en" xmlns:th="http://www.thymeleaf.org" xmlns="http://www.w3.org/1999/html">

<head>

<meta charset="utf-8" />

<link rel="apple-touch-icon" sizes="76x76" href="assets/img/apple-icon.png"/>

<link rel="icon" type="image/png" sizes="96x96" href="assets/img/favicon.png"/>

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<title>User Profile</title>

<meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0' name='viewport' />

<meta name="viewport" content="width=device-width" />

<!-- Bootstrap core CSS -->

<link href="assets/css/bootstrap.min.css" rel="stylesheet" />

<!-- Animation library for notifications -->

<link href="assets/css/animate.min.css" rel="stylesheet"/>

<!-- Paper Dashboard core CSS -->

<link href="assets/css/paper-dashboard.css" rel="stylesheet"/>

<!-- Fonts and icons -->

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/latest/css/font-awesome.min.css" rel="stylesheet"/>

<link href='https://fonts.googleapis.com/css?family=Muli:400,300' rel='stylesheet' type='text/css'/>

<link href="assets/css/themify-icons.css" rel="stylesheet"/>

</head>

<body>

<div class="wrapper">

<div class="sidebar" data-background-color="black" data-active-color="warning">

<!--

Tip 1: you can change the color of the sidebar's background using: data-background-color="white | black"

Tip 2: you can change the color of the active button using the data-active-color="primary | info | success | warning | danger"

-->

<div class="sidebar-wrapper">

<div class="logo">

<a class="simple-text" th:text="${user.getUname()}"></a>

</div>

<ul class="nav">

<li>

<a th:href="@{/dash}">

<i class="ti-panel"></i>

<p>Dashboard</p>

</a>

</li>

<li class="active">

<a th:href="@{/user}">

<i class="ti-user"></i>

<p>User Profile</p>

</a>

</li>

<li>

<a th:href="@{/ptests}">

<i class="ti-view-list-alt"></i>

<p>Previous Tests</p>

</a>

</li>

</ul>

</div>

</div>

<div class="main-panel">

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar bar1"></span>

<span class="icon-bar bar2"></span>

<span class="icon-bar bar3"></span>

</button>

<a class="navbar-brand" href="#">User Profile</a>

</div>

<div class="collapse navbar-collapse">

<ul class="nav navbar-nav navbar-right">

<li>

<a th:href="@{/logout}">

<i class="ti-settings"></i>

<p>Logout</p>

</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="content">

<div class="container-fluid">

<div class="row">

<div class="col-lg-8 col-md-7">

<div class="card">

<div class="header">

<h4 class="title">Edit Profile</h4>

</div>

<div class="content">

<form th:action="@{/update}" method="post">

<div class="row">

<div class="col-md-4">

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control border-input" placeholder="Email" th:value="${user.getEmail()}" readonly/>

</div>

</div>

</div>

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label>First Name</label>

<input type="text" name="fname" class="form-control border-input" placeholder="Company" th:value="${user.getFname()}"/>

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label>Last Name</label>

<input type="text" name="lname" class="form-control border-input" placeholder="Last Name" th:value="${user.getLname()}"/>

</div>

</div>

</div>

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label>Old Password</label>

<input type="text" name="opass" class="form-control border-input" placeholder="Old Password"/>

</div>

<div class="form-group">

<label>New Password</label>

<input type="text" name="npass" class="form-control border-input" placeholder="New Password"/>

</div>

</div>

</div>

<div class="text-center">

<button type="submit" class="btn btn-info btn-fill btn-wd">Update Profile</button>

</div>

<div class="clearfix"></div>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

<footer class="footer">

<div class="container-fluid">

<div class="copyright pull-right">

&copy; <script>document.write(new Date().getFullYear())</script>, made with <i class="fa fa-heart heart"></i> by our team.

</div>

</div>

</footer>

</div>

</div>

</body>

<!-- Core JS Files -->

<script src="assets/js/jquery.min.js" type="text/javascript"></script>

<script src="assets/js/bootstrap.min.js" type="text/javascript"></script>

<!-- Checkbox, Radio & Switch Plugins -->

<script src="assets/js/bootstrap-checkbox-radio.js"></script>

<!-- Charts Plugin -->

<script src="assets/js/chartist.min.js"></script>

<!-- Notifications Plugin -->

<script src="assets/js/bootstrap-notify.js"></script>

</html>

1. **Update.java**

package student.evaluation.StudentEvaluation;

import entity.User;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.DocumentBuilder;

import org.w3c.dom.Document;

import org.w3c.dom.NodeList;

import org.w3c.dom.Node;

import org.w3c.dom.Element;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

public class Update {

private static NodeList fnameList;

private static NodeList lnameList;

private static NodeList passwordList;

private static NodeList mobileList;

public static void updateName(User obj) {

obj.setFiName(fnameList.item(0));

obj.setLaName(lnameList.item(0));

}

public static void updatePassword(User obj) {

obj.setPass(passwordList.item(0));

}

public static void updateMNumber(User obj) {

obj.setmNumber(mobileList.item(0));

}

public static void main(String[] arg) {

try {

User obj = new User();

*// String filepath = "/tmp/"+obj.getUname()+".xml";*

String filepath = "e:\\gaurav.xml";

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(filepath);

doc.getDocumentElement().normalize();

fnameList = doc.getElementsByTagName("fname");

lnameList = doc.getElementsByTagName("lname");

passwordList = doc.getElementsByTagName("password");

mobileList = doc.getElementsByTagName("mobile");

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filepath));

transformer.transform(source, result);

}

catch (Exception e) {

e.printStackTrace();

}

}

}

1. **UpdateController.java**

package student.evaluation.StudentEvaluation;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.SessionAttributes;

import org.springframework.web.servlet.ModelAndView;

import entity.User;

@Controller

@SessionAttributes("user")

public class UpdateController {

@Autowired

UpdateService updateService;

@RequestMapping(value="/update", method = RequestMethod.POST)

*// @ResponseBody*

public ModelAndView updateProfile(

@ModelAttribute("user")User user,

@RequestParam(name="fname")String fname,

@RequestParam(name="lname")String lname,

@RequestParam(name="opass")String opass,

@RequestParam(name="npass")String npass) {

ModelAndView model = new ModelAndView("user");

user.setFname(fname);

user.setLname(lname);

user.setPassword(npass);

updateService.updateUser(user);

model.addObject("user", user);

return model;

*// return user.getEmail();*

}

}

1. UpdateService.java

package student.evaluation.StudentEvaluation;

import entity.User;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.DocumentBuilder;

import org.springframework.stereotype.Service;

import org.w3c.dom.Document;

import org.w3c.dom.NodeList;

@Service

public class UpdateService {

private static NodeList fnameList;

private static NodeList lnameList;

private static NodeList passwordList;

public void updateName(User obj) {

obj.setFiName(fnameList.item(0));

obj.setLaName(lnameList.item(0));

}

public void updatePassword(User obj) {

obj.setPass(passwordList.item(0));

}

public void updateUser(User obj) {

try {

String filepath = "/tmp/"+obj.getUname()+".xml";

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(filepath);

doc.getDocumentElement().normalize();

fnameList = doc.getElementsByTagName("fname");

lnameList = doc.getElementsByTagName("lname");

passwordList = doc.getElementsByTagName("password");

this.updateName(obj);

this.updatePassword(obj);

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filepath));

transformer.transform(source, result);

}

catch (Exception e) {

e.printStackTrace();

}

}

}

1. TestRegistrationService.java

package student.evaluation.StudentEvaluation;

import static org.junit.Assert.\*;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.Random;

import org.junit.Before;

import org.junit.FixMethodOrder;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.junit.runners.MethodSorters;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit4.SpringRunner;

import entity.User;

@SpringBootTest

@FixMethodOrder(MethodSorters.NAME\_ASCENDING)

public class TestRegistrationService {

private String generatePassword() {

Random rand = new Random();

int length = rand.nextInt((15 - 8) + 1) + 8;

int j;

String password = "";

for(j=0;j<length;j++) {

password = password + (char)(rand.nextInt((122 - 48) + 1) + 48);

}

return password;

}

private String generateUsername() {

Random rand = new Random();

int length = rand.nextInt((15 - 6) + 1) + 6;

int j;

String username = "";

for(j=0;j<length;j++) {

username = username + (char)(rand.nextInt((122 - 97) + 1) + 97);

}

return username;

}

@Test

public void testB() {

*//Step 1: Read TestFile.csv to fetch the username and password.*

try{

BufferedReader p = new BufferedReader(new InputStreamReader(new FileInputStream("/tmp/TestFile.csv")));

*// BufferedReader p = new BufferedReader(new InputStreamReader(new FileInputStream("e:\\TestFile.csv")));*

LoginService loginService = new LoginService();

String str;

String username,password;

while((str=p.readLine())!=null) {

String strArr[] = str.split(",");

username = strArr[0];

password = strArr[3];

*//Step 2: Pass the username and correct password to the checkpassword() method.*

Boolean flag = loginService.checkPassword(username, password);

*//Step 3: Assert the result.*

assertEquals("true",String.valueOf(flag));

*//Step 4: Pass the username and incorrect password to the checkpassword() method.*

password = this.generatePassword();

flag = loginService.checkPassword(username, password);

*//Step 5: Assert the result.*

assertEquals("false",String.valueOf(flag));

*//Step 6: Create some random usernames and passwords to the checkpassword() method.*

username = this.generateUsername();

flag = loginService.checkPassword(username, password);

*//Step 7: Assert the result.*

assertEquals("false",String.valueOf(flag));

}

}

catch(Exception e) {

e.printStackTrace();

fail(e.toString());

}

}

*// @Autowired*

*// private RegisterService registerService;*

*//*

private String getQuestion() {

String questions[] = {"What is your first school name?",

"Which city you were born in?",

"What is the first name of your best friend?"};

Random rand = new Random();

int index = rand.nextInt((2-0)+1) + 0;

return questions[index];

}

private String getName() {

Random rand = new Random();

int length = rand.nextInt((15 - 6) + 1) + 6;

int j;

String username = "";

for(j=0;j<length;j++) {

username = username + (char)(rand.nextInt((122 - 97) + 1) + 97);

}

return username;

}

private String getPassword() {

Random rand = new Random();

int length = rand.nextInt((15 - 8) + 1) + 8;

int j;

String password = "";

for(j=0;j<length;j++) {

password = password + (char)(rand.nextInt((122 - 48) + 1) + 48);

}

return password;

}

private String getEmail() {

int len1,len2,i,j;

String email;

email = "";

Random rand = new Random();

len1 = rand.nextInt((15-8)+1)+8;

len2 = rand.nextInt(10-7+1)+7;

for(i=0;i<len1;i++) {

email = email + (char)(rand.nextInt((122 - 97) + 1) + 97);

}

email = email+"@";

for(i=0;i<len1;i++) {

email = email + (char)(rand.nextInt((122 - 97) + 1) + 97);

}

email = email+".com";

return email;

}

*// @Before*

public void createTestCases(){

HashMap<String, Boolean> userMap= new HashMap<>();

int i;

String uname,fname,lname,pass,ques,ans,email;

String str;

try{

FileWriter file = new FileWriter("/tmp/TestFile.csv");

*// FileWriter file = new FileWriter("e:\\TestFile.csv");*

BufferedWriter bufferedWriter =

new BufferedWriter(file);

for(i=0;i<200;i++) {

str = "";

while(true) {

uname = this.getName();

if(userMap.containsKey(uname)) {

continue;

}

else {

userMap.put(uname, true);

break;

}

}

fname = this.getName();

lname = this.getName();

pass = this.getPassword();

ans = this.getName();

ques = this.getQuestion();

email = this.getEmail();

str = uname+','+fname+','+lname+','+pass+','+ans+','+ques+','+email+','+"true";

bufferedWriter.write(str);

bufferedWriter.newLine();

str = uname+','+fname+','+lname+','+pass+','+ans+','+ques+','+email+','+"false";

bufferedWriter.write(str);

bufferedWriter.newLine();

}

bufferedWriter.close();

}

catch (IOException e) {

fail("Cannot create the csv file.");

}

}

@Test

public void testA() {

RegisterService registerService = new RegisterService();

TestRegistrationService obj = new TestRegistrationService();

obj.createTestCases();

try {

BufferedReader p = new BufferedReader(new InputStreamReader(new FileInputStream("/tmp/TestFile.csv")));

*// BufferedReader p = new BufferedReader(new InputStreamReader(new FileInputStream("e:\\TestFile.csv")));*

String str;

while((str=p.readLine())!=null) {

String strArr[] = str.split(",");

User user = new User();

user.setUname(strArr[0]);

user.setFname(strArr[1]);

user.setLname(strArr[2]);

user.setPassword(strArr[3]);

user.setAnswer(strArr[4]);

user.setQuestion(strArr[5]);

user.setEmail(strArr[6]);

Boolean flag = registerService.register(user);

*// assertEquals(Boolean.parseBoolean(strArr[7]),flag);*

assertEquals(strArr[7],String.valueOf(flag));

}

}

catch(Exception e) {

fail("Exception occured while testing. : "+e);

}

}

}

1. Questions.java

package entity;

public class Questions {

private String question;

private String choice[] = new String[4];

private int answer;

private int id;

public String getQuestion() {

return question;

}

public void setQuestion(String question) {

this.question = question;

}

public String[] getChoice() {

return choice;

}

public void setChoice(String[] choice) {

this.choice = choice;

}

public int getAnswer() {

return answer;

}

public void setAnswer(int answer) {

this.answer = answer;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

}

1. Tests.java

package entity;

public class Tests {

private String name ="fas";

private int marks = 2;

private int time = 1;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getMarks() {

return marks;

}

public void setMarks(int marks) {

this.marks = marks;

}

public int getTime() {

return time;

}

public void setTime(int time) {

this.time = time;

}

}

1. User.java

package entity;

import java.util.ArrayList;

import org.w3c.dom.Node;

public class User {

private ArrayList<Tests> testList;

private String uname;

private String fname;

private String lname;

private String password;

private String question;

private String answer;

private String mobile;

private String email;

private int counter;

*//For use in updation code of profile*

private Node fiName;

private Node laName;

private Node mNumber;

private Node pass;

public void setFiName(Node fiName) {

this.fiName = fiName;

this.fiName.setTextContent(this.fname);

}

public void setLaName(Node laName) {

this.laName = laName;

this.laName.setTextContent(this.lname);

}

public void setmNumber(Node mNumber) {

this.mNumber = mNumber;

this.mNumber.setTextContent(this.mobile);

}

public void setPass(Node pass) {

this.pass = pass;

this.pass.setTextContent(this.password);

}

*//for use in updation code of profile end*

public String getUname() {

return uname;

}

public void setUname(String uname) {

this.uname = uname;

}

public String getFname() {

return fname;

}

public void setFname(String fname) {

this.fname = fname;

}

public String getLname() {

return lname;

}

public void setLname(String lname) {

this.lname = lname;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getQuestion() {

return question;

}

public void setQuestion(String question) {

this.question = question;

}

public String getAnswer() {

return answer;

}

public void setAnswer(String answer) {

this.answer = answer;

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public int getCounter() {

return counter;

}

public void setCounter(int counter) {

this.counter = counter;

}

public User() {

this.setUname("rajendar");

this.setFname("asf");

this.setLname("aszxvf");

this.setPassword("12412 ");

this.setQuestion("favourite animal?");

this.setAnswer("cat");

this.setMobile("98492316");

this.setEmail("dasf@gmail.com");

this.setCounter(0);

}

}

1. Build.gradle

buildscript {

ext {

springBootVersion = '1.5.16.RELEASE'

}

repositories {

mavenCentral()

}

dependencies {

classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")

}

}

apply plugin: 'java'

apply plugin: 'eclipse'

apply plugin: 'org.springframework.boot'

test {

testLogging {

exceptionFormat = 'full'

}

}

group = 'student.evaluation'

version = '0.0.1-SNAPSHOT'

sourceCompatibility = 1.8

repositories {

mavenCentral()

}

dependencies {

compile('org.springframework.boot:spring-boot-starter-web')

compile('org.springframework.boot:spring-boot-devtools')

compile('net.sourceforge.nekohtml:nekohtml')

compile('org.springframework.boot:spring-boot-starter-thymeleaf')

runtime('org.springframework.boot:spring-boot-devtools')

testCompile('org.springframework.boot:spring-boot-starter-test')

testCompile group: 'junit', name: 'junit', version: '4.12'

}

1. .travis.yml

language: java

jdk: oraclejdk8

branches:

only:

- gaurav

- ankit

- Test

deploy:

provider: heroku

app: evening-refuge-46144

api\_key:

secure: "1f3a7cae-76d9-45d9-b1fa-b3a8809d04d2"

on:

branch: Test