**CHAPTER 1**

**INTRODUCTION**

* 1. **Objective**

To provide the EAMCET counselling data to the aspirants based on the rank, category, district and branch.

* 1. **Purpose and Scope**

The main purpose of this project is to provide the Engineering, Agriculture and Medical aspirants with the data to fill the best options in official counselling site.

There is no need for the aspirants to go through the counselling books. There is no need for them to mine the data, create the list of options from those books and then fill the options. This work can be skipped and the required data can be retrieved from by using this application.

* 1. **Existing System**
* The existing systems contained of improper data.
* The complexity of existing systems is high.
* Some systems did not provide the facility to predict the colleges based on ranks.
* They are providing data of only top colleges.
* They ask for user registration, which is not necessary.
  1. **Proposed System**
* The application is exclusively only for AP EAMCET counselling.
* The application is provided with latest data.
* The application provides the data even for medical aspirants.
* The required data can be downloaded in the form of an excel document.
  1. **Advantages**
* Data based on selected district or based on the criteria can be retrieved and downloaded in the form of excel document.
* The latest data is present with the application.
* The data can be uploaded to database from an excel document.
  1. **System Requirements**

**1.6.1 Software Requirements**

* JDK 1.7
* Eclipse IDE
* Gradle
* MySQL

**1.6.2 Hardware Requirements**

|  |  |
| --- | --- |
| * RAM | : 1GB |
| * Hard Disk Space | : 20GB |
| * Processor | : Intel i3 or higher |

**CHAPTER 2**

**LITERATURE SURVEY**

This project is similar to college predictor applications which are present online. But, those applications are having only limited amount of data. Some do not predict colleges based on ranks.

**2.1 Some similar applications online**

* <http://www.myengg.com/>
* <http://www.engineering.careers360.com/colleges/ranking/2015/top-engineering-colleges-in-andhra-pradesh>
* <http://www.inspirenignite.com/eamcet-cutoff-and-possible-colleges/>
* <http://www.minglebox.com/engineering/colleges/exam/eamcet>

**2.2 Similar features in the existing applications**

* Provides the data of only top colleges in the state.
* Provides the data of other competitive examinations.

**2.3 Limitations of existing applications**

* They do not predict based on ranks.
* Complete data of a particular type of competitive exam is displayed to the user.

**CHAPTER 3**

**DESIGN PROCESS**

**3.1 Introduction to Unified Modelling Language (UML)**

The Unified Modelling Language (UML) allows the software engineer to express and analysis model using the modelling notations that is governed by set of syntactic, semantic and pragmatic rules. The UML is a language for:

* Visualizing
* Specifying
* Constructing
* Documenting

These are the artefacts of the software-intensive system.

**A conceptual model of UML**: The three major elements of UML are:

* The UML’s basic building blocks.
* The rules that dictate how those building blocks may be put together.
* Some common mechanisms that apply throughout the UML.

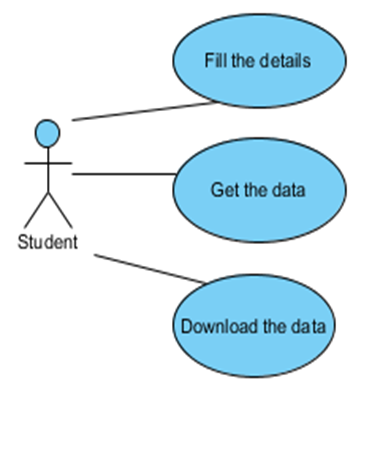
Each view is defined by a set of diagrams, which is as follows:

* **User Model View**
  + This view represents the system from the user’s perspective.
  + The analysis representation describes the usage scenario from the end-user perspective.
* **Structural Model View**
  + In this model the data and functionalities are arrived from inside the system.
  + This model view models the static structures.
* **Behavioural Model View**
  + It represents the dynamic behaviour as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.
* **Implementation Model View**
* In this the structural and behavioural as parts of the system are represented as they are to be built.
* **Environmental Model View**
  + In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**3.2 UML DIAGRAMS:**

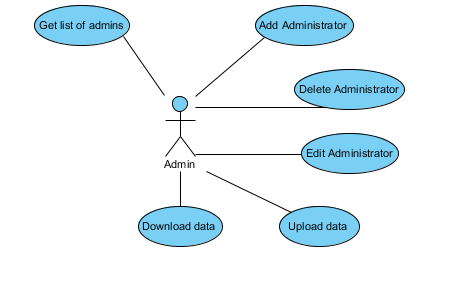
**3.2.1 Use Case Diagram**

The **Use Case diagram** is used in describing the behaviour of the target system from an external point of view. In addition to drawing the diagram, helps us to visualize the entire system and allow us to document the requirements.



**Fig. 3.1** Use case Diagram of Student

In the above use case diagram, the student must fill the details and submit to get the data. If the student is willing to download the data, they can do that.

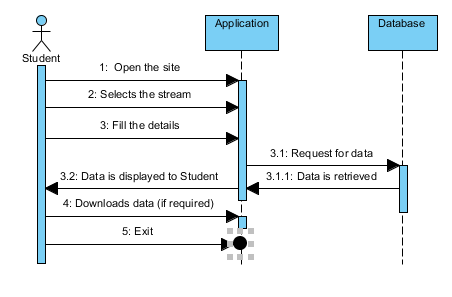


**Fig. 3.2** Use case Diagram of Admin

In the above use case diagram, the admin is able to check the list of admins, add a new admin, update or delete the existing admin(s), upload the data into the database, download the data present in database.

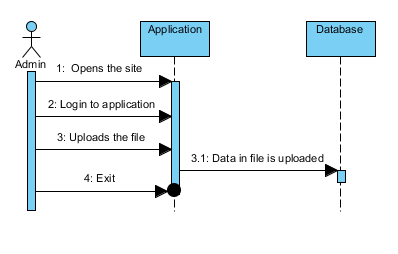
**3.2.2 Sequence Diagram**

A **Sequence diagram** is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



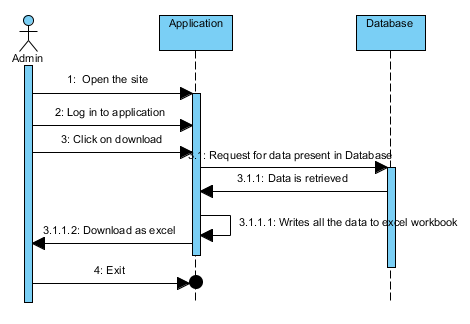
**Fig. 3.3** Sequence Diagram of Student

The above sequence diagram represents the order of interactions that a student has to do, to get the data. Here the user opens the site, selects the stream and submit the details. The data is retrieved on to the screen. If the student wants to download the data, he/she can do it.



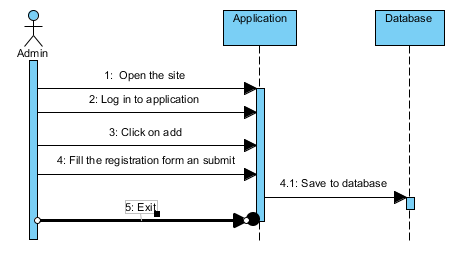
**Fig. 3.4** Sequence Diagram of Admin to upload data

The above sequence diagram is used to show the interactions to upload the content from excel sheet to the database table.



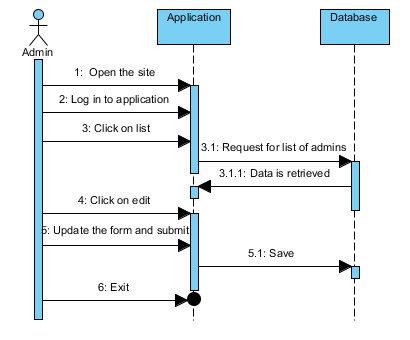
**Fig. 3.5** Sequence Diagram of Admin to download data

The above sequence diagram is used to show the interactions to download the content from database table to an excel file.



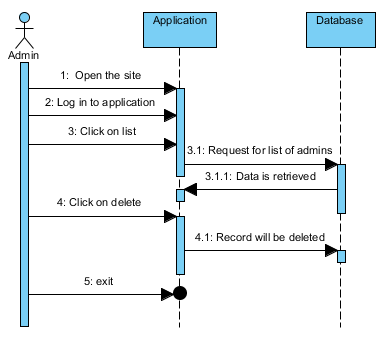
**Fig. 3.6** Sequence Diagram of Admin to add admin

The above sequence diagram is used to show the interactions to add a new admin for the application.



**Fig. 3.7** Sequence Diagram of Admin to update admin details

The above sequence diagram is used to show the interactions to update the details of existing admin.

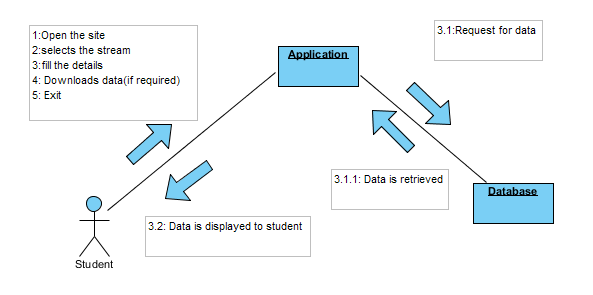


**Fig. 3.8** Sequence Diagram of Admin to delete admin

The above sequence diagram is used to show the interactions to remove the existing admin from the application.

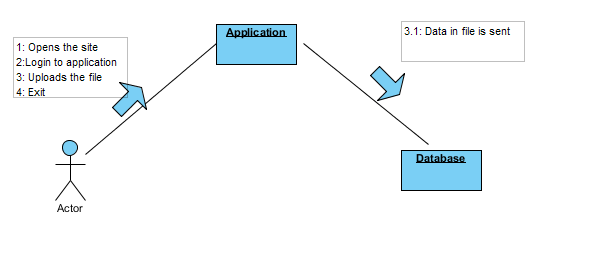
**3.2.3 Collaboration Diagram**

A Collaboration diagram is very similar to a Sequence diagram in the purpose it achieves; in other words, it shows the dynamic interaction of the objects in a system. A distinguishing feature of a Collaboration diagram is that it shows the objects and their association with other objects in the system apart from how they interact with each other.



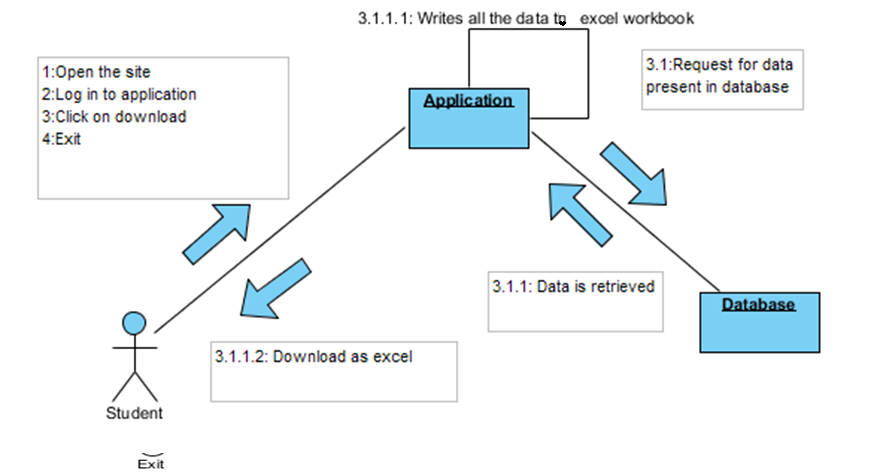
**Fig. 3.9** Collaboration Diagram of Student

The above collaboration diagram represents the interactions. Here the user opens the site, selects the stream and submit the details, download the data are the interactions done by the student. The request to retrieve data is done by the application.



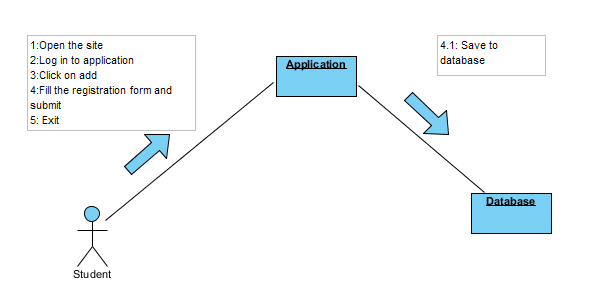
**Fig. 3.10** Collaboration Diagram of Admin to upload data

The above collaboration diagram is used to show the interactions to upload the content from excel sheet to the database table.



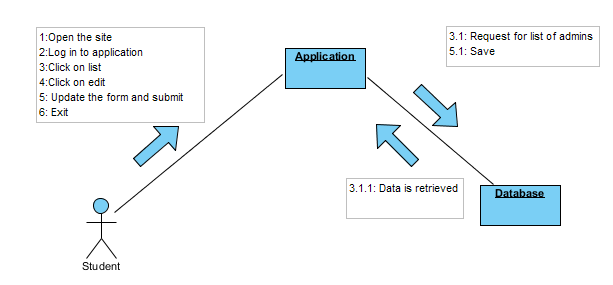
**Fig. 3.11** Collaboration Diagram of Admin to download data

The above collaboration diagram is used to show the interactions to download the content from database table to an excel file.



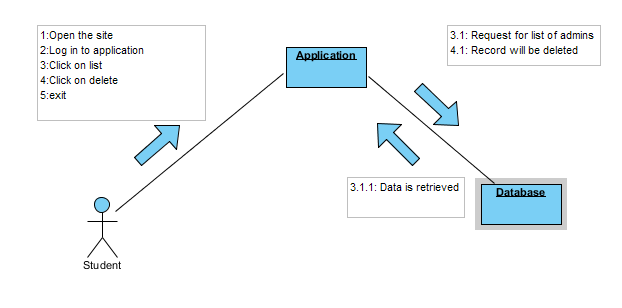
**Fig. 3.12** Collaboration Diagram of Admin to add admin

The above collaboration diagram is used to show the interactions to add a new admin for the application.



**Fig. 3.13** Collaboration Diagram of Admin to update details admin

The above collaboration diagram is used to show the interactions to update the details of existing admin

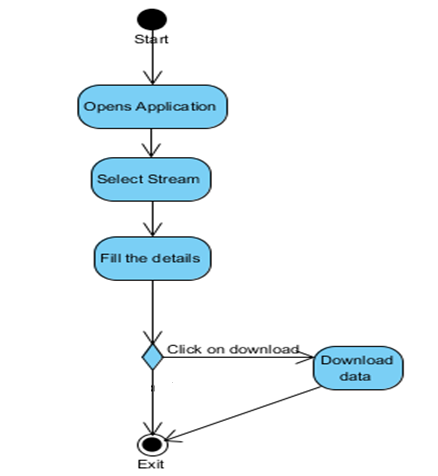


**Fig. 3.14** Collaboration Diagram of Admin to delete admin

The above collaboration diagram is used to show the interactions to remove the existing admin from the application.

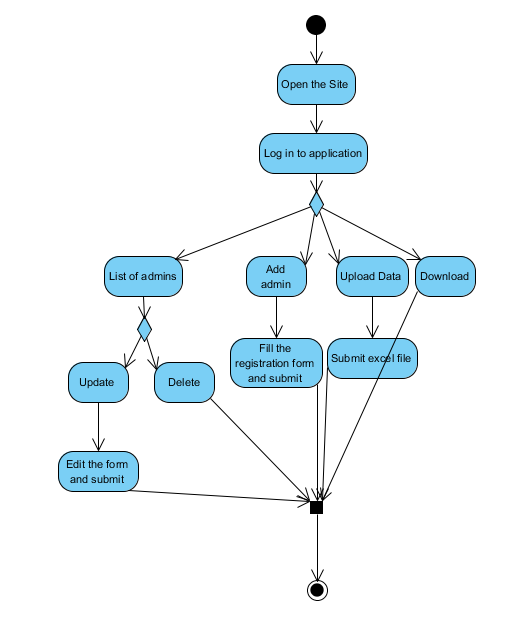
**3.2.4 Activity Diagram**

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram are not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim-lane etc.



**Fig. 3.15** Activity Diagram of student

The above activity diagram represents the control flow of actions that a student has to make to get the data.



**Fig. 3.16** Activity Diagram of admin

The above activity diagram represents the various control flow of actions that an admin can perform.

**CHAPTER 4**

**CONCEPTS AND IMPLEMENTATIONS**

**4.1 Programming Platforms**

The platforms used for the implementation of the project are described below:

1) JDK 1.7

2) Eclipse IDE

3) Gradle plugin in Eclipse

4) Spring boot

5) Java Persistence API

6) Thymeleaf

7) Twitter bootstrap

8) Apache POI

9) MySQL

* 1. **Java Development Kit (JDK) 1.7**

### **4.2.1 Evolution of JAVA**

Java is related to C++ which is direct decedent of C. Much of the character of Java is inherited from these two languages. From C, Java derives syntax. Many of Java’s Object- Oriented features were influenced by C++. In fact, several of Java defining characteristics come from or responses to its predecessors.

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Micro Systems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called “OAK”, but was renamed as “JAVA” in 1995. Between the initial implementation of OAK in the fall of 1992 and the public announcement of Java in the spring of 1995, many more people contributed to this design and the evolution of the language. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin and Tim Lidholm were key contributors to the maturing of original prototype.

The original impetus for Java was not the Internet! Instead, the primary motivation was the need for a platform – independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls. The trouble with C and C++ is that they are designed to be compiled for a specific target. Here the problem is that the cost of compilers is very expensive and time-consuming to create. So an easier and most efficient solution was needed.

In attempt to find such a solution, Gosling and others began to work on a portable, platform- independent language that could be used to produce code that would run on a variety of CPUs under differing environments. This effort is ultimately led to the creation of Java.

Here the output of JAVA compiler is not executable code. Rather, it is byte code. Byte code is a highly optimized set of instructions designed to be executed by Java run-time system, which is called the Java Virtual Machine (JVM). In essence, the original JVM was designed as an interpreter for byte code. Here the modern languages are designed to be compiled into executable code because of performance concerns. However, the fact that a Java Program is executed by the JVM helps solve the major problems associated with web- based programs. Because translating a Java program into byte code makes it much easier to run a program in a variety of environments because only the JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Here the JVM differs for platform to platform, all understand the same Java byte code. If a Java program were compiled to native code, then different versions of the same program would have to exist for each type of CPU connected to the Internet. The fact is that the java program which is executed by JVM also helps to make it secure.

Some of the versions of java:

|  |  |  |
| --- | --- | --- |
| **Java Version** | **Release date** | **Year** |
| JDK 1.0 | January 21 | 1996 |
| JDK 1.1 | February 19 | 1997 |
| J2SE 1.2 | December 8 | 1998 |
| J2SE 1.3 | May 8 | 2000 |
| J2SE 1.4 | February 6 | 2002 |
| J2SE 5.0 | September 30 | 2004 |
| Java SE 6 | December 11 | 2006 |
| Java SE 7 | July 28 | 2011 |

#### Table 5.1: Different release versions of JAVA

### 4.2.1.1 Features of JAVA

#### Simple

* + - * Java is Easy to write and more readable and eye catching.
      * Java has a concise, cohesive set of features that makes it easy to learn and use.
      * Most of the concepts are drew from C++ thus making Java learning simpler.

#### Secure

* + - * Java program cannot harm other system thus making it secure.
      * Java provides a secure means of creating Internet applications.
      * Java provides secure way to access web applications.

#### Portable

* + - * Java programs can execute in any environment for which there is a Java run-time system.(JVM)
      * Java programs can be run on any platform (Linux, Window, and Mac).
      * Java programs can be transferred over world wide web (e.g. applets)

#### Object-oriented

* + - * Java programming is object-oriented programming language.
      * Like C++ java provides most of the object oriented features.
      * Java is pure OOP. Language. (while C++ is semi object oriented)

#### Robust

* + - * Java encourages error-free programming by being strictly typed and performing run- time checks.

#### Multithreaded

* + - * Java provides integrated support for multithreaded programming.

#### Architecture-neutral

* + - * Java is not tied to a specific machine or operating system architecture.
      * Machine Independent i.e. Java is independent of hardware.

#### Interpreted

* + - * Java supports cross-platform code through the use of Java byte code.
      * Byte code can be interpreted on any platform by JVM.

#### High performance

* + - * Byte codes are highly optimized.
      * JVM can execute them much faster.

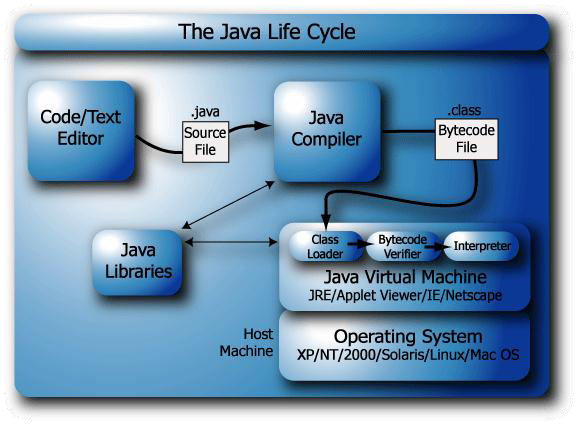
#### Distributed

* + - * Java was designed with the distributed environment.
      * Java can transmit, run over internet.

#### Dynamic

* + - * Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time.

### **4.2.3. Life Cycle of JAVA**



**Fig. 5.1** Java Life Cycle

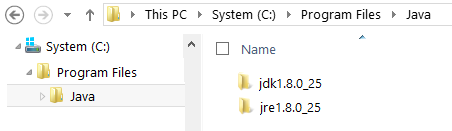
**4.2.4: Installation of JDK:**

**Step 1:** Download JDK

* Goto JavaSE download site: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
* Under "Java Platform, Standard Edition" ⇒ "Java SE 8u{xx}" ⇒ Click the "JDK Download" button.
* Check "Accept License Agreement".
* Choose your operating platform, e.g., "Windows x64 for 64-bit Windows OS" or "Windows x32 for 32-bit Windows OS". You can check whether your Windows OS is 32-bit or 64-bit via "Control Panel" ⇒ System ⇒ "System Type".

**Step 2:** Install JDK and JRE

* Run the downloaded installer (e.g., "jdk-8u{xx}-windows-x64.exe"), which installs both the JDK and JRE. By default, the JDK will be installed in directory "C:\Program Files\Java\jdk1.8.0\_xx", where xx denotes the latest upgrade number; and JRE in "C:\Program Files\Java\jre1.8.0\_xx".
* For novices, accept the defaults. Follow the screen instructions to install JDK and JRE.
* Take note of your JDK installed directory (which you will need to use in the next step). Check the JDK installed directory by inspecting these folders using Windows' Explorer.



**Fig. 4.2** Location of Java

**Step 3:** Include JDK's "bin" Directory in the PATH

* Windows OS searches the current directory and the directories listed in the PATH environment variable for executable programs. JDK's programs (such as Java compiler javac.exe and Java runtime java.exe) reside in directory "<JAVA\_HOME>\bin" (where <JAVA\_HOME> denotes the JDK installed directory). You need to include the "<JAVA\_HOME>\bin" in the PATH.
* To edit the PATH environment variable in Windows XP/Vista/7/8/10:
* Control Panel ⇒ System ⇒ Advanced system settings
* Switch to "Advanced" tab ⇒ Environment Variables
* In "System Variables", scroll down to select "PATH" ⇒ Edit
* In "Variable value" field, INSERT "c:\Program Files\Java\jdk1.8.0\_xx\bin" (Replace xx with your installation upgrade number!!!) IN FRONT of all the existing directories, followed by a semi-colon (;) which separates the JDK's binary directory from the rest of the existing directories.  
  DO NOT DELETE any existing entries; otherwise, some existing applications may not run.
* Variable name : PATH
* Variable value : c:\Program Files\Java\jdk1.8.0\_xx\bin;[exiting entries...]
* For Latest Windows 10: You see a table listing the existing entries. Click "New" ⇒ Enter the JDK's binary directory "c:\Program Files\Java\jdk1.8.0\_xx\bin" (Replace xx with your installation upgrade number!!!) ⇒ Select "Move Up" to move the JDK's binary directory to the top.

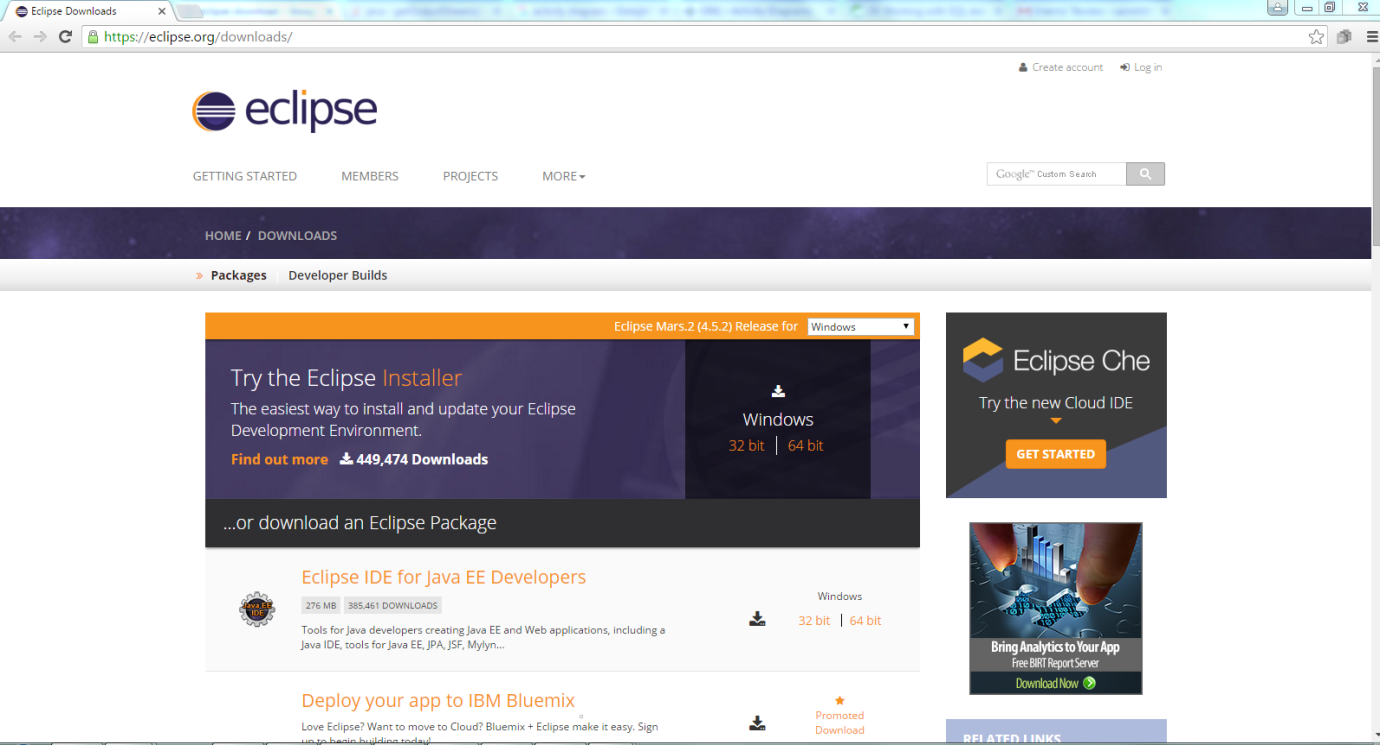
**Step 4:** Verify the JDK Installation

* Launch a CMD shell (Click "Start" button ⇒ run... ⇒ enter "cmd"; OR from "Start" button ⇒ All Programs ⇒ Accessories ⇒ Command Prompt).
* Issue "path" command to list the contents of the PATH environment variable. Check to make sure that your <JAVA\_HOME>\bin is listed in the PATH.
* prompt> path
* PATH=c:\Program Files\Java\jdk1.8.0\_xx\bin;[other entries...]
* Don't type “prompt>”, which denotes the command prompt. Key in the command only.
* Issue the following commands to verify that JDK/JRE are properly installed and display their version:
* prompt> java -version
  + java version "1.8.0\_xx"
  + Java(TM) SE Runtime Environment (build 1.8.0\_xx-b13)
  + Java HotSpot(TM) 64-Bit Server VM (build 25.5-b02, mixed mode)

**4.3Eclipse IDE:**

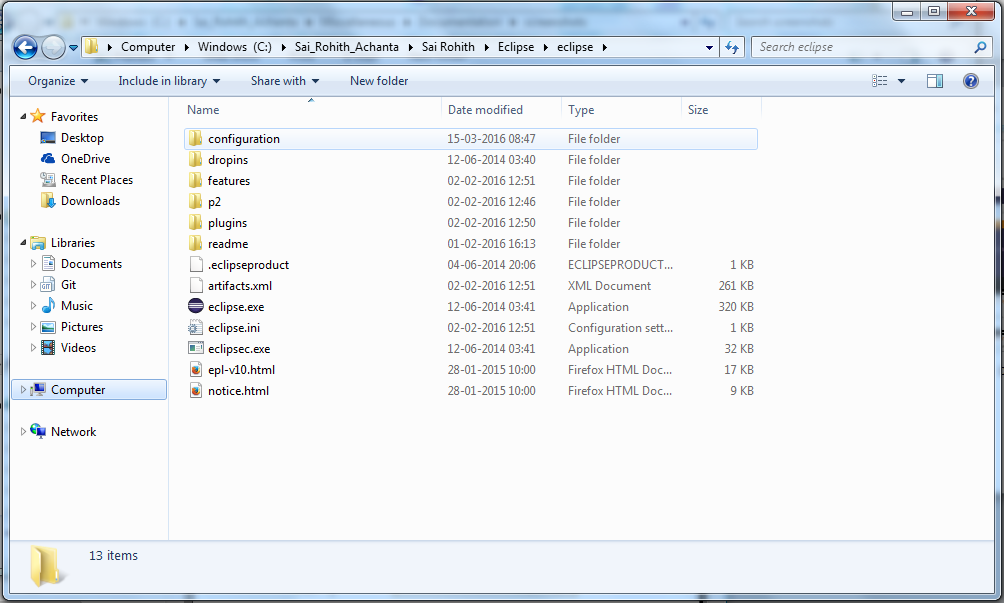
**Step 1:** To use Eclipse for Java programming, you need to first install Java Development Kit

(JDK). Refer 5.2 for JDK installation.



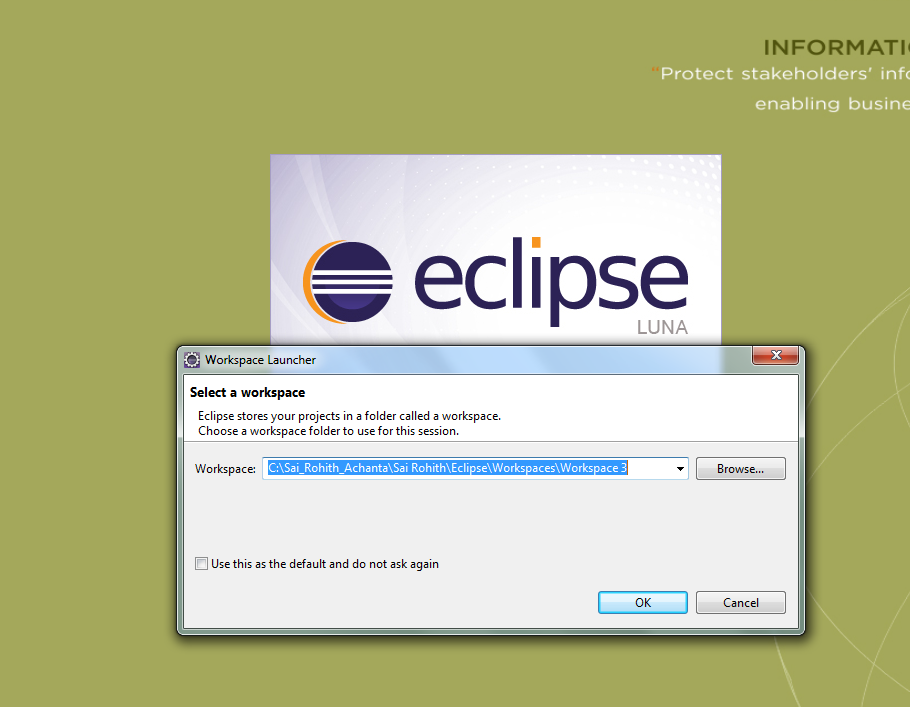
**Fig. 4.3** Download Eclipse IDE

**Step 2:** To install Eclipse, simply unzip the downloaded file into a directory of your choice.



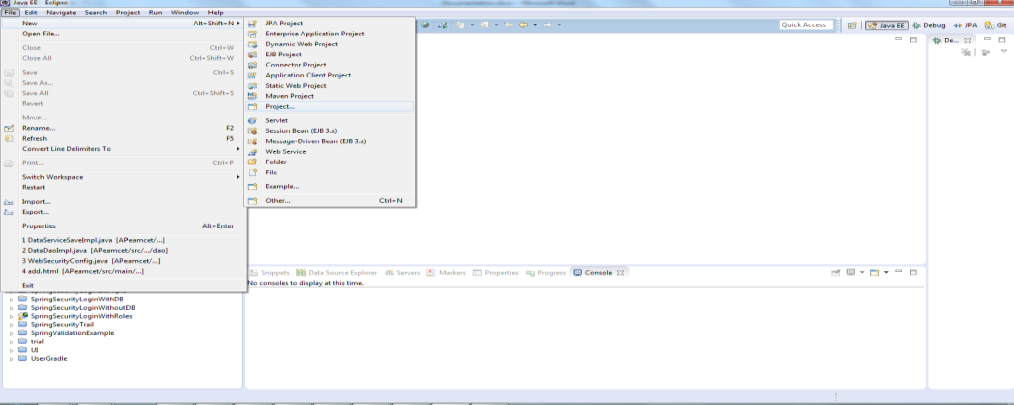
**Fig. 4.4** Unzip file and set IDE location

**Step 3:** Choose a path to save all projects into your workspace.



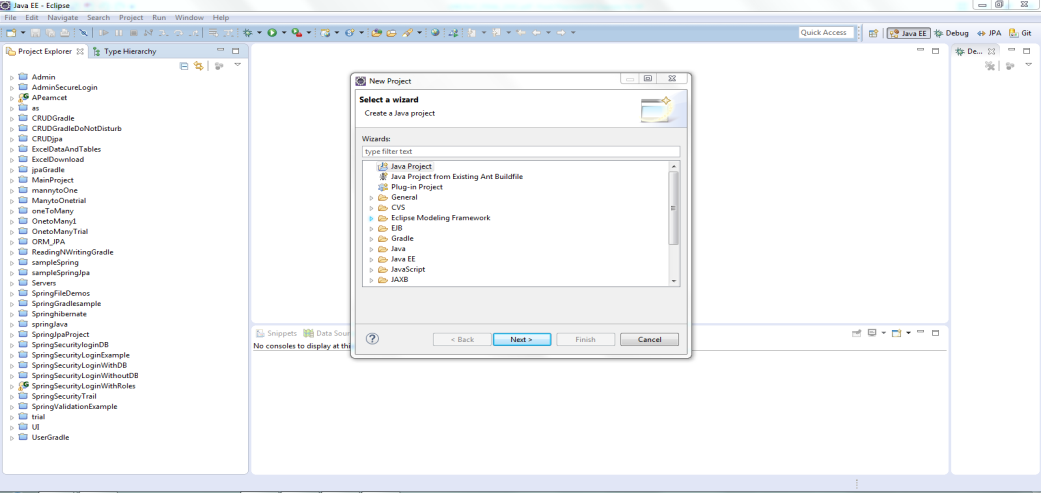
**Fig. 4.5** selecting the workspace

**Step 4:** Test eclipse by writing a sample Java Application.

****

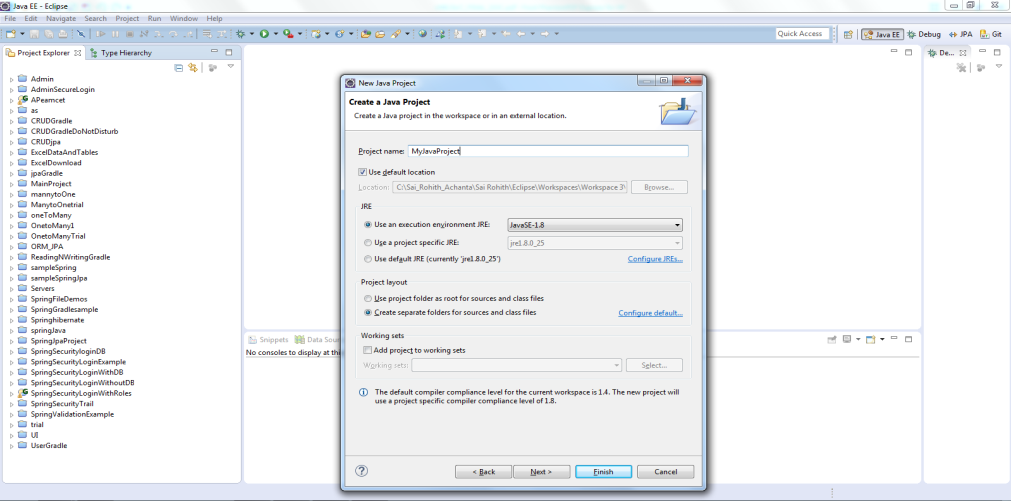
**Fig. 4.6** creating a new project

**Step 5:** Choose Java project and click “Next”.

****

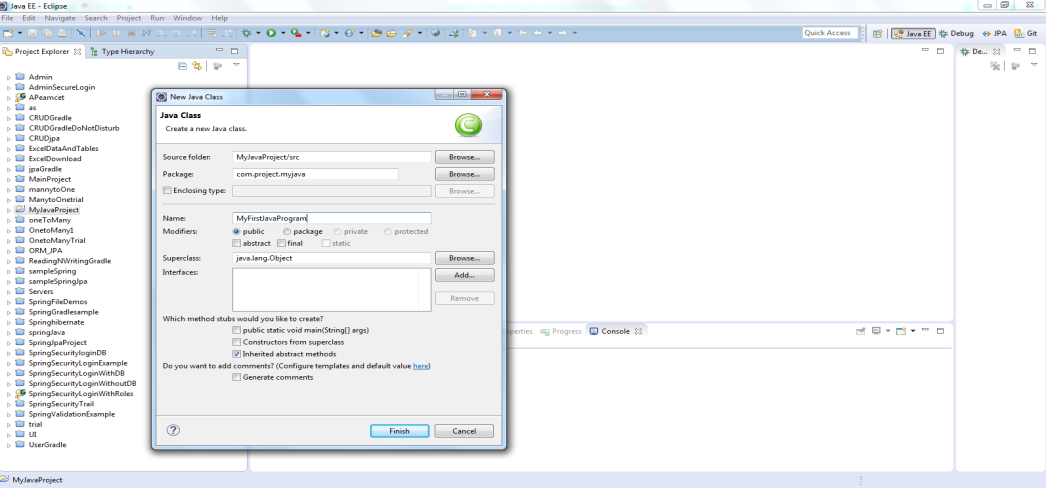
**Fig. 4.7** selecting a java project

**Step 6:**  Choose a Project name and Click “Finish”.

****

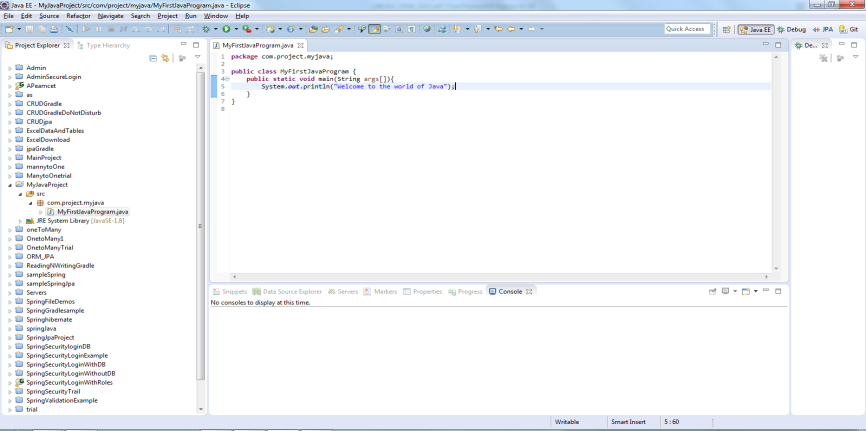
**Fig 4.8** Naming the project

**Step 7:** Give a package name and a class name and click “Finish”.

****

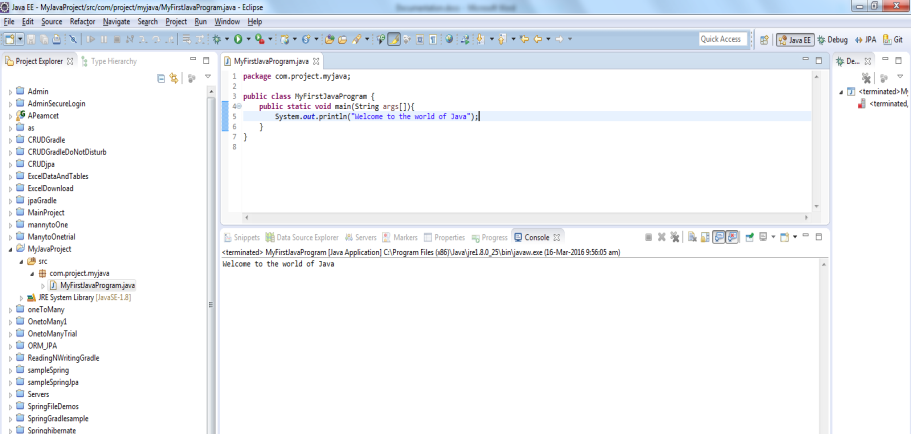
**Fig 4.9** creating a package and a class in it.

**Step 8:**  Write a sample Java program to test the IDE.



**Fig 4.10** A sample Java program.

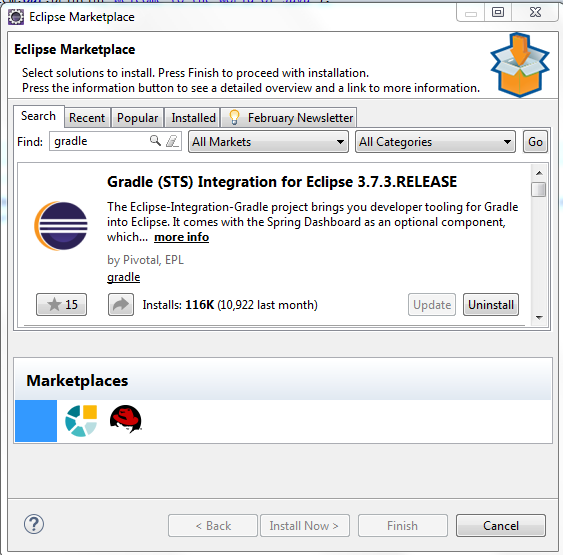
**Step 9:**  Save the code and run the program.



**Fig 4.11** Output console

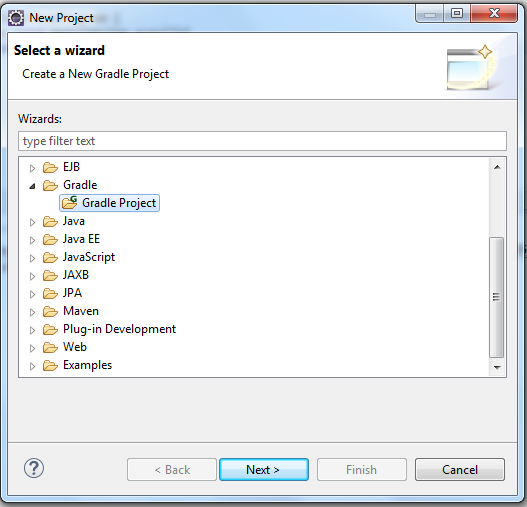
**4.4 Adding Gradle plug-in to Eclipse**

**Step 1:** Launch Eclipse IDE -> click on Help -> select Eclipse Marketplace. Search for gradle and install it.



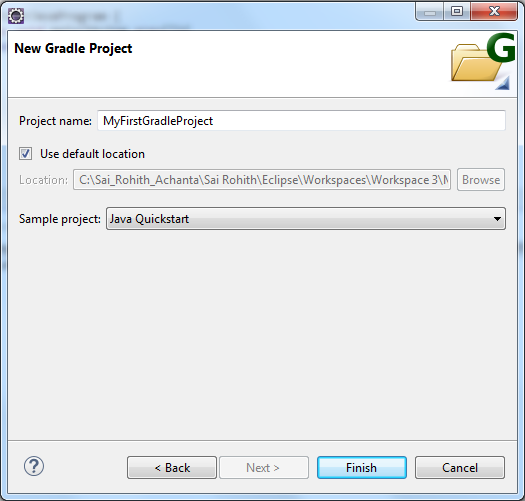
**Fig 4.12** Eclipse marketplace

**Step 2:** Now in create a new project window check for “Gradle” followed by “Gradle Project”.



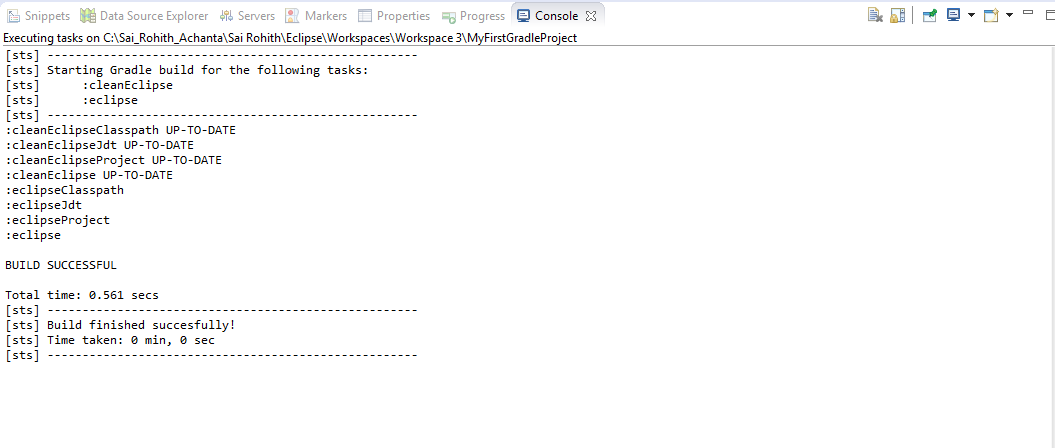
**Fig 4.13** Verifying Gradle Installation

**Step 3:**  Creating a gradle project.



**Fig 4.14** Naming the project

**Step 5:** Now the console should display following output.



**Fig 4.15** Output console

* 1. **Spring framework**

Spring is a *lightweight* 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, AOP, DAO, Context, ORM, WEB MVC etc.

### **4.5.1 Advantages of Spring Framework:**

There are many advantages of Spring Framework. They are as follows:

#### 1) Predefined Templates

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

#### 2) Loose Coupling

The Spring applications are loosely coupled because of dependency injection.

#### 3) Easy to test

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

**4) Lightweight**

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

#### 5) Fast Development

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

#### 6) Powerful abstraction

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

#### 7) Declarative support

It provides declarative support for caching, validation, transactions and formatting.

**4.5.2 Dependency Injection:**

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

### **4.5.2.1 Two ways to perform Dependency Injection in Spring framework**

Spring framework provides two ways to inject dependency

* Constructor Injection
* Setter Injection

**4.5.2.1.1 Constructor Injection**

We can achieve constructor injection, by creating constructors in the model class. The constructors can be an empty constructor or constructors with different signatures.

**4.5.2.1.2 Setter Injection**

We can achieve Setter Injection, by creating the getters and setters for the fields present in the model class.

**4.5.3 Spring Security:**

Spring Security is a Java/Java EE framework that provides authentication, authorization and other security features for enterprise applications.

* **Authentication**

**Authentication** is any process by which a system verifies the identity of a User who wishes to access it. Since Access Control is normally based on the identity of the User who requests access to a resource.

* **Authorization**

Authorization is the process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use.

**4.5.3.1 OWASP:**

The **Open Web Application Security Project** (**OWASP**) is an online community which creates freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security.

The Spring Security protects the application from top 10 OWASP vulnerabilities.

The OWASP Top 10 - 2013 is as follows:

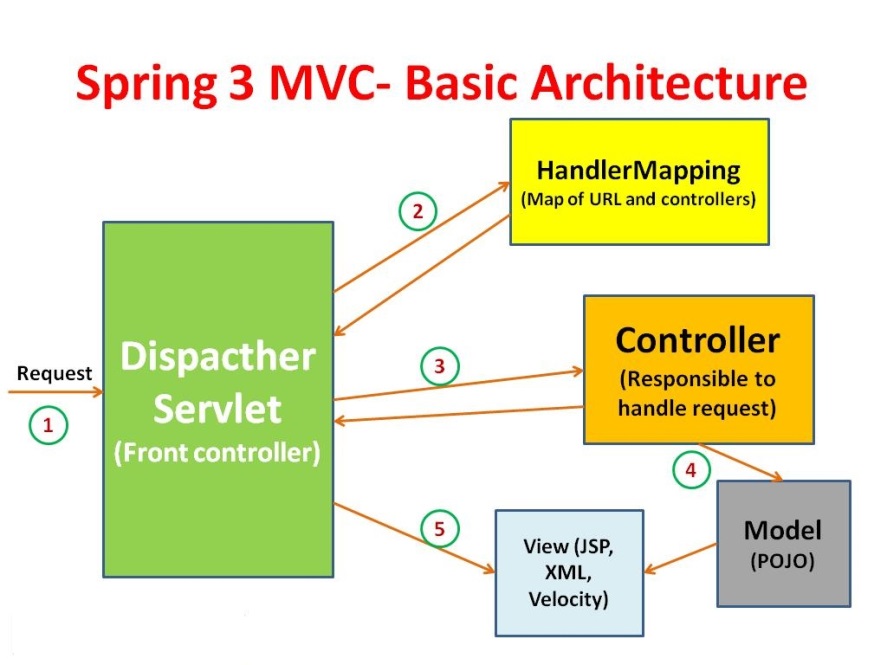
* A1 Injection
* A2 Broken Authentication and Session Management
* A3 Cross-Site Scripting (XSS)
* A4 Insecure Direct Object References
* A5 Security Misconfiguration
* A6 Sensitive Data Exposure
* A7 Missing Function Level Access Control
* A8 Cross-Site Request Forgery (CSRF)
* A9 Using Components with Known Vulnerabilities
* A10 Un validated Redirects and Forwards

**4.5.4 Spring MVC:**

In Spring Web MVC, DispatcherServlet class works as the front controller. It is responsible to manage the flow of the spring mvc application.

The @Controller annotation is used to mark the class as the controller in Spring 3.

The @RequestMapping annotation is used to map the request url. It is applied on the method.



**Fig. 4.16** Spring MVC Architecture

As displayed in the figure, all the incoming requests are intercepted by the DispatcherServlet that works as the front controller. The DispatcherServlet gets entry of handler mapping from the xml file and forwards the request to the controller. The controller returns an object of ModelAndView. The DispatcherServlet checks the entry of view resolver in the xml file and invokes the specified view component.

**4.5.5 Spring Data JPA:**

Spring Data JPA API provides JpaTemplate class to integrate spring application with JPA. JPA (Java Persistent API) is the sun specification for persisting objects in the enterprise application. It is currently used as the replacement for complex entity beans.

The implementations of JPA specification are provided by many vendors such as:

* Hibernate
* Toplink
* iBatis
* OpenJPA etc.

## 4.5.5.1 Advantage of Spring JpaTemplate

You don't need to write the before and after code for persisting, updating, deleting or searching object such as creating Persistence instance, creating EntityManagerFactory instance, creating EntityTransaction instance, creating EntityManager instance, commiting EntityTransaction instance and closing EntityManager.

**4.5.5.2 JPA Repository**

The JPA module of Spring Data contains a custom namespace that allows defining repository beans. It also contains certain features and element attributes that are special to JPA.

**4.5.5.2.1 Query Methods**

* **Default Queries**

The JPA Repository provides certain query methods. The query methods can be like “findAll(); , save(Entity entity); deleteAll(); saveAndFlush(Entity entity); etc..,”.

* **Query Creation**

Besides the presence of default query methods, the spring is provided with the facility to generate or create query methods for the programmer.

The supporting keywords for query creation are as follows.

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Sample** | **JPQL snippet** |
| And | findByLastnameAndFirstname | … where x.lastname = ?1 and x.firstname = ?2 |
| Or | findByLastnameOrFirstname | … where x.lastname = ?1 or x.firstname = ?2 |
| Between | findByStartDateBetween | … where x.startDate between 1? and ?2 |
| LessThan | findByAgeLessThan | … where x.age < ?1 |
| GreaterThan | findByAgeGreaterThan | … where x.age > ?1 |
| After | findByStartDateAfter | … where x.startDate > ?1 |
| Before | findByStartDateBefore | … where x.startDate < ?1 |
| IsNull | findByAgeIsNull | … where x.age is null |
| IsNotNull,NotNull | findByAge(Is)NotNull | … where x.age not null |
| Like | findByFirstnameLike | … where x.firstname like ?1 |
| NotLike | findByFirstnameNotLike | … where x.firstname not like ?1 |
| StartingWith | findByFirstnameStartingWith | … where x.firstname like ?1 (parameter bound with appended %) |
| EndingWith | findByFirstnameEndingWith | … where x.firstname like ?1 (parameter bound with prepended %) |
| Containing | findByFirstnameContaining | … where x.firstname like ?1 (parameter bound wrapped in %) |
| OrderBy | findByAgeOrderByLastnameDesc | … where x.age = ?1 order by x.lastname desc |

**Table 4.2** Keywords for Query creation

**4.6 Thymeleaf**

Thymeleaf is a Java library. It is a template engine capable of processing and generating HTML, XML, JavaScript, CSS and text, and can work both in web and non-web environments. It is better suited for serving the view layer of web applications, but it can process files in many formats, even in offline environments.

It provides an optional module for integration with Spring MVC, so that you can use it as a complete substitute of JSP in your applications made with this technology, even with HTML5.

The main goal of Thymeleaf is to provide an elegant and well-formed way of creating templates. Its Standard and Spring Standard dialects allow you to create powerful natural templates that can be correctly displayed by browsers and therefore work also as static prototypes. You can also extend Thymeleaf by developing your own dialects.

Some Thymeleaf attributes are as follows:

|  |  |
| --- | --- |
| Attribute | Purpose |
| th:if=”${expression}” | If condition |
| th:field=”\*{fieldname}” | Take the data from field |
| th:each=”objectname : ${objectlist}” | Iterates to display data on view |
| th:text=”${message}” | Displays the message sent by controller |
| th:rows=”${name}” | Used to get the total number of rows |
| th:action=”@{url}” | Requests to the url (POST) |
| th:href=”${url}” | Requests to the url (GET) |
| th:object=”${objectname}” | Sets the object reference to view |

**Table 4.3** Some basic attributes in thymeleaf

**4.7 Twitter bootstrap**

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications.

Bootstrap is a front end web framework, that is, an interface for the user, unlike the server-side code which resides on the "back end" or server.

**4.7.1 Structure and function**

Bootstrap is modular and consists essentially of a series of [LESS style sheets](https://en.wikipedia.org/wiki/LESS_(stylesheet_language)) that implement the various components of the toolkit. A style sheet called bootstrap less includes the components style sheets. Developers can adapt the Bootstrap file itself, selecting the components they wish to use in their project.

The use of LESS style sheet language allows the use of variables, functions and operators, nested selectors, as well as so-called [mix-ins](https://en.wikipedia.org/wiki/Mixin).

Since version 2.0, the configuration of Bootstrap also has a special "Customize" option in the documentation. Moreover, the developer chooses on a form the desired components and adjusts, if necessary, the values of various options to their needs. The subsequently generated package already includes the pre-built CSS style sheet.

Grid system and responsive design comes standard with a 1170 pixel wide, grid layout. Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions and types of devices: mobile phones, portrait and landscape, tablets and PCs with low and high resolution. Each variation adjusts the width of the columns.

### **4.7.2 Style sheets**

Bootstrap provides a set of style sheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements.

### **4.7.3 Re-usable components**

In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. These include buttons with advanced features (e.g. grouping of buttons or buttons with drop-down option, make and navigation lists, horizontal and vertical tabs, navigation, breadcrumb navigation, pagination, etc.), labels, advanced typographic capabilities, thumbnails, warning messages and a progress bar. The components are implemented as CSS classes, which must be applied to certain HTML elements in a page.

### **4.7.4 JavaScript components**

Bootstrap comes with several JavaScript components in the form of [jQuery](https://en.wikipedia.org/wiki/JQuery" \o "JQuery) plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields. In version 2.0, the following JavaScript plugins are supported: Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel and Type ahead.

* 1. **Apache POI**

Apache POI is a popular API that allows programmers to create, modify, and display MS Office files using Java programs. It is an open source library developed and distributed by Apache Software Foundation to design or modify Microsoft Office files using Java program. It contains classes and methods to decode the user input data or a file into MS Office documents.

* + 1. **Workbook**

This is the super-interface of all classes that create or maintain Excel workbooks. It belongs to the **org.apache.poi.ss.usermodel** package. The two classes that implement this interface are as follows:

* **HSSFWorkbook** : This class has methods to read and write Microsoft Excel files in .xls format. It is compatible with MS-Office versions 97–2003.
* **XSSFWorkbook** : This class has methods to read and write Microsoft Excel and OpenOffice xml files in .xls or .xlsx format. It is compatible with MS-Office versions 2007 or later.
  + 1. **Sheet**

Sheet is an interface under the org.apache.poi.ss.usermodel package and it is a super-interface of all classes that create high or low level spreadsheets with specific names. The most common type of spreadsheet is worksheet, which is represented as a grid of cells.

* **HSSFSheet:** This is a class under the org.apache.poi.hssf.usermodel package. It can create excel spreadsheets and it allows to format the sheet style and sheet data.
* **XSSFSheet:** This is a class which represents high level representation of excel spreadsheet. It is under org.apache.poi.hssf.usermodel package.
  + 1. **Row**

This is an interface under the org.apache.poi.ss.usermodel package. It is used for high-level representation of a row of a spreadsheet. It is a super-interface of all classes that represent rows in POI library.

* **XSSFRow:** This is a class under the org.apache.poi.xssf.usermodel package. It implements the Row interface, therefore it can create rows in a spreadsheet. Listed below are the methods and constructors under this class.
  + 1. **Cell**

This is an interface under the org.apache.poi.ss.usermodel package. It is a super-interface of all classes that represent cells in the rows of a spreadsheet.

Cells can take various attributes such as blank, numeric, date, error, etc. Cells should have their own numbers (0 based) before being added to a row.

* **XSSFCell:** This is a class under the org.apache.poi.xssf.usermodel package. It implements the Cell interface. It is a high-level representation of cells in the rows of a spreadsheet.
  1. **MySQL**

MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications.

## 4.9.1 MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**CHAPTER 5**

**CODING**

**5.1 Introduction:**

**5.1.1: Domain classes:**

The domain classes are the Plain Old Java Objects and they are the replica of the database table’s columns. The domain classes consists of variables, getters and setters and constructors and annotations like @Entity, @Id, @Column etc.., The annotations are present in “javax.persistence.\*” library.

**5.1.1.1: Data Domain class:**

@Entity

**public** **class** Data {

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**int** id;

@Column

String code;

@Column

String institutename;

@Column

String coursecode;

@Column

String region;

@Column

String district;

@Column

String place;

@Column

Integer ocb;

@Column

Integer ocg;

@Column

Integer scb;

@Column

Integer scg;

@Column

Integer stb;

@Column

Integer stg;

@Column

Integer bcab;

@Column

Integer bcag;

@Column

Integer bcbb;

@Column

Integer bcbg;

@Column

Integer bccb;

@Column

Integer bccg;

@Column

Integer bcdb;

@Column

Integer bcdg;

@Column

Integer bceb;

@Column

Integer bceg;

@Column

String type;

@Column

**int** year;

//Constructors, getters and setters  
}

**5.1.1.2: Role Domain class:**

@Entity

**public** **class** Role {

@Id

**int** roleid;

@Column

String rolename;

//Constructors, getters and setters  
 }

**5.1.1.3: User Domain class:**

@Entity

**public** **class** User {

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**int** id;

@Column

String username;

@Column

String password;

@Column

String email;

@Column

String contact;

@ManyToOne  
Role role;

//Constructors, getters and setters  
}

**5.1.2: User Model classes:**

The model classes are the also the Plain Old Java Objects, but these are used to take the data from the view layer (User Interface). The members of these classes are same as the members of the domain classes but the difference between domain classes and model classes arises at the annotations. The model classes consists of @Max, @Min, @Email, @NotEmpty etc.., The annotations are present in ” org.hibernate.validator.constraints” library.

**5.1.2.1: User Model Class:**

**public** **class** UserModel {

**int** id;

@Range(min=5,max=20)

String username;

@NotEmpty

String password;

@NotEmpty

String confirmPassword;

@Email

String email;

@NotEmpty

String contact;

**int** role;  
 String rolename;

//Constructors, getters and setters  
 }

**5.2 Code to save/update a record:**

**5.2.1: View:**

The data from the user is taken from the view and it is set to the model object. On submitting the form, the request mapping is done and the control goes to the controller.

<form method=*"post"* th:action=*"@{/submit}"* th:object=*"${userPojo}"* action=*"#"*>

<label for=*"username"* class=*"col-sm-2 control-label text-right"*>UserName:</label>

<input type=*"text"* class=*"form-control"* th:field=*"\*{username}"* id=*"username"* minlength=*"7"* name=*"username"* placeholder=*"username"* required=*"required"* />

<label for=*"password"* class=*"col-sm-2 control-label text-right"*>Password:</label>

<input type=*"password"* class=*"form-control"* th:field=*"\*{password}"* id=*"password"* minlength=*"7"* name=*"password"* placeholder=*"password"* required=*"required"*/>

<label for=*"confirmPassword"* class=*"col-sm-2 control-label text-right"*>Confirm Password:</label>

<input type=*"password"* class=*"form-control"* th:field=*"\*{confirmPassword}"* id=*"confirmPassword"* minlength=*"7"* name=*"confirmPassword"* placeholder=*"confirm password"* required=*"required"* oninput=*"check(this)"*/>

<label for=*"email"* class=*"col-sm-2 control-label text-right"*>E-mail:</label>

<input type=*"email"* class=*"form-control"* th:field=*"\*{email}"* id=*"email"* name=*"email"* placeholder=*"email"* required=*"required"*/>

<label for=*"contact"* class=*"col-sm-2 control-label text-right"*>Contact:</label>

<input type=*"tel"* class=*"form-control"* th:field=*"\*{contact}"* id=*"contact"* name=*"contact"* placeholder=*"contact"* required=*"required"* />

<label for=*"role"* class=*"col-sm-2 control-label text-right"*>Role:</label>

<select th:field=*"\*{role}"* id=*"role"* name=*"role"* class=*"form-control"*>

<option value=*"2"* name=*"Admin"*>Admin</option>

</select>

<button type=*"submit"* class=*"btn btn-default"*>Add</button>

<input type=*"reset"* class=*"btn btn-default*

</form>

**5.2.2: Controller:**

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

**public** String edit(@ModelAttribute UserModel userPojo, BindingResult result,Map<String, Object> map, Model model){

**if**(result.hasErrors()){

**return** Constants.***VIEW\_ADDRUPDATE***;

}

userService.edit(userPojo);

List<UserModel> userPojoList=userService.getAll();

map.put(Constants.***CONST\_USER\_LIST***,userPojoList );

model.addAttribute(Constants.***CONST\_USER\_COUNT***,userPojoList.size());

**return** Constants.***VIEW\_ADMINHOME***;

}

**5.2.3: Service:**

@Transactional

**public** **void** edit(UserModel userPojo) {

User user= **new** User();

user.setUsername(userPojo.getUsername());

user.setPassword(userPojo.getPassword());

user.setContact(userPojo.getContact());

user.setEmail(userPojo.getEmail());

user.setId(userPojo.getId());

**int** roleid=userPojo.getRole();

Role role= **new** Role();

role=roleDao.getUserRole(roleid);

user.setRole(role);

userDao.edit(user);

}

**5.2.4: Data Access Object (DAO):**

**public** **void** edit(User user) {

userRepo.saveAndFlush(user);

}

**5.2.5: Repository:**

**public** **interface** UserRepo **extends** JpaRepository<User, Integer> {

}

**5.3: Retrieve data:**

**5.3.1: View:**

<a th:href=*"@{/list}"*>List Users</a>

**5.3.2: Controller:**

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

**public** String getall(@ModelAttribute UserModel userPojo,Map<String, Object> map, Model model){

List<UserModel> userPojoList=userService.getAll();

map.put(Constants.***CONST\_USER\_LIST***,userPojoList );

model.addAttribute(Constants.***CONST\_USER\_COUNT***,userPojoList.size());

**return** Constants.***VIEW\_ADMINHOME***;

}

**5.3.3: Service:**

@Transactional

**public** List<UserModel> getAll() {

List<User> userList= **new** ArrayList<User>();

List<UserModel> userPojoList= **new** ArrayList<UserModel>();

userList= userDao.getAll();

**for**(User user:userList){

UserModel userPojo= **new** UserModel();

userPojo.setContact(user.getContact());

userPojo.setEmail(user.getEmail());

userPojo.setId(user.getId());

userPojo.setUsername(user.getUsername());

String rolename=user.getRole().getRollname();

userPojo.setRolename(rolename);

userPojoList.add(userPojo);

}

**return** userPojoList;

}

**5.3.4: Data Access Object (DAO):**

**public** List<User> getAll() {

**return** userRepo.findAll();

}

**5.3.5: Repository:**

**public** **interface** UserRepo **extends** JpaRepository<User, Integer> {

}

**5.4: Excel data upload to database table:**

**5.4.1: View:**

<form method=*"post"* th:action=*"@{/uploaddata}"* action=*"#"* enctype=*"multipart/form-data"* th:object=*"${dataPojo}"*>

<input type=*"file"* name=*"file"* value=*"file"* required=*"required"*/><br/>

<select th:field=*"\*{type}"*>

<option value=*"All"* name=*"All"* >All</option>

<option value=*"Engineering"* name=*"Engineering"* >Engineering</option> <option value=*"Medical"* name=*"Medical"*>Medical</option>

<option value=*"Agricultural"* name=*"Agricultural"*>Agricultural</option>

</select>

<select th:field=*"\*{year}"*>

<option value=*"2014"* name=*"2014"* >2014</option>

<option value=*"2015"* name=*"2015"*>2015</option>

<option value=*"2016"* name=*"2016"*>2016</option>

<option value=*"2017"* name=*"2017"*>2017</option>

<option value=*"2018"* name=*"2018"*>2018</option>

<option value=*"2019"* name=*"2019"*>2019</option>

<option value=*"2020"* name=*"2020"*>2020</option>

</select>

<input type=*"Submit"* value=*"Upload"*/>

</form>

**5.4.2: Controller:**

@RequestMapping(value="/uploaddata",headers = "content-type=multipart/\*", method=RequestMethod.***POST***)

**public** String handleFormUpload(@RequestParam("file") MultipartFile file, @ModelAttribute DataModel dataPojo, Model model,Map<String, Object> map)**throws** IOException {

logger.log(Level.***INFO***,"Controller");

String status = **null**;

**try**{

**if** (!file.isEmpty()) {

String check=ValidateUpload.*validateOfficeData*(file);

**if**(check.equals(Constants.***CONST\_DATA\_VALID***)){

**try**{

status=dataServiceSave.saveContents(file, dataPojo);

}**catch**(InvalidException invalidException){

**return** Constants.***VIEW\_DATAMANAGE***;

}

model.addAttribute(Constants.***CONST\_DATA\_MESSAGE***, status);

map.put(Constants.***CONST\_DATA\_POJO***, dataPojo);

}

**else**{

model.addAttribute(Constants.***CONST\_DATA\_MESSAGE***, check);

}

}

}**catch**(MultipartException multipartException){

logger.log(Level.***SEVERE***,multipartException.getMessage(), multipartException);

}

**return** Constants.***VIEW\_DATAMANAGE***;

}

**5.4.3: Validation of Excel file:**

**public** **static** String validateOfficeData(MultipartFile file){

/\*

\* Check whether file is of excel type

\*/

**if**(!file.getContentType().equals(Constants.***CONST\_EXCEL\_SHEET\_TYPE***)){

**return** Constants.***CONST\_ERROR\_IN\_FILE\_TYPE***;

}

/\*

\* Check whether file size is less than 10 MB

\*/

**if**(file.getSize() > 10\*1024\*1024) {

**return** Constants.***CONST\_ERROR\_IN\_FILE\_SIZE***;

}

**return** Constants.***CONST\_DATA\_VALID***;

}

**5.4.4: Service:**

@Transactional

**public** String saveContents(MultipartFile file, DataModel dataPojo) **throws** InvalidException {

XSSFWorkbook offices;

**int** rowValue=1;

**try** {

offices = **new** XSSFWorkbook(file.getInputStream());

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

XSSFSheet worksheet=offices.getSheetAt(0);

Iterator<Row> rite= worksheet.rowIterator();

**if**(rite.hasNext()){

XSSFRow row=(XSSFRow)rite.next();

**if**(!ValidateUpload.*validateHeader*(row)){

**return** Constants.***CONST\_ERROR\_IN\_EXCEL\_FORMAT***;

}

}

Iterable<Data> dataList= **new** ArrayList<Data>();

**while**(rite.hasNext()){

XSSFRow dataRow=(XSSFRow)rite.next();

ArrayList<Integer> intList= **new** ArrayList<Integer>();

ArrayList<String> strList=**new** ArrayList<String>();

Iterator<Cell> cite = dataRow.cellIterator();

**while**(cite.hasNext()){

Cell cell = cite.next();

**switch**(cell.getCellType()){

**case** Cell.***CELL\_TYPE\_STRING***: strList.add(cell.getStringCellValue().toString()); **break**;

**case** Cell.***CELL\_TYPE\_NUMERIC***: intList.add((**int**) cell.getNumericCellValue()); **break**;

**case** Cell.***CELL\_TYPE\_BLANK***: **throw** **new** InvalidException(Constants.***CONST\_ERROR\_BLANK\_DATA\_CELLS***);

**default** : **throw** **new** InvalidException(Constants.***CONST\_ERROR\_IN\_EXCEL\_CELL***);

}

}

rowValue++;

Data data=**new** Data();

**if**((intList.size()<17)||(strList.size()<7)){

**return** Constants.***CONST\_ERROR\_INVALID\_DATA***+rowValue;

}

data.setCode(strList.get(0));

data.setInstitutename(strList.get(1));

data.setCoursecode(strList.get(2));

data.setRegion(strList.get(3));

data.setdistrict(strList.get(4));

data.setPlace(strList.get(5));

data.setType(strList.get(6));

data.setOcb(intList.get(0));

data.setOcg(intList.get(1));

data.setScb(intList.get(2));

data.setScg(intList.get(3));

data.setStb(intList.get(4));

data.setStg(intList.get(5));

data.setBcab(intList.get(6));

data.setBcag(intList.get(7));

data.setBcbb(intList.get(8));

data.setBcbg(intList.get(9));

data.setBccb(intList.get(10));

data.setBccg(intList.get(11));

data.setBcdb(intList.get(12));

data.setBcdg(intList.get(13));

data.setBceb(intList.get(14));

data.setBceg(intList.get(15));

data.setYear(dataPojo.getYear());

((ArrayList<Data>)dataList).add(data);

intList.clear();

strList.clear();

}

String type=dataPojo.getType();

**int** year = dataPojo.getYear();

**if**(dataPojo.getType().equalsIgnoreCase("All")){

dataDao.deleteAll(year);

}

**else**{

dataDao.delete(type, year);

}

dataDao.saveAll(dataList);

**return** Constants.***CONST\_SUCCESSFULL***;

}

**5.4.5: Data Access Object (DAO):**

**public** **void** saveAll(Iterable<Data> dataList) {

dataRepo.save(dataList);

}

**5.4.6: Repository:**

**public** **interface** DataRepo **extends** JpaRepository<Data, Long> {

}

**5.5: Download data from database table in the form of excel file:**

**5.5.1: View:**

<form method=*"post"* th:action=*"@{/downloadData}"* action=*"#"* enctype=*"multipart/form-data"* th:object=*"${dataPojo}"*>

<select th:field=*"\*{type}"*>

<option value=*"All"* name=*"All"* >All</option>

<option value=*"Engineering"* name=*"Engineering"* >Engineering</option> <option value=*"Medical"* name=*"Medical"*>Medical</option>

<option value=*"Agricultural"* name=*"Agricultural"*>Agricultural</option>

</select>

<select th:field=*"\*{year}"*>

<option value=*"2014"* name=*"2014"* >2014</option>

<option value=*"2015"* name=*"2015"*>2015</option>

<option value=*"2016"* name=*"2016"*>2016</option>

<option value=*"2017"* name=*"2017"*>2017</option>

<option value=*"2018"* name=*"2018"*>2018</option>

<option value=*"2019"* name=*"2019"*>2019</option>

<option value=*"2020"* name=*"2020"*>2020</option>

</select>

<input type=*"submit"* value=*"download"*/>

</form>

**5.5.2: Controller:**

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

**public** String generateExcel(HttpServletRequest request, HttpServletResponse response, @ModelAttribute DataModel dataPojo, Model model, Map<String, Object> map) **throws** Exception {

List<Data> list = dataServiceSave.download(dataPojo);

**if**(list.size()==0){

. . model.addAttribute(Constants.***CONST\_DATA\_MESSAGE***,Constants.***CONST\_DATA\_NOT\_AVAILABLE***);

map.put(Constants.***CONST\_DATA\_POJO***, dataPojo);

}**else**{

response.setHeader(Constants.***CONST\_EXCEL\_CONTENT***, Constants.***CONST\_EXCEL\_ATTACHEMNT***);

response.setHeader(Constants.***CONST\_EXCEL\_CONTENT\_TYPE***, Constants.***CONST\_EXCEL\_SHEET\_TYPE***);

XSSFWorkbook workbook=ExcelDownload.*create*(list);

OutputStream outputStream=response.getOutputStream();

workbook.write(outputStream);

outputStream.flush();

outputStream.close();

}

**return** Constants.***VIEW\_DOWNLOAD***;

}

**5.5.3: Service:**

**public** List<Data> download(DataModel dataModel) **throws** FileNotFoundException {

List<Data> dataList= **new** ArrayList<Data>();

**if**(dataModel.getType().equalsIgnoreCase("All")){

dataList= dataDao.retrieve(dataModel.getYear());

}**else**{

dataList =dataDao.retrieve(dataModel.getType(), dataModel.getYear());

}

**return** dataList;

}

**5.5.4: Data Access Objects (DAO):**

**public** java.util.List<Data> retrieve(String type, Integer year) {

**return** dataRepo.findByTypeAndYear(type, year);

}

@Override

**public** List<Data> retrieve(**int** year) {

**return** dataRepo.findByYear(year);

}

**5.5.5: Repository:**

**public** **interface** DataRepo **extends** JpaRepository<Data, Long> {

List<Data> findByTypeAndYear(String type, Integer year);

List<Data> findByYear(**int** year);

}

**5.5.6: Writing data to excel workbook:**

**public** **static** XSSFWorkbook create(List<Data> data1){

XSSFWorkbook workbook = **null**;

**try**{

workbook = **new** XSSFWorkbook();

XSSFSheet spreadsheet = workbook.createSheet(Constants.***CONST\_DOWNLOAD\_FILE\_NAME***);

**int** rowIndex=0,columnIndex=0;

XSSFRow row =spreadsheet.createRow(rowIndex);

**for**(ExcelDataHeaders dataHeaders:ExcelDataHeaders.*values*()){

XSSFCell cell=row.createCell(columnIndex);

cell.setCellValue(dataHeaders.toString());

columnIndex++;

}

**for**(Data data:data1){

row=spreadsheet.createRow(++rowIndex);

XSSFCell id1=row.createCell(0);

id1.setCellValue(data.getCode());

XSSFCell code1=row.createCell(1);

code1.setCellValue(data.getInstitutename());

XSSFCell name1=row.createCell(2);

name1.setCellValue(data.getCoursecode());

XSSFCell region1=row.createCell(3);

region1.setCellValue(data.getRegion());

XSSFCell district1=row.createCell(4);

district1.setCellValue(data.getDistrict());

XSSFCell place1=row.createCell(5);

place1.setCellValue(data.getPlace());

XSSFCell ocb1=row.createCell(6);

ocb1.setCellValue(data.getOcb());

XSSFCell ocg1=row.createCell(7);

ocg1.setCellValue(data.getOcg());

XSSFCell scb1=row.createCell(8);

scb1.setCellValue(data.getScb());

XSSFCell scg1=row.createCell(9);

scg1.setCellValue(data.getScg());

XSSFCell stb1=row.createCell(10);

stb1.setCellValue(data.getStb());

XSSFCell stg1=row.createCell(11);

stg1.setCellValue(data.getStg());

XSSFCell bcab1=row.createCell(12);

bcab1.setCellValue(data.getBcab());

XSSFCell bcag1=row.createCell(13);

bcag1.setCellValue(data.getBcag());

XSSFCell bcbb1=row.createCell(14);

bcbb1.setCellValue(data.getBcbb());

XSSFCell bcbg1=row.createCell(15);

bcbg1.setCellValue(data.getBcbg());

XSSFCell bccb1=row.createCell(16);

bccb1.setCellValue(data.getBccb());

XSSFCell bccg1=row.createCell(17);

bccg1.setCellValue(data.getBccg());

XSSFCell bcdb1=row.createCell(18);

bcdb1.setCellValue(data.getBcdb());

XSSFCell bcdg1=row.createCell(19);

bcdg1.setCellValue(data.getBcdg());

XSSFCell bceb1=row.createCell(20);

bceb1.setCellValue(data.getBceb());

XSSFCell bceg1=row.createCell(21);

bceg1.setCellValue(data.getBceg());

XSSFCell type1=row.createCell(22);

type1.setCellValue(data.getType());

XSSFCell year1=row.createCell(23);

year1.setCellValue(data.getYear());

}

}**catch**(Exception exception){

***logger***.log(Level.***SEVERE***,exception.getMessage(), exception);

}

**return** workbook;

}

**CHAPTER 6**

**TESTING**

**6.1 Software Testing**

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the features of a software item.

Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

**Verification**

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

**Validation**

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

**6.1.1. Basics of software testing**

There are two basics of software testing: black box testing and white box testing.

**Black box Testing**

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

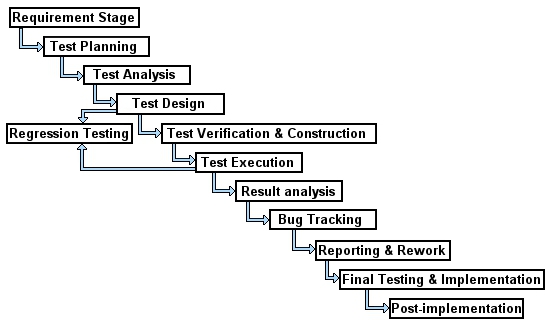
**White box Testing**

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

**6.2. Software Testing Life Cycle**

Software testing Life Cycle (STLC) defines the stages or steps or phases in testing of software. However, there is no fixed standard STLC in the world and it varies as per the Software development Life Cycle.

****

**Fig 7.1** Phases of Software Testing

**6.2.1 Requirement Stage**

This is the initial stage if the software testing process. In this phase, developer takes part in analysing the requirements for designing a product. The role of software tester is also necessary in this phase as they can think from user’s point of view, which developers may not. Thus a team of developers, testers and users can be formed to analyse the requirements.

**6.2.2 Test Planning**

Test planning means to determine a plan well in advance to reduce further risks. A well designed test plan document plays an important role in achieving a process oriented approach. Once the requirements of the project are confirmed, a test plan is documented.

**6.2.3 Test Analysis**

Once the test plan documentation is done, the next stage is to analyse what types of software testing should be carried out at the various stages of the Software Development Life Cycle (SDLC).

**6.2.4 Test Design**

Test design is done based on the requirements of the project documented in the SRS.

This phase decides whether manual or automated testing is to be done. In automation testing, different paths for testing are to be identified first and writing of scripts has to be done, if required. An end-to-end check list that covers all the features of the project is necessary in the test design process.

**6.2.5 Test Verification and Construction**

In this phase, the test plan, test design and automated test script are completed. Stress and performance testing plans are also completed at this stage. When the development team is done with a unit of code, the testing team is required to help them in testing that unit and reporting any bug in the product, if found. Integration testing and bug reporting is done in this phase of software testing.

**6.2.6 Test Execution**

Planning and execution of various test cases is done in this phase. Once the unit testing is completed, the functionality of the tests is done in this phase. At first, top level testing is done to find out the top level failures and bugs are reported immediately to the development team to get the required workaround. Test reports have to be documented properly and the bugs have to be reported to the development team.

**6.2.7. Result Analysis**

After the successful execution of the test case, the testing team has to re-test it to compare the expected values with the actual values and declare the results pass or fail.

**6.2.8 Bug Tracking**

This is one of the important stages, as the Defect Profile Document (DPD) has to be updated to let developer know about the defect. The Defect Profile Document (DPD) contains the following:

**Defect ID:** Each bug gets its own unique identification to track it in the future.

**Test case ID:** Test Case identification for a particular defect. It is useful to track progress.

**Description:** Detailed description of the bug. Tracking tools like Bugzilla or TTPro may be used to create unique descriptions for each bug.

**Summary:** This field contains some keyword information about the bug, which can help in minimizing the number of records to be searched.

**Defect Submitted By:** Name of the tester who detected/reported the bug.

**Date of Submission**: Date at which the bug was detected and reported.

**Build Number:** The build number is the number of test runs requires satisfying a testing criterion.

**Version Number**: The version information of the software application in which the bug was detected and fixed.

**Assigned To:** Name of the developer who is supposed to fix the bug.

**6.2.9. Reporting and Rework**

Testing is an iterative process. The bug that is reported and fixed by the development team has to undergo the testing process again to ensure that the problems have been solved.

Regression testing has to be done. Once the Quality Analyst assures that the product is ready, the software is released for production. Before release, the software has to undergo one more round of top level testing. Thus the testing is an on-going process.

**6.2.10 Final testing and Implementation**

This phase focuses on the remaining levels of testing, such as acceptance, load, stress, performance and recovery testing. The application needs to be verified under specified conditions with respect to the SRS. Various documents are updated and different matrices for testing are completed at this stage of software testing.

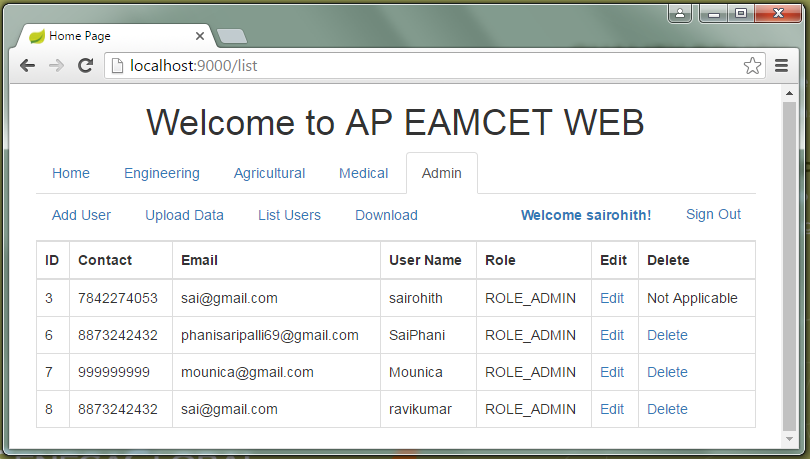
**6.2.11 Post Implementation**

Once the test results are evaluated, the recording of the errors that occurred during the various levels of testing process is done. Creating plans for improvement and enhancement is an on-going process. This helps to prevent similar problems from occurring in future projects.

In short, planning for improvement of testing process for future applications is done in this phase.

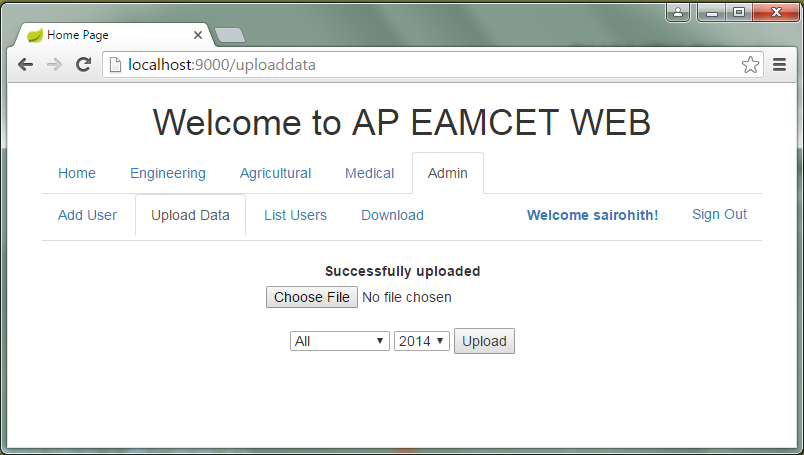
**6.3 Test cases:**

**7.3.1: Test case 1:** To Display the list of admins

****

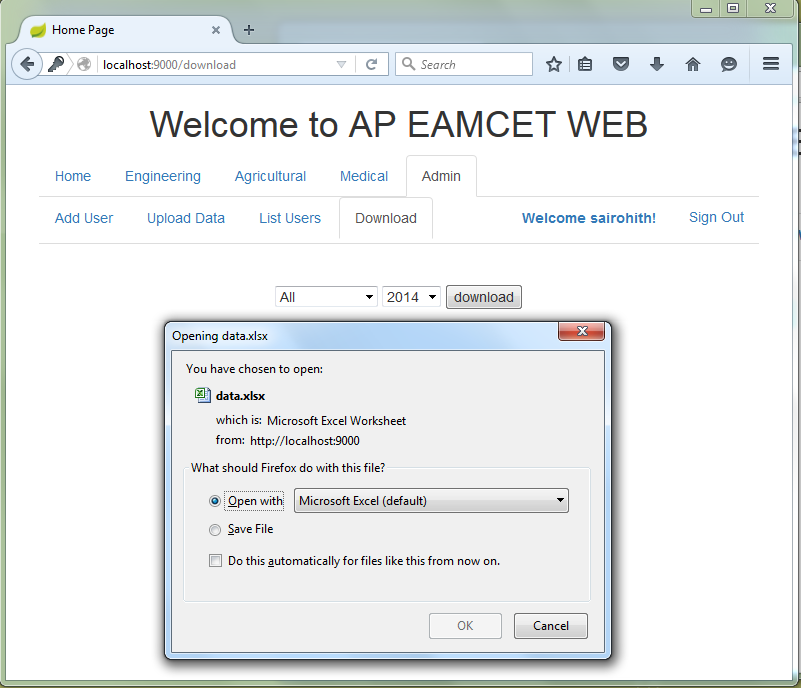
**Fig 6.2** Test case showing the list of existing admins

**6.3.2: Test case 2:** To upload data from excel to database table.

****

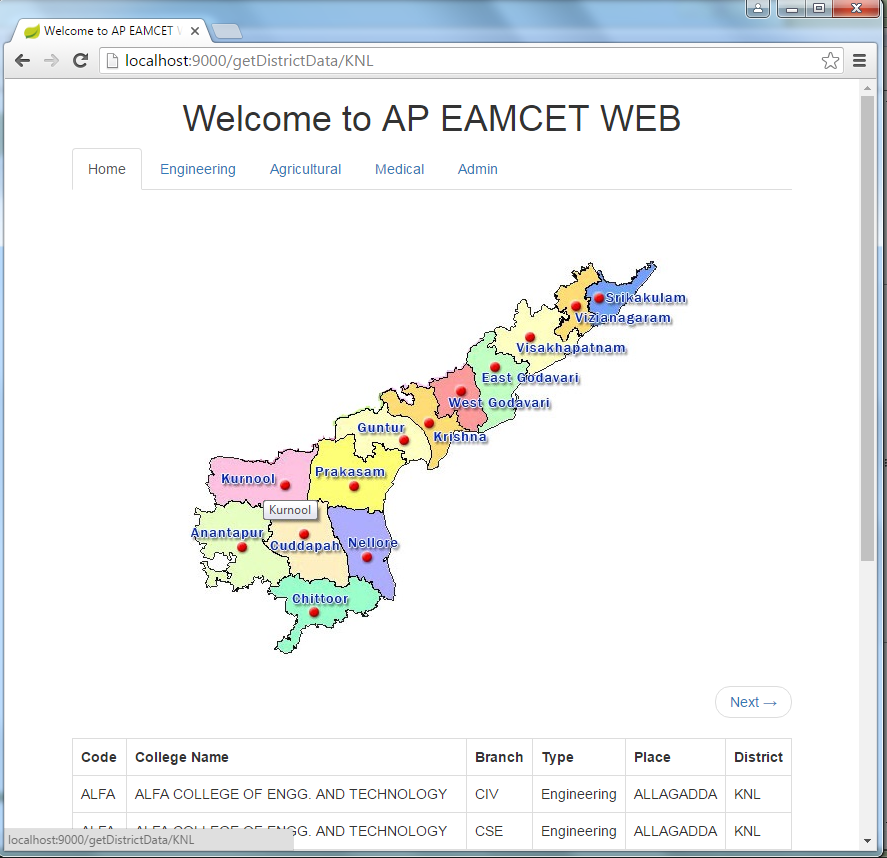
**Fig 6.3** Test case showing the successful insertion of data

**6.3.3: Test case 3:**  To download the data present in database table to an excel file.

****

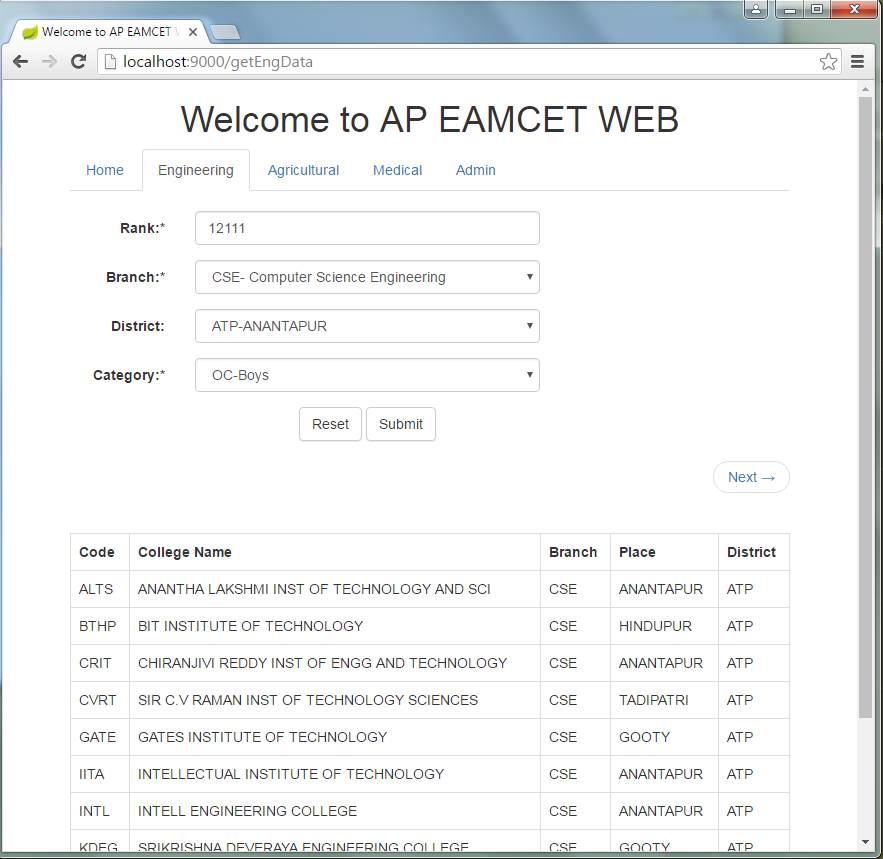
**Fig 6.4** Test case to download data present in database

**6.3.4: Test case 4:** To display district wise data, when clicked on a particular district.

****

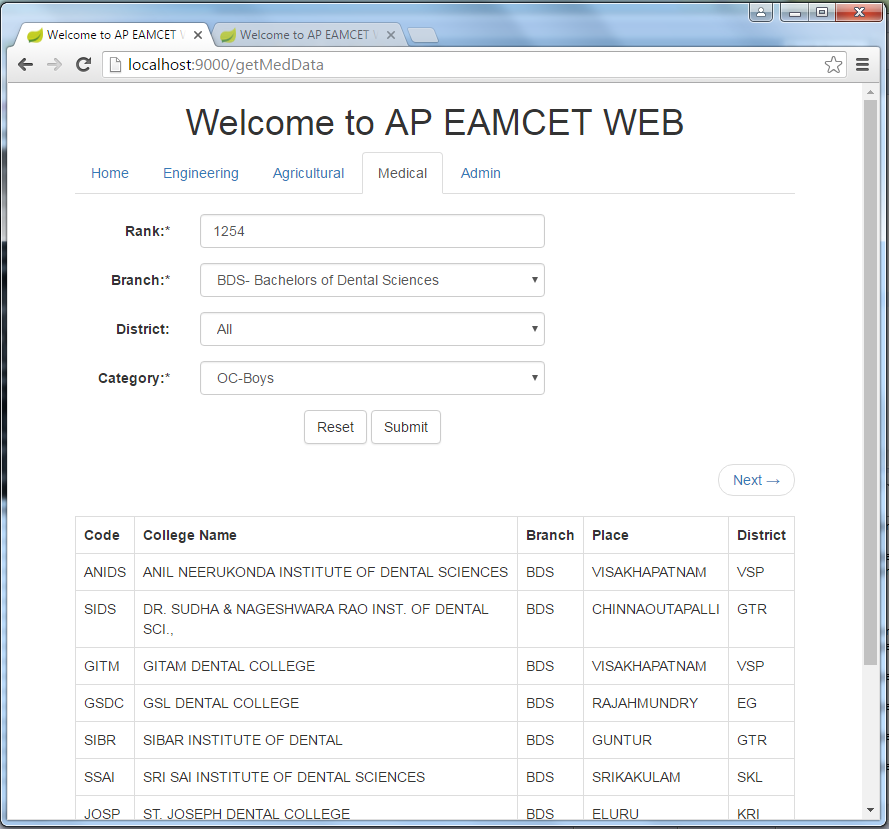
**Fig 6.5** Test case showing the data based on district

**6.3.5: Test case 5:** To display the engineering colleges list based on rank, category, branch and district.

****

**Fig. 6.6** Test case showing the data based on rank, category, branch and district.

**6.3.6: Test case 6:** To display the medical colleges list based on rank, category, branch and district.

****

**Fig. 6.7** Test case showing the data based on rank, category, branch and district.

**Bibilography:**