**Chapter 1**

**INTRODUCTION**

**1.1 Company profile**

**Company Name :** OMVSAB IT SOLUTION  
**Address:**

Hingane Home Colony,   
Near Karve nagar Bus Stop,   
Karve nagar,  
Pune 411052

**Telephone :** 020-65222250

**Website :** www.omvsabitsolution.in

**OMVSABIT Solution Pvt. Ltd.**

OMVSAB is Software & Web Development Company in pune with Hosting, Domain Registration, E- Marketing. OMVSAB is implementers of Business Software and all Web Applications. OMVSAB is a global IT services and solutions provider. The most important factor contributing to the success of OMVSAB is the mutual trust between the client, team and the management of the company. We have a well-equipped infrastructure and 24\*7 customer supports through email, phone or live chat. OMVSAB is offering ITES (IT Enabled Services) like Software solutions, website design and development, Web hosting, domain name registration. OMVSAB is a well-known software development company in pune. We help our clients to improve their business processes. We believe in fully satisfied clients with our technology and we will be fully transparent to maintain perfect relationships with clients.

**Mission:**

To build and retain a pool of talented professionals within OmVSab  
To provide opportunities for continuous professional and personal growth to the employees at OmVSab.  
To respect individual opinion while boosting team spirit across the organization  
To operate with a strong belief in an all-level employee participation for continuous knowledge and process up-gradation.

**Vision:**

To achieve sustainable quality and productivity we divide our vision into four parts.  
**Profit**: To create value and make difference.  
**Employee**: To work where people are motivated to be the best.  
**Partners**: Develop a winning network of clients.  
**Productivity**: Achieving productivity through efficiency and effectiveness in software products.

**Values:**

**Customer-centered**: We are fully committed to true customer satisfaction and success, which leads to long-lasting partnerships.  
**Collaboration**: Each of us has individual strengths and weaknesses. By acting together in synergy, we contribute as a whole to our common goal.  
**Excellence**: We continuously strive for excellence by raising the bar and delivering the very best to our customers.  
**Care**: We are a people oriented business. We genuinely care for each other.

**1.2 Existing System and Need for System :**

The college maintains the information, about the student details, student marks, exam details, department wise staff details etc., also keep the contacts with student and staff manually. Each and every activity that belongs to college is maintained by paperwork.

In the existing system the whole work is done manually. All the information is recorded in register and MS Excel application. All student entries and student results process is very time consuming and error prone. It needs lots of paper work. When any student related data needed it is required to search in the records. Several functions like generate the results for details were given manually.

In the present system, the authority is required to collect the information about the students through manually communication.

**Limitations of existing system:**

* Problem in maintaining the student’s record as the time of admission as various details are to rerecorded at master data.
* Problem in keeping the data of various departments together.
* In the existing system all the information is recorded into registers. Register entries and results process is very time consuming and error possibilities.
* Several functions like generate exam time table were given manually.
* Problem in keeping data of staff’s number of teacher’s increases.
* Report generation is very slow.

**Need for system:**

The “Exam Result ERP for Savitribai College of Arts Pimplegao Pisa is intended to maintain the information, about the students, new admissions, staff details, department details, students results details.

The main motive of new system is to maintain details of students and generate results easily.

**Followings are the drawback of existing system:**

* Manual system is Time-consuming.
* Manual system is more prone to errors as compared to computerized system.
* Manual system is not user friendly and not secured.
* Handling manual system is tedious job.
* File management is needed in manual system.
* In manual system, it is not possible to keep back-up for each and every transaction .It is created for study only.

**1.3 Scope of Work :**

Easy to maintain Data.

Easy to generate report.

User (Authorized Person/staff) feels Secure to maintain student data.

User (Authorized Person/staff) store new admission details in database.

User (Authorized Person/staff) updates admission details.

Student details save in centralize database.

Student and staff receive important information through message

User (Authorized Person/staff ) generates results more securely.

User (Authorized Person/staff ) generates time table for exam.

**1.4 Operating Environment – Hardware and Software :**

**HARDWARE REQUIREMENTS (MINIMUM):-**

* RAM Capacity : 128MB
* Pentium processor : 233 MHZ
* Hard Disk : 20GB
* CD-ROM Drive : 32 HZ
* Keyboard : 108 Standard
* Mouse : Optical
* Monitor : 15’’ Color Monitor

**SOFTWARE REQUIREMENTS:-**

* Web Presentation : HTML,CSS
* Client – side Scripting : JavaScript
* Programming Language : Java
* Web based Technologies : Servlets, jsp
* database Connectivity : JDBC
* Java Version : JDK1.6
* Backend Database : MySQL
* Operating System : Windows
* Web Server : Tomcat 7.0
* Browser : IE/Mozilla

**1.5 Detail Description of Technology Used:**

**Introduction: (Java)**

Java is a high-level, third generation programming language, like C, FORTRAN, Smalltalk, Perl, and many others. You can use Java to write computer applications that play games, store data or do any of the thousands of other things computer software can do. Compared to other programming languages, Java is most similar to C. However although Java shares much of C's syntax, it is not C. Knowing how to program in C or, better yet, C++, will certainly help you to learn Java more quickly, but you don't need to know C to learn Java. A Java compiler won't compile C code, and most large C programs need to be changed substantially before they can become Java programs. What's most special about Java in relation to other programming languages is that it lets you write special programs called applets that can be downloaded from the Internet and played safely within a web browser. Java language is called as an Object-Oriented Programming language and before begining for Java, we have to learn the concept of OOPs(Object-Oriented Programming).

**Basic Concept of OOP(Object-Oriented Programming):**

There are some basic concepts of object oriented programming as follows:

1. Object

2. Class

3. Data abstraction

4. Data encapsulation

5. Inheritance

6. Polymorphism

7. Dynamic binding

**1. Object**

Objects are important runtime entities in object oriented method. They may characterize a location, a bank account, and a table of data or any entry that the program must handle.

**2. Classes**

A class is a set of objects with similar properties (attributes), common behaviour (operations), and common link to other objects. The complete set of data and code of an object can be made a user defined data type with the help of class.

The objects are variable of type class. A class is a collection of objects of similar type. Classes are user defined data types and work like the build in type of the programming language. Once the class has been defined, we can make any number of objects belonging to that class. Each object is related with the data of type class with which they are formed.

**3. Data Abstraction**

Data abstraction refers to the act of representing important description without including the background details or explanations.

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, cost and functions operate on these attributes. They summarize all the important properties of the objects that are to be created.

Classes use the concepts of data abstraction and it is called as Abstract Data Type (ADT).

**4. Data Encapsulation**

Data Encapsulation means wrapping of data and functions into a single unit (i.e. class). It is most useful feature of class. The data is not easy to get to the outside world and only those functions which are enclosed in the class can access it.

These functions provide the boundary between Object‘s data and program. This insulation of data from direct access by the program is called as **Data hiding**.

**5**. **Inheritance**

Inheritance is the process by which objects of one class can get the properties of objects of another class. Inheritance means one class of objects inherits the data and behaviours from another class. Inheritance maintains the hierarchical classification in which a class inherits from its parents.

Inheritance provides the important feature of OOP that is reusability. That means we can include additional characteristics to an existing class without modification. This is possible deriving a new class from existing one.

In other words, it is property of object-oriented systems that allow objects to be built from other objects. Inheritance allows openly taking help of the commonality of objects when constructing new classes. Inheritance is a relationship between classes where one class is the parent class of another (derived) class. The derived class holds the properties and behaviour of base class in addition to the properties and behaviour of derived class.

**6. Polymorphism**

(Poly means ―many‖ and morph means ―form‖). Polymorphism means the ability to take more than one form. Polymorphism plays a main role in allocate objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific activities associated with each operation may differ. Polymorphism is broadly used in implementing inheritance.

It means objects that can take on or assume many different forms. Polymorphism means that the same operations may behave differently on different classes. Booch defines polymorphism as the relationship of objects many different classes by some common super class. Polymorphism allows us to write generic, reusable code more easily, because we can specify general instructions and delegate the implementation detail to the objects involved.

**7. Dynamic Binding**

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code related with a given procedure call is not known until the time of the call at run time.

Dynamic binding is associated polymorphism and inheritance.

**Java(Hibernate)**

Hibernate ORM (Hibernate in short) is an object-relational mapping library for the [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance

mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate is a free software that is distributed under the GNU Lesser General Public License.

Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to [SQL](http://en.wikipedia.org/wiki/SQL) data types). Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from manual result set handling and object conversion. Applications using Hibernate are portable to supported SQL databases with little performance overhead

**Mapping**

Mapping Java classes to database tables is accomplished through the configuration of an XML file or by using Java Annotations. When using an XML file, Hibernate can generate skeleton source code for the persistence classes. This is unnecessary when annotations are used. Hibernate can use the XML file or the annotations to maintain the database schema.

Facilities to arrange [one-to-many](http://en.wikipedia.org/wiki/Point-to-multipoint_communication) and [many-to-many](http://en.wikipedia.org/wiki/Many-to-many_(data_model)) relationships between classes are provided. In addition to managing associations between objects, Hibernate can also manage[reflexive](http://en.wikipedia.org/wiki/Reflexive_relation) associations where an object has a one-to-many relationship with other instances of its own [type](http://en.wikipedia.org/wiki/Data_type).

Hibernate supports the mapping of custom value types. This makes the following scenarios possible:

* Overriding the default SQL type that Hibernate chooses when mapping a column to a property.
* Mapping Java [Enum](http://en.wikipedia.org/wiki/Enumerated_type" \o "Enumerated type) to columns as if they were regular properties.
* Mapping a single property to multiple columns.

Definition: Objects in a front-end application follow [OOP](http://en.wikipedia.org/wiki/Object-oriented_programming) principles, while objects in the back-end follow [database normalization](http://en.wikipedia.org/wiki/Database_normalization) principles, resulting in different representation requirements. This problem is called "[object-relational impedance mismatch](http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch)". Mapping is a way of resolving the impedance mismatch problem.

Mapping tells the ORM tool which java class object to store in which database table

**Hibernate Query language(HQL)**

Hibernate provides an [SQL](http://en.wikipedia.org/wiki/SQL) inspired language called [Hibernate Query Language](http://en.wikipedia.org/wiki/Hibernate_Query_Language) (HQL) which allows SQL-like queries to be written against Hibernate's data objects. *Criteria Queries*are provided as an [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) alternative to HQL. Criteria Query is used to modify the objects and provide the restriction for the objects.

**Persistence**

Hibernate provides transparent persistence for [Plain Old Java Objects](http://en.wikipedia.org/wiki/Plain_Old_Java_Object) (POJOs). The only strict requirement for a persistent class is a [no-argument constructor](http://en.wikipedia.org/wiki/Nullary_constructor), not necessarily*public*. Proper behavior in some applications also requires special attention to the *equals()* and *hashCode()* methods.[[2]](http://en.wikipedia.org/wiki/Hibernate_%28Java%29#cite_note-2)

Collections of data objects are typically stored in Java collection objects such as Set and List. [Java generics](http://en.wikipedia.org/wiki/Generics_in_Java), introduced in Java 5, are supported. Hibernate can be configured to [lazy load](http://en.wikipedia.org/wiki/Lazy_load) associated collections. Lazy loading is the default as of Hibernate 3.

Related objects can be configured to [*cascade*](http://en.wikipedia.org/wiki/Propagation_constraint) operations from one to the other. For example, a parent Album object can be configured to cascade its save and/or delete operation to its child Track objects. This can reduce development time and ensure [referential integrity](http://en.wikipedia.org/wiki/Referential_integrity). A *dirty checking* feature avoids unnecessary database write actions by performing SQL updates only on the modified fields of persistent objects.

**Integration**

Hibernate can be used both in standalone [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) applications and in [Java EE](http://en.wikipedia.org/wiki/Java_EE) applications using [servlets](http://en.wikipedia.org/wiki/Java_Servlet), [EJB](http://en.wikipedia.org/wiki/EJB) session beans, and [JBI](http://en.wikipedia.org/wiki/JBI) service components. It can also be included as a feature in other programming languages. For example, [Adobe](http://en.wikipedia.org/wiki/Adobe_Systems) integrated Hibernate into version 9 of [ColdFusion](http://en.wikipedia.org/wiki/ColdFusion) (which runs on J2EE app servers) with an abstraction layer of new functions and syntax added into [CFML](http://en.wikipedia.org/wiki/CFML)

**Entities And Components**

In Hibernate [jargon](http://en.wikipedia.org/wiki/Jargon), an *entity* is a stand-alone object in Hibernate's [persistent](http://en.wikipedia.org/wiki/Persistence_(computer_science)) mechanism which can be manipulated independently of other objects. In contrast, a *component* is subordinate to an entity and can be manipulated only with respect to that entity. For example, an Album object may represent an entity but the Tracks object associated with the Album objects would represent a *component* of the Album entity if it is assumed that Tracks can only be saved or retrieved from the database through the Album object. Unlike J2EE, it can switch databases.

**MySql**

For a more general introduction see the SQL Wikibook.

**S**tructured **Q**uery **L**anguage is a third generation language for working with relational databases. Being a 3G language it is closer to human language than machine language and therefore easier to understand and work with.

* Dr. E. F. Ted Codd who worked for IBM described a relational model for database in 1970.
* In 1992, ANSI (American National Standards Institute), the apex body, standardized most of the basic syntax.
* Its called SQL 92 and most databases (like Oracle, MySQL, Sybase, etc.) implement a subset of the standard (and proprietary extensions that makes them often incompatible).

**Why MySql**

* Free as in Freedom - Released with GPL version 2 license (though a different license can be bought from Oracle, see below)
* Cost - Free!
* Support - Online tutorials, forums, mailing list (lists.mysql.com), paid support contracts.
* Speed - One of the fastest databases available.
* Functionality - supports most of ANSI SQL commands.
* Ease of use - less need of training / retraining.
* Portability - easily import / export from Excel and other databases
* Scalable - Useful for both small as well as large databases containing billions of records and terabytes of data in hundreds of thousands of tables.
* Permission Control - selectively grant or revoke permissions to users.

**Java Script**

JavaScript is a Client-side scripting language, which empowers the developers to write functionality to run at the users client .It is very useful and powerful if the developers want to preprocess data on the client before submission to the server, this can reduce traffic over the network and offload processing from the server to the client. Since JavaScript is a Scripting language (not compiled) and runs primarily on the client. Most JavaScript used today is used to manipulate different aspects of the browsers. JavaScript uses an object-oriented programming (OOP) approach.

There are four primary components to JavaScript’s OOP methodology:

**1.** **Object**: A JavaScript object can be almost any elements of a web documents such as the documents, tables, forms, buttons, images.

**2.** **Properties:** Each object has its own properties such as the history of the browser.

History length contains the count of the items in the history list.

**3.** **Methods:** Objects can do certain things known as methods. Basically a method

Is a means of a making an object do something like open or close.

**4. Events:** An events an actually triggers the function or method to run.

JavaScript is good to the do the following:

* User input Validation.
* Calculations and simple graphing.
* Mouse over buttons.
* Counters.
* Quick reaction to user actions..

**CHAPTER 2**

**PROPOSED SYSTEM**

**2.1 Proposed System :**

The proposed system is a web based application. The system will be maintained by Admin .

This system is providing all features regarding college administration like –i.e.

* Save student details.
* New admission.
* Update admission.
* Cancel admission.
* Save exam time table.
* Generate seat number.
* Generate admit card.
* Generate results.
* Generate results summary.
* Add/Update/Delete Department details.
* Add/Update/Delete staff details.
* Generate Reports.

**2.1.1 Feasibility Study:**

A feasibility study is conducted to select the best system that meets performance requirement. The system required performance is defined by a statement of constraints, the identification specific system objective and a description of outputs.

**The Key Considerations in feasibility Analysis:-**

2.1.1.A. Economic feasibility

2.1.1.B Technical feasibility

2.1.1.C Operational feasibility

**2.1.1.A Economic feasibility:-**

The proposed system is economically feasible because the cost involve in purchasing the hardware and software are within approachable personal cost like salaries of employee hired are also nominal. The operating environment costs are marginal. The less time involve also helped in its economic feasible.

**2.1.1.B Technical feasibility:-**

It is possible with current technical support and hardware development in the organization to fulfill defines objectives and proposed activities. It is always desirable to select the best package that will be easy to implement for the system design and must be a flexible as possible with other programmers and outside developer as interactive as possible to use. The technical requirement regarding hardware and software should not be very high.

**2.1.1.C Operational feasibility:**

This system is very user friendly and very easy to use. The application does not have complex menus and not having complex operations. All the work is going very smoothly, all the facilities are having very good user interface so that’s why there is no any difficulty while doing work on the system.

**2.2 Objectives of System:**

The “**Exam Result ERP for Savitribai College of Arts, Pimplegaon Pisa”** provides rapid access to student’s details, new admission details, staff details, result processing result summery etc. System should be able to fulfill on its successful compilation and implementation.

**Objective of the system includes**:

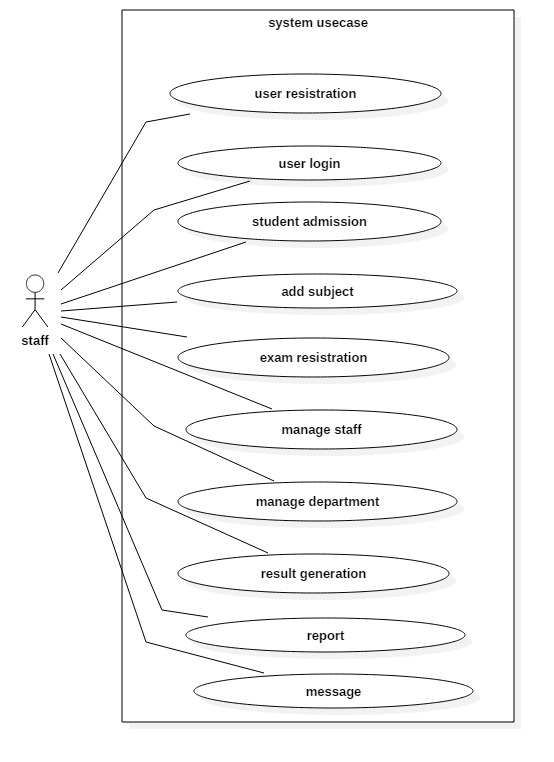
* Faster storage and retrieval of the information saving valuable time.
* No need to extra employee and extra work.
* To avoid redundancy in data storage through normalization.
* To generate accurate anew formatted reports.
* To provide accurate and timely reports.

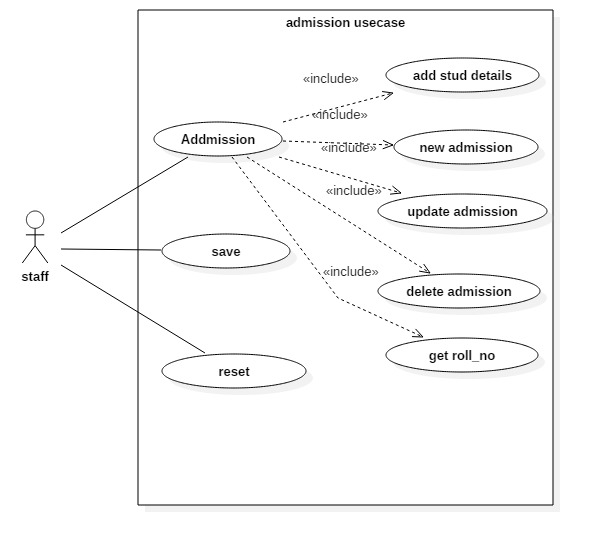
**2.3 User Requirements**

* To save time and cost in maintaining the all records of college.
* Faster access of Data.
* Increasing efficiency.
* Quick generate reports.
* Easy interface.
* To provide centralized database support for the user to access project related information.
* Performance of the system should be display the requested windows in very short time.
* Processing should be quicker.
* System should be storage the large number of data and records.

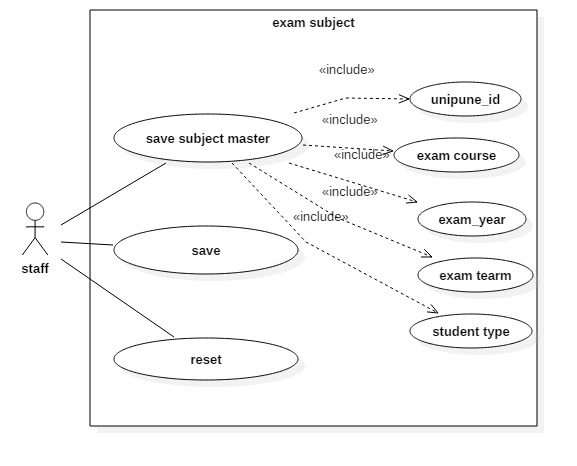
**CHAPTER 3**

**Analysis and Design**

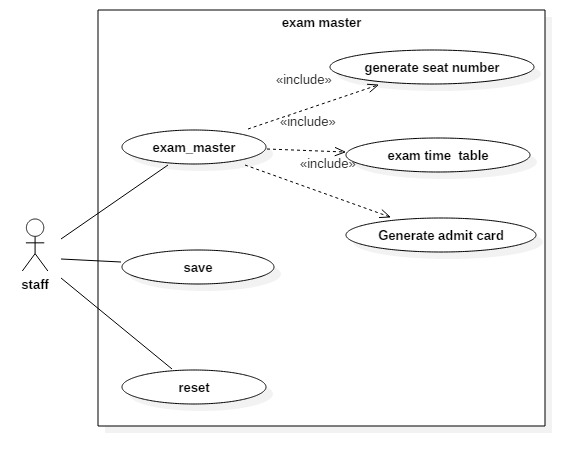
**3.2 Usecase Diagram 1.System Usecase2.Admission Usecase**

****

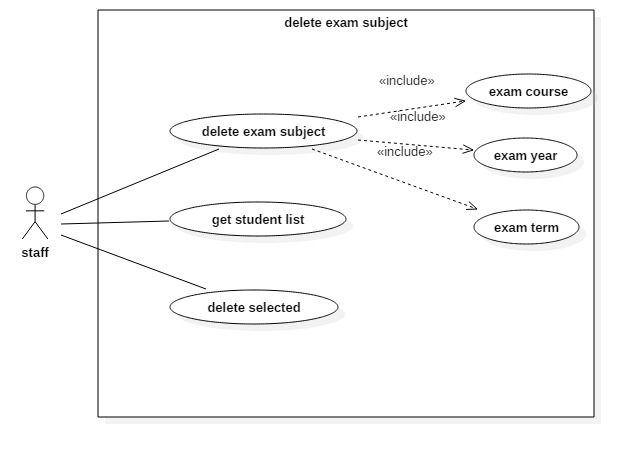
**3.Subject Master**

****

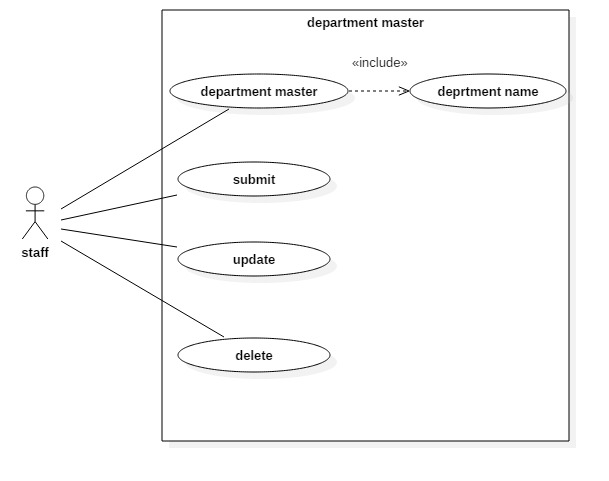
**4.Exam master**

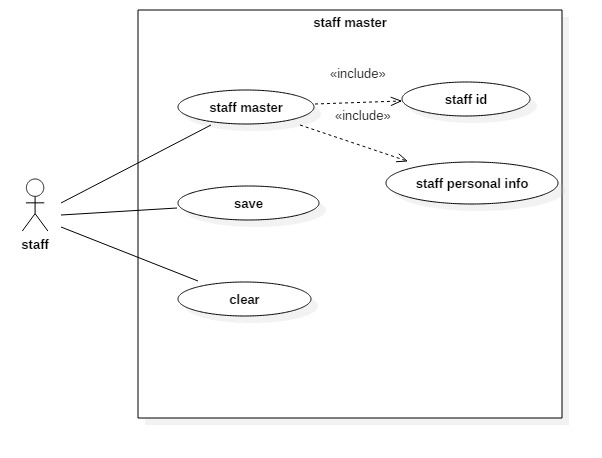
****

**5.Delete exam subject**

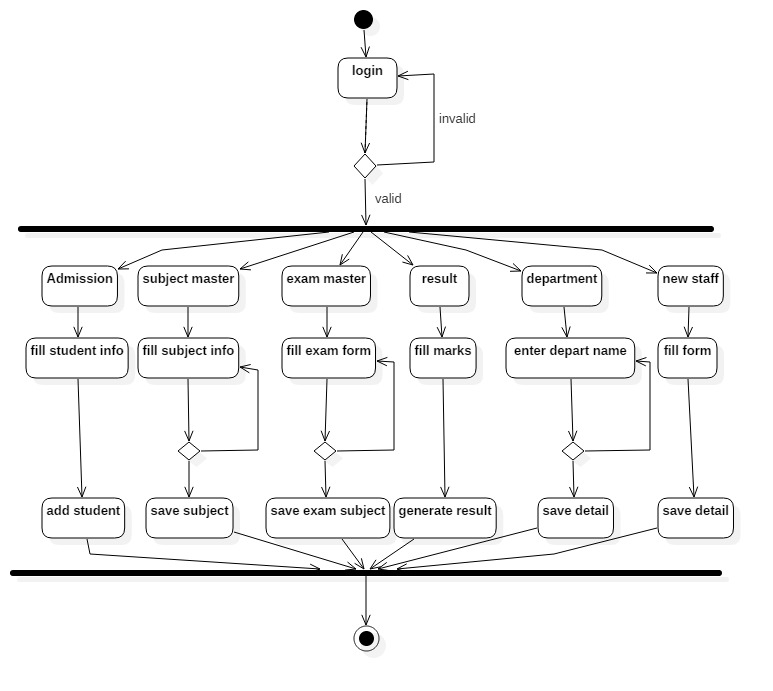
****

**6.Department Master**

****

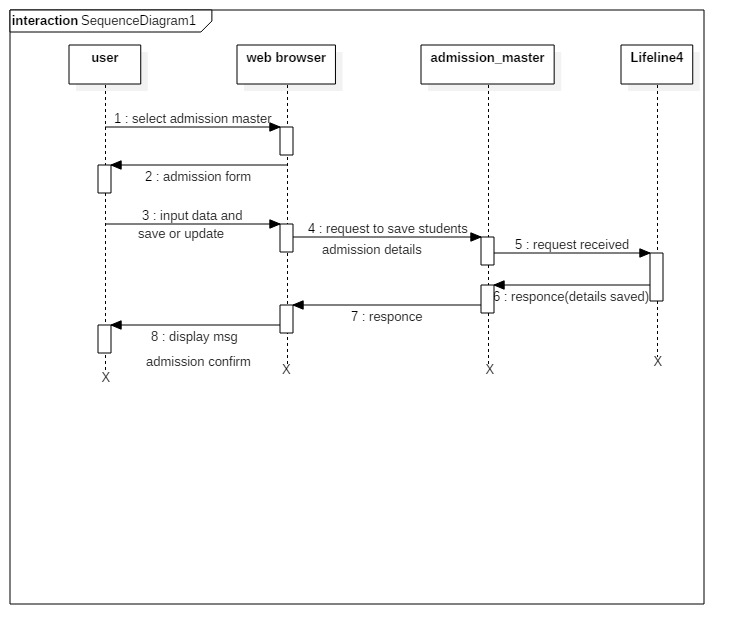
**7.Staff master**

**3.3 Activity diagram**

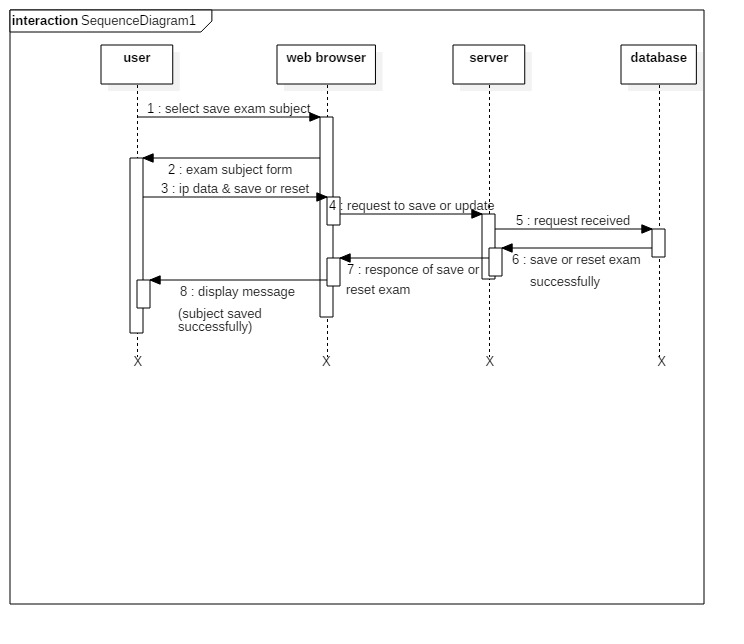
****

**3.4 Sequence diagram**

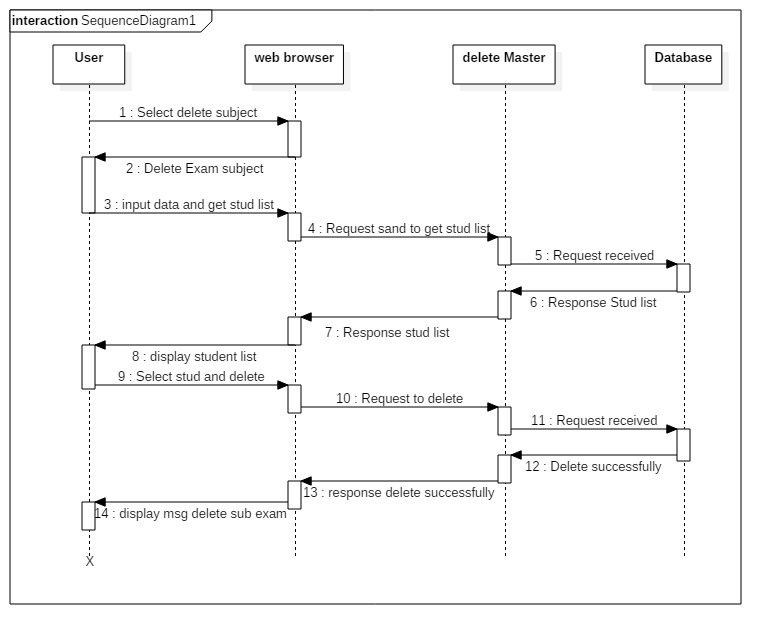
**1.Admission master**

****

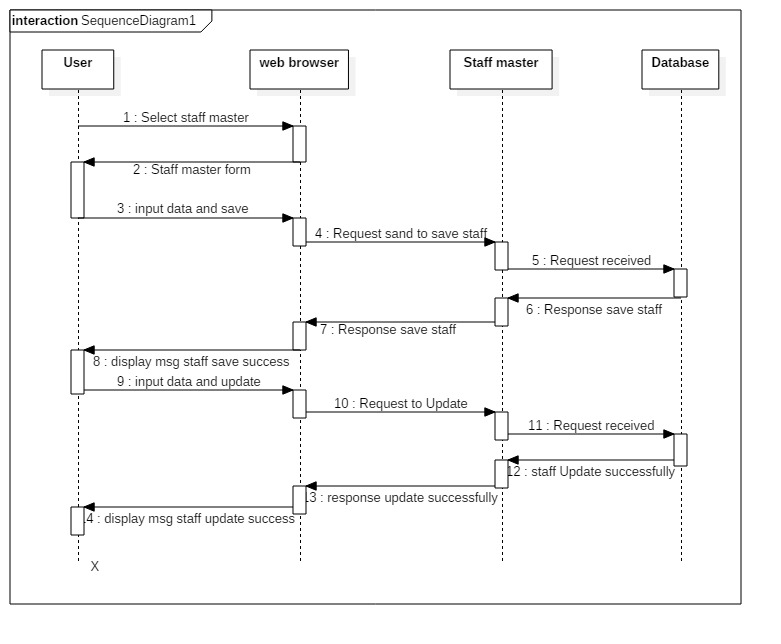
**2.Exam subject**

****

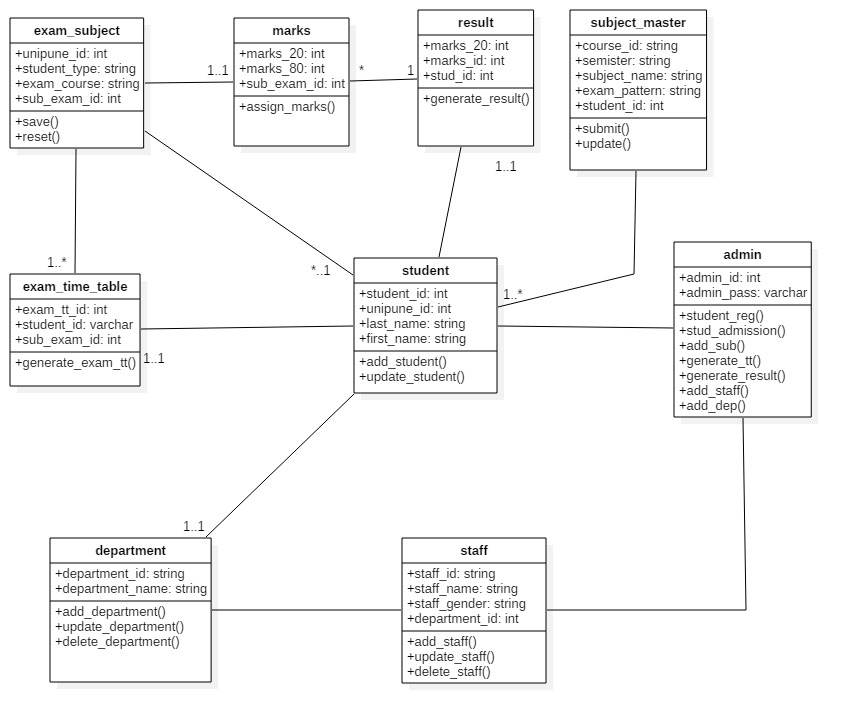
**3.Delete exam subject**

****

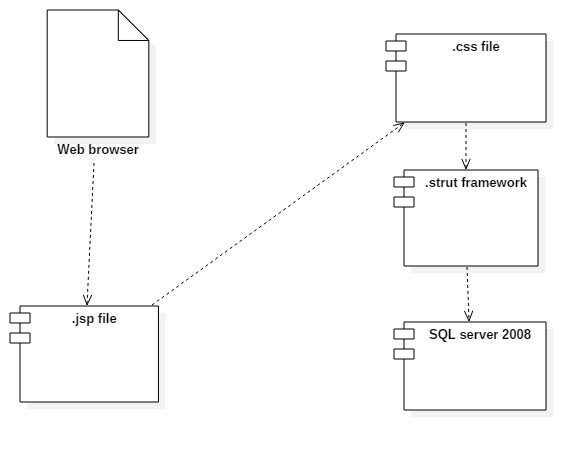
**4.Staff master**

****

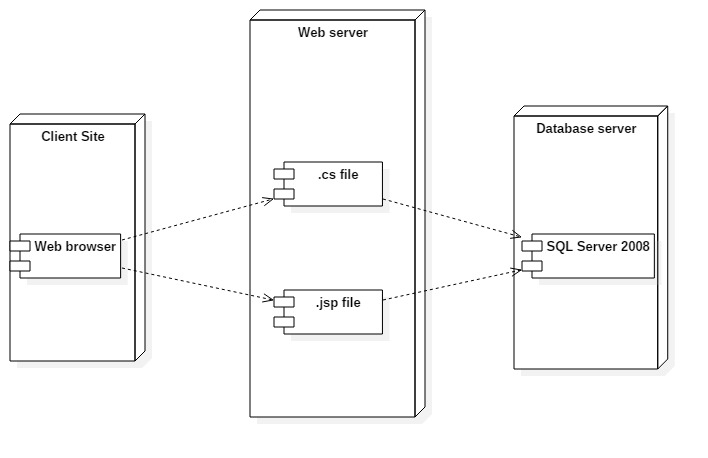
**3.5 CLASS DIAGRAM**

****

**3.7 Component Diagram**

****

**Deployment Diagram**

****

**Database Tables**

**1.Table name:- Admin**

Description:- Login information

Primary Key:- login\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | login\_id | Varchar | 10 | Primary Key | Store the login id |
| 2 | login\_pwd | Varchar | 20 | Not Null | Store the login password |

**2. Table name:- Admission**

Description:- Store the information about student academic year

Primary Key:- admission\_id

Foreign Key: student\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | admission\_id | Int | 10 | Primary Key | Store the Addmission id of students. |
| 2 | course\_name | Varchar | 20 | Not Null | Store the Course name of students. |
| 3 | course\_year | Varchar | 5 | Not Null | Store the course year of students. |
| 4 | course\_term | Varchar | 10 | Not Null | Store the course year of students. |
| 5 | student\_id | Varchar | 5 | Foreign Key | Store the student id of students. |
| 6 | roll\_no | Varchar | 5 | Not Null | Store roll number |

**3. Table name:- Department**

Description:- Store information about department

Primary Key:- department\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | department\_id | Int | 10 | Primary Key | Store the Department id of college |
| 2 | department\_name | Varchar | 10 | Not Null | Store the department name of the college. |

**4. Table name:- Exam\_time\_table**

Description:- Exam related data like time table, appeared students

Primary Key:-exam\_time\_table\_id

Foreign Key: student\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | exam\_time\_table\_id | Int | 10 | Primary Key | Store the Addmission id of students. |
| 2 | course\_name | Varchar | 20 | Not Null | Store the Course name of students. |
| 3 | course\_year | Varchar | 10 | Not Null | Store the course year of students. |
| 4 | course\_term | Varchar | 10 | Not Null | Store the course year of students. |
| 5 | student\_id | Varchar | 5 | Foreign key | Store the student id of students |
| 6 | exam\_date | Varchar | 10 | Not Null | Store the exam date. |
| 7 | from\_time | Varchar | 10 | Not Null | Store the exam start time. |
| 8 | to\_time | Varchar | 10 | Not Null | Store the exam finish time . |
| 9 | order\_exam\_date | Varchar | 10 | Not Null | Store the order exam date . |

**5. Table name:- Marks \_20**

Description:-internal marks of students out of 20 marks.

Primary key:-marks\_20\_id

Foreign Key: subject\_exam\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | marks\_20\_id | Int | 10 | Primary Key | Store the mark id out of 20 marks |
| 2 | subject\_exam\_id | Int | 10 | Foreign Key | Store the subject exam id |
| 3 | out\_of\_20 | Int | 2 | Not Null | Store the out of 20 marks |
| 4 | is\_absent | Varchar | 7 | Not Null | It stores the student absentee if student absent for exam .by default its value is present |

**6. Table name:- Marks \_80**

Description:-internal marks of students out of 80 marks.

Primary Key:- marks\_80\_id

Foreign Key: subject\_exam\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | marks\_80\_id | Int | 10 | Primary Key | Store the mark id out of 80 marks |
| 2 | subject\_exam\_id | Int | 10 | Foreign Key | Store the subject exam id |
| 3 | out\_of\_80 | Int | 2 | Not Null | Store the out of 20 marks |
| 4 | is\_absent | Varchar | 7 | Not Null | It stores the student absenty if student absent for exam .by default its value is present |

**7.Table name:- Marks**

Description:-internal marks of students out of 80 marks.

Primary Key:- marks\_id

Foreign Key: subject\_exam\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | marks\_id | Int | 10 | Primary Key | Store the mark id |
| 2 | subject\_exam\_id | Int | 10 | Foreign Key | Store the subject exam id |
| 3 | out\_of\_100 | Int | 3 | Not Null | Store the out of 20 + 80 marks |
| 4 | is\_absent | Varchar | 7 | Not Null | It stores the student absenty if student absent for exam .by default its value is present |
| 5 | Marks\_20\_id | Int | 2 | F \_key | Marks out of 20 |
| 6 | Marks\_80\_id | Int | 3 | F\_key | Marks out of 80 |

**8. Table name:-Seat\_no**

Description: List of seat numbers of students

Primary Key:- seat\_no\_id

Foreign Key: subject\_exam\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | seat\_no\_id | Int | 10 | Primary Key | Store the seat no id of student |
| 2 | seat\_no | Varchar | 7 | Not Null | Store the seat no of student |
| 3 | subject\_exam\_id | Int | 10 | Foreign Key | Store the student subject exam id |
| 4 | order\_pre | Varchar | 10 | Not Null | Ascending order |
| 5 | order\_post | Varchar | 10 | Not Null | Deseeding order |

**9. Table name:-Student**

Description :-Student information during admission

Primary id:- student\_id

Foreign Key: unipuune\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | student\_id | Int | 10 | Primary Key | Store the student id |
| 2 | unipune\_id | Varchar | 10 | Not Null | Store the uni pune id |
| 3 | last\_name | Varchar | 20 | Not Null | Store the last name of student |
| 4 | first\_name | Varchar | 20 | Not Null | Store the first name of student |
| 5 | middle\_name | Varchar | 20 | Not Null | Store the middle name of student |
| 6 | father\_name | Varchar | 20 | Not Null | Store the fathers name of student |
| 7 | mother\_name | Varchar | 20 | Not Null | Store the mothers name of student |
| 8 | date\_birth | Date | 10 | Not Null | Store the birth date of student |
| 9 | Gender | Varchar | 6 | Not Null | Store the gender of student |
| 10 | Contact\_no | Varchar | 10 | Not Null | Store the contact no of student |
| 11 | Category | Varchar | 10 | Not Null | Store the categories of student |
| 12 | At\_post | Varchar | 10 | Not Null | Store the address of student |
| 13 | Taluka | Varchar | 10 | Not Null | Store the taluka of student |
| 14 | Destrict | Varchar | 10 | Not Null | Store the destrict of student |
| 14 | Pin\_code | Varchar | 6 | Not Null | Store the pin code of student |
| 15 | Dept\_id | Int | 6 | F\_key | Store dept id |

**10.** **Table name:-Staff**

Description:-Store the information about the staff working collage

Primary Key:- staff\_id

Foreign Key: department\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | staff\_id | Int | 10 | Primary Key | Store the staff id |
| 2 | last\_name | Varchar | 10 | Not Null | Store the last name of staff |
| 3 | staff\_fname | Varchar | 10 | Not Null | Store the first name of staff |
| 4 | Staff\_bdate | Date | 10 | Not Null | Store the birth date of staff |
| 5 | Gender | Varchar | 6 | Not Null | Store the gender of staff |
| 6 | Contact\_no | Varchar | 10 | Not Null | Store the contact no of staff |
| 7 | Staff\_date\_join | Date | 3 | Not Null | Store the joining date |
| 8 | Staff\_date\_retire | Date | 3 | Not Null | Store the retire date |
| 9 | Department\_id | Varchar | 5 | Foreign key | Store the department id of staff |

**11. Table name:- subject**

Description:-Information about subject with according to group

Primary Key:- subject\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | subject\_id | Int | 10 | Primary Key | Store the subject id |
| 2 | subject\_code | Varchar | 10 | Not Null | Store the subject code |
| 3 | subject\_name | Varchar | 10 | Not Null | Store the subject name |
| 4 | group\_name | Varchar | 10 | Not Null | Store the group name |
| 5 | subject\_type | Varchar | 15 | Not Null | Store the subject type |
| 6 | course\_name | Varchar | 20 | Not Null | Store the course name |
| 7 | course\_year | Varchar | 10 | Not Null | Store the course year |

**12.Table name:- subject\_exam**

Description:- Exam subject list with according to student

Primary Key:- subject\_exam\_id

Foreign Key: student\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| 1 | subject\_exam\_id | Int | 10 | Primary Key | Store the subject exam id |
| 2 | course\_name | Varchar | 20 | Not Null | Store the course name |
| 3 | Course\_year | Varchar | 10 | Not Null | Store the course year |
| 4 | Course\_term | Varchar | 7 | Not Null | Store the course term |
| 5 | Student\_type | Varchar | 5 | Not Null | Store the subject type |
| 6 | Student\_id | Int | 10 | Foreign key | Store the student id |

**3.8 Data dictionary :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr No** | **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** | **Table\_name** |
| 1 | admission\_id | Int | 10 | Primary Key | Store the Addmission id of students. | Admission |
| 2 | at\_post | Varchar | 10 | Not Null | Store the address of student | Student |
| 3 | Category | Varchar | 7 | Not Null | Store the categories | Student |
| 4 | Contact\_no | Varchar | 10 | Not Null | Store the contact no of student | Student |
| 5 | Contact\_no | Varchar | 10 | Not Null | Store the contact no of staff | Student |
| 6 | course\_name | Varchar | 10 | Not Null | Store the Course name of students. | Admission |
| 7 | course\_name | Varchar | 20 | Not Null | Store the Course name of students. | Exam\_time\_table |
| 8 | course\_name | Varchar | 20 | Not Null | Store the course name | Subject |
| 9 | course\_name | Varchar | 20 | Not Null | Store the course name | Subject\_exam |
| 10 | course\_term | Varchar | 10 | Not Null | Store the course year of students. | Exam\_time\_table |
| 11 | course\_term | Varchar | 10 | Not Null | Store the course year of students. | Admission |
| 12 | Course\_term | Varchar | 10 | Not Null | Store the course term | Exam\_time\_table |
| 13 | course\_year | Varchar | 10 | Not Null | Store the course year of students. | Admission |
| 14 | course\_year | Varchar | 10 | Not Null | Store the course year of students. | Exam\_time\_table |
| 15 | course\_year | Varchar | 10 | Not Null | Store the course year | Subject |
| 16 | course\_year | Varchar | 10 | Not Null | Store the course year | Subject\_exam |
| 17 | date\_birth | Date | 10 | Not Null | Store the birth date of student | Student |
| 18 | department\_id | Int | 5 | Primary Key | Store the Department id of college | department |
| 19 | department\_id | Varchar | 5 | Foreign key | Store the department id of staff | Staff |
| 20 | department\_name | Varchar | 10 | Not Null | Store the department name of the college. | Department |
| 21 | destrict | Varchar | 10 | Not Null | Store the destrict of student | Student |
| 22 | exam\_date | Date | 10 | Not Null | Store the exam date. | Exam\_time\_table |
| 23 | exam\_time\_table\_id | Int | 5 | Primary Key | Store the Addmission id of students. | Exam\_time\_table |
| 24 | father\_name | Varchar | 20 | Not Null | Store the fathers name of student | Student |
| 25 | first\_name | Varchar | 20 | Not Null | Store the first name of student | Student |
| 26 | from\_time | Varchar | 5 | Not Null | Store the exam start time. | Exam\_time\_table |
| 27 | Gender | Varchar | 6 | Not Null | Store the gender of student | Student |
| 28 | Gender | Varchar | 6 | Not Null | Store the gender of staff | Staff |
| 29 | group\_name | Varchar | 10 | Not Null | Store the group name | Subject |
| 30 | is\_absent | Varchar | 7 | Not Null | It stores the student absentee if student absent for exam .by default its value is present | Marks\_20 |
| 31 | is\_absent | Varchar | 7 | Not Null | It stores the student absenty if student absent for exam .by default its value is present | Marks\_80 |
| 32 | is\_delete | Varchar | 4 | Not Null | Use to delete the deparment | Department |
| 33 | is\_delete | Varchar | 4 | Not Null | Use to delete the subject | Subject |
| 34 | Last\_name | Varchar | 20 | Not Null | Store the last name of student | Student |
| 35 | last\_name | Varchar | 20 | Not Null | Store the last name of staff | Staff |
| 36 | login\_id | varchar | 10 | Primary Key | Store the login id | Login |
| 37 | login\_pwd | Varchar | 20 | Not Null | Store the login password | Login |
| 38 | marks\_20\_id | Int | 10 | Primary Key | Store the mark id out of 20 marks | Marks\_20 |
| 39 | marks\_80\_id | Int | 10 | Primary Key | Store the mark id out of 80 marks | Marks\_80 |
| 40 | middle\_name | Varchar | 20 | Not Null | Store the middle name of student | Student |
| 41 | mother\_name | Varchar | 20 | Not Null | Store the mothers name of student | Student |
| 42 | order\_exam\_date | Varchar | 10 | Not Null | Store the order exam date . | Exam\_time\_table |
| 43 | order\_post | Int | 4 | Not Null | Deseeding order | Seat\_no |
| 44 | order\_pre | Varchar | 4 | Not Null | Ascending order | Seat\_no |
| 45 | out\_of\_20 | Int | 2 | Not Null | Store the out of 20 marks | Marks\_20 |
| 46 | out\_of\_80 | Int | 2 | Not Null | Store the out of 20 marks | Marks\_80 |
| 47 | Pin\_code | Varchar | 6 | Not Null | Store the pin code of student | Student |
| 48 | roll\_no | Varchar | 4 | Not Null | Store the roll no of students | Admission |
| 49 | seat\_no | Varchar | 7 | Not Null | Store the seat no of student | Seat\_no |
| 50 | seat\_no\_id | Int | 7 | Primary Key | Store the seat no id of student | Seat\_no |
| 51 | Staff\_bdate | Date | 10 | Not Null | Store the birth date of staff | Staff |
| 52 | Staff\_date\_join | Date | 10 | Not Null | Store the joining date | Staff |
| 53 | Staff\_date\_retire | Date | 10 | Not Null | Store the retire date | Staff |
| 54 | staff\_fname | Varchar | 10 | Not Null | Store the first name of staff | Staff |
| 55 | Staff\_id | Int | 10 | Primary Key | Store the staff id | Staff\_id |
| 56 | student\_id | Varchar | 10 | Foreign key | Store the student id of students. | Subject\_exam |
| 57 | student\_id | Varchar | 10 | Foreign key | Store the student id of students | Exam\_time\_table |
| 58 | student\_id | Int | 10 | Primary Key | Store the student id | Student |
| 59 | Student\_id | Int | 10 | Foreign key | Store the student id | Admission |
| 60 | Student\_type | Varchar | 7 | Not Null | Store the subject type | Subject |
| 61 | subject\_code | Varchar | 10 | Not Null | Store the subject code | Subject |
| 62 | subject\_exam\_id | Int | 10 | Not Null | Store the subject exam id | Subject\_exam\_id |
| 63 | subject\_exam\_id | Int | 10 | Foreign key | Store the subject exam id | Marks\_20 |
| 64 | subject\_exam\_id | Int | 10 | Foreign key | Store the student subject exam id | Marks\_80 |
| 65 | subject\_exam\_id | Int | 10 | Primary Key | Store the subject exam id | Subject\_exam |
| 66 | subject\_id | Int | 10 | Primary Key | Store the subject id | Subject |
| 67 | subject\_name | Varchar | 10 | Not Null | Store the subject name | Subject |
| 68 | subject\_type | Varchar | 7 | Not Null | Store the subject type | Subject |
| 69 | Taluka | Varchar | 10 | Not Null | Store the taluka of student | Student |
| 70 | to\_time | Varchar | 5 | Not Null | Store the exam finish time . | Exam\_time\_table |
| 71 | unipune\_id | Varchar | 10 | Not Null | Store the uni pune id | Student |

**CHAPTER 4**

**User Manual**

**4.1 User Manual and purpose :**

The user manual is provided with the system to guide the user of the system. The user by the help of this user manual will be guided right from the login till receiving the facilities

such as information .

* **How to login:**

User is required to login with proper login name and password. After pressingenter key user directly access to home page of system

* **How to use Admin Home page:**

Admin home page provides various menus such as save exam subject, Shift exam subject from tern end to final, delete exam subject.

Admin can store student details, new admission. Also store staff details, department details etc.

**Purpose:-**

For any system to be successful it is important that the intended user find the system easy to operate. The purpose of the user manual is to make user acquainted with the system and help user understand the system and operate it conveniently.

The manual contain several screen shots that describes how to use the entire system. The purpose of the user manual is to make user acquainted with the system and help user understand the system and operate it conveniently. The manual contain several screenshots that describes how to use the entire system.

**Requirements:-**

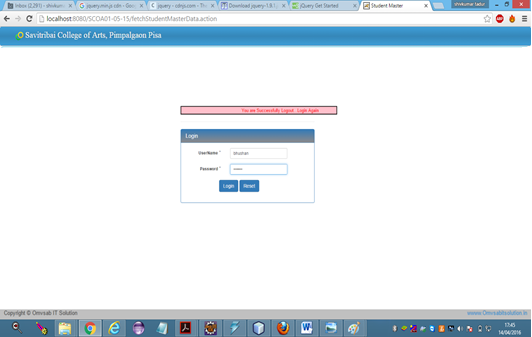
1. Computer system

2. Internet connection

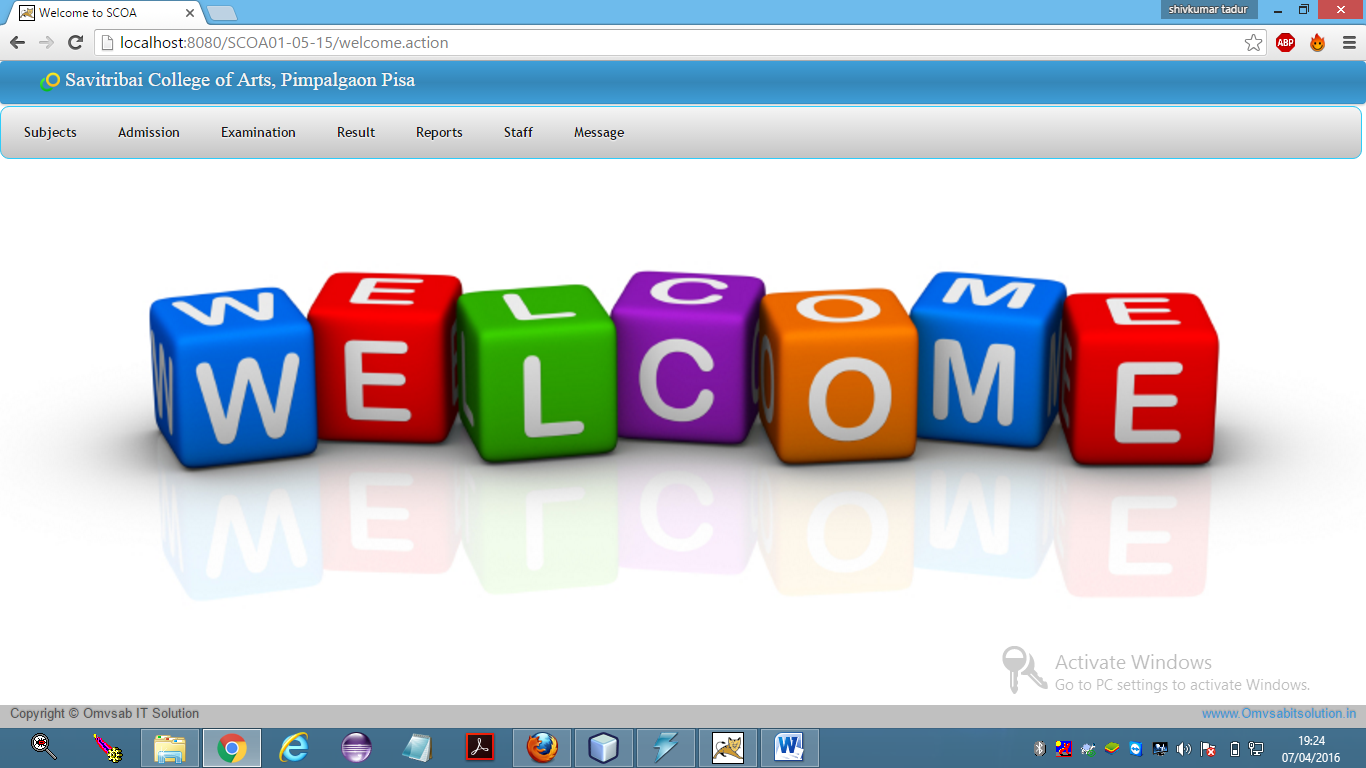
3. Client must have some knowledge of computer.

**4.2 Forms and Reports**

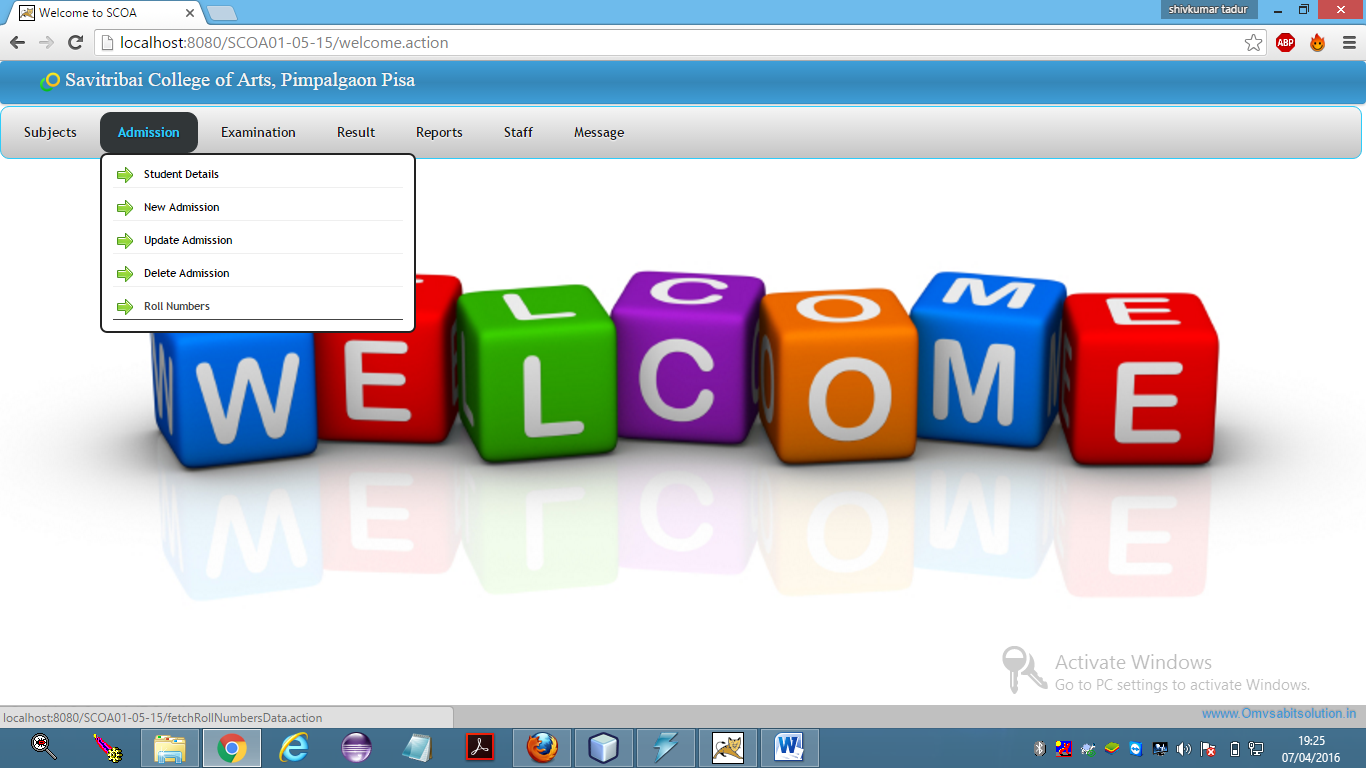
**1.Login:**

****

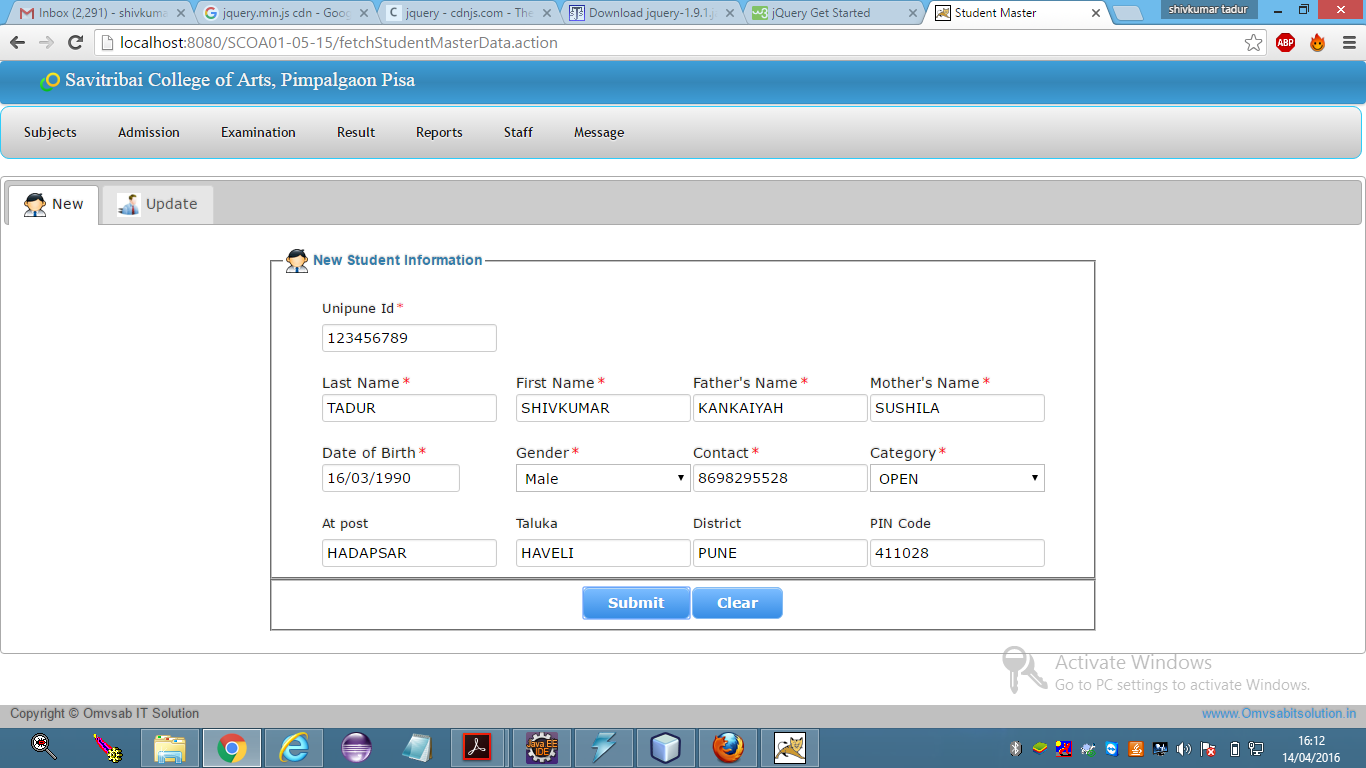
**2.Homepage-**



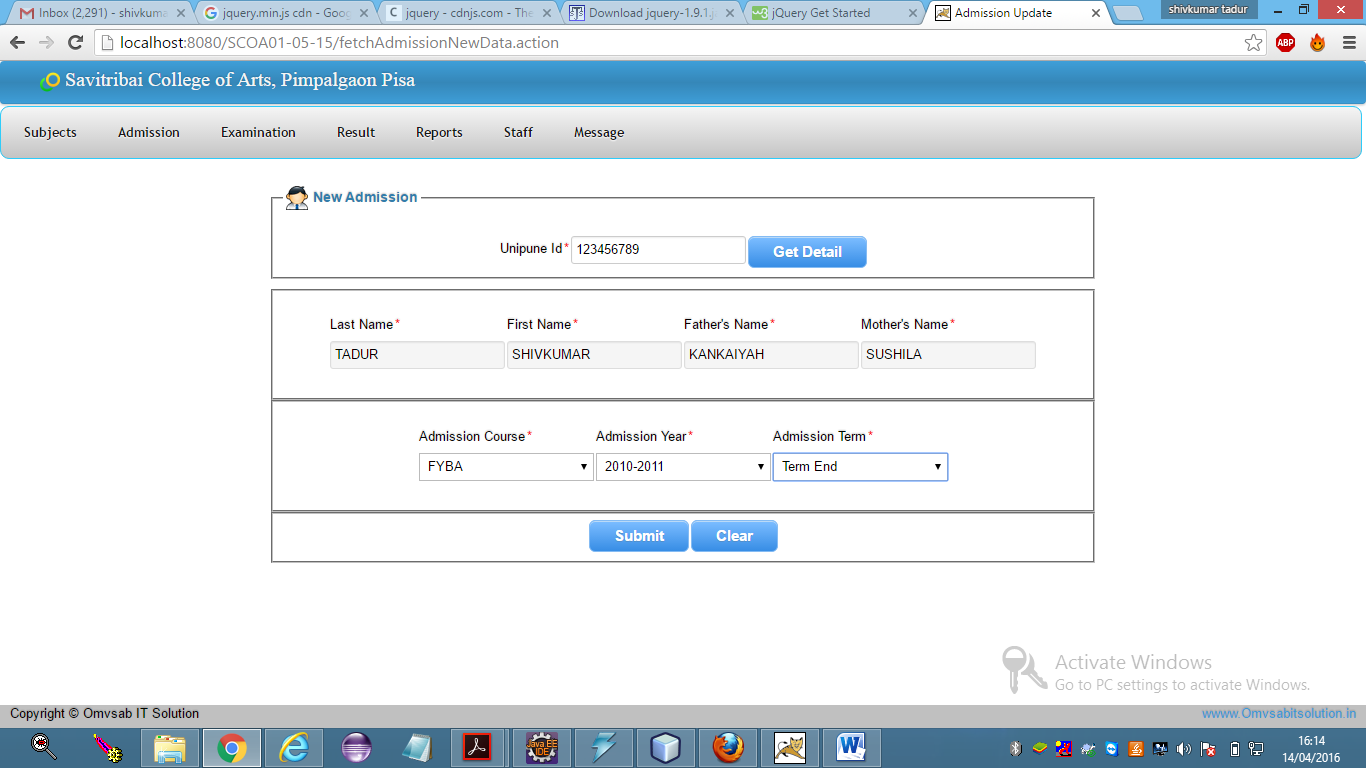
**3.Admission**



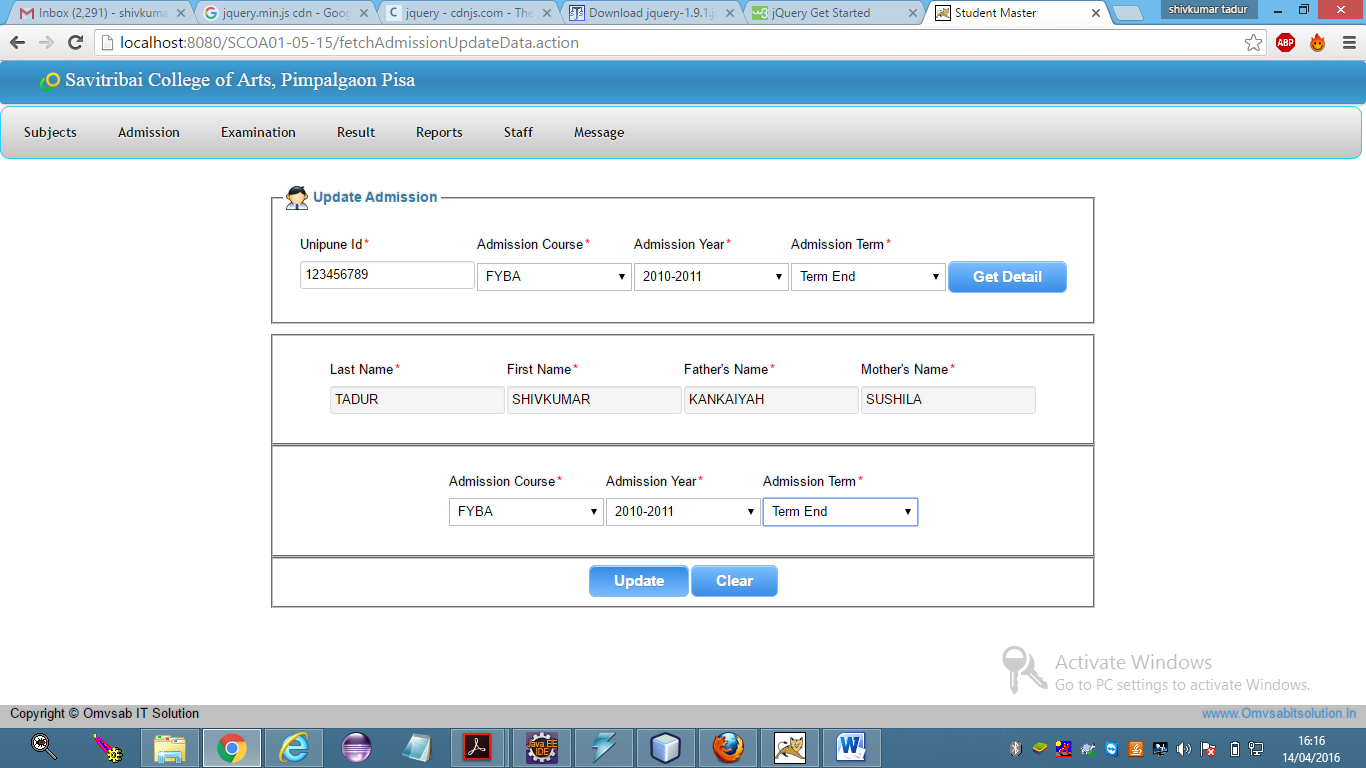
**4.New Student Detail**



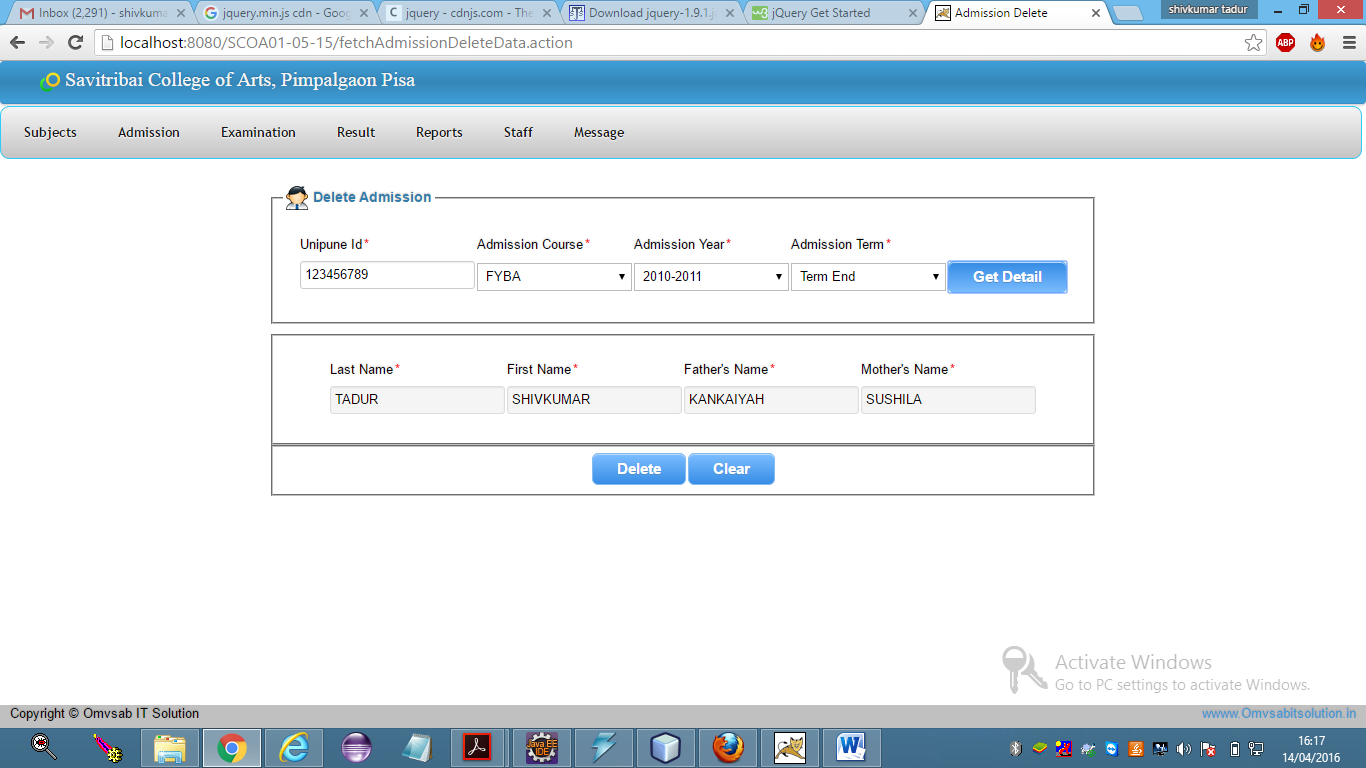
**5.New student admission:**



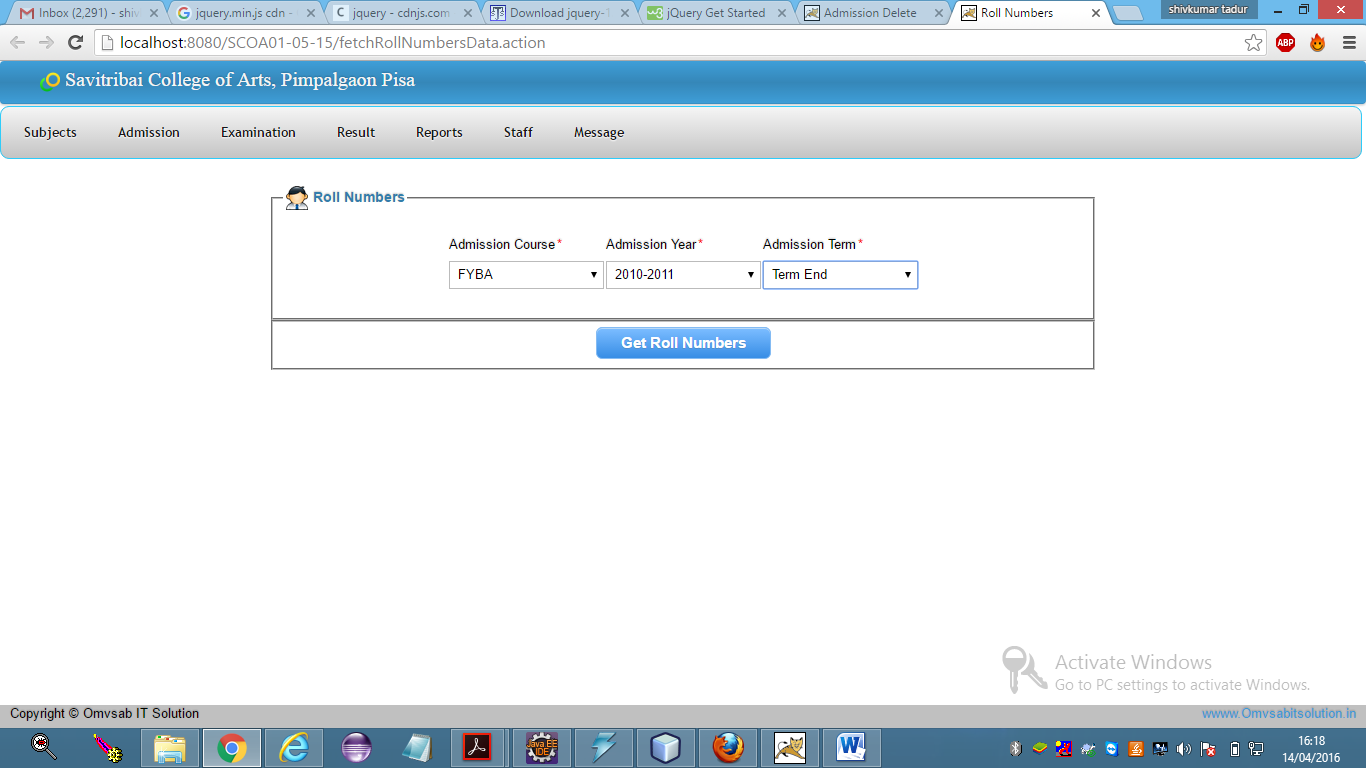
**6.Update admission:**



**7.Delete Admission**



**8.Generate roll number:**





**9.Save exam subject:**



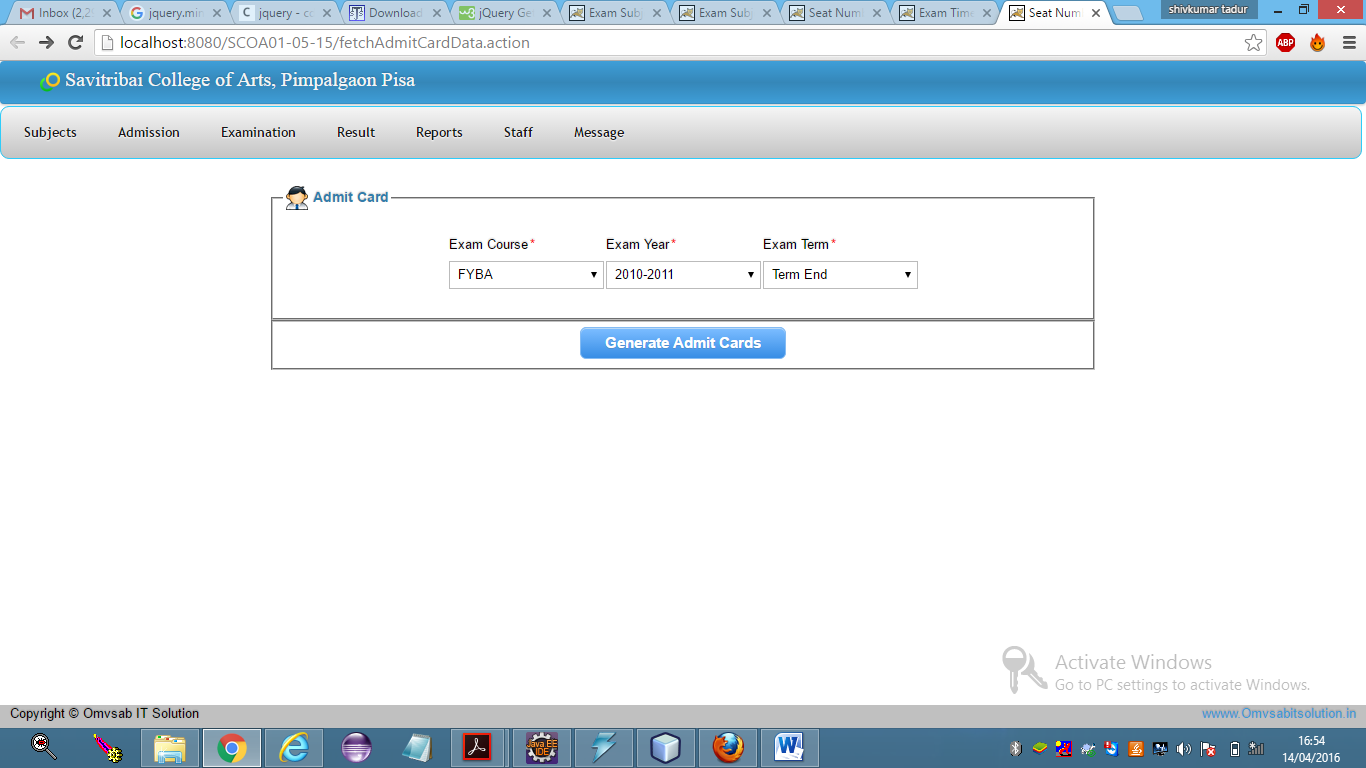
**10.Generate seat number:**



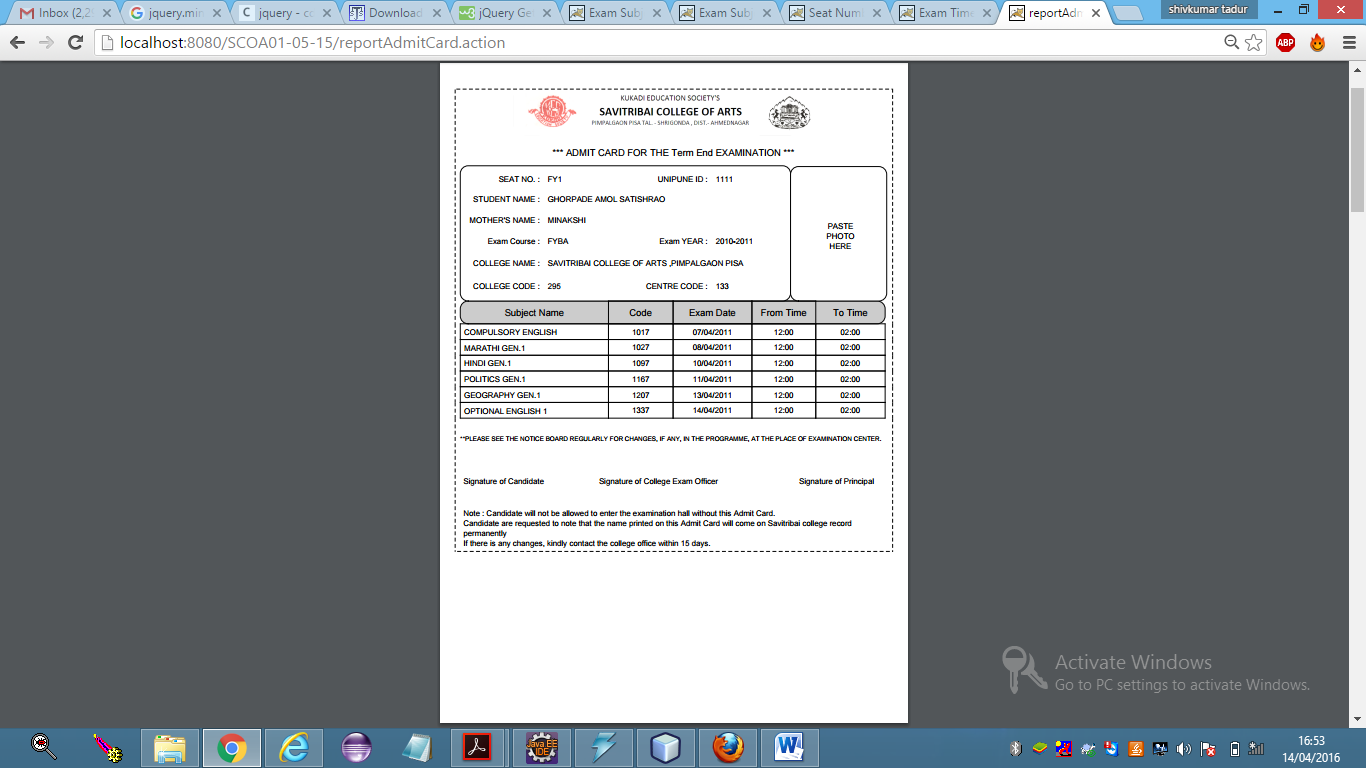
**11.Exam time table:**



**12.Generate admit card:**



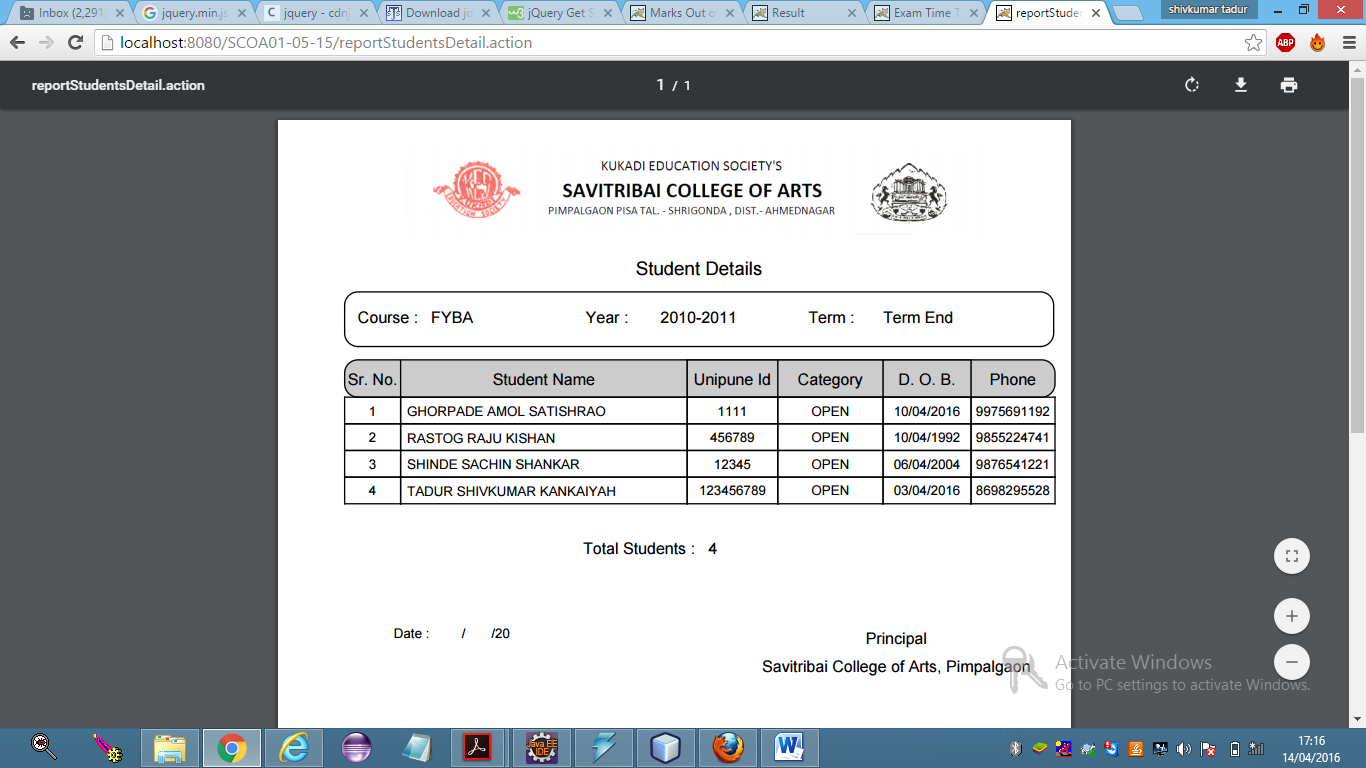
**13.Admit card:**



**14.Marks out of 20:**



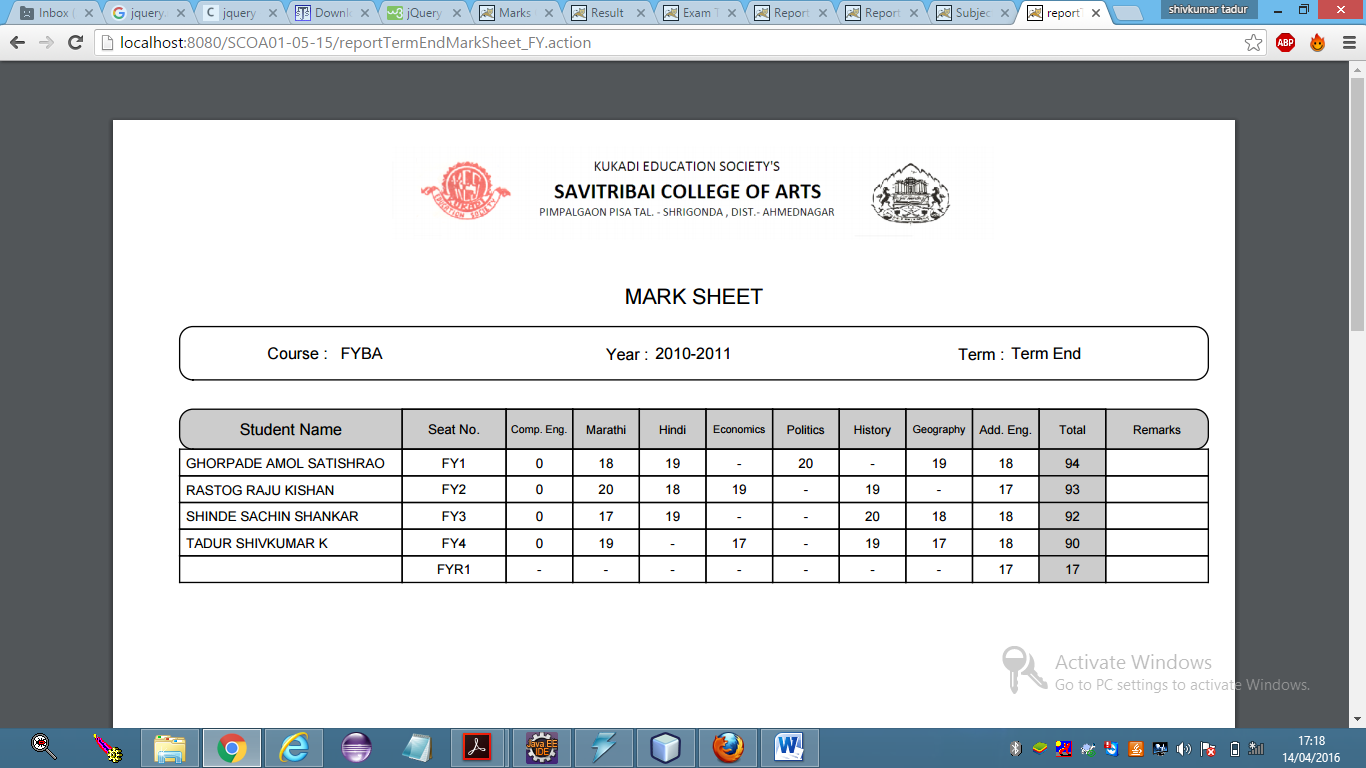
**15.Student admission details report:**



**16.Subject wise report:**



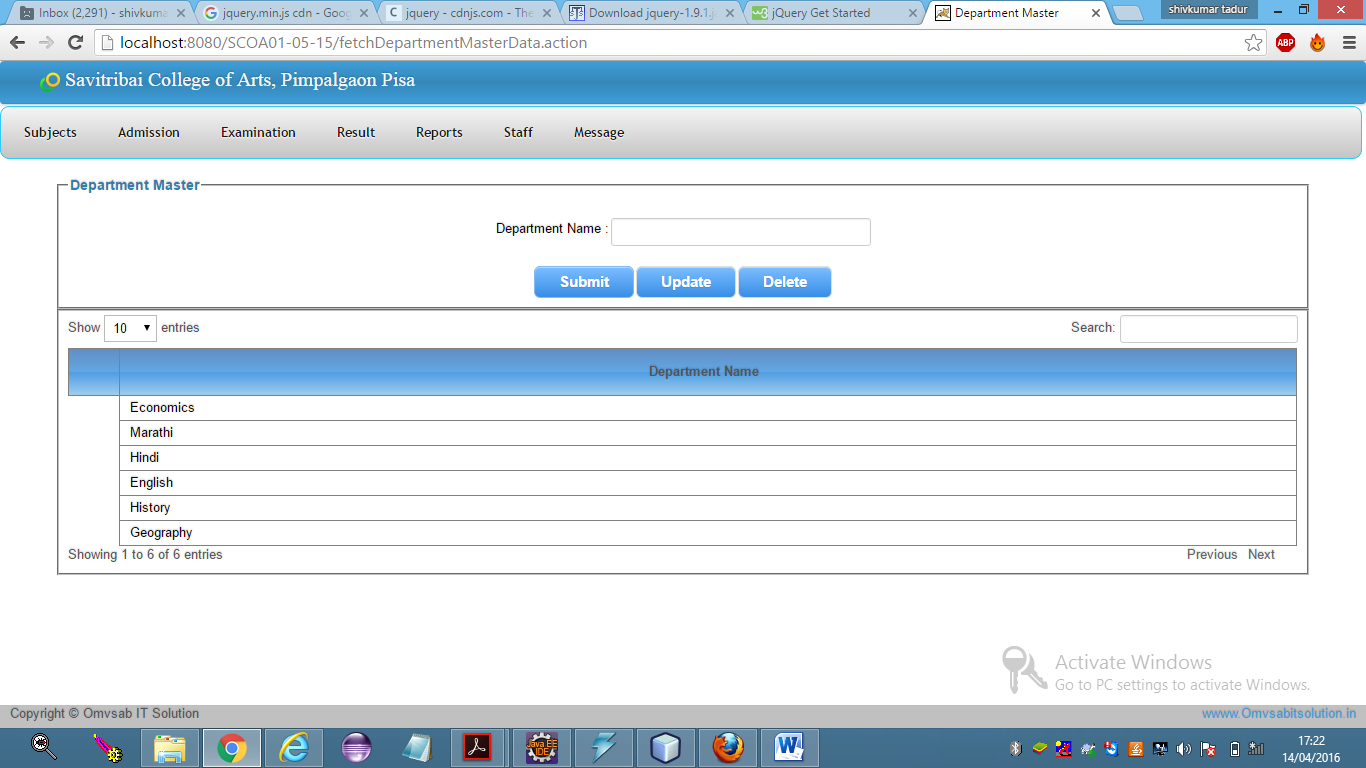
**17.Student markshit report:**



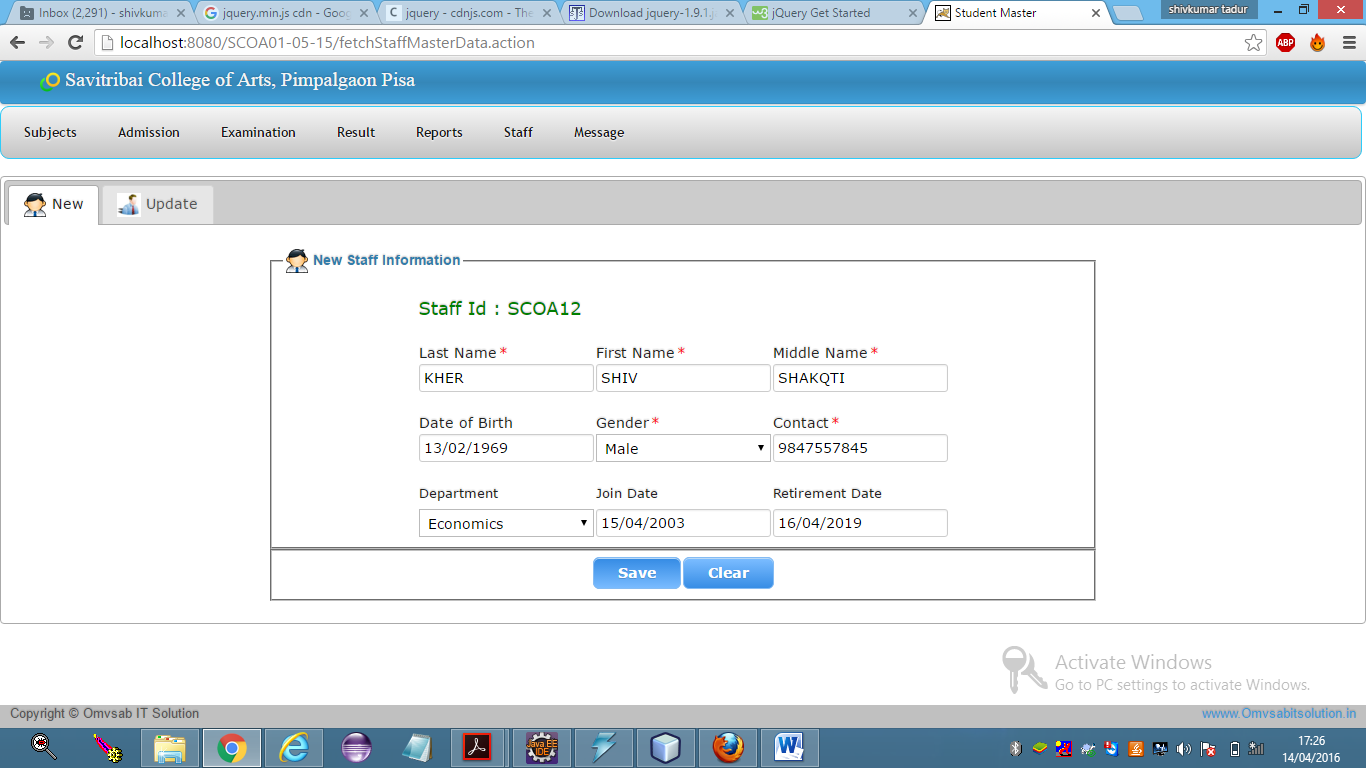
**18.Group(subject) wise result report:**



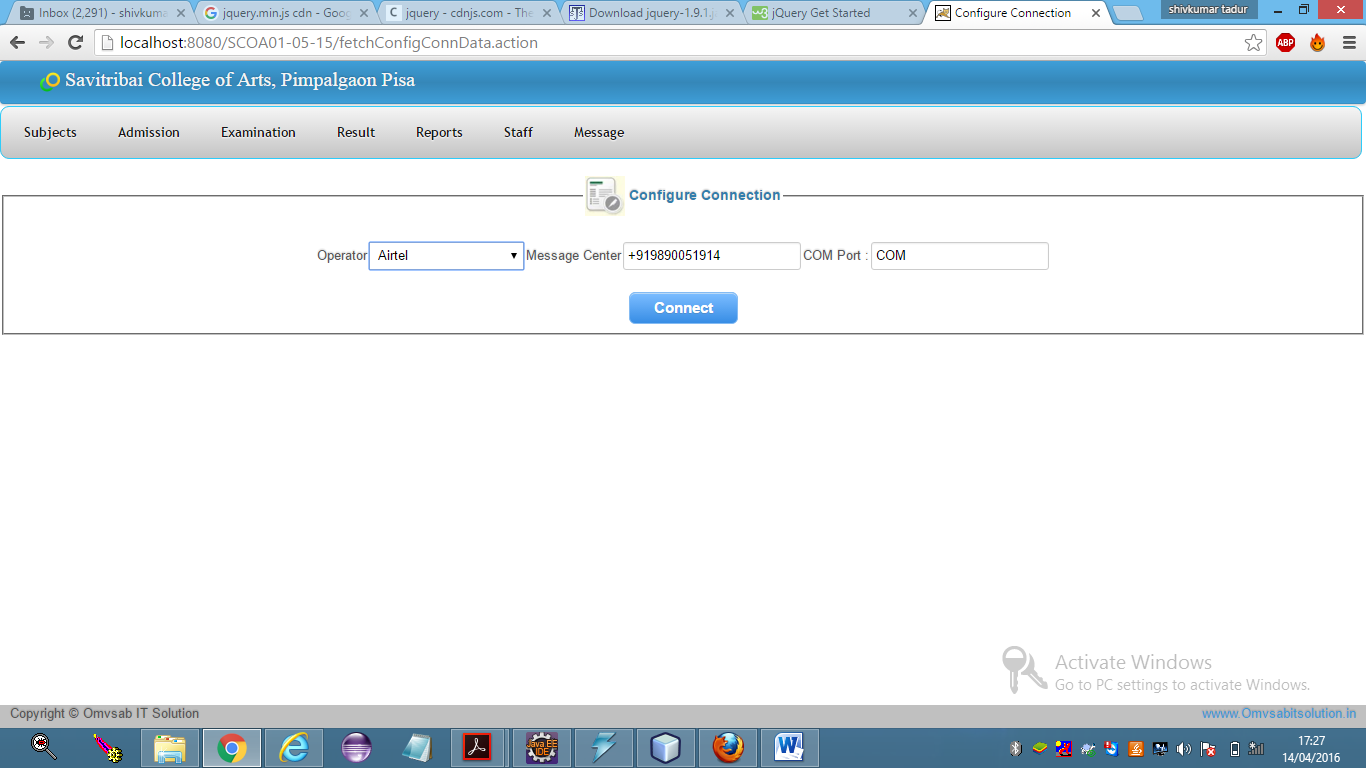
**19.Department master:**



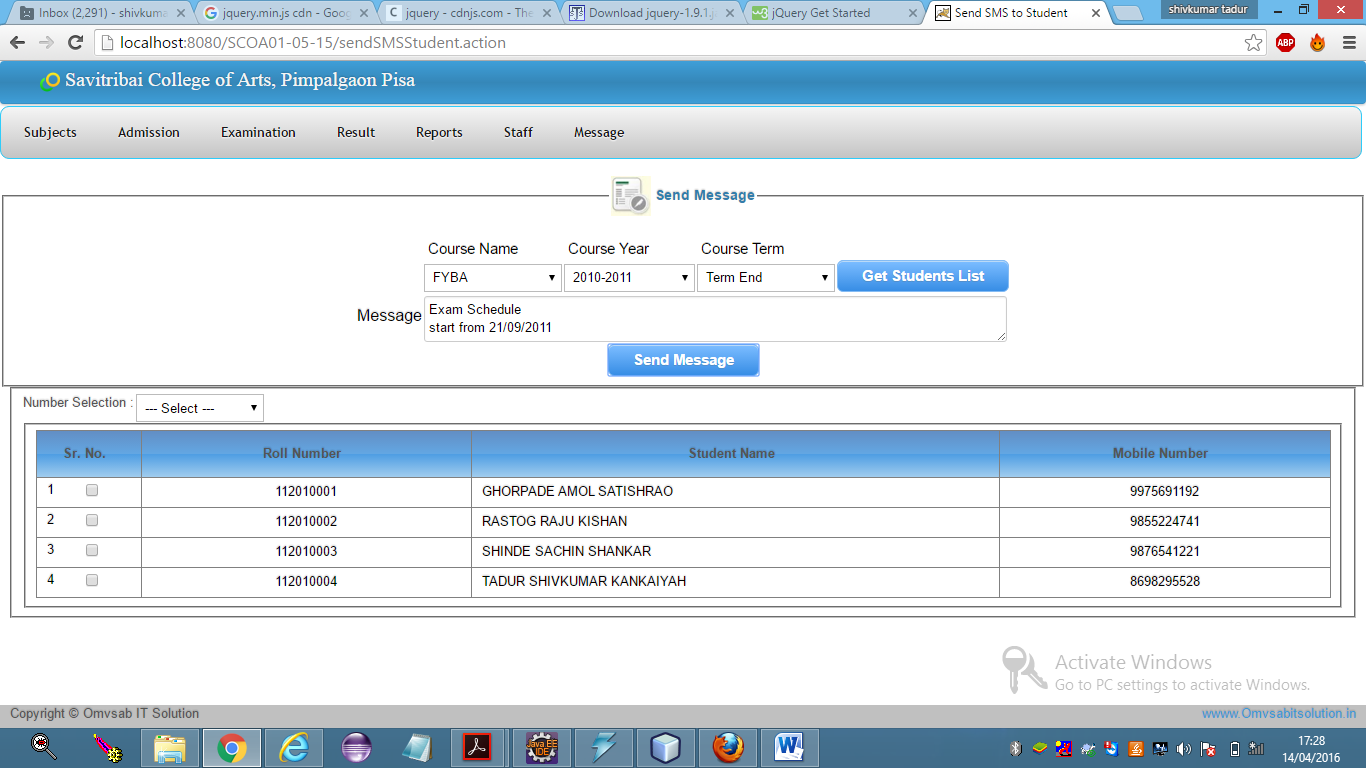
**20.New staff:**



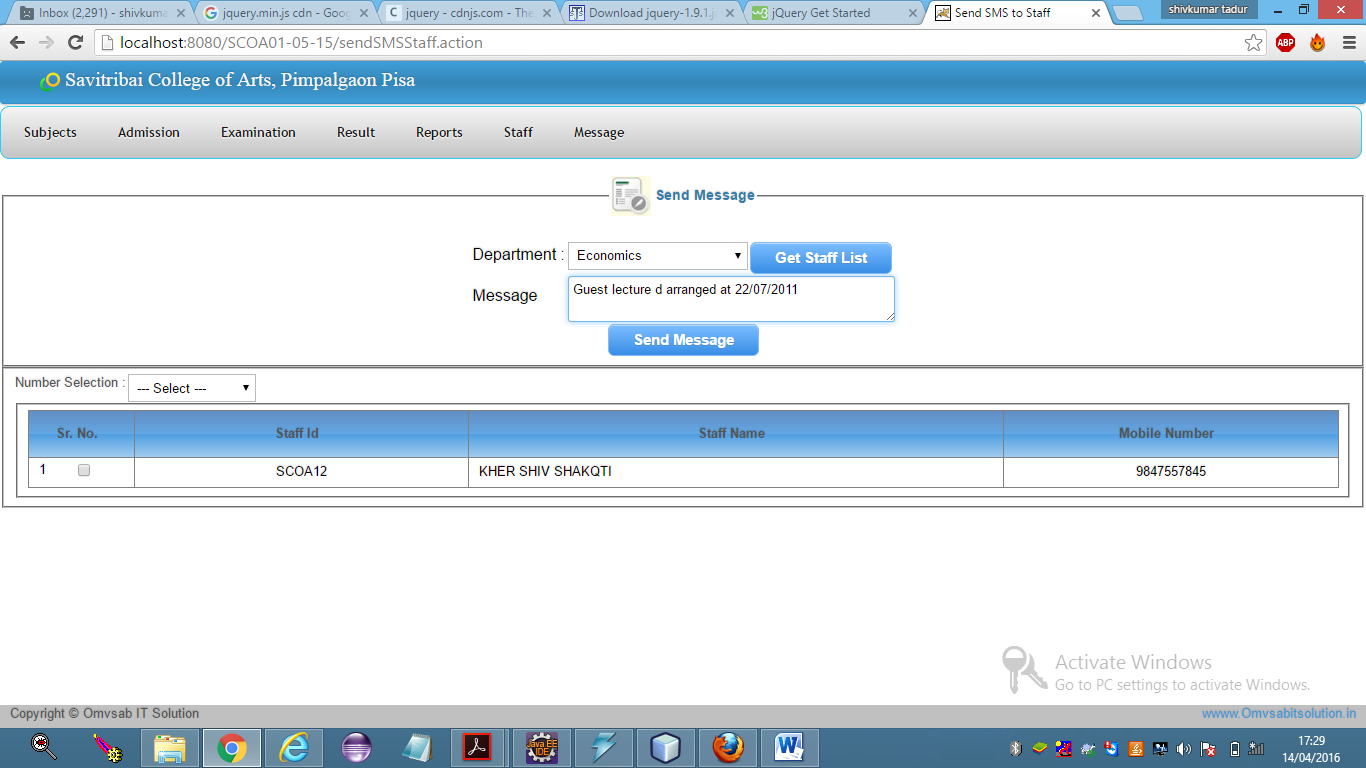
**21.Message master: 21 A. Configure connection:**



**22. Send sms to student:**



**23.Send sms to staff:**



**4.3 TEST PROCEDURE AND IMPLEMENTATION**

**Test Cases:**

****

****

****

****

**Test Procedure and Implementation:**

Testing is vital to the success of the system. Testing has to use to be done for accuracy. Efficiency and effectiveness.

**Preparation of test data:-**

Software testing in the process of executing a program with the intent of finding an error. A successful test one that covers an error yet undiscovered. Testing servers to demonstrate that all functions in the system are fully operational and system performs according to specifications. The test data is prepared on the Requirements specified by the user. The user for the system also gives the sample data.

Following strategies are applied while Testing:-

***Unit Testing:-***

As and when modules implemented. It is tested at the same time. In this testing primarily the data structure defined in the module were checked. Input parameters, return values (interface) for input and output operations were examined

Error reporting is done with proper error message.

***Integration Testing:-***

After unit testing integration testing was supplied on integrated modules in order to fix the bugs occurred after module combination. In the testing, interface integrity and functional validity of the integrate module were the major checking issues.

***Requirements Testing:-***

System as a whole at the completion is tested against requirements specified in the analysis phase. This testing resulted into satisfaction of the user requirements that were targeted for the system.

***Alpha and Beta Testing:-***

Alpha and beta testing was done after all above testing stages to recover errors that only user can find. In this phase recording of software usage problems was done. As a result of problems reported report during beta testing, modification and enhancements are done.

**CHAPTER 5**

**DRAWBACKS AND LIMITATIONS**

**DRAWBACKS AND LIMITATIONS**

* The cost of computer and associated equipments is falling steadily. Still these equipments are and continue to be much more costly as compared to manual processing equipments like pen and pencil.
* Also technological complexity of these equipment makes it more difficult to learn and maintain these equipments as compared to equipments of manual information processing.
* Use of computers require additional infrastructure, such as power supply and software back up. This increases the chances of problems due to failure of infrastructure.
* Computers generally reduces the errors of processing. However if there is an error in input data, it can result in major blunders in the processing which the computer system is unable to detect.

**CHAPTER 6**

**Proposed Enhancements**

**Proposed Enhancements:**

* In future enhancement we can add payment module
* We can add new functionality as per college requirement
* Future expansion can be done in concise manner in order to improve the efficiency of the system
* The software has been developed in such way that it can accept modification and further changes.

**CHAPTER 7**

**CONCLUSION**

**CONCLUSION**

Computerized system is more effective than manual system.

It reduces paper work and also provides flexibility for data Retrieval.

Burdon of manual work is reduced as whenever transaction take place, there is no need to record it in many places.

**CHAPTER 8**

**BIBLIOGRAPHY**

**Bibliography:**

* Professional JSP 2nd Edition- Brown, S.(2001).
* Jdbc, Servlet, And Jsp Black Book, New Edition. Dreamtech Press- Kumar, S. K. (2008).
* Java2 complete reference. -Schildt, H. (2006).
* The Java Language Specification Addision Wesley Publishing Company.- Goshling(1996).
* The Java Language Environment Sun Microsystems- McGilton(1996)..

**Web site References:**

* <http://jakartha.apache.org>
* [www.programmerheaven.com](http://www.programmerheaven.com)
* <http://indiastudychannel.com>
* [www.develoeradvantage.com](http://www.develoeradvantage.com)

**ANNEXURE**

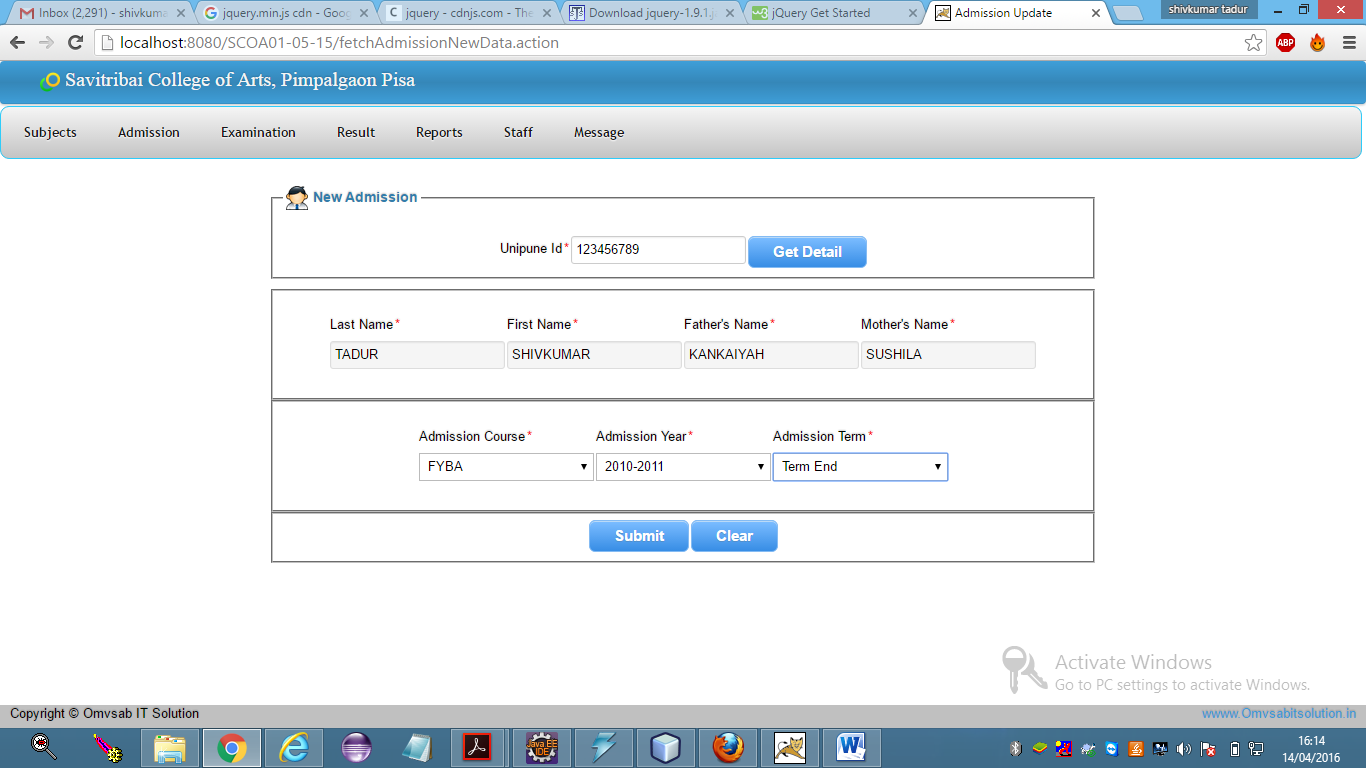
ANNEXURE 1:Input forms with Data

ANNEXURE 2:Output forms with Data

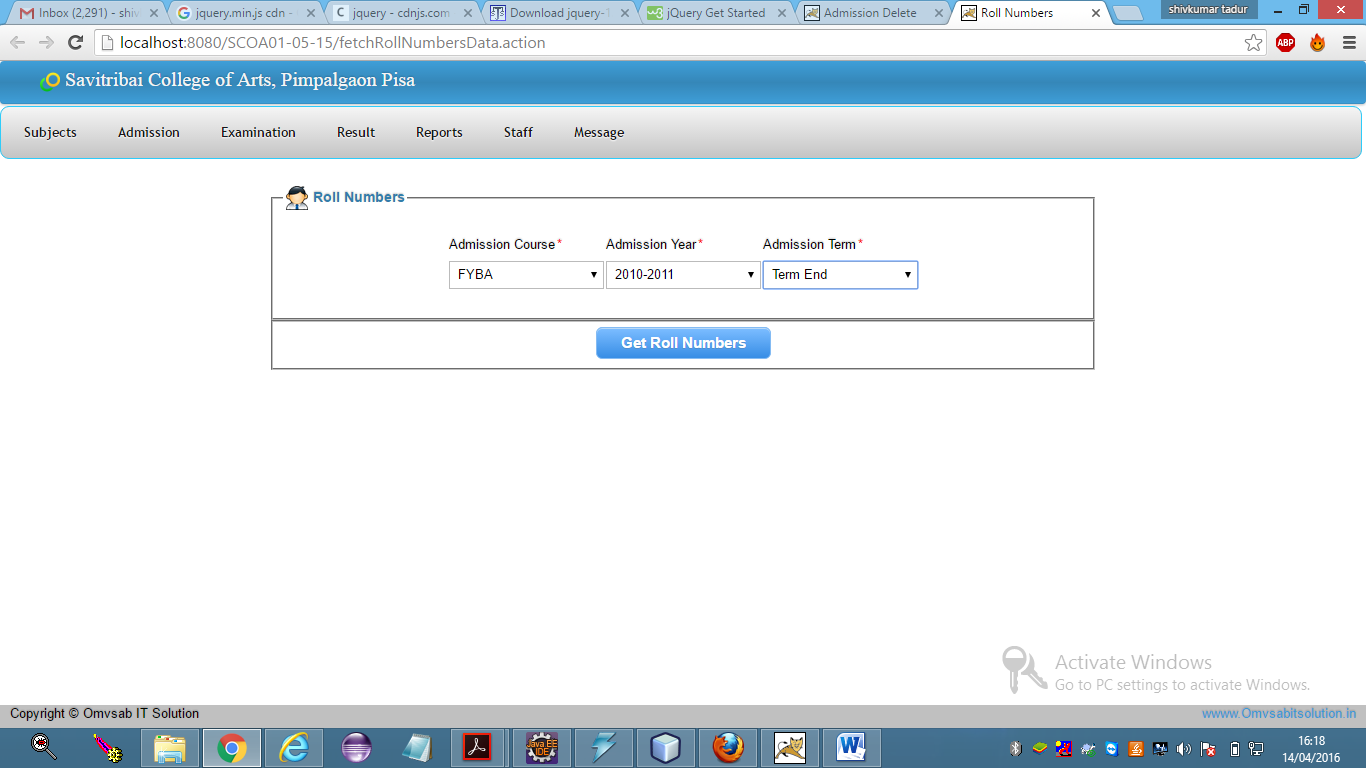
ANNEXURE 3:Sample Code

**1.Input forms with data**

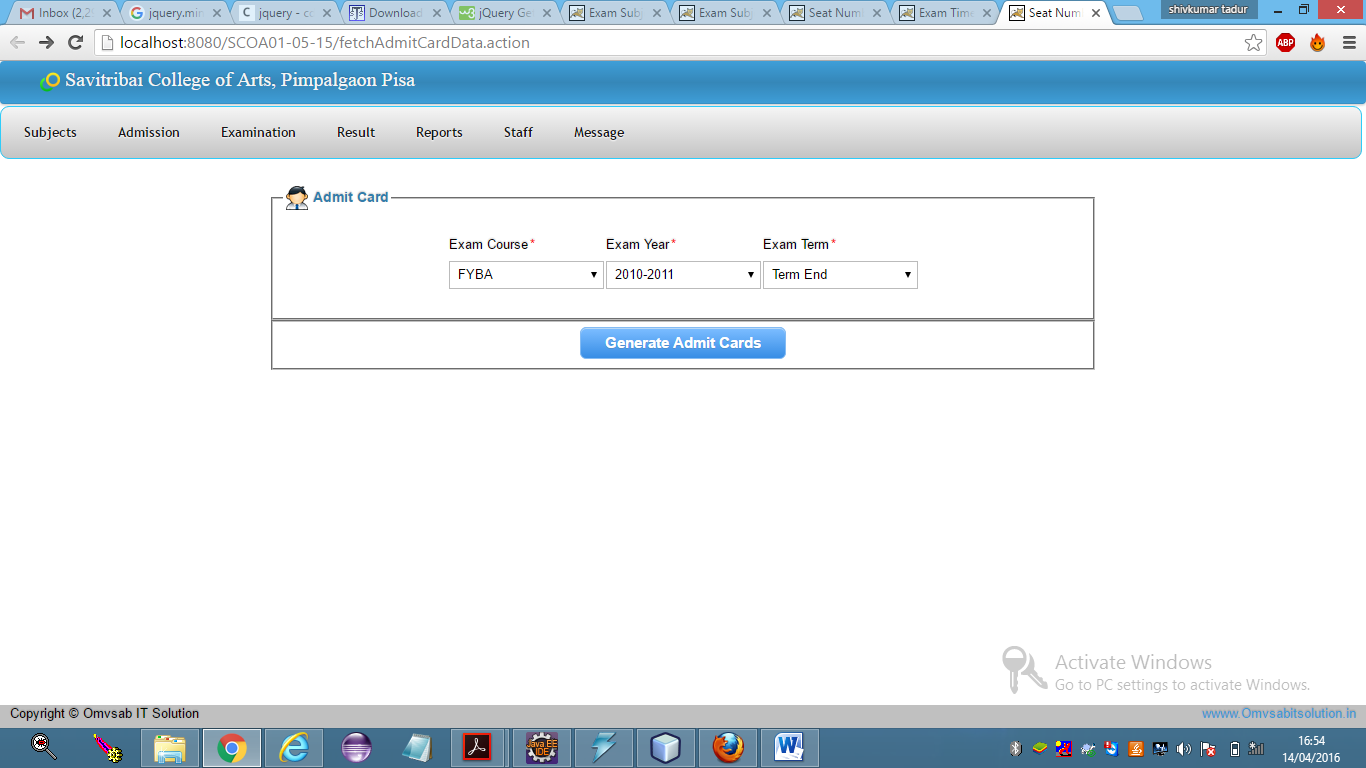
**1.New admission**



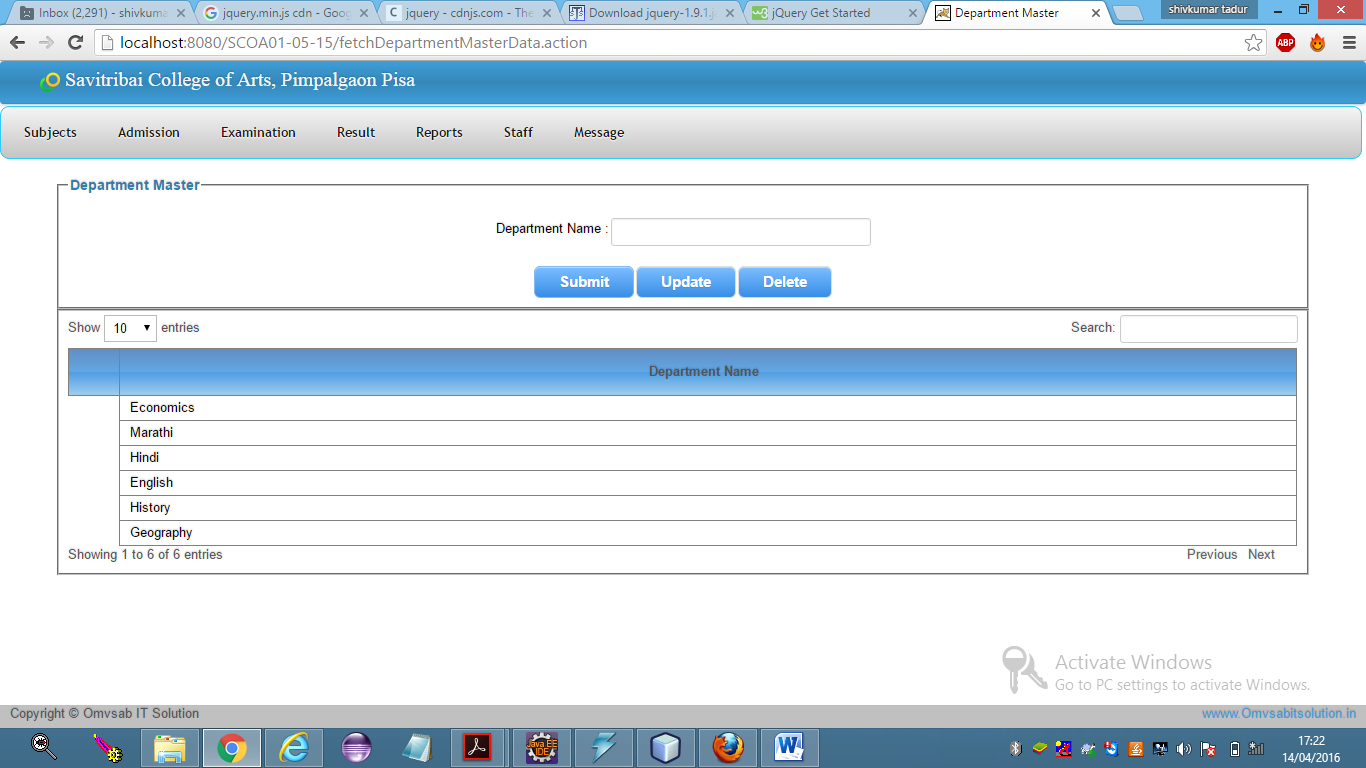
**2.Get roll number**



**3.Get Admit Card**



**4.Department Master**



**2.Output Reports with Data**

**1.Get roll number**



**2.Save exam subject**

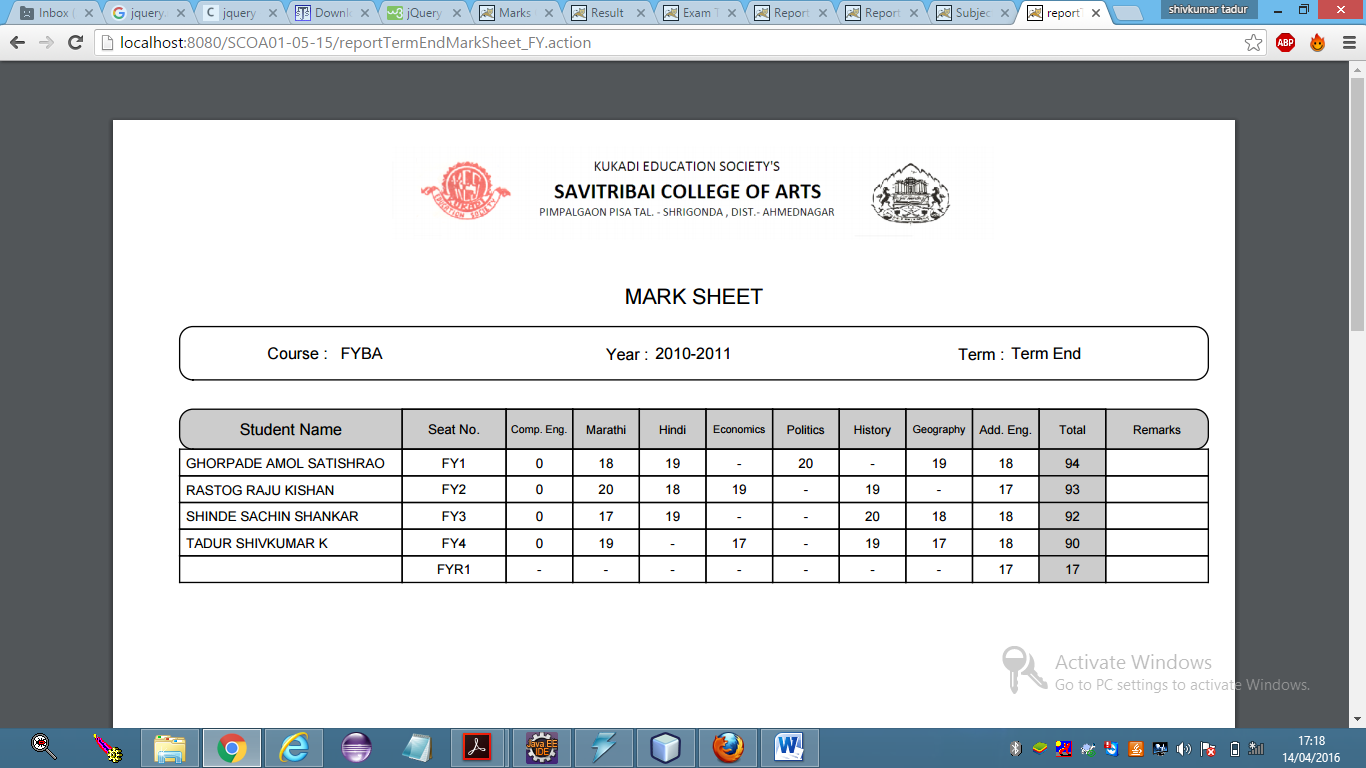


**3.Generate roll number**



**4.Exam Time table**



**5.Student marksheet**

**ANNEXURE 3 : SAMPLE PROGRAM CODE**

**Code for Authentication**

Package com.extermity.filter;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class AuthenticationFilter implements Filter{

@Overtide

public void destroy() {

}

@Override

public void doFilter(ServletRequest req,ServletResponse resp,

FilterChain filterChain)throws IOException,

ServletException{

HttpServletRequest request=(HttpServletRequest) req;

HttpServletResponse response=(HttpServletResponse)resp;

HttpSession session=request.getSession(flase);

if(session=null)[

System.out.println(“Other”);

RequestDispatcher rd= request.getRequestDispatcher(“login.jsp”);

rd.forward(request,response);

}

System.out.println(“Same”);

filterChain.doFilter(request,response);

}

@Override

public void init(FilterConfig filterConfig)throws ServletException{

}

}

**Sample code for login in (jsp page)**

<!DOCTYPE html>

<%@page import= “com.sun.xml.internal.ws.api.ha.StickyFeature”%>

<html>

<head>

<meta charset=”utf-8”>

<link herf=”dts\_web\_content/css/login.css”rel =’stylesheet’ type = ‘text/css’/>

<link herf=”dts\_web\_content/css/loginalert.css”rel =’stylesheet’ type = ‘text/css’/>

<meta name =”viewpoint” content=width=device-width,initial-scale=1”>

<script type=”application/x-javascript”>

addEventListener(“load”,function())

{

setTimeout(hideURLbar,0);

}

false;

function hideURLbar()

{

window.scrollTo(0,1);

}

</script>

<!—webfonts-->

<!—

<link href =’http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,700,800’

rel=’stylesheet’type=’text.css’?>-->

<!--//webfonts-->

</head>

<body>

<div class=”main”>

<%String error=(String)request.getAttribute(“msg”);

if(error!=null)

{

<%--<%--=error%>--%>

<div class = “alert alert-error>

<a class= “close” data-dismiss=”alert”> </a>

<strong>Incorrect</strong> The User Id and Password Does not Belong To Any Account.

</div>

<%}%>

<form action = “login.demo” method =”post”>

<h1>

<span>

<font size =”6px”> Employer </font></span><lable><font size=”6px”>Login </font>

</lable>

</h1>

<div class =”inset”>

<p>

<lable for = “username”><font color=”white size=”5px”><b>User\_Id </b>

</font>

</label>

<input type=”text” placeholder=” “ required name = “username”/>

</p>

<p>

<label for = “password”><font color=”white “ size =”5px” ><b> Password </b></font>

</lable>

<input type=”password” placeholder=” “ name = “password” required/>

</p>

<p>

<input type=”chackbox “ name = “remember”id= “remember”/>

<lable for=”remember”> Remember me for 14 days </lable>

</p>

</div>

<p class = “p- container”>

<span> < a href =’#’> Forget password? </a></span>

< input type= “submit” value =”Login”>

</p>

</form>

</div>

<!-----start-copyright----->

<div class =”copy-right”>

<p>Designed By:- <a href =<http://anuthamtechonology.com>>AnuthamTechonology</a></p>

</div>

<!-------//end-copyright------>

</body>

</html>

**Sample code for Systm Login**

package com. Extremity.model;

public class SystemLogin {

private int userid;

private String username;

private String password;

private String uertype;

public int getUserid() {

return userid;

}

public void serUserid(int userid){

this.userid= userid;

}

public String getUsername() {

return username;

}

public void serUsername(String username){

this.username= username;

}

public String getPassword() {

return password;

}

public void setPassword (String password) {

this.password =password;

}

public String getUsertype() {

return usertype;

}

public void serUsertype(String usertype) {  
 this. usertype = usertype;

}

}