Introduction

Introduction

VCS (Virtual Classroom System) aims to promote a greater count of students to splurge into the field of Education. It integrates the benefits of a physical classroom with the convenience of a 'no-physical-bar' virtual learning environment, minus the commuting hazards and expenses. It will usher in the immense flexibility and sophistication in the existing learning platform structures, with the perfect blend of synchronous and asynchronous interaction. It provides a means of collaborative learning for the students.

With the ever-increasing popularity and accessibility of the Internet, it is only natural that the educational community should want to make use of this tremendous resource. Use of the Internet and Web are leading to significant changes in educational models. Effective exploitation of these changes requires adequate attention to understanding the technology, the educational processes and issues, student's characteristics, etc. As this use of Internet is increasing, a traditional classroom has shifted to E-Learning. While advancements in communication tools were easily adapted to learning methods, it was the introduction of the personal computer and the development of the Internet that would create the most radical transformation in higher education. Learning by computer can be as easy as communicating with your professor and fellow classmates via email, student utilizing an interactive CD-ROM. Thus, E-Learning can be defined an approach to facilitate and enhance learning by means of personal computers, CD-ROMs, and the Internet. It may be as simple as that teacher may simply post their material on Internet; students can read it online or can download it for further access. Since student won't be in a classroom with professor and classmates, he will need to be capable of independent learning. Instructor will provide him with a syllabus, course documents, and required readings. The interaction between the professor and the student will happen via e-mail, discussion board, forums etc. Since the class doesn't meet in a physical space at a scheduled time, the student will have to learn independently. He will be responsible for keeping up with the assigned reading and completing assignments according to the timeline on the syllabus. The growing popularity of E-Learning has introduced new terms to education, as Virtual Classroom, where student will be present with his professor and fellow learners in a classroom. They will not be present physically in the classroom but connected to the classroom via Internet.

Virtual classroom aims to simulate the experience of attending a class over the web. So everyone is able to see other participant virtually.

Objective of the project

The main objective of the project is to provide an online learning experience to registered students of the portal. The registered students and faculties can also interact with each other thus providing a healthy and friendly environment to learn while not present physically in the lecture hall.
Some advantages of VCS are:

- Removal of geographical barriers (Anywhere learning)
- Quicker to organize
- One to one communication

Scope of the project

- Students can choose courses, attend lectures, take tests, view their performance records as per their convenience.
- Attend lectures either at scheduled time or view lectures at a later time.
- Faculties can take lectures, upload assignments, evaluate answer sheets and also can upload lectures and other discussions in various formats as in videos, presentations.
- Upload and download of various files such as journals, notices and videos.
- Asynchronous communication in the form of Email, discussion boards that enable communication to occur at "convenient times" that suit student schedules and are not accessed at simultaneous or prearranged times.
- There can be forums etc to discuss various queries and to put up suggestions posted both by students and faculties.
- Admin can generate reports, logs, backup and recovery of database at any time.
- Students or faculty can be accessible to the books in the library at their convenient time for reference.
- One to one, many to one and many to many information sharing.
- Provision of resources to arouse the interest of students in extracurricular activities like public speaking etc and to grasp the chance to enhance their personal abilities.
- Students can also take up various quizzes which can help them to realize their inbuilt talents in various fields.

System Analysis

Identification of Need

Education over the Internet is the latest concept in spreading education to everyone. The conveniences of learning on line are numerous. Students and those interested in learning over the computer and can choose their own timing that is convenient to them and the classes can be taken at their own place.

This **Virtual Classroom System** is designed in such a way that the student i.e. client can communicate with the server when it is logged on and the client can retrieve the text files from the server by issuing the request. Clients have his or her own login name and password, which help them to get connected with the server. Here the students are provided with the facility of attending the class of their choice and can choose the faculty according to their wish.

Developing a **Virtual Classroom System** to promote a greater count of students to splurge into the field of Education. It integrates the benefits of a physical classroom with the convenience of a 'no-physical-bar' virtual learning environment, minus the commuting hazards and expenses. It will usher in the immense flexibility and sophistication in the existing learning platform structures, with the perfect blend of synchronous and asynchronous interaction. It provides a means of collaborative learning for the students.

If you're a teacher or student, you probably know that Virtual Class Room designed to help professors and instructors create and teach courses online or use online technology to help run classes. In educational software circles, it's also often called an e-learning system, a learning management system, or a virtual learning environment.

In the recent era of globalization, technological advancement has increased dramatically in every sphere including mainstream education. These advances have introduced new educational nomenclature i.e. "**Virtual Classroom**". Profound investments in technology in this decade have given rise to a worldwide explosion of information. Many educational institutions have been mystified by this information chaos. They are driven by the goal to use newly found access to global data communication. This step will increase enrolment and will award a vast range of degrees through massive investments in distance education programs. There has been much talk among educators that these acts begin to modify the students' worth to the academic world, as the students begin to assume both the tangible and intangible characteristics associated with those of a "Customer" as opposed to the characteristics of a student. Marketing strategies abound that beseech the "students-customer" to take advantage of "fast, universal access", "earn a degree in a short period of time", and other creative approaches that guarantee satisfaction and quick delivery of the degree-of-choice. Moreover, in the fast growing competition in the job market, there have been increasing demands for specialists, professionals over population, increasing awareness as well as demand for higher education, shortage of qualified teachers and infrastructure facility. **Virtual classroom system** has taken a lead role in the teaching-learning process.

Generically, the **virtual classroom system** is a teaching and learning environment located within a computer mediated communication system. It consists of asset of group communication and work "spaces" and facilities that are constructed in software. **Virtual Classroom System** allows you to incorporate dynamic, interactive training into your learning landscape and manage it across the enterprise. This reduces training costs while increasing impact, scope, and frequency of training to keep pace with your business-using only a Web browser. Ensure customers, partners, and employees are always up-to-date on new product releases, corporate initiatives, and soft skills. Train the widest audience possible with anytime, anywhere access to recorded training sessions.

Preliminary Investigation

This system provides an online solution to provide teaching and learning environment located within a computer mediated communication system. It consists of asset of group communication and work "spaces".

Existing System

- Existing system is not providing the information about faculty's achievements.
- Existing system doesn't provide online exams.
- Existing system doesn't have the facility to send the mails to other students.
- Existing system is not having the facility for faculty to upload the assignments.
- The Existing System doesn't provide the facility for the students to download the assignments.

Concept of Virtual Classroom

Just as the term virtual means a simulation of the real thing, Virtual Classroom is a simulated classroom via Internet, which provides a convenient communication environment for distance learners just like traditional face-to-face classroom. A virtual classroom allows learners to attend a class from anywhere in the world and aims to provides a learning experience that is similar to a real class room. When we go to college we have a schedule of lectures, which we must attend. Student must arrive on time, and when he enters the classroom, he finds a teacher, fellow learners, a blackboard or whiteboard, LCD projector, optionally a television screen with videos. Likewise, a Virtual Classroom is a scheduled, online, teacher-led training session where teachers and learners interact together using computers linked to a network such as the Internet.

A virtual classroom enables to bring learners from around the world together online in highly interactive virtual classes while greatly reducing the travel, time, and expense of on-site teaching/training programs. It can be used as a solution for live

delivery and interaction that addresses the entire process of creating and managing our teaching-learning process. It facilitates instructor and student in teaching-learning events, such as a seminar, online discussion or a live training for employees in company. As in traditional classroom, there are professor and fellow learners present with the student; we have many participants present in virtual classroom. They can talk with each other as in the traditional classroom via chat. Similarly presenter uses whiteboard, gives notes/resources, and gives presentation as given in traditional one. Thus, virtual classroom can be visualized as a classroom where a lecture or session is conducted using Internet. Now, that we have some idea about virtual classroom, we will discuss some advantages that virtual classroom offers over traditional classroom.

Feasibility Study

Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

Understand the different technologies involved in the proposed system:

1. Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.
2. Find out whether the organization currently possesses the required technologies:
    - Is the required technology available with the organization?
    - If so is the capacity sufficient?

Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Have the user been involved in the planning and development of the project? Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

"Will the current printer be able to handle the new reports and forms required for the new system?"

Economical Feasibility

Economical feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

Project Planning

The key to a successful project is in the planning. Creating a project plan is the first thing we should do when undertaking any kind of project.

Often project planning is ignored in favor of getting on with the work. However, many people fail to realize the value of a project plan in saving time, money and many problems.

1. Identifying project Goals

A project is successful when the needs of the stakeholders have been met. A stakeholder is anybody directly or indirectly impacted by the project.

As a first step, it is important to identify the stakeholders in our project. It is not always easy to identify the stakeholders of a project, particularly those impacted indirectly.

Examples of stakeholders are:

- The project sponsor.
- The customer who receives the deliverables.
- The users of the project outputs.
- The project manager and project team.

Once we understand who the stakeholders are, the next step is to find out their needs. The best way to do this is by conducting stakeholder interviews. Take time during the interviews to draw out the true needs that create real benefits. Often stakeholders will talk about needs that aren't relevant and don't deliver benefits. These can be recorded and set as a low priority.

The next step, once we have conducted all the interviews, and have a comprehensive list of needs is to prioritize them. From the prioritized list, create a set of goals that can be easily measured. A technique for doing this is to review them against the SMART principle. This way it will be easy to know when a goal has been achieved. The acronym SMART has a number of slightly different variations, which can be used to provide a more comprehensive definition for goal setting:
S - Specific, significant, stretching
M - Measurable, meaningful, motivational
A - Agreed upon, attainable, achievable, acceptable, action-oriented
R - Realistic, relevant, reasonable, rewarding, results-oriented
T - time-based, timely, tangible, tractable

Once we have established a clear set of goals, they should be recorded in the project plan. It can be useful to also include the needs and expectations of our stakeholders. This is the most difficult part of the planning process completed. It's time to move on and look at the project deliverables.

2. Project Deliverables

Using the goals we have defined, create a list of things the project needs to deliver in order to meet those goals. Specify when and how each item must be delivered. Add the deliverables to the project plan with an estimated delivery date. More accurate delivery dates will be established during the scheduling phase, which is next.

3. Project Schedule

Create a list of tasks that need to be carried out for each deliverable identified. For each task identify the following:

- The amount of effort (hours or days) required to complete the task.
- The resources who will carry out the task.

Once we have established the amount of effort for each task, we can work out the effort required for each deliverable, and an accurate delivery date. Update our deliverables section with the more accurate delivery dates. At this point in the planning, you could choose to use a software package such as Microsoft Project to create our project schedule. Alternatively, use one of the many free templates available. Input all of the deliverables, tasks, durations and the resources who will complete each task.

A common problem discovered at this point, is when a project has an imposed delivery deadline from the sponsor that is not realistic based on your estimates. If you discover that this is the case, you must contact the sponsor immediately. The options we have in this situation are:

- Renegotiate the deadline (project delay).
- Employ additional resources (increased cost).
- Reduce the scope of the project (less delivered).

Use the project schedule to justify pursuing one of these options.

4. Supporting Plans

This section deals with plans you should create as part of the planning process. These can be included directly in the plan.

Human Resource Plan:

Identify by name, the individuals and organizations with a leading role in the project. For each, describe their roles and responsibilities on the project. Next, describe the number and type of people needed to carry out the project. For each resource detail start dates, estimated duration and the method you will use for obtaining them. Create a single sheet containing this information.

Communications Plan:

Create a document showing who needs to be kept informed about the project and how they will receive the information. The most common mechanism is a weekly or monthly progress report, describing how the project is performing, milestones achieved and work planned for the next period.

Risk Management Plan:

Risk management is an important part of project management. Although often overlooked, it is important to identify as many risks to your project as possible, and be prepared if something bad happens.

Here are some examples of common project risks:

- Time and cost estimates too optimistic.
- Customer review and feedback cycle too slow.
- Unexpected budget cuts.
- Unclear roles and responsibilities.
- Stakeholder input is not sought, or their needs are not properly understood.
- Stakeholders changing requirements after the project has started.
- Stakeholders adding new requirements after the project has started.
- Poor communication resulting in misunderstandings, quality problems and rework.
- Lack of resource commitment.

Risks can be tracked using a simple risk log. Add each risk we have identified to our risk log; write down what you will do in the event it occurs, and what we will do to prevent it from occurring. Review our risk log on a regular basis, adding new risks as they occur during the life of the project. Remember, when risks are ignored they don't go away.

Project Scheduling

Project scheduling is the art of planning dates for starting and completing activities and milestones. According to the Project Management, here are five key processes to developing a project schedule.

Define Activities

The goal of the activity definition step is to identify all the tasks required to accomplish the product. This frequently results in identifying all the work products and deliverables that comprise the project. These deliverables are found as the components of a Work Breakdown Structure (WBS). The project schedule further decomposes these deliverables into the actual activities required to complete the work.

If the project team doesn't have an established scope statement, WBS, or sufficient scope definition, you may need to host a workshop or two to gather the requirements and further develop the project schedule. Since you need to produce a project schedule by next week, you will likely create tasks in your project schedule for "Analysis" or "Scope Definition." At this point in the project, it is OK to not have all the project details. You can build activities in your project schedule to gather the information. It is perfectly acceptable to build a plan for the analysis of the project before committing to the implementation or delivery phase of the project.

Assume for now you either have a WBS available or have enough information to build a sample set of tasks to further define the scope. Once you have all the activities defined, the next step is the sequence the activities.

Sequence Activities

At this point you've entered all the task names and have further decomposed the deliverables listed in the WBS. The next step is to sequence the activities with dependencies. During this step, you'll identify any dependencies of related tasks and document them in the project schedule. You'll need to analyze each of the tasks to understand which task has a dependency on additional tasks. In your favorite project schedule development book, be sure to read about the different types of dependency relationships include Finish-to-Start and Start-to-Start dependencies. These relationships will impact your task start and finish dates.

Estimate Activity Resources

The next step is to identify the resources and their availability to your project. Remember that not all team members will be 100% available to your project as some team members will be working on multiple projects. In this step, you'll also assign resources to each of the tasks. I usually assign resource to tasks using the standard Gantt chart view in Microsoft Project. For each task at the lowest point in

the WBS, click on the drop down box in the Resource Names column and select the available team member.

I recommend breaking down the tasks so you can assign one task to one resource to avoid adding multiple resources to a given task. It creates a larger project schedule, but it allows me better control in allocating and tracking resources as the project executes.

Estimate Activity Durations

With resources assigned, the next step is to estimate each task's duration. The activity's duration is the number of working periods required to complete the task. In Microsoft Project, this can be defined in days, weeks, and even months! It is also important to understand the difference of the different duration types including Fixed Work, Fixed Duration and Fixed Units. Selecting the correct duration type impacts the resource availability and the forecasted task end date.

Develop Schedule

The next step is to analyze the project schedule and examine the sequences, durations, resources and inevitable scheduling constraints. The goal of this step is to validate the project schedule correctly models the planned work. In this step you'll not only validate the duration estimates are accurate, but validate the resource allocations are correct.

Resource leveling is a key step in ensuring the project dates are realistic and the resources are appropriately assigned. Microsoft Project has an automatic resource leveling feature, but I recommend against using it. Instead of automatic leveling, I recommend using a manual process to resolve resource over-allocation. This manual process of resource leveling is time consuming, but it results in a better end project with realistic end dates.

Plan and Schedule More Complex Projects using Gantt Charts

Gantt charts are useful tools for analyzing, planning and controlling complex multi-stage projects. Gantt charts can:

- Assist in identifying the tasks and sub-tasks to be undertaken
- Help you lay out the tasks that need to be completed
- Assist in scheduling when these tasks will be carried out and in what order
- Assist in planning resources needed to complete the project
- Assist in working out the critical path for a project where it needs to be completed by a particular date

When a complex or multi-task project is under way, Gantt charts assist in monitoring whether the project is on schedule, or not. If not, the Gantt chart allows you to easily identify what actions need to be taken in order to put the project back onto schedule.

An essential concept behind project planning is that some activities depend upon other activities being completed first. For example, it is not a good idea to start building the walls in an office block before you have laid the foundations; neither is it a good idea to put the cake mix into the tin without greasing the tin first.

These are dependent activities which need to be completed in a sequence, with each stage being more-or-less completed before the next stage can begin. We can call such dependent activities 'sequential'.
Non-sequential activities are not dependent on the completion of any other tasks. These activities may be done at any time before or after a particular stage in the project is reached. These activities are called are non-dependent or "parallel" tasks.

To create a Gantt chart:

List all Activities/Tasks in the Plan

For each task, show the earliest possible start date, how long you estimate the length of time it should take, and whether it is parallel or sequential. If tasks are sequential, show which stages they depend on.
Head up a sheet of graph paper (using pencil and a ruler) with the days, weeks or months through to task completion on the top x-axis. The y-axis can be used to itemize each task in its order. You may want to use a spreadsheet for this instead of graph paper if you prefer.

Plot the Tasks onto the Plan

Next list the tasks in the first column on the left hand side of the page, the y-axis. To draw up a rough first draft of the Gantt chart; plot each task on the plan, showing it starting on the earliest possible date. Draw each task as a horizontal bar, with the length of the bar being the length of time you estimate the task will take. Above each task bar, mark the estimated time taken to complete the task. At this stage there is no need to include scheduling - all you are doing is setting up the first draft.

Schedule the Tasks/Activities

Now on a fresh sheet redraw the Gantt chart to schedule actions and tasks. Schedule these in such a way that sequential actions are carried out in the desired sequence e.g. dig holes, lay foundations, begin construction. Ensure that these dependent activities do not start until the activities they depend on have been fully completed.

Where possible, schedule parallel tasks so that they do not interfere with sequential actions on the critical path. While scheduling, ensure that you make best use of the time and resources you have available. Do not over-commit resources and allow some time in the schedule for holdups, overruns, quality rejections, failures in delivery, etc.

Once the Gantt chart is drawn, you can see how long will it take to complete your project. The key steps to be carried out to ensure successful completion of the project should be clearly visible.

The following is the Gantt chart drawn for Virtual classroom systems:

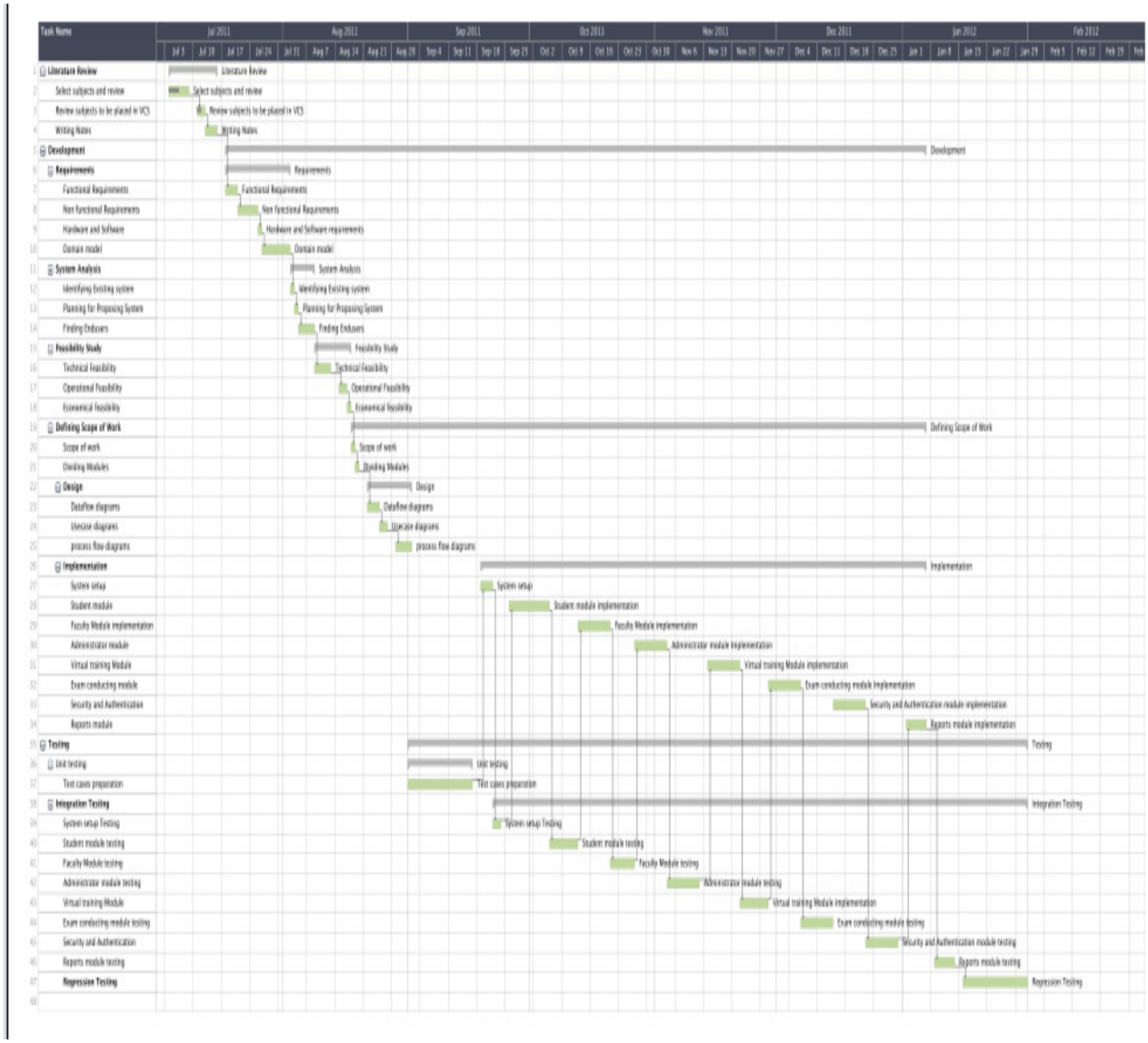


Figure 2.1: GANTT CHART for VCS

Pert Chart

The **Program Evaluation and Review Technique** commonly abbreviated PERT is a model for project management invented by United States Department of Defense's US Navy Special Projects Office in 1958 as part of the Polaris mobile submarine launched ballistic missile project. Pert chart is a statistical tool, used in project management, designed to analyze and represent the tasks involved in completing a given project.

PERT is a method for analyzing the tasks involved in completing a given project, especially the time needed to complete each task and identifying the minimum time needed to complete the total project.

Conventions:

- A PERT chart is a tool that facilitates decision making. The first draft of a PERT chart will number its events sequentially in 10s (10, 20, 30, etc.) to allow the later insertion of additional events.
- Two consecutive events in a PERT chart are linked by activities, which are conventionally represented as arrows (see the diagram above).
- The events are presented in a logical sequence and no activity can commence until its immediately preceding event is completed.
- The planner decides which milestones should be PERT events and also decides their "proper" sequence.
- A PERT chart may have multiple pages with many sub-tasks.

PERT is valuable to manage where multiple tasks are occurring simultaneously to reduce redundancy.

The first step to scheduling the project is to determine the tasks that the project requires and the order in which they must be completed. The order may be easy to record for some tasks (e.g. When building a house, the land must be graded before the foundation can be laid) while difficult for others (There are two areas that need to be graded, but there are only enough bulldozers to do one). Additionally, the time estimates usually reflect the normal, non-rushed time. Many times, the time required to execute the task can be reduced for an additional cost or a reduction in the quality.

In the following example there are seven tasks, labeled A through G. Some tasks can be done concurrently (A and B) while others cannot be done until their predecessor task is complete (C cannot begin until A is complete). Additionally, each task has three time estimates: the optimistic time estimate (O), the most likely or normal time estimate (M), and the pessimistic time estimate (P). The expected time (T_E) is computed using the formula (O + 4M + P) ÷ 6.

Activity	Predecessor	Time estimates			Expected time
		Opt. (O)	Normal (M)	Pess. (P)	
A	—	2	4	6	4.00
B	—	3	5	9	5.33
C	A	4	5	7	5.17
D	A	4	6	10	6.33
E	B, C	4	5	7	5.17
F	D	3	4	8	4.50
G	E	3	5	8	5.17

Advantages

- PERT chart explicitly defines and makes visible dependencies (precedence relationships) between the WBS elements
- PERT facilitates identification of the critical path and makes this visible
- PERT facilitates identification of early start, late start, and slack for each activity,
- PERT provides for potentially reduced project duration due to better understanding of dependencies leading to improved overlapping of activities and tasks where feasible.
- The large amount of project data can be organized & presented in diagram for use in decision making.

Disadvantages

- There can be potentially hundreds or thousands of activities and individual dependency relationships
- The network charts tend to be large and unwieldy requiring several pages to print and requiring special size paper
- The lack of a timeframe on most PERT/CPM charts makes it harder to show status although colors can help (e.g., specific color for completed nodes)
- When the PERT/CPM charts become unwieldy, they are no longer used to manage the project.

During project execution, however, a real-life project will never execute exactly as it was planned due to uncertainty. It can be ambiguity resulting from subjective estimates that are prone to human errors or it can be variability arising from unexpected events or risks. The main reason that the Project Evaluation and Review Technique (PERT) may provide inaccurate information about the project completion time is due to this schedule uncertainty. This inaccuracy is large enough to render such estimates as not helpful.

One possibility to maximize solution robustness is to include safety in the baseline schedule in order to absorb the anticipated disruptions. This is called proactive scheduling. A pure proactive scheduling is a utopia; incorporating safety in a baseline schedule that allows to cope with every possible disruption would lead to a baseline schedule with a very large make-span. A second approach, reactive

scheduling, consists of defining a procedure to react to disruptions that cannot be absorbed by the baseline schedule.

Software Requirement Specifications (SRS)

Developing a **virtual classroom system** to promote a greater count of students to splurge into the field of Education. It integrates the benefits of a physical classroom with the convenience of a 'no-physical-bar' virtual learning environment, minus the commuting hazards and expenses. It will usher in the immense flexibility and sophistication in the existing learning platform structures, with the perfect blend of synchronous and asynchronous interaction. It provides a means of collaborative learning for the students.

If you're a teacher or student, you probably know that Virtual Class Room designed to help professors and instructors create and teach courses online or use online technology to help run classes. In educational software circles, it's also often called an e-learning system, a learning management system, or a virtual learning environment.

There are basically 4 types of users:

- ✓ Student
- ✓ Faculty
- ✓ Management
- ✓ Admin

Functionality

Following are some functional requirements of VCS

- Students can choose courses, attend lectures, take exams, view their attendance records, progress reports etc as per their convenience.
- Registration for multiple courses.
- Attend lectures either at the scheduled time or on request view lecture at a later time.
- Faculties can take lectures, upload assignments, announcements, evaluate answer sheets and also can upload lectures and other discussions in various formats as in videos, power point presentation etc.
- Upload and Download of various assignments, college notices, student's notices, journals, videos.
- Management can take reports or class, students, faculties. Also can approve faculty registrations, exams creations approvals, upload notices, add news.
- There can be forums, blogs, polls etc to discuss various queries and to put up suggestions posted both by students and teachers.
- Administrator can generate reports, log files, backup/recovery of data at any time.
- Shared documents and media library that can help in active learning of a student.

- Users must have valid User ID and password to login thus creating their individual profiles.
- Students can take up various quizzes which can help them to realize their inbuilt talents in various fields.
- Students also can have fun at VCS by playing online games like Sudoku, logical games etc developed in Java

Non functional requirements

Secure access of confidential data (user details) SSL can be used.
24 x 7 availability
Better component design to get better performance at peak time
Flexible service based architecture will be highly desirable for further extension.

Product description

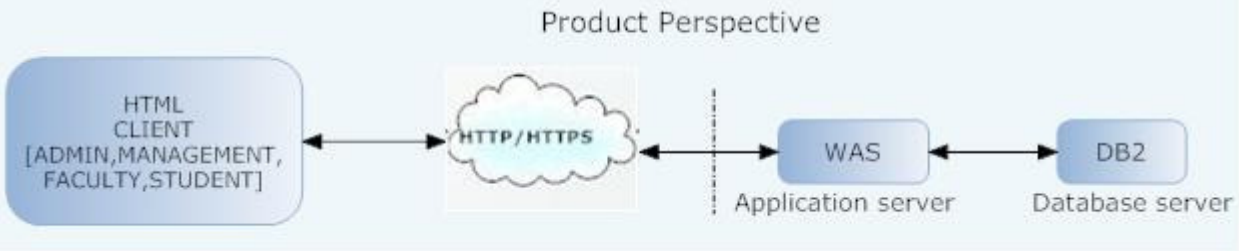


Figure 2.2 : Product perspective

- The web pages (XHTML/JSP) are present to provide the user interface on customer client side. Communication between customer and server is provided through HTTP/HTTPS protocols.
- The Client Software is to provide the user interface on system user client side and for this TCP/IP protocols are used.
- On the server side web server is for EJB and database server is for storing the information.

Software Interface

Client: Web Browser, Operating System (any)
Web Server: WAS, Operating System (any)
Data Base Server: DB2/My SQL, Operating System (any)
Development End: Eclipse 3.5

Hardware Interface

CLIENT SIDE

	PROCESSOR	**RAM**	**DISK SPACE**
INTERNET EXPLORER 6 AND ABOVE or Firefox	PENTIUM III 1 GHz AND ABOVE	256 MB	1 GB

SERVER SIDE

WEB SPHERE APPLICATION SERVER V5.0/Tomcat	PENTIUM III at 1 GHz	512 MB	2 GB
DB2 V9.1/My SQL	PENTIUM III at 1 GHz	512 MB	1GB(Excluding data size)

Communication interface:

- Client on internet will be using HTTP/HTTPS protocol
- Client on intranet will be using TCP/IP protocol.

Data Flow Diagram (DFD)

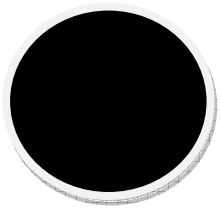
Data flow diagram:

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

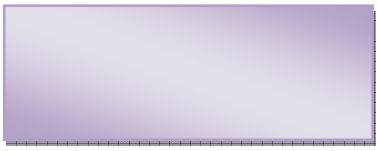
DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.

2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.



4. **Data Store:** Here data are stored or referenced by a process in the System.

Context Level DFD

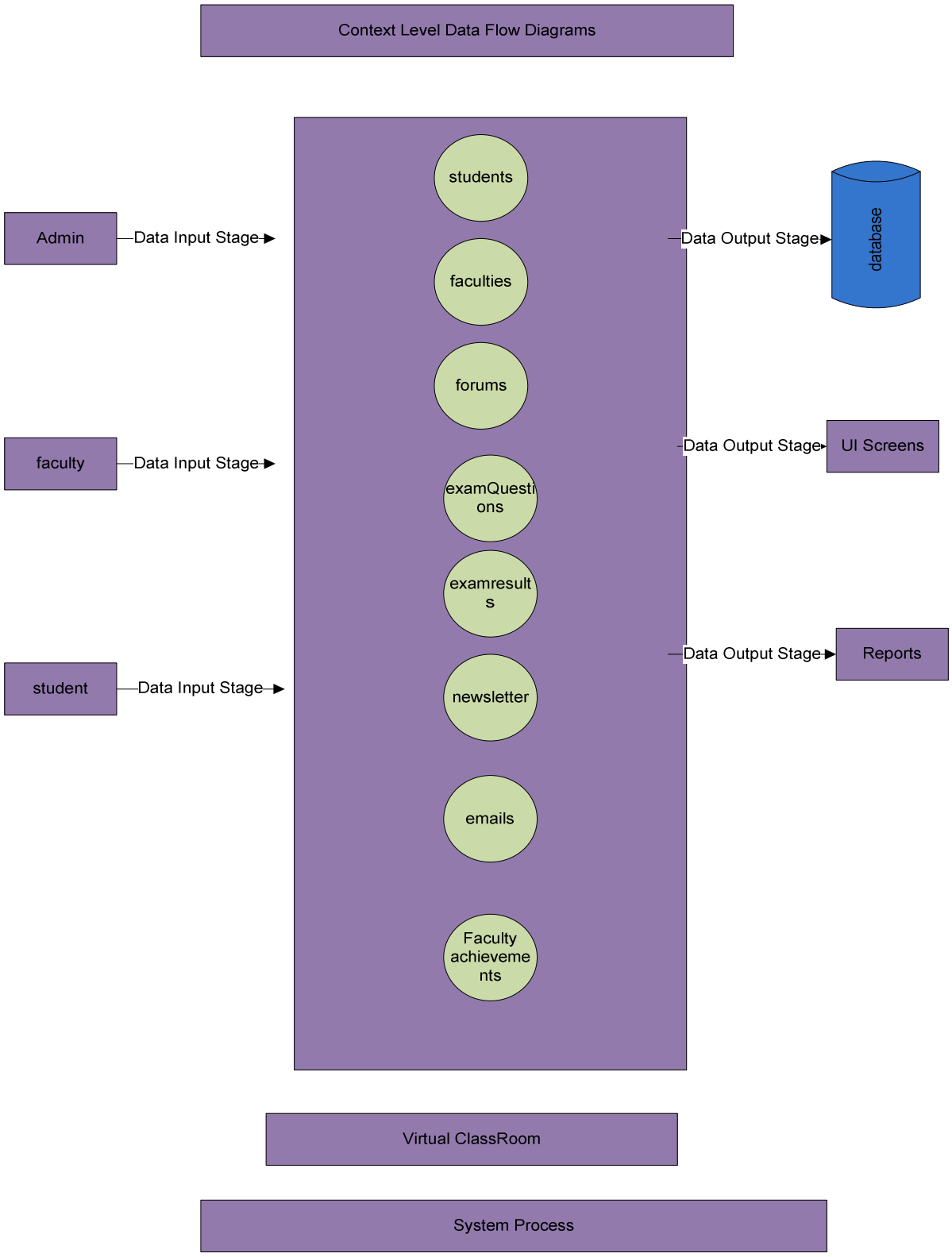


Figure 2.3: Context Level DFD

Level - 0 DFD:

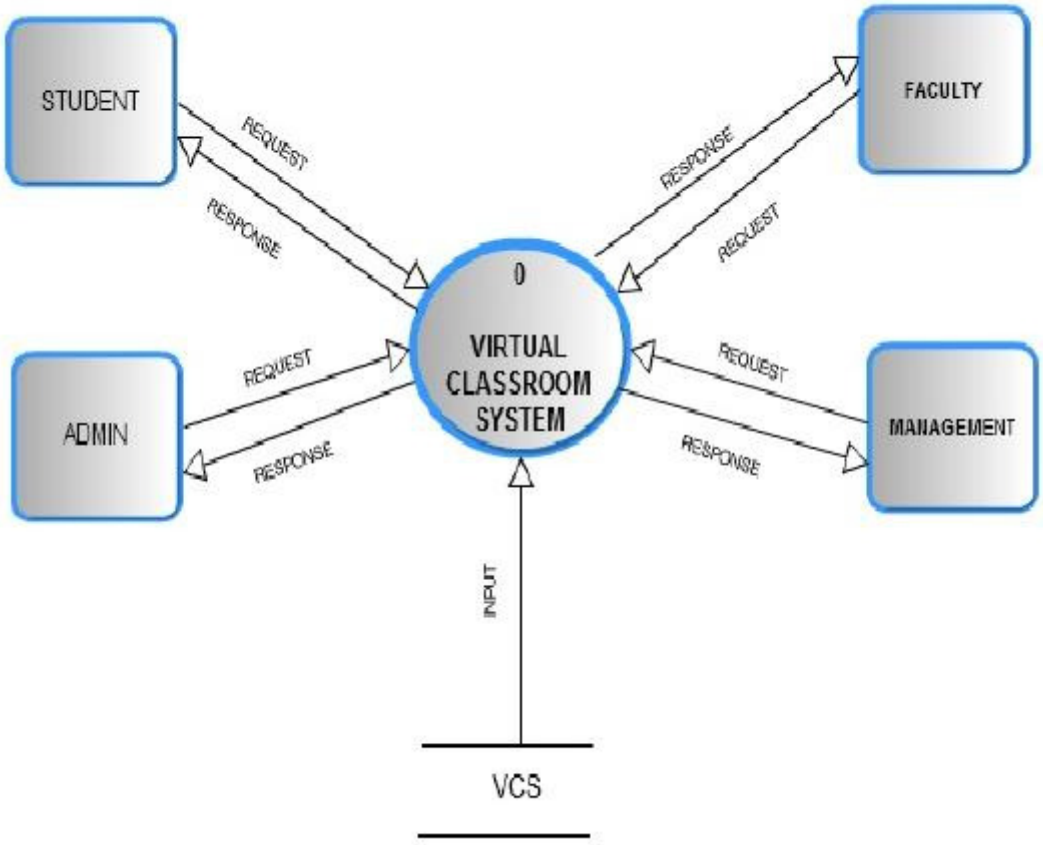


Figure 2.4: Level - 0 DFD

Level – 1 DFD:

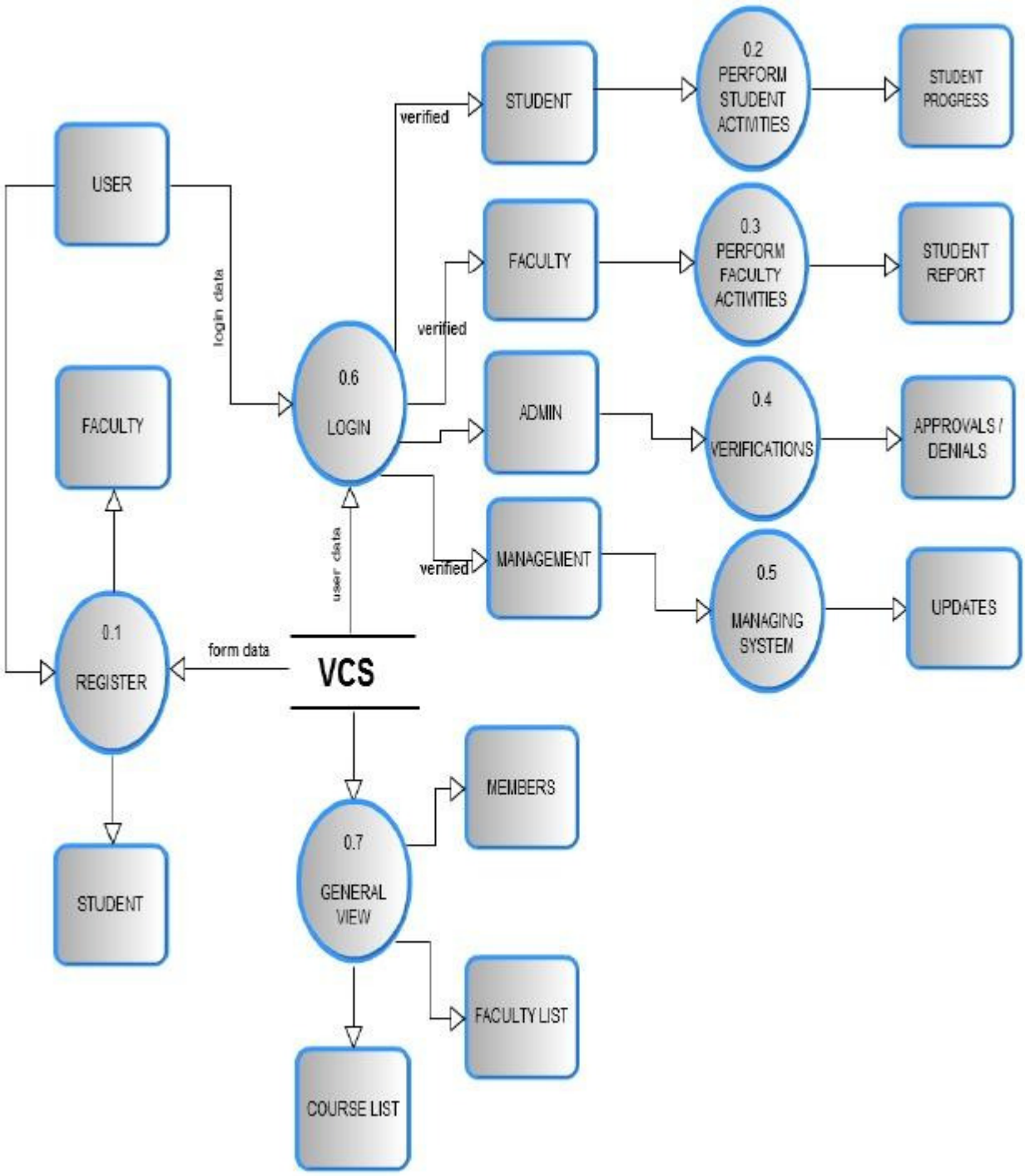


Figure 2.5: Level 1 DFD

Level – 2 DFD:

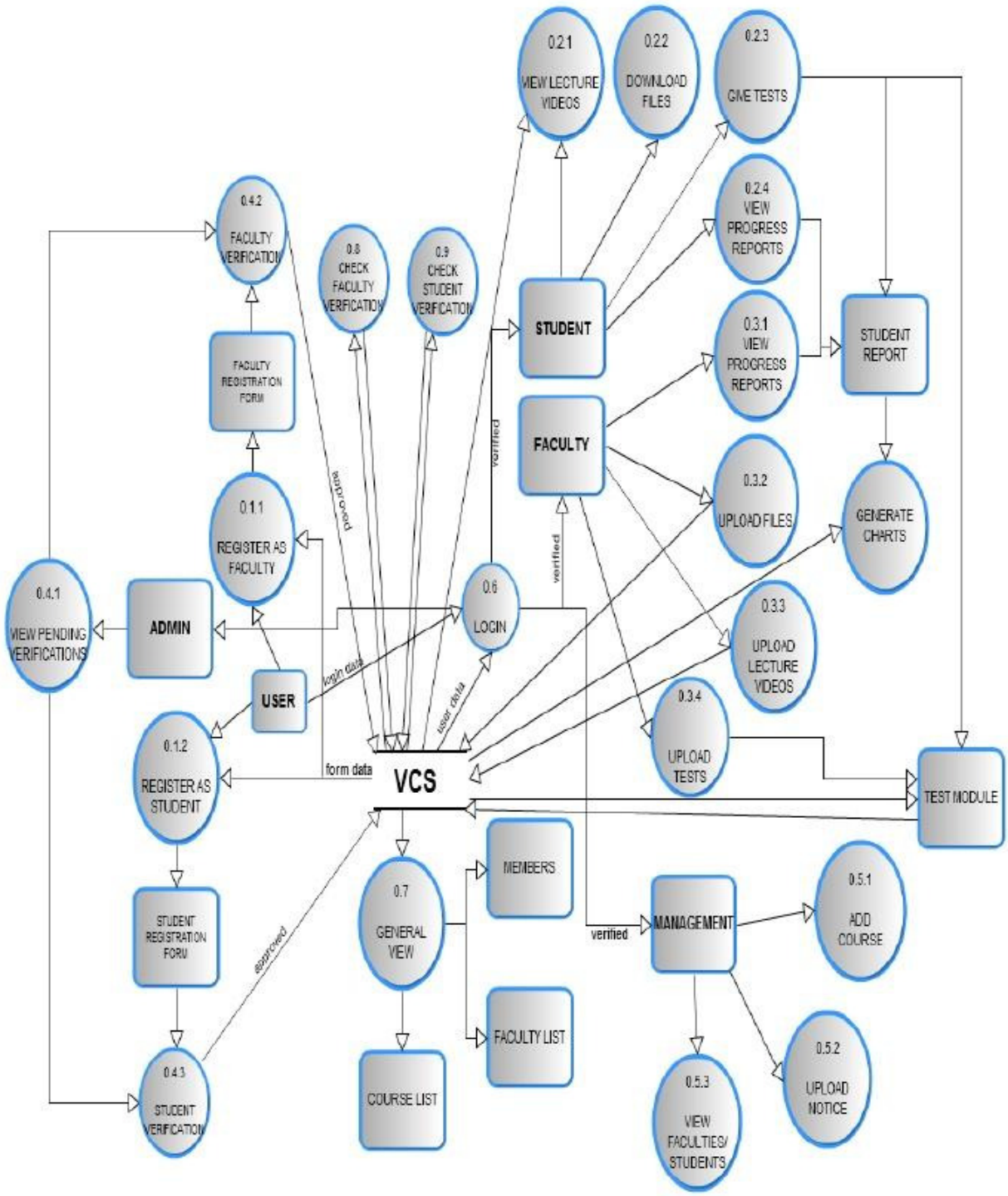


Figure 2.6: Level 2 DFD

UML Diagrams

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

User Model View:

i. This view represents the system from the users perspective.
ii. The analysis representation describes a usage scenario from the end-users perspective.

Structural model view:

i. In this model the data and functionality are arrived from inside the system.
ii. This model view models the static structures.

Behavioral Model View:
It represents the dynamic of behavioral 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 behavioral as parts of the system are represented as they are to be built.

Environmental Model View:
In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

i. UML Analysis modeling, this focuses on the user model and structural model views of the system.
ii. UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, Counselor, Student …etc., or another system like central database.

Use Case diagrams

Course Registration Use case

If a student wants an admission in the VCS he/she has to fill in the details in the student registration form then he/she has to select the courses from the available one's where they can choose their subjects only the optional one's and then he/she can become the authorized student of the school after making payment for the course. But only after the use has been verified by the Admin after making payment. Once registered user can avail all the facilities of VCS.

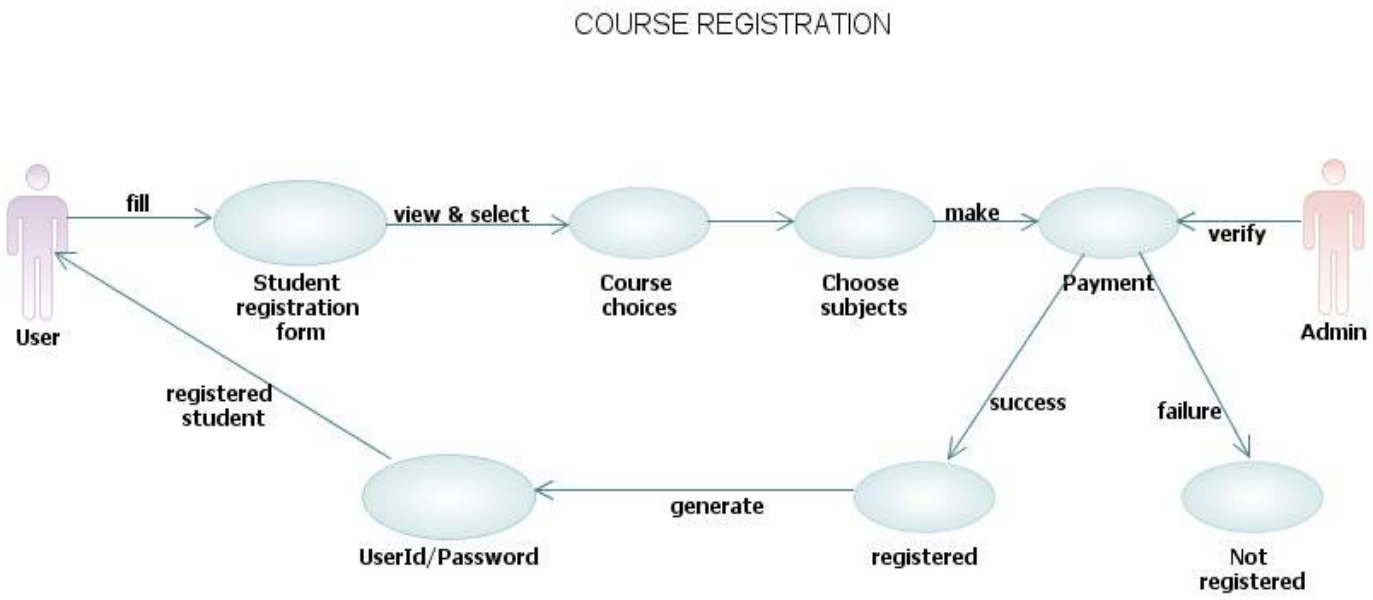


Figure 2.7: Course Registration Use case

Faculty Registration use case

A User can register as faculty in the VCS only after filling in the faculty registration form. If the user has the required qualifications then after the detailed verification of all his/her details by the Management, he/she becomes the authorized faculty of VCS and can teach students.

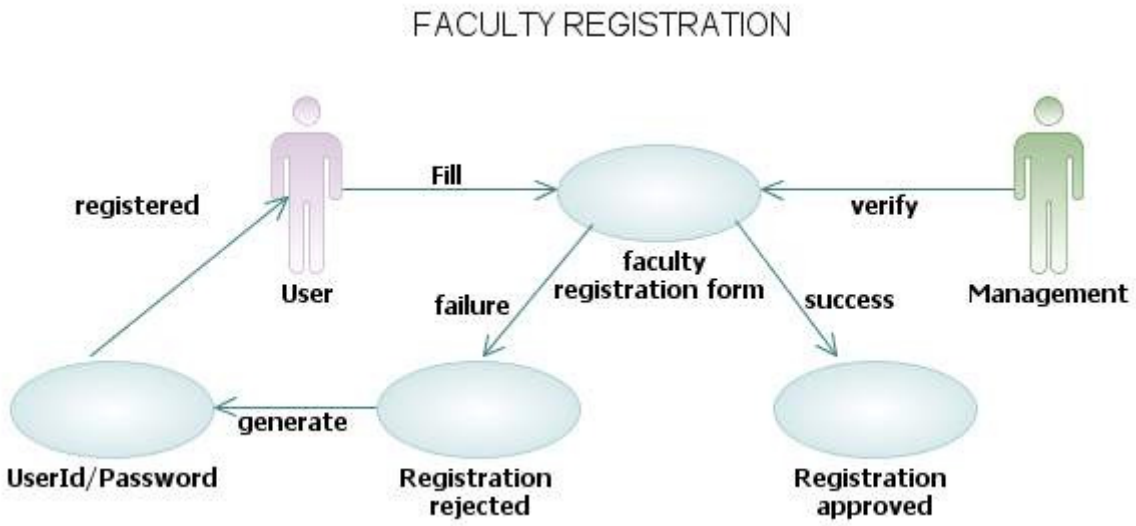


Figure 2.8: Faculty Registration use case

File Management use case

A User (admin/management/faculty/student) can upload a file that may be an assignment, a video, a presentation or any other file. All four types of the users (admin/management/faculty/student) have the privilege of viewing the uploaded files. Uploaded file can only be deleted by the user who uploaded it.

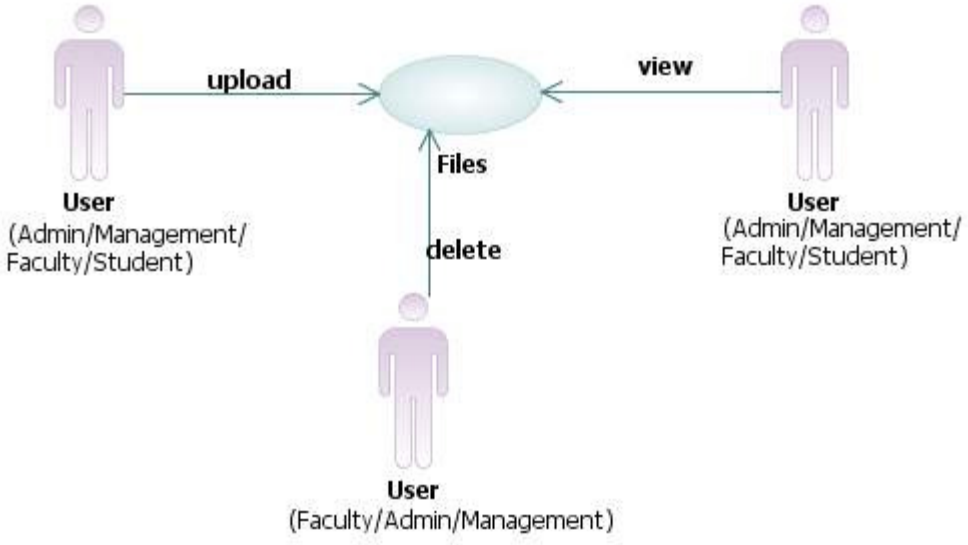


Figure 2.9: File Management use case

Notices Management use case

A User (admin/management/faculty/student) can upload a notice. All four types of the users (admin/management/faculty/student) have the privilege of viewing the uploaded notices. An uploaded notice can only be deleted by the user who had uploaded it.

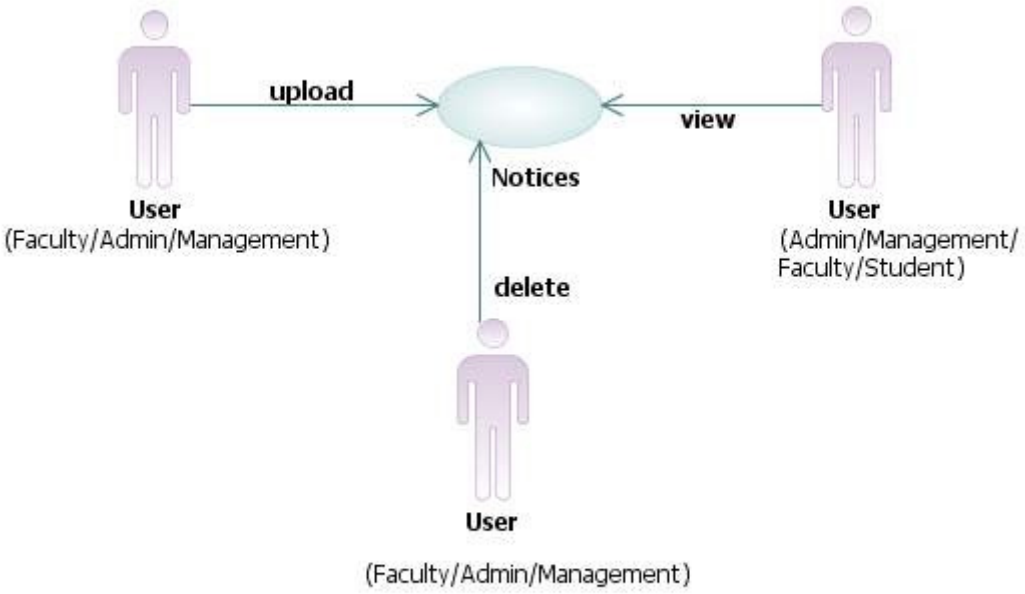


Figure 2.10: Notices Management use case

Discussion Management Use case

Faculty will upload a schedule of the Discussion Time for a particular subject/topic.Students can view the schedule and according to the schedule, faculty will organize the Discussion Time on a particular date and students will attend the Discussion.Here students can put up queries to the faculty. Discussion may take place among students too.

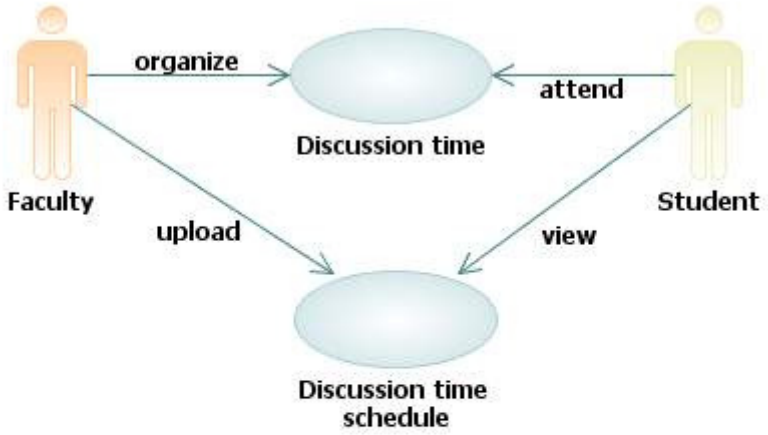


Figure 2.11: Discussion Management Use case

Tests Management use case

Faculty will upload the Test Paper of a particular subject/topic on a particular date. Student will take the test and submit the answer sheet. Faculty will then evaluate the answer sheet of the student and will generate his/her progress report on the basis of the marks obtained by the student. Student will then be able to view his/her progress report.

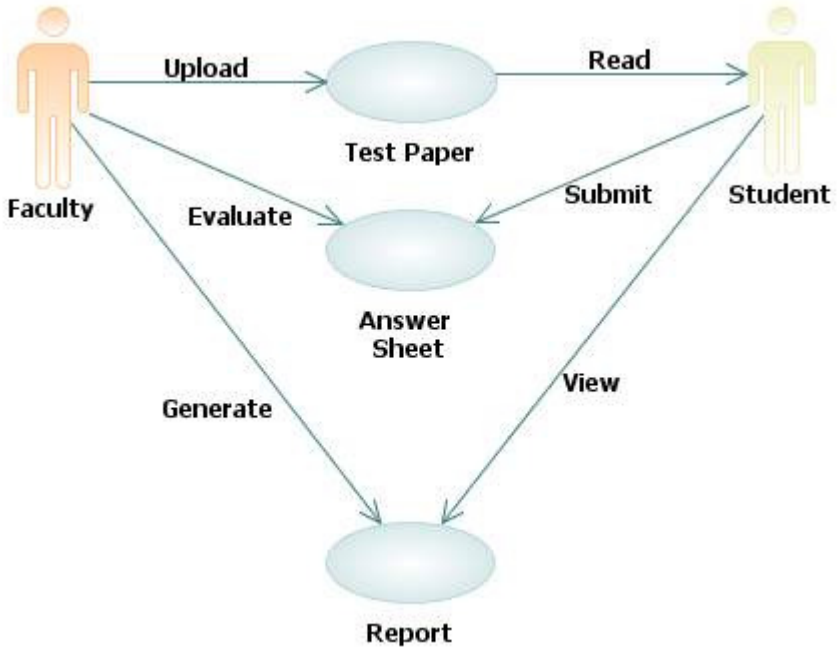


Figure 2.12: Tests Management use case

Admin Use case

The admin has the following controls:

- Manage Emails
- Manage Financial Transactions
- View Requests
- View Activity User Records
- View Crystal Reports
- Update/Delete Notices
- Update/Delete Files
- View Notices
- View Files
- Take backup of database
- Generate Reports

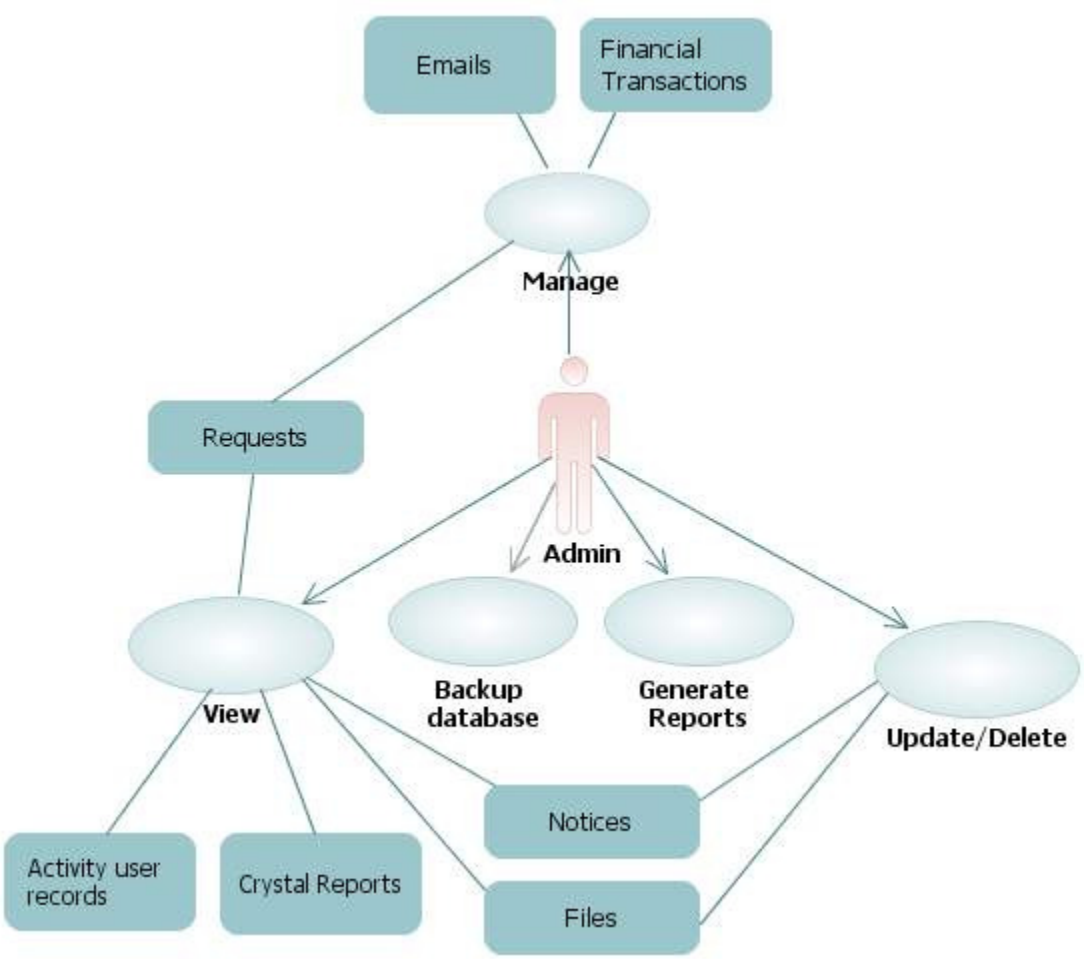


Figure 2.13: Admin Use case

Sequence Diagrams

Login Sequence diagram:

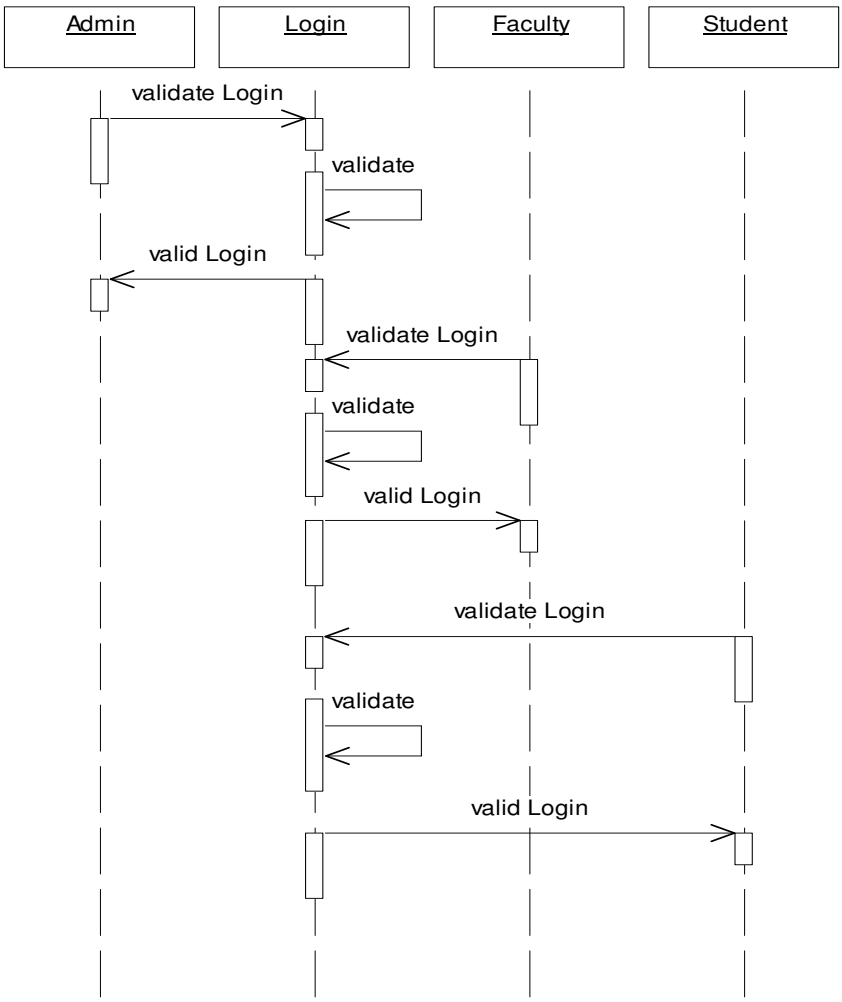


Figure 2.14: Login Sequence diagram

Student Registration Sequence Diagram :

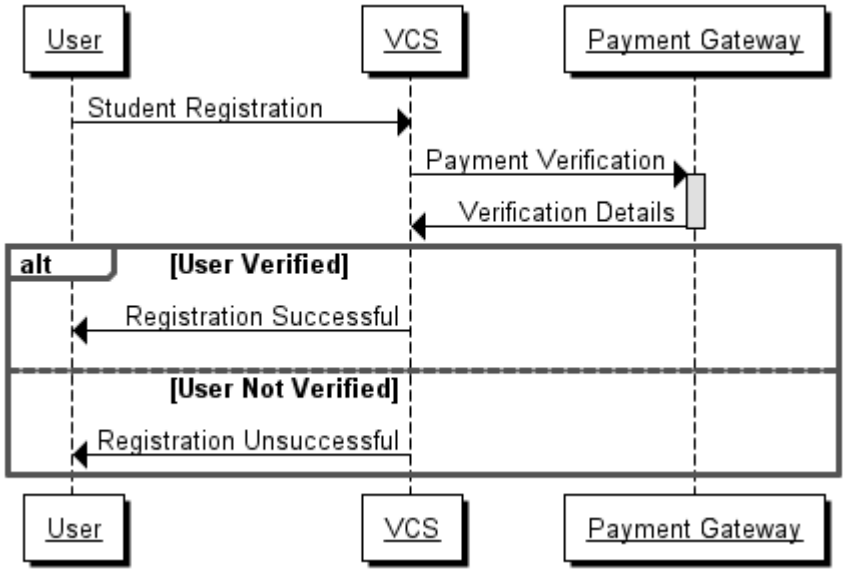


Figure 2.15: Student Registration Sequence diagram

Faculty Registration Sequence Diagram:

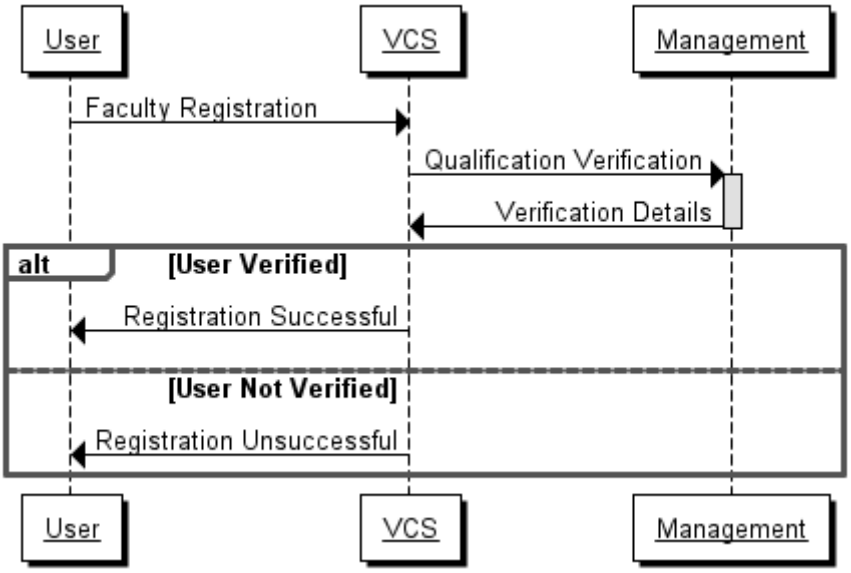


Figure 2.16: Faculty Registration Sequence diagram

Notice Management Sequence Diagram:

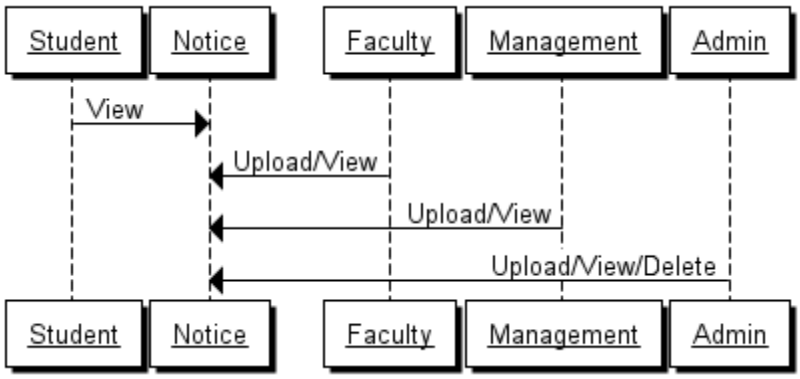


Figure 2.17: Notice Management Sequence Diagram

Files Management Sequence Diagram (Assignments):

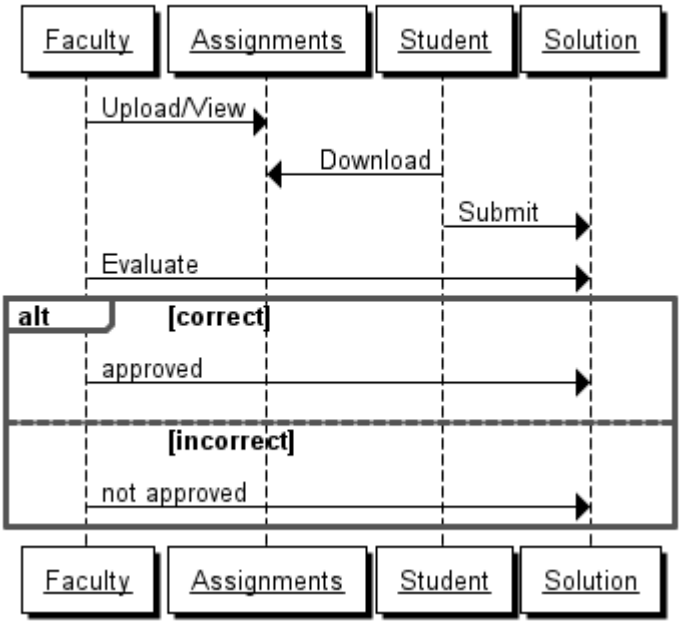


Figure 2.18: Files Management Sequence Diagram

Files Management (Lectures):

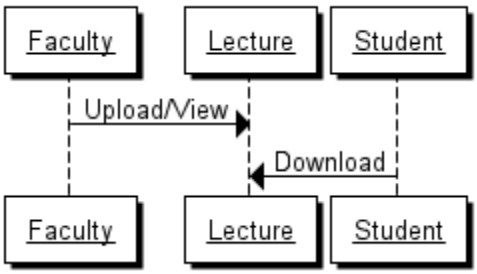


Figure 2.19: Files Management (Lectures)

Files Management (General):

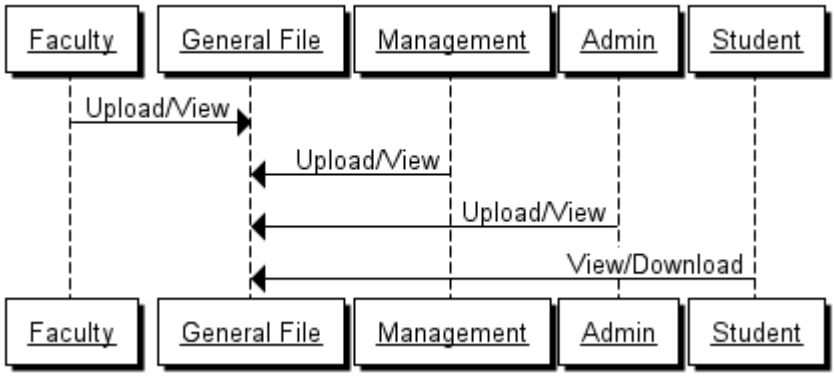


Figure 2.20: Files Management (General)

Tests:

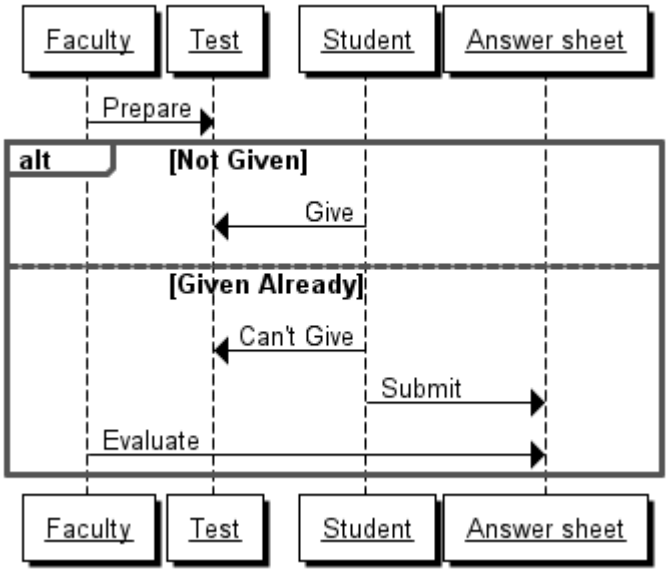


Figure 2.21: Tests

Adding Courses:

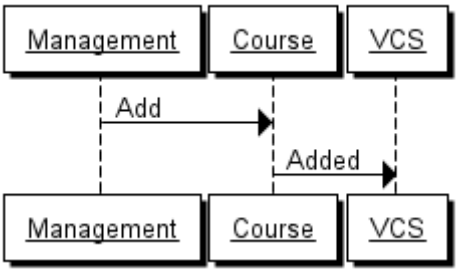


Figure 2.22: Adding Courses

Discussion:

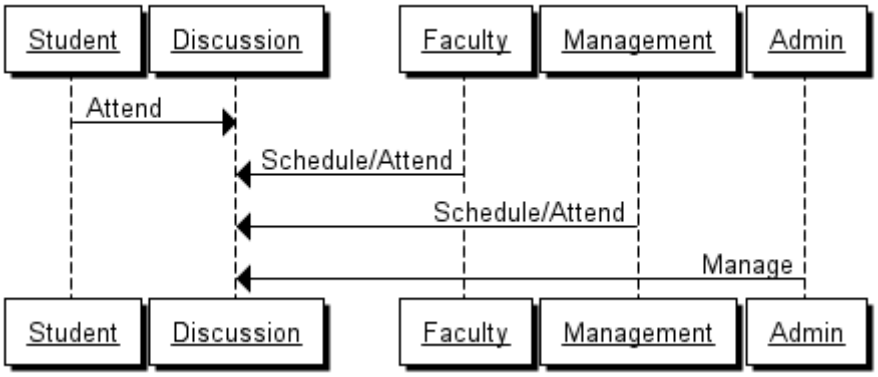


Figure 2.23: Discussion

Report:

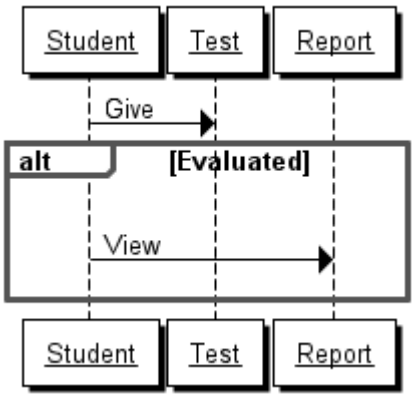


Figure 2.24: Report

Syllabus:

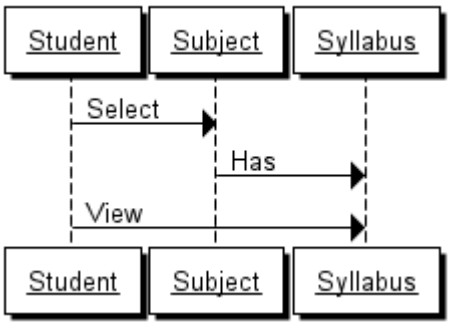


Figure 2.25: Syllabus

Profile Management:

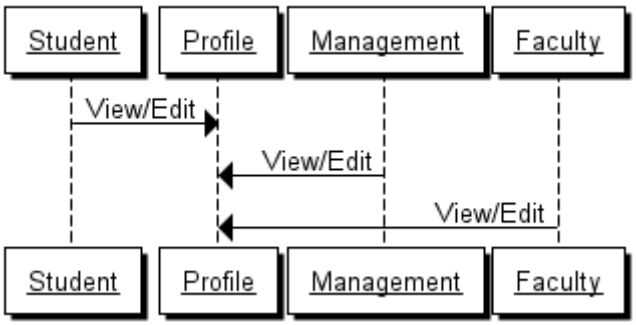


Figure 2.26: Profile Management

Admin Sequence diagram:

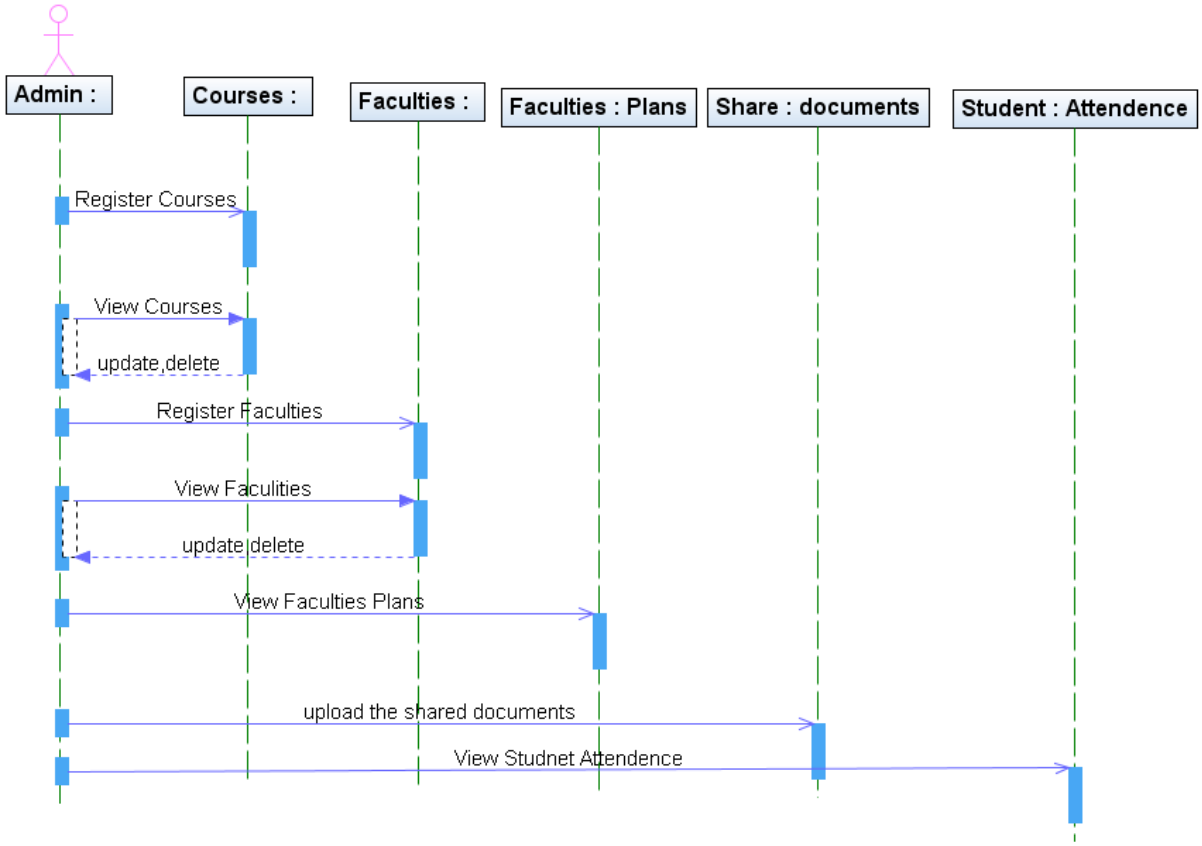


Figure 2.27: Admin Sequence diagram

Collaboration Diagram

Login:

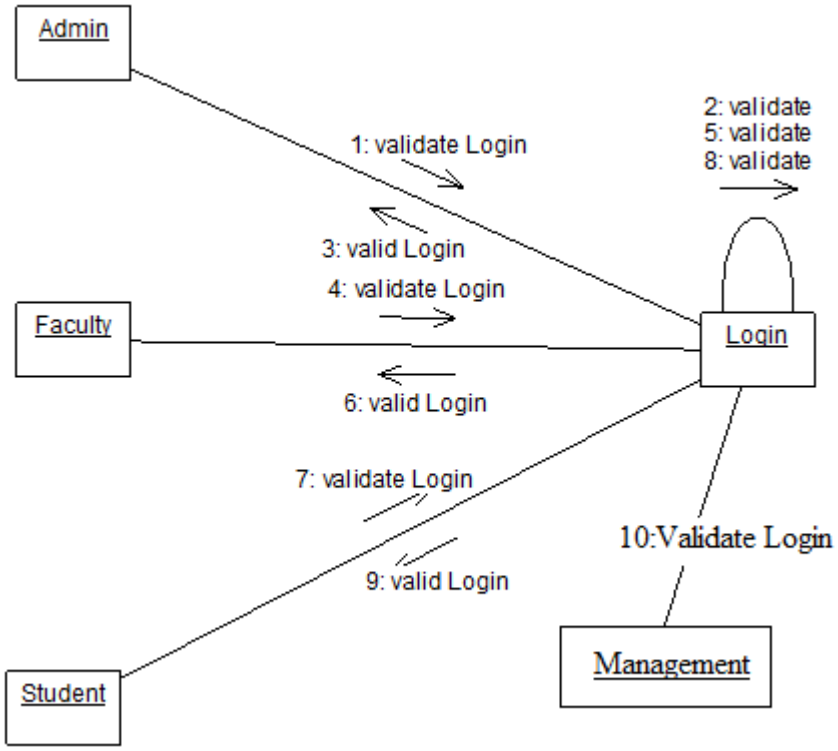


Figure 2.28: Login

Adding Faculties Collaboration Diagram:

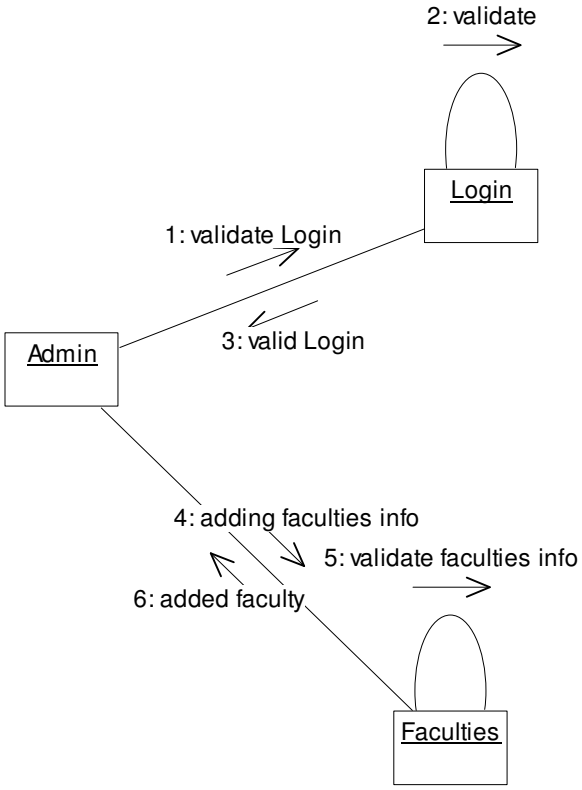


Figure 2.29: Adding Faculties Collaboration Diagram

Add/ View Collaboration Diagram:

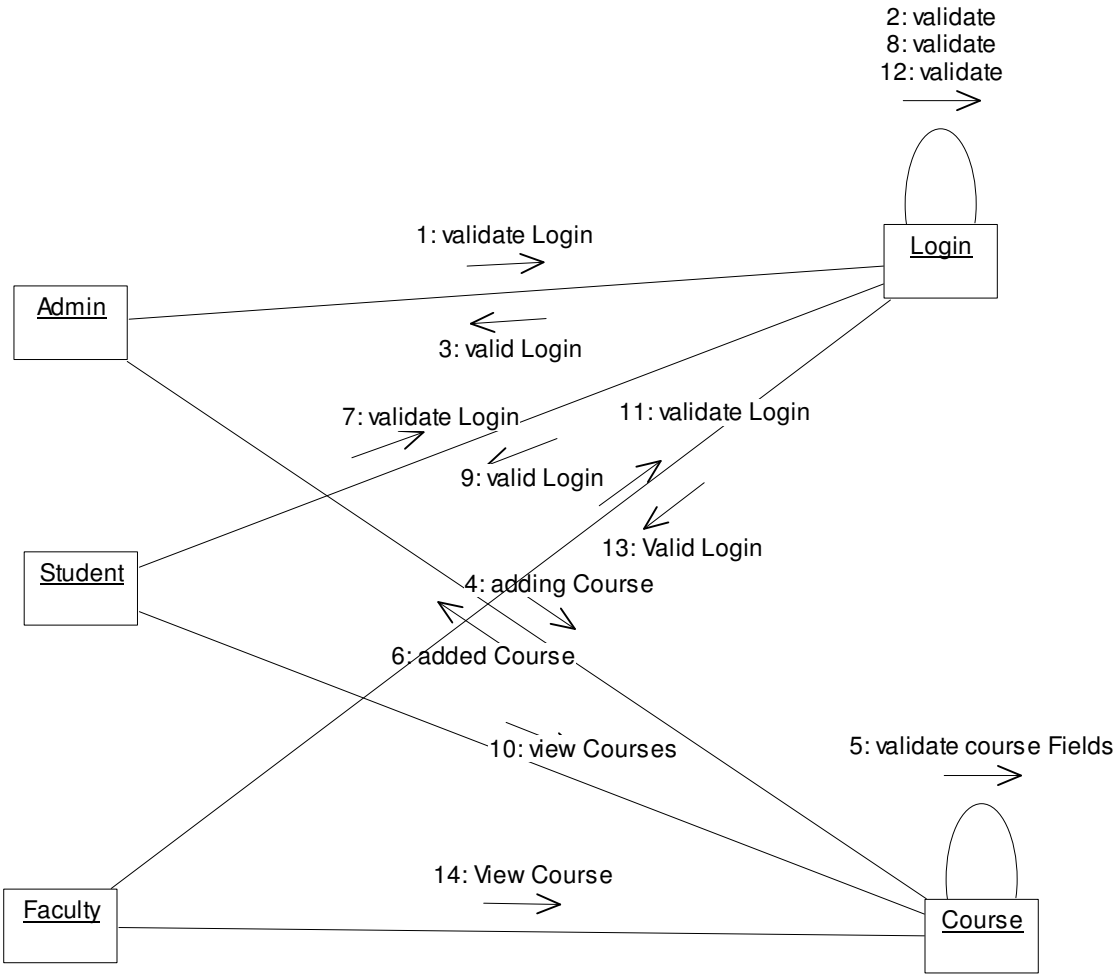


Figure 2.30: Add/ View Collaboration Diagram

Activity Diagrams

Admin Activity Diagram:

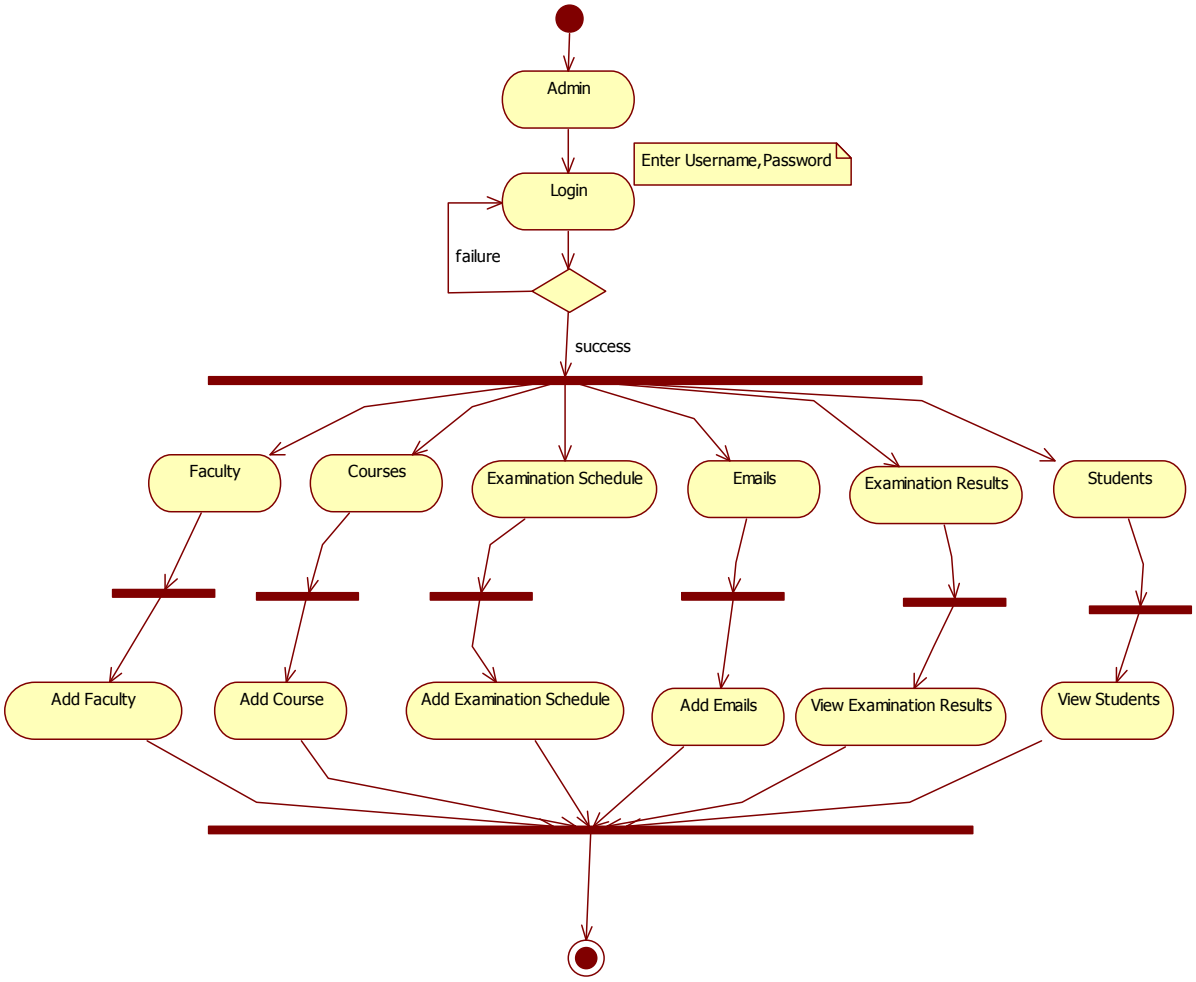


Figure 2.31: Admin Activity Diagram

Student Activity Diagram:

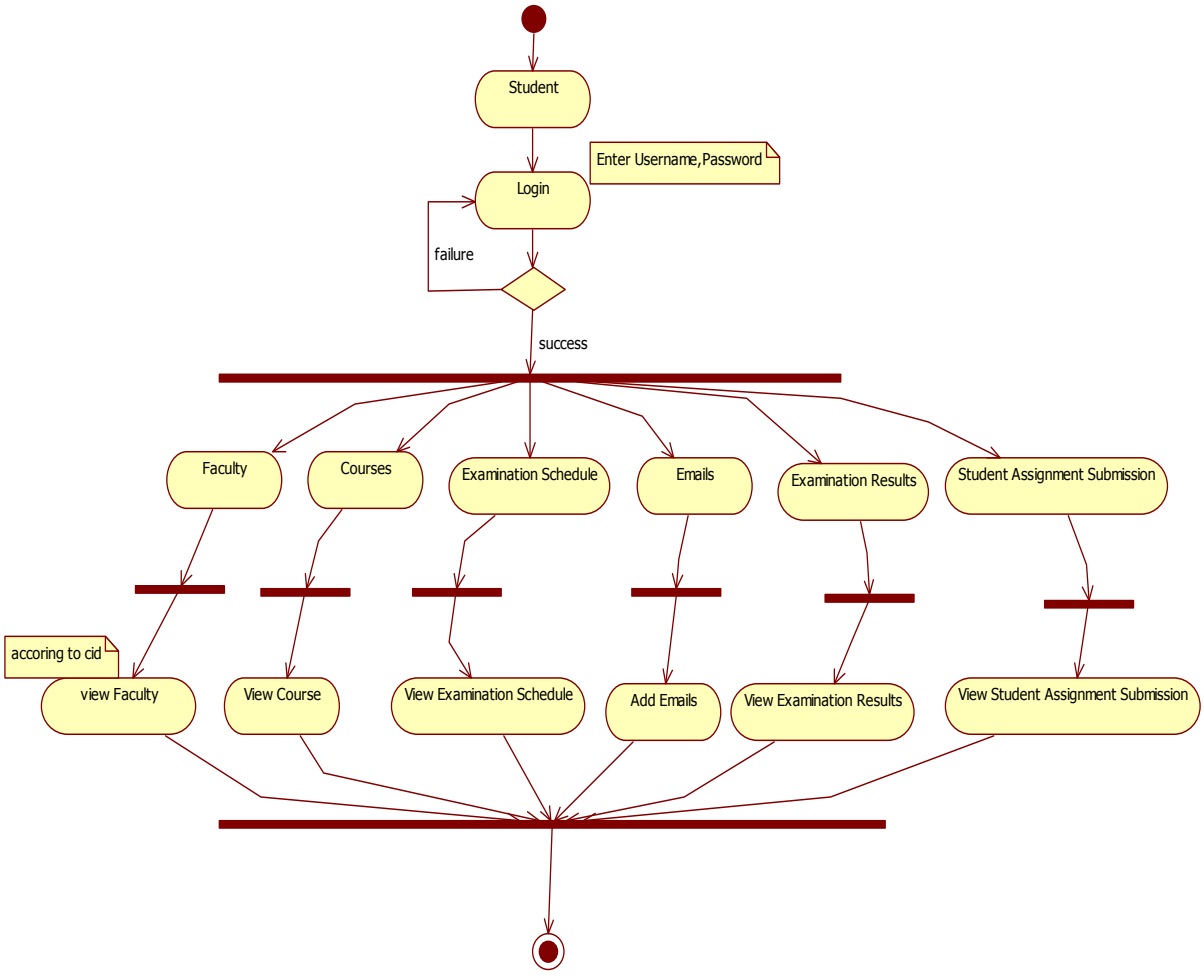


Figure 2.32: Student Activity Diagram

Faculty Activity Diagram:

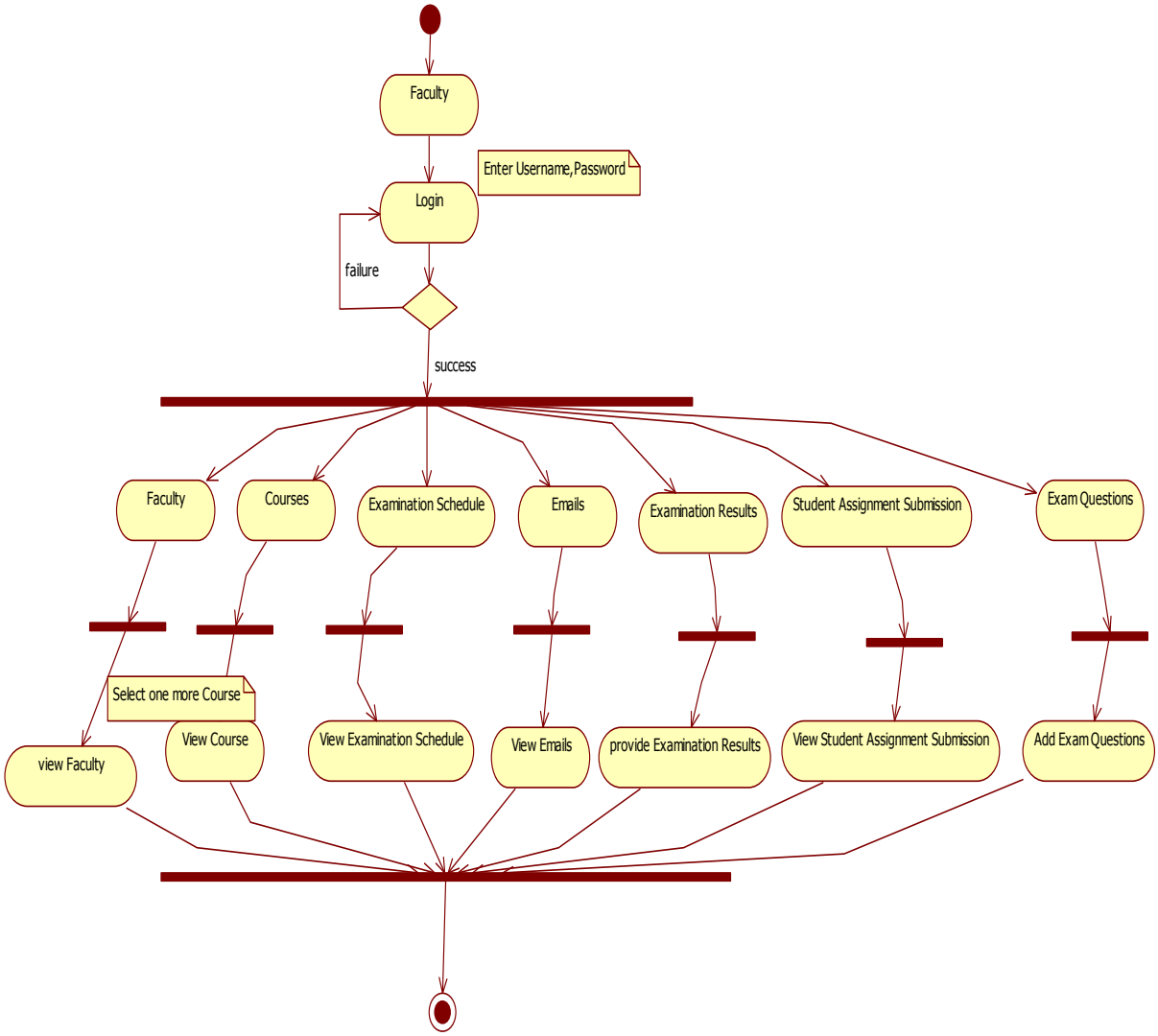


Figure 2.33: Faculty Activity Diagram

Component Diagram

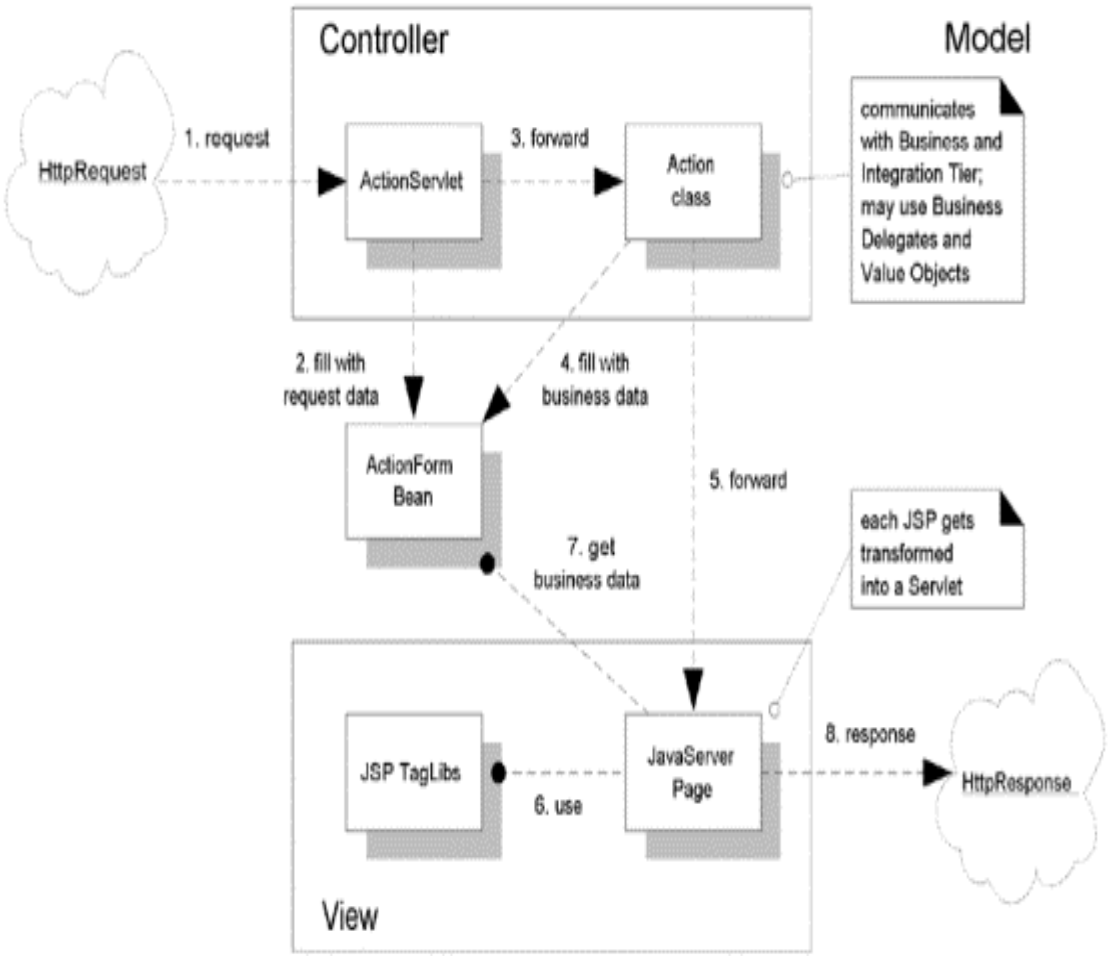


Figure 2.34: Component Diagram

Deployment Diagram

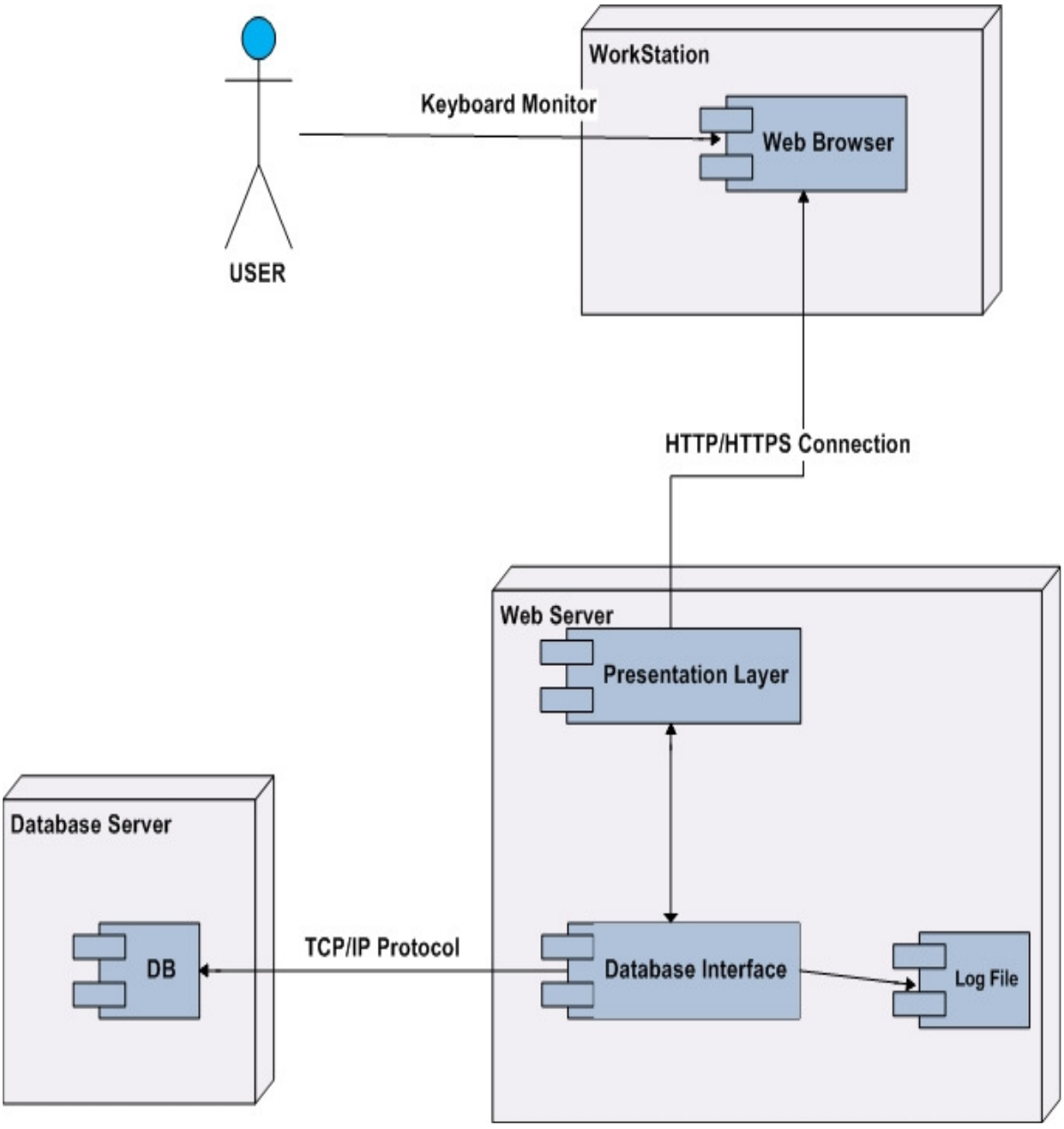


Figure 2.35: Deployment Diagram

ER Diagram

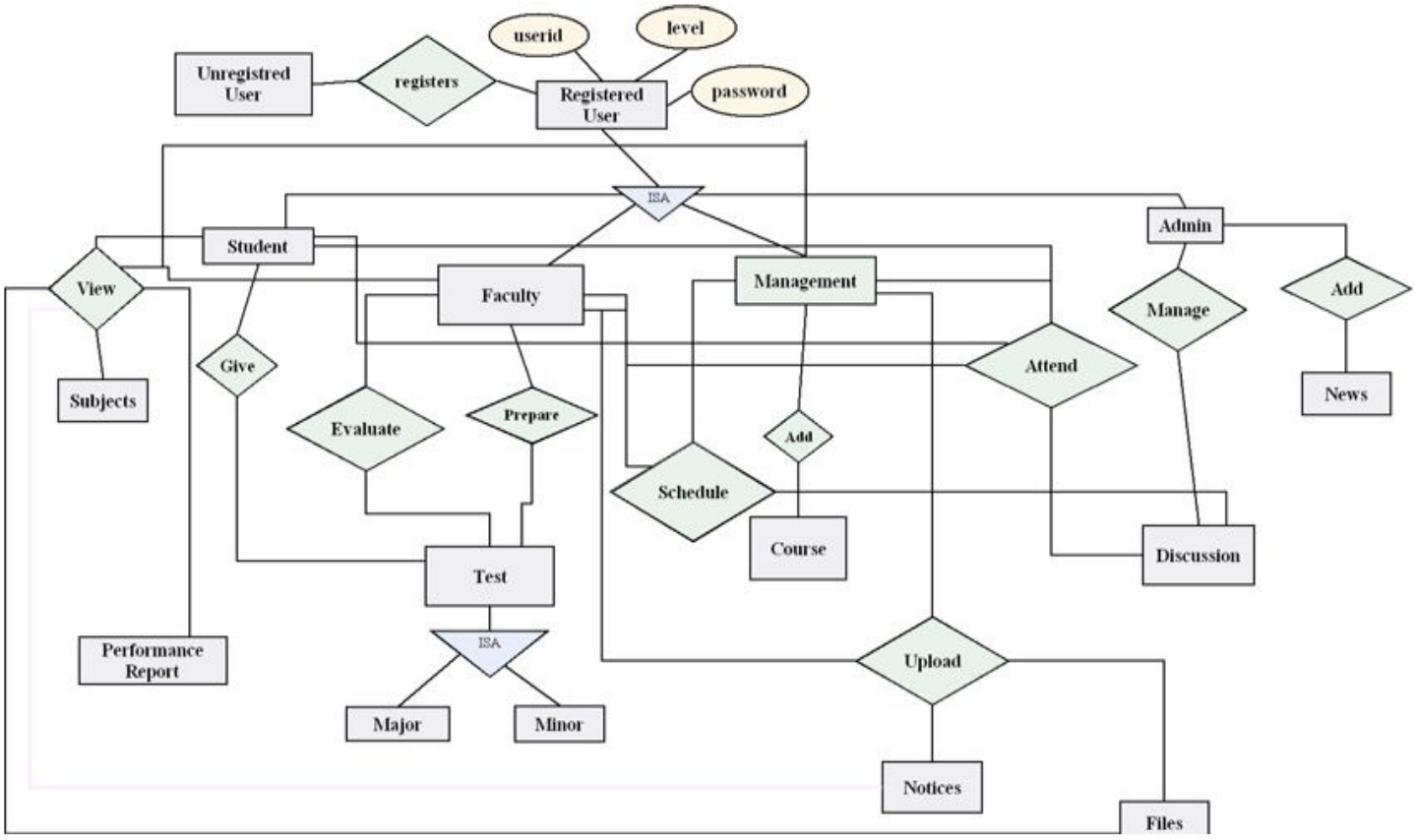


Figure 2.36: Entity Relationship Diagram of Virtual Classroom system

System Design

System Design

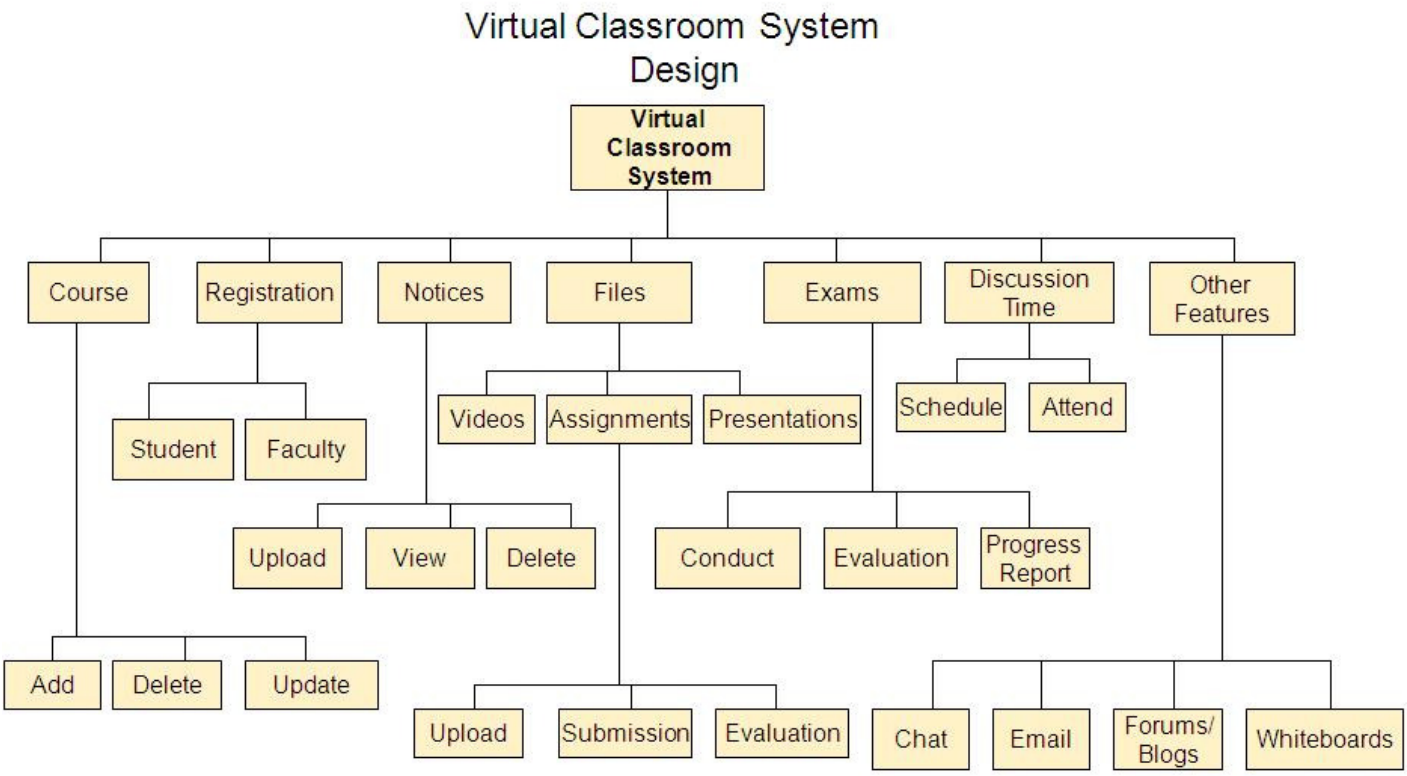


Figure 3.1: System Design

Data dictionary:

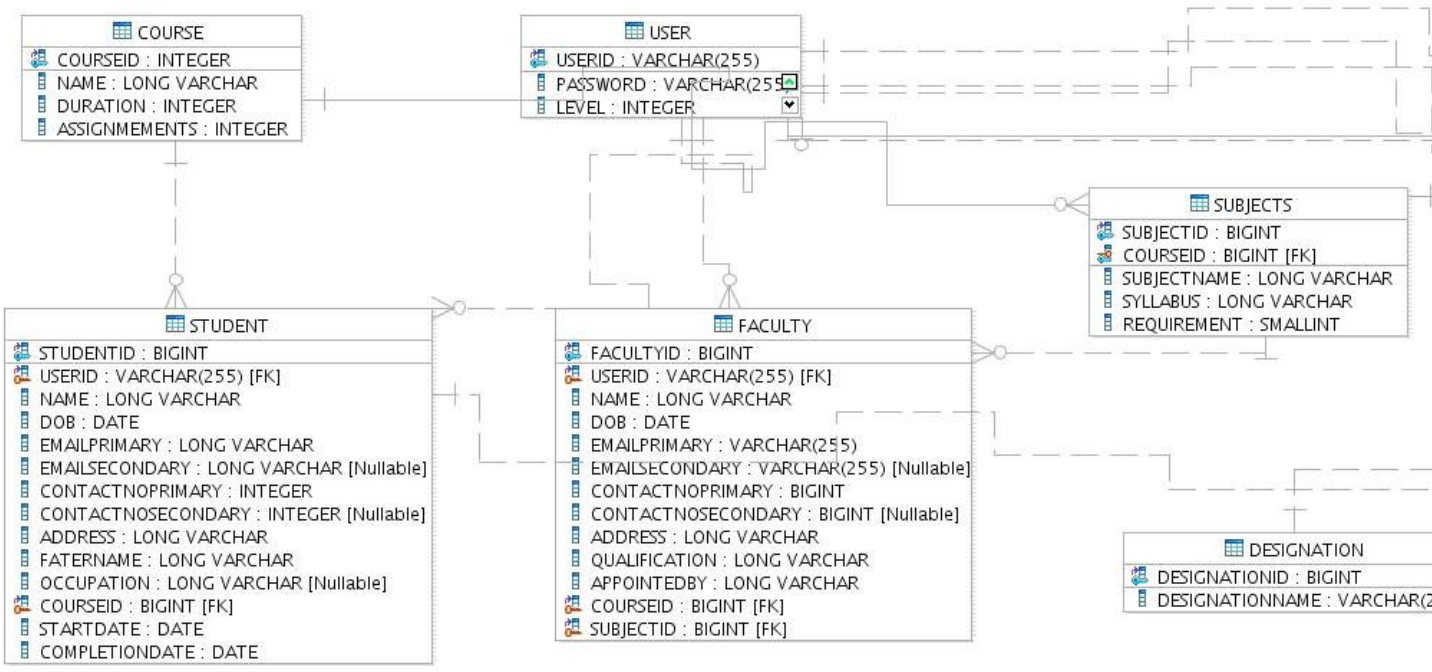


Figure 3.2: Database design

Description of each Table is as follows:

User: Contains login related details of users registered to the system.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
USERID	VARCHAR(255)	✓		BINARY		
PASSWORD	VARCHAR(255)	✓		BINARY		
LEVEL	VARCHAR(255)	✓		BINARY		
ONLINE	SMALLINT(5)	✓		UNSIGNED ZEROFILL	0	VCS_SCHEM...
Blocked	INTEGER	✓		UNSIGNED ZEROFILL	0	
ARCHIVED	VARCHAR(45)	✓		BINARY		
Image	VARCHAR(45)	✓		BINARY		

Figure 3.1: User Table

Student: Contains details of all Students registered to the system.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
STUDENTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
USERID	VARCHAR(255)	✓		☐ BINARY		
NAME	VARCHAR(255)	✓		☐ BINARY		
DOB	DATETIME	✓				
EMAILPRIMARY	VARCHAR(255)	✓		☐ BINARY		
EMAILSECONDARY	VARCHAR(255)	✓		☐ BINARY		
CONTACTNOPRIMARY	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
CONTACTNOSECON...	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
ADDRESS	VARCHAR(500)	✓		☐ BINARY		
OCCUPATION	VARCHAR(45)	✓		☐ BINARY		
COURSEID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
FATHERNAME	VARCHAR(255)	✓		☐ BINARY		
COMPLETIONDATE	DATETIME	✓				
PAYMENTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

Figure 3.2: Student Table

Faculty: Contains details of all faculties registered to the system.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
FACULTYID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
USERID	VARCHAR(45)	✓		☐ BINARY		
DOB	DATETIME	✓				
EMAILPRIMARY	VARCHAR(255)	✓		☐ BINARY		
EMAILSECONDARY	VARCHAR(255)	✓		☐ BINARY		
CONTACTNOPRIMARY	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
CONTACTNOSECON...	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
ADDRESS	VARCHAR(500)	✓		☐ BINARY		
QUALIFICATION	VARCHAR(255)	✓		☐ BINARY		
APPOINTEDBY	VARCHAR(255)	✓		☐ BINARY		
COURSEID	VARCHAR(255)	✓		☐ BINARY		
SUBJECTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
NAME	VARCHAR(255)	✓		☐ BINARY		

Figure 3.3: Faculty Table

Management: Contains details of all Management staff registered to the system.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
MANAGEMENTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
USERID	VARCHAR(255)	✓		☐ BINARY		
NAME	VARCHAR(500)	✓		☐ BINARY		
DOB	DATETIME	✓				
EMAILP	VARCHAR(255)	✓		☐ BINARY		
EMAILS	VARCHAR(255)	✓		☐ BINARY		
CONTACTP	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
CONTACTS	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
ADDRESS	VARCHAR(255)	✓		☐ BINARY		
PHOTOGRAPH	VARCHAR(255)	✓		☐ BINARY		
DESIGNATION	VARCHAR(255)	✓		☐ BINARY		

Figure 3.4: Management Table

Discussions: Contains discussions of faculties and students:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
DISCUSSIONID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
ORGANIZEDBY	VARCHAR(255)	✓		☐ BINARY		
ORGANIZEDFOR	VARCHAR(20)	✓		☐ BINARY		
COURSEID	VARCHAR(30)	✓		☐ BINARY		
SUBJECTID	VARCHAR(30)	✓		☐ BINARY		
TITLE	VARCHAR(255)	✓		☐ BINARY		
DATE	DATETIME	✓				
DURATION	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

Figure 3.5: Discussions Table

Course: Contains details of all courses.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
COURSEID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
NAME	VARCHAR(500)	✓		☐ BINARY		
DURATION	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
FEES	VARCHAR(255)	✓		☐ BINARY		

Figure 3.6: Course Table

Subjects: Contains all subject details related to all courses.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
SUBJECTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
SUBJECTNA...	VARCHAR(500)	✓		☐ BINARY		

Figure 3.7: Subjects Table

Files: Contains details of all files uploaded on the system.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
FILEID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
TITLE	VARCHAR(50)	✓		☐ BINARY		
DESCRIPTION	VARCHAR(500)	✓		☐ BINARY		
UPLOADEDBY	VARCHAR(20)	✓		☐ BINARY		
TOPICS	VARCHAR(500)	✓		☐ BINARY		
FILETYPE	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
SOLUTIONFILE	VARCHAR(20)	✓		☐ BINARY		
UPLOADEDTO	VARCHAR(20)	✓		☐ BINARY		
COURSEID	VARCHAR(20)	✓		☐ BINARY		
SUBJECTID	VARCHAR(20)	✓		☐ BINARY		
UPLOADDATE	DATETIME	✓				
FilePath	VARCHAR(255)	✓		☐ BINARY		
Image	VARCHAR(45)	✓		☐ BINARY		

Figure 3.8: Files Table

Notices: Contains details of all notices posted on the system.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
NOTICEID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
KEYWORD	VARCHAR(500)	✓		☐ BINARY		
DESCRIPTION	VARCHAR(500)	✓		☐ BINARY		
POSTEDTO	VARCHAR(500)	✓		☐ BINARY		
POSTEDBY	VARCHAR(255)	✓		☐ BINARY		
POSTINGDATE	DATETIME	✓				
EXPIRESON	DATETIME	✓				
TITLE	VARCHAR(255)	✓		☐ BINARY		
ARCHIVED	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
COURSEID	VARCHAR(255)	✓		☐ BINARY		
SUBJECTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

Figure 3.9: Notices Table

Test: Contains all exams details.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
TESTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
TESTTYPE	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
TESTNAME	VARCHAR(255)	✓		☐ BINARY		
PREREQUISITES	VARCHAR(255)	✓		☐ BINARY		
SYLLABUS	VARCHAR(500)	✓		☐ BINARY		
TESTPAPER	VARCHAR(1000)	✓		☐ BINARY		
TAKENBY	VARCHAR(255)	✓		☐ BINARY		
COURSEID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
SUBJECTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
MAXMARKS	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
PASSMARKS	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
DURATION	VARCHAR(20)	✓		☐ BINARY		

Figure 3.10: Test Table

Answer Sheets: contains all the answer for student tests.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
TESTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
STUDENTID	VARCHAR(20)	✓		☐ BINARY		
ANSWERS	VARCHAR(1000)	✓		☐ BINARY		
EVALUATED	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
TOTALMARKS	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

Figure 3.11: Answer Sheets Table

Appointment: This table contains all the appointments of faculty which is maintained by the management.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
APPOINTMENTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
NAME	VARCHAR(500)	✓		☐ BINARY		
RESUME	VARCHAR(255)	✓		☐ BINARY		
PASSWORD	VARCHAR(255)	✓		☐ BINARY		
SUBJECTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
DOB	DATETIME	✓				
EMAILPRIMARY	VARCHAR(255)	✓		☐ BINARY		
EMAILSECONDARY	VARCHAR(255)	✓		☐ BINARY		
CONTACTNOPRIMARY	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
ADDRESS	VARCHAR(255)	✓		☐ BINARY		
QUALIFICATION	VARCHAR(255)	✓		☐ BINARY		
PASSYEAR	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
APPROVED	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
COURSEID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
APPROVEDBY	VARCHAR(255)	✓		☐ BINARY		
USERID	VARCHAR(255)	✓		☐ BINARY		

Figure 3.12: Appointment Table

Feed Back: This table used to store the feedback provided by students.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
FEEDBACKID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
TESTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
FILEID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
STUDENTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
FACULTYID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
FEEDBACK	VARCHAR(500)	✓		☐ BINARY		
APPROVAL	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

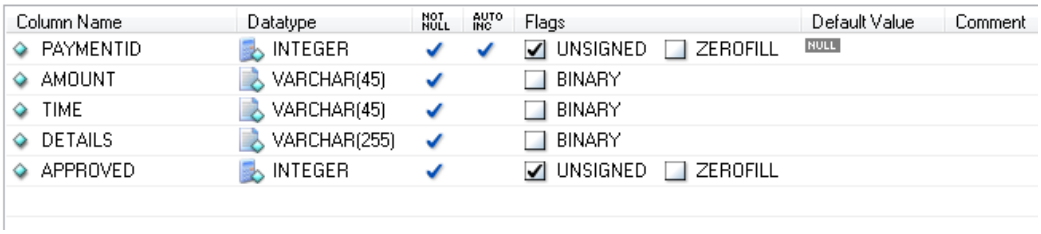
Figure 3.13: Feed Back Table

News: This table contains the news which is created by the faculty or management or admin users. The new can be any useful information for the students such as interesting forums, programming resources etc.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
NEWSID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
SUBJECT	VARCHAR(255)	✓		☐ BINARY		
DATE	DATETIME	✓				
DESCRIPTION	VARCHAR(500)	✓		☐ BINARY		

Figure 3.14: News Table

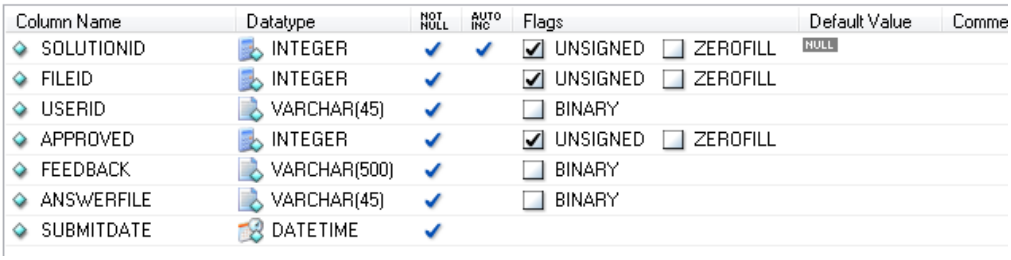
Payment: This table contains all the payment information of students who registered the course.



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
PAYMENTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
AMOUNT	VARCHAR(45)	✓		☐ BINARY		
TIME	VARCHAR(45)	✓		☐ BINARY		
DETAILS	VARCHAR(255)	✓		☐ BINARY		
APPROVED	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

Figure 3.15: Payment Table

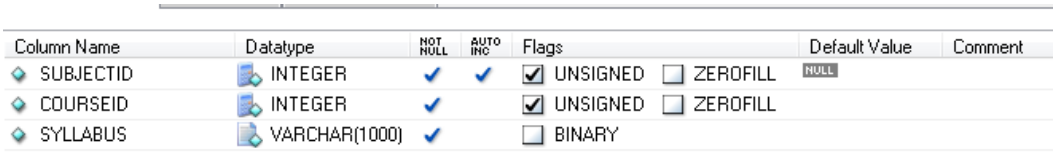
Student solutions: This table contains the student solutions to the assignments submitted by students and reviewed by faculty.



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
SOLUTIONID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
FILEID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
USERID	VARCHAR(45)	✓		☐ BINARY		
APPROVED	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
FEEDBACK	VARCHAR(500)	✓		☐ BINARY		
ANSWERFILE	VARCHAR(45)	✓		☐ BINARY		
SUBMITDATE	DATETIME	✓				

Figure 3.16: Student solutions Table

Syllabus: This table contains the syllabus for each course/subject.



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
SUBJECTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
COURSEID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
SYLLABUS	VARCHAR(1000)	✓		☐ BINARY		

Figure 3.17: Syllabus Table

Exams: This table contains all the exams created by the faculty.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
examId	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
examName	VARCHAR(100)	✓		☐ BINARY		
subjectId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
courseId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
questionIds	VARCHAR(500)	✓		☐ BINARY		
maxMarks	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
passMarks	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
duration	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
examAppovalComm...	VARCHAR(200)	✓		☐ BINARY		
examAppovalStatus	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
examCreatedBy	VARCHAR(100)	✓		☐ BINARY		
examCreationDate	DATETIME	✓				
approvedBy	VARCHAR(100)	✓		☐ BINARY		
approvalDate	DATETIME	✓				

Figure 3.18: exams Table

Questions: This table contains all questions of a particular subject which are used to create an exam.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
questionId	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
subjectId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
courseId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
questiontype	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
question	VARCHAR(1000)	✓		☐ BINARY		
options	VARCHAR(1000)	✓		☐ BINARY		
correctAnswer	VARCHAR(500)	✓		☐ BINARY		
marks	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
explanation	VARCHAR(500)			☐ BINARY	NULL	
createdBy	VARCHAR(100)	✓		☐ BINARY		
createdDate	DATETIME	✓				
updatedBy	VARCHAR(500)	✓		☐ BINARY		
updatedDate	DATETIME	✓				

Figure 3.19: Questions Table

Polls: This table contains poll questions and options.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
POLLID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
POLLQUESTION	VARCHAR(100)	✓		☐ BINARY		
POLLOPTIONS	VARCHAR(800)	✓		☐ BINARY		
CREATEDBY	VARCHAR(80)	✓		☐ BINARY		
CREATIONDATE	DATETIME	✓				
status	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		

Figure 3.20: Polls Table

PollStatus: This table contains poll status like who polled and option selected etc.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
ANSWERID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
POLLID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
SELECTEDOPTION	VARCHAR(100)	✓		☐ BINARY		
SELECTIONBY	VARCHAR(80)	✓		☐ BINARY		
SELECTIONDATE	DATETIME	✓				

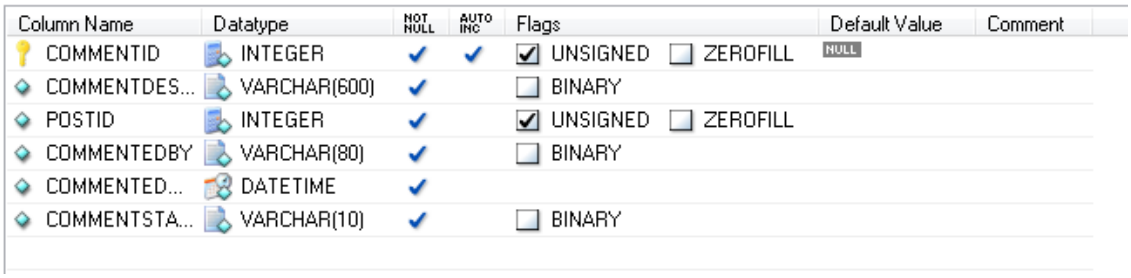
Figure 3.21: PollStatus Table

Posts: This table stores all the blog posts.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
POSTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
POSTTITLE	VARCHAR(100)	✓		☐ BINARY		
POSTDESCRIPTION	VARCHAR(600)	✓		☐ BINARY		
SUBJECTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
POSTEDBY	VARCHAR(80)	✓		☐ BINARY		
POSTEDDATE	DATETIME	✓				
POSTSTATUS	VARCHAR(10)	✓		☐ BINARY		

Figure 3.22: Posts Table

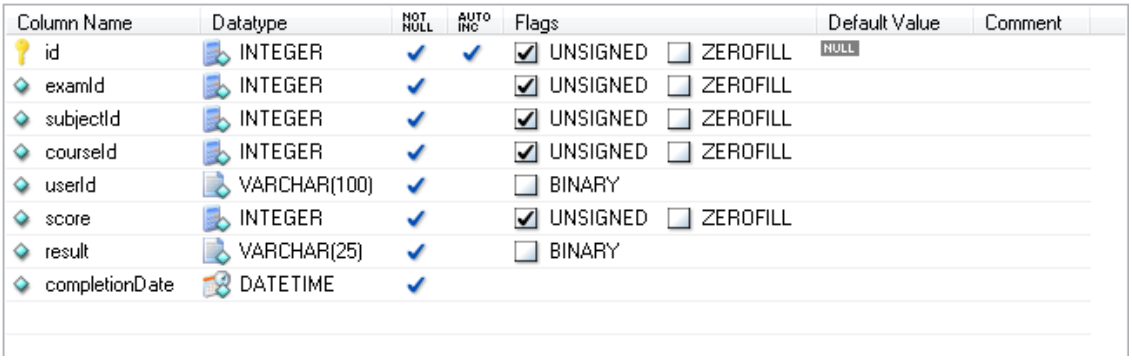
Comments: This table contains all the comments for blog post.



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
COMMENTID	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
COMMENTDES...	VARCHAR(600)	✓		☐ BINARY		
POSTID	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
COMMENTEDBY	VARCHAR(80)	✓		☐ BINARY		
COMMENTED...	DATETIME	✓				
COMMENTSTA...	VARCHAR(10)	✓		☐ BINARY		

Figure 3.23: Comments Table

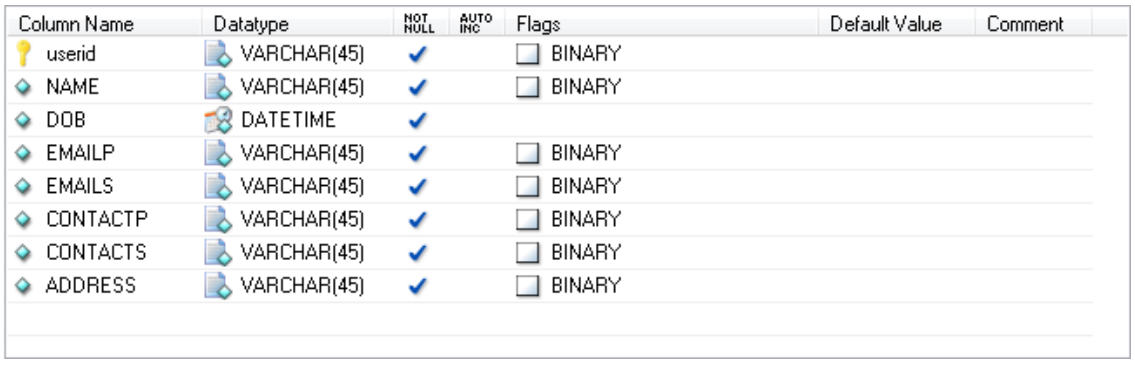
Student Exam Status: This table contains all the information on the status of exam of student. It stores exam result, score, completions date etc.



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
examId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
subjectId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
courseId	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
userId	VARCHAR(100)	✓		☐ BINARY		
score	INTEGER	✓		☑ UNSIGNED ☐ ZEROFILL		
result	VARCHAR(25)	✓		☐ BINARY		
completionDate	DATETIME	✓				

Figure 3.24: StudentExamStatus table

Admin: This table contains admin details.



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
userid	VARCHAR(45)	✓		☐ BINARY		
NAME	VARCHAR(45)	✓		☐ BINARY		
DOB	DATETIME	✓				
EMAILP	VARCHAR(45)	✓		☐ BINARY		
EMAILS	VARCHAR(45)	✓		☐ BINARY		
CONTACTP	VARCHAR(45)	✓		☐ BINARY		
CONTACTS	VARCHAR(45)	✓		☐ BINARY		
ADDRESS	VARCHAR(45)	✓		☐ BINARY		

Figure 3.25: Admin table

Technology Description

Technology Description

HTML

HTML, an initialism of Hypertext Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document — by denoting certain text as headings, paragraphs, lists, and so on — and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of labels (known as tags), surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behavior of web browsers and other HTML processors.

HTML is also often used to refer to content of the MIME type text/html or even more broadly as a generic term for HTML whether in its XML-descended form (such as XHTML 1.0 and later) or its form descended directly from SGML

Hyper Text Markup Language:
Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

Basic HTML Tags:

<! -- --> specifies comments
<A>…….</A> Creates hypertext links
<B>…….</B> Formats text as bold

<BIG>…….</BIG> Formats text in large font.
<BODY>…</BODY> Contains all tags and text in the HTML document
<CENTER>...</CENTER> Creates text
<DD>…</DD> Definition of a term
<DL>...</DL> Creates definition list
<FONT>…</FONT> Formats text with a particular font
<FORM>...</FORM> Encloses a fill-out form
<FRAME>...</FRAME> Defines a particular frame in a set of frames
<H#>…</H#> Creates headings of different levels(1 – 6)
<HEAD>...</HEAD> Contains tags that specify information about a document
<HR>...</HR> Creates a horizontal rule
<HTML>…</HTML> Contains all other HTML tags
<META>...</META> Provides meta-information about a document
<SCRIPT>…</SCRIPT> Contains client-side or server-side script
<TABLE>…</TABLE> Creates a table
<TD>…</TD> Indicates table data in a table
<TR>…</TR> Designates a table row
<TH>…</TH> Creates a heading in a table

Attributes:

The attributes of an element are name-value pairs, separated by "=", and written within the start label of an element, after the element's name. The value should be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML).Leaving attribute values unquoted is considered unsafe.

Most elements take any of several common attributes: id, class, style and title. Most also take language-related attributes: lang and dir.
The id attribute provides a document-wide unique identifier for an element. This can be used by stylesheets to provide presentational properties, by browsers to focus attention on the specific element or by scripts to alter the contents or presentation of an element. The class attribute provides a way of classifying similar elements for presentation purposes. For example, an HTML document (or a set of documents) may use the designation class="notation" to indicate that all elements with this class value are all subordinate to the main text of the document (or documents). Such notation classes of elements might be gathered together and presented as footnotes on a page, rather than appearing in the place where they appear in the source HTML.

An author may use the style non-attributal codes presentational properties to a particular element. It is considered better practice to use an element’s son- id page and select the element with a stylesheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach subtextual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tooltip. The generic inline span element can be used to demonstrate these various non-attributes.

The preceding displays as HTML (pointing the cursor at the abbreviation should display the title text in most browsers).

Advantages:

- A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- HTML is platform independent.
- HTML tags are not case-sensitive.

Javascript

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browser's display accordingly.

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

<SCRIPTS>.. </SCRIPT>.
<SCRIPT LANGUAGE = "JavaScript">
JavaScript statements
</SCRIPT>

Here are a few things we can do with JavaScript:
- Validate the contents of a form and make calculations.
- Add scrolling or changing messages to the Browser's status line.
- Animate images or rotate images that change when we move the mouse over them.
- Detect the browser in use and display different content for different browsers.
- Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.
JavaScript Vs Java

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.
Advantages
- JavaScript can be used for Sever-side and Client-side scripting.
- It is more flexible than VBScript.
- JavaScript is the default scripting languages at Client-side since all the browsers supports it.

JAVA

Initially the language was called as "oak" but it was renamed as "Java" in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.
- Finally, Java is to Internet programming where C was to system programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java – compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of Java Security

Every time you that you download a "normal" program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a "firewall" between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient.

The Byte code

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

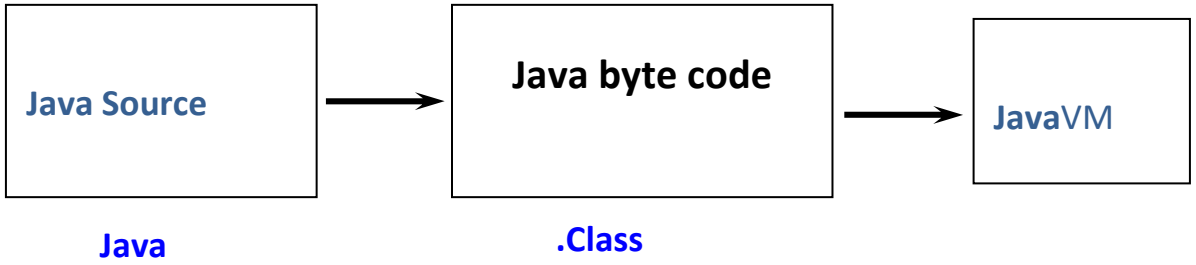


Figure 4.1: Picture showing the development process of JAVA Program

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The .Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

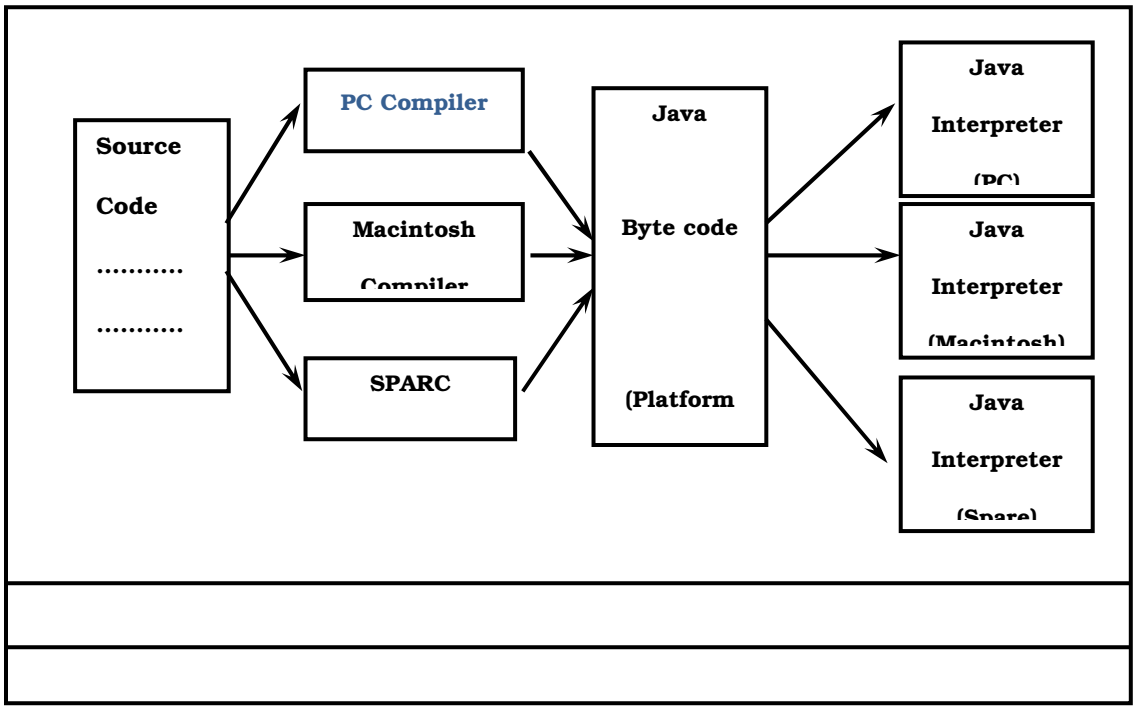
Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Figure 4.2 : Compiling and interpreting Java Source Code



During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or SunSARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

Simple

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

Object-Oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.
Java virtually eliminates the problems of memory management and de-allocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

Java DataBase Connectivity (JDBC)

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.
What Does JDBC Do?

Simply put, JDBC makes it possible to do three things:
- Establish a connection with a database
- Send SQL statements
- Process the results.

JDBC versus ODBC and other APIs
At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms.

So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question:

1. ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
2. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void *". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers.
3. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.
4. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

Two-tier and Three-tier Models

The JDBC API supports both two-tier and three-tier models for database access.
In the two-tier model, a Java applet or application talks directly to the database. This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. The database may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the database as the server. The network can be an Intranet, which, for example, connects employees within a corporation, or it can be the Internet.

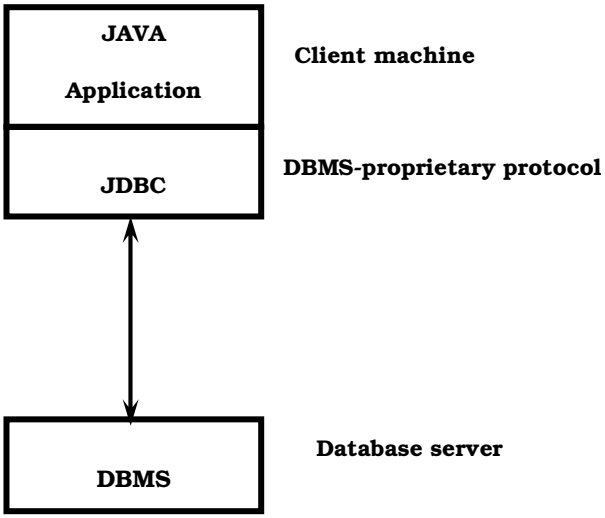


Figure 4.3: Two tier Model

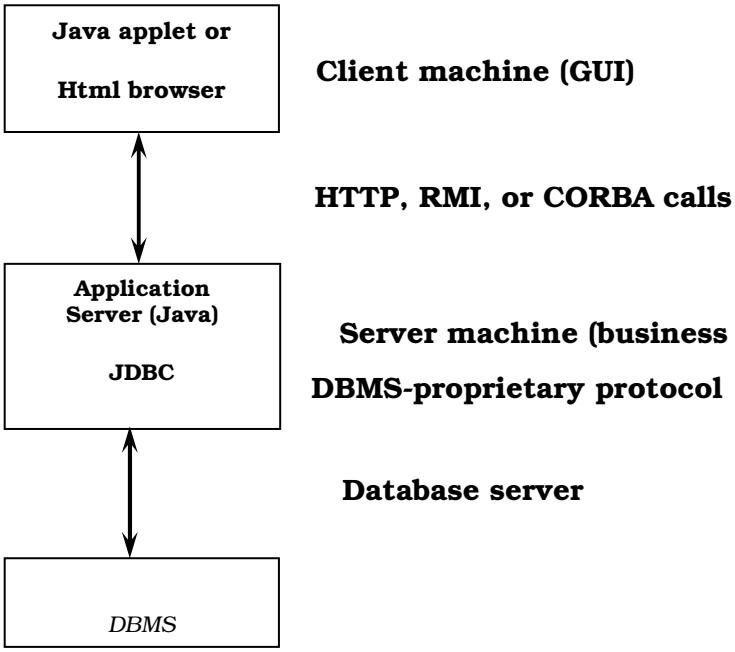
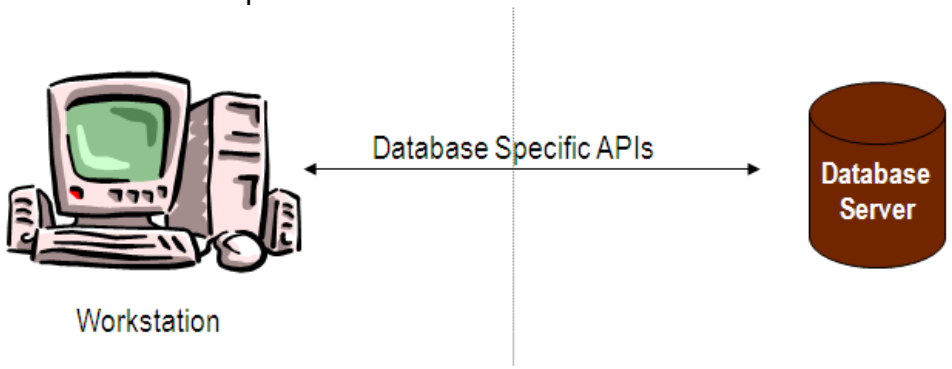


Figure 4.4: Three tier Model

In the three-tier model, commands are sent to a "middle tier" of services, which then send SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally, in many cases the three-tier architecture can provide performance advantages.

Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code, it is becoming practical to implement the middle tier in Java. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features. JDBC is important to allow database access from a Java middle tier.



- Presentation, business and data model processing logic into client application
- Server is typically a database server
- Client sends SQL statements, retrieves raw data

JDBC Driver Types

The JDBC drivers that we are aware of at this time fit into one of four categories:

- JDBC-ODBC bridge plus ODBC driver
- Native-API partly-Java driver
- JDBC-Net pure Java driver
- Native-protocol pure Java driver

JDBC-ODBC Bridge

If possible, use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, the ODBC driver library, and the database client library).

What Is the JDBC- ODBC Bridge?

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the Sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of Innersole and Java Soft.

JDBC connectivity

The JDBC provides database-independent connectivity between the J2EE platform and a wide range of tabular data sources. JDBC technology allows an Application Component Provider to:

- Perform connection and authentication to a database server
- Manager transactions
- Move SQL statements to a database engine for preprocessing and execution
- Execute stored procedures
- Inspect and modify the results from Select statements

Database:

A database management system (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems.

Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

Description

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

- ✓ A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
  - The four most common types of organizations are the hierarchical, network, relational and object models. Inverted lists and other methods are also used. A given database management system may provide one or more of the four models. The optimal structure depends on the natural organization of the application's data, and on the application's requirements (which include transaction rate (speed), reliability, maintainability, scalability, and cost).
  - The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.
- ✓ Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory).

- ✓ A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.
  - It also controls the security of the database.
  - Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called subschemas. For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data.

- If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.

✓ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).
- It also maintains the integrity of the data in the database.
- The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. New categories of data can be added to the database without disruption to the existing system.

Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with RAID disk arrays used for stable storage. Connected to one or more servers via a high-speed channel, hardware database accelerators are also used in large volume transaction processing environments.
DBMSs are found at the heart of most database applications. Sometimes DBMSs are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

SQL

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

In the relational model, data is stored in structures called relations or tables.
SQL statements are issued for the purpose of:

Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes).

Data manipulation: Used to manipulate the data within those schema objects (DML Inserting, Updating, Deleting the data, and Querying the Database).

A schema is a collection of database objects that can include: tables, views, indexes and sequences

List of SQL statements that can be issued against an Oracle database schema are:

- **ALTER** - Change an existing table, view or index definition (DDL)
- **AUDIT** - Track the changes made to a table (DDL)
- **COMMENT** - Add a comment to a table or column in a table (DDL)
- **COMMIT** - Make all recent changes permanent (DML - transactional)
- **CREATE** - Create new database objects such as tables or views (DDL)
- **DELETE** - Delete rows from a database table (DML)
- **DROP** - Drop a database object such as a table, view or index (DDL)
- **GRANT** - Allow another user to access database objects such as tables or views (DDL)
- **INSERT** - Insert new data into a database table (DML)
- **No AUDIT** - Turn off the auditing function (DDL)
- **REVOKE** - Disallow a user access to database objects such as tables and views (DDL)
- **ROLLBACK** - Undo any recent changes to the database (DML - Transactional)
- **SELECT** - Retrieve data from a database table (DML)
- **TRUNCATE** - Delete all rows from a database table (can not be rolled back) (DML)
- **UPDATE** - Change the values of some data items in a database table (DML)

SERVLETS

The Java web server is JavaSoft's own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

About Servlets

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side - object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

- They’re faster and cleaner than CGI scripts
- They use a standard API (the Servlets API)
- They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

Attractiveness of Servlets

There are many features of Servlets that make them easy and attractive to use. These include:

- Easily configured using the GUI-based Admin tool
- Can be loaded and invoked from a local disk or remotely across the network.
- Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.
- Can be called dynamically from within HTML pages, using server-side include tags.
- Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

Advantages of the Servlet API

One of the great advantages of the Servlet API is protocol independence. It assumes nothing about:

- The protocol being used to transmit on the net
- How it is loaded
- The server environment it will be running in

These qualities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the Servlet API as well. These include:

- It's extensible - you can inherit all your functionality from the base classes made available to you.
- It's simple, small, and easy to use.

- Web components
  - Servlets or JSP pages
  - JavaBeans (optional)

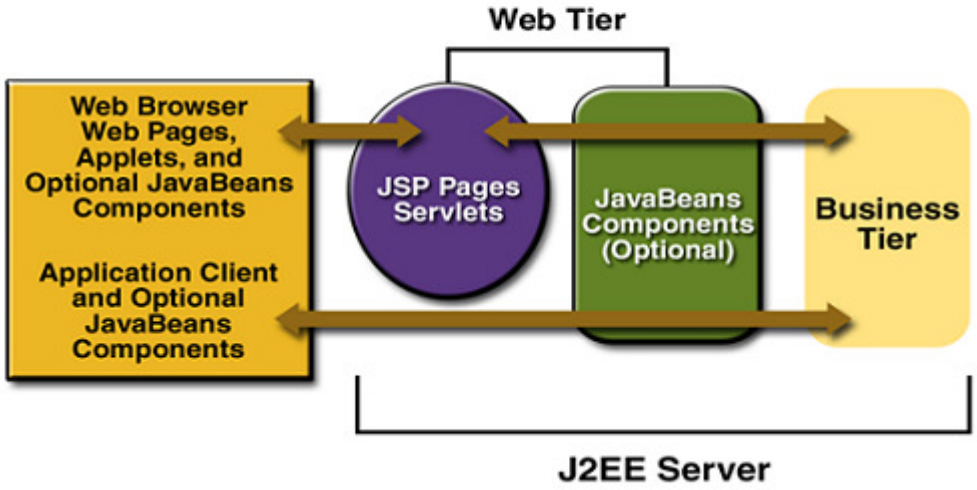


Figure 4.5 : Servlets

Features of Servlets:

- Servlets are persistent. Servlet are loaded only by the web server and can maintain services between requests.
- Servlets are fast. Since Servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
- Servlets are platform independent.
- Servlets are extensible. Java is a robust, object-oriented programming language, which easily can be extended to suit your needs
- Servlets are secure.
- Servlets can be used with a variety of clients.

Loading Servlets:

Servlets can be loaded from three places
From a directory that is on the CLASSPATH. The CLASSPATH of the JavaWebServer includes service root/classes/ which is where the system classes reside.

From the <SERVICE_ROOT /Servlets/ directory. This is *not* in the server's class path. A class loader is used to create Servlets from this directory. New Servlets can be added - existing Servlets can be recompiled and the server will notice these changes.

From a remote location, for this a code base like http: // nine.eng / classes / foo / is required in addition to the Servlets class name. Refer to the admin GUI docs on Servlet section to see how to set this up.

Loading Remote Servlets

Remote Servlets can be loaded by:

1. Configuring the Admin Tool to setup automatic loading of remote Servlets
2. Setting up server side include tags in. shtml files
3. Defining a filter chain configuration

Invoking Servlets

A Servlet invoker is a Servlet that invokes the "service" method on a named Servlet. If the Servlet is not loaded in the server, then the invoker first loads the Servlet (either from local disk or from the network) and the then invokes the "service" method. Also like applets, local Servlets in the server can be identified by just the class name. In other words, if a Servlet name is not absolute, it is treated as local.

A client can invoke Servlets in the following ways**:**

- The client can ask for a document that is served by the Servlet.
- The client (browser) can invoke the Servlet directly using a URL, once it has been mapped using the Servlet Aliases section of the admin GUI.
- The Servlet can be invoked through server side include tags.
- The Servlet can be invoked by placing it in the Servlets/ directory.
- The Servlet can be invoked by using it in a filter chain.

Java Server Pages (JSP)

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model .The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches; it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other's work.

Features of JSP

Portability
Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

Components
It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.

Processing
A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

Access Models
A Java Server Pages file may be accessed in at least two different ways. A client's request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

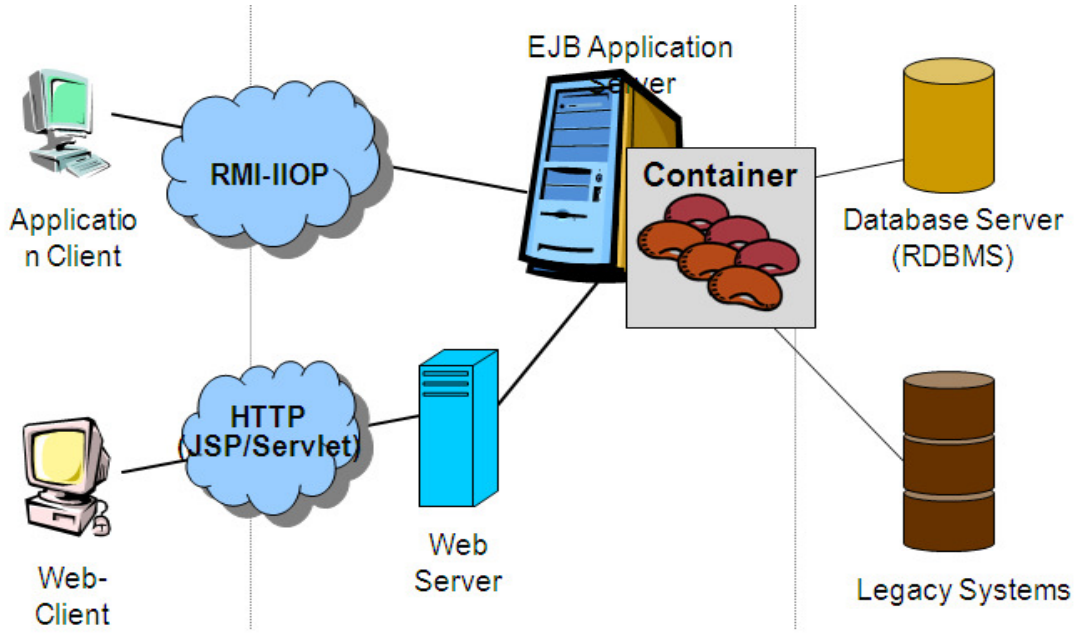


Figure 4.6 : JSP

Steps in the execution of a JSP Application:

1. The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.
2. This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
3. JSP engine is program which can under stands the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

Struts:

What is a Web Application?

Applications built using the Struts framework are at their core, web applications. A web application is a collection of individual components that once bound together, form a complete application that can be installed and executed by a web container. The components are tied together due to the fact that they reside in the same web context and in many cases, may refer to one another, directly or indirectly.

Elements of a Web Application

Obviously, not all web applications are created equal. They will not have the same functional and non-functional requirements across organizations, departments, or even the same vertical markets. Therefore, not all web applications will contain the same types of resources. In general however, web applications can consist of one or more of the following types of components:

- Servlets
- JSP Pages
- Standard JavaBeans and Utility Classes
- HTML Documents
- Multimedia Files (Images, Audio and Video Files, CAD Drawings, etc…)
- Client side Applets, Stylesheets, and JavaScript Files
- Text Documents

Meta information that ties all of the above components together

JSP Model 1 and Model 2 Architectures

The early JSP specifications presented two approaches for building web applications using JSP technology. These two approaches were described in the specification as JSP Model 1 and Model 2 architectures. Although the terms are no longer used in the JSP specification, their usage throughout the web tier development community is still widely used and referenced.

The two JSP architectures differed in several key areas. The major difference was how and by which component the processing of a request was handled. With the Model 1architecture, the JSP page handles all of the processing of the request and is also responsible for displaying the output to the client.

In direct comparison to the Model 1 approach, in the Model 2 architecture, the client request is first intercepted by a servlet, most often referred to as a Controller servlet. The servlet handles the initial processing of the request and also determines which JSP page to display next.

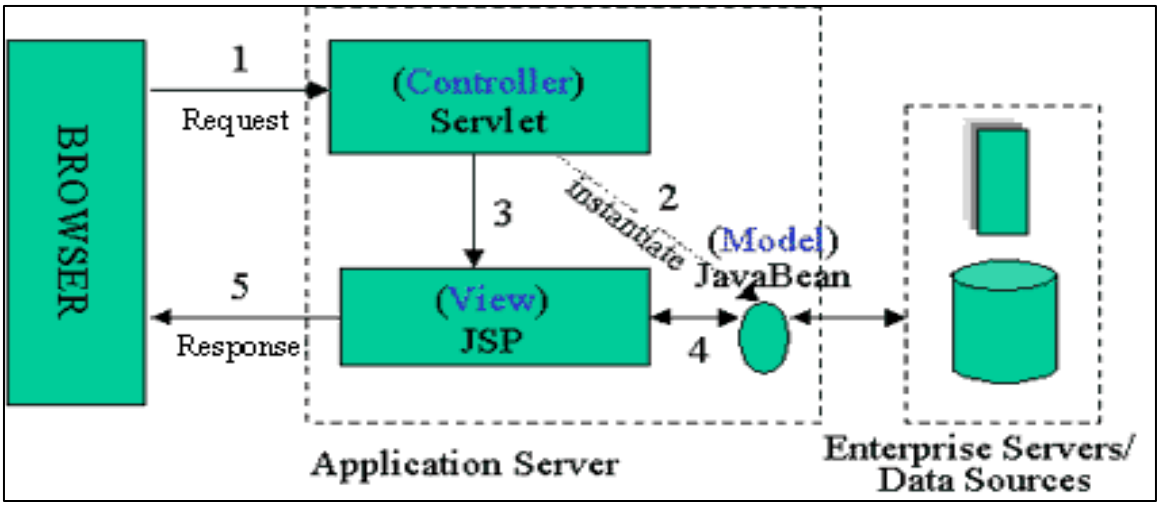


Figure 4.7 : Model 2 Architecture

As you can see from the above figure, in the Model 2 architecture, a client never sends a request directly to a JSP page. The controller servlet acts as sort of a traffic cop. This allows the servlet to perform front-end processing like authentication and authorization, centralized logging, and possibly help with Internationalization. Once processing of the request has finished, the servlet directs the request to the appropriate JSP page. How exactly the next page is determined can vary widely across different applications. For example, in simpler applications, the next JSP page to display may be hard coded in the servlet based on the request, parameters, and current application state. In other more sophisticated web applications, a workflow/rules engine may be used.

As you can see, the main difference between the two approaches is that the Model 2 architecture introduces a controller servlet that provides a single point of entry and also encourages more reuse and extensibility than Model 1. With the Model 2 architecture, there is also a clear separation of the business logic, presentation output, and request processing. This separation is often referred to as a Model-View-Controller (MVC) pattern. While the Model 2 architecture might seem overly complicated, it can actually simplify an application greatly. Web applications built using the Model 2 approach are generally easier to maintain and can be more extensible than comparable applications built around the Model 1 architecture.

All of this doesn't mean that applications built using the Model 1 approach are incorrectly designed. The Model 1 architecture might be the best decision for smaller applications that have simple page navigation, no need for centralized features, and are fairly static. However, for more larger enterprise-size web applications, it would be more advantageous to utilize the Model 2 approach.

Why is Model-View-Controller So Important?

Model-View-Controller is an architectural pattern that by it self has nothing to do with web applications directly. As we saw from the previous section, the JSP Model 2 approach is clearly about separating responsibilities in a web application built

using Servlet and JSP technologies. Allowing a JSP page to handle the responsibilities of receiving the request, executing some business logic, and then determining the next view to display can really make for an unattractive JSP page, not to mention the problems this entanglement causes for maintenance and extensibility. By having components within a web application that have very clear and distinct responsibilities, the development and maintenance on an application can be made more efficient. This is also true for software development as a whole. The MVC pattern is categorized as a design pattern in many software design books. Although there is usually much disagreement on the precise definition of the pattern, there are some fundamental ideas.

The MVC pattern has three key components:
The Model Component
Responsible for the business domain state knowledge
The View Component
Responsible for a presentation view of the business domain
The Controller Component
Responsible for controlling flow and state of the user input

Normally with the MVC pattern, there's a form of event notification that takes place to notify the view when some portion of the model changes. However, since a browser in a typical web application has a stateless connection, the notification from the model to the view can't easily occur. Of course, an application could perform some type of push action to push data changes all the way to a client; but this doesn't and probably shouldn't happen in most web applications. A user can close at a browser anytime and there isn't warning or notification sent to the server. There's a great deal of overhead necessary to management remote clients from the server side. This type of behavior is overkill for typical B2C and B2B web applications.

The MVC Model

Depending on the type of architecture of your application, the model portion of the MVC pattern can take many different forms. In a two-tier application, where the web tier interacts directly with a data store like a database, the model classes may be a set of regular Java objects. These objects may be populated manually from a result set returned by a database query or they can even be instantiated and populated automatically by an Object-to-Relational Mapping (ORM) framework.

In a more complex enterprise application where the web tier communicates with an EJB server for example, the model portion of the MVC pattern might be Enterprise JavaBeans. Although the EJB 2.0 Specification made some improvements in performance through the use of local interfaces, there can still be a significant performance impact if the web tier attempted to use entity beans directly as the model portion of the application. In many cases, JavaBeans are returned from Session beans and used within the web tier. These JavaBeans are commonly referred to as value objects and are used within the views to build the dynamic content.

The MVC View

The views within the web tier MVC pattern typically consist of HTML and JSP pages. HTML pages are used to serve static content, while JSP pages can be used to serve both static and dynamic content. Most dynamic content is generated in the web tier. However, Web applications are considered stateless because the browser doesn't typically maintain an open socket to the web server. However, a web application may still maintain session data for a user or even store data within the browser on behalf of the user. some applications may require the need for client-side JavaScript. This does not interface or infringe upon the MVC concept.

The MVC Controller

The controller portion of the web tier MVC design is generally a Java servlet. The controller in a web tier application performs the following duties:

1. Intercepts HTTP requests from a client.
2. Translates the request into a specific business operation to perform.
3. Either invokes the business operation itself or delegates to a handler.
4. Helps to select the next view to display to the client.
5. Returns the view to the client.

The Front Controller pattern, which is part of the J2EE Design Patterns, describes how a web tier controller should be implemented. Since all client requests and responses go through the controller, there is a centralized point of control for the web application. This aides in maintenance and when adding new functionality. Code that would normally need to be put in every JSP page can be put in the controller servlet, since it processes all requests. The controller also helps to decouple the presentation components (views) from the business operations, which also aids development.

What is a Framework?

I have been throwing the word framework around in this chapter without having really defined what exactly it is or how it adds value in software development. In its simplest form, a framework is a set of classes and interfaces that cooperate to solve a specific type of software problem. A framework has the following characteristics:

- ✓ A framework is made up of multiple classes or components, each of which may Provide an abstraction of some particular concept.
- ✓ The framework defines how these abstractions work together to solve a problem.
- ✓ The framework components are reusable

A good framework should provide generic behavior that can be utilized across many different types of applications. There are many interpretations of what constitutes a framework. Some might consider the classes and interfaces provided by the Java language a framework, but it's really a library. There's a subtle, but very distinct difference between a software library and a framework. With a software library, your application is the main code that executes and it invokes routines on the library. With a framework, it contains the executing routines and invokes operations onto your extensions through inheritance and other means. The places where the framework can be extended are known as extension points. A framework is commonly referred to an "upside-down" library because of the alternate manner in which it operates.

Creation of the Struts Framework

By now you should have a foundation for JSP and Servlet technology and you should also understand the benefits that the Web MVC design and JSP Model 2 architecture adds to a web application. This section provides a little background and history on the Struts framework, which is an implementation of all of these ideas.

High-level Diagram of Struts Framework:

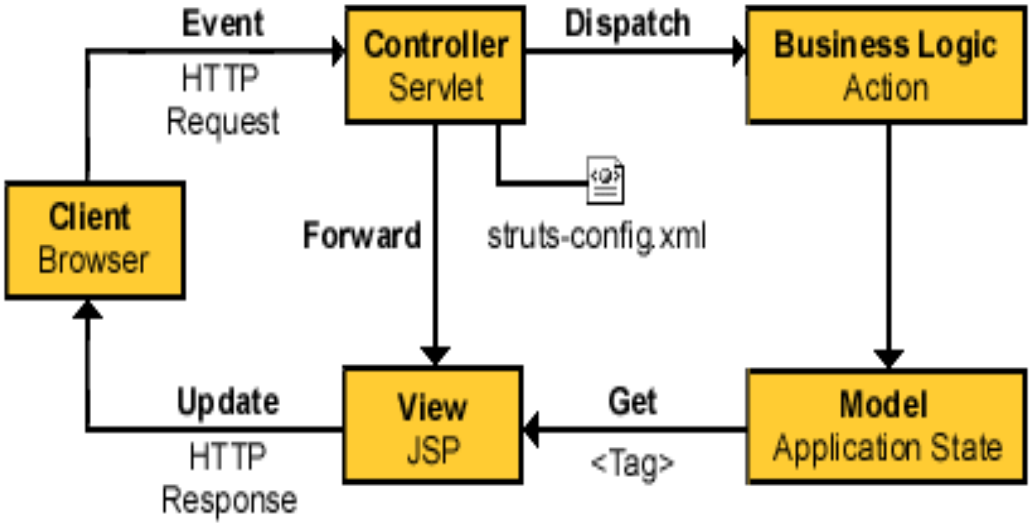


Figure 4.8 : High-level Diagram of struts framework

Typical Diagram of Struts Framework:

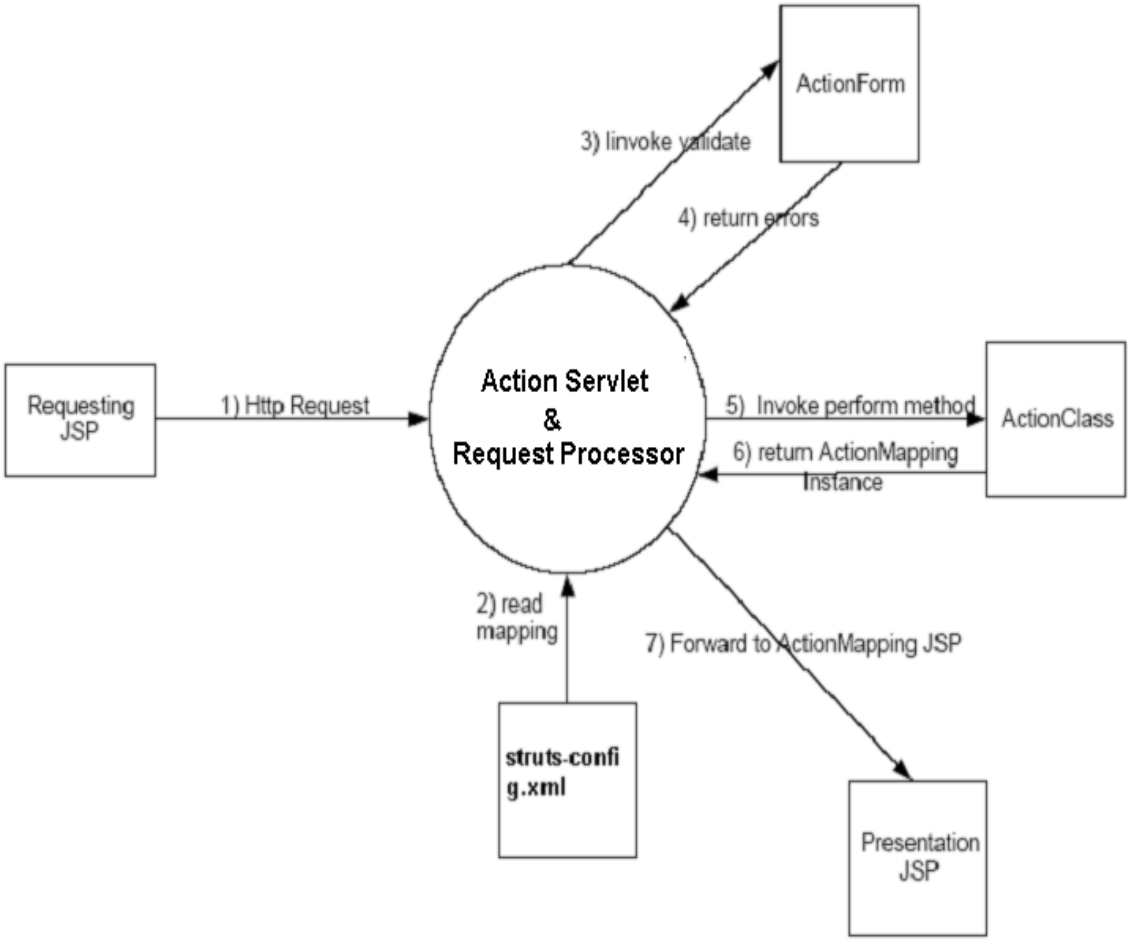


Figure 4.9 : Typical Diagram of struts framework

The Struts Component Packages

The Struts framework is made up of approximately 200 Java classes, divided into 15 Java packages. Approximately is an appropriate term because the framework is continuously growing and being shaped.

Struts Controller Components

The controller component in a MVC application has several Responsibilities. Those responsibilities include receiving input from a client, invoking a business operation, and coordinating the view to return back to the client. Of course, there are many other functions that the controller may perform, but these are a few of the primary ones.

You also learned that, with the JSP Model 2 architecture, on which Struts was fashioned the controller was implemented by a Java servlet. This servlet becomes the centralized point of control for the web application. The controller servlet maps user actions into business operations and then helps to select the view to return to the client based on the request and other state information.

In the Struts framework however, the controller responsibilities are implemented by several different components, one of which is an instance of the org.apache.struts.action.ActionServlet class.

The Struts ActionServlet

The **ActionServlet** extends the javax.servlet.http.HttpServlet class and is responsible for packaging and routing HTTP traffic to the appropriate handler in the framework. The ActionServlet class is not abstract and therefore can be used as a concrete controller by your applications. Prior to version 1.1 of the Struts framework, the ActionServlet was solely responsible for receiving the request and processing it by calling the appropriate handler. In version 1.1, a new class called org.apache.struts.action.RequestProcessor has been introduced to process the request for the controller. The main reason for decoupling the request processing from the ActionServlet is to provide you with the flexibility to subclass the RequestProcessor with your own version and modify how the request is processed. For the banking application example, we are going to keep it simple and utilize the default ActionServlet and RequestProcessor classes provided by the framework.

Once the controller receives a client request, it delegates the handling of the request to a helper class. This helper knows how to execute the business operation that is associated with the requested action. In the Struts framework, this helper class is a descendant of the org.apache.struts.action.Action class.

Struts Action Classes

An **org.apache.struts.action.Action** class in the Struts framework is an extension of the controller component. It acts as an Adaptor between a user action and a business operation. The Action class decouples the client request from the business model. This decoupling allows for more than a one-to-one mapping between the user request and an Action class. The Action class can perform other functions, such as authorization, logging, and session validation, before invoking the business operation.

The Struts Action class contains several methods, but the most important is the execute() method. Here is the method signature:

```
public ActionForward execute(ActionMapping mapping,
ActionForm form, HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException;
```

The execute() method is called on an instance of an Action class by the controller when a request is received from a client. The controller will create an instance of the Action class if one doesn't already exist. The Struts framework will only create a single instance of each Action class in your application. Since there is only one instance for all users, you must ensure that all of your Action classes operate properly in a multi-threaded environment. Although the execute() method is not abstract, the default implementation just returns null so you will need to create your own Action class implementations and override this method.

Mapping the Actions

At this point, you might be asking yourself, "How does the controller know which Action instance to invoke when it receives a request?" The answer is by inspecting the request information and utilizing a set of action mappings. Action mappings are part of the Struts configuration information that is configured in a special XML file. This configuration information is loaded into memory at startup and made available to the framework at runtime. Each <action> element is represented in memory by an instance of the org.apache.struts.action.ActionMapping class. The ActionMapping object contains a path attribute that is matched against a portion of the URI of the incoming request.

Whenever the controller receives a request where the path in the URI contains the string "/login", the execute() method of the LoginAction instance will be invoked. The Struts framework also uses the mappings to identify the resource to forward the user to once the action has completed.

Determining the Next View

We've talked about how the controller receives the request and how the action mappings and request information are used to determine the correct action instance to invoke and pass the request to. What hasn't been discussed is how or what determines the view to return back to the client. If you looked closely at the execute() method signature in the Action class from the previous section, you might have noticed that the return type for the method is an **org.apache.struts.action.ActionForward** class. The ActionForward class represents a destination to which the controller may send control once an Action has completed. Instead of specifying an actual JSP page in the code, you can declaratively associate an action forward mapping with the JSP and then use that ActionForward throughout your application. The action forwards are specified in the configuration file, similar to action mappings.

The logout action declares a <forward> element that is named "Success", which forwards to a resource of "/login.jsp". Notice in this case, a redirect attribute is set to "true". Instead of performing a forward using a RequestDispatcher, the request that invokes the logout action mapping will be redirected instead.

The action forward mappings can also be specified in a global section independent of any specific action mapping. In the previous case, only the logout action

mapping could reference the action forward named "Success". In the case of forwards declared in the global forwards section, all action mappings can reference them. Here is an example global forwards section from the configuration file:

```
<global-forwards>
<forward name="SystemFailure" path="/systemerror.jsp" />
<forward name="SessionTimeOut" path="/sessiontimeout.jsp" />
</global-forwards>
```

The forwards defined in the global section are more general and don't apply to a specific action. Notice that every forward must have a name and path, but the redirect flag is optional. If you don't specify a redirect attribute, its default value is false and thus performs a forward. The path attribute in an ActionForward can also specify another Struts Action.

Struts Model Components

There are several different ways to look at what constitutes a model for Struts. The lines between business and presentation objects can get quite blurry when dealing with web applications. One application's business objects are another's value objects. It's important to keep the business objects separate from the presentation, so that the application is not tightly coupled to one type of presentation. It's very likely that the look and feel of a web site will change over time. Studies show that the freshness of a web site's appearance helps to attract new customers and also keep existing customers coming back. This may not be as true in the Business-to-Business (B2B) world, but it's definitely true for Business-to-Consumer (B2C) applications, which make up the majority of the web applications used today.

The Struts View Components

The last of the MVC components to discuss are the Struts View components. Arguably, it's the easiest to understand. The view components that are typically employed in a Struts application are:

JavaServer Pages
Custom Tags
HTML
Java Resource Bundles
Struts Action Forms
Value Objects

Using the Struts ActionForm

Struts ActionForm objects are used in the framework to pass client input data back and forth between the user and the business layer. The framework will automatically collect the input from the request and pass this data onto an Action using a form bean, which then can be passed along to the business layer. To keep the presentation layer decoupled from the business layer, you should not pass the action form itself to the business layer, but rather create the appropriate value

objects using the data from the form and pass these objects as argument to the business layer. The following steps that illustrate how the framework processes an ActionForm for every request.

1. Check the mapping for the action and see if a form bean has been configured for it.
2. If so, use the name attribute to lookup the form bean configuration information.
3. Depending on the scope configured for the form bean for the action, check to see if there's already an instance of the form bean at the appropriate scope.
4. If an ActionForm instance is present in the appropriate scope and it's the same type as needed for this new request, reuse it.
5. Otherwise, create a new instance of the required form bean and store it into the appropriate scope that is set by the scope attribute for the action mapping.
6. The reset() method is called on the ActionForm instance.
7. Iterate through the request parameters and for every parameter name that has a corresponding set method name on the ActionForm, populate it with the value for that parameter.
8. Finally, if the validate attribute is set to true, then invoke the validate() method on the ActionForm instance and return any errors.

Using JavaServer Pages for Presentation

JavaServer Pages make up the majority of what has to be built for the Struts view components. Combined with custom tag libraries and HTML, it becomes easy to provide a set of views for an application. Although JavaServer Pages make up the majority of what organizations and developers are using to display the dynamic content, it's not the only technology. There are other forms of presentation technologies that can be combined with the Struts framework. One very popular one is the XML/XSLT combination. This alternate model is being referred to as Model 2X, which is a combination of the controller servlet from the Struts framework and XSLT and beans serialized from the value objects to render the views. Many developers feel that JSP has the following problems:

- Developers are free to embed application logic into the JSP pages. This can lead to an application that is difficult to maintain.
- JSP syntax is not currently XML compliant, which may cause the XML or HTML that gets generated, not to be "well formed".
- Developers must learn the JSP syntax and how to program custom tags.
- Developing a processing pipeline where each node in the pipeline may modify the data or layout is not possible with JSP pages. This makes it difficult to separate layout and style.

A recompile of the JSP page is necessary for each change made to the page.

Custom Tag Libraries

The Struts framework provides five core tag libraries that can be used by your applications. Each one has a different purpose and can be used individually or along side the others. You may also create your own custom tags and can even extend the Struts tags if you need them to perform extra functionality. The custom tag libraries that are included with the framework are:
1) HTML Tag Library
2) Bean Tag Library
3) Logic Tag Library
4) Template Tag Library
5) Nested Tag Library

Unless you are planning on using templates as part of your application, the Template Tag library might not be necessary, but the others are invaluable to making your application easy to develop and maintain.

To use the libraries in your application, you need to first register them with the web application. To do this, you should add the following lines to the deployment descriptor for each web application that you wish to use Struts for.

```
<web-app>
<taglib>
<taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
<taglib-location>/WEB-INF/struts-html.tld</taglib-location>
</taglib>
<taglib>
<taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
<taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
</taglib>
<taglib>
<taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
<taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
</taglib>
<taglib>
<taglib-uri>/WEB-INF/struts-template.tld</taglib-uri>
<taglib-location>/WEB-INF/struts-template.tld</taglib-location>
</taglib>
<taglib>
<taglib-uri>/WEB-INF/struts-nested.tld</taglib-uri>
<taglib-location>/WEB-INF/struts-nested.tld</taglib-location>
</taglib>
</web-app>
```

The next step is to create your JSP pages and include the following lines.

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
<%@ taglib uri="/WEB-INF/struts-nested.tld" prefix="nested" %>
```

Once this is done and the Struts JAR file is in the web application's CLASSPATH, you are then able to use the custom tags in your JSP pages.

Message Resource Bundles

The Java library includes a set of classes to support reading message resources from either a Java class or a properties file. The core class in this set is the java.util.ResourceBundle. The Struts framework provides a similar set of classes, based around the org.apache.struts.util.MessageResources class that provides similar functionality, but provides for a little more flexibility that the framework requires.

The standard Java support for Internationalization has grown with the past several releases and the Struts framework could probably use what's included with 1.2 and newer, but since Struts was created before 1.2, they had to build in their own support for several key pieces. With a Struts application, you must provide a Java message bundle for each language that you wish to support A message resource bundle is used for reasons other than just for localization. It can also save time during application maintenance. For example, if you use the same text messages or labels throughout various parts of your web site or application, when one or more of these values need to change, you only need to make the change in a single location. Even if you don't have requirements for Internationalization, you should still utilize resource bundles. With Struts 1.1, you now have the ability to define multiple MessageResources for an application. This allows you isolate certain types of resources, into separate bundles. For example, you might want to store the image resources for an application into one bundle and the rest of the resources into another. How you organize your application's resources is up to you, but you now have the flexibility to separate them based on some criteria. Some applications choose to separate along components lines

Multiple Application Support

Prior to version 1.1, each Struts application was limited to having a single configuration file. The single instance of the file, which is normally called struts-config.xml, was specified in the web application deployment descriptor. It was the sole provider of the configuration information for the Struts application. The fact that there was only a single place to put configuration information made it very difficult for larger projects because it often became a bottleneck and caused contentions to use and modify this file. With the creation of 1.1, this problem has been alleviated with the advent of multi application support. You can now define multiple configuration files and allow developers to work better in parallel.

Configuring the Struts Application

The Struts framework uses two separate, but somewhat related types of configuration files, which must be properly configured before an application will function properly. Due to the popularity and flexibility of the self-describing nature of XML, both of these configuration files are based on XML.

The web application deployment descriptor web.xml is described fully in the Java Servlet specification. This configuration file is necessary for all web applications, not just those built with the Struts framework. There is however, Struts specific deployment information that must be configured within it when building web applications using Struts.

The second configuration file is the Struts configuration file, commonly named struts-config.xml. As you'll see however, the name can be just about anything that you want it to. The Struts configuration file makes it possible for you to declaratively configure many of your application's settings. You can think of the Struts configuration file as the rules for the web application.

Template for struts-config.xml file:

```
<struts-config>

<data-sources>
<form-beans>
<global-exceptions>
<global-forwards>
<action-mappings>
<controller>
<message-resources>
<plug-in>

</struts-config>
```

Note that we have to follow the above order strictly, otherwise it will throw exceptions.

What are ActionForms?

Almost every web application has a requirement to accept input from users. Some examples of user input are credit card information, billing and shipping address information, or even something as small as a username and password. The HTML language provides the necessary components to render the input fields in a browser, including text boxes, radio buttons, check boxes, buttons, and many more. When building these types of pages, the input components must be nested inside of an HTML form Elements. The Struts framework relies on the org.apache.struts.action.ActionForm class as the key component for handling these tasks.

The ActionForm class is used to capture input data from an HTML form and transfer it to the Action class. Since the HTML input components don't natively include an input buffer and users quite often enter invalid data, web applications need a way to store the input data temporarily, so that it can be redisplayed when an error occurs. In this sense, the ActionForm class acts as a buffer to hold the state of what the user entered while it is being validated. The ActionForm also acts as a "firewall" for your application in that it helps to keep suspect or invalid input out of your business tier until it can be scrutinized by the validation rules. Finally, the ActionForm is also used to transfer data from the Action class back to the HTML form. This allows more consistency for your HTML forms, because they are always pulling data from the ActionForm. When the user input data does pass input validation, the ActionForm is passed into the execute() method of the Action class. From there, the data can be retrieved from the ActionForm and passed on to the business tier. Because the ActionForm imports packages from the Servlet API, you shouldn't pass the ActionForm to the business tier. Doing so would couple the business methods to the Servlet API and make it more difficult to reuse the business tier components. Instead, the data within the ActionForm should be transferred to an object from the domain model instead of being passed as an argument to the business tier. A common approach is to create a data transfer object and populate it with the data from the ActionForm. You don't have to declare an ActionForm for every HTML form in your application. The same ActionForm can be associated with one or more action mappings. This means that they can be shared across multiple HTML forms. For example, if you had a wizard interface, where a set of data was entered and posted across multiple pages, a single ActionForm can be used to capture all of this data, a few fields at a time.

ActionForms and Scope

ActionForms can have two different levels of scope, request and session. If request scope is used, an ActionForm will only be available until the end of the request/response cycle. Once the response has been returned to the client, the ActionForm and the data within it is no longer accessible.

If you need to keep the form data around longer than the request, you can configure an ActionForm to have session scope. This might be necessary if your application captures data across multiple pages, similar to a wizard dialog. An ActionForm that has been configured with session scope will remain in the session until it's removed, replaced with another object, or until the session times out. The framework doesn't have a built-in facility for cleaning up session scoped ActionForm objects automatically. Like any other object placed into the HttpSession, it's up to the application to routinely perform clean up on the resources stored there. This is slightly different from objects placed into the request, because once the request is finished, they can be reclaimed by the garbage collector since they can no longer be referenced. Unless you need to specifically hold the form data across multiple requests, you should use request scope for your ActionForm objects.

When the controller receives a request, it will attempt to recycle an ActionForm instance from either the request or the session, depending on the scope that the

ActionForm has in the action element. If no instance is found, a new instance will be created.

Creating an ActionForm

The ActionForm class provided by the Struts framework is abstract. You need to create subclasses of it to capture your application specific form data. Within your subclass, you should define a property for each field that you wish to capture from the HTML form.

The ActionForm is populated from request parameters, not request attributes. If you are forwarding from one action to anther, you can'tadd a request attribute and expect that the ActionForm will be populated from it. Request parameters and request attributes are two separate resources.

The ActionForm validate() Method

The validate() method may be called by the RequestProcessor for every request. Whether it's called or not depends on two things. First, an ActionForm must be configured for an action mapping. This means that the name attribute for an action element must correspond to the name attribute of one of the form-bean elements in the configuration file.

The second condition that must be met before the RequestProcessor will invoke the validate() method is that the validate attribute must have a value of "true".

```
<action
path="/signin" type=" LoginAction" scope="request" name="loginForm"
validate="true" input="/security/signin.jsp">
<forward name="Success" path="/index.jsp" redirect="true"/>
<forward name="Failure" path="/security/signin.jsp" redirect="true"/>
</action>
```

When the signin action is invoked, the framework will populate an instance of a LoginForm using values it finds in the request. Because the validate attribute has a value of "true", the validate() method in the LoginForm will be called. Even if the validate attribute is set to "false", the ActionForm will still be populated from the request if an ActionForm is configured for the action.

The validate() method in the base ActionForm class simply returns null. If you want to perform validation on the data that is submitted with the request, you'll need to override the validate() method in your ActionForm subclasses.

The validate() method may return an ActionErrors object, depending on whether or not any validation errors were detected. You also can return null if there are no errors; the framework will check for both null and an empty ActionErrors object. This saves you from having to create an instance of ActionErrors when there are no errors.

The ActionForm reset() Method

The reset() method has been a bane for much of the Struts user community at one time or another. Exactly when the reset() method is called and what should be done within it is almost always misinterpreted. This doesn't mean that one implementation is more correct than another, but there are misconceptions that many new Struts developers pick up and then have a hard time shaking regarding the reset().

It's called before the ActionForm has been populated from the request. The method was added to the ActionForm class originally to help facilitate resetting boolean properties back to their defaults. To understand why they need to be reset, it's helpful to know how the browser and the HTML form submit operation processes checkboxes. When an HTML form contains checkboxes, only the values for the checkboxes that are checked are sent in the request. Those that are not checked are not included as a request parameter.

Therefore, the reset() method was added to allow applications to reset the boolean properties in the ActionForm back to false, since false wasn't included in the request and the boolean values would possibly be stuck in the "true" state. The reset() method in the base ActionForm contains no default behavior, since no properties are defined in this abstract class. Applications that extend the ActionForm class are allowed to override this method and reset the ActionForm properties to whatever state they wish. This may include setting boolean properties to true or false, setting String values to null or some initialized value, or even instantiating instances of other objects that the ActionForm holds on to.

For an ActionForm that has been configured with request scope, the framework will essentially create a new instance for each new request. Since a new instance is created, there's not much need to reset() the values back to any default state. ActionForms that are configured with session scope are different however. This is the time that the reset() method comes in handy.

Declaring ActionForms in the Struts Configuration File

Once you have created a class that extends ActionForm, you need to configure the class in the Struts configuration file. The first step is to add a new form-bean element to the form-beans section of the file:

```
<form-beans>
<form-bean name="loginForm" type=" LoginForm"/>
</form-beans>
```

The value for the type field must be a fully qualified Java class name that is a descendant of ActionForm. Once you have defined your form-bean, you can now use it in one or more action elements. It's very common to share one ActionForm across several actions.

Declaring ActionForm Properties as Strings

All request parameters that are sent by the browser are Strings. This is true regardless of the type that the value will eventually map to in Java. For example, dates, times, Booleans, and other values are all strings when they are pulled out of the request. They will also be converted into strings when they are written back out to the HTML page.Therefore, it makes sense that all of the ActionForm properties where the input may be invalid, should be of type String. The reason for this is to support displaying the data back out in its original form to the user, when there is an error. For example, if a user types in "12Z" for a property expecting to be an Integer, there's no way to store "12Z" into an int or Integer property. However, you can store it into a String until it can be validated. This same value, which is stored in a String, can be used to render the input field with the value, so the user can see their mistake. This is functionality that even the most inexperienced users have come to expect and look for.

Using ActionErrors

Earlier in the chapter, you saw that the validate() method returned an ActionErrors object. The ActionErrors class encapsulates one or more errors that have been discovered by the application. Each problem discovered is represented by an instance of org.apache.struts.action.ActionError. An ActionErrors object has request scope. Once an instance is created and populated by the validate() method, it is stored into the request. Later, the JSP page can retrieve the object from the request and use the ActionError objects contained within it to display errors messages to the user.

An instance of ActionErrors can be instantiated in the validate() method and populated by adding instances of the ActionError class to it.

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request){
ActionErrors errors = new ActionErrors();
if( getEmail() == null || getEmail().length() < 1 ){
errors.add("email", new ActionError("security.error.email.required"));
}
if( getPassword() == null || getPassword().length() < 1 ){
errors.add("password",new ActionError("security.error.password.required"));
}
return errors;
}
```

The validate() method in this fragment checks to make sure that the email and password fields have been set with values other than an empty string. If not, ActionError objects are added to the ActionErrors instance.
The ActionError class contains several useful constructors.
Several are listed here:

```
public ActionError(String key);
```

public ActionError(String key, Object value0);
public ActionError(String key, Object value0, Object value1);
public ActionError(String key, Object[] values);

The key argument is a String value that corresponds to a key from one of the application's resource bundles. The custom tag ErrorsTag uses this value to lookup the message to display to the user. The remaining arguments are used as parametric replacement values for the message. For example, if you had a bundle message defined like this:

global.error.login.requiredfield=The {0} field is required for login
then we could create an instance of an ActionError like this:
ActionError error = new ActionError("global.error.login.requiredfield", "Email");

The message displayed to the user after substituting in the "Email" string would be:
The Email field is required for login

When adding instances of the ActionError class to the ActionErrors object, the first argument in the add() method is a property that can be used to retrieve a specific ActionError instance. If all of your ActionError instances can be treated the same and you have no need to retrieve them individually, you can use the constant ActionErrors.GLOBAL_ERROR similar to this:
errors.add(ActionErrors.GLOBAL_ERROR,
new ActionError("security.error.password.required"));

Using Dynamic ActionForms

Using the ActionForm class has many advantages over performing the functionality yourself in the Action class or some set of helper utility classes. Since the behavior that the ActionForm class provides is needed in nearly every web application, as well as many times in the same application, using the framework to perform the work can really reduce the development time and your frustration level. Having stated the benefits of using ActionForms, there are a few very important downsides to using them.

The first and foremost problem with using ActionForms is the sheer number of classes that it can add to a project. Even if you share ActionForm definitions across many pages, the additional classes make it more difficult to manage a project and provide maintenance. This is why some developers might create a single ActionForm and implement the properties for all of the HTML forms within these. The problem with this of course, is that combining the fields into this one class makes it a point of contention on a project that has more than just a few developers. Another major liability is the requirement to define the properties in the ActionForm that need to be captured from the HTML form. If a property is added or removed from the HTML form, the ActionForm class may need to be modified and recompiled. For these reasons, a new type of ActionForm was added to the framework, which is dynamic in nature and allows you to avoid having to create concrete ActionForm classes for your application. The dynamic ActionForm is

implemented by the base class org.apache.struts.action.DynaActionForm, which extends the ActionForm class.

The properties that the ActionForm defines:
1. The validate() method
2. The reset() method

The properties for a DynaActionForm are configured in the Struts configuration file, which you'll see how to do in the next section. The reset() method is called at exactly the same time during request processing as it is for a standard ActionForm. The one difference is that you have a little less control over what you do during the method. However, you can always subclass the DynaActionForm to override the reset behavior.

The validation of the presentation data is a little more complicated, because we'll need to wait until we talk about the Struts Validator components before talking about how validation occurs in a dynamic form.

Configuring Dynamic ActionForm

To use the DynaActionForm in your Struts application, the first step is to add a form-bean element to the configuration file. There are two very important differences between a form-bean element for a regular ActionForm and one that is dynamic. First, a form-bean element for a dynamic ActionForm is required to have an attribute called dynamic, which must have a value of "true". This is necessary for the framework to understand that it should handle this ActionForm differently. The second difference is that you must include one or more form-property elements in order for the dynamic form to have properties. The DynaActionForm uses a java.util.Map internally to store key/value pairs. The form-property elements are loaded into the Map and become the properties that get populated by the framework.

```
<form-beans>
<form-bean name="loginForm" dynamic="true"
type="org.apache.struts.action.DynaActionForm">
<!-- Specify the dynamic properties of the form -->
<form-property name="email" type="java.lang.String "/>
<form-property name="password" type="java.lang.String "/>
<!-- You can also set the initial value of a property -->
<form-property initial="false"
name="rememberMe" type="java.lang.Boolean "/>
</form-bean>
<form-beans>
```

The declarative properties are what make the ActionForm dynamic. At runtime, the framework creates an instance of the DynaActionForm class and makes it possible to set and get the configured property values. To add new properties, you only need to modify the configuration file. No source code needs to be changed. The power and flexibility that this provides for you is immense.

The form-beam element also allows you to specify the initial value for each property. The framework will set the property to that value when the application is started. The initial value is also used when the reset() method is called to reset the values back to their original state. If you don't include the initial attribute, then properties will be assigned default values based on the Java programming language; numbers to zero (0) and properties of type Object will be assigned a null value by the framework.

Performing Validation using Dynamic ActionForms

Since the DynaActionForm is used for every dynamic ActionForm and you don'tprovide subclasses of ActionForm, there's no way to override the validate() method. Fortunately, the framework comes to your aid again with a feature called the Struts Validator. The Struts Validator was created by David Winterfeldt and is now in the main Struts distribution. The validator is a framework that was intended to work with Struts from the beginning. It supports basic validation rules like checking for required fields, email, date and time fields, and many others.

Struts Built-in Actions:

The following are the built-in actions given under the struts framework:

- ✓ ForwardAction
- ✓ IncludeAction
- ✓ SwitchAction
- ✓ DispatchAction

ForwardAction

The **org.apache.struts.actions.ForwardAction** dispatches the request to the given path without performing any action. The ForwardAction is configured if we want our request to be forwarded to the given path may be a JSP, Servlet or any other resource. The ForwardAction must be configured in struts-config.xml.

Syntax:

```
<action path="/mypath"
type="org.apache.struts.actions.ForwardAction" parameter="/MyPage.jsp"/>
                                (OR)
<action path="/mypath" forward="/MyPage.jsp"/>
```

This is the recommended approach.

IncludeAction

The **org.apache.struts.actions.IncludeAction** dispatches the request to the given path using the include option of RequestDispatcher. IncludeAction is same as of ForwardAction but this uses RequestDispatcher include method to dispatch the request to the given path instead of returning ActionForward to the RequestProcessor. The IncludeAction must be configured in struts-config.xml.

Syntax:
<action path="/mypath"
type="org.apache.struts.actions.IncludeAction" parameter="/MyPage.jsp"/>
(OR)
<action path="/mypath" include="/MyPage.jsp"/>
This is the recommended approach.

SwitchAction

The **org.apache.struts.actions.SwitchAction** is used to switch the request from one module to other. The SwitchAction is configured as a normal action;SwitchAction accepts parameters named page and prefix. The page parameter takes the module relative URI beginning with "/" character to which page the request should be forwarded after switching the module.
The prefix parameter takes module name beginning with "/", i.e, to which module the request should be switched.

Configuration in struts-config.xml:

<action path="/changeModule"
type="org.apache.struts.actions.SwitchAction"prefix="/admin"
page="/home.jsp"/>

DispatchAction

The **org.apache.struts.actions.DispatchAction** class allows us to combine set of similar actions into a single Action class, in order to simplify the application design by eliminating the need to create separate action classes for each of the action. This class provides a mechanism for modularizing a set of related actions into a single Action.

The org.apache.struts.actions.DispatchAction class is an abstract class with no abstract methods and is a sub type of BaseAction class, which extends org.apache.struts.action.Action class. The execute() method of DispatchAction dispatches the request to a public method that is named by the request parameter value, the parameter name is specified through parameter attribute of <action> tag in struts-config.xml file.

Steps to use DispatchAction class:

- ✓ Create an action class
- ✓ Configure action mapping

Creating action class:
Subclass the Action class with DispatchAction class and provides a set of methods that will be called by the execute() method of DispatchAction.
Note that this class should not override execute() method like other actions.

Configuring action mapping:

<action path="/mypath" type="com.mk.struts.MyAction" name="myform" parameter="submit" validate="false">
<forward name="success" path="/home.jsp"/>
</action>

Eclipse IDE:

Eclipse is an open-source software framework written primarily in Java. In its default form it is an Integrated Development Environment (IDE) for Java developers, consisting of the Java Development Tools (JDT) and the Eclipse Compiler for Java (ECJ). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs are available for over a dozen languages.

Architecture:
The basis for Eclipse is the Rich Client Platform (RCP). The following components constitute the rich client platform:

- OSGi - a standard bundling framework
- Core platform - boot Eclipse, run plug-ins
- the Standard Widget Toolkit (SWT) - a portable widget toolkit
- JFace - viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors
- the Eclipse Workbench - views, editors, perspectives, wizards

Eclipse's widgets are implemented by a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also leverages an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the rich client platform, in contrast to some other applications where functionality is typically hard coded. This plug-in mechanism is a lightweight software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX, networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK.

The key to the seamless integration of tools with Eclipse is the plug-in. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that a plug-in you develop integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are created equal.

The Eclipse SDK includes the Eclipse Java Development Tools, offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards. The Visual Editor project allows interfaces to be created interactively, hence allowing Eclipse to be used as a RAD tool.

The following is a list of notable projects and plug-in for the Eclipse IDE.

These projects are maintained by the Eclipse community and hosted by the Eclipse Foundation.

1. **Core projects:**

Rich Client Platform (Platform) is the core framework that all other Eclipse projects are built on. Java Development Tools (JDT) provides support for core Java SE. This includes a standalone fast incremental compiler.

2. **Tools projects:**

C/C++ Development Tools (CDT) adds support for C/C++ syntax highlighting, code formatting, and debugger integration and project structures. Unlike the JDT project, the CDT project does not add a compiler and relies on an external tool chain. Graphical Editing Framework (GEF) allows developers to build standalone graphical tools. Example use includes circuit diagram design tools, activity diagram editors and WYSIWYG document editors.

3. **Web projects:**

J2EE Standard Tools (JST) extends the core JDT to include support for Java EE projects. This includes EJBs, JSPs and Servlets.
PHP Development Tools (PDT)
Web Standard Tools (WST) adds standards compliant web development tools. These tools include editors for XML, HTML and CSS.

4. **Modeling projects:**

Eclipse Modeling Framework (EMF) a modeling framework and code generation facility for building tools and other applications based on a structured data model, from a model specification described in XMI.
Graphical Modeling Framework (GMF) is a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF.

5. Other projects:

Test and Performance Tools Platform (TPTP) which provides a platform that allows software developers to build test and performance tools, such as debuggers, profilers and benchmarking applications. Business Intelligence and Reporting Tools Project (BIRT), an Eclipse-based open source reporting system for web applications, especially those based on Java EE.

WebServer/Application Server:

An application server is a software engine that delivers applications to client computers or devices, typically through the Internet and using the Hypertext Transfer Protocol. Application servers are distinguished from web servers by the extensive use of server-side dynamic content and frequent integration with database engines.

Common features:

Application server products typically bundle middleware to enable applications to intercommunicate with dependent applications, like web servers, database management systems, and chart programs. Some application servers also provide an API, making them operating system independent. Portals are a common application server mechanism by which a single point of entry is provided to multiple devices.

Java application servers:

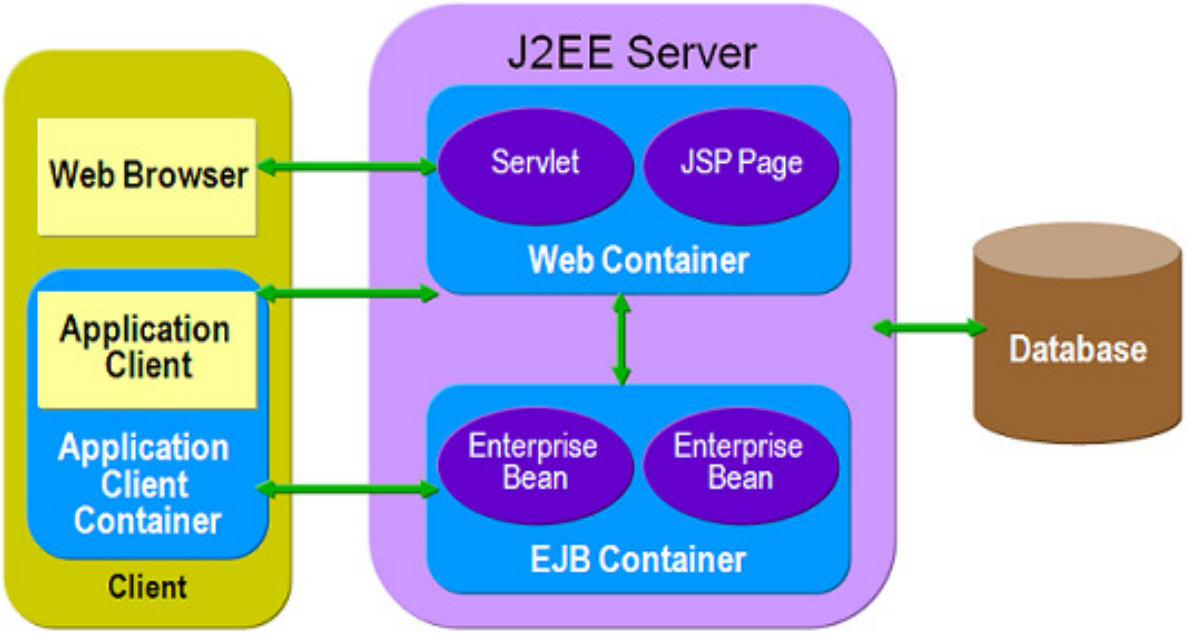


Figure 4.10 : Java Application Server

Java EE Servers:

Following the success of the Java platform, the term application server sometimes refers to a Java Platform--Enterprise Edition (J2EE) or Java EE 5 application server. Among the better known Java Enterprise Edition application servers are WebLogic Server (BEA), JBoss (Red Hat), WebSphere (IBM), JRun (Adobe), Apache Geronimo (Apache Foundation, based on IBM WebSphere), Oracle OC4J (Oracle Corporation), Sun Java System Application Server (Sun Microsystems) and Glassfish Application Server (based on Sun Java System Application Server).

Java application server was the first open source application server to have achieved official compliance with the Java Enterprise Specification. BEA delivered the first Java EE 5 certified application server followed by Sun Microsystems' reference implementation Glassfish.

The Web modules are servlets and Java Server Pages, and business logic is built into Enterprise JavaBeans (EJB-3 and later). The Hibernate project offers an EJB-3 container implementation for the JBoss Application server. Tomcat from Apache and JOnAS from ObjectWeb are typical of containers into which these modules can be put.

A Java Server Page (JSP) is a servlet from Java that executes in a Web container—the Java equivalent of CGI scripts. JSPs are a way to create HTML pages by embedding references to the server logic within the page. HTML coders and Java programmers can work side by side by referencing each other's code from within their own. JavaBeans are the independent class components of the Java architecture from Sun Microsystems.

The application servers mentioned above mainly serve Web applications. Some application servers target networks other than the Web: Session Initiation Protocol servers, for instance, target telephony networks.

Scope of The Development Project

Database Tier: The concentration is applied by adopting the Oracle 8.1 Enterprise versions. SQL is taken as the standard query language. The overall business rules are designed by using the power of PL/SQL components like stored procedures stored functions and database triggers.

User Tier: The use interface is developed is a browser specific environment to have centralized architecture. The components are designed using Dreamweaver and Java server pages power the dynamic of the page design.

Data Base Connectivity Tier
The communication architecture is designed by concentrated on the standards of struts and Java Beans. The database connectivity is established using the Java Database connectivity.

JBOSS

JBoss Application Server (or JBoss AS) is a free software / open source Java EE-based application server. Because it is Java-based, JBoss AS is cross-platform, usable on any operating system that Java supports.

Environment

JBoss AS 4.0 is a J2EE 1.4 application server, with embedded Tomcat 5.5. Any JVM between 1.4 and 1.5 is supported. JBoss can run on numerous operating systems including Windows, Mac OS X, many POSIX platforms, and others, as long as a suitable JVM is present. JBoss AS 4.2 is also a J2EE 1.4 application server, but EJB 3 is deployed by default. It requires JDK 5. Tomcat 6 is bundled with it. Next JBoss AS 5 will be Java EE 5 application server.

Product features

- Failover (including sessions)
- Load balancing
- Distributed caching (using JBoss Cache, a standalone product)
- Distributed deployment (farming)
- Enterprise JavaBeans version 3

Coding

Coding

The final phase of the progress process is the implementation of the new system. This phase is culmination of the previous phases and will be performed only after each of the prior phases has been successfully completed to the satisfaction of both the user and quality assurance. The tasks, comprise the implementation phase, include the installation of hardware, proper scheduling of resources needed to put the system in to introduction, a complete of instruction that support both the users and its environment.

Below code is the sample code for virtual classroom system:

Index.jsp:

```
<%-- tpl:insert page="/theme/VCSTemplate.jtpl" --%><!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@page import="java.util.Date"%>
<%@page import="java.text.DateFormat"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="com.ignou.vcs.commons.beans.UserBean"%>
<%@page
import="com.ignou.vcs.commons.database.CommonsDatabaseActivities"%>
<html:html>
<head>
<script type="text/javascript" language="javascript" >
function formValidator()
{
    var searchString = window.document.searchForm.search;
    if(searchString.value.length == 0)
    {
        alert("Enter the search query");
        return false;
    }
    return true;

}
function loadCss() {
    var browser = navigator.appName.toLowerCase();
    // document.write(browser);
    var stylesheet = document.getElementById("pagestyle");
    var menusheet = document.getElementById("menustyle");
    if(browser.indexOf("microsoft internet explorer") != -1) {
```

```
        stylesheet.href="${pageContext.request.contextPath}/theme/css/style_ie.cs
s";

        menusheet.href="${pageContext.request.contextPath}/theme/css/menu_ie.
css";
        }
        else {

        stylesheet.href="${pageContext.request.contextPath}/theme/css/style1.css"
;

        menusheet.href="${pageContext.request.contextPath}/theme/css/menu.css
";
        }
}
</script>
<link              id="pagestyle"              type="text/css"              rel="stylesheet"
href="${pageContext.request.contextPath}/theme/css/style1.css" />
<link              id="menustyle"              type="text/css"              rel="stylesheet"
href="${pageContext.request.contextPath}/theme/css/menu.css" />
<script                                                                    type="text/javascript"
src="${pageContext.request.contextPath}/theme/js/transmenu_Packed.js"></scri
pt>

<!-- LightBox css and scripts -->
<%

//response.sendRedirect("http://localhost:8080/VCS/UnderMaintainance.jsp");


        String usid = (String) request.getSession().getAttribute("userId");
        if(usid != null) {
 %>
        <link                                                                    rel="stylesheet"
href="${pageContext.request.contextPath}/theme/css/lightbox_vid.css"
media="screen,projection" type="text/css" />
<script                                                                    type="text/javascript"
src="${pageContext.request.contextPath}/theme/js/prototype.js"></script>
<script                                                                    type="text/javascript"
src="${pageContext.request.contextPath}/theme/js/lightbox.js"></script>


 <%
 }

 %>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="GENERATOR" content="Rational Application Developer">
<%-- tpl:put name="headarea" --%>
<title>Virtual Classroom System</title>
<link rel="stylesheet"
href="${pageContext.request.contextPath}/theme/css/lightbox.css"
media="screen,projection" type="text/css" />
<script type="text/javascript"
src="${pageContext.request.contextPath}/theme/js/prototype.js"></script>
<script type="text/javascript"
src="${pageContext.request.contextPath}/theme/js/lightbox.js"></script>
<%-- /tpl:put --%>
</head>
<body onLoad="javascript:loadCss()">
<%
	String userIDForName = (String)
request.getSession().getAttribute("userId");
	System.out.println("UserId: " + userIDForName);
	String userName = "";
	String userLevel = "";
	CommonsDatabaseActivities dbObjectForName = new
CommonsDatabaseActivities();
	if (userIDForName != null ) {

		UserBean userBeanForName =
(UserBean)dbObjectForName.getUserInfo(userIDForName) ;
		userName = userBeanForName.getName();
		userLevel = userBeanForName.getLevel();
		System.out.println("UserNAME: " + userName);
		System.out.println("UserLEVEL: " + userLevel);
	}
 %>
	<div id="content">
		<div id="header">
			<%
				if(userIDForName == null) {
			 %>
				<%@ include file="/theme/right_links/default.jsp" %>

			<%
				} else {
					String link =
"/theme/right_links/logged_in.jsp?userName=" + userName;
			%>
				<jsp:include page="<%= link %>" />
			<%
				}
```

```
                %>
                <%-- tpl:put name="top_menu_right" --%>

            <%-- /tpl:put --%>
                <div id="logo">
                <%-- tpl:put name="logo_image" --%>
                    <img
src="${pageContext.request.contextPath}/theme/images/vcs.jpg"       height="71"
width="190">
                <%-- /tpl:put --%>
                </div>
            </div>

            <div id="tabs">
                <%
                    if(userLevel.equals("0")) {
                %>
                    <%@                                                 include
file="/theme/main_menu/student_menu.jsp" %>
                <%
                    } else if (userLevel.equals("1")){
                %>
                    <%@                                                 include
file="/theme/main_menu/faculty_menu.jsp" %>
                <%
                    } else if (userLevel.equals("2")){
                %>
                    <%@                                                 include
file="/theme/main_menu/management_menu.jsp" %>
                <%
                    } else if (userLevel.equals("3")){
                %>
                    <%@ include file="/theme/main_menu/admin_menu.jsp"
%>
                <%
                    } else if (userLevel.equals("")){
                %>
                    <%@ include file="/theme/main_menu/default.jsp" %>
                <%
                    }
                %>

                <%-- tpl:put name="top_menu_middle" --%>

                <%-- /tpl:put --%>

                <div id="search">
```

```
                        <form method="post" action = "${pageContext.request.contextPath}/search/search.jsp" onsubmit = "return formValidator()" name = "searchForm">
                        <p>
                                <input type="text" name="search" class="search"/>
                                <input type="submit" value="Search" class="button" />

                        </p>
                        </form>
                    </div>
                </div>


                <div class="left">
                    <div class="left_articles">
                        <div class="buttons">
                        <%-- tpl:put name="buttons_blue_green" --%>

                        <%-- /tpl:put --%>
                        </div>
                        <%
                                Date date = new Date();
                                DateFormat formatterMonth = new SimpleDateFormat("MMM");
                                DateFormat formatterDay = new SimpleDateFormat("dd");

                                String month = formatterMonth.format(date);
                                String day = formatterDay.format(date);
                                month = month.toUpperCase();

                                if(day.equals("1") || day.equals("21") || day.equals("31")) {
                                        day = day + "st";
                                } else if(day.equals("2") || day.equals("22")) {
                                        day = day + "nd";

                                } else if(day.equals("3") || day.equals("23")) {
                                        day = day + "rd";

                                } else {
                                        day = day + "th";
```

```
                }

                %>
            <%-- tpl:put name="calendar" --%>
            <div class="calendar">
                <p><%= month %><br /><%= day %></p>
            </div>
            <%-- /tpl:put --%>
            <%-- tpl:put name="centre_heading" --%>
            <h2><a href="index.jsp"><u>Virtual Classroom System</u></a></h2>
            <p class="description">Studying the e-way.</p>
            <%-- /tpl:put --%>
            <%-- tpl:put name="centre_content" --%>
            <p>
                <img src="${pageContext.request.contextPath}/theme/images/thumb1.jpg" class="thumbnail" alt="thumbnail" />

                VCS (Virtual Classroom System) aims to promote a greater count of students to splurge into the field of Education.
                It integrates the benefits of a physical classroom with the convenience of a "no-physical-bar" virtual learning environment, minus the commuting hazards and expenses.
                It will usher in the immense flexibility and sophistication in the existing learning platform structures, with the perfect blend of synchronous and asynchronous interaction.
                It provides a means of collaborative learning for the students.
                <br />
            </p>

            <%-- /tpl:put --%>
        </div>


        <%-- tpl:put name="bottom_box" --%>

        <div class="thirds">
            <div class="smallboxtop"></div>
            <div class="smallbox">
                <p>
                    <b><u>Coming Soon...</u></b><br />
                    <a href="#" accesskey="m"><span class="key">I</span>nteractive White Boards</a><br />
                    <a href="#" accesskey="m"><span class="key">V</span>ideo Conference with friends, Faculties</a><br />
```

```
                                    <a href="#" accesskey="m"><span
class="key">V</span>oice Mailboxes</a><br /><br>
                                </p>
                            </div>
                        </div>

                        <div class="thirds">
                            <div class="smallboxtop"></div>
                            <div class="smallbox">
                                <p><u><b>Important Links</b></u><br>
                                    <a href =
"${pageContext.request.contextPath}/faculty_registration.jsp">Become a
Faculty.</a><br>
                                    <a href
="${pageContext.request.contextPath}/faculty_registration.jsp">Become a
Manager</a> <br>
                                    <a href = "#">Bookmark Virtual Classroom
Systems.</a><br>
                                    <a href =
"http://localhost:8080/VCS/player3_vid.jsp?filePath=http://localhost:8080/VCS/DA
TA/lectures/48.flv"  class=lbOn>Demo Course.</a>

                                </p>
                            </div>
                        </div>

                        <div class="thirds">
                            <div class="smallboxtop"></div>
                            <div class="smallbox">
                                <p><b><u>Contacts</u></b><br>
                                    <a href =
"contacts/GeneralInformation.jsp">General Information</a><br>
                                    <a href =
"contacts/GeneralInformation.jsp">Students Enquiries</a> <br>
                                    <a href = "contacts/techques.jsp">Websites
Technical Questions</a><br>
                                    <a href =
"contacts/GeneralInformation.jsp">Faculty Enquiry</a>
                                </p>
                            </div>
                        </div>
                        <%-- /tpl:put --%>
                    </div>


                    <div id="right">
                        <%-- tpl:put name="right_boxes" --%>
                        <div class="boxtop"></div>
```

```
                    <div class="box">
                        <p>
                            <b><u>News</u></b><br />
                            1.  <a  href="news/AdobeSummit.jsp"
accesskey="m">New Java Courses launched. </a><br />
                            2.  <a  href="news/AdobeSummit.jsp"
accesskey="m">Exclusive video training offered for all adobe courses.</a><br />
                            3.  <a  href="news/SunTechDays.jsp"
accesskey="m">Sun Tech days here back again for 2011.</a><br />
                            4.  <a  href="news/AdobeSummit.jsp"
accesskey="m">Watch out this space for more up coming courses.</a><br />
                            5.  <a  href="news/AdobeSummit.jsp"
accesskey="m">Ansca Mobile offers special subscription pricing for current
students and educators. </a><br />
                            6.  <a  href="news/AdobeSummit.jsp"
accesskey="m">Adobe is back again with Dev Summit on mobile platform.</a><br
/>
                        </p>
                        <div  class="buttons"><p><a  href="#"
class="bluebtn"><span>More</span></a></p></div>
                    </div>

                    <!--  <div class="boxtop"></div>
                    <div class="box">
                        <p>
                            <b><u>Coming Soon...</u></b><br />
                            <a  href="#"  accesskey="m"><span
class="key">I</span>nteractive White Boards</a><br />
                            <a  href="#"  accesskey="m"><span
class="key">C</span>hat with friends, Faculties</a><br />
                            <a  href="#"  accesskey="m"><span
class="key">V</span>oice Mailboxes</a><br />
                        </p>
                        <div  class="buttons"><p><a
href="http://localhost:8080/login.jsp"
class="bluebtn"><span>Enter..</span></a></p></div>
                    </div>-->
                    <%-- /tpl:put --%>
                </div>
                <div class="footer">

                    <%-- tpl:put name="footer" --%>
                    <p>
                        <a href="mailbox/index.html">About Us</a> &middot;
                        <a  href="activitiesIndex.jsp">Activities  at  VCS</a>
&middot;
                        <a href="#">Members</a> &middot;
                        <a href="#">Contact Us</a> &middot;
```

```
                        <br/>
                    &copy; Copyright 2011 Virtual Classroom System</p>
                    <%-- /tpl:put --%>
                </div>
        </div>
</body>
</html:html>
<%-- /tpl:insert --%>
```

Home.jsp:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@page import="java.util.Date"%>
<%@page import="java.text.DateFormat"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="com.ignou.vcs.commons.beans.UserBean"%>
<%@page
import="com.ignou.vcs.commons.database.CommonsDatabaseActivities"%>
<html:html>
<head>
<script type="text/javascript" language="javascript">
function loadCss() {
        var browser = navigator.appName.toLowerCase();
        // document.write(browser);
        var stylesheet = document.getElementById("pagestyle");
        var menusheet = document.getElementById("menustyle");
        if(browser.indexOf("microsoft internet explorer") != -1) {

        stylesheet.href="${pageContext.request.contextPath}/theme/css/style_ie2.c
ss";

        menusheet.href="${pageContext.request.contextPath}/theme/css/menu_ie.
css";
        }
        else {

        stylesheet.href="${pageContext.request.contextPath}/theme/css/style3.css"
;

        menusheet.href="${pageContext.request.contextPath}/theme/css/menu.css
";
        }
```

```
}
</script>
<script src="${pageContext.request.contextPath}/theme/js/player/flowplayer-3.0.3.min.js"></script>
<script language="JavaScript">
	flowplayer("player", "${pageContext.request.contextPath}/theme/player_swf/flowplayer-3.0.3.swf");
</script>

<link rel="shortcut icon" href="${pageContext.request.contextPath}/theme/images/logo_small.JPG" type="image/x-icon">

<script src="${pageContext.request.contextPath}/theme/js/ajax.js"></script>
<link id="pagestyle" type="text/css" rel="stylesheet" href="${pageContext.request.contextPath}/theme/css/style3.css">
<link id="menustyle" type="text/css" rel="stylesheet" href="${pageContext.request.contextPath}/theme/css/menu.css">
<script type="text/javascript" src="${pageContext.request.contextPath}/theme/js/transmenu_Packed.js"></script>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="GENERATOR" content="Rational Application Developer"><title>Virtual Classroom System</title></head>
<body onload="javascript:loadCss()">
<%
	String userIDForName = (String) request.getSession().getAttribute("userId");
	System.out.println("UserId: " + userIDForName);
	String userName = "";
	String userLevel = "";
	CommonsDatabaseActivities dbObjectForName = new CommonsDatabaseActivities();
	if (userIDForName != null ) {

		UserBean userBeanForName = (UserBean)dbObjectForName.getUserInfo(userIDForName) ;
		userName = userBeanForName.getName();
		userLevel = userBeanForName.getLevel();
		System.out.println("UserNAME: " + userName);
		System.out.println("UserLEVEL: " + userLevel);
	}
 %>
	<div id="content">
		<div id="header">
			<%
				if(userIDForName == null) {
			 %>
```

```
                        <%@include file="/theme/right_links/default.jsp"%>
                    <%
                        } else {
                            String link = "/theme/right_links/logged_in.jsp?userName=" + userName;
                    %>
                        <jsp:include page="<%= link %>" />
                    <%
                        }
                     %>

                    <div id="logo">
                        <img src="${pageContext.request.contextPath}/theme/images/vcs1.jpg" height="71" width="190">
                    </div>
                </div>

                <div id="tabs">
                    <%
                        if(userLevel.equals("0")) {
                     %>
                        <%@include file="/theme/main_menu/student_menu.jsp"%>
                    <%
                        } else if (userLevel.equals("1")){
                    %>
                        <%@include file="/theme/main_menu/faculty_menu.jsp"%>
                    <%
                        } else if (userLevel.equals("2")){
                    %>
                        <%@include file="/theme/main_menu/management_menu.jsp"%>
                    <%
                        } else if (userLevel.equals("3")){
                    %>
                        <%@include file="/theme/main_menu/admin_menu.jsp"%>
                    <%
                        } else if (userLevel.equals("")){
                    %>
                        <%@include file="/theme/main_menu/default.jsp"%>
                    <%
                        }
                     %>

                                <div id="search">
```

```
                        <form                                    method="post"
action="${pageContext.request.contextPath}/search/search.jsp"    onsubmit    =
"return formValidator()" name = "searchForm">
                        <p>
                              <input type="text" name="search" class="search"
size = "15">
                              <input       type="submit"       value="Search"
class="button">

                        </p>
                        </form>
                  </div>

            </div>


            <div class="left">
                  <div class="left_articles">

                        <%
                                    Date date = new Date();
                                    DateFormat    formatterMonth    =    new
SimpleDateFormat("MMM");
                                    DateFormat    formatterDay    =    new
SimpleDateFormat("dd");

                                    String              month              =
formatterMonth.format(date);
                                    String day = formatterDay.format(date);
                                    month = month.toUpperCase();

                                    if(day.equals("1")  ||  day.equals("21")  ||
day.equals("31")) {
                                          day = day + "st";
                                    }     else     if(day.equals("2")     ||
day.equals("22")) {
                                          day = day + "nd";

                                    }     else     if(day.equals("3")     ||
day.equals("23")) {
                                          day = day + "rd";

                                    } else {
                                          day = day + "th";

                                    }

                        %>
```

```
                        <div class="calendar">
                                <p><%= month %><br><%= day %></p>
                        </div>
                        <h2><a          href="#"><u>Virtual          Classroom
System</u></a></h2>
                        <p class="description">Studying the e-way.</p>
                        <div id="id">
                        <table>

                        <%
                                if(userLevel.equals("0")) {
                        %>
                        <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/furl_48.png"
class="thumbnail">
                                <a
href="http://localhost:8080/VCS/files/view_files.jsp?fileType=0"
accesskey="m""><b><u>View General Files</u></b></a>
                                <p>View all the relevant general files.</p></td>
                        </tr>
                        <tr>
                                <td><a
href="${pageContext.request.contextPath}/profile/profile.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/user_48.png"
class="thumbnail">
                                <b><u>My Profile</u></b></a>
                                <p>View or edit your profile.</p></td>
                        </tr>
                        <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/book_48.png"
class="thumbnail">
                                <a
href="${pageContext.request.contextPath}/student/viewSyllabus.jsp"><b><u>My
Subjects</u></b></a>
                                <p>Know about your subjects and their respective
syllabus.</p></td>
                        </tr>
                        <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/globe_48.png"
class="thumbnail">
                                <a
href="${pageContext.request.contextPath}/student/explore.jsp"><b><u>Suggest
ed Reading</u></b></a>
                                <p>Time to gear up your learning process in a
more innovative manner.</p></td>
```

```
                        </tr>
                        <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/yahoo_48.png"
class="thumbnail">
                                <a
href="${pageContext.request.contextPath}/student/fun.jsp"><b><u>Fun@VCS</
u></b></a>
                                <p>Its fun time at VCS..enjoy the
stride..!!</p></td>
                        </tr>
                        <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/table_green_48.pn
g" class="thumbnail">
                                <a
href="${pageContext.request.contextPath}/student/generateReport.jsp"><b><u>
My Report Card</u></b></a>
                                <p>My Report Card</p></td>
                        </tr>
                        <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/blue_speech_bubbl
e_48.png" class="thumbnail">
                                <a
href="${pageContext.request.contextPath}/skin_multilanguage/login.jsp"><b><u
>Chat</u></b></a>
                                <p>Chat with Friends, Faculty and
Management.</p></td>
                        </tr>


                        <%
                                } else if (userLevel.equals("1")) {
                         %>
                         <tr>
                                <td><img
src="${pageContext.request.contextPath}/theme/images_links/furl_48.png"
class="thumbnail">
                                <a
href="http://localhost:8080/VCS/files/view_files.jsp?fileType=0"
accesskey="m""><b><u>View General Files</u></b></a>
                                <p>View all relevant General Files.</p></td>
                        </tr>

                        <tr>
```

```
                                    <td><img
src="${pageContext.request.contextPath}/theme/images_links/user_48.png"
class="thumbnail">
                                    <a
href="${pageContext.request.contextPath}/profile/profile.jsp"><b><u>My
Profile</u></b></a>
                                    <p>View or edit your profile.</p></td>
                                </tr>
                                <tr>
                                    <td><img
src="${pageContext.request.contextPath}/theme/images_links/book_48.png"
class="thumbnail">
                                    <a
href="http://localhost:8080/VCS/faculty/course/course_faculty.jsp"><b><u>My
Courses</u></b></a>
                                    <p>Check the syllabus of your Subject in various
courses that you teach.</p></td>
                                </tr>
                                <tr>
                                    <td><img
src="${pageContext.request.contextPath}/theme/images_links/navigate_48.png"
class="thumbnail">
                                    <a
href="http://localhost:8080/VCS/help_manual/Help_Manual.html"    target="_new"
accesskey="m"><b><u>Help Manual</u></b></a>
                                    <p>Complete Help at your finger-tips.</p></td>
                                </tr>
                                <tr>
                                    <td><img
src="${pageContext.request.contextPath}/theme/images_links/questionmark_48.p
ng" class="thumbnail">
                                    <a
href="http://localhost:8080/VCS/faculty/view_news.jsp"><b><u>News          @
VCS</u></b></a>
                                    <p>Keep yourself updated with latest news from
vcs.</p></td>
                                </tr>
                                <tr>
                                    <td><img
src="${pageContext.request.contextPath}/theme/images_links/newspaper_48.png
" class="thumbnail">

                                    <a
href="${pageContext.request.contextPath}/faculty/booksJournals.jsp"><b><u>Eb
ooks/Journals</u></b></a>
                                    <p>Enhance    your    knowledge    and    General
Awareness here!!</p></td>
                                </tr>
```

```
                        <%
                                } else if (userLevel.equals("2")) {
                         %>
                         <tr>
                                <td><a
href="${pageContext.request.contextPath}/profile/profile.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/user_48.png"
class="thumbnail">
                                <b><u>My Profile</u></b></a>
                                <p>View or edit your profile.</p></td>
                        </tr>

                        <tr>
                                <td><a
href="${pageContext.request.contextPath}/management/viewCourses.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/users_two_48.png"
class="thumbnail">
                                <b><u>View Students</u></b></a>
                                <p>View all the students and their profiles who are
taking education at VCS</p></td>
                        </tr>

                        <tr>
                                <td><a
href="http://localhost:8080/VCS/faculty/view_news.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/newspaper_48.png
" class="thumbnail">
                                <b><u>News @ VCS</u></b></a>
                                <p>Get yourself equipped with all the latest
happenings at VCS !!!</p></td>
                        </tr>
                        <tr>
                                <td><a
href="http://localhost:8080/VCS/administrator/view_faculties.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/table_48.png"
class="thumbnail">
                                <b><u>View Faculties</u></b></a>
                                <p>View and Manage faculties here and the
courses and subjects alloted to them.Also view students feedback for their
faculties.</p></td>
                        </tr>
                        <%
                                } else if (userLevel.equals("3")) {
                         %>
                         <tr>
                                <td><a
href="${pageContext.request.contextPath}/profile/profile.jsp"><img
```

```
src="${pageContext.request.contextPath}/theme/images_links/user_48.png"
class="thumbnail">
                                <b><u>My Profile</u></b></a>
                                <p>View or edit your profile.</p></td>
                        </tr>
                        <tr>
                                <td><a
href="http://localhost:8080/VCS/administrator/archive_users.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/mail_write_48.png"
class="thumbnail">
                                <b><u>Archive Users</u></b></a>
                                <p>Archive Users that have completed their
course.</p></td>
                        </tr>
                        <tr>
                                <td><!--<a
href="http://localhost:8080/VCS/administrator/backup_db_form.jsp">--><img
src="${pageContext.request.contextPath}/theme/images_links/app_48.png"
class="thumbnail">
                                <b><u>Backup Database</u></b></a>
                                <p>Take Backup of VCS Database.</p></td>
                        </tr>
                        <tr>
                                <td><a
href="http://localhost:8080/VCS/faculty/view_news.jsp"><img
src="${pageContext.request.contextPath}/theme/images_links/questionmark_48.p
ng" class="thumbnail">
                                <b><u>News @ VCS</u></b></a>
                                <p>Get yourself equipped with the latest
hapenings @ VCS.</p></td>
                        </tr>
<!--                    <tr>-->
<!--                            <td><a
href="http://localhost:8080/VCS/admin/"><img
src="${pageContext.request.contextPath}/theme/images_links/questionmark_48.p
ng" class="thumbnail">-->
<!--                            <b><u>Manage Discussion</u></b></a>-->
<!--                            <p>Discuss with fa.</p></td>-->
<!--                    </tr>-->


                        <%
                                }

                        %>
                        </table>
                        </div>
                        </div>
```

```
		</div>


		<div id="right">
		<%
		if((userLevel.equals("0")) || (userLevel.equals("1")))
		{
		%>
			<div class="boxtop"></div>
<!--			<div class="box">-->
<!--				<p>-->
<!--					<%@ include file = "top_box.jsp" %>-->
<!--				</p>-->
<!--			</div>-->
		<%}
		else
		{
		 %>
<!--			<div class="boxtop"></div>-->
			<div class="box">
				<p>
					<b><u>News</u></b><br />
					<a href="#" accesskey="m">IIIrd Standard students to undergo 2 more major tests prior to module V. </a><br />
					<a href="#" accesskey="m">Vth Standard Major Test Evaluated. Check your Progress.</a><br />
					<a href="#" accesskey="m">Annual Programming contest to begin from 15th November 2008.</a><br />
					<a href="#" accesskey="m">IOPC (IIT Kanpur) starting 21st November Check http://www.iitk.ac.in for more dtails.</a><br />
				</p>
				<div class="buttons"><p><a href="#" class="bluebtn"><span>More</span></a></p></div>
			</div>

		<%}
		if((userLevel.equals("0")) || (userLevel.equals("1")) || (userLevel.equals("2")))
		{
		 %>

			<div class="boxtop"></div>
			<div class="box">
				<p>
					<%@ include file = "bottom_box.jsp" %>
				</p>
			</div>
```

```
<%}
else
{
 %>

<%} %>
</div>
</div>
<div class="footer">
	<p>
		<a href="#">About Us</a> &middot;
		<a href="#">Activities at VCS</a> &middot;
		<a href="#">Members</a> &middot;
		<a href="#">Contact Us</a> &middot;
		<br>
	Best viewed in Mozilla Firefox 2.0 and later. Designed and developed by <b>Pradeepthi S</b></b></p>
	</div>

</body>
</html:html>
```

Login.jsp:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@page import="java.util.Date"%>
<%@page import="java.text.DateFormat"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="com.ignou.vcs.commons.beans.UserBean"%>
<html:html>
<head>
<script type="text/javascript" language="javascript">
function loadCss() {
	var browser = navigator.appName.toLowerCase();
	// document.write(browser);
	var stylesheet = document.getElementById("pagestyle");
	var menusheet = document.getElementById("menustyle");
	if(browser.indexOf("microsoft internet explorer") != -1) {

	stylesheet.href="${pageContext.request.contextPath}/theme/css/style_ie2.css";
```

```
        menusheet.href="${pageContext.request.contextPath}/theme/css/menu_ie.
css";
        }
        else {

        stylesheet.href="${pageContext.request.contextPath}/theme/css/style3.css"
;

        menusheet.href="${pageContext.request.contextPath}/theme/css/menu.css
";
        }
}
</script>
<script language="javascript">
        function userfocus() {
                document.fm.username.focus();
        }

        function check() {
                var u, p;
                u = document.fm.username.value;
                p = document.fm.password.value;
                if (u.length < 1) {
                        alert("You must enter User Name");
                        document.fm.username.focus();
                        return false;
                } else if (p.length < 1) {
                        alert("You must enter Password");
                        document.fm.password.focus();
                        return false;
                }
        }
</script>
<script       src="${pageContext.request.contextPath}/theme/js/player/flowplayer-
3.0.3.min.js"></script>
<script language="JavaScript">
        flowplayer("player",
"${pageContext.request.contextPath}/theme/player_swf/flowplayer-3.0.3.swf");
</script>

<link                                rel="shortcut                                icon"
href="${pageContext.request.contextPath}/theme/images/logo_small.JPG"
type="image/x-icon">

<script src="${pageContext.request.contextPath}/theme/js/ajax.js"></script>
<link              id="pagestyle"              type="text/css"              rel="stylesheet"
href="${pageContext.request.contextPath}/theme/css/style3.css">
```

```
<link id="menustyle" type="text/css" rel="stylesheet"
href="${pageContext.request.contextPath}/theme/css/menu.css">
<script type="text/javascript"
src="${pageContext.request.contextPath}/theme/js/transmenu_Packed.js"></scri
pt>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="GENERATOR" content="Rational Application
Developer"><title>Virtual Classroom System</title></head>
<body onload="javascript:loadCss(),userfocus()">
    <div id="content">
        <div id="header">
            <%@include file="/theme/right_links/default.jsp"%>

            <div id="logo">
                <img
src="${pageContext.request.contextPath}/theme/images/vcs1.jpg" height="71"
width="190">
            </div>
        </div>

        <div id="tabs">
            <%@include file="/theme/main_menu/default.jsp"%>

            <div id="search">
            <form method="post"
action="${pageContext.request.contextPath}/search/search.jsp" onsubmit =
"return formValidator()" name = "searchForm">
            <p>
            <input type="text" name="search" class="search"
size = "15">
            <input type="submit" value="Search"
class="button">

            </p>
            </form>
        </div>

        </div>

        <div class="left">
            <div class="left_articles">

            <%
                Date date = new Date();
                DateFormat formatterMonth = new
SimpleDateFormat("MMM");
```

```
                                    DateFormat formatterDay = new
SimpleDateFormat("dd");

                                    String month =
formatterMonth.format(date);
                                    String day = formatterDay.format(date);
                                    month = month.toUpperCase();

                                    if(day.equals("1") || day.equals("21") ||
day.equals("31")) {
                                        day = day + "st";
                                    } else if(day.equals("2") ||
day.equals("22")) {
                                        day = day + "nd";

                                    } else if(day.equals("3") ||
day.equals("23")) {
                                        day = day + "rd";

                                    } else {
                                        day = day + "th";

                                    }

                                %>
                            <div class="calendar">
                                <p><%= month %><br><%= day %></p>
                            </div>
                            <h2><a href="#"><u>Virtual Classroom
System</u></a></h2>
                            <p class="description">Studying the e-way.</p>
                            <div id="id">
                            <h2><p align="center">Login</p></h2>
                        <center>
                        <FORM method="post" name="fm" action="/verifyLogin"
onsubmit="return check()"><BR>
                        <br>
                        <TABLE align="center">
                            <TBODY>
                                <TR>
                                    <TD>Username</TD>
                                    <TD><INPUT type="text"
name="username" size="20"></TD>
                                </TR>
                                <TR>
                                    <TD>Password</TD>
                                    <TD><INPUT type="password"
name="password" size="20"></TD>
```

```
                        </TR>
                    </TBODY>
                </TABLE>
                <BR>
                <BR>
                <INPUT type="submit" name="Submit" value="Login"> <BR>

                </FORM>
                <BR>

                </center>
                    </div>
                    </div>

                </div>


            <div class="footer">
                <p>
                    <a href="#">About Us</a> &middot;
                    <a href="#">Activities at VCS</a> &middot;
                    <a href="#">Members</a> &middot;
                    <a href="#">Contact Us</a> &middot;
                    <br>
                Best viewed in Mozilla Firefox 2.0 and later. Designed and
developed by <b>Pradeepthi S</b></b></p>
                </div>

</body>
</html:html>
```

AddCourseAction.java:

```
package com.ignou.vcs.actions;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

/**
 * @version 1.0
 * @author Pradeepthi S
```

```
 */
public class AddcourseAction extends Action

{

	public ActionForward execute(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		ActionErrors errors = new ActionErrors();
		ActionForward forward = new ActionForward(); // return value
		com.ignou.vcs.forms.AddCourseForm courseform =
(com.ignou.vcs.forms.AddCourseForm) form;
		String subject = request.getParameter("subjectValues");
		request.getSession().setAttribute("subject", subject);

		try {

			com.ignou.vcs.database.VCSDatabaseActivities dbObj = new
com.ignou.vcs.database.VCSDatabaseActivities();

			String courseid = dbObj.insertCourse(courseform);

			request.getSession().setAttribute("courseid", courseid);

		} catch (Exception e) {

			// Report the error using the appropriate name and ID.
			errors.add("name", new ActionError("id"));

		}

		// If a message is required, save the specified key(s)
		// into the request for use by the <struts:errors> tag.

		if (!errors.isEmpty()) {
			saveErrors(request, errors);

			// Forward control to the appropriate 'failure' URI (change name
as
			// desired)
			// forward = mapping.findForward(failure");

		} else {

			// Forward control to the appropriate 'success' URI (change
name as
			// desired)
```

```
                forward = mapping.findForward("success");

            }

            // Finish with
            return (forward);

        }
}
```

AddSubjectAction.java:

```
package com.ignou.vcs.actions;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

/**
 * @version 1.0
 * @author Pradeepthi S
 */
public class AddSubjectAction extends Action

{

        public ActionForward execute(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

            ActionErrors errors = new ActionErrors();
            ActionForward forward = new ActionForward(); // return value

            try {

                // do something here

            } catch (Exception e) {

                // Report the error using the appropriate name and ID.
                errors.add("name", new ActionError("id"));
```

```
            }

            // If a message is required, save the specified key(s)
            // into the request for use by the <struts:errors> tag.

            if (!errors.isEmpty()) {
                  saveErrors(request, errors);

                  // Forward control to the appropriate 'failure' URI (change name
as
                  // desired)
                  // forward = mapping.findForward(failure");

            } else {

                  // Forward control to the appropriate 'success' URI (change
name as
                  // desired)
                  forward = mapping.findForward("success");
            }

            // Finish with
            return (forward);
}
}
```

Screens

Default View:

Figure 5.1: Virtual Classroom system home page

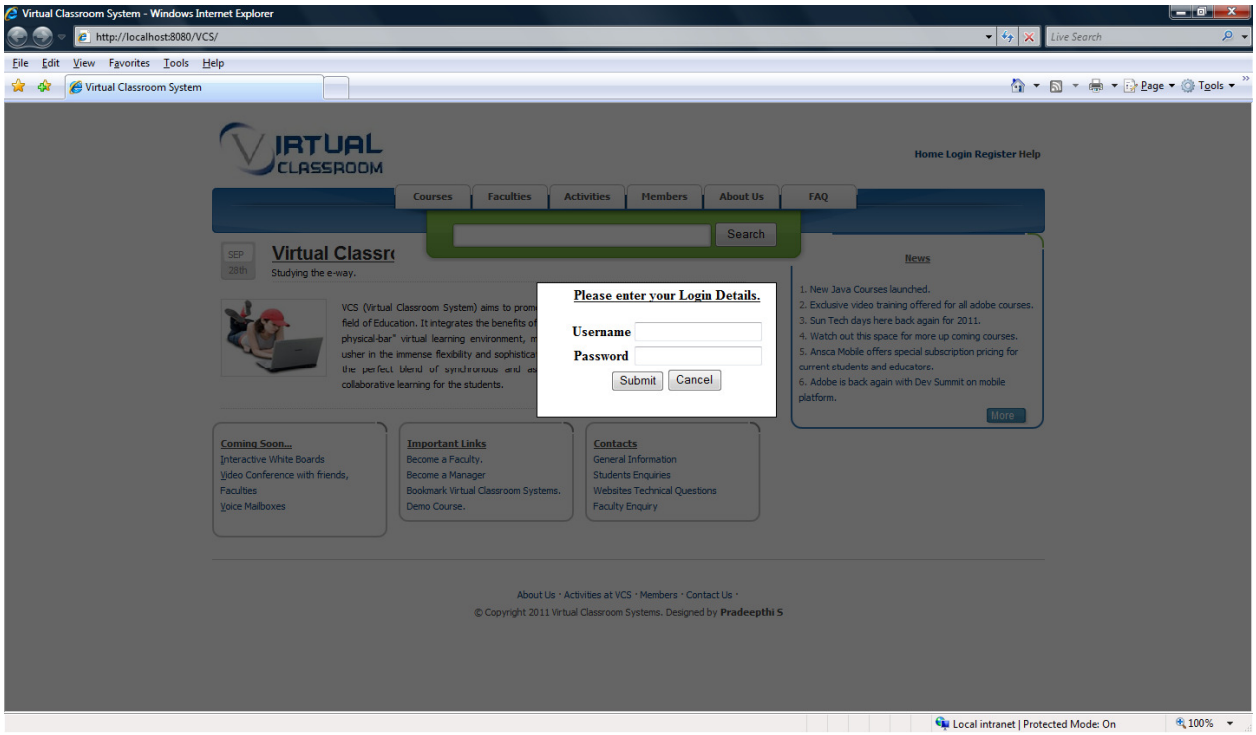


Figure 5.2: Login page

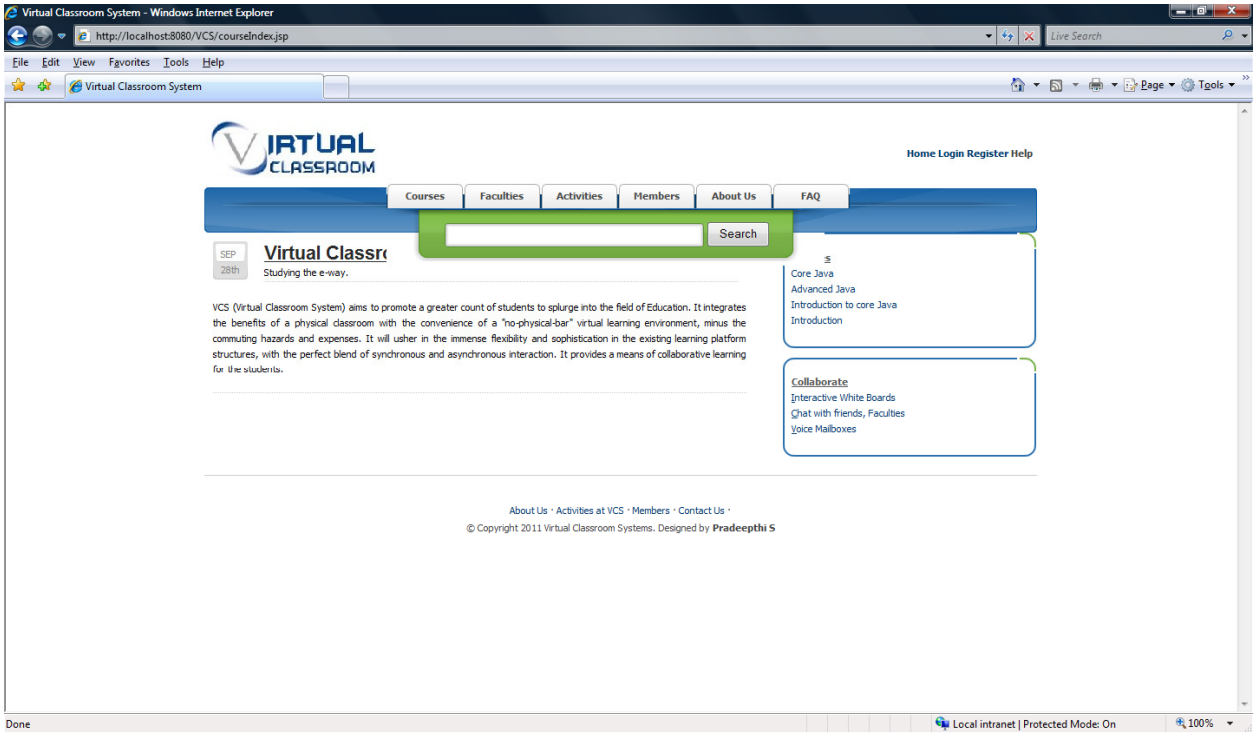


Figure 5.3: List of courses without logging

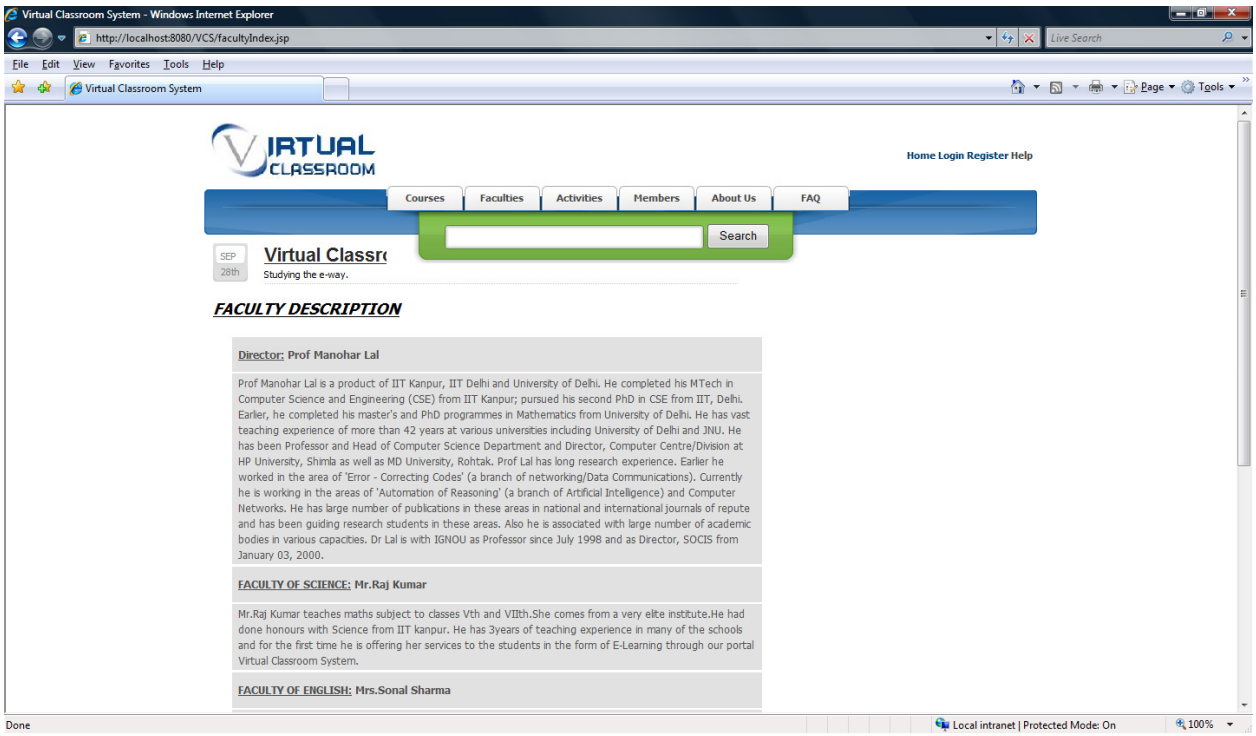


Figure 5.4: Faculty details

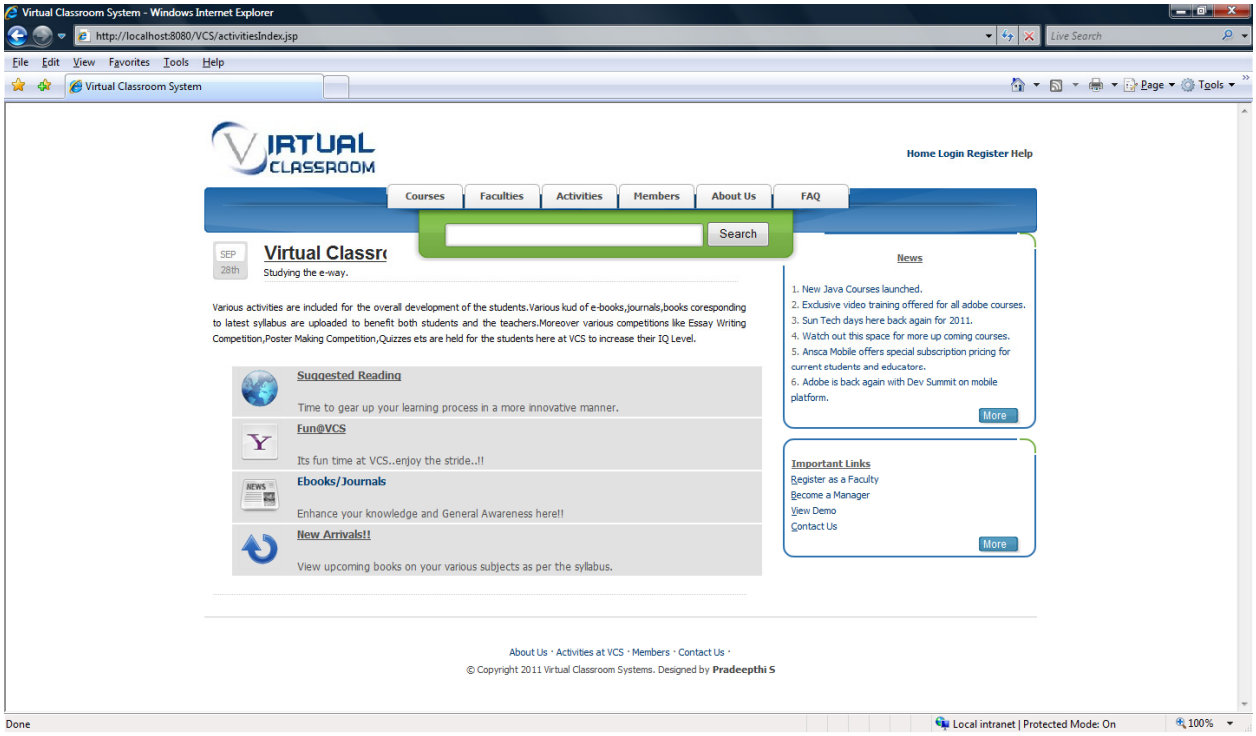


Figure 5.5: List of activities available without logging in

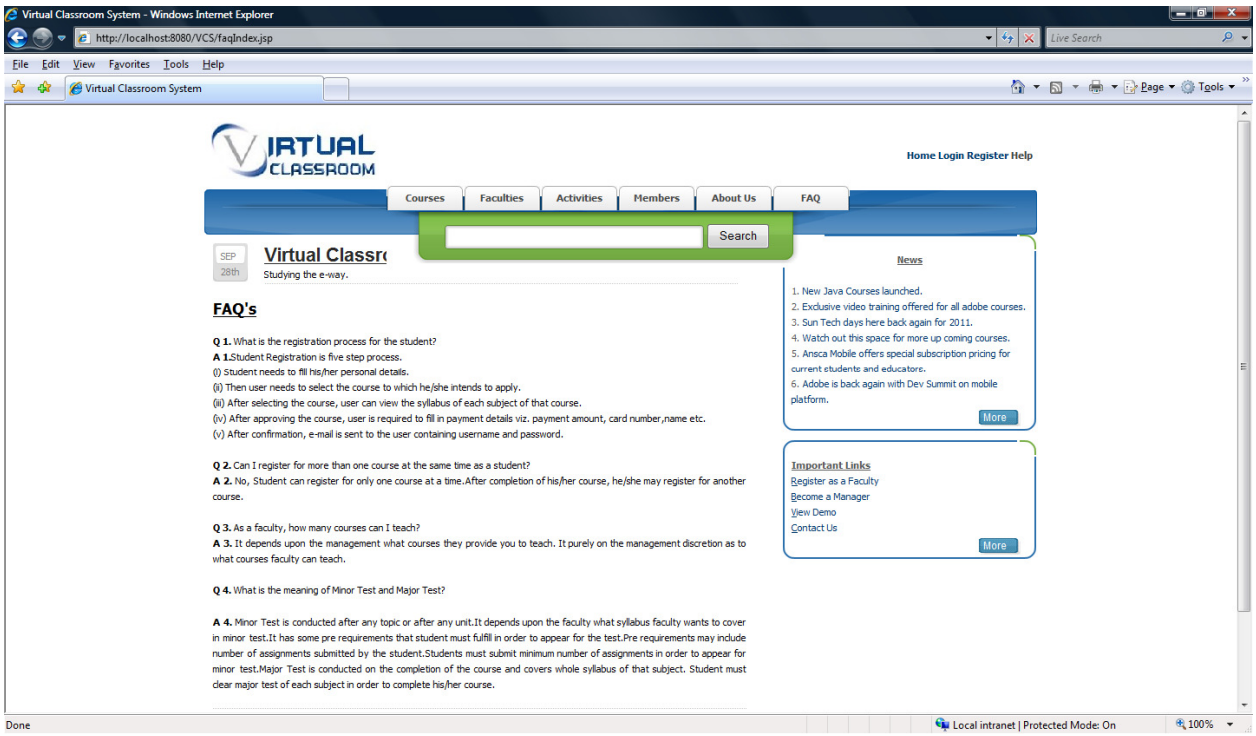


Figure 5.6: Frequently asked questions

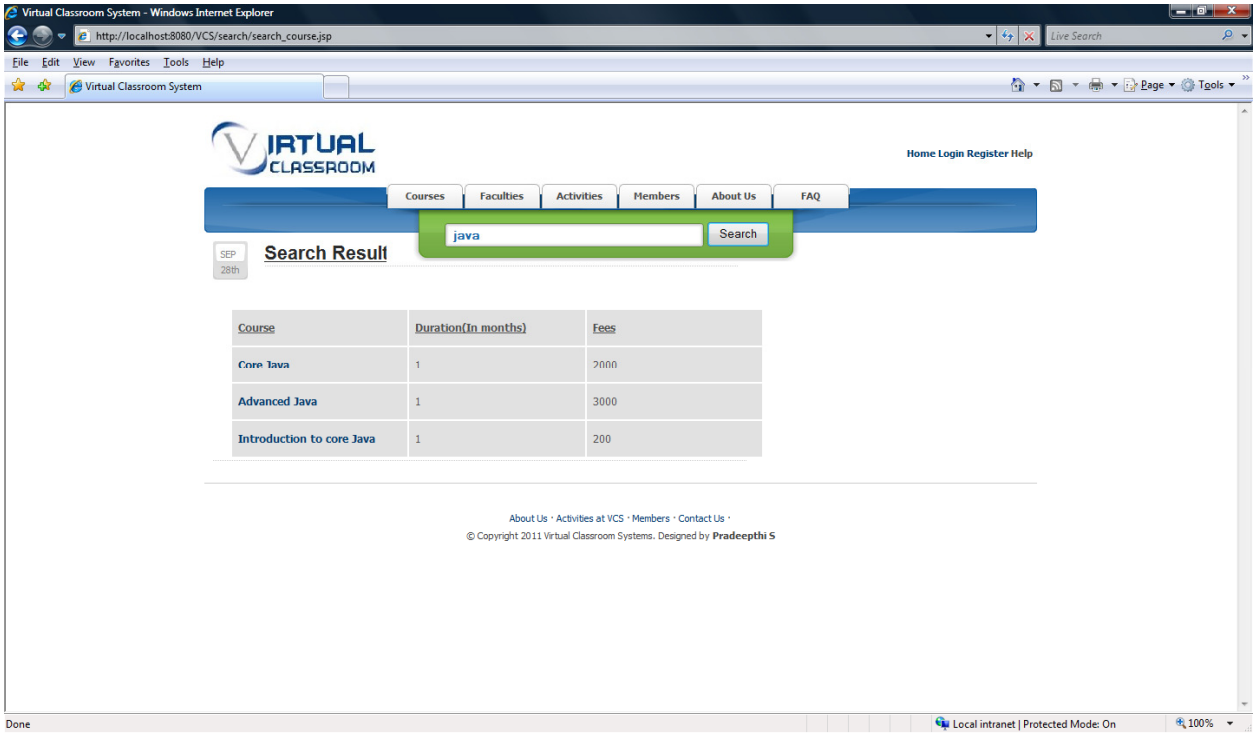


Figure 5.7: List of courses available for "java" keyword

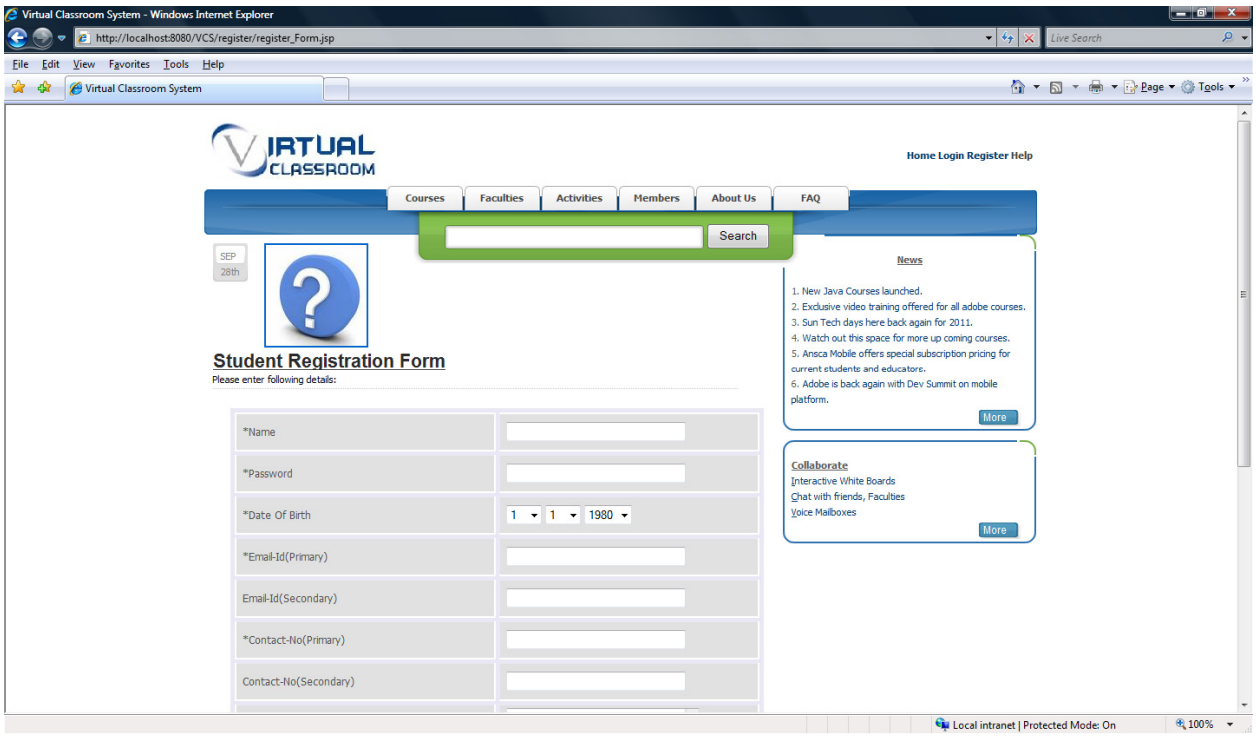


Figure 5.8: Student Registration

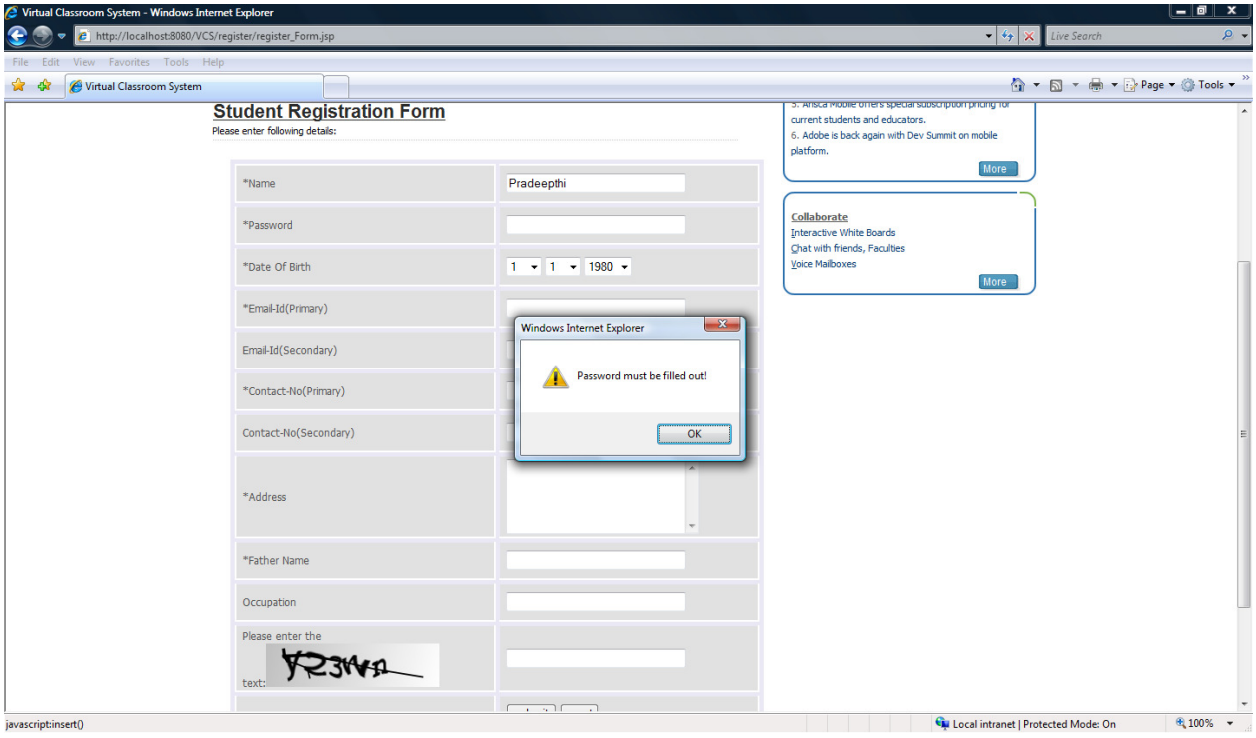


Figure 5.9: Password in Student registration

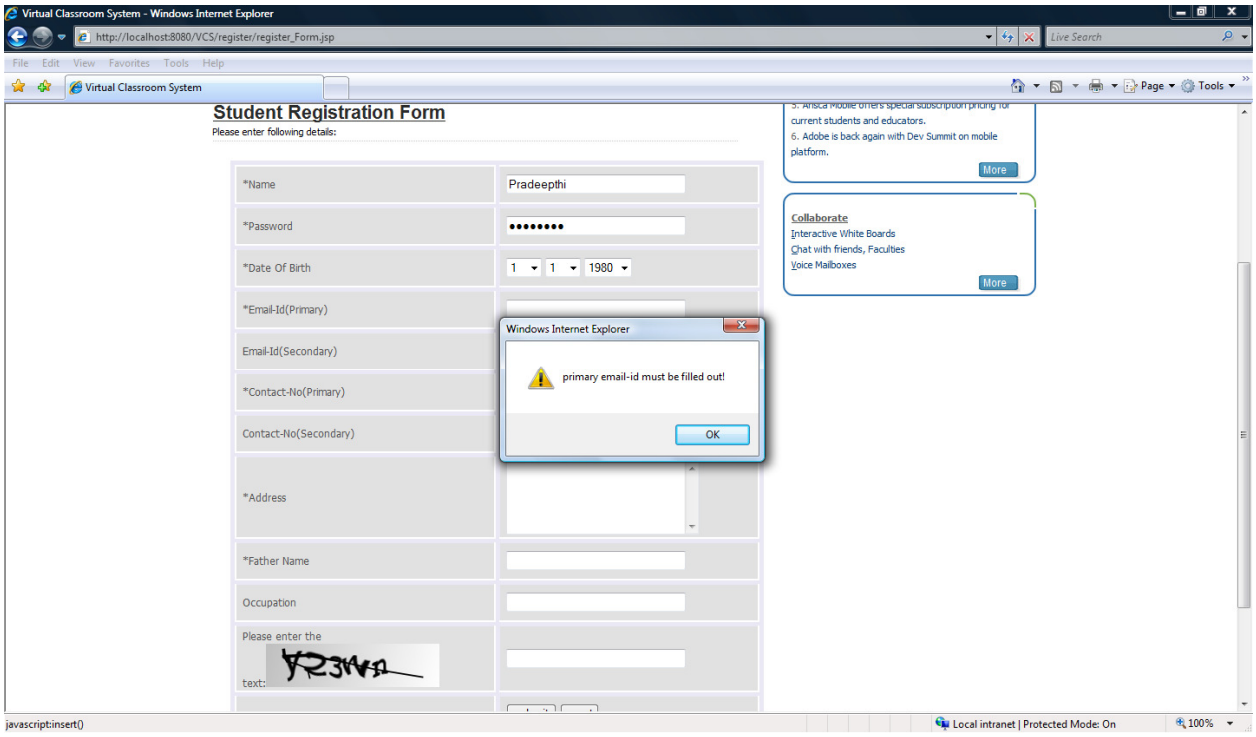


Figure 5.10: Email validation in Student registration

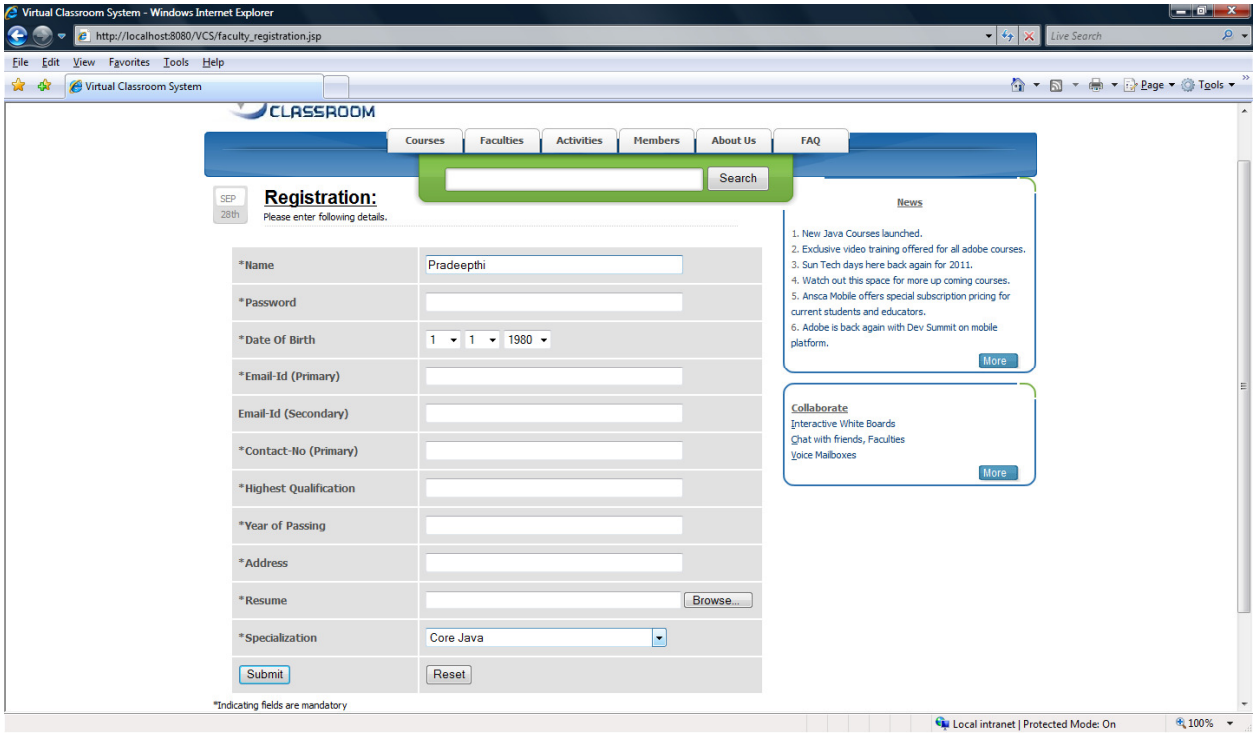


Figure 5.11: Faculty Registration page

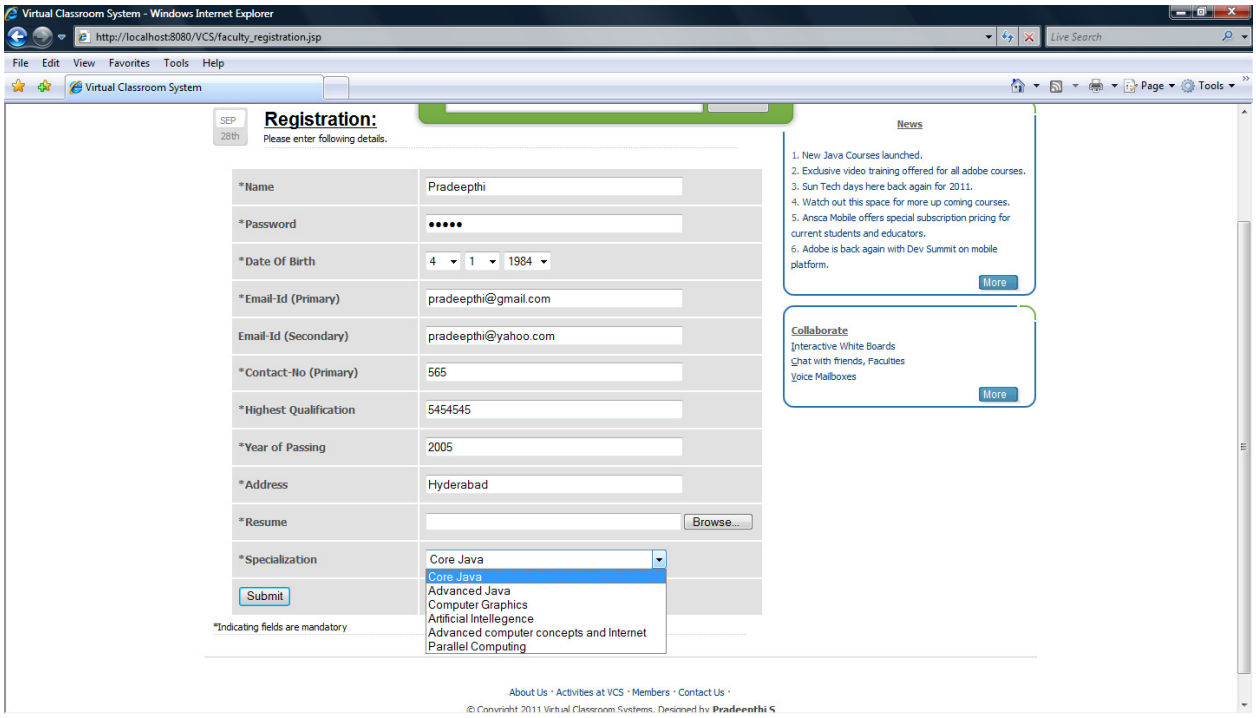


Figure 5.12: list of courses that a faculty can register

Admin Home

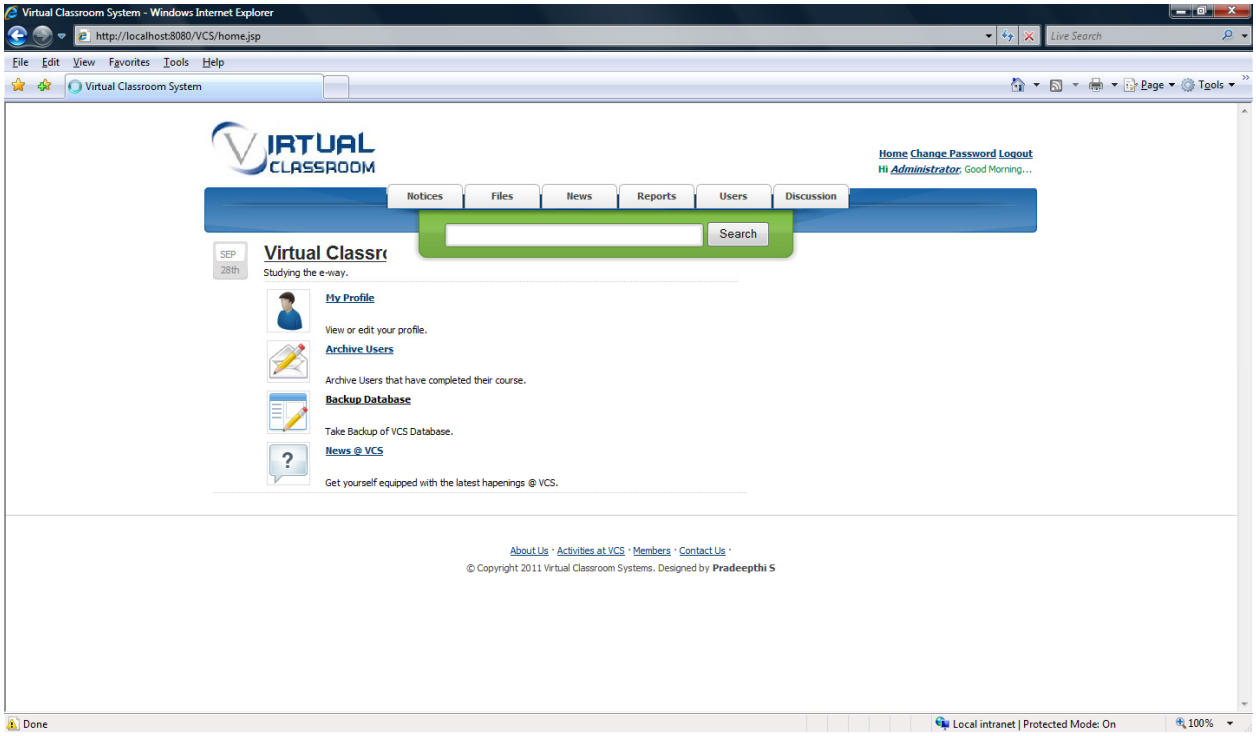


Figure 5.13: Admin home page

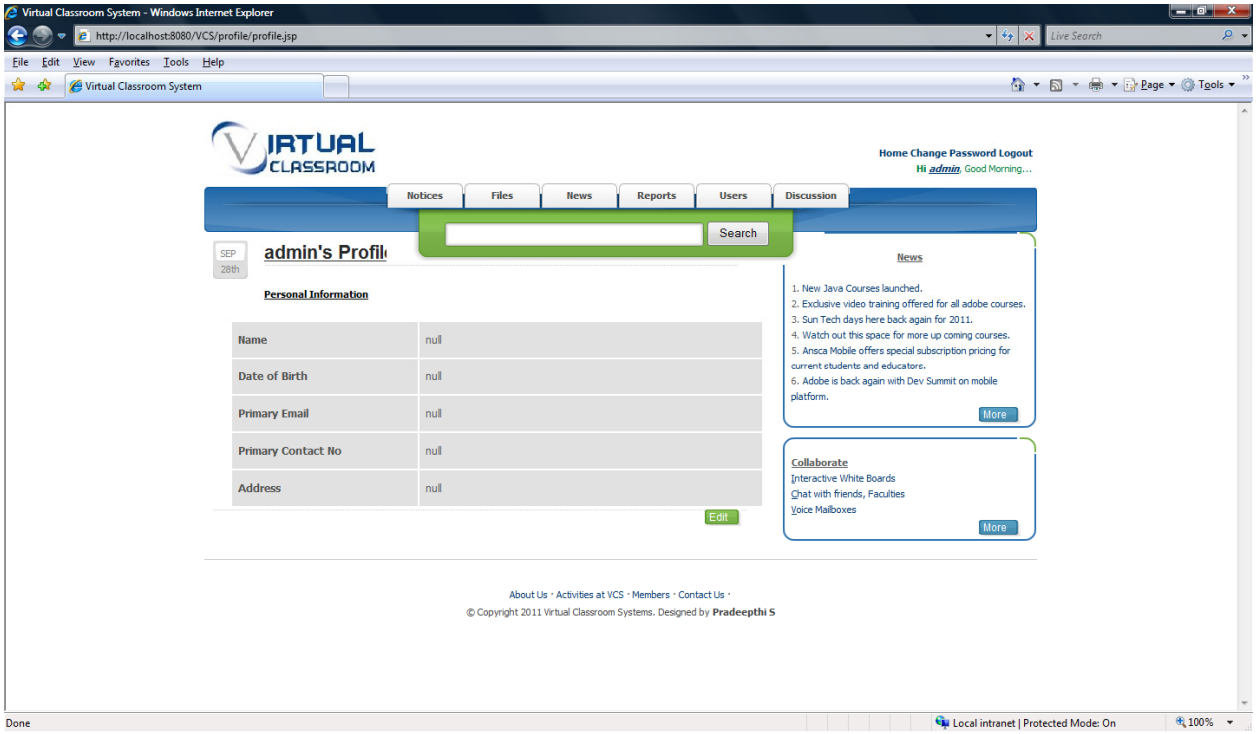


Figure 5.14: Profile of Admin user

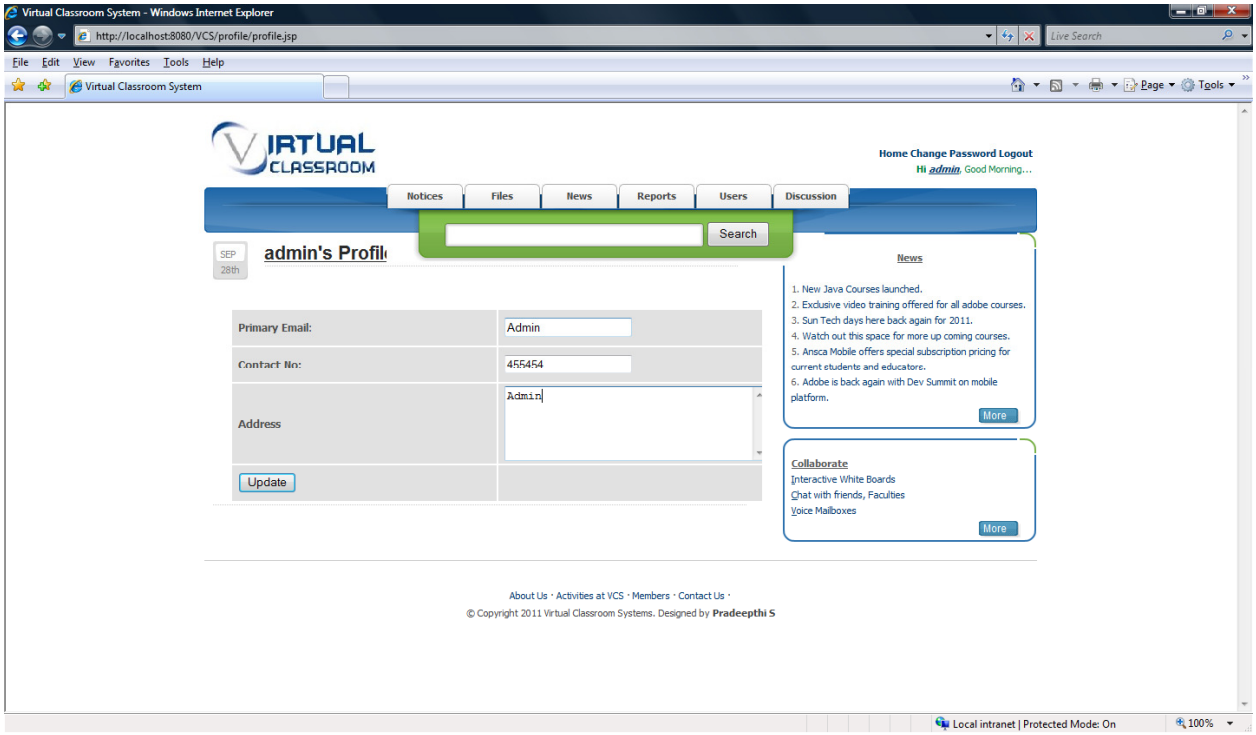


Figure 5.15: Profile edit

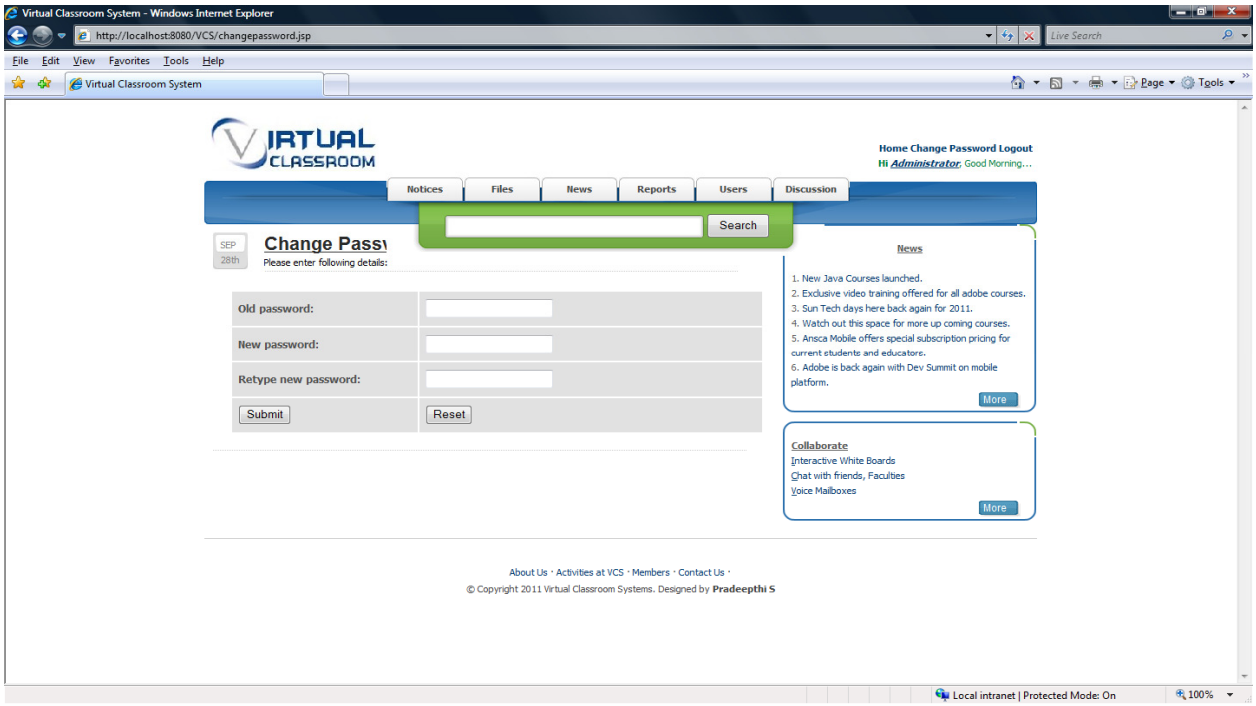


Figure 5.16: Changing password

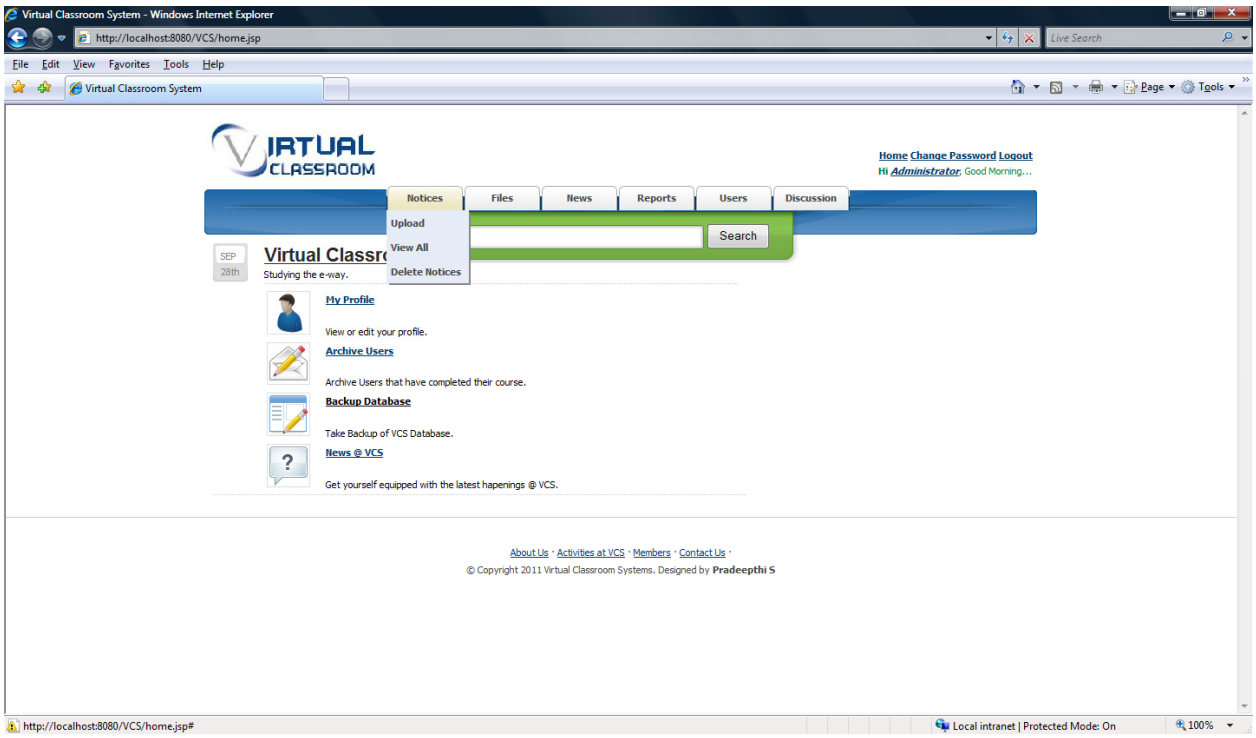


Figure 5.17: Notices Menu for Admin user

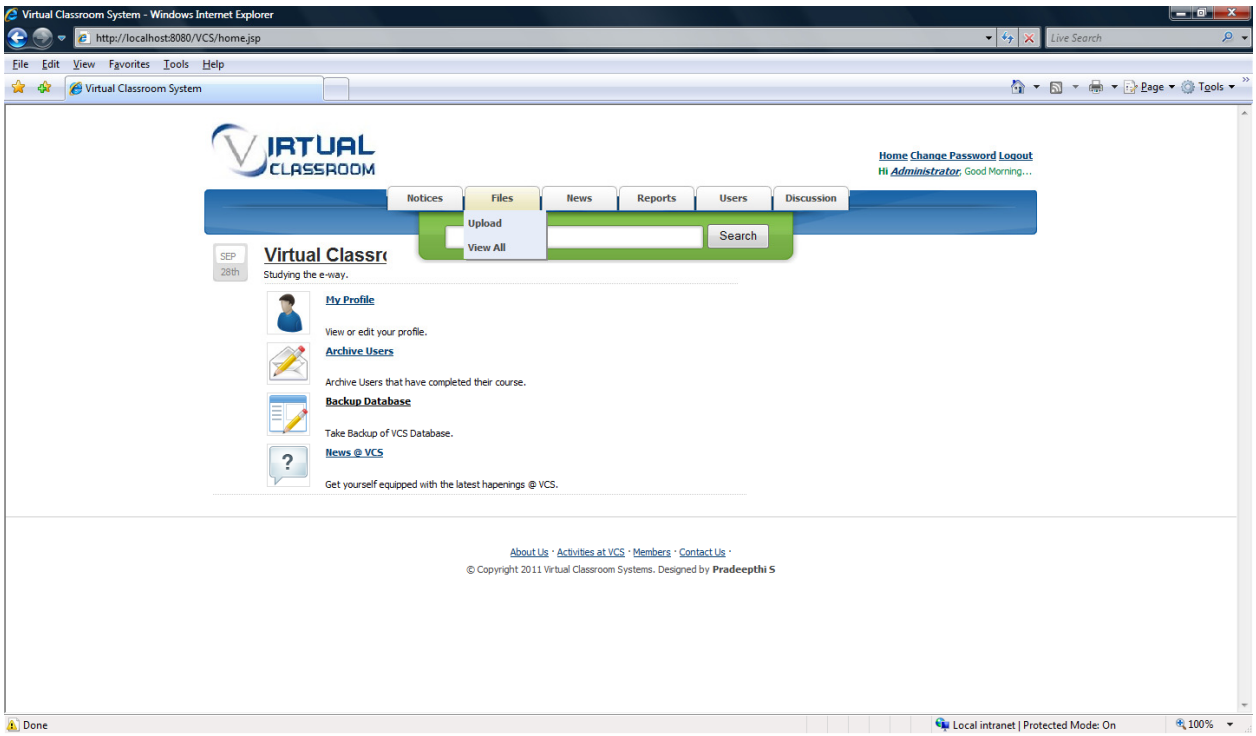


Figure 5.18: Files Menu for Admin

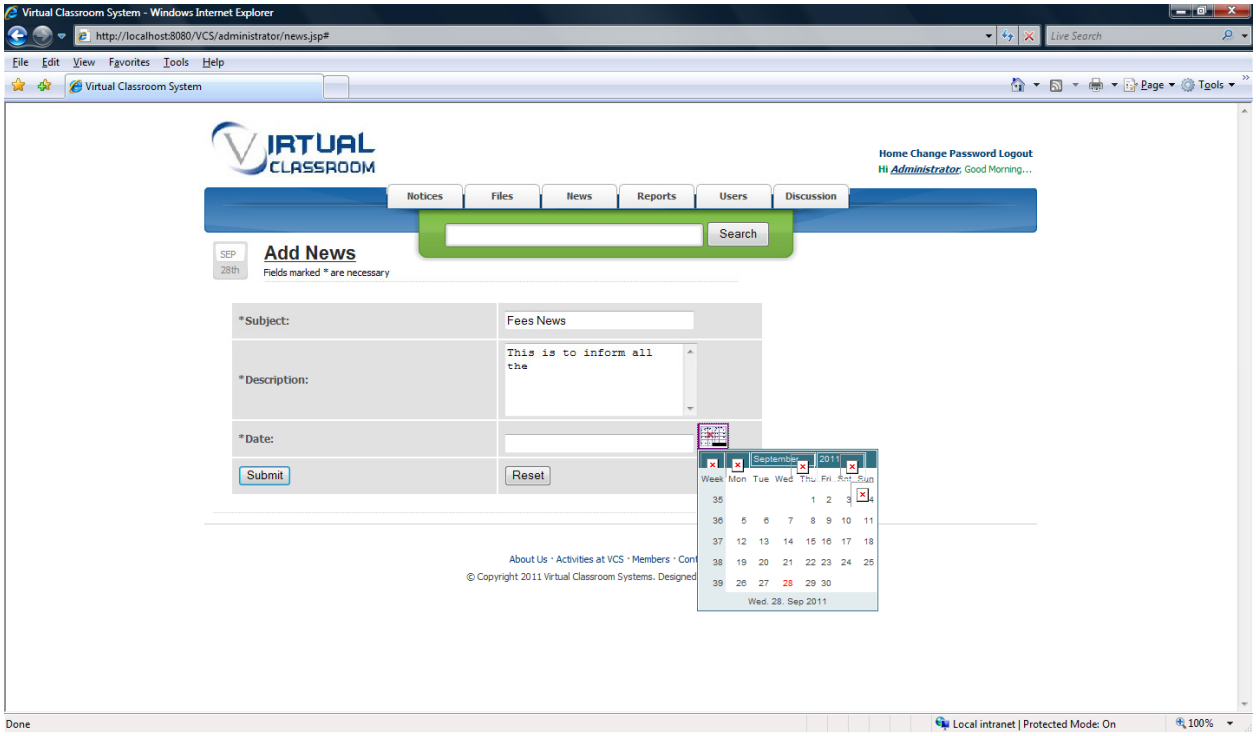


Figure 5.19: Adding News with end date

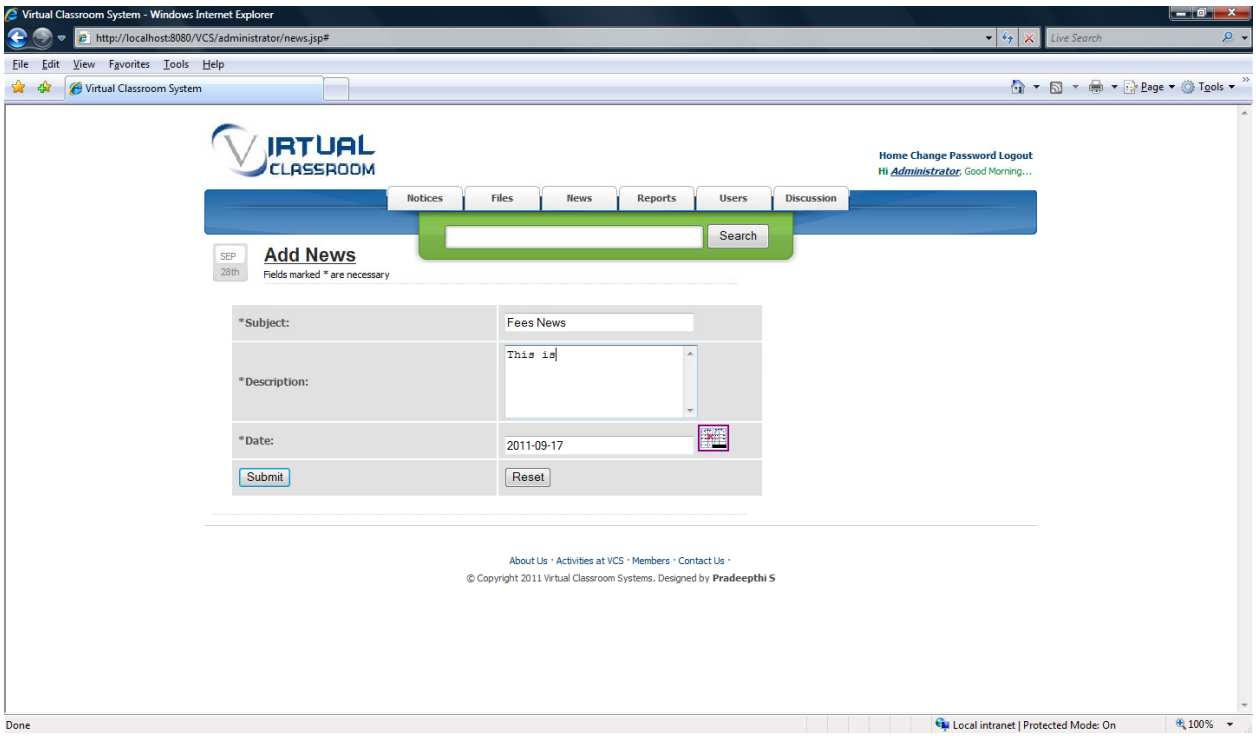


Figure 5.20: Adding news

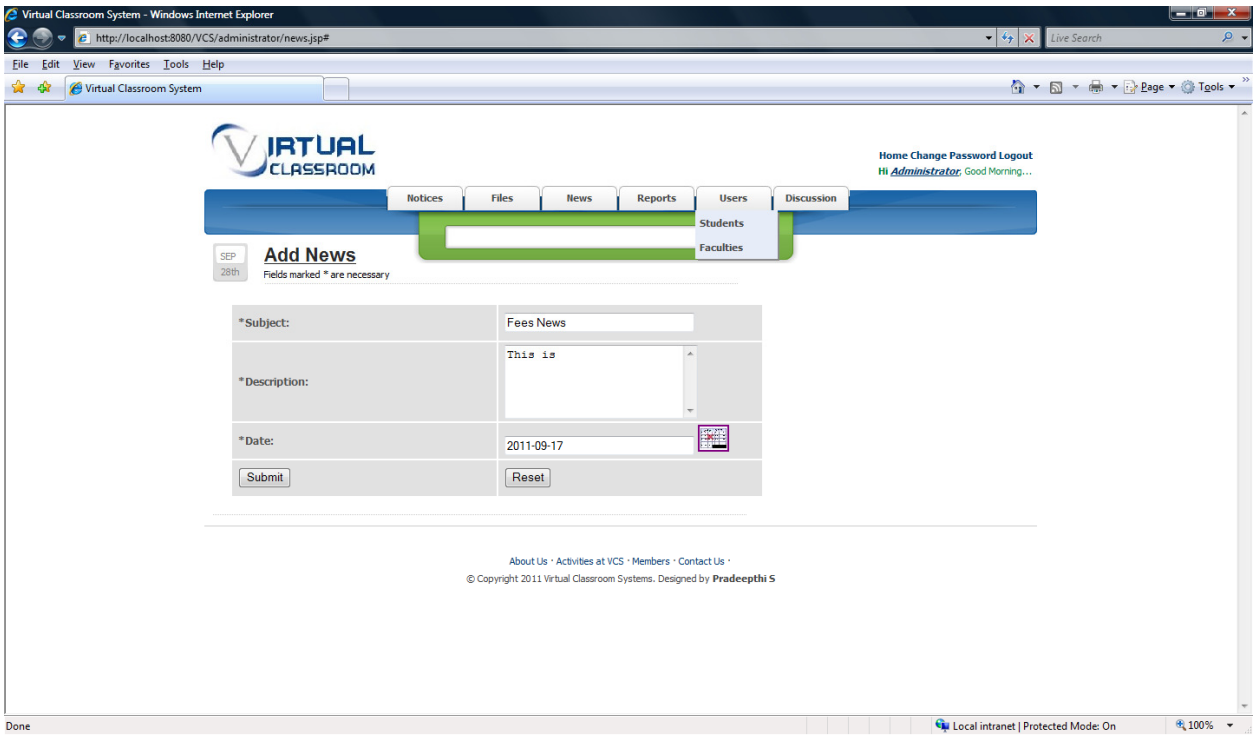


Figure 5.21: Users menu options (Students, Faculties)

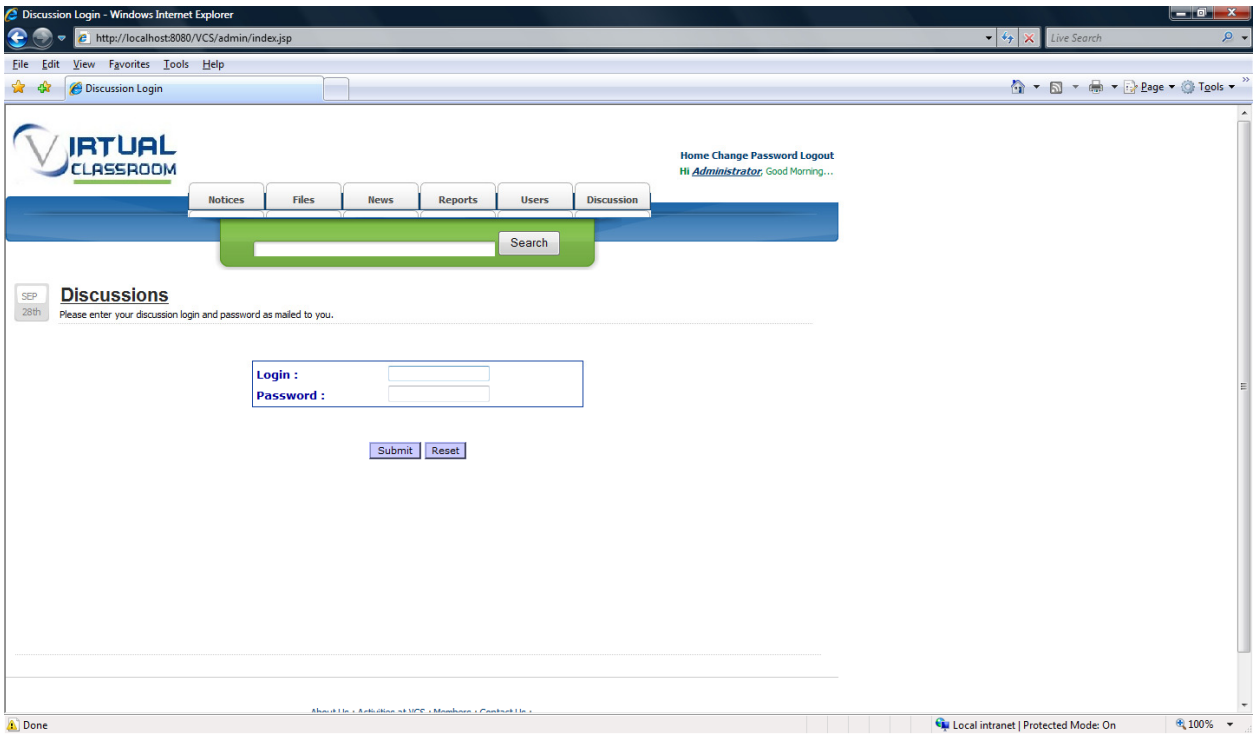


Figure 5.22: Discussions login page

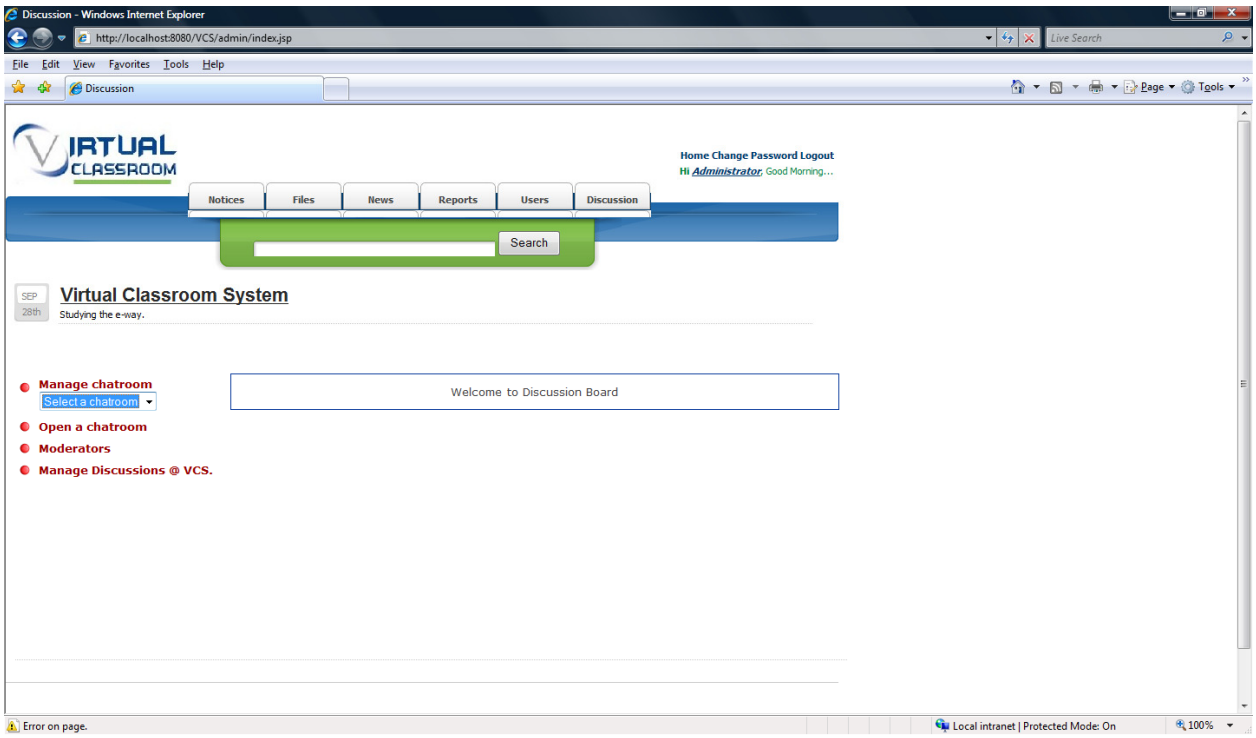


Figure 5.23: Managing discussions

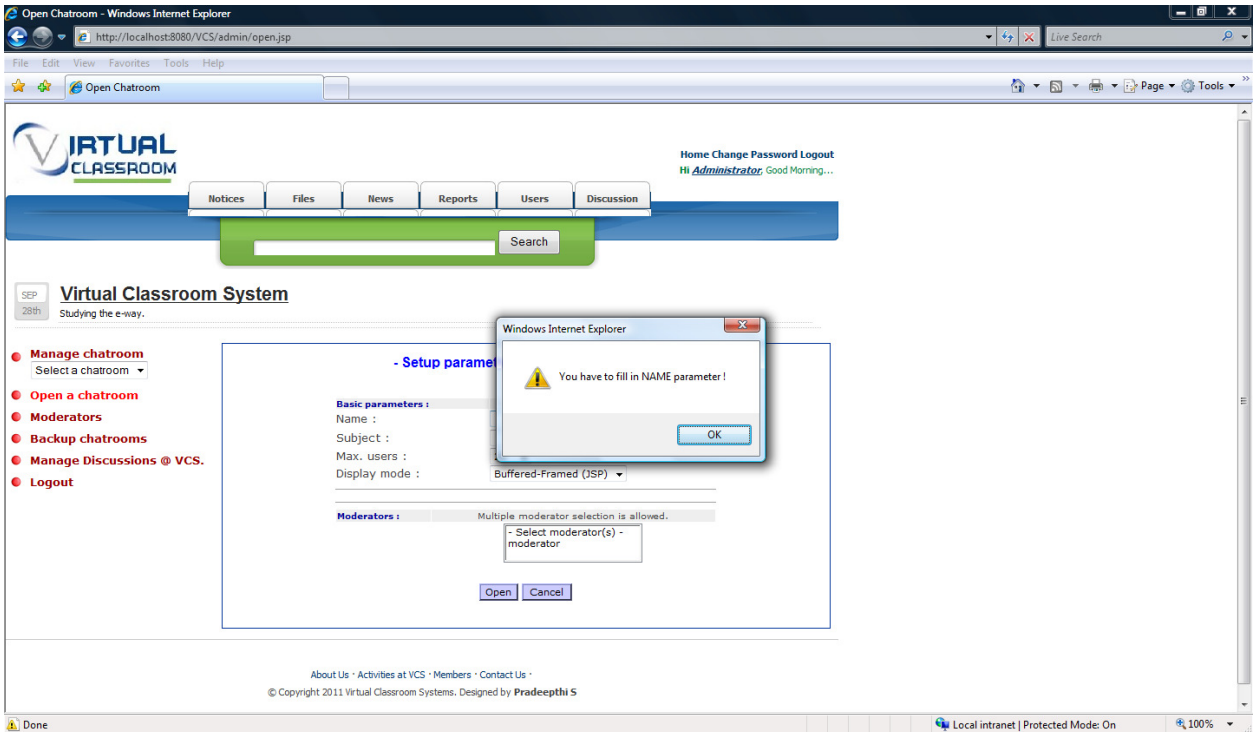


Figure 5.24: Validations in creating discussion

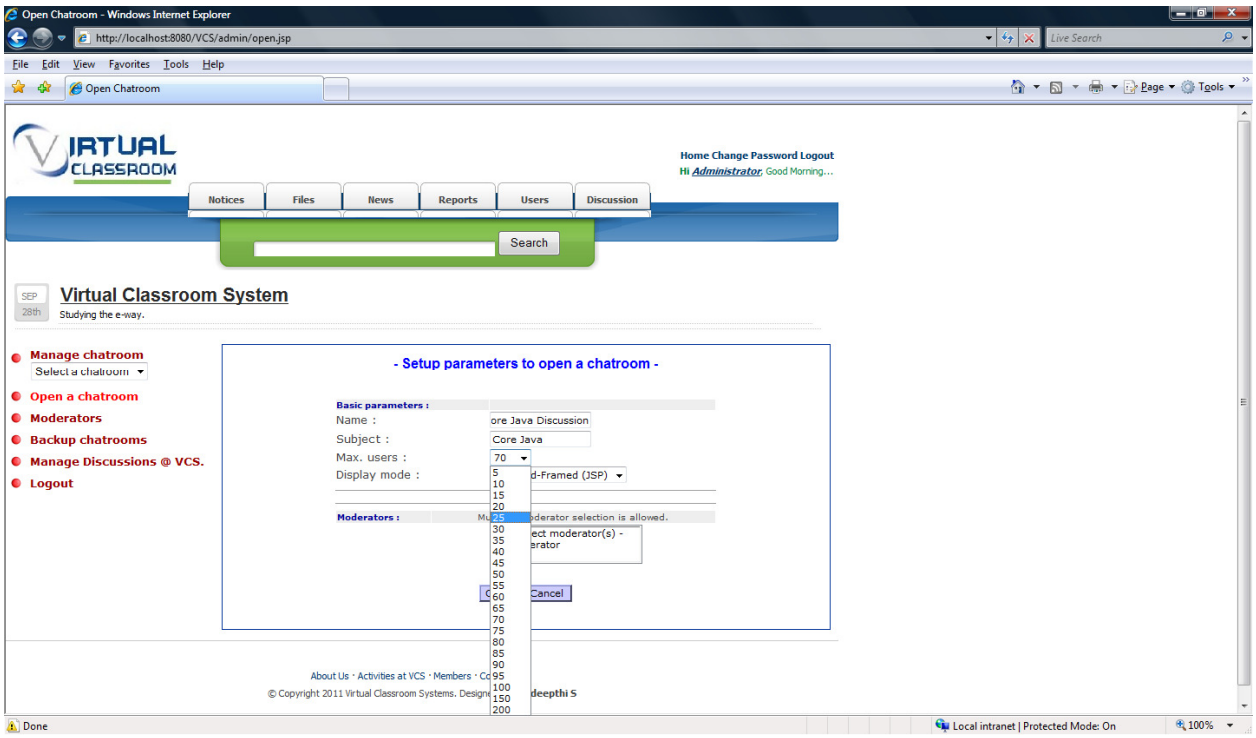


Figure 5.25: selecting maximum number of people for a discussion

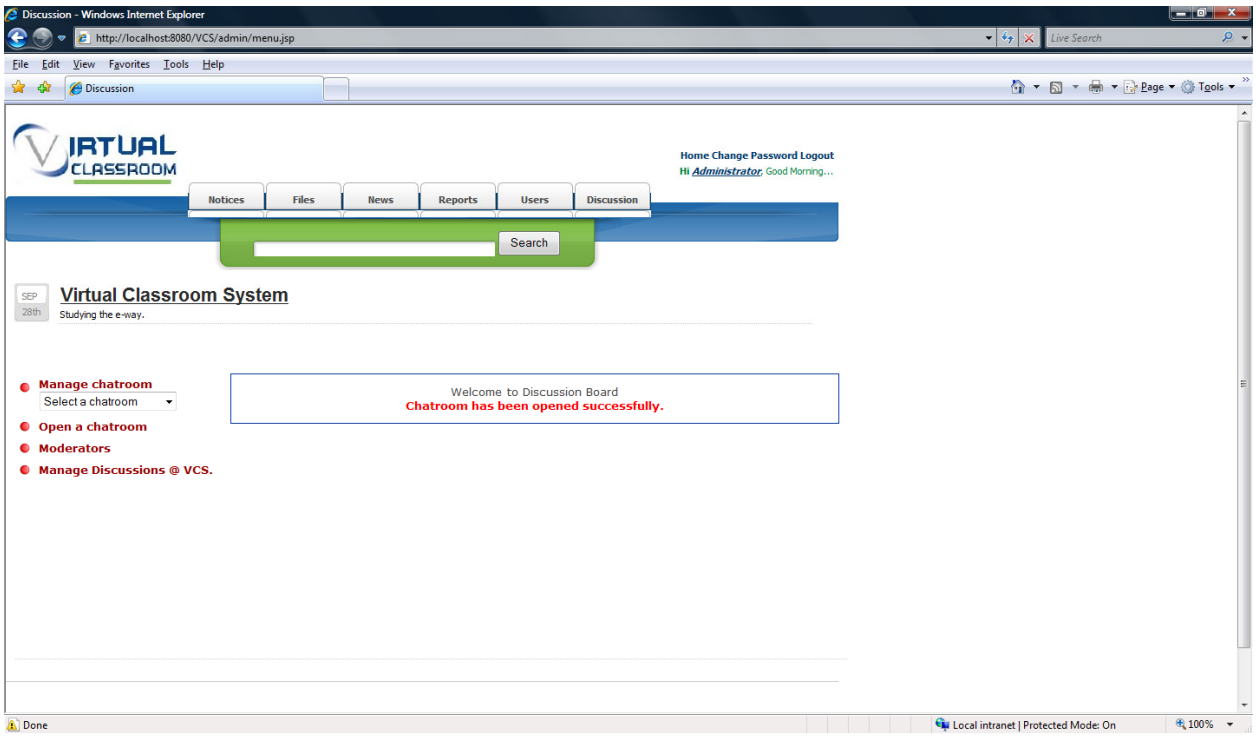


Figure 5.26: Discussion creation success message

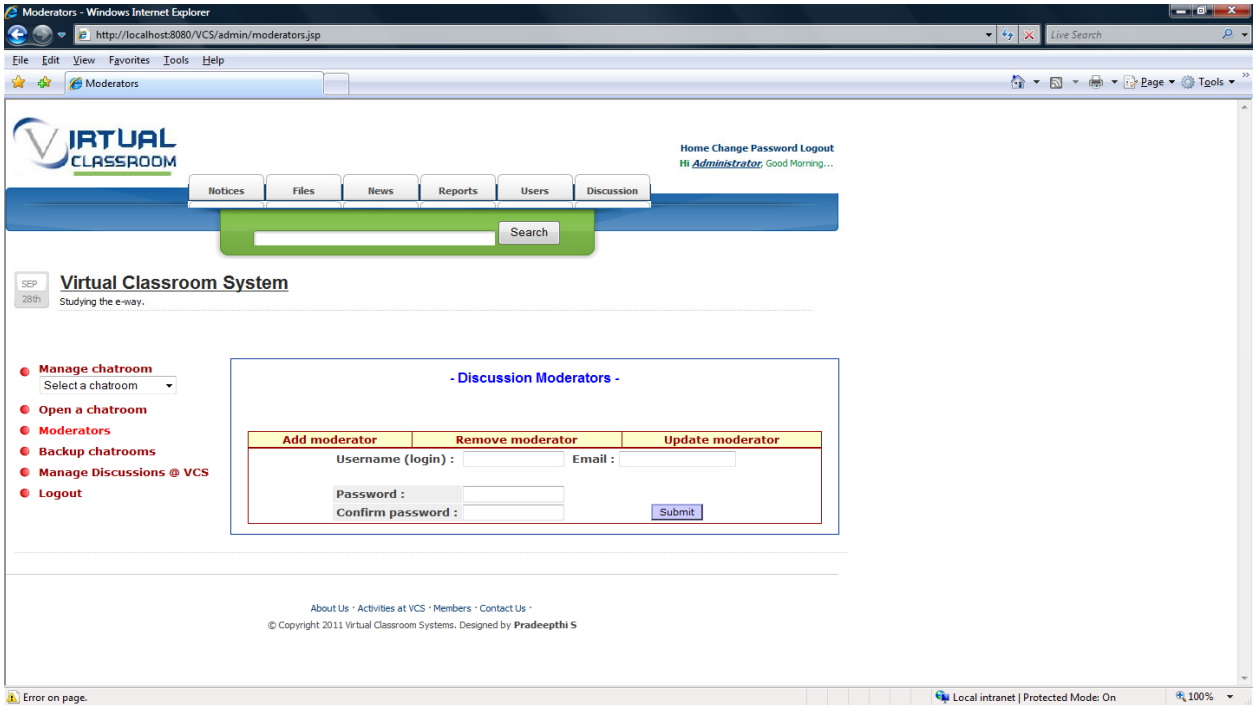


Figure 5.27: Creating moderator for a discussion

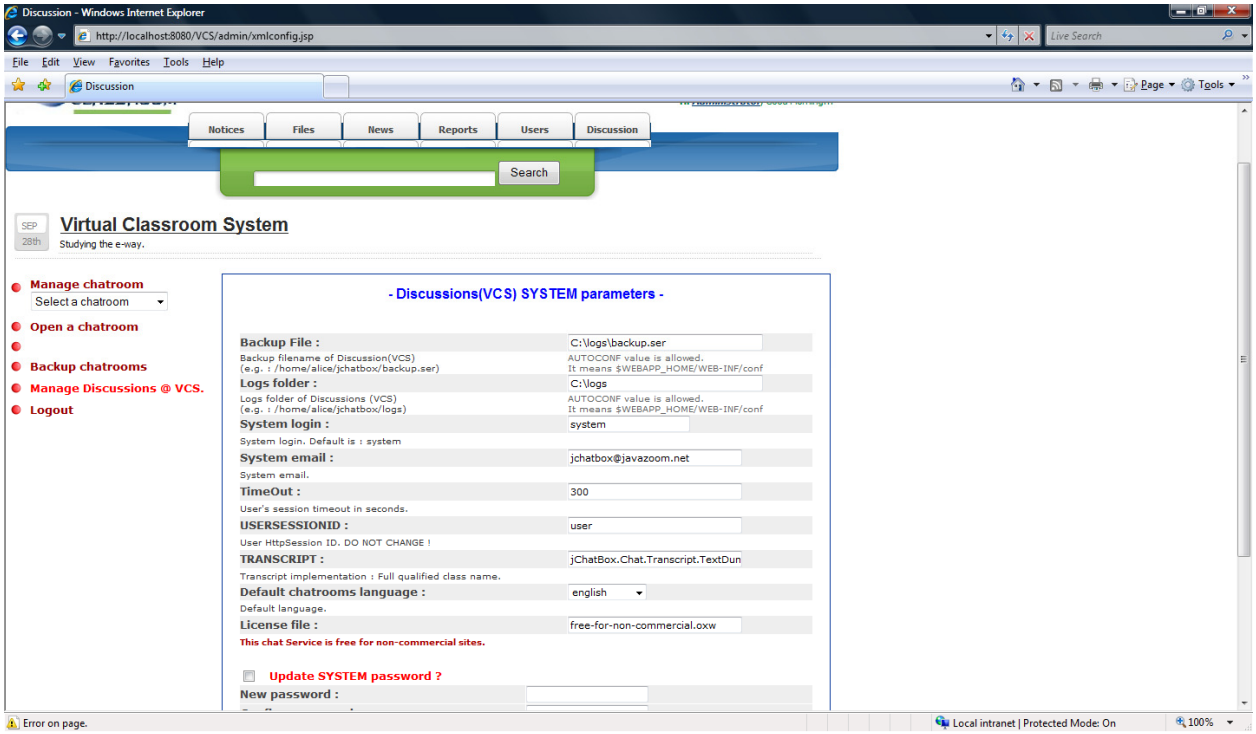


Figure 5.28: Managing the Discussions

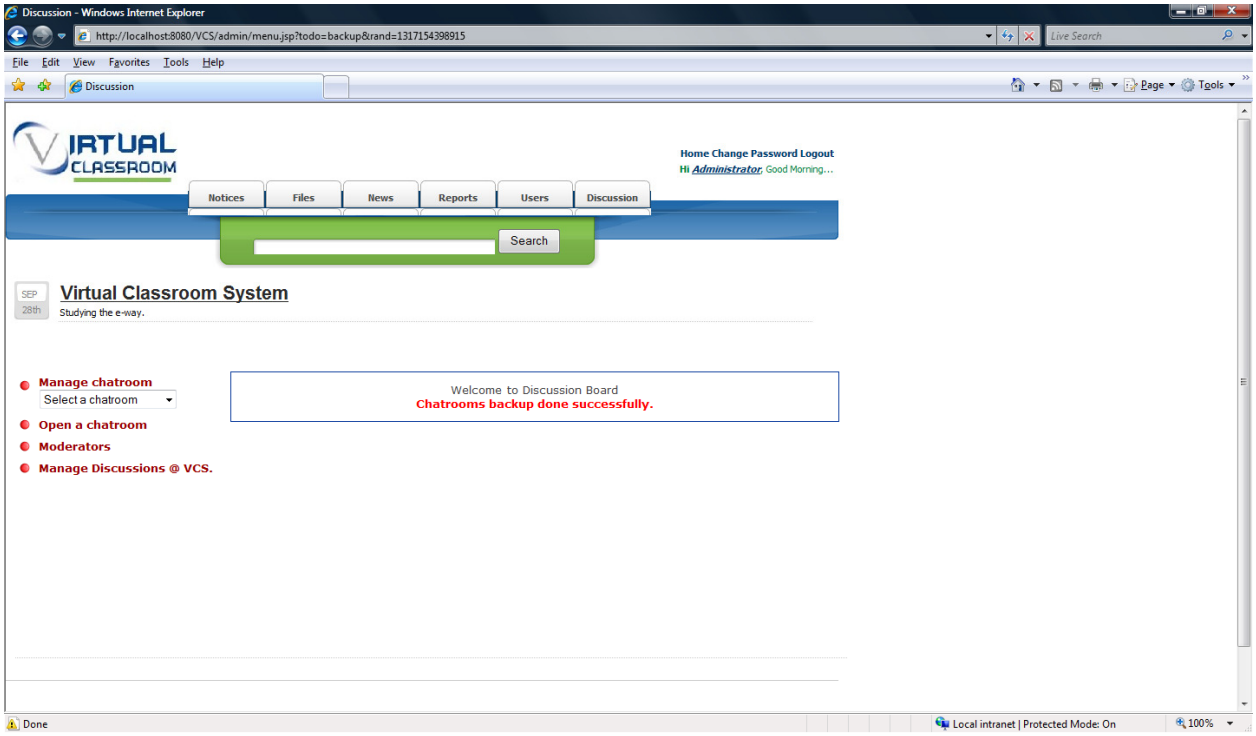


Figure 5.29: Taking Backup of the discussions.

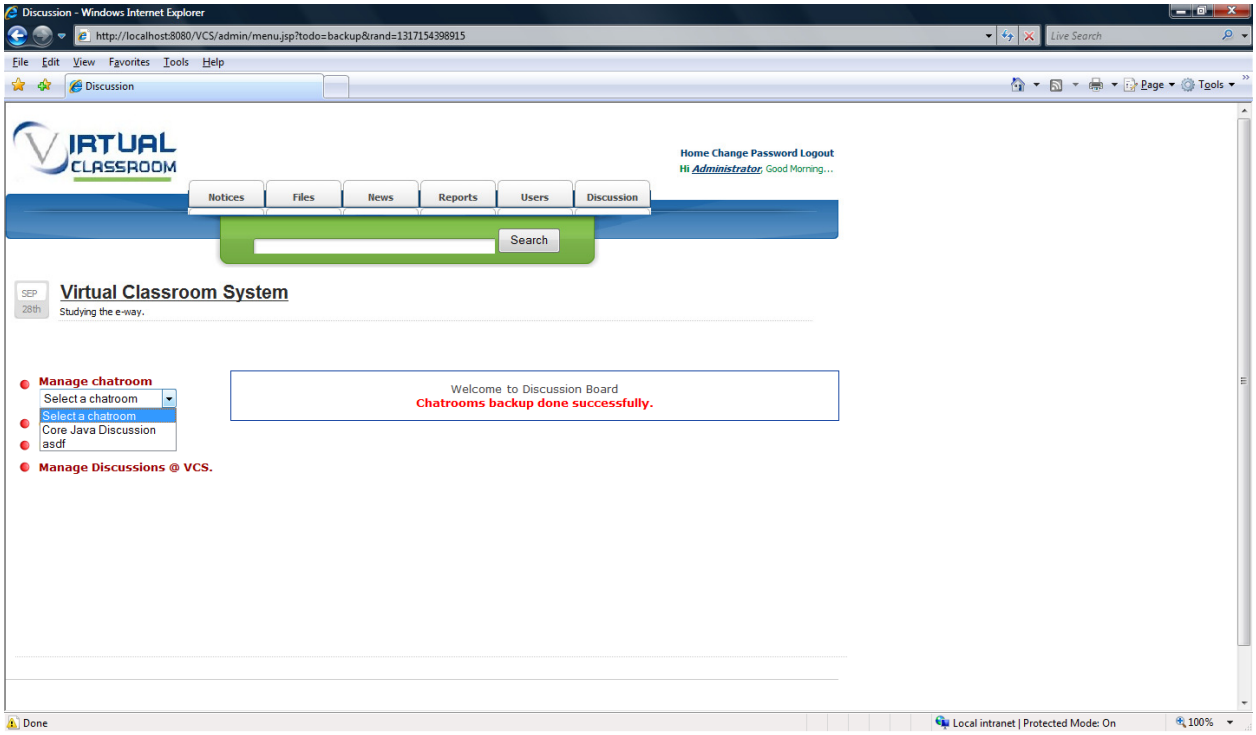


Figure 5.30: Selecting the discussion to manage

Figure 5.31: Uploading files

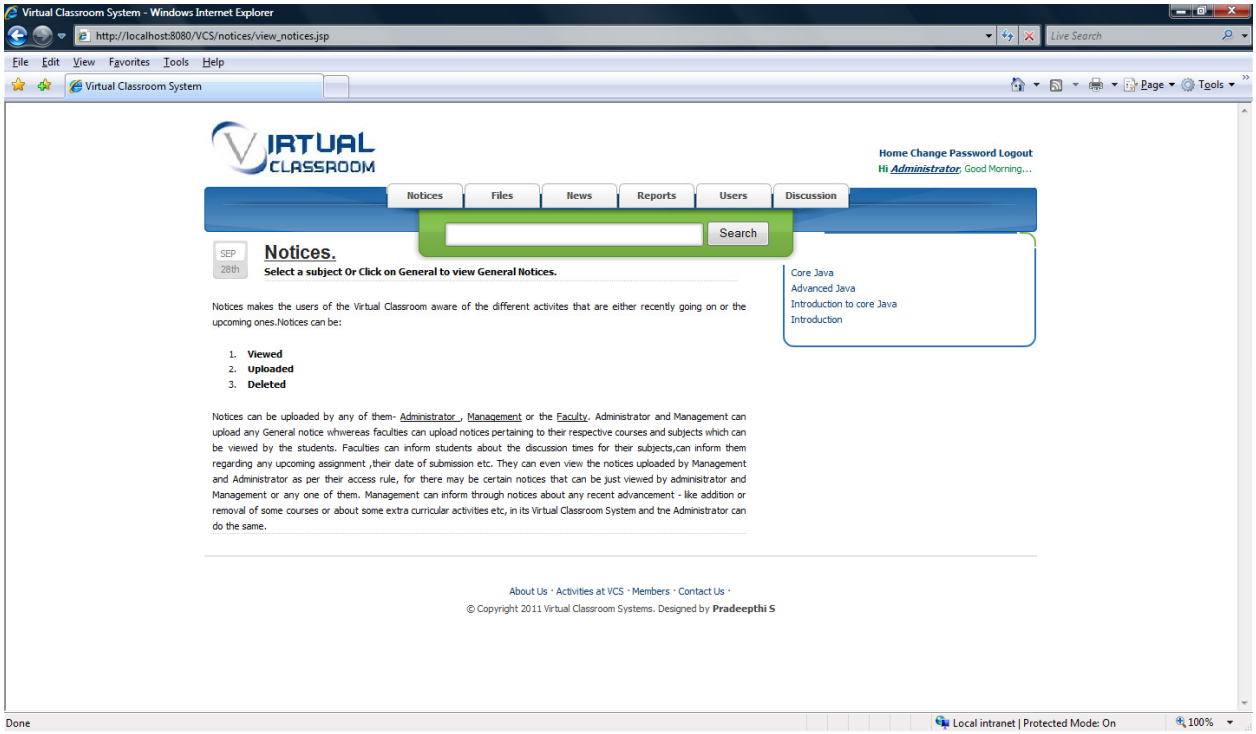


Figure 5.32: Listing all the notices by selecting a particular subject Dele

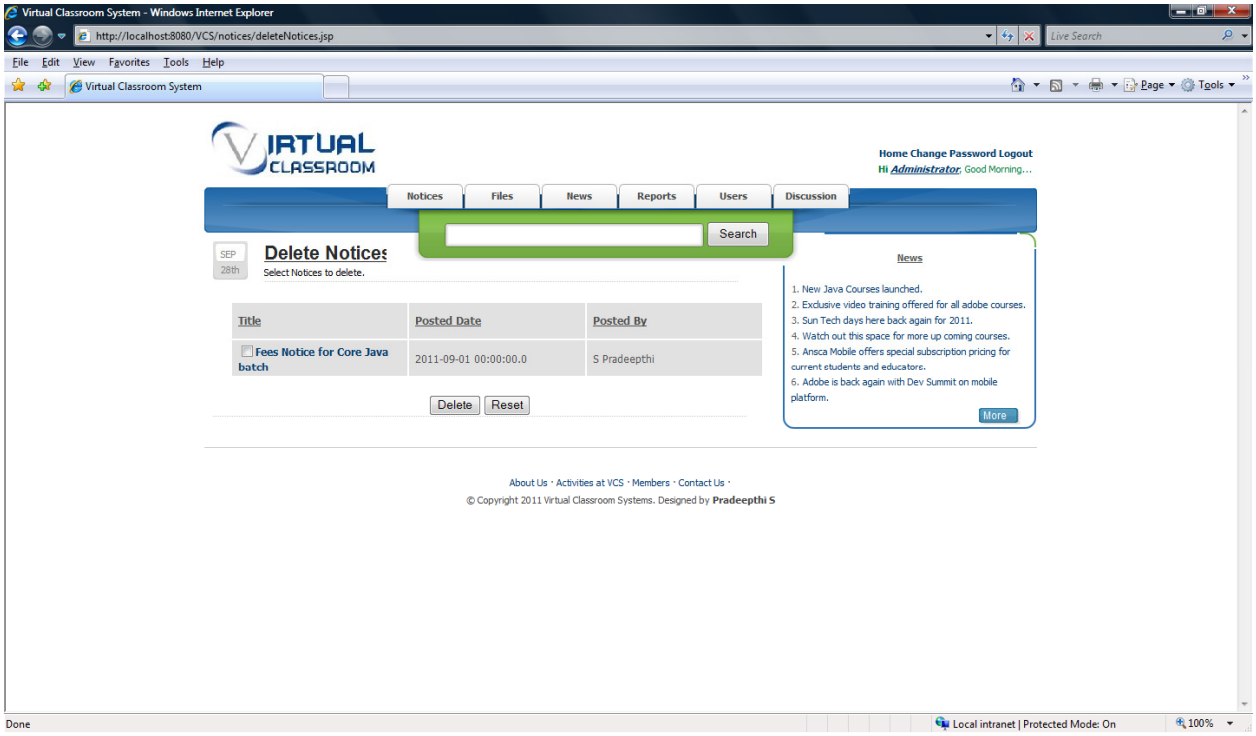


Figure 5.33: Deleting notices

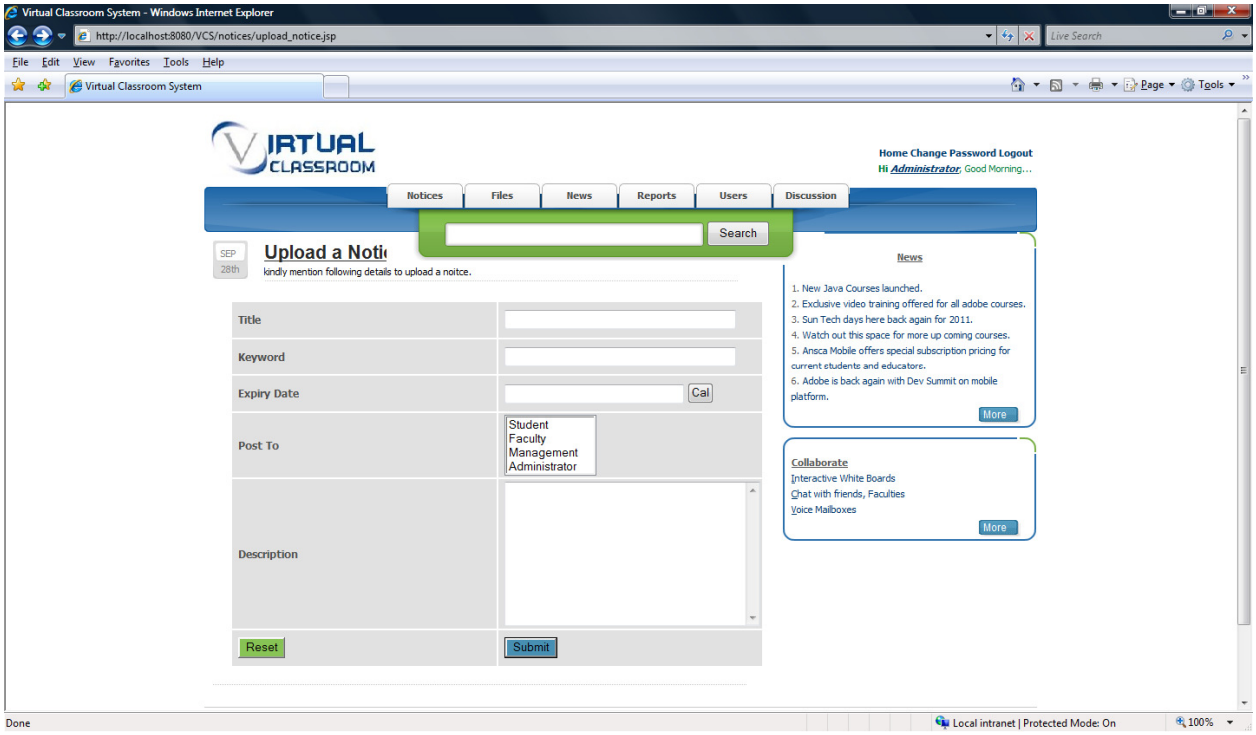


Figure 5.34: uploading a notice

Figure 5.35: uploading general/lecture/assignment files

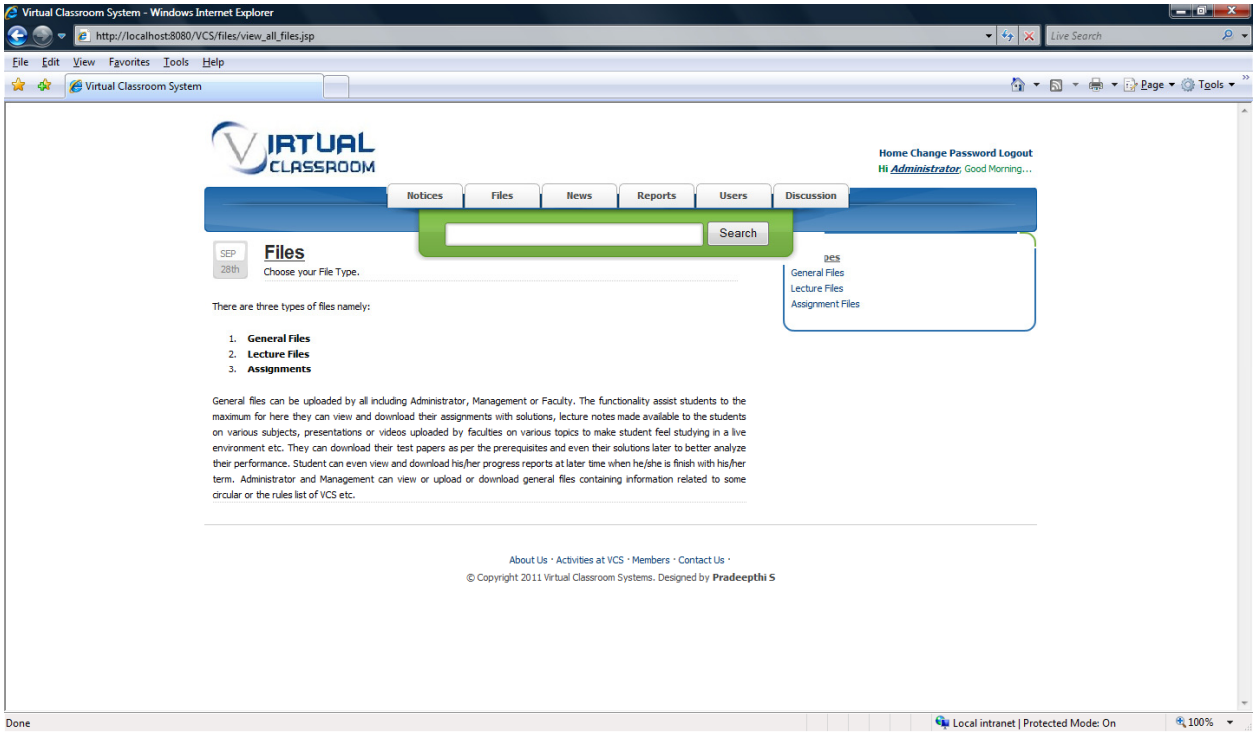


Figure 5.36: viewing all the files

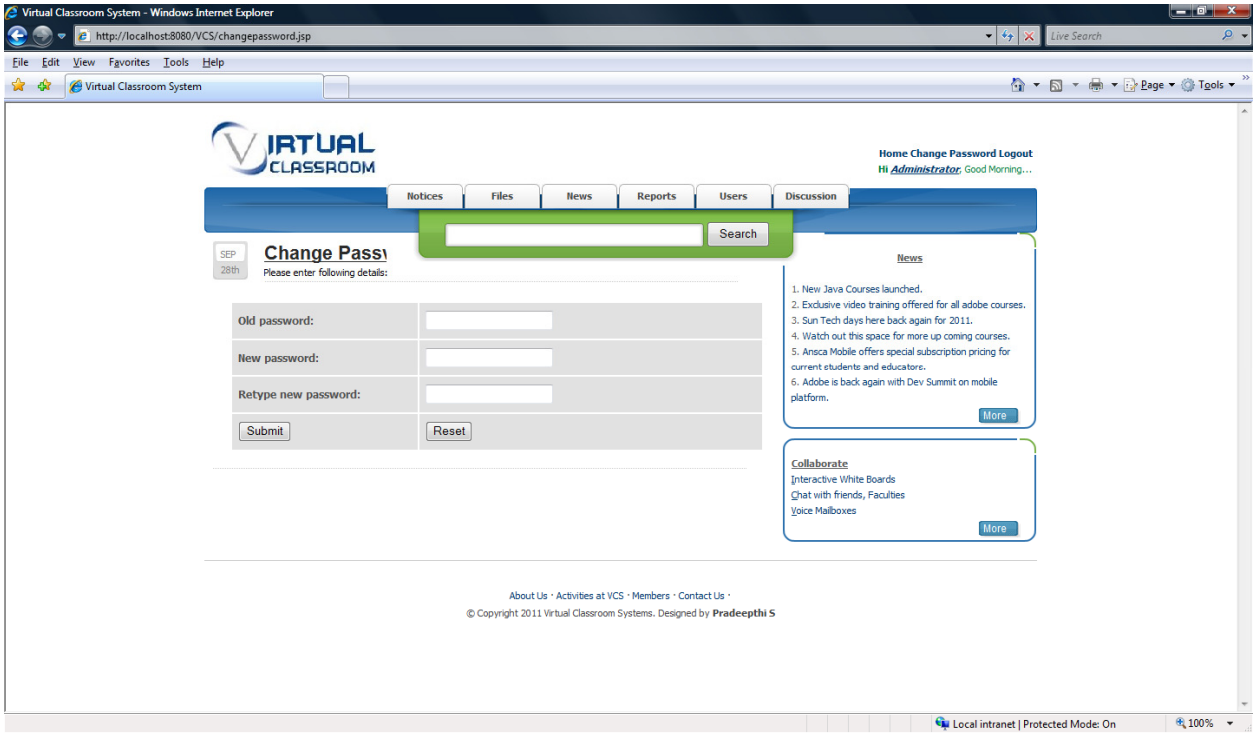


Figure 5.37: Changing password

Faculty

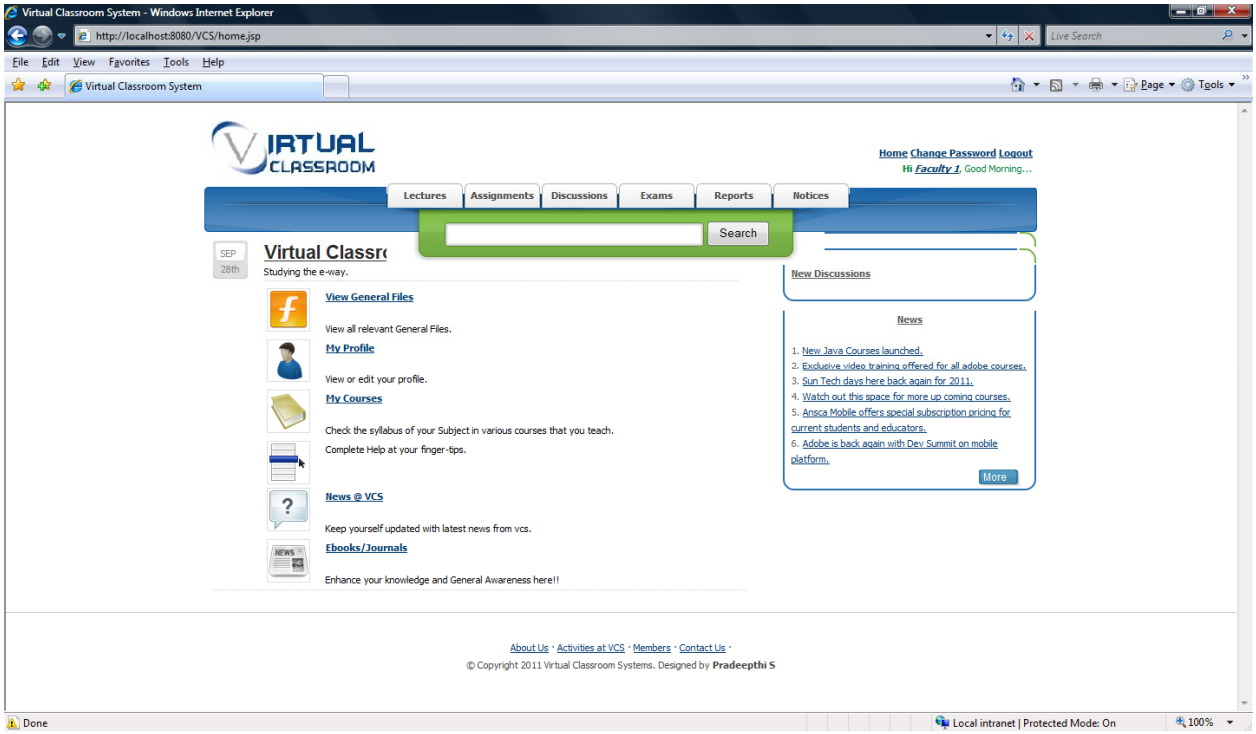


Figure 5.38: Faculty home page

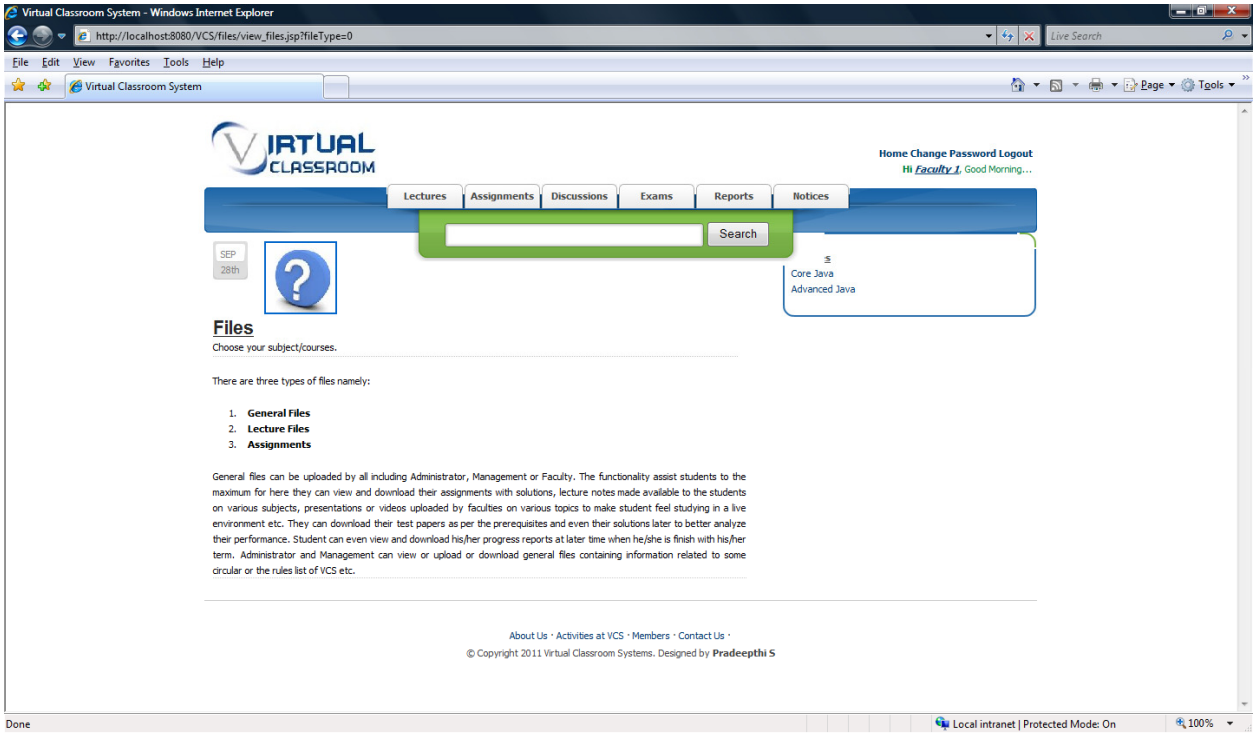


Figure 5.39: Faculty files

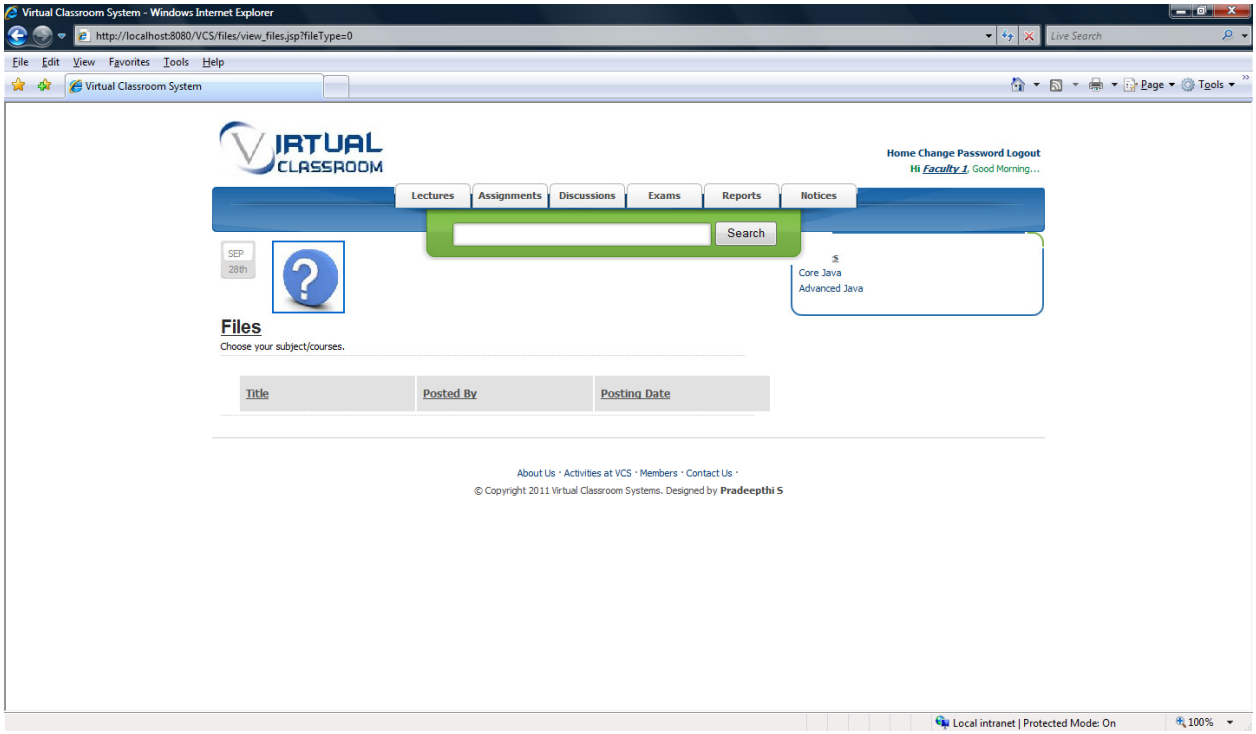


Figure 5.40: Listing all files – subject wise

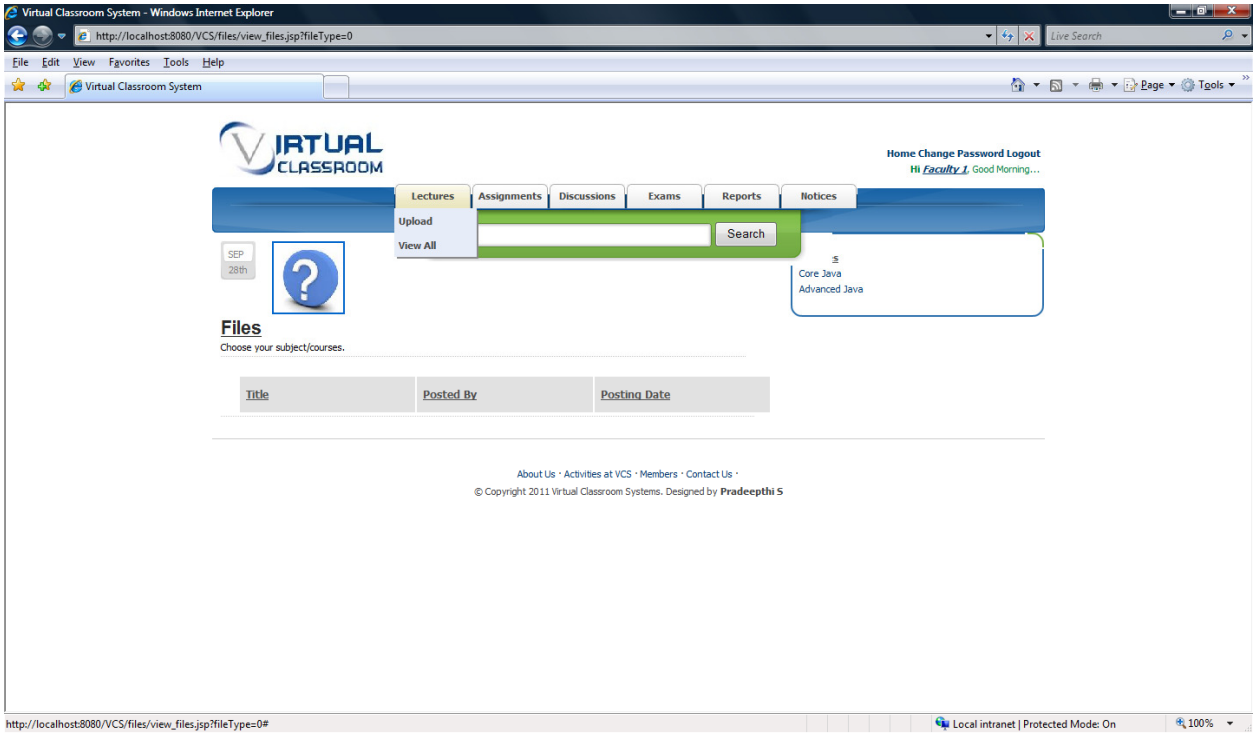


Figure 5.41: lectures menu options

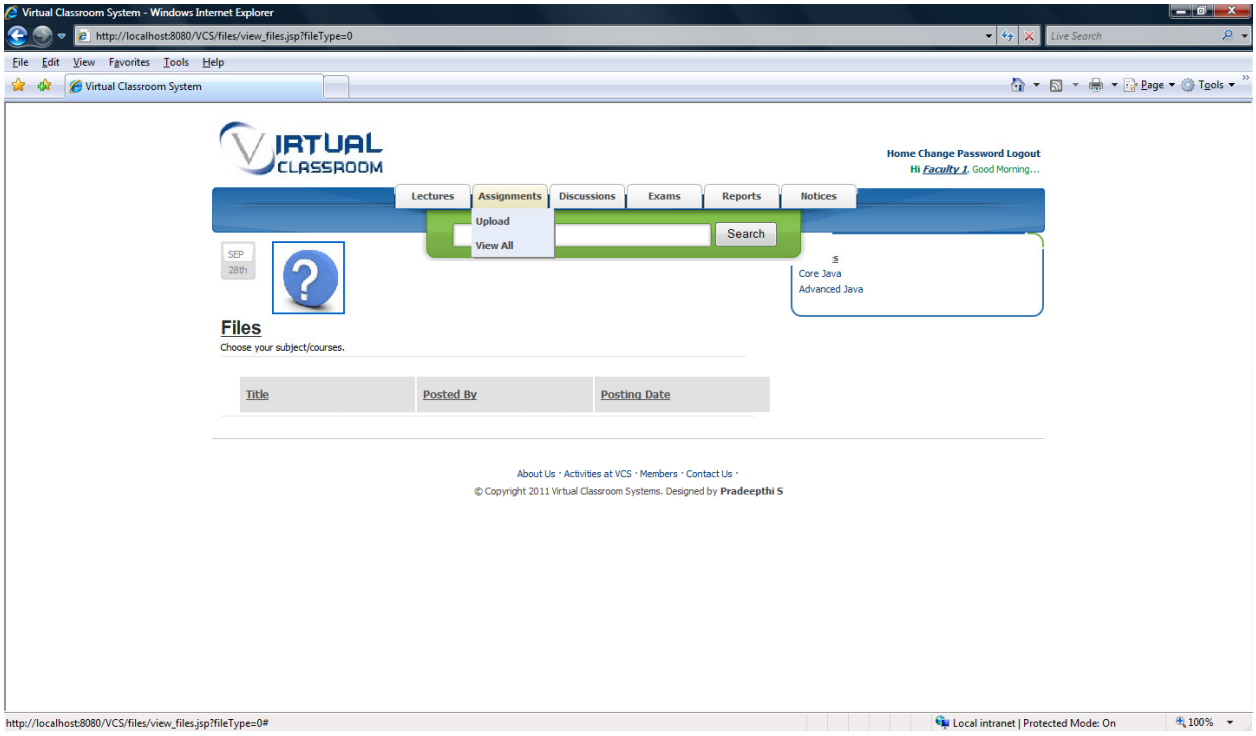


Figure 5.42: assignment menu options

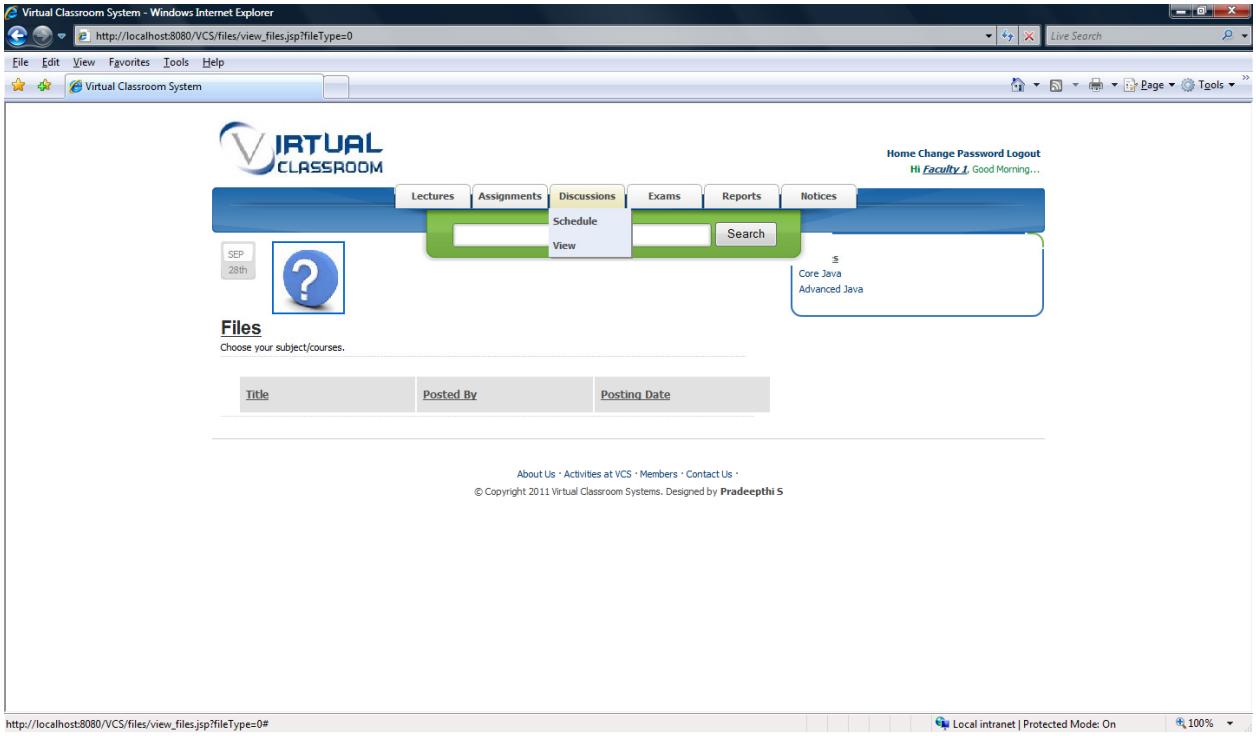


Figure 5.43: discussions menu options

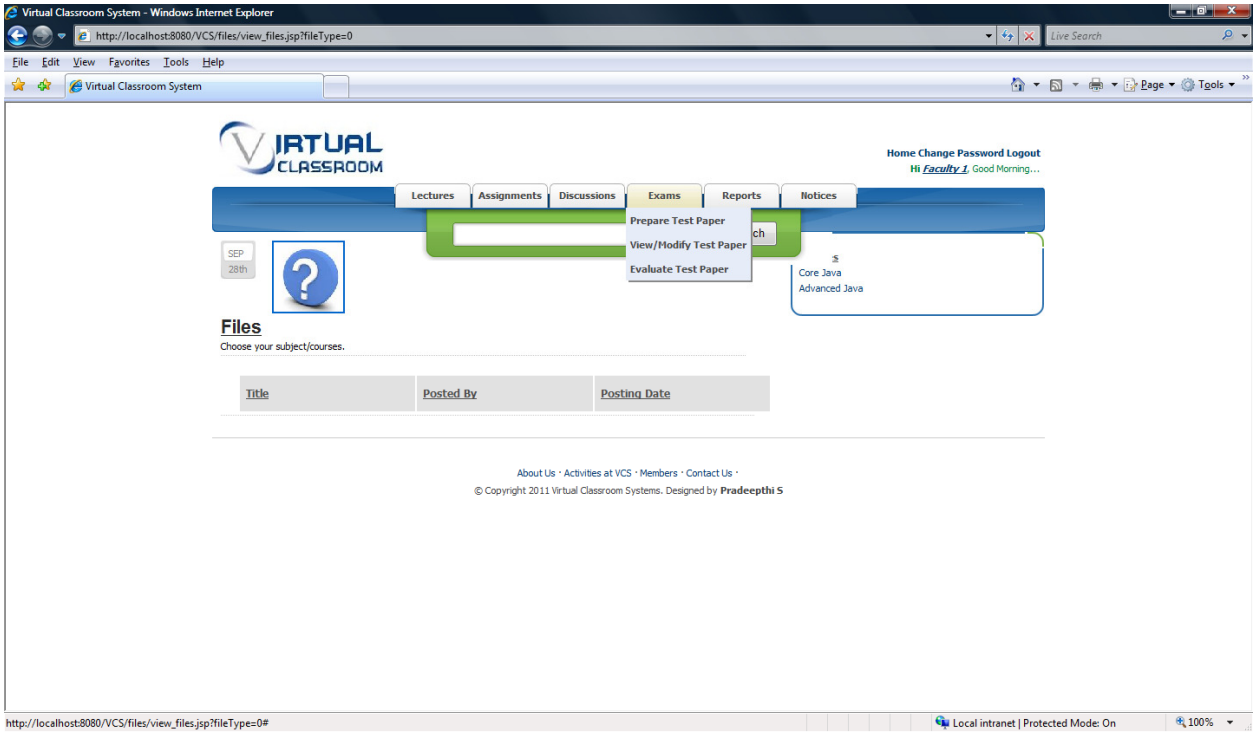


Figure 5.44: Exam menu options

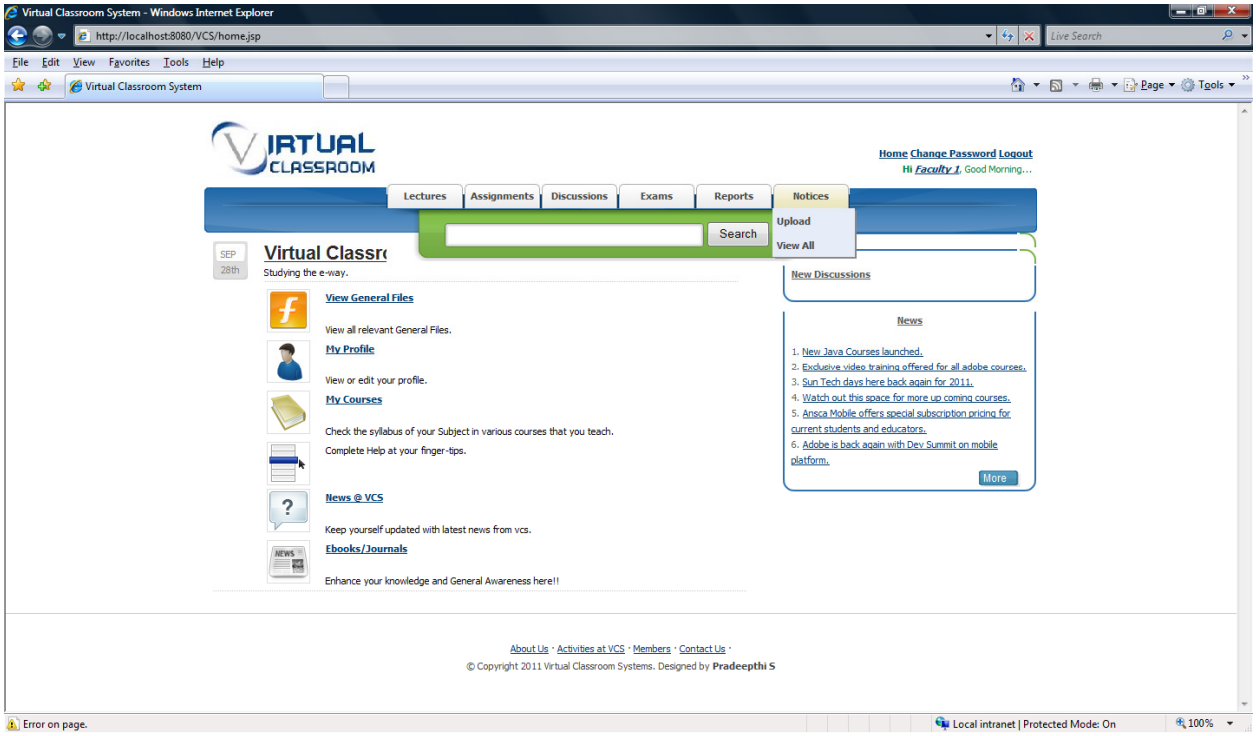


Figure 5.45: Notices menu options

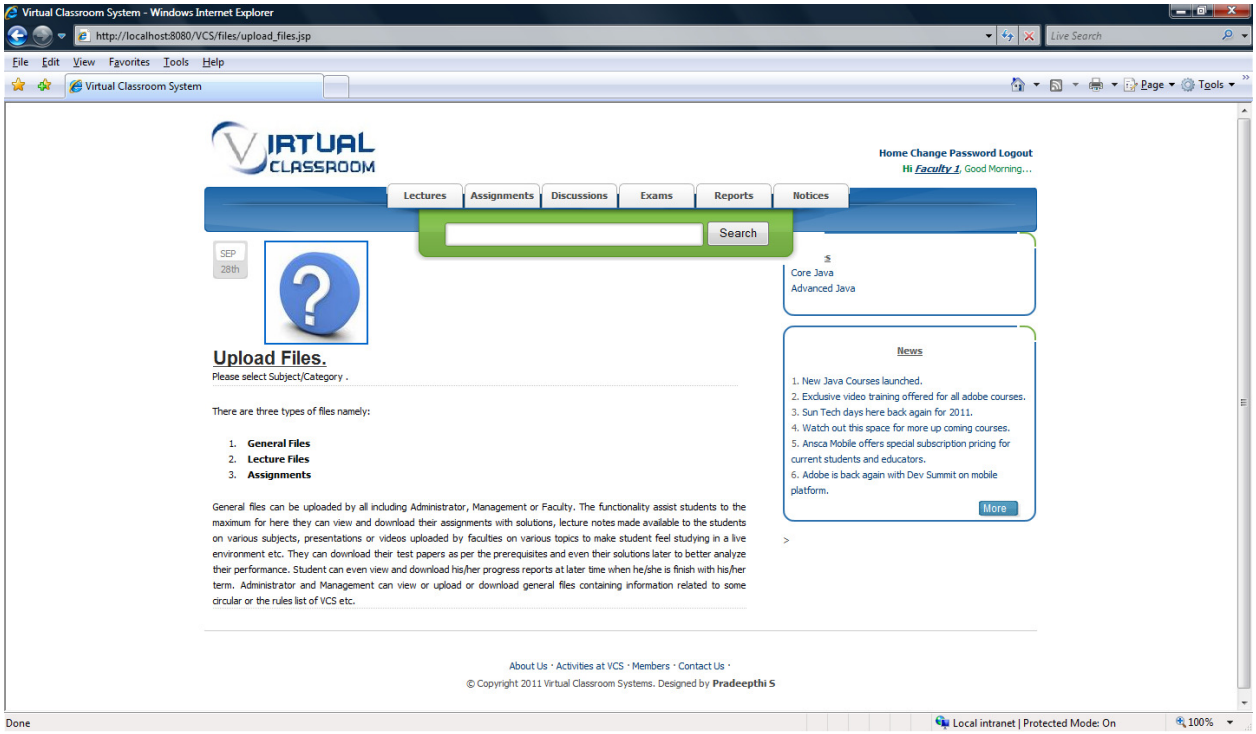


Figure 5.46: uploading files (general/lectures/assignments)

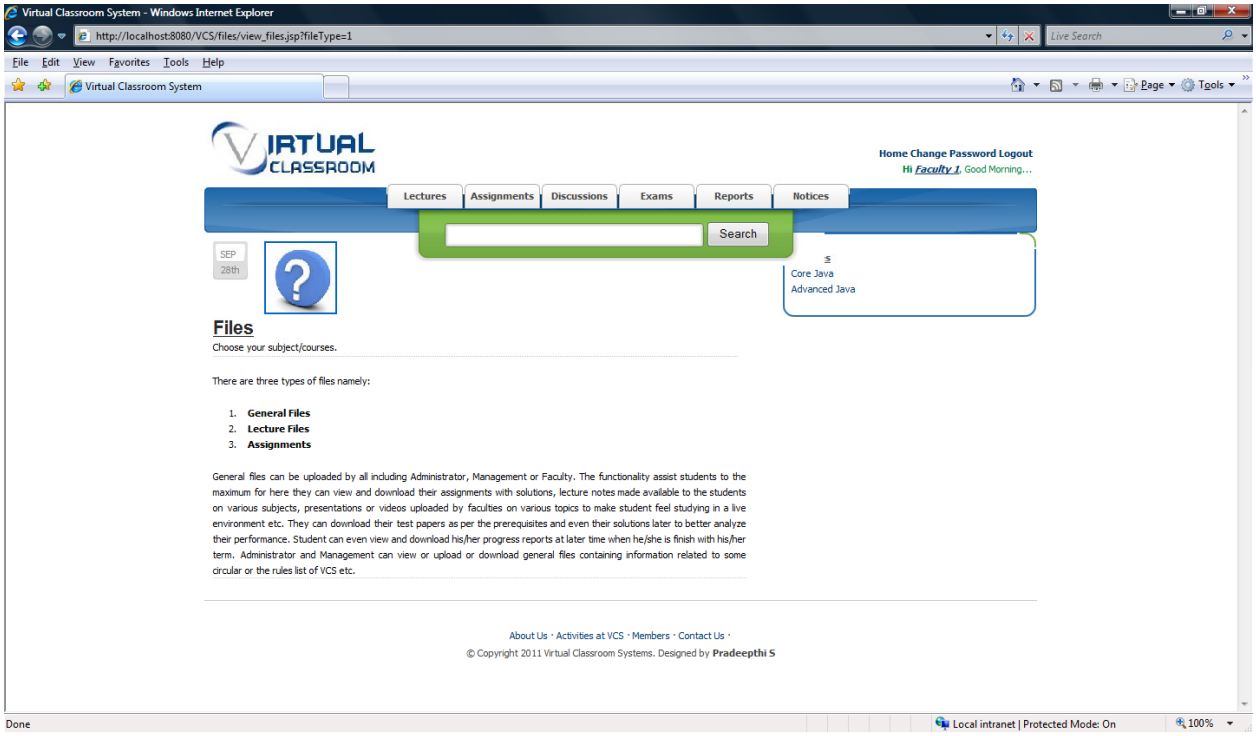


Figure 5.47: viewing files

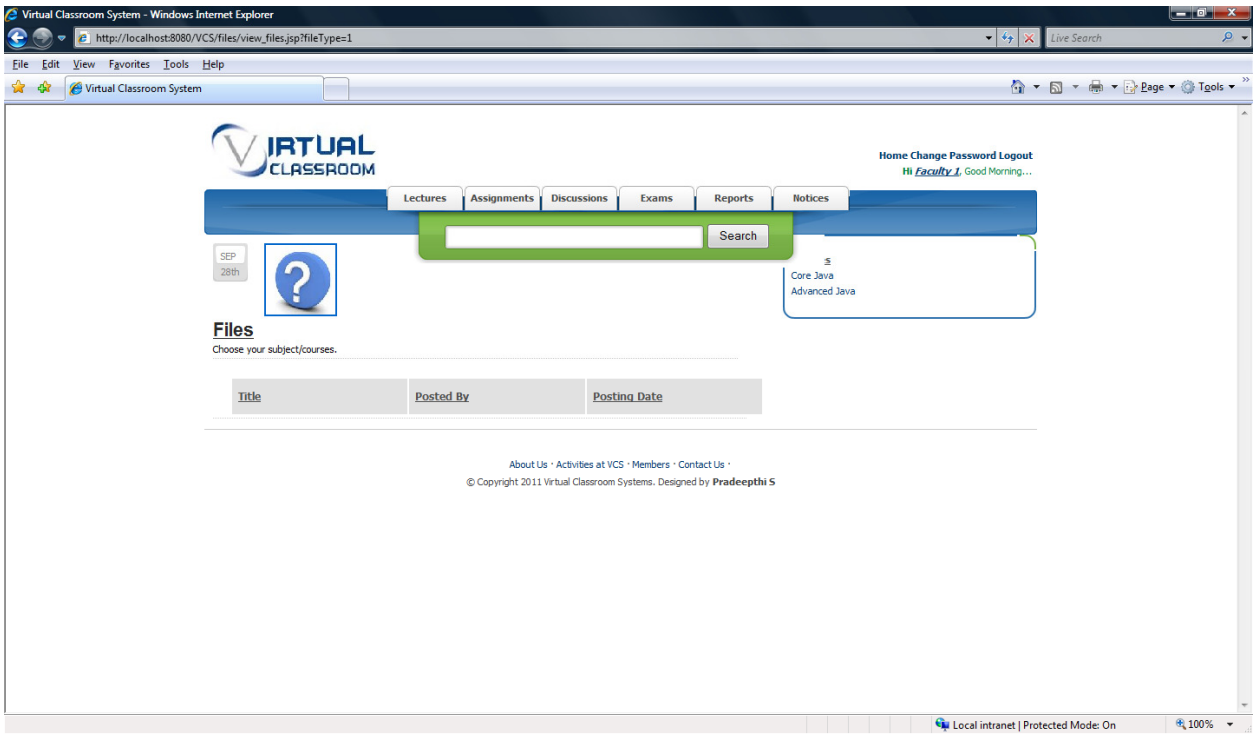


Figure 5.48: viewing files subject wise

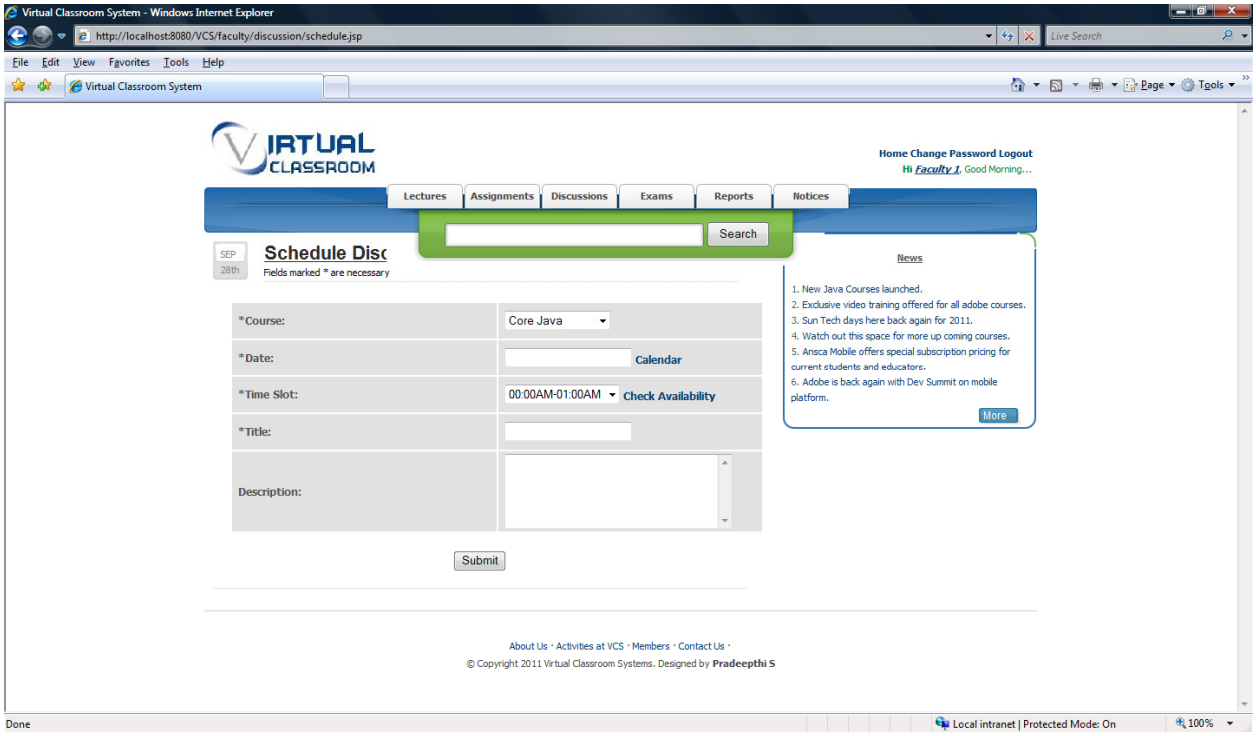


Figure 5.49: Scheduling a discussion

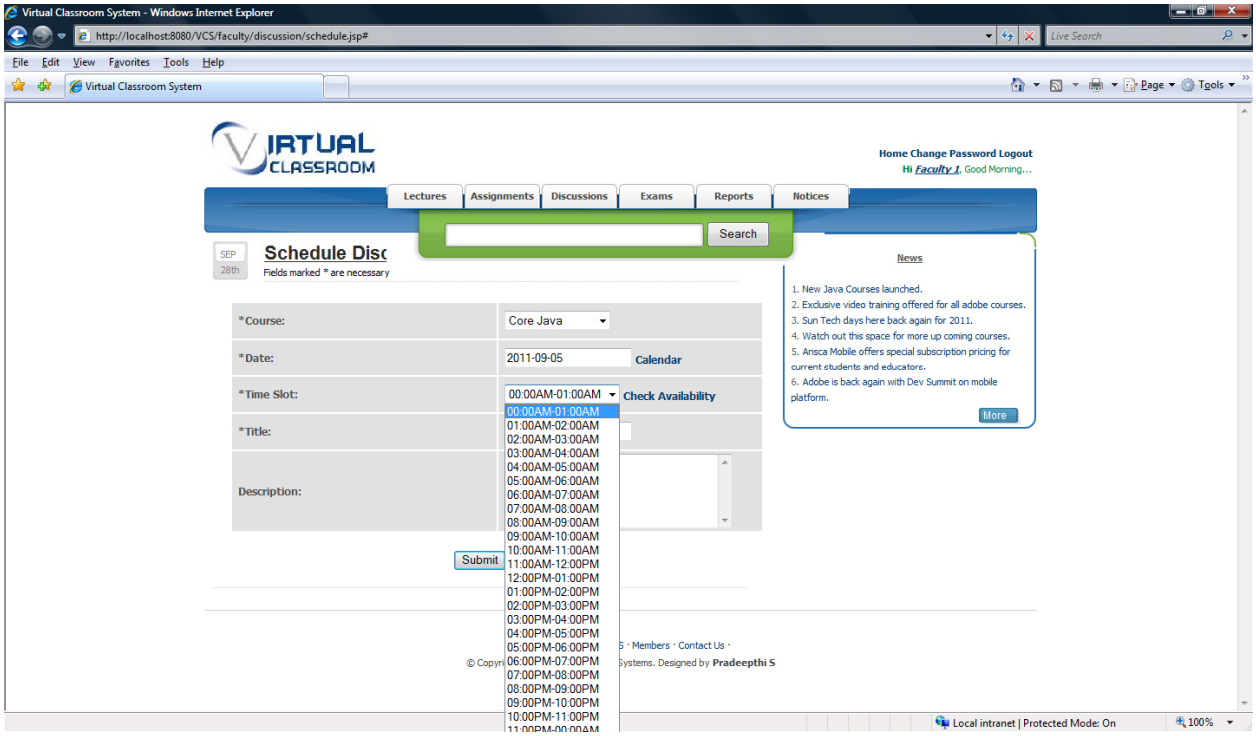


Figure 5.50: selecting time for discussion

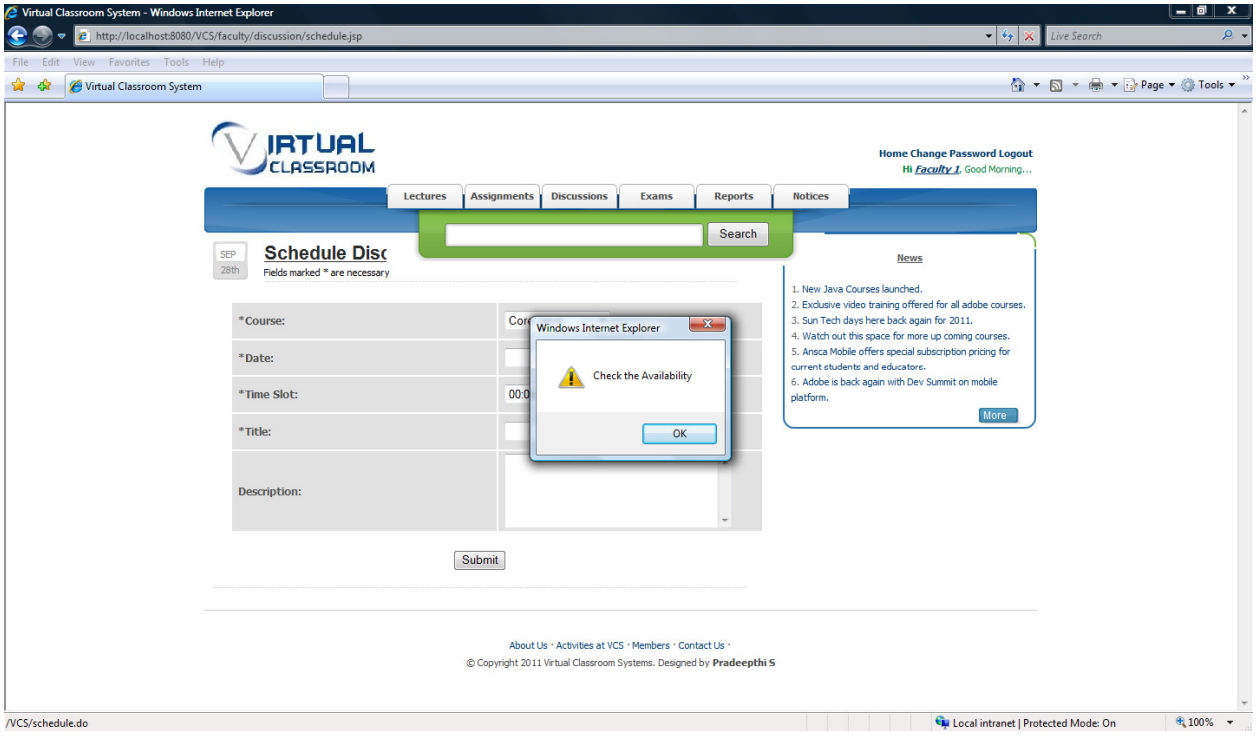


Figure 5.51: validating the availability while creating discussion

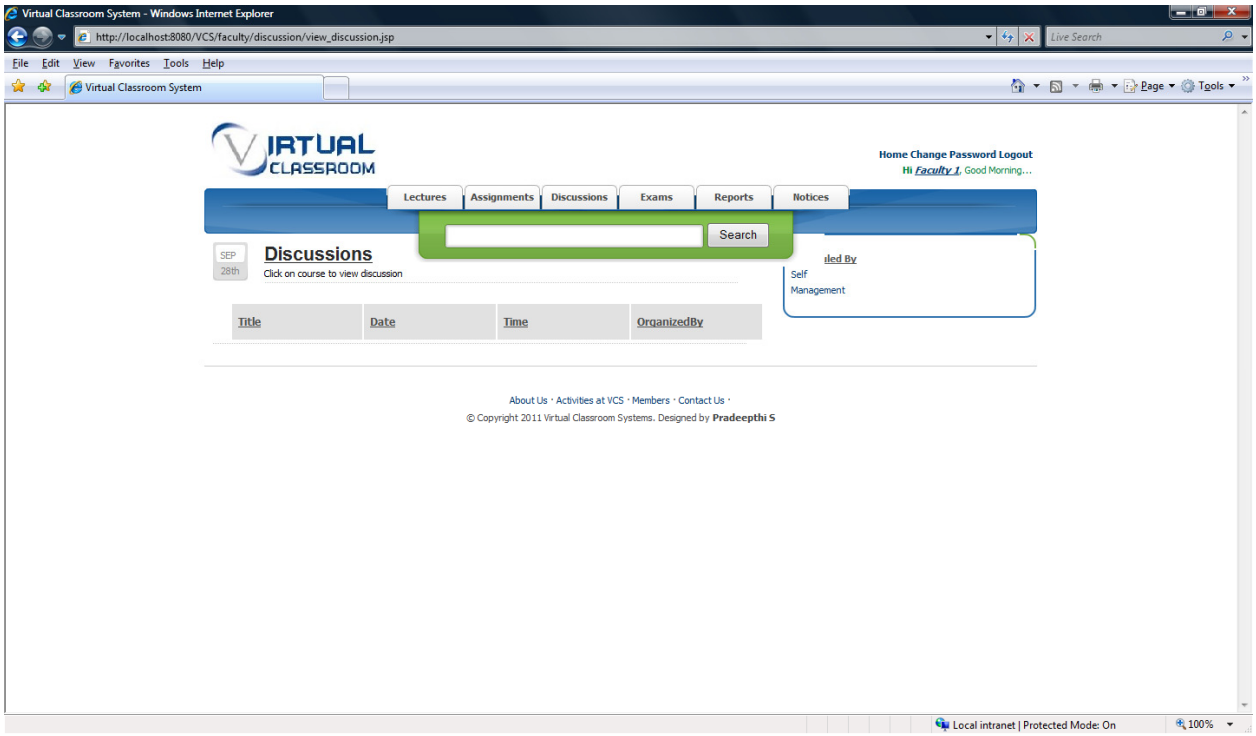


Figure 5.52: viewing list of all discussions

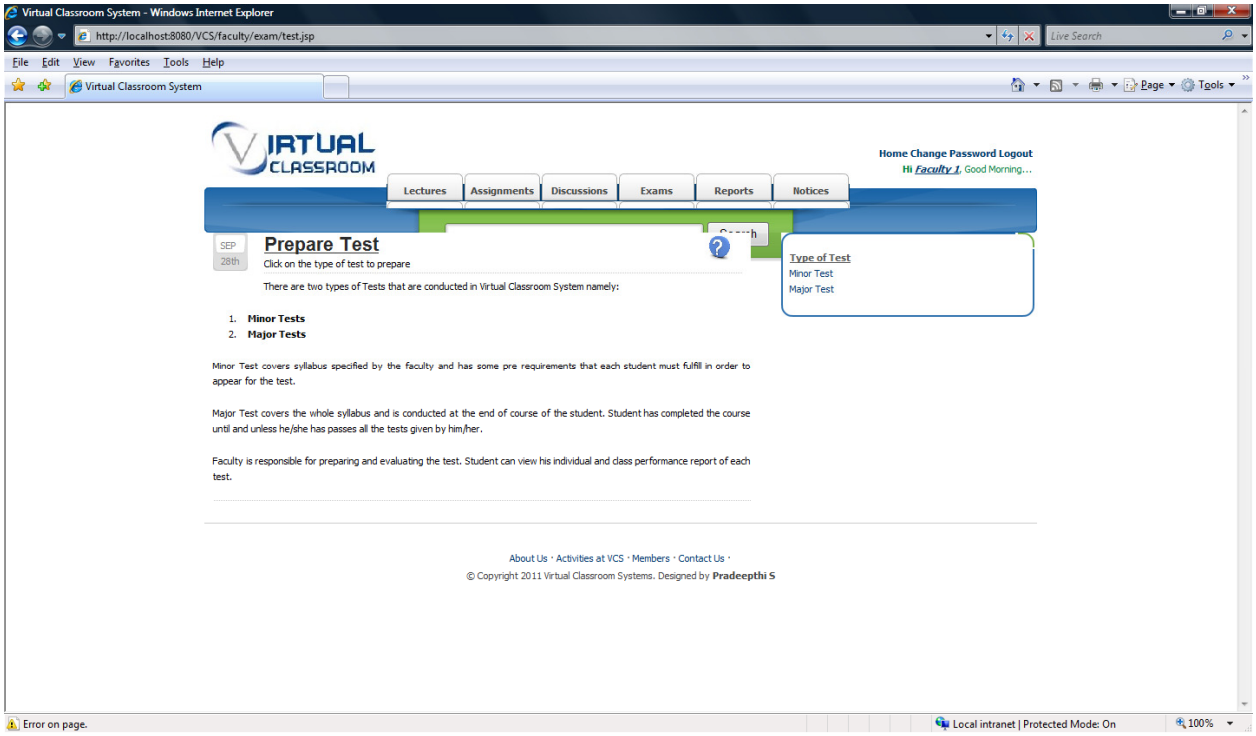


Figure 5.53: Preparing for test

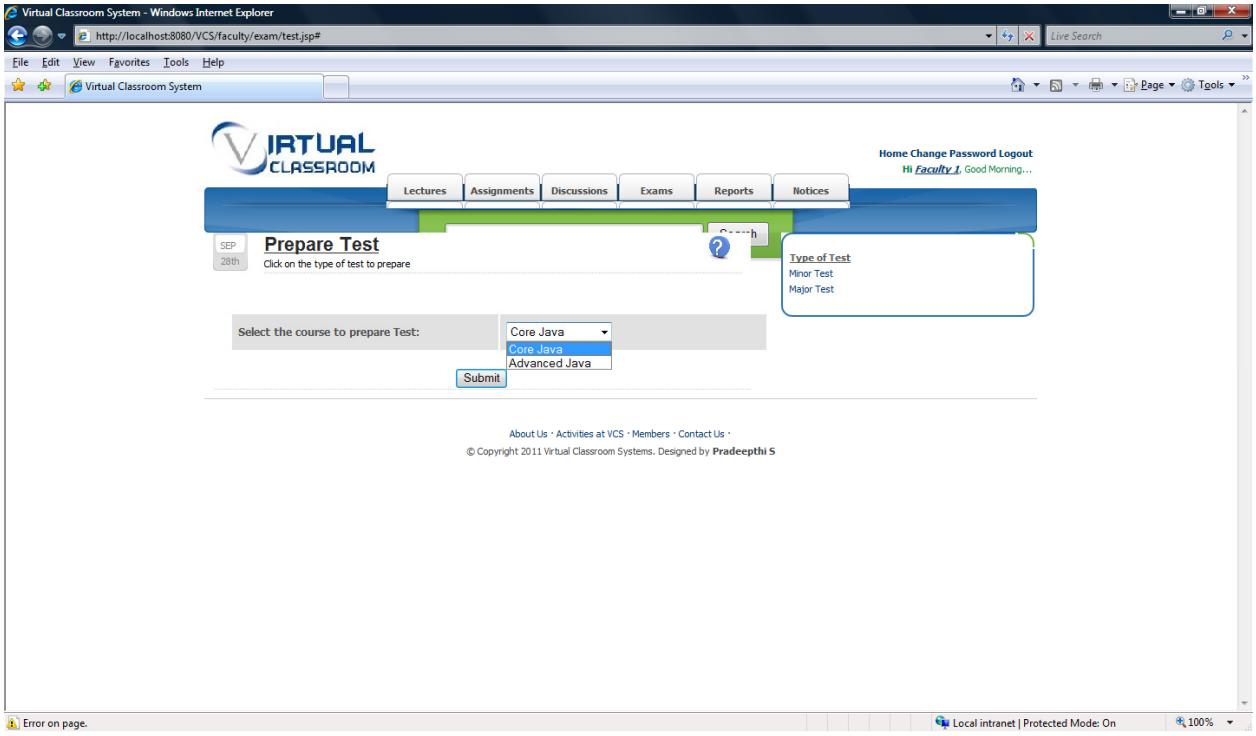


Figure 5.54: Selecting subject

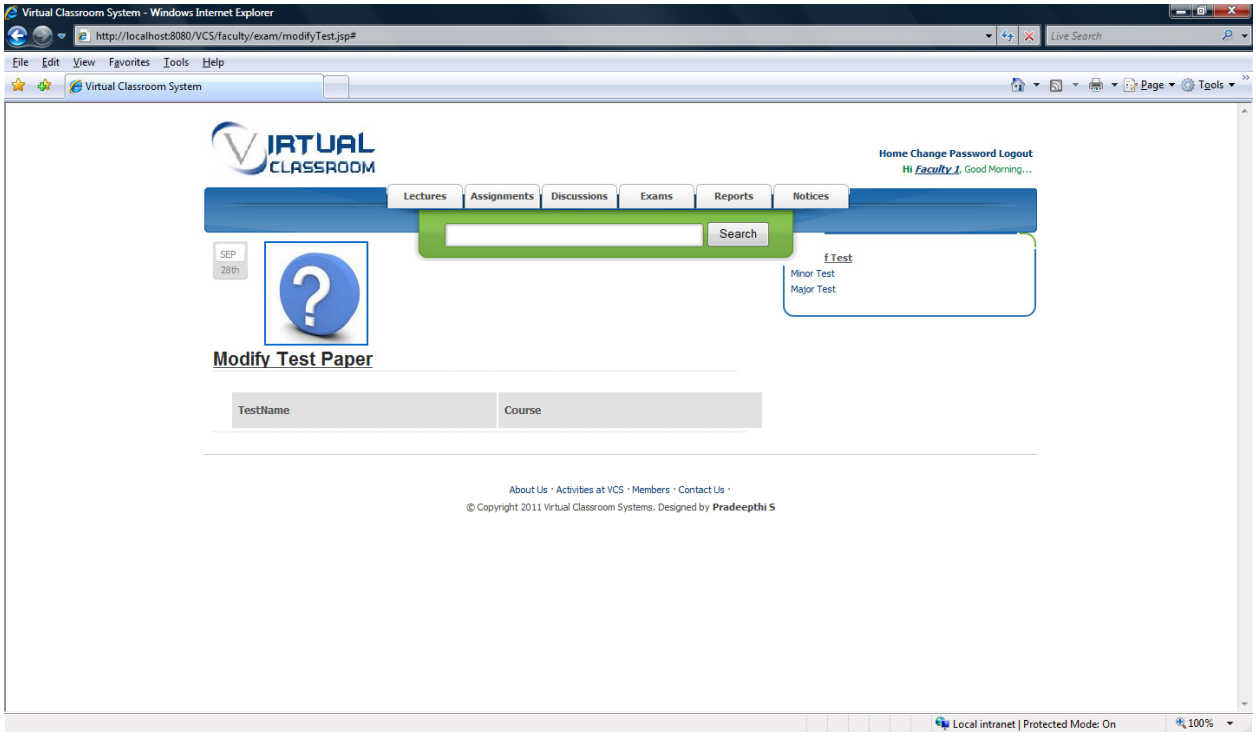


Figure 5.55: Modifying the test paper evaluation

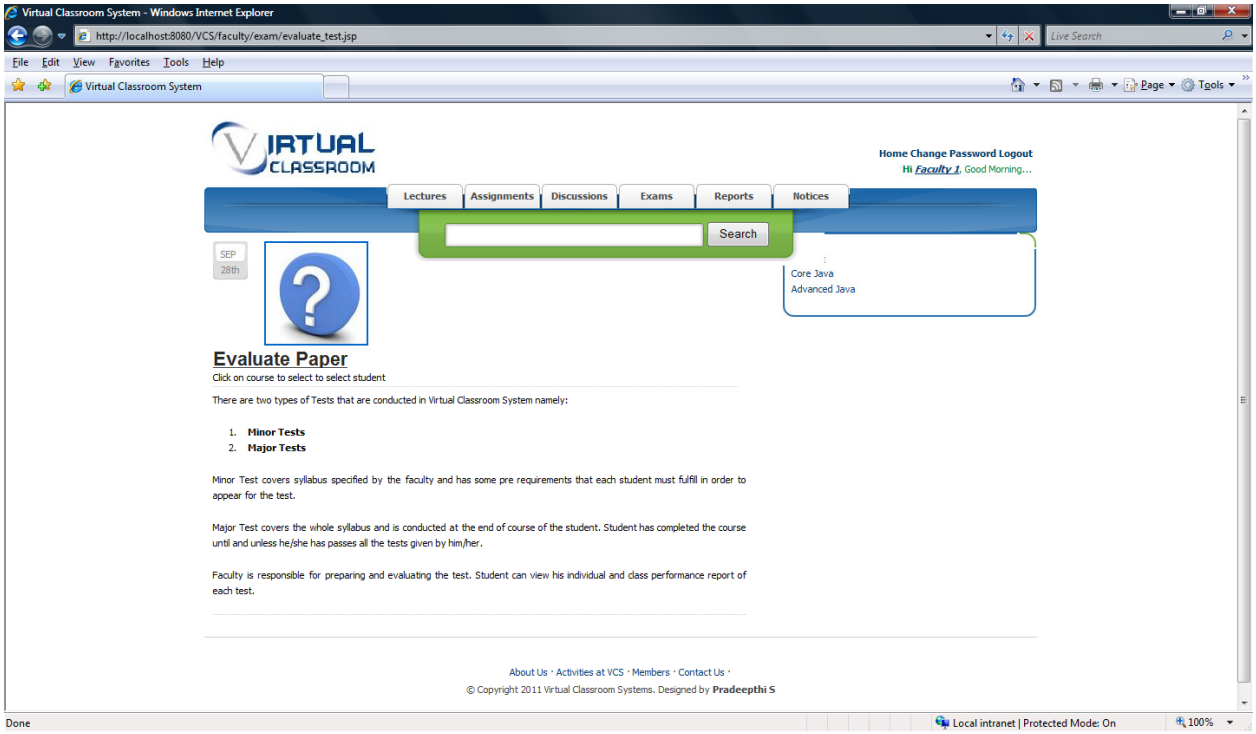


Figure 5.56: Evaluating the paper

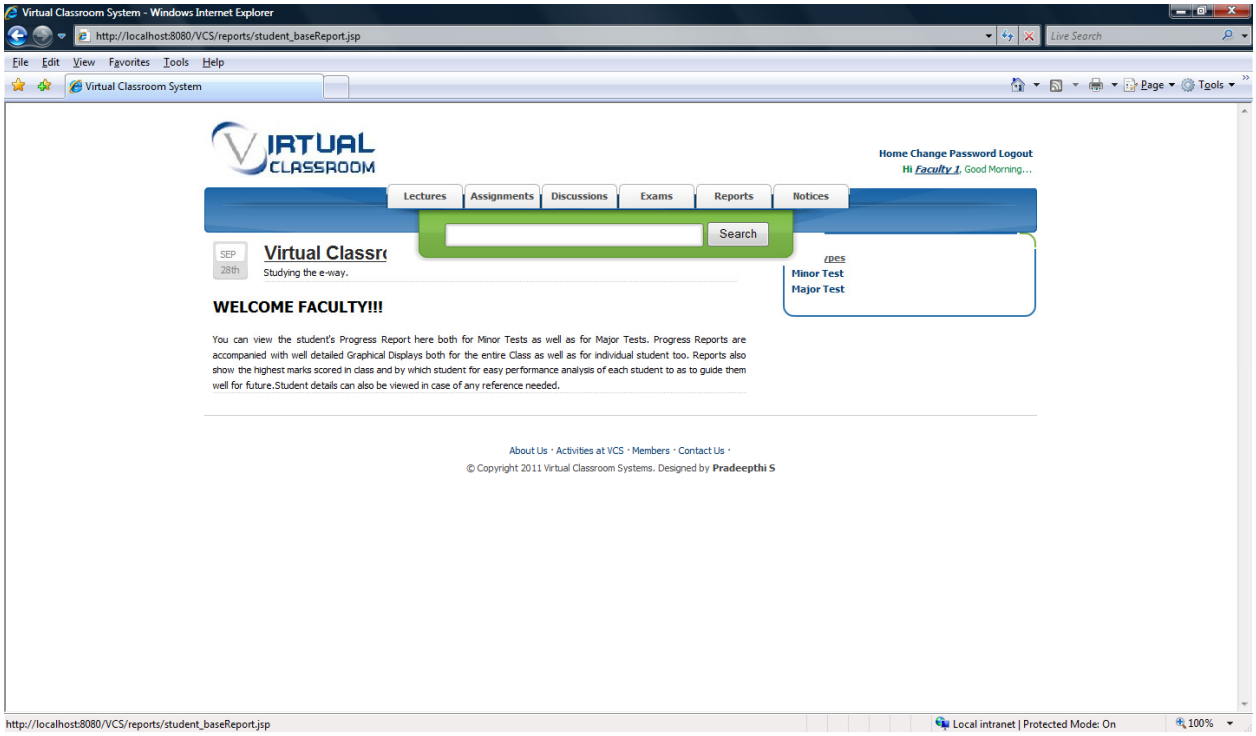


Figure 5.57: Tests creation

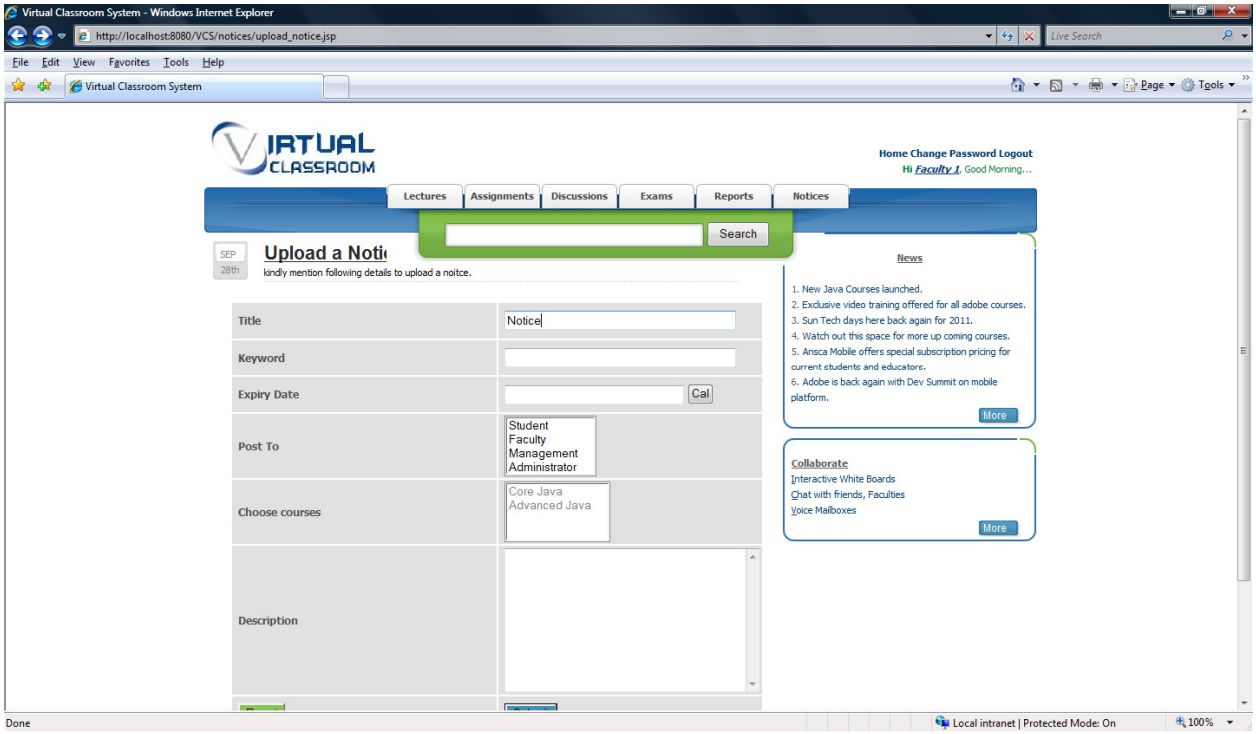


Figure 5.58: uploading notices

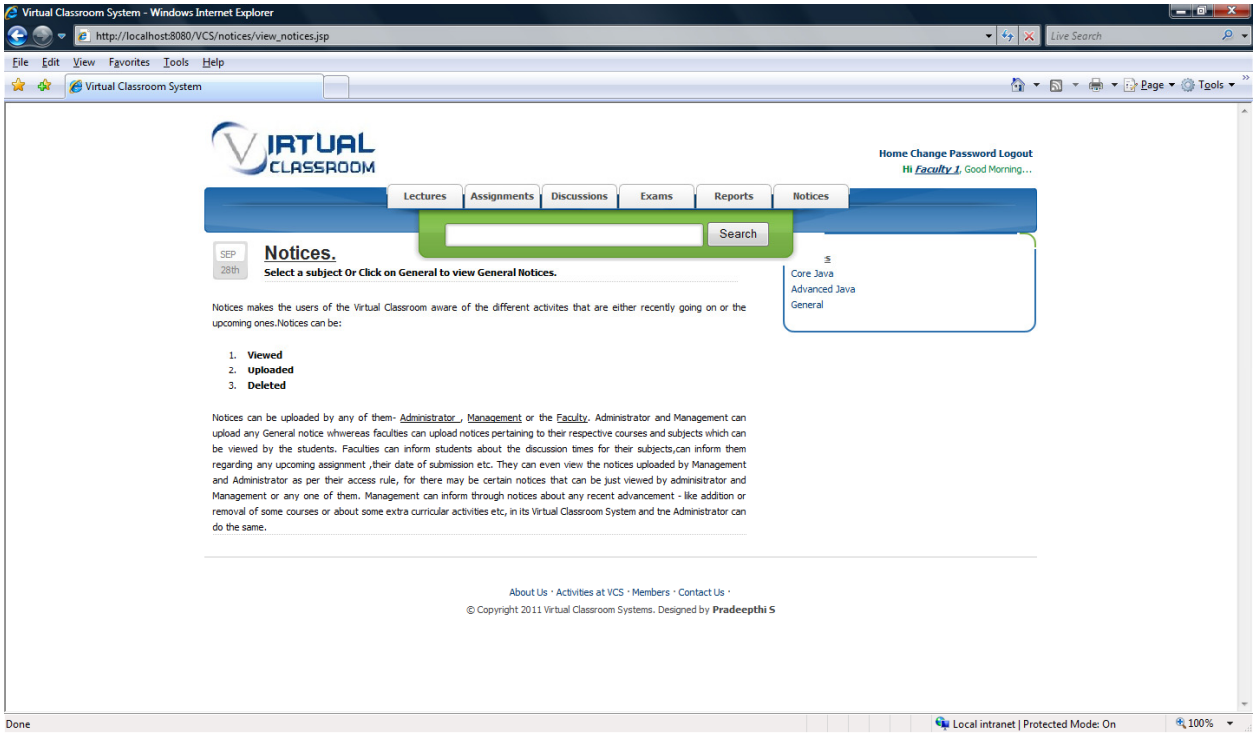


Figure 5.59: viewing notices subject wise

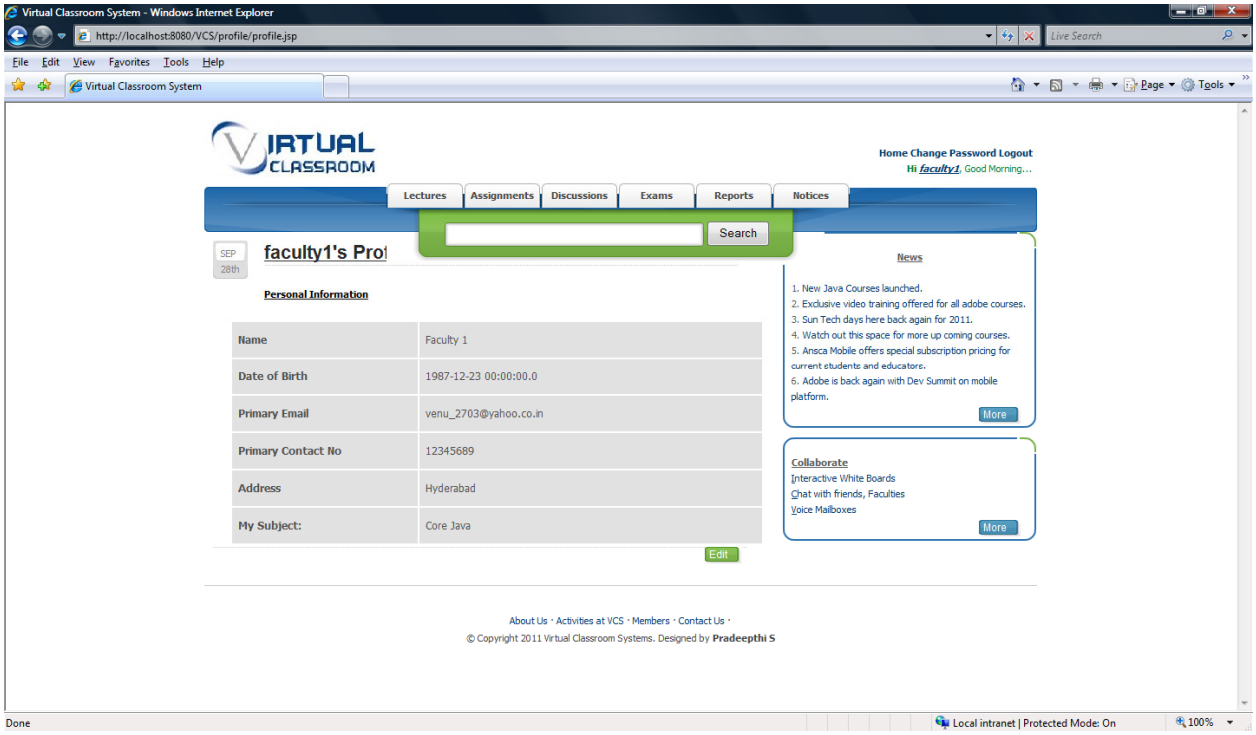


Figure 5.60: Faculty profile

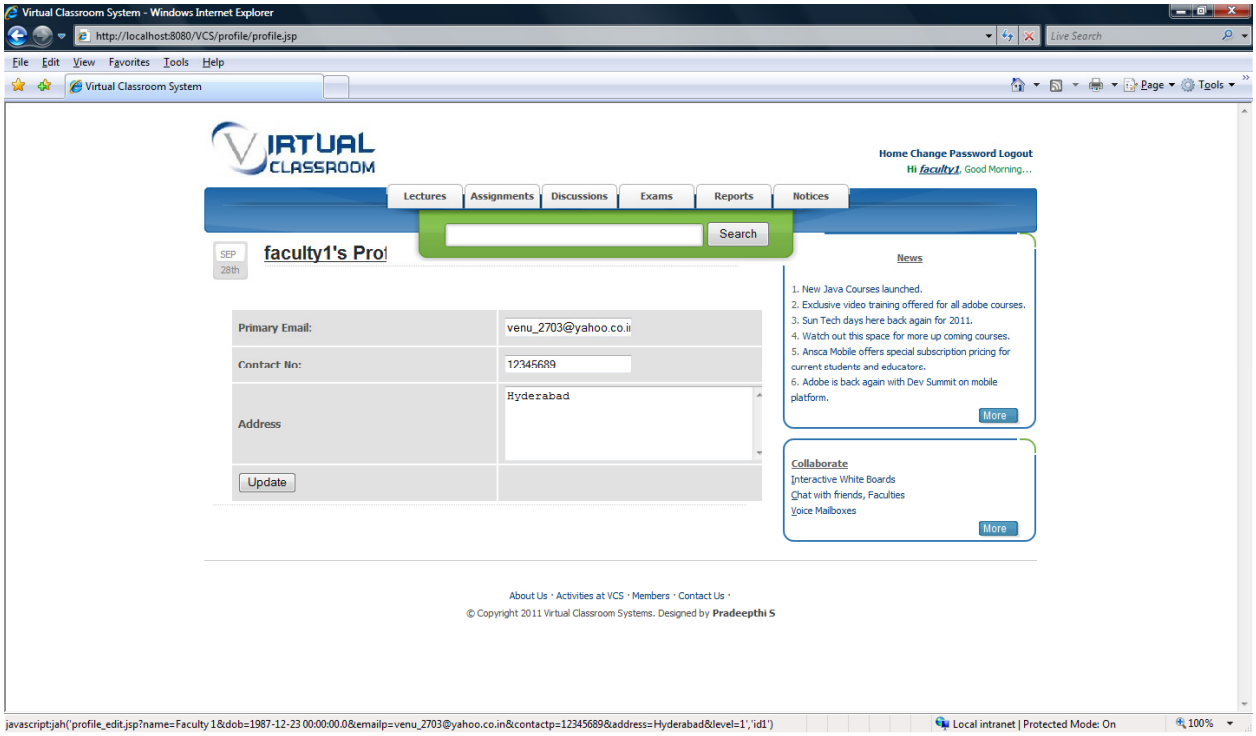


Figure 5.61: Editing profile

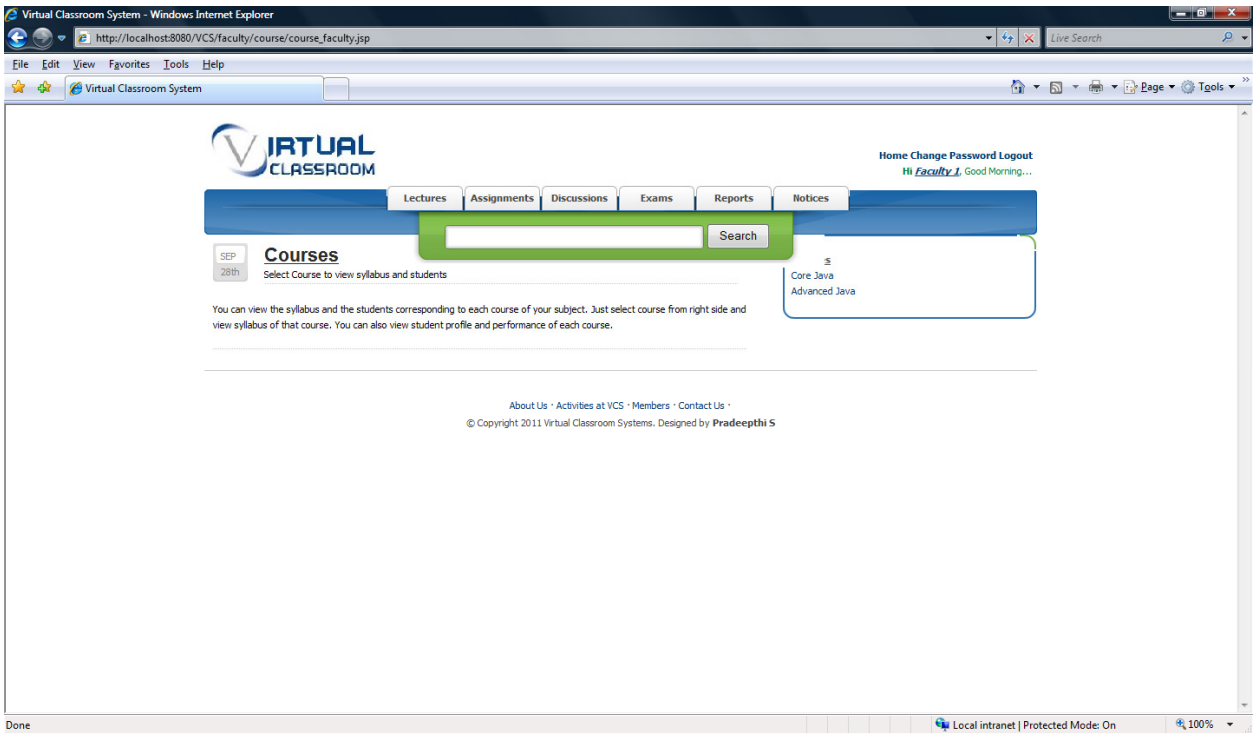


Figure 5.62: Viewing courses -Faculty

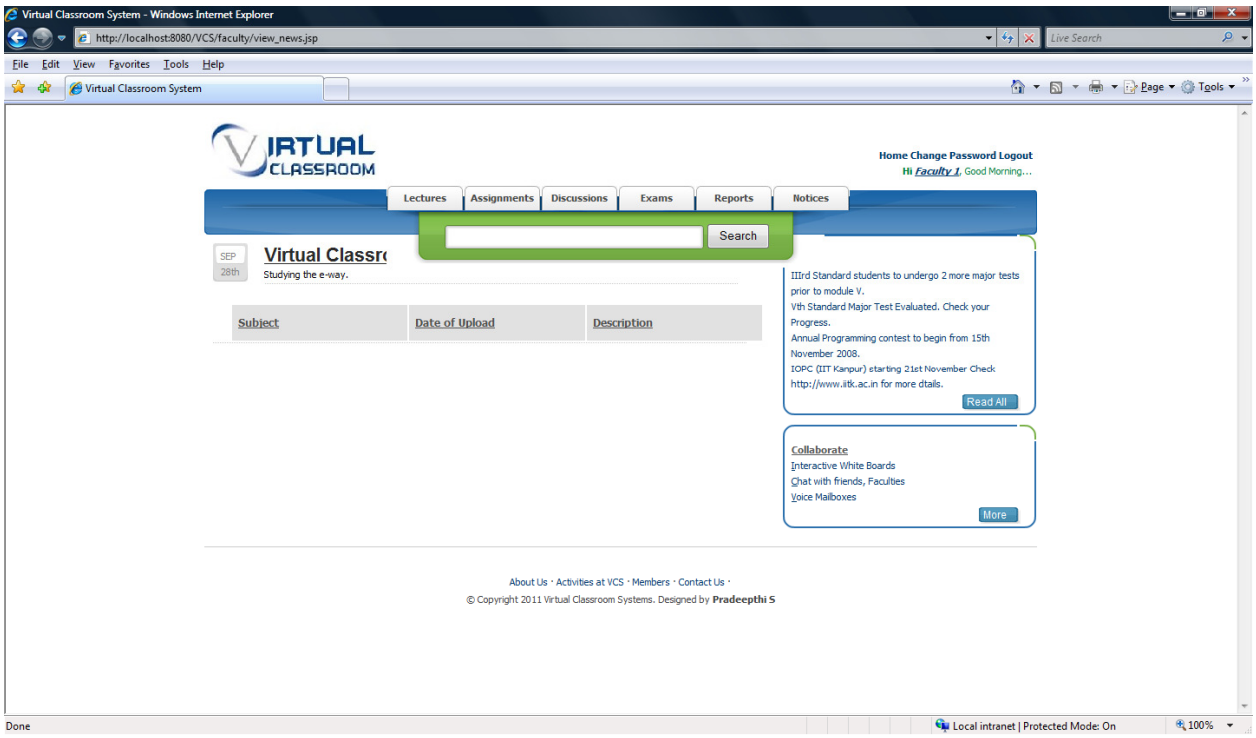


Figure 5.63: Viewing courses -Faculty

Management

Figure 5.64: Management user home page

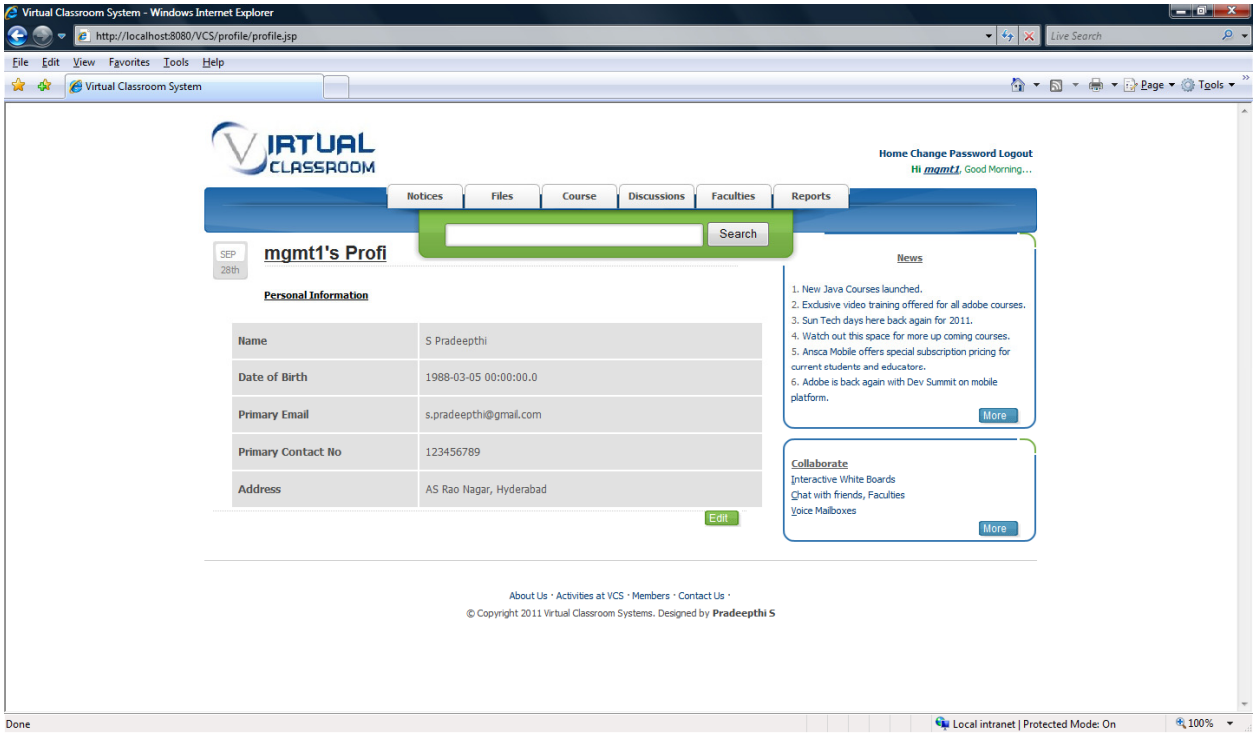


Figure 5.65: management user profile display

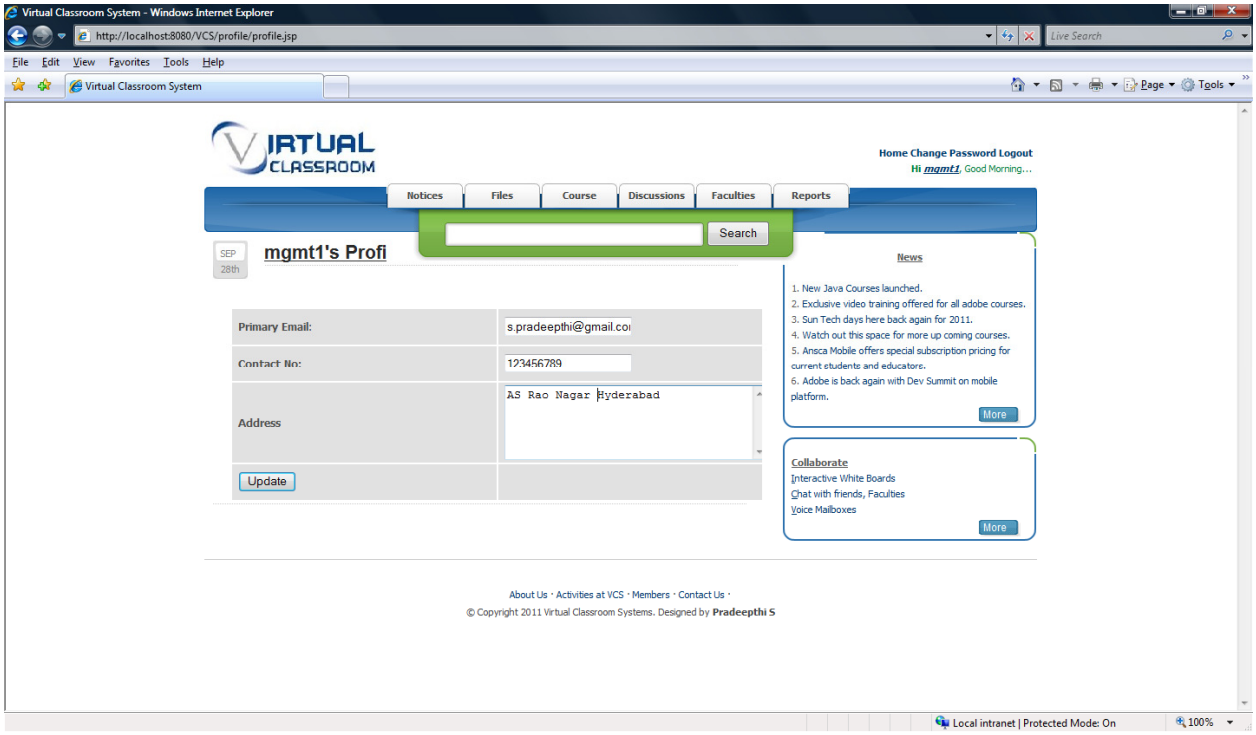


Figure 5.66: Edit the profile

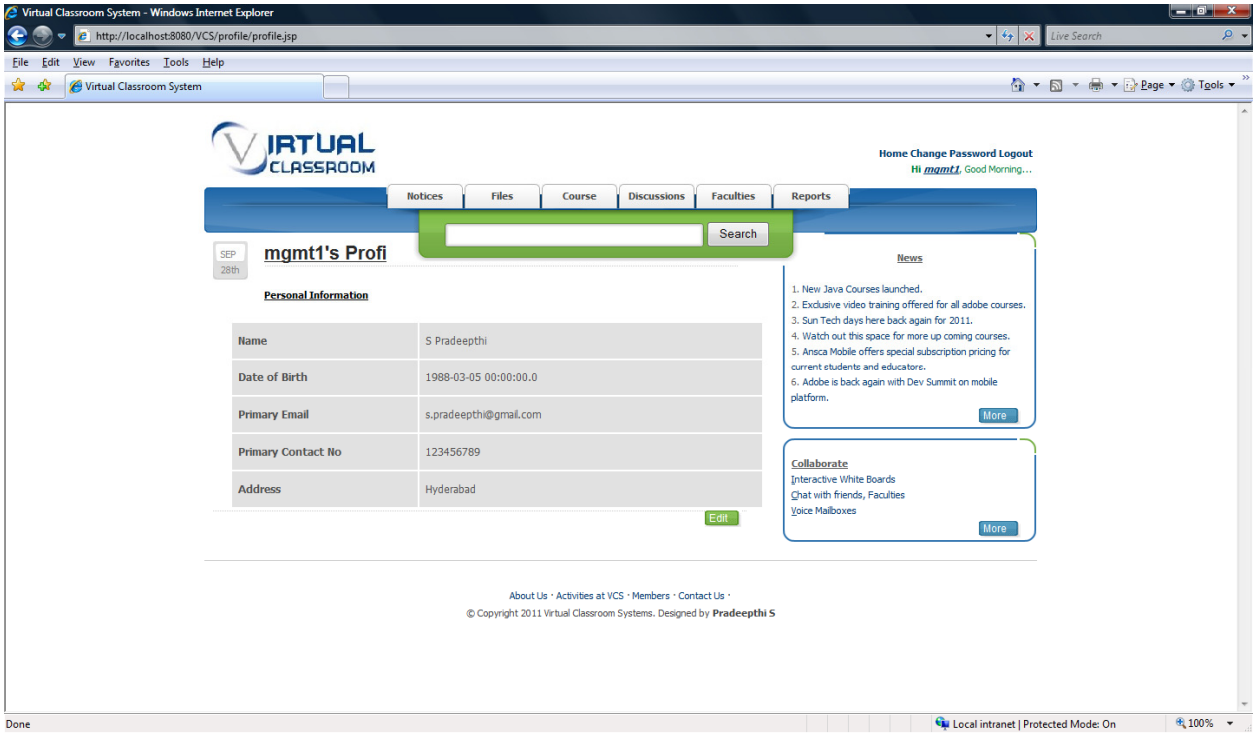


Figure 5.67: modified profile

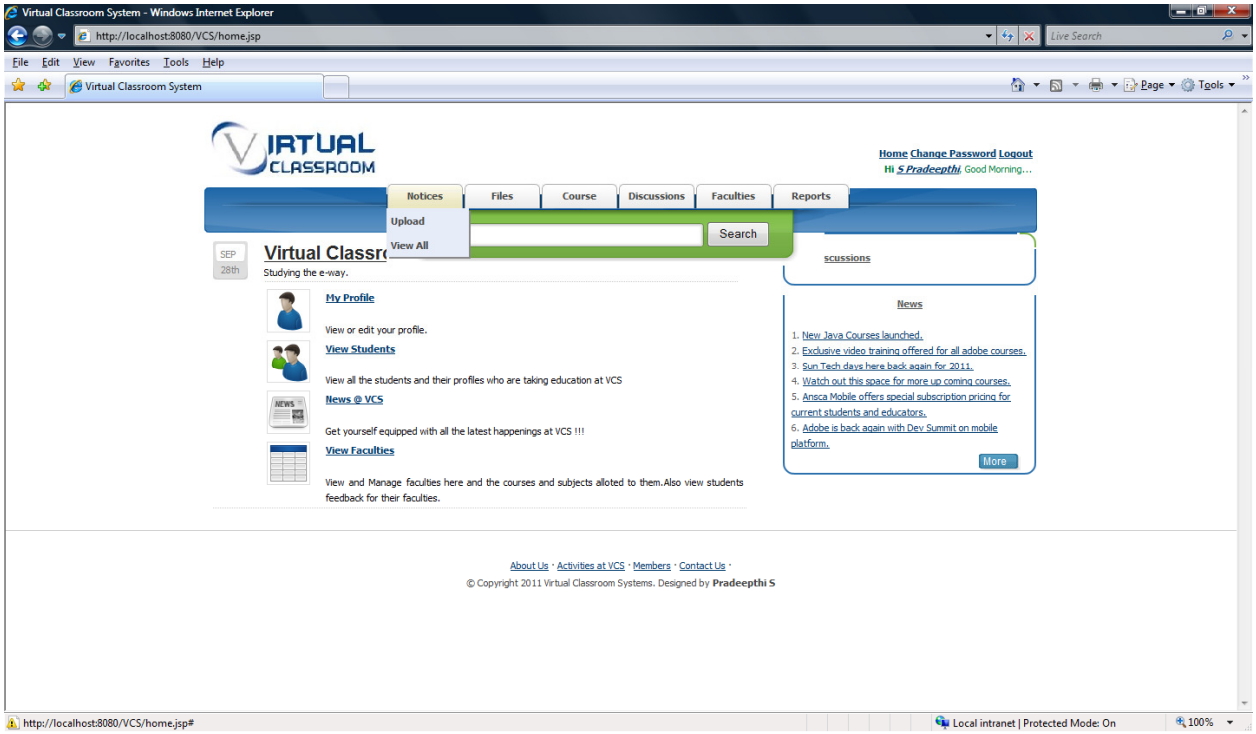


Figure 5.68: Notices menu option available to management user

Figure 5.69: Files menu options for management user

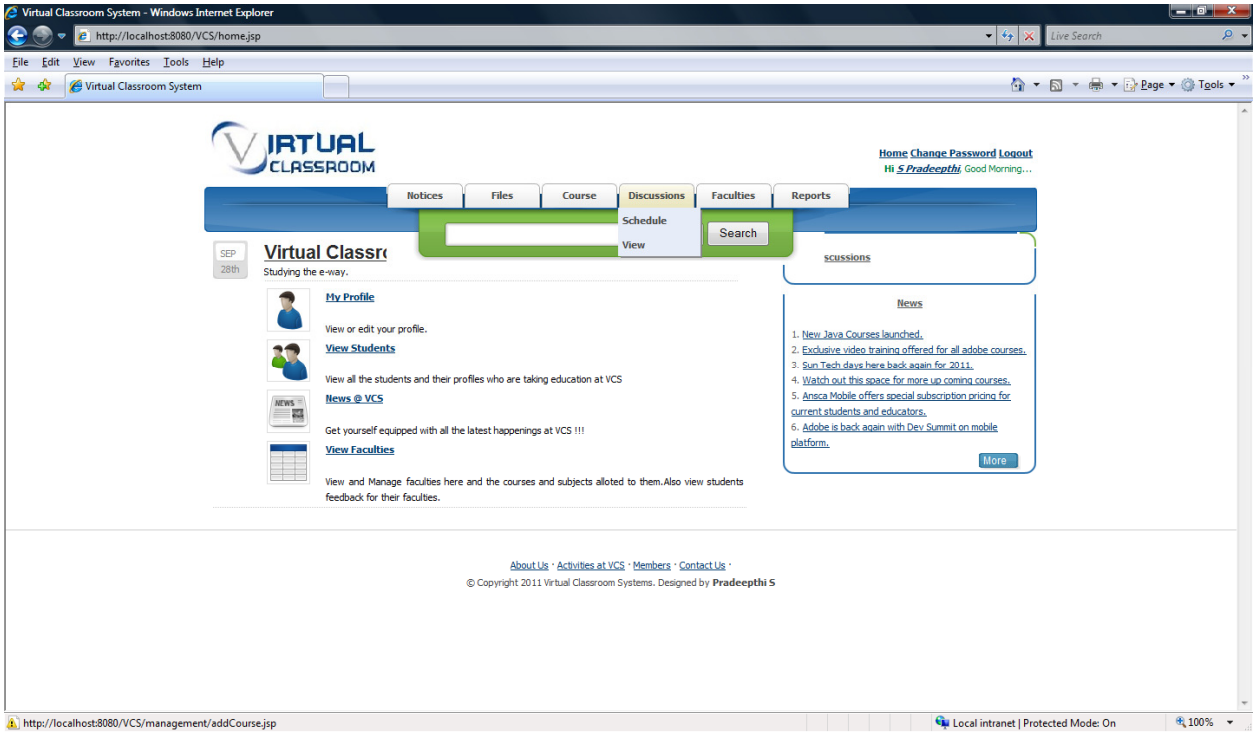


Figure 5.70: discussion menu options

Figure 5.71: Reports available for management user

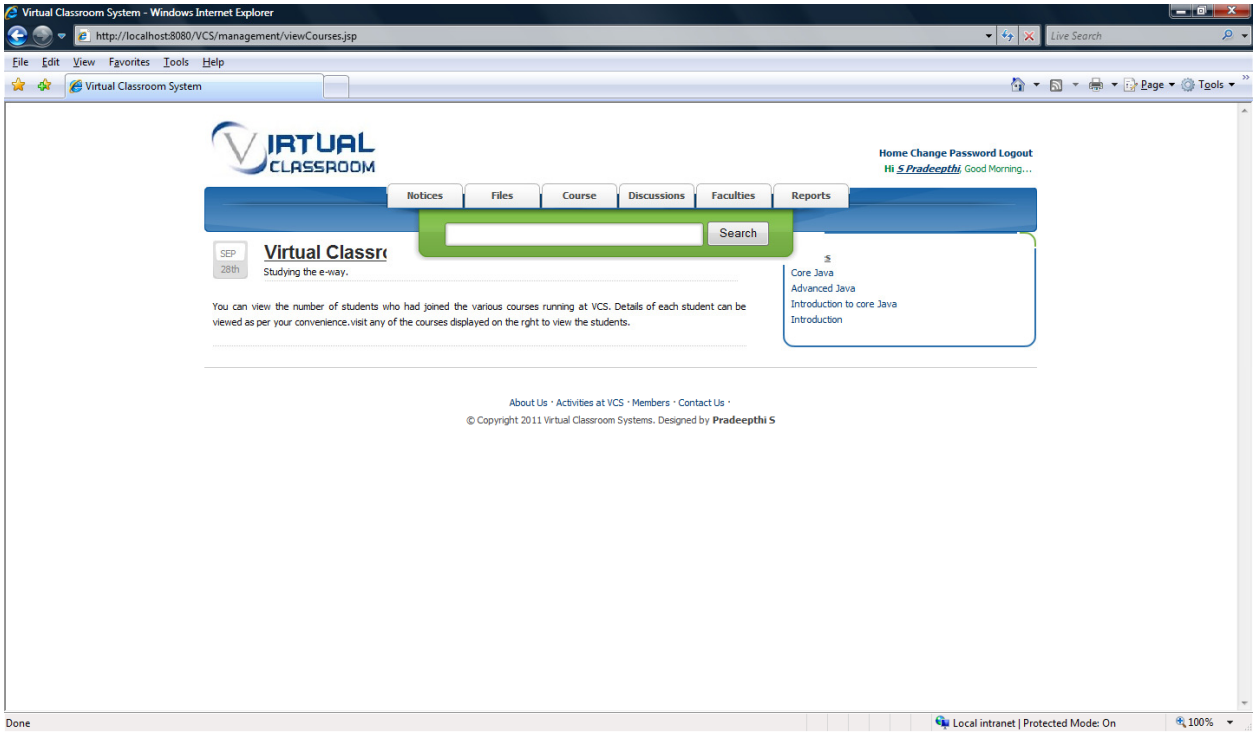


Figure 5.72: Faculty report subject wise

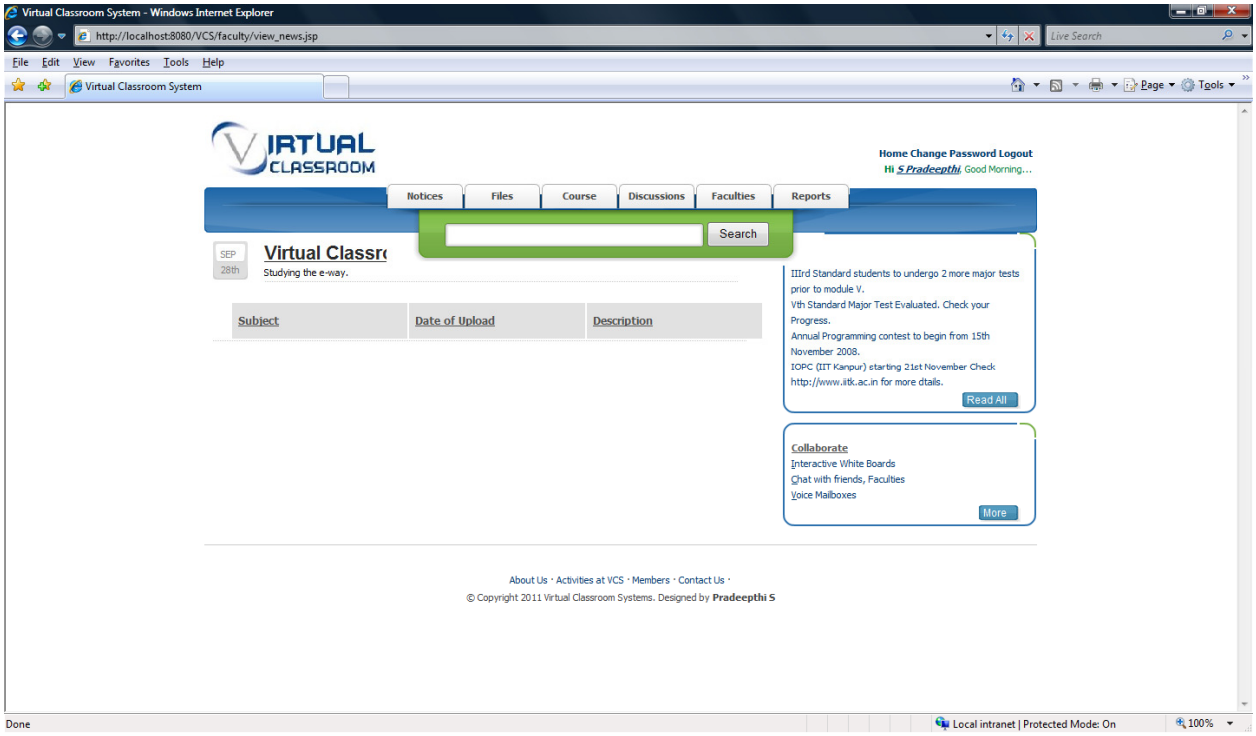


Figure 5.73: Student report

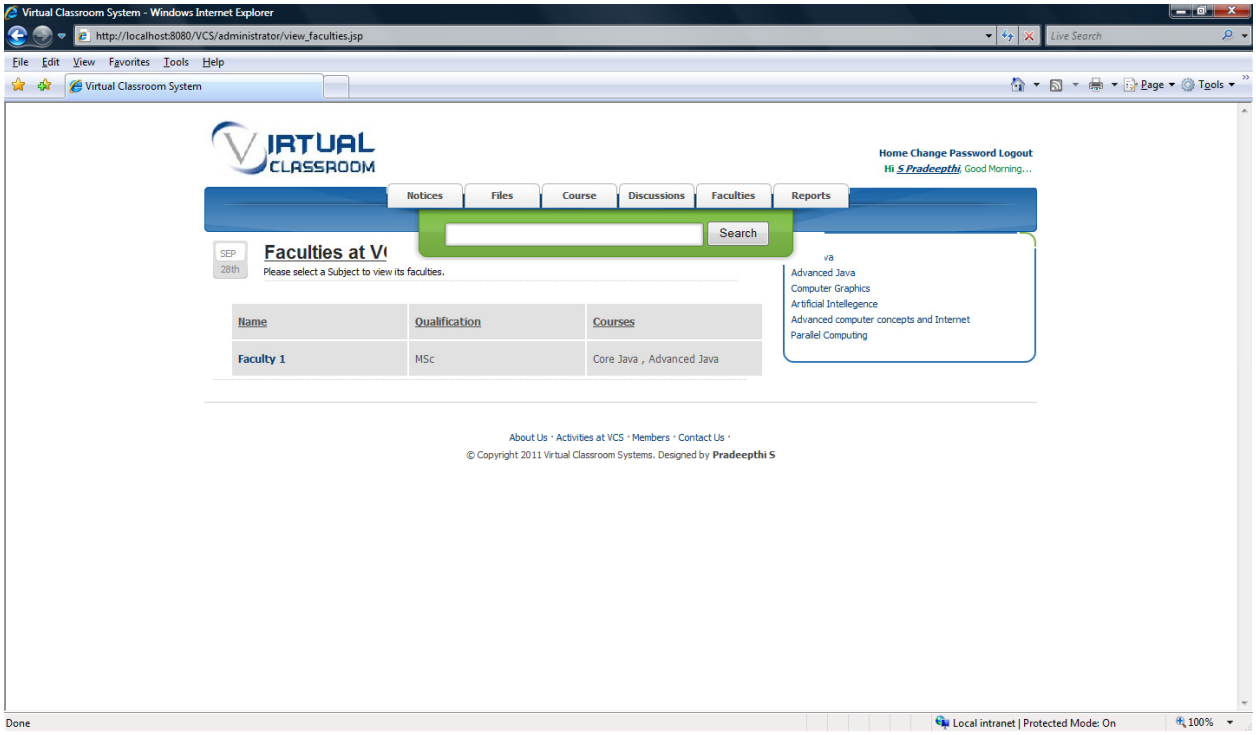


Figure 5.74: Listing the faculties available

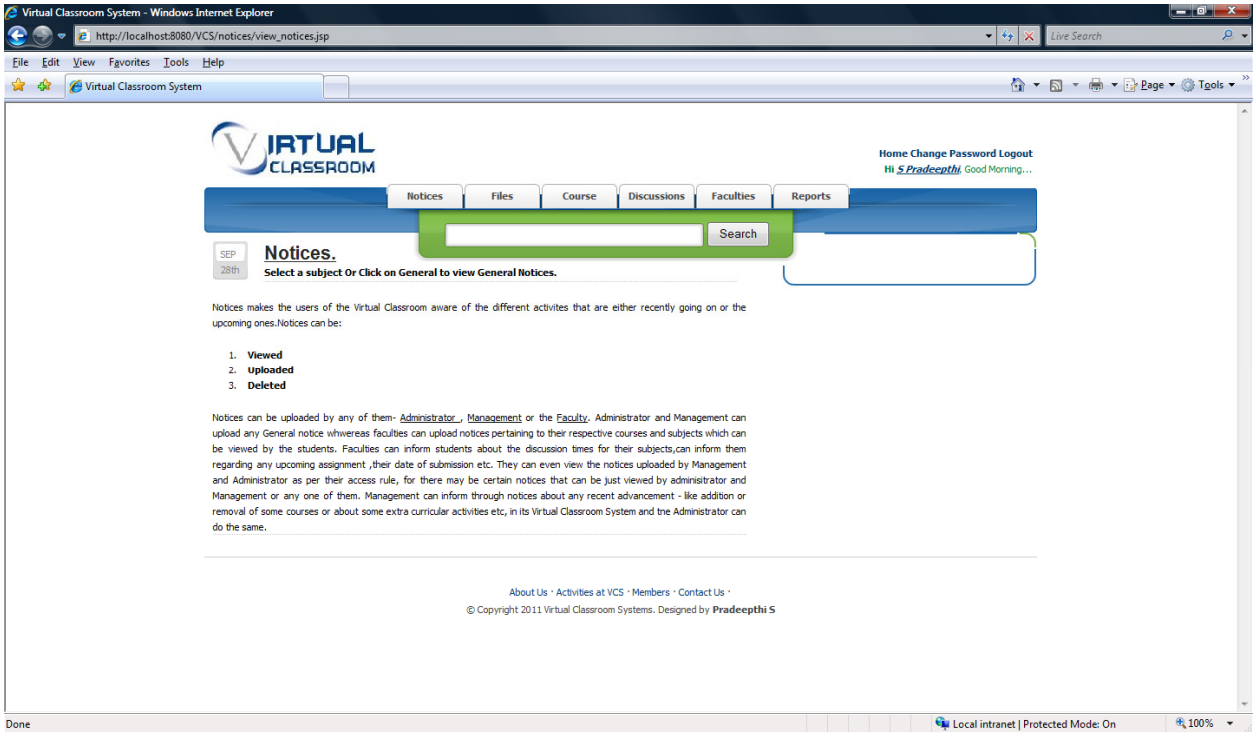


Figure 5.75: Notices listing

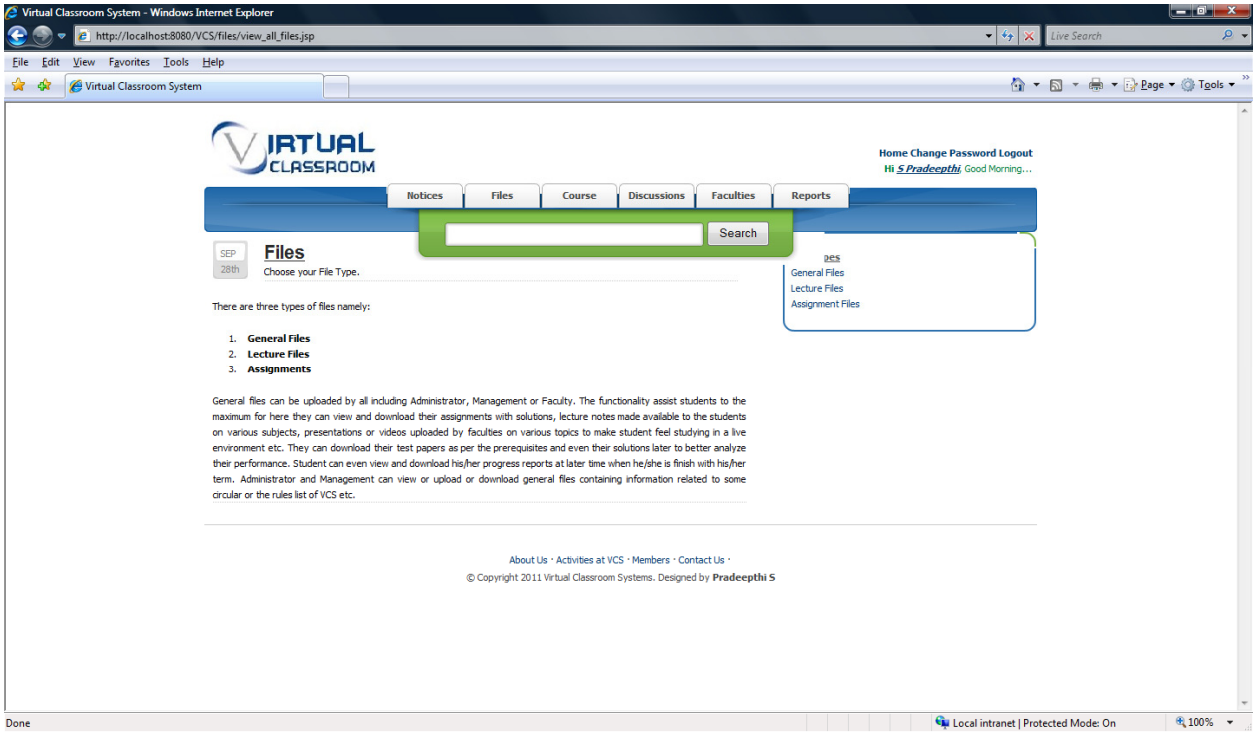


Figure 5.76: Files display

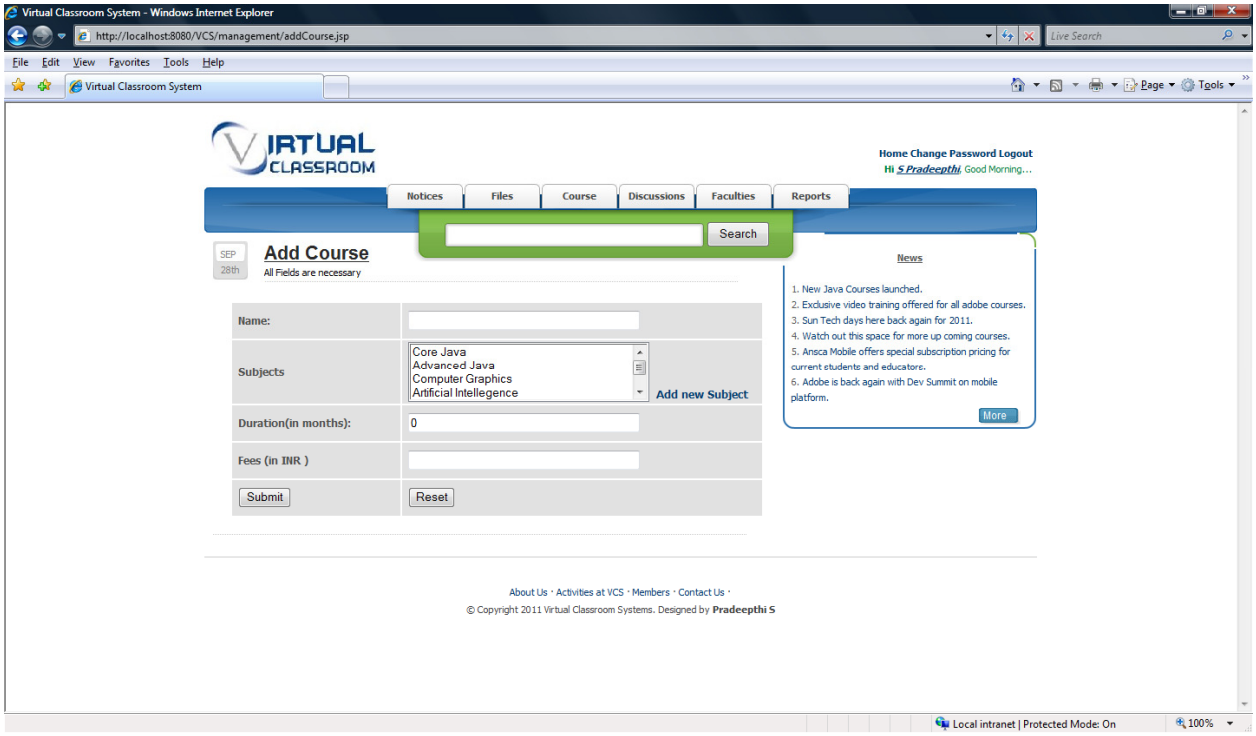


Figure 5.77: Adding course

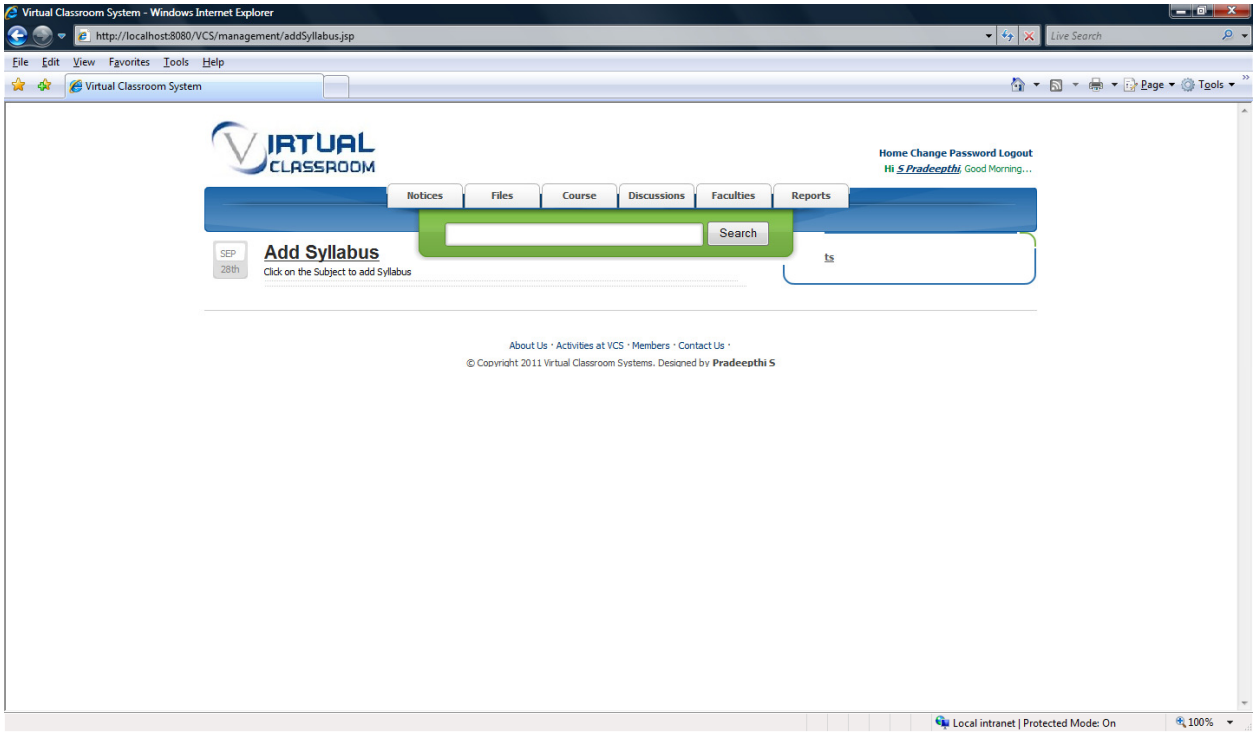


Figure 5.78: Adding syllabus for a particular course

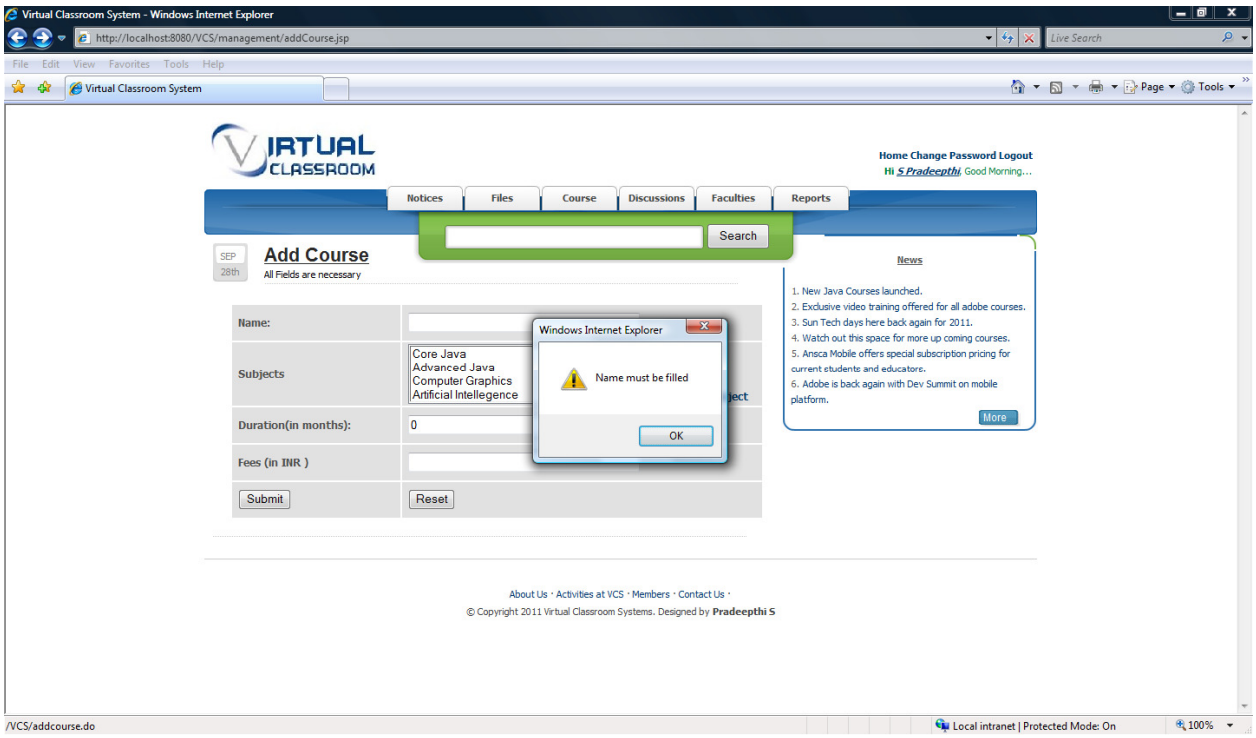


Figure 5.79: Adding course validation

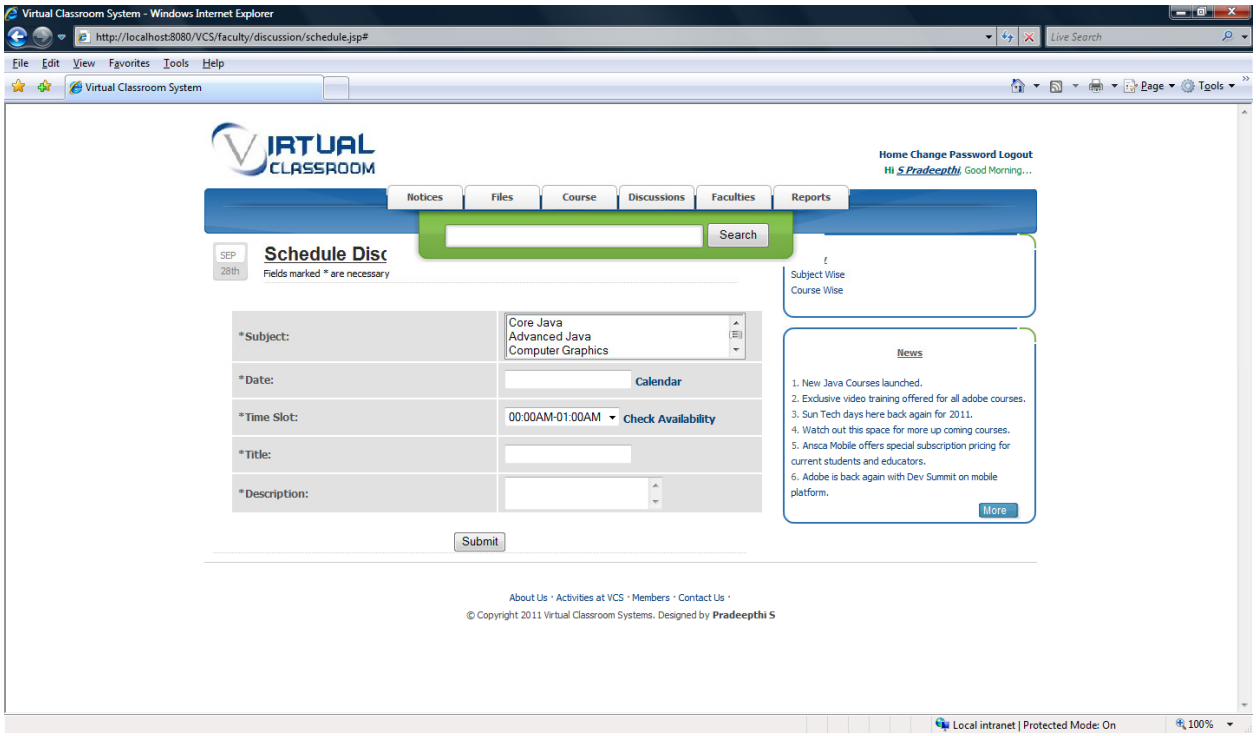


Figure 5.80: Scheduling for a discussion

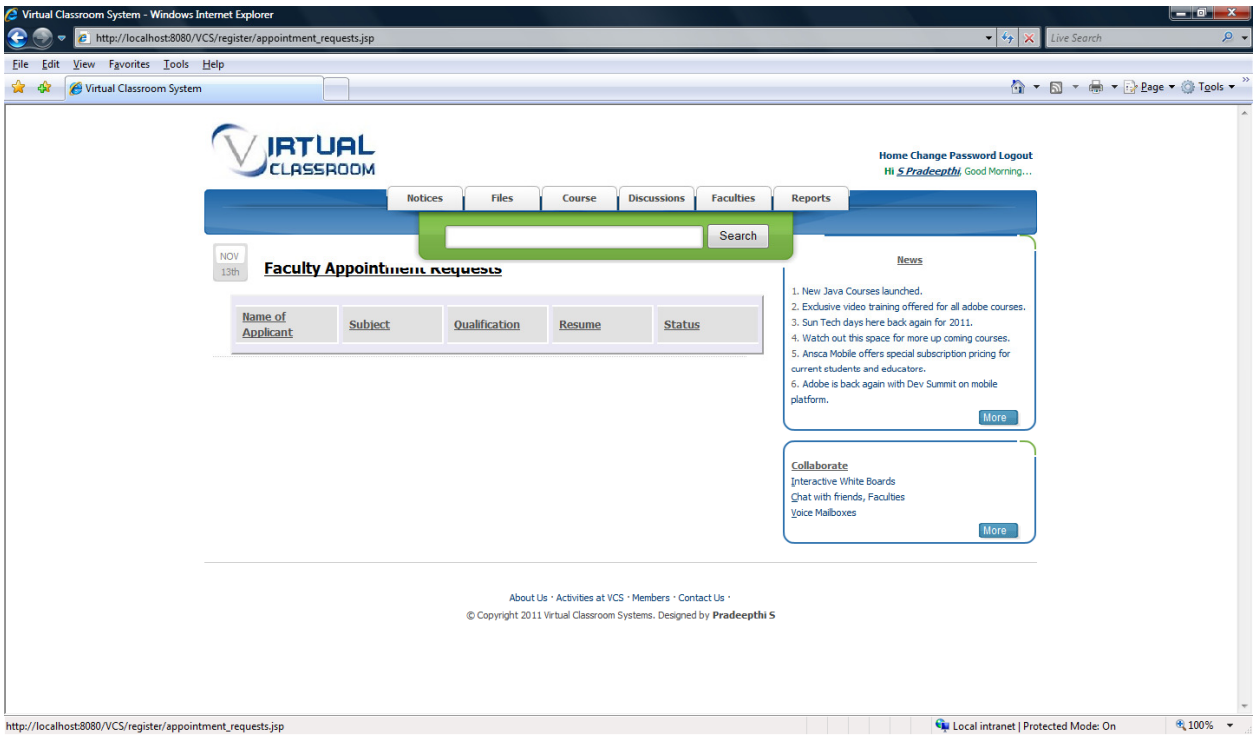


Figure 5.81: Faculty appointments

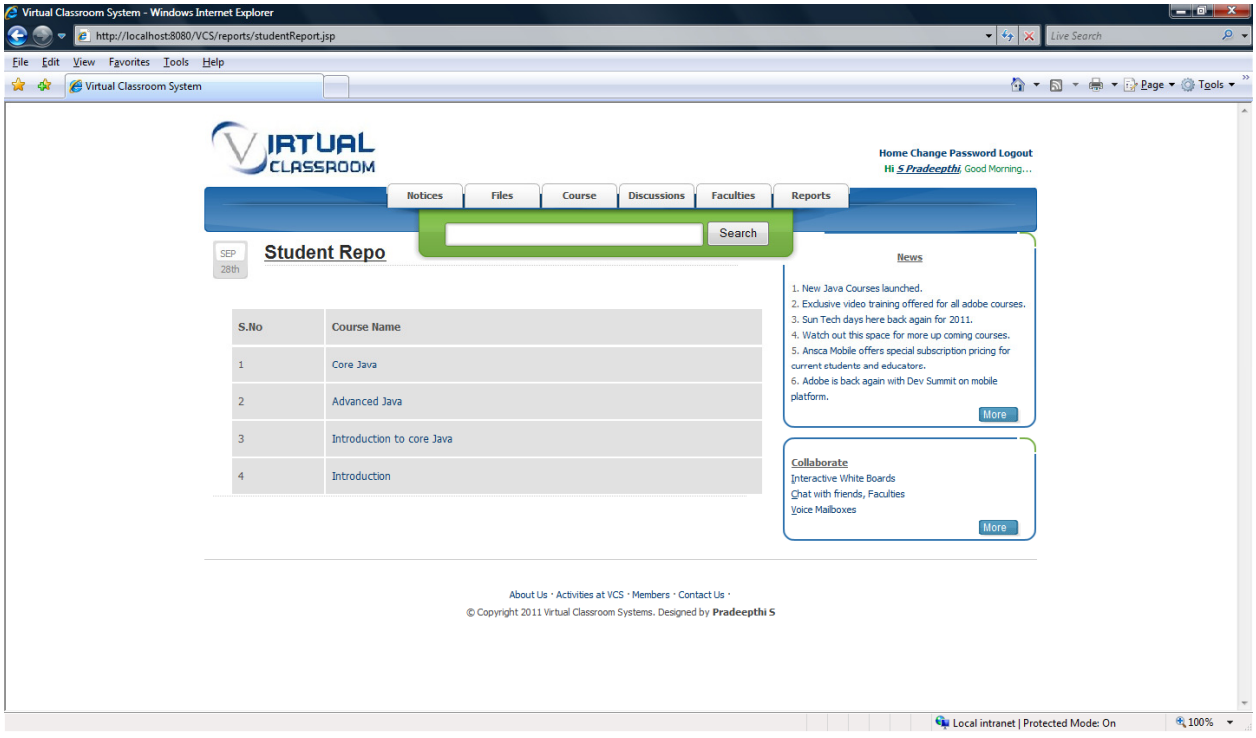


Figure 5.82: Student report subject wise

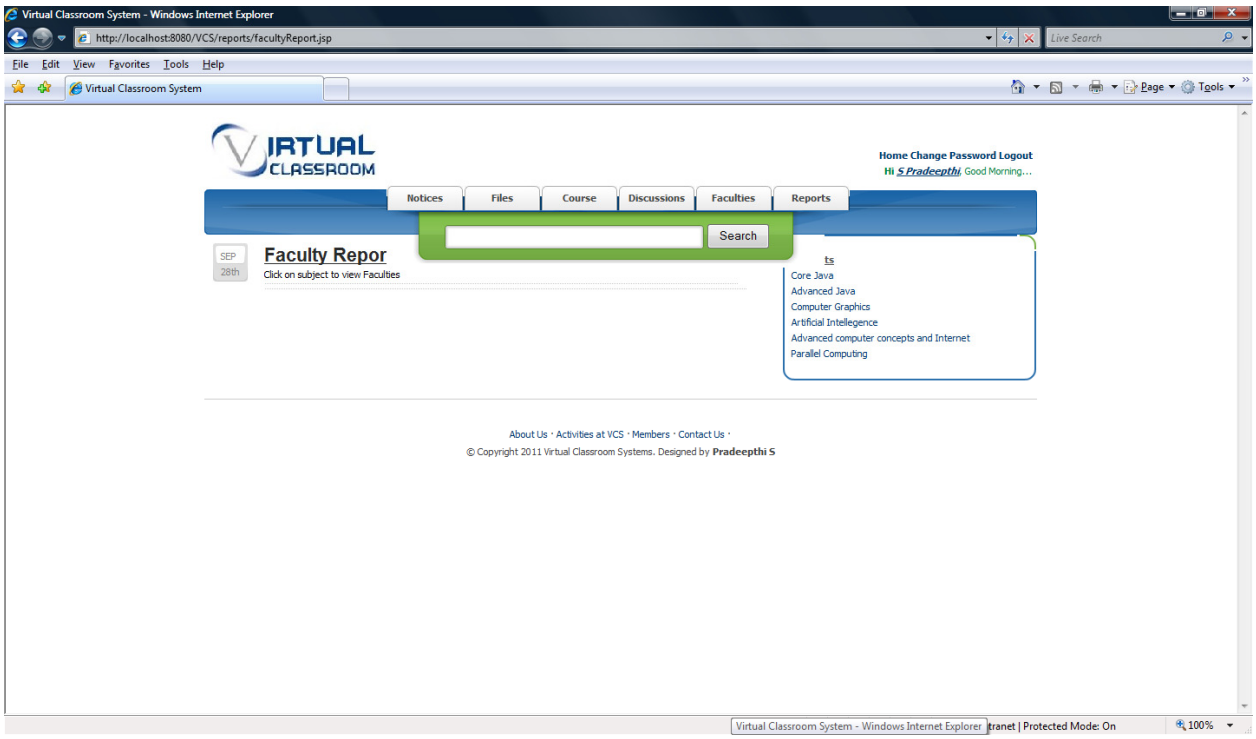


Figure 5.83: Faculty report

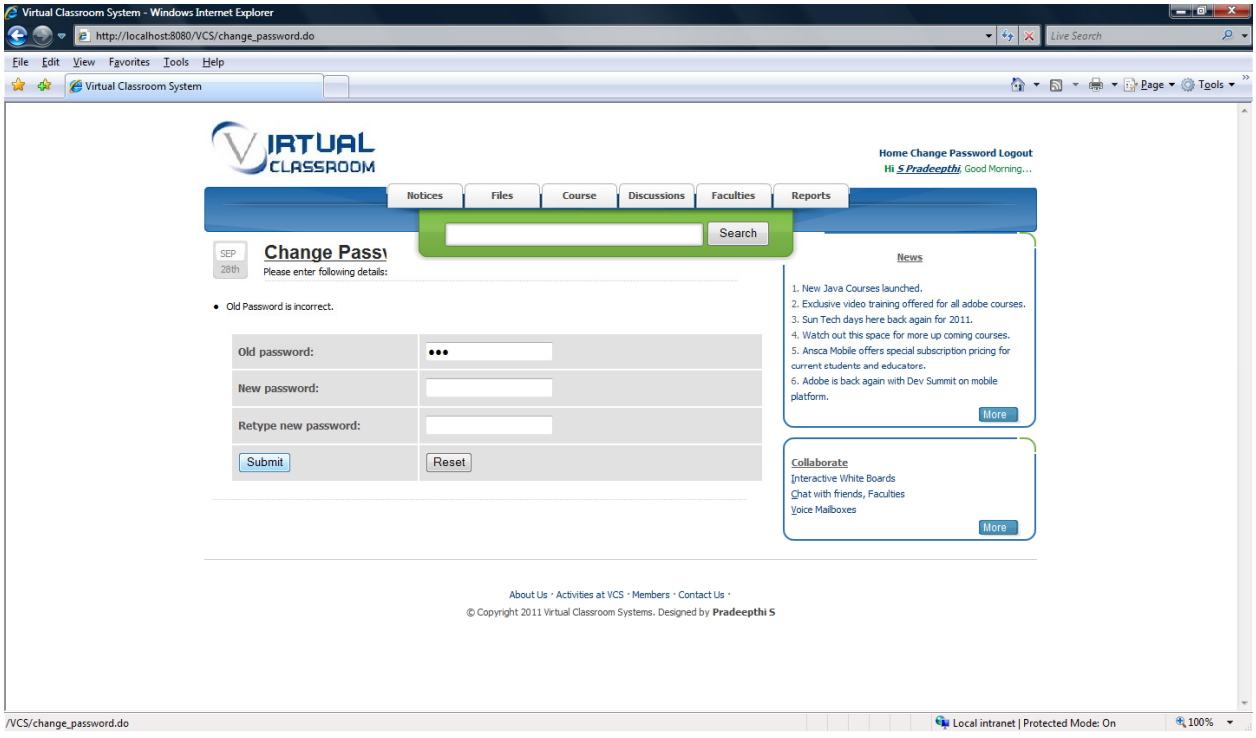


Figure 5.84: Changing password

Student

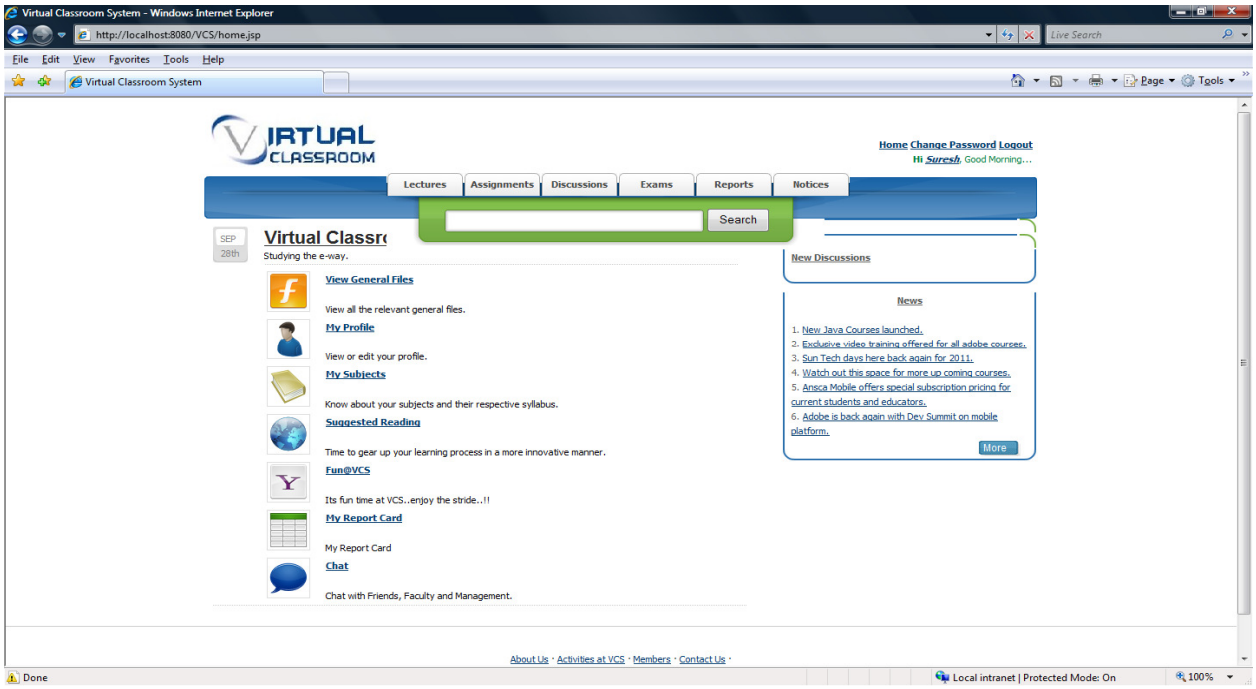


Figure 5.85: Student home page

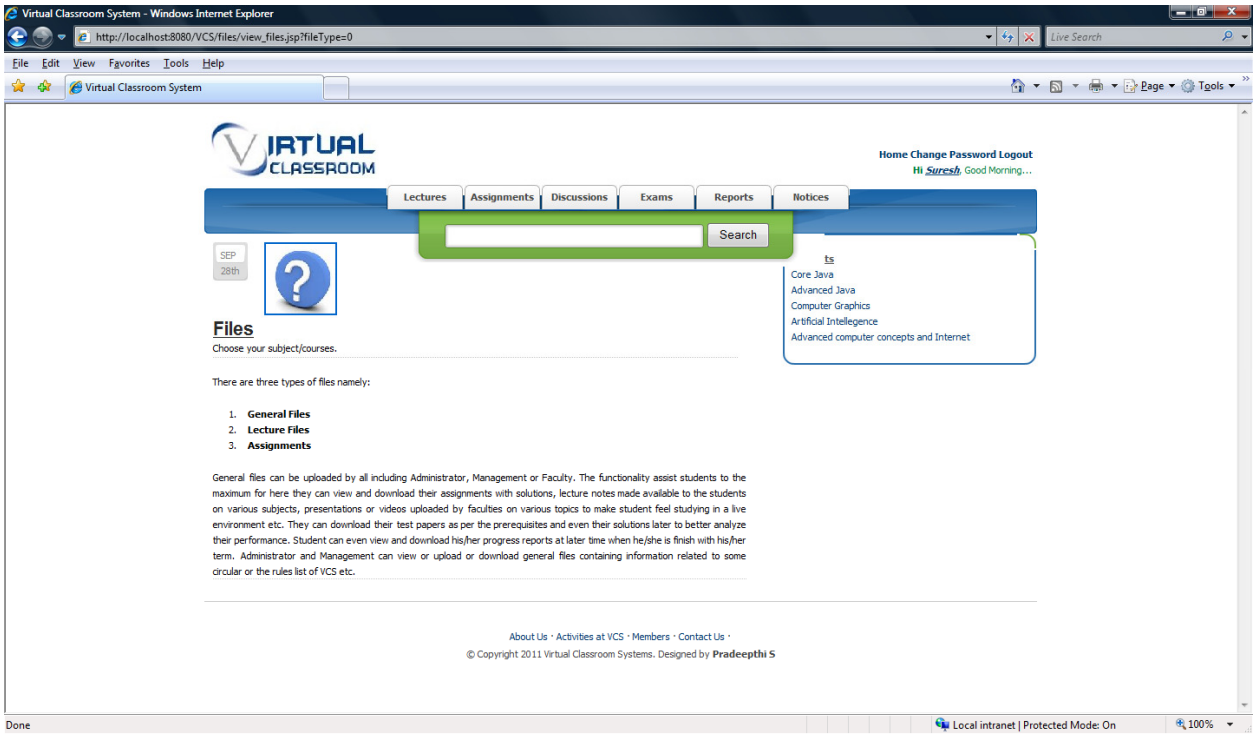


Figure 5.86: Student files (Subject wise)

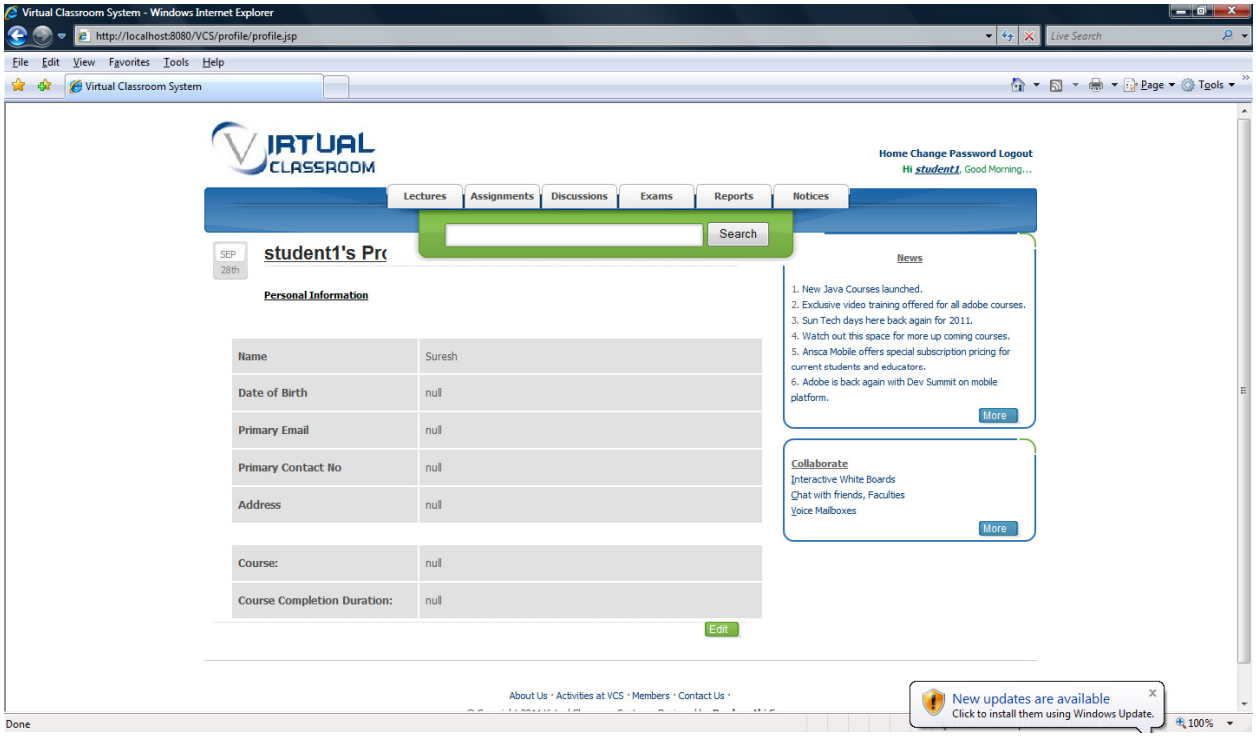


Figure 5.87: Profile of logged in user

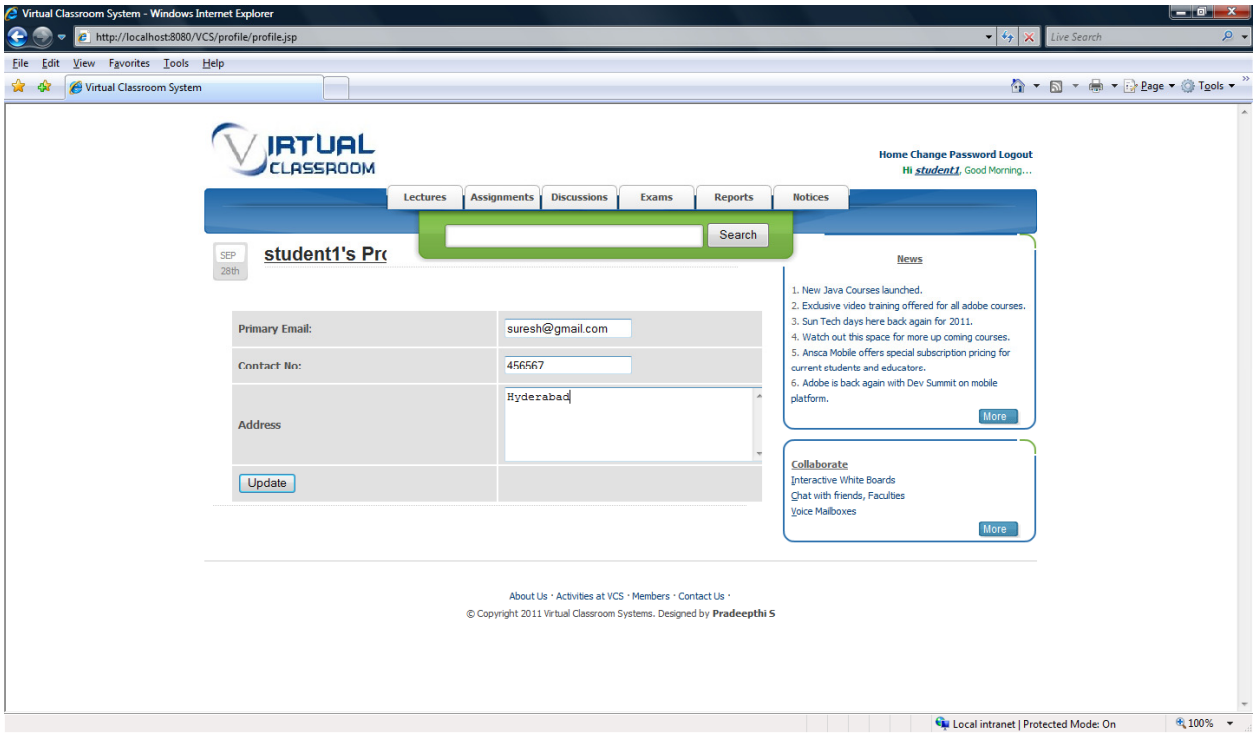


Figure 5.88: Edit profile

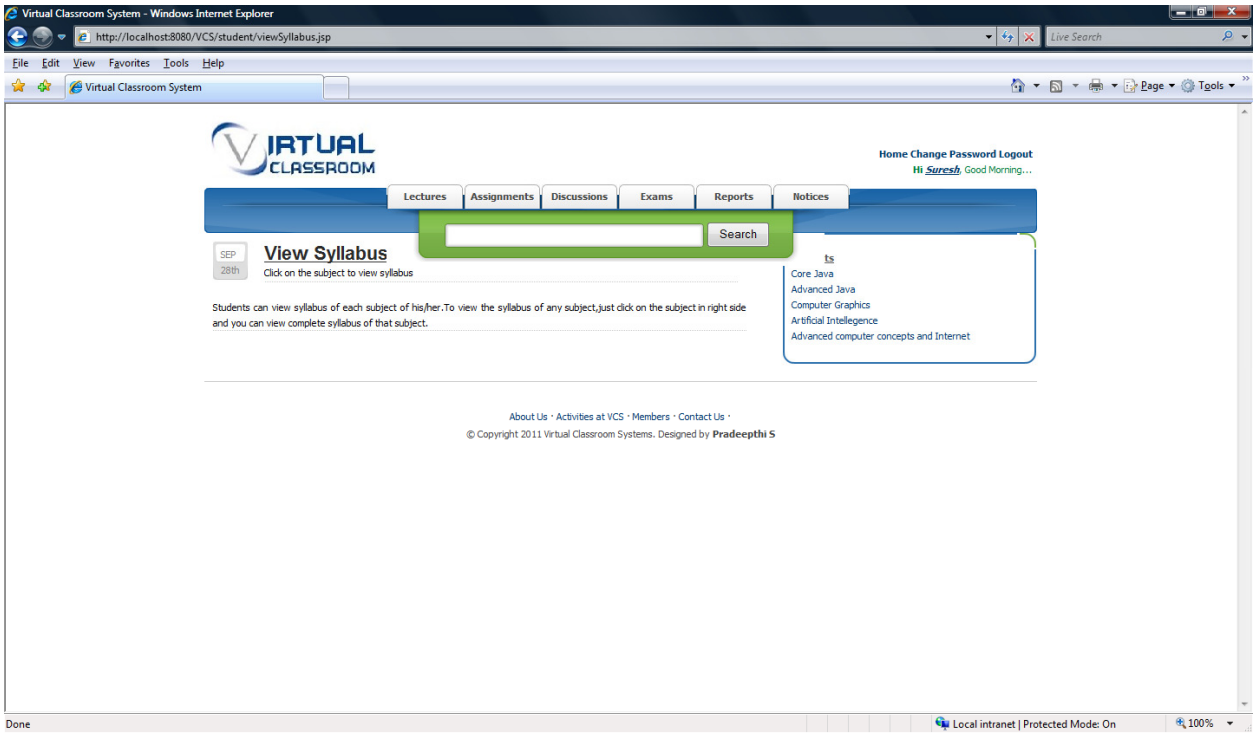


Figure 5.89: Viewing syllabus

Figure 5.90: Suggestive reading material for student

Figure 5.91: A suggesting book for logged in user

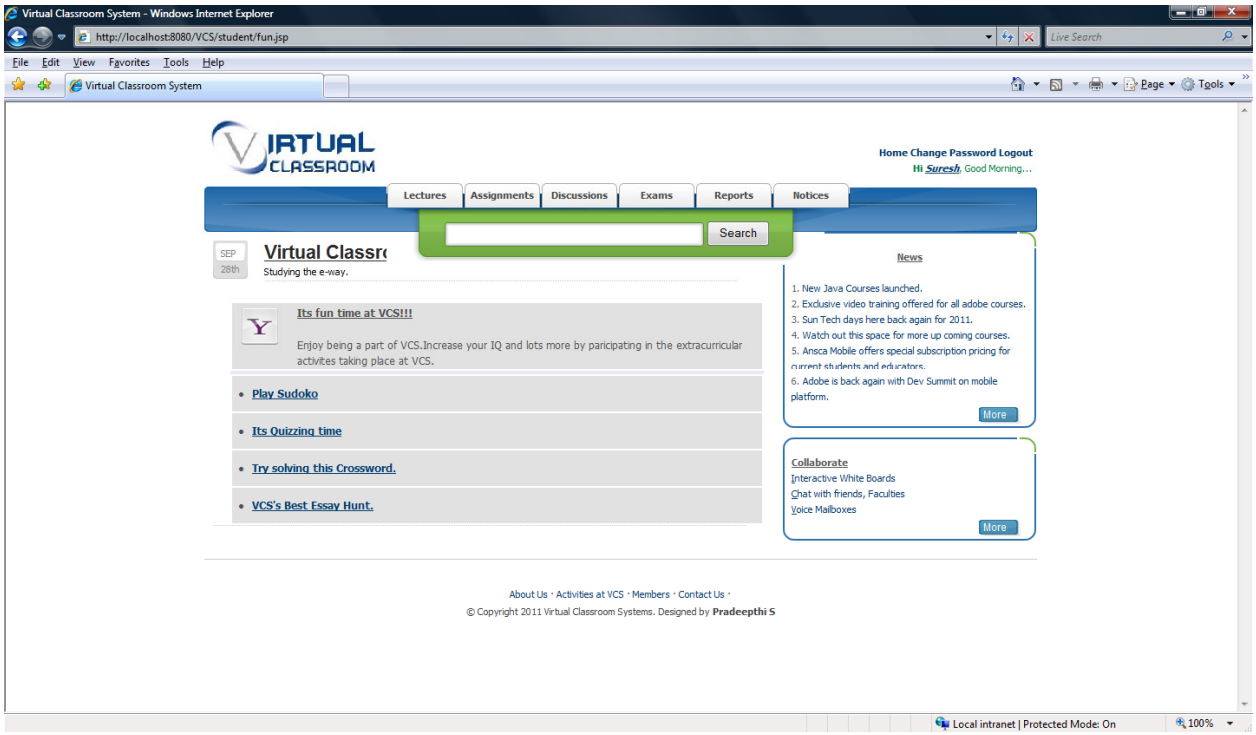


Figure 5.92: Fun @ VCS corner for student

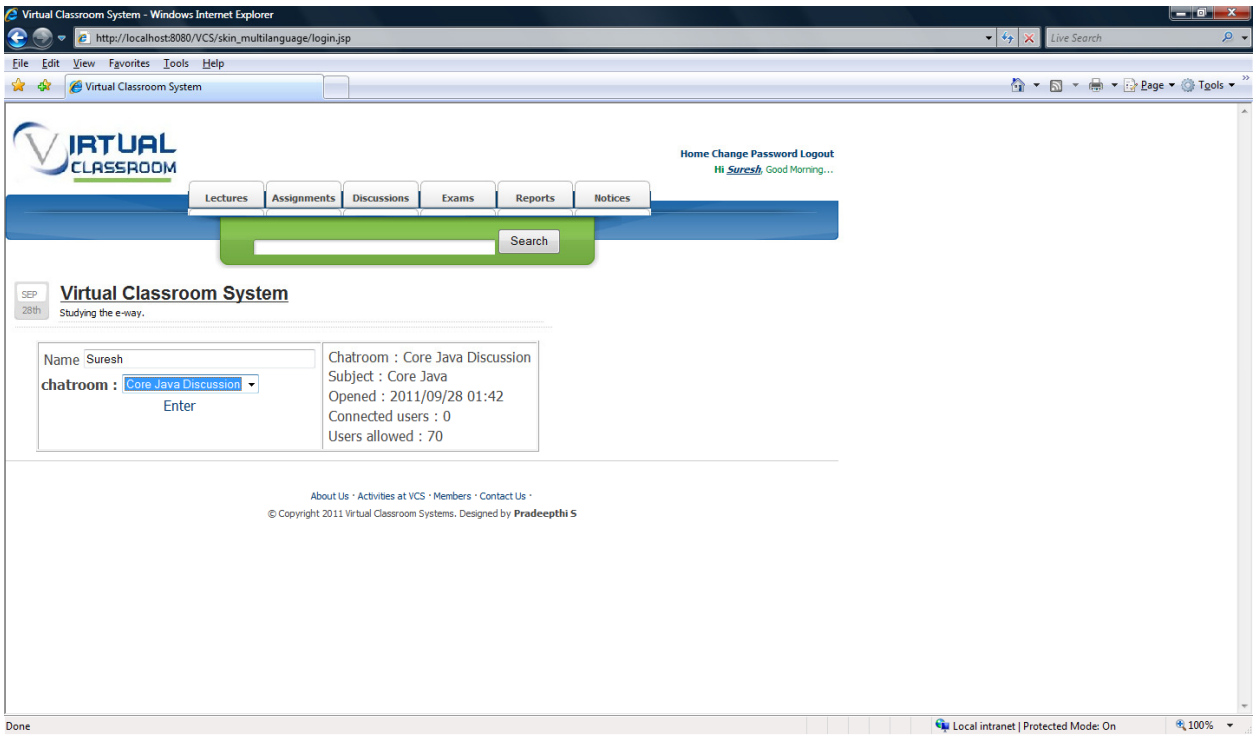


Figure 5.93: logging into discussion board

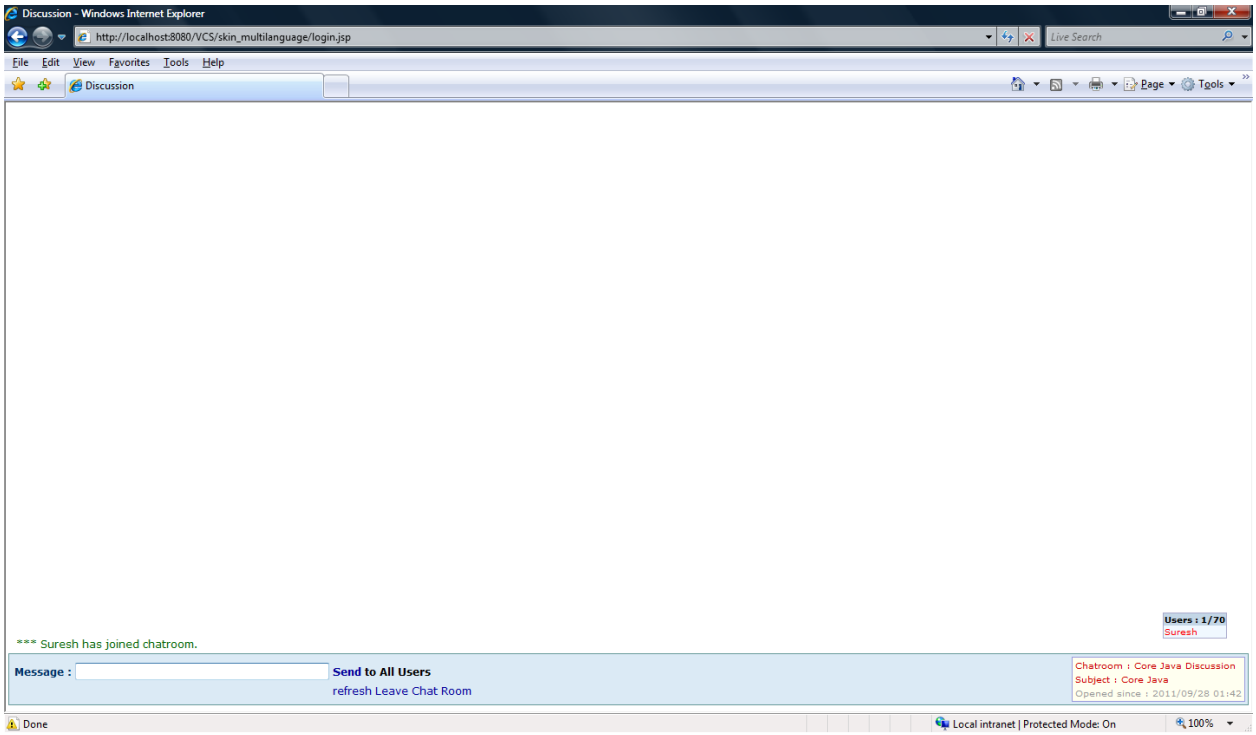


Figure 5.94: Discussion room

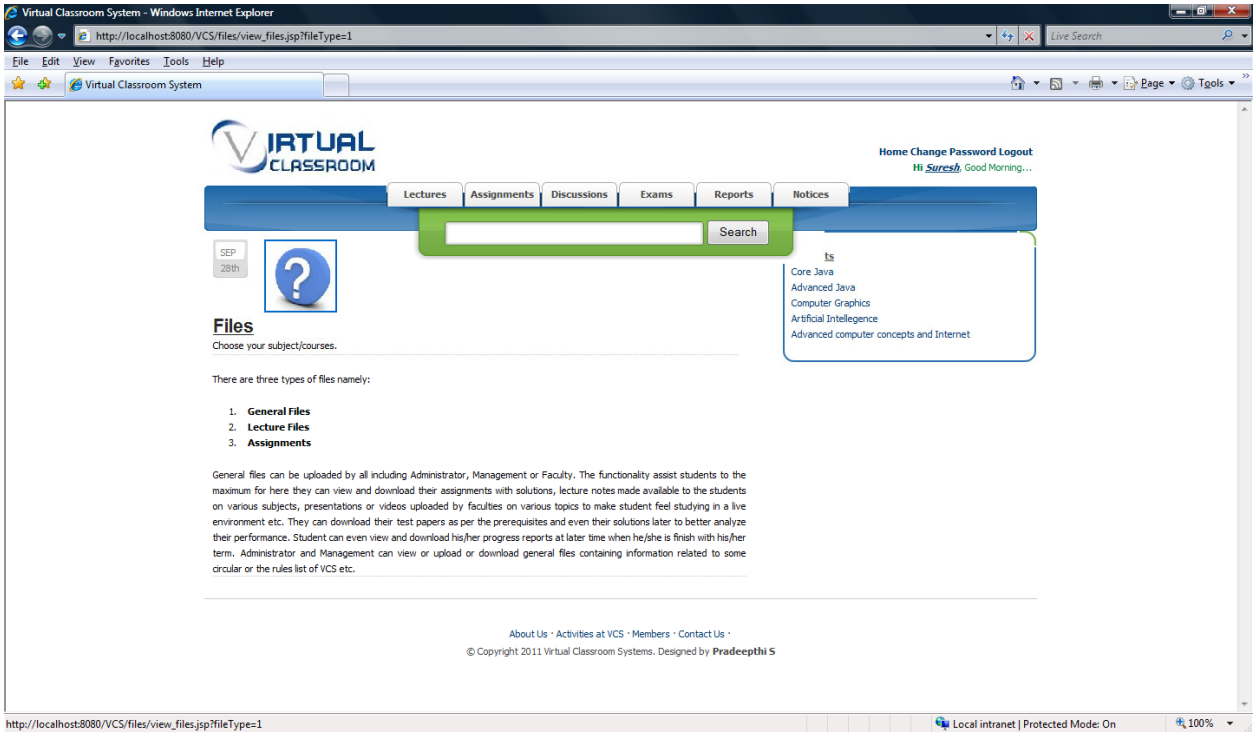


Figure 5.95: listing files

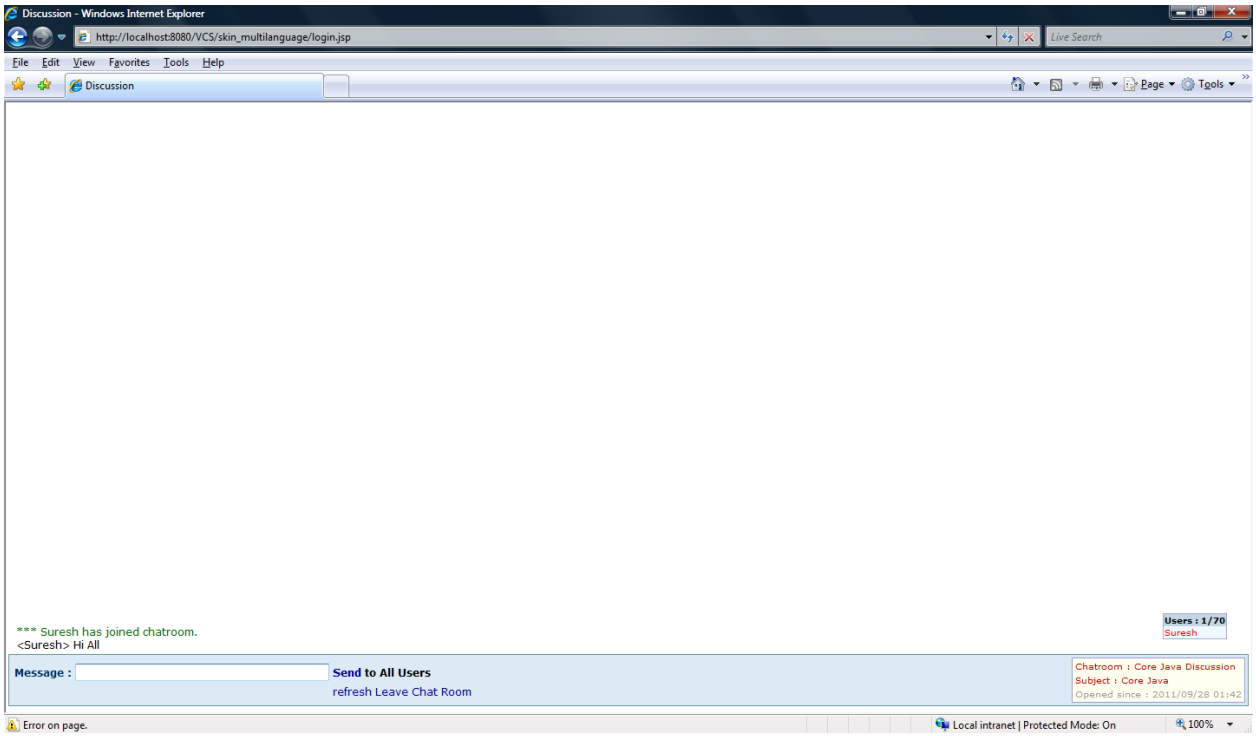


Figure 5.96: discussion messages

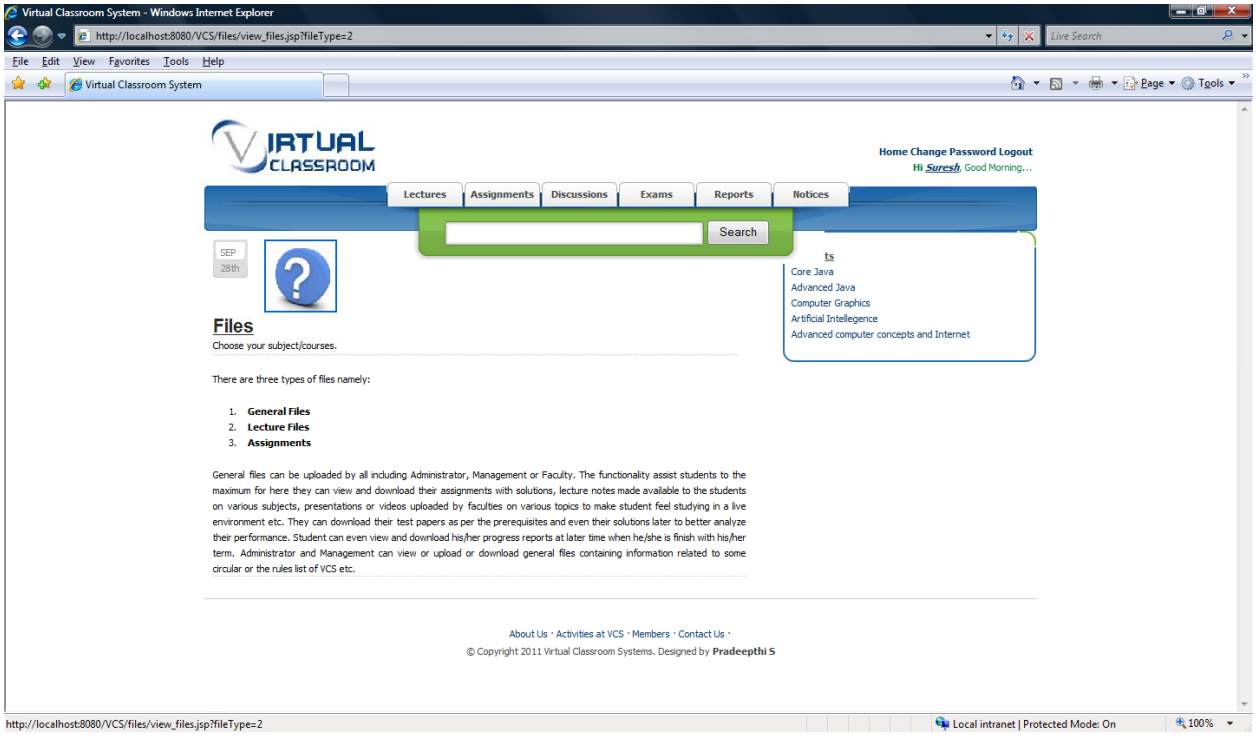


Figure 5.97: Subject wise files

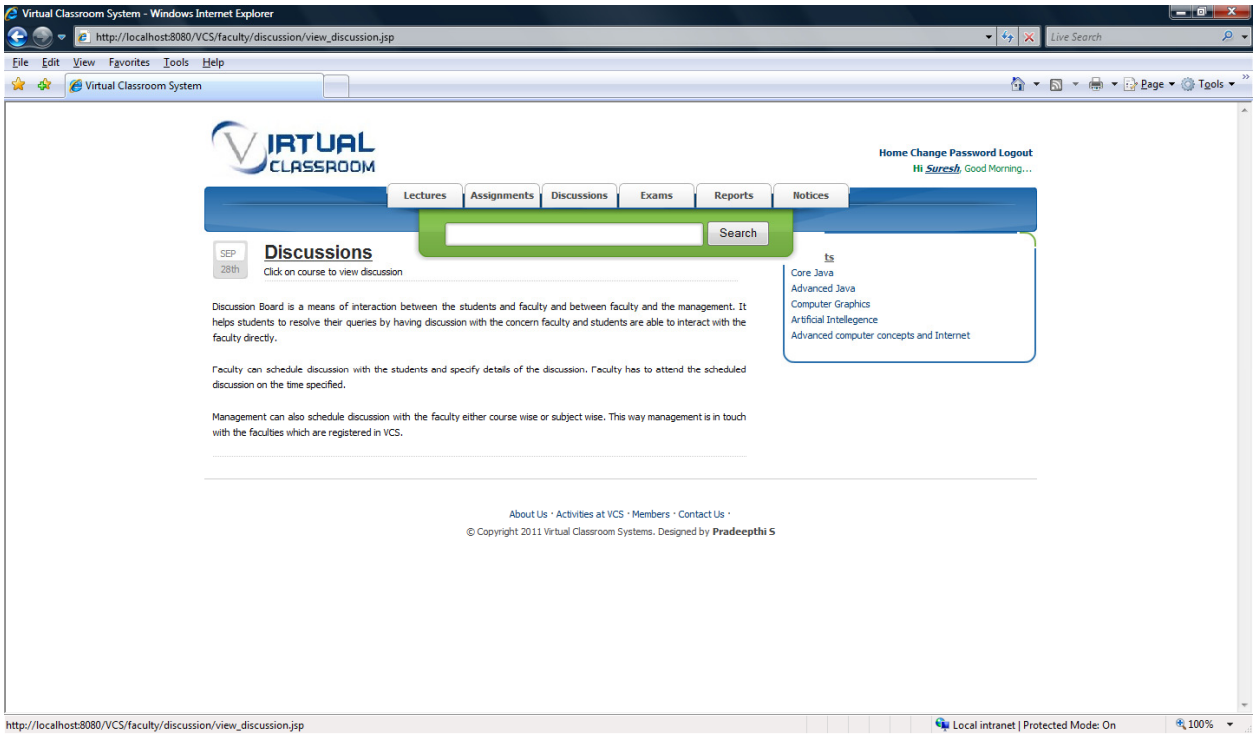


Figure 5.98: Discussion home page

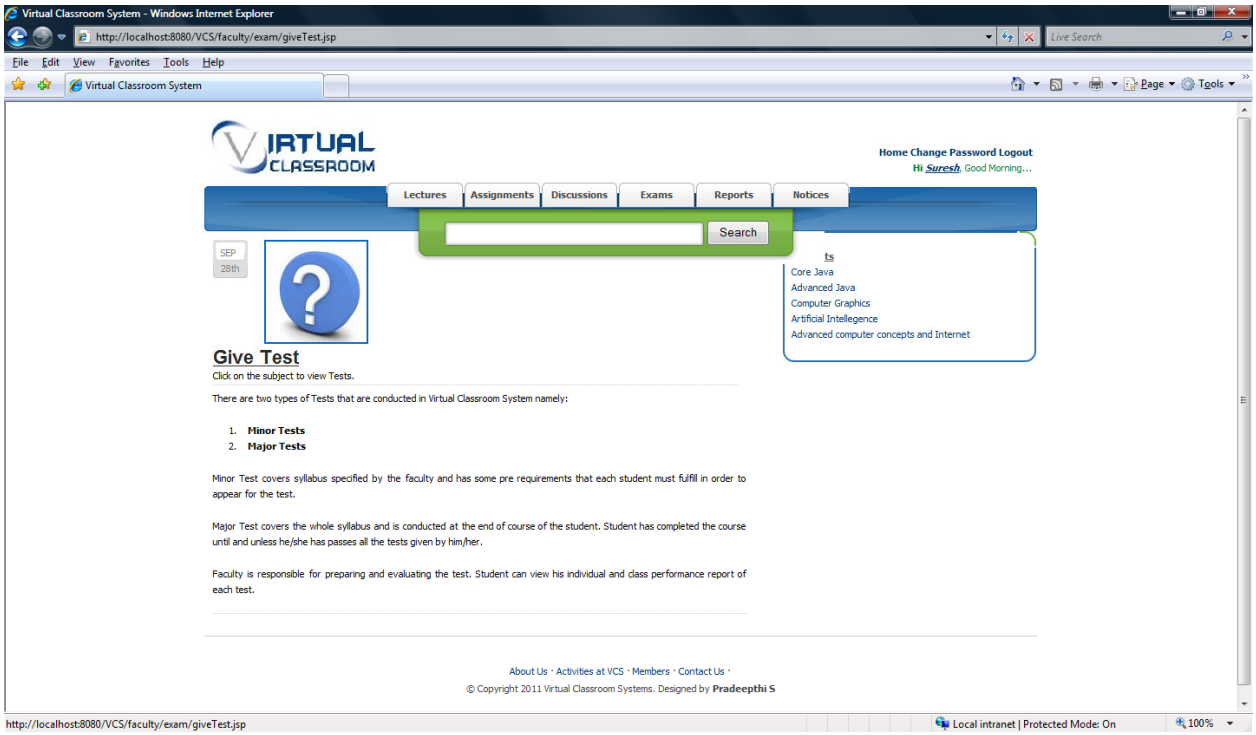


Figure 5.99: Examination corner

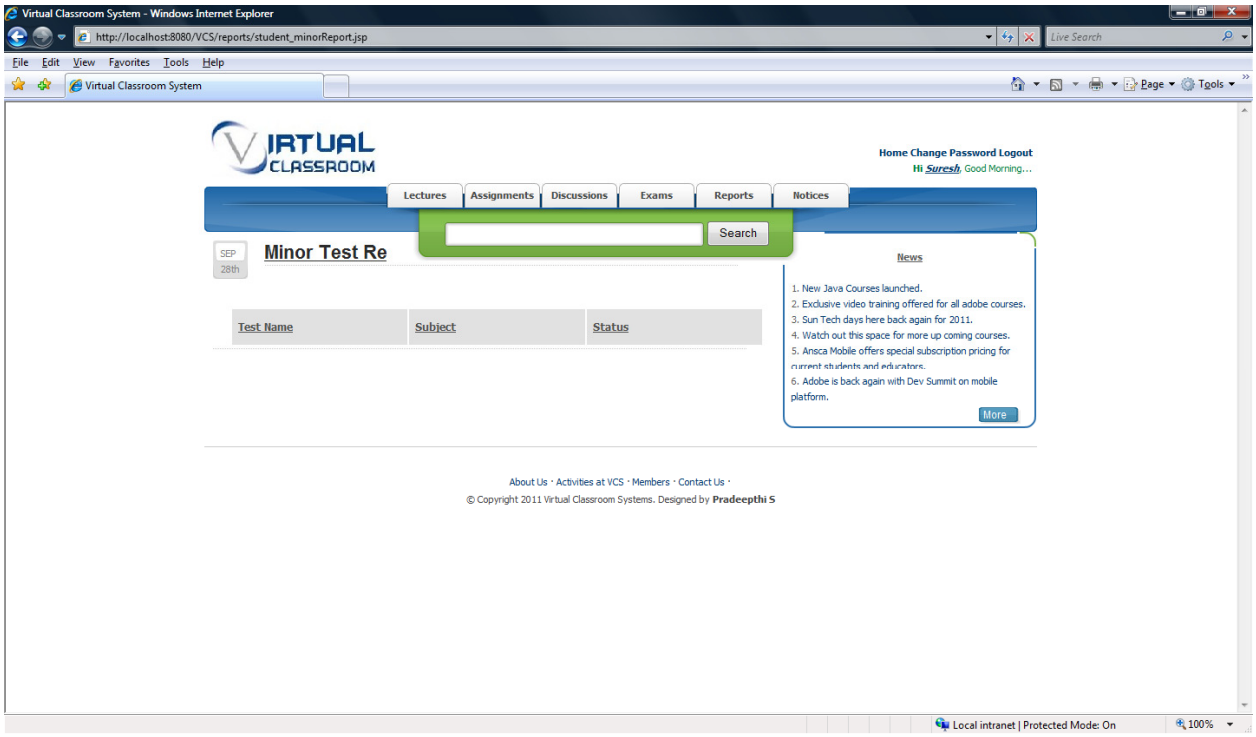


Figure 5.100: viewing list of tests available

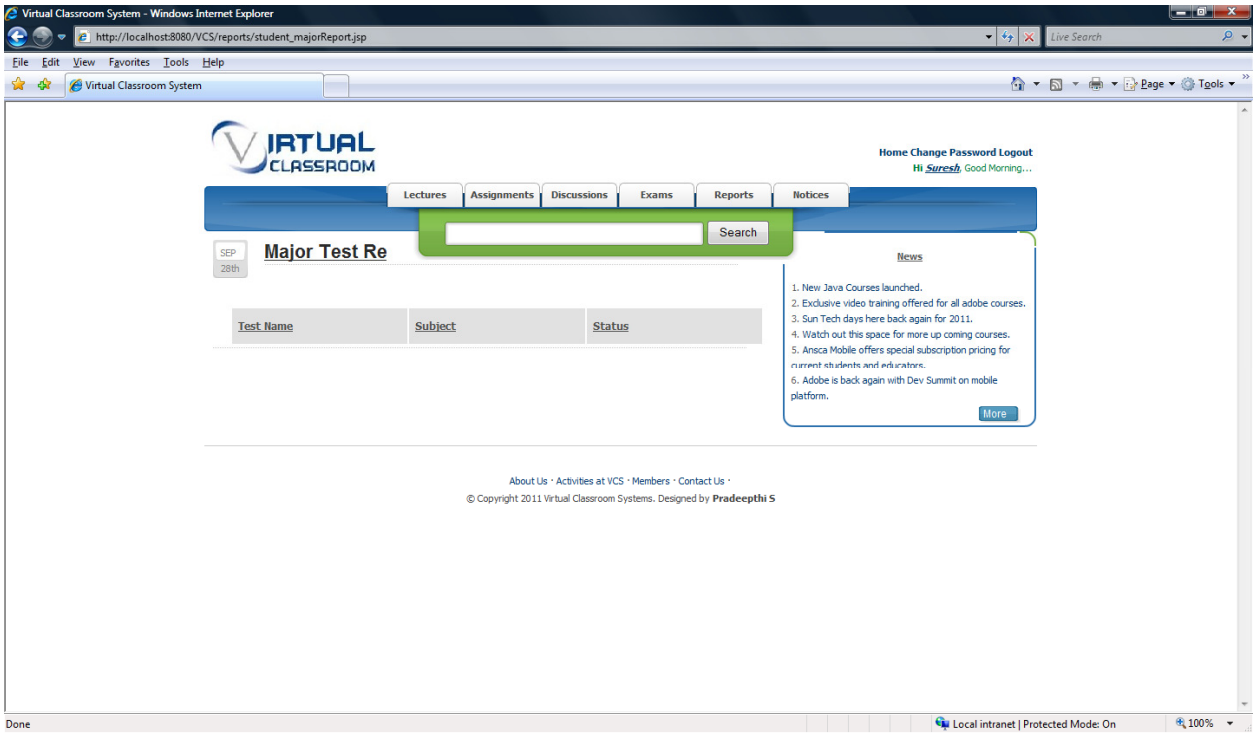


Figure 5.101: listing available tests(Major)

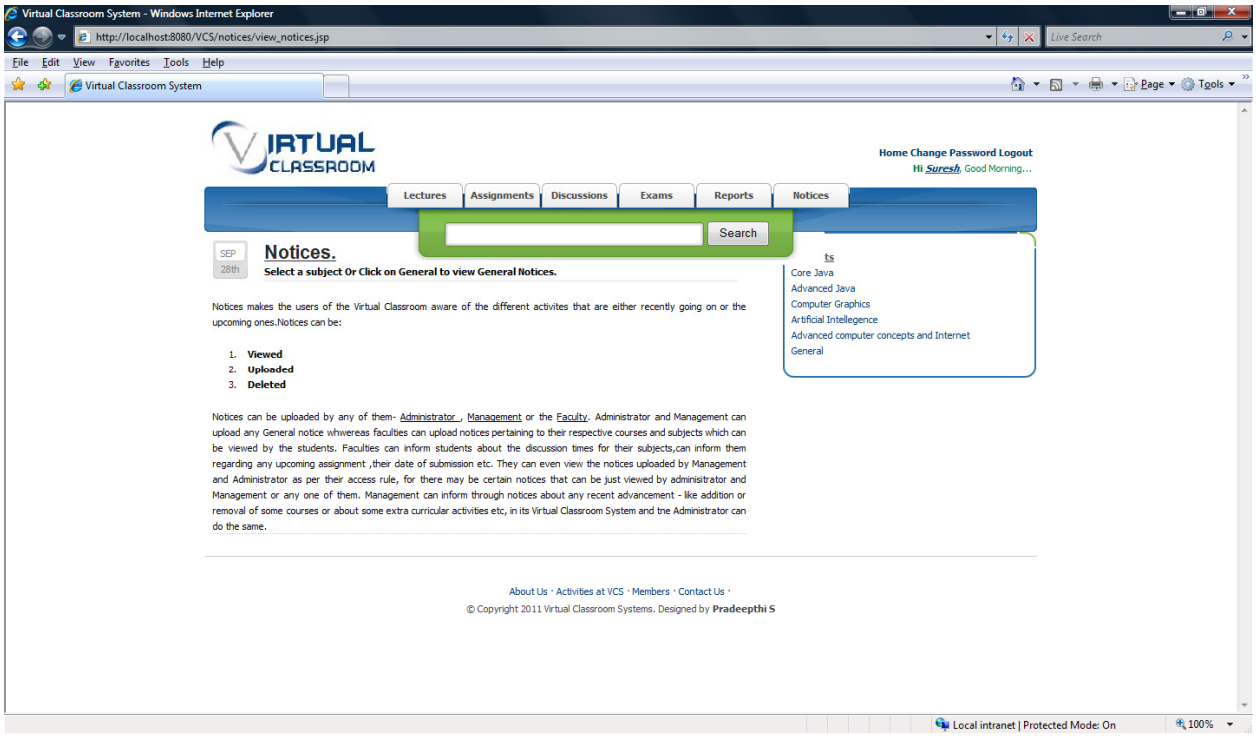


Figure 5.102: Viewing notices

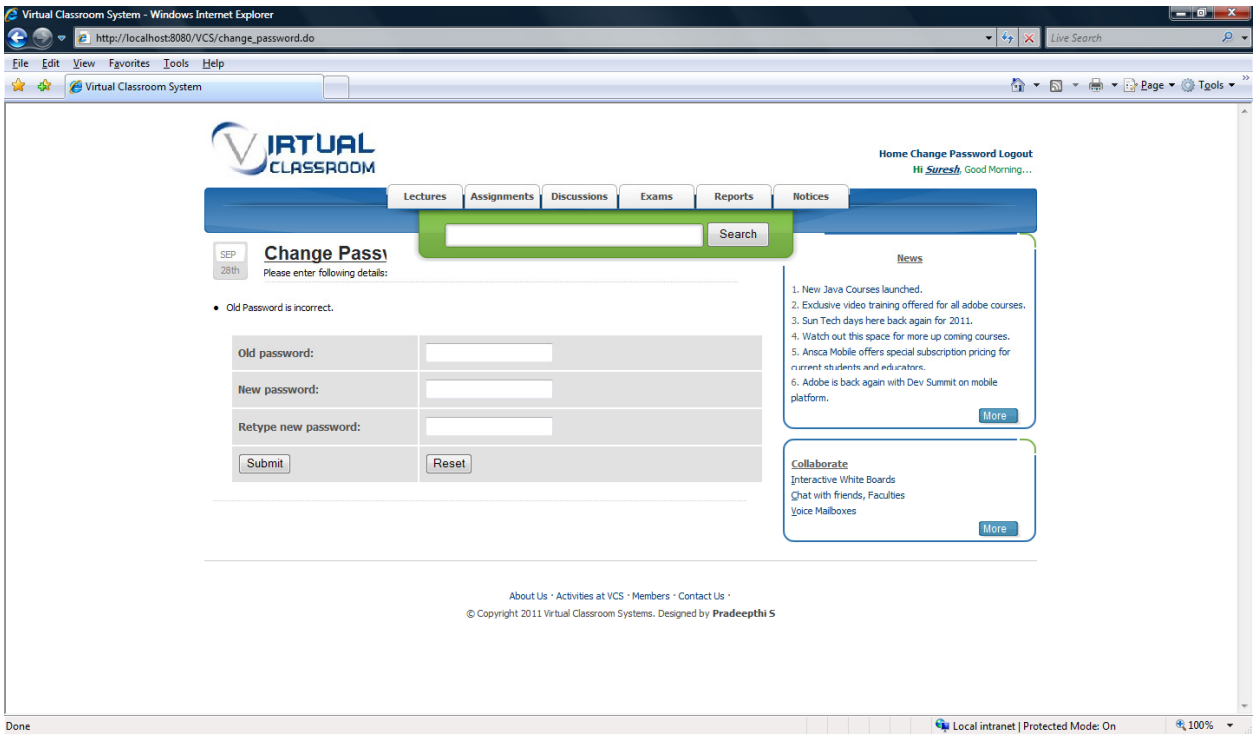


Figure 5.103: Changing password

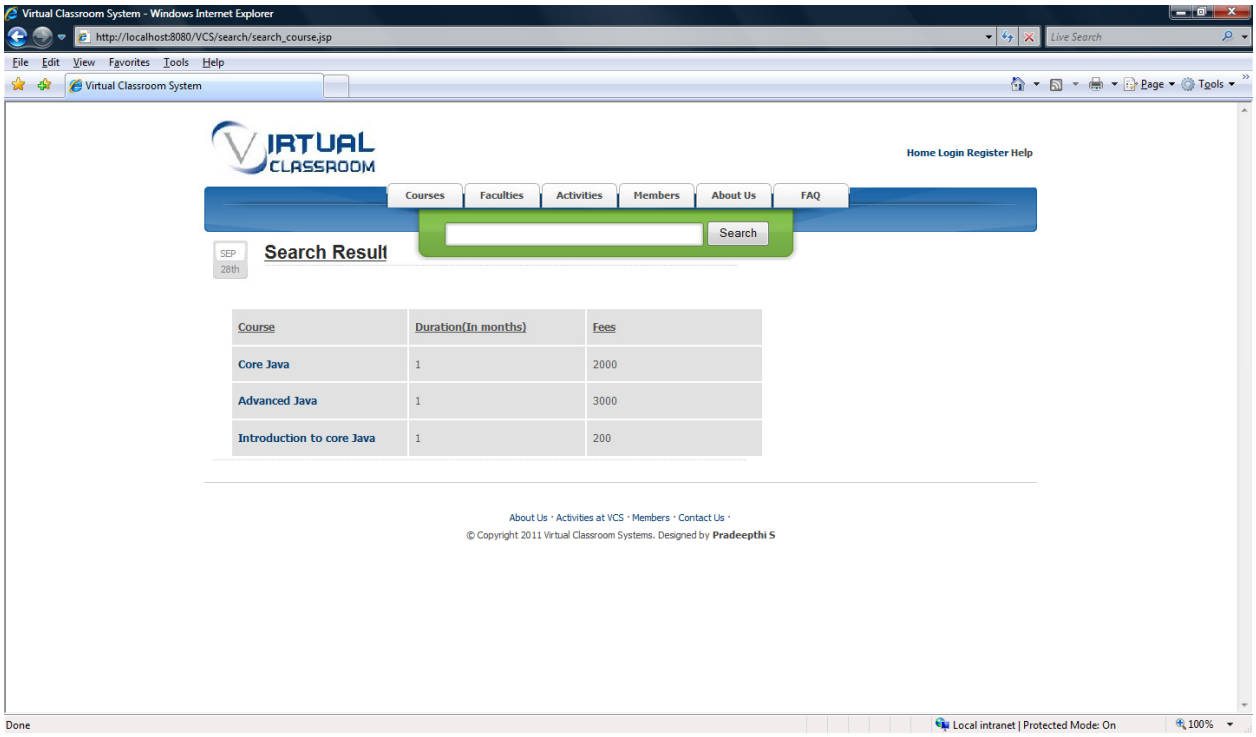


Figure 5.104: Searching subjects

Testing

Testing

Testing Methodology:

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to someone who matters within the project community.

Introduction:

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some

companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.
A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures. Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes. It is commonly believed that the earlier a defect is found the cheaper it is to fix it.

	Time Detected				
Time Introduced	Requirements	Architecture	Construction	System Test	Post-Release
Requirements	1	3	5-10	10	10-100
Architecture	-	1	10	15	25-100
Construction	-	-	1	10	10-25

In counterpoint, some emerging software disciplines such as extreme programming and the agile software development movement, adhere to a "test-driven software development" model. In this process unit tests are written first, by the programmers (often with pair programming in the extreme programming methodology). Of course these tests fail initially; as they are expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed.

Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).
The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

Static testing VS Dynamic testing:

There are many approaches to software testing. Reviews, walkthroughs or inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. The former can be, and unfortunately in practice often is, omitted, whereas the latter takes place when programs begin to be used for the first time - which is normally considered the beginning of the testing stage. This may actually begin before the program is 100% complete in order to test particular sections of code (modules or discrete functions). For example, Spreadsheet programs are, by their very nature, tested to a large extent "on the fly" during the build process as the result of some calculation or text manipulation is shown interactively immediately after each formula is entered.

Verification VsValidation:
Software testing is used in association with verification and validation:

- **Verification:** Have we built the software right? (i.e., does it match the specification?) It is process based.
- **Validation:** Have we built the right software? (i.e., is this what the customer wants?) It is product based.
-

Testing Methodology:

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

- **Black box testing:** Black box testing treats the software as a black box without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.
- **White box testing:** White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these).

Testing Strategy:

A software testing strategy is a well-planned series of steps that result in the successful construction of the software. It should be able to test the errors in software specification, design & coding phases of software development. Software

testing strategy always starts with coding & moves in upward direction. Thus a testing strategy can also divide into four phases:

- **Unit Testing:** Used for coding.
- **Integration Testing:** Used for design phase.
- **System Testing:** For system engineering.
- **Acceptance Testing:** For user acceptance.

Test Development:

1. Test case Development (check list)
2. Test Procedure preparation. (Description of the Test cases).
3. Implementation of test cases. Observing the result.

Result Analysis:

1. Expected value: is nothing but expected behavior of application.
2. Actual value: is nothing but actual behavior of application

Bug Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

Types of Testing:

> Smoke Testing: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

> Sanity Testing: is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

> Regression Testing: is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

>Re-Testing: is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

>Static Testing: is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

>Dynamic Testing: is the testing which is performed on an application when it is being executed.ex: Functional testing.

>Alpha Testing: it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

> **Beta-Testing:** it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

> **Monkey Testing:** is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

>**Compatibility testing:** it is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.

>**Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

>**Adhoc Testing:** Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, without that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmetic issues.

TCD (Test Case Document):

Test Case Document Contains

Test Scope (or) Test objective

- **Test Scenario**
- **Test Procedure**
- **Test case**

This is the sample test case document for the Academic details of student project:

Test scope:

- Test coverage is provided for the screen " Academic status entry" form of a student module of university management system application
- Areas of the application to be tested

Test Scenario:

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively

Test Cases:

- Template for Test Case

T.C.No	Description	Exp	Act	Result

Guidelines for Test Cases:

1. GUI Test Cases:

- Total no of features that need to be check
- Look & Feel
- Look for Default values if at all any (date & Time, if at all any require)
- Look for spell check

Example for GUI Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Check for all the features in the screen	The screen must contain All the features		
2	Check for the alignment of the objects as per the validations	The alignment should be in proper way		

2. Positive Test Cases:

- The positive flow of the functionality must be considered
- Valid inputs must be used for testing
- Must have the positive perception to verify whether the requirements are justified.

Example for Positive Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Check for the date Time Auto Display	The date and time of the system must be displayed	The date and time of the system must be displayed	Success
2	Enter the valid lecture id into the lecture id field	It should accept	It should accept	Success

3. Negative Test Cases:

- Must have negative perception.
- Invalid inputs must be used for test.

Example for Negative Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Try to modify The information in date and time	Modification should not be allow	Modification should not be allow	failure
2	Enter invalid data in to the faculty form, click on save	It should not accept invalid data, save should not allow	It should not accept invalid data, save should not allow	failure

Maintenance

Maintainance

Software maintenance is becoming an important activity of a large number of software organizations. When the hardware platform is changed, and a software product performs some low-level functions, maintenance is necessary. Also, whenever the support environment of a software product changes, the software product requires rework to cope up with the newer interface. For instance, a software product may need to be maintained when the operating system changes. Thus, every software product continues to evolve after its development through maintenance efforts. Therefore it can be stated that software maintenance is needed to correct errors, enhance features, port the software to new platforms, etc.

There are basically three types of software maintenance. These are:

- **Corrective:** Corrective maintenance of a software product is necessary to rectify the bugs observed while the system is in use.

- **Adaptive:** A software product might need maintenance when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware or software.

- **Perfective:** A software product needs maintenance to support the new features that users want it to support, to change different functionalities of the system according to customer demands, or to enhance the performance of the system.

- **Preventive:** Preventive maintenance is conducted to keep equipment working and/or extend the life of the equipment.

System Security

System Security

Password Encryption :

The JavaTM Cryptography Extension (JCE) provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. Support for encryption includes symmetric, asymmetric, block, and stream ciphers. The software also supports secure streams and sealed objects.

JCE is based on the same design principles found elsewhere in the JCA: implementation independence and, whenever possible, algorithm independence. It uses the same "provider" architecture. Providers signed by a trusted entity can be plugged into the JCE framework, and new algorithms can be added seamlessly.

User Profiles and Access Rights :

There are 4 levels of users in VCS : Admin, Student, Faculty and Management

- Complete control to set up classes as internal only (behind the college or university firewall) or accessible to outside parties through the Internet
- Access authentication options that require all participants to authenticate themselves before gaining access to classes or materials
- Integration with college or university directories to ensure only legitimate students can access classes
- Encryption options, to protect sensitive information such as students' or instructors' personal data
- In-session classroom controls that allow class leaders to change participant permissions, require passwords, lock a class, and eject unwanted attendees

Reports

Reports

Faculty Performance Report:

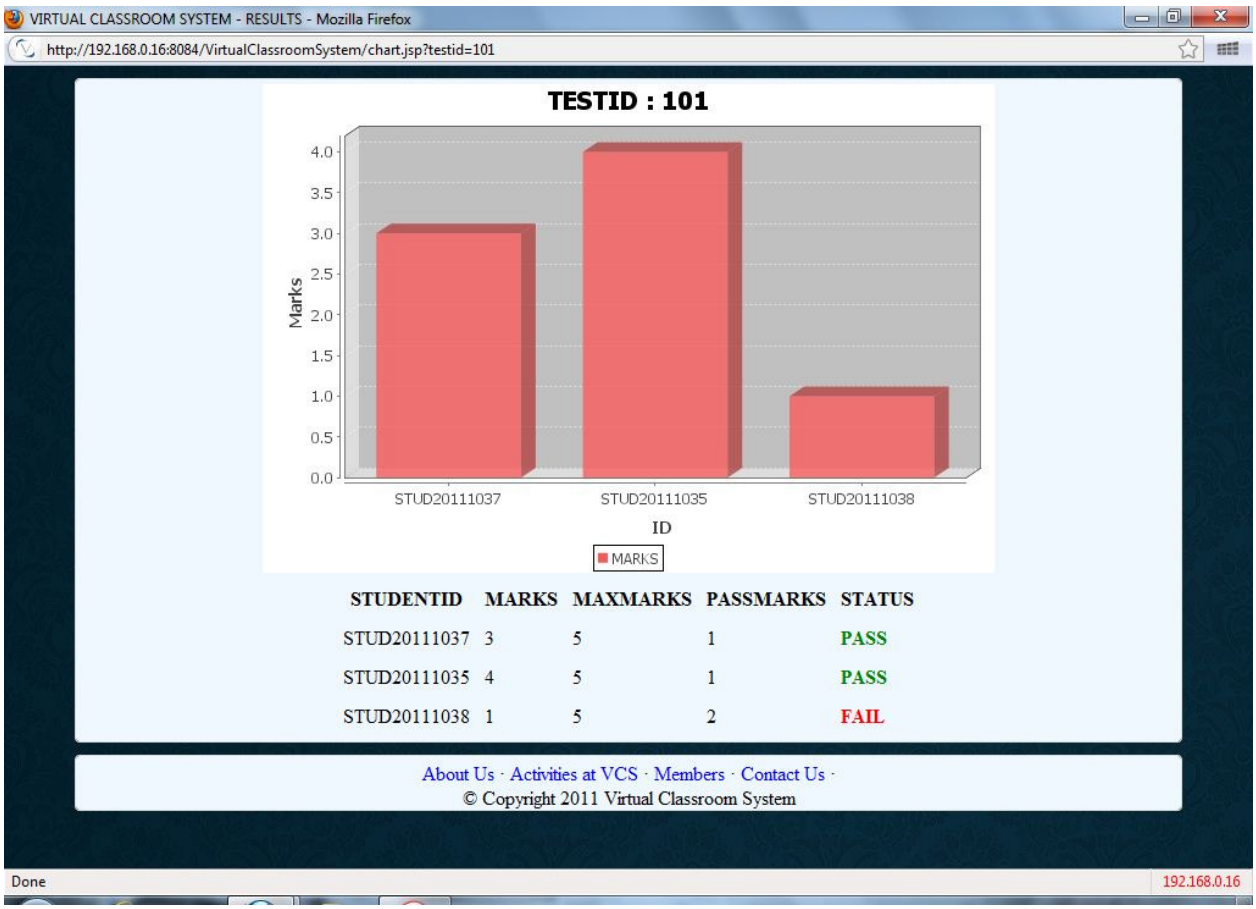


Figure 1.49: Faculty Performance Report

Student Performance Report:

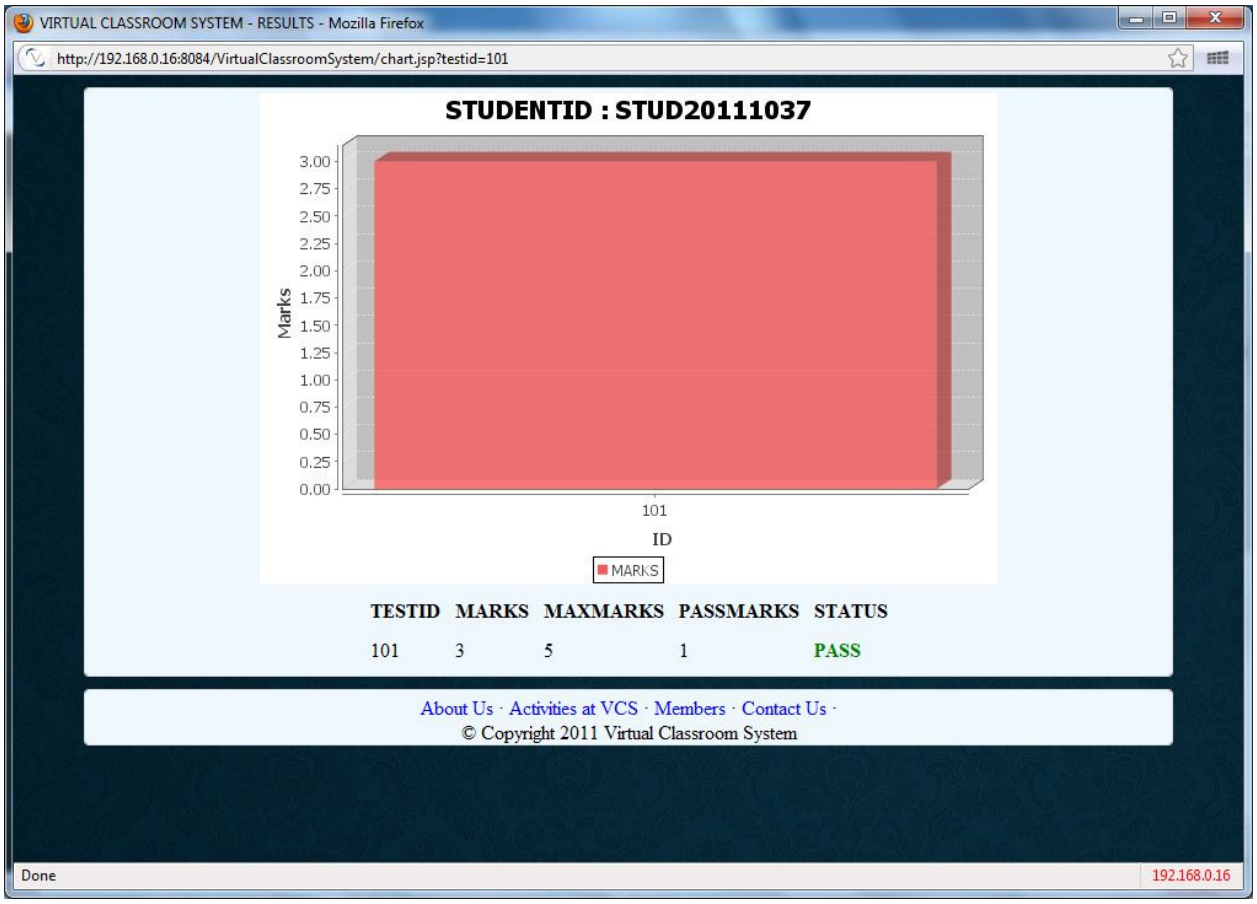


Figure 1.50: Student Performance Report

Limitations and Future Enhancements

Limitations of the system:

- Only the permanent employees can access all the features in the system.
- Advanced techniques are not used to check the authorization.
- Cannot Export data as a comma separated text file to be used with other spreadsheet application eg? EXCEL,
- Interactive whiteboards are not implemented.
- Voice mail box and voice mails not implemented.

Future Enhancements:

Since this project is now only working now at local level, if it is hosted on the domain, then the project can be provided on the state and national level also. For web hosting, VCS requires a domain name to be registered so that it becomes available to all.

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
- Because it is based on object-oriented design, any further changes can be easily adaptable.
- Based on the future security issues, security can be improved using emerging technologies.
- Attendance module can be added
- sub admin module can be added
- An in-built web browser can be added

Conclusion

Conclusion:

The **"virtual classroom system"** was successfully designed and is tested for accuracy and quality.

During this project we have accomplished all the objectives and this project meets the needs of the organization. The developed will be used in searching, retrieving and generating information for the concerned requests.

Goals

- ✓ Reduced entry work.
- ✓ Easy retrieval of information.
- ✓ User friendly menus.
- ✓ Reduced errors due to human intervention
- ✓ User friendly screens to enter the data
- ✓ Portable and flexible for further enhancement
- ✓ Web enabled.
- ✓ Fast finding of information requested

Bibliography

Bibliography:

Core Java™ 2 Volume I – Fundamentals 7th Edition Pearson Education – Sun Microsystems	Cay S. Hortsman Gary Cornell
Core Java™ 2 Volume II – Advanced Pearson Education – Sun Microsystems	Cay S. Hortsman Gary Cornell
Head First Servlets & JSP O'Reilly – SPD	Eric Freeman Elisabeth Freeman
The Book of JavaScript 2nd Edition SPD	thau
Effective Java – Programming Language Guide Pearson Education – Sun Microsystems	Joshua Bloch
Java Database Best Practices O'Reilly – SPD	George Reese
JBoss – A Developers Notebook O'Reilly – SPD	Norman Richards Sam Griffith
Phil Hanna, *JSP 2.0*	Tata McGraw-Hill , 2003
Activity Diagrams	*http://www.ibm.com/developerworks/rational/library/2802.html*