**INTRODUCTION**

**INTRODUCTION**

An e-resource is an electron information resource that you can access through university or college websites, on or off campus. You can get information you want, when you need, 24 hours a day, seven days a week.

MCA E-Resource is web application that is developed for education purpose only for MCA department. Student can download notes and question paper from anywhere and anytime, but the student must be registered on web site. Faculty can upload as well as download notes and question paper, but the main task of faculty is uploading notes and question paper on web site.

MCA E-Resource provide a platform to student that can access all available resources on web site. The web site provide a user friendly GUI for both student as well as faculty.



**OBJECTIVE**

**OBJECTIVE**

The main objective of this project is uploading and downloading notes and question paper.

* Provide a facility that student can download notes and question paper.
* Provide a facility that faculty can upload as well as download notes and question paper.

**SYSTEM**

**SPECIFICATION**

**SOFTWARE REQUIREMENT**

**FRONT END**

* PHP
* HTML
* JAVASCRIPT
* CSS

**BACK END**

* MYSQL

**OPERATING SYSTEM**

* Windows XP Onwards

**HARDWARE REQUIREMENT**

**PROCESSOR**

* Intel Pentium 4 or Higher

**RAM**

* Minimum 512 MB

**HARD DISK**

* Minimum 6 GB

**OTHERS**

* Monitor
* Mouse
* Keyboard

**FEASIBILITY**

**STUDY**

**FEASIBILITY STUDY**

The feasibility study is an evaluation and analysis of the potential of a proposed project which is based on extensive investigation and research to support the process of decision making.

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

A well-designed feasibility study should provide a historical background of the business or project, a description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based

The acronym [TELOS](http://en.wikipedia.org/wiki/TELOS_%28project_management%29) refers to the five areas of feasibility - Technical, Economic, Legal, Operational, and Scheduling.

**TECHNICAL FEASIBILITY**

The technical feasibility considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements. If the internal technical capability cannot support the project, the project still can be considered technically feasible if an outside entity could be utilized to support the technical requirements and the costs associated with this outsourcing would still be within the scope of the proposed budget and economic feasibility.

**ECONOMIC FEASIBILITY**

The economic feasibility will review the expected costs to see if they are in-line with the projected budget or if the project has an acceptable return on investment. At this point, the projected costs will only be a rough estimate. The exact costs are not required to determine economic feasibility. It is only required to determine if it is feasible that the project costs will fall within the target budget or return on investment.

**LEGAL FEASIBILITY**

Determines whether the proposed system conflicts with legal requirements, e.g. a data processing system must comply with the local Data Protection Acts.

**OPERATIONAL FEASIBILITY**

Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the feasibilities to gauge. In order to determine this feasibility, it is important to understand the management commitment to the proposed project. If the request was initiated by management, it is likely that there is management support and the system will be accepted and used. However, it is also important that the employee base will be accepting of the change.

**SCHEDULING FEASIBILITY**

Similar to economic feasibility, a rough estimate of the project schedule is required to determine if it would be feasible to complete the systems project within a required timeframe. The required timeframe would need to be set by the organization.

**OTHER FEASIBILITY FACTORS**

**MARKET AND REAL ESTATE FEASIBILITY**

Market feasibility studies typically involve testing geographic locations for a real estate development project, and usually involve parcels of real estate land. Developers often conduct market studies to determine the best location within a jurisdiction, and to test alternative land uses for given parcels. Jurisdictions often require developers to complete feasibility studies before they will approve a permit application for retail, commercial, industrial, manufacturing, housing, office or mixed-use project. Market Feasibility takes into account the importance of the business in the selected area.

**RESOURCE FEASIBILITY**

This involves questions such as how much time is available to build the new system, when it can be built, whether it interferes with normal business operations, type and amount of resources required, dependencies, and developmental procedures with company revenue prospectus.

**CULTURAL FEASIBILITY**

In this stage, the project's alternatives are evaluated for their impact on the local and general [culture](http://en.wikipedia.org/wiki/Culture). For example, environmental factors need to be considered and these factors are to be well known. Further an enterprise's own culture can clash with the results of the project.

**FINANCIAL FEASIBILITY**

In case of a new project, financial viability can be judged on the following parameters:

* Total estimated cost of the project
* Financing of the project in terms of its capital structure, debt equity ratio and promoter's share of total cost
* Existing investment by the promoter in any other business
* Projected cash flow and profitability

The financial viability of a project should provide the following information:

* Full details of the assets to be financed and how liquid those assets are.
* Rate of conversion to cash-liquidity (i.e. how easily can the various assets be converted to cash?).
* Project's funding potential and repayment terms.
* Sensitivity in the repayments capability to the following factors:
  + Time delays.
  + Mild slowing of sales.
  + Acute reduction/slowing of sales.
  + Small increase in cost.
  + Large increase in cost.
  + Adverse economic conditions.

**MARKET RESEARCH STUDY AND ANALYSIS**

This is one of the most important sections of the feasibility study as it examines the marketability of the product or services and convinces readers that there is a potential market for the product or services. If a significant market for the product or services cannot be established, then there is no project.

Typically, market studies will assess the potential sales of the product, absorption and market capture rates and the project's timing.

The feasibility study outputs the feasibility study report, a report detailing the evaluation criteria, the study findings, and the recommendations.

**FEATURES**

**FEATURES**

* Students can download notes and question paper.
* Faculty upload as well as download notes and question paper.
* Student can download any subject notes and any year of question paper.
* Websites runs 24\*7.

**TECHNOLOGY**

**USED**

**TECHNOLOGY USED**

This project is a web application that is developed in PHP, JAVASCRIPT HTML and CSS along with My SQL as back end.

* PHP
* HTML
* JAVASCRIPT
* CSS
* XAMPP 3.2.1
* MYSQL

**PHP**

**PHP** is a server-side scripting language designed for web development but also

used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million servers. Originally created by Rasmus Lerdorf in 1995, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for Personal Home Page; it now stands for PHP: Hypertext Preprocessor, a recursive backronym.

PHP code is interpreted by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical application.

PHP is free software released under the PHP License, which is incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP.PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

**HTML**

Hyper Text Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser.

HTML is written in the form of HTML elements consisting of  *tags* enclosed in

angle brackets (like <html>), within the web page content. HTML tags most

commonly come in pairs like <h1> and </h1>, although some tags represent  *empty*

*elements* and so are unpaired, for example <img>. The first tag in a pair is the  *start*

*tag*, and the second tag is the  *end tag* (they are also called *opening tags* and *closing*

*tags*). In between these tags web designers can add text, further tags, comments

and other types of text-based content.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML.

**JAVA SCRIPT**

JavaScript is the scripting language of the Web! JavaScript is an interpreted computer programming language. JavaScript can interact with HTML source code, enabling Web authors to spice up their sites with dynamic content.

The most common use of JavaScript is to write functions that are embedded in or included from HTML pages and that interact with the Document Object Model (DOM) of the page. Some simple examples of this usage are:

* Loading new page content or submitting data to the server via AJAX without reloading the page (for example, a social network might allow the user to post status updates without leaving the page)
* Animation of page elements, fading them in and out, resizing them, moving them, etc.
* Interactive content, for example games, and playing audio and video.
* JavaScript was designed to add interactivity to HTML pages.
* JavaScript is a scripting language.
* A scripting language is a lightweight programming language.
* A JavaScript is usually embedded directly into HTML pages.
* JavaScript is an interpreted language (means that scripts execute without

preliminary compilation).

* Everyone can use JavaScript without purchasing a license.

**JAVASCRIPT FEATURES**

* JavaScript gives HTML designers a programming tool- HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small “snippets” of code into their HTML pages.
* JavaScript can put dynamic text into an HTML page – A JavaScript statement like this: document. write(“<h1>” + name + ”</h1>”) can write a variable text into an HTML page.
* JavaScript can react to events – A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on a HTML element.
* JavaScript can read and write HTML elements – A JavaScript can read and change the content of an HTML element.
* JavaScript can be used to validate data – A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing.
* JavaScript can be used to detect the visitor’s browser – A JavaScript can be used to detect the visitor’s browser, and – depending on the browser – load another page specifically designed for that browser.
* JavaScript can be used to create cookies – A JavaScript can be used to store and retrieve information on the visitor’s computer.
* JavaScript includes an eval function that can execute statements provided as strings at run-time.
* JavaScript uses prototypes where many other object oriented languages use classes for inheritance. It is possible to simulate many class-based features with prototypes in JavaScript.

**CSS**

Cascading Style Sheets (CSS) and JavaScript Style Sheets used in concert with

Scripting is what puts the dynamism in Dynamic HTML. It allows you to precisely format, embellish, and position the content in your documents instead of acquiescing to the whims of each browser's rendering choices. You can also create documents that change by themselves, or in response to user interaction, and have inline animations contained within them. Inline is the key word here because that means faster downloading and no plug-ins to load or have compatibility issues with or annoy the user because, if after downloading

seventy-five plug-ins, the one for your content isn't on his/her system. It also means that you can have an image that not only animates but can move around the page instead of being confined to a stationary rectangular space.

You embellish your content by making stylistic choices that affect the color, size, font face, boldness, and other aspects of text markup, and you can create margins and borders for your text. Then you can position your content exactly where you want it to appear in the page instead of the old inline flow method. Transparent and opaque blocks of content are now possible which you can change, move, resize, make appear or disappear on-the-fly, or respond to user input. If so inclined, you could let each user custom design how the website will appear for them.

Time will definitely be saved if you have a particular style that you want to use for your whole site or section of it by creating the Style Sheet just once and saving it in its own file and then accessing it multiple times with one line of code. You just LINK to the Style Sheet when you want it to apply to that specific page.

The text layout paradigm of Style Sheets has been directly ported from the desktop publishing world where programs like QuarkXpress have had Style Sheets for years. If you are familiar with that then you're already halfway home to understanding Cascading Style Sheets in HTML. There are two main differences between Style Sheets as used in Quark and those used in Navigator. The first is that in Quark you are working in a WYSIWYG environment so you just make your choices from radio buttons, text-boxes, and pop-up menus in a dialog window and then Quark does all the post-script coding for you.

In HTML with Cascading Style Sheets you have to do the coding yourself but you're doing the same thing to your content and you get very similar results. In fact, getting HTML pages to behave more like documents in the publishing world is half the point. The other difference, and this one goes way beyond desktop publishing, is the dynamic and interactive aspects of Cascading Style Sheets.

**XAMPP**

XAMPP SERVER is a windows web development environment. It allows you to create web applications with Apache, PHP, and a MySQL database.

Alongside Php MyAdmin allows you to manage easily your databases.

**MYSQL**

MySQL is the most popular database system used with PHP.

What is MySQL?

* MySQL is a database system used on the web
* MySQL is a database system that runs on a server
* MySQL is ideal for both small and large applications
* MySQL is very fast, reliable, and easy to use
* MySQL supports standard SQL
* MySQL compiles on a number of platforms
* MySQL is free to download and use
* MySQL is developed, distributed, and supported by Oracle Corporation
* MySQL is named after co-founder Monty Widenius's daughter: My

The data in MySQL is stored in tables. A table is a collection of related data, and it consists of columns and rows.

**SDLC**

**SDLC**

SDLC is an acronym for **Software Development Life Cycle**. It is also sometimes referred to as **System Development Life Cycle**. In simple words it the process, methods or a set of methodologies applied to create or alter software projects. Each of these methodologies defines unique way to create a new software module or program.

A software development process, also known as a software development life-cycle (SDLC), is a structure imposed on the development of a software product. Similar terms include software life cycle and software process. It is often considered a subset of systems development life cycle. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Some people consider a life-cycle model a more general term and a software development process a more specific term. For example, there are many specific software development processes that 'fit' the spiral life-cycle model. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

The large and growing body of software development organizations implements process methodologies. Many of them are in the defense industry, which in the U.S. requires a rating based on 'process models' to obtain contracts.

The international standard for describing the method of selecting, implementing and monitoring the life cycle for software is ISO/IEC 12207.

A decades-long goal has been to find repeatable, predictable processes that improve productivity and quality. Some try to systematize or formalize the seemingly unruly task of writing software. Others apply project management techniques to writing software. Without effective project management, software projects can easily be delivered late or over budget. With large numbers of software projects not meeting their expectations in terms of functionality, cost, or delivery schedule, it is effective project management that appears to be lacking.

Organizations may create a Software Engineering Process Group (SEPG), which is the focal point for process improvement. Composed of line practitioners who have varied skills, the group is at the center of the collaborative effort of everyone in the organization who is involved with software engineering process improvement.

**SOFTWARE**

**DEVELOPMENT**

**ACTIVITIES**

* Planning
* Implementing, testing and documenting
* Deployment and maintenance

**PLANNING**

Planning is an objective of each and every activity, where we want to discover things that belong to the project. An important task in creating a software program is extracting the requirements or requirements analysis. Customers typically have an abstract idea of what they want as an end result, but do not know what software should do. Skilled and experienced software engineers recognize incomplete, ambiguous, or even contradictory requirements at this point. Frequently demonstrating live code may help reduce the risk that the requirements are incorrect.

Once the general requirements are gathered from the client, an analysis of the scope of the development should be determined and clearly stated. This is often called a scope document.

Certain functionality may be out of scope of the project as a function of cost or as a result of unclear requirements at the start of development. If the development is done externally, this document can be considered a legal document so that if there are ever disputes, any ambiguity of what was promised to the client can be clarified.

**IMPLEMENTING, TESTING AND DOCUMENTING**

Implementation is the part of the process where software engineers actually program the code for the project.

Software testing is an integral and important phase of the software development process. This part of the process ensures that defects are recognized as soon as possible.

Documenting the internal design of software for the purpose of future maintenance and enhancement is done throughout development. This may also include the writing of an API, be it external or internal. The software engineering process chosen by the developing team will determine how much internal documentation (if any) is necessary. Plan-driven models (e.g., Waterfall) generally produce more documentation than Agile models

**DEPLOYMENT AND MAINTENANCE**

Deployment starts directly after the code is appropriately tested, approved for release, and sold or otherwise distributed into a production environment. This may involve installation, customization (such as by setting parameters to the customer's values), testing, and possibly an extended period of evaluation.

Software training and support is important, as software is only effective if it is used correctly.

Maintaining and enhancing software to cope with newly discovered faults or requirements can take substantial time and effort, as missed requirements may force redesign of the software

**ADVANTAGE**

**ADVANTAGE**

* This website is only develop for education purpose.
* There is no need to go to faculty ask for notes and question paper.
* Time saving for both students and faculty.
* Faculty can upload as well as download notes and question paper.
* Student can download notes and question paper any time after uploading faculty.

**DISADVANTAGE**

**DISADVANTAGE**

* This web application is developed only for single department of an organization or college or university.
* Only authenticated student and faculty are used available resources on website.
* Only limited number of resources are available.
* This web application run only in organization or college or university.
* This web application properly work on Pc’s or laptop.

**DATAFLOW**

**DIAGRAM**

**(DFD)**

**DATA FLOW DIAGRAM**

The Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system. It enables you to represent the processes in your information system from the viewpoint of data. The DFD lets you visualize how the system operates, what the system accomplishes and how it will be implemented, when it is refined with further specification.

Data flow diagrams are used by systems analysts to design information-processing systems but also as a way to model whole organizations. You build a DFD at the very beginning of your business process modeling in order to model the functions your system has to carry out and the interaction between those functions together with focusing on data exchanges between processes. You can associate data with conceptual, logical, and physical data models and object-oriented models.

A DFD shows what kind of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

There are two types of DFDs, both of which support a top-down approach to systems analysis, whereby analysts begin by developing a general understanding of the system and gradually break components out into greater detail:

**LOGICAL DATA FLOW DIAGRAMS**

Are implementation-independent and describe the system, rather than how activities are accomplished.

**PHYSICAL DATA FLOW DIAGRAMS**

Are implementation-dependent and describe the actual entities (devices, department, people, etc.) involved in the current system.

When it comes to conveying how information data flows through systems (and how that data is transformed in the process), data flow diagrams (DFDs) are the method of choice over technical descriptions for three principal reasons.

(1) DFDs are easier to understand by technical and nontechnical audiences.

(2) DFDs can provide a high level system overview, complete with boundaries

and connections to other systems.

(3) DFDs can provide a detailed representation of system components.

DFDs help system designers and others during initial analysis stages visualize a current system or one that may be necessary to meet new requirements. Systems analysts prefer working with DFDs, particularly when they require a clear understanding of the boundary between existing systems and postulated systems. DFDs represent the following:

1. External devices sending and receiving data

2. Processes that change that data

3. Data flows themselves

4. Data storage locations

The hierarchical DFD typically consists of a top-level diagram (Level 0) underlain by cascading lower level diagrams (Level 1, Level 2…) that represent different parts of the system.

**VALID AND NON-VALID DATA FLOWS**

Before embarking on developing your own data flow diagram, there are some general guidelines you should be aware of. Data stores are storage areas and are static or passive; therefore, having data flow directly from one data store to another doesn’t make sense because neither could initiate the communication. Data stores maintain data in an internal format, while entities represent people or system external to them.

Because data from entities may not be syntactically correct or consistent, it is not a good idea to have a data flow directly between a data store and an entity regardless of direction.

Data flow between entities would be difficult it would be impossible for the system to know about any communication between them. The only type of communication that can be modeled is that which the system is expected to know or react to.

Process on DFDs have no memory, so it would not make sense to show data flows between two asynchronous processes (between two processes that may or may not be active simultaneously) because they may respond to different external events.

Therefore, data flow should only occur in the following scenarios:

* Between a process and an entity (in either direction)
* Between a process and a data store (in either direction)
* Between two processes that can only simultaneously

Data flow diagram is a highly effective technique for showing the flow of information through a system. DFDs are used in the preliminary stages of system analysis to help understand the current system and to represent a required system. The DFDs themselves represent external entities sending and receiving information (entities), the processes that change information (processes), the information flows themselves (data flow), and where information is stored (data stores). The hierarchical DFDs consist up a single top layer (Level 0 or the control diagram) that can be decomposed into many lower level diagrams (Level 1, Level 2, Level N), each representing different areas of the system.

DFDs are extremely useful in the systems analysis as they help structure the steps in object-oriented design and analysis. Because DFDs and object technology share the same syntax constructs, DFDs are appropriate for the 00 domain.

DFDs are a form of information development and as such provide key insight into how information is transformed as it passes through a system. Having the skills to develop DFDs from functional specs and being able to interpret them is values add skill set that is well within the domain of technical communications.

**RULES FOR DRAWING DFDS**

• A process must have at least one input and one output data flow.

• A process begins to perform its tasks as soon as it receives the necessary

input data flows.

• A primitive process performs a single well-defined function.

• Never label a process with an IF-THEN statement.

• Never show time dependency directly on a DFD.

• Be sure that data stores, data flows, data processes have descriptive titles.

Processes should use imperative verbs to project action.

• All processes receive and generate at least one data flow.

• Begin/end data flows with a bubble.

**DFD NOTATIONS**

The DFD may be partitioned into levels that represent increasing information flow and functional details. Five simple notations are used to complete a DFD. These notations are given below:-

* Data flow
* Process
* External Entity
* Data store
* Output

**DATA FLOW**

Data flow represents the input (or output) of data to (or from) a process, data store or an actor. Data flow only data, not control. Represent the minimum essential data the process needs. Using only the minimum essential data reduces the dependence between processes. Data flows must begin and/or end at a process.

Data flows are always named. Name is not to include the word "data". It should be given unique names. Names should be some identifying noun. For example, marks, order, payment, complaint, registration no.

A data flow is represented by an arrow as shown in Fig below.

Data flow

An arrow between the producer and the consumer of the data value represents a

data flow. Arrow is labeled with description of data. Data can be elementary or

aggregate. Input arrow indicates storing data in the data store and output arrow

indicates accessing of data from data store.

**PROCESS**

Processes are work or actions performed on incoming data flows to produce

outgoing data flows. These show data transformation or change. Data coming into a process must be "worked on" or transformed in some way. Thus, all processes must have inputs and outputs. In some cases, data inputs or outputs will only be shown at more detailed levels of the diagrams. Each process is always "running" and ready to accept data. Major functions of processes are computations and making decisions .Each process may have dramatically different timing: yearly, weekly, daily etc.

A process is depicted by a circle as shown below:

Process

Every process is named. Processes are named with one carefully chosen verb and an object of the verb. There is no subject. Name is not to include the word "process". Each process should represent one function or action. If there is an "and" in the name, you likely have more than one function (and process).

Processes should generally move from top to bottom and left to right.

**EXTERNAL ENTITY**

The External Entity symbol represents sources of data to the system or destinations of data from the system. They determine the system boundary. They are external to the system being studied. They are often beyond the area of influence of the developer. They can represent another system or subsystem. These go on margins/edges of data flow diagram. They are represented by a rectangle symbol and are named with appropriate name as shown in Fig below.

External Entity

Some authors call them actors as they are active objects that drive the data flow

diagram by producing or consuming values. Actors are attached to the inputs and outputs of a data flow diagram. Actors are also called as terminators as they act as source and sink for data.

**DATA STORE**

Data stores are repository for data that are temporarily or permanently recorded

within the system. It is an "inventory" of data. These are common link between data and process models. Only processes may connect with data stores.

There can be two or more systems that share a data store. This can occur in the

case of one system updating the data store, while the other system only accesses

the data. Data stores are represented by open rectangle or two parallel lines as

shown below.

Data store

Orr

Data store

Data stores are named with an appropriate name, not to include the word "file",

Names should consist of plural nouns describing the collection of data. Like

customers, orders, and products. These may be duplicated.

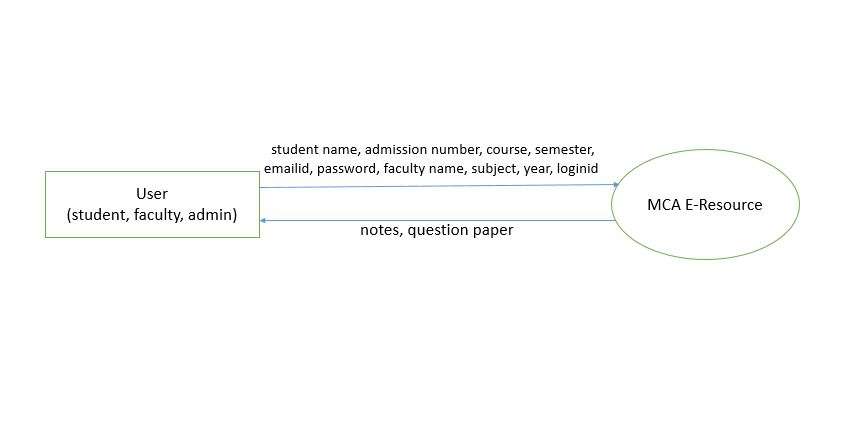
They store data for later use. They do not generate any operation on its own but can respond to request. That is why they are passive objects in a data flow diagram.

**OUTPUT**

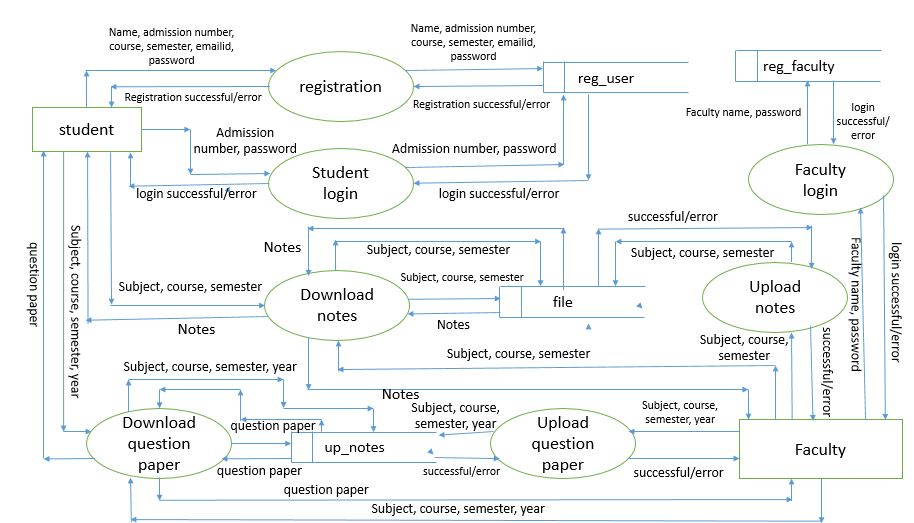
The output symbol is used when a hard copy is produced and the user of the copies cannot clearly specify or there are several users of output.

Output

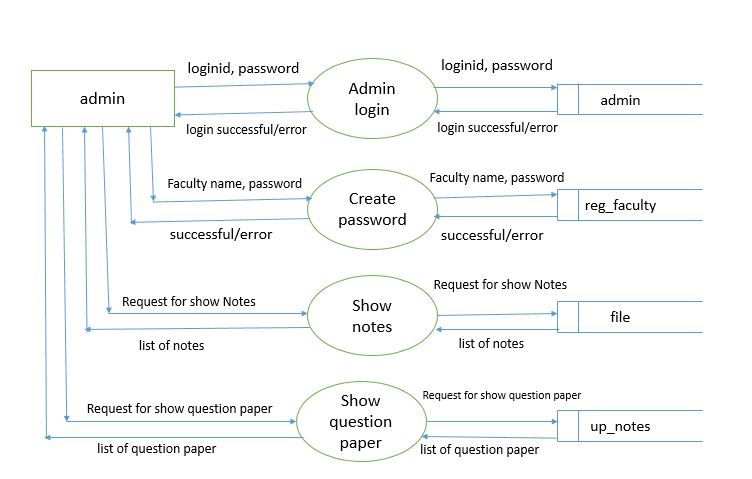
**0-LEVEL DFD**

****

**1-LEVEL DFD**

****

1-Level DFD Continue….



**DATABASE**

**DESIGN**

**DATABASE DESIGN**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

* Determine the relationships between the different data elements.
* Superimpose a logical structure upon the data on the basis of these

relationships.

**THE DESIGN PROCESS**

1. **Determine the purpose of the database**

This helps prepare for the remaining steps.

1. **Find and organize the information required**

Gather all of the types of information to record in the database, such as product name and order number.

1. **Divide the information into tables**

Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.

1. **Turn information items into columns**

Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table.

1. **Specify primary keys**

Choose each table’s primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be User ID or Registration ID.

1. **Set up the table relationships**

Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.

1. **Refine the design**

Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.

1. **Apply the normalization rules**

Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables.

**DATABASE FORMAT**

A properly designed database provides you with access to up-to-date, accurate information. Because a correct design is essential to achieving your goals in working with a database, investing the time required to learn the principles of good design makes sense. In the end, you are much more likely to end up with a database that meets your needs and can easily accommodate change. This article provides guidelines for planning a database. You will learn how to decide what information you need, how to divide that information into the appropriate tables

and columns, and how those tables relate to each other.

In this project, we have databases wherever it is required. The databases are required for the entities (as shown in the E-R diagram) present in the project.

The entities are:

1. admin table
2. file table
3. reg\_faculty table
4. reg\_user table
5. up\_notes table

**admin table**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Type | Size | Key |
| loginid | varchar | 15 | primary key |
| admin\_pass | varchar | 20 |  |

**file table**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Type | Size | Key |
| id | Int | 10 | primary key |
| course | varchar | 5 |  |
| sem | varchar | 10 |  |
| sub | varchar | 20 |  |
| name | varchar | 255 |  |
| mine | varchar | 50 |  |
| size | bigint | 25 |  |
| data | mediumblob |  |  |
| created | datetime |  |  |

**reg\_faculty table**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Type | Size | Key |
| faculty\_name | varchar | 25 | Primary key |
| faculty\_pass | varchar | 15 |  |

**reg\_user table**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Type | Size | Key |
| std\_name | varchar | 20 | primary key |
| adm\_no | varchar | 11 |  |
| course | varchar | 6 |  |
| sem | Int | 3 |  |
| email | varchar | 25 | unique key |
| pass | varchar | 15 |  |
| compass | varchar | 15 |  |

**up\_notes table**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Type | Size | Key |
| id | Int | 10 | primary key |
| course | varchar | 5 |  |
| sem | varchar | 10 |  |
| sub | varchar | 20 |  |
| year | varchar | 10 |  |
| name | varchar | 255 |  |
| mine | varchar | 50 |  |
| size | bigint | 20 |  |
| data | mediumblob |  |  |
| created | datetime |  |  |

**TESTING**

**WHAT IS SOFTWARE**

**TESTING**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the [software](http://en.wikipedia.org/wiki/Software) to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding [software bugs](http://en.wikipedia.org/wiki/Software_bug) (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

* meets the requirements that guided its design and development,
* works as expected,
* can be implemented with the same characteristics,
* and satisfies the needs of stakeholders.

Software testing, depending on the testing method employed, can be implemented at any time in the software development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the [Agile](http://en.wikipedia.org/wiki/Agile_software_development) approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

**BASICS OF SOFTWARE TESTING**

There are two basics of software testing: blackbox testing and whitebox testing.

**WHITE-BOX TESTING**

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. [in-circuit testing](http://en.wikipedia.org/wiki/In-circuit_test) (ICT).

While white-box testing can be applied at the [unit](http://en.wikipedia.org/wiki/Unit_testing), [integration](http://en.wikipedia.org/wiki/Integration_testing) and [system](http://en.wikipedia.org/wiki/System_testing) levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

* [API](http://en.wikipedia.org/wiki/Application_programming_interface) testing (application programming interface) – testing of the application using public and private APIs
* [Code coverage](http://en.wikipedia.org/wiki/Code_coverage) – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* [Fault injection](http://en.wikipedia.org/wiki/Fault_injection) methods – intentionally introducing faults to gauge the efficacy of testing strategies
* [Mutation testing](http://en.wikipedia.org/wiki/Mutation_testing) methods
* [Static testing](http://en.wikipedia.org/wiki/Static_testing) methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important [function points](http://en.wikipedia.org/wiki/Function_points) have been tested.[[22]](http://en.wikipedia.org/wiki/Software_testing#cite_note-22) Code coverage as a [software metric](http://en.wikipedia.org/wiki/Software_metric) can be reported as a percentage for:

* Function coverage, which reports on functions executed
* Statement coverage, which reports on the number of lines executed to complete the test

100% statement coverage ensures that all code paths or branches (in terms of [control flow](http://en.wikipedia.org/wiki/Control_flow)) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

**BLACK-BOX TESTING**

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: [equivalence partitioning](http://en.wikipedia.org/wiki/Equivalence_partitioning), [boundary value analysis](http://en.wikipedia.org/wiki/Boundary_value_analysis), [all-pairs testing](http://en.wikipedia.org/wiki/All-pairs_testing), [state transition tables](http://en.wikipedia.org/wiki/State_transition_table), [decision table](http://en.wikipedia.org/wiki/Decision_table) testing, [fuzz testing](http://en.wikipedia.org/wiki/Fuzz_testing), [model-based testing](http://en.wikipedia.org/wiki/Model-based_testing), [use case](http://en.wikipedia.org/wiki/Use_case) testing, [exploratory testing](http://en.wikipedia.org/wiki/Exploratory_testing) and specification-based testing.

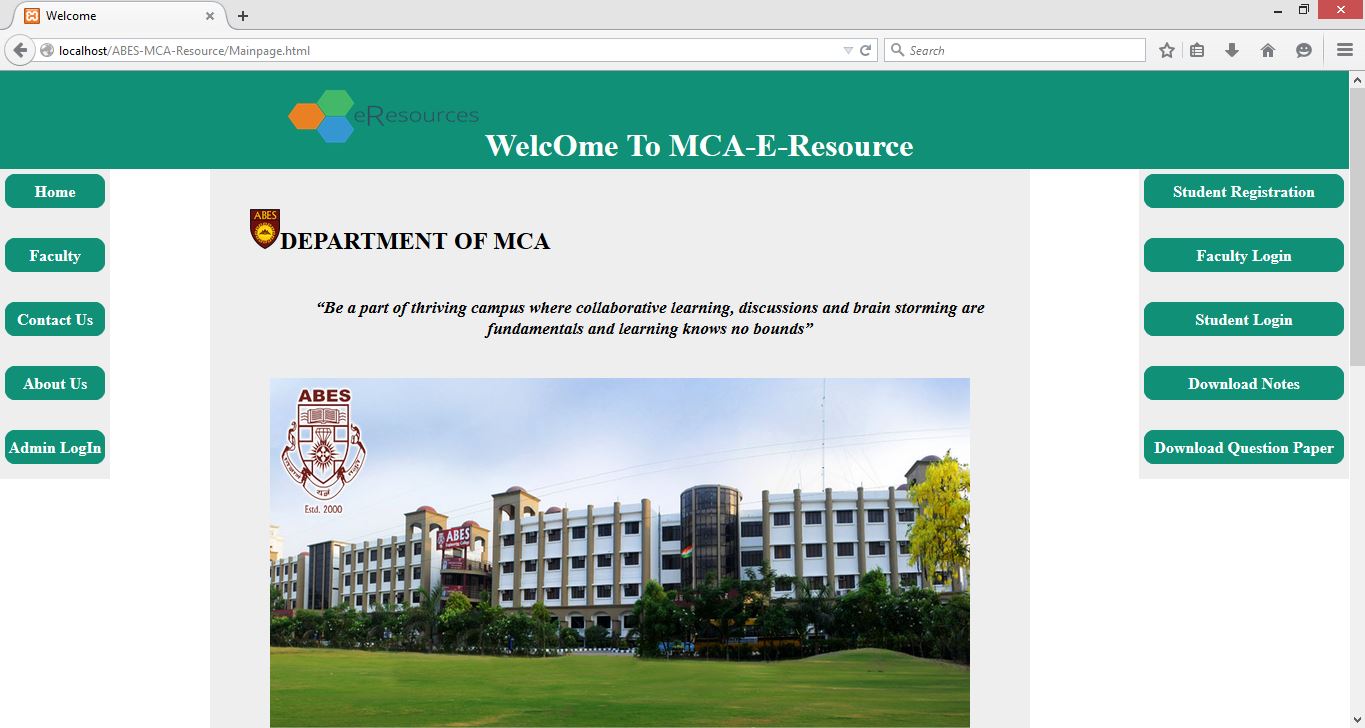
Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough [test cases](http://en.wikipedia.org/wiki/Test_case) to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be [functional](http://en.wikipedia.org/wiki/Functional_testing) or [non-functional](http://en.wikipedia.org/wiki/Non-functional_testing), though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

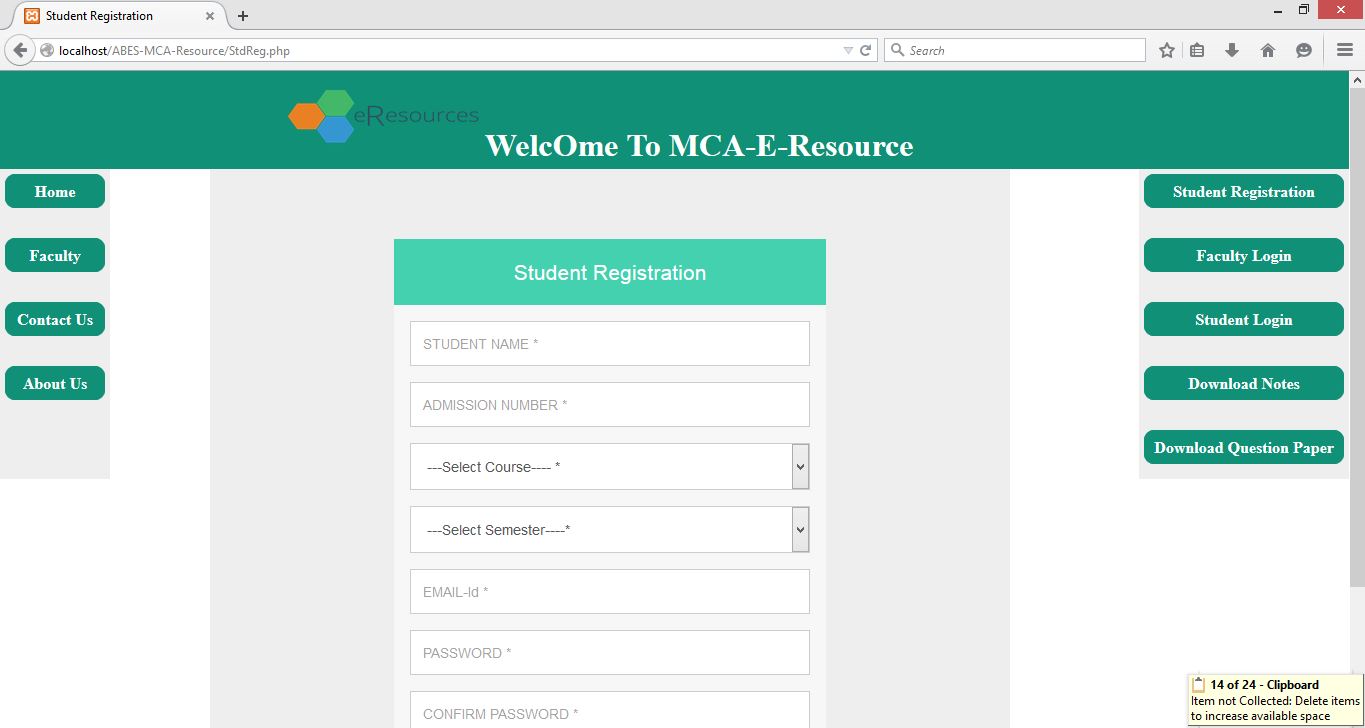
One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight". Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: [unit](http://en.wikipedia.org/wiki/Unit_test), [integration](http://en.wikipedia.org/wiki/Integration_testing), [system](http://en.wikipedia.org/wiki/System_testing) and [acceptance](http://en.wikipedia.org/wiki/Acceptance_test). It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

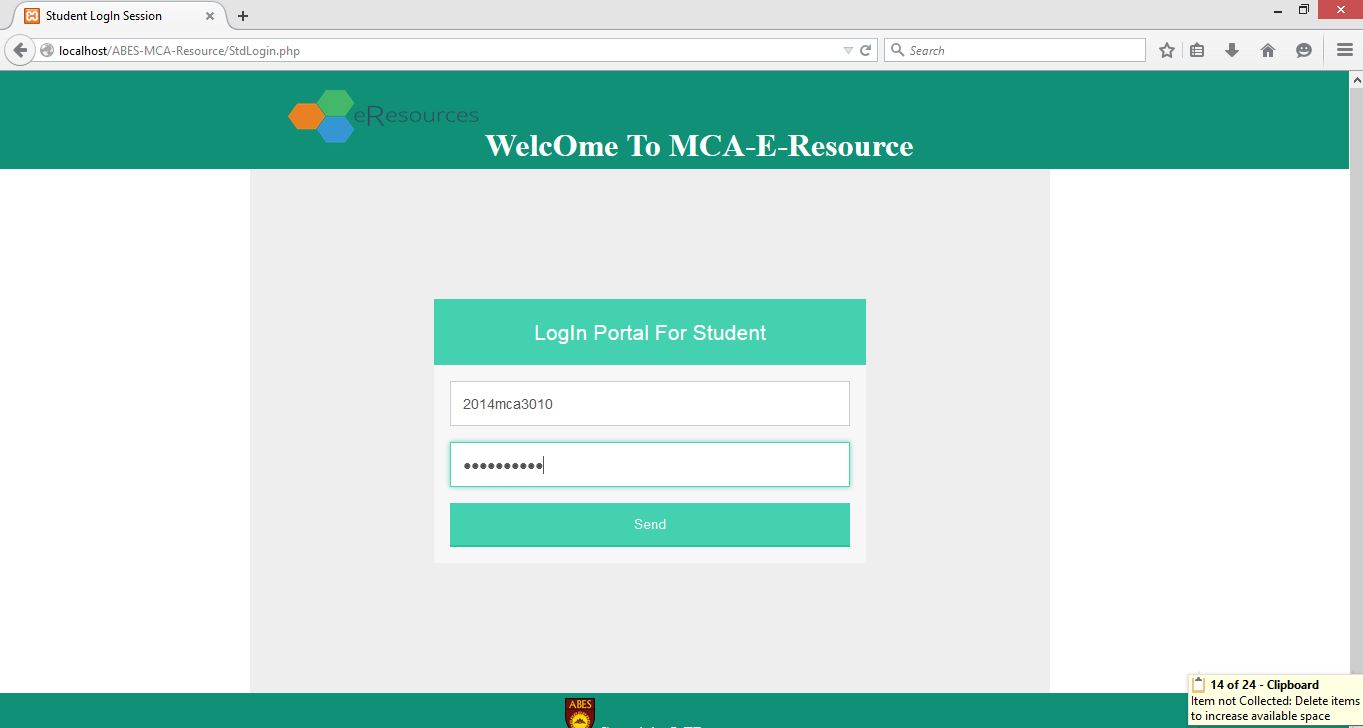
**IMPLEMENTATION**

****

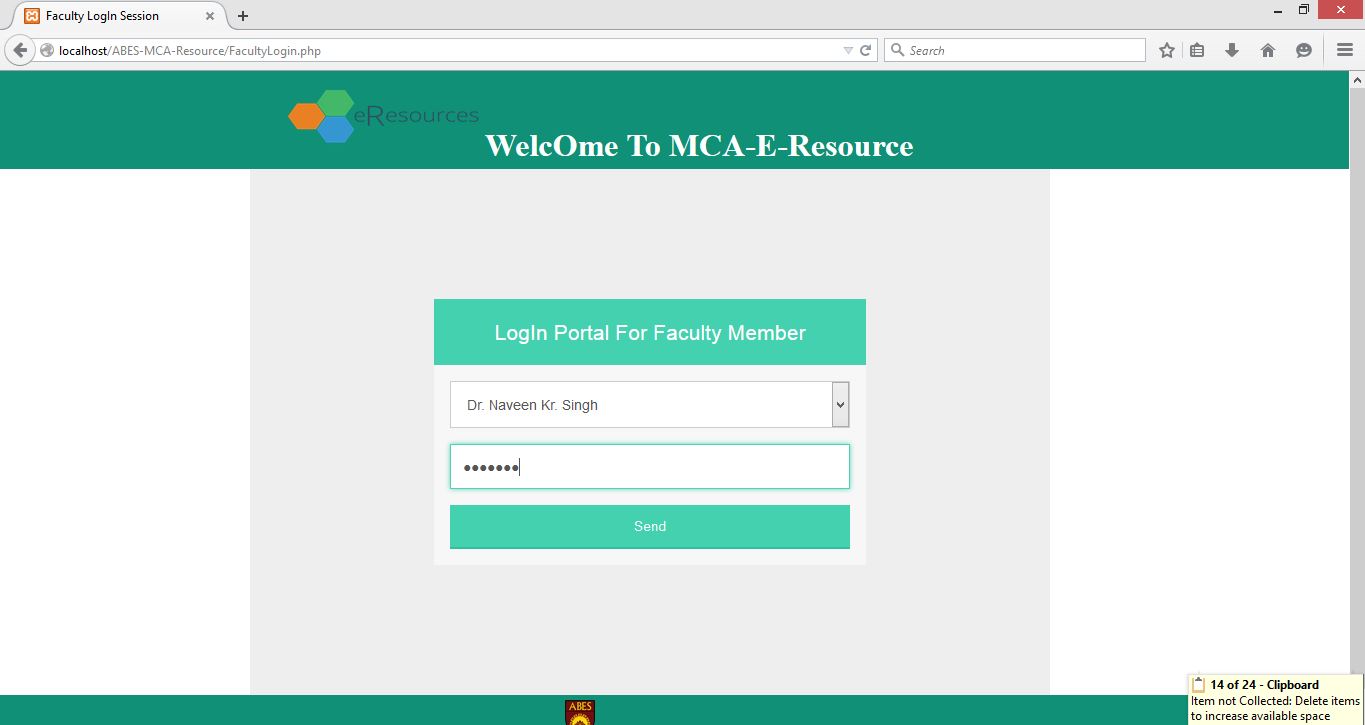
Home page of website

****

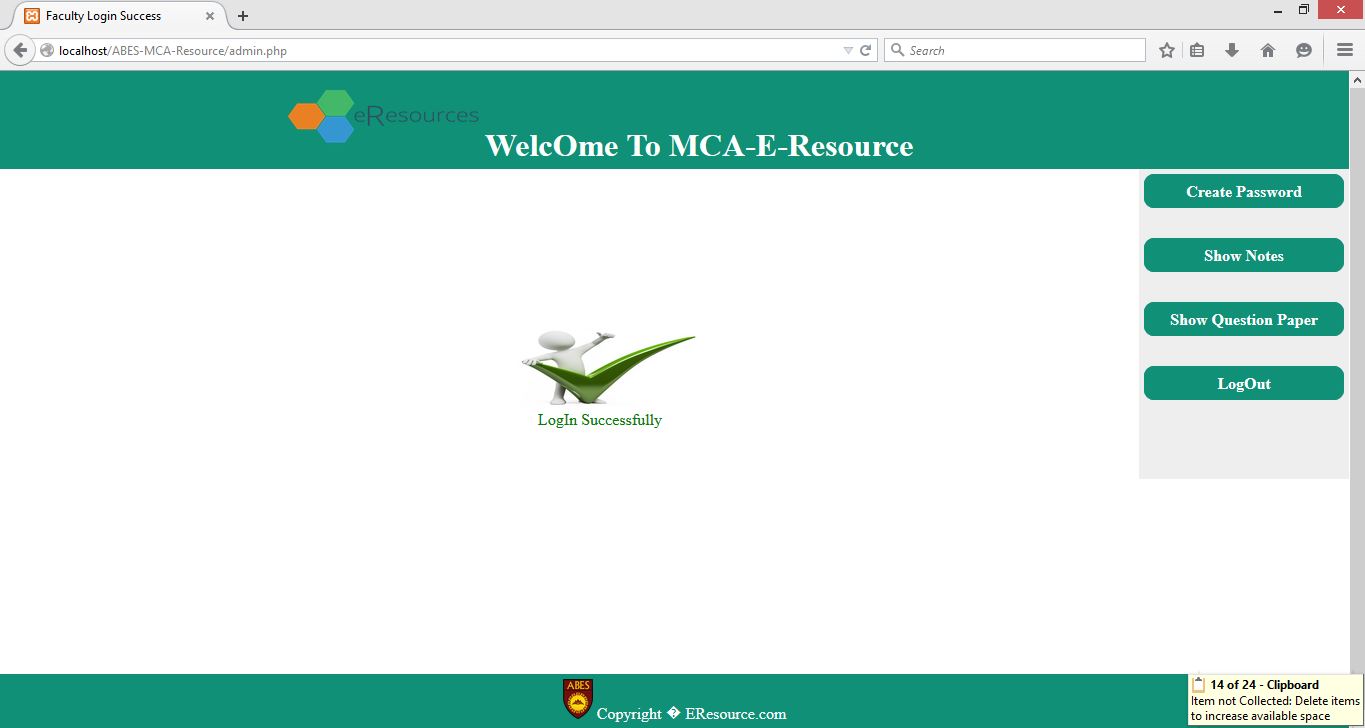
Registration form for students

****

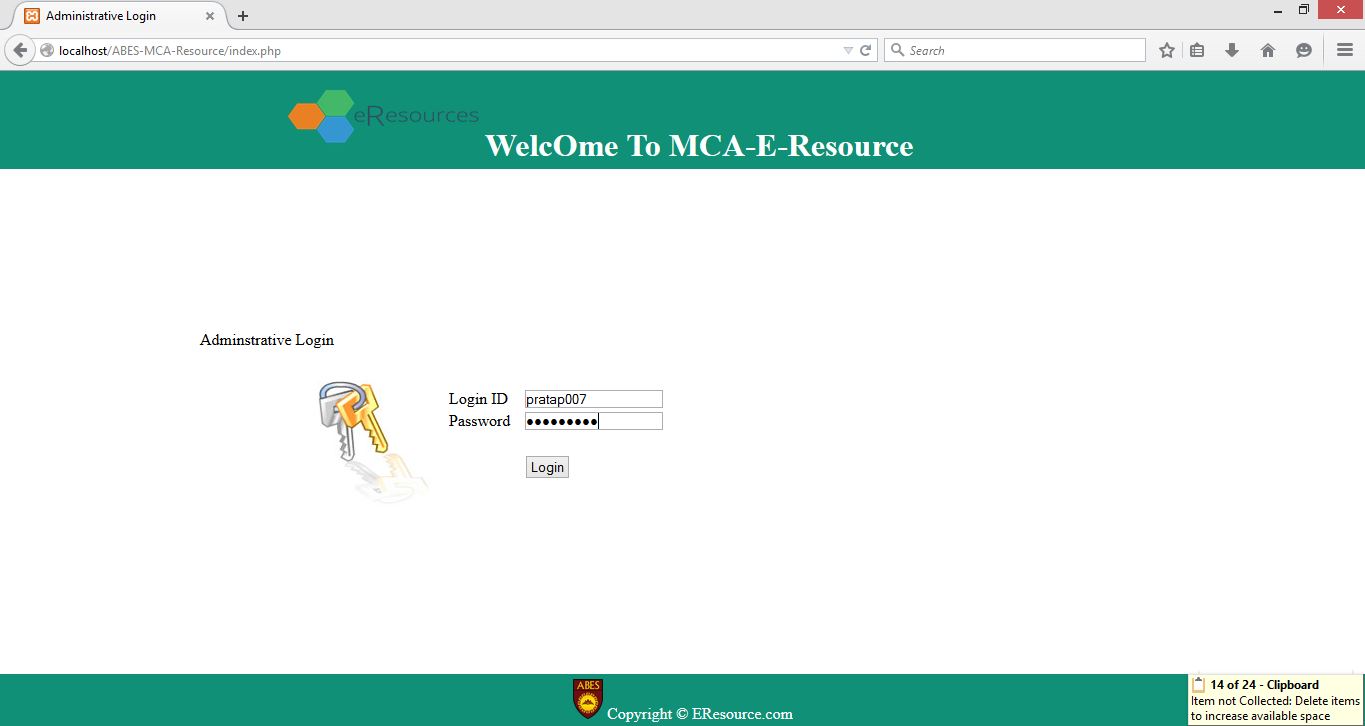
Login portal for student

****

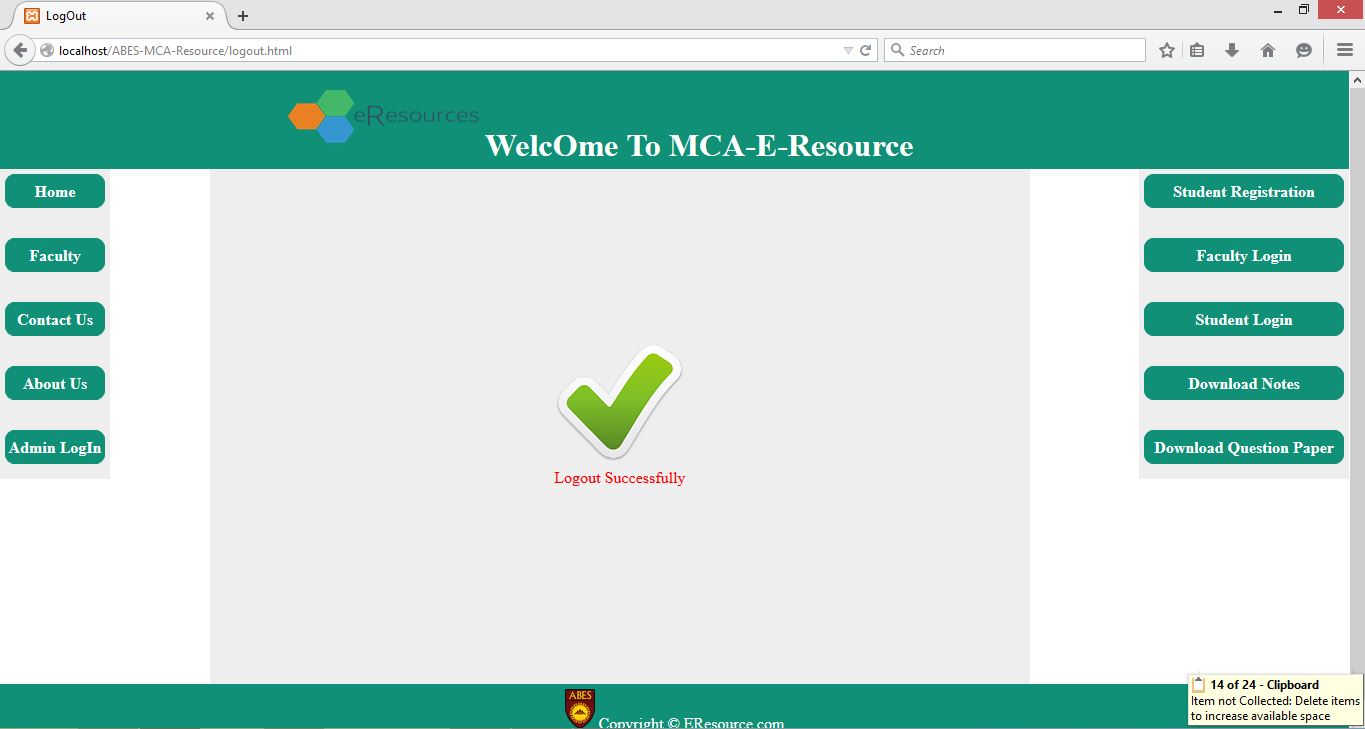
Login portal for faculty member

****

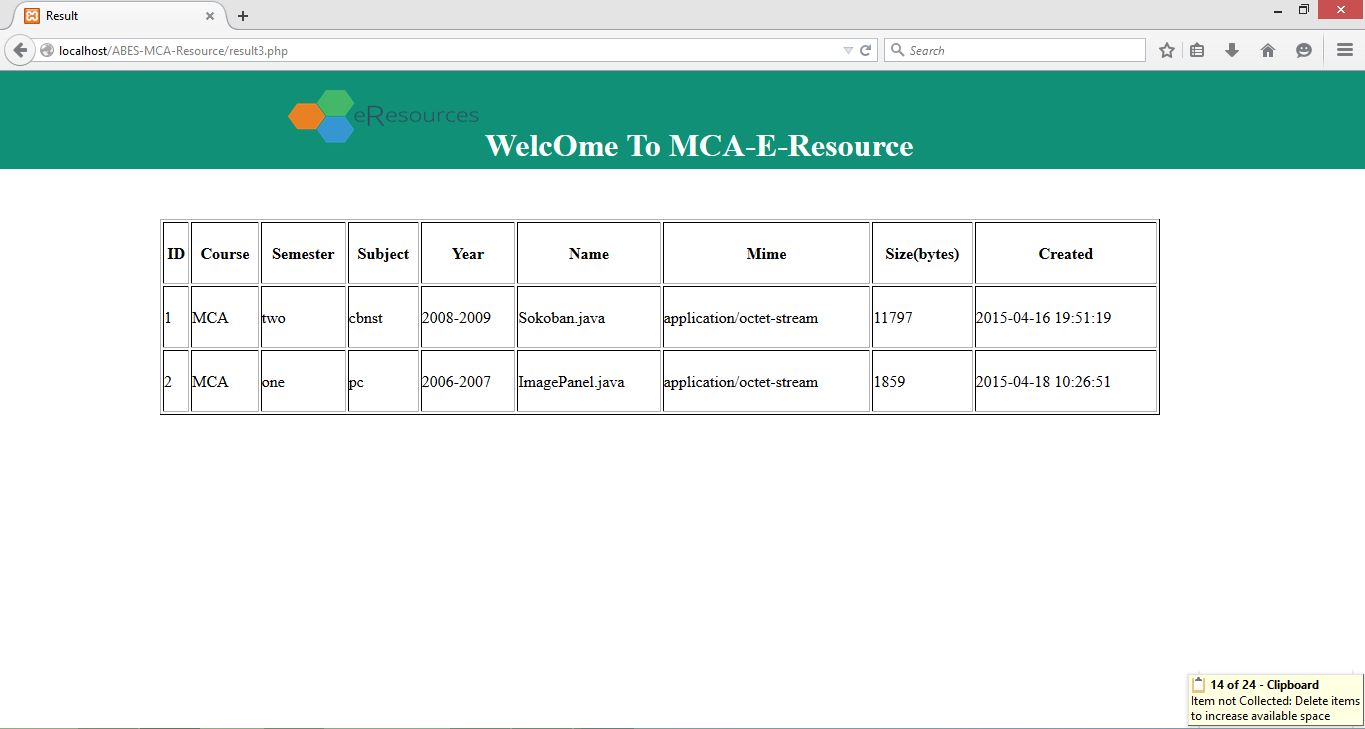
Successful login page for admin

****

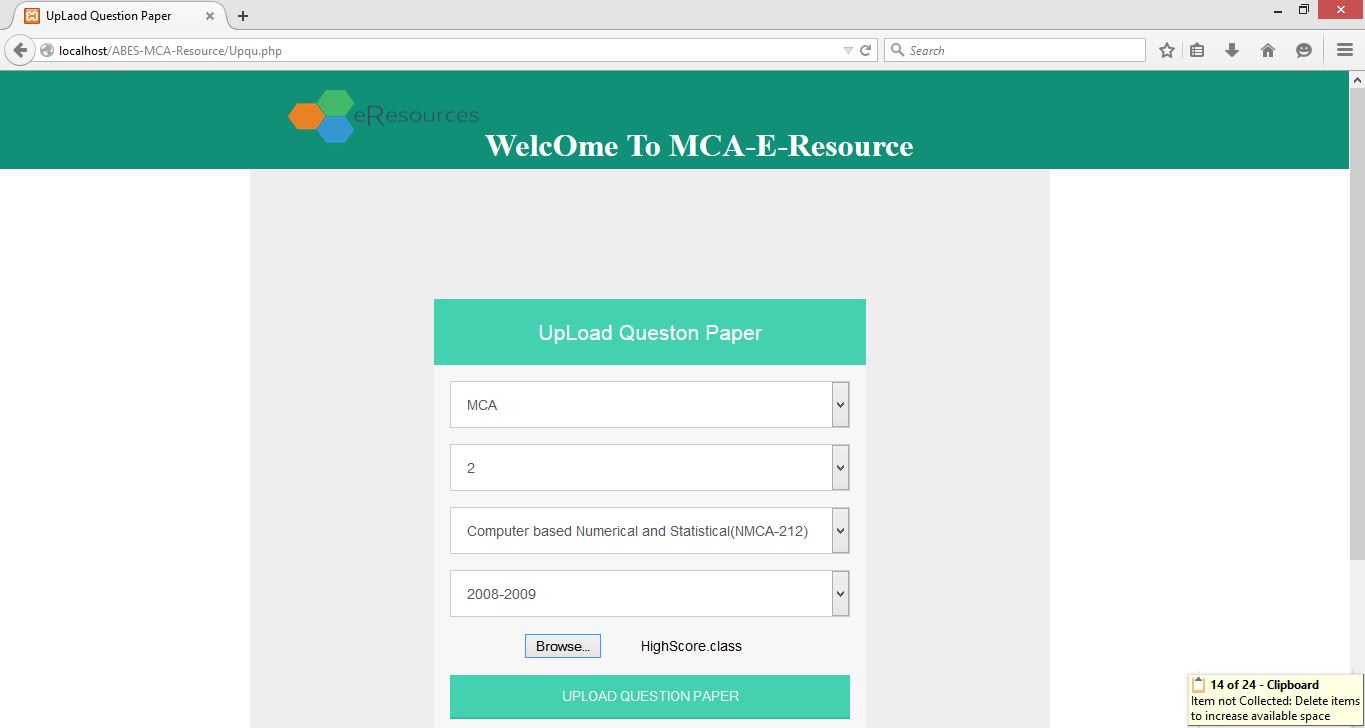
Login portal for admin

****

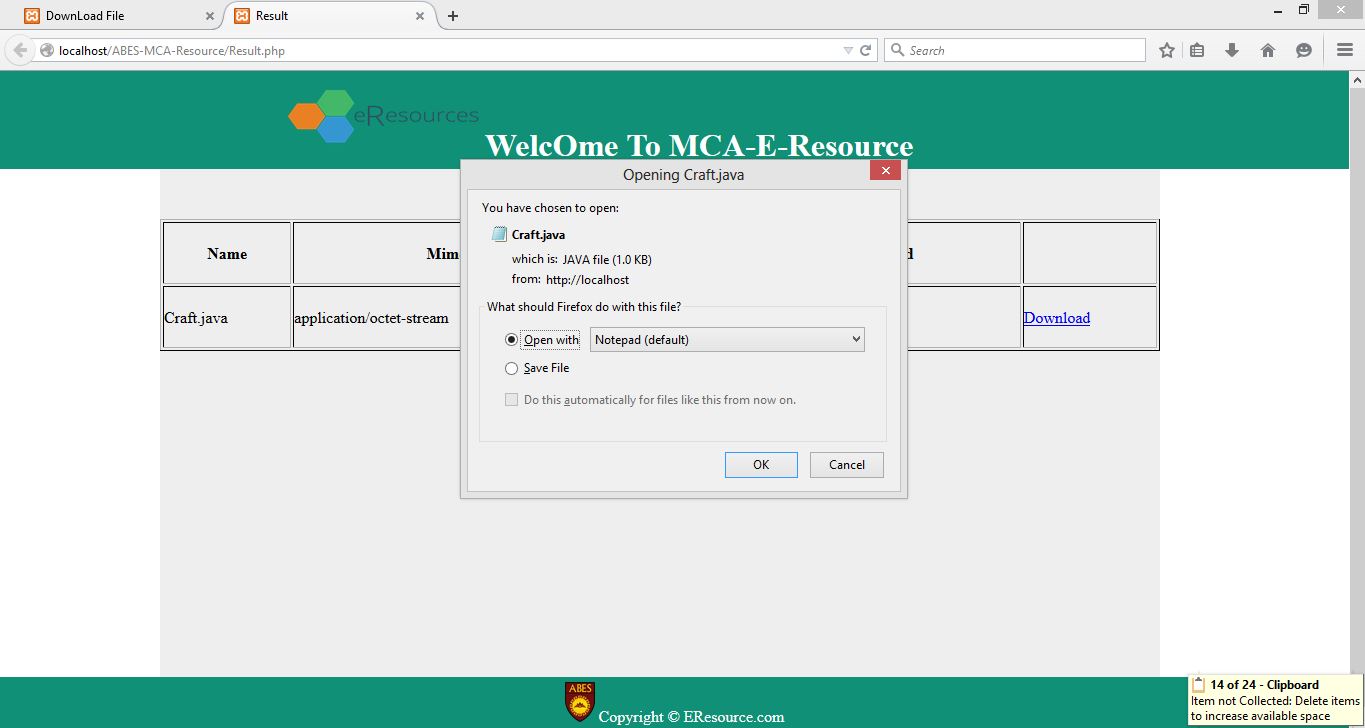
Logout page

****

List of uploaded question paper

****

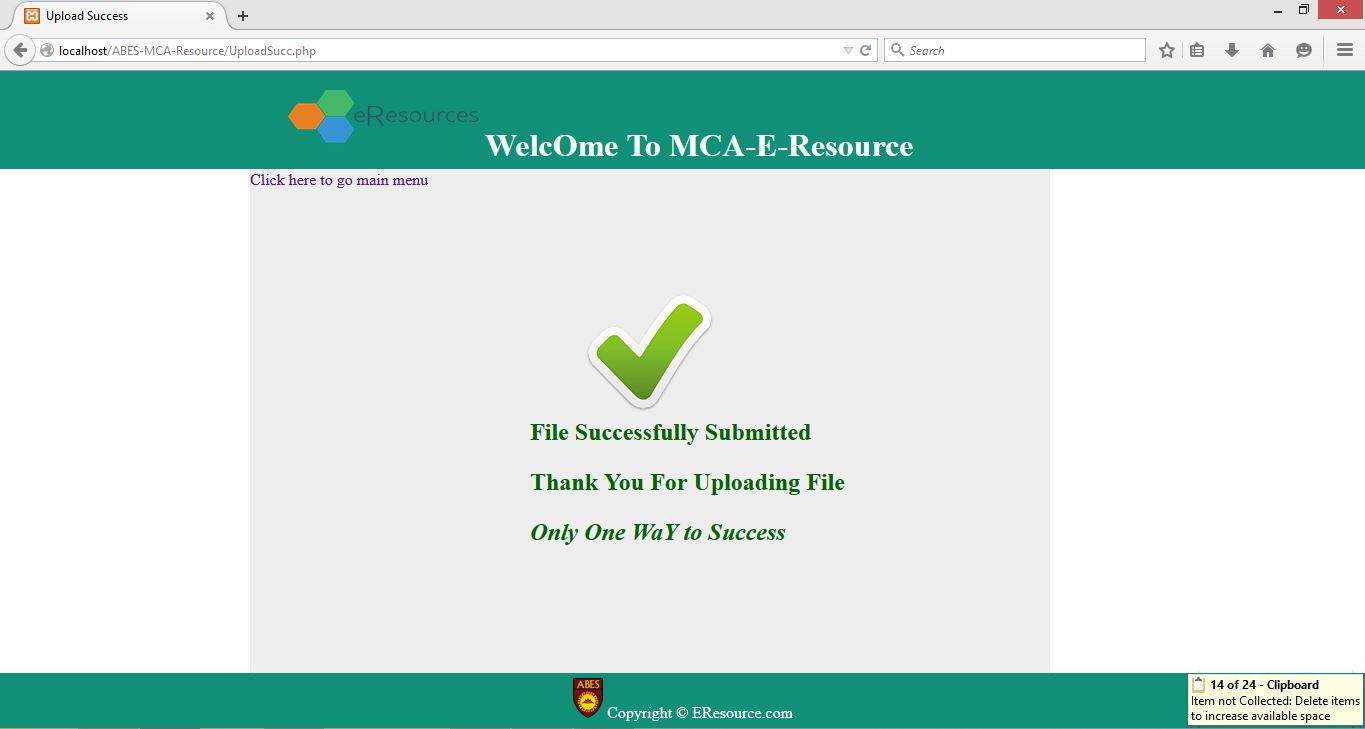
Upload Question paper

****

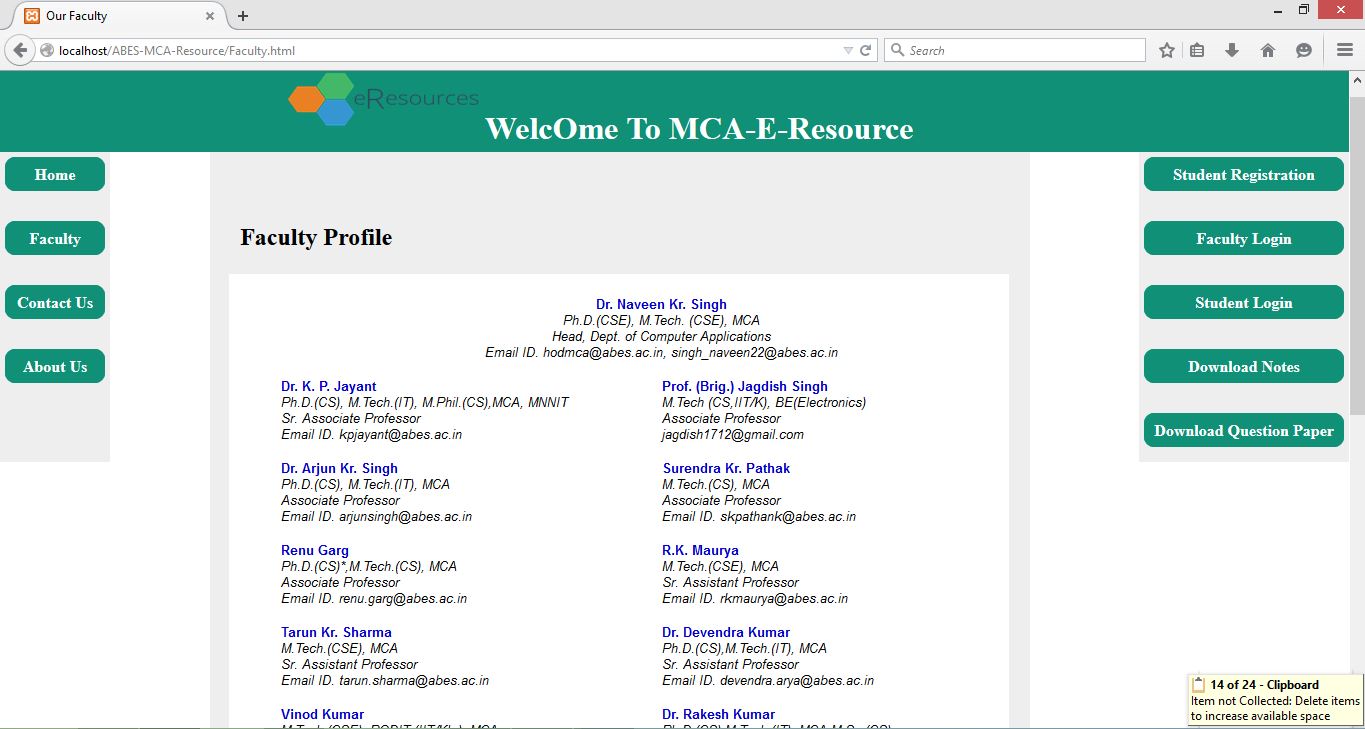
Downloading notes

****

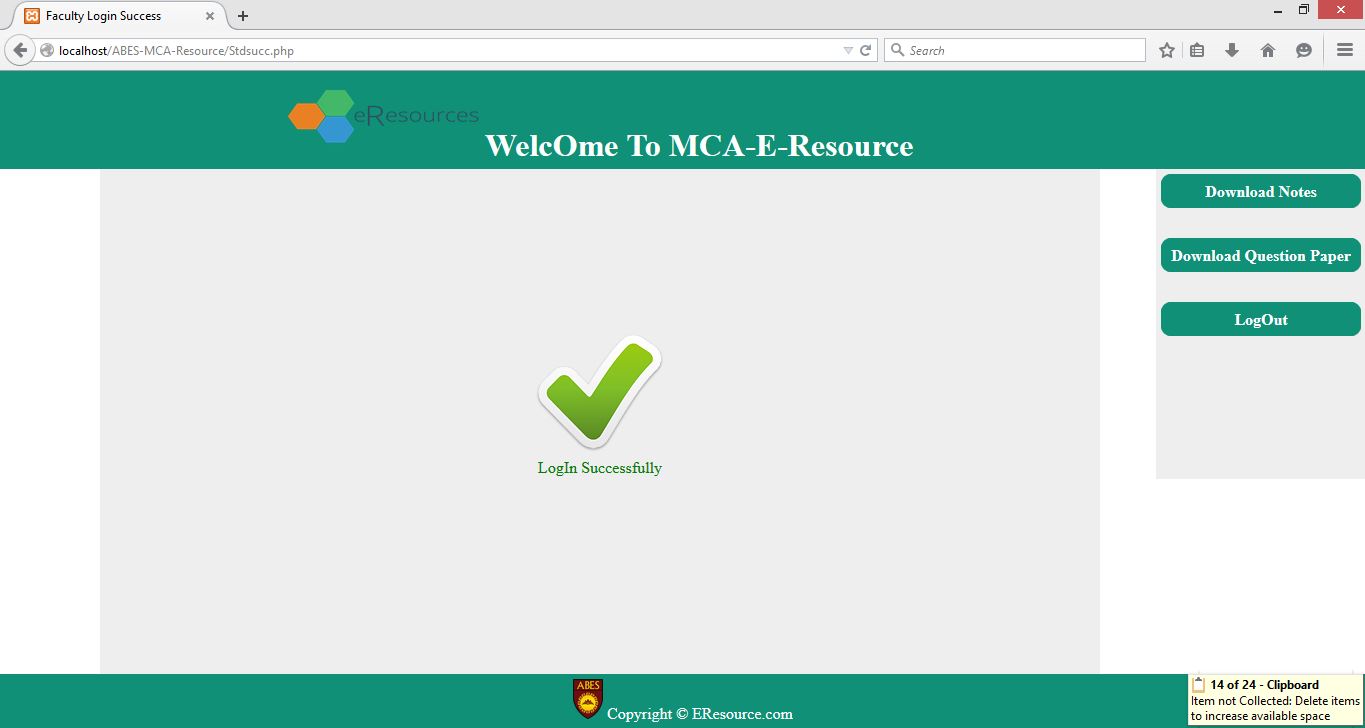
Contact detail of college

****

File submitted successfully

****

Faculty detail

****

Login successfully

**LIMITATION**

**LIMITATION**

* This web application is developed only for single department of an organization or college or university.
* Only authenticated student and faculty are used available resources on website.
* Only limited number of resources are available.
* This web application run only in organization or college or university.
* This web application properly work on Pc’s or laptop.

**FUTURE**

**ENHANCEMENT**

**FUTURE ENHANCEMENT**

* Web application can be access from all over the world.
* Web application can be develop for smart phone/mobile phone also.
* Every student can download notes or question paper.
* We can add many features like student attendance record, implement forum, notice board etc.

**CONCLUSION**

**CONCLUSION**

After study of all points of our project which are mostly connected to this project we consider these points and we add in this project. My web application known as MCA E-Resource Stick worked really well. It is now performing all the features that we aimed for.The best part of my web application is that it’s very user friendly. We have documented the design of our web application in such a way that anybody can alter it by reading my documentation.

In making of this project also help concept of software engineering such as making of Software Requirement Specification (SRS), Data Flow Diagram (DFD), and many other points we have an idea about web application that my project how make.

So we get complete demo of my project which are also making in convert it into code.

**BIBLIOGRAPHY**

**BIBLIOGRAPHY**

**WEBSITE REFERANCE**

* <http://wiki.answers.com>
* [www.wikipedia.com](http://www.wikipedia.com)
* www.w3.org
* [www.blogspot.com](http://www.blogspot.com)
* hotwired.lycos.com/webmonkey/

**BOOK REFERANCE**

* Web Programming Building Internet Applications --- Chris Bates
* Web Design A Beginner’s Guide --- Wendy Willard
* HTML: A Beginner’s Guide --- Wendy Willard
* HTML Goodies --- Joe Burns
* Designing with JavaScript --- Nick Heinle
* DHTML: The Definitive Guide --- Danny Goodman