**PANJAB UNIVERSITY**

**UNIVERSITY INSTITUTE OF COMPUTING**

**Project Report**

**On**

**“ONLINE EXAMINATION SYSTEM”**

**Submitted By: Kanav Singla**

**ONLINE EXAMINATION SYSTEM**

The software system is an online examination system that helps students facing problems in online examination system taken by many of the competitive and entrance examination system. The project consists of a several exams that student can access and give to for free. These examinations are generally based on current online competitive exam pattern. These exams will helps the students to get familiar with and can able to achieve time management and question attempting strategies. The project is an online system that will allow the students to access it from anywhere, anytime and any number of time user can give the exam to improve themselves. Moreover the system is also to prove latest exams information to user’s via email to students mailboxes. This online system is developed on javascript frameworks and supported by an mongodb database to exam,questions and user specific details and results.

Features:

* **Online Registration/Login :** The Student that need to give online exam they have to register and then have to login to give exam.
* **Student Registration:** Student have to register by providing their details.
* **Exam Notification:** Students will get notified about new exam that will be added.
* Student will feel comfort with our examination dashboard which will have similar layout as original.
* **Results to inbox:** The Final Result will be delivered to the students inbox.

**INTRODUCTION**

Online examination system is a web based application to provide online examination, tutorials and previous year questions online.

Online examination is conducting a test online to measure the knowledge of the participants on a given topic. In the olden days everybody had to gather in a classroom at the same time to take an exam. With online examination students can do the exam online, in their own time and with their own device, regardless where they life. You online need a browser and internet connection.

This website is basically for students to get familiar with the online based examination that are most popular and adapted examination system now a days..In this examination portal student have to register for the exam online. He will be provided a dashboard from which they can access the services provided my us . So, there is no need to walk around everything can be done at this own place.Once the exam is over so the result will be shown don’t need to wait for the result. Students will be proved their statistics of the tests given by them in their dashboard for improving themselves.

**OBJECTIVES**

The main objective of this Project is to conduct online entrance exam and allow student learn by giving online test . Students will able to familiar with online examination system and improve their speed to get able to success in online entrance or competitive examinations .

we are providing following features to the students:

* Free online exam portal
* Exam layout as similar to many online competitive examinations.
* Results and statistics in their inbox.

**EXISTING SYSTEM**

The whole process of assigning test and evaluating their score after the test was done manually till date.Processing the test paper i.e checking and distributing respective score score used to take time when the software was not installed.

Currently, some of websites on Online Examination System are available but we have found that they are not free, so we are working to provide these facilities for free to students those who can’t able get those facilities.

We tried our best to have our project on camping on the mark.

Limitation of Existing System:

* Time consuming for creating question paper.
* Time to check right and wrong answer.
* calculation of marks.
* human error.

**Kind of information needed**

* Before one determines where to go and what tool to use, first requirement is to figure out what information to gather. The basic information required is what problem does students face while giving online examinations.

**Sources of information**

* Information is gathered from two main sources namely personal and feedback from others students.
* Two main sources of information are: -
* External sources
* Internal sources.

**Proposed System along with intended objectives**

* Create a website dynamic in nature.
* Create a User friendly interface.
* Emphasis on exam dashboard to make it similar as original online exam’s.
* Easy accessible by every person

**Project Objective**

* To provide free online exam experience to users.
* Every time accessibility online.
* Providing best organizing to every user anywhere.

**Project Scope**

This solution is in web-based in which it can only be accessed from the Internet. So, every effort is taken to implement this project of ONLINE EXAMINATION SYSTEM , on successful implementation of this ONLINE EXAMINATION SYSTEM , we can target all competitive exams.

**Feasibility Study**

The feasibility of the system can be judged according to its workability impact on the organization, ability to meet user needs and the effective use of resources. One should keep in mind the need of the user and how does a candidate system meet it.

**Technical Feasibility**

This Information System for Project Online Examination System to serves free online exam portal and is very much technically feasible. We have technical guarantee and reliability as it has been tested by our team members. This project can be run on computers having internet access and web browser installed on it. It is easy to use even by the persons with little knowledge of computers.

**Economic Feasibility**

This system is economically feasible also as the cost of making the project is very low as it is based on node.js server ,angular 4 as front-end javascript, online mongodb database server(mlab) and can be deployed to any cloud as github repository and access from anywhere. Cost in developing the software is very low, as compared to the money spent on the existing system. This project can be run on any free cloud providing PAAS(platform as a service) like Heroku now what we are using .

**Operational Feasibility**

The client’s requirement is also taken into consideration then the system is designed. Operational feasibility of Information System for Project Online Examination system is satisfied as the running of this system satisfies the user and the client is also satisfied by the economic and technical feasibility.

**Team Structure**

Often a team of people is assigned a project. For team to work as a group and contribute most to the project, the people in the team have to be organized in some manner. This structure of Team has a direct impact on the product quality and project productivity. The structure of my team democratic. Democratic team consists of ten or fewer. The structure allowed input from all the members, which led to better decisions in difficult situations.

**FUNCTIONAL SPECIFICATION**

• **Admin** : Admin will be protected by username and password .Ordinary user of website will not be permitted to enter into the Admin Panel. Admin will able control and manage all the application data.Admin is also responsible for uploading questions and adding more users(sub admin) for uploading and managing exams.

• **Students** : Students have to register to get access to the exams he/she wants to give.Students can also see their previous records and results.

**Software Specifications**

**Table 1.1 Configuration setting for E- Examination.**

|  |  |
| --- | --- |
| **Technology Implemented** | **Nodejs (8.9.4)** |
| **L Language Used** | **Javascript (Es 2015)** |
| **D Database** | **MongoDB (3.5.12)** |
| **Middleware** | **ExpressJS (4.16.0)** |
| **U User Interface Design** | **Angular 5, MDBootstrap(css framework)** |
| **w Web Browser** | **Mozilla, Google Chrome, Internet Explorer** |

**Hardware Requirements (Server)**

**Table 1.2 Configuration setting for E-Examination.**

|  |  |
| --- | --- |
| **P Processor** | **P Core i2, AMD or Higher Version** |
| **Operating system** | **Windows server 2008,linux** |
| **R RAM** | **2 2GB recommended** |
| **H Hardware Devices** | **K keyboard with mouse** |
| **H Hard disk** | **10GB or More** |
| **D Display** | **S Standard Output Display** |

**Hardware Requirements (Client)**

**Table 1.3 Configuration setting for E-Examination.**

|  |  |
| --- | --- |
| **B**  **Browser** | **C**  **Chrome,Opera,Safari,Mozilla Firefox.**  **( (Latest)** |
| **O**  **Operating System** | **A**  **Any** |
| **D**  **Device** | **D**  **Desktop** |
| **H**  **Hardware Devices** | **Keyboard with mouse** |

**JAVASCRIPT**

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The [ECMA-262 Specification](http://www.ecma-international.org/publications/index.html) defined a standard version of the core JavaScript language.

* JavaScript is a lightweight, interpreted programming language.
* Designed for creating network-centric applications.
* Complementary to and integrated with Java.
* Complementary to and integrated with HTML.
* Open and cross-platform

## Client-side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

## Advantages of JavaScript

The merits of using JavaScript are −

* Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
* Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

## 

## Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features :−

* Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
* JavaScript cannot be used for networking applications because there is no such support available.
* JavaScript doesn't have any multithreading or multiprocessor capabilities.

Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

## MEAN STACK

## Mean is an acronym made up of commonly used technologies for an all JavaScript web stack. You don’t have to use this combination and there are many alternative choices, especially on the client-side such as Backbone,Ember etc.This particular combination of tools has generated a lot of traction in the enterprise area and is framework based, making it a good place to start.

The key components are:

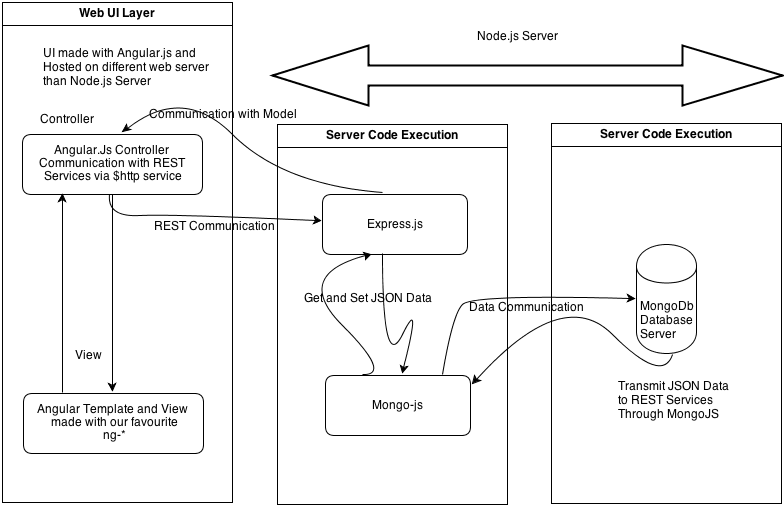
* **M**ongoDB (Database)
* **E**xpressJS (Web Framework)
* **A**ngularJS (Front-end Framework)
* **N**odeJS (Application Server)

MEAN brings together four of the most used and appreciated technologies for JavaScript development, laying down the foundation for easily building complex web applications.

**Database fig.1.1**



**MEAN ARCHITECTURE**



## 

## Node.js

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v**8.9.4**.

Node.js is a platform built on [Chrome's JavaScript runtime](https://code.google.com/p/v8/) for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

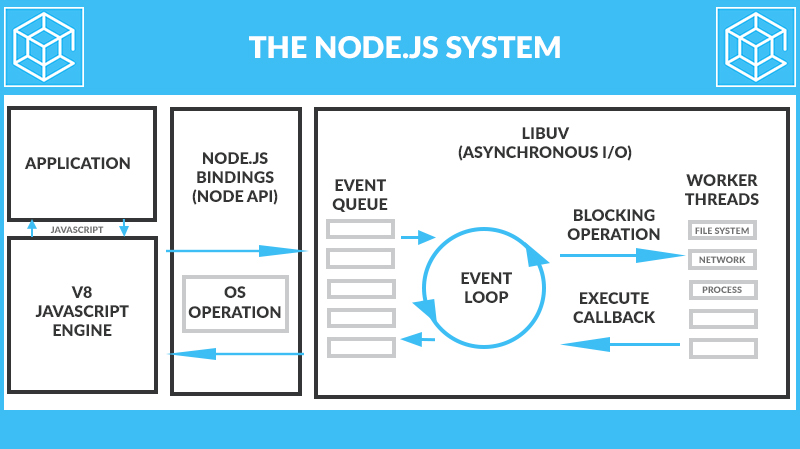
Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

**System diagram of nodejs server**

## 

## 



## 

## Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

* Asynchronous and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* Very Fast − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* Single Threaded but Highly Scalable − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* No Buffering − Node.js applications never buffer any data. These applications simply output the data in chunks.
* License − Node.js is released under the [MIT license](https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE).

## 

## 

## 

## Concepts

The following diagram depicts some important parts of Node.js which we will discuss in detail in the subsequent chapters.



**NPM**

Node Package Manager (NPM) provides two main functionalities −

* Online repositories for node.js packages/modules which are searchable on [search.nodejs.org](https://search.nodejs.org/)
* Command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

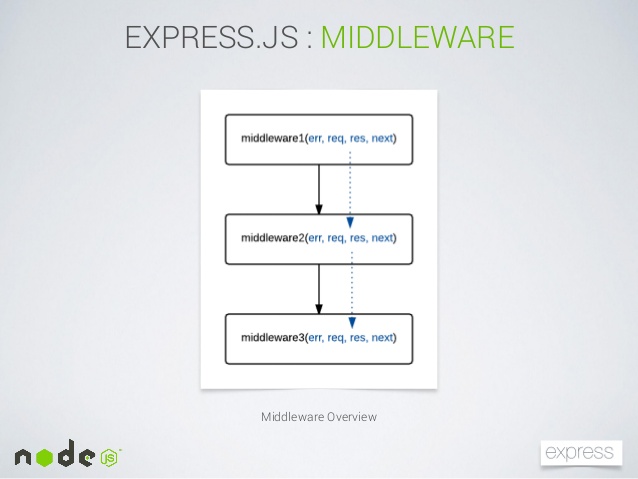
NPM comes bundled with Node.js installables after v**8.9.4** version. To verify the same, open console and type the following command and see the result −

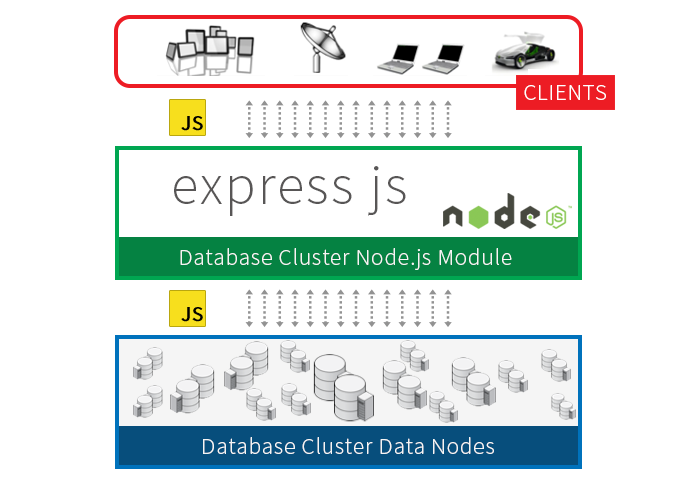
### 

### EXPRESS.JS FRAMEWORK BASICS

* **Server-side web and mobile application framework**
* **Language**: written in JavaScript
* **Express builds**:
  + Single-page, multi-page, and hybrid mobile and web apps
  + Common back-end functions for web applications
  + APIs (application programming interfaces)
* **Templating engines**: Express comes with two templating engines, Jade and EJS, which facilitate the flow of data into a website structure.
* **MVC pattern**: Express supports the Model-View-Controller architecture, a really helpful way to build websites in a model-driven format.
* **Platform**: Node.js
* **Operating system**: It’s cross-platform, so it’s not limited to one OS.
* **The Express Generator** lets you create complex applications quickly.

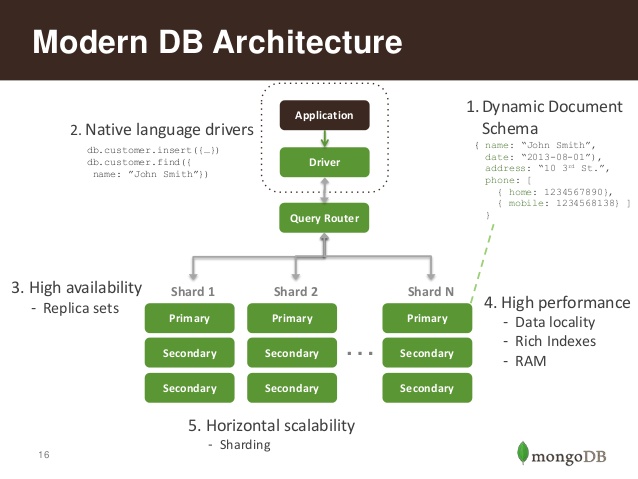
**System diagram of Express.js middleware**



****

**MongoDB**

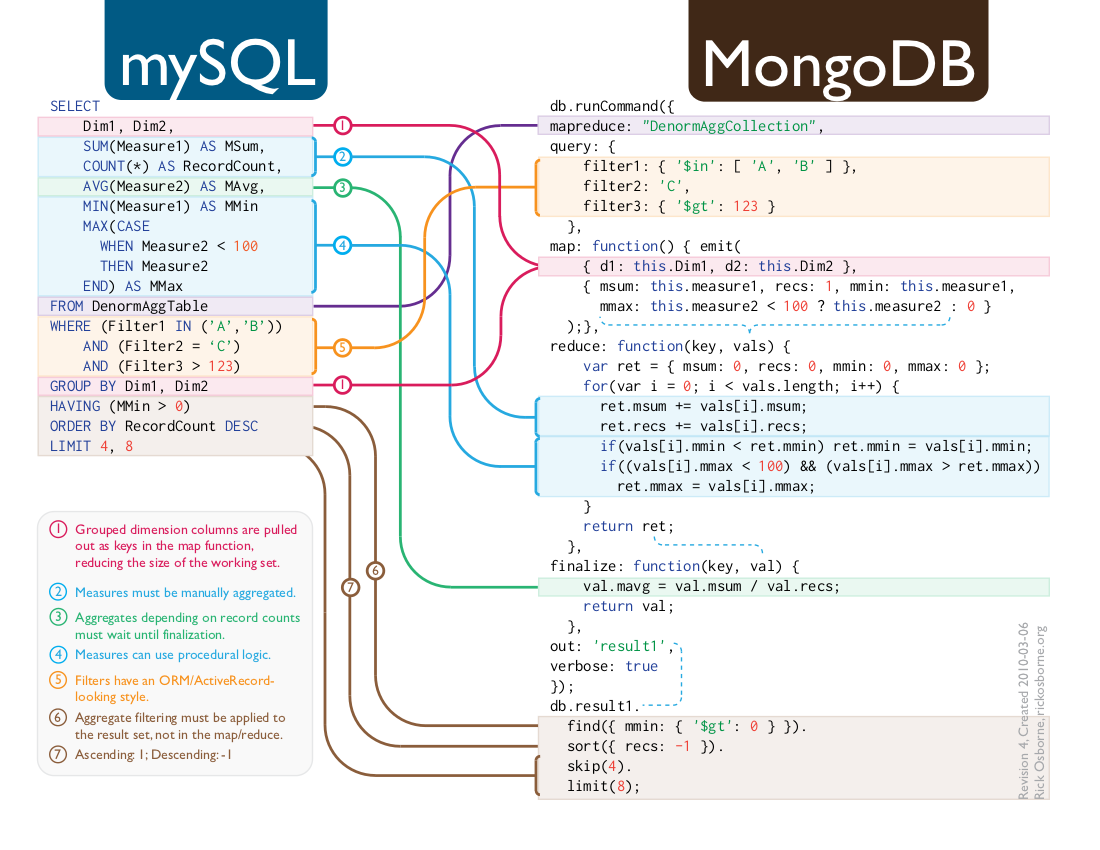
MongoDB is a schemaless NoSQL database system. MongoDB saves data in binary JSON format which makes it easier to pass data between client and server.



**MongoDB features:**

* MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
* The document model maps to the objects in your application code, making data easy to work with
* Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
* MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
* MongoDB is free and open-source, published under the GNU Affero General Public License
* Ad hoc queries - supports search by field, regular expression searches, and range queries.
* Indexing - any field in the BSON document can be indexed.
* Replication - provides high availability via replica sets which consists of two or more copies of the original data.
* Load balancing - sharding is the method used to allow MongoDB to scale horizontally, meaning that data will be distributed and split into ranges and then stored in different shards which can be located in different servers. Shard keys are used to determine how the data will be distributed.
* Aggregation - MapReduce can be applied to enable batch processing of data as well as perform aggregation operations.
* File storage - MongoDB can be used as file system which makes use of the above functions and acting in a distributed manner through sharding.

**Difference between Mysql and MongoDb**



## What Is NoSQL?

A common misconception is that the term *NoSQL* stands for “No SQL.” *NoSQL* actually stands for “Not only SQL,” to emphasize the fact that NoSQL databases are an alternative to SQL and can, in fact, apply SQL-like query concepts.

NoSQL covers any database that is not a traditional relational database management system (RDBMS). The motivation behind NoSQL is mainly simplified design, horizontal scaling, and finer control over the availability of data. NoSQL databases are more specialized for types of data, which makes them more efficient and better performing than RDBMS servers in most instances.

NoSQL seeks to break away from the traditional structure of relational databases, and enable developers to implement models in ways that more closely fit the data flow needs of their system. This means that NoSQL databases can be implemented in ways that traditional relational databases could never be structured.

Several different NoSQL technologies exist, including the HBase column structure, the Redis key/value structure, and the Virtuoso graph structure. However, this book uses MongoDB and the document model because of the great flexibility and scalability offered in implementing back-end storage for web applications and services. In addition, MongoDB is by far the most popular and well-supported NoSQL language currently available. The following sections describe some of the NoSQL database types.

**AngularJS**

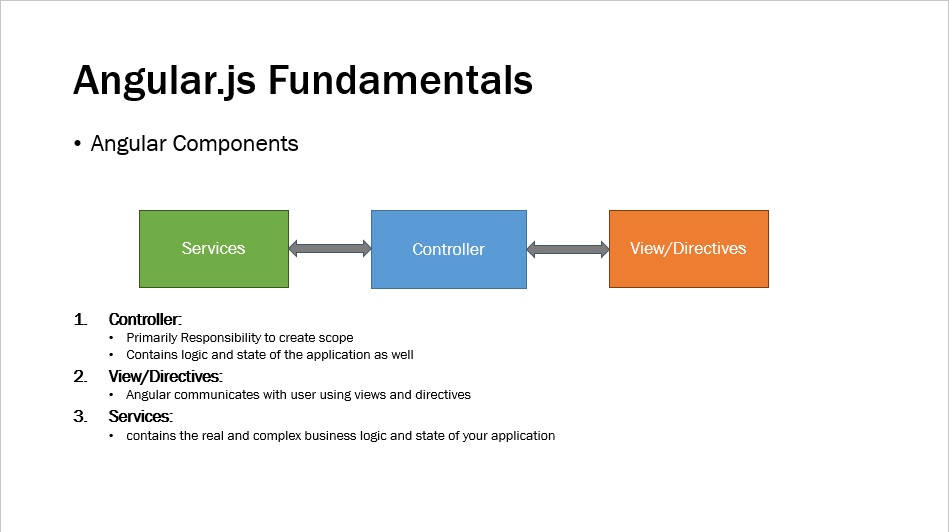
AngularJS is a JavaScript framework developed by Google. It provides some awesome features like the two-way data binding. It’s a complete solution for rapid and awesome front end development.

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in what do I have to do to trick the browser into doing what I want?

AngularJS takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. AngularJS teaches the browser new syntax through a construct we call directives. Examples include:

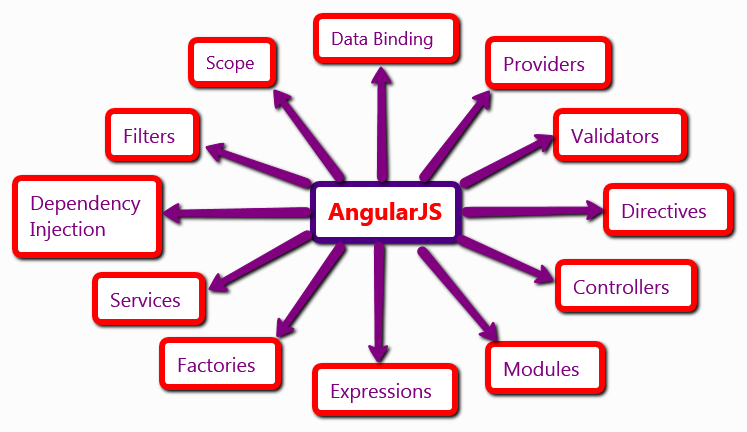
* Data binding, as in {{}}.
* DOM control structures for repeating, showing and hiding DOM fragments.
* Support for forms and form validation.
* Attaching new behavior to DOM elements, such as DOM event handling.
* Grouping of HTML into reusable components.



**Complete client-side solution**

AngularJS is not a single piece in the overall puzzle of building the client-side of a web application. It handles all of the DOM and AJAX glue code you once wrote by hand and puts it in a well-defined structure. This makes AngularJS opinionated about how a CRUD (Create, Read, Update, Delete) application should be built. But while it is opinionated, it also tries to make sure that its opinion is just a starting point you can easily change. AngularJS comes with the following out-of-the-box:

* Everything you need to build a CRUD app in a cohesive set: Data-binding, basic templating directives, form validation, routing, deep-linking, reusable components and dependency injection.
* Testability story: Unit-testing, end-to-end testing, mocks and test harnesses.
* Seed application with directory layout and test scripts as a starting point.



**The Zen of AngularJS**

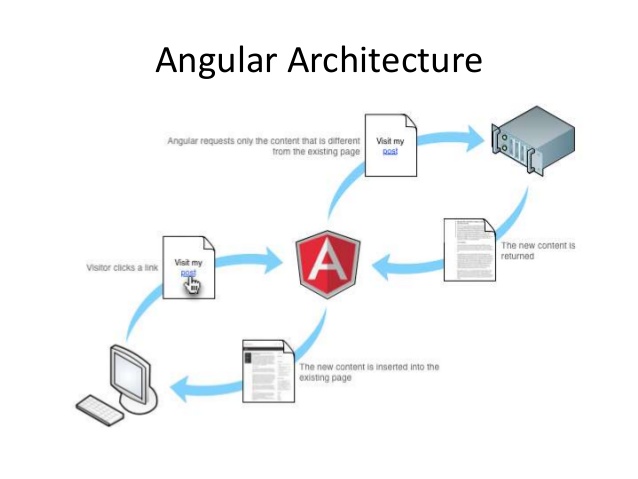
AngularJS is built around the belief that declarative code is better than imperative when it comes to building UIs and wiring software components together, while imperative code is excellent for expressing business logic.

* It is a very good idea to decouple DOM manipulation from app logic. This dramatically improves the testability of the code.
* It is a really, *really* good idea to regard app testing as equal in importance to app writing. Testing difficulty is dramatically affected by the way the code is structured.
* It is an excellent idea to decouple the client side of an app from the server side. This allows development work to progress in parallel, and allows for reuse of both sides.
* It is very helpful indeed if the framework guides developers through the entire journey of building an app: From designing the UI, through writing the business logic, to testing.
* It is always good to make common tasks trivial and difficult tasks possible.

**AngularJS frees you from the following pains:**

* **Registering callbacks:** Registering callbacks clutters your code, making it hard to see the forest for the trees. Removing common boilerplate code such as callbacks is a good thing. It vastly reduces the amount of JavaScript coding *you* have to do, and it makes it easier to see what your application does.
* **Manipulating HTML DOM programmatically:** Manipulating HTML DOM is a cornerstone of AJAX applications, but it's cumbersome and error-prone. By declaratively describing how the UI should change as your application state changes, you are freed from low-level DOM manipulation tasks. Most applications written with AngularJS never have to programmatically manipulate the DOM, although you can if you want to.
* **Marshaling data to and from the UI:** CRUD operations make up the majority of AJAX applications' tasks. The flow of marshaling data from the server to an internal object to an HTML form, allowing users to modify the form, validating the form, displaying validation errors, returning to an internal model, and then back to the server, creates a lot of boilerplate code. AngularJS eliminates almost all of this boilerplate, leaving code that describes the overall flow of the application rather than all of the implementation details.
* **Writing tons of initialization code just to get started:** Typically you need to write a lot of plumbing just to get a basic "Hello World" AJAX app working. With AngularJS you can bootstrap your app easily using services, which are auto-injected into your application in a Guice-like dependency-injection style. This allows you to get started developing features quickly. As a bonus, you get full control over the initialization process in automated tests.

**Architecture of Angularjs**

****

**DATA FLOW DIAGRAM**

Data flow diagram is graphical tool used to describe and analyze the movement of data through 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 process may be described logically and independently of the physical components associated with the system. They are termed logical data

**Flow Diagrams**

Components of DFD’s

Data Flow: Data move in a specific direction from an origin to a destination in a form of a document letter, telephone call or virtually any medium.

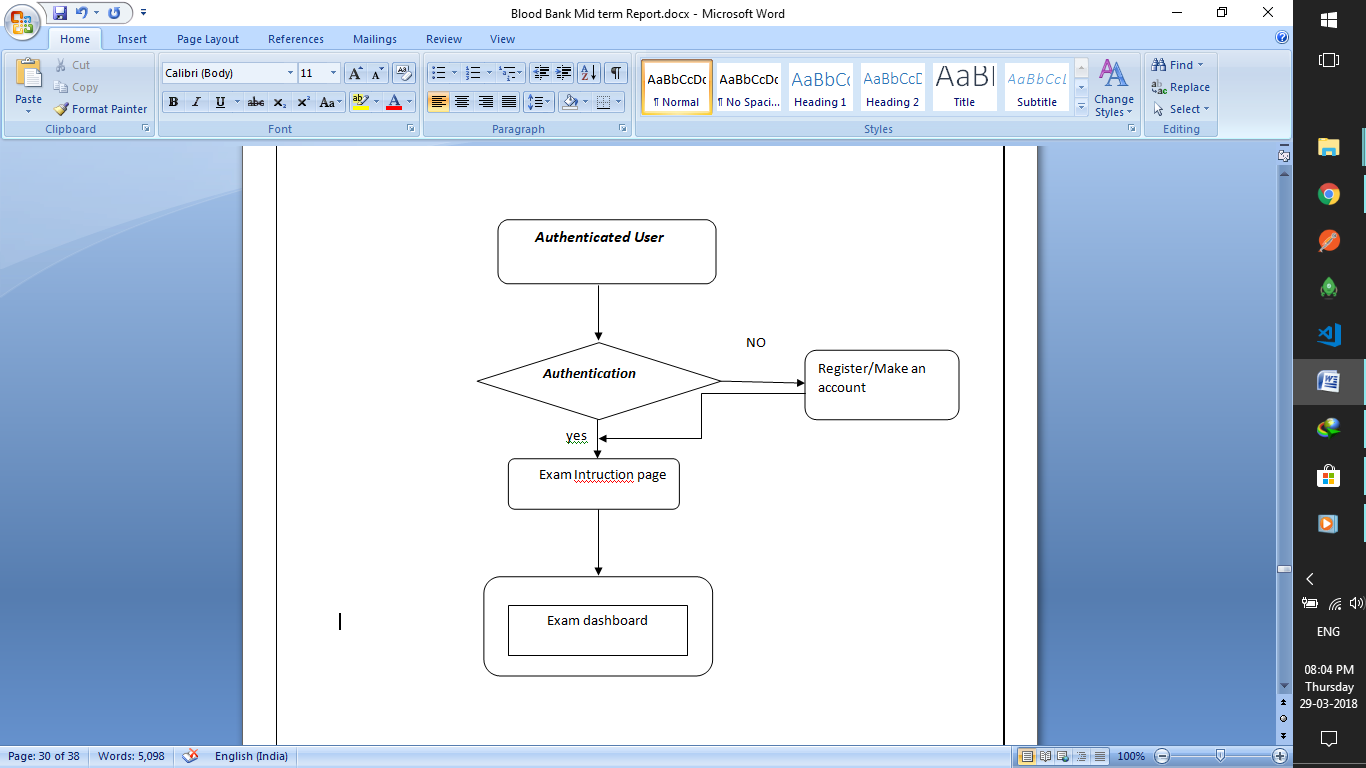
* Process: People, procedures or devices that are used to produce data. The physical component is not identified
* Source or Destination of Data: External sources or destinations of data which may be people, programs, organizations or other entities, interact with the system but are outside its boundary. The term source and sink are interchangeable with origin and destination.
* Data Store: Here data are stored or referenced by a process in the system. The data store may represent computerized or non-computerized devices.

# 

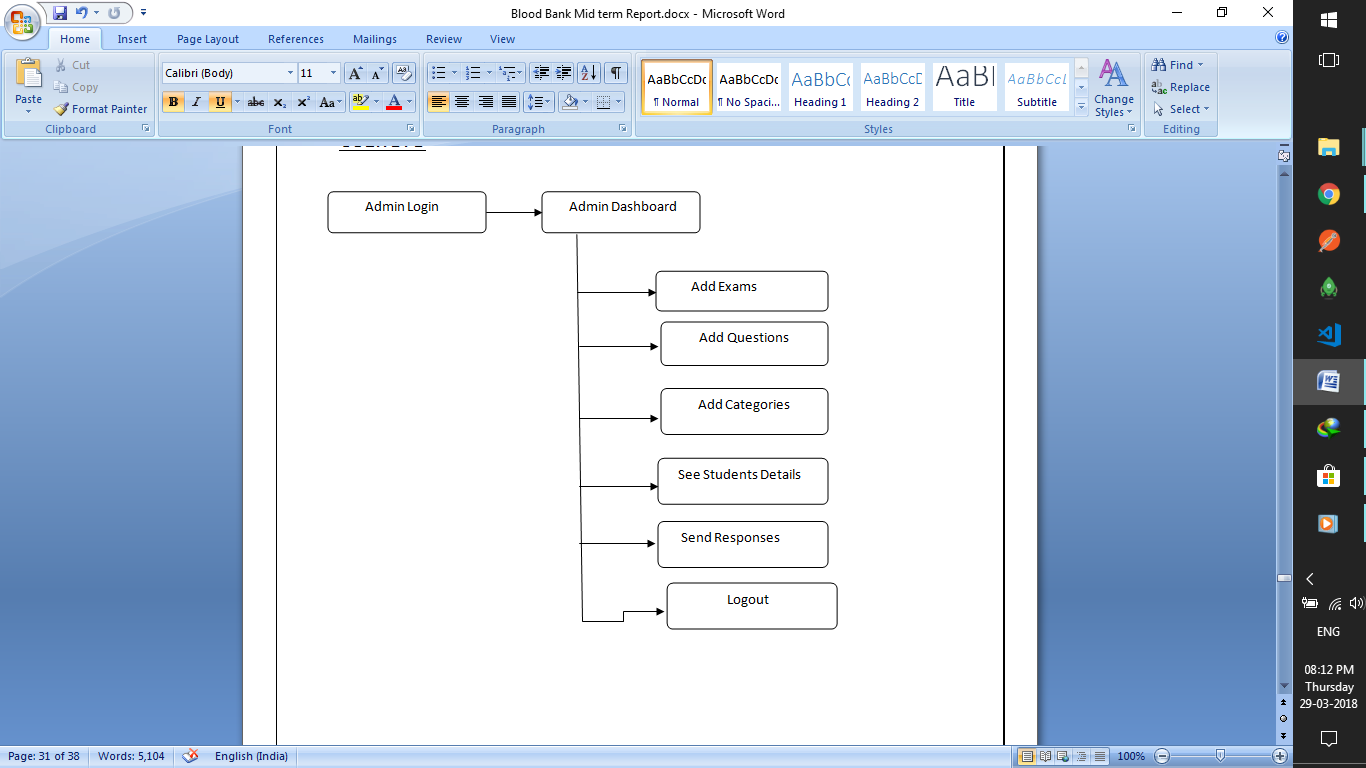
# 

# Project

**User Login DFD**

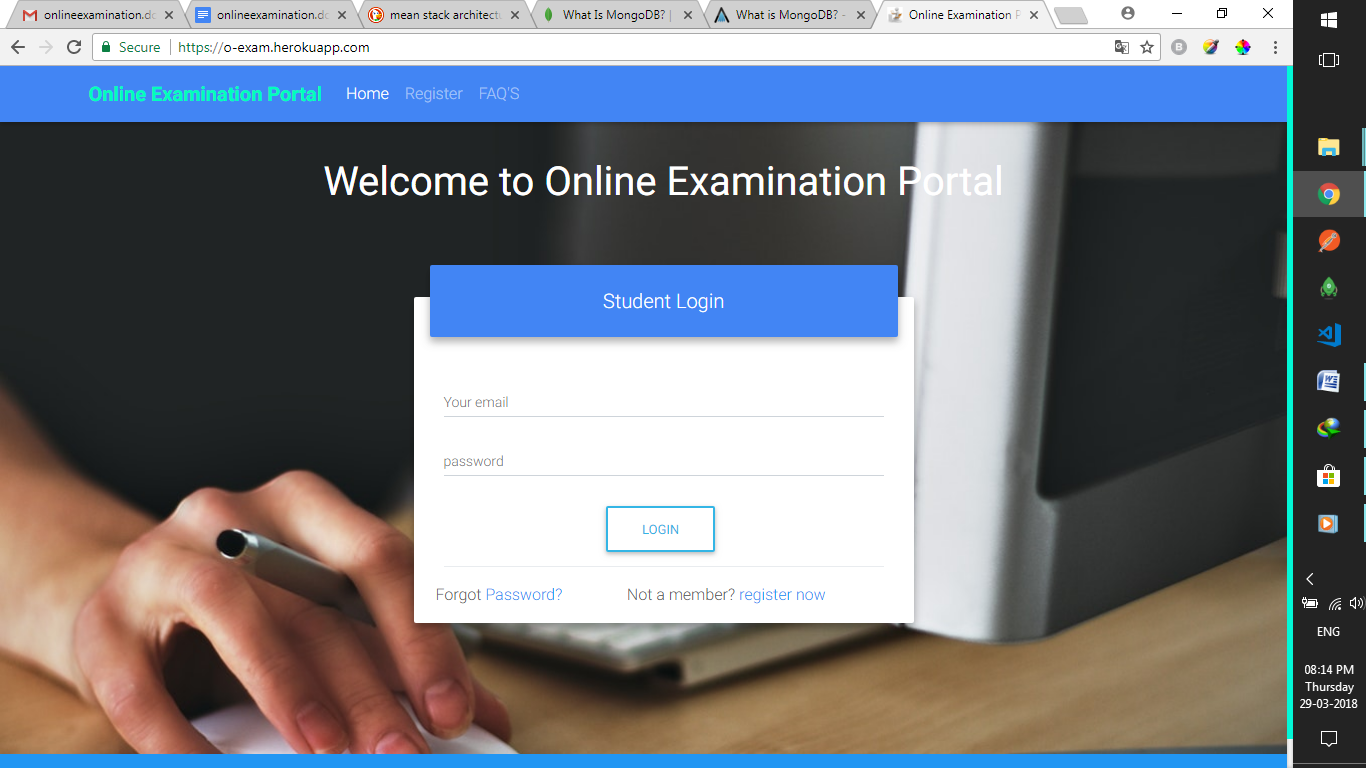


**Admin Login DFD**



Here is some of the work that we have finished till now ,it is not sufficient but we are doing our best to make this project enough interactive and user friendly.so that we can make it popular in web market.

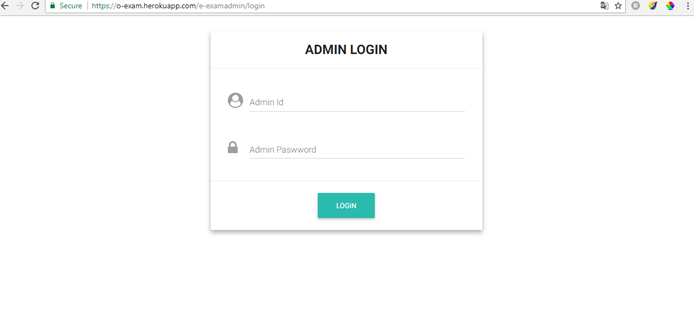
**HOME PAGE**



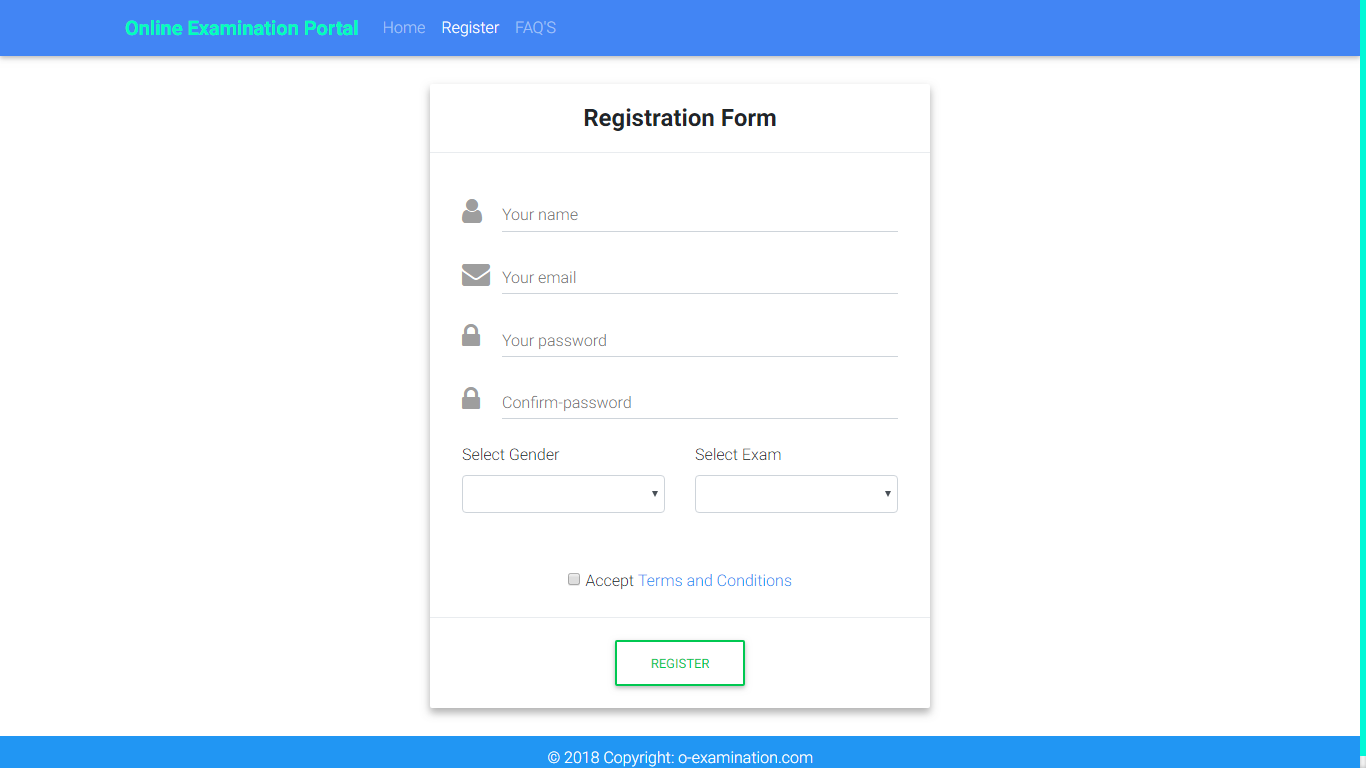
**Fig 1.1 student login screen**

**ADMIN LOGIN**

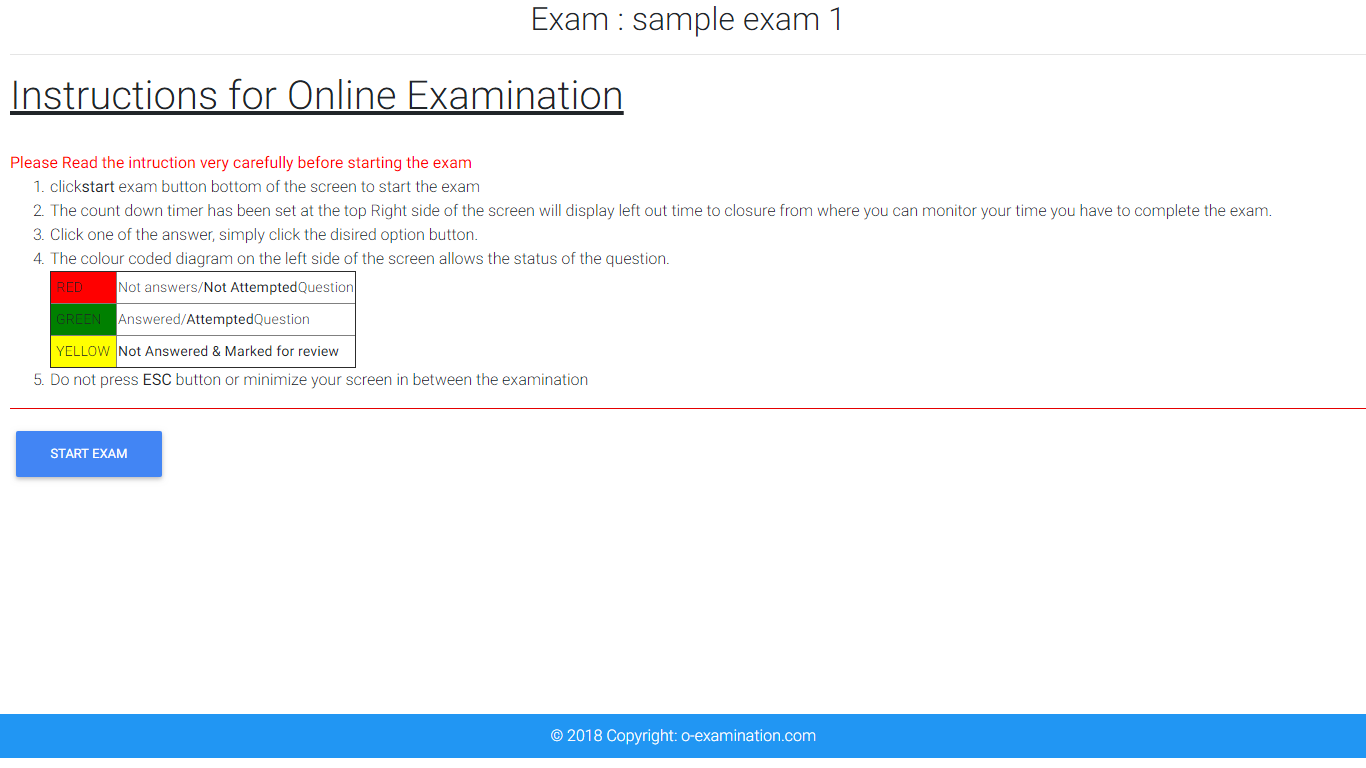
**Fig 1.2 Admin login screen**

****

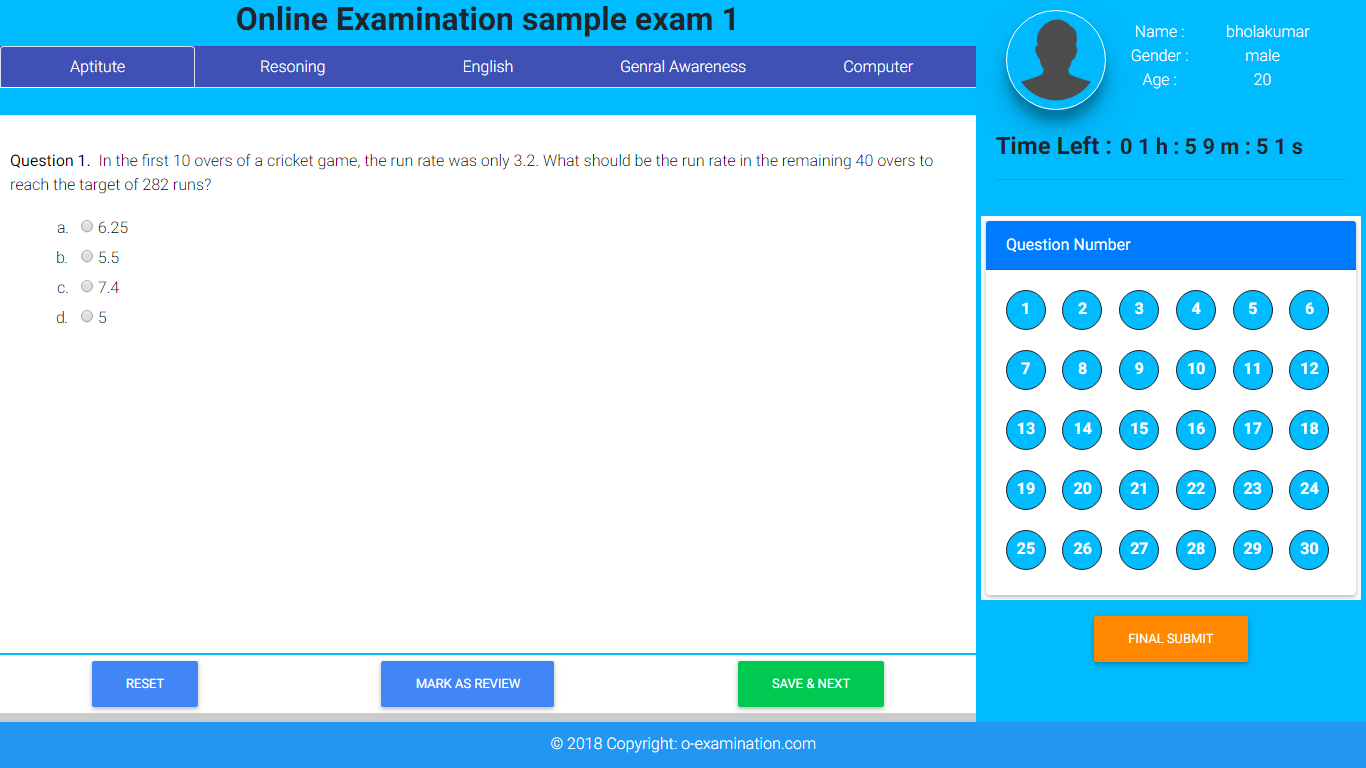
**Fig 1.3 Student registration screen**

****

**Fig 1.4 Exam Instruction page screen**

****

**Fig 1.5 ExamDashBoard screen**

****

**References**

**Websites :-**

§ <https://www.youtube.traversy> media

§ <https://www.mongodb.github.io/node-mongodb-native/2.2/tutorials/conect/>

§ <https://www.learnangular2.com>

§ <https://www.expressjs.com>

§ <https://www.youtube/traversy> media

§ <https://www.w3schools.com/TheNetNinja>/

§ <https://www.w3schools.com/nodejs/>

§ <https://www.w3schools.com/mongodb/>

§ <https://www.w3schools.com/angularjs/>