**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

**www.msrit.edu**

**SOFTWARE REQUIREMENT SPECIFICATIONS**

**AND**

**GANTT CHART**

**Project Title: RIT Data Center**

**Team Infinite Diversions:**

**Sashank Agarwal 1MS16CS090**

**Anshul Gupta 1MS15CS093**

**Product Overview:**

Managing the data in a very large and long-established institution is a tedious job as the size of the institution keeps on extending and to maintain the data of every year in a format that is easily accessible becomes very important. RIT Data-Center records all the data regarding the Institution, various committees and departments in the Institution, achievements, and activities conducted by various departments, admission and scholarship details, faculty details etc in a structured manner that can be read, queried and updated in an easier manner.

It provides various levels of authentication and authorization to the members of the Institution to view and update the data in the database. The primary function of the system is to generate dynamic reports for the departments and the Institution in any specified formats for the given requirements whenever required. It facilitates the authority at the higher level to monitor the activities happening in various departments by selecting the fields he/she wants to view. It also provides a platform for managing up-to-date information of various activities being conducted in the Institution and allows every member of the Institution to manage and update his/her information directly in the database.

1. **Hardware Requirements:**

Our hardware requirement is restricted to just one, that is to set up a high-performance server that could take up the load for 30-40 users at a time. We would need a server with minimum this specification -

* RAM - 8GB or more
* Secondary Storage - SSD 80GB
* GPU - Not Required
* Processor - At least 4 core processor with a clock speed of 3 GHz

**Software Requirements:**

* Node JS - for Backend
* MySQL - database management
* Node JS - For extracting the data from the raw excel sheets
* EJS View Engine - For Frontend
* HTML/Bootstrap/JS - For making attractive UI

1. **Functional Requirements:**
   1. Dynamic report generation: The web application should be capable of producing dynamic reports in excel format. In order to obtain such reports, the user will provided with filters (drop-down menus, checkboxes, etc) which allow him/her to customize the report and download it in excel format.
   2. Authorization levels: The web application should also provide a secured login to each individual accessing the website. Based on the user's designation and level of authorization, they will be able to manipulate the data contained in different entities of the database. We have categorized the users based on their access rights as:
2. Faculty - will be having permission to create, modify or delete his/her individual information and can view department level and institution level information. The faculty should not be able to modify or delete information at the department and institution level.
3. Head of the department - HODs will be given two different logins, one to manage his/her personal information and the other to monitor the department level information. The HOD will also have the permission for generating dynamic reports at the department level.
4. Department Coordinators / Office Staff - This is the group of users that will be dealing with updating and managing the information at the department level. They will have authority to create, modify and delete data and also generate reports at the department level.
5. Principal - The principal will be given three different logins, one to manage his/her personal information, second to monitor the department level information and third to monitor the information at the institution level. The Principal will also have the permission for generating dynamic reports at the institution level.
6. Database Administrator - DBA will have overall access to all the data and will also be responsible for authorizing users.
   1. Extracting data from excel sheets: The project also covers the crucial part of uploading the humongous amount of data into the database appropriately. This complex task is achieved by a module present in the server, which simply takes in an excel sheet and uploads it to an appropriate table present in the database.
   2. Storing views of recently joined tables: The principle of caching is used here. Since, there is a high probability of generating reports, which were generated recently, it is a good practice to store the view of the table obtained after joining two or more tables. This decreases the computational time required by the server to join these tables again, hence the user gets his/her reports quickly.
   3. Hassle free seminar management: The web application allows for creating new seminars without any effort. This considerably reduces the time required to conduct seminars.
   4. File manager: There needs to be constant checks on the number of excel sheets that are getting stored on the server. For incorporating this mundane task, a module is required on the server side, which keeps a count of the number of sheets present on the server and deletes the files one by one if the count crosses a particular limit based upon the FIFO principle.
   5. Graph generator: A database application, which stores the information about the college, would have tons of data. We use this data, to obtain different kinds of graphs and conclusions (E.g. number of students placed v/s year, CET cut-off rank v/s year, etc).
7. **Non-Functional Requirements:**
   1. Security Requirements:

Since all the data will be transferred on the web, the system should also use an encryption and decryption mechanism to ensure that only intended user can access the data. Also, we need to ensure different levels of authorization for different members of the Institution. Some users would have access rights to all the functionality of the system whereas some users will be having restricted access rights.

* 1. Availability

We would like the Data Center to be available to the users 24/7. Because of this requirement, we are hosting the website on AWS (Amazon web services) which will provide 24-hour service in contrast to the college server that demands the user to be in the college campus to update the details.

* 1. Recoverability

We will take the backup of the database from time to time. Or write a script to take the backup weekly.

* 1. Manageability

There will be a great deal of manageability in the data center. The admin will have the power to change and update almost all the data in the database. There are 5 categories of authorization. Each category will have different privileges regarding the updating they can do in the database. The admin will be provided with the admin panel where they will have options to insert, update or delete data.

1. **Performance Requirements:**

The website will be hosted on the internet so that the faculty and other users can access the application from anywhere. Though the number of users authorized to use the website is less, we can have situations where multiple users are working simultaneously. Thus we need a powerful server that can handle multiple users at any point of time. Given a critical period of time, we can expect a maximum of 60 users at the same time.

The website should have a fast response time, given a request to update or delete data in the database. Updating shouldn’t take more than 3-5 seconds. Node.js for backend development and MySQL for database management ensures the faster processing of the requests.

We might have new committees or even new departments in our college in the future. Also, since the data that is being stored in the database keeps on expanding every year we might need more storage space and processing power to handle the requests. Because of this reason, the server needs to be scalable. We should be able to expand the server and get more resources at a given time.

1. **Database Requirements:**

Our database consists of all the data regarding admissions, committee members, scholarships, infrastructure, faculty achievements, academics, R&D details, achievements etc. Since our database has to be maintained in a structured manner for the purpose of querying and generating reports we are using MySQL database.

We have categorized the data in our database into three levels based on the different category of users who will be monitoring the data.

1. Institution-level - This consists of the data regarding sports, NSS, library, placements, scholarships and different committees such as governing body, academic council, finance committee etc.
2. Department level - The institution then branches to departments that contain the data regarding infrastructure, admissions, student achievements and publications, teaching and non-teaching staff data, professional bodies and activities conducted by department such as guest lectures, industrial visits, seminars and workshops etc.
3. Faculty level - The faculty comes under the department and all the data regarding faculty is maintained at this level. This constitutes faculty qualification details, service details, R&D information, academic details, conferences and FDPs attended and organized by the faculty and publication details that include books, book chapters and papers published in journals and conferences.
4. **Design Constraints:**

Due to the structure of the database and the requirements being unstable and frequently changing we come across few design constraints for this project -

* We are using SQL for our database, to keep our data structured and have a fast querying and data access. But this fact leads us to restrict our database structure to a minimal change.
* Making any changes in the DB structure (due to any change in the requirements) forces us to change almost the entire query related to that schema, which leads to a lot of redundant work.
* On the other hand, using a NoSQL Db structure could also have a lot of problems, as it is really difficult to set up the integrity in NoSQL.
* NoSQL could have given us the flexibility to change the DB structure more often but that leads to the most of the cases where exceptions are raised due to non-consistency of the database.
* In all this dilemma of choosing a SQL over NoSQL, we were followed by many arguments and had many discussions on that. But in the end, considering the final requirement that we need for this project, we concluded that SQL DB could be the best and it would suffice most of our use cases.
* This leads to our design constraint of changing the DB Structure the least we can and hence save a lot of hours in development.