**CHAPTER 1**

**INTRODUCTION**

This document is a Software Requirement Specification (SRS) and it presents the main Objective of this project being  to design  a todo list application.This is the initial draft for the SRS and it will be used for the extensions. Todo list application will allow a user to create new

tasks, assign them a title and due date, and choose a project for that task to belong to. They will

need to use a text based user interface via the command-line.Once they are using the application, the user should be able to also edit,

mark as done or remove tasks. They can also quit and save the current task list to file, and then

restart the application with the former state restored.

 Also, This document is prepared by following IEEE conventions for software requirement specification.

**PURPOSE**

The purpose of this document is to build  a todo list application. The application will allow a user to create new tasks, assign them a title , due date, choose a project for that task to belong to.  edit task,

mark as done or remove tasks, quit and save the current task list to file, and finally the user will be able to

restart the application with the former state restored.

**PROJECT SCOPE**

Presently the application  web-based and also provides a clean and user-friendly interface to the users.

**CHAPTER 2**

**LITERATURE SURVEY**

The todo list application is a new attempt to speed up the process of managing time of engineers and others working with computers and varied operating systems more often and also those  in  educational institute.s The existing systems are time-consuming and there are many difficulties faced by these groups of persons to get information about each individual and work schedules. This software provides a solution to these problems. It provides an interactive user interface and helps users in an organisation to get information immediately at that instant of time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | |  | | --- | |  | | | | |

**Chapter 3**

**REQUIREMENTS**

**3.1 Functional Requirements**

● Model a task with a task title, due date, status and project

* ● Display a collection of tasks, sorted by date or filtered by project
* ● Support the ability to add, edit, mark as done, or remove tasks
* ● Support a text-based user interface
* ● Load and save task list to file

**3.2 Non- Functional Requirements**

**Safety Requirements**

The application should not  provide catastrophic failure, such as a disk crash, and operating system incomnpartibility.

**Also the application will provide other  Software  Quality  Attributes as USABILITY satisfactions**

.

**3.3  Hardware Requirements**

* Computers

**3.4 Software Requirements**

* ● Mac, Windows , Linux operating systems
* ● Internet
* **Model Implementation**
* **Agile Model**

**Diagram:**

**Chapter 4**

**System Architecture**

**4.1 Clint-Server Architecture**

The client-server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients.

Client-server software architecture is versatile and flexible in today’s fast-changing IT landscape. It is modular in structure and relies on messaging services for communication between components. They were designed to improve flexibility, usability, scalability, and interoperability

Diagram:

**Chapter 5**

**Design and Implementation**

**5.1  Product  Features**

The major features of e-administration of computer lab system are as listed below.

* ● Model a task with a task title, due date, status and project
* ● Display a collection of tasks, sorted by date or filtered by project
* ● Support the ability to add, edit, mark as done, or remove tasks
* ● Support a text-based user interface
* ● Load and save task list to file.

       By achieving above features, responsiveness and hence outcomes of application will greatly increases.

**class diagram design**

**Use case diagram**

**Sequence diagram**

**Activity diagram**

**Entity diagram**

**Chapter 6**

**Snapshots**

**Chapter 7**

**TESTING AND RESULT**

The reason behind testing was to find errors. Every program or software has errors in it, against the common view that there are no errors in it if the program or software is working. Executing the programs with the intention of finding the errors in it is therefore testing; hence a successful test is one which finds errors. Testing is an activity, however, it is restricted to being performed after the development phase is complete, but is carried parallel with all stages of system development, starting with requirement specification.

Test cases were devised with a purpose in mind. A test case is a set of the data that a system will process as normal input. The software units developed in the system are modules and routines that are assembled and integrated to perform the required function of the system. Test results once gathered and evaluated, provide a qualitative indication of the software quality and reliability and serve as the basis for design modification if required. In this phase, testing is done at different levels. Actually testing phase of the implementations works accurately and efficiently before live operation commences.

**7.1 Unit Testing**

The unit testing was done after the coding phase. The purpose of the unit testing was to locate errors in the current module, independent of the other modules. Some changes in the coding were done during the testing phase. Finally, all the modules were individually tested following bottom to top approach, starting with smallest and lowest modules and then testing one at a time.

**7. 6 Acceptance  Testing**

This is the final stage in the testing process before the system is accepted for operational use. Any requirement problem or requirement definition problem revealed from acceptance testing are considered and made error free.

**Chapter  8**

**TIME FRAME**

**Chapter 9**

**CONCLUSION**

.