SOFTWARE ENGINEERING AND PROJECT MANGANEMENT LAB

(Code 18CSC207J)

B.Tech (CSE) – 2nd year/4th Semester

Name: MOKSH VIJ

Registration no: RA1811003030124

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY,

DELHI NCR CAMPUS, MODINAGAR

SIKRI KALAN, DELHI MEERUT ROAD, DIST. – GHAZIABAD - 201204

www.srmimt.net

Even Semester (2019-2020)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Registration no.RA1811003030124

Certified to be the bonafide record of work done by Moksh Vij of 4th semester 2nd year B.TECH degree course in SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, NCR Campus of Department of Computer Science & Engineering, in Software Engineering and Project Management Lab, during the academic year 2019-2020.

Lab in charge **Head of the Department (CSE)**

Submitted for university examination held on __/____/___ at SRM IST, NCR Campus.

Internal Examiner-I **Internal Examiner-II**

Index

S.No.	Name of Experiment	Date of Experiment
1	Identify the Software Project, Create Business Case, Arrive at a Problem Statement	6/1/2020
2	Stakeholder and User Description, Identify the appropriate Process Model, Comparative study with Agile Model	11/1/2020
3	Identify the Requirements, System Requirements, Functional Requirements, Non-Functional Requirements	13/1/2020
4	Prepare Project Plan based on scope, Find Job roles and responsibilities, Calculate Project effort based on resources	20/1/2020
5	Prepare the Work, Breakdown Structure based on timelines, Risk Identification and Plan	3/2/2020
6	Design a System Architecture, Use Case Diagram, ER Diagram (Database), DFD Diagram (process) (Upto Level 1), Class Diagram (Applied For OOPS based Project), Collaboration Diagram (Applied For OOPS based Project) (Software – Rational Rose)	17/2/2020
7	State and Sequence Diagram, Deployment Diagram, Sample Frontend Design (UI/UX)	24/2/2020
8	Implementing code for the desired project	16/3/2020
9		
10	To write about coding standards and about agile	17/4/2020
11	Development. Regarding coding standards of the project.	

EXPERIMENT 1

AIM: To design and implement a text editor in java using

Netbeans IDE

REQUIREMENTS:

Hardware:

- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Programming Language:

- JAVA

PROCEDURE:

Project Title: NOTEPAD

Business Case:

- The text editor is designed and implemented in Netbeans IDE.
- It contains buttons for saving and opening text files.
- The files would be saved in a folder in the system.
- It also contains button which is when clicked tells the information about the notepad.

Problem Statement:

Text Editor is software that edits plain text and comprises of Java Swings and AWT. This project has all the frames prepared in Swing. It can be used to create new files with different file extensions like .txt, .java etc.

RESULT: The text editor is designed and implemented using java.

EXPERIMENT 2

AIM: Identify the stakeholders, create a user description, identify an appropriate process model for your project and compare it with agile methodology.

REQUIREMENTS:

Hardware:

- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Programming Language:

- JAVA

Software used for experiment:

- MS WORD

PROCEDURE:

Stakeholders:

The stake holders for the project are:

1) The development team: the team members have to make sure that they deliver the desirable project within the time limit and the quoted budget.
2) Product Owner: the product owner is accountable for delivering the product to the users, publicity etc. they are the investors and have their financial stake.
3) User: the team and the owner has to make sure that the users are satisfied with the product.

User Description:

A text editor, in general, is a type of program used to edit plain text files. Text editors are often provided with operating systems or systems of software development packages, and can be used to configure files and programming Language source code. Various text editors are available in the

market, e.g., vi editor and Emacs editor in UNIX, Simple Text and TextEdit editors in Macintosh, Notepad in Microsoft Windows, and so on.

A text editor can be Graphical User Interface (GUI) based or Character User Interface (CUI) based. A GUI based application is always preferred over a CUI based one since we have to click with the help of mouse on the icons of readymade commands instead of typing them.

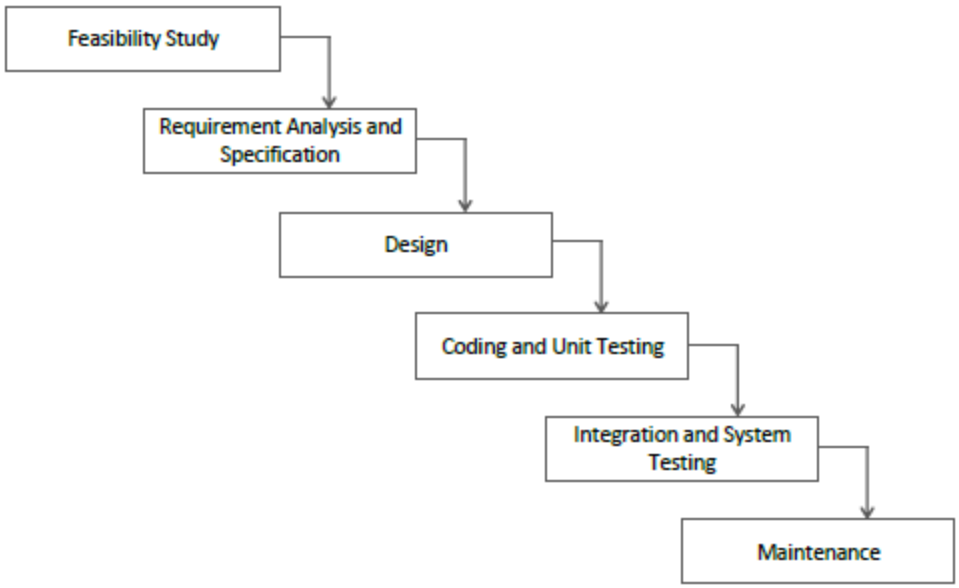
Notepad is the most common text editor used by almost all computer literate people. It is used to create simple documents. It saves files in plain text format (ANCI text) and supports only a few formatting options.

It is used to view or edit text files (files having the extension .txt). Users find Notepad an idea tool for creating Web pages.

Process Model:

Classical Waterfall Model:

Classical waterfall model is the basic **software development life cycle** model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.
Classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other.



Advantages:

Classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered as the basis for other software development life cycle models. Below are some of the major advantages of this SDLC model:

- This model is very simple and is easy to understand.
- Phases in this model are processed one at a time.
- Each stage in the model is clearly defined.
- This model has very clear and well undestood milestones.
- Process, actions and results are very well documented.
- Reinforces good habits: define-before- design, design-before-code.
- This model works well for smaller projects and projects where requirements are well understood.

Comparison with Agile Methodology:

Waterfall basically is a sequential model where software development is segregated into a sequence of pre -defined phases – including feasibility, planning, design, build, test, production, and support. On the other hand, Agile development methodology follows a linear sequential approach while providing flexibility for changing project requirements, as they occur.

Here are the differences between Agile and Waterfall Methodology:

1. The software development process is divided into different phases in the Waterfall model while Agile methodology segregates the project development lifecycle into sprints
2. Waterfall is a structured software development methodology, and often times can be quite rigid, whereas the Agile methodology is known for its flexibility
3. According to the Waterfall model, software development is to be completed as one single project, which is then divided into different phases, with each phase appearing only once during the SDLC. However, the Agile methodology can be considered as a collection of many different projects, which are nothing but the iterations of the different phases focusing on improving the overall software quality with feedbacks from users or the QA team
4. If you want to use the Waterfall model for software development, then you have to be clear with all the development requirements beforehand as there is no scope of changing the requirements once the project development starts. The Agile methodology, on the other hand, is quite flexible, and allows for changes to be made in the project development requirements even after the initial planning has been completed
5. All the project development phases such as designing, development, testing, etc. are completed once in the Waterfall model while as part of the Agile methodology, they follow an iterative development approach. As a result, planning, development, prototyping and other software development phases can appear more than once during the entire SDLC
6. One of the major differences between Agile and Waterfall development methodology is their individual approach towards quality and testing. In the Waterfall model, the “Testing” phase comes after the “Build” phase, but, in the Agile methodology, testing is typically performed concurrently with programming or at least in the same iteration as programming
7. While Waterfall methodology is an internal process and does not require the participation of customers, the Agile software development approach focuses on customer satisfaction and thus, involves the participation of customers throughout the development phase
8. The Waterfall model can be regarded as a stringently sequential process, however, the Agile methodology is a highly collaborative software development process, thereby leading to better team input and faster problem solving

9. The Waterfall model is best suited for projects which have clearly defined requirements and in which change is not expected at all, while Agile development supports a process in which the requirements are expected to change and evolve. Thus, if you are planning to develop a software that would require frequent overhauls and has to keep up with the technology landscape and customer requirements, Agile is the best approach to follow
10. The Waterfall model exhibits a project mindset and lays its focus strictly on the completion of project development, while Agile introduces a product mindset that focuses on ensuring that the developed product satisfies its end customers, and changes itself as the requisites of customers change.

Result: the experiment is executed successfully.

EXPERIMENT 3

Aim: To gather the user requirements, system requirements and the functionalities of the project and hence design the SRS document for the project.

Procedure:

User Requirement: The user requirements for a notepad would be:

a. Change the font, size and the style of text.
b. Enabling the cut copy and paste features.
c. Database of the saved files.
d. User description.

System Requirement:

Hardware:

- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Functional Requirement: The software provides an interface for editing, viewing and creating text files. The project also enables user to change the font, size and style of text. It tells the user about the no words and lines written in the particular text and we can also cut copy and paste text from other files.

Non-Functional Requirement:

The software is highly reliable and can work on different systems if the system fulfills the basic requirements for the software. The software provides a user friendly interface and is easy to use. The software can be utilized in many different ways other than creating text files. It can be used to create source code files and after further development can be used for playback.

SRS [Software requirement specification]:

Introduction: A text editor, in general, is a type of program used to edit plain text files. Text editors are often provided with operating systems or systems of software development packages, and can be used to configure files and programming

Language source code. Various text editors are available in the market, e.g., vi editor and Emacs editor in UNIX, Simple Text and TextEdit editors in Macintosh, Notepad in Microsoft Windows, and so on.

A text editor can be Graphical User Interface (GUI) based or Character User Interface (CUI) based. A GUI based application is always preferred over a CUI based one since we have to click with the help of mouse on the icons of readymade commands instead of typing them.

Notepad is the most common text editor used by almost all computer literate people. It is used to create simple documents. It saves files in plain text format (ANCI text) and supports only a few formatting options.

It is used to view or edit text files (files having the extension .txt). Users find Notepad an idea tool for creating Web pages.

Overview:

1) **Customers:** A text editor is program that allows you to open, view, and edit plain text files. Unlike word processors, text editors do not add formatting to text, instead focusing on editing functions for plain text. Text editors are used by a wide variety of people, for a wide variety of purposes.
2) **Platform:** Our goal is to develop a notepad using Java on Netbeans IDE. The files will be stored in .txt format in the database of the source file. The notepad is developed using swing and AWT.
3) **Deliverables:** the deliverables from the team would be:
    a. Working software within the required time.
    b. The document stating all the functionalities and the user description.
    c. Maintenance for the project for a period of six months.
4) **Functionality:** the text editor can be used to edit, open and create plain text files. The files created will generally be of .txt format and also can be used to create source file with extensions like .java, .html etc

Goals and Scope: At the highest level, Notepad's scope covers the creation, storage, sharing, retrieval and collaborative editing. The scope also covers the creation of user accounts, user logons, and project permission management.

The future Goals of the project would be to use it for playback for musical source project.

Scheduling and Milestones:

1. First we would conduct the feasibility study to check whether to take this project or not. After the feasibility study we conduct requirement analysis. This will take around 3 days.
2. Next we start the designing phase of the project. This phase will take around 7 days.
3. Then we go in the development phase where the coding is done and the project is developed and tested to make sure it fulfills all the requirements. This phase takes around 30 days.
4. Then we conduct the system and integration testing. This takes around 7 days.
5. Then the project is delivered and goes under maintenance for a period of 180 days.

Result: We have gathered the user requirements, system requirements and the functionalities of the project and the SRS document is prepared.

EXPERIMENT 4

AIM: Prepare project plan based on the scope find job, roles and responsibilities and calculate project effort based on the resources.

REQUIREMENTS:

Hardware:

- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Programming Language:

- JAVA

Software used for experiment:

- MS WORD

Procedure:

Project Plan:

1. First we would conduct the feasibility study to check whether to take this project or not. After the feasibility study we conduct requirement analysis. This will take around 20 days.
2. Next we start the designing phase of the project. This phase will take around 20 days.
3. Then we go in the development phase where the coding is done and the project is developed and tested to make sure it fulfills all the requirements. This phase takes around 180 days.
4. Then we conduct the system and integration testing. This takes around 40 days.
5. Then the project is delivered and goes under maintenance for a period of 180 days.

Jobs, Roles and Responsibilities:

The team compromises of six members. Every team member has a very specific role in this project. The individual roles for each team member are:

1) Member1: The first member of the team is mainly involved in the feasibility study and requirement analysis.
2) Member2: The role of the second member is designing the project and choosing the right model for the project development.
3) Member3, Member4 and Member 5: These members are involved in the coding and the testing phase and also prepare the documentation of the project.
4) Member6: this member along with the other members takes care of the maintenance phase of the project.

   Here the tasks are not specific and other team members can also help them in completing their tasks.

Estimate Project Effort:

1. **Determine how accurate your estimate needs to be.** Typically, the more accurate the estimate, the more detail is needed, and the more time that is needed. If you are asked for a rough order of magnitude (ROM) estimate (-25% - +75%), you might be able to complete the work quickly, at a high-level, and with a minimum amount of detail. On the other hand, if you must provide an accurate estimate within 10%, you might need to spend quite a bit more time and understand the work at a low level of detail.
2. **Create the initial estimate of effort hours for each activity and for the entire project.** There are many techniques you can use to estimate effort including task decomposition (Work Breakdown Structure), expert opinion, analogy, Pert, etc.
3. **Add specialist resource hours.** Make sure you include hours for part-time and specialty resources. For instance, this could include freelance people, training specialists, procurement, legal, administrative, etc.
4. **Consider rework (optional).** In a perfect world, all project deliverables would be correct the first time. On real projects, that usually is not the case. Workplans that do not consider rework can easily end up underestimating the total effort involved with completing deliverables.
5. **Add project management time.** This is the effort required to successfully and proactively manage a project. In general, add 15% of the effort hours for project management. For instance, if a project estimate is 12,000 hours (7 - 8 people), a full-time project manager (1,800 hours) is needed. If the project estimate is 1,000 hours, the project management time would be 150 hours.
6. **Add contingency hours.** Contingency is used to reflect the uncertainty or risk associated with the estimate. If you're asked to estimate work that is not well defined, you may add 50%, 75%, or more to reflect the uncertainty. If you have done this project many times before, perhaps your contingency would be very small -- perhaps 5%.
7. **Calculate the total effort by adding up all the detailed work components.**
8. **Review and adjust as necessary.** Sometimes when you add up all the components, the estimate seems obviously high or low. If your estimate doesn't look right, go back and make adjustments to your estimating assumptions to better reflect reality. I call this being able to take some initial pushback from your manager and sponsor. If your sponsor thinks the estimate is too high, and you don't feel comfortable to defend it, you have more work to do on the estimate. Make sure it seems reasonable to you and that you are prepared to defend it.
9. **Document all assumptions.** You will never know all the details of a project for certain. Therefore, it is important to document all the assumptions you are making along with the estimate

Result: The project plan is prepared and the estimate project effort is calculated.

EXPERIMENT 5

Aim: Prepare work breakdown structure based on timeliness, risk identification and plan.

Requirement:

Hardware:

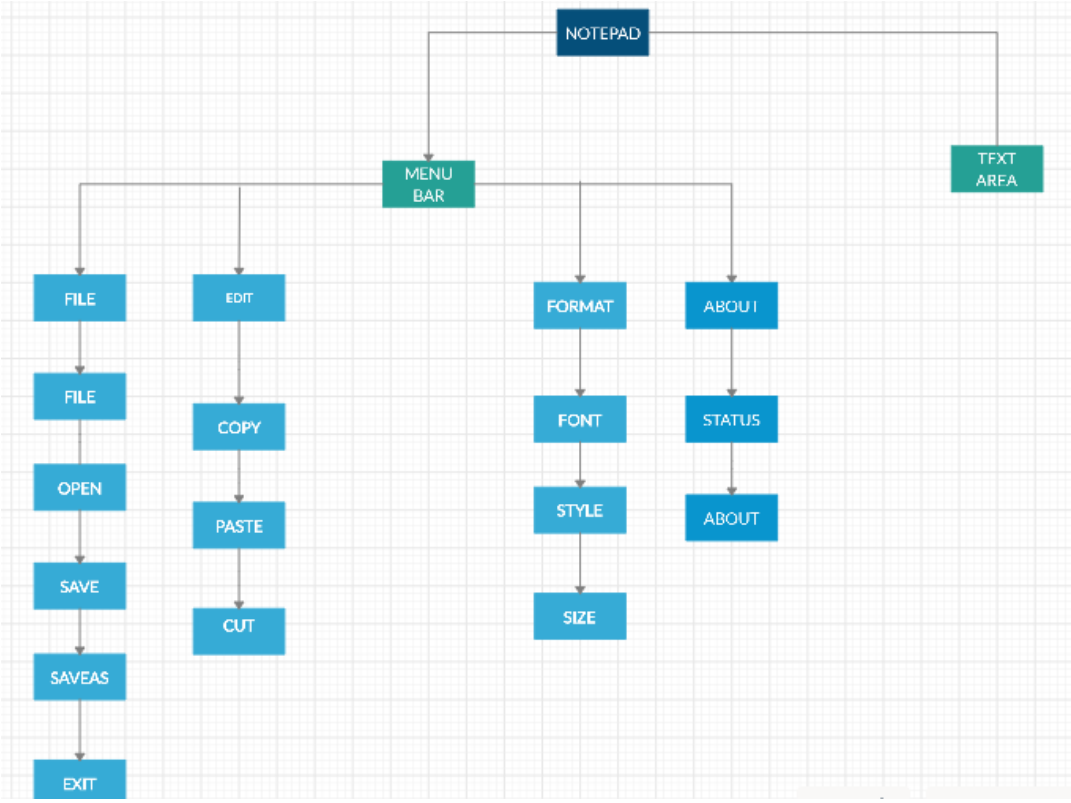
- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Procedure:

- **Work Breakdown Structure:**



RESULT: work breakdown structure based on timeliness, risk identification and plan is prepared.

EXPERIMENT 6

AIM: Creating use case, ER, Data flow Diagram and Class Diagram for the Software Project.

REQUIREMENTS:

Hardware:

- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Programming Language:

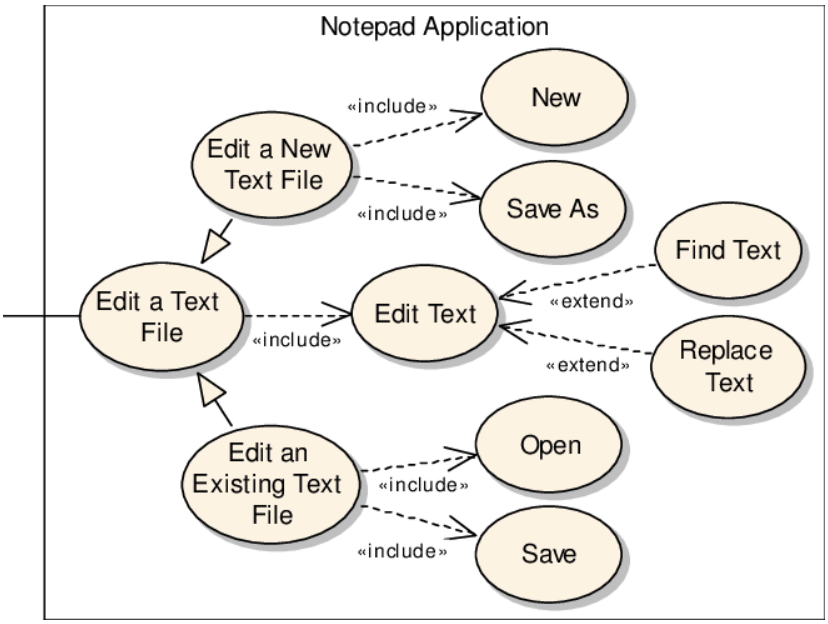
- JAVA

Software used for experiment:

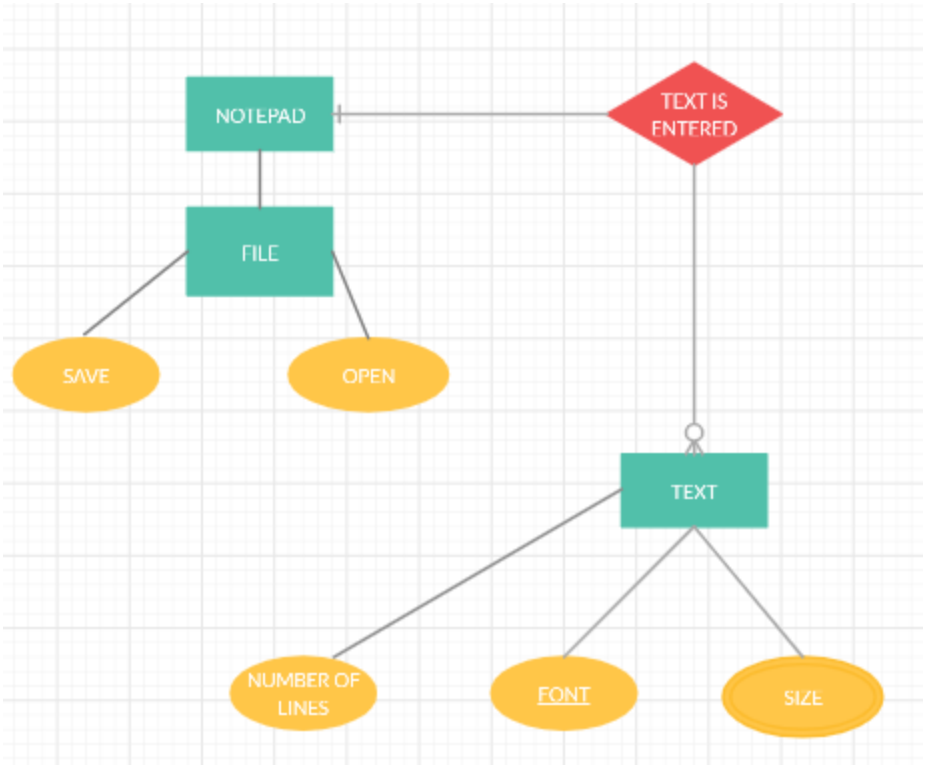
- MS WORD

Procedure:

1. USE CASE DIAGRAM:



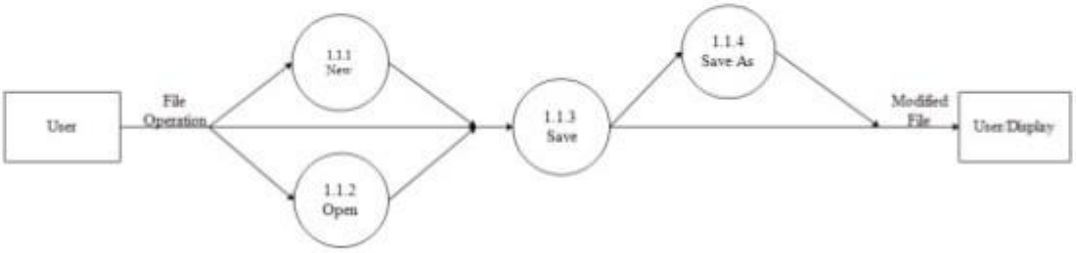
2. ER DIAGRAM



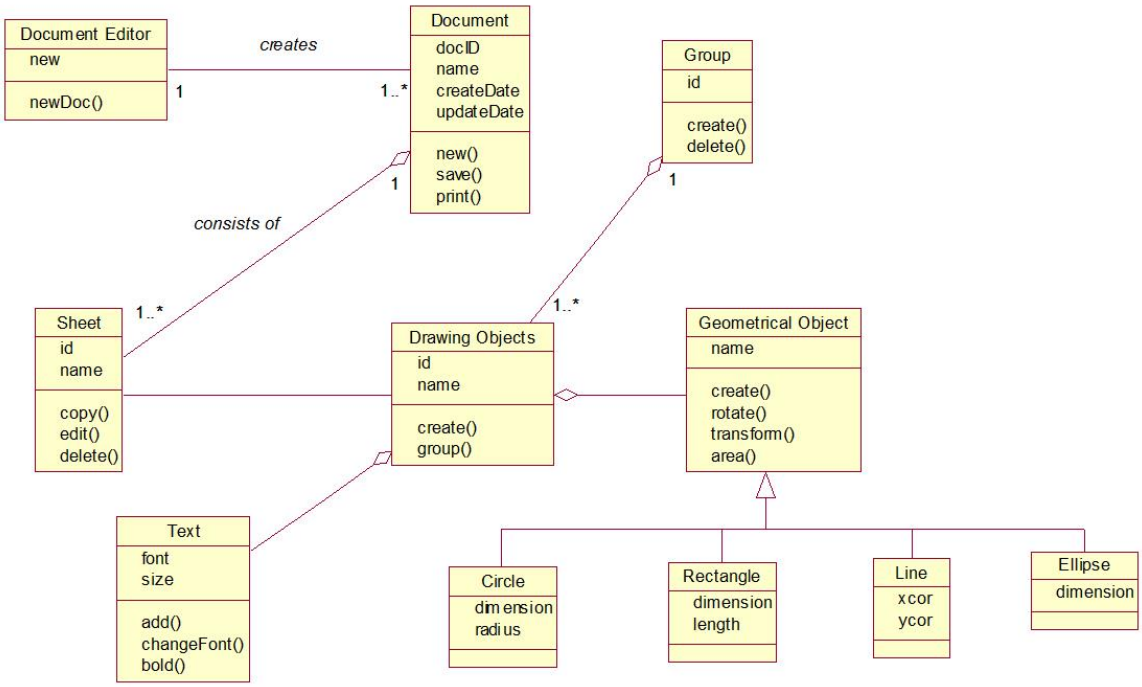
3. DATA FLOW DIAGRAM:

Level 0

Level 1



4. CLASS DIAGRAM:



Some of the features in the class diagram will be added in the upcoming updates.

RESULT: Use case, ER, Data flow Diagram and Class Diagram for the Software Project are created.

EXPERIMENT 7

AIM: Creating state machine, sequence and deployment diagram and designing the frontend of the project.

REQUIREMENTS:

Hardware:

- RAM: 2GB OR MORE
- ROM: 500GB OR MORE
- Intel i3or more

Software:

- JDK
- Netbeans IDE

Programming Language:

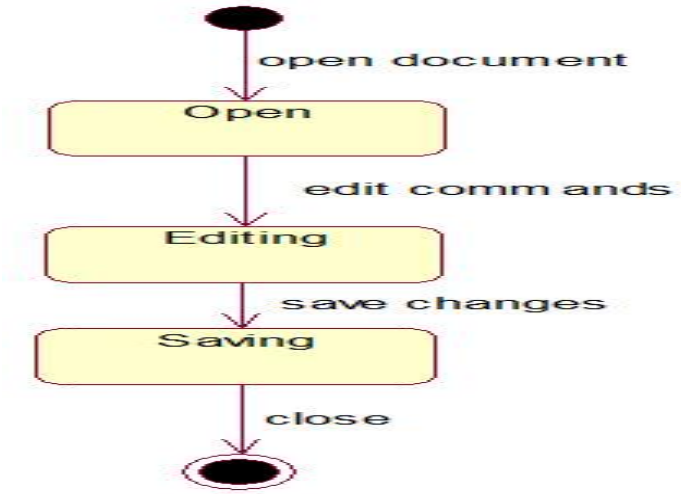
- JAVA

Software used for experiment:

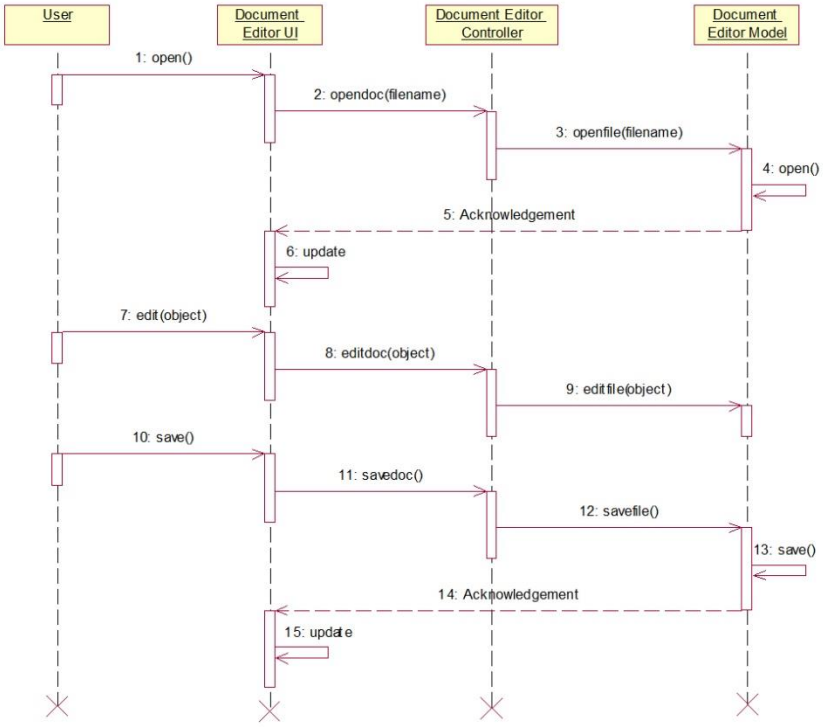
- MS WORD
- UML diagrams
- Lucid chart
- PDF to word converter

- ## State Machine Diagram:

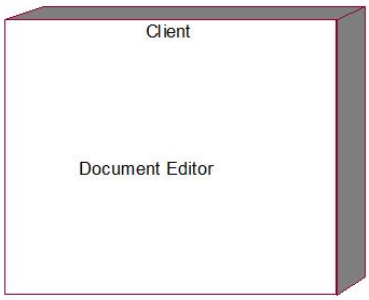
  A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.



- **Sequence Diagram:**
  A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

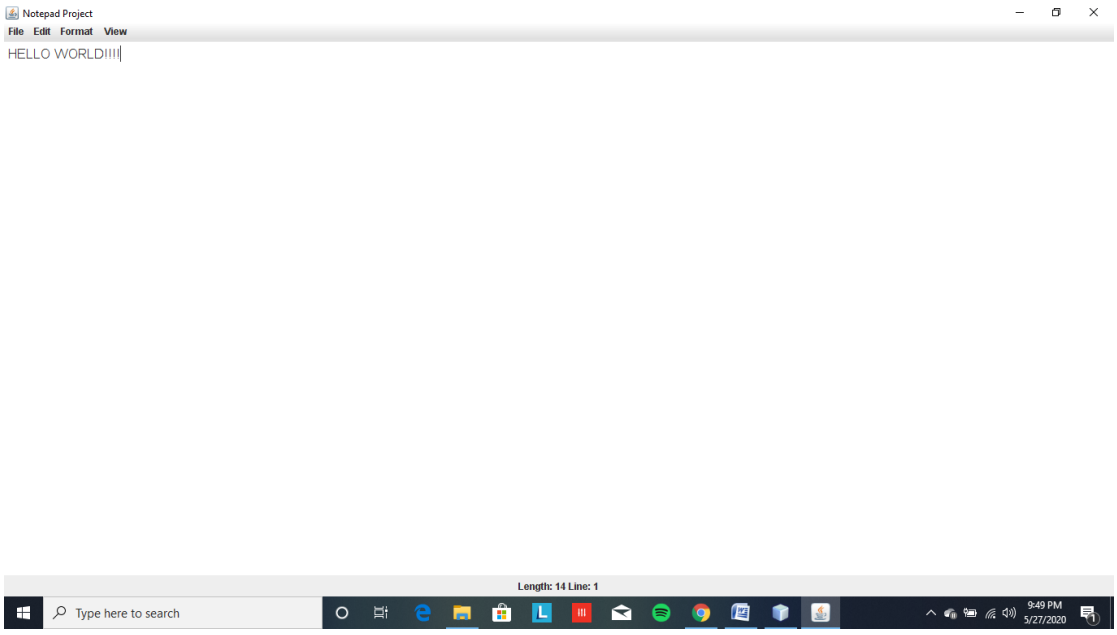


- **Deployment Diagram:**
  A deployment diagram in the UML models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components exist, what software components run on each node, and how the different pieces are connected.



Designing the Frontend:

The frontend for the software project is:



RESULT: state machine, sequence and deployment diagram and designing the frontend of the project has been created.

EXPERIMENT 8 AND 9

AIM: Implementing code for the desired project.

REQUIREMENTS:

- An IDE
- Knowledge of programming language
- Skills on particular coding language
- System
- Internet

Introduction:

- Coding for the desired project is the most time consuming part in the SDLC or Software Development Life Cycle.

- Here we are going to code for our project i.e (Global Messenger Application Software).

Software Code:

The Language used is JAVA.

The software code is:

```
import java.io.*;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;



public class notepad extends KeyAdapter  implements ActionListener, KeyListener

{


        static int active =0;

        static int fsize=17;
```

```
        JFrame frame1;
        JMenuBar npMenuBar;
        JMenu file, edit, format, view;
        JMenuItem newdoc, opendoc, exit, savedoc, saveasdoc, copydoc, pastedoc, remdoc,  fontfamily, fontstyle, fontsize, status,about;
        JTextArea maintext;
        JTextField title;
        Font font1;
        JPanel bottom;
        JLabel details, pastecopydoc;
        JList familylist, stylelist, sizelist;
        JScrollPane sb;

        String familyvalue[]={"Agency FB","Antiqua","Architect","Arial","Calibri","Comic Sans","Courier","Cursive","Impact","Serif"};
        String sizevalue[]={"5","10","15","20","25","30","35","40"};
        int [] stylevalue={ Font.PLAIN, Font.BOLD, Font.ITALIC };
        String [] stylevalues={ "PLAIN", "BOLD", "ITALIC" };
        String ffamily, fsizestr, fstylestr;
        int fstyle;
        int cl;
        int linecount;
        String tle ;
        String topicstitle = "";
        JScrollPane sp;

notepad(){

        frame1 = new JFrame("Notepad Project");

        font1=new Font("Arial",Font.PLAIN,17);

        bottom = new JPanel();
        details = new JLabel();
        pastecopydoc = new JLabel();
```

```
familylist = new JList(familyvalue);
stylelist = new JList(stylevalues);
sizelist = new JList(sizevalue);


familylist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
sizelist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
stylelist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

bottom.add(details);

maintext = new JTextArea();

sp=new JScrollPane(maintext);
title = new JTextField(100);

sb = new JScrollPane(maintext);

maintext.setMargin( new Insets(5,5,5,5) );

maintext.setFont(font1);
frame1.add(maintext);

npMenuBar = new JMenuBar();

file = new JMenu("File");
edit = new JMenu("Edit");
format = new JMenu("Format");
view = new JMenu("View");

newdoc = new JMenuItem("New");
opendoc = new JMenuItem("Open");
savedoc = new JMenuItem("Save");
saveasdoc = new JMenuItem("Saveas");
exit = new JMenuItem("Exit");
```

```
        copydoc = new JMenuItem("Copy");
        remdoc = new JMenuItem("Cut");
        pastedoc = new JMenuItem("Paste");

        fontfamily = new JMenuItem("Font");
        fontstyle = new JMenuItem("Style");
        fontsize = new JMenuItem("Size");
        status = new JMenuItem("Status");
about = new JMenuItem("About");

        file.add(newdoc);
        file.add(opendoc);
        file.add(savedoc);
        file.add(saveasdoc);
        file.add(exit);

        edit.add(copydoc);
        edit.add(pastedoc);
        edit.add(remdoc);

        format.add(fontfamily);
        format.add(fontstyle);
        format.add(fontsize);

        view.add(status);
view.add(about);

        npMenuBar.add(file);
        npMenuBar.add(edit);
        npMenuBar.add(format);
        npMenuBar.add(view);

        frame1.setJMenuBar(npMenuBar);
        frame1.add(bottom, BorderLayout.SOUTH);
```

```
		newdoc.addActionListener(this);
		copydoc.addActionListener(this);
		pastedoc.addActionListener(this);
		remdoc.addActionListener(this);
		status.addActionListener(this);
		savedoc.addActionListener(this);
		saveasdoc.addActionListener(this);

		fontfamily.addActionListener(this);
		fontsize.addActionListener(this);
		fontstyle.addActionListener(this);
	about.addActionListener(this);
		exit.addActionListener(this);

		maintext.addKeyListener(this);

		frame1.setSize(600,600);
		frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame1.setLocationRelativeTo(null);
		frame1.setVisible(true);
}
public void actionPerformed(ActionEvent ae)
{
		if(ae.getSource()== newdoc)
		{
		frame1.setTitle("New Document.txt");
		maintext.setText("");
		title.setText("");
		}
		else if (ae.getSource()== copydoc)
		{
			String texts= maintext.getText();
			pastecopydoc.setText(texts);
			JOptionPane.showMessageDialog(null, "Copy Text "+texts);
```

```
}
else if(ae.getSource()== remdoc)
{
		maintext.setText("");
		JOptionPane.showMessageDialog(null, "Removed");
}
else if (ae.getSource() == pastedoc)
{
		if(maintext.getText().length() != 0)
		{
			maintext.setText(maintext.getText());
		}
		else
		{
		maintext.setText(pastecopydoc.getText());
		}
}
else if(ae.getSource()== status)
{
		try{
			if(active ==0)
			{
				File f = new File(tle+".txt");
				details.setText("Size: "+f.length());
			}
		}
		catch (Exception e)
		{

		}
}
else if (ae.getSource()== fontfamily)
		{

	JOptionPane.showConfirmDialog(null, familylist, "Choose Font Family", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.PLAIN_MESSAGE);
```

```
                ffamily=String.valueOf(familylist.getSelectedValue());
                font1=new Font(ffamily,fstyle,fsize);
                maintext.setFont(font1);
                }
        else if (ae.getSource()== fontstyle)
                {

          JOptionPane.showConfirmDialog(null, stylelist, "Choose Font Style", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.PLAIN_MESSAGE);
                fstyle=stylevalue[stylelist.getSelectedIndex()];
                font1=new Font(ffamily,fstyle,fsize);
                maintext.setFont(font1);
                }
        else if (ae.getSource()== fontsize)
                {

          JOptionPane.showConfirmDialog(null, sizelist, "Choose Font Size", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.PLAIN_MESSAGE);
                fsizestr=String.valueOf(sizelist.getSelectedValue());
                fsize =Integer.parseInt(fsizestr);
                font1=new Font(ffamily,fstyle,fsize);
                maintext.setFont(font1);
                }
        else if(ae.getSource()==exit)
                {
                        frame1.dispose();
                }
        else if (ae.getSource()==savedoc)
        {
                title.setText(topicstitle);
                tle= title.getText();
                try{
          FileOutputStream filesave= new FileOutputStream(topicstitle+".txt");
                String s= maintext.getText();
                for(int i=0;i<s.length();i++)
                {
```

```
                filesave.write(s.charAt(i));
                }
                filesave.close();
                }
                catch (Exception e){

                }
  }
  else if (ae.getSource()==saveasdoc)
  {
                if(title.getText().length() == 0)
    {
                        topicstitle = JOptionPane.showInputDialog(null,
"Enter Your File Title?",
                                "Your File Name",
JOptionPane.QUESTION_MESSAGE);

                                title.setText(topicstitle);
                tle= title.getText();
                try{
    FileOutputStream filesave= new FileOutputStream(tle+".txt");
                String s= maintext.getText();
                for(int i=0;i<s.length();i++)
                {
                 frame1.setTitle(topicstitle+".txt");
                 filesave.write(s.charAt(i));
                }
                filesave.close();
                }
                catch (Exception e){

                }
                }
  }
  else if (ae.getSource()== opendoc)
```

```
                {


JFileChooser chooser = new JFileChooser();

                }
        else if(ae.getSource()==about)
        {
           JOptionPane.showMessageDialog(about,"this notepad is created by the students of cse c using java language in netbeans ide");
        }
}


public void keyTyped(KeyEvent ke){
                cl= maintext.getText().length();
                linecount = maintext.getLineCount();
                details.setText("Length: "+cl+" Line: "+linecount);
}


public static void main(String ar[])
{
new notepad();
}
}
```