Software Design Specification

for

Project Zelda

Version 1.2 approved

Prepared by Shravan Sridhar, Shivendra Singh, Siddharth Bisht, Suraj Kande and Shreyash Tiwari

NIIT University

11/11/2018

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
SDS v1.0	19/10/2018	Initial document.	1.0
SDS v1.1	02/11/2018	Addition of sequence and state diagrams.	1.1
SDS v1.2	11/11/2018	Addition of pseudocodes.	1.2

1. Introduction

1.1 Purpose

The product is a plugin created for our university's Moodle (moodle.niituniverity.in) for the sole-purpose of aiding in the smooth function of NEPL (Nano-Electronics Premier League). The proposed product will help the organizers and the faculty in-charges to keep track of each student's webQuests/research paper. It'll also help simplify the process of grading. In other words, the proposed software provides a web-based research development and publication platform for the university where the students can create their research papers online. This document is intended to help the final development of the product by providing all the necessary information pertaining to the production of the product.

1.2 Definitions, Acronyms and Abbreviations

- *IEEE: Institute of Electrical and Electronics Engineers*
- *SDS: Software Design Specifications*
- *HLD: High Level Design*
- *TBD: To be decided*

1.3 Product Scope

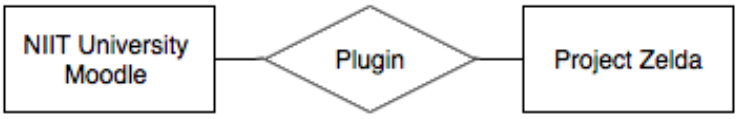
The software being described here will let the project managers and the faculty divide the students into groups and assign each group a topic of research which the students can then access. The students will be provided with an efficient text editor to work on their research paper. They can also keep track of the deadline from within the website. The faculty will have plagiarism checkers readily available. They will also have a window for grading each student. The goal here is to simplify this process of student-teacher interaction and the proposed model seems to efficiently fulfill this goal.

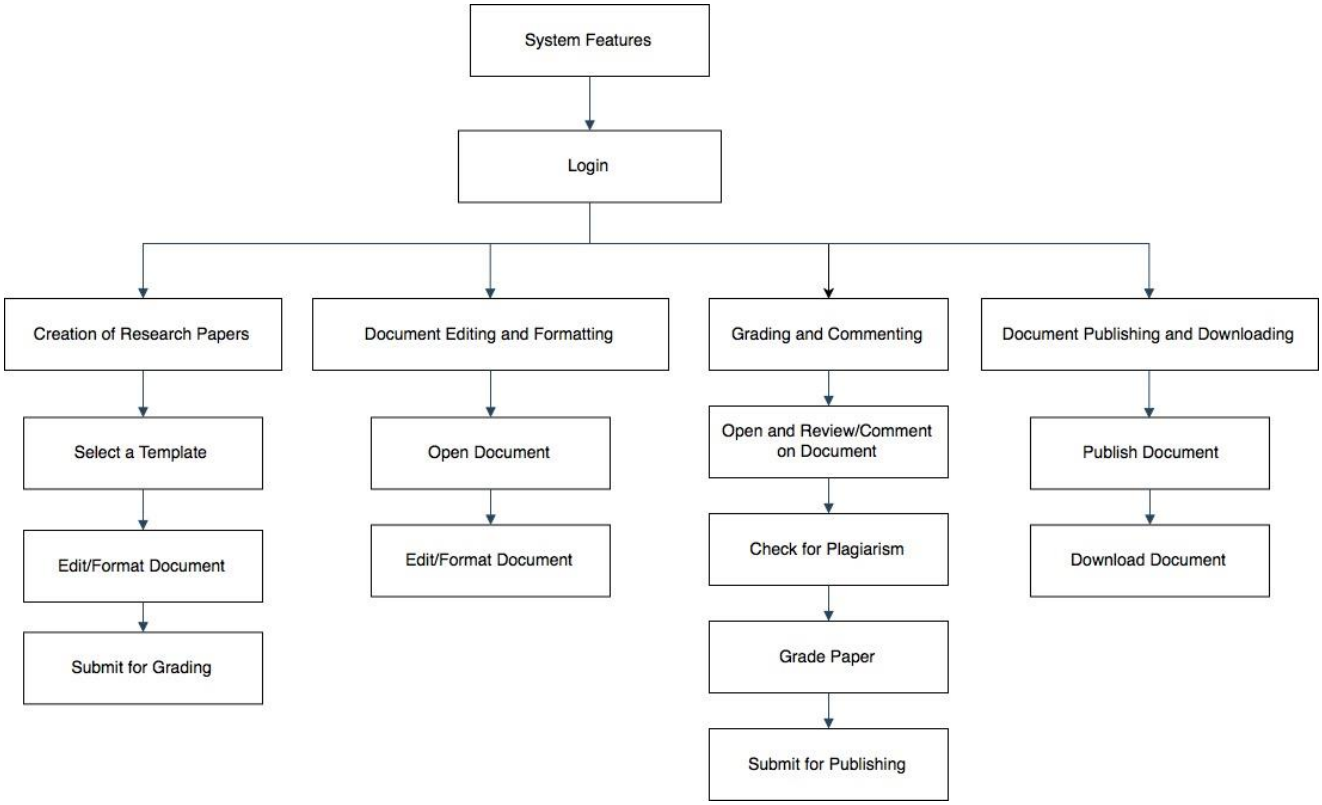
1.4 References

- *NIIT University's Moodle UI guidelines (We do not have access to the documentation yet but the website is accessible).*
- *830-1998 - IEEE Recommended Practice for Software Requirements Specifications.*

2. Conceptual Architecture

While the project was initially going to be proposed as a stand-alone web portal for submission and maintenance of research papers, the process later seemed to be redundant considering how all the students and faculty members are already registered on the university's Moodle. As a result, we later shifted from the idea of creating a separate portal to simply creating a plugin to provide all the functionalities that current Moodle cannot provide in order to fulfill our goal.





2.1 Creation of Research Papers

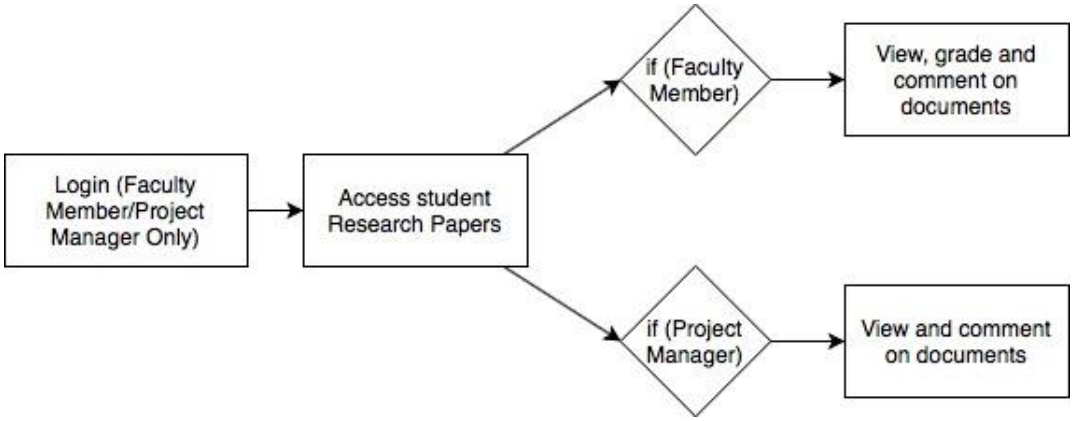
Allowing users to create standardized documents from a set of templates, that are of the format the teacher wants. Initially, the application will be used only for submitting papers but later can also be used to create official research dockets according to good paper writing conventions.

2.2 Document Editing/Formatting

Allows users to edit already created documents, provide predefined templates for ease of work and also format the documents to make them more systematic and appealing.

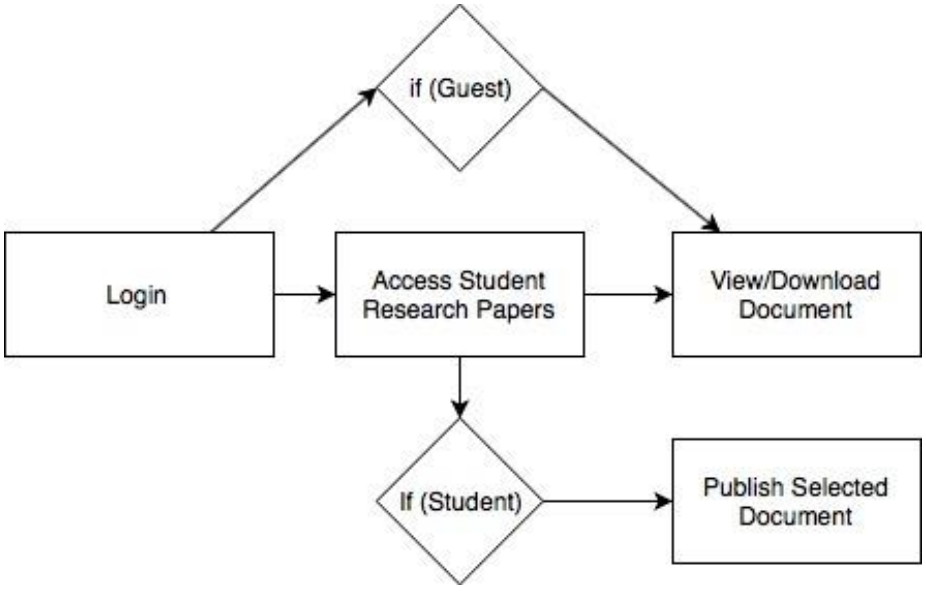
2.3 Grading and Commenting

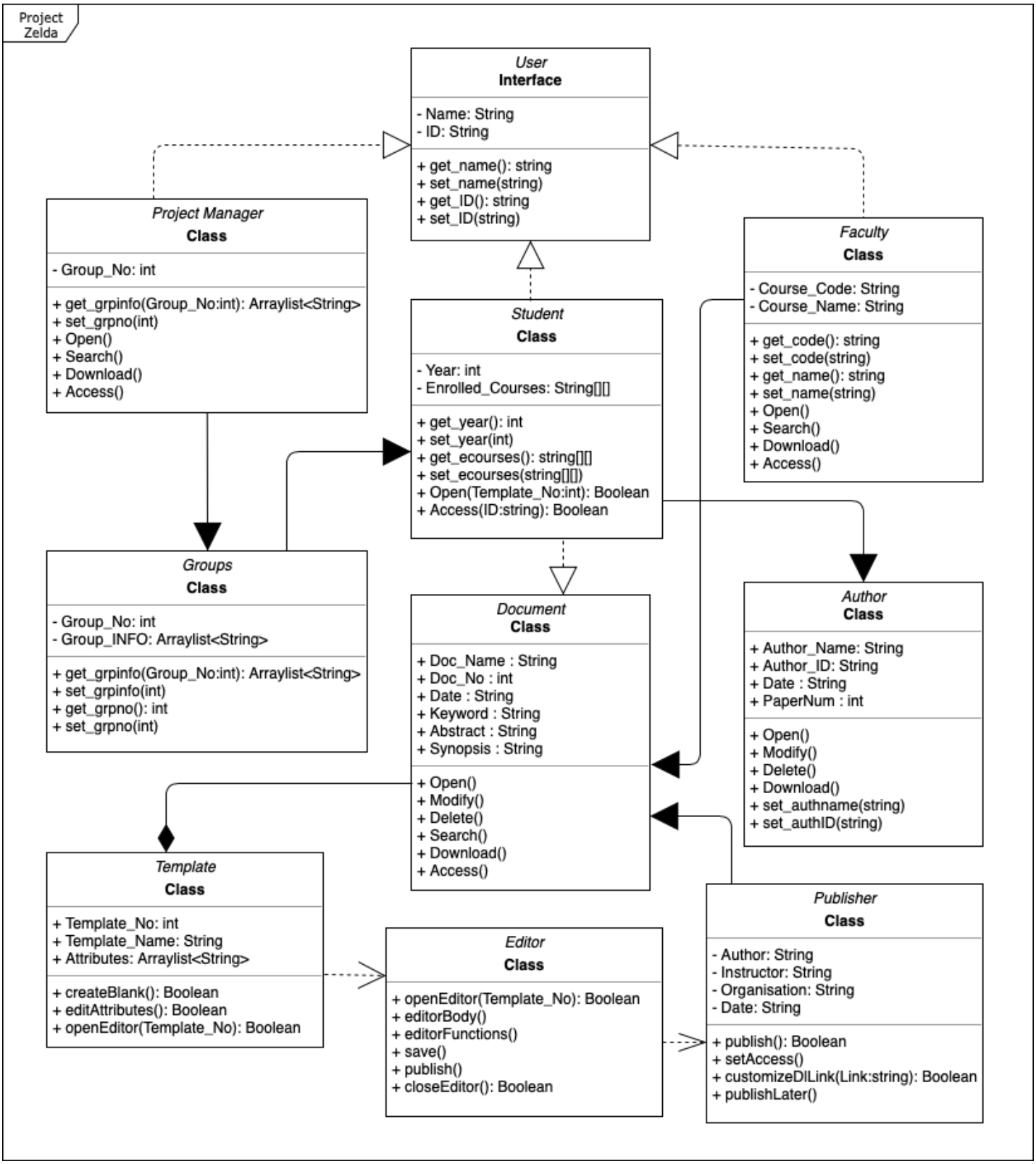
Allowing users with permission to grade (faculty members) and comment (both faculty members and project managers) on research papers of students working under them. They can view the document but cannot make changes to it.



2.4 Document Publishing and Viewing/Downloading

Allowing students (initially, only students will be allowed to create a research paper, this plugin will later be offered to anyone who has an account on Moodle) to create their documents online, on a browser and access it from anywhere as long as they are connected to the internet. Therefore, a student can keep his document on the cloud and can view/download it whenever and in whatever format (.pdf, .docx, etc.) he/she wants.



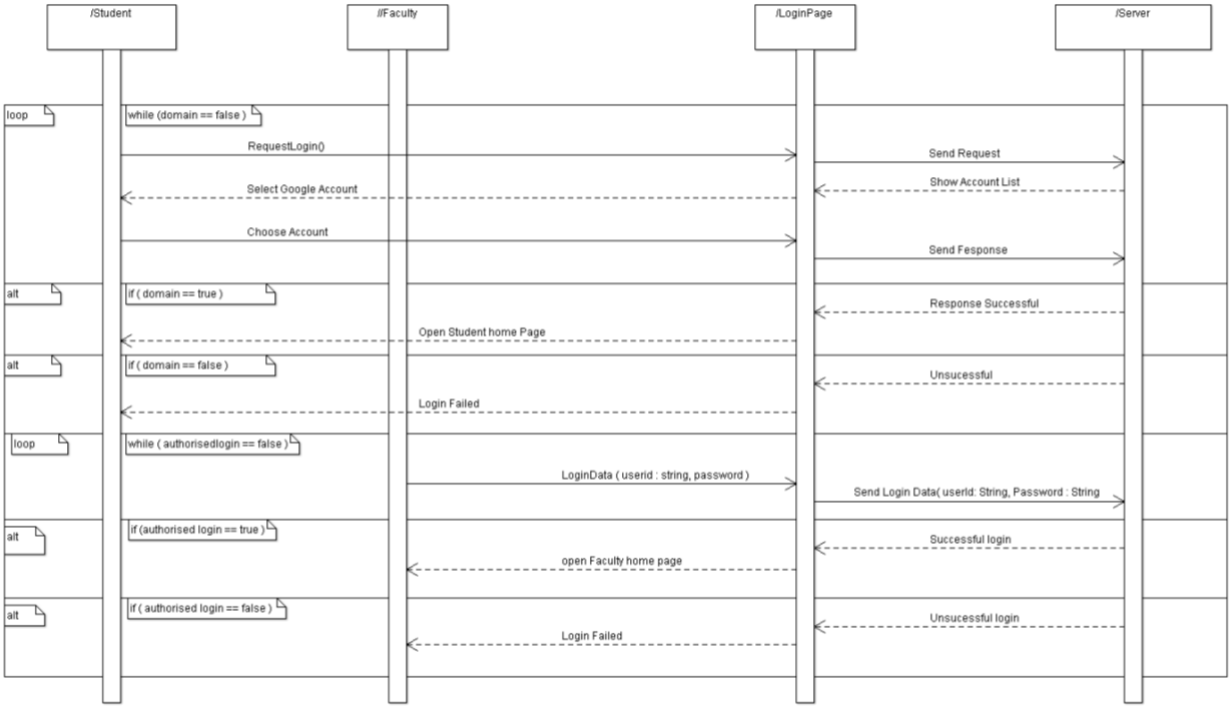


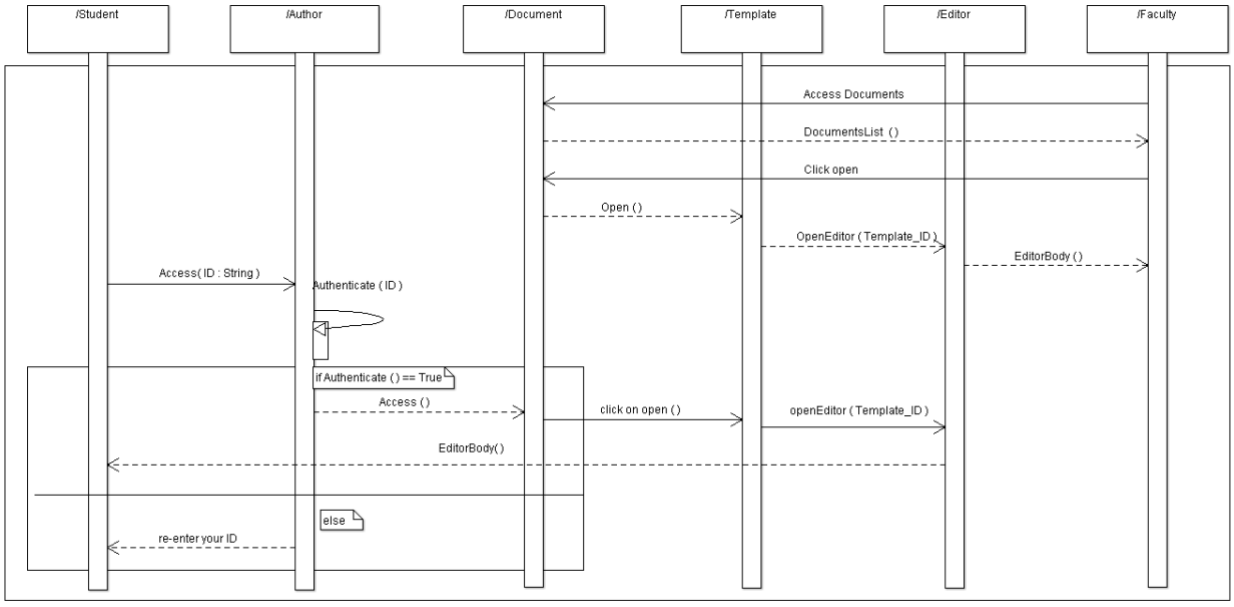
Class Diagram

3. Logical Architecture

3.1 Login and View

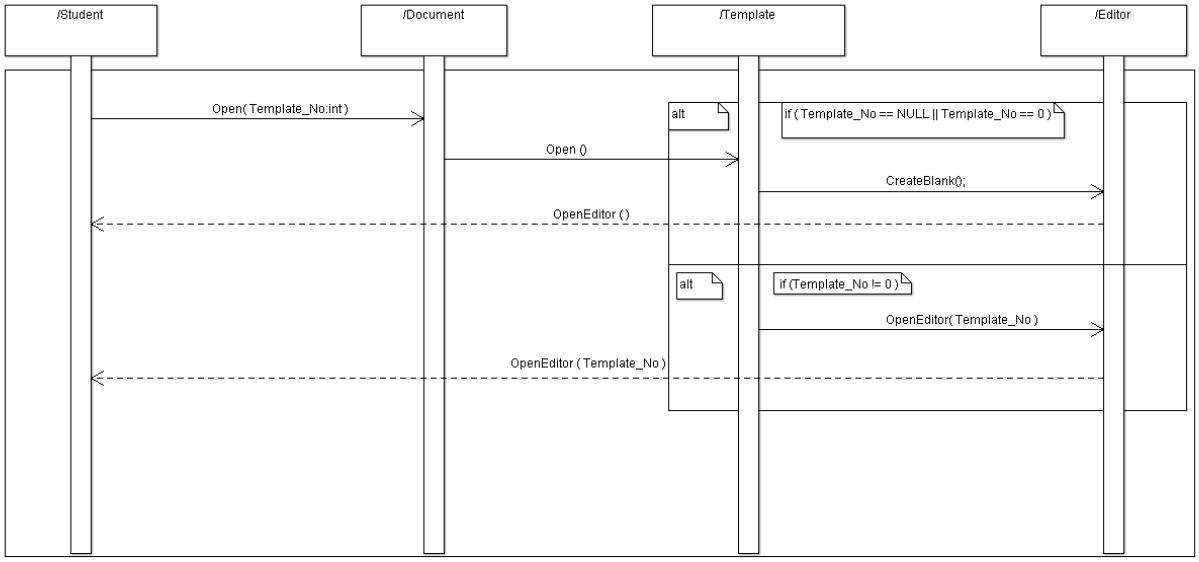
A user has to first login, after which he/she can access a document and view it. Since the document in consideration here is not just that of the owner, the user must have permission to view.

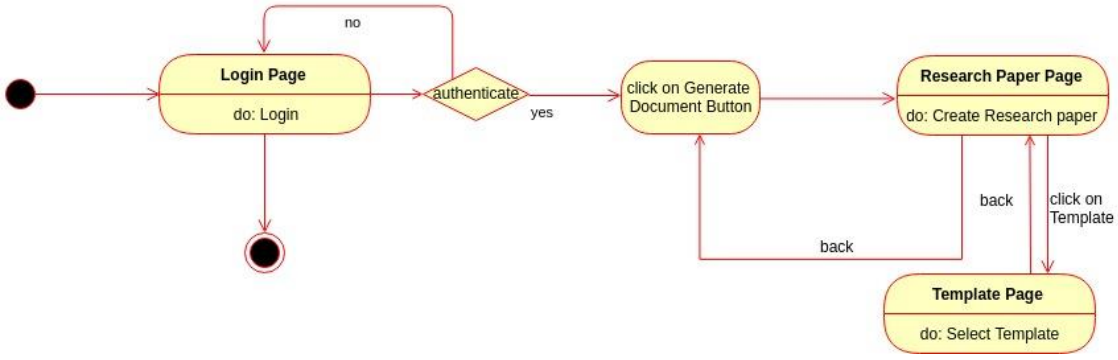




3.2 Creation of Research Papers

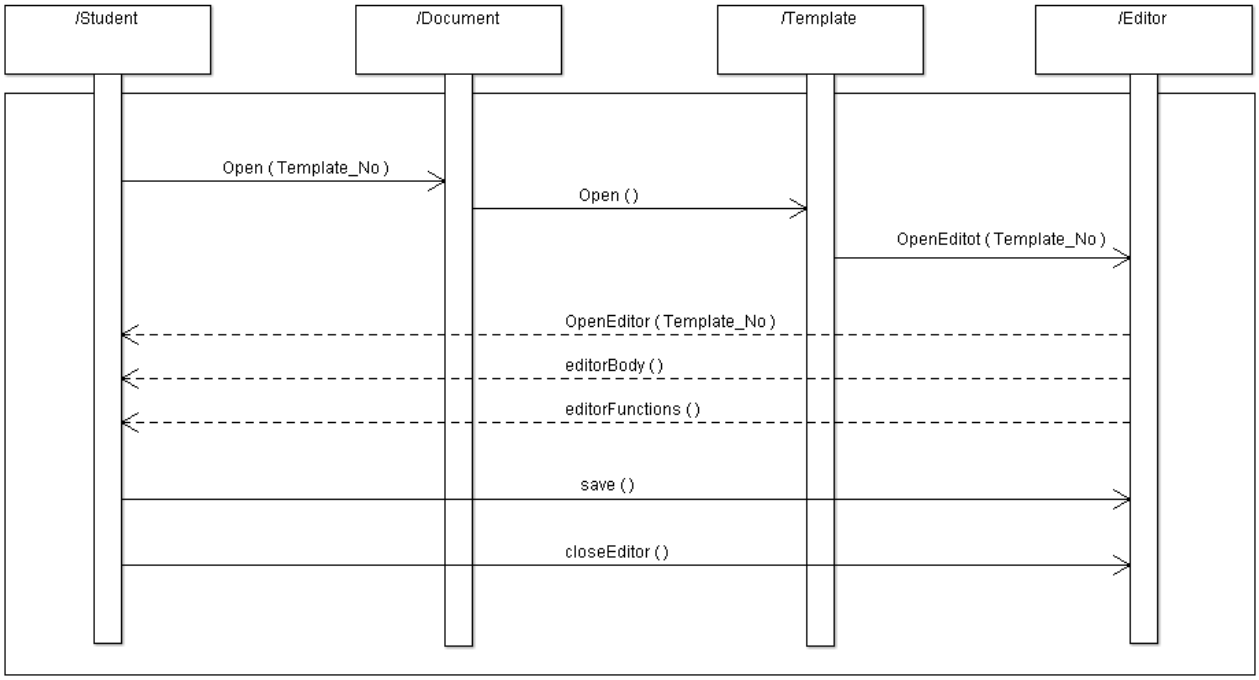
Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of usage of this feature will be displayed: it could be an icon or appear as a clickable item on Moodle.

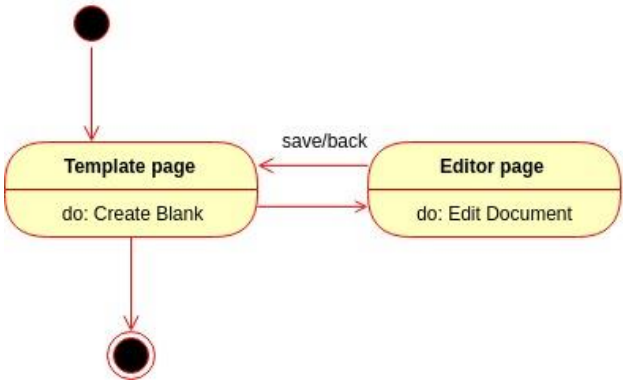


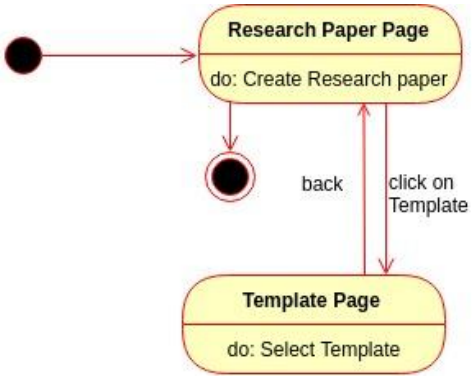


3.3 Select a Template

Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of creating a new document will be displayed: it could be an icon or appear as a clickable item on Moodle. Upon successful creation of document, the user will be displayed a list of templates that he/she can choose from.

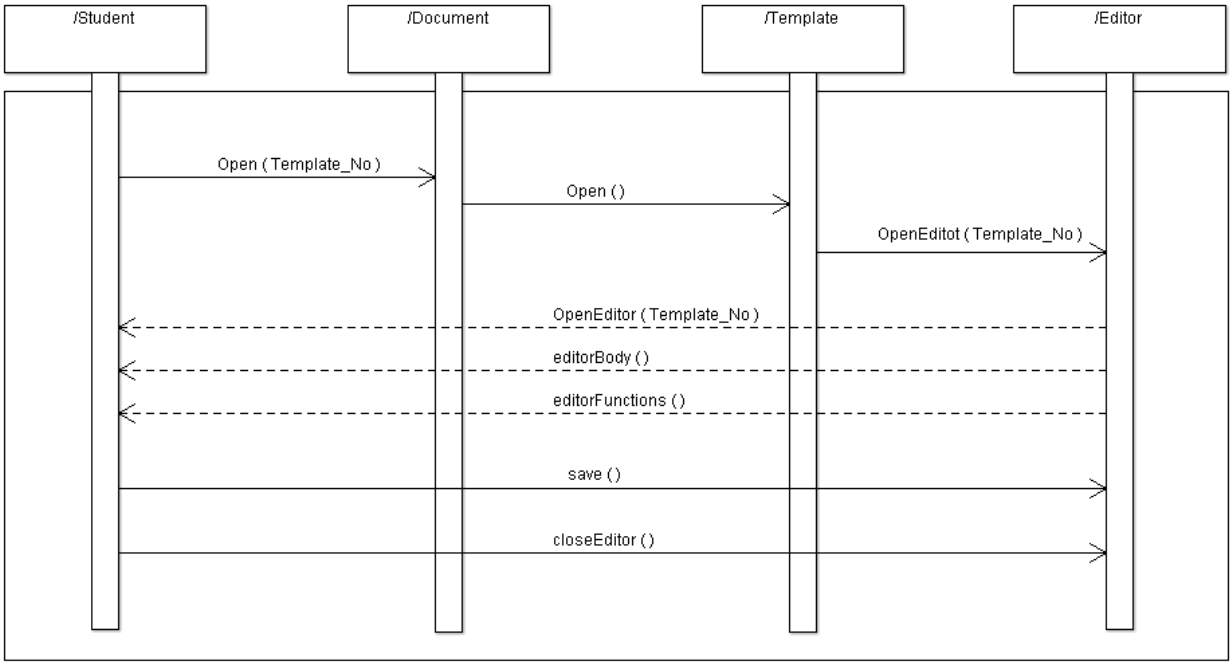


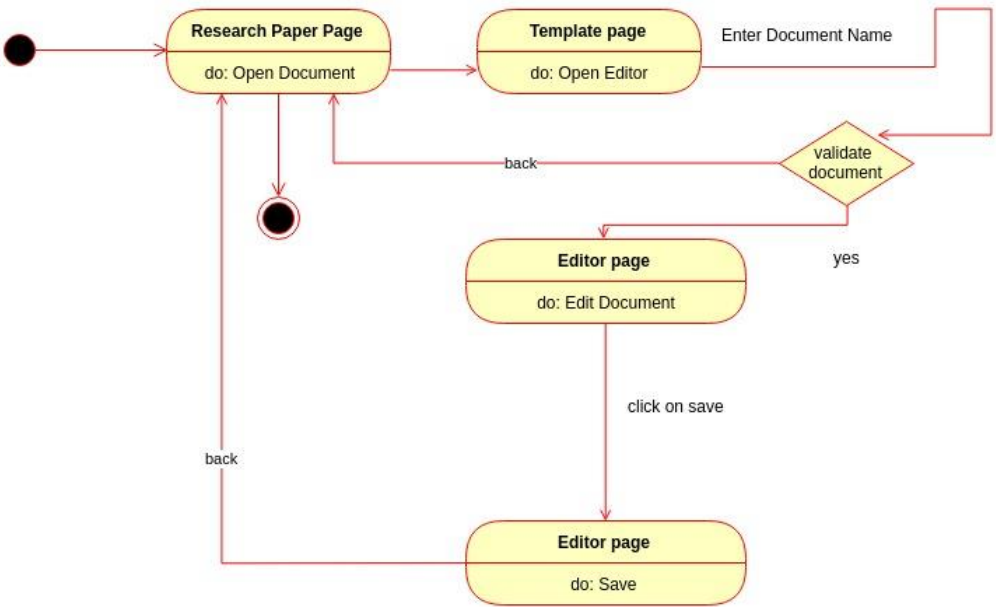




3.4 Edit/Format Documents

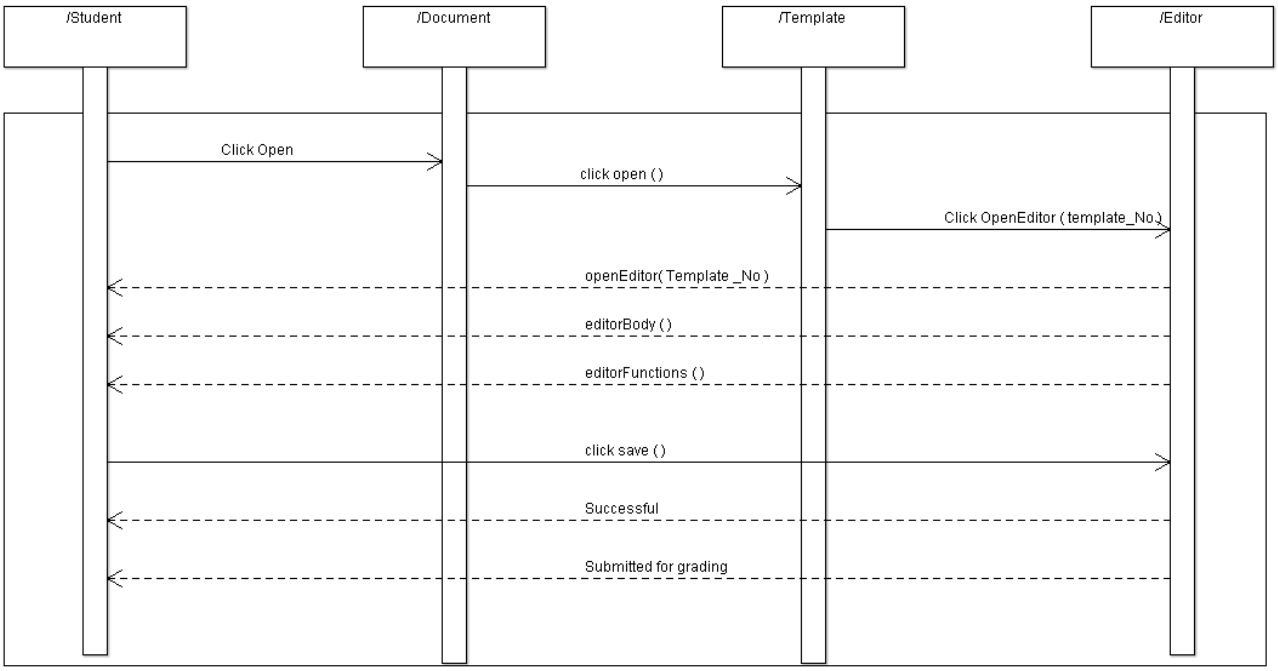
Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of creating a new document or opening an already created document will be displayed: it could be an icon or appear as a clickable item on Moodle. On opening a document, the user will be able to edit/format the document.

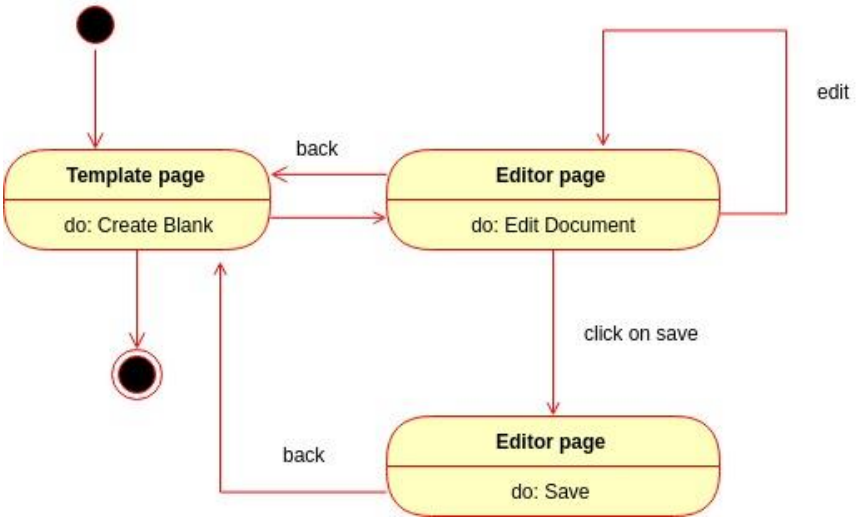




3.5 Submit for Grading

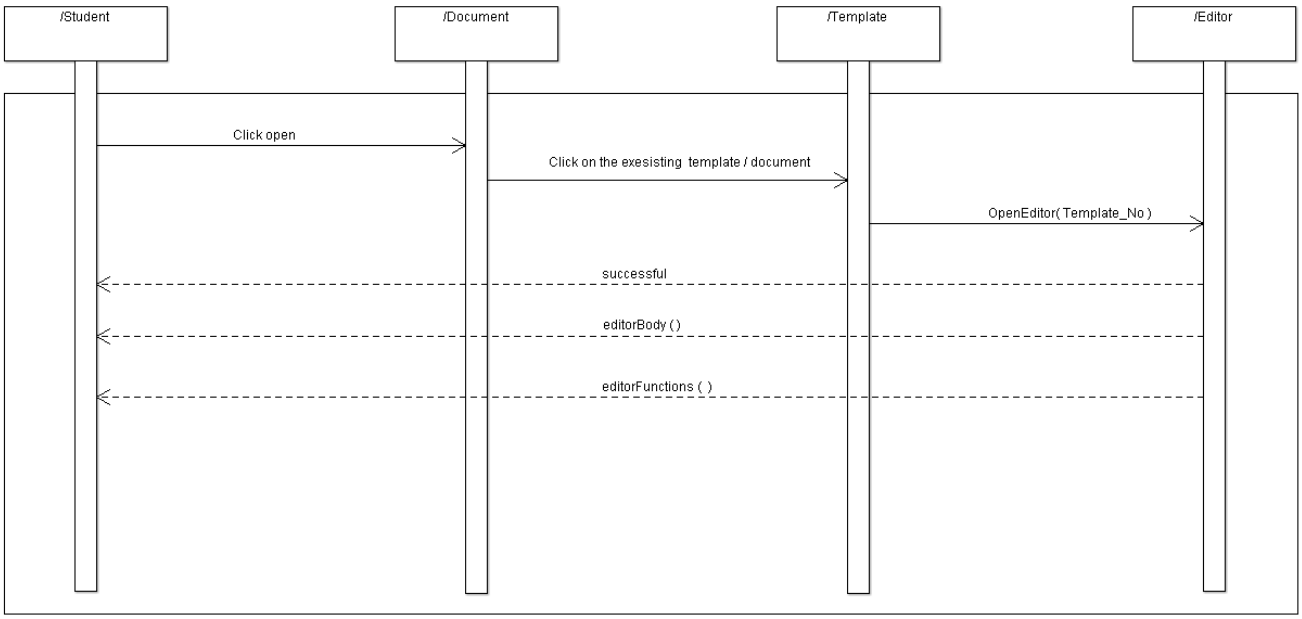
Once a teacher logs into his/her Moodle account, he/she will have the liberty to use the plugin. If student's have submitted their documents for grading, it'll be visible and they will in turn be able to grade.

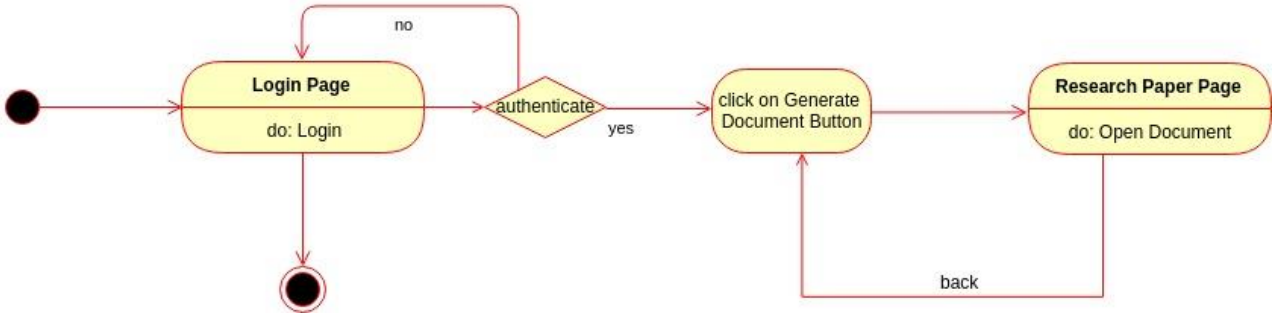




3.6 Open Document

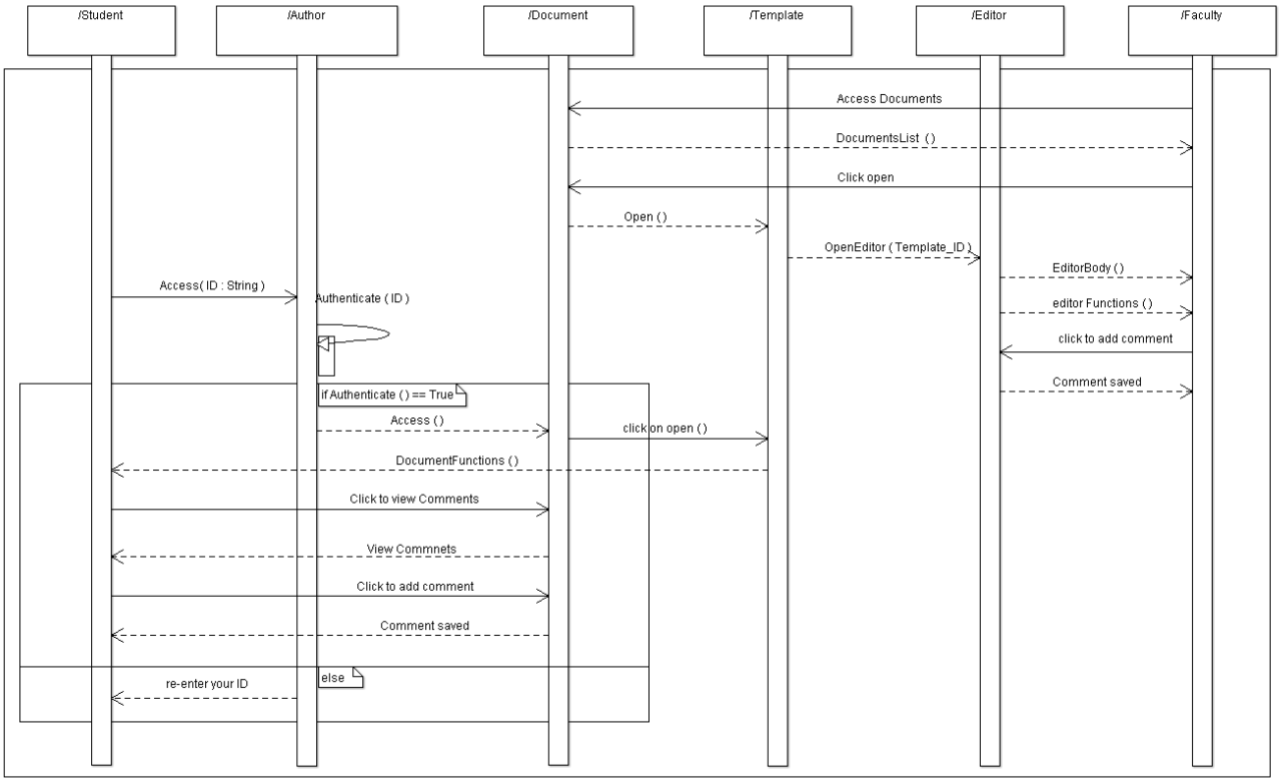
Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of opening a document will be displayed: it could be an icon or appear as a clickable item on Moodle. The user can then open and edit/format the document.

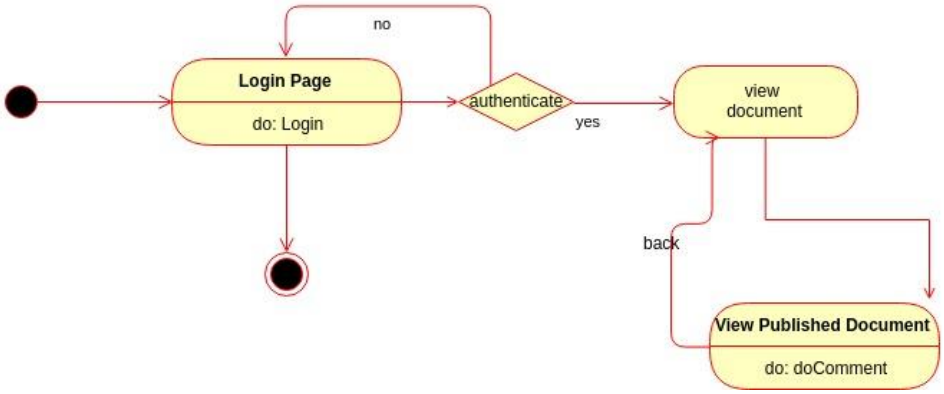




3.7 Comment on a Document

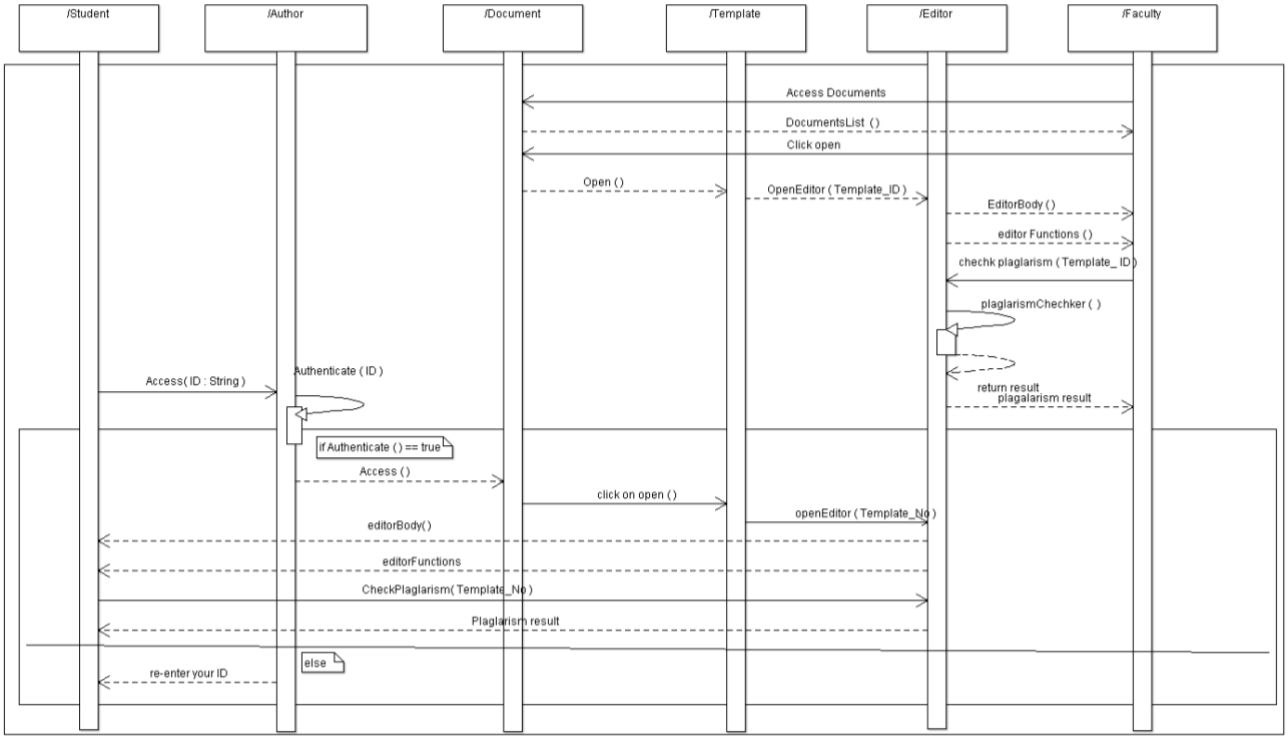
Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of opening a document which can be commented on will be displayed: it could be an icon or appear as a clickable item on Moodle. After opening, the user can comment on the document.





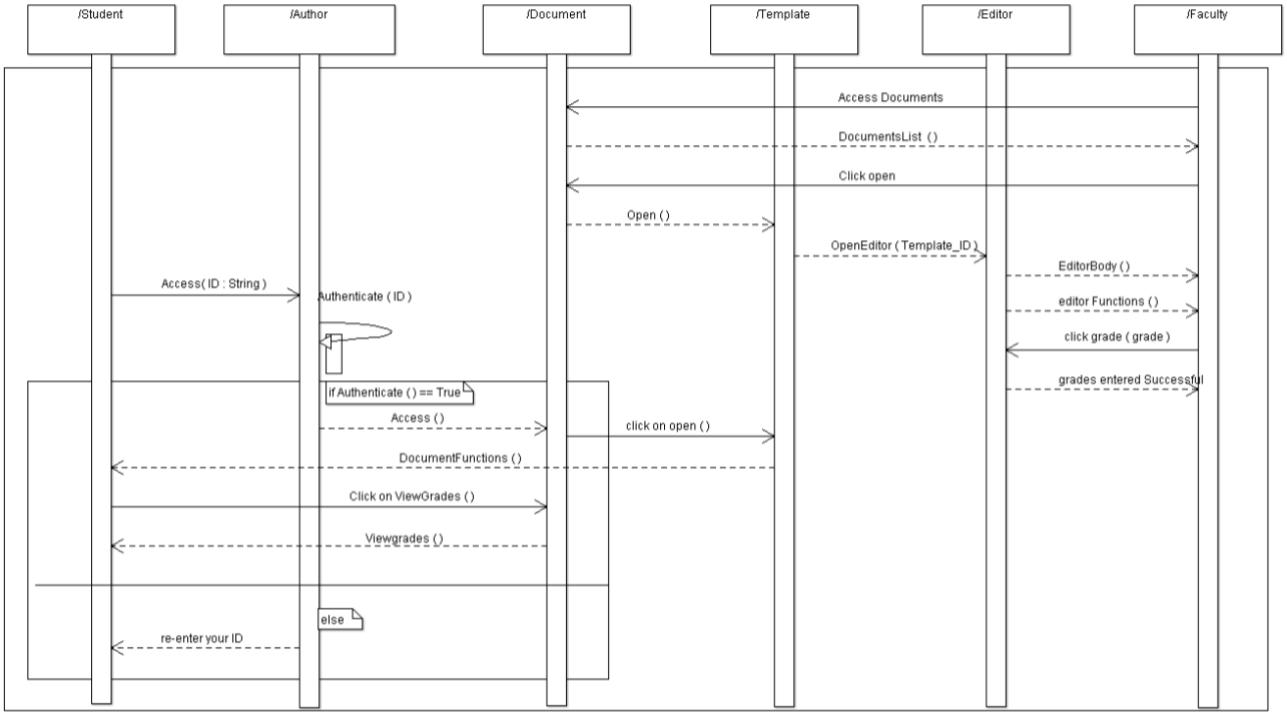
3.8 Check for Plagiarism

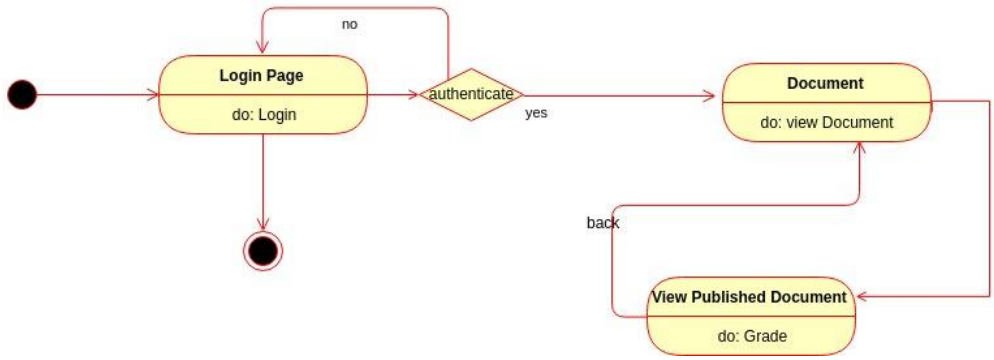
Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. The teacher will be able to open all those documents that have been submitted for grading and can check for plagiarism before grading.



3.9 Grading

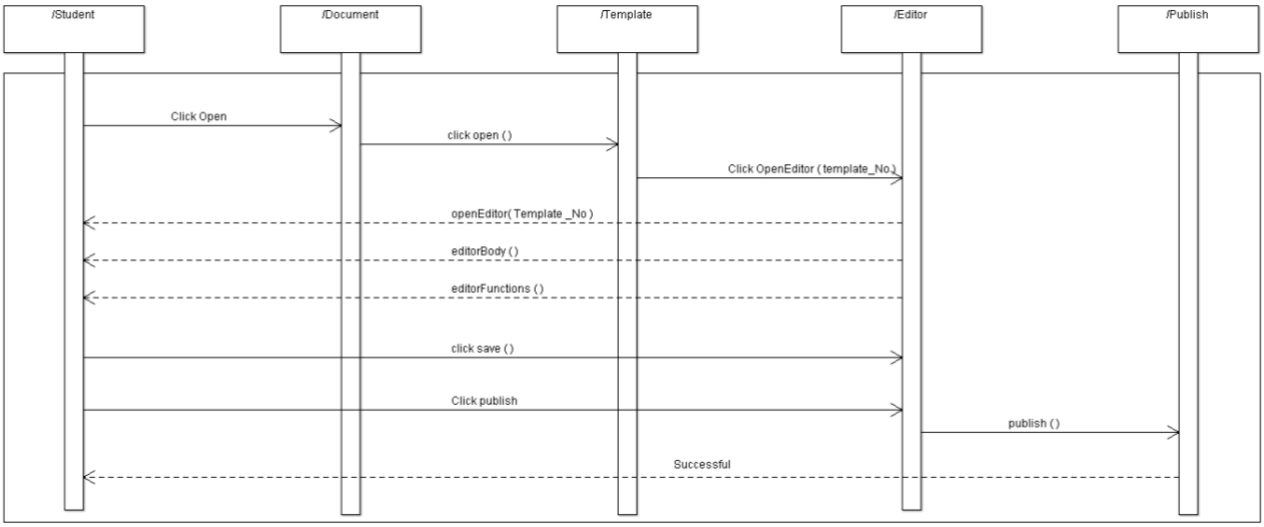
An option of usage of this feature will be displayed: it could be an icon or appear as a menu item on the document page.

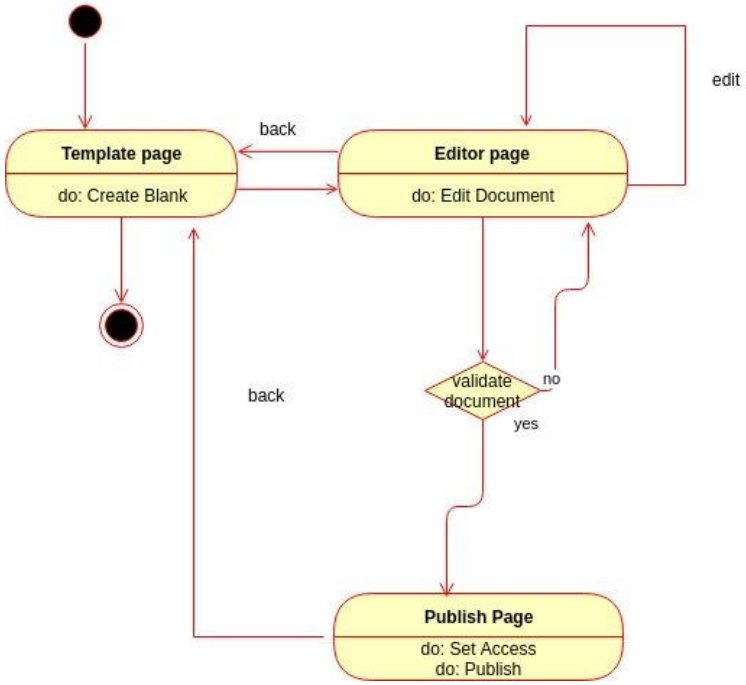




3.10 Document Publishing

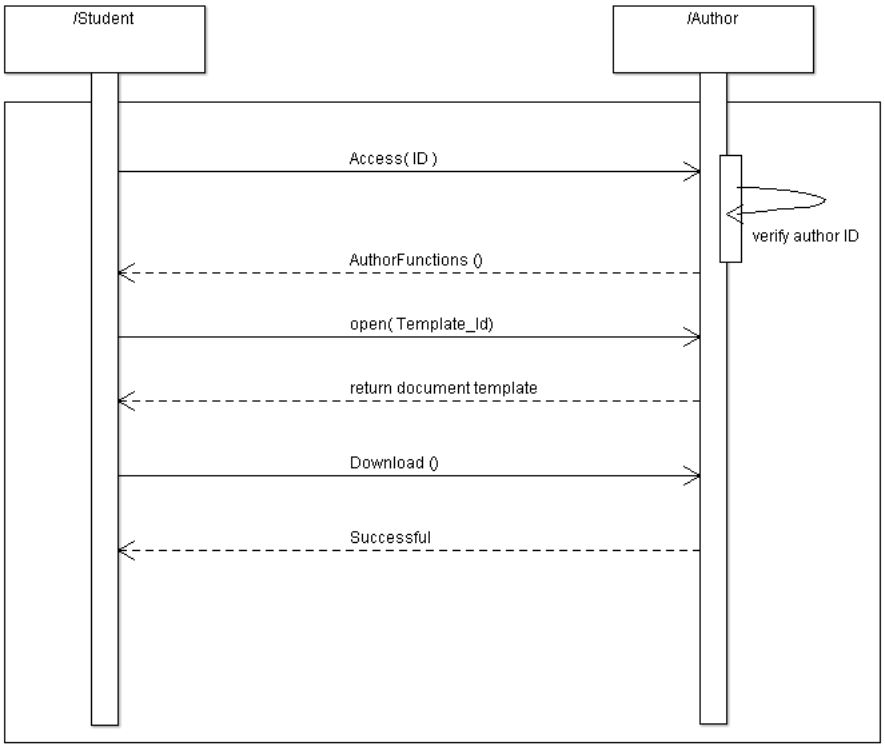
Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of usage of this feature will be displayed after a document is opened up. It will appear as a drop-down list on Moodle (list will contain format types).

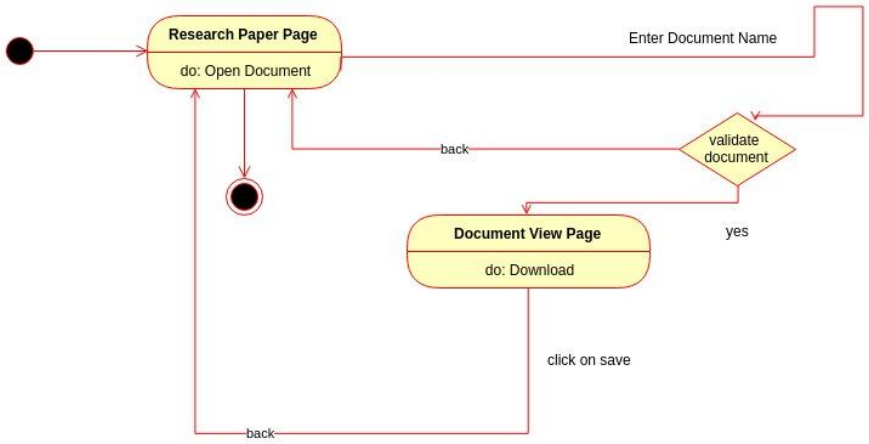




3.11 Download Document

Once a user logs into his/her Moodle account, he/she will have the liberty to use the plugin. An option of opening a document will be displayed: it could be an icon or appear as a clickable item on Moodle. After opening the document, the user can download in different formats.





4. Pseudocode for Components

1. *MainFrame()*

{

1. *start*
2. *initComponents()*
3. *Filter Files to display*
4. *Set JFileChooser to accept only text files*
5. *FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt", "text")*
6. *fileOpener.setFileFilter(filter)*
7.
8. *Launch the application on the middle of Screen*
9. *this.setLocationRelativeTo(null);*
10. *this.addWindowListener(new WindowAdapter()*
11. *end*

}

2. *WindowListener()*

{

1. *start*
2. *takes input of window adapter*
3. *call windowclosing*
4. *end*

}

3. *WindowClosing()*

1. *start*
2. *{*
3. *takes input object of windowevent*
4. *super.windowClosing(e)*
5. *change body of generated methods*
6. *choose Tools then Templates*
7. *int ans = JOptionPane.showConfirmDialog(rootPane, "Save Changes ?", "Confirm", JOptionPane.YES_NO_OPTION, JoptionPane.QUESTION_MESSAGE)*
8. *if (ans == JoptionPane.YES_OPTION) 9. then saveChanges() 10.*
11. *}*

12. *End*

4. *FileMain()*

{

1. *Start*
2. *initComponents()*
3. *this.setLocationRelativeTo(null)*
4. *currentEditingFile = file*
5. *FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt", "text")*
6. *fileOpener.setFileFilter(filter)*
7. *readTheParamFile(file)*
8. *end*

}

5. *ReadFile()*

{

1. *start*
2. *take input object of File{*
3. *try {*
4. *Scanner scn = new Scanner(file)*
5. *String buffer = ""*
6. *while (scn.hasNext()) {*
7. *buffer += scn.nextLine() + "\n"*
8. *}*
9. *display.setText(buffer)*
10. *}*
11. *catch (FileNotFoundException ex)*
12. *{*
13. *Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex)*
14. *}*
15. *}*
16. *end*

}

6. *SaveDisplay()*

{

1. *start*
2. *try {*
3. *PrintWriter printWriter = new PrintWriter(currentEditingFile)*
4. *printWriter.write(display.getText())*

```
5.      printWriter.close()
6.      JOptionPane.showMessageDialog(rootPane, "Saved", "Done",
        JOptionPane.INFORMATION_MESSAGE)
7.      }
8.      catch (FileNotFoundException ex)
9.      {
10.     Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex)
11.     }
12.
13.     end
}
```

7. *initialise_component()*

```
{
1.      start
2.      fileOpener = new javax.swing.JFileChooser()
3.      saveDialog = new javax.swing.JFileChooser()
4.      jPanel1 = new javax.swing.JPanel()
5.      jToolBar1 = new javax.swing.JToolBar()
6.      open= new javax.swing.JButton()
7.      save = new javax.swing.JButton()
8.      access = new javax.swing.JButton()
9.      modify = new javax.swing.Jbutton()
10.     download= new javax.swing.Jbutton()
11.     edit = new javax.swing.Jbutton()
12.     delete = new javax.swing.Jbutton()
13.     search = new javax.swing.JButton()
14.     jScrollPane = new javax.swing.JScrollPane()
15.     display = new javax.swing.JTextArea()
16.     jLabel = new javax.swing.JLabel()
17.     end

}
```

8. *ExitAction()*

```
{
1.      start
2.      saveDialog.setDialogType(javax.swing.JFileChooser.SAVE_DIALOG)
3.
saveDialog.setFileSelectionMode(javax.swing.JFileChooser.DIRECTORIES_ONLY)
4.
5.      setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
6.      setTitle("Java Text Editor")
```

```
}
```

9. *OpenButton()*

```
1.                  {
2.                  start
3.                  open.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  open.setText("Open");6.          open.setFocusable(false); 7.
8.                  open.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.                  open.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.                 open.addActionListener(new java.awt.event.ActionListener()
11.                 {
12.                 public void actionPerformed(java.awt.event.ActionEvent evt)
13.                 {
14.                 openActionPerformed(evt);15.
16.  }
17.  }
18.
19.       );
20.       jToolBar1.add(open);
21.       end
22.       }
```

10. *SaveButton()*

```
1.                  {
2.                  start
3.                  save.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  save.setText("save");6.          save.setFocusable(false); 7.
8.                  save.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.                  save.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.                 save.addActionListener(new java.awt.event.ActionListener()
11.                 {
12.                 public void actionPerformed(java.awt.event.ActionEvent evt)
13.                 {
14.                 saveActionPerformed(evt);15.
16.  }
```

```
17. }
18.
19.     );
20.     jToolBar1.add(save);
21.     end
22.     }
```

11. *AccessButton()*

```
1.              {
2.              start
3.              access.setIcon(new javax.swing.ImageIcon
4.              (getClass().getResource("image.png")));
5.              access.setText("access");6.     access.setFocusable(false);
6.
7.              access.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
8.              access.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
                8.      access.addActionListener(new java.awt.event.ActionListener()
                9.      {
                10.     public void actionPerformed(java.awt.event.ActionEvent evt)
                11.     {
                12.     accessActionPerformed(evt);15.
16. }
17. }18.
19.     );
20.     jToolBar1.add(access);
21.     end
22.     }
```

12. *ModifyButton()*

```
1.              {
2.              start
3.              modify.setIcon(new javax.swing.ImageIcon
4.              (getClass().getResource("image.png")));
5.              modify.setText("modify");6.             modify.setFocusable(false); 7.
8.          modify.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.          modify.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
```

```
10.          modify.addActionListener(new java.awt.event.ActionListener()
11.          {
12.          public void actionPerformed(java.awt.event.ActionEvent evt)
13.          {
14.          modifyActionPerformed(evt);
15.
16. }
17. }18.
19.      );
20.      jToolBar1.add(modify);
21.      end
22.      }
```

13. *EditButton()*

```
1.                {
2.                start
3.                edit.setIcon(new javax.swing.ImageIcon
4.                (getClass().getResource("image.png")));
5.                edit.setText("edit");6.          edit.setFocusable(false); 7.
8.          edit.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.          edit.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.         edit.addActionListener(new java.awt.event.ActionListener()
11.         {
12.         public void actionPerformed(java.awt.event.ActionEvent evt)
13.         {
14.         editActionPerformed(evt);15.
16. }
17. }18.
19.      );
20.      jToolBar1.add(edit);
21.      end
22.      }
```

14. *DownloadButton()*

```
1.                {
2.                start
```

```
3.                  download.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  download.setText("Open");6.       download.setFocusable(false); 7.
8.              download.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.              download.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.             download.addActionListener(new java.awt.event.ActionListener()
11.             {
12.             public void actionPerformed(java.awt.event.ActionEvent evt)
13.             {
14.             downloadActionPerformed(evt);
15.
16.  }
17.  }18.
19.         );
20.         jToolBar1.add(download);
21.         end
22.         }
```

15. *DeleteButton()*

```
1.                  {
2.                  start
3.                  delete.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  delete.setText("Open");6.         delete.setFocusable(false); 7.
8.              delete.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.              delete.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.             delete.addActionListener(new java.awt.event.ActionListener()
11.             {
12.             public void actionPerformed(java.awt.event.ActionEvent evt)
13.             {
14.             deleteActionPerformed(evt);15.         } }
19.         );
20.         jToolBar1.add(delete);
21.         end
22.         }
```

16. *SearchButton()*

```
1.                  {
```

```
2.                  start
3.                  search.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  search.setText("Open");6.       search.setFocusable(false); 7.
8.          search.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.          search.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.         search.addActionListener(new java.awt.event.ActionListener()
11.         {
12.         public void actionPerformed(java.awt.event.ActionEvent evt)
13.         {
14.         searchActionPerformed(evt);15.
16. }
17. }18.
19.       );
20.       jToolBar1.add(search);
21.       end
22.       }
```

17. *OpenAction()*

```
{
      1.
      2.    start
      3.    take input Object of Action event
      4.    {
      5.
      6.          Show File Open dialouge here
      7.          int status = fileOpener.showOpenDialog(rootPane);
      8.          if (status == JFileChooser.APPROVE_OPTION) {
      9.          if (currentEditingFile != null) {
      10.         // A file is opened and is being edited. Open the new file in new window
      11.         MainFrame newWindow = new MainFrame(fileOpener.getSelectedFile());
      12.         newWindow.setVisible(true);
      13.         newWindow.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
      14.         newWindow.pack();
      15.         return;
      16.         }
      17.         currentEditingFile = fileOpener.getSelectedFile();
      18.         System.out.println("File chosen. File name = " +
                  fileOpener.getSelectedFile().getName()); 19.
      20.             try {
```

```
21.             Now read the contents of file
22.             Scanner scn = new Scanner(new FileInputStream(currentEditingFile));
23.             String buffer = "";
24.             while (scn.hasNext()) {
25.             buffer += scn.nextLine() + "\n";
26.             }
27.             display.setText(buffer);
28.             } catch (FileNotFoundException ex) {
29.             Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
30.             }
31.
32.         } else {
33.         System.out.println("No file selected");
34.         }35.    } 36.
37. end

}
```

18. *SaveAction()*

```
1.              {
2.              {//saveActionPerformed
3.              //If we are editing a file opened, then we have to save the contents on
                the same file, currentEditingFile
4.              if (currentEditingFile != null) {
5.              try {
6.              PrintWriter printWriter = new PrintWriter(currentEditingFile);
7.              printWriter.write(display.getText());
8.              printWriter.close();
9.              JOptionPane.showMessageDialog(rootPane, "Saved to " +
                currentEditingFile.getName(), "Done",
                JOptionPane.INFORMATION_MESSAGE);
10.             } catch (FileNotFoundException ex) {
11.             Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null,
                ex);
12.             }
13.             } else {
14.             int status = saveDialog.showOpenDialog(rootPane);
15.             if (status == JFileChooser.APPROVE_OPTION) {
16.             //We got directory. Now needs file name
```

```
17.  String fileName = JOptionPane.showInputDialog("File Name",
     "Untitled.txt");
18.  if (!fileName.contains(".txt")) {
19.  fileName += ".txt";
20.  }
21.  File f = new File(saveDialog.getSelectedFile() + "\\" + fileName);
22.  if (f.exists()) {
23.  JOptionPane.showMessageDialog(rootPane, "File Already Exist.",
     "Error",
JOptionPane.ERROR_MESSAGE);
24.  return;
25.  } else {
26.  try {
27.  f.createNewFile();
28.  PrintWriter printWriter = new PrintWriter(f);
29.  printWriter.write(display.getText());
30.  printWriter.close();
31.  JOptionPane.showMessageDialog(rootPane, "Saved", "Done",
JOptionPane.INFORMATION_MESSAGE);
32.  } catch (IOException ex) {
33.  Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null,
     ex);
34.  }
35.  }
36.  } else {
37.  JOptionPane.showMessageDialog(rootPane, "Error Occured", "Cant
     Save",
JOptionPane.ERROR_MESSAGE);
38.  }
39.  }40.
41. }
```

19. *TemplateButton()*

```
1.  {
2.  start
3.  template.setIcon(new javax.swing.ImageIcon
4.  (getClass().getResource("image.png")));
5.  template.setText("Open");6.  template.setFocusable(false); 7.
8.  template.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
```

```
9.          template.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.         template.addActionListener(new java.awt.event.ActionListener()
11.         {
12.         public void actionPerformed(java.awt.event.ActionEvent evt)
13.         {
14.         templateActionPerformed(evt);15.
16. }
17. }18.
19.       );
20.       jToolBar1.add(template);
21.       end
22.       } 23.
```

20. *PublishButton()*

```
1.                  {
2.                  start
3.                  publish.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  publish.setText("Open");6.       publish.setFocusable(false); 7.
8.          publish.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.          publish.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.         publish.addActionListener(new java.awt.event.ActionListener()
11.         {
12.         public void actionPerformed(java.awt.event.ActionEvent evt)
13.         {
14.         publishActionPerformed(evt);15.
16. }
17. }18.
19.       );
20.       jToolBar1.add(publish);
21.       end
22.       } 23.
24.
```

19. *openEditorButton()*

```
1.                  {
2.                  start
```

```
3.                  openEditor.setIcon(new javax.swing.ImageIcon
4.                  (getClass().getResource("image.png")));
5.                  openEditor.setText("Open");6.          openEditor.setFocusable(false); 7.
8.            openEditor.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
9.            openEditor.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
10.           openEditor.addActionListener(new java.awt.event.ActionListener()
11.           {
12.           public void actionPerformed(java.awt.event.ActionEvent evt)
13.           {
14.           openEditorActionPerformed(evt);
15.
16. }
17. }18.
19.       );
20.       jToolBar1.add(openEditor);
21.       end
22.       }
```

Class: ProjectManager

```
21. group_info()
1.                  {
2.                  void setid(String id){
3.                  this.id = id;
4.                  }5.
6.          public getid(){
7.                   return id;
8.                  }
9.
10.      }
```

search()
download() access()

Interface: User

```
1.   {
2.                  void setname(String name){
3.                  this.name = name;
```

```
4.
    } 5.

6.          public getname(){
7.          return name;
8.          }
9.          void setid(String id){
10.         this.id = id; 11.       }
12.
13.     public getid(){
14.         return id;
15.         }
16.
17. }
```

Class: Student

```
    2.      {
    3.          {
    4.          void    setyear(String
                year){
5.          this.year = year;
6.          }7.
8.          public getyear(){
9.          return year;}
    10.
    11.         void setcourses(String courses){
    12.         this.courses = courses; 13.     }
    14.
    15.     public getcourses(){
    16.         return courses;
    17.         }
    18.
    19. }
```

opentemplate()
accessdoc()

Class: Faculty

```
1.  {
2.          void setcode(String code){
3.          this.code = code;
```

```
4.
 } 5.
6.          public getcode(){
7.          return code;
8.          }
9.          void setname(String name){
10.         this.name = name; 11.       }
12.
13.         public getname(){
14.         return name;
15.         }
16.
17.     }
```

opendoc()
searchdoc()
downloaddoc()
accessdoc()

Class:Author

open()
modify()
delete()
download()
set_authname ()

```
1.  {
2.          void setauthor(String author){
3.          this.author = author;
4.      }
5.
6.
7.
8.      }
```

set_authID()

```
1.  {
2.          void setname(String name){
3.          this.name = name;
```

```
4.          }
5.
6.
7.      }
```

Class: Publisher

publish()
setaccess()
customizeIDlink()
publishlater()

Class:Editor

openeditor()
editorbody()
save()
publish()
closeeditor()

Class: Template

createblank()
editattributes()
openeditor()

Class: Document

open()
modify()
delete()
search()
download()
access()