Final Project

COMP 354- Fall 23

Apekshaba Gohil		Anam A. Shaikh		Marzieh Adeli

apekshabagohil.it@gmail.com	anams403@gmail.com	marzieh.adeli@gmail.com

Components of Project Report

1. **Project Schedule**
2. **Risk Analysis and Mitigation Plan**
3. **Materialized Risks**
4. **Implementation**
5. **Testing Strategies**
6. **Quality Control**
7. **Revision History and Contributions**

1. Schedule

Define a Feasible Schedule with the following:

1. Task Description
2. Task Ownership
3. Estimated Time to Complete the Task(for estimation define which technique you feel according to you is suitable for your project)
4. Also, give brief on which tool did you use to manage and keep track of your processes or version control, etc.

Estimation Technique

Estimation techniques

1. Reconciling- Based
2. Problem Based
3. LOC-Based Estimation
4. FP-Based Estimation
5. Use-Case Point Estimation

2. Risk Analysis and Mitigation

1. Define the technique you used to identify the risks.

2. Define the impact of your risks (categorize them based on severity, low, high, critical, catastrophic)

3. Devise a Mitigation plan for each type of risk

3. Materialized Risks

Materialization **occurs when the thing we thought was possible actually happens**.

Report the materialization related to the project and also when working within the team.

4. Implementation

How did you implement your Project using the **design strategy** (as described in Assignment 2)

Describe your implemented components based on that design strategy.

5. Testing Strategy

Define the unit tests you practically applied on your components as designed in Assignment 3.

What tool did you use to perform unit tests (e.g. Junit, Mockito, etc.)

As the Integration Strategy defined in your Assignment 3 how did you apply it to integrate your software ?

Any tools used and how did that tool help you in the integration.

7. Quality Control

Describe if there were any changes in your requirements, design strategies, test-cases or integration plan.

Explain what was the change and why did you change with a clear and reasonable justification.

With the change how did you preserve the quality of your software.

8. Revision History and Contribution

Like every assignment we defined the revision history and contribution make sure to add this in your final report.

Project Report

Important!

Keep this report as simple and concise as possible.

Maximum number of pages of your report must **not exceed more than 20**.

Be brief and precise about why and how did you plan and use the implementation.

During implementation if there is any change in your project (e.g. change in your implementation plan itself) do mention it in your project with a justification.

Project

Your ConcordiaCart Web App must focus on mainly

For Customer :

1. Display Most Selling/ Least Selling Items (User Recommendation you must use any parameter for recommendation)
2. Display Discount based on recommendations
3. Used Products Panel

For Admin:

1. Discount for most selling/ least selling items
2. Stock Management

Project

It’s a web-app so make sure depending on the user login we get the information based on the user account.

In your demo, make sure to show your test cases as described in the project report.

Components of Project Report

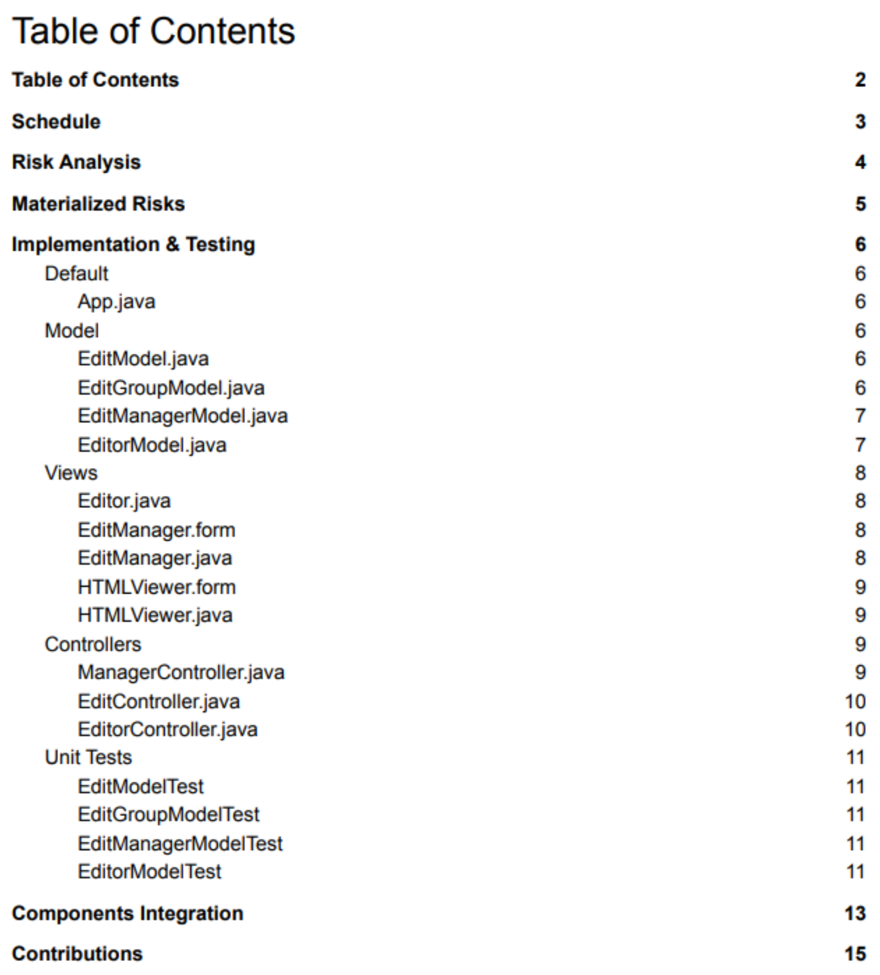
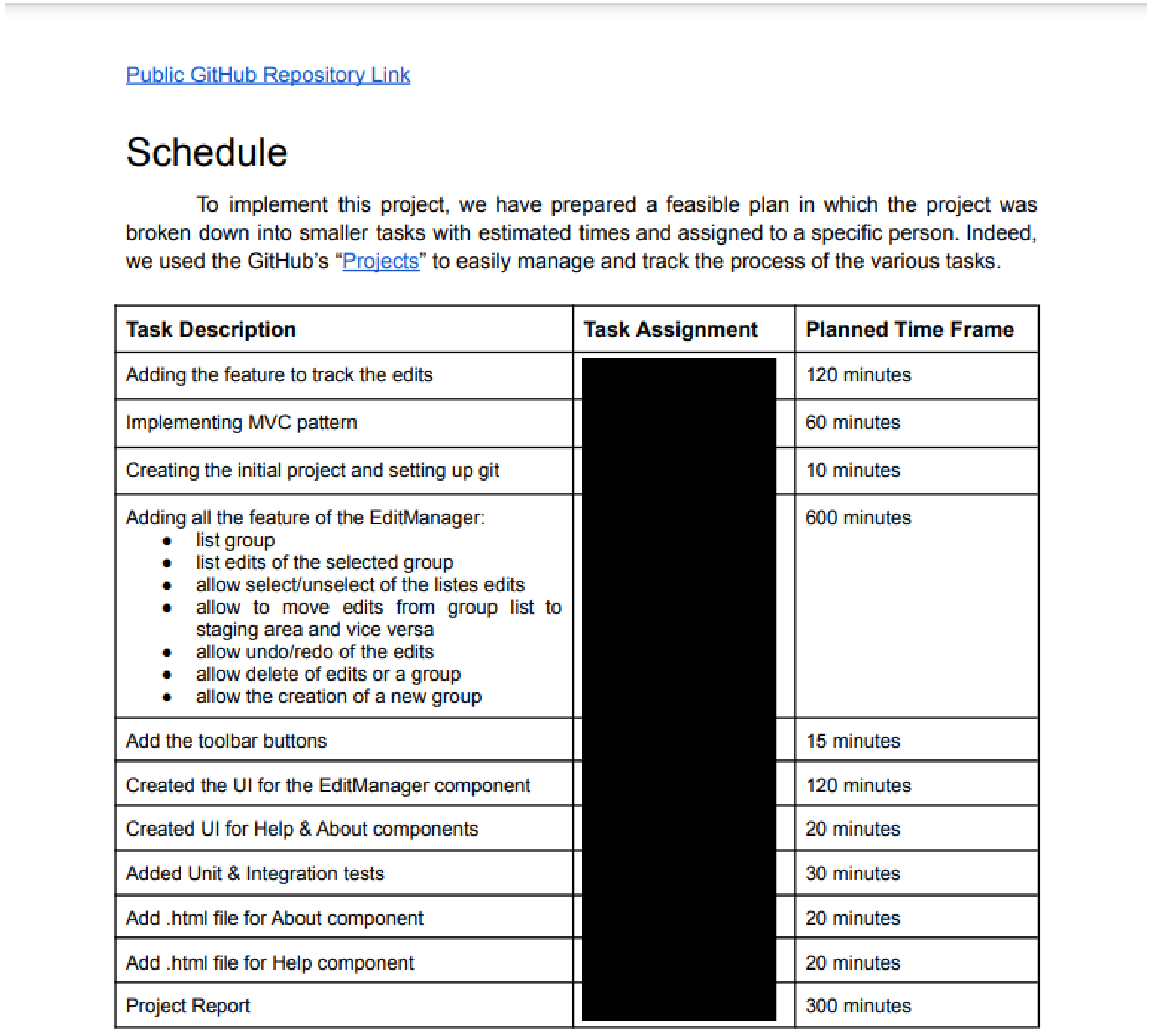


Table of Contents

1. Schedule



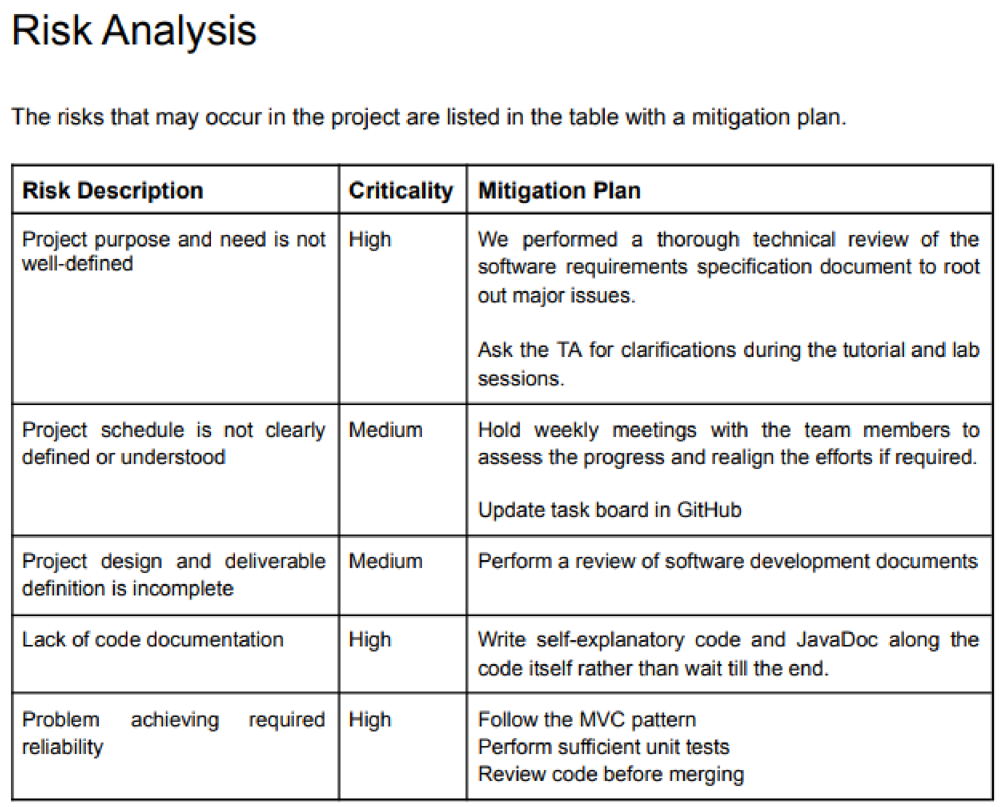
Public GitHub Repository Link

Schedule

To implement this project, we have prepared a feasible plan in which the project was broken down into smaller tasks with estimated times and assigned to a specific person. Indeed, we used the GitHub's "Projects" to easily manage and track the process of the various tasks.

Task Description	Task Assignment	Planned Time Frame
Adding the feature to track the edits	[redacted]	120 minutes
Implementing MVC pattern		60 minutes
Creating the initial project and setting up git		10 minutes
Adding all the feature of the EditManager: • list group • list edits of the selected group • allow select/unselect of the listes edits • allow to move edits from group list to staging area and vice versa • allow undo/redo of the edits • allow delete of edits or a group • allow the creation of a new group		600 minutes
Add the toolbar buttons		15 minutes
Created the UI for the EditManager component		120 minutes
Created UI for Help & About components		20 minutes
Added Unit & Integration tests		30 minutes
Add .html file for About component		20 minutes
Add .html file for Help component		20 minutes
Project Report		300 minutes

2. Risk Analysis

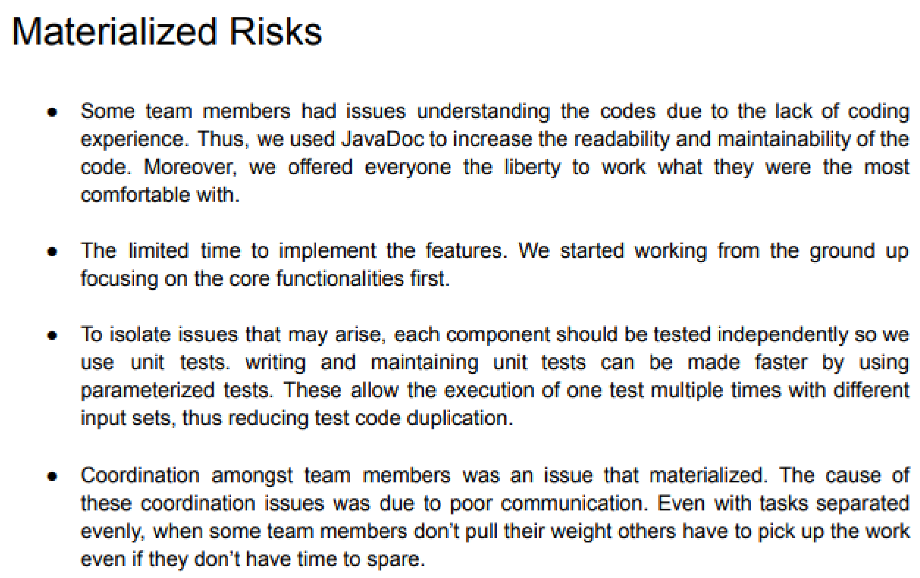


Risk Analysis

The risks that may occur in the project are listed in the table with a mitigation plan.

Risk Description	**Criticality**	**Mitigation Plan**
Project purpose and need is not well-defined	High	We performed a thorough technical review of the software requirements specification document to root out major issues. Ask the TA for clarifications during the tutorial and lab sessions.
Project schedule is not clearly defined or understood	Medium	Hold weekly meetings with the team members to assess the progress and realign the efforts if required. Update task board in GitHub
Project design and deliverable definition is incomplete	Medium	Perform a review of software development documents
Lack of code documentation	High	Write self-explanatory code and JavaDoc along the code itself rather than wait till the end.
Problem achieving required reliability	High	Follow the MVC pattern Perform sufficient unit tests Review code before merging

3. Materialized Risk

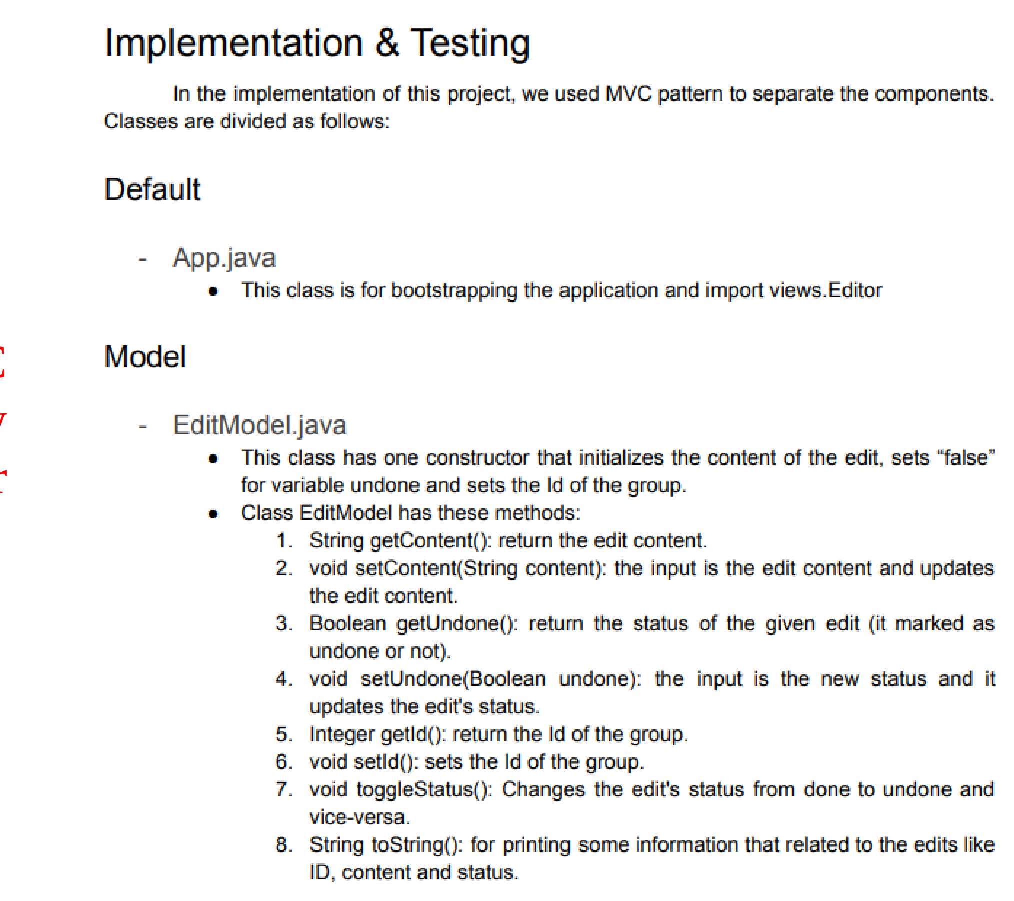


Materialized Risks

- Some team members had issues understanding the codes due to the lack of coding experience. Thus, we used JavaDoc to increase the readability and maintainability of the code. Moreover, we offered everyone the liberty to work what they were the most comfortable with.
- The limited time to implement the features. We started working from the ground up focusing on the core functionalities first.
- To isolate issues that may arise, each component should be tested independently so we use unit tests. writing and maintaining unit tests can be made faster by using parameterized tests. These allow the execution of one test multiple times with different input sets, thus reducing test code duplication.
- Coordination amongst team members was an issue that materialized. The cause of these coordination issues was due to poor communication. Even with tasks separated evenly, when some team members don't pull their weight others have to pick up the work even if they don't have time to spare.

4. Implementation

They have used the MVC Architecture and that's how they have explained their implementation task.



Implementation & Testing

In the implementation of this project, we used MVC pattern to separate the components. Classes are divided as follows:

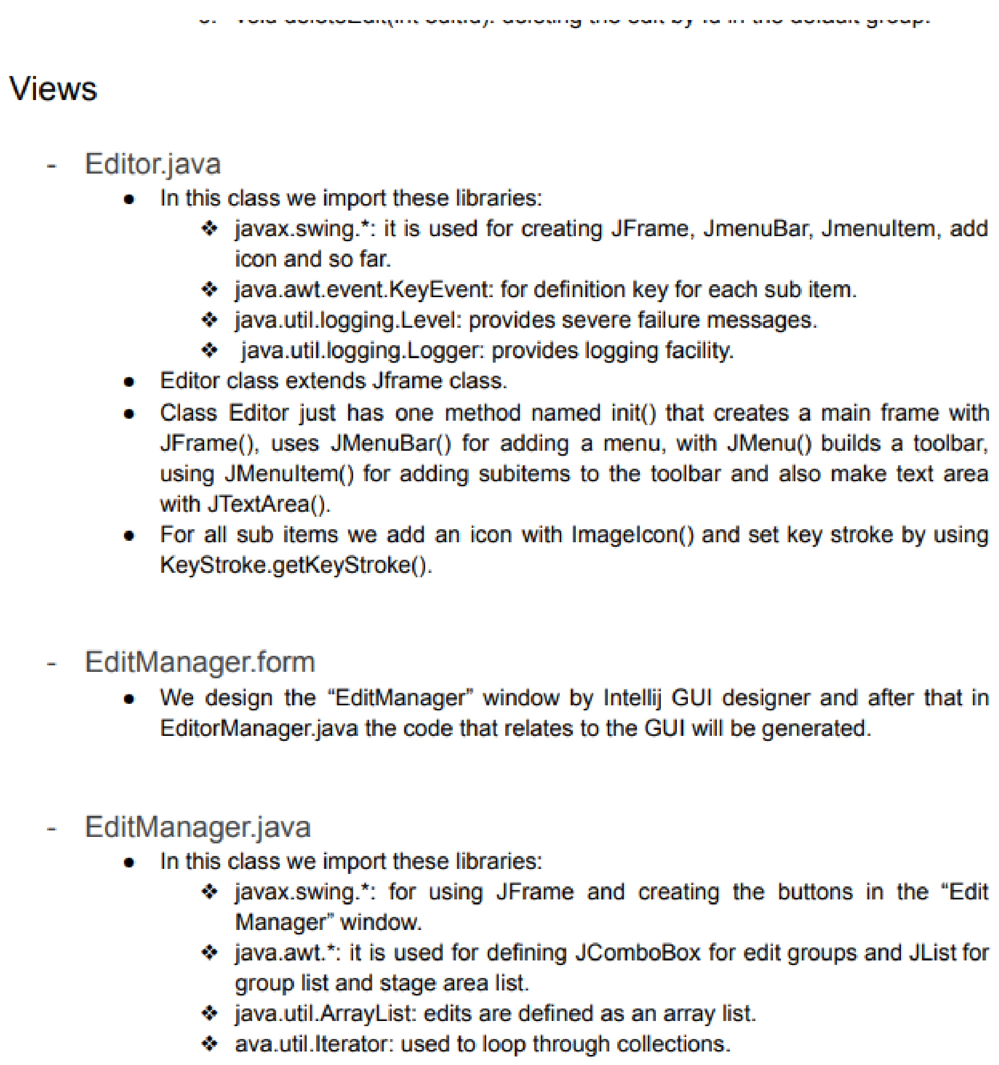
Default

- App.java
  - This class is for bootstrapping the application and import views.Editor

Model

- EditModel.java
  - This class has one constructor that initializes the content of the edit, sets "false" for variable undone and sets the Id of the group.
  - Class EditModel has these methods:
    1. String getContent(): return the edit content.
    2. void setContent(String content): the input is the edit content and updates the edit content.
    3. Boolean getUndone(): return the status of the given edit (it marked as undone or not).
    4. void setUndone(Boolean undone): the input is the new status and it updates the edit's status.
    5. Integer getId(): return the Id of the group.
    6. void setId(): sets the Id of the group.
    7. void toggleStatus(): Changes the edit's status from done to undone and vice-versa.
    8. String toString(): for printing some information that related to the edits like ID, content and status.

4. Implementation



void deleteEdit(int editId): deleting the edit by id in the default group.

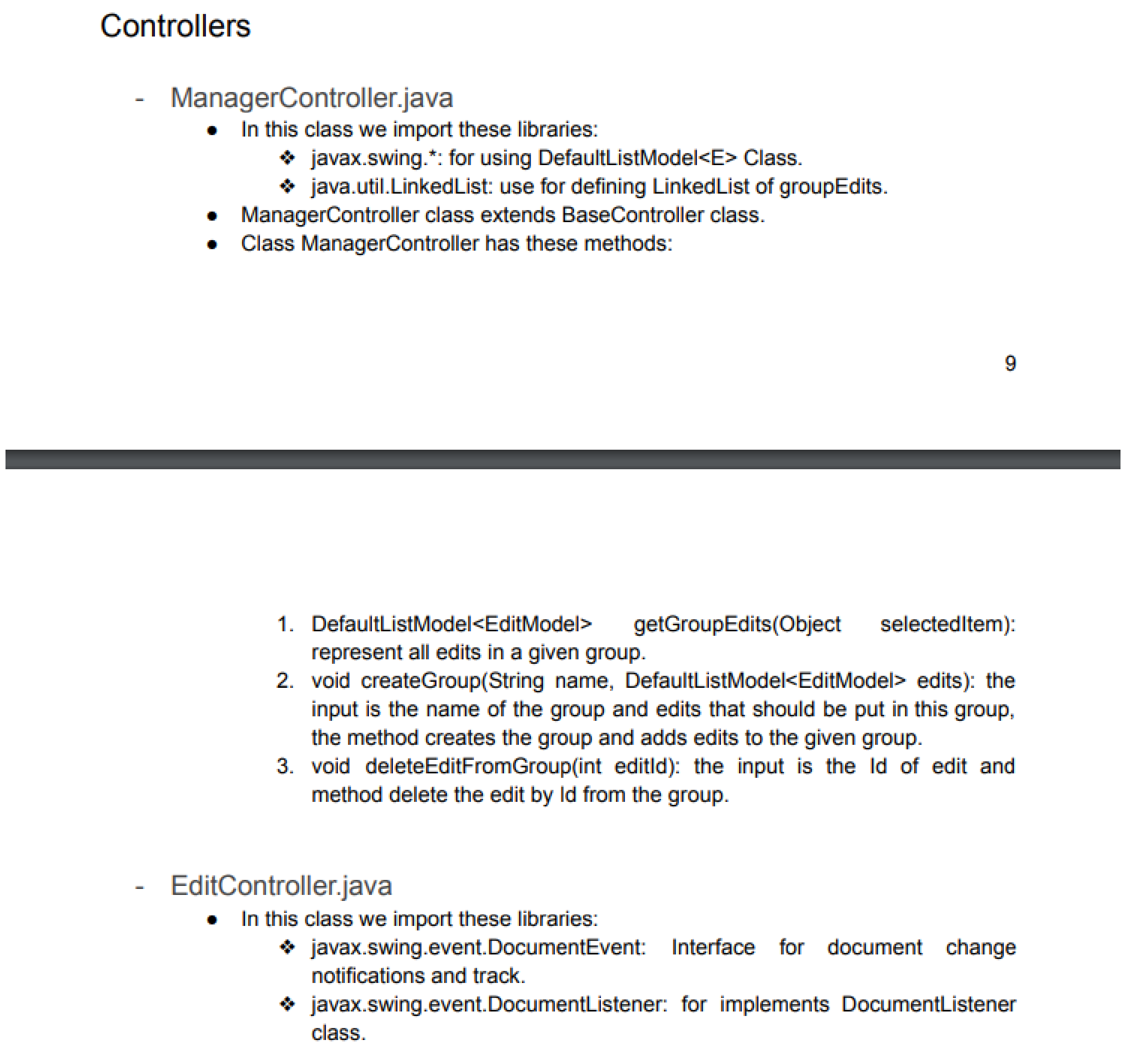
Views

- Editor.java
  - In this class we import these libraries:
    - ❖ javax.swing.*: it is used for creating JFrame, JmenuBar, JmenuItem, add icon and so far.
    - ❖ java.awt.event.KeyEvent: for definition key for each sub item.
    - ❖ java.util.logging.Level: provides severe failure messages.
    - ❖ java.util.logging.Logger: provides logging facility.
  - Editor class extends Jframe class.
  - Class Editor just has one method named init() that creates a main frame with JFrame(), uses JMenuBar() for adding a menu, with JMenu() builds a toolbar, using JMenuItem() for adding subitems to the toolbar and also make text area with JTextArea().
  - For all sub items we add an icon with ImageIcon() and set key stroke by using KeyStroke.getKeyStroke().

- EditManager.form
  - We design the “EditManager” window by Intellij GUI designer and after that in EditorManager.java the code that relates to the GUI will be generated.

- EditManager.java
  - In this class we import these libraries:
    - ❖ javax.swing.*: for using JFrame and creating the buttons in the “Edit Manager” window.
    - ❖ java.awt.*: it is used for defining JComboBox for edit groups and JList for group list and stage area list.
    - ❖ java.util.ArrayList: edits are defined as an array list.
    - ❖ ava.util.Iterator: used to loop through collections.

4. Implementation



Controllers

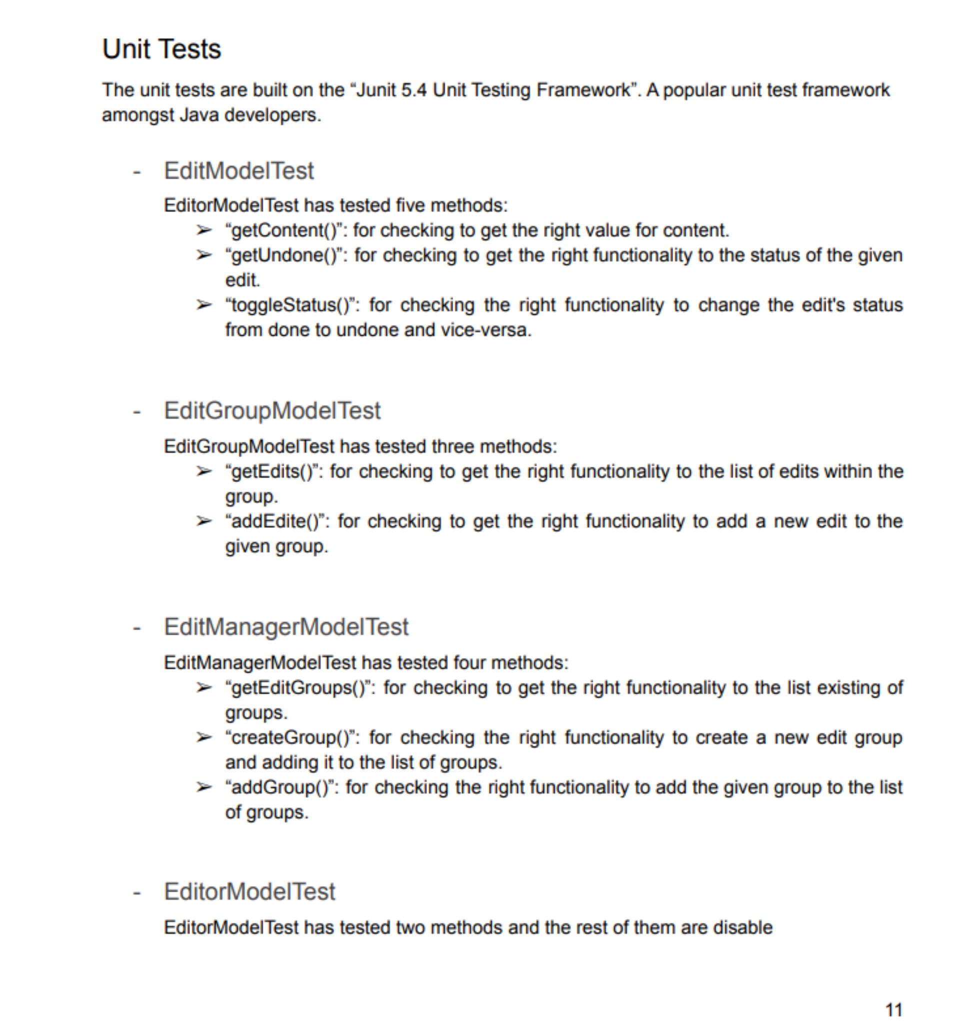
- ManagerController.java
  - In this class we import these libraries:
    - ❖ javax.swing.*: for using DefaultListModel<E> Class.
    - ❖ java.util.LinkedList: use for defining LinkedList of groupEdits.
  - ManagerController class extends BaseController class.
  - Class ManagerController has these methods:

9

1. DefaultListModel<EditModel> getGroupEdits(Object selectedItem): represent all edits in a given group.
2. void createGroup(String name, DefaultListModel<EditModel> edits): the input is the name of the group and edits that should be put in this group, the method creates the group and adds edits to the given group.
3. void deleteEditFromGroup(int editId): the input is the Id of edit and method delete the edit by Id from the group.

- EditController.java
  - In this class we import these libraries:
    - ❖ javax.swing.event.DocumentEvent: Interface for document change notifications and track.
    - ❖ javax.swing.event.DocumentListener: for implements DocumentListener class.

5. Testing Strategy



Unit Tests

The unit tests are built on the "Junit 5.4 Unit Testing Framework". A popular unit test framework amongst Java developers.

- EditModelTest

  EditorModelTest has tested five methods:
  - ➢ "getContent()": for checking to get the right value for content.
  - ➢ "getUndone()": for checking to get the right functionality to the status of the given edit.
  - ➢ "toggleStatus()": for checking the right functionality to change the edit's status from done to undone and vice-versa.

- EditGroupModelTest

  EditGroupModelTest has tested three methods:
  - ➢ "getEdits()": for checking to get the right functionality to the list of edits within the group.
  - ➢ "addEdite()": for checking to get the right functionality to add a new edit to the given group.

- EditManagerModelTest

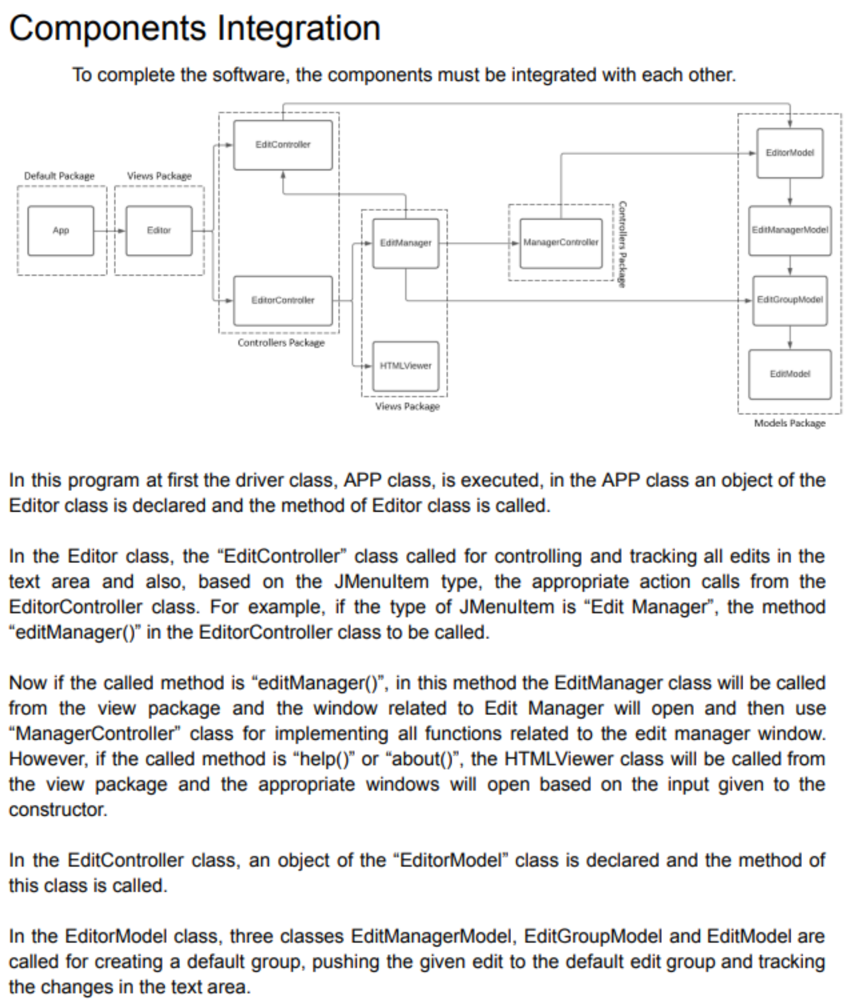
  EditManagerModelTest has tested four methods:
  - ➢ "getEditGroups()": for checking to get the right functionality to the list existing of groups.
  - ➢ "createGroup()": for checking the right functionality to create a new edit group and adding it to the list of groups.
  - ➢ "addGroup()": for checking the right functionality to add the given group to the list of groups.

- EditorModelTest

  EditorModelTest has tested two methods and the rest of them are disable

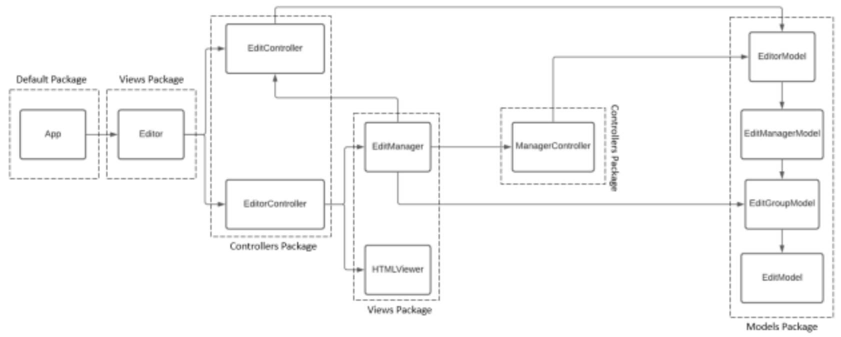
11

5. Testing Strategies



Components Integration

To complete the software, the components must be integrated with each other.



In this program at first the driver class, APP class, is executed, in the APP class an object of the Editor class is declared and the method of Editor class is called.

In the Editor class, the "EditController" class called for controlling and tracking all edits in the text area and also, based on the JMenuItem type, the appropriate action calls from the EditorController class. For example, if the type of JMenuItem is "Edit Manager", the method "editManager()" in the EditorController class to be called.

Now if the called method is "editManager()", in this method the EditManager class will be called from the view package and the window related to Edit Manager will open and then use "ManagerController" class for implementing all functions related to the edit manager window. However, if the called method is "help()" or "about()", the HTMLViewer class will be called from the view package and the appropriate windows will open based on the input given to the constructor.

In the EditController class, an object of the "EditorModel" class is declared and the method of this class is called.

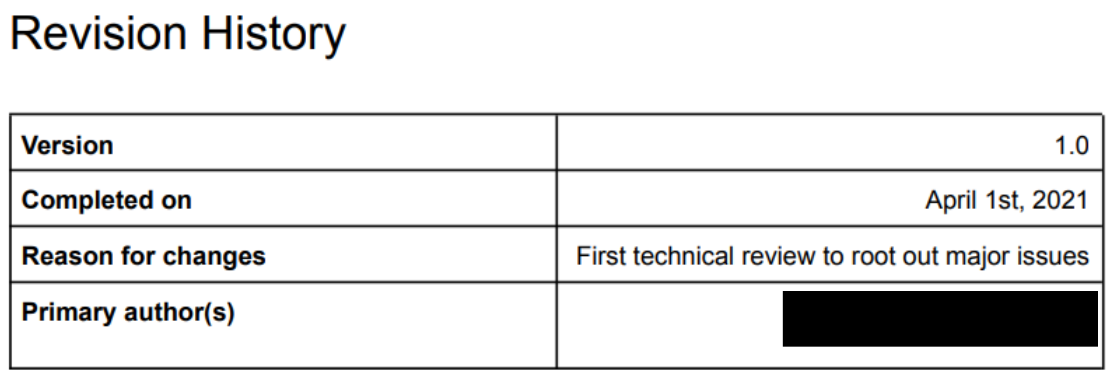
In the EditorModel class, three classes EditManagerModel, EditGroupModel and EditModel are called for creating a default group, pushing the given edit to the default edit group and tracking the changes in the text area.

7. Quality Control

The example doesn’t contain this part but you do need to mention it in your project report.

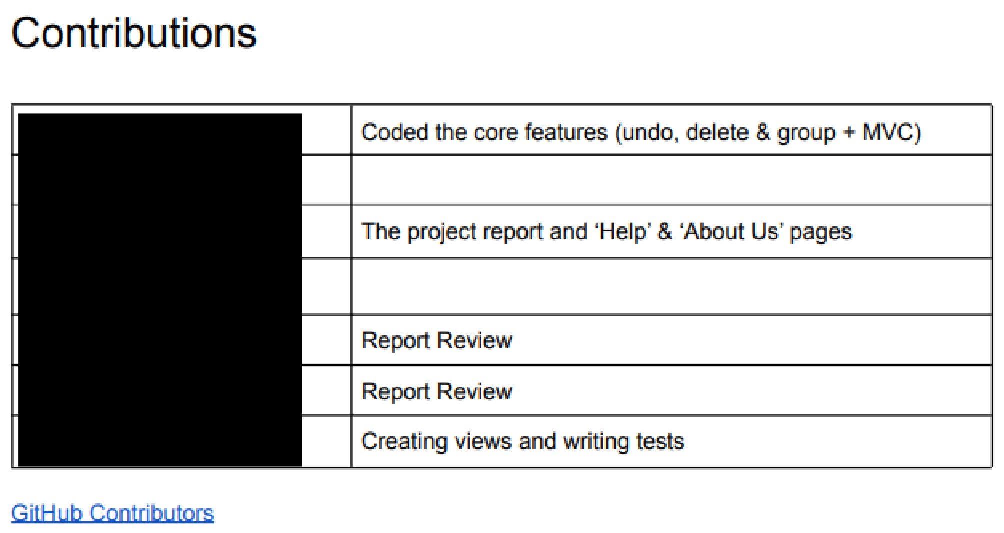
Discuss the changes to the requirements, designs and test cases to ensure the quality of your design software.

8. Revision History and Contributions



Revision History

Version	1.0
Completed on	April 1st, 2021
Reason for changes	First technical review to root out major issues
Primary author(s)	[redacted]



Contributions

[redacted]		Coded the core features (undo, delete & group + MVC)
[redacted]		
[redacted]		The project report and 'Help' & 'About Us' pages
[redacted]		
[redacted]		Report Review
[redacted]		Report Review
[redacted]		Creating views and writing tests

GitHub Contributors