**Plan the Application**

1. **Define Features:**
   * Create a text area for typing and editing text.
   * Add a menu bar with "File" and "Edit" menus.
   * Include options like New, Open, Save, Print, Cut, Copy, and Paste.
   * Add a "Save and Submit" button to save and close the file.
2. **Choose Technologies:**
   * Use **Java Swing** for the GUI (Graphical User Interface).
   * Use **Java AWT** for event handling and basic window management.

**2. Set Up the Project**

1. **Create a Java Project:**
   * Set up a new Java project in your preferred IDE (e.g., IntelliJ, Eclipse, or NetBeans).
2. **Create the Main Class:**
   * Create a class (e.g., TextEditor) to serve as the entry point for the application.

**3. Design the User Interface**

1. **Create the Main Window:**
   * Use a JFrame to create the main application window.
   * Set the title, size, and default close operation for the window.
2. **Add a Text Area:**
   * Use a JTextArea to allow users to input and edit text.
   * Wrap the JTextArea in a JScrollPane to enable scrolling for large texts.
3. **Create a Menu Bar:**
   * Use JMenuBar to create a menu bar at the top of the window.
   * Add two main menus: "File" and "Edit".
4. **Add Menu Items:**
   * Under the "File" menu, add items like New, Open, Save, and Print.
   * Under the "Edit" menu, add items like Cut, Copy, and Paste.
5. **Add a "Save and Submit" Button:**
   * Use a JButton to create a button that saves the file and closes the application.

**4. Implement Functionality**

1. **Handle File Operations:**
   * **New:** Clear the text area to start a new document.
   * **Open:** Use a file chooser (JFileChooser) to open a text file and load its content into the text area.
   * **Save:** Use a file chooser to save the content of the text area to a file.
   * **Print:** Implement functionality to print the text area content.
2. **Handle Edit Operations:**
   * **Cut:** Remove the selected text and copy it to the clipboard.
   * **Copy:** Copy the selected text to the clipboard.
   * **Paste:** Insert the clipboard content at the cursor position.
3. **Handle the "Save and Submit" Button:**
   * Save the current text to a file (if not already saved).
   * Close the application after saving.
4. **Use Action Listeners:**
   * Attach ActionListener to menu items and buttons to handle user actions (e.g., clicking "Save" or "Cut").

**5. Test the Application**

1. **Test File Operations:**
   * Verify that New, Open, Save, and Print work as expected.
   * Ensure files are correctly loaded and saved.
2. **Test Edit Operations:**
   * Verify that Cut, Copy, and Paste work correctly.
   * Test edge cases like copying without selecting text or pasting without content in the clipboard.
3. **Test the "Save and Submit" Button:**
   * Ensure the application saves the file and closes properly.

**6. Refine and Improve**

1. **Add Error Handling:**
   * Handle exceptions (e.g., file not found, permission issues) gracefully.
   * Display user-friendly error messages.
2. **Enhance the UI:**
   * Add icons to menu items for better visual appeal.
   * Customize the font and color of the text area.
3. **Add Additional Features (Optional):**
   * Add a "Find and Replace" feature.
   * Support for multiple tabs or documents.
   * Add keyboard shortcuts (e.g., Ctrl+S for Save).

**7. Document the Project**

1. **Write a User Guide:**
   * Explain how to use the text editor (e.g., how to open, save, and edit files).
2. **Write a Developer Guide:**
   * Document the code structure, classes, and methods.
   * Include instructions for setting up and running the project.

**8. Package and Distribute**

1. **Create an Executable JAR:**
   * Package the application into a JAR file for easy distribution.
2. **Test the Executable:**
   * Ensure the JAR file runs correctly on different systems.