Group symbol: **3**

Team: **3**

Project title: **A fast, lightweight and intuitive text-editor**

Team members (*filled by PM, Team Leader*):

No	Name	Surname	Student ID	Role
1	Gabriele	Simoni	293981	*PM, Team Leader*
2	Hüdalfa Bera	Dalgın	293988	*Team member*
3	Nozomi Malke	Shirasaki	288599	*Team member*
4	Erik	Parra Mejido	293864	*Team member*
5	Cédric Minh	Prétet	293891	*Team member*

1. Elaboration of application concept (S1)

1.1. Project (business) goals

Main Goal

The primary business goal of releasing the text editor as open source is to establish an initial presence within the software ecosystem. By providing a reliable, visible tool, the project demonstrates technical capability and commitment to quality, building credibility and recognition among developers and potential collaborators. This initial positioning lays the groundwork for future opportunities and engagement without relying on immediate commercial returns.

1. User Adoption

A key sub-goal is achieving widespread adoption among developers and general users. Broad usage increases visibility, attracts contributions, and drives community engagement. By prioritizing usability, performance, and cross-platform support, the project encourages continuous use and helps establish the text editor as a recognized and trusted tool within the target audience.

2. Transparency and Open Source

As an open-source initiative, the project will foster collaboration, peer review, and community driven improvement. The source code will be freely accessible, allowing developers and users to inspect, modify, and redistribute it under a permissive license. This openness will ensure long-term sustainability, encourage innovation, and build trust through shared ownership and accountability.

3. Ecosystem Integration

The text editor is designed to integrate with existing developer workflows, plugins, and extensions, creating a foundation for broader ecosystem participation. This integration allows the software to become a platform upon which other tools and services can be built, increasing its strategic value and potential for future commercial opportunities while maintaining the core product as open and freely available.

1.2. Identification of project's internal and external Stakeholders

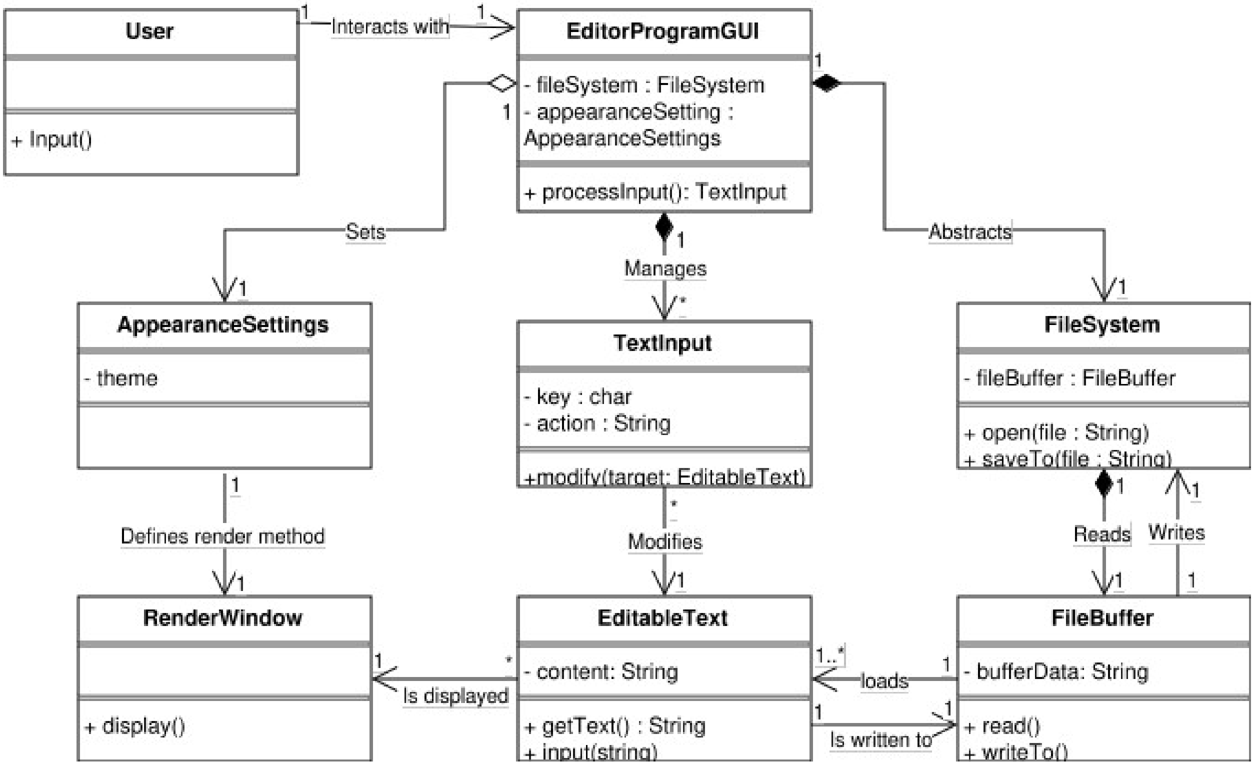
- **Internal Stakeholders**

Symbol	Name	Role	Description
1	Gabriele Simoni	Project Manager	Plans tasks, coordinates the team, monitors progress and contributes to development
2	Hüdalfa Bera Dalgın	Team Member	Works on tasks assigned by PM, contributes to development, testing, or documentation as needed.
3	Nozomi Malke Shirasaki	Team Member	Works on tasks assigned by PM, contributes to development, testing, or documentation as needed.
4	Erik Parra Mejido	Team Member	Works on tasks assigned by PM, contributes to development, testing, or documentation as needed.
5	Cédric Minh Prétet	Team Member	Works on tasks assigned by PM, contributes to development, testing, or documentation as needed.

- **External Stakeholders**

Symbol	Name	Role	Description
1	Computer users	End users	Generic computer users, ranging from casual users to programmers who may use the program to write text files, edit configuration files or write code.
2	Open source community	Community	The program may be adopted by open source projects as part of toolchains. The program may help expand the set of open source tools available.
3	Future contributors	Expand the project	Individuals who may contribute to the project's future development to fix bugs and add features to gain experience.

1.3. Domain description



- **Entities**

Entities	Description
User	The one who controls the application. Interaction is done only through the Editor Program GUI.
Editor Program GUI	The primary controller for the application. It processes GUI interactions and keyboard inputs from the user.
File System	File manipulation is done through this system. It opens files to read and edit. Several files can be open at the same time and this system keeps track of them.
File Buffer	The content of each open file is loaded in a file buffer. When saving to a file, the data stored in the file buffer will be written to the corresponding file.
Text Input	Processed from the Editor Program GUI, text input is taken to use to edit an editable text.

Editable Text	This is where the content of each file is stored and edited in the application. Each file has a corresponding Editable Text to use.
Appearance Settings	Controls the way that the application renders the text editor window based on saved settings.
Render Window	This is the section of the application which displays the text to be edited. It takes into account the appearance settings and the syntax highlighting when displaying known file types.

- ## Relationships

Relationships	**Name**	**Description**
User -> Editor Program GUI	Interacts	All user input is processed through the Editor Program GUI.
Editor Program GUI -> File System	Abstracts	The file system isolates the file related tasks from the Editor Program GUI to create a more organized set up.
File System -> File Buffer	Reads	Data from opened files are read into the File Buffer.
File Buffer -> File System	Writes	Data from the File Buffer is written into a file opened in the file system.
Editor Program GUI -> Text Input	Manages	Input is processed then only the text input is passed to be used to edit the focused text.
Text Input -> Editable Text	Modifies	The text input is used to edit the Editable Text that has focus.
Editor Program GUI -> Appearance Settings	Sets	The application sets the settings for the appearance of the application window.
Appearance Settings -> Render Window	Defines render method	The appearance settings stored in the application are used to initialize the appearance of the window.
File Buffer -> Editable Text	Loads	The content of an Editable Text is loaded from the File Buffer.
Editable Text -> File Buffer	Is written to	When saving a file, the content of an Editable Text is written to the File Buffer to prepare to write to the

		corresponding file.
Editable Text -> Render Window	Is displayed in	The text content from an Editable Text is displayed in the Render Window.

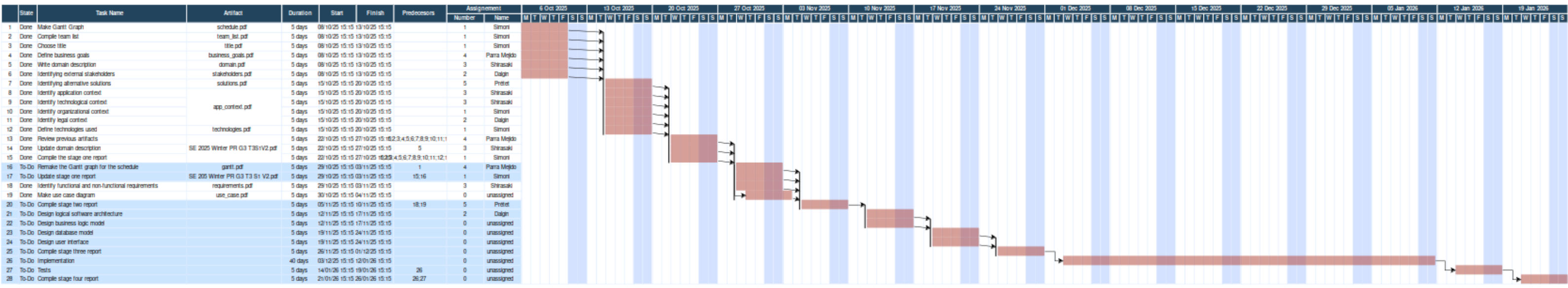
- ## Events

Events	Description
File Opened	Triggered when the user selects a file to open through the Editor Program GUI. The File System will then read the contents of the file to the buffer and assign an Editable Text to hold the contents.
File Saved	Triggered when the SaveButton is pressed. Overwrites the contents of the TextBox onto the selected file.
Text Edited	Change the contents of the focused Editable Text based on the user's processed text input.
Document Closed	Remove the corresponding Editable Text and any associated data.
Settings Changed	Save the new settings to a file and render the window again with the new settings.

1.4. Project schedule (Gantt chart)

The project schedule is regularly updated in a shared Google Drive spreadsheet.
It's accessible through the following link:
https://docs.google.com/spreadsheets/d/1ZAp8YltL8X0dDxpcvXPFH82-Os02tLYX/edit?usp=drive_link&ouid=107940987460351811396&rtpof=true&sd=true

The following is a current snapshot of the project schedule:



	State	Task Name	Artifact	Duration	Start	Finish	Predecesors	Assignement	
								Number	Name
1	Done	Make Gantt Graph	schedule.pdf	5 days	08/10/25 15:15	13/10/25 15:15		1	Simoni
2	Done	Compile team list	team_list.pdf	5 days	08/10/25 15:15	13/10/25 15:15		1	Simoni
3	Done	Choose title	title.pdf	5 days	08/10/25 15:15	13/10/25 15:15		1	Simoni
4	Done	Define business goals	business_goals.pdf	5 days	08/10/25 15:15	13/10/25 15:15		4	Parra Mejido
5	Done	Write domain description	domain.pdf	5 days	08/10/25 15:15	13/10/25 15:15		3	Shirasaki
6	Done	Identifying external stakeholders	stakeholders.pdf	5 days	08/10/25 15:15	13/10/25 15:15		2	Dalgin
7	Done	Identifying alternative solutions	solutions.pdf	5 days	15/10/25 15:15	20/10/25 15:15		5	Prétet
8	Done	Identify application context	app_context.pdf	5 days	15/10/25 15:15	20/10/25 15:15		3	Shirasaki
9	Done	Identify technological context		5 days	15/10/25 15:15	20/10/25 15:15		3	Shirasaki
10	Done	Identify organizational context		5 days	15/10/25 15:15	20/10/25 15:15		1	Simoni
11	Done	Identify legal context		5 days	15/10/25 15:15	20/10/25 15:15		2	Dalgin
12	Done	Define technologies used	technologies.pdf	5 days	15/10/25 15:15	20/10/25 15:15		1	Simoni
13	Done	Review previous artifacts	SE 2025 Winter PR G3 T3S1V2.pdf	5 days	22/10/25 15:15	27/10/25 15:15	1;2;3;4;5;6;7;8;9;10;11;1	4	Parra Mejido
14	Done	Update domain description		5 days	22/10/25 15:15	27/10/25 15:15	5	3	Shirasaki
15	Done	Compile the stage one report		5 days	22/10/25 15:15	27/10/25 15:15	1;2;3;4;5;6;7;8;9;10;11;12;1	1	Simoni
16	To-Do	Remake the Gantt graph for the schedule	gantt.pdf	5 days	29/10/25 15:15	03/11/25 15:15	1	4	Parra Mejido
17	To-Do	Update stage one report	SE 205 Winter PR G3 T3 S1 V2.pdf	5 days	29/10/25 15:15	03/11/25 15:15	15;16	1	Simoni
18	Done	Identify functional and non-functional requirements	requirements.pdf	5 days	29/10/25 15:15	03/11/25 15:15		3	Shirasaki
19	Done	Make use case diagram	use_case.pdf	5 days	30/10/25 15:15	04/11/25 15:15		0	unassigned
20	To-Do	Compile stage two report		5 days	05/11/25 15:15	10/11/25 15:15	18;19	5	Prétet
21	To-Do	Design logical software architecture		5 days	12/11/25 15:15	17/11/25 15:15		2	Dalgin
22	To-Do	Design business logic model		5 days	12/11/25 15:15	17/11/25 15:15		0	unassigned
23	To-Do	Design database model		5 days	19/11/25 15:15	24/11/25 15:15		0	unassigned
24	To-Do	Design user interface		5 days	19/11/25 15:15	24/11/25 15:15		0	unassigned
25	To-Do	Compile stage three report		5 days	26/11/25 15:15	01/12/25 15:15		0	unassigned
26	To-Do	Implementation		40 days	03/12/25 15:15	12/01/26 15:15		0	unassigned
27	To-Do	Tests		5 days	14/01/26 15:15	19/01/26 15:15	26	0	unassigned
28	To-Do	Compile stage four report		5 days	21/01/26 15:15	26/01/26 15:15	26;27	0	unassigned

1.5. Identification of existing or alternative solutions

Alternatives	**Main features**	**Advantages**	**Disadvantages**
Visual Studio Code	• Syntax Highlighting • Extensive plugin ecosystem • Cross-platform compatibility • Integrated terminal and Git support • Customizable interface	• Highly extensible through extensions • Active open-source community • Frequent updates and feature improvements • Integrated debugging and development tools	• Built on Electron, leading to higher memory usage • Slower performance compared to lightweight editors
Sublime Text	• Syntax Highlighting • Cross-platform support • Extensive plugin system via Package Control • Fast startup and performance	• Extremely fast and responsive • Rich ecosystem of plugins • Highly customizable	• Closed source, limiting community involvement • Paid license after trial period
Notepad++	• Syntax Highlighting • Open source • Plugin support • Lightweight and fast • Windows-focused	• Lightweight and efficient • Active open-source community • Supports many programming languages	• Limited native support on non-Windows platforms • Outdated UI compared to modern editors
Atom	• Syntax Highlighting • Cross-platform • Plugin and theme support • GitHub integration	• Open-source and hackable • Good for customization and prototyping	• Discontinued since 2022 • Slower performance (Electron-based)

		• Rich plugin ecosystem	• No longer actively maintained
Windows Notepad	• Basic text editing • Included with Windows • Fast startup	• Very lightweight and simple to use • Instant startup time	• Closed source, limiting community engagement • No syntax highlighting or advanced features • Windows-only
Gedit	• Syntax Highlighting • Plugin support • Open source • Cross-platform (Linux, Windows, macOS)	• Simple and user-friendly • Open-source and community-driven • Integrates well with GNOME desktop	• Lacks advanced developer tools • Not as customizable as other editors
Kate	• Syntax Highlighting • Multiple document interface • Plugin support • Open source • Cross-platform	• Feature-rich for an open-source editor • Strong community support • Good performance	• Interface may be overwhelming for beginners • Less popular on Windows and macOS
Obsidian	• Markdown-based text editing • Cross-platform (Windows, macOS, Linux, mobile) • Plugin and theme support • Graph view for linking notes • Offline support	• Powerful organization and linking features • Large and active user community • Plugin system allows customization • Excellent for notetaking and knowledge	• Closed source (limits community engagement) • Heavier and slower startups than lightweight editors • Focused more on note management

		management	rather than pure text editing
Notion	• Online collaborative editor • Cross-platform (web, desktop, mobile) • Rich-text editing with databases, tables, and media embedding • Real-time synchronization and collaboration	• Excellent for structured content and team collaboration • Clean and modern interface • Integrates with many external services	• Closed source requires an internet connection for most use cases • Slower and heavier than traditional text editors • Not optimized for code or plain-text editing
VIM	• Modal text editing (normal, insert, visual, command modes) • Syntax Highlighting • Cross-platform (Linux, macOS,Windows) • Highly configurable via scripts and plugins • Open-source and terminalbased	• Extremely lightweight and fast • Fully open source with a large, active community • Runs efficiently even on low-end systems • Highly customizable and extensible	• Steep learning curve for new users • Minimal GUI (primarily terminal-based) • Configuration and plugin management can be complex

1.6. Project context

Application Context

- *Operating System*

  Windows, macOS and Linux. The OS will act as a bridge between the application and the hardware. It will handle memory access, file system interaction and user Input/Output.

- *User environment*

  Single-user desktop session with keyboard, mouse and display interfaces, allowing the user to interact with the program.

- *File system*

  Storage for creating, reading and editing text files.

Technological Context

- *File systems*

  NTFS, ext4, APFS or other storage technologies the editor reads from and writes to.

- *Graphical subsystems*

  Libraries provided by the operating systems to allow the program to integrate with the system graphical interfaces. These are libraries like X11 on Linux and GDI on Windows.

- *Text encoding*

  Standards used to encode text documents. The program will have to adapt this technology to correctly display the text. Encodings like ASCII, UTF-8 and UTF-16.

- *Input/Output mechanisms*

  Keyboard, mouse, display interfaces managed by the operating system and accessed by the editor.

Legal Context

1. Licensing

The Text Editor project will be distributed under the GNU General Public License (GPL v3). This license allows users to freely use, study, modify, and redistribute the software, ensuring that all versions remain open source. By adopting the GPL, we aim to support transparency, collaboration, and innovation within the open-source community.

2. Intellectual Property

All members of the project team are joint owners of the original codebase and resources created for this project. Once the software is released under the GPL license, it becomes open to public contribution. Any modifications or extensions made by external contributors must remain under the same GPL terms and include proper credit to the original authors.

3. Compliance and Ethical Use

The team commits to adhering to both legal and academic standards throughout the development process. No copyrighted or proprietary materials will be used without authorization, and any external dependencies included will be compatible with the GPL license. We also commit to ethical software engineering practices, fostering transparency and respect for intellectual property rights.

4. Liability Disclaimer

In compliance with GPL v3, the software will be provided 'as is', without any warranty. The development team is not responsible for any potential damage, data loss, or issues that may arise from its use. Users assume full responsibility when utilizing the application.

1.7. Technologies used in the project

1.7 Technologies used

Name	Description	Justification	Responsibilities	Website
C Programming Language	C is a general-purpose programming language. It provides low-level memory access, structured programming constructs, and is widely used for system software, embedded systems, and performance-critical applications.	C is widely adopted for open-source projects for its portability. Its efficiency aligns with our goal to deliver a text-editor with optimal performance.	• Defining software logic • Managing memory • Facilitating interaction between hardware and other software components	https://en.wikipedia.org/wiki/C_(programming_language)
GNU Toolchain	The GNU Toolchain is a collection of open-source programming tools used to develop software.	It provides a reliable set of tools. The ones concerning our project are the C compiler (GCC), the debugger (GDB) and the build system (Make). It makes our project more accessible to open-source contributors, who are already familiar with the toolkit.	• Compiling source code into executables • Debugging applications • Automating the build and installation process	https://en.wikipedia.org/wiki/GNU_toolchain
GTK	GTK is an open-source library used for creating applications that have graphical user interfaces. It supports multiple platforms and languages.	It allows for cross-platform compatibility, good performance and is open-source.	• Handling user interaction with the application • Integrating GUI with the application logic	https://www.gtk.org/
GitHub	GitHub is a web-based platform for version control and collaborative software development using Git. It allows developers to host repositories, track changes and collaborate in teams.	GitHub allows for collaborative development. Because Github is widely recognized in the software development community, hosting a project there increases its visibility.	• Hosting the project code • Tracking version history • Facilitating team collaboration • Tracking team members contributions to the code	https://github.com/about
Draw.io	Draw.io is a diagramming tool for creating flowcharts and UML diagrams.	It provides a simple and free way to create diagrams. It is open-source.	• Offering a simple and effective way to design diagrams	https://www.drawio.com/about
LibreOffice Writer	LibreOffice Writer is a free and open-source word processor.	It's free and open-source. It has all the features of a modern word processor.	• Allowing the team to create, edit, and format text documents, including reports, artifacts and technical documentation.	https://www.libreoffice.org/discover/writer/