Group symbol: **3**

Team: **3**

Project title: **wrtext**

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*

2. Requirements specification **(S2)**

2.1. Functional Requirements Specification

In this section, provide the table for functional requirements, including symbol, type (e.g. business logic, user interface, data exchange, etc.) description, significance (MoSCoW) and source (Stakeholder).

Understanding: Functional requirements describe what the system should do like its *behaviors, features, and interactions*
They define how the system responds to inputs, processes data, and produces results.

Symbol	Type	Description	Significance	Source
FR1	User Interface	The editor shall allow users to create, open, edit, and save text files through a graphical interface	Must have	End users
FR4	Business Logic	The system shall maintain an internal buffer for each opened file to track unsaved changes.	Must have	Development team
FR5	Security	The system shall provide error handling to prevent data loss during save operations.	Must have	Development team
FR6	File Handling	The editor shall support opening and saving plain-text files across multiple platforms (Windows, macOS, Linux).	Must have	End users
FR7	User Interaction	The application shall support common editing actions such as undo and redo.	Should have	End users
FR8	Data Management	The editor shall handle multiple files simultaneously using separate buffers.	Should have	End users
FR9	Error Management	The application shall notify users in case of failed operations (e.g., file not found, insufficient permissions).	Should have	End users
FR10	User Interface	The editor shall include a search and replace functionality.	Could have	End users
FR11	Configuration	The editor shall allow basic appearance customization (e.g., theme, font size, word wrap).	Could have	End users

2.2. Non-Functional Requirements

In this section provide the table for non-functional requirements that includes symbol, type (e.g., efficiency, standards, constraints, etc.), description, significance (MoSCoW) for the project, source (Stakeholder). Each requirement should also have specified a verification method – a description of a confirmation method whether a requirement has been fulfilled or not in the most measurable and objective way.

Understanding: Non-functional requirements describe how the system performs its functions ratherthan what those functions are.
They define qualities, constraints, or standards that affect the overall user experience and system performance, such as speed, reliability, usability, or maintainability.

Symbol	Type	Description	Significance	Source	Verification method
NFR1	Efficiency	The application shall load and open files under 2 second for files under 5MB.	Must have	End users	Performance testing
NFR2	Portability	The application shall run natively on Windows, macOS, and Linux with the same core functionality.	Must have	End users	Testing on target OS
NFR3	Reliability	The system shall not lose data in case of unexpected crashes; temporary buffers or autosave features may be used.	Should have	Development team	Stress and fault testing
NFR4	Usability	The interface shall remain simple, intuitive, and easy to navigate for all user types.	Must have	End users	User testing and feedback
NFR5	Maintainability	The source code shall be structured and documented to allow new contributors to easily understand and extend it.	Must have	Open source community	Code review and documentation audit
NFR6	Security	File operations shall be restricted to the user's permissions, avoiding unauthorized file access.	Should have	Development team	Manual testing and code inspection
NFR7	Scalability	The system shall handle up to 10 files open simultaneously without degradation in responsiveness.	Could have	End users	Performance benchmark
NFR8	Compliance	The source code shall be publicly available and modifiable under GPL v3.	Must have	Community	License audit
NFR9	Extensibility	The system shall be structured to support future extensions such as syntax highlighting or plugin integration.	Must have	Development team	Code review
NFR10	Transparency	The project repository shall include changelogs and version history for traceability.	Should have	Open source community	Repository audit

2.3. Use Case Diagram

You should prepare the use case diagram in UML 2.5 depicting the roles of stakeholders who are users of the project. It should also present the high-level concept of system usage divided into modules. Use case diagram should not cover common business logic (e.g. registration and logging in, CRUD operations).