Group symbol: **3**

Team: **3**

Project title: **wrtext**

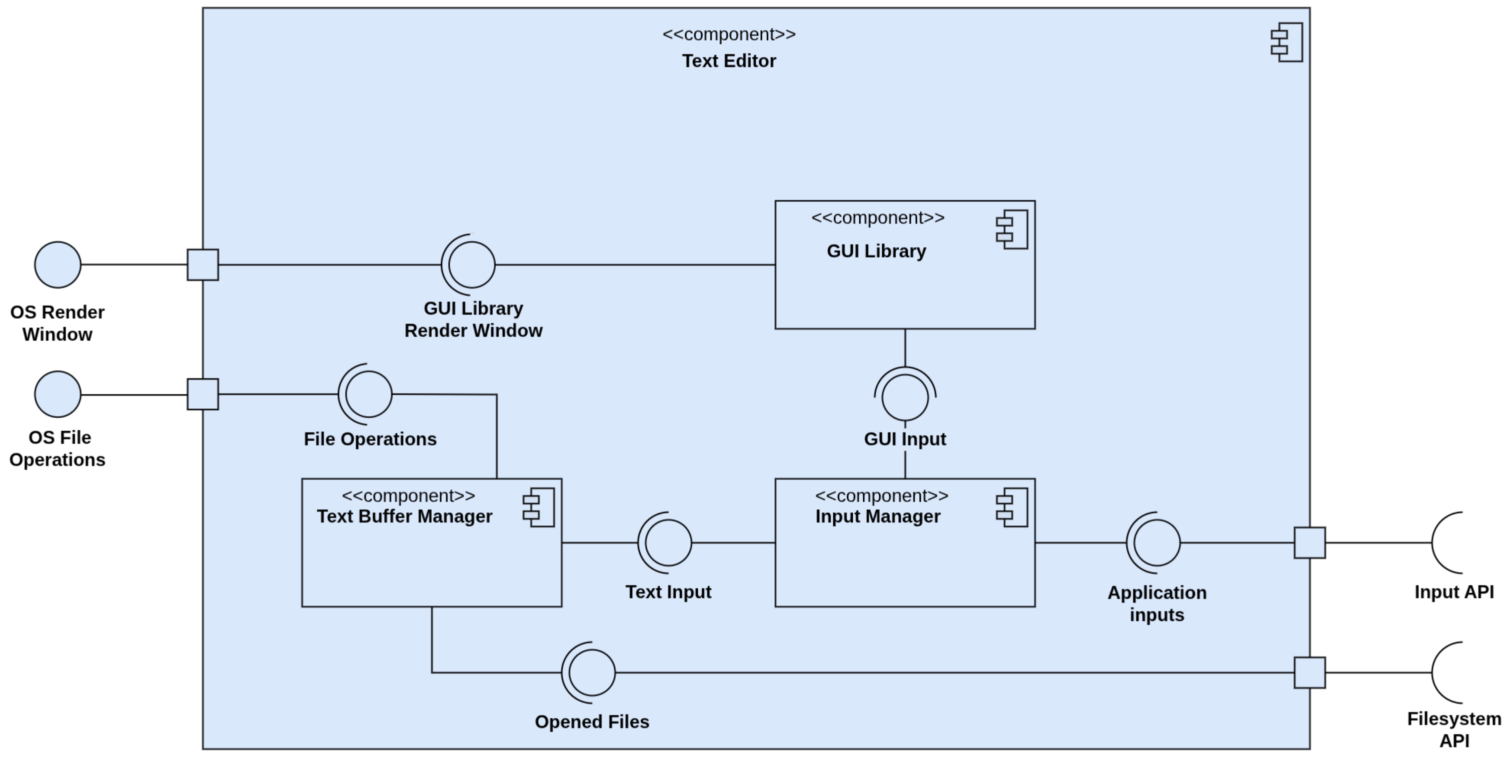
Team members (*filled by PM, Team Leader*):

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*

3. Design **(S3)**

3.1. Logical Software Architecture

The logical architecture aspect of the system should be present as a Component Diagram expressed in UML 2.5.



Provided Interfaces

The Text Editor application must provide two major interfaces to the external system. A Render Window that the operating system can interpret and draw to the screen and a sequence of File Operations the operating system will read to modify and create files.

Required Interfaces

The Text Editor application requires an interface that can provide access to the user inputs registered by the operating system, in the diagram this interface is called “Input API”.
The editor also requires an interface that can allow the application to read files from the filesystem. The name of this interface is “Filesystem API”.

Input Manager

The Input Manager reads user input and dispatches it to other components based on the type of input.

GUI Library

The GUI Library keeps an internal state that is used to create a render window which can be displayed on the screen. User inputs can alter the internal state of the GUI (for example which file is currently being displayed).
The window is first generated according to the GUI Library, and cannot be interpreted by the operating system. The GUI Library can translate it into a window that can be rendered by the operating system.

Text Buffer Manager

The Text Buffer Manager receives user inputs which detail how to modify or create files. It uses the interface to the Filesystem to access files. After processing user inputs, it exposes the file operations that are necessary to modify or create the files according to what the user has edited.

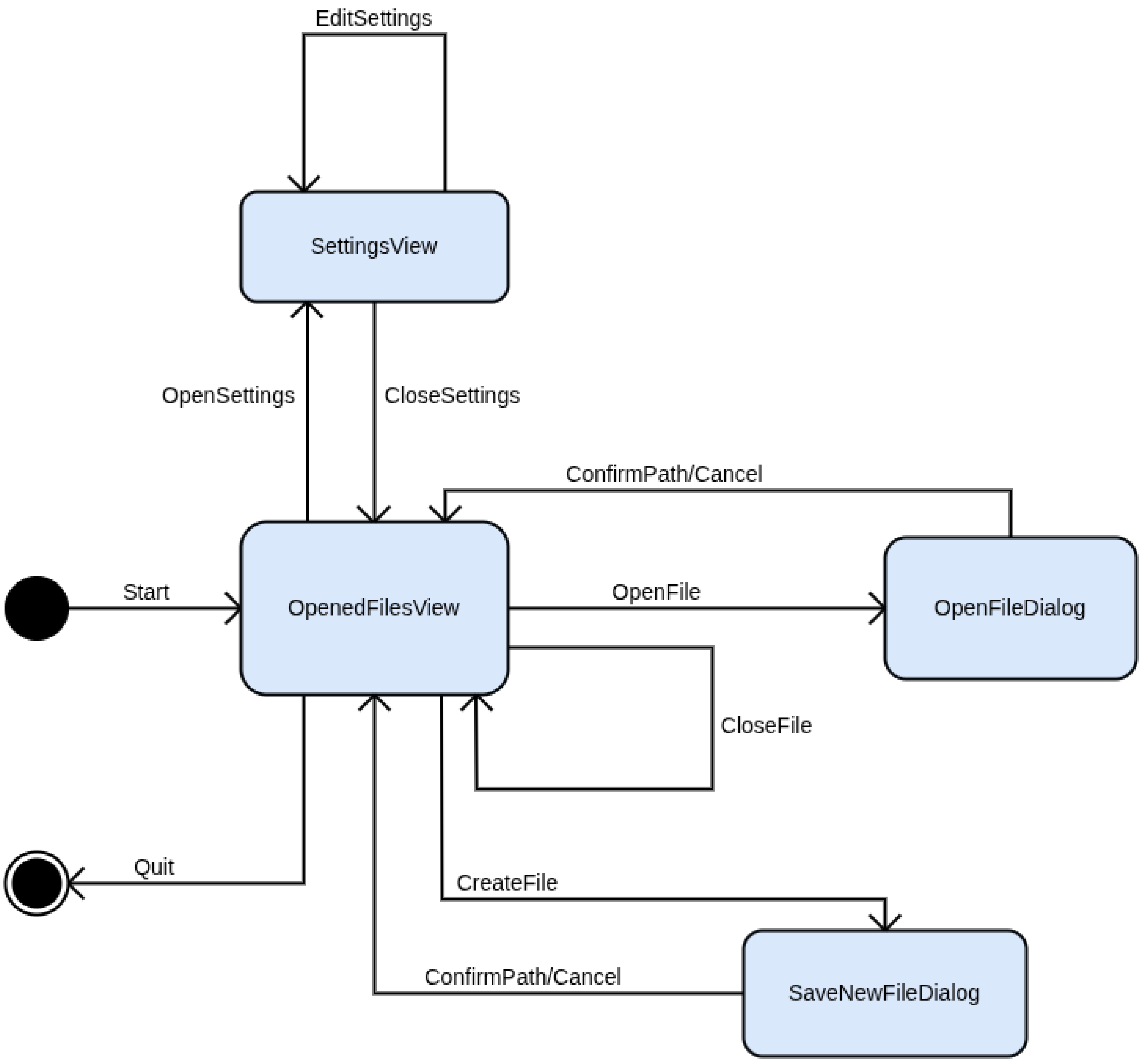
3.2. Business Logic Model

3.2.1. Behavioural Model

In this section present:

- *state diagram for whole the system*
- *behaviour of 2 crucial use cases using the UML 2.5 diagrams, namely sequence diagram, and activity diagram.*

State Diagram



Activity and sequence diagram of the *New File* use case

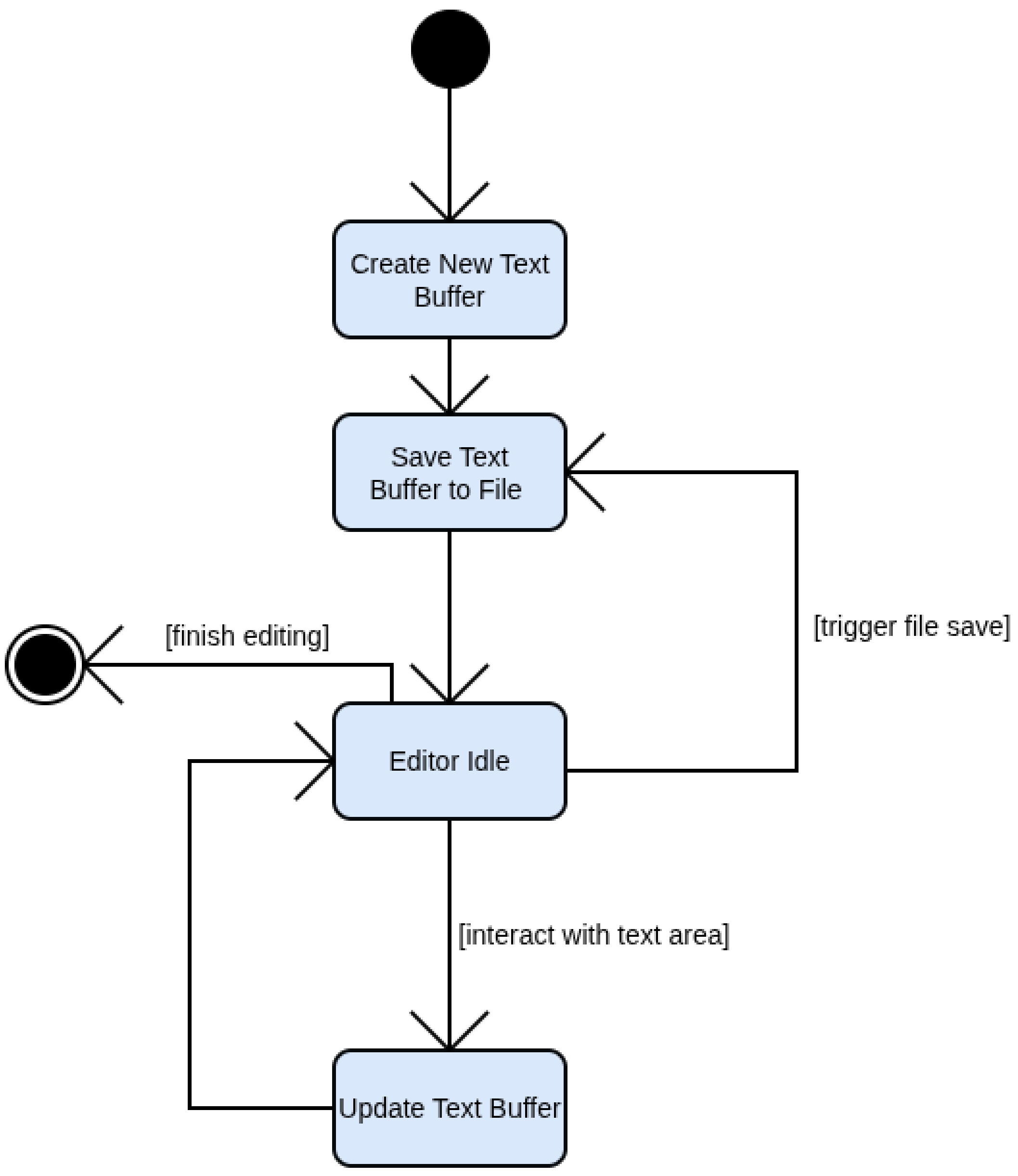
In this section we analyze the behaviour of the application during the New File use case. The process through which a user can create a new file and edit its contents.

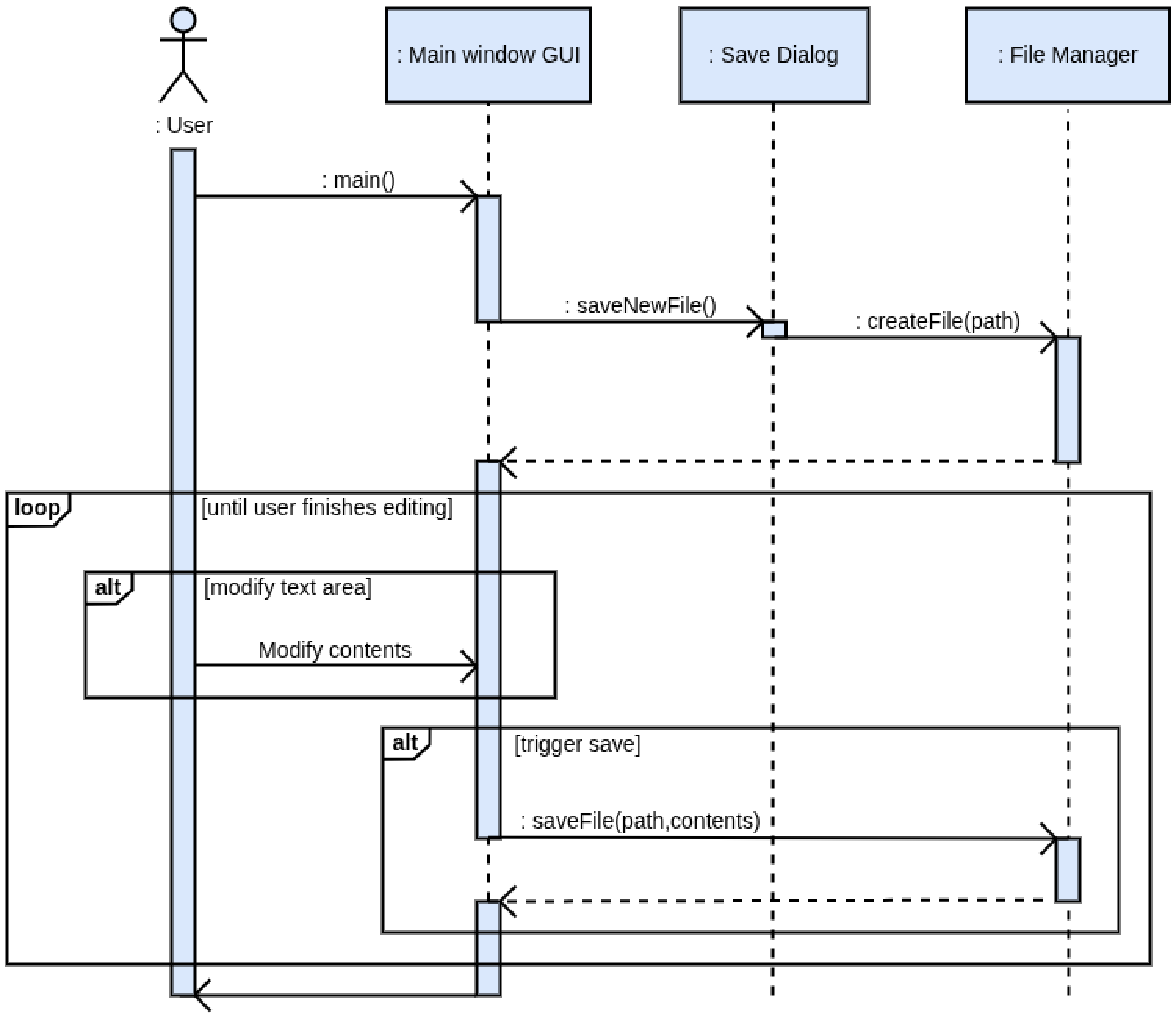
- **Pre-conditions**

  The application must be running.
  The settings window mustn't be open.
  The application must have reading and writing access to the desired saving location.

- **Post-conditions**

  The new file is now present in the filesystem.
  The new file on the filesystem contains the text the user has written on the editor.




: User
: Main window GUI
: Save Dialog
: File Manager
: main()
: saveNewFile()
: createFile(path)
loop
[until user finishes editing]
alt
[modify text area]
Modify contents
alt
[trigger save]
: saveFile(path,contents)

Activity and sequence diagram of the *Change Settings* use case

In this section we analyze the behaviour of the application during the Change Settings use case. The process through which a user can alter the application settings.

- **Pre-conditions**

  The application must be running.

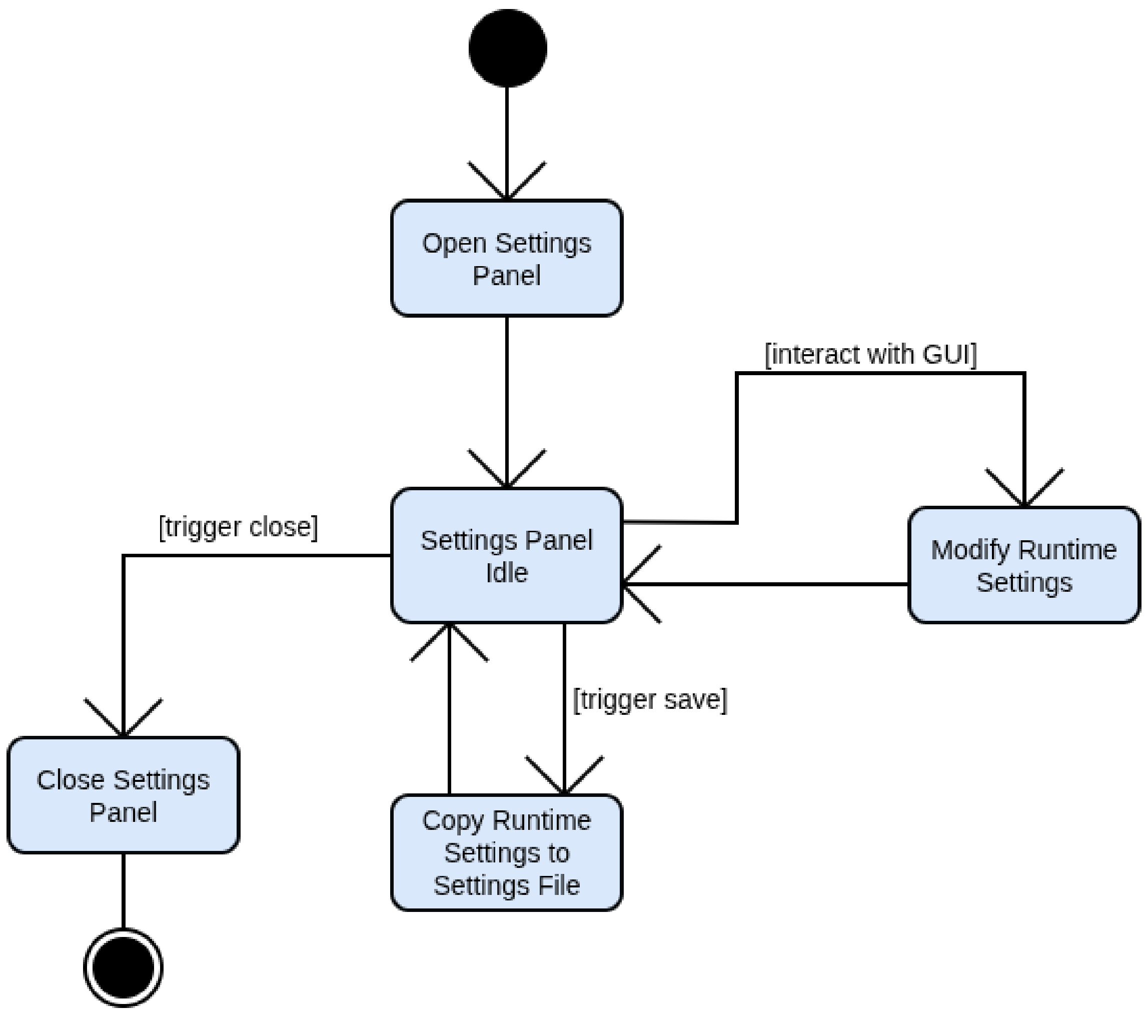
  The settings window mustn’t be open.

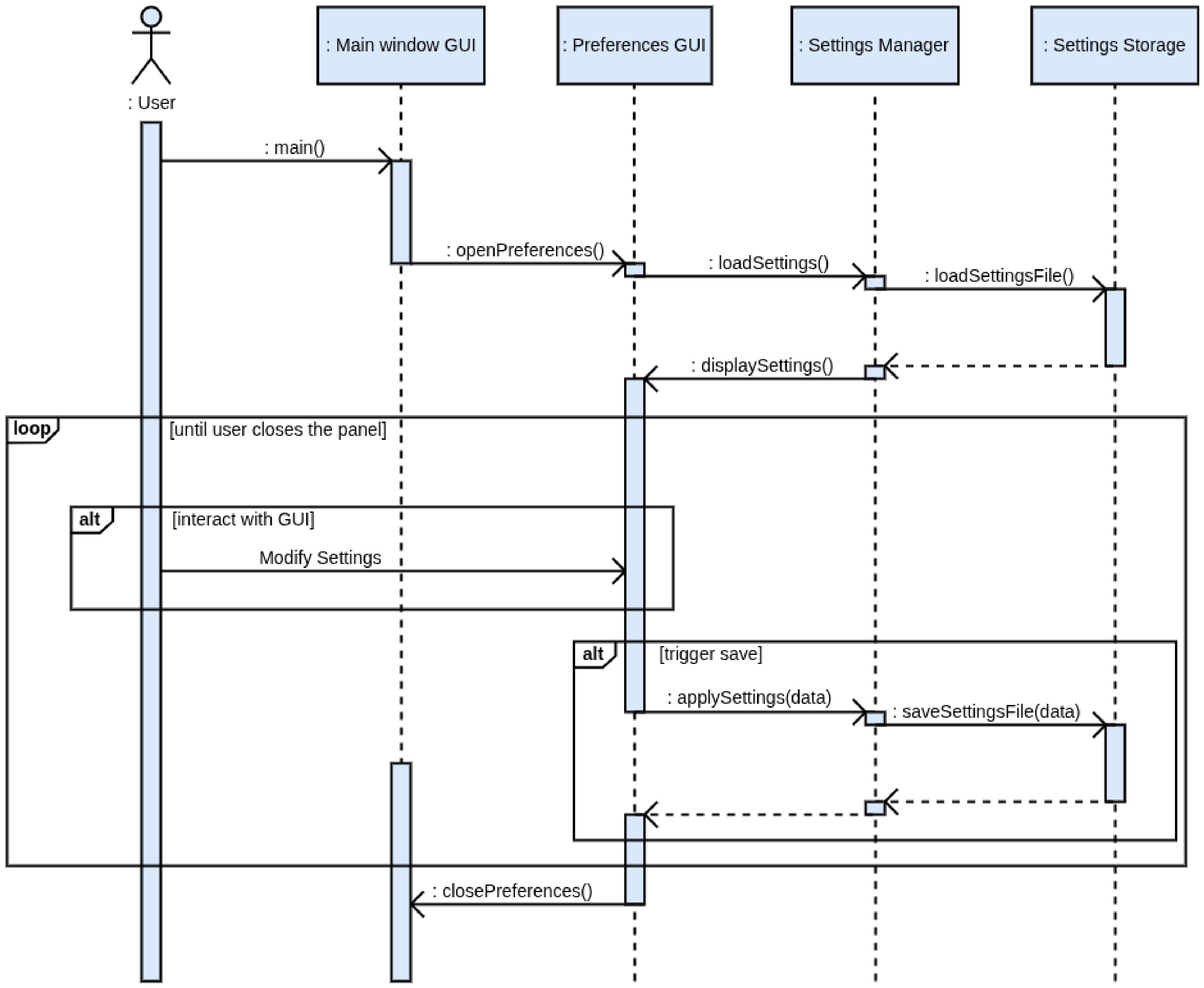
  The application must have reading and writing access to the settings file location.

- **Post-conditions**

  The settings file has changed accordingly to the user desired changes

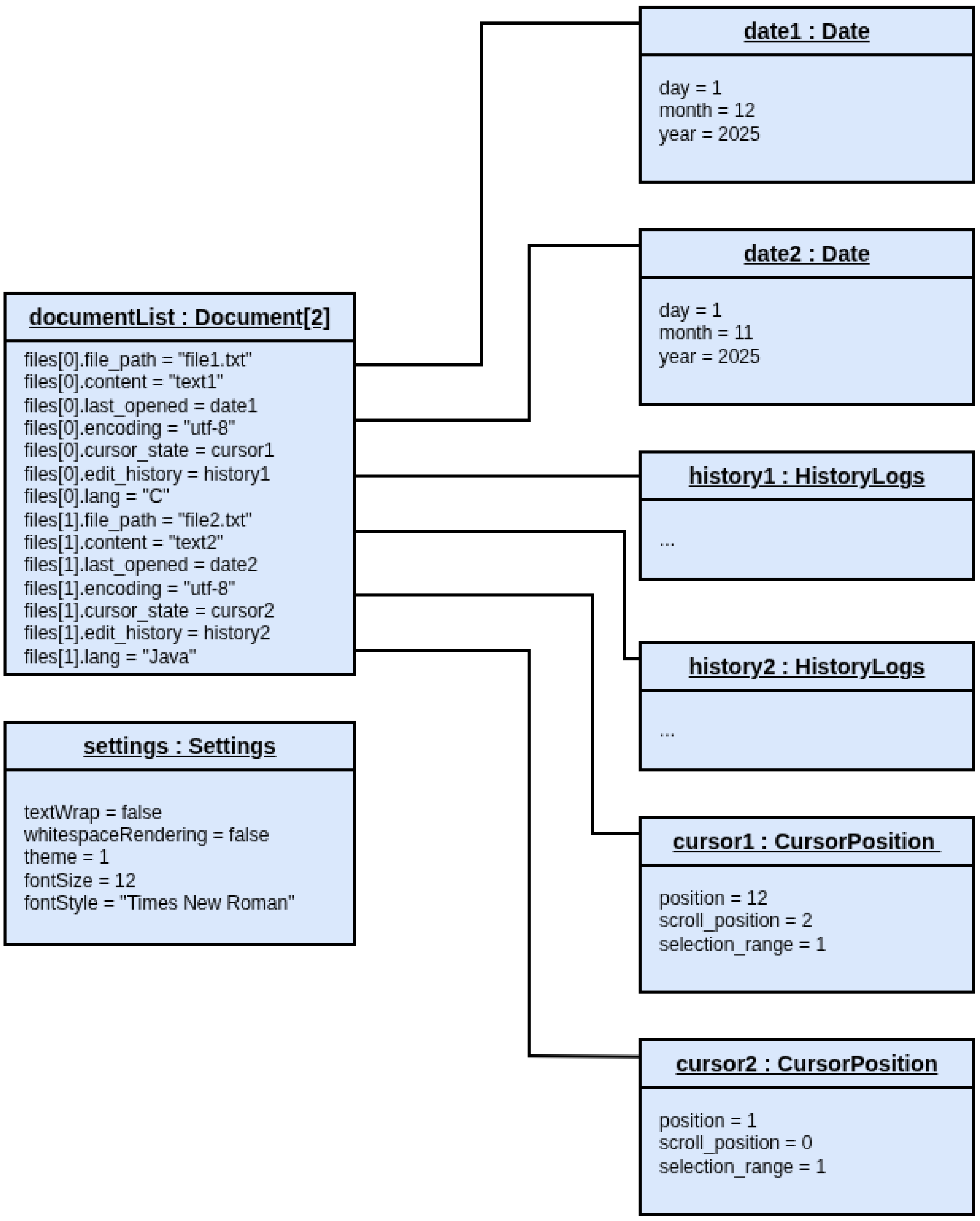
  The application behaves accordingly to the new settings




: User
: Main window GUI
: Preferences GUI
: Settings Manager
: Settings Storage
: main()
: openPreferences()
: loadSettings()
: loadSettingsFile()
: displaySettings()
loop
[until user closes the panel]
alt
[interact with GUI]
Modify Settings
alt
[trigger save]
: applySettings(data)
: saveSettingsFile(data)
: closePreferences()

Object diagram

Due to the language used by our application, which doesn't follow the Objected-Oriented Programming paradigm, there cannot be a class diagram for our project. This section displays an object diagram that shows data structures in an instance of execution. In this instance two files have been opened: *file1.txt* and *file2.txt*.


date1 : Date
day = 1
month = 12
year = 2025
date2 : Date
day = 1
month = 11
year = 2025
documentList : Document[2]
files[0].file_path = "file1.txt"
files[0].content = "text1"
files[0].last_opened = date1
files[0].encoding = "utf-8"
files[0].cursor_state = cursor1
files[0].edit_history = history1
files[0].lang = "C"
files[1].file_path = "file2.txt"
files[1].content = "text2"
files[1].last_opened = date2
files[1].encoding = "utf-8"
files[1].cursor_state = cursor2
files[1].edit_history = history2
files[1].lang = "Java"
history1 : HistoryLogs
...
history2 : HistoryLogs
...
settings : Settings
textWrap = false
whitespaceRendering = false
theme = 1
fontSize = 12
fontStyle = "Times New Roman"
cursor1 : CursorPosition
position = 12
scroll_position = 2
selection_range = 1
cursor2 : CursorPosition
position = 1
scroll_position = 0
selection_range = 1

3.3. Database Model

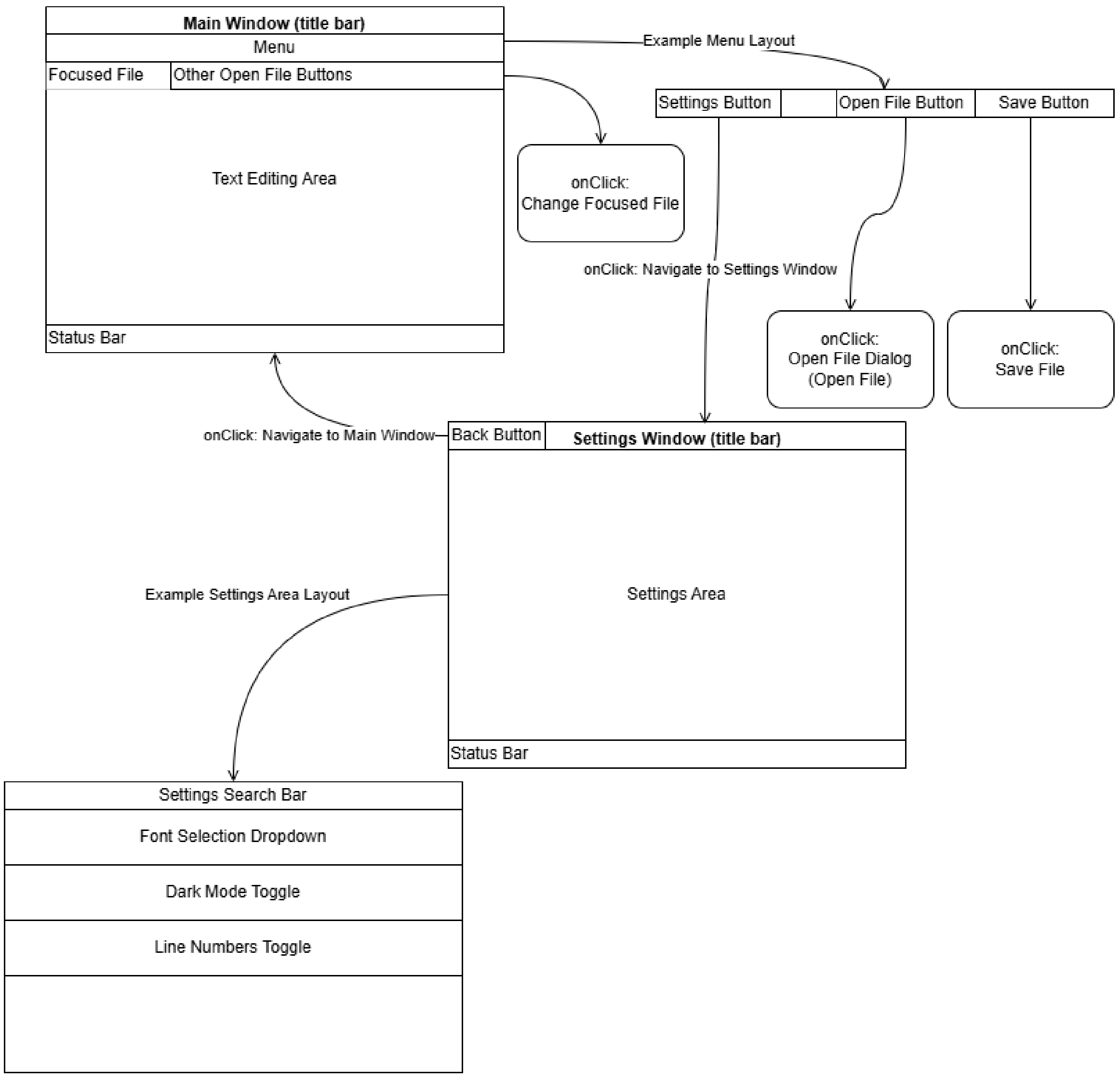
The application does not access a database.

3.4. User Interface Design

In this section, you should present the distribution of user interface components with events and their assignments to the behaviour specifications, which these elements initiate or take part.

The application will consist of two pages, the main window where most of the functionality is, and the settings windows where various parameters and settings can be adjusted within the application. The general layout is shown in the diagram below.

Each UI component is represented as a rectangle while each event is represented as a rounded rectangle. Behavior specifications are written in each event box. Arrows are used to show association between ui components and events, show example layouts, and show navigation between pages. The example layouts show a single possibility for the application but there will likely be differences in the completed application such as more buttons or more settings.


Main Window (title bar)
Menu
Focused File
Other Open File Buttons
Text Editing Area
Status Bar
Example Menu Layout
Settings Button
Open File Button
Save Button
onClick: Change Focused File
onClick: Navigate to Settings Window
onClick: Open File Dialog (Open File)
onClick: Save File
onClick: Navigate to Main Window
Back Button
Settings Window (title bar)
Settings Area
Status Bar
Example Settings Area Layout
Settings Search Bar
Font Selection Dropdown
Dark Mode Toggle
Line Numbers Toggle