Timer Duration

USER

Enter

Activity Interval

Enter

Productivity Interval

User Data

Timer Controller

Save Change

Reset Timer

Report change

Reset Timer

Set Data

Report

Change

The user can enter the desired time for the Activity interval. The Value is saved in User Data under the associated activity set data. This change is reported to the timer controller which resets the current timer and sets it to the applicable value meaning if the timer is in the same state as the changed interval type. The rest interval entered will only be saved to userdata, as it is not tied to a specific activity set. This change is reported to the timer controller which resets the current timer and sets it to the applicable value meaning if the timer is in the same state as the changed interval type.

Timer Usage

USER

Button press

Start

User Data

Timer Controller

Resume Timer

Start

Productive

Timer

Set Data

Validate

Button press

Pause

Stop

Timer

Button press

Next

If productive next

If Activity next

Validate

Start

Activity

Timer

Validate

User can start the timer, which will validate with User Data for productivity interval, and validate for Activity interval, to be sure of the duration of the timer, then the timer will resume its valid time, or set to correct time and begin. User can pause the timer and it will stop the timer. User can press next and will start the next productive or activity interval, and validate the time and activity that is next up.

Design Sequence

USER

Enter

int: Activity\_Duration

Enter

Int:

Productivity\_Duration

User Data

Timer Controller

SetInterval(Int, Activity)

Reset

Timer()

SetTimer(Productivity\_Duration)

Reset Timer()

Set Data

SetTimer(Activity\_Duraction)

Pause

Bool:isTimerActive

Pause

Timer()

The user can enter an integer value that corresponds to seconds. That value will set the activity duration in Set Data, a subclass of Userdata. The timer controller will reset after it receives the new activity duration. The user can enter an integer value that corresponds to seconds. This value will notify the timer controller which will reset after it receives the new productivity duration.

Permission Toggle



In this diagram, the user in on the permissions screen and clicks a toggle button to turn a permission off or on. We see that the input in the user clicking the toggle button, then it determines if the permission is being turned off or on. Depending on the answer, it asks the android permission manager to do different things. If turning on, it prompts the user to make sure they want to turn it on. If turning off, it simply disables the permission and returns confirmation.

Go to Permission Settings

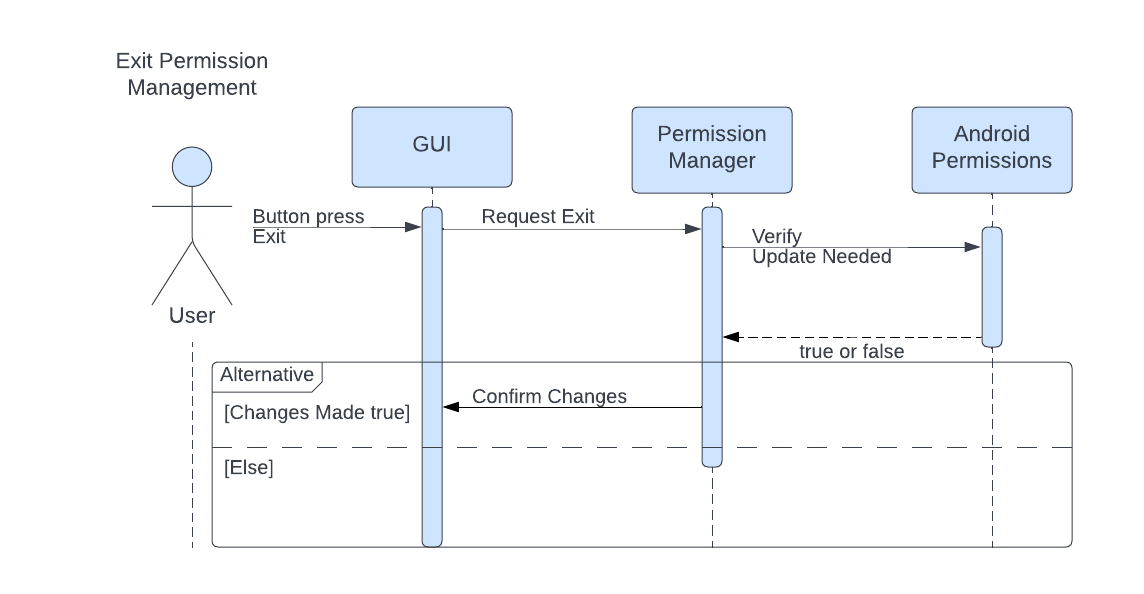


While on the settings screen, the user selects the Permission Settings button. Doing this will bring them to the Permissions screen gui. The app will have to generate the Permission Screen GUI object and display it.

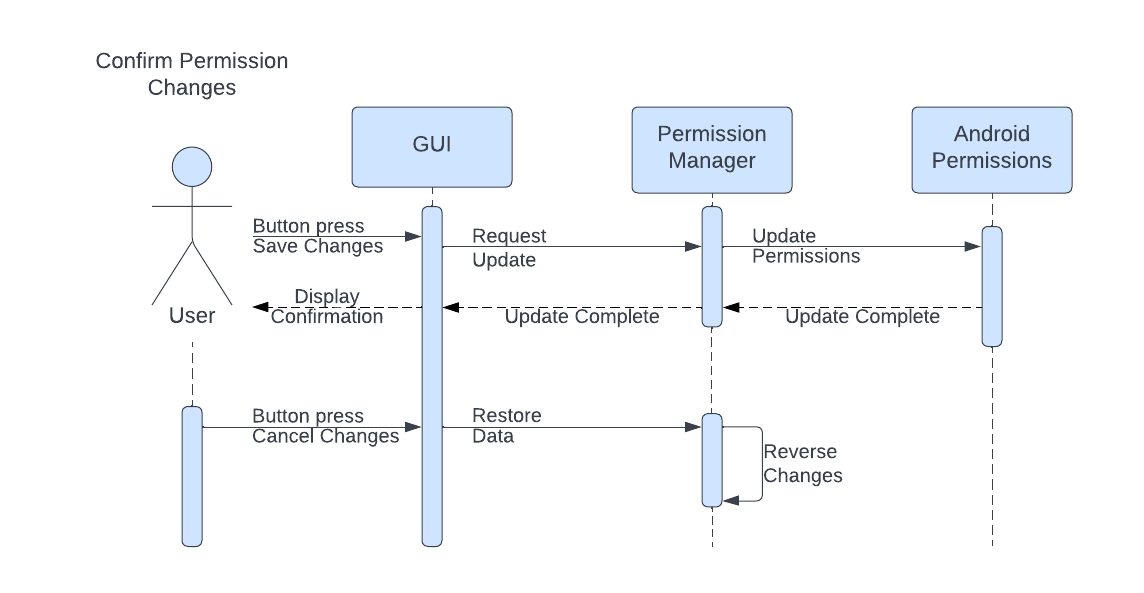
Toggle permissions design sequence diagram



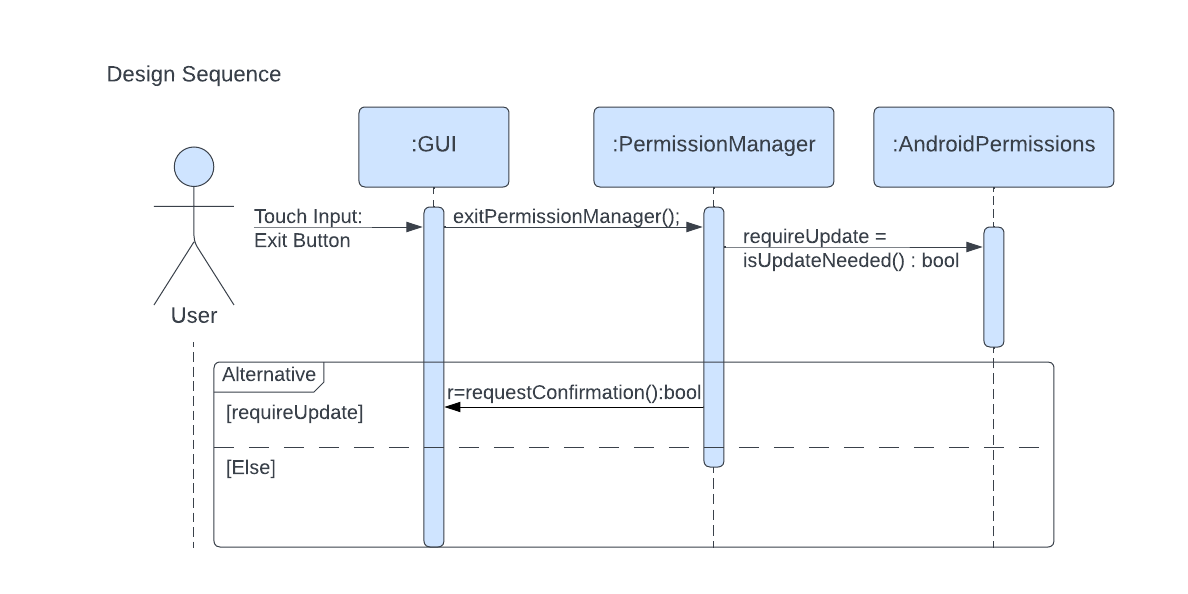
This is just a more technical view at the toggle permissions analysis diagram. The user is on the Permission Settings GUI and presses one of the permission toggle buttons. The app then checks if they’re turning the permission on or off, then acts accordingly. If its being turned on then it prompts the user for permission to turn it on. If it’s turning it off then it disables the permissions and returns confirmation that it has been turned off.



The user can press a button to exit from permission management. Once pressed by the user, the GUI sends a request to the Permission Manager to exit, this request is done to ensure that no other actions are required by the user before exiting. When the Permission Manager receives this requests it checks whether the user has made any changes by verifying current permissions with the saved Android permissions. If any are changed then Permission Manager will have the user confirm them. However, if none are changed then the Permission Manager will simply close out.

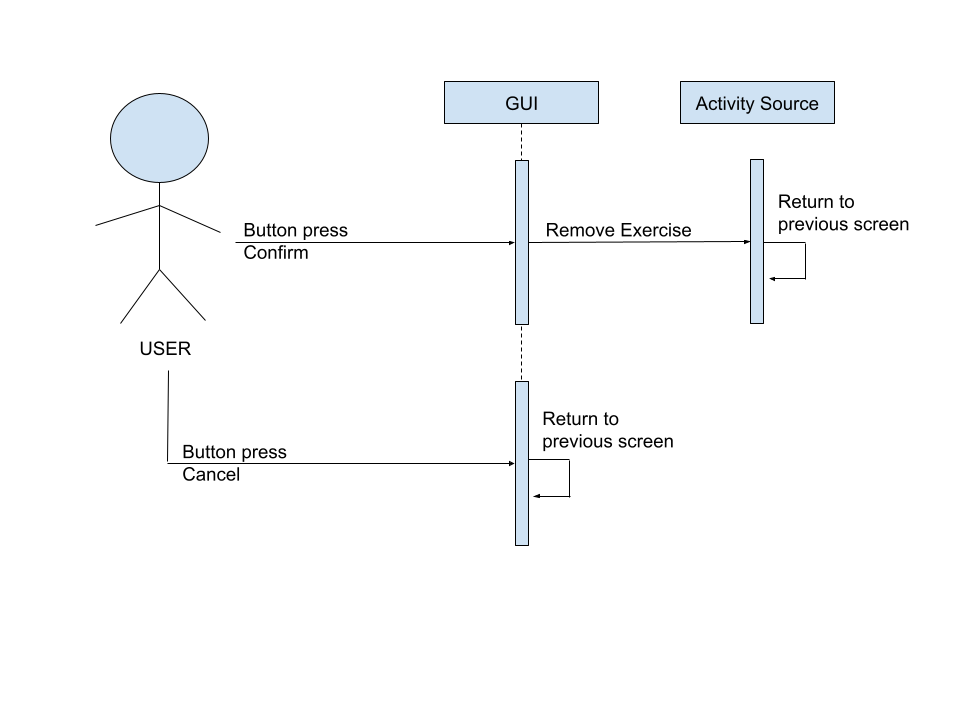


The user can press a button to save the changes made while in permission management. Once pressed by the user, the GUI sends the request to save the changes. Permission Manager receives this request and updates the saved Android Permissions. Once completed, a confirmation of the update being successfully is displayed to the user. The user can also press a button to cancel the changes they made. Once pressed by the user, the GUI sends a request to restore the permissions to their former state. When received by the Permission Manager, permissions are changed to reflect their initial state.



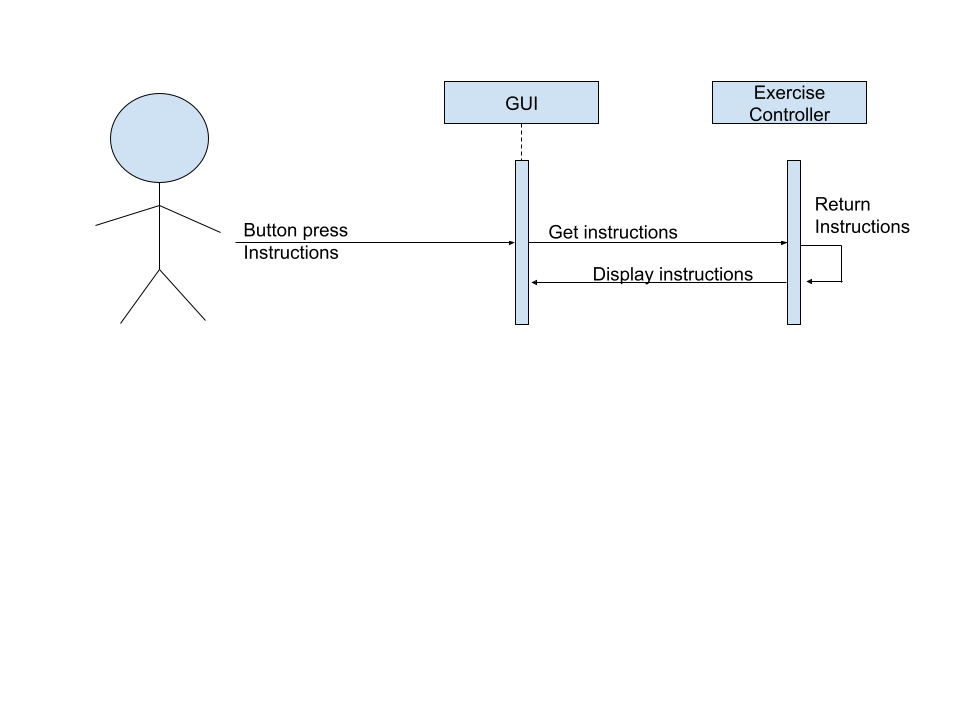
Through the GUI the user can provide touch input to call a function to exit from PermissionManager. This function will instruct PermissionManager to verify whether any changes were made and need to be reflected in AndroidPermissions. By using the isUpdateNeeded() function, a boolean value is returned and assigned to the ‘requireUpdate’ variable. If requireUpdate is true, PermissionManager will call a function that will begin the process of confirming whether or not the user would like to save their changes before exiting. Else, PermissionManager will simply exit.

Remove Exercise from Activity Source



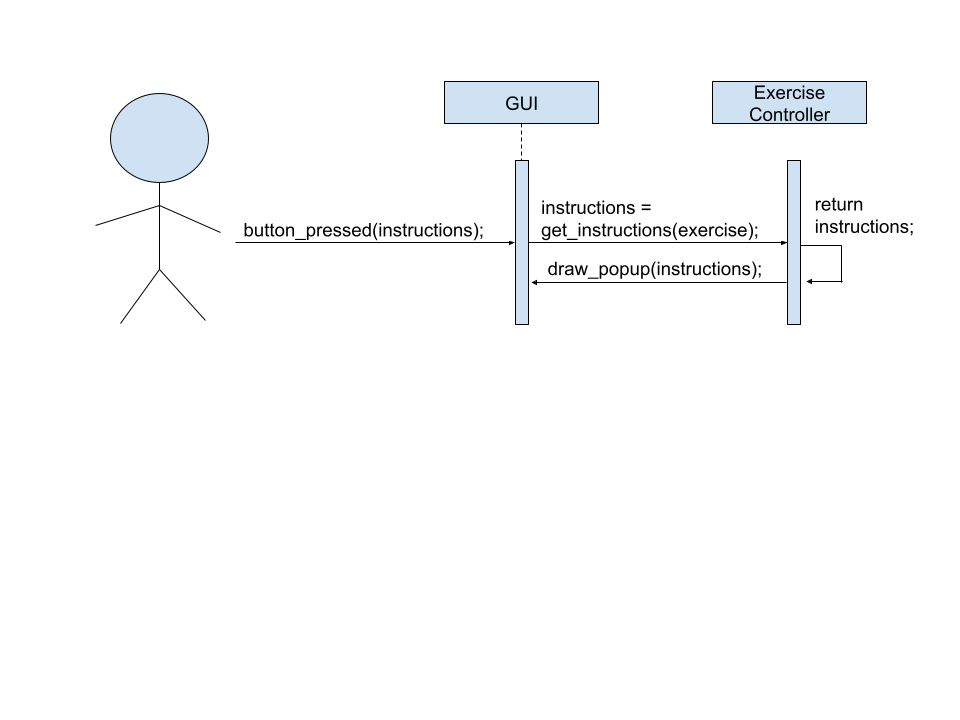
When the user confirms a prompt to remove an exercise from the Activity Source on the GUI, the GUI will instruct the Activity Source to remove the exercise, after which the app will return to the previous screen. Otherwise, if the user chooses to cancel the removal, the app will return to the previous screen.

Display Interval Instructions



When the user is on the timer screen, they may choose to press the “Instructions” button. Upon doing so, the GUI will instruct the exercise controller to return the instructions of the current exercise to be viewed in a pop-up on screen.

Design Sequence: Display Interval Instructions



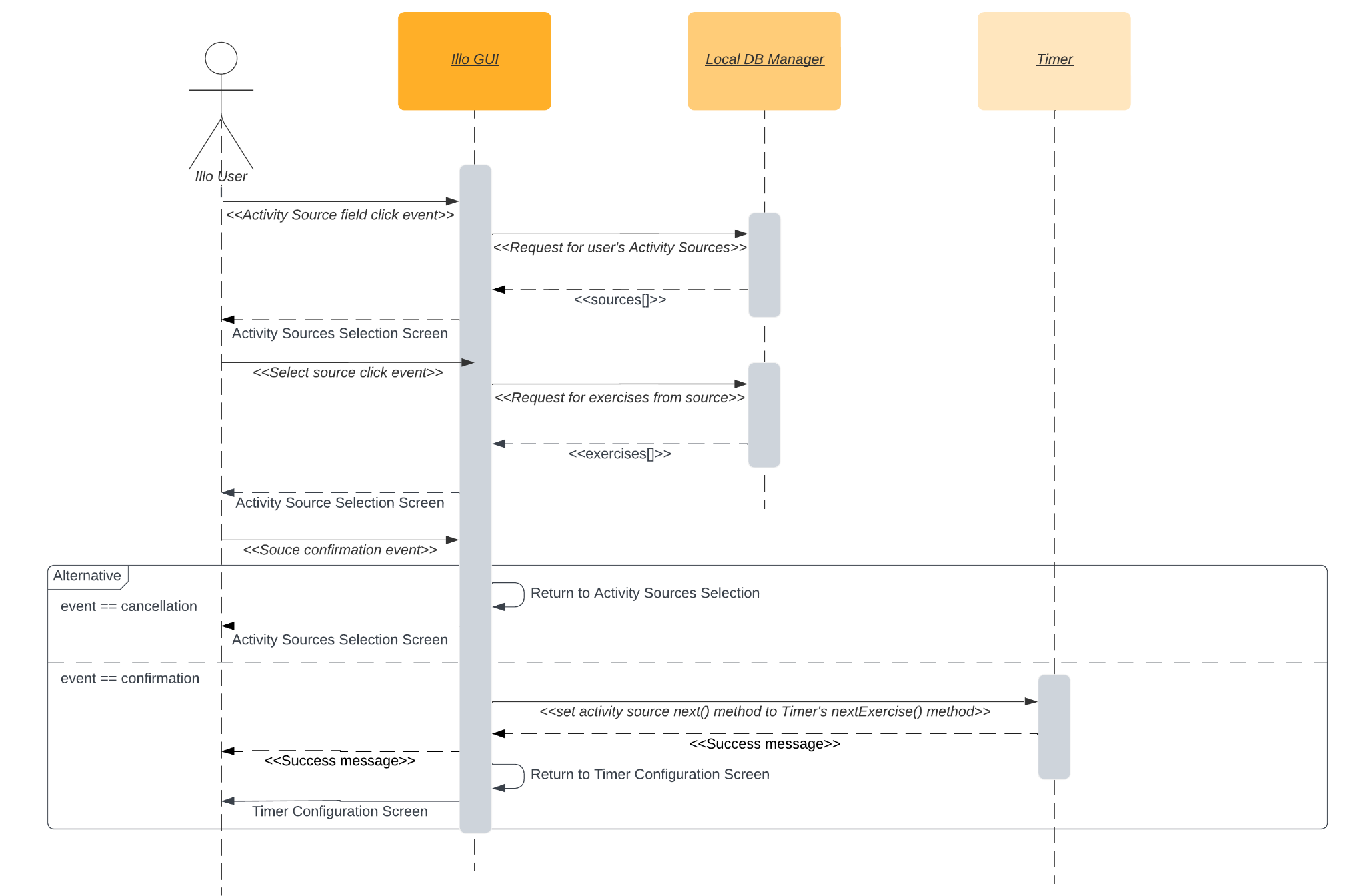
Upon interacting with the instructions on the Timer screen’s graphical interface, a new pop-up object called “instructions” will be created .This object will have the capability of storing information in a distinct format to be drawn to the screen. Upon its creation, the exercise controller will assign the pop-up its instructions and necessary information. Then, the object will be sent to the GUI to be drawn to the screen for the user to view.

**Use Case: Set Activity Source**

*Analysis Description*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Subject | Subject Action | Parameters | Object Acted Upon |
| 1 | User | selects | Activity Source field | Illo GUI |
| 2.1 | Illo GUI | retrieves | Activity Sources | Local DB Manager |
| 2.2 | Illo GUI | presents | User’s Activity Sources | User |
| 3 | User | selects | an ActivitySource | Illo GUI |
| 4.1 | Illo GUI | retrieves | Exercises[] from given Activity Source | Local DB Manager |
| 4.2 | Illo GUI | presents | Exercises[], confirm/cancel buttons | User |
| 5 | User | confirms | ActivitySource | Illo GUI |
| 6.1 | Illo GUI | retrieves | next() method from ActivitySource | Local DB Manager |
| 6.2 | Illo GUI | sets | The timer’s nextExercise() method to the retrieved next() method | Timer |
| 6.3 | Illo GUI | sends | Timer Configuration Screen | User |

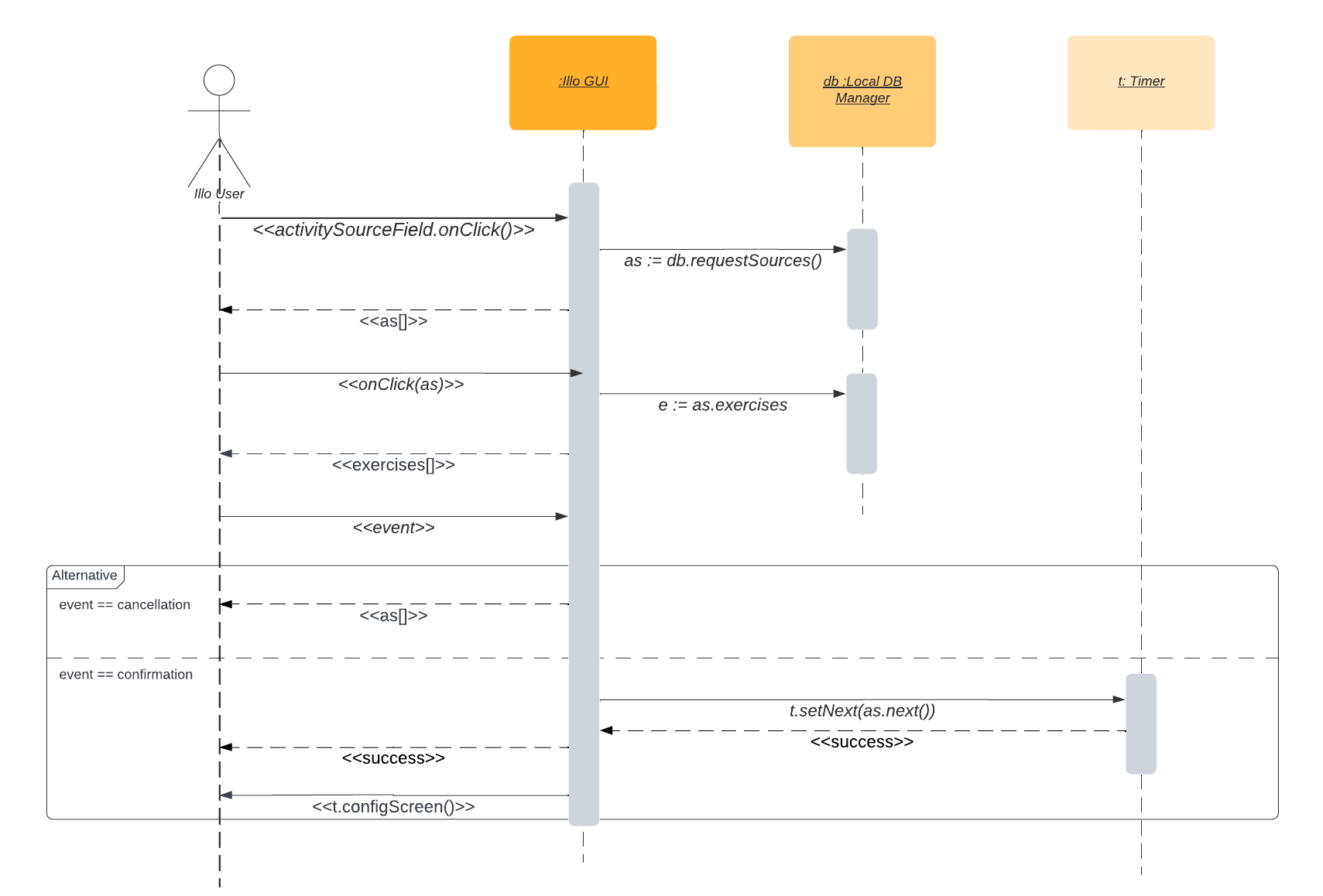
Analysis Sequence for Above



*Design Sequence Description*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Subject | Subject Action | Parameters | Object Acted Upon |
| 1 | User | *onClick()* | activitySourceField | Illo GUI |
| 2.1 | Illo GUI | *requestSources* |  | Local DB Manager |
| 2.2 | Illo GUI | renders | as: Activity Sources | User |
| 3 | User | *onClick()* | *as* | Illo GUI |
| 4.1 | Illo GUI | *exercises* |  | Local DB Manager |
| 4.2 | Illo GUI | presents | *exercises* | User |
| 5 | User | *onClick()* | Confirmation event | Illo GUI |
| 6.1 | Illo GUI | *setNext()* | *as.next()* | Timer |
| 6.2 | Illo GUI | sends user to | *t.configScreen()* | User |

Design Sequence for Above



**Use Case: Make New Activity Source**

*Description*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Subject | Subject Action | Parameters | Object Acted Upon |
| 5 | User | enters | *Activity source name, activity source type (*Exercise set | Workout) | Illo GUI |
| 6.1 | Illo GUI | saves | new ActivitySource(*name, type)* | Illo Local DB |
| 6.2 | Illo GUI | generates | Edit-Exercises-Screen for new ActivitySource() |  |
| 6.3 | Illo GUI | presents | Success message for creation of new ActivitySource() | User |
| 6.4 | Illo GUI | sends to | To new edit screen for the activity source | User |
| The expected, but not necessary, behavior is that the user proceeds to edit the new Activity Source. See the **Edit Activity Source** use case. | | | | |

Analysis Sequence for Above

