Number	Test Group	Test Type	Name	Test case data	Preconditions	Expected Results
1	Test DatabaseManager::TaskManager	Automated	successul task creation	Call the method TaskManager.createTask three times with different task names and check the results.	The TaskModel collection is empty.	A new task is created each time with the expected attributes and status code. The tasks are correctly stored in the database.
2		Automated	unsuccessul task creation	Call the method TaskManager.createTask with various invalid task data	The TaskModel collection is empty.	Errors.BAD_REQUEST is returned
3		Automated	successul task note creation	Call the method TaskManager.createTaskNotes and pass a valid taskId and notes	The TaskModel has a few tasks created where notes can be added	Notes for tasks are created
4		Automated	unsuccessul task note creation	Call the method TaskManager.createTaskNotes with invalid parameters, and/or with non existing task	The TaskModel has a few tasks created where notes can be added	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
5		Automated	successful task report schedule creation	Call TaskManager.createTaskReportSchedule with valid report schedule data	The TaskModel has a few tasks created where schedules can be added	A successful status code is returned and the task report schedules are created correctly.
6		Automated	unsuccessful task report schedule creation	Call TaskManager.createTaskReportSchedule with invalid report schedule data	The TaskModel has a few tasks created where schedules can be added	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
7		Automated	successful task update	Call the method TaskManager.updateTask and pass valid parameters	The TaskModel has a few tasks created that can be edited for the test	Tasks are successfully updated with the correct data
8		Automated	unsuccessful task update	Call the method TaskManager.updateTask and pass non valid parameters	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
9		Automated	successful update last task update	Call TaskManager.updateTaskLastUpdated and pass the id of an existing task	The TaskModel has a few tasks created that can be edited for the test	The task's lastUpdated field is updated correctly, with the current time
10		Automated	unsuccessful update last task update	Call TaskManager.updateTaskLastUpdated and pass the id of a non existing task	The TaskModel has a few tasks created that can be edited for the test	The expected status code for each update attempt is Errors.NOT_FOUND
11		Automated	successful update task name	Call the method TaskManager.updateTaskName with valid task and organization IDs, and a new task name	The TaskModel has a few tasks created that can be edited for the test	The task's name is successfuly updated in the database
12		Automated	unsuccessful update task name	Call the method TaskManager.updateTaskName with invalid task and organization IDs, and/or an invalid name	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
13		Automated	successful update task description	Call the method TaskManager.updateTaskDescription with valid task data	The TaskModel has a few tasks created that can be edited for the test	The task's description is successfuly updated in the database
14		Automated	unsuccessful update task description	Call the method TaskManager.updateTaskDescription with invalid task data	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
15		Automated	successful update task status	Call the method TaskManager.updateTaskStatus with valid task and status	The TaskModel has a few tasks created that can be edited for the test	The task's status is successfuly updated in the database
16		Automated	unsuccessful update task status	Call the method TaskManager.updateTaskStatus with invalid task ids and states id	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
17		Automated	successful update task notes	Call the method TaskManager.updateTaskNotes with valid data	The TaskModel has a few tasks created that can be edited for the test	The task's notes are successfuly updated in the database
18		Automated	unsuccessful update task notes	Call the method TaskManager.updateTaskNotes with invalid data or task id	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
19		Automated	successful add new assaignee	Call the method TaskManager.addNewAssignee with valid assignee ID and task id	The TaskModel has a few tasks created that can be edited for the test	The task's assainees list is successfuly updated in the database
20		Automated	unsuccessful add new assaignee	Call the method TaskManager.addNewAssignee with invalid assignee ID or task id	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
21		Automated	successful update task manager	Call the method TaskManager.updateTaskManager with a valid new manager, and task id	The TaskModel has a few tasks created that can be edited for the test	The task's manager is successfuly updated in the database
22		Automated	unsuccessful update task manager	Call the method TaskManager.updateTaskManager with an invalid new manager, or task id	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
23		Automated	successful enable and disable notification	Call TaskManager.enableNotification and TaskManager.disableNotification with valid task IDs	The TaskModel has a few tasks created that can be edited for the test	The task's notification flag is successfuly updated in the database
24		Automated	unsuccessful enable and disable notification	Call TaskManager.enableNotification and TaskManager.disableNotification with invalid task IDs	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
25		Automated	successful add child task	Call the method TaskManager.addChildTask with a valid task id and child task id	The TaskModel has a few tasks created that can be edited for the test	The task's notes are successfuly updated in the database
26		Automated	unsuccessful add child task	Call the method TaskManager.addChildTask with an invalid task id or child task id	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
27		Automated	successful update recursive permission value	Call the method TaskManager.updateTaskRecursivePermissionsValue with valid task id and permission value	The TaskModel has a few tasks created that can be edited for the test	The task's recursive permission value issuccessfuly updated in the database
28		Automated	unsuccessful update recursive permission value	Call the method TaskManager.updateTaskRecursivePermissionsValue with invalid task id or permission value	The TaskModel has a few tasks created that can be edited for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
29		Automated	successful delete task	Call the method TaskManager.deleteTask with valid taskId and organizationId	The TaskModel has a few tasks created that can be deleted for the test	the specified task is successfuly deleted in the database
30		Automated	unsuccessful delete task	Call the method TaskManager.deleteTask with invalid taskId or organizationId	The TaskModel has a few tasks created that can be deleted for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
31		Automated	successful delete task notes	Call TaskModel.deleteTaskNote with a valid task and note id	The TaskModel has a few tasks created that can be deleted for the test	the specified task note is successfuly deleted in the database
32		Automated	unsuccessful delete task notes	Call TaskModel.deleteTaskNote with an invalid task or note id	The TaskModel has a few tasks created that can be deleted for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
33		Automated	successful delete task assignee	Call the method TaskManager.deleteTaskAssignee with valid task id, user id, and assignee id	The TaskModel has a few tasks created that can be deleted for the test	the specified task assignee is successfuly deleted from the assignees list in the database
34		Automated	usuccessful delete task assignee	Delete task assignee with invalid task id, assignee id, or organization id	The TaskModel has a few tasks created that can be deleted for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
35		Automated	successful delete report schedule	Delete report schedule with valid task id, report schedule id, and organization id	The TaskModel has a few tasks created that can be deleted for the test	the specified task report schedule is successfuly deleted in the database
36		Automated	unsuccessful delete report schedule	Delete report schedule with invalid task id, report schedule id, or organization id	The TaskModel has a few tasks created that can be deleted for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
37		Automated	successful remove child task	Remove child task with valid parent task id, child task id, and organization id	The TaskModel has a few tasks created that can be deleted for the test	the specified child task is successfuly deleted from the child tasks list in the database
38		Automated	unsuccessful remove child task	Remove child task with invalid parent task id, child task id, or organization id	The TaskModel has a few tasks created that can be deleted for the test	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
39		Automated	successful get task	Get task with valid task id and organization id	The TaskModel has a few tasks created that can be red for the test	Successfully retrieve task details, returning Errors.OK
40		Automated	unsuccessful get task	Get task with invalid task id or organization id	The TaskModel has a few tasks created that can be red for the test	Task not found, returning Errors.NOT_FOUND
41	Test OrganizationManager	Automated	test successfull add user to organization API	Call the API to add an user to the organization with a valid requeat	User and Organizations are already present in the database	The api returns OK (200) and the database is updated
42		Automated	test unsuccesfull add user to organization API	Call the API to add an user to the organization with invalid users/organizations	User and Organizations are already present in the database	Not found error (404) or Bad request error (400) is returned based on the invalid parameters
43		Automated	test unautorized add user to organization API	Call the API to add an user to the organization with invalid/missing autorization token	User and Organizations are already present in the database	Non autorized error (403)
44		Automated	test successfull remove user from organization API	Call the API to remove an user from an organization with a valid requeat	User and Organizations are already present in the database	The api returns OK (200) and the database is updated
45		Automated	test unsuccesfull remove user from organization API	Call the API to remove an user from an organization with invalid users/organizations	User and Organizations are already present in the database	Not found error (404) or Bad request error (400) is returned based on the invalid parameters
46		Automated	test unautorized remove user from organization API	Call the API to remove an user from an organization with invalid/missing autorization token	User and Organizations are already present in the database	Non autorized error (403)
47		Automated	test successful calculate subscription price	Call the method OrganizationManager.calculateSubscriptionPrice with a valid organization id	an Organization with some users and soem tasks is present in the database	The function successfuly caluclate the price the ortanization should pay based on usage
48		Automated	test unsuccessful calculate subscription price	Call the method OrganizationManager.calculateSubscriptionPrice with an invalid organization id	an Organization with some users and soem tasks is present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters

49		Automated	test succesful verify subscription	Call the method OrganizationManager.verifySubscription with a valid organization id	an Organization is present in the database	The function verify the current status of the organization subscription and returns it
50		Automated	test unsuccesful verify subscription	Call the method OrganizationManager.verifySubscription with an invalid organization id	an Organization is present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
51		Automated	test successful pay subscription	Call the API to pay the subscription with valid organization, and credit card informations	an Organization is present in the database, A connection with a bancking service has been established, A valid tast credit card is ready	The payment is successful
52		Automated	test unsuccessful pay subscription	Call the API to pay the subscription with invalid organization, or credit card informations	an Organization is present in the database, A connection with a bancking service has been established, A valid tast credit card is ready	The payment is not successful
53	Test Report Page	Manual	test add periodic report	Go to the report page and take a look at the list of scheduled reports. Then add a few new repors by using the button "Add report schedule"	The backend, the front end and the database are running. You have access to a testing account	The added report are displayed in the table and all the informations are corrects. This is true even after a log out and a browser cash refresh
54		Manual	test delete periodic report	Go to the report page and take a look at the list of scheduled reports. Then click the delete button on some of them.	The backend, the front end and the database are running. You have access to a testing account	The deleted reports are no longer displayed in the list This is true even after a log out and a browser cash refresh. The user no longer receive notifications about the scheduler reports
55		Automated	test successfull send notification	Call method NotificationManager.sendNotification with valid parameters	At least one user and one notification are present in the database	The notification is sent successfully
56		Automated	test unsuccessfull send notification	Call method NotificationManager.sendNotification with invalid parameters	At least one user and one notification are present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
57		Automated	test successfull send notification	Call method ExternalNotificationManager.sendNotification with valid parameters	At least one user and one notification are present in the database, we have a mail account with api access to test the notification	The notification is sent successfully
58		Automated	test unsuccessfull send notification	Call method ExternalNotificationManager.sendNotification with invalid parameters	At least one user and one notification are present in the database, we have a mail account with api access to test the notification	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
59		Automated	test successfull delete notification	Call method InternalNotificationManager.deleteNotification with valid parameters (userId, notificationId, userToken)	At least one user and one notification are present in the database	The notification is deleted successfully
60		Automated	test unsuccessfull delete notification	Call method InternalNotificationManager.deleteNotification with invalid parameters (userId, notificationId, userToken)	At least one user and one notification are present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
61	Test NotificationManager	Automated	test successfull mark notification as read	Call method InternalNotificationManager.markNotificationAsRead with valid parameters (userId, notificationId, userToken)	At least one user and one notification are present in the database	The notification gets marked as read
62		Automated	test unsuccessfull mark notification as read	Call method InternalNotificationManager.markNotificationAsRead with invalid parameters (userId, notificationId, userToken)	At least one user and one notification are present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
63		Automated	test successfull get notification details	Call method InternalNotificationManager.getNotificationDetails with valid parameters (userId, notificationId, userToken)	At least one user and one notification are present in the database	A notification object is returned
64		Automated	test unsuccessfull get notification details	Call method InternalNotificationManager.getNotificationDetails with invalid parameters (userId, notificationId, userToken)	At least one user and one notification are present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
65		Automated	test successfull get notification list	Call method InternalNotificationManager.getNotificationDetails with valid parameters (userId, userToken)	At least one user is present in the database	The list of notifications linked to the userId are returned
66		Automated	test unsuccessfull get notification list	Call method InternalNotificationManager.getNotificationDetails with invalid parameters (userId, userToken)	At least one user is present in the database	Errors.BAD_REQUEST is returned
67		Automated	test successfull set new report schedule	Call method reportScheduler.scheduleReport with valid parameters (organizationId, taskId, reportPrompt, reportKind, reportFrequency, userId, userToken)	Organization, Task, and User are already present in the database	A new report schedule is set for said task and with said frequency
68		Automated	test unsuccessfull set new report schedule	Call method reportScheduler.scheduleReport with invalid parameters (organizationId, taskId, reportPrompt, reportKind, reportFrequency, userId, userToken)	Organization, Task, and User are already present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
69		Automated	test successfull get report schedules	Call method reportScheduler.getReportSchedules with valid parameters (organizationId, taskId, userId, userToken)	Organization, Task, and User are already present in the database	The list of report schedules set for the task are returned
70		Automated	test unsuccessfull get report schedules	Call method reportScheduler.getReportSchedules with invalid parameters (organizationId, taskId, userId, userToken)	Organization, Task, and User are already present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
71		Automated	test successfull remove report schedule	Call method reportScheduler.deleteReportSchedule with valid parameters (organizationId, taskId, reportId, userId, userToken)	Organization, Task, User and report are already present in the database	The report schedule gets deleted successfully
72		Automated	test unsuccessfull remove report schedule	Call method reportScheduler.deleteReportSchedule with invalid parameters (organizationId, taskId, reportId, userId, userToken)	Organization, Task, User and report are already present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
73	Test ReportManager	Automated	test execute all pending periodic reports	Call method reportScheduler.executeScheduledReport with valid parameters (organizationId)	An Organization is present in the database	All the pending reports gets executed
74		Automated	test unsuccessfull execute all pending periodic reports	Call method reportScheduler.executeScheduledReport with invalid parameters (organizationId)	An Organization is present in the database	Errors.BAD_REQUEST is returned
75		Automated	test successfull generate automatic report	Call method AutomaticReportManager.generateAutomaticReports with valid parameters (organizationId, taskId, reportPrompt)	Organization and Task are already present in the database, connection with LLM APIs is valid	The report gets generated successfully
76		Automated	test unsuccessfull generate automatic report	Call method AutomaticReportManager.generateAutomaticReports with invalid parameters (organizationId, taskId, reportPrompt)	Organization and Task are already present in the database, connection with LLM APIs is valid	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
77		Automated	test successfull generate manual report	Call method ManualReportManager.generateManualReports with valid parameters (organizationId, taskId, reportPrompt)	Organization and Task are already present in the database	The report gets generated successfully
78		Automated	test unsuccessfull generate manual report	Call method ManualReportManager.generateManualReports with invalid parameters (organizationId, taskId, reportPrompt)	Organization and Task are already present in the database	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
79		Automated	test successfull generate report	Call method ReportManager.generateReport with valid parameters (organizationId, taskId, reportPrompt, reportKind, userId, userToken)	Organization, Task and User are already present in the database, connection with LLM APIs is valid	The report gets generated successfully
80		Automated	test unsuccessfull generate report	Call method ReportManager.generateReport with invalid parameters (organizationId, taskId, reportPrompt, reportKind, userId, userToken)	Organization, Task and User are already present in the database, connection with LLM APIs is valid	Errors.NOT_FOUND or Errors.BAD_REQUEST is returned based on the invalid parameters
81		Automated	successful organization creation	Call the method OrganizationManager.createOrganization with valid data and check the result	OrganizationModel database is empty	New organization is successfully created with provided data, and all attributes are correctly saved in the database
82		Automated	invalid organization creation	Call the method OrganizationManager.createOrganization with various invalid data and check the results	OrganizationModel database is empty	The response status code should be Errors.BAD_REQUEST for each invalid organization data combination.
83		Automated	successful add user to organization	Call the method OrganizationManager.addUserToOrganization with valid input and check the result	Organization and User should already exist in the database	Errors.OK is returned. The user should be successfully added to the organization, and the organization's user list should include the added user ID.
84	Test DatabaseManager::OrganizationManager	Automated	unsuccessful add user to organization	Call the method OrganizationManager.addUserToOrganization with invalid input and check the result		Invalid attempts should return Errors.NOT_FOUND or Errors.BAD_REQUEST
85		Automated	update license	Call the method OrganizationManager.updateLicense and check the results	An organization with the specified ID exists in the database	Valid attempts should return Errors.OK and license is updated. Invalid attempts should return Errors.NOT_FOUND or Errors.BAD_REQUEST
86		Automated	update license expiration date	Call the method OrganizationManager.updateLicenseExpirationDate and check the results	An organization with the specified ID exists in the database	Valid attempts should return Errors.OK and date is updated. Invalid attempts should return Errors.NOT_FOUND or Errors.BAD_REQUEST
87		Automated	read organization	Call the method OrganizationManager.readOrganization and check the results	An organization with the specified ID exists in the database	The response status code should be Errors.OK. The organization data should match the those created earlier.
88		Automated	test successful find user by name	Call the method UserManager::readUserByName with valid input and check the result	A user with the specified name exists in the database	The response status code should be Errors.OK.
89		Automated	test unsuccessful find user by name	Call the method UserManager::readUserByName with invalid input and check the result		The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST depending on the incorrect imput inserted
90		Automated	test getGoogleOauthLogInInfo	Call the method UserManager::readGoogleUserInfo	The specified user should exist in the database and have performed signup with Google	The user infos are returned correctly if the user exists

91	Test AccountManager	Automated	test getFacebookOauthLogInInfo	Call the method UserManager::readFacebookUserInfo	The specified user should exist in the database and have performed signup with Facebook	The user infos are returned correctly if the user exists
92		Automated	test successful custom login	Perform user login using correct username and password	The specified user should exist in the database	The user is correctly logged in
93		Automated	test unsuccessful custom login	Perform user login trying to use incorrect parameters	The specified user should not exist in the database or the password should be incorrect	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and logging in is not performed
94		Automated	test successful Google login	Perform user login using Google	The specified user should exist in the database and have performed signup with Google	The user is correctly logged in using Google
95		Automated	test unsuccessful Google login	Perform user login using Google inserting invalid credentials	The specified user should not exist in the database or not having performed signup with Google	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and logging in is not performed
96		Automated	test successful Facebook login	Perform user login using Facebook	The specified user should exist in the database and have performed signup with Facebook	The user is correctly logged in using Facebook
97		Automated	test unsuccessful Facebook login	Perform user login using Facebook inserting invalid credentials	The specified user should not exist in the database or not having performed signup with Facebook	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and logging in is not performed
98		Automated	test successful custom signup	Call the method UserManager::createUser with valid input and check the result	The UserModel collection is empty	The response status code should be Errors.OK and the user is correctly created
99		Automated	test unsuccessful custom signup	Call the method UserManager::createUser with invalid input and check the result	The UserModel collection is empty	The response status code should be Errors.BAD_REQUEST and an error message should be returned
100		Automated	test Google signup	Perform user signup using Google	The UserModel collection is empty	True is returned if the user is created successfully, otherwise False is returned
101		Automated	test Facebook signup	Perform user signup using Facebook	The UserModel collection is empty	True is returned if the user is created successfully, otherwise False is returned
102		Automated	test successful edit username	Call the method UserManager::updateUserName with valid input	The user should exist in the database	The response status code should be Errors.OK and the username is updated correctly
103		Automated	test unsuccessful edit username	Call the method UserManager::updateUserName with invalid input		The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST, the username is not updated
104		Automated	test successful change password	Change user password with correct old password and valid new one	The user should exist in the database and the password inserted should correspond to the original one	The response status code should be Errors.OK and the password is updated
105		Automated	test unsuccessful change password	Change user password with incorrect old password or invalid new one	The user should exist in the database	The response status code should be Errors.BAD_REQUEST and an error message should be returned
106		Automated	test successful delete account	Call the method UserManager::deleteAccount with valid input and check the result	The account should exist in the database	The response status code should be Errors.OK and the account is correctly deleted
107		Automated	test unsuccessful delete account	Call the method UserManager::deleteAccount with invalid input and check the result	The account should not exist in the database	The response status code should be Errors.NOT_FOUND and an error message is displayed
108	Test DatabaseManager::UserManager	Automated	test successful create user	Call the method UserManager.createUser with several correct inputs and check the result	The UserModel collection is empty	The response status code should be Errors.OK and the user is correctly created
109		Automated	test unsuccessful create user	Call the method UserManager.createUser with several invalid inputs and check the result	The UserModel collection is empty	The response status code should be Errors.BAD_REQUEST and an error message is displayed
110		Automated	test successful find user	Call the method UserManager.findUser for an existing user and check the result	The UserModel collection should contain the user	The response status code should be Errors.OK and the user is correctly returned
111		Automated	test unsuccessful find user by name	Call the method UserManager.findUser for a non existing and/or with invalid parameters	The UserModel collection should not contain the user	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
112		Automated	test successful find every user	Call the method UserManager.findEveryUser and check the results	The UserModel collection should be non-empty	The response status code should be Errors.OK and every user is correctly found
113		Automated	test successful create notification	Call the method UserManager.createNotification with invalid inputs and an existing user and check the result	The corrisponding user should be contained the UserModel collection	The response status code should be Errors.OK and the notification is correctly created
114		Automated	test unsuccessful crete notification	Call the method UserManager.createNotification with invalid inputs and/or a non existing user	The corrisponding user should not be contained the UserModel collection and/or the input for the method should be invalid	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
115		Automated	test successful read notification	Call the method UserManager.readNotification for an existing user and check the result	The related user should already be contained in the UserModel collection	The response status code should be Errors.OK and the notification is returned
116		Automated	test unsuccessful read notification	Call the method UserManager.readNotification for a non existing user and check the result	The corrisponding user should not be contained the UserModel collection and/or the input for the method should be invalid	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
117		Automated	test successful mark notification as read	Call the method UserManager.MarkNotificationAsRead for an existing user and a valid notification Id and check the result	The related user should already be contained in the UserModel collection and the related notification should already exist	The response status code should be Errors.OK and the flag for the notification is correctly updated
118		Automated	test unsuccessful mark notification as read	Call the method UserManager.MarkNotificationAsRead for a non existing notification id and check the result	Either the user or the notification should be not contained in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
119		Automated	test successful update username	Call the method UserManager.updateUserName with a valid new username for an existing user and check the result	The user should be already contained in the UserModel collection and the new username shoud be valid and not already taken	The response status code should be Errors.OK and the username is correctly updated
120		Automated	test unsuccessful update username	Call the method UserManager.updateUserName with an invalid new username and/or for a non existing user and check the result	The user should not be already contained in the UserModel collection and/or the new username should be invalid/already taken	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
121		Automated	test successful update user kind	Call the method UserManager.updateUserKind with a valid user kind for an existing user and check the result	The user should be already contained in the UserModel collection and the new user kind shoud be valid and not already taken	The response status code should be Errors.OK and the user kind is correctly updated
122		Automated	test unsuccessful update user kind	Call the method UserManager.updateUserKInd with an invalid user kind and/or for a non existing user and check the result	The user should not be already contained in the UserModel collection and/or the new user kind should be invalid	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
123		Automated	test successful update user mail	Call the method UserManager.updateUserMail with valid new mail for an existing user and check the result	The user should be already contained in the UserModel collection and the new mail shoud be valid and not already taken	The response status code should be Errors.OK and the user email is correctly updated
124		Automated	test unsuccessful update user mail	Call the method UserManager.updateUserMail with an invalid new mail and/or for a non existing user and check the result	The user should not be already contained in the UserModel collection and/or the new email should be invalid/already taken	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
125		Automated	test successful create task	Call the method TaskCreator.createTask with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task is correctly created
126		Automated	test unsuccessful create tak	Call the method TaskCreator.createTask with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
127		Automated	test successful create task notes	Call the method TaskCreator.createTasNoteswith valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task notes are correctly created
128		Automated	test unsuccessful create task notes	Call the method TaskCreator.createTaskNotes with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
129		Automated	test successful delete task	Call the method TaskEditor.deleteTask with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task is correctly deleted
130		Automated	test unsuccessful delete task	Call the method TaskEditor.deleteTask with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
131		Automated	test successful delete task assignee	Call the method TaskEditor.deleteTaskAssignee with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task assignee is correctly deleted
132		Automated	test unsuccessful delete task assignee	Call the method TaskEditor.deleteTaskAssignee with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection, assigness should be 2 at least	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
133		Automated	test successful delete task notes	Call the method TaskEditor.deleteTaskNotes with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task notes are correctly deleted
134		Automated	test unsuccessful delete task notes	Call the method TaskEditor.deleteTaskNotes with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed

135	Test TaskManager	Automated	test successful update task	Call the method TaskEditor.updateTask with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task is correctly updated
136		Automated	test unsuccessful update task	Call the method TaskEditor.updateTask with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
137		Automated	test successful update task name	Call the method TaskEditor.updateTaskName with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task name is correctly updated
138		Automated	test unsuccessful update task name	Call the method TaskEditor.updateTaskName with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
139		Automated	test successful update task description	Call the method TaskEditor.updateTaskDescription with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task description is correctly updated
140		Automated	test unsuccessful update task description	Call the method TaskEditor.updateTaskDescription with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
141		Automated	test successful update task status	Call the method TaskEditor.updateTaskStatus with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task status is correctly updated
142		Automated	test successful update task status	Call the method TaskEditor.updateTaskStatus with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
143		Automated	test successful update task notes	Call the method TaskEditor.updateTaskNotes with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task notes are correctly updated
144		Automated	test successful update task notes	Call the method TaskEditor.updateTaskNotes with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
145		Automated	test successful update task assignee	Call the method TaskEditor.updateTaskAssignee with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task assignee is correctly updated
146		Automated	test unsuccessful update task assignee	Call the method TaskEditor.updateTaskAssignee with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
147		Automated	test successful add new assignee	Call the method TaskEditor.addNewAssignee with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection, user should not yet belong to it	The response status code should be Errors.OK and a task tree array is correctly returned
148		Automated	test unsuccessful add new assignee	Call the method TaskEditor.addNewAssignee with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection or user already belongs to it	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
149		Automated	test successful update task manager	Call the method TaskEditor.updateTaskManager with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the manager for the task is correctly updated
150		Automated	test unsuccessful update task manager	Call the method TaskEditor.updateTaskManager with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
151		Automated	test unsuccessful enable notification	Call the method TaskEditor.enableNotification with existing collections as input	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the flag is corretly modified
152		Automated	test unsuccessful enable notification	Call the method TaskEditor.enableNotification with non-existing collections as input	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
153		Automated	test successful disable notification	Call the method TaskEditor.disableNotification with existing collections as input	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the flag is corretly modified
154		Automated	test unsuccessful disable notification	Call the method TaskEditor.disableNotification with non-existing collections as input	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
155		Automated	test successful update task recursive permissions	Call the method TaskEditor.updateTaskRecursivePermissionsValue with existing collections as input	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the permissions are correctly modified
156		Automated	test unsuccessful update task recursive permissions	Call the method TaskEditor.updateTaskRecursivePermissionsValue with non-existing collections as input and/or invalid input	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
157		Automated	test successful get tasks for user	Call the method TaskGetter.getTasksForUser with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the tasks for the user are correctly returned
158		Automated	test unsuccessful get tasks for user	Call the method TaskGetter.getTasksForUser with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
159		Automated	test successful get task id	Call the method TaskGetter.getTask with valid input and existing related collections	The user token should be valid and the inputs should be valid and existing in their corrisponding collection	The response status code should be Errors.OK and the task is correctly returned
160		Automated	test unsuccessful get task id	Call the method TaskGetter.getTask with invalid input and/or one or more non-existing related collection	The user token should not be valid and/or the inputs should be invalid or non existing in their corrisponding collection	The response status code should be Errors.NOT_FOUND or Errors.BAD_REQUEST and an error message is displayed
161	Test Home Page	Manual	test successful get user's tasks	Go to the home page and click on the task related button, then click on the "list" related button	A valid session is being running and both front-end and back-end are correcly working	The task list is correctly displayed
162		Manual	test successful get task visualization info	Go to the home page and click on the task related button, then click on the "visualize" related button	A valid session is being running and both front-end and back-end are correcly working	The task tree hierarchy is correctly displayed
163		Manual	test successful create task	Go to the home page and click on the task related button, then click on the "create new one" related button	A valid session is being running and both front-end and back-end are correcly working	The new task is correctly added and the list is updated
164		Manual	test successful delete task	Go to the home page and click on the task related button, select a task and then click on the "delete" related button	A valid session is being running and both front-end and back-end are correcly working	The selected task is correctly removed and is no longer displayed on the corrisponding list
165		Manual	test successful edit task	Go to the home page and click on the task related button, select a task and then click on the "list" related button	A valid session is being running and both front-end and back-end are correcly working	The selected task is correctly modified and the new attributes are correctly displayed in the corrisponding list
166	Test Notification View	Manual	test delete notification	Go to notification view and see the list of notifications. Then click on one of them and select the delete option	At least one user and one notification should exist in the database	Notifications are deleted and no longer displayed in the notifications list
167		Manual	test mark notification as read	Go to notification view and see the list of notifications. Then click on one of them and mark it as read	At least one user and one unread notification should exist in the database	The notification is marked as read but is still displayed in the notifications list
168		Manual	test get notifications list	Go to notification view and click on notification list to see the list of all notifications	At least one user should exist in the database	The notification list is displayed. If there are no notifications, an empty list should be displayed
169		Manual	test get notifications details	Go to notification view and see the list of notifications. Then click on one of them and select get details	At least one user and one notification should exist in the database	A new page is displayed with all the details about the selected notification (Notification and user ids, notification text, date and read flag)
170	Test Sign In Page	Manual	test successful sign in	Go to the sign in page and insert valid parameters as username and password, then click on the sign in button	The user must already exist in the database but not be already signed in. The credentials should be the correct ones associated to that user	The user is correctly logged in and is taken to the home page
171		Manual	test unsuccessful sign in	Go to the sign in page and insert incorrect parameters as username and password, then click on the sign in button	The user must not be already signed in	The user is not logged in and an error message is displayed. The user is taken back to the login page
172	Test Sign Up Page	Manual	test successful sign up	Go to the sign up page and insert all the requested valid parameters, then click on the sign up button	The user should not already exist in the database. The parameters should be valid	The user is correctly siged up and is taken to the home page
173		Manual	test unsuccessful sign up	Go to the sign up page and insert invalid parameters or leave some parameters blank, then click on the sign up button	The user should not already exist in the database	The user is not signed up and an error message is displayed, saying if there are missing parameters or invalid ones. The user is taken back to the sign up page
174	Test Edit Profile	Manual	test successful change password	Go to the edit profile page, select change password, in the first textbox insert the old password and then in the second textbox the new password	The user must be already signed in	The user is redirected to the edit profile page, the new password is set
175		Manual	test unsuccessful change password	Go to the edit profile page, select change password, in the first textbox insert a wrong password and then in the second textbox the new password	The user must be already signed in	The user gets a wrong password message and the page is reloaded.
176		Manual	test usuccessful change password	Go to the edit profile page, select change password, in the first textbox insert the old password and then in the second textbox an old password	The user must be already signed in	The user gets a "You can't use an old password" message, the page is reloaded

177		Manual	test successful delete user account	Go to the edit profile page, select delete account, insert your password and then click on the confirm button	The user must be already signed in	The user is redirected to the home page, the account has been deleted successfully
178		Manual	test unsuccessful delete user account	Go to the edit profile page, select delete account, insert a wrong password and then click on the confirm button	The user must be already signed in	The user gets a wrong password message and the page is reloaded.
179	Test Organization View	Manual	test add user to organization	Go to the Organization view page, select add user to organization, search the new user to add, select it and click on the confirm button	Organization and new User must be already defined in the database. The Organization manager must be signed in	The new user is added into the Organization, Organization manager is redirected into Organization view page
180		Manual	test delete Organization	Go to the Organization view page, select delete Organization button, insert password, and click to confirm button	Organization must be already defined in the database. The Organization manager must be signed in	The Owner gets redirected into the home page and the Organization gets removed from the database, all task and reports included
181		Manual	test unsuccessful delete Organization	Go to the Organization view page, select delete Organization button, insert wrong password, and click to confirm button	Organization must be already defined in the database. The Organization manager must be signed in	The Owner gets a wrong password message and the page gets reloaded
182		Manual	test successful change payment method	Go to the Organization view page, select change payment method, select the payment method of choise, authenticate into new selected bank account, confirm	Organization must be already defined in the database. The Organization manager must be signed in, connection with the bank must be authenticated	The Owner gets redirected into the Organization view page, new payment method is set
183		Manual	test unsuccessful change payment method	Go to the Organization view page, select change payment method, select the payment method of choise, fail to authenticate into new selected bank account	Organization must be already defined in the database. The Organization manager must be signed in	The Owner gets redirected into the Organization view page, all changes are reverted