Deliverable 3

Team Name: Lambda++

Team Number: 18

Sponsor: EPI-USE

Project title: EPI-USE Game Farm Manager

Team Members

Gage Nott

Ariel Sischy

Kevin Kiyo

Abuzar Bulbulia

Background

The Melorane Game Reserve (1) is a habitat for elephants and rhinos that works in tandem with the Barokologadi Community. The goals of the reserve are to create economic opportunities for the community and to serve as a nature reserve for the wildlife. ERP (Elephant, Rhinos and People) (2), the nonprofit organization in charge of this reserve, started this project in 2021 and has a 90-year lease, meaning this reserve is a long-term strategic project. ERP (working under GroupElephant.com (3)) also plans to franchise this model for future conservation efforts.

Currently, the reserve typically handles fewer than 30 concurrent employees, but with a high likelihood of expansion. The employees often work far apart, spread across the reserve, but often work together in teams on tasks. There is currently only a single administrator.

Problem Statement

It is difficult for the sole administrator to manage activities across the reserve. There is no formal system in place to assign activities to ERP employees and track progress, leading to miscommunication and disorganization. Furthermore, there is no system to track the various employees' skill sets and availability, resulting in scheduling conflicts and mismanaged skillsets when tasks are assigned. This leads to wasted time and effort to sort out these issues. Finally, there are no formal means to track and request stock and equipment for a task or for the administrator to manage these resources. This, again, leads to wasted time and effort due to conflicts over resources.

Proposed Solution

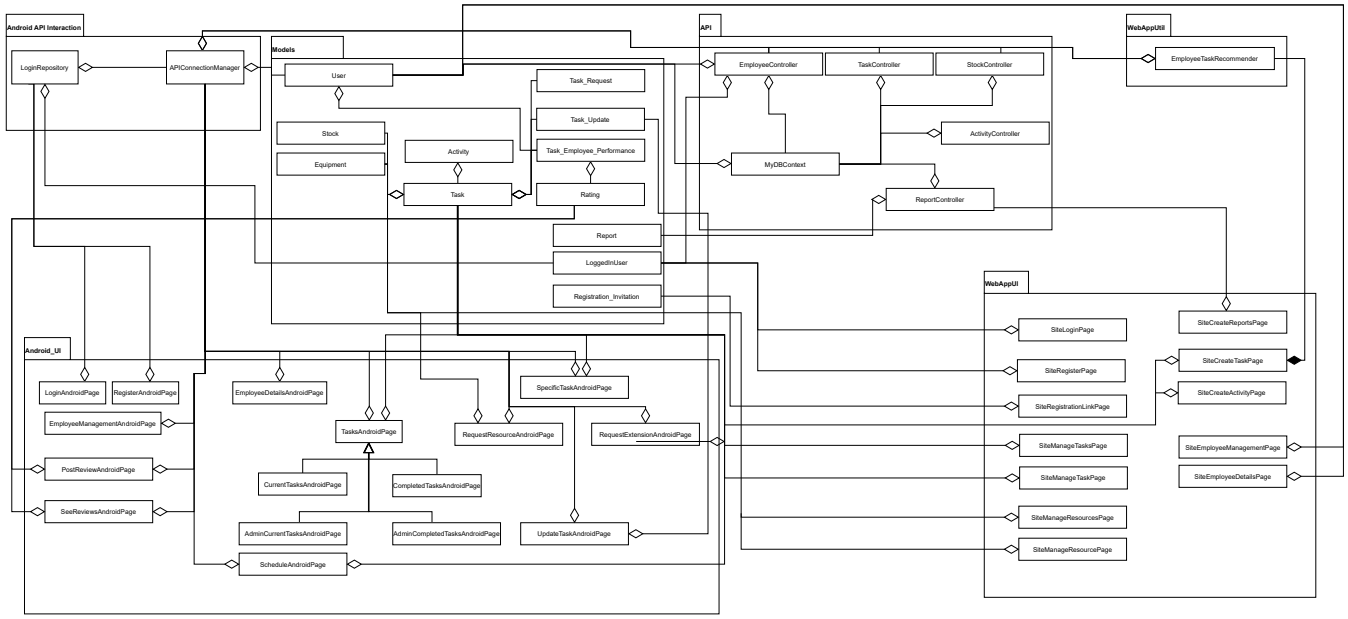
We seek to develop a system that will enable the management of work activities and supplies. This will consist of an administrative site to create and monitor activities and stock, as well as a mobile application to allow employees to keep track of the activities they do and changes to the stock. By the construction of the

administrative site, it should be much easier for the administrator to manage the completed, in progress, and upcoming farm activities and keep track of what is going on. The extensive reports the system generates should empower the administrator to make decisions based on up-to-date information and the monitored performance of employees. The ability to track the stock and equipment will allow informed decisions to be made about what stock to buy, how to schedule the usage of limited equipment, and keep track of which tasks use certain resources and how much.

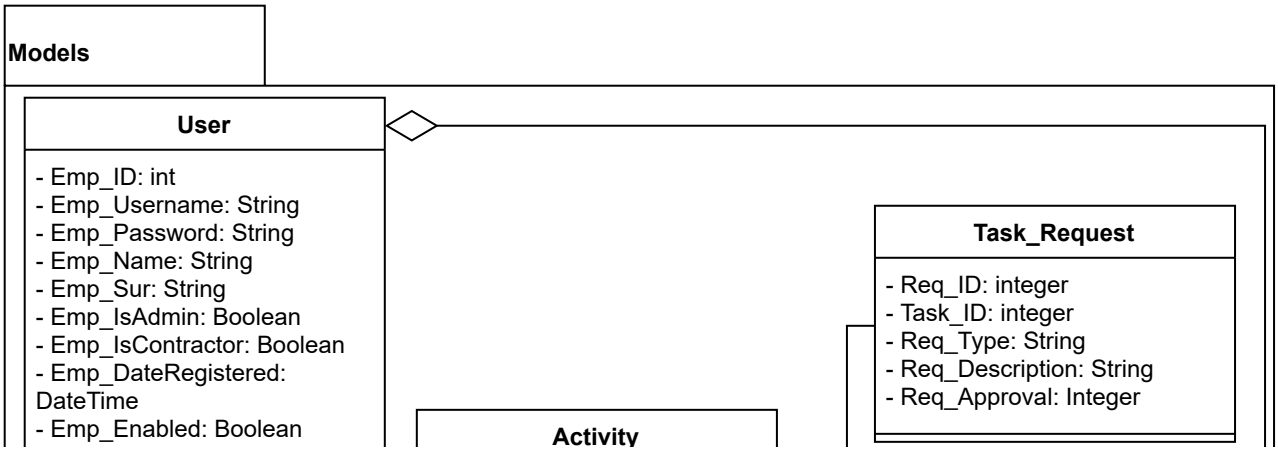
The mobile application will enable employees to interact with this system on the go, as they are typically outdoors across the reserve. They will be able to keep track of what tasks they are assigned to, whom they are working with, the associated locations and deadlines, the status of the task and any updates that are made to it. It will allow for clear and concise communication on the task's progress between task members themselves and with the administrator. It will also allow for formalised requests to be made to the administrator, for example for stock/equipment, or for extensions to the deadlines. An administrative branch of the app will also be required as the administrator is often outside too. This will allow for them to be kept up to date on the status of the tasks and any requests made by employees.

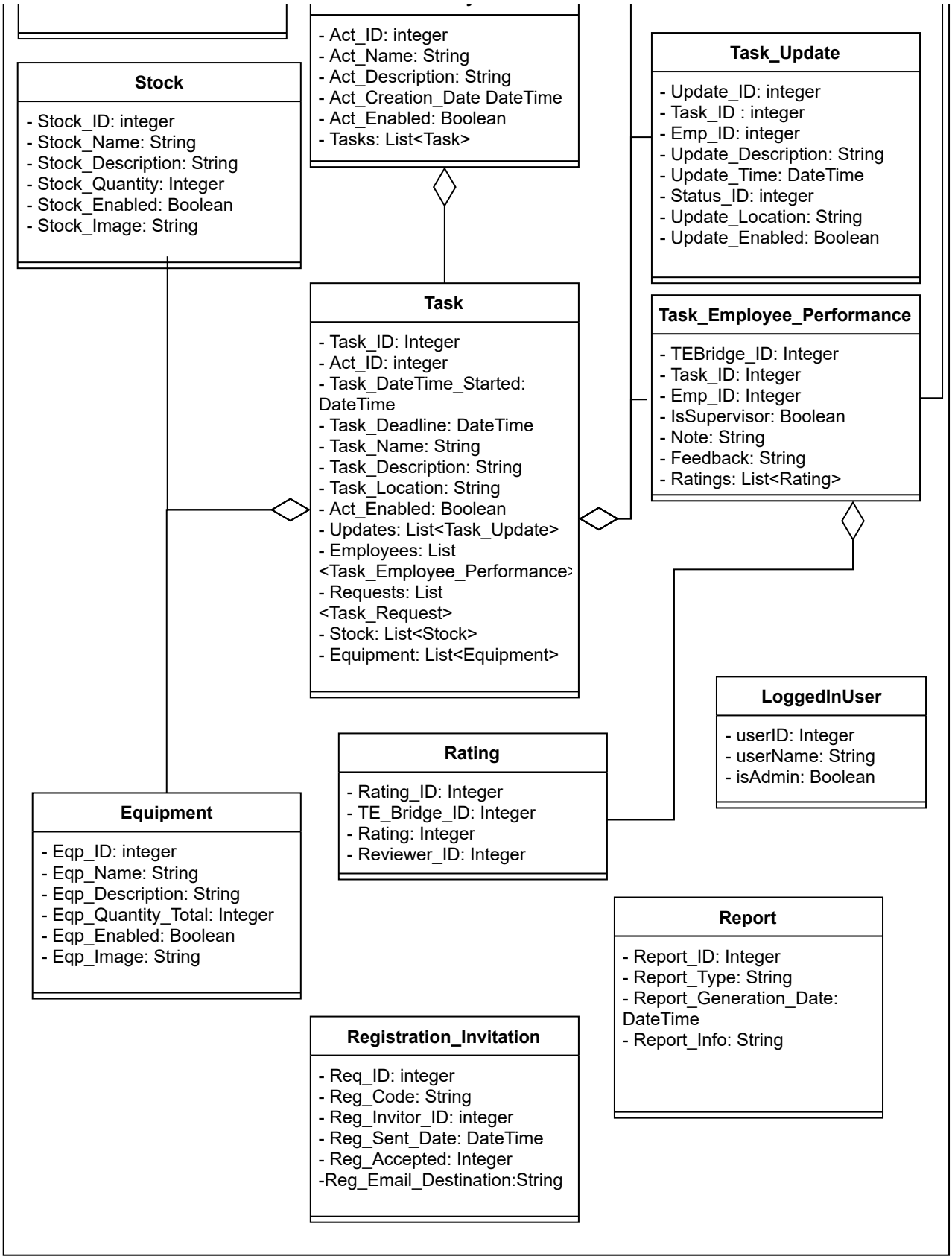
Class Diagrams

Overview and relationships

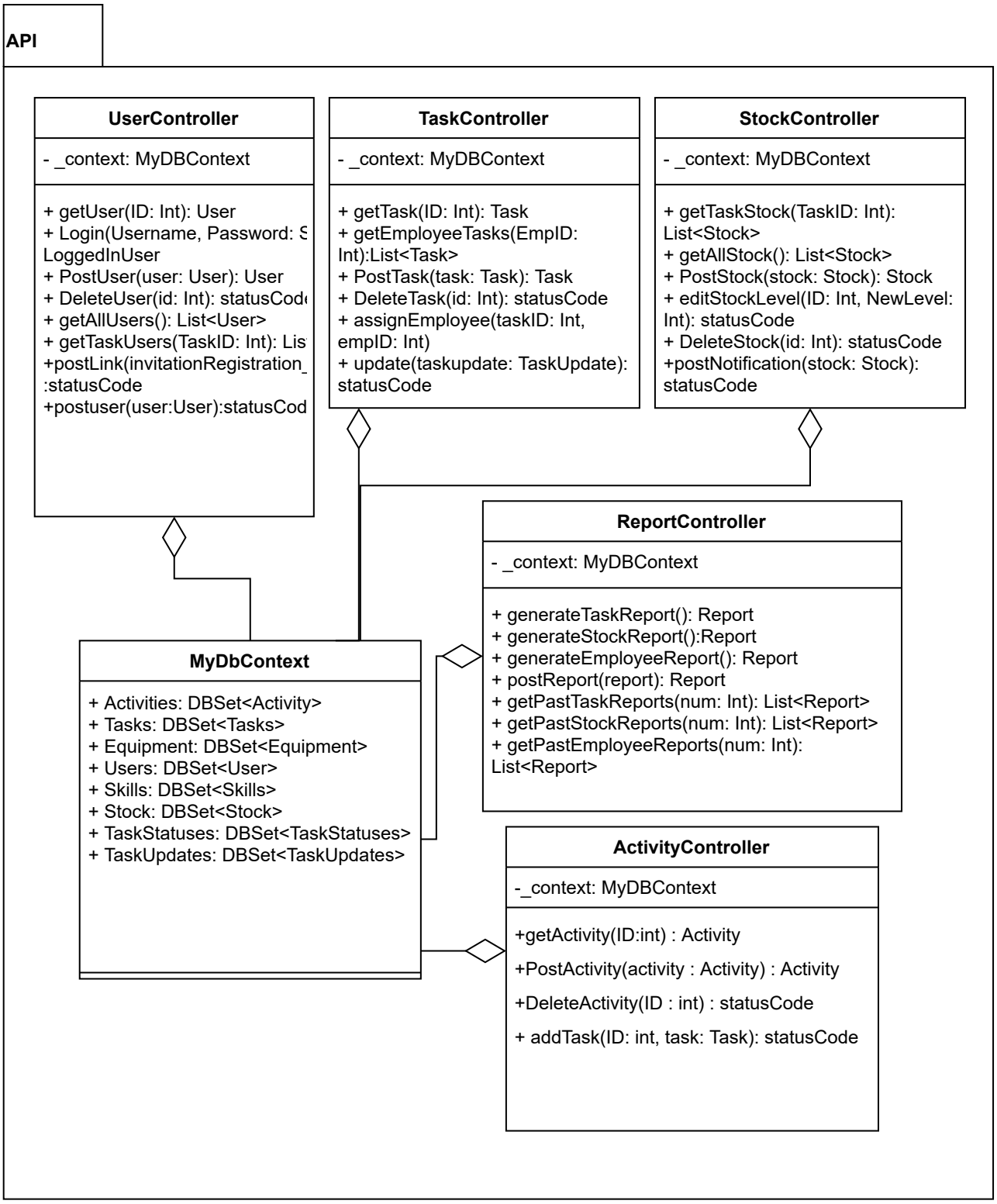


Models

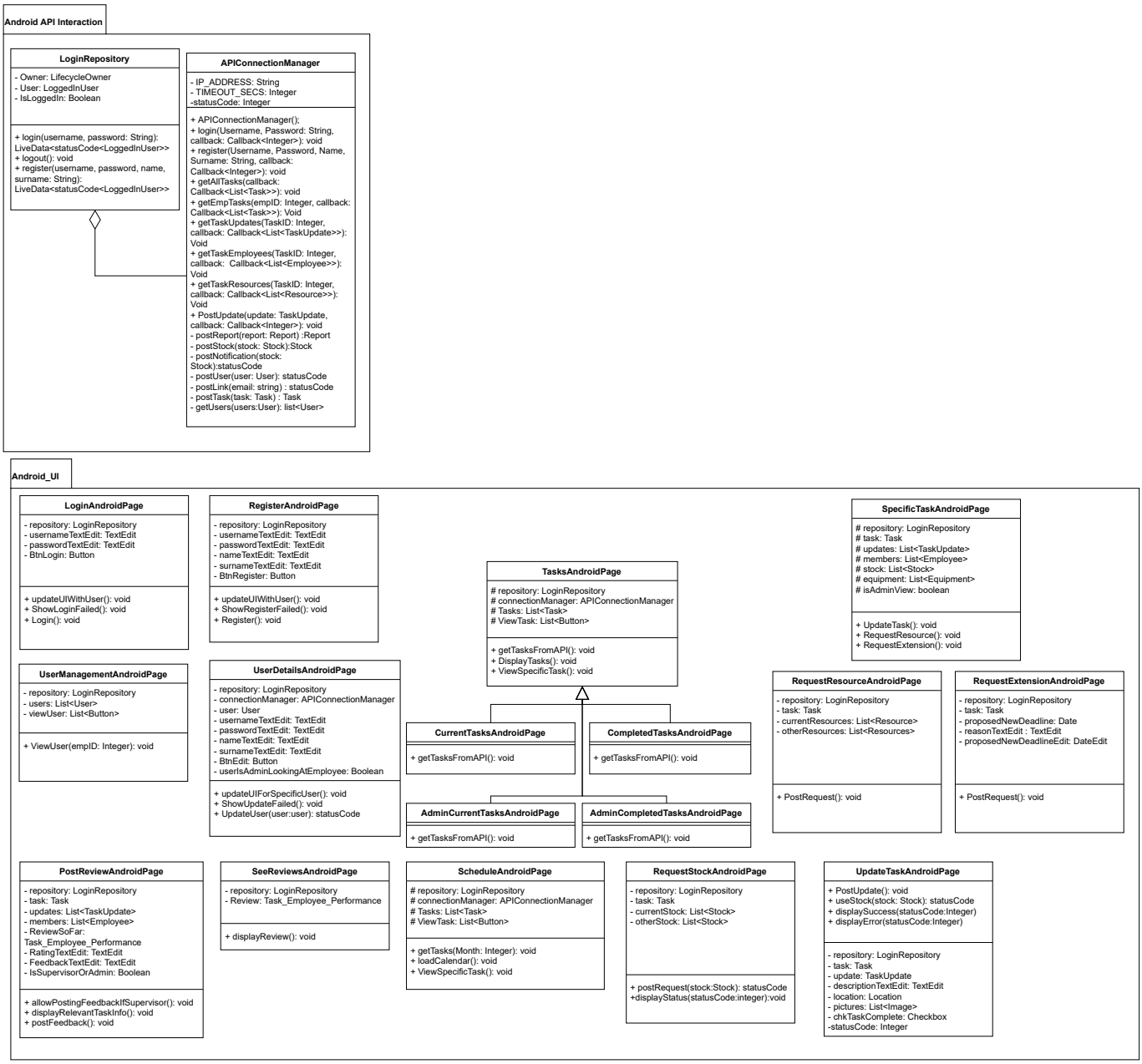



- Act_ID: integer
- Act_Name: String
- Act_Description: String
- Act_Creation_Date DateTime
- Act_Enabled: Boolean
- Tasks: List<Task>
Stock
- Stock_ID: integer
- Stock_Name: String
- Stock_Description: String
- Stock_Quantity: Integer
- Stock_Enabled: Boolean
- Stock_Image: String
Task_Update
- Update_ID: integer
- Task_ID : integer
- Emp_ID: integer
- Update_Description: String
- Update_Time: DateTime
- Status_ID: integer
- Update_Location: String
- Update_Enabled: Boolean
Task
- Task_ID: Integer
- Act_ID: integer
- Task_DateTime_Started: DateTime
- Task_Deadline: DateTime
- Task_Name: String
- Task_Description: String
- Task_Location: String
- Act_Enabled: Boolean
- Updates: List<Task_Update>
- Employees: List <Task_Employee_Performance>
- Requests: List <Task_Request>
- Stock: List<Stock>
- Equipment: List<Equipment>
Task_Employee_Performance
- TEBridge_ID: Integer
- Task_ID: Integer
- Emp_ID: Integer
- IsSupervisor: Boolean
- Note: String
- Feedback: String
- Ratings: List<Rating>
LoggedInUser
- userID: Integer
- userName: String
- isAdmin: Boolean
Rating
- Rating_ID: Integer
- TE_Bridge_ID: Integer
- Rating: Integer
- Reviewer_ID: Integer
Equipment
- Eqp_ID: integer
- Eqp_Name: String
- Eqp_Description: String
- Eqp_Quantity_Total: Integer
- Eqp_Enabled: Boolean
- Eqp_Image: String
Report
- Report_ID: Integer
- Report_Type: String
- Report_Generation_Date: DateTime
- Report_Info: String
Registration_Invitation
- Req_ID: integer
- Reg_Code: String
- Reg_Invitor_ID: integer
- Reg_Sent_Date: DateTime
- Reg_Accepted: Integer
-Reg_Email_Destination:String

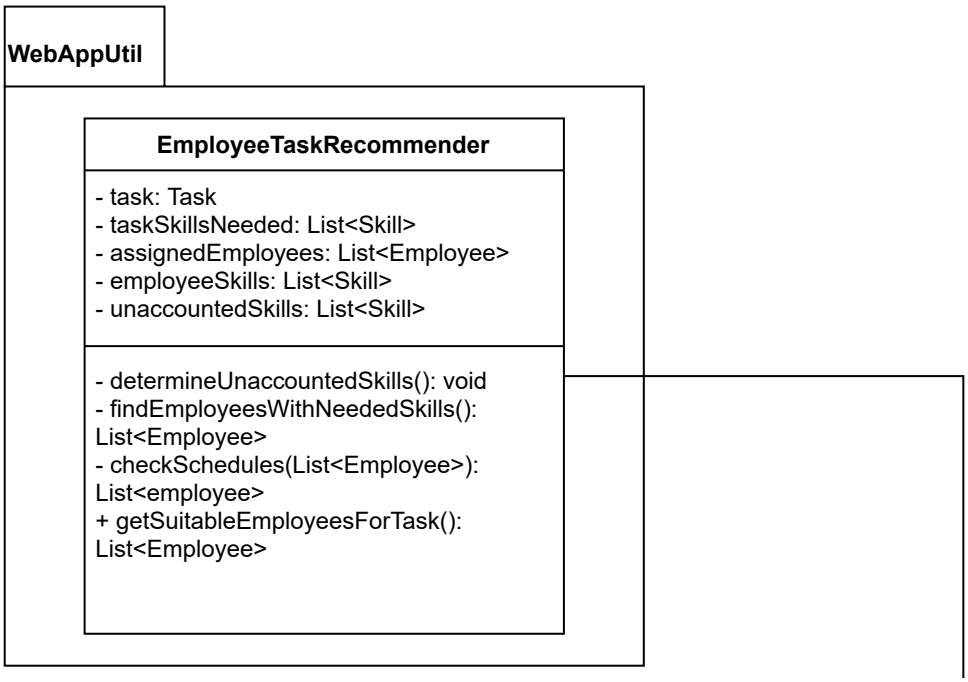
API

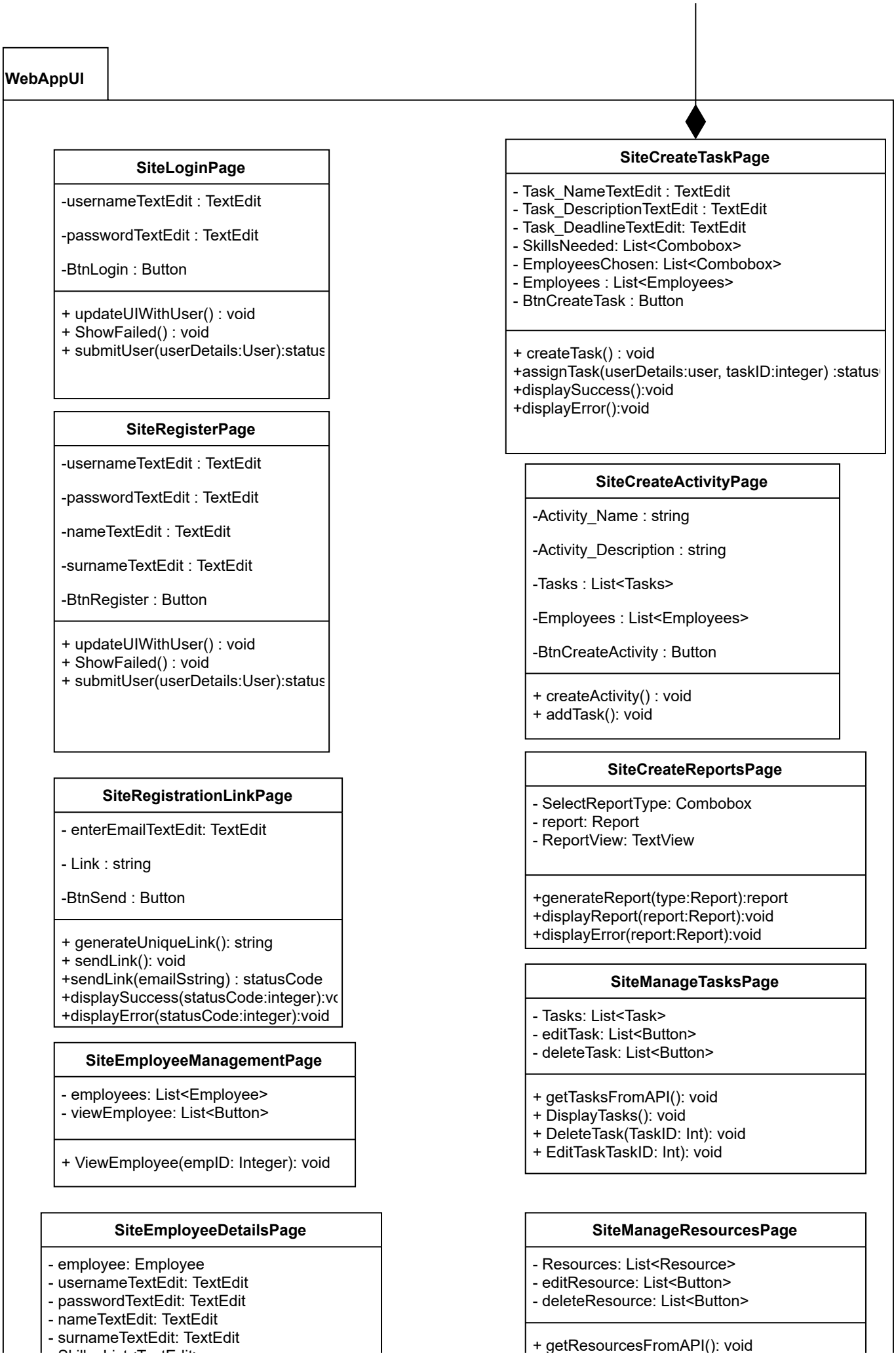

API
UserController
- _context: MyDBContext
+ getUser(ID: Int): User
+ Login(Username, Password: S
LoggedInUser
+ PostUser(user: User): User
+ DeleteUser(id: Int): statusCod
+ getAllUsers(): List<User>
+ getTaskUsers(TaskID: Int): Lis
+postLink(invitationRegistration_
:statusCode
+postuser(user:User):statusCod
TaskController
- _context: MyDBContext
+ getTask(ID: Int): Task
+ getEmployeeTasks(EmpID:
Int):List<Task>
+ PostTask(task: Task): Task
+ DeleteTask(id: Int): statusCode
+ assignEmployee(taskID: Int,
empID: Int)
+ update(taskupdate: TaskUpdate):
statusCode
StockController
- _context: MyDBContext
+ getTaskStock(TaskID: Int):
List<Stock>
+ getAllStock(): List<Stock>
+ PostStock(stock: Stock): Stock
+ editStockLevel(ID: Int, NewLevel:
Int): statusCode
+ DeleteStock(id: Int): statusCode
+postNotification(stock: Stock):
statusCode
ReportController
- _context: MyDBContext
+ generateTaskReport(): Report
+ generateStockReport():Report
+ generateEmployeeReport(): Report
+ postReport(report): Report
+ getPastTaskReports(num: Int): List<Report>
+ getPastStockReports(num: Int): List<Report>
+ getPastEmployeeReports(num: Int):
List<Report>
MyDbContext
+ Activities: DBSet<Activity>
+ Tasks: DBSet<Tasks>
+ Equipment: DBSet<Equipment>
+ Users: DBSet<User>
+ Skills: DBSet<Skills>
+ Stock: DBSet<Stock>
+ TaskStatuses: DBSet<TaskStatuses>
+ TaskUpdates: DBSet<TaskUpdates>
ActivityController
-_context: MyDBContext
+getActivity(ID:int) : Activity
+PostActivity(activity : Activity) : Activity
+DeleteActivity(ID : int) : statusCode
+ addTask(ID: int, task: Task): statusCode

Android



Web Application




WebAppUI
SiteLoginPage
-usernameTextEdit : TextEdit
-passwordTextEdit : TextEdit
-BtnLogin : Button
+ updateUIWithUser() : void
+ ShowFailed() : void
+ submitUser(userDetails:User):status
SiteRegisterPage
-usernameTextEdit : TextEdit
-passwordTextEdit : TextEdit
-nameTextEdit : TextEdit
-surnameTextEdit : TextEdit
-BtnRegister : Button
+ updateUIWithUser() : void
+ ShowFailed() : void
+ submitUser(userDetails:User):status
SiteRegistrationLinkPage
- enterEmailTextEdit: TextEdit
- Link : string
-BtnSend : Button
+ generateUniqueLink(): string
+ sendLink(): void
+sendLink(emailSstring) : statusCode
+displaySuccess(statusCode:integer):vc
+displayError(statusCode:integer):void
SiteEmployeeManagementPage
- employees: List<Employee>
- viewEmployee: List<Button>
+ ViewEmployee(empID: Integer): void
SiteEmployeeDetailsPage
- employee: Employee
- usernameTextEdit: TextEdit
- passwordTextEdit: TextEdit
- nameTextEdit: TextEdit
- surnameTextEdit: TextEdit
SiteCreateTaskPage
- Task_NameTextEdit : TextEdit
- Task_DescriptionTextEdit : TextEdit
- Task_DeadlineTextEdit: TextEdit
- SkillsNeeded: List<Combobox>
- EmployeesChosen: List<Combobox>
- Employees : List<Employees>
- BtnCreateTask : Button
+ createTask() : void
+assignTask(userDetails:user, taskID:integer) :status
+displaySuccess():void
+displayError():void
SiteCreateActivityPage
-Activity_Name : string
-Activity_Description : string
-Tasks : List<Tasks>
-Employees : List<Employees>
-BtnCreateActivity : Button
+ createActivity() : void
+ addTask(): void
SiteCreateReportsPage
- SelectReportType: Combobox
- report: Report
- ReportView: TextView
+generateReport(type:Report):report
+displayReport(report:Report):void
+displayError(report:Report):void
SiteManageTasksPage
- Tasks: List<Task>
- editTask: List<Button>
- deleteTask: List<Button>
+ getTasksFromAPI(): void
+ DisplayTasks(): void
+ DeleteTask(TaskID: Int): void
+ EditTaskTaskID: Int): void
SiteManageResourcesPage
- Resources: List<Resource>
- editResource: List<Button>
- deleteResource: List<Button>
+ getResourcesFromAPI(): void

- Skills: List<TextEdit>
- Images: List<Picture>
- BtnEdit: Button

+ updateUIForSpecificUser(): void
+ ShowUpdateFailed(): void
+ UpdateEmployee(): void
-deleteEmployee(id:int):statusCode
-displaySuccess(statusCode:int):void
-displayError(statusCode:int):void

+ DisplayResources(): void
+ DeleteResource(RID: Int): void
-displayStock(stock:Stock):void
-displayError(stock:Stock):void
+modifyStock(stock:Stock):stock

SiteManageTaskPage

- Task: Task
- editTask: Button
- deleteTask: Button
- nameTextEdit : TextEdit
- LocationTextEdit : TextEdit
- DescriptionTextEdit : TextEdit
- Employees: List<Combobox>
- SkillsNeeded: List<Combobox>

+ DeleteTask(): void
+ EditTask(): void

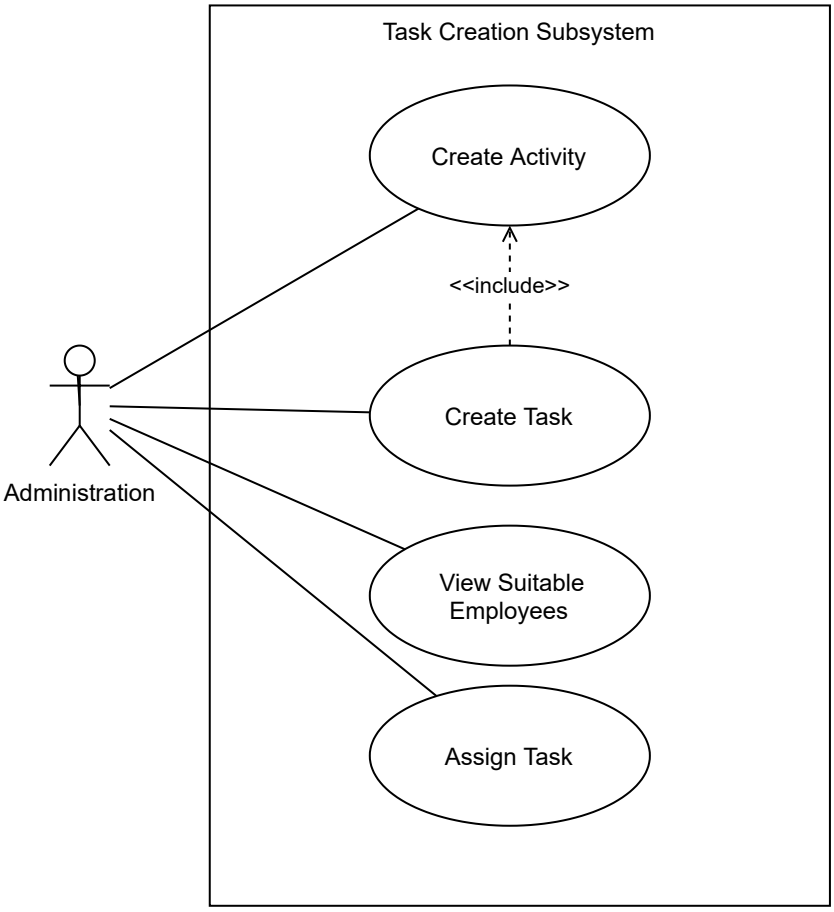
SiteManageResourcePage

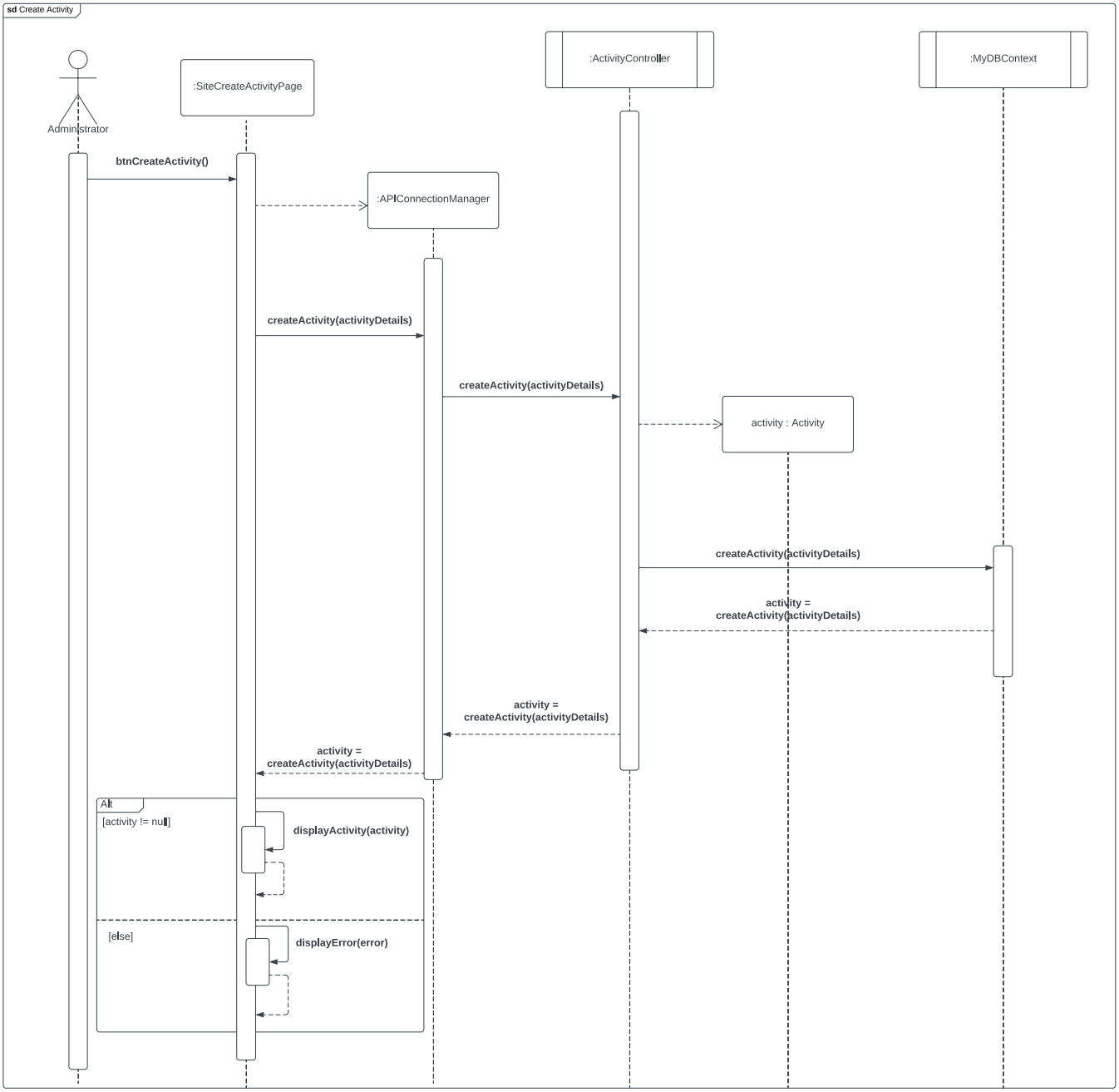
- Resource: Resource
- editResource: Button
- deleteResource: Button
- nameTextEdit : TextEdit
- QuantityTextEdit : TextEdit
- DescriptionTextEdit : TextEdit

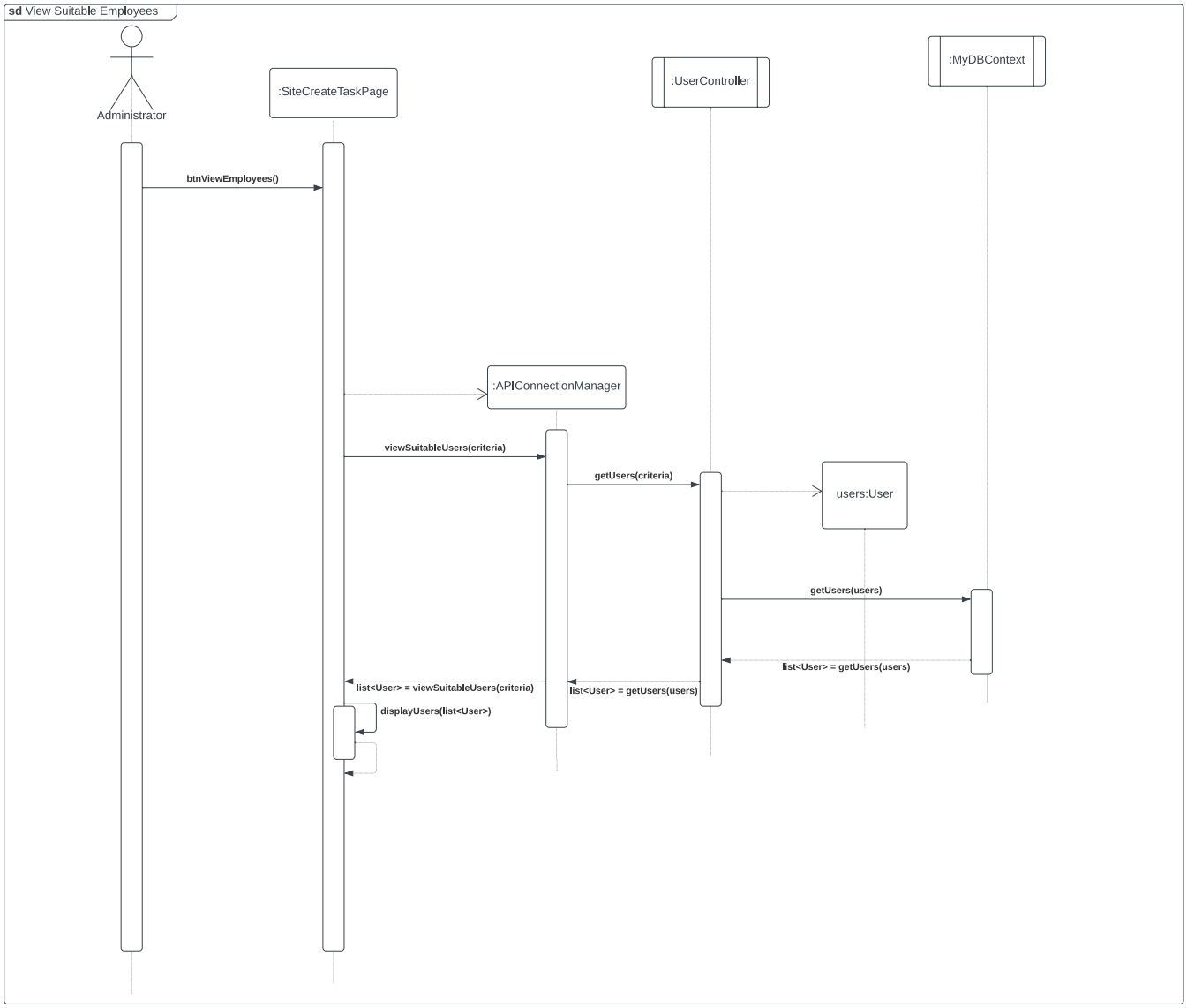
+ DeleteResource(): void
+ EditResource(): void

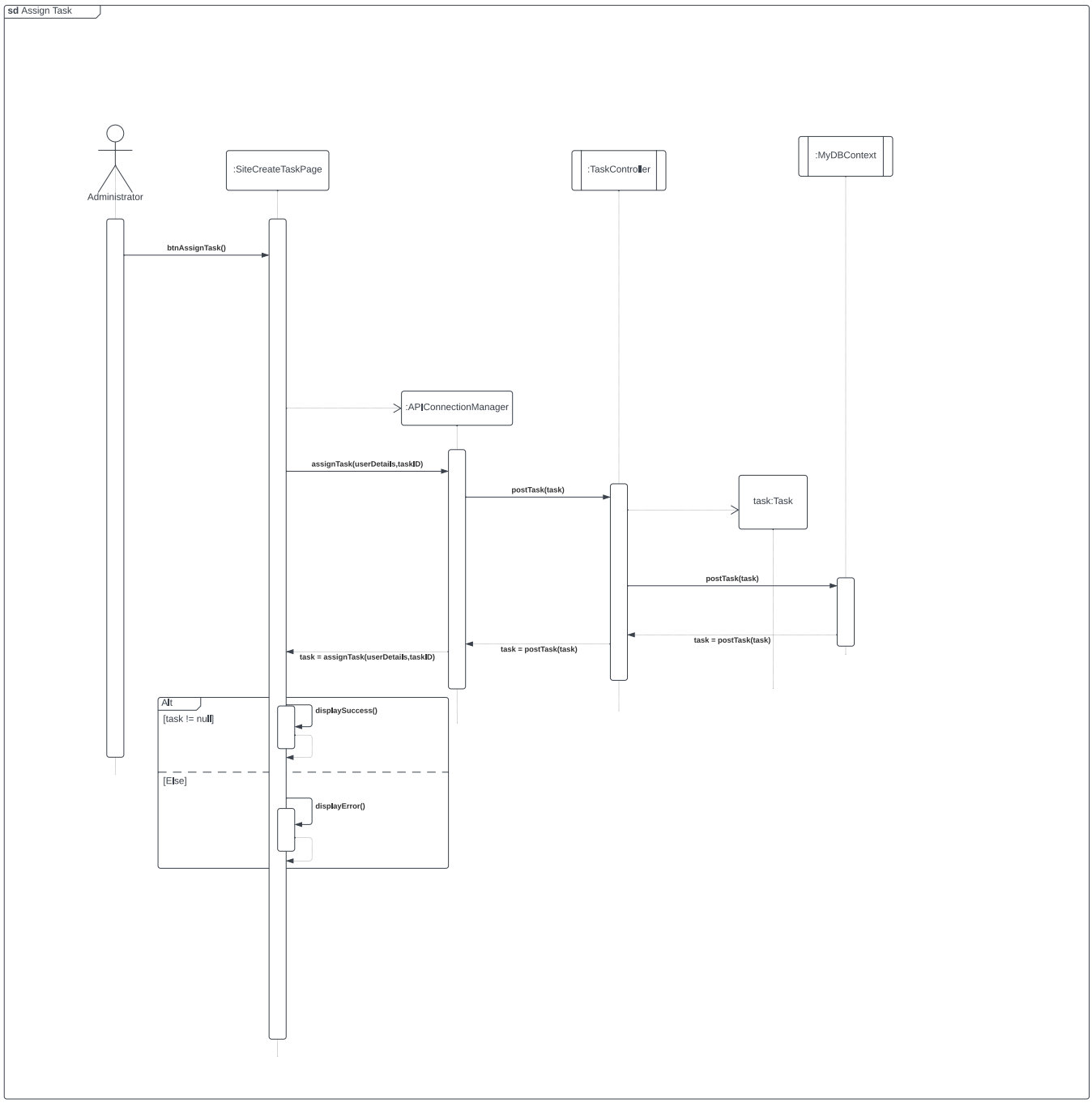
Interaction Sequence Diagrams

Activity & Task Creation

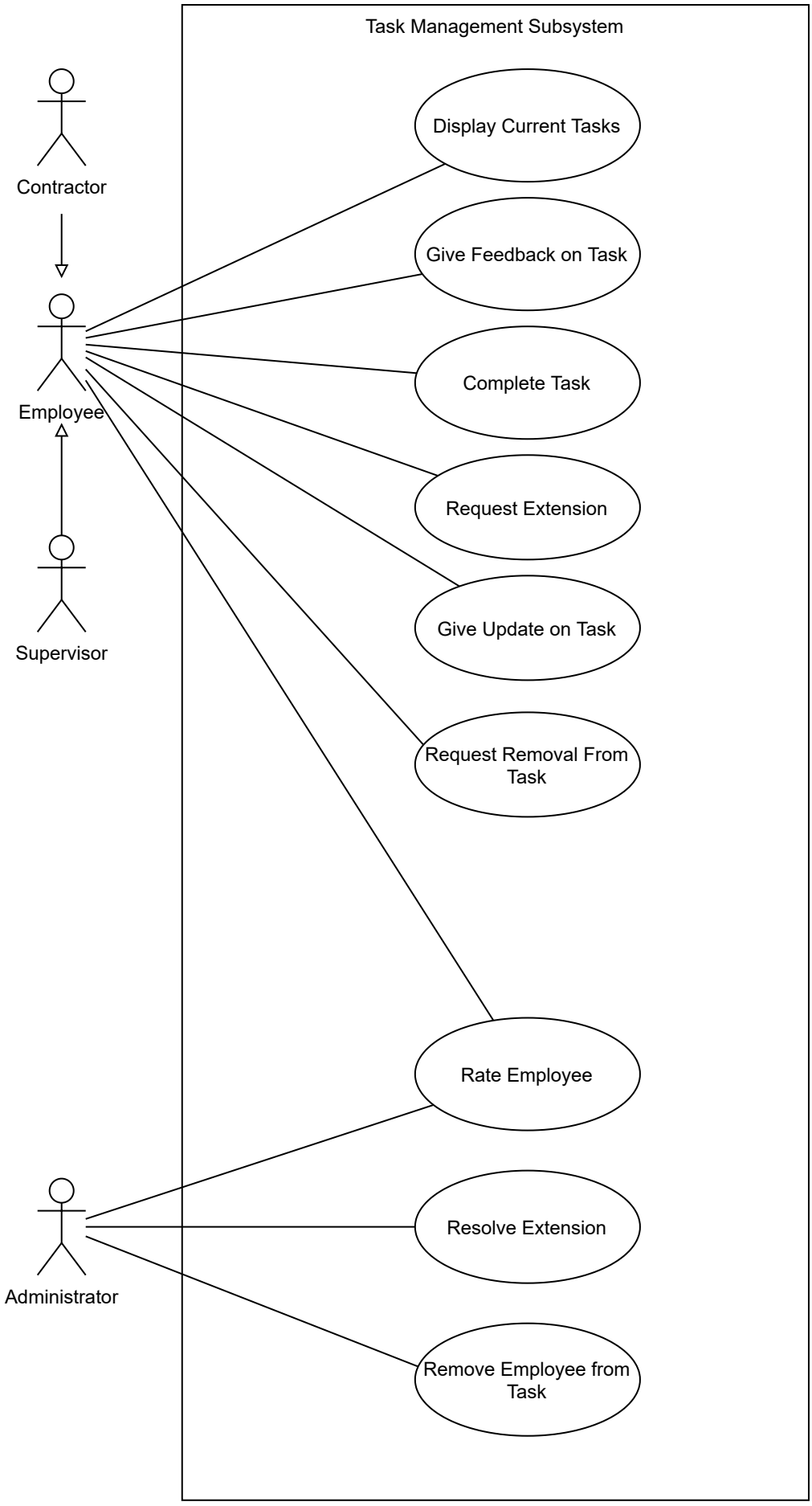


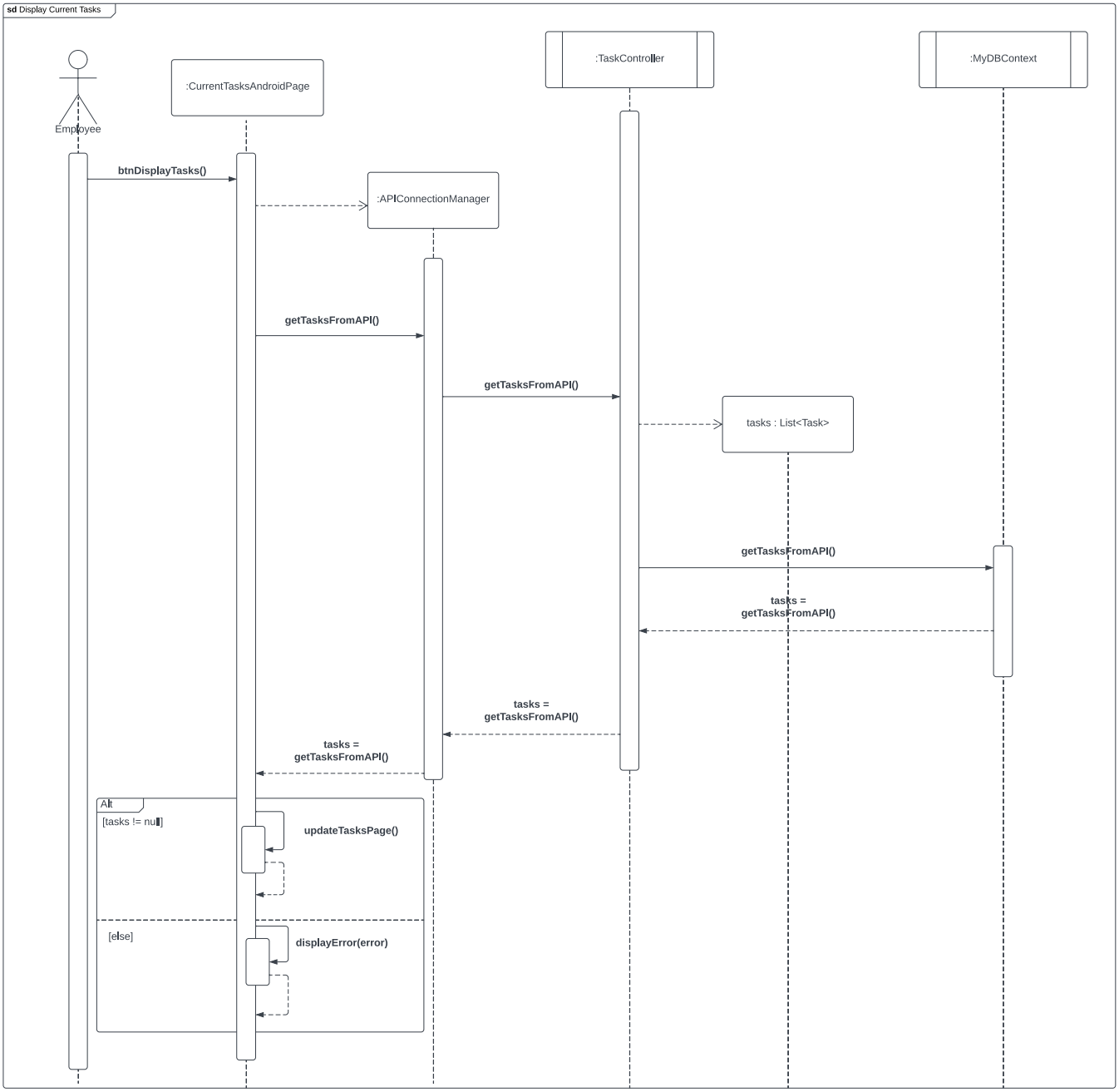

sd Create Activity
Administrator
:SiteCreateActivityPage
:APIConnectionManager
:ActivityController
:MyDBContext
activity : Activity
btnCreateActivity()
createActivity(activityDetails)
createActivity(activityDetails)
createActivity(activityDetails)
activity =
createActivity(activityDetails)
activity =
createActivity(activityDetails)
activity =
createActivity(activityDetails)
Alt
[activity != null]
displayActivity(activity)
[else]
displayError(error)

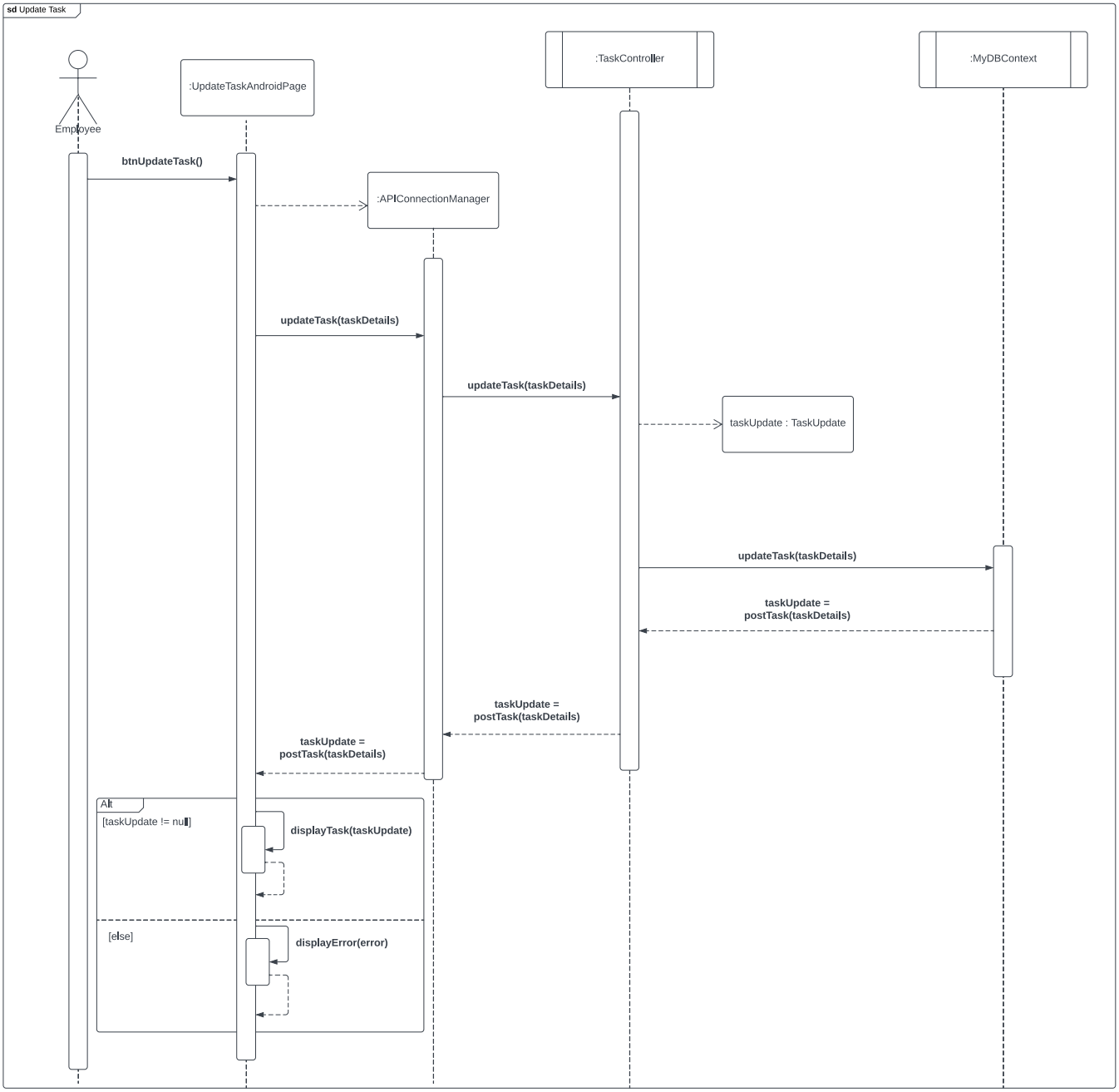

sd View Suitable Employees
Administrator
:SiteCreateTaskPage
:UserControlller
:MyDBContext
btnViewEmployees()
:APIConnectionManager
viewSuitableUsers(criteria)
getUsers(criteria)
users:User
getUsers(users)
list<User> = getUsers(users)
list<User> = viewSuitableUsers(criteria)
list<User> = getUsers(users)
displayUsers(list<User>)

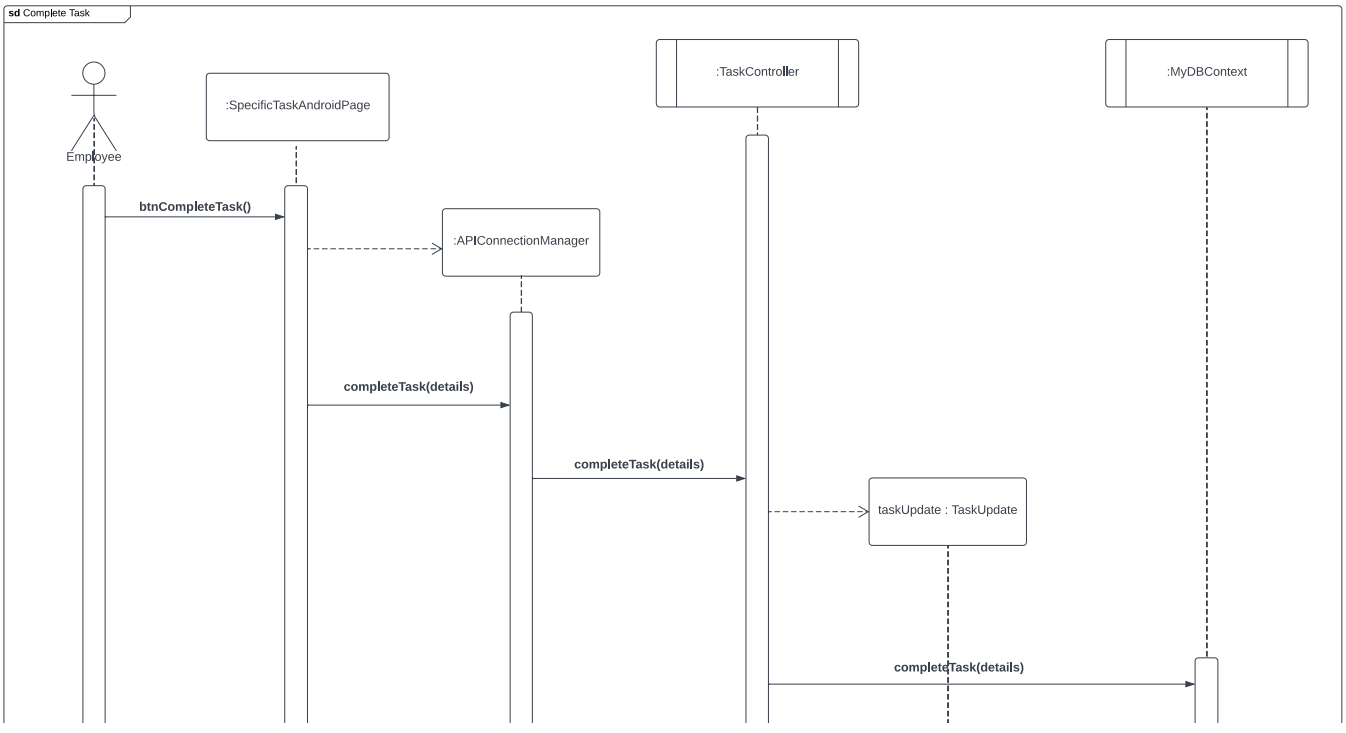


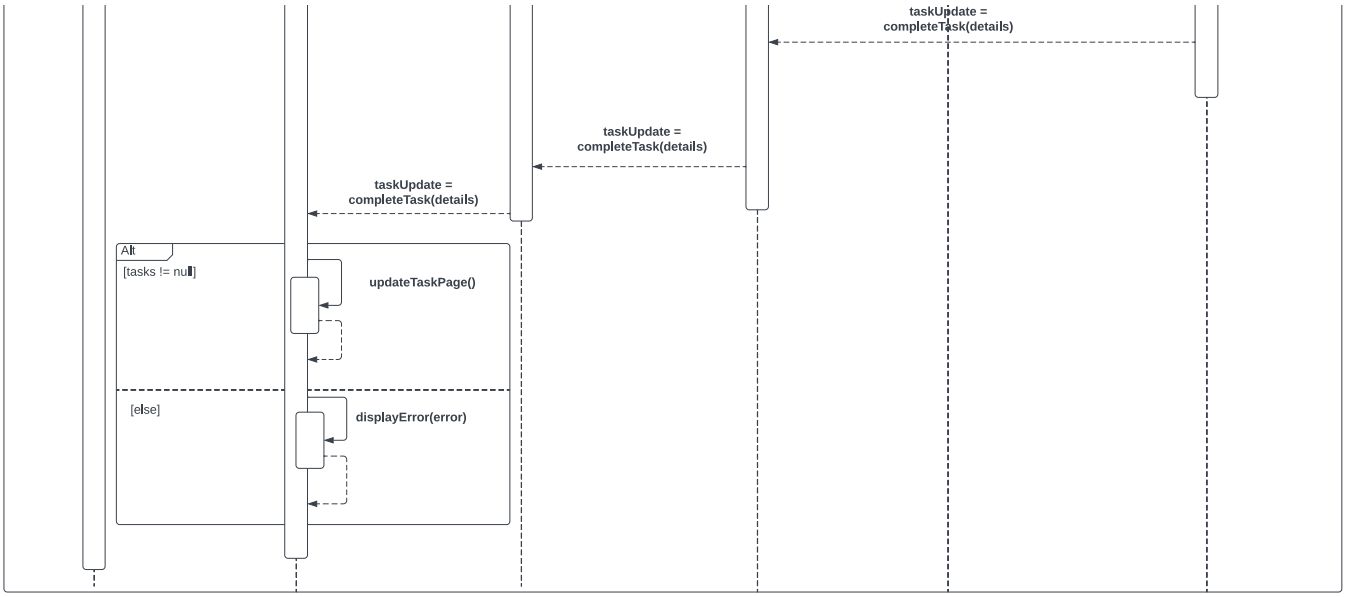
Task Management

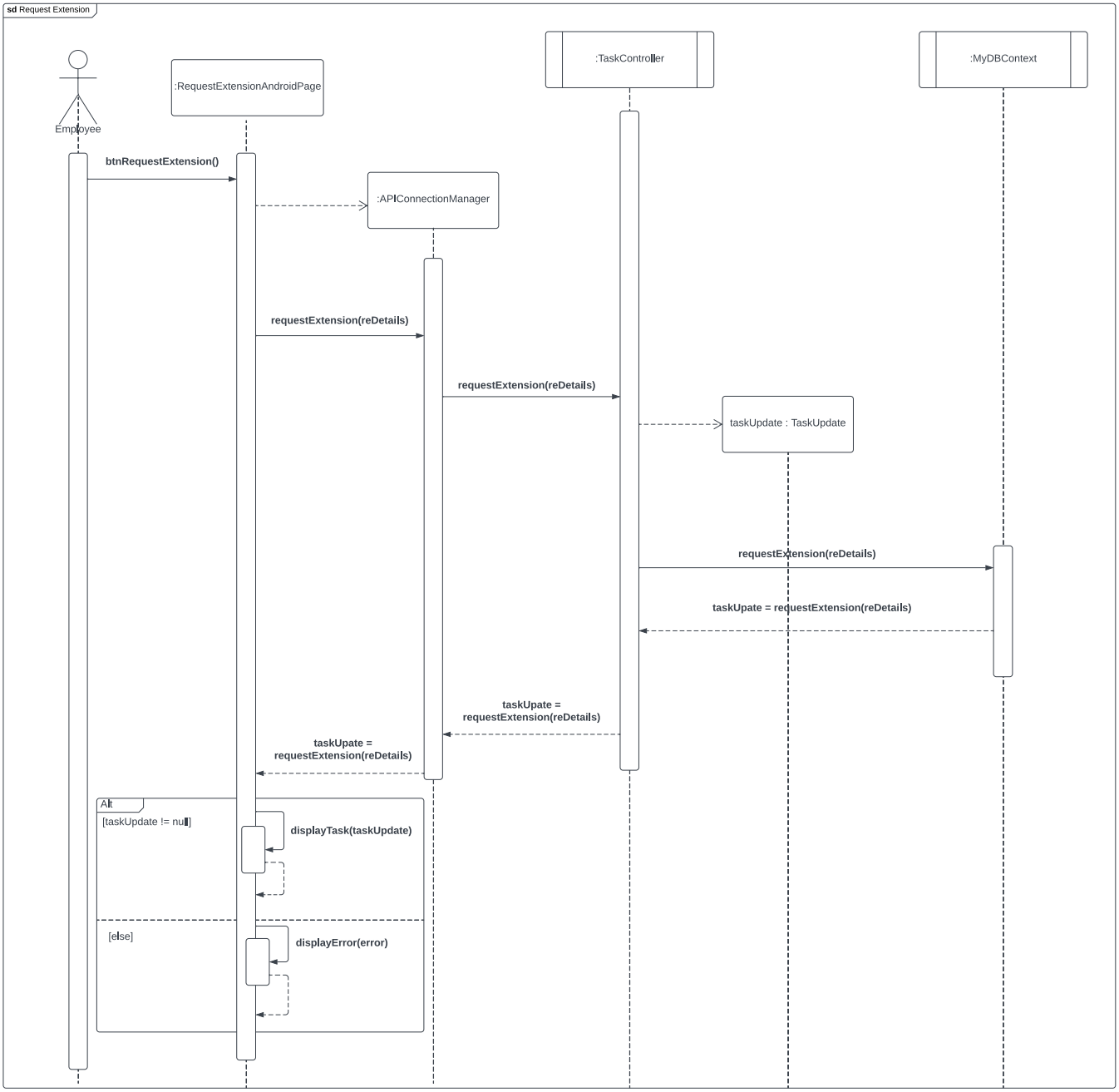

Task Management Subsystem
Contractor
Employee
Supervisor
Administrator
Display Current Tasks
Give Feedback on Task
Complete Task
Request Extension
Give Update on Task
Request Removal From Task
Rate Employee
Resolve Extension
Remove Employee from Task

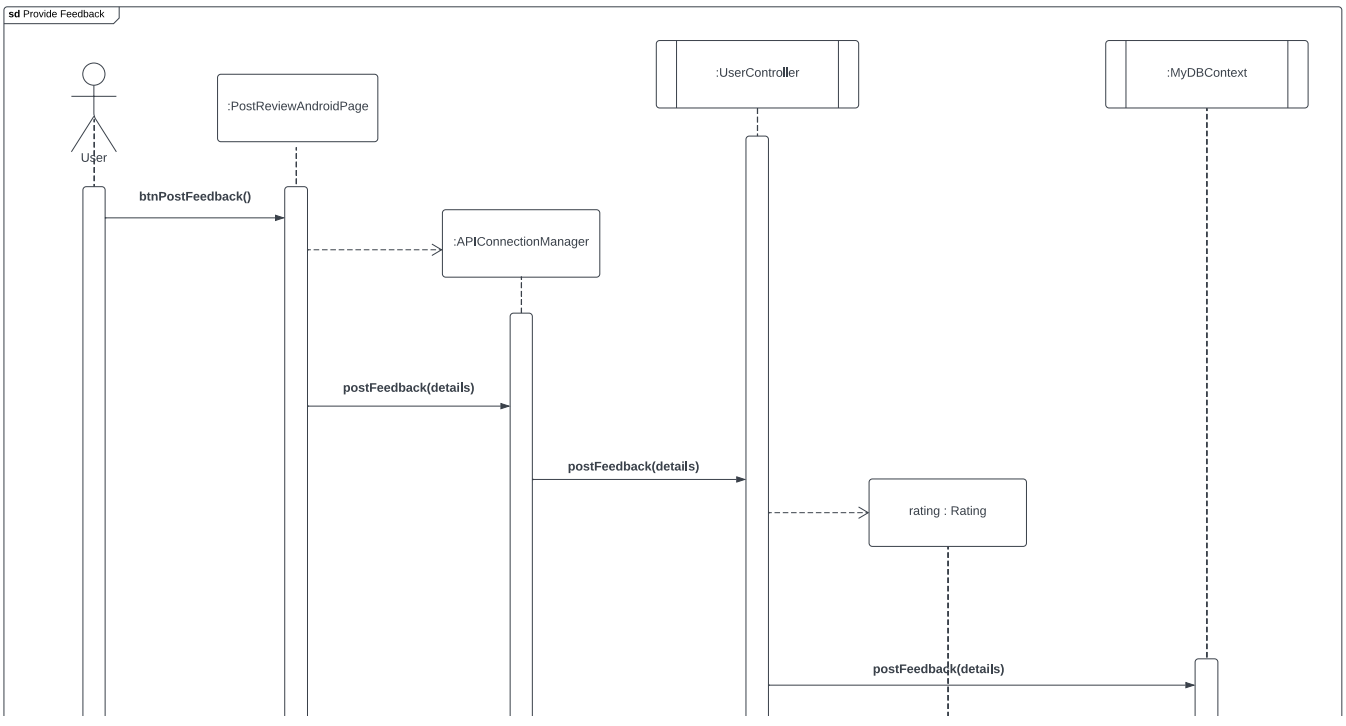

sd Display Current Tasks
Employee
:CurrentTasksAndroidPage
:TaskController
:MyDBContext
btnDisplayTasks()
:APIConnectionManager
getTasksFromAPI()
getTasksFromAPI()
tasks : List<Task>
getTasksFromAPI()
tasks =
getTasksFromAPI()
tasks =
getTasksFromAPI()
tasks =
getTasksFromAPI()
Alt
[tasks != null]
updateTasksPage()
[else]
displayError(error)

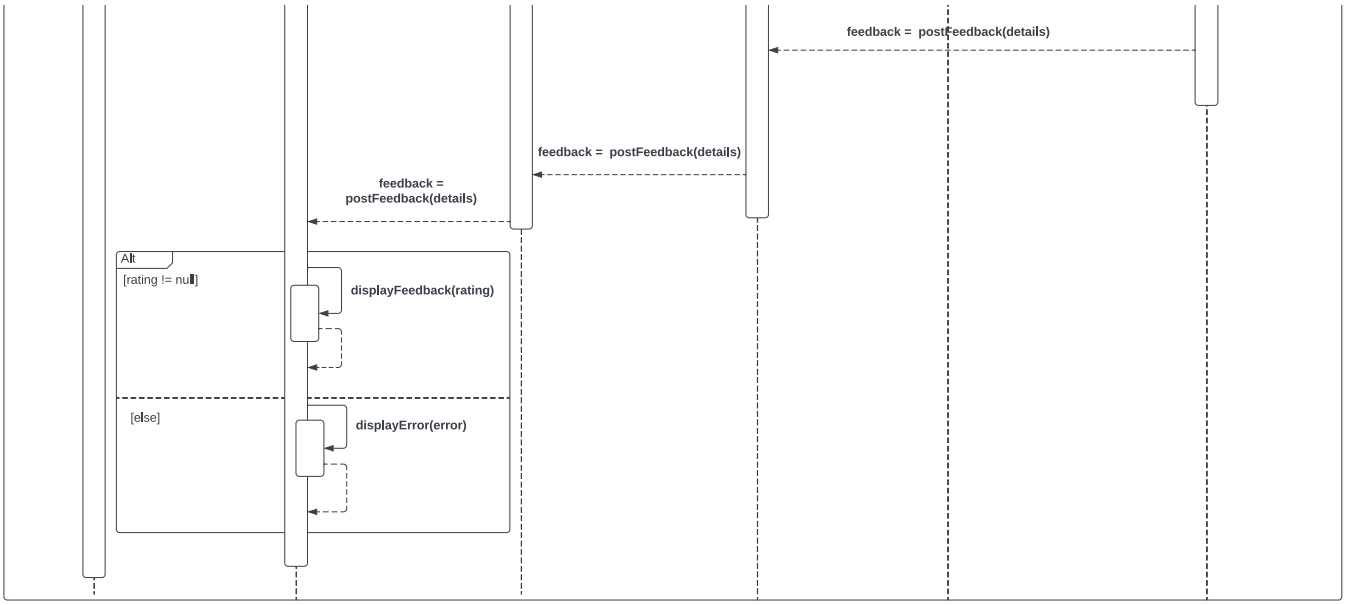

sd Update Task
Employee
:UpdateTaskAndroidPage
:TaskController
:MyDBContext
btnUpdateTask()
:APIConnectionManager
updateTask(taskDetails)
updateTask(taskDetails)
taskUpdate : TaskUpdate
updateTask(taskDetails)
taskUpdate =
postTask(taskDetails)
taskUpdate =
postTask(taskDetails)
taskUpdate =
postTask(taskDetails)
Alt
[taskUpdate != null]
displayTask(taskUpdate)
[else]
displayError(error)


sd Complete Task
Employee
:SpecificTaskAndroidPage
:TaskController
:MyDBContext
btnCompleteTask()
:APIConnectionManager
completeTask(details)
completeTask(details)
taskUpdate : TaskUpdate
completeTask(details)

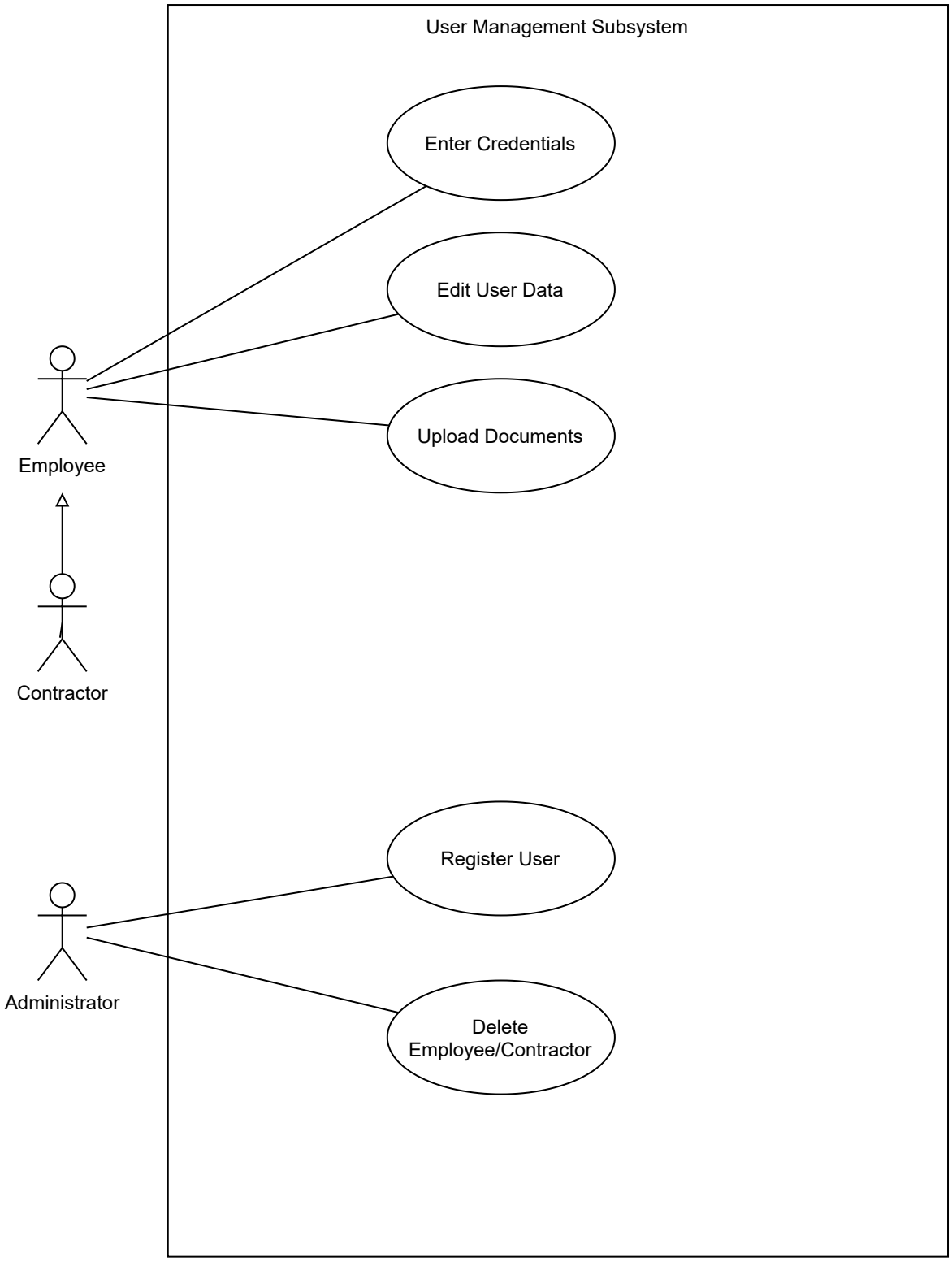

taskUpdate =
completeTask(details)
taskUpdate =
completeTask(details)
taskUpdate =
completeTask(details)
Alt
[tasks != null]
updateTaskPage()
[else]
displayError(error)


sd Request Extension
Employee
:RequestExtensionAndroidPage
:TaskController
:MyDBContext
btnRequestExtension()
:APIConnectionManager
requestExtension(reDetails)
requestExtension(reDetails)
taskUpdate : TaskUpdate
requestExtension(reDetails)
taskUpate = requestExtension(reDetails)
taskUpate = requestExtension(reDetails)
taskUpate = requestExtension(reDetails)
Alt
[taskUpdate != null]
displayTask(taskUpdate)
[else]
displayError(error)

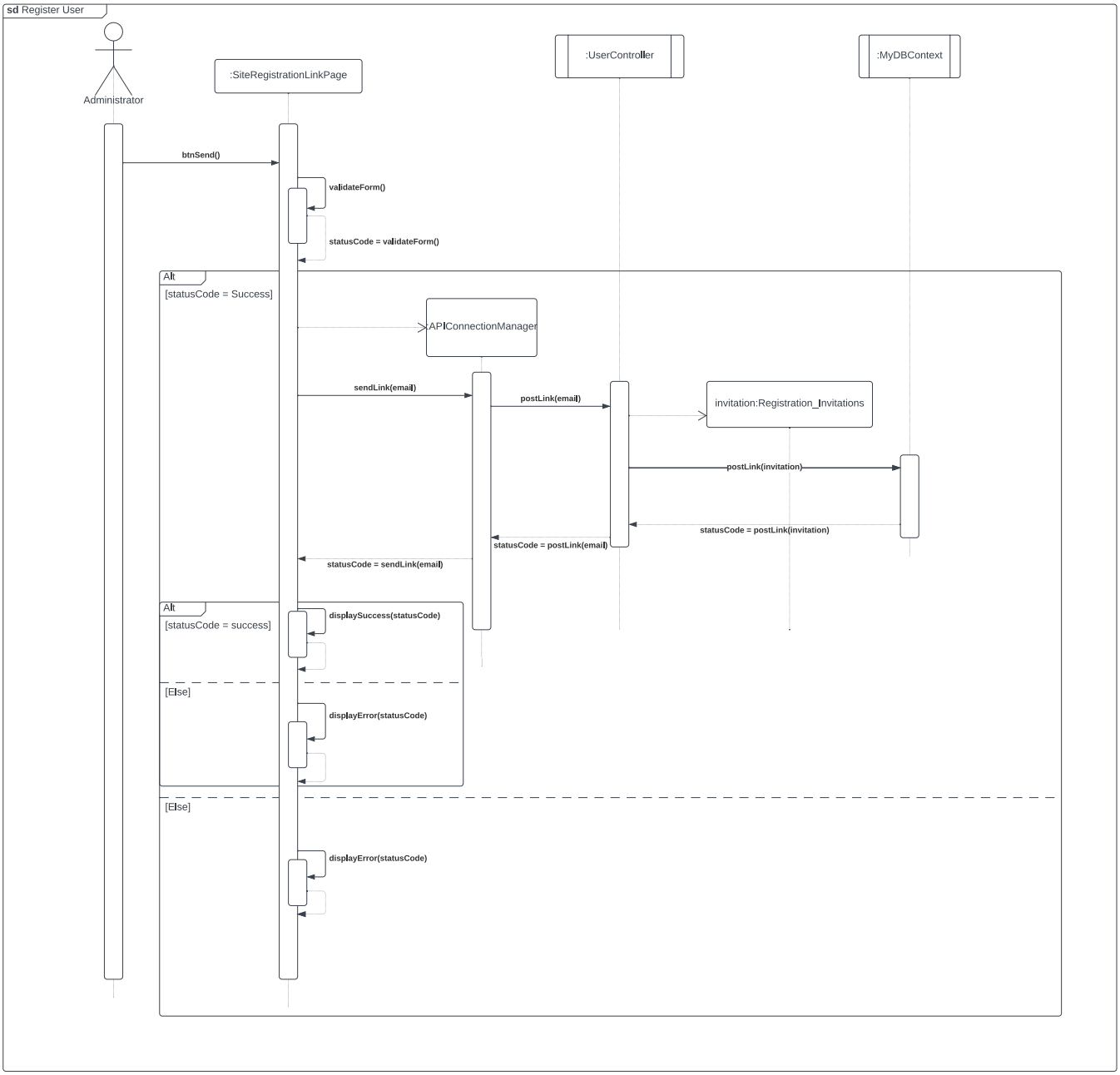

sd Provide Feedback
User
:PostReviewAndroidPage
:UserController
:MyDBContext
btnPostFeedback()
:APIConnectionManager
postFeedback(details)
postFeedback(details)
rating : Rating
postFeedback(details)

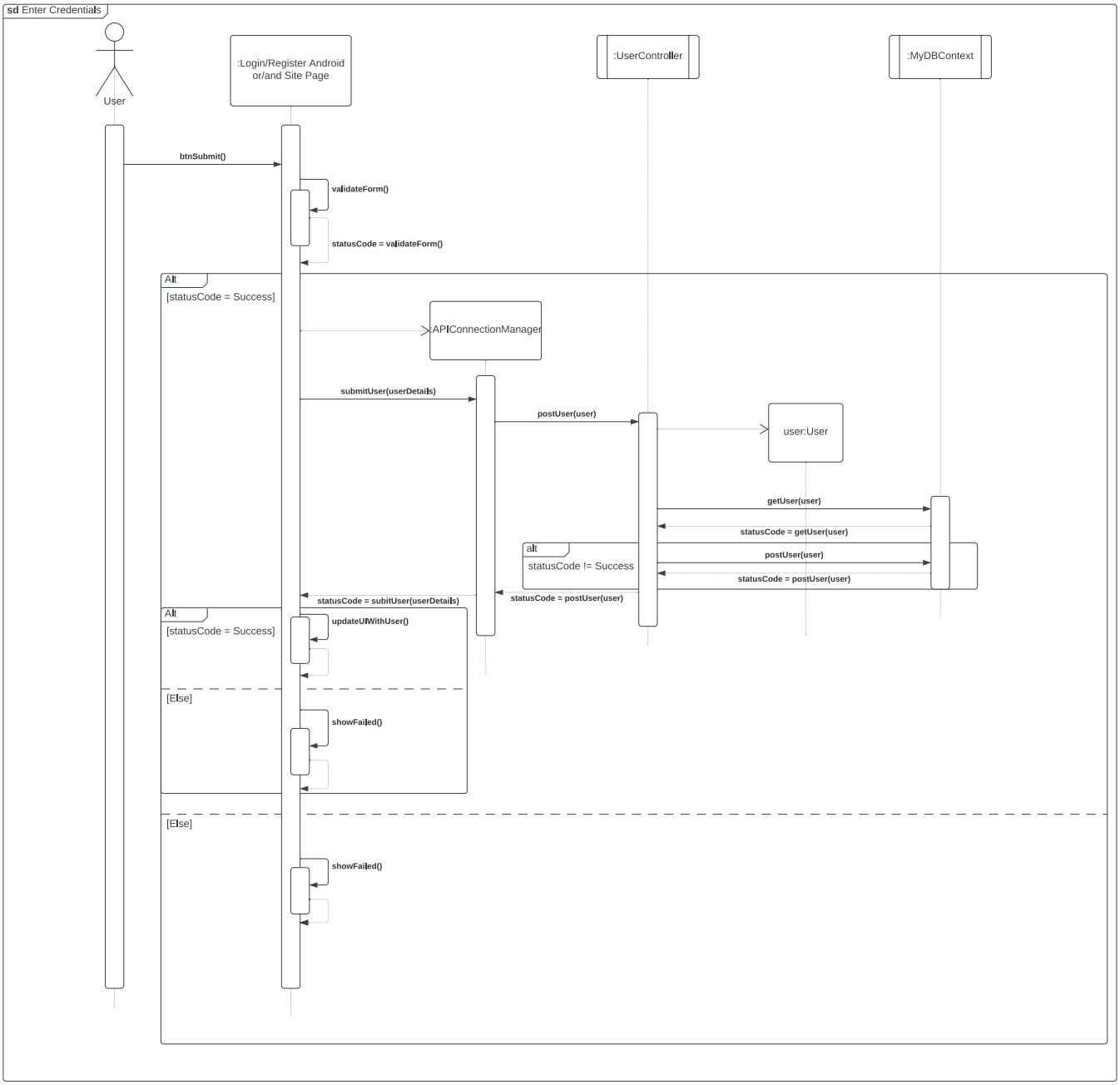


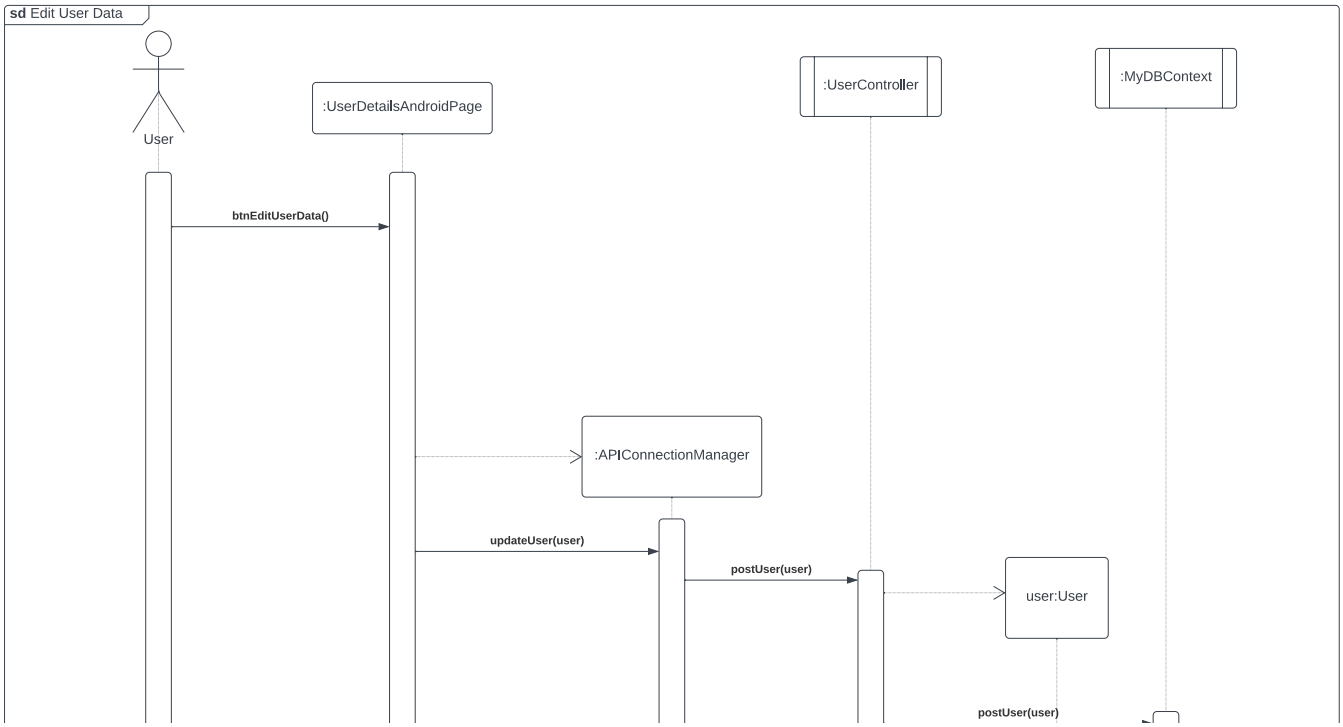
User Management

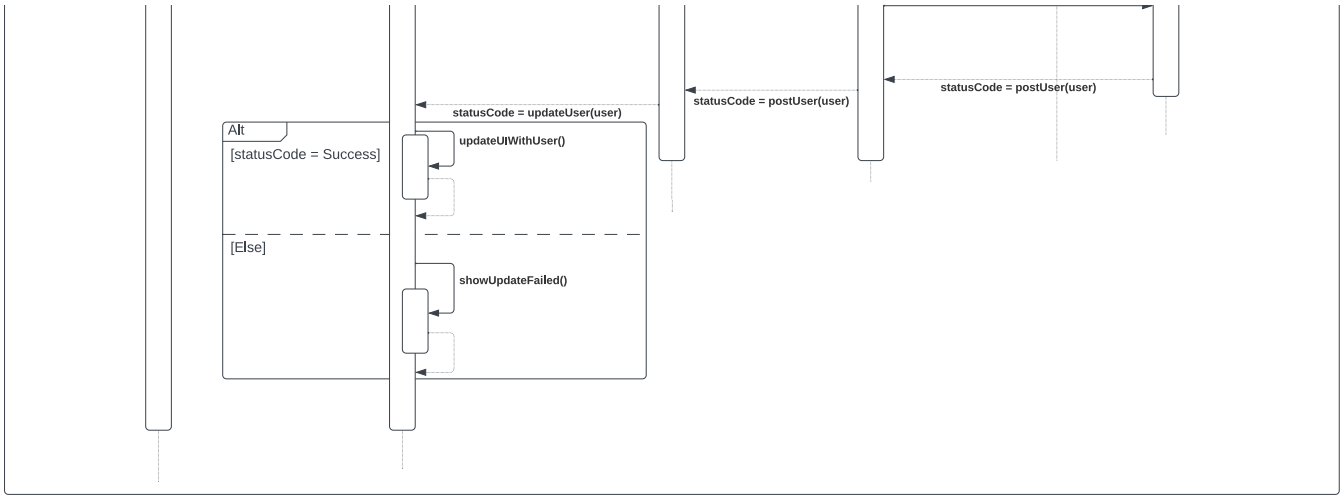


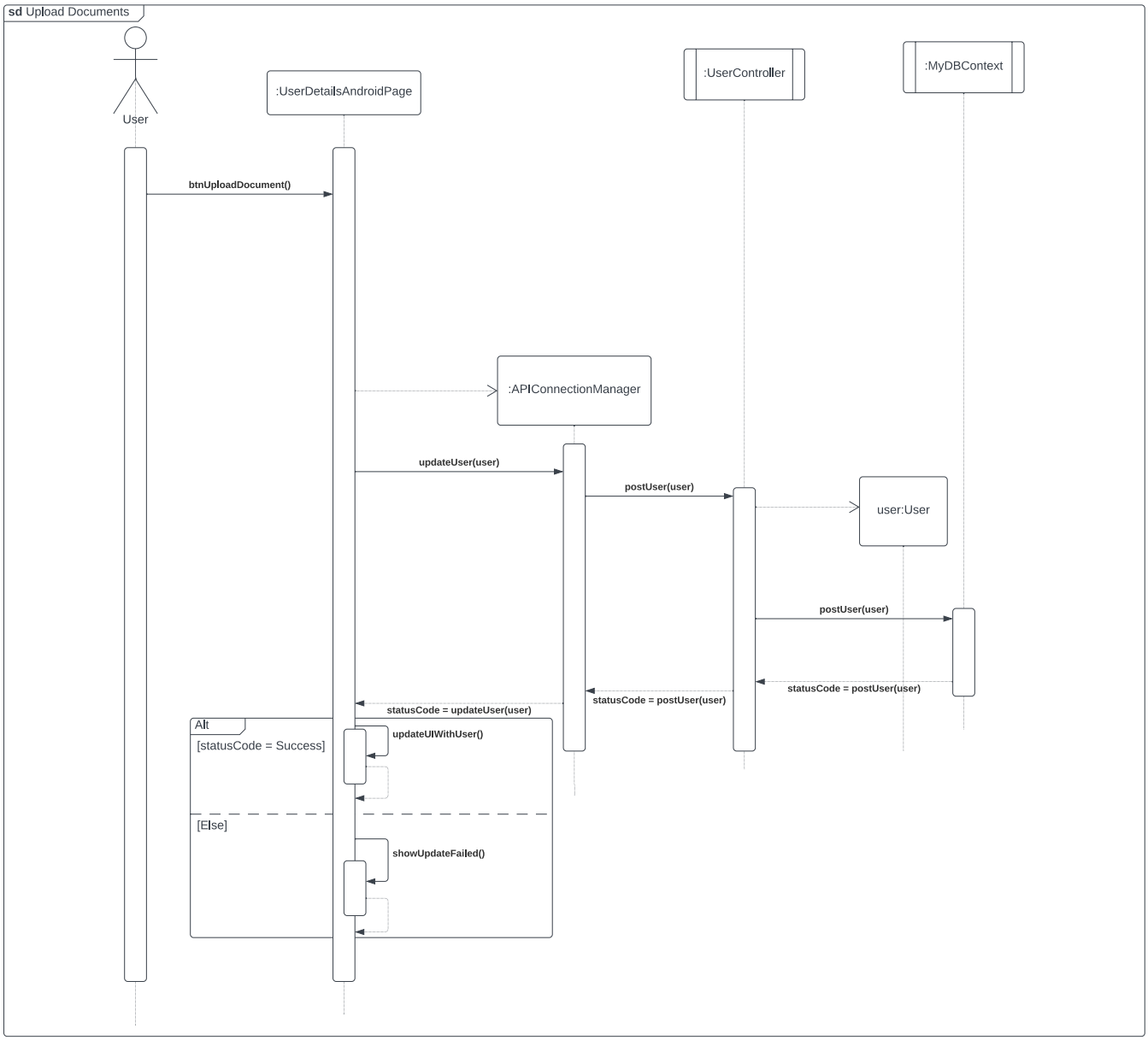
Note: Enter credentials is used for both logging in, and registering..

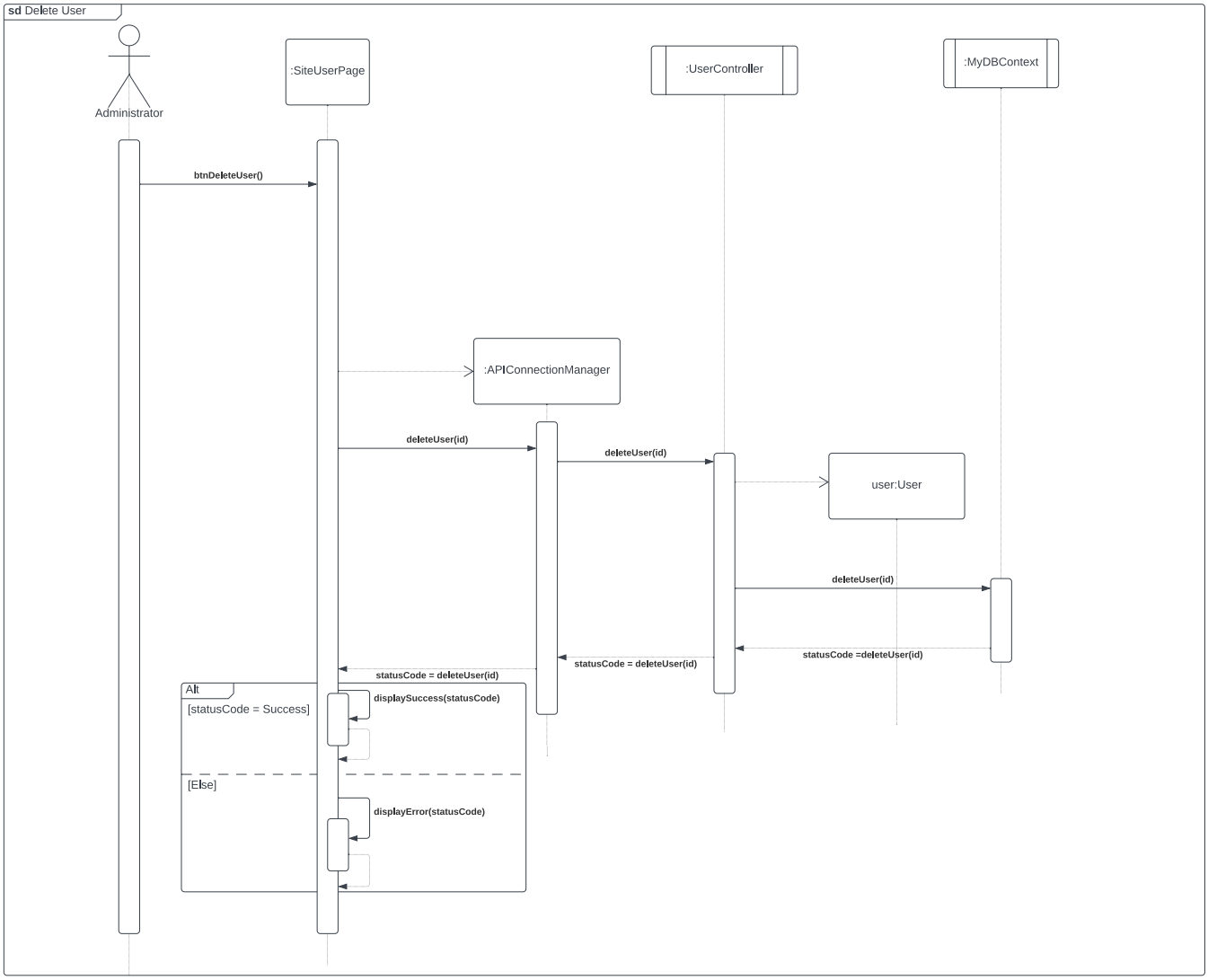

sd Register User
Administrator
:SiteRegistrationLinkPage
:UserControlller
:MyDBContext
btnSend()
validateForm()
statusCode = validateForm()
Alt
[statusCode = Success]
APIConnectionManager
sendLink(email)
postLink(email)
invitation:Registration_Invitations
postLink(invitation)
statusCode = postLink(invitation)
statusCode = postLink(email)
statusCode = sendLink(email)
Alt
[statusCode = success]
displaySuccess(statusCode)
[Else]
displayError(statusCode)
[Else]
displayError(statusCode)


sd Enter Credentials
User
:Login/Register Android or/and Site Page
:UserController
:MyDBContext
btnSubmit()
validateForm()
statusCode = validateForm()
Alt
[statusCode = Success]
APIConnectionManager
submitUser(userDetails)
postUser(user)
user:User
getUser(user)
statusCode = getUser(user)
alt
statusCode != Success
postUser(user)
statusCode = postUser(user)
statusCode = subitUser(userDetails)
Alt
[statusCode = Success]
updateUIWithUser()
[Else]
showFailed()
[Else]
showFailed()

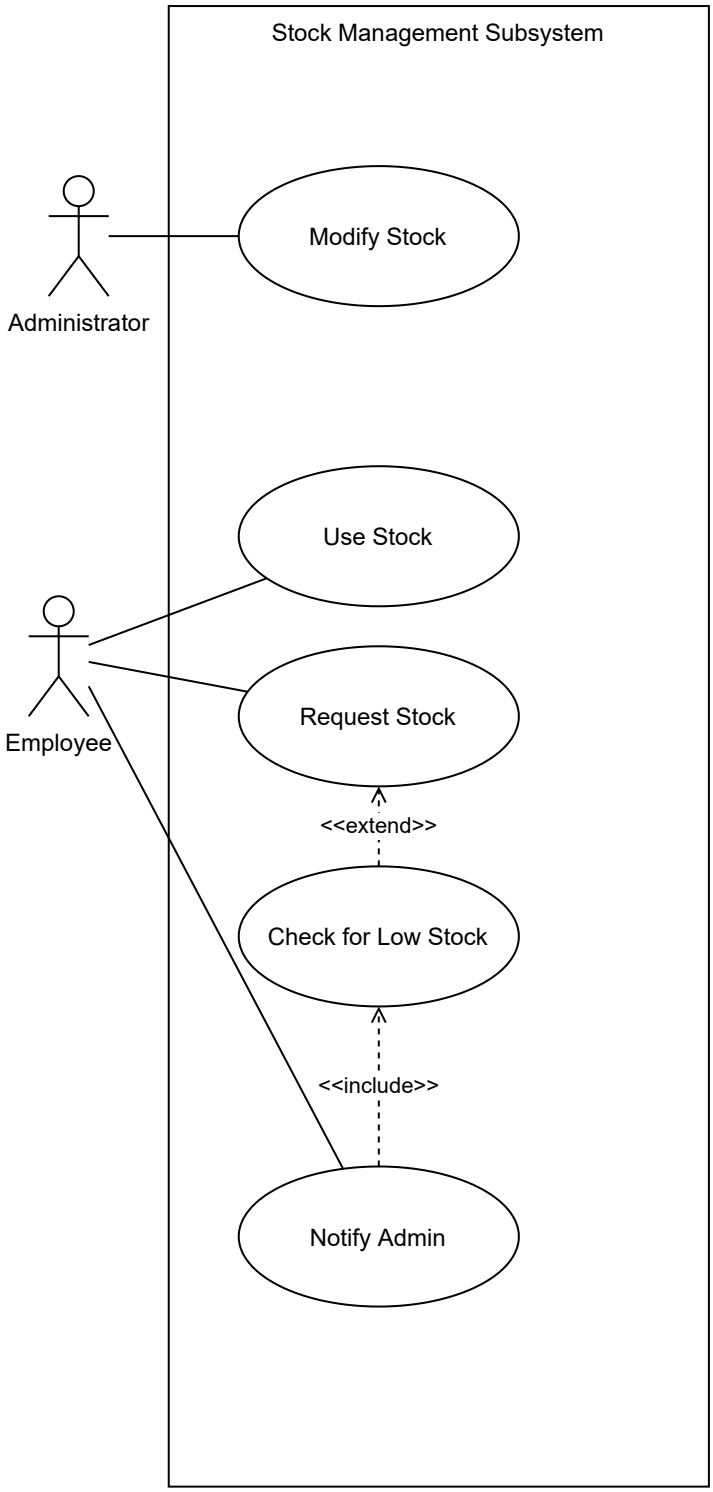

sd Edit User Data
User
:UserDetailsAndroidPage
:UserController
:MyDBContext
btnEditUserData()
:APIConnectionManager
updateUser(user)
postUser(user)
user:User
postUser(user)


statusCode = updateUser(user)
statusCode = postUser(user)
statusCode = postUser(user)
Alt
[statusCode = Success]
updateUIWithUser()
[Else]
showUpdateFailed()

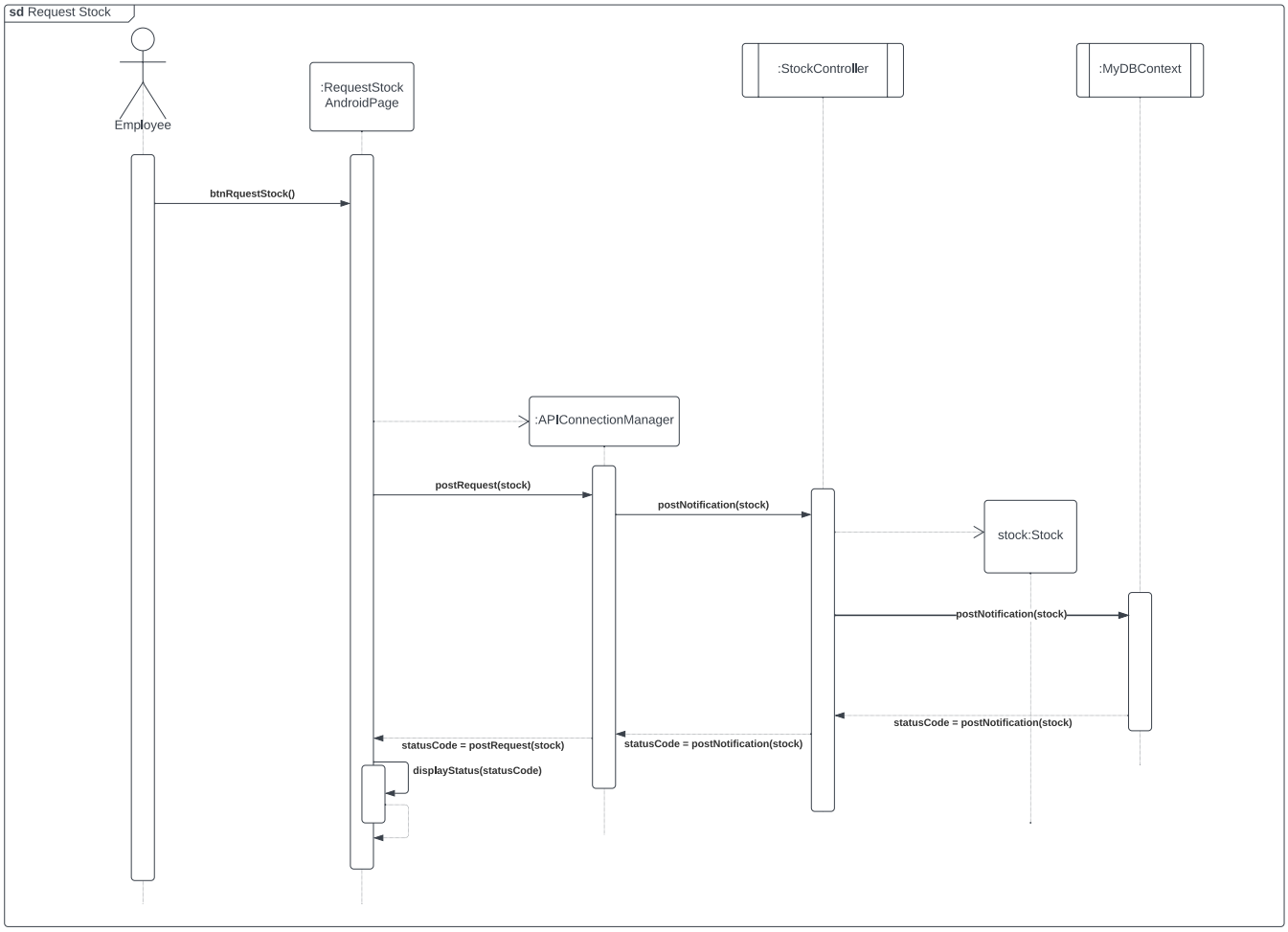

sd Upload Documents
User
:UserDetailsAndroidPage
:UserController
:MyDBContext
btnUploadDocument()
:APIConnectionManager
updateUser(user)
postUser(user)
user:User
postUser(user)
statusCode = postUser(user)
statusCode = updateUser(user)
Alt
[statusCode = Success]
updateUIWithUser()
[Else]
showUpdateFailed()

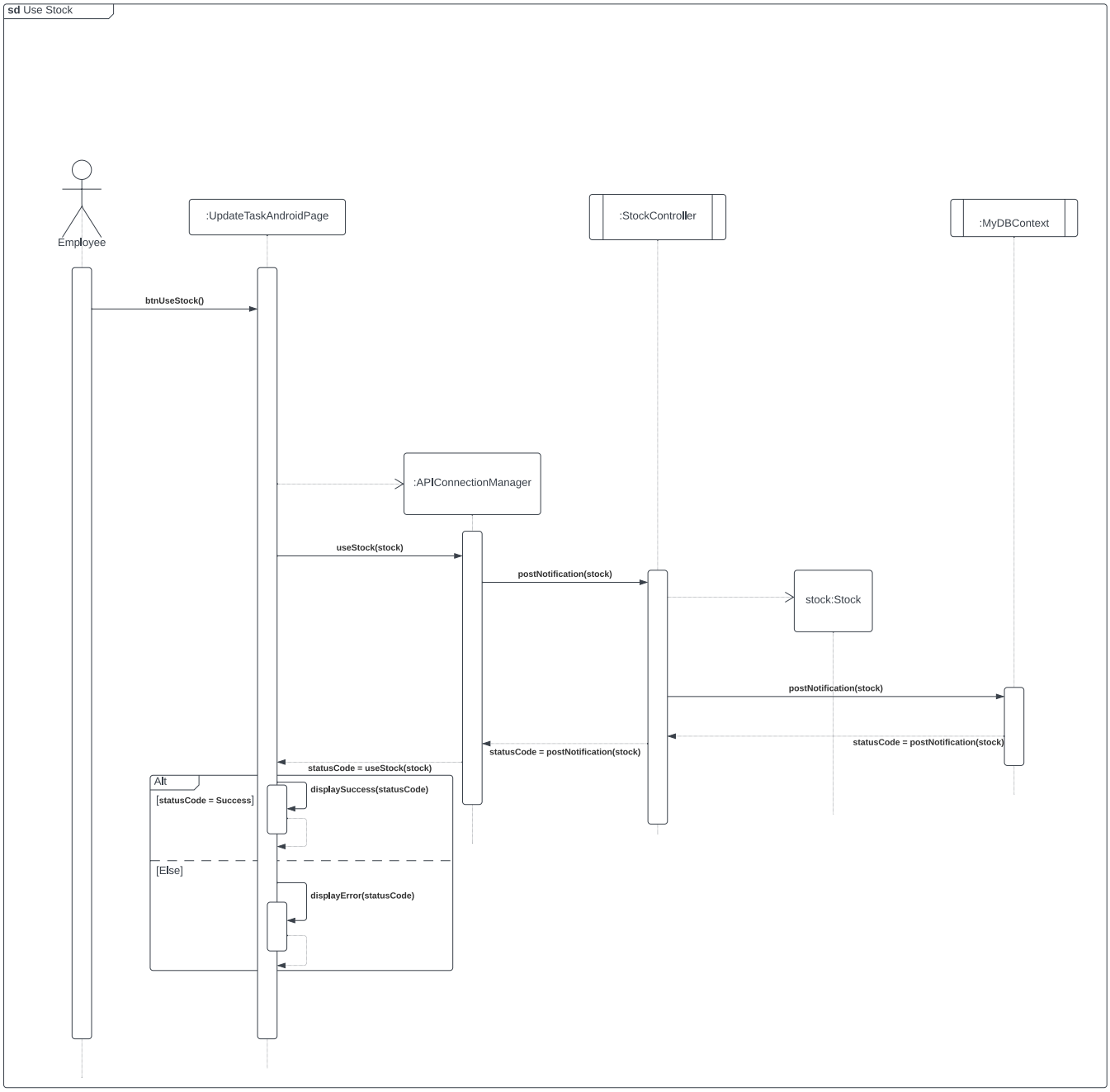


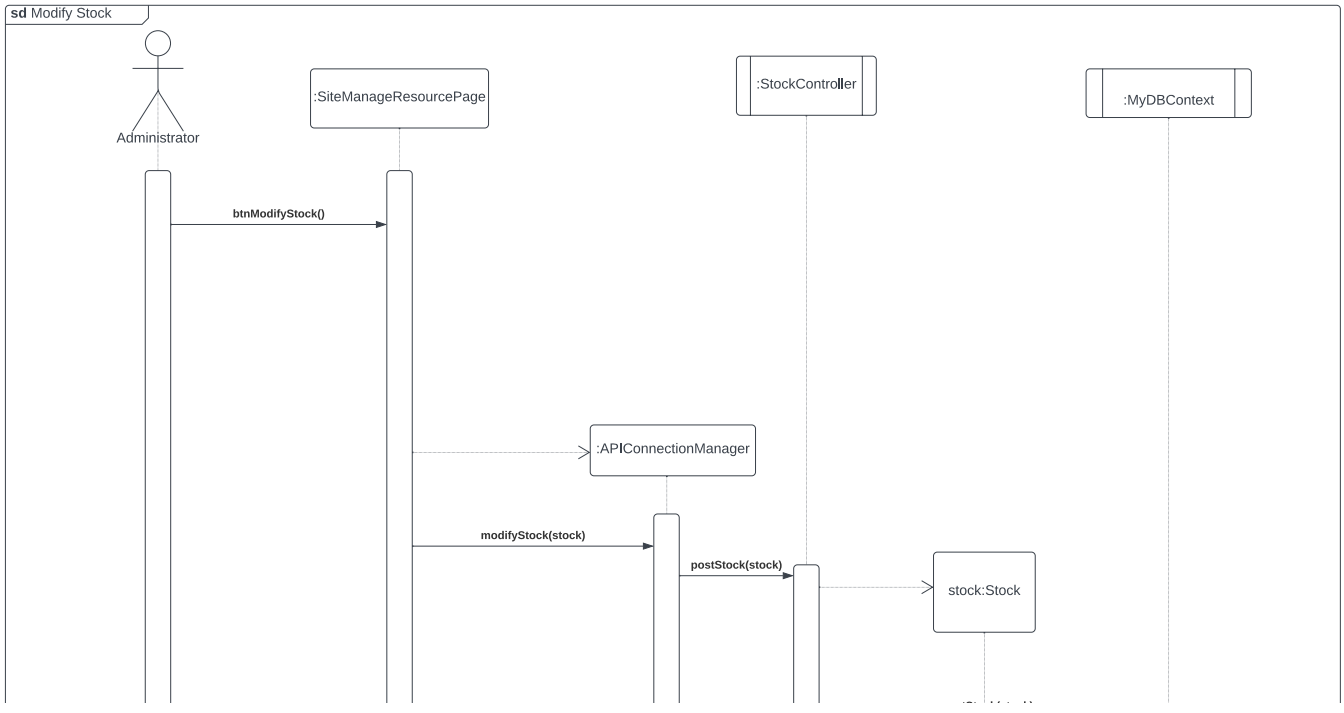
Stock Management

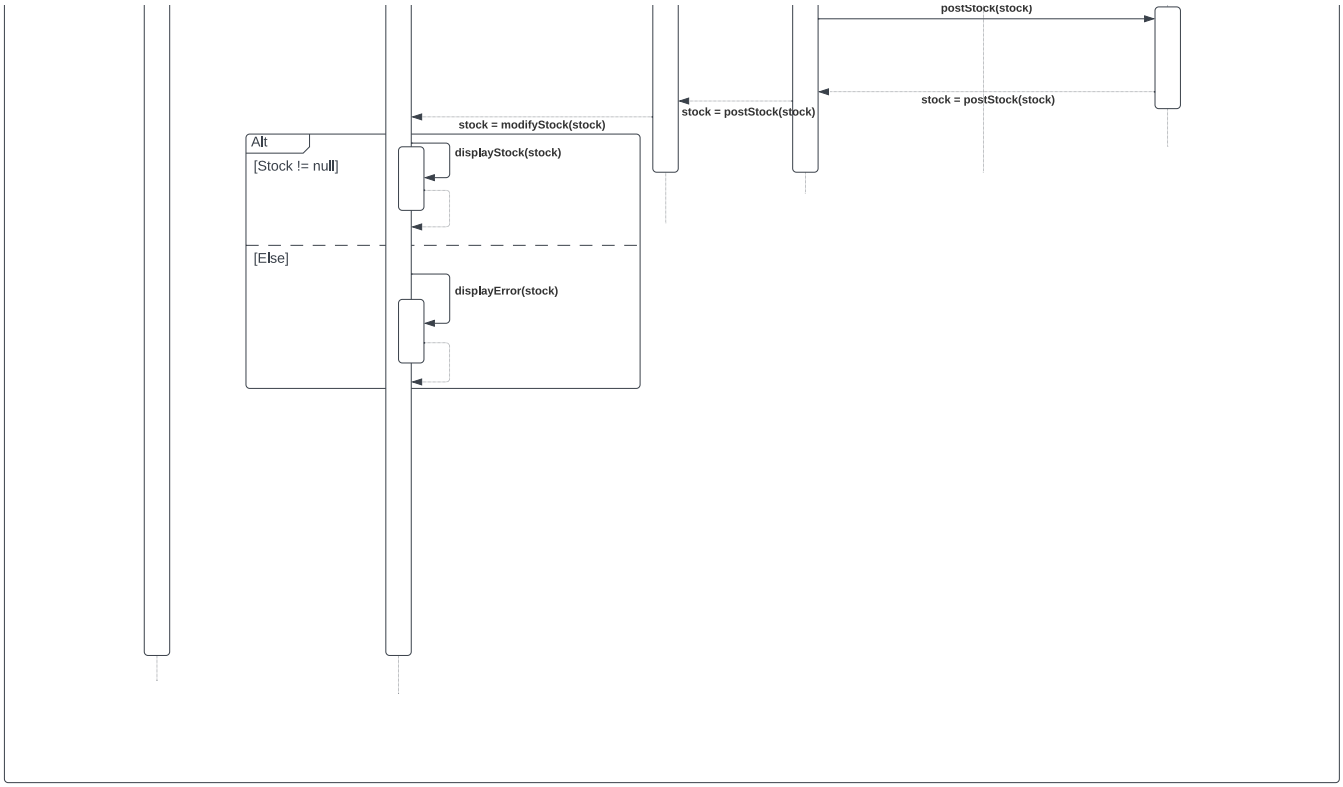


Note: Stock referes to both consumables, and equipment.

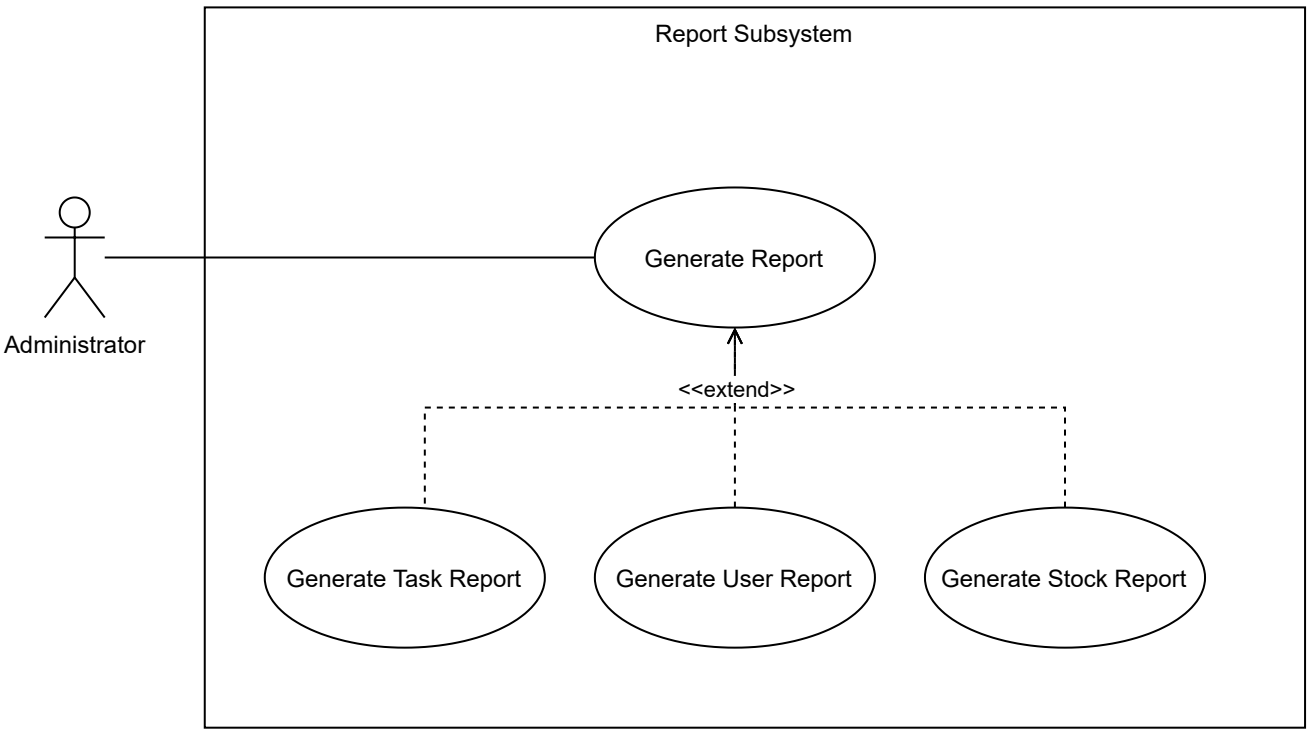

sd Request Stock
Employee
:RequestStock AndroidPage
:StockController
:MyDBContext
btnRquestStock()
:APIConnectionManager
postRequest(stock)
postNotification(stock)
stock:Stock
postNotification(stock)
statusCode = postNotification(stock)
statusCode = postRequest(stock)
statusCode = postNotification(stock)
displayStatus(statusCode)


sd Use Stock
Employee
:UpdateTaskAndroidPage
:StockController
:MyDBContext
btnUseStock()
:APIConnectionManager
useStock(stock)
postNotification(stock)
stock:Stock
postNotification(stock)
statusCode = postNotification(stock)
statusCode = postNotification(stock)
statusCode = useStock(stock)
Alt
[statusCode = Success]
displaySuccess(statusCode)
[Else]
displayError(statusCode)

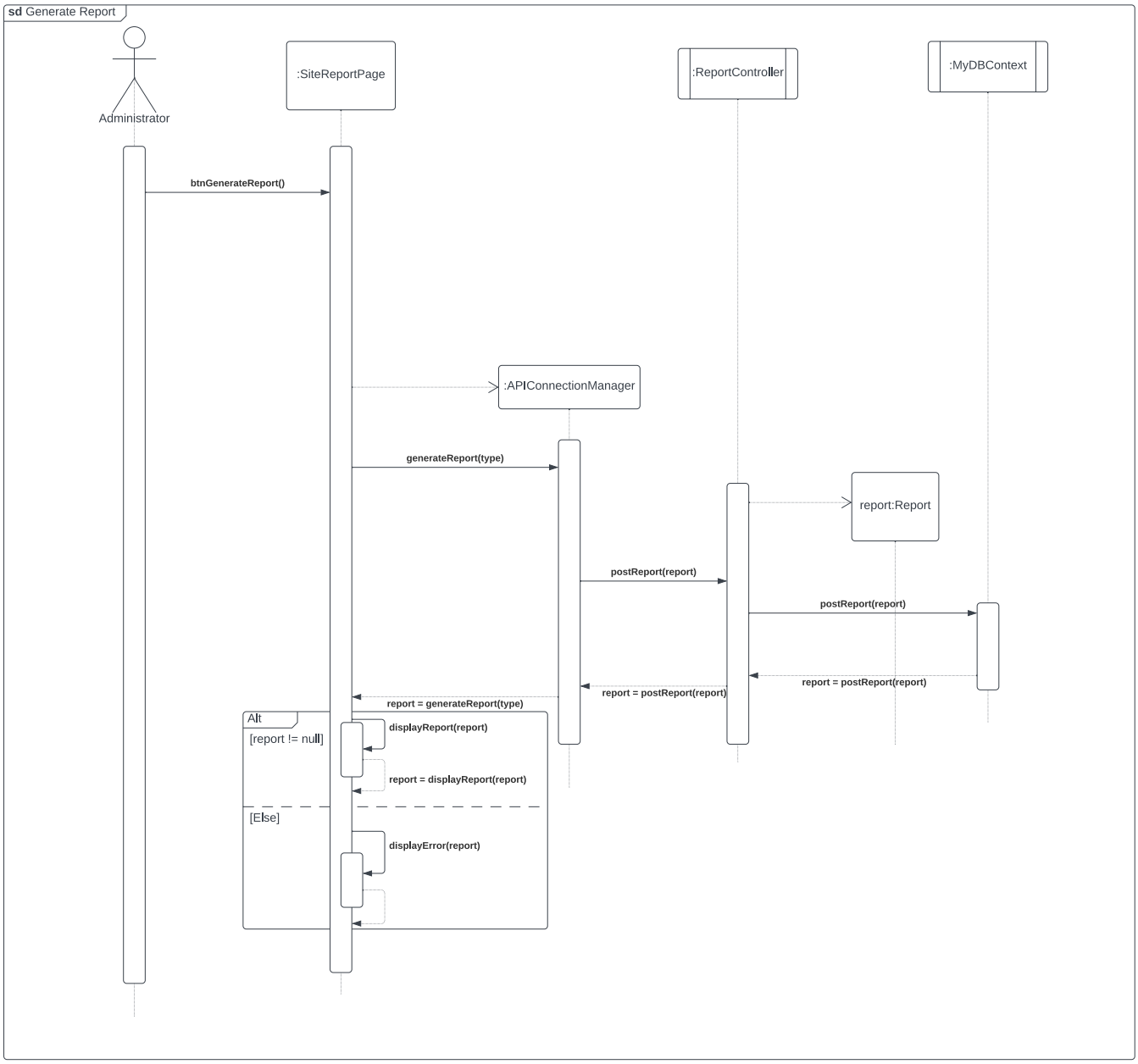

sd Modify Stock
Administrator
:SiteManageResourcePage
:StockController
:MyDBContext
btnModifyStock()
:APIConnectionManager
modifyStock(stock)
postStock(stock)
stock:Stock



Report Management



Note: Users can be Employees, Contractors or Supervisors.



Footnotes

1. https://www.erp.ngo/updates/2021/12/21/erp-melorane-game-reserve
2. https://www.erp.ngo/
3. https://www.groupelephant.com/