HAROKOPIO UNIVERSITY
DEPARTMENT OF INFORMATION & TELEMATICS

Work 2022:

Distributed Systems

Group 7:

it22033, Christos Kazakos

it22045, Konstantinos Katsaras

it22064, Manousos Linardakis

Content:

- 1st Deliverable: Design
  - Admissions
  - Architecture Diagram
  - Class Diagram
  - ER Diagram
  - Use Case Diagram
  - Activity Diagrams
  - State Diagrams
  - Sequence Diagrams

- 2nd Deliverable: Implementation
  - Component/Deployment Diagram
  - Repository Link(old)
  - REST API User Manual(backend token auth)

- 3rd Deliverable: Design:
  - Safety
  - Front-Back communication
  - Access
  - Screenshots of Frontend

- 4th Deliverable: Implementation:
  - Repository links(new)
  - User Manual(backend basic & frontend)

1st Deliverable: Design

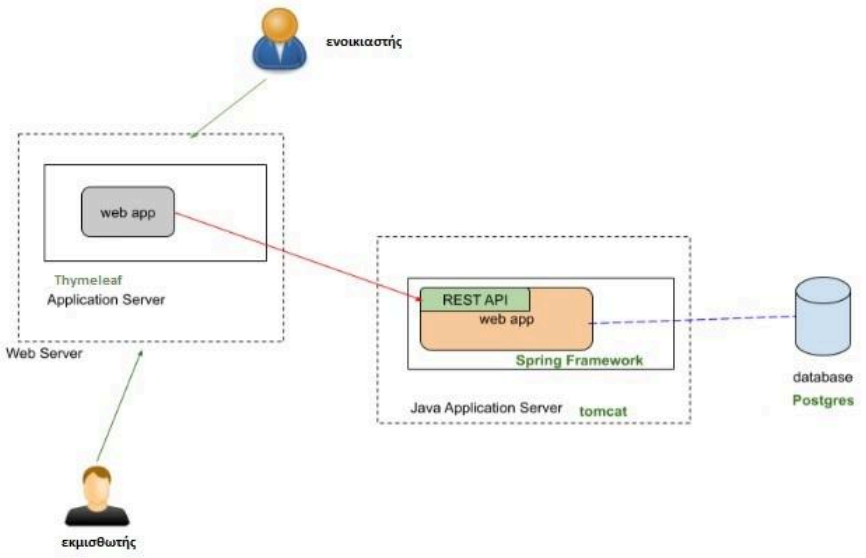
Assumptions:

- The information about the property has been provided either by anothersite, or from related advertisements. The application we will implement only aims to issue contracts for the rental of the selected property.
- The tenant contacts the lessor by phone to express interest in the property (after first registering in the system). After telling him the username, the lessor takes the necessary data (of the tenant) from the database to complete the lease.

- The tenant reads the details of the lease and then agrees or sends comments. The contract is finalized only if the tenant accepts the details of the lease. This leads to the deletion of the lease (from the database) and the creation of the corresponding contract.
- Lease details can be checked/completed by a tenant/landlord (respectively).

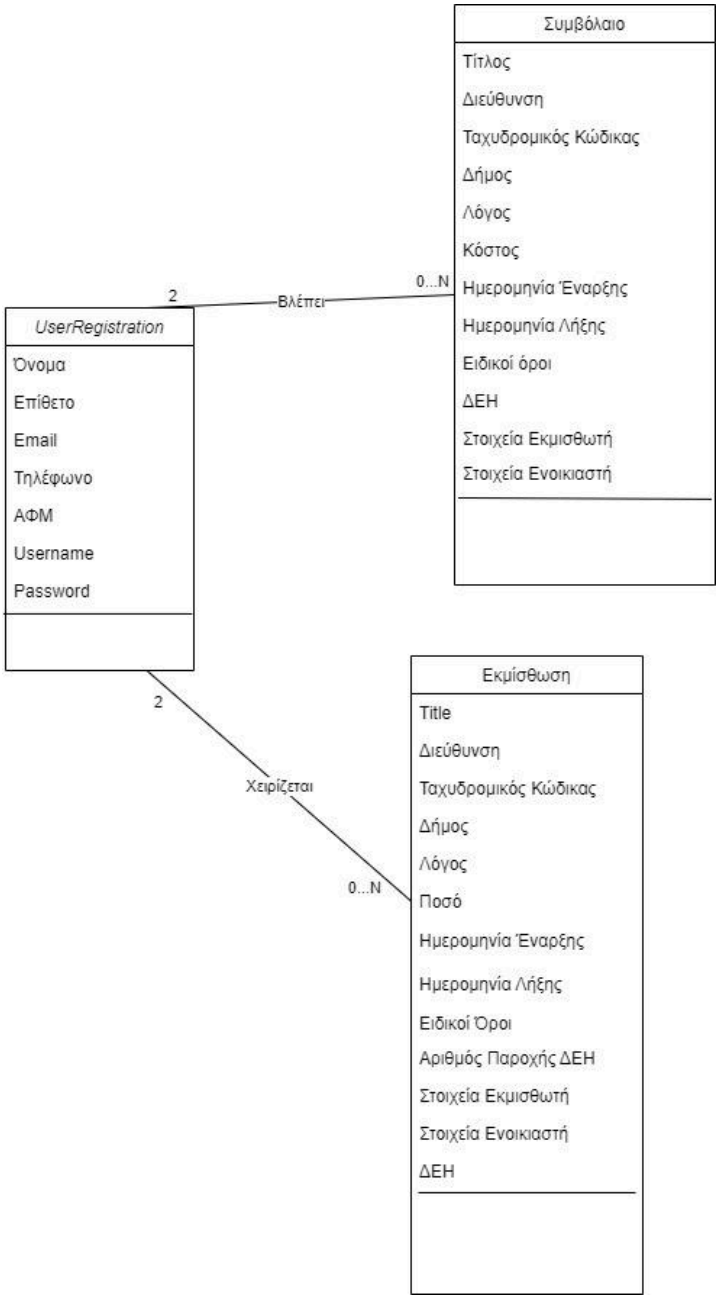
- To complete – create the lease, the details of the respective tenant, as well as the lessor, need to be recorded. This presupposes their "existence" in the base before the creation of the lease.
- Functions for each user are available after login/signup at the beginning of the application (authentication is done with tokens).
- Each lease has only one landlord and tenant.
- Each user can haveonly a role.

Figure with the overall architecture of all parts of the system:

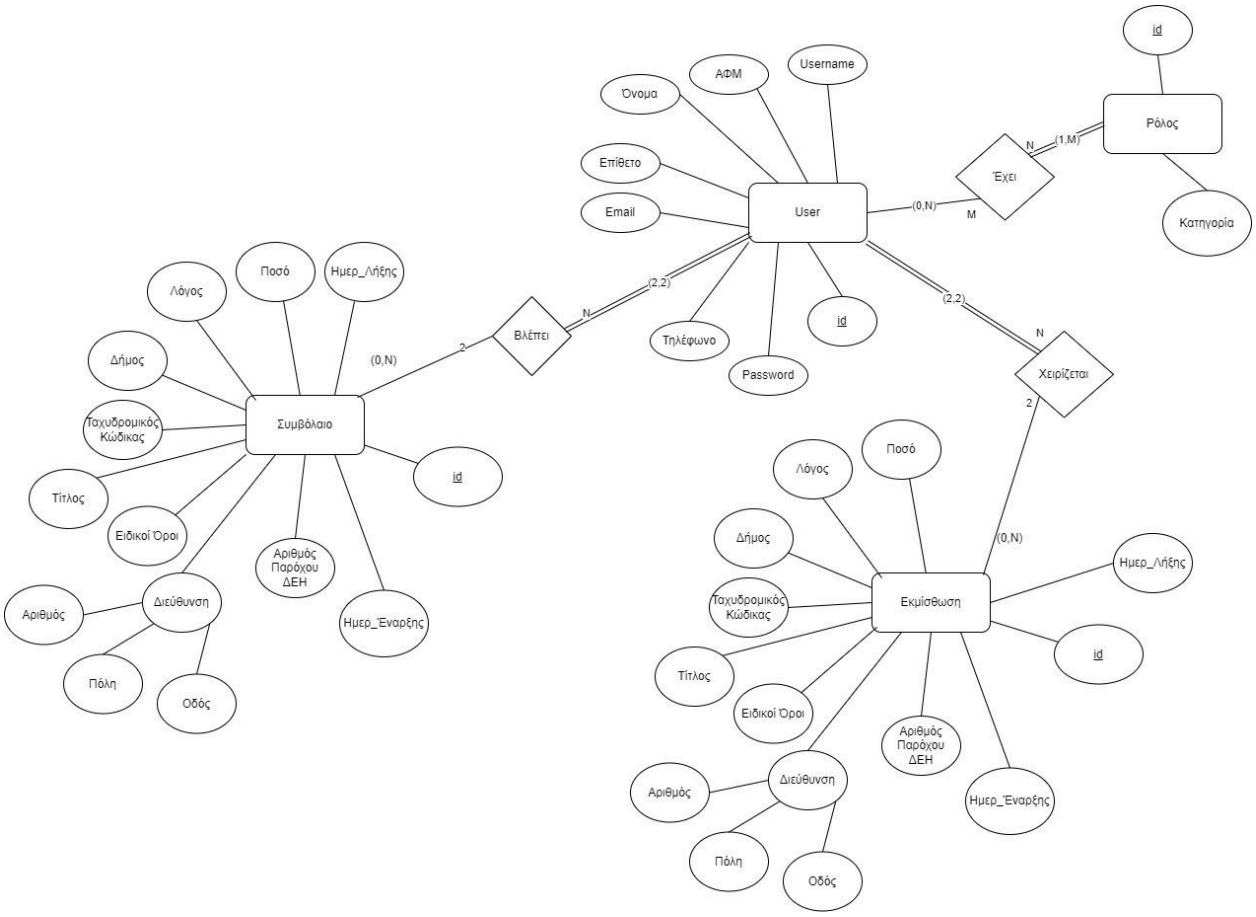


The basic structure for implementing and managing a user directory, using standardized methodologies, is as follows:

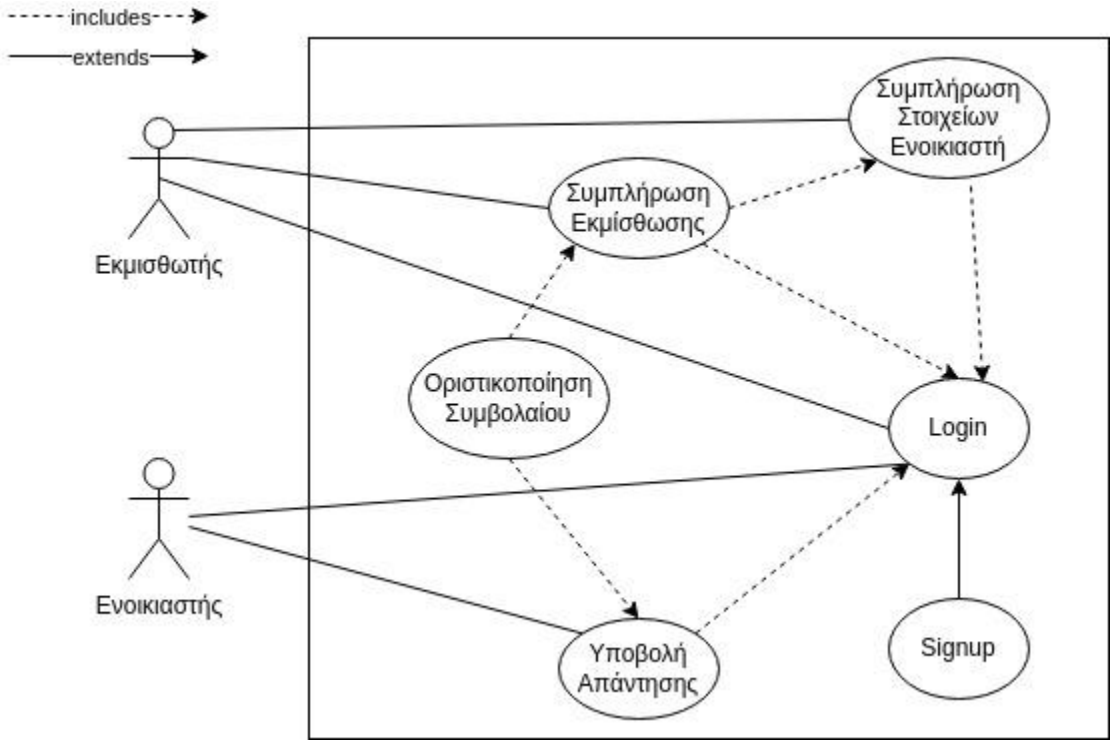
Class Diagram:



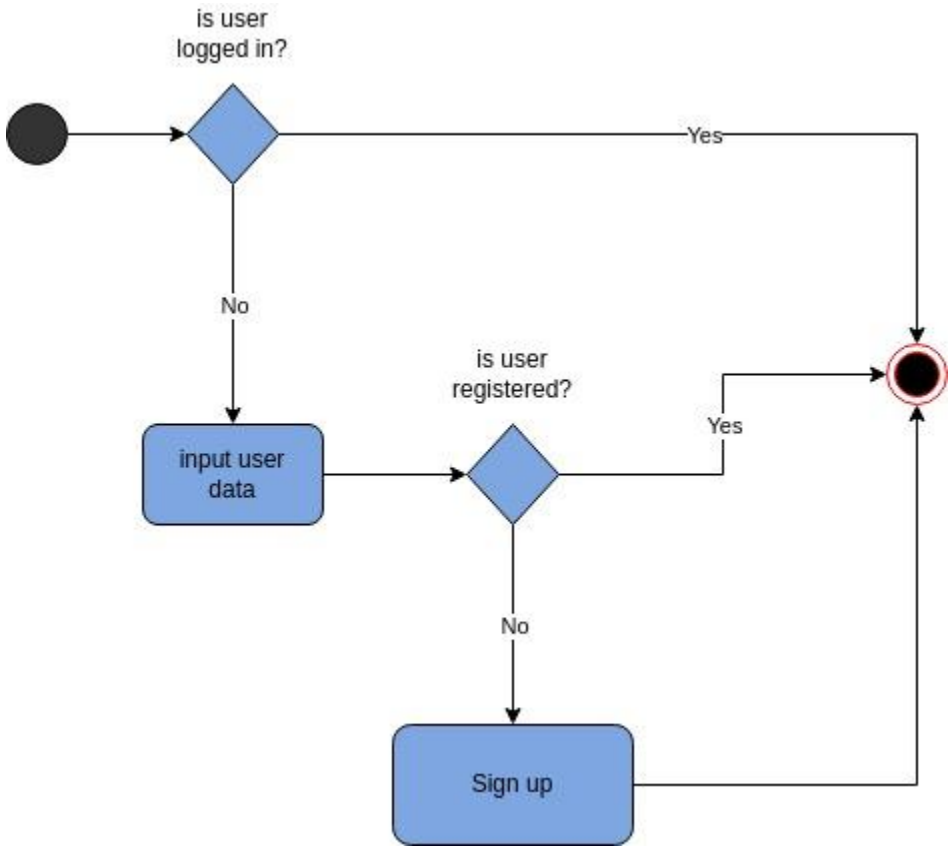
ER Diagram:


id
Ρόλος
Κατηγορία
Έχει
N
(1,M)
M
(0,N)
User
Username
ΑΦΜ
Όνομα
Επίθετο
Email
Τηλέφωνο
Password
id
Βλέπει
(2,2)
N
2
(0,N)
Συμβόλαιο
Λόγος
Ποσό
Ημερ_Λήξης
Δήμος
Ταχυδρομικός Κώδικας
Τίτλος
Ειδικοί Όροι
Αριθμός Παρόχου ΔΕΗ
Ημερ_Έναρξης
id
Διεύθυνση
Αριθμός
Πόλη
Οδός
Χειρίζεται
(2,2)
N
2
(0,N)
Εκμίσθωση
Λόγος
Ποσό
Δήμος
Ταχυδρομικός Κώδικας
Τίτλος
Ειδικοί Όροι
Αριθμός Παρόχου ΔΕΗ
Ημερ_Έναρξης
Ημερ_Λήξης
id
Διεύθυνση
Αριθμός
Πόλη
Οδός

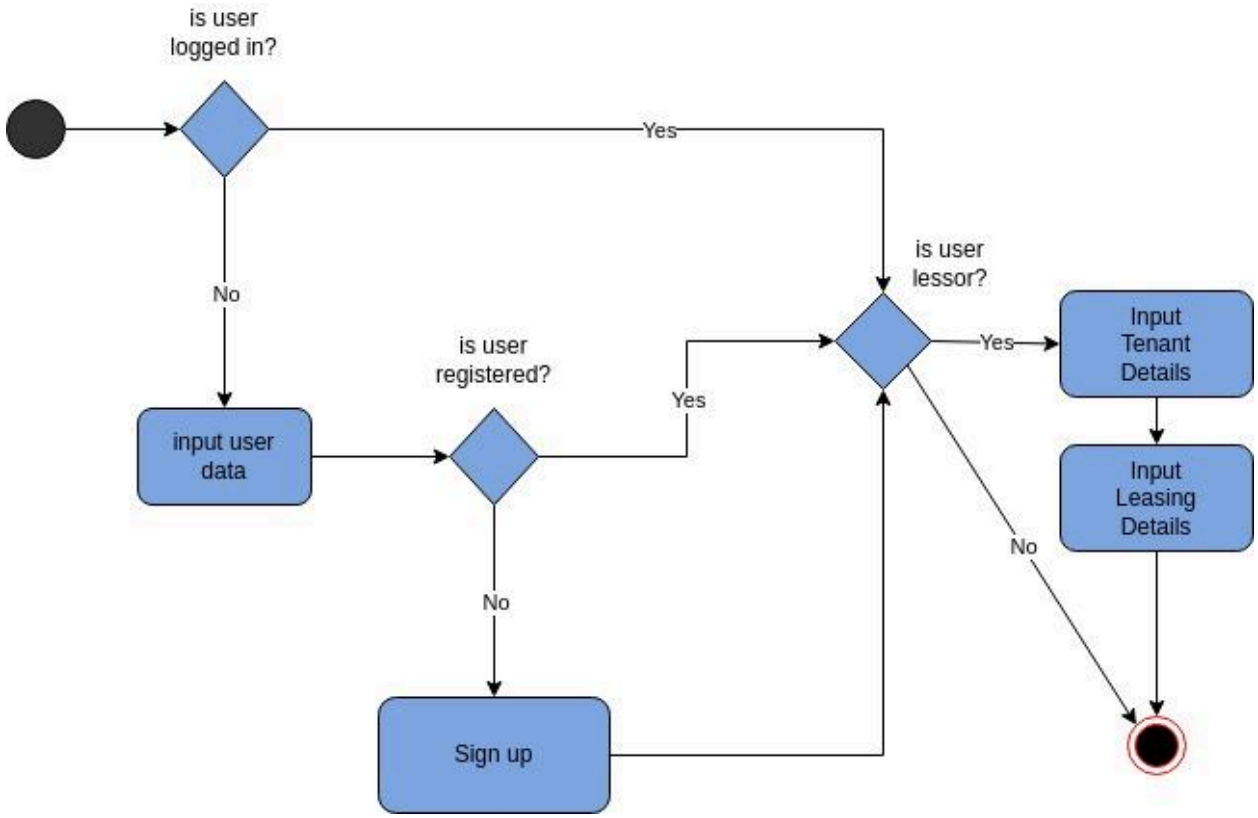
Reporting and commenting on the services provided in each different role is described with the following Use Case Diagram:



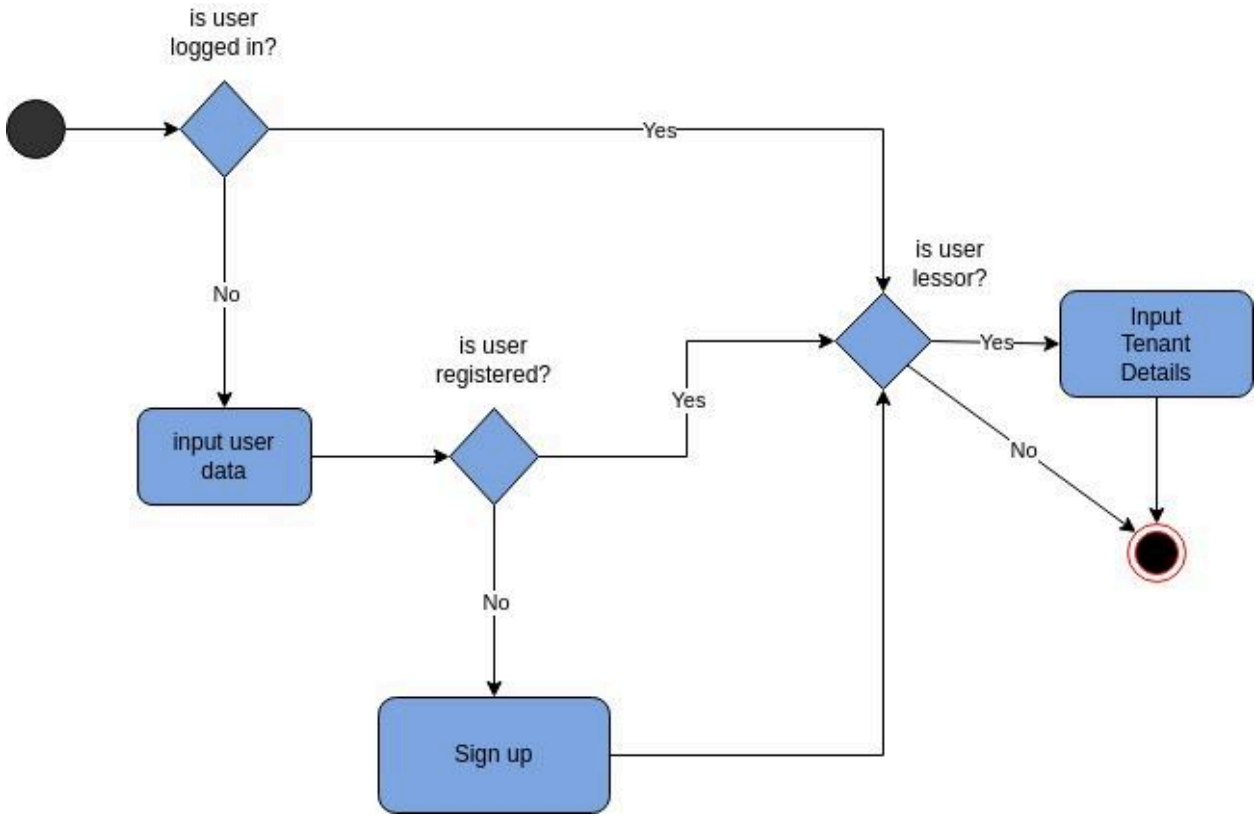
Activity Diagram for the “Login” (and “Sign Up”) Function:



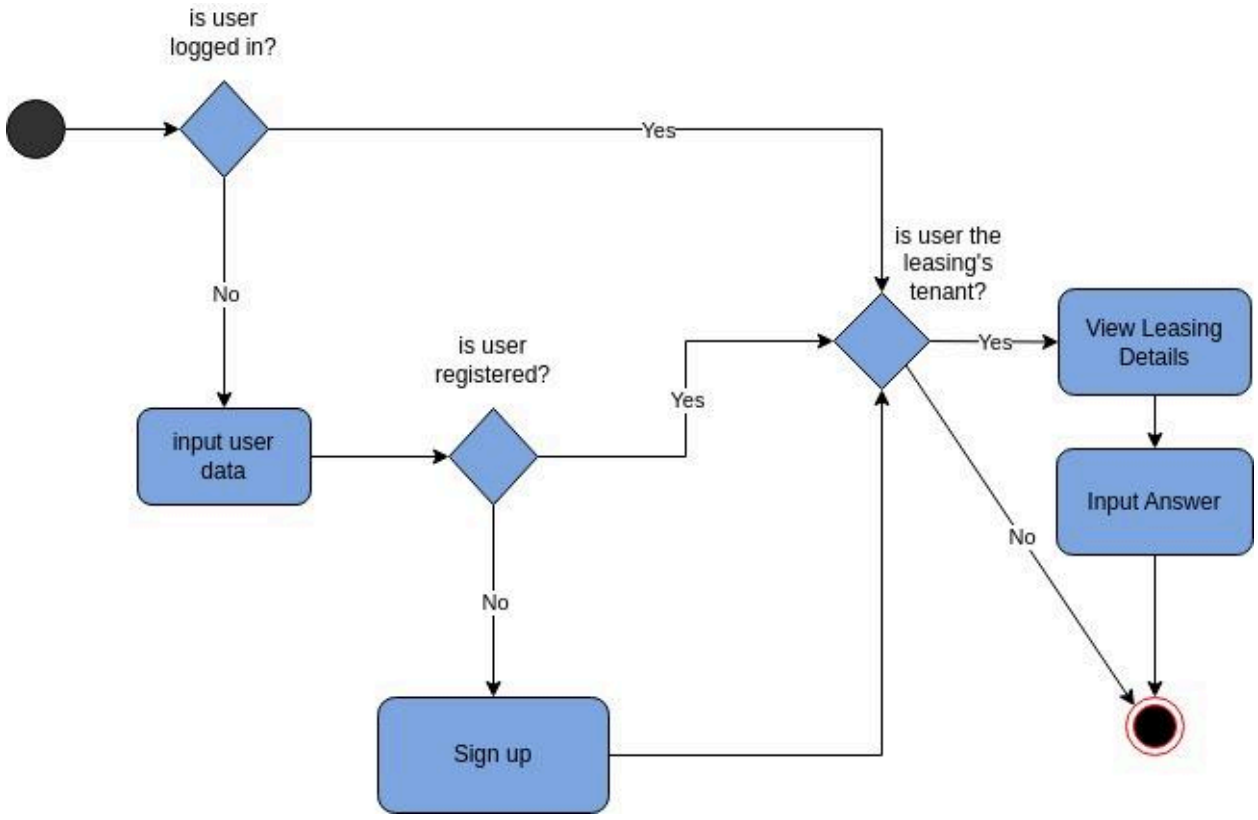
Activity Diagram for the “Fill Lease” Function:



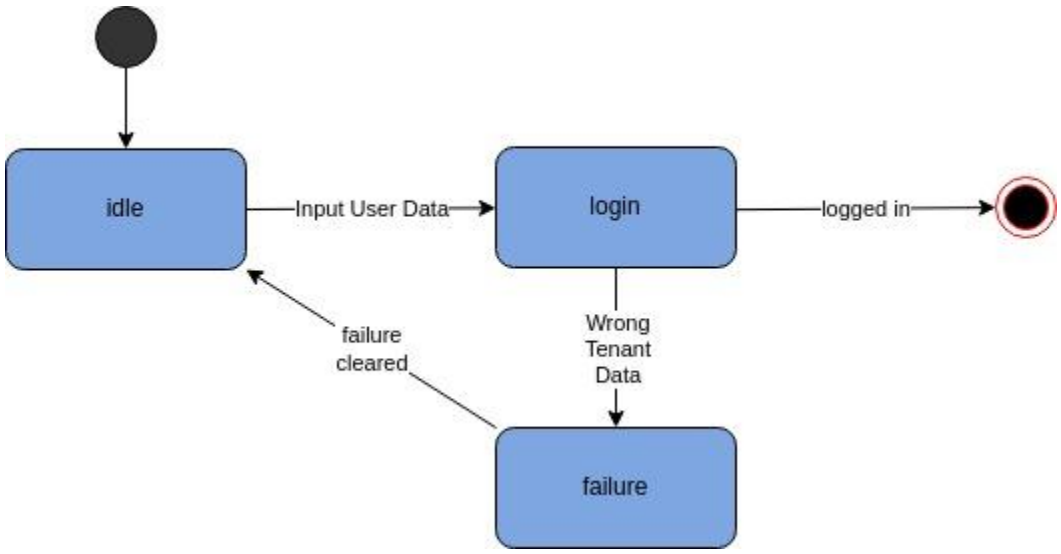
Activity Diagram for the "Filling in Tenant Details" Function:



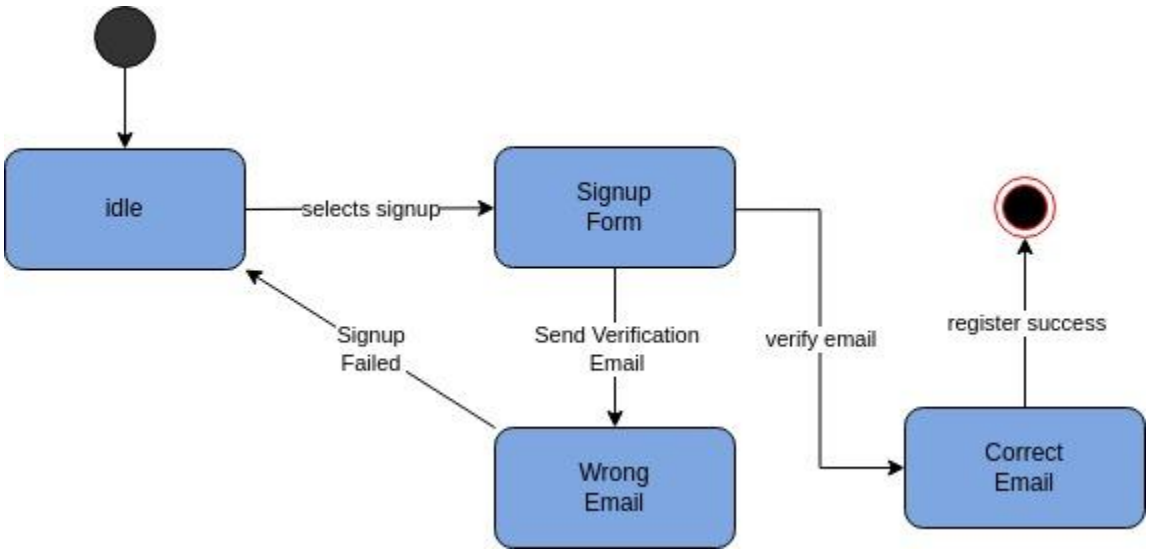
Activity Diagram for the “Submit Answer” Function:



State Diagram for the “Login” Function:

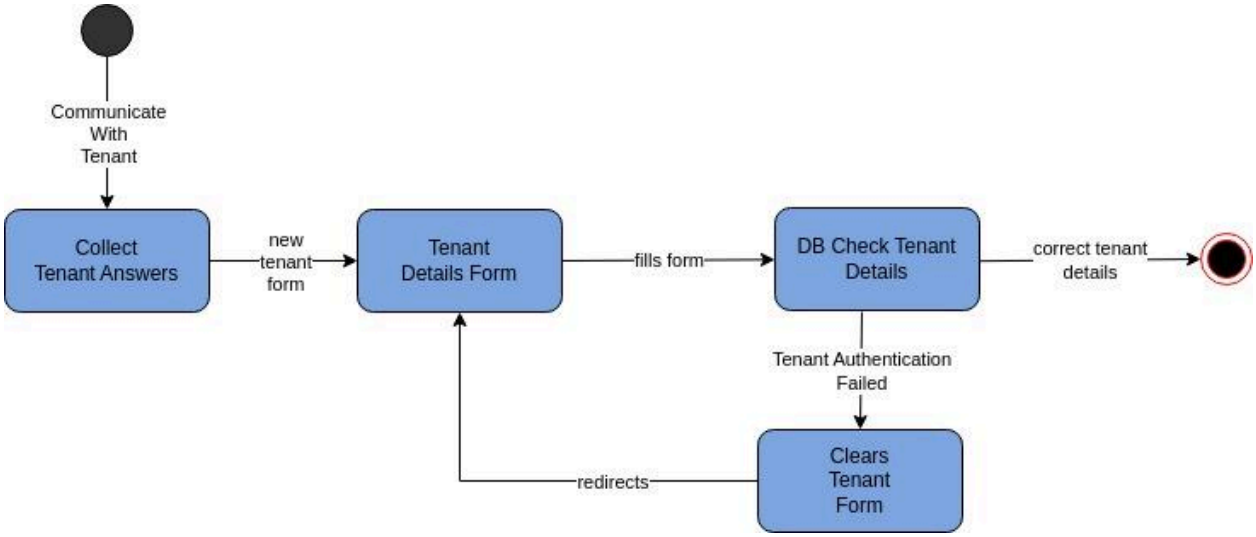


State Diagram for the “Signup” Function:

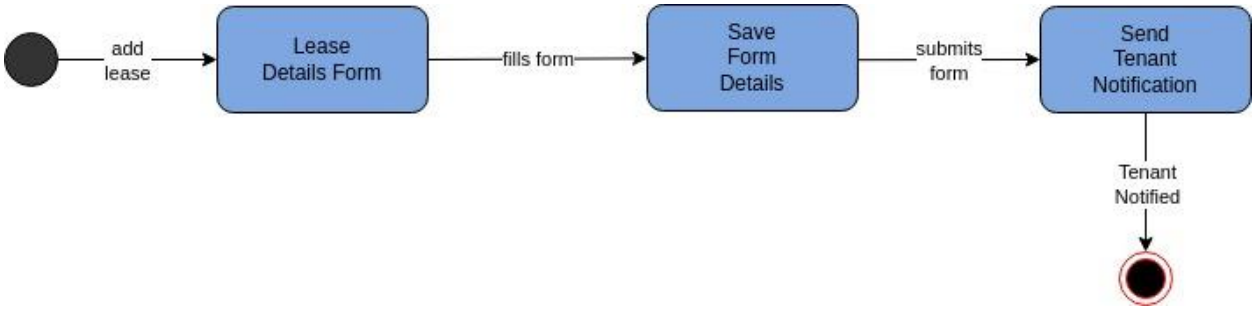


State Diagram for the “Submit Answer” Function:

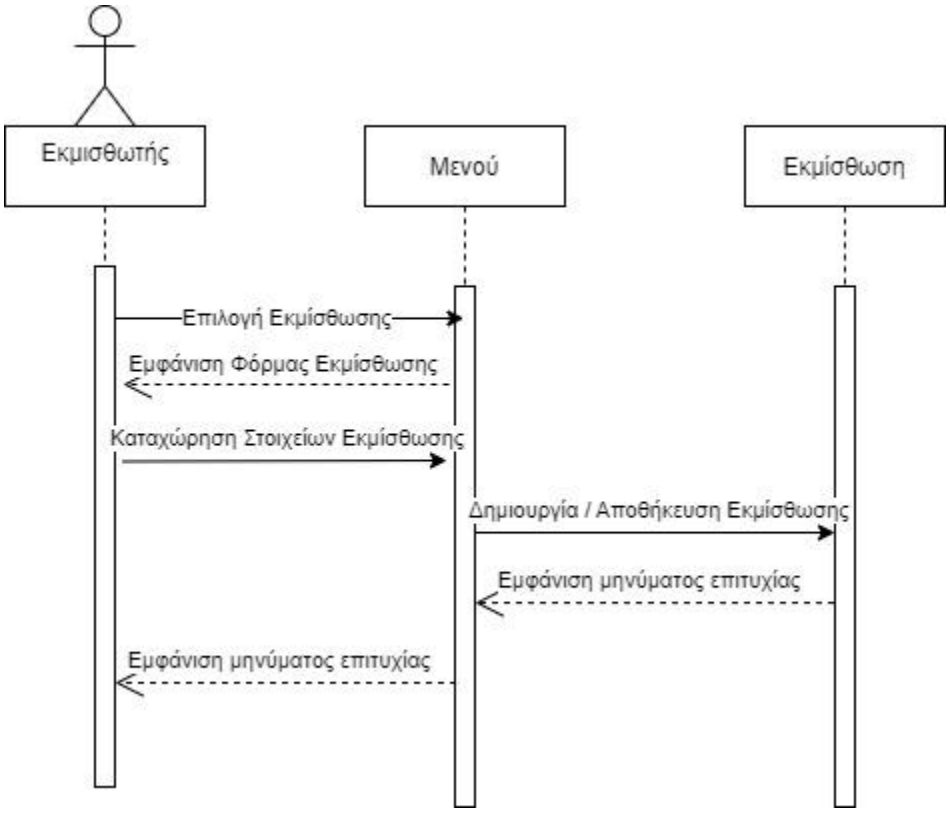
State Diagram for the "Filling in tenant details" Function:



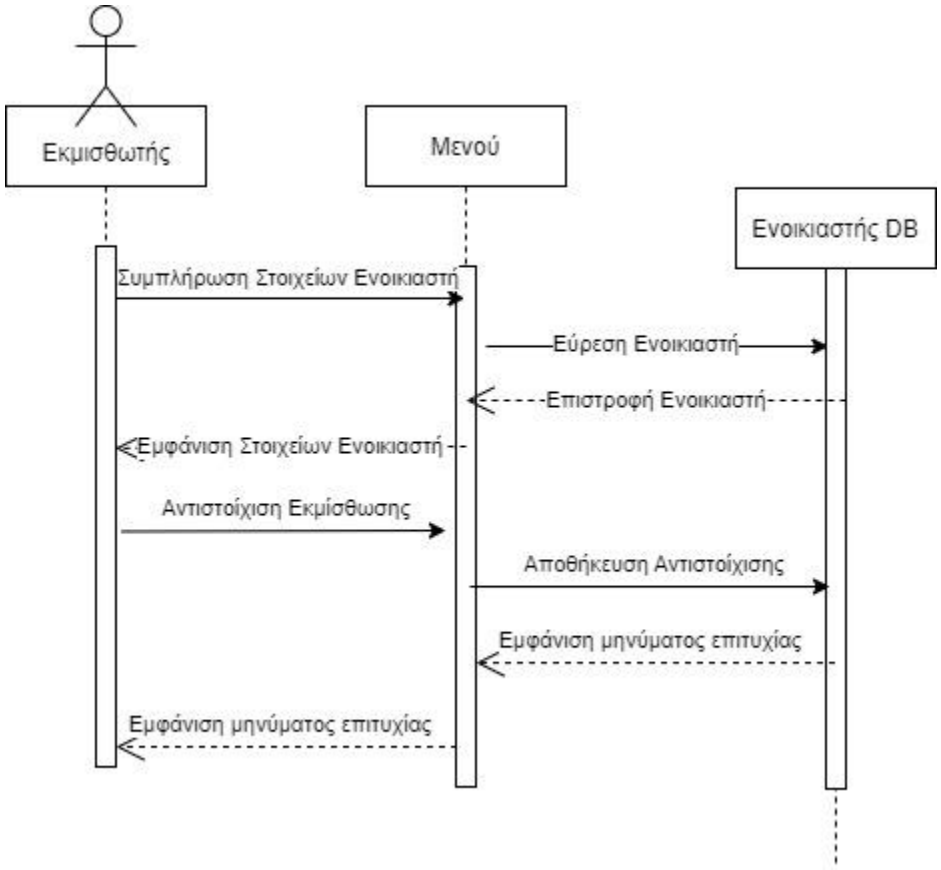
State Diagram for the "Fill Lease" Function:



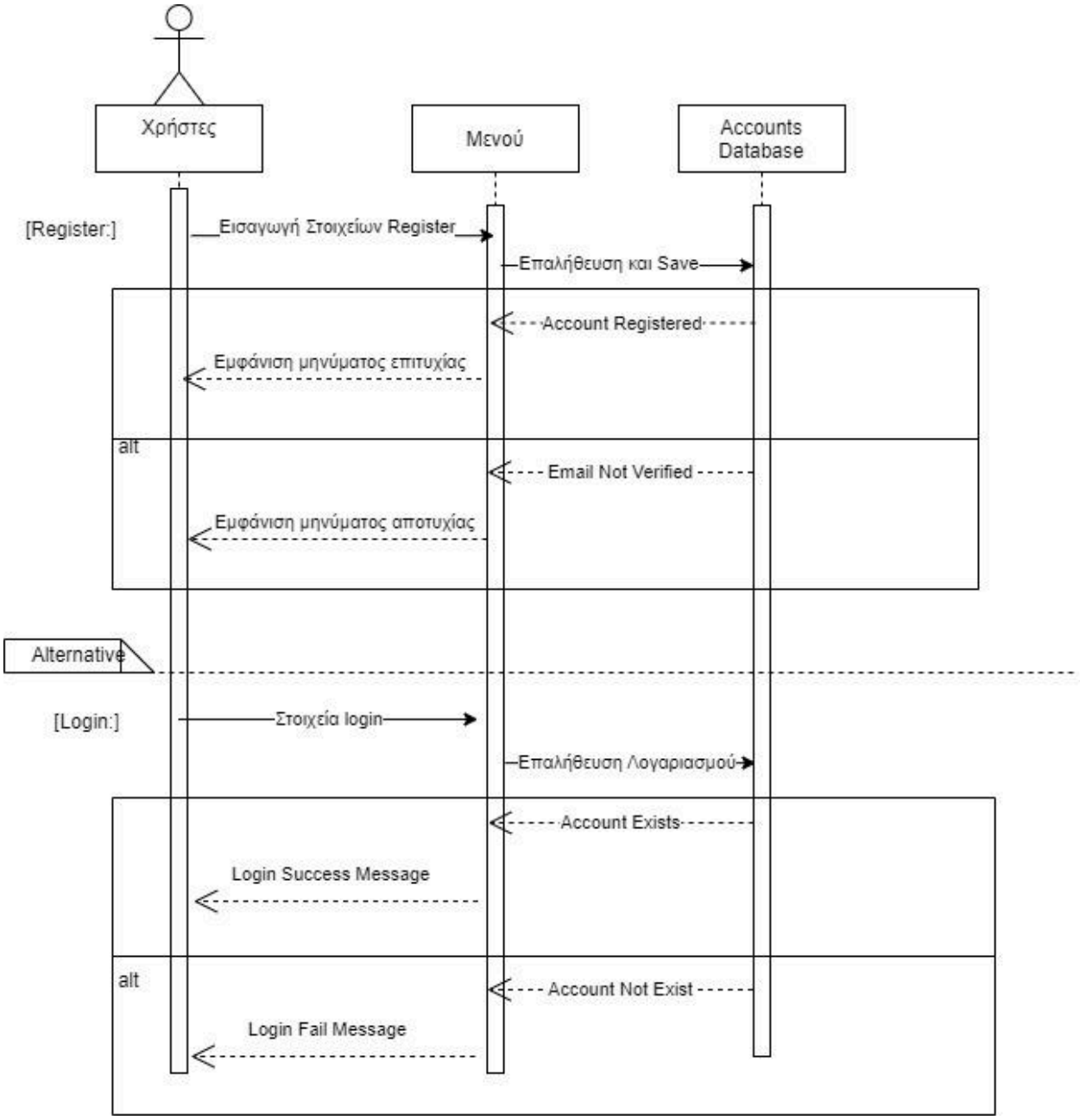
Sequence Diagram for the “Fill Out Lease” Function:



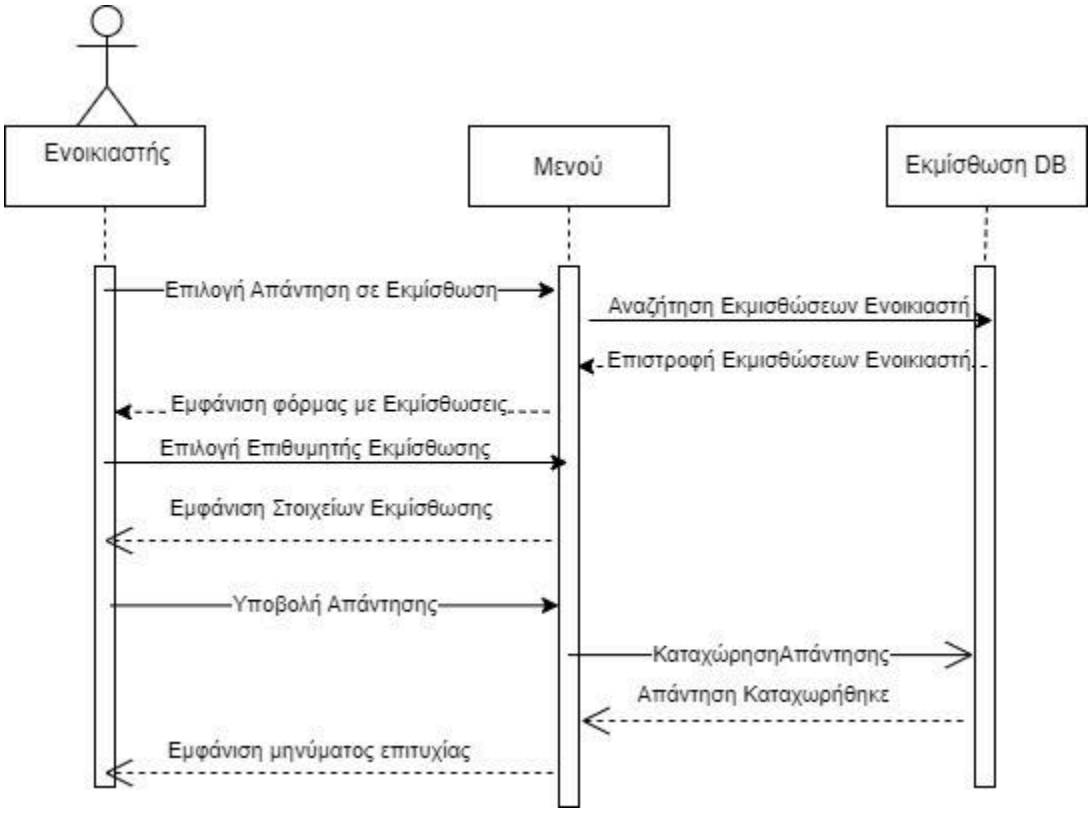
Sequence Diagram for the "Filling in Tenant Details" Function:



Sequence Diagram for the “Login” (and “Sign Up”) Function:

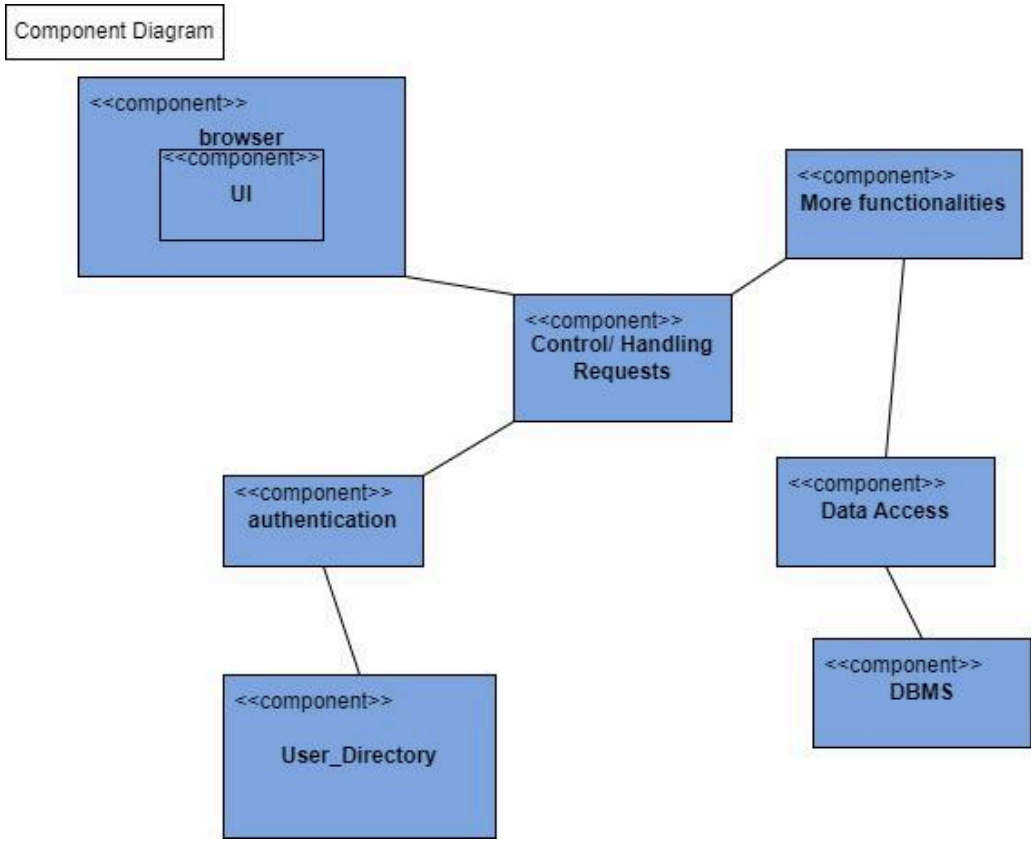


Sequence Diagram for the “Submit Answer” Function:



2nd Deliverable - Implementation:

Here is the Component/Deployment diagram we used to implement the application:



Repository Link:

The link to the github repository for the backend (token authentication) is as follows:

https://github.com/manouslinard/dist_sys_2022/tree/backend-token-auth

The link for the repository (general):

https://github.com/manouslinard/dist_sys_2022

We have also implemented the backend with basic authentication (along with the additional functions we need in thefrontend). The link of the backend repository (with basic authentication) is as follows:

https://github.com/manouslinard/dist_sys_2022/tree/backend-basic

Accordingly, the user manual for the new functions can be found herehere .

REST API Usage Manual:

In order to use the following "links", the request needs to start with "http://localhost:8080 "(eg the method "/lessor/getAllLessors " would be written as follows to make it work: "http://localhost:8080/lessor/getAllLessors" – respectively for the other methods).

It is noteworthy that in the methods we use curly brackets "{ ... }" to indicate that this "piece" is variable and will need to be changed accordingly, in order to have the desired result (see also the method descriptions). Also, in the final request**not**curly brackets are placed but only the value of the variable.

OR**access**of "methods" starting with /lessor/... can be executed by the lessor. Accordingly, the access of the "methods" starting with /tenant/... can be executed by the tenant. Additionally anything starting with /api/auth/... can be executed by everyone in the token and basic authentication (backend). It is also important to note that each user can only see their own data and**No**of other users (unless logged in as an admin, who can additionally execute functions that create leases, delete users or view their details – mainly to help solve user problems if any).

Authentication (only works for the token authentication)	Description	Method	Test Input (JSON)
/api/auth/signup	Users do signup to the system.	POST	Lessor Signup: {"username":"lessor1", "e-mail":" lessor1@gmail.com

			",password:"pass123", role:["lessor"]} Tenant Signup: {"username":"tenant1", "e-mail":" tenant1@gmail.com ",password:"pass123"} Admin Signup: {"username":"admin1", "e-mail":" admin1@gmail.com " , password:"pass123", role:["admin"]} (optional: firstName, lastName, afm and phone attributes — see structure of entire jsonhere)
/api/auth/signin	Users do signin to the system.	POST	{"username":tenant, password:"pass123"}

Lessor (/lessor)	Description	Method	Test Input(JSON)
/lessor/getAllLessors	Returns all the lessors.	GET	
/lessor/{lessorUsername}/leases	It brings all the lessor's leases with it given lessorUsername.	GET	
/lessor/{lessorUsername}/leases/{lid}	Brings a lease with id=lid of the lessor with the given lessorUsername	GET	

/lessor/{lessorUsername}/leases/{lid}	Deletes the lease with id=lid of the lessor with the given lessorUsername	DELETE	
/lessor/{lessorUsername}/leases/{lid}	Updates the lease with id=lid of the lessor with the given lessorUsername.	PUT	{"title":"lease2","startDate":"2023-01-01", "endDate":"2023-01-13"} (only mandatory attr are lease's title, start and end Date – the others are optional – see other attr in test JSONcreateLease)
/lessor/{lessorUsername}/{tenantUsername}/createLease	Creates a lease and enters the tenant details (with the given tenantUsername) as well as details of the lessor (with the given lessorUsername) within the new lease.	POST	{"title": "test", "address": "address", "tk": "12137", "dimos": "Peristeri","reason": "Reason","cost": 15000, "startDate": "2023-01-01", "endDate": "2023-01-13", "sp_con": "Special Conditions", "day": "1234567"}
/lessor/createTenant	It creates one tenant.	POST	{"username":"christos3", "e-mail":"christos3@gmail.com", password:"pass123", "firstName":"Chris", "lastName":"Papas", "afm":"12345678901", phone:"698731311"} (attributes firstName,

			lastName, afm and phone are optional — afm should be numbers only and exact size of 11! Also afm is unique and cannot be "used" again)
/lessor/getAllTenants	It shows all of them tenants.	GET	
/lessor/{lessorUsername}/contracts	It brings everything contracts of the lessor with the given lessorUsername.	GET	
/lessor/{lessorUsername}/contracts/{cid}	He brings a contract with id=cid of the lessor with the given lessorUsername.	GET	
/lessor/{lessorUsername}	Returns them elements of the lessor with the input lessorUsername.	GET	
/lessor/{lessorUsername}	Deletes the lessor with the input lessorUsername.	DELETE	

Tenant	Description	Method	Test Input (JSON)
/tenant/{tenantUsername}/leases	It brings all tenant leases with it given tenantUsername.	GET	

/tenant/{tenantUsername}/leases/{lid}	Brings a lease with id=lid of the given tenant tenantUsername.	GET	
/tenant/{tenantUsername}/leases/{lid}/answer	He saves her tenant response with the given tenantUsername to a lease of, which has id=lid. If tenantAnswer: hasAgreeed == true, is finalized contract.	POST	{"hasAgreeed":**false**, "tenantComment":"Needs fixing"} OR: {"hasAgreeed":**true**} (comment is optional)
/tenant/getAllLessors	Returns all the lessors who it is registered in system.	GET	
/tenant/{tenantUsername}/contracts	It brings everything tenant contracts with the given tenantUsername	GET	
/tenant/{tenantUsername}/contracts/{cid}	He brings a contract with id=cid of the given tenant tenantUsername.	GET	
/tenant/{tenantUsername}	Deletes the tenant with the input tenantUsername.	DELETE	

/tenant/{tenantUsername}	Returns them elements of the tenant with the input tenantUsername.	GET	

3rd Deliverable: Design

Safety:

In the backend, we set the security with Spring Boot Security in the fileSecurityConfig . In addition, we use the basic http request for authentication from the frontend to the backend, for this the user needs to login (or register) to enter the application. About basic auth in frontend, if user enters wrong credentials then "Bad Credentials" message is displayed. Additionally, each user's password is encrypted after registration.

In addition the user's role is parametrically checked to ensure correct operation and "integrity" of the application. To implement this on the frontend, we use Thymeleaf, which provides direct ways to check a user's role, whether they are authenticated, and other useful functions.

Front-Back Communication:

The front-back communication is done with REST that we have implemented in the backend. To "easily" call some of the methods we have implemented, we use jquery (eg to delete a user by pressing a button). It is noteworthy that we make use of the functions of thymeleaf to check if the data already exists in the database, informing the user accordingly (eg if the user during registration puts a username that already exists in the database → return an appropriate message in the corresponding field of the filling form in front end). Another reason we used thymeleaf in the frontend is to directly check whether or not the page viewer has been authenticated and, depending on their role, show them the appropriate content. To implement the various functions - controls (for the input) and other functions (such as brightness slider) in the frontend we used javascript.

Access:

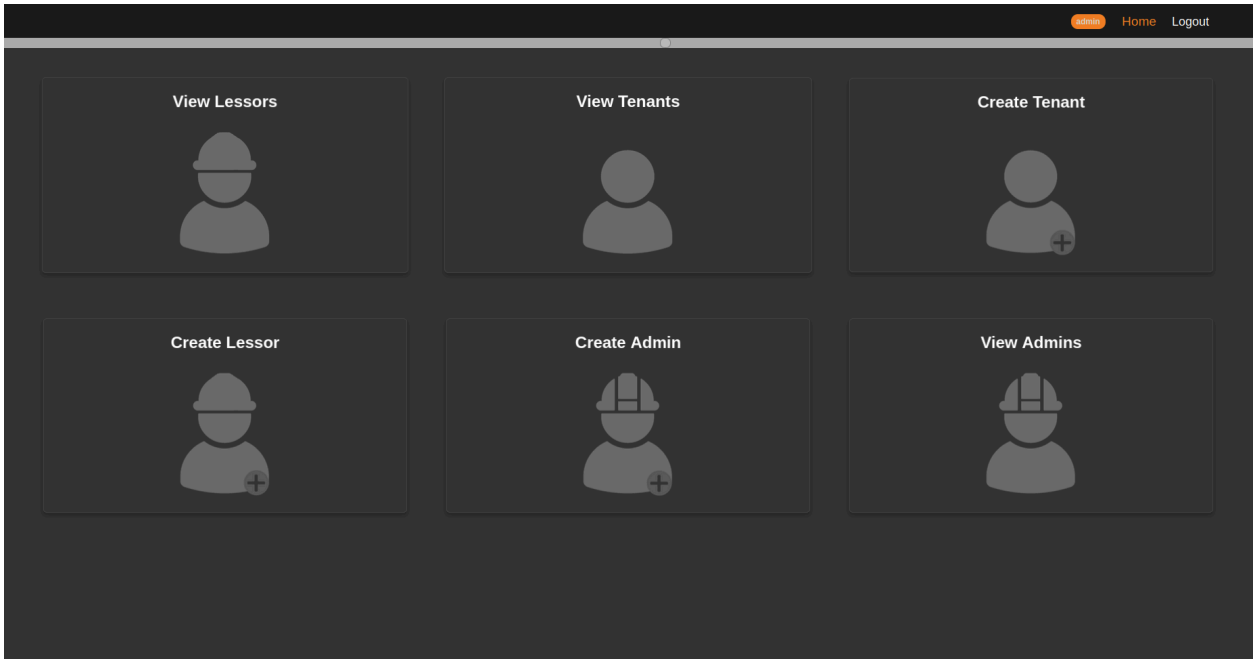
Each user's access is described in the corresponding methods in the "Access" column (see at user manual). For user access to the backend, we have defined with Spring Security (specifically in the fileSecurityConfig) which REST calls each role is allowed to use. Also, in the frontend we use Thymeleaf sec:authorize to show users only the methods - functions they have

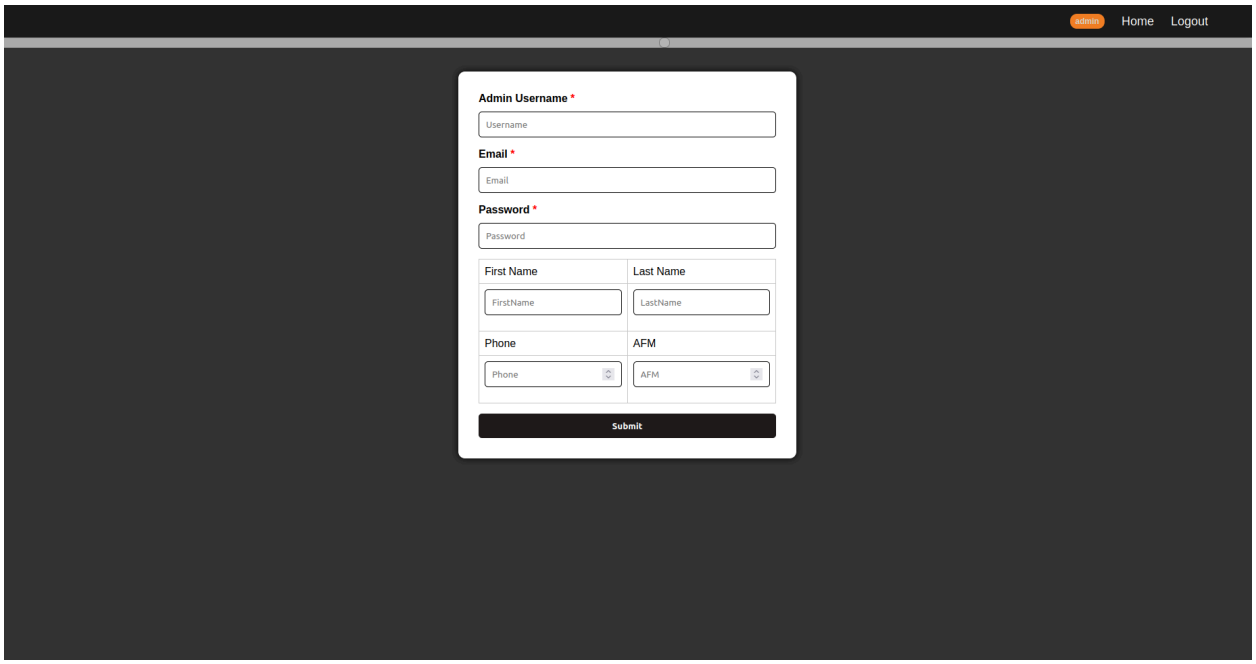
access. It is noteworthy that the security we have defined in the backend "agrees" with the functions we show in each role (in the frontend).

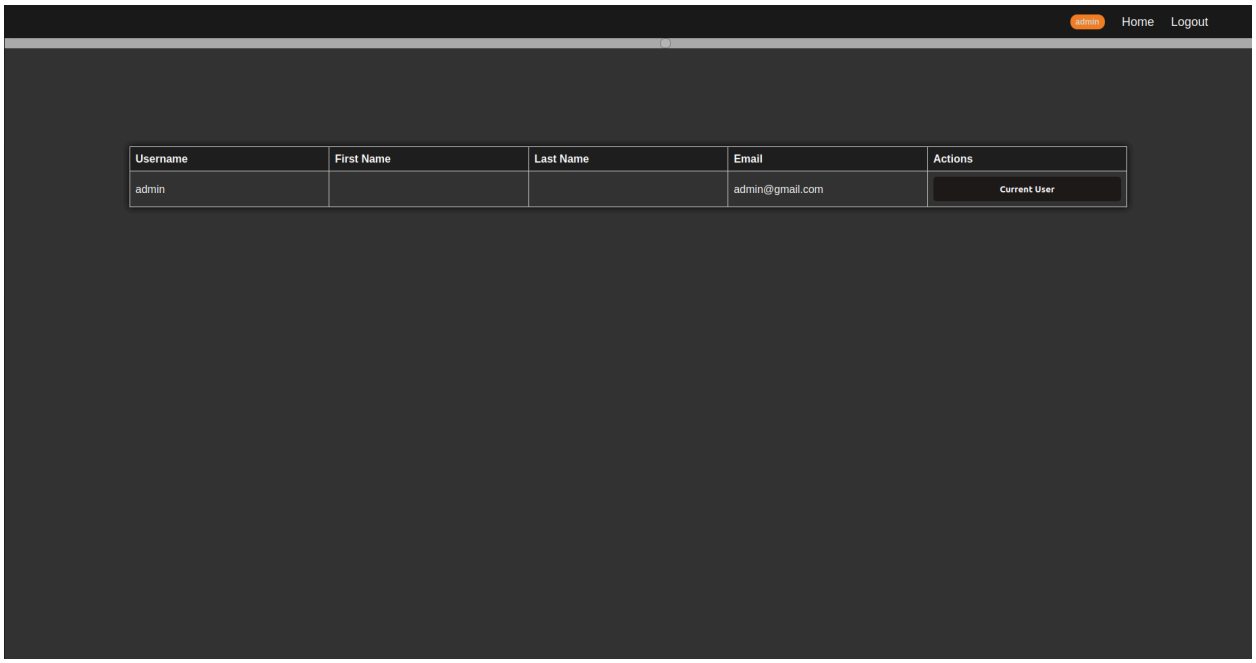
Screenshots of Frontend:

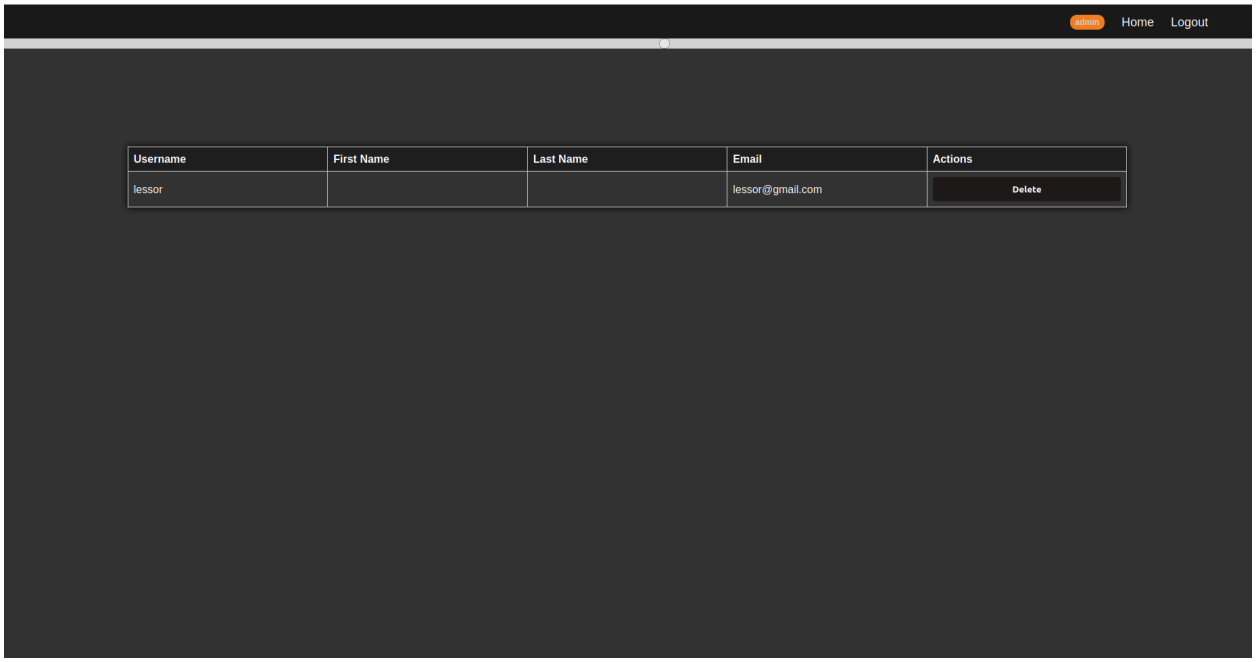
In the frontend we have a brightness slider, in which the user can adjust the brightness (dark & light mode). Indicative of the light mode, see the screens of the lessor.

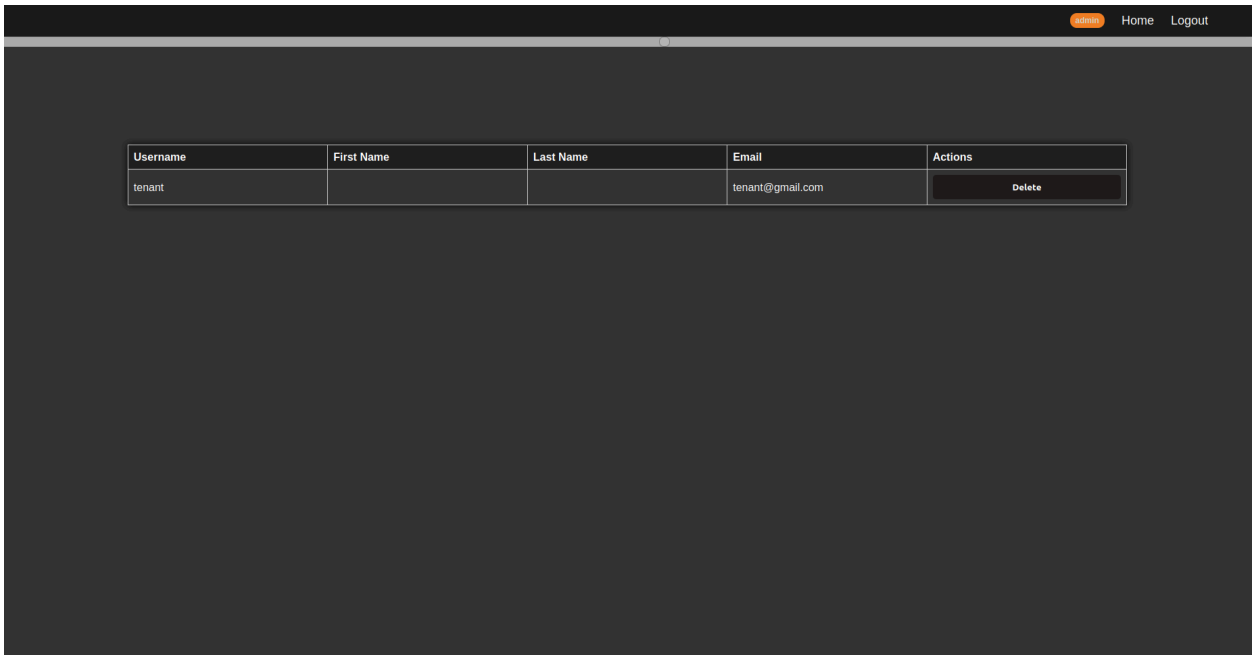
Admin Frontend (Dark Mode):



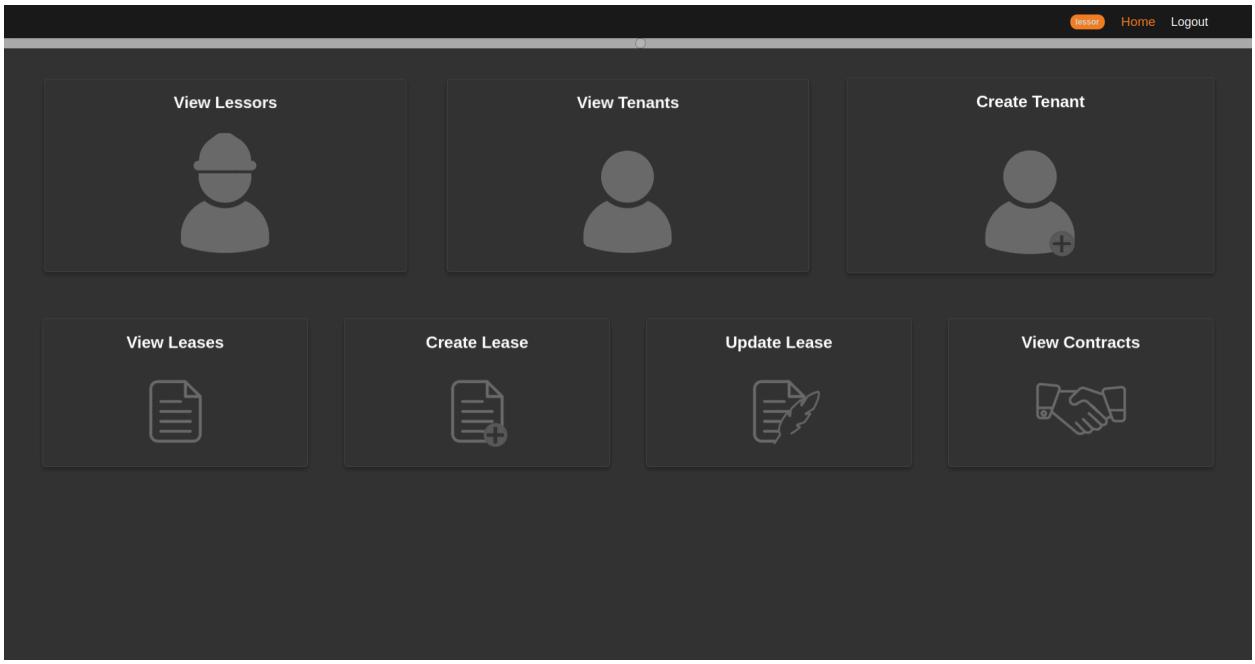

admin
Home
Logout
Admin Username *
Username
Email *
Email
Password *
Password
First Name
Last Name
FirstName
LastName
Phone
AFM
Phone
AFM
Submit

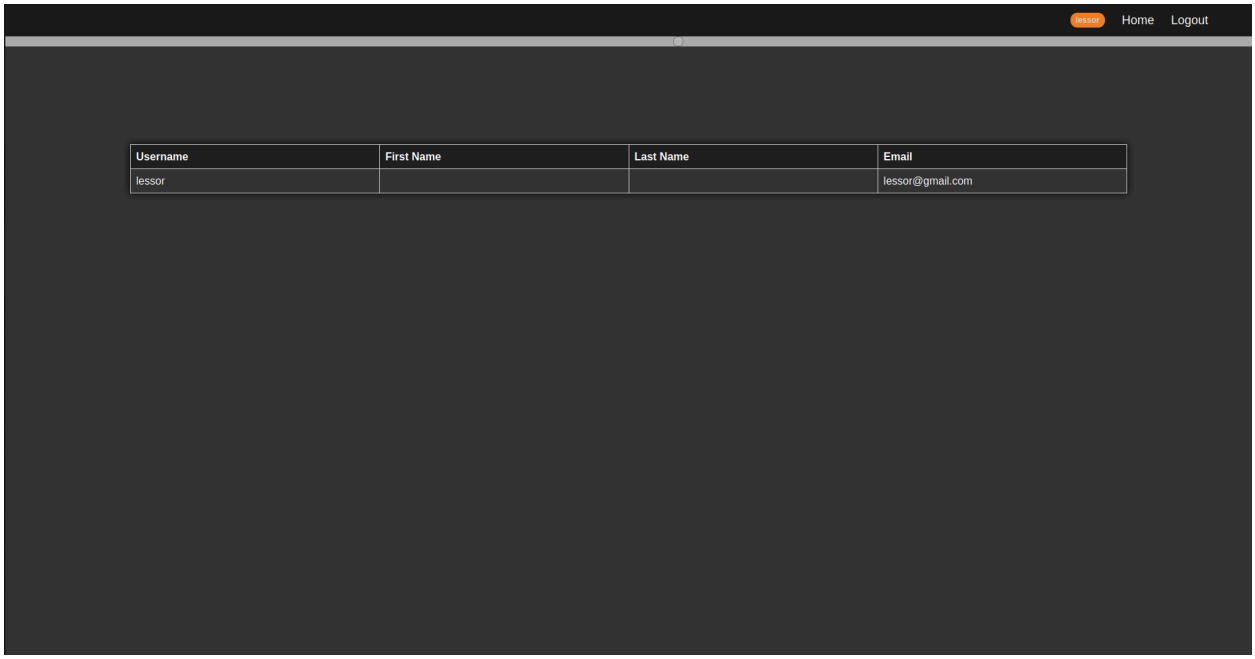

admin
Home
Logout
Username
First Name
Last Name
Email
Actions
admin
admin@gmail.com
Current User

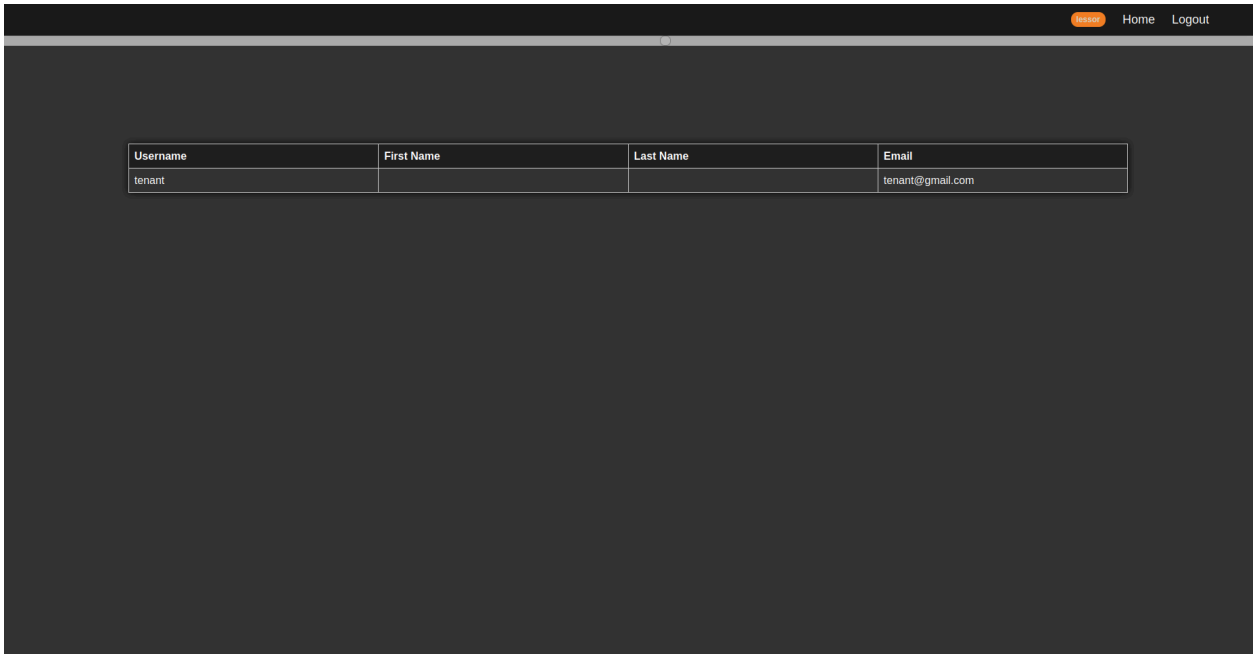


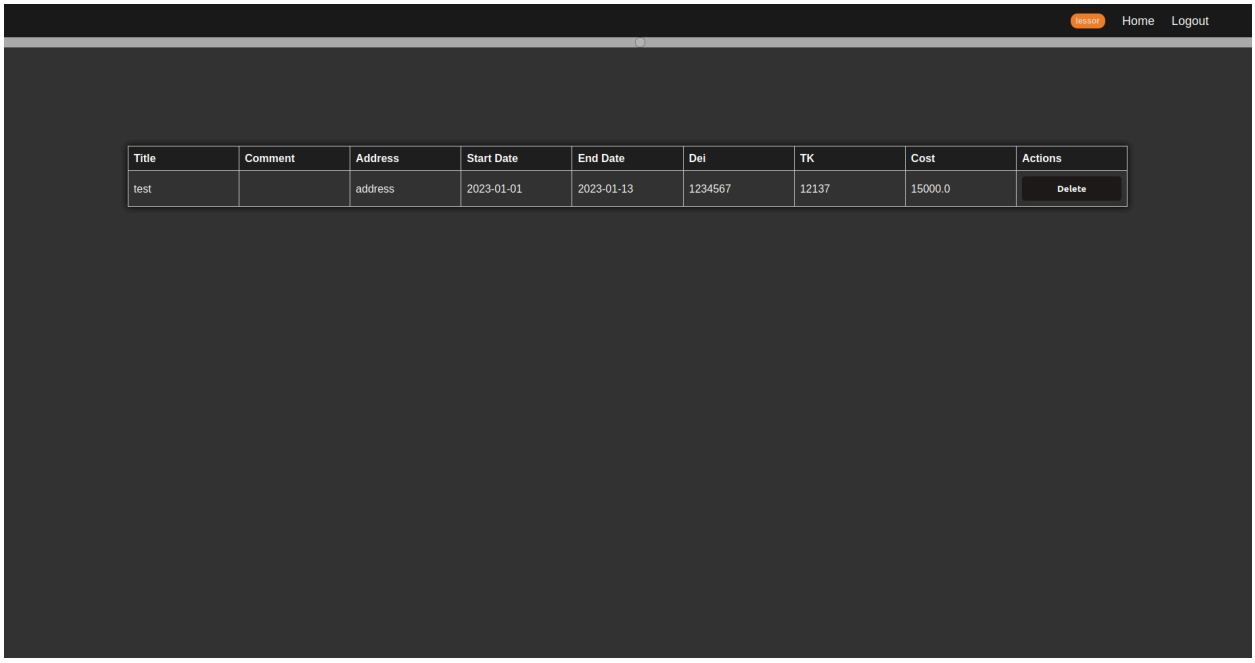


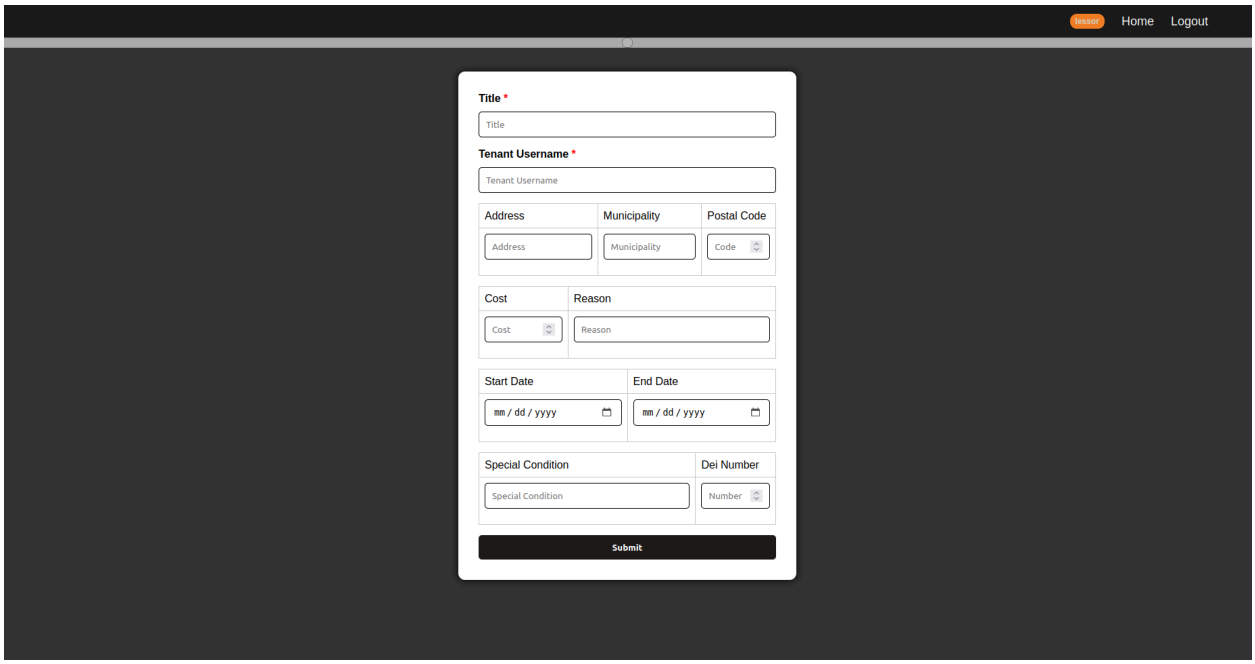
Lesser Frontend (Dark Mode):

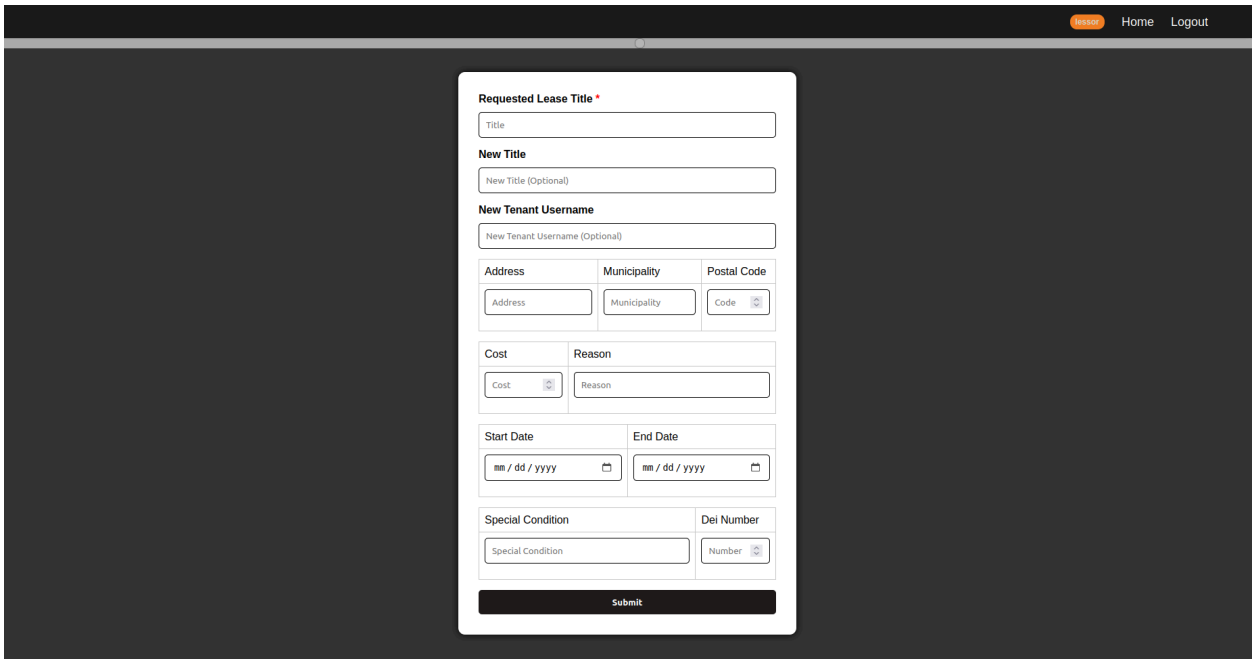

lessor
Home
Logout
View Lessors
View Tenants
Create Tenant
View Leases
Create Lease
Update Lease
View Contracts

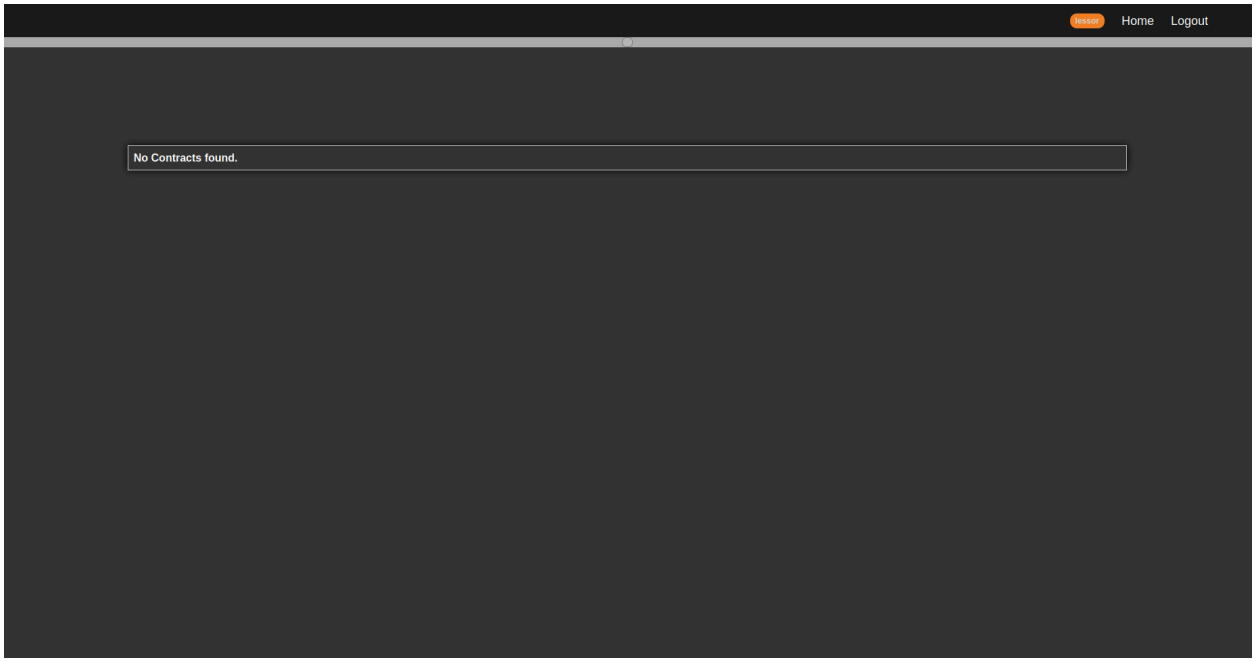

lessor
Home
Logout
Username
First Name
Last Name
Email
lessor
lessor@gmail.com


lessor
Home
Logout
Username
First Name
Last Name
Email
tenant
tenant@gmail.com

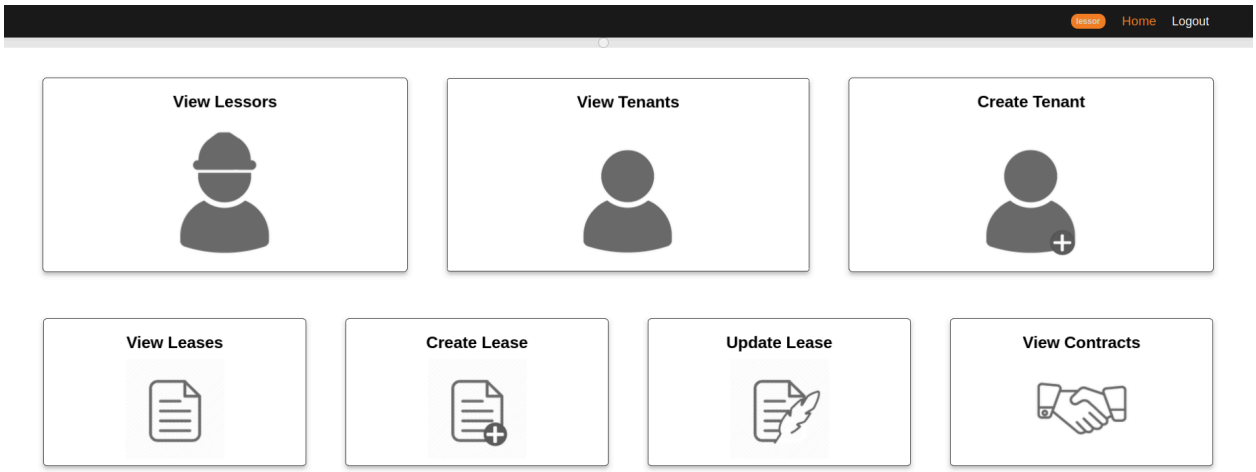

lessor
Home
Logout
Title
Comment
Address
Start Date
End Date
Dei
TK
Cost
Actions
test
address
2023-01-01
2023-01-13
1234567
12137
15000.0
Delete


lessor
Home
Logout
Title *
Title
Tenant Username *
Tenant Username
Address
Municipality
Postal Code
Address
Municipality
Code
Cost
Reason
Cost
Reason
Start Date
End Date
mm / dd / yyyy
mm / dd / yyyy
Special Condition
Dei Number
Special Condition
Number
Submit

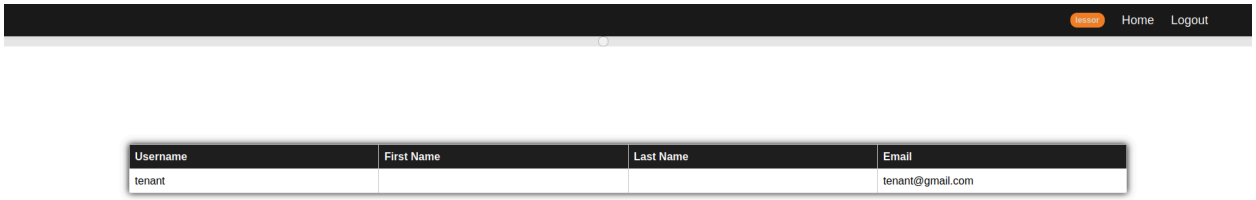

lessor
Home
Logout
Requested Lease Title *
Title
New Title
New Title (Optional)
New Tenant Username
New Tenant Username (Optional)
Address
Municipality
Postal Code
Address
Municipality
Code
Cost
Reason
Cost
Reason
Start Date
End Date
mm / dd / yyyy
mm / dd / yyyy
Special Condition
Dei Number
Special Condition
Number
Submit

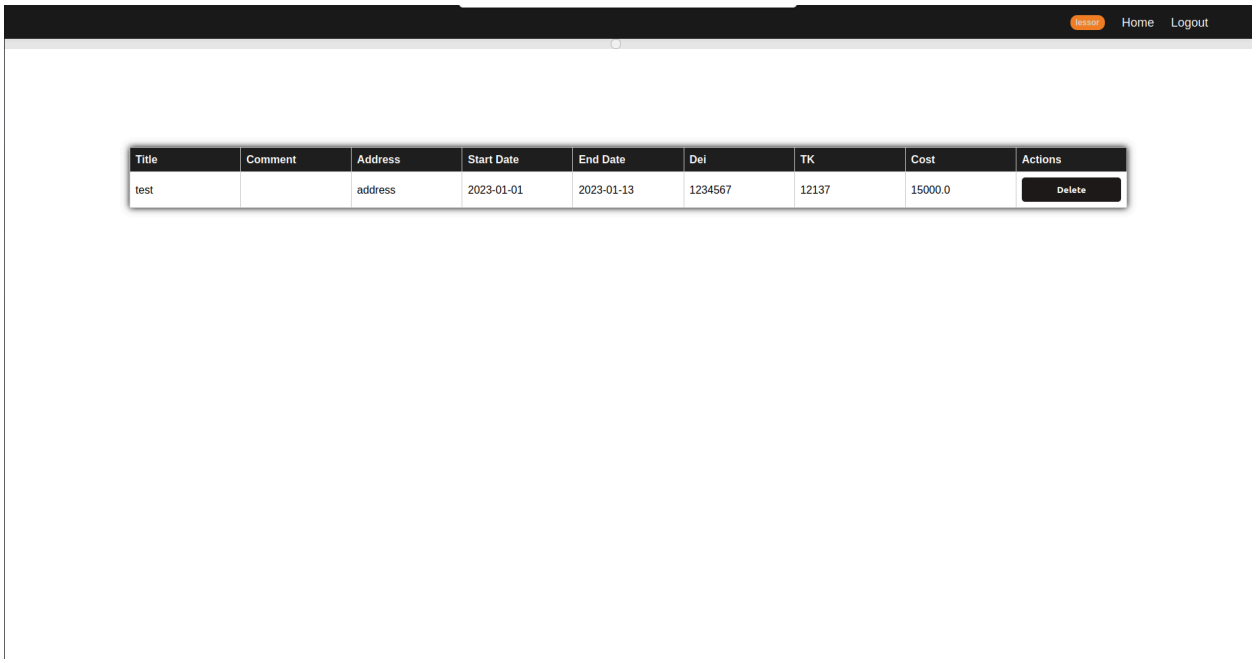


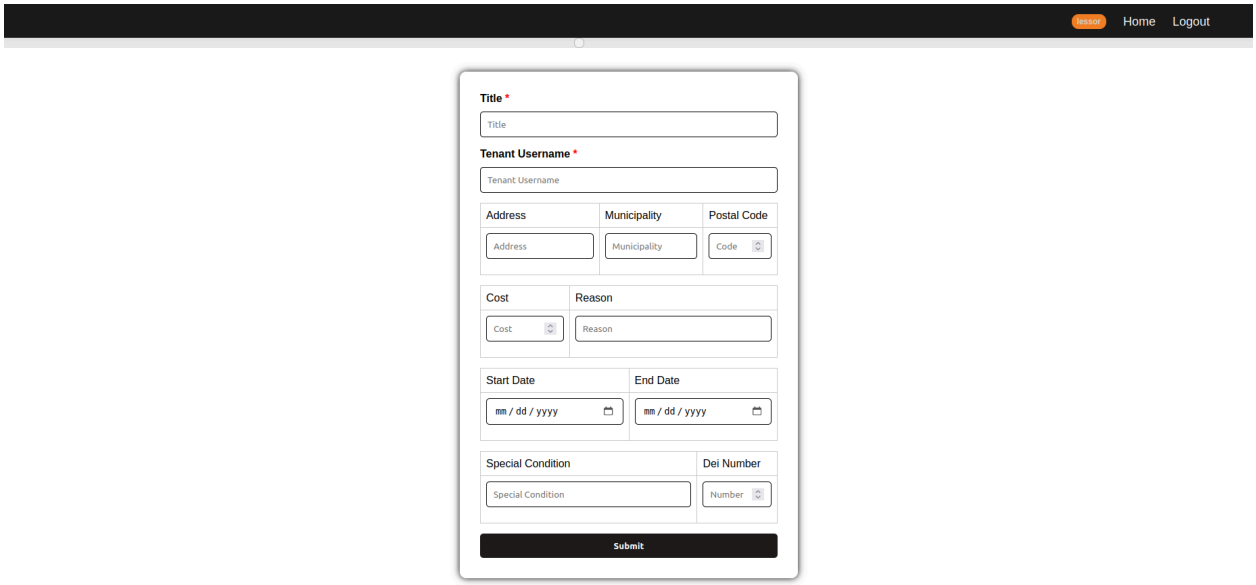
Lesser Frontend (Light Mode):

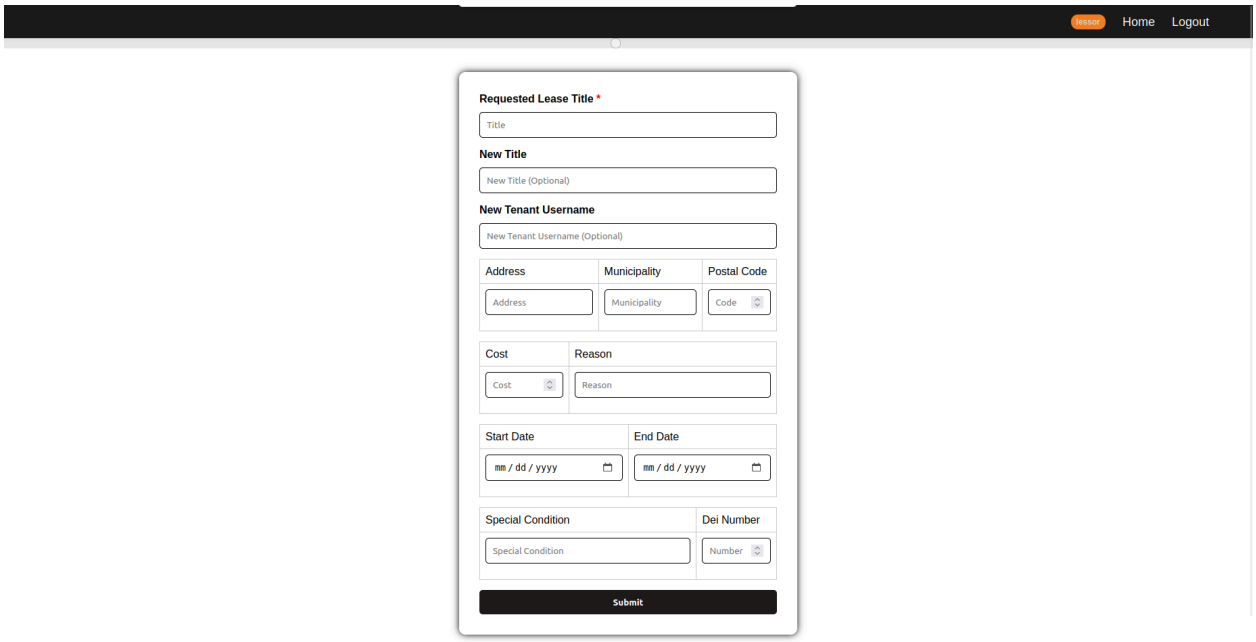


Username	First Name	Last Name	Email
lessor			lessor@gmail.com

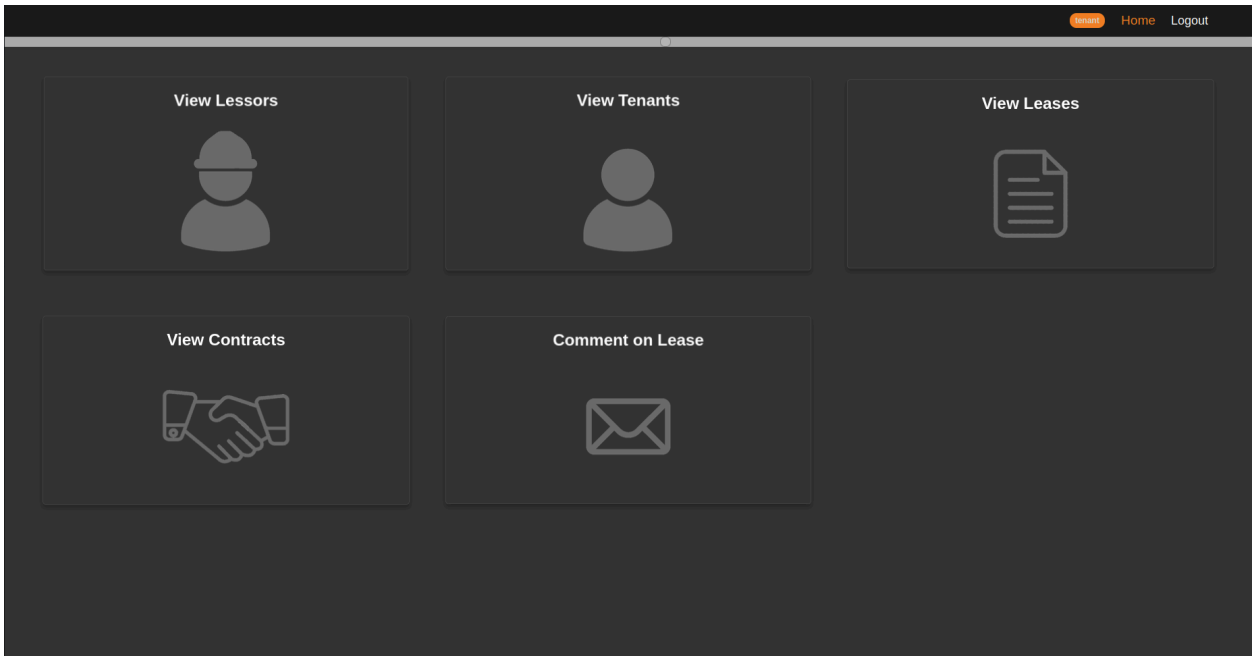

lessor
Home
Logout
Username
First Name
Last Name
Email
tenant
tenant@gmail.com

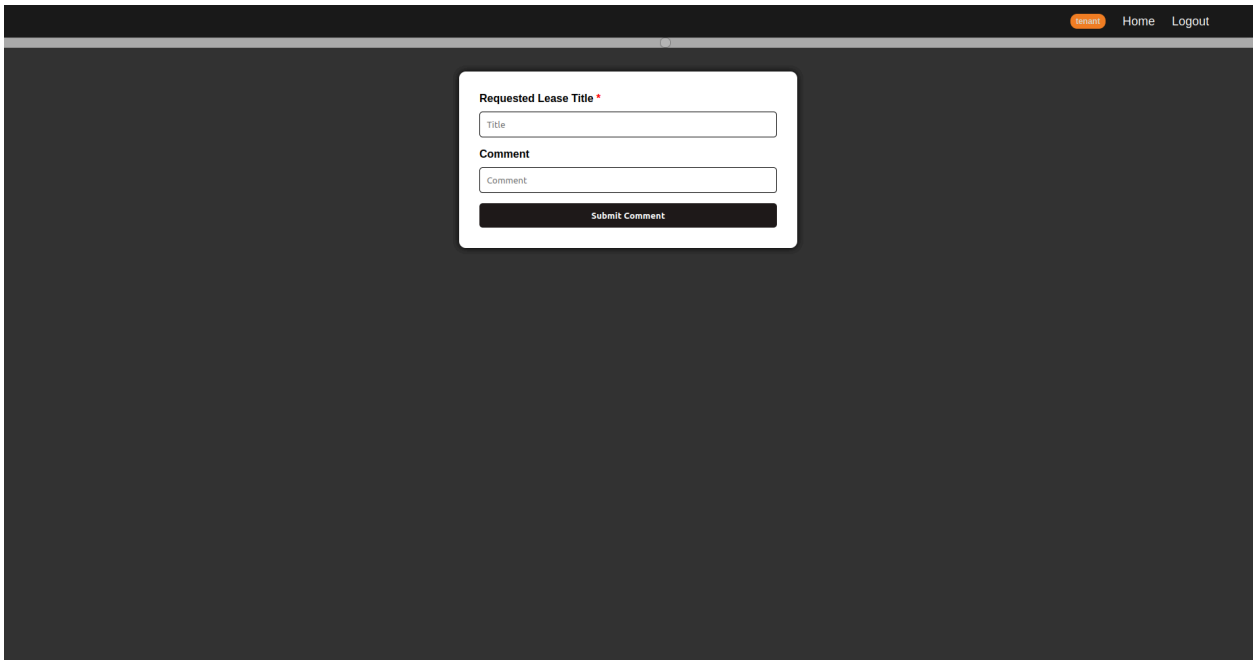

lessor
Home
Logout
Title
Comment
Address
Start Date
End Date
Dei
TK
Cost
Actions
test
address
2023-01-01
2023-01-13
1234567
12137
15000.0
Delete

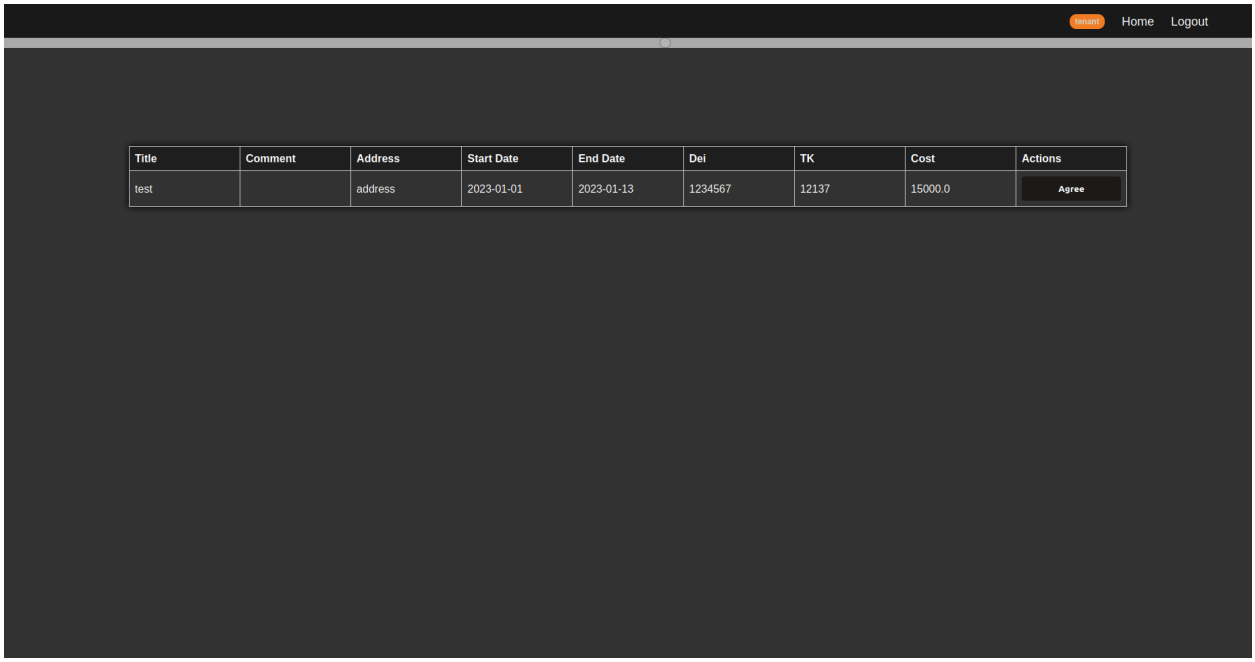

lessor
Home
Logout
Title *
Title
Tenant Username *
Tenant Username
Address
Municipality
Postal Code
Address
Municipality
Code
Cost
Reason
Cost
Reason
Start Date
End Date
mm / dd / yyyy
mm / dd / yyyy
Special Condition
Dei Number
Special Condition
Number
Submit

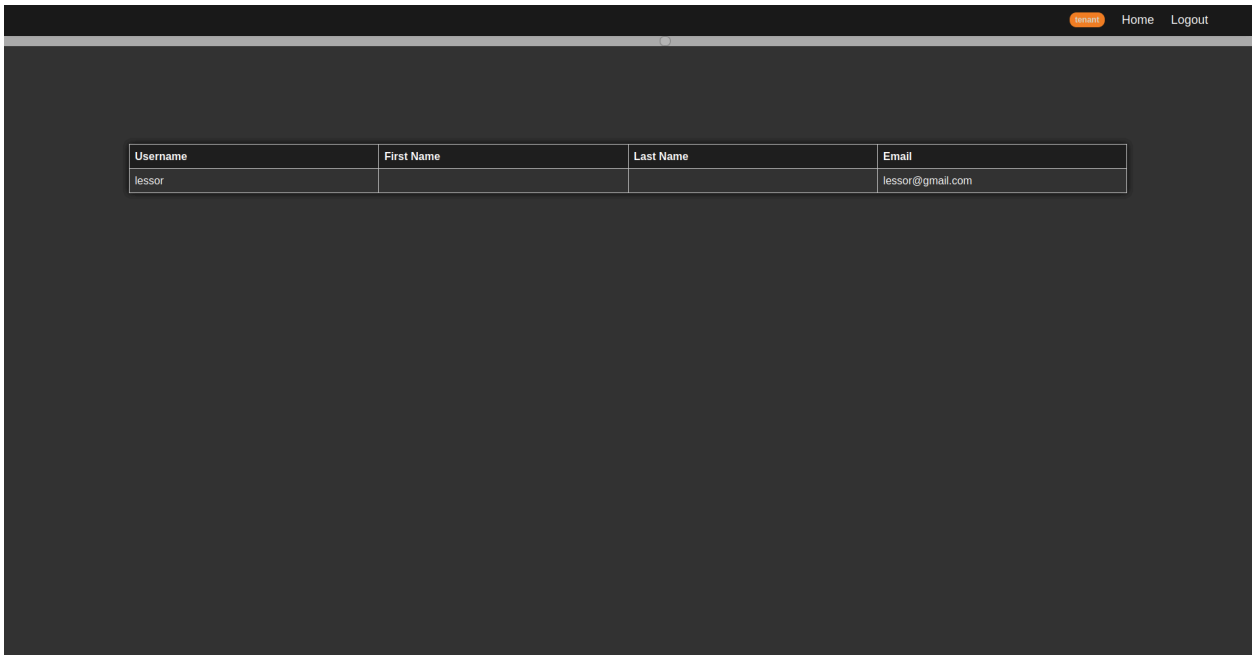

lessor
Home
Logout
Requested Lease Title *
Title
New Title
New Title (Optional)
New Tenant Username
New Tenant Username (Optional)
Address
Municipality
Postal Code
Address
Municipality
Code
Cost
Reason
Cost
Reason
Start Date
End Date
mm / dd / yyyy
mm / dd / yyyy
Special Condition
Dei Number
Special Condition
Number
Submit

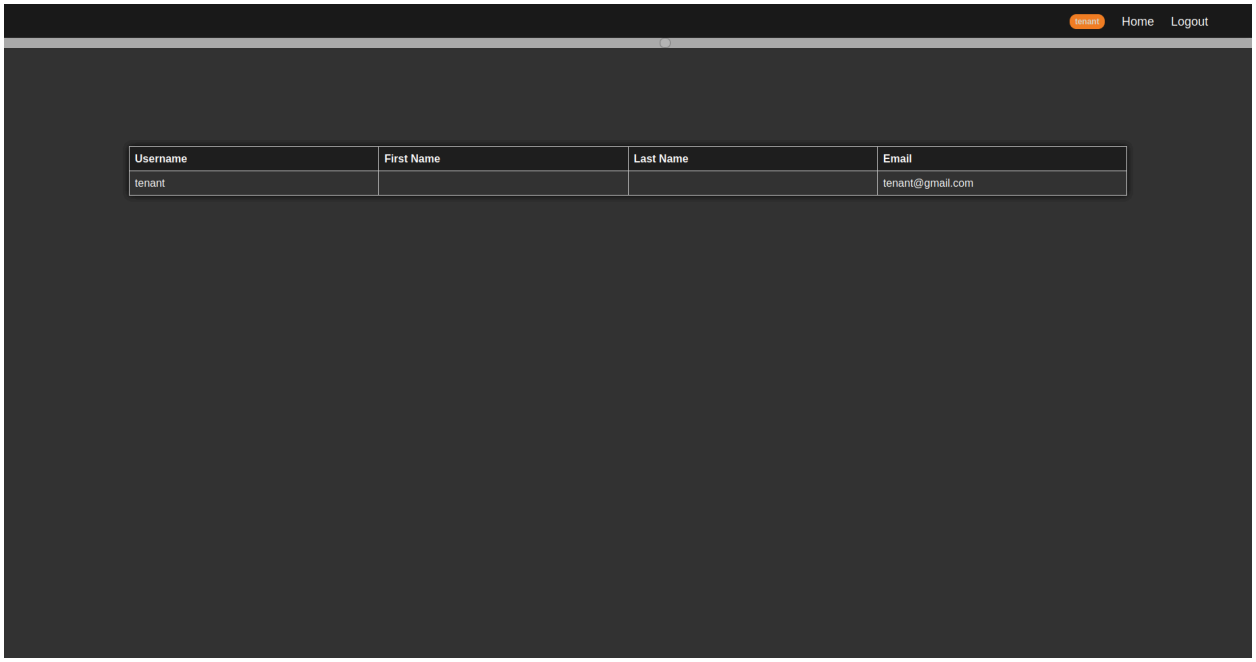
Tenant Frontend (Dark Mode):






tenant
Home
Logout
Title
Comment
Address
Start Date
End Date
Dei
TK
Cost
Actions
test
address
2023-01-01
2023-01-13
1234567
12137
15000.0
Agree


tenant
Home
Logout
Username
First Name
Last Name
Email
lessor
lessor@gmail.com



4th Deliverable: Implementation

Repository Links:

- Main
- Backend with Token authentication
- Backend with Basic authentication
- Front end

User Manual:

The frontend has the same basis as the backend-basic-auth & backend-token-auth, which means that whatever elements are created/deleted in the frontend (or the corresponding ones in backend-basic-auth or backend-token-auth), there will be a similar update for the rest.

It is important to note that in the methods we use curly brackets “{ ... }” to indicate that this “piece” is a variable and will need to be changed accordingly, in order to have the desired result (see also the method descriptions). Also, in the final request**not**curly brackets are placed but only the value of the variable.

See the previous deliverable (and the manual with the rest of the backend methods)here .

User (Backend)	Description	Method	Access	Test Input (JSON)
/user/tenant/{id}	Deletes the tenant with the input id.	DELETE	ADMIN	-
/user/lessor/{id}	Deletes the lessor with the input id.	DELETE	ADMIN	-
/user/leases/{id}	Deletes the lease with the input id.	DELETE	LESSOR	-
/user/admin/{id}	Deletes the admin with the input id.	DELETE	ADMIN	-
/user/leases/agree/{id}	Agree with it lease with input id (creates the corresponding contract).	POST	TENANT	-

UserForm (Front end)	Description	Method	Access	Test Input (JSON)
/	Returns the index	GET	ALL	-
/lessorform	Displays the lessor form	GET	ADMIN	-

/lessorlist	Displays the list of lessors	GET	LESSOR, TENANT	-
/leaselist	Displays the list of leases	GET	LESSOR, TENANT	-
/contractlist	Displays the list of contracts	GET	LESSOR, TENANT	-
/lessorform	Stores the lessor.	POST	ADMIN	-
/tenantform	Displays the tenant form	GET	ADMIN, LESSOR	-
/tenantlist	Displays the list of tenants.	GET	All the logged in users.	-
/tenantform	Stores the tenant.	POST	admin, LESSOR	-
/leaseform	Displays the form creation of the lease.	GET	LESSOR	-
/leaseform	Stores the lease.	POST	LESSOR	-
/leaseupdate	Displays the form for updating a lease.	GET	LESSOR	-
/leaseupdate	Refreshes the data of the lease.	POST	LESSOR	-
/adminlist	Displays the list of admins.	GET	ADMIN	-
/adminform	Saves Admin.	POST	ADMIN	-

/adminform	Displays the form of admin.	GET	ADMIN	-
/leasecom	Displays the form to fill it out lease.	GET	TENANT	-
/leasecom	It finds the lease that the comment is going to it is listed based on the title, and saves the comment.	POST	TENANT	-
/verifyuser	It does a Verify one verification code.	POST	ANY	-
/verifyuser	Displays the verification code form.	GET	ANY	-