Aleeza Balbas, Ayman Zaher, Daniel Zhuo

ASSIGNMENT 1 INSTRUCTIONS:

Each group will select one of the following three projects to analyze the requirements of the project for the Front End (only) at this stage. The basic idea is to identify the requirements and write test cases to test the requirements.

CHOSEN: PROJECT #1 - RENTAL MANAGEMENT WEB APPLICATION

This project aims to develop a web application for managing various rental properties, including residential and commercial spaces. Key features will include tenant management, lease agreement creation, rent collection, and maintenance. The goal is to enhance operational efficiency and improve communication between property owners and tenants.

UI DESIGN LINK:

https://www.figma.com/design/nvKhRHu02neP3D1sCZmWJY/CISC327%3A-Group-57AA-UI?node-id=2106-610&t=hPV9BrS4x3Y4PIrM-1

FRONT END REQUIREMENTS:

REQUIREMENT #1: User Login Authentication and Authorization

The system shall provide secure login and account-creation functionality with role-based access control.

Features:

- Different access levels for property owners, tenants, and administrators.
- Password recovery and multi-factor authentication options.

Test Case Name**:** Valid User Registration

Objective: To verify that users can successfully register for a property owner or tenant account after providing valid account information.

Action:

1. In the User Registration page, fill in valid account details including name, email and role (property owner or tenant) and click on the “Register” button.

Assert:

1. The user is successfully registered and is redirected to the MFA Setup page with a message indicating registration success.

Test Case Name**:** Invalid User Registration

Objective: To verify that users cannot register for a property owner or tenant account after providing invalid account information.

Action:

1. In the User Registration page, fill in account details such as name, email and role (property owner or tenant) with invalid or missing information and click on the “Register” button.

Assert:

1. The user is not registered and a message indicating the registration failure reason is displayed to the user.

Test Case Name**:** MFA Setup

Objective: To verify that users can setup a Multi Factor Authentication method to be used during future authentication.
Action:

1. In the MFA Setup page, use the MFA setup token to generate the next six-digit MFA code and enter it into the verification code text input.
2. Click on "Verify".

Assert:

1. MFA has been setup and the user is redirected to the homepage with a message indicating MFA Setup was completed successfully.

Test Case Name**:** Valid User Login

Objective: To verify that users with a registered account can successfully login into the system after providing valid credentials and verify their identity with their registered MFA method.

Action:

1. In the Login page, fill in valid account credentials for a registered user and click on the "Login" button.
2. After the prompt to verify the users identity using their registered MFA method is displayed, generate the users current six-digit verification code and insert it into the verification code text input field and then click "Verify".

Assert:

1. The user is successfully logged in and is redirected to the homepage with a message indicating login success.

Test Case Name**:** Invalid User Login

Objective: To verify that users cannot login into the system without providing valid account credentials.

Action:

1. In the User Login page, click on the "Login" button.

Assert:

1. The user is not logged in and a message indicating the login failure reason is displayed to the user.

REQUIREMENT #2: Property Management

The system shall allow property owners to add, edit, and delete property listings for residential and commercial spaces.

Features:

- Upload property details, including address, type, size, and images.
- Categorize properties as available or occupied.
- View a dashboard of all managed properties.

Test Case Name**:** View Managed Properties

Objective: To verify that a logged in registered property owner can view all of the properties they have managed by the system.

Action:

1. After logging in as a property owner, visit the Managed Properties page.

Assert:

1. All of the user's managed properties are visible on the page.

Test Case Name**:** Add a New Property to Manage

Objective: To verify that a logged in registered property owner can add a new property to be managed by the system.

Action:

1. After logging in as a property owner, visit the Add New Managed Property page.
2. Fill in valid property details such as address, type, size and images and click on the "Add" button.

Assert:

1. The new property is successfully added.
2. The user is redirected to the Managed Property Details page for the new property and a message indicating success is displayed to the user.
3. All of the details entered during property addition are visible on the Managed Property Details page.

Test Case Name**:** View Managed Property Details

Objective: To verify that a logged in registered property owner can view the details of a property the user has added to be managed by the system.

Action:

1. After logging in as a property owner, visit the Managed Property Detail page for one of the users managed properties.

Assert:

1. The page is accessible and all of the details for the managed property are displayed on the page.

Test Case Name**:** Edit Managed Property Details

Objective: To verify that a logged in registered property owner can edit the information of a property the user has added to be managed by the system.

Action:

1. After logging in as a property owner, visit the Managed Property Detail page for one of the users managed properties.
2. Click the "Edit" button.
3. After the details turn into editable fields, modify the value of one of the fields.

4. Click the “Save” button.

Assert:

1. The page is accessible and all of the details for the managed property are displayed on the page.
2. After clicking the “Edit” button, the details turn into editable fields.
3. After clicking the “Save” button, the details are no longer editable and a message indicating saving the edited information was successful.

Test Case Name**:** Delete Managed Property

Objective: To verify that a logged in registered property owner can remove a property that the user had previously added to be managed by the system.

Action:

1. After logging in as a property owner, visit the Managed Property Details page for one of the users managed properties.
2. Click on the “Delete” button.
3. When the prompt to confirm the deletion is displayed, click on “Delete”.

Assert:

1. The page is accessible and all of the details for the managed property are displayed on the page.
2. The prompt to confirm the deletion is displayed to the user after clicking “Delete”
3. The user is redirected to the Managed Properties page and the deleted property is not visible.

REQUIREMENT #3: Tenant Management

The system shall enable property owners to manage tenant information.

Features:

- Add new tenants with personal and contact details.
- Edit tenant information and lease status.
- Access tenant rental history and payment records.

Test Case Name**:** View All Tenants

Objective: To verify that a logged in registered property owner can view all pending, current, and past tenants.

Action:

1. After logging in as a property owner, visit the Tenants page.
2. Expand the past tenants section.

Assert:

1. The page is accessible and all pending, current and past tenants are visible.

Test Case Name**:** View Property Tenants

Objective: To verify that a logged in registered property owner can view pending, current, and past tenants for a given property.

Action:

1. After logging in as a property owner, visit the Managed Property Details page for one of the users managed properties.
2. Expand the past tenants section.

Assert:

1. The page is accessible and all pending, current and past tenants for the given managed property are visible.

Test Case Name**:** View Tenant History

Objective: To verify that a logged in registered property owner can view the property owner's current and previously managed properties a given tenant is interested in renting, is currently renting or has rented and all of the payments made by the tenant to the property owner.

Action:

1. After logging in as a property owner, visit the Tenant Details page for a tenant that is interested in renting, is currently renting or has rented a current or previously managed property added to the system by the property owner.

Assert:

1. The page is accessible and all pending, current and past managed properties owned by the property owner are visible.
2. A history of all the payments the tenant has made to the property owner is visible.

Test Case Name**:** Add Tenant to Property

Objective: To verify that a logged in registered property owner can add a tenant to a property owned by the property owner and currently managed by the system as a renter.

Action:

1. After logging in as a property owner, visit the Managed Property Details page for one of the users currently managed properties that has had a lease agreement prepared using the system.

2. Click on the “Add Tenant” button.
3. Once the prompt to enter the tenant's information has appeared, insert the email address of a controlled mailbox and some additional information about the tenant into the text fields present in the prompt.
4. Select an existing lease agreement to add the tenant with.
5. Click “Add Tenant”.

Assert:

1. The prompt to enter the new tenant’s information is no longer visible.
2. A message indicating adding the tenant was a success is displayed to the user.
3. The new tenant is listed on the currently visited Managed Property Details page.
4. The tenant is notified by email and the system, if registered, that they have been added to the property.

Test Case Name**:** Edit Tenant Information

Objective: To verify that a logged in registered property owner can edit previously saved information for a given tenant that is interested in renting, is currently renting or has rented at least one of the property owner's current or previously managed properties.

Action:

1. After logging in as a property owner, visit the Tenant Details page for a tenant that is interested in renting, is currently renting or has rented a current or previously managed property added to the system by the property owner.
2. Click the “Edit Information” button.
3. After the field containing the tenant’s previously saved information becomes editable, modify the value in the text box.
4. Click the “Save Information” button.

Assert:

1. The tenant saved information field is no longer editable and contains the updated information.
2. A message indicating saving the information was a success is displayed to the user.

Test Case Name**:** Remove Tenant from Property

Objective: To verify that a logged in registered property owner can remove a given tenant that is currently renting one of the property owner's current managed properties from the renters of the property.

Action:

1. After logging in as a property owner, visit the Managed Property Details page for one of the property owner’s currently managed properties that has at least one active tenant.
2. In the tenants list, click the “Remove” button next to one of the current tenants’ names.
3. After the prompt to confirm the removal appears, click the “Remove” button.

Assert:

1. The tenant is moved to the past tenants section and is not immediately visible.
2. A message indicating removing the tenant from the properties renters was a success is displayed to the user.
3. The tenant is notified by email and the system that they have been removed from the property.

REQUIREMENT #4: Lease Agreement Creation

The system shall facilitate the creation and management of lease agreements.

Features

- Generate customizable lease agreement templates.
- Allow digital signatures for lease agreements.
- Store and retrieve signed agreements securely.

Test Case Name: View Lease Agreements

Objective: To verify that a logged in registered property owner or tenant can view the pending, current and previous lease agreements between the tenant and the property owner for a given property owned by the property owner and currently or previously managed by the system and the tenant is interested in, currently or previously renting the property.

Action:

1. After logging in as a property owner or tenant, visit the Property Details page for one of the property owner's currently managed properties that has at least one active tenant or a property the tenant is interested in renting, is currently renting or previously rented.
2. In the tenants list, click the "Agreement" button next to one of the current tenants' names.

Assert:

1. The user is redirected to the Lease Agreement Viewing page.
2. The proper lease agreement between the tenant and the property owner for the selected property is displayed.

Test Case Name: Create Lease Agreement

Objective: To verify that a logged in registered property owner can create a lease agreement for a given property owned by the property owner and currently managed by the system.

Action:

1. After logging in as a property owner, visit the Create Lease Agreement page for one of the property owner's currently managed properties.
2. Enter the information required into all of the fields including price and payment frequency and then click on "Create".

Assert:

1. The user is redirected to the Managed Property Details page.
2. The newly created lease agreement is listed on the page.
3. A message indicating the creation of the agreement was a success is displayed to the user.

Test Case Name: Sign Lease Agreement

Objective: To verify that a logged in registered tenant can sign a lease agreement for a managed property a property owner has added them to as a tenant.

Action:

1. After logging in as a tenant that has been added to an actively managed property and has an agreement to sign, visit the Lease Agreement Viewing page for a lease agreement the tenant has yet to sign.
2. Click the "Sign" button.
3. After the prompt to fill in the required information to sign the lease, insert the requested information into the text fields on the prompt..
4. Click on the "Sign" button.

Assert:

1. The user is redirected to the Managed Property Details page.

2. The signed lease agreement is listed on the page and indicates it has been signed.
3. A message indicating the signing of the agreement was a success is displayed to the user.

REQUIREMENT #5: Rent Collection Interface

The system shall provide a secure platform for tenants to make rent payments online.

Features:

- Support multiple payment methods (credit card, bank transfer, etc.).
- Send automatic payment reminders to tenants.
- Generate receipts and update payment records upon successful transactions.

Test Case Name: Make a Rent Payment

Objective: To verify that a logged in registered tenant that is currently renting a property managed by the system can make rent payments to the property owner.

Action:

1. After logging in as a tenant that is renting at least one property managed by the system, visit the Make a Payment page for one of the properties currently rented out by the tenant.
2. Select the credit card payment option and enter the required information required into all of the fields such as card number.
3. Click the “Pay” button.

Assert:

1. The user is redirected to the Managed Property Details page.
2. The payment is listed under the payments section of the page.
3. A message indicating the payment was a success is displayed to the user.

Test Case Name: View Rent Payment Receipt

Objective: To verify that a logged in registered tenant that is currently renting or previously rented a property managed by the system can view receipts for all the payments they have made.

Action:

1. After logging in as a tenant that is renting or has rented at least one property managed by the system, visit the Property Details page for one of the properties the tenant has made a payment for.
2. In the Payment History list, click the “Receipt” button next to one of the payments.

Assert:

1. The user is redirected to a page with the receipt visible to the user.
2. The correct receipt is displayed and the payment details are correct.

Test Case Name: Receive Upcoming Payment Reminder

Objective: To verify that a logged in registered tenant receives a reminder seven days before a rent payment is due for a managed property they are currently renting.

Action:

1. After logging in as a tenant that has a payment due in under seven days, open the Notification tray.

Assert:

1. A notification indicating an upcoming payment is due is listed in the notification tray.
2. The notification was added seven days before the payment is due.

REQUIREMENT #6: Maintenance Requests

The system shall allow tenants to submit maintenance requests and enable property owners to manage them.

Features:

- Tenants can submit requests with descriptions and attach images.
- Property owners receive notifications of new requests.
- Track the status of maintenance tasks (e.g., pending, in-progress, completed).

Test Case Name: Submit Maintenance Request

Objective: To verify that a logged in registered tenant can submit a maintenance request for a managed property they are currently renting.

Action:

1. After logging in as a tenant that is currently renting a managed property, visit the Property Details page.
2. Click on the "New Maintenance Request" button.
3. After the maintenance request creation prompt is displayed, enter all the required information into the fields on the prompt and attach an image to the request using the button on the prompt.
4. Click on the "Submit Maintenance Request" button.

Assert:

1. The prompt to enter maintenance information is no longer visible.
2. A message indicating submitting the maintenance request was successful is displayed to the user.
3. The newly created maintenance request is visible in the Maintenance Requests list on the Property Details page.

Test Case Name: View Maintenance Request

Objective: To verify that a logged in registered property owner can view all maintenance requests for a currently or previously managed property.

Action:

1. After logging in as a property owner that has a managed property with a maintenance request, visit the Managed Property Details page for a property with at least one maintenance request.
2. Click on one of the listed Maintenance Requests.

Assert:

1. The user is redirected to the Maintenance Request Detail page and the details and progress of the maintenance request are visible.

Test Case Name: Update Maintenance Request Status

Objective: To verify that a logged in registered property owner can update the status of an active maintenance request for a currently managed property.

Action:

1. After logging in as a property owner that has a managed property with a maintenance request, visit the Maintenance Request Details page for an active maintenance request for a managed property.
2. Click the "Update" button.
3. After the fields on the page containing the maintenance details are editable, modify some of the data already present.
4. Click on the "Save" button.

Assert:

1. The fields on the page are no longer editable and have been updated with the modified values.
2. A message indicating saving the maintenance request was successful is displayed to the user.

Test Case Name**:** Maintenance Request Added Notification

Objective: To verify that a logged in registered property owner can update the status of an active maintenance request for a currently managed property.

Action:

1. After logging in as a property owner that has a managed property with a maintenance request, open the notification tray

Assert:

1. A notification is present in the tray indicating a request has been created for one of the users managed properties.

REQUIREMENT #7: In-Web Messaging Tool

The system shall include a messaging feature for direct communication between property owners and tenants.

Features:

- Send and receive messages within the application.
- Support for attachments (e.g., documents, images).
- Notifications for new messages.

Test Case Name**:** Send Message

Objective: To verify that a logged in property owner can message any tenant interested in, currently or previously renting one of the users properties and any tenant can message a property owner they are interested in, currently or previously renting from.

Action:

1. After logging in as a registered user, visit the Messaging page.
2. Select a user in the list of available users to message.
3. After redirection to the Message User page, insert some text and attach a photo to the message.
4. Click on the "Send" button.

Assert:

1. The message is no longer in the message input fields.
2. The message is listed on the page as a sent message.

Test Case Name**:** Receive Message

Objective: To verify that a logged in user can receive and view a m essage sent to them using the system.

Action:

1. After logging in as a registered user that has been sent a message, visit the Messaging User page for another user that has messaged the logged in user.

Assert:

1. The message the other user sent to the current user is visible on the page.

Test Case Name**:** Message Received Notification

Objective: To verify that a logged in user receives a notification from the system when a message is sent to them using the system.

Action:

1. After logging in as a registered user that has received a message through the system, open the Notification tray.

Assert:

1. A notification indicating another user sent a message to the current user is visible in the Notification tray.

REQUIREMENT #8: Reporting and Analytics

The system shall generate reports on financials, occupancy rates, and maintenance activities.

Features:

- Exportable reports in PDF or Excel formats.
- Visual dashboards with charts and graphs.

Test Case Name**:** View Financial Report

Objective: To verify that a logged in property owner can view a report including charts and graphs of all the financial records stored by the system relating to the property owner.

Action:

1. After logging in as a registered property owner, visit the Property Owner Reports page.
2. Click the "Financials" navigation link.

Assert:

1. The user is redirected to the View Report page.
2. A report of all of the financial records stored by the system relating to the property owner is visible and includes charts and graphs.

Test Case Name**:** View Occupancy Report

Objective: To verify that a logged in property owner can view a report including charts and graphs of the occupancy records stored by the system for all of the property owner's currently and previously managed properties.

Action:

1. After logging in as a registered property owner, visit the Property Owner Reports page.
2. Click the "Occupancy" navigation link.

Assert:

1. The user is redirected to the View Report page.
2. A report of all of the occupancy records stored by the system for the property owner's currently and previously managed properties is visible and includes charts and graphs.

Test Case Name**:** View Maintenance Report

Objective: To verify that a logged in property owner can view a report including charts and graphs of the maintenance records stored by the system for all of the property owner's currently and previously managed properties.

Action:

1. After logging in as a registered property owner, visit the Property Owner Reports page.
2. Click the "Maintenance" navigation link.

Assert:

1. The user is redirected to the View Report page.
2. A report of all of the maintenance records stored by the system for the property owner's currently and previously managed properties is visible and includes charts and graphs.

Test Case Name**:** Export Report to PDF

Objective: To verify that a logged in property owner can export a report available to them through the Property Owner Reports page to a PDF file.

Action:

1. After logging in as a registered property owner, visit the View Report page for one of the reports available to the property owner.
2. Click the "Export" button.

Assert:

1. A PDF version of the visible report is exported and saved to the user's device.
2. The PDF version contains all of the information available on the View Report page.

REQUIREMENT #9: Notifications

The system shall send automated notifications and alerts.

Features:

- Email and SMS notifications for rent due dates, lease expirations, and maintenance updates.
- Configurable notification settings for users.

Test Case Name: Receive Rent Due Notification

Objective: To verify that tenants receive notifications for upcoming rent due dates according to their notification preferences.

Action:

1. As a tenant with a rent payment due in 7 days, log in to the system.
2. Open the Notification Tray.

Assert:

1. A notification indicating the upcoming rent payment is present in the Notification Tray.
2. The notification includes the due date and the amount due.
3. If email notifications are enabled, an email is received with the same information.
4. If SMS notifications are enabled, an SMS message is received with the same information.

Test Case Name: Receive Lease Expiration Notification

Objective: To verify that tenants receive notifications about upcoming lease expirations according to their notification preferences.

Action:

1. As a tenant whose lease expires in 30 days, log in to the system.
2. Open the Notification Tray.

Assert:

1. A notification about the upcoming lease expiration is present in the Notification Tray.
2. The notification includes the expiration date and options to renew the lease.
3. If email notifications are enabled, an email is received with the same information.
4. If SMS notifications are enabled, an SMS message is received with the same information.

Test Case Name: Receive Maintenance Update Notification

Objective: To verify that tenants receive notifications when the status of their maintenance requests is updated.

Action:

1. As a property owner, update a tenant's maintenance request status (e.g., from "In Progress" to "Completed").
2. As the tenant who submitted the request, log in to the system.
3. Open the Notification Tray.

Assert:

1. A notification about the maintenance request status update is present in the Notification Tray.
2. The notification includes the request ID and new status.
3. If email notifications are enabled, an email is received with the same information.
4. If SMS notifications are enabled, an SMS message is received with the same information.

Test Case Name: Receive Message Notification

Objective: To verify that users receive notifications when they receive a direct message (DM) in the system.

Action:

1. As User A, send a direct message to User B using the in-app messaging feature.
2. As User B, log in to the system.
3. Open the Notification Tray.

Assert:

1. A notification indicating a new message from User A is present in the Notification Tray.
2. If email notifications are enabled, an email is received indicating a new message from User A.
3. If SMS notifications are enabled, an SMS message is received indicating a new message from User A.

Test Case Name**:** Receive Maintenance Offer Notification

Objective: To verify that tenants receive notifications when a property owner offers or schedules maintenance services.

Action:

1. As a property owner, schedule a maintenance service for a property rented by Tenant A.
2. As Tenant A, log in to the system.
3. Open the Notification Tray.

Assert:

1. A notification about the scheduled maintenance is present in the Notification Tray.
2. The notification includes details such as date, time, and type of maintenance.
3. If email notifications are enabled, an email is received with the maintenance details.
4. If SMS notifications are enabled, an SMS message is received with the maintenance details.

Test Case Name**:** Receive Lease Offer Notification

Objective: To verify that prospective tenants receive notifications when they are offered a lease agreement.

Action:

1. As a property owner, create a lease agreement and assign it to a prospective tenant (Tenant B).
2. As Tenant B, log in to the system.
3. Open the Notification Tray.

Assert:

1. A notification about the new lease offer is present in the Notification Tray.
2. The notification includes details about the property and lease terms.
3. If email notifications are enabled, an email is received with the lease offer details.
4. If SMS notifications are enabled, an SMS message is received with the lease offer details.

Test Case Name**:** Configure Notification Settings

Objective: To verify that users can configure their notification preferences.

Action:

1. As a logged-in user, navigate to Settings.
2. Go to the Notification Settings section.
3. Change preferences (e.g., enable email notifications, disable SMS notifications).
4. Click on Save Settings.

Assert:

1. Notification preferences are saved successfully.
2. A confirmation message is displayed indicating settings have been updated.
3. Future notifications adhere to the updated preferences.

REQUIREMENT #10: Document Management

The system shall manage documents related to properties and tenants.

Features:

- Upload and store documents like IDs, contracts, and invoices.
- Secure access and sharing options.

Test Case Name: Upload Document

Objective: To verify that users can upload documents to the system.

Action:

1. As a logged-in user, navigate to the Documents section.
2. Click on Upload Document.
3. Select a valid document file (e.g., PDF, DOCX).
4. Enter a description and select a document type (e.g., ID, Contract).
5. Click on Upload.

Assert:

1. The document is successfully uploaded and listed in the user's document list.
2. A success message is displayed.
3. Document metadata (name, type, description) is correctly displayed.

Test Case Name: View Uploaded Document

Objective: To verify that users can view documents they have uploaded.

Action:

1. In the Documents section, locate the uploaded document.
2. Click on the document name or View button.

Assert:

1. The document opens within the application or in a new browser tab.
2. The content of the document is displayed accurately.

Test Case Name: Delete Document

Objective: To verify that users can delete documents they have uploaded.

Action:

1. In the Documents section, locate the document.
2. Click on the Delete button.
3. Confirm the deletion when prompted.

Assert:

1. The document is removed from the document list.
2. A confirmation message is displayed.
3. The document is no longer accessible.

Test Case Name: Share Document with Another User

Objective: To verify that users can securely share documents with other users.

Action:

1. In the Documents section, locate the document.

2. Click on the Share button.
3. Select a user to share with (e.g., tenant or property owner).
4. Set access permissions (e.g., view-only, download allowed).
5. Click on Share Document.

Assert:

1. The selected user receives a notification about the shared document.
2. The shared document appears in the recipient's Shared Documents section.
3. Access permissions are enforced correctly.

Test Case Name**:** Edit Document Sharing Settings

Objective: To verify that users can edit the sharing settings of a document, including access permissions, expiration date, and custom URL.

Action:

1. As a logged-in user, navigate to the Documents section.
2. Locate a document that has been shared with another user.
3. Click on the Edit Sharing Settings button.
4. Modify the sharing settings:
5. Change access permissions (e.g., from view-only to edit).
6. Set an expiration date for access.
7. Customize the sharing link (custom URL).
8. Click on Save Changes.

Assert:

1. The sharing settings are updated successfully.
2. A confirmation message is displayed indicating the settings have been updated.
3. The recipient's access to the document reflects the updated permissions.
4. If an expiration date is set, access is revoked after the specified date.
5. The custom URL directs to the document with updated settings.

Test Case Name**:** Verify Access Permissions for Documents

Objective: To ensure that document access permissions are enforced.

Action:

1. User A shares a document with User B with view-only permissions.
2. User B attempts to download or delete the document.

Assert:

1. User B can view the document but cannot download or delete it.
2. Unauthorized actions prompt an error message.