Dhirubhai Ambani University Technology

Formerly DA-IICT

Group 15

IT314 Software Engineering
Project Title: Real Estate Listing Website

Group Members :

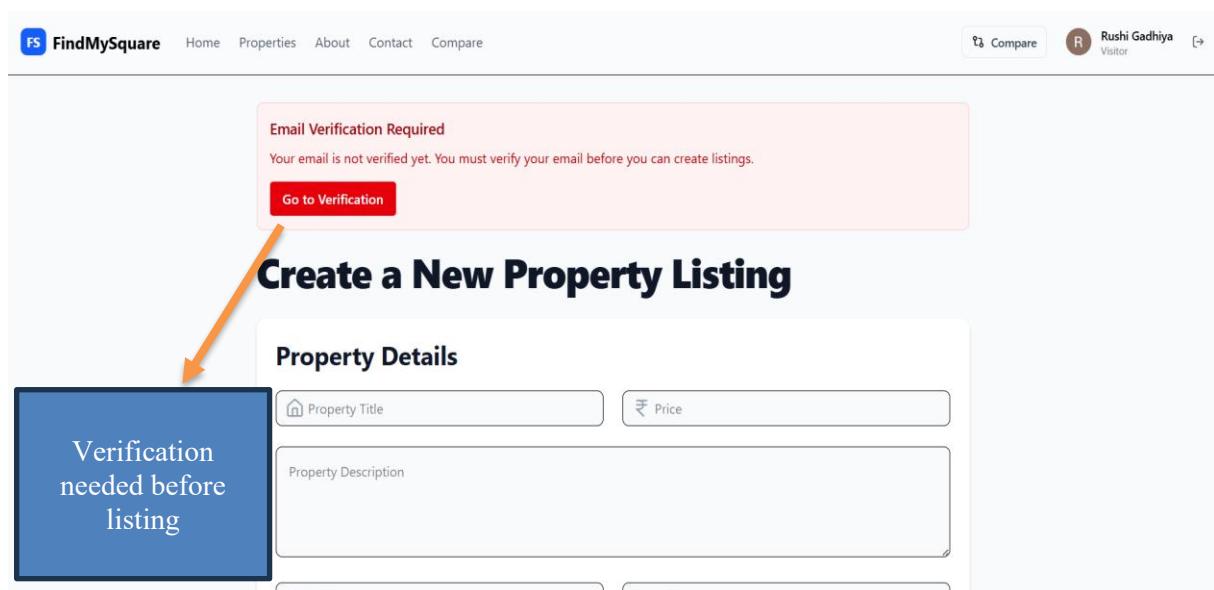| SID	Name
202301190	VALA HARDIK VINUBHAI
202301138	SRI SAI MADHAVA TEJA YANDURI
202301193	GADHVI NANDINI INDRABHAI
202301196	VYAS GURU PRAVINCHANDRA
202301134	SNEH SHAH
202301156	RUSHI M GADHIYA
202301142	KHUSHKUMAR PATEL
202301162	NINAMA MITWA DINESHKUMAR
202301178	PATEL PRINCE JITUBHAI
202301147	PATEL JEINILBHAI VIJESHBHAI

Prof- Dr. Saurabh Tiwari
Group Mentor- Hetul

Sprint 4: Property Management (Lister/User)

In this sprint, we implemented all the function from an lister side which is also an user/actor in our project ,only verified listers are allowed to add and manage their Listing Dashboard. We implemented the following functionalities:

Add New Property

- Functionality :Only Verified Users can post and add new property and listing on website .Note, that an ordinary user only also list an website, the only situation will be the verification needed.

- Implementation: Developed form like card to get and post desired property details like Title, Price, Location, Amenities .Also ,Lister can upload images while filling this form.

- Image Handling: Integrated Cloudinary,an secures image storage platform. Users can upload multiple property images, which are processed and stored in the cloud, returning a URL to our database.

- Data Integrity: Implemented MongoDB Transactions to ensure that a Property and its corresponding Listing are created simultaneously.

Bedrooms

Bathrooms

Size (sq ft)

Balconies

DD/MM/YYYY

Select Property Type

Location

Street

City

Locality

State

Zip Code

Country

Property Details to be uploaded by lister including drop and drag image[Property Details Form]

Amenities

☐ Parking   ☐ Gym   ☐ SwimmingPool   ☐ Wifi

☐ Security   ☐ PowerBackup   ☐ Garden   ☐ Lift
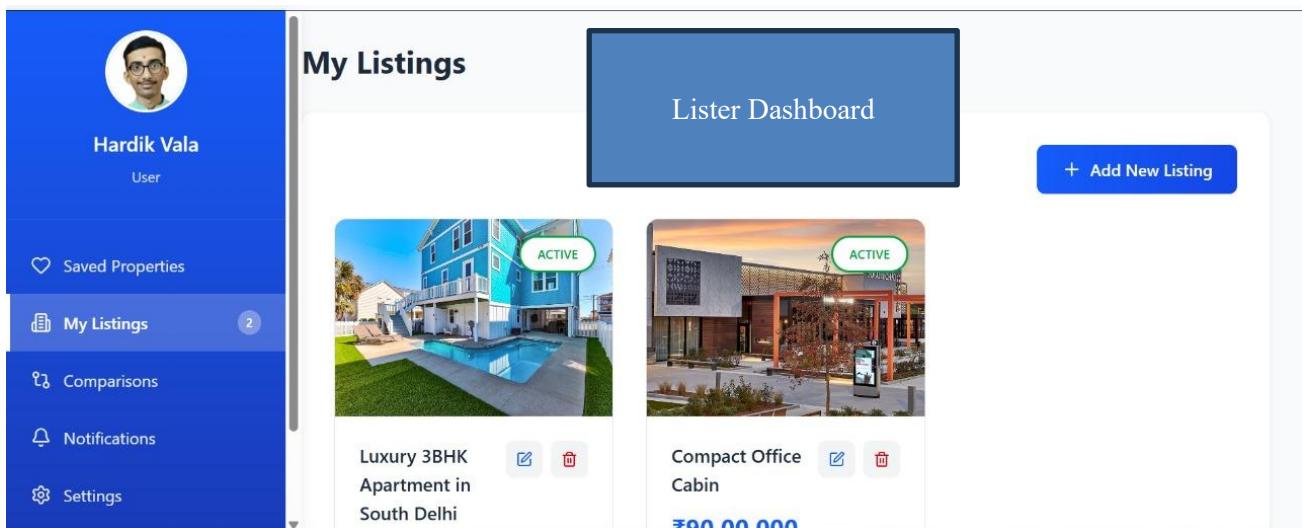
☐ Clubhouse   ☐ PlayArea   ☐ Furnished

Property Images

Upload files or drag and drop
PNG, JPG, GIF up to 10MB

Final Add property Button after filling the form
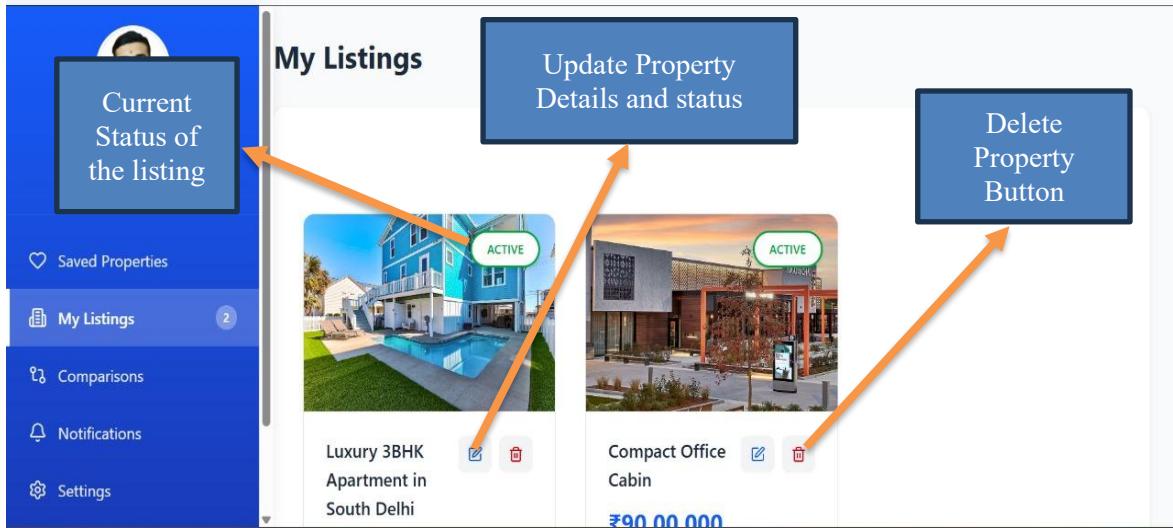
⬆ Add Property

Lister Dashboard (My Properties)

- Functionality: Listers will have access to view a personalized dashboard which will show the property he has listed.

- Implementation: Created a secure API endpoint (/my-properties) that fetches only the listings linked to the logged-in user's ID.

- Status Visibility: The dashboard will display many things like the details he added and also the current status of each property (e.g., "Pending Verification," "Active," "Hidden") so listers can be able to track approval progress.

Update Property Details

- Functionality: Listers will be able to update and edit the details of their existing listings from the lister dashboard by selecting the the option in the propery card of their desired listing ,he will be able to change every detail he initially uploaded including the image.Again only for the verifies user only.

- Implemenation:Added an update button at property card of Lister dashboard of every listing made by him.

- Re-Verification : Added a check that, whenever crucial details related to Price or Location were changed, the listing status automatically moved to "Pending" for re-verification by the admin.As, it can be used as loophole to verify listing with dummy details and once verified, then edit it so that fraud properties can also be seen as verified on website.

Hide/Show Listing

- Functionality: Listers will be able to temporarily hide any desired property from lister dashboard without actually deleting it from their listing Profile.

- Implementation: Added an switch in the dashboard inside update button,an update status button was added the edit which will hide and unhide the selected desired property.

- Logic: This updates the listing's status to "Hidden" or "Active" instantly, removing it from or adding it back to the public search results via the updatePropertyStatus controller.

Delete Property

- Functionality: Listers will be also able to remove any desired property from the their listing dashboard permanently.Note that once deleted every single details uploaded including the image will be permanently removed from main database and cloudinary database permanently.

- Cleanup: The system will perform a "Cascading Delete" of the Property data, the Listing record, and all of its associated Reviews within the database. It will also fire a cleanup function that will remove the associated images from Cloudinary to free up storage space.

Frontend & Integration
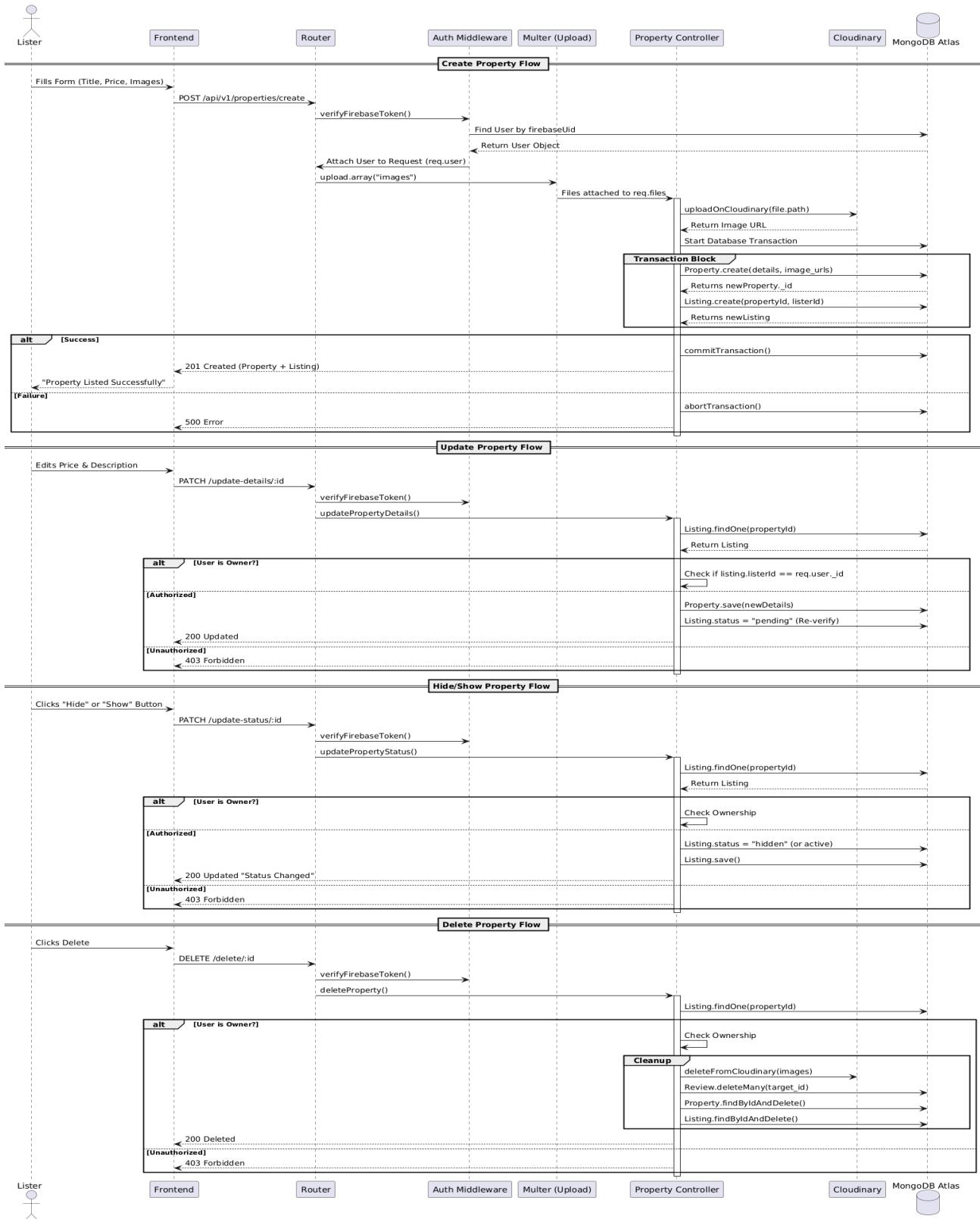
Frontend Components (React)

- Add Property Form: Designed a form with React Hook Form and client-side validation; for example, it ensures that the price is a positive number.

- Uploader Images: Developing a drag-and-drop image uploader that will preview selected images before submission.

- Dashboard UI: Created and implemented a card-like form for the "My Properties" page, with "Edit,"/"update", "Delete," and "Hide" action buttons on each of the listed property detail card on website.API Integration

- Secured Requests: Establish Axios interceptors to attach the Firebase Auth Token to the headers of every request. In return, it allows the backend middleware - verifyFirebaseToken- to securely authenticate the user.• Error Handling: Added global error handling; shows backend messages - for instance, "Upload Failed", "Unauthorized" - directly to the user as toast notifications.

- State Management: Utilized React state to have the Dashboard view update instantly when a user decides to hide or delete a property, rather than having to reload the page.

Overview of Sprint

- What this Sprint means is that we've successfully built the core Supply-Side of the platform.. The verified listers fully control their inventory through a secure Dashboard. We implemented complex backend logic like Cloudinary image uploads, Atomic Transactions to maintain data safety, and Role-Based Access Control to ensure against unauthorized modifications. The Frontend was successfully integrated with these backend Part,APIs which will be providing a seamless experience wherein listers can manage their assets and track their verification in real time.So,at last our lister-specific functionalities were implemented(Frontend+Backend) including integrating in sprint 4.

Sequence Diagram

Lister | **Frontend** | **Router** | **Auth Middleware** | **Multer (Upload)** | **Property Controller** | **Cloudinary** | **MongoDB Atlas**

Create Property Flow

Lister → Frontend: Fills Form (Title, Price, Images)
Frontend → Router: POST /api/v1/properties/create
Router → Auth Middleware: verifyFirebaseToken()
Auth Middleware → MongoDB Atlas: Find User by firebaseUid
MongoDB Atlas → Auth Middleware: Return User Object
Auth Middleware → Router: Attach User to Request (req.user)
Router → Multer (Upload): upload.array("images")
Multer (Upload) → Property Controller: Files attached to req.files
Property Controller → Cloudinary: uploadOnCloudinary(file.path)
Cloudinary → Property Controller: Return Image URL
Property Controller: Start Database Transaction

Transaction Block
Property Controller → MongoDB Atlas: Property.create(details, image_urls)
MongoDB Atlas → Property Controller: Returns newProperty._id
Property Controller → MongoDB Atlas: Listing.create(propertyId, listerId)
MongoDB Atlas → Property Controller: Returns newListing

alt [Success]
Property Controller → MongoDB Atlas: commitTransaction()
Property Controller → Frontend: 201 Created (Property + Listing)
Frontend → Lister: "Property Listed Successfully"

[Failure]
Property Controller → MongoDB Atlas: abortTransaction()
Property Controller → Frontend: 500 Error

Update Property Flow

Lister → Frontend: Edits Price & Description
Frontend → Router: PATCH /update-details/:id
Router → Auth Middleware: verifyFirebaseToken()
Router → Property Controller: updatePropertyDetails()
Property Controller → MongoDB Atlas: Listing.findOne(propertyId)
MongoDB Atlas → Property Controller: Return Listing

alt [User is Owner?]
Property Controller: Check if listing.listerId == req.user._id

[Authorized]
Property Controller → MongoDB Atlas: Property.save(newDetails)
Property Controller → MongoDB Atlas: Listing.status = "pending" (Re-verify)
Property Controller → Frontend: 200 Updated

[Unauthorized]
Property Controller → Frontend: 403 Forbidden

Hide/Show Property Flow

Lister → Frontend: Clicks "Hide" or "Show" Button
Frontend → Router: PATCH /update-status/:id
Router → Auth Middleware: verifyFirebaseToken()
Router → Property Controller: updatePropertyStatus()
Property Controller → MongoDB Atlas: Listing.findOne(propertyId)
MongoDB Atlas → Property Controller: Return Listing

alt [User is Owner?]
Property Controller: Check Ownership

[Authorized]
Property Controller → MongoDB Atlas: Listing.status = "hidden" (or active)
Property Controller → MongoDB Atlas: Listing.save()
Property Controller → Frontend: 200 Updated "Status Changed"

[Unauthorized]
Property Controller → Frontend: 403 Forbidden

Delete Property Flow

Lister → Frontend: Clicks Delete
Frontend → Router: DELETE /delete/:id
Router → Auth Middleware: verifyFirebaseToken()
Router → Property Controller: deleteProperty()
Property Controller → MongoDB Atlas: Listing.findOne(propertyId)

alt [User is Owner?]
Property Controller: Check Ownership

Cleanup
Property Controller → Cloudinary: deleteFromCloudinary(images)
Property Controller → MongoDB Atlas: Review.deleteMany(target_id)
Property Controller → MongoDB Atlas: Property.findByIdAndDelete()
Property Controller → MongoDB Atlas: Listing.findByIdAndDelete()
Property Controller → Frontend: 200 Deleted

[Unauthorized]
Property Controller → Frontend: 403 Forbidden

Class Diagram

User

o firebaseUid : String
o email : String
o name : String
o role : String
o createdAt : Date

Lister

o verificationStatus : String

—————Methods (Controller Logic)—————

o createProperty(data, images) : JSON
o updatePropertyDetails(propertyId, data) : JSON
o updatePropertyStatus(propertyId, status) : JSON
o deleteProperty(propertyId) : JSON
o getMyProperties() : Array

1 ▼ manages
0..*

Listing

o _id : ObjectId
o propertyId : ObjectId
o listerId : ObjectId
o status : String
o viewsCount : Number
o rejectionReason : String

Status Values:
- "pending" (Default/After Update)
- "verified" (Publicly Visible)
- "hidden" (Lister hid it)
- "rejected" (Admin rejected)

1 ▼ describes
1

Property

o _id : ObjectId
o title : String
o description : String
o price : Number
o location : Object
o images : String[]
o amenities : Object
o priceHistory : Array