Design Document

for

DealSimplified

Version 1.0

Group 14 **Group Name: Kasukabe Defence Group**

Esra Fatima	**220391**	**esra22@iitk.ac.in**
Kanika Chaturvedi	**220497**	**kanikac22@iitk.ac.in**
Krishiv Geriani	**220545**	**krishivg22@iitk.ac.in**
Manavjeet Singh	**220616**	**manavjeetw22@iitk.ac.in**
Rachit Choudhary	**220845**	**rachitc22@iitk.ac.in**
Riddhima Vijayvargiya	**220883**	**riddhima22@iitk.ac.in**
Ritul	**220896**	**ritul22@iitk.ac.in**
Riya Agarwal	**220899**	**ariya22@iitk.ac.in**
Vishal Singh	**221207**	**vishals22@iitk.ac.in**
Vishap Raj	**221208**	**vishapraj22@iitk.ac.in**

COURSE: CS253

Mentor TA: Vikrant Chouhan

Date: 7-02-2025

CONTENTS II

REVISIONS II

1 CONTEXT DESIGN 1

1.1 Context model 1
1.2 human Interface Design 1

2 ARCHITECTURE DESIGN 2

3 OBJECT-ORIENTED DESIGN 3

3.1 Use case diagram 3
3.2 Class diagram 3
3.3 sequence diagram 3
3.4 State diagram 3

4 PROJECT PLAN 4

5 OTHER DETAILS 5

APPENDIX A - GROUP LOG 6

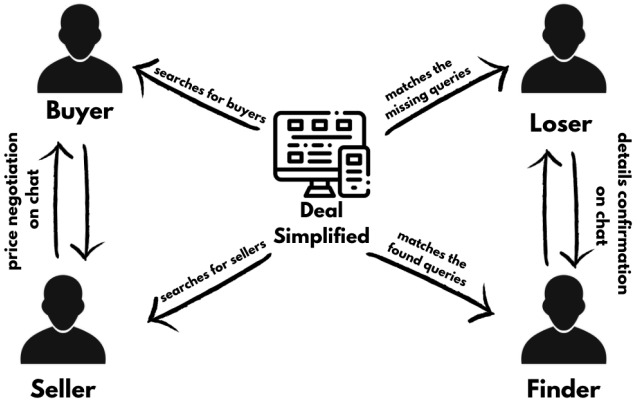
Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Esra Fatima Kanika Chaturvedi Krishiv Geriani Manavjeet Singh Rachit Choudhary Riddhima Vijayvargiya Ritul Riya Agarwal Vishal Singh Vishap Raj	First draft	07/02/25

1. Context Design

1.1. Context Model

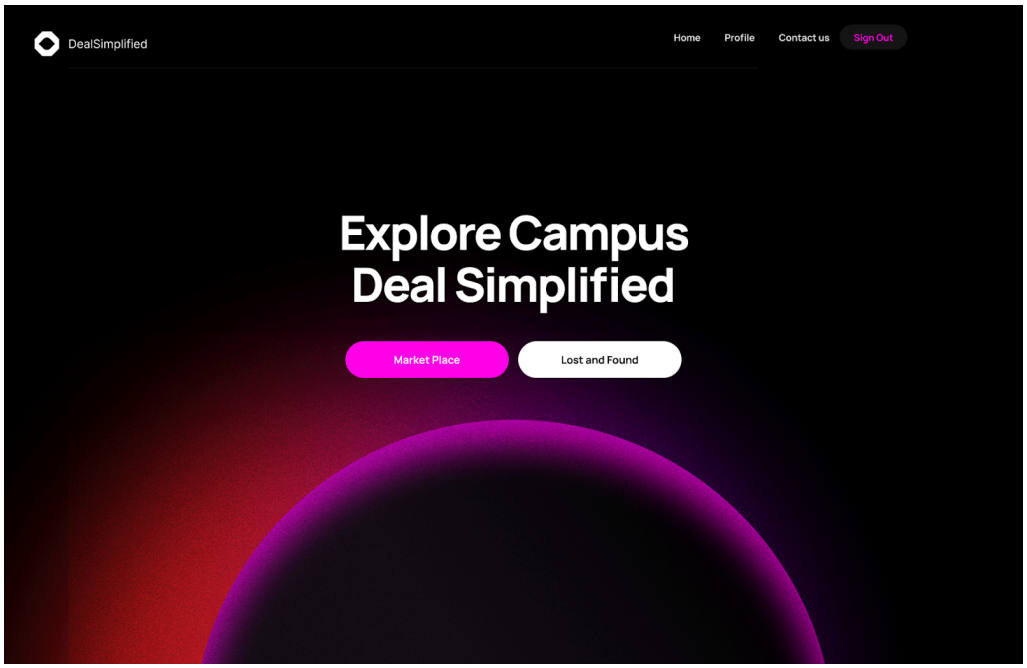
The context model shows the various aspects of the system which interact with each other. The entities are



seller, buyer, loser, finder and the management system for both types of user. The database stores all the data of the system. There will be an OTP API which sends unique codes for login and registration purposes.

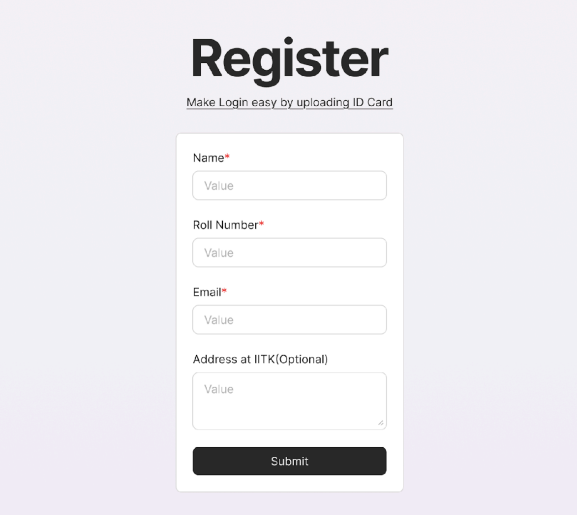
1.2. Human Interface Design

Landing page
user_type(market_place, lost_found) *User selects whether they are using the marketplace or lost and found portal*



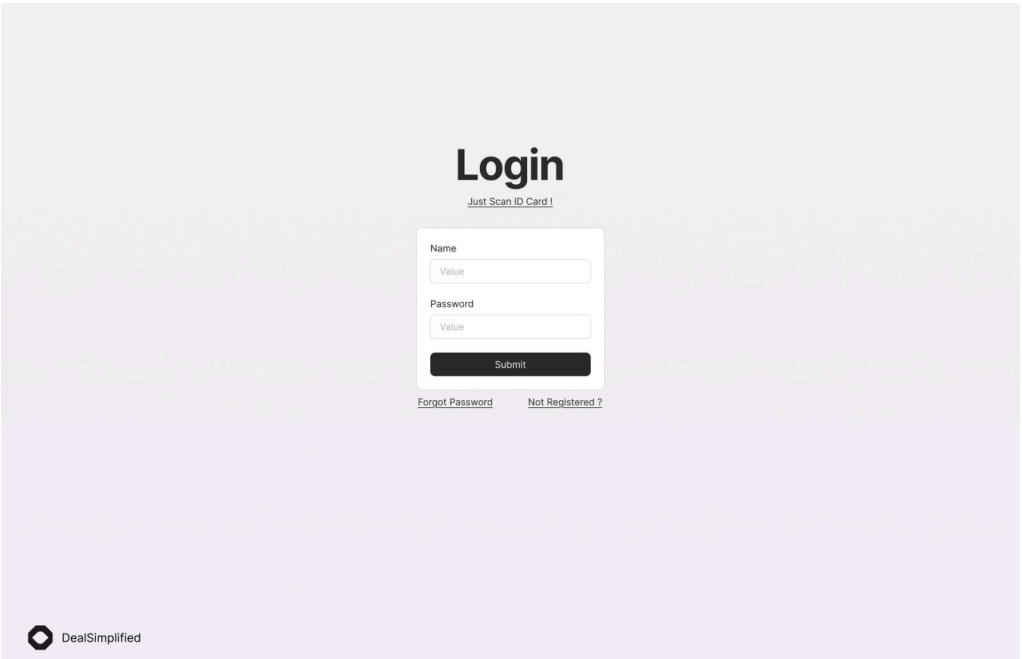
Register page

register_user(name, roll_no, email_id, address)
For all types of campus residents to register on the platform



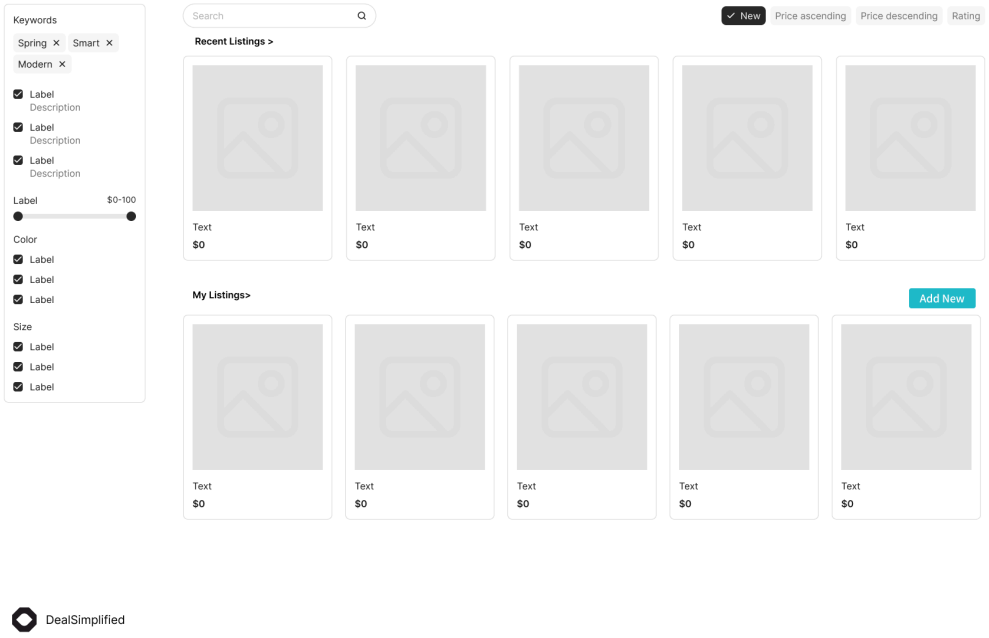
Login page

login_user(user_id,password)
For all types of campus residents to login on the platform
not_registered()
takes the user to the register page
forgot_password(user_id,roll_no, email_id)
Users can reset their password



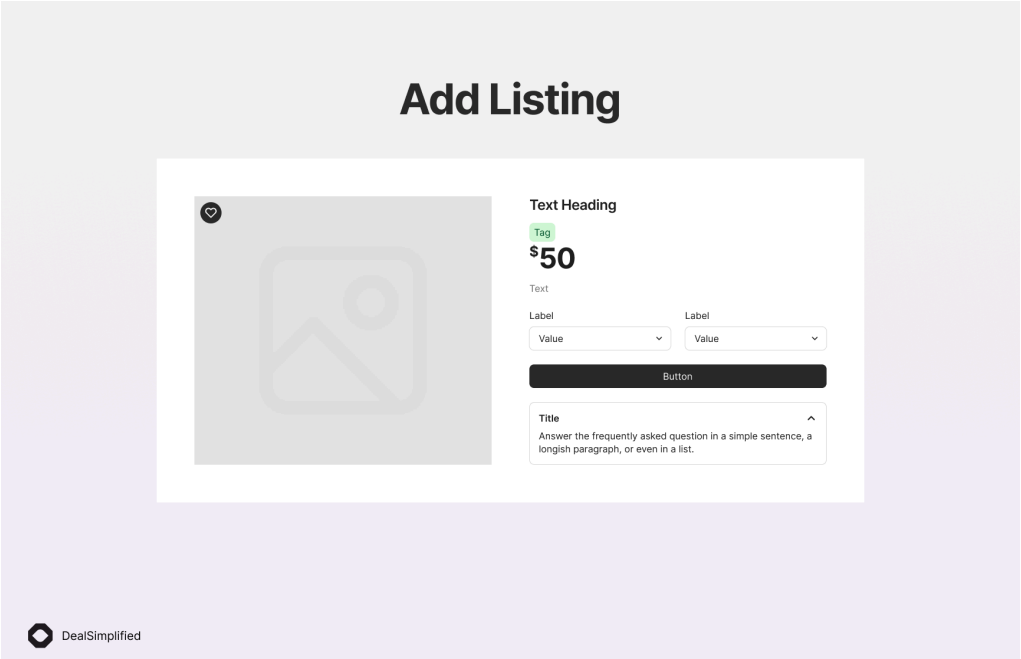
Marketplace listing page

search_item(item_name, item_category item_age)
Users can search for desired items by name, category or age.
user_wishlist(user_id, item_name, item_category)
Users can add items of interest to be notified whenever it appears on listing.
view_user_profile(user_id)
Users can view their profiles and their past history on portal.



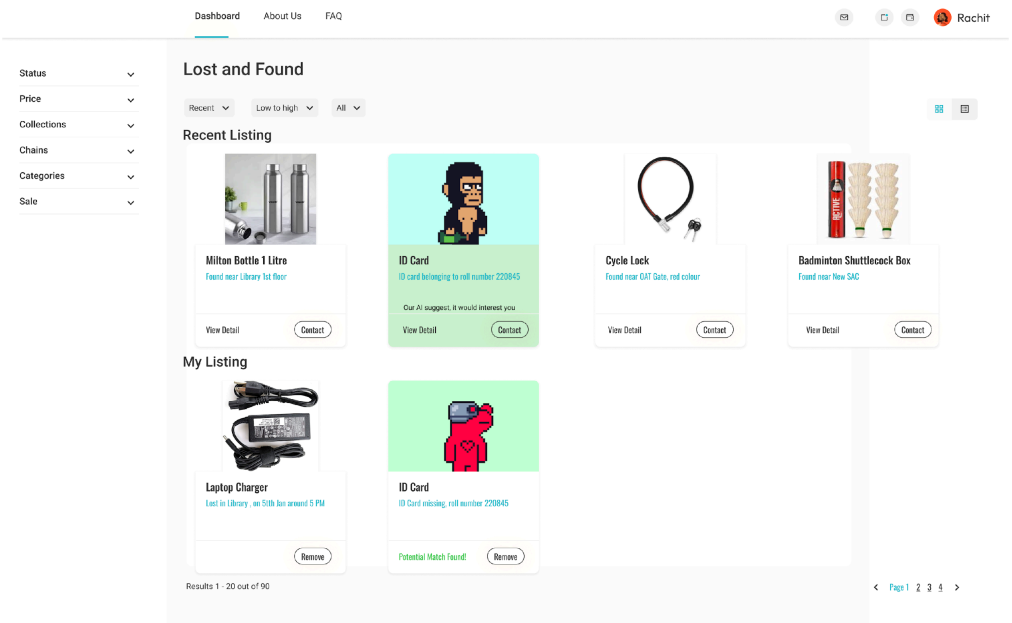
Seller listing page

add_listing(item_name, item_category, item_photo, item_price)
Users can fill in information about the item along with a photo to be listed in the marketplace.

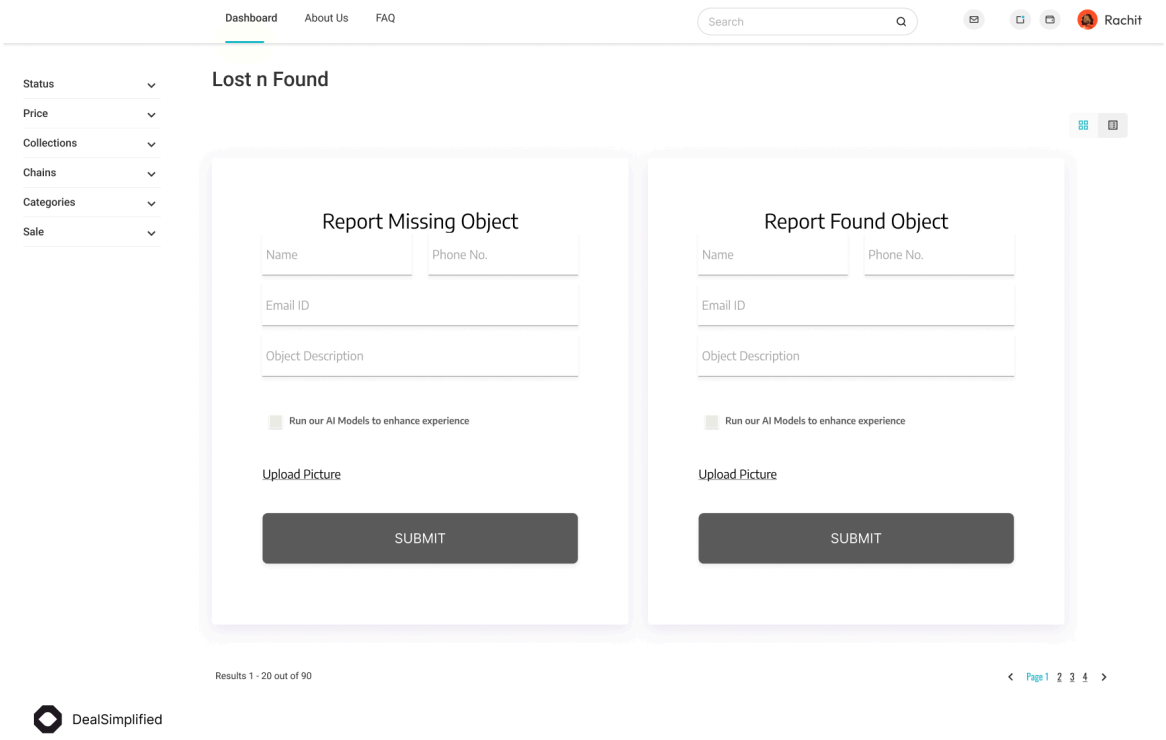


Found items listing page

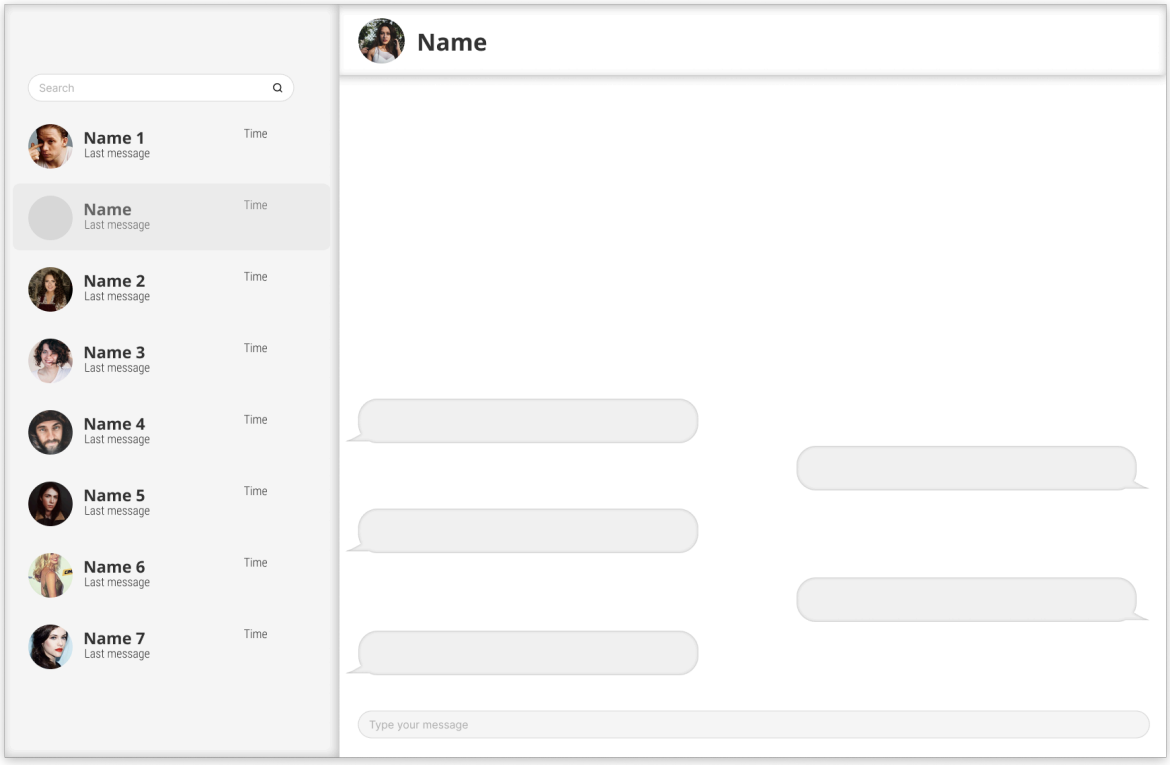
search_item(item_name, item_category, item_colour, lost_days)
Users can search for the lost items by name, category, colour, number of days since item lost.
add_query(item_name, item_category, item_colour, lost_days,)
Users can add a query for the lost items by name, category, colour, number of days since item lost.



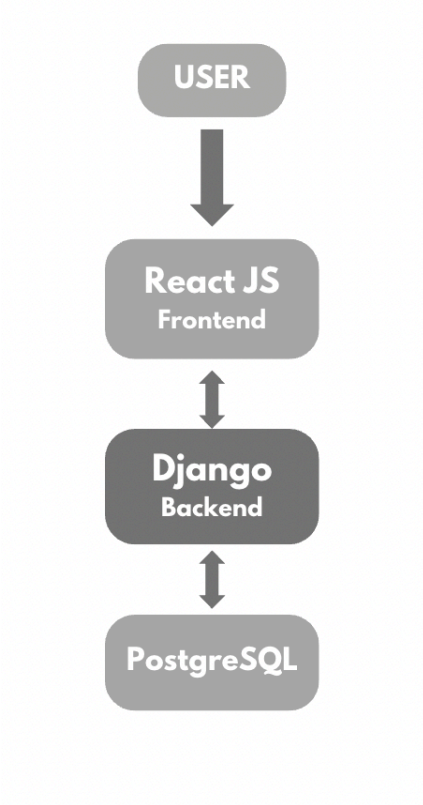
Lost and found query page
add_missing_query(item_name, location, phone_number, item_category, item_colour, lost_days,) *Users can add a query for the lost items by name, location, category, colour, number of days since item lost.* ***add_found_query(item_name, location, phone_number, item_category, item_colour, found_days,)*** *Users can add a query for the found items by name, location, category, colour, number of days since item found.*



Chat page
start_chat(item_id, user_id,) *Users can send a message to any other user by entering the chosen item id or user id.*



2. Architecture Design



This architecture should meet the non-functional requirements as described in the SRS document as follows:

• **Scalability:** The architecture is scalable, meaning it can handle an increased number of users and data. The backend and database can be scaled horizontally by adding more instances, and the frontend can be scaled vertically by increasing the resources of the server. To implement horizontal scaling, we can increase the number of servers which are managing database handling that run our application and distribute the workload among them. This can handle more requests and tolerate failures or outages of individual servers. Apart from this, if need be, we can also use cloud-based services for managing database requests while maintaining privacy of the users.

• **Performance:** The architecture is designed for high performance while being simple to understand and implement, with a separation of concerns between frontend and backend, and the use of APIs to communicate between them, along with a powerful PostGre SQL database for data retrieval and management. This allows for optimized data access and

processing.

• **Safety and Security:** Security is of prime importance in our software, since we are handling sensitive private information of the users such as medical reports and diagnostics.
For this reason, our architecture implements security measures such as the protection of sensitive data and enforcing access control. This is done by devising an authentication system which verifies the phone number and license (in case of doctor) of the user and prevents unauthorised access, all of which is implemented using the backend framework.

• **Interoperability:** The given architecture uses React.js and Django for the frontend and backend respectively, both of which are extremely popular and reliable frameworks for software development, and hence compatible across almost all major browsers on the internet. This allows for smooth functioning of the software on all servers and for all users, irrespective of the devices or browsers used.

• **Reliability:** Through rigorous testing and beta testing, as well as a robust design achieved by using enhanced and flexible code quality, we plan to provide continuous integration and deployment within our architecture so that any bugs or glitches are prevented from occurring and the software works smoothly. We also plan to cover all details through thorough and up-to-date documentation, which will make it easy to use and convenient for users, while being accessible most of the time and dynamically updated as well.

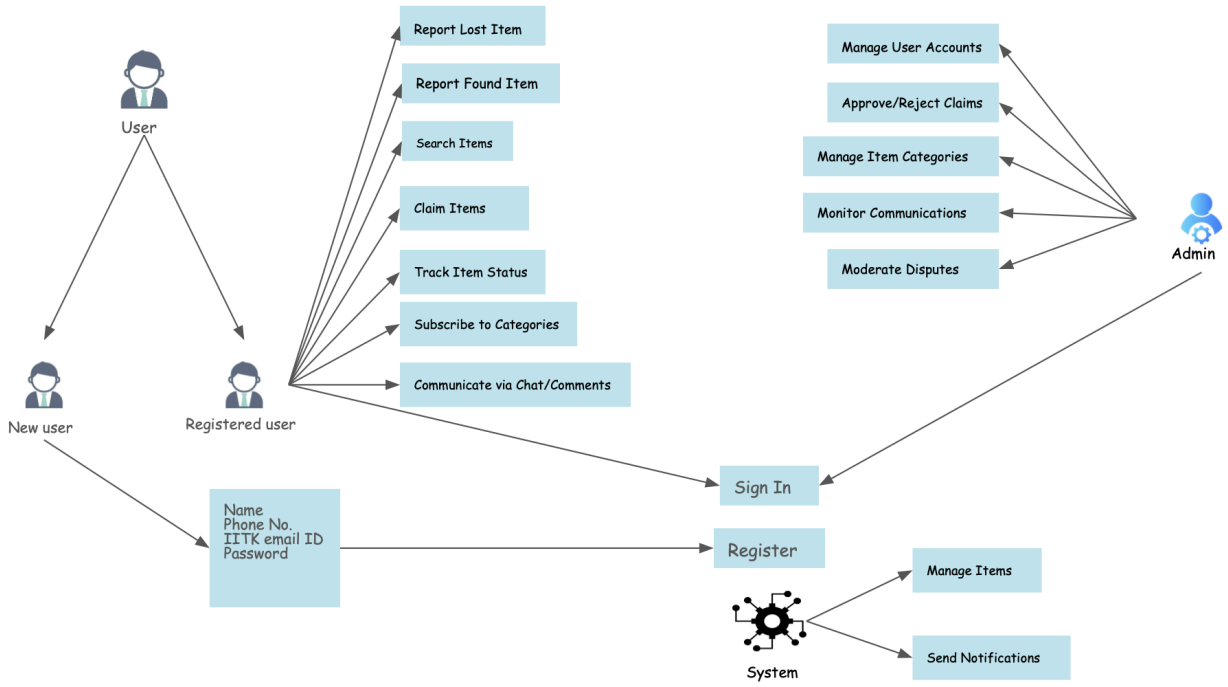
3.Object Oriented Design

3.1 Use Case Diagrams

Lost and Found System

The following diagram illustrates the various use cases involved in the system:

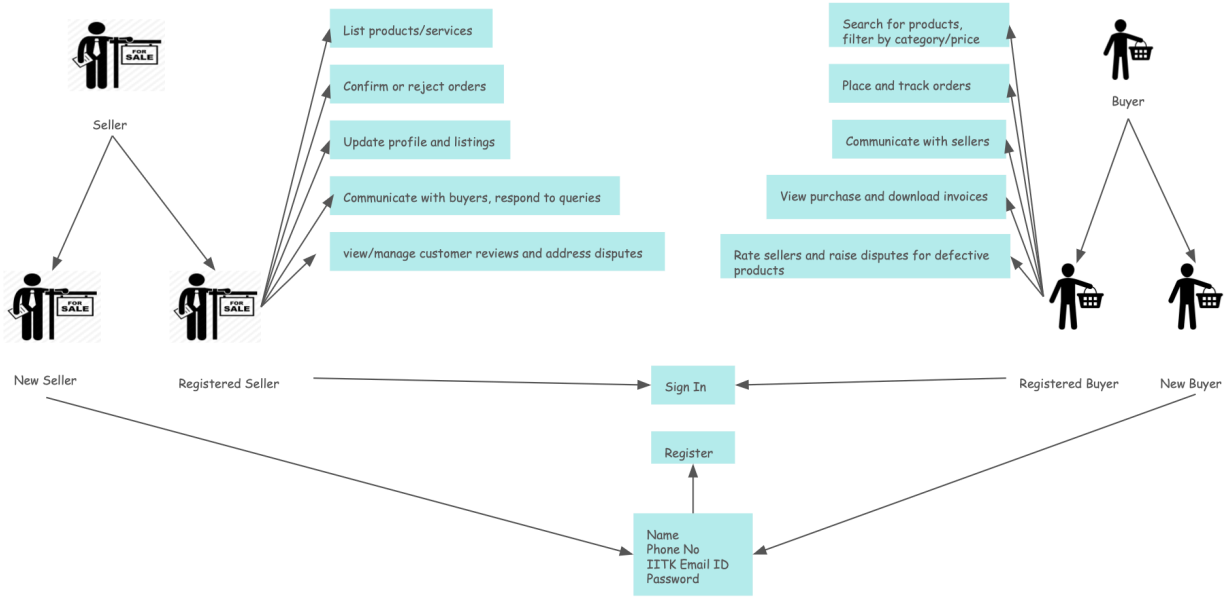
- **Students register** using their official IITK email ID.
- **Registered students and admins sign in** using their credentials with role-based access control.
- **Students report lost items** by submitting details like description, category, images, location, and date.
- **Students report found items** with similar details for easy identification.
- **Students search for items** using keywords, categories (e.g., electronics, books, ID cards), and locations.
- **Students filter search results** by date, category, location, or status (lost/found/claimed).
- **Students claim items** by submitting verification details such as proof of ownership.
- **System notifies reporters** when someone initiates a claim for their reported item.
- **Admins approve or reject claims** if disputes arise, ensuring proper validation.
- **Admins manage user accounts**, including activating or deactivating accounts when necessary.
- **Admins manage item categories** for easier reporting and searching (e.g., electronics, personal belongings).
- **Students track the status of reported items** through stages like Reported, Matched, and Claimed.
- **System automatically recommends potential matches** for lost and found items using NLP for text descriptions and computer vision for image analysis.
- **System sends notifications** to users about potential matches, claims initiated, or status updates.
- **Users subscribe to specific categories or keywords**, receiving notifications when new items are reported.
- **Students and claimants communicate** via commenting or a chat feature to facilitate easier verification.
- **Admins monitor and moderate communication** to resolve disputes effectively.



Marketplace System

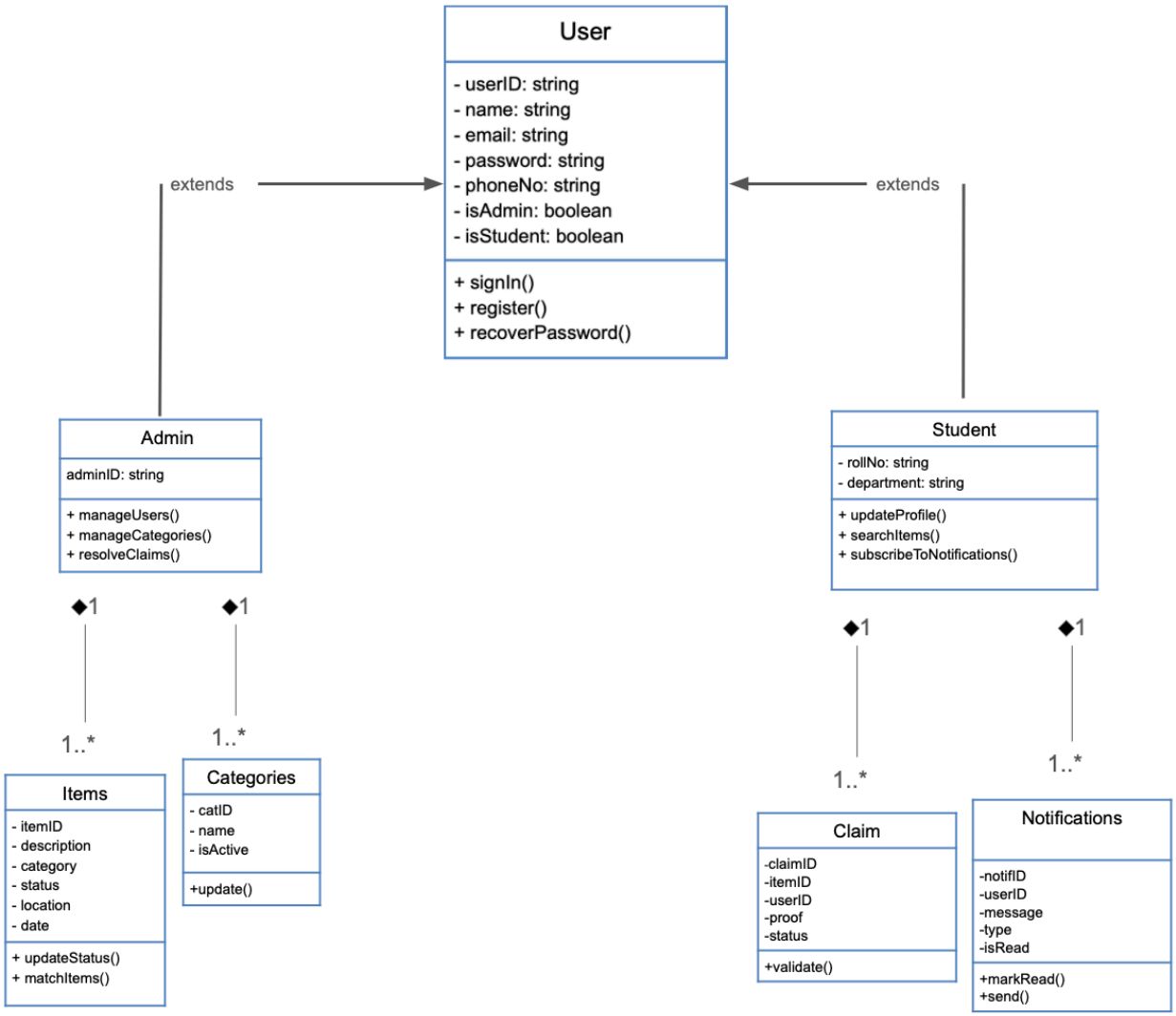
The following diagram illustrates the various use cases involved in the system.

- New users register as buyers or sellers, with sellers providing verification details.
- Registered buyers and sellers sign in using their credentials.
- Sellers list products/services with details like name, category, price, and images.
- Buyers search for products using filters like category, price, and seller rating.
- Buyers place orders by adding items to their cart and selecting payment options.
- Sellers manage orders by confirming, rejecting, or updating order status.
- Buyers and sellers communicate via a built-in messaging system.
- Sellers update their profile, business details, and product listings.
- Buyers view their purchase history and download invoices.
- Buyers raise disputes for defective products, and sellers address them.
- Admin manually verifies seller business credentials before allowing sales.
- Sellers manage and respond to customer reviews and ratings.
- Buyers rate and review sellers after a purchase.

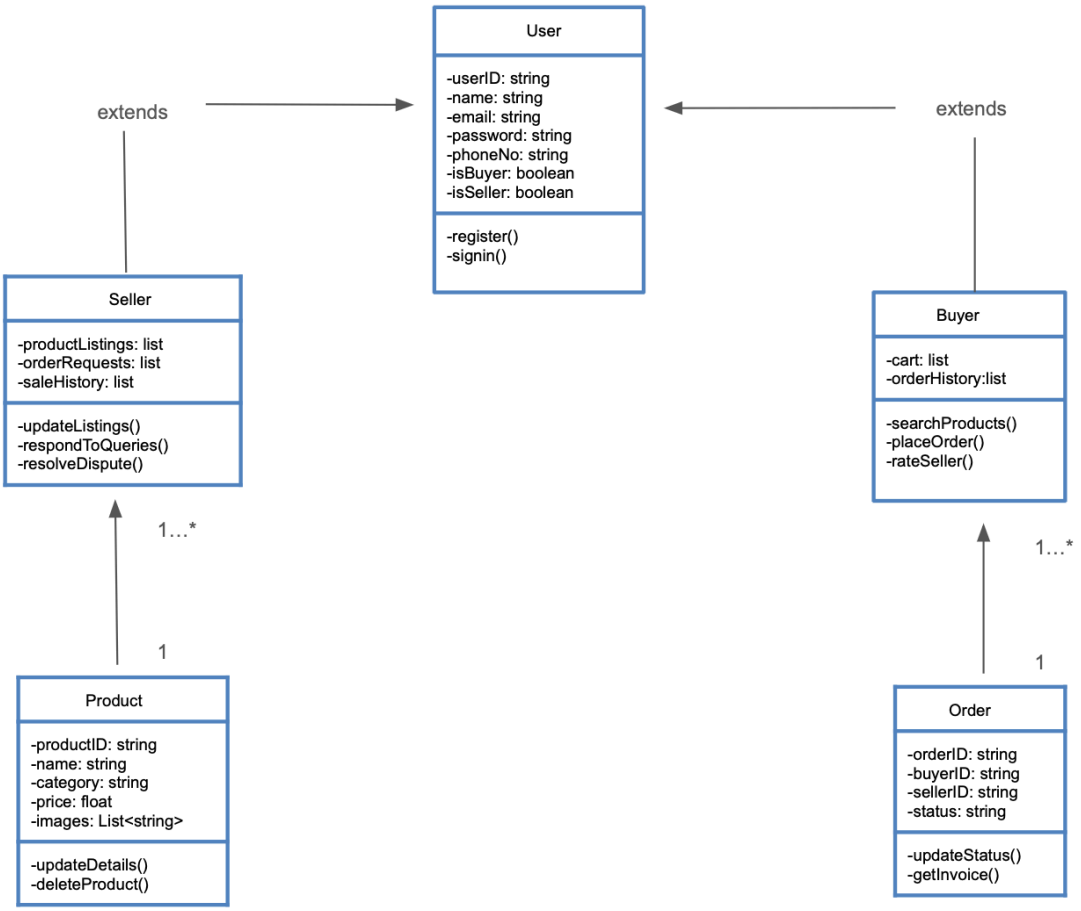

Seller
List products/services
Confirm or reject orders
Update profile and listings
Communicate with buyers, respond to queries
view/manage customer reviews and address disputes
Search for products, filter by category/price
Place and track orders
Communicate with sellers
View purchase and download invoices
Rate sellers and raise disputes for defective products
Buyer
New Seller
Registered Seller
Sign In
Registered Buyer
New Buyer
Register
Name
Phone No
IITK Email ID
Password

3.2 Class Diagrams

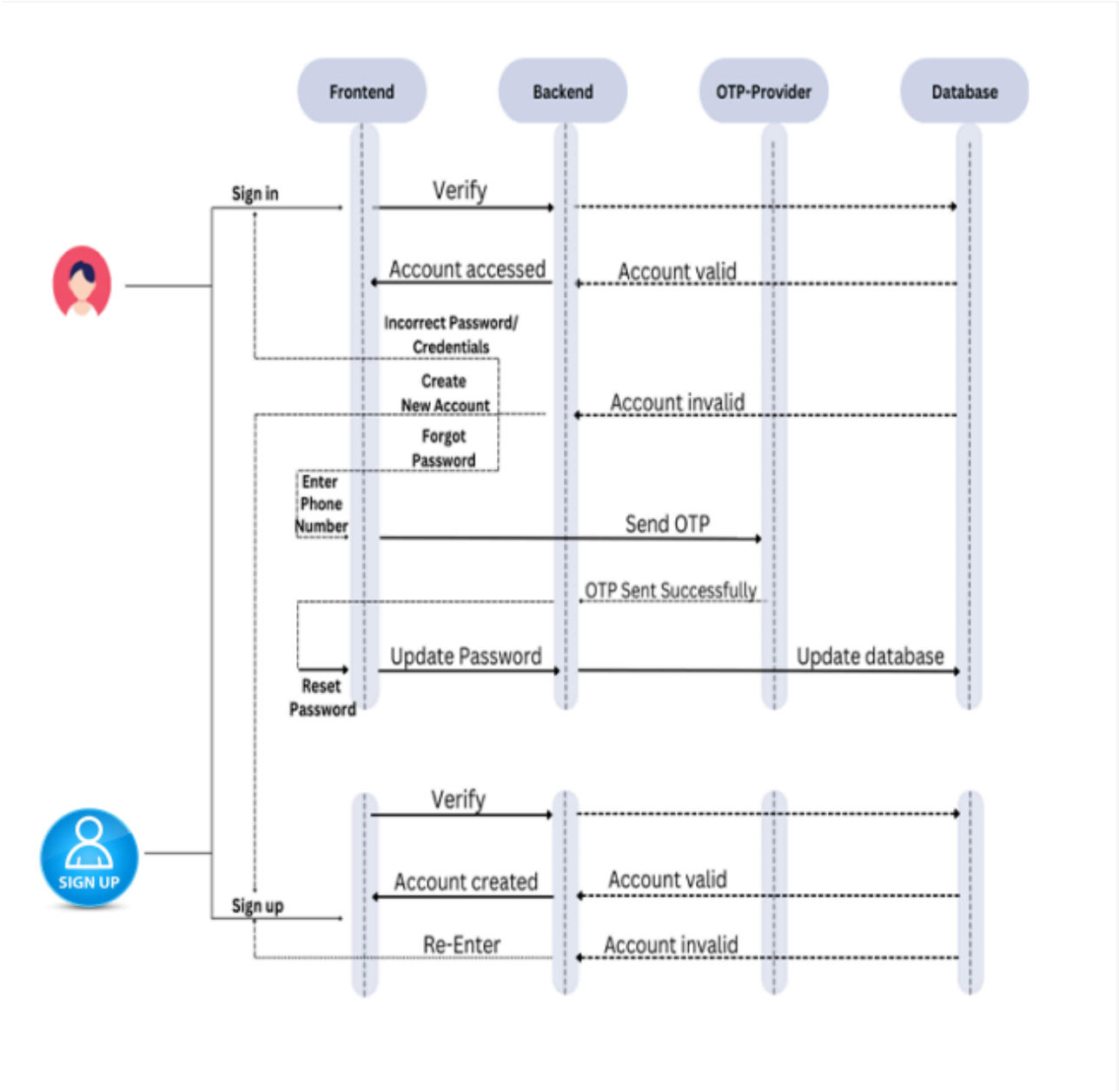
Lost and Found System

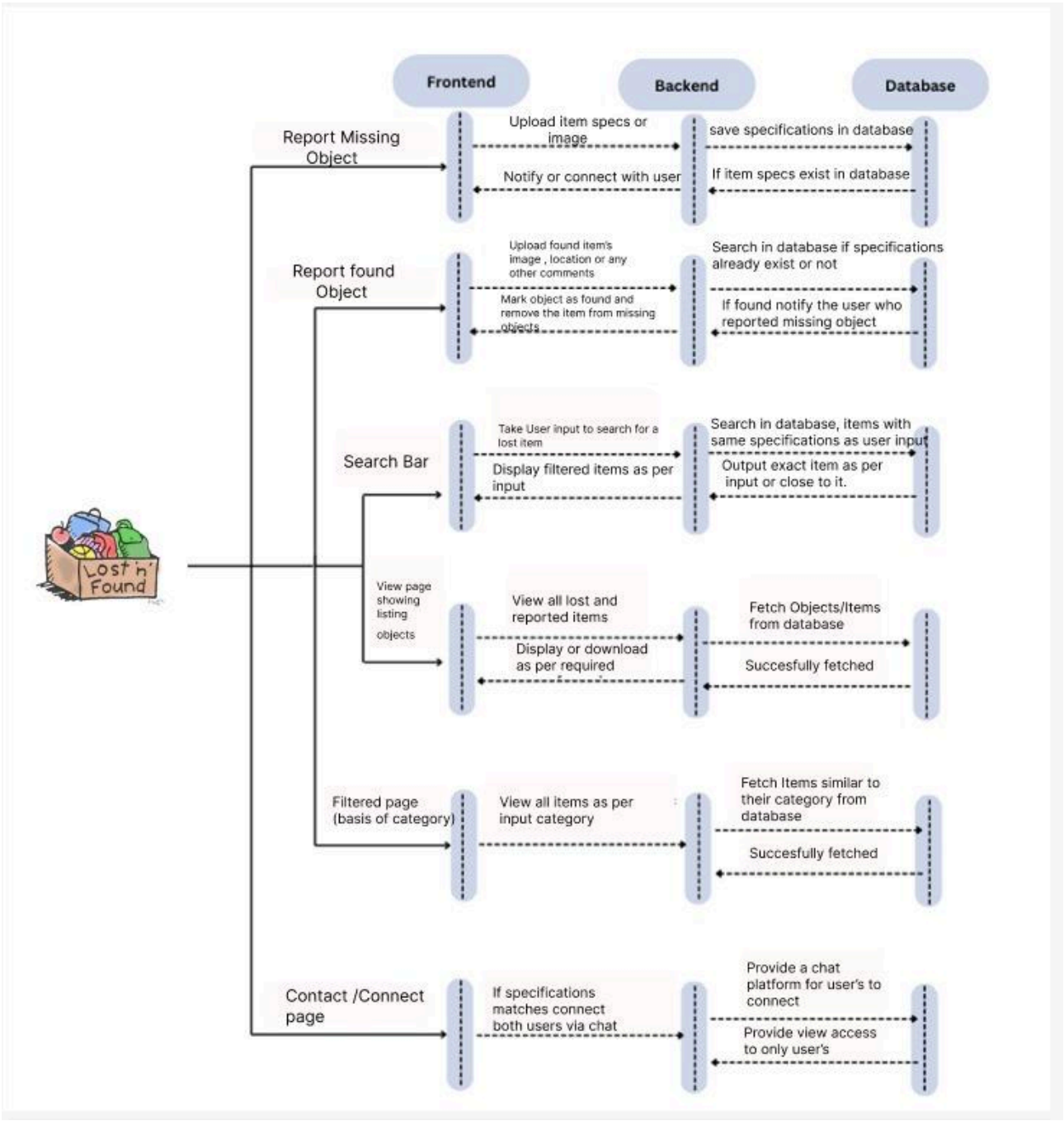


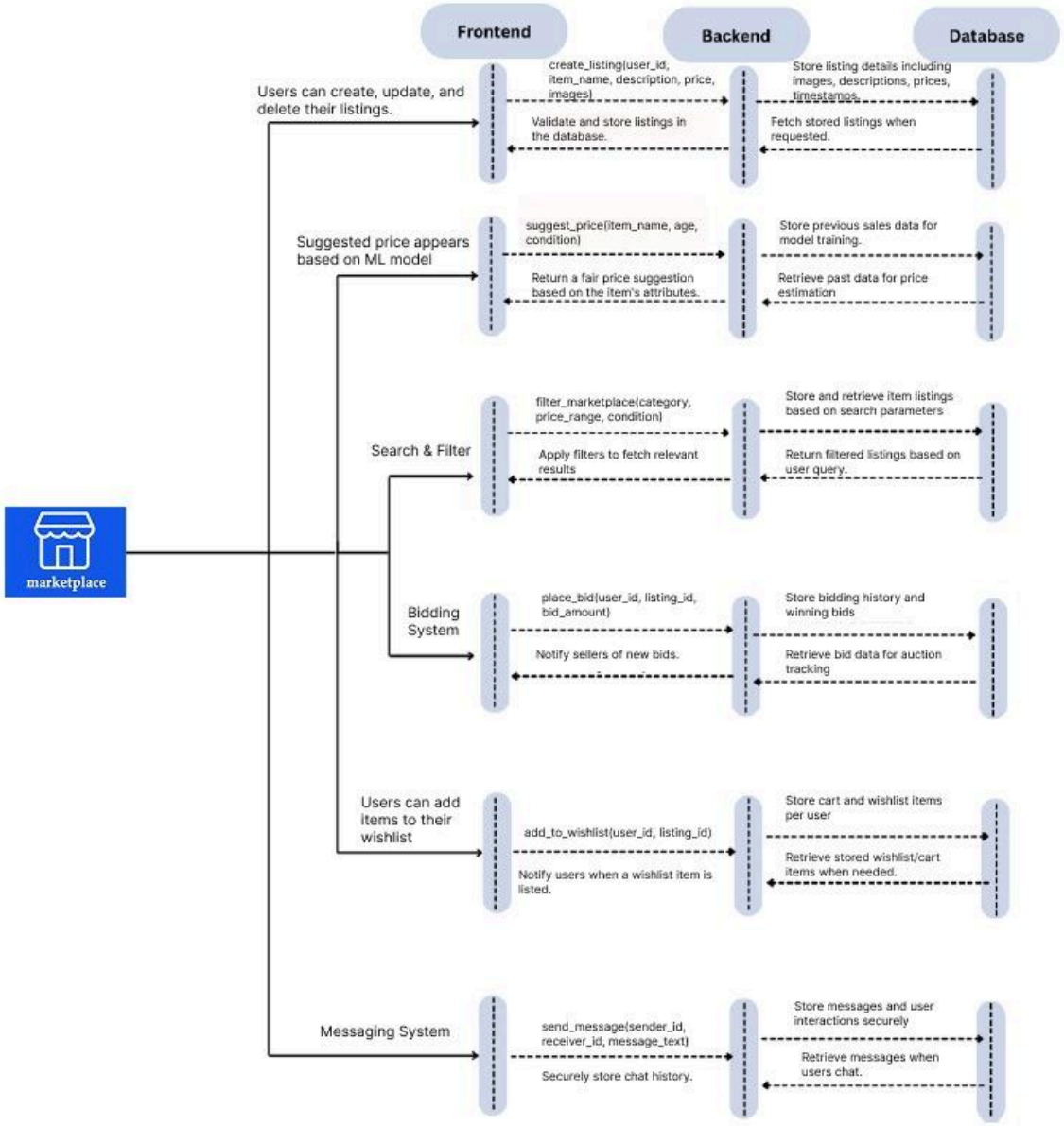
Marketplace System



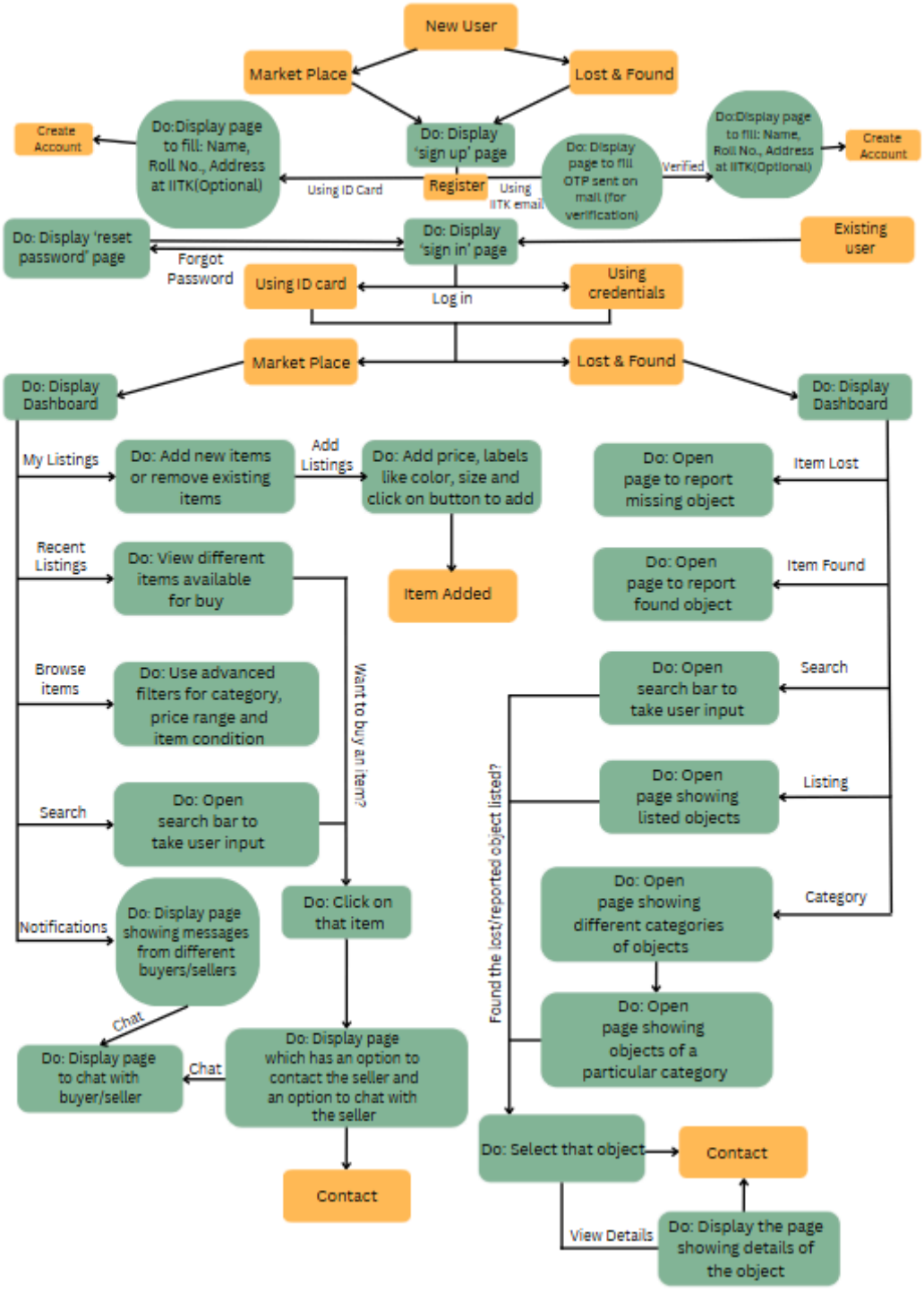
3.3 Sequence Diagrams



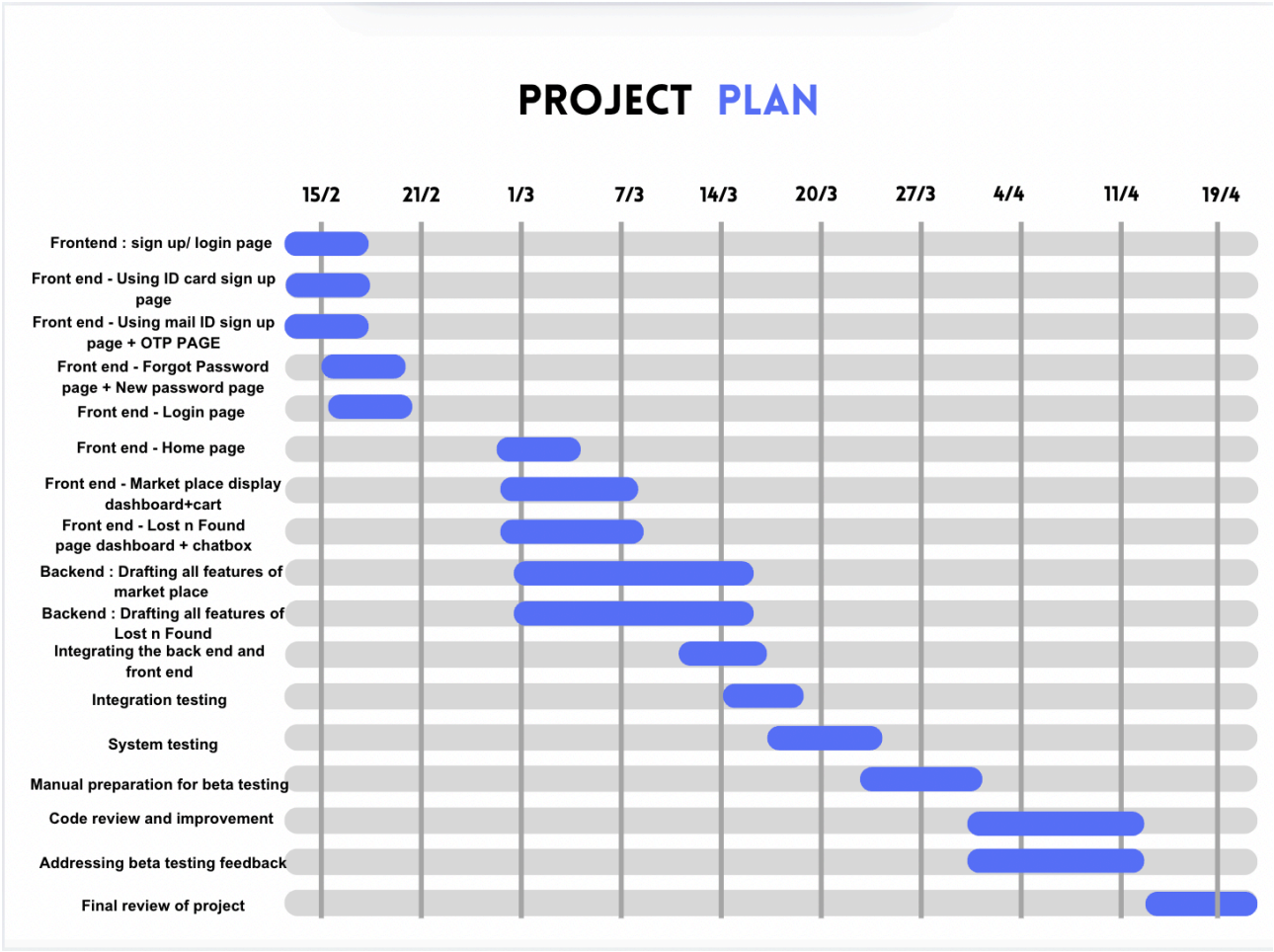

Frontend
Backend
Database
Report Missing Object
Upload item specs or image
save specifications in database
Notify or connect with user
If item specs exist in database
Report found Object
Upload found item's image , location or any other comments
Search in database if specifications already exist or not
Mark object as found and remove the item from missing objects
If found notify the user who reported missing object
Search Bar
Take User input to search for a lost item
Search in database, items with same specifications as user input
Display filtered items as per input
Output exact item as per input or close to it.
Lost 'n' Found
View page showing listing objects
View all lost and reported items
Fetch Objects/Items from database
Display or download as per required
Succesfully fetched
Filtered page (basis of category)
View all items as per input category
Fetch Items similar to their category from database
Succesfully fetched
Contact /Connect page
If specifications matches connect both users via chat
Provide a chat platform for user's to connect
Provide view access to only user's


Frontend
Backend
Database
marketplace
Users can create, update, and delete their listings.
create_listing(user_id, item_name, description, price, images)
Validate and store listings in the database.
Store listing details including images, descriptions, prices, timestamps.
Fetch stored listings when requested.
Suggested price appears based on ML model
suggest_price(item_name, age, condition)
Return a fair price suggestion based on the item's attributes.
Store previous sales data for model training.
Retrieve past data for price estimation
Search & Filter
filter_marketplace(category, price_range, condition)
Apply filters to fetch relevant results
Store and retrieve item listings based on search parameters
Return filtered listings based on user query.
Bidding System
place_bid(user_id, listing_id, bid_amount)
Notify sellers of new bids.
Store bidding history and winning bids
Retrieve bid data for auction tracking
Users can add items to their wishlist
add_to_wishlist(user_id, listing_id)
Notify users when a wishlist item is listed.
Store cart and wishlist items per user
Retrieve stored wishlist/cart items when needed.
Messaging System
send_message(sender_id, receiver_id, message_text)
Securely store chat history.
Store messages and user interactions securely
Retrieve messages when users chat.

3.4 State Diagram



Project Plan



Activity	Start Date	Duration	End date	Team Members
Frontend : sign up/ login page	15/02/25	2	17/02/25	Riddhima,Ritul
Front end - Using ID card sign up page	15/02/25	2	17/02/25	Krishiv, Kanika

Front end - Using mail ID sign up page + OTP PAGE	15/02/25	2	17/02/25	Riya, Rachit
Front end - Forgot Password page + New password page	17/02/25	2	19/02/25	Vishal, Vishap
Front end - Login page	17/02/25	2	19/02/25	Manavjeet, Esra
Front end - Home page	1/03/25	3	4/03/25	Riddhima,Ritul
Front end - Market place display dashboard+cart	1/03/25	3	4/03/25	Krishiv, Kanika
Front end - Lost n Found page dashboard + chatbox	1/03/25	3	4/03/25	Riya, Rachit
Backend : Drafting all features of market place	1/03/25	13	14/03/25	Vishal, Vishap, Krishiv, Riddhima
Backend : Drafting all features of Lost n Found	1/03/25	13	14/03/25	Manavjeet, Esra, Kanika, Ritul
Integrating the back end and front end	13/03/25	3	16/03/25	Entire team
Integration testing	15/03/25	2	17/03/25	Entire team
System testing	17/03/25	3	21/03/25	Entire team
Manual preparation for	25/03/25	3	28/03/25	Entire team

beta testing				
Code review and improvement	29/03/25	12	11/04/25	Entire team
Addressing beta testing feedback	29/03/25	12	11/04/25	Entire team
Final review of project	12/04/25	7	19/04/25	Entire team

Group Log

• Offline meetings were hosted on a regular basis to discuss the software ideas, consistently monitor the completion of tasks assigned to every team member.

• Apart from this, a WhatsApp Group was also created to discuss daily updates over the documentation, along with brainstorming over some sudden change of thoughts.

Date, Time & Duration	*Meet Type*	*Ongoing Discussions in the meet*
26/01/2025	*Offline Meet*	*Forming groups among our team to tackle different aspects of the project*
2/02/2025	*Online Meet*	*Dividing sections of the SDD, and assigning them to members of the team*
4/02/2025	*Online Meet*	*To check on the progress of all team members*
5/02/2025	*Offline Meet*	*Final meet amongst the team members as a robust draft of the SDD documentation*