CSC322

Professor Wei

10/31/24

Jezlea Ortega, Aprajita Srivastava, Nadia, Suhana Lama, and Tahsina

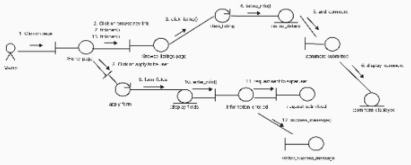
Phase II: Design Report

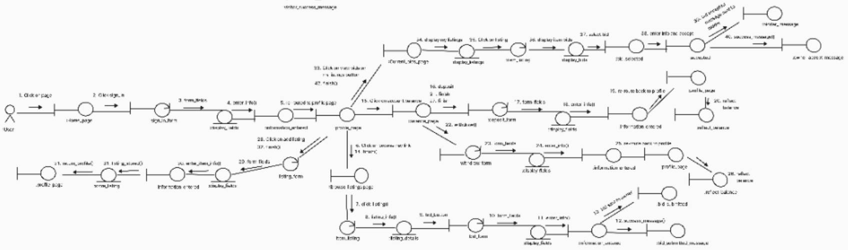
1. Introduction

The TrustSphere project is an e-bidding system that enables visitors to browse listings and apply to become verified users, allowing them to participate in bidding, listing items, and transaction rating functionalities. Superusers in the system have administrative capabilities, such as approving applications and moderating user behavior. Our goal is to build a user-friendly, robust platform with secure handling of user roles, transactions, and application processes, structured around Node.js, MongoDB, and React.

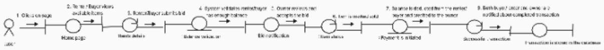
For the collaboration diagram, we would show primary interactions between core classes, including **Visitor**, **User**, **Superuser**, **Listing**, **Bid**, **Transaction**, and **Application**. Each class would have clear associations depicting how data flows through the system for key use cases like browsing, bidding, approving, rating, and monitoring users.

Deleting After: Just put which one requirements you did it here for organization. There are ten requirements each person has to do two NAME	REQUIREMENTS CHOSEN
Joshua	DONE
Apregita	Done
Sai	Done
Hindia	Done
Ishmael	Done



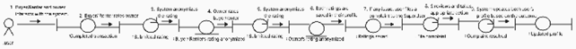


4th requirement



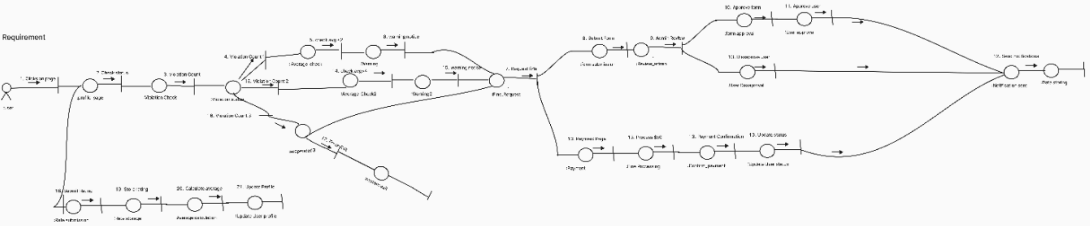
1. Clicks on page
User
2. Renter / Buyer views available items
:Home page
3. Renter/Buyer submits bid
:Items details
4. System validates renter/buyer has enough balance
:Balance validation
5. Owner reviews and accepts the bid
:Bid notification
6. Item is marked sold
:Item status
7. Balance is deducted from the renter/ buyer and credited to the owner
:Payment is initiated
8. Both buyer / renter and owner are notified about completed transaction
:Successful transaction
Transaction is stored in the database

5th requirement

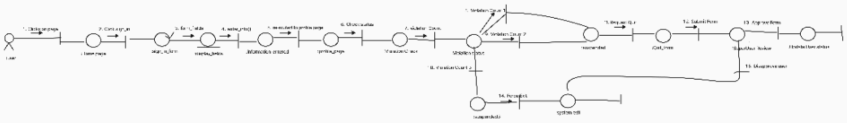


1. Buyer/Renter and owner interacts with the system
User
2. Buyer/ Renter rates owner
:Completed transaction
3. System anonymizes the rating
:Submitted rating
4. Owner rates buyer/renter
:Buyer / Renters rating anonymized
5. System anonymizes the rating
:Submitted rating
6. Both ratings are saved in their profile
:Owner's rating anonymized
7. If any issues, user files a complaint to the SuperUser
:Ratings saved
8. SU reviews and takes appropriate action
:File complaint
9. System replaces both user's profile based on the outcome
:Complaint resolved
:Updated profile

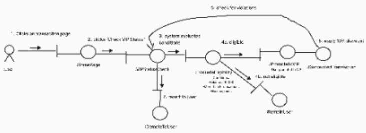
6th Requirement



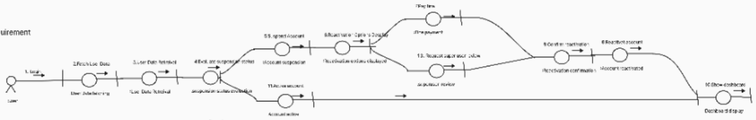
7th Requirement



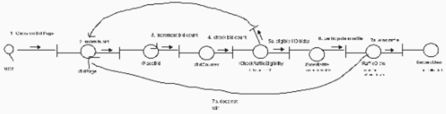
8th Requirement



9th Requirement



10th Requirement

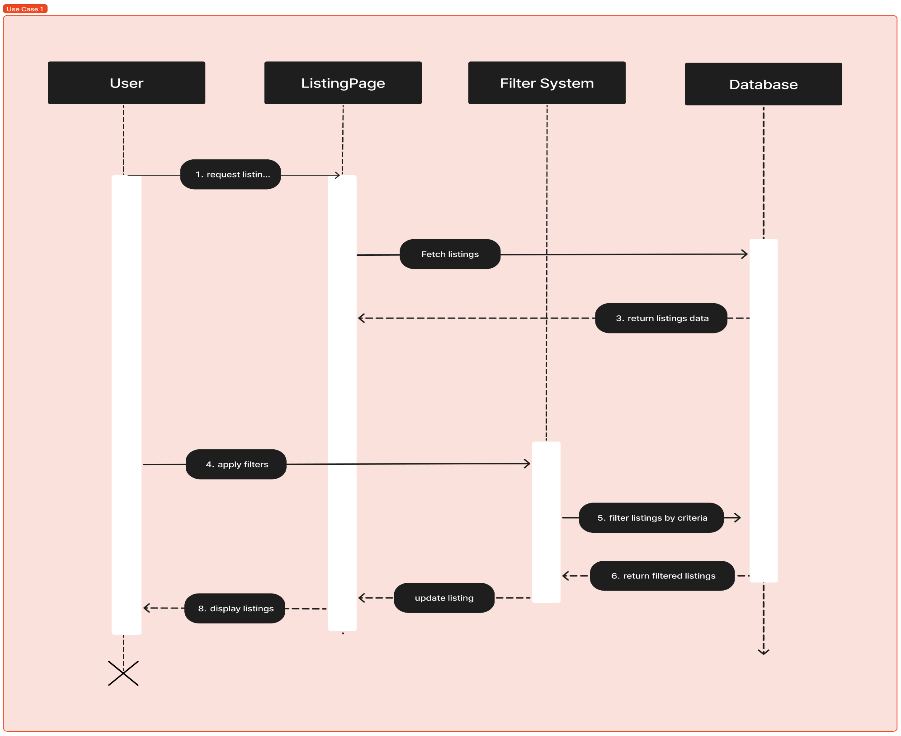


2. All use cases

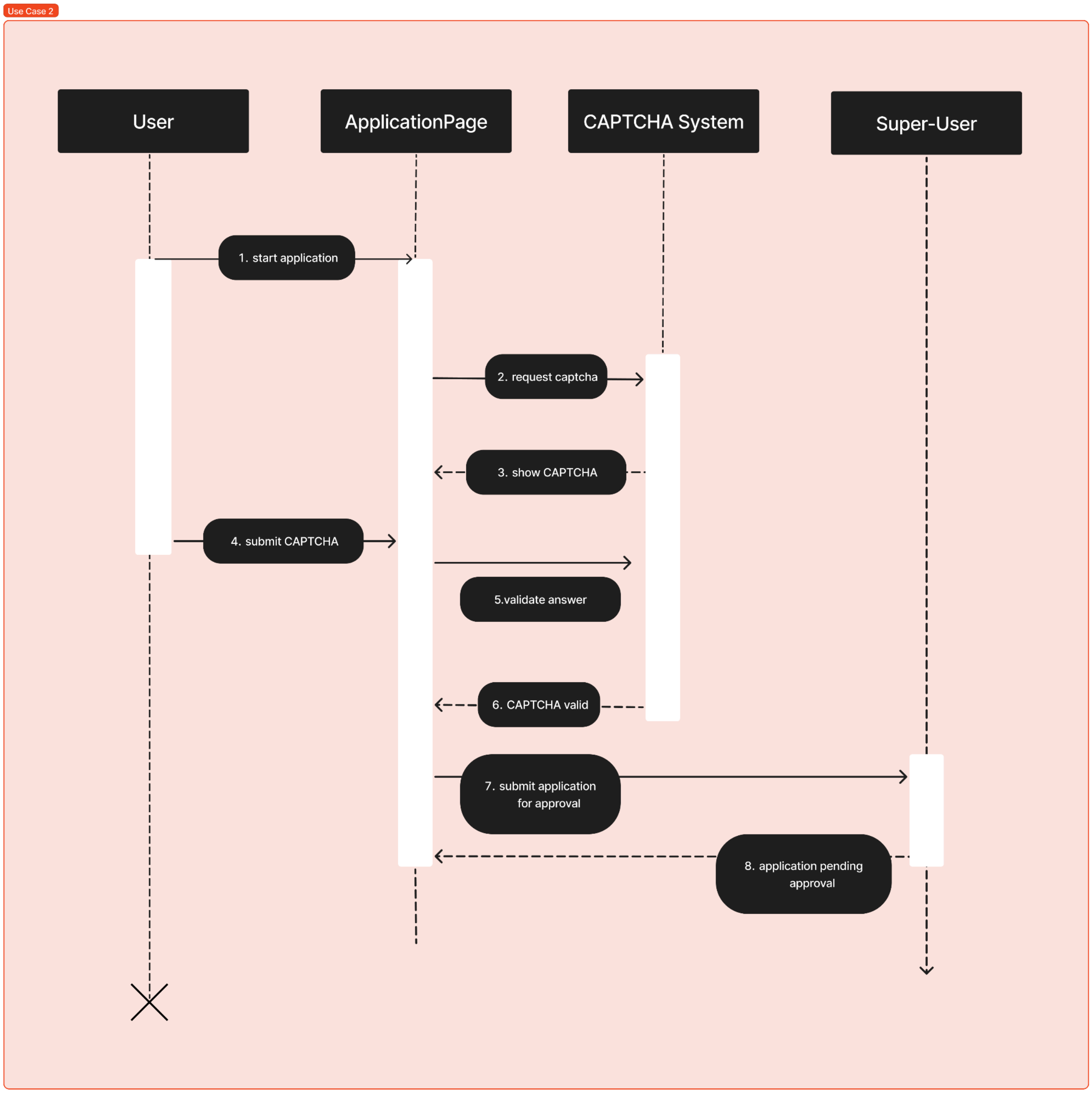
o Scenarios for each use case: normal AND exceptional scenarios

o Collaboration or sequence class diagram for each use case, choose 3 or more use cases: draw the Petri-nets instead of class diagrams

1. **Browse Listings (Visitor):**
   - **Description:** Visitors can browse all listings on the platform. They can apply filters (price, availability, category) but cannot make bids or complete transactions.
   - **Precondition:** The visitor must have stable internet access.
   - **Postcondition:** The visitor is shown a list of items, but no interaction beyond viewing is permitted.

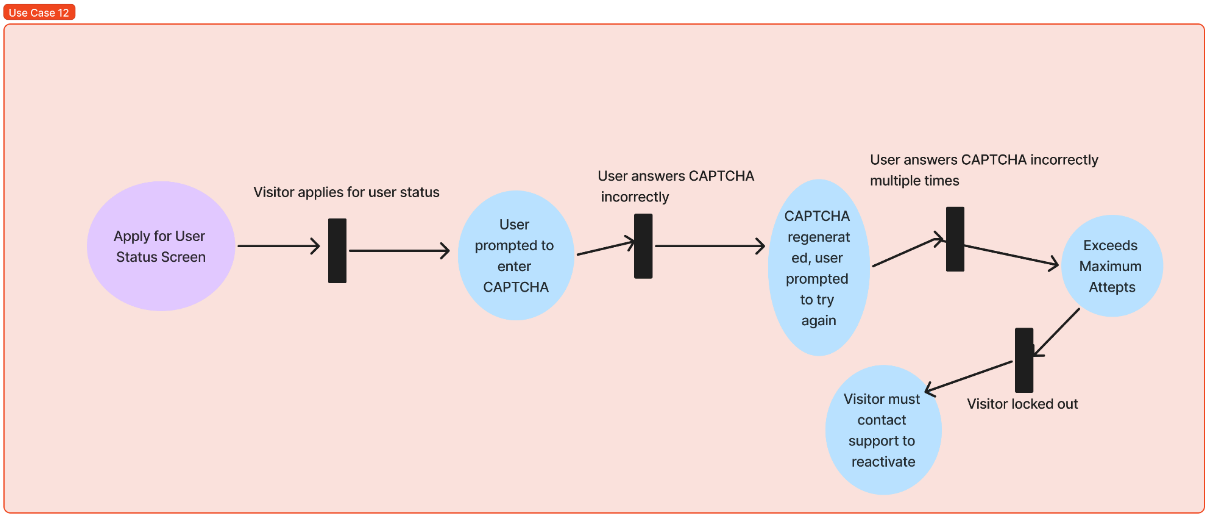


2. **Apply to Become a User (Visitor):**
   - **Description:** A visitor can apply for user status by answering an arithmetic-based CAPTCHA. The application is sent to a superuser for approval.
   - **Precondition:** The visitor must correctly answer the CAPTCHA.
   - **Postcondition:** The visitor's application is submitted for approval.


Use Case 2
User
ApplicationPage
CAPTCHA System
Super-User
1. start application
2. request captcha
3. show CAPTCHA
4. submit CAPTCHA
5.validate answer
6. CAPTCHA valid
7. submit application for approval
8. application pending approval

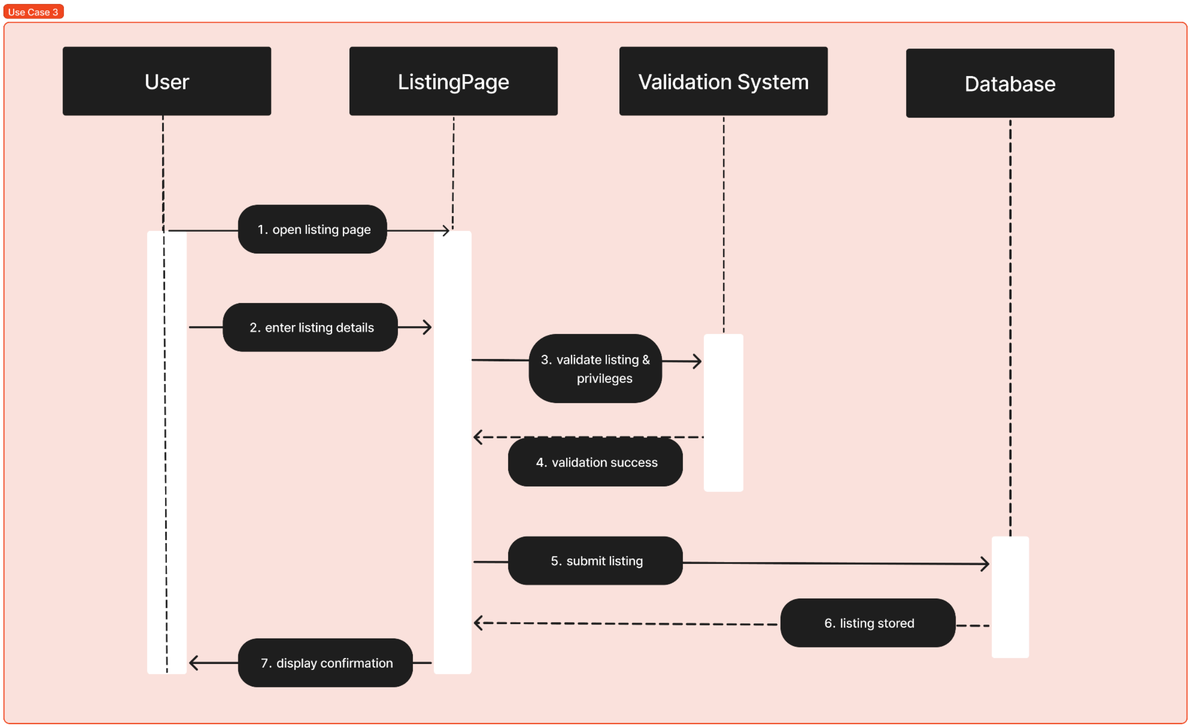
3. Exception:

The visitor attempts to apply for user status by answering an arithmetic-based CAPTCHA, but the visitor answers the CAPTCHA incorrectly multiple times and exceeds the maximum allowed attempts.


Use Case 12
Apply for User Status Screen
Visitor applies for user status
User prompted to enter CAPTCHA
User answers CAPTCHA incorrectly
CAPTCHA regenerated, user prompted to try again
User answers CAPTCHA incorrectly multiple times
Exceeds Maximum Attepts
Visitor locked out
Visitor must contact support to reactivate

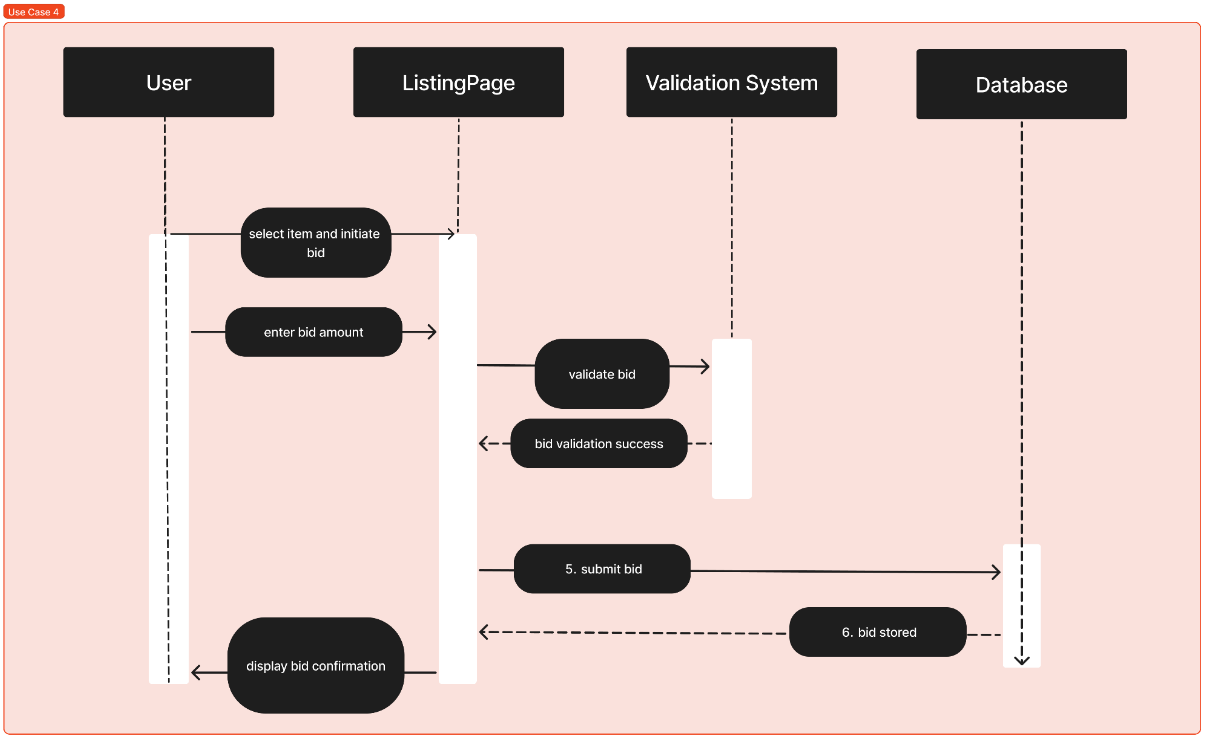
4. **List Item for Sale/Rent (User):**

- **Description:** Registered users can list items or services for sale or rent. The listing must include a description, price, and deadline.
- **Precondition:** The user must be logged in with sufficient privileges.
- **Postcondition:** The item is visible in the listings for other users and visitors to view.



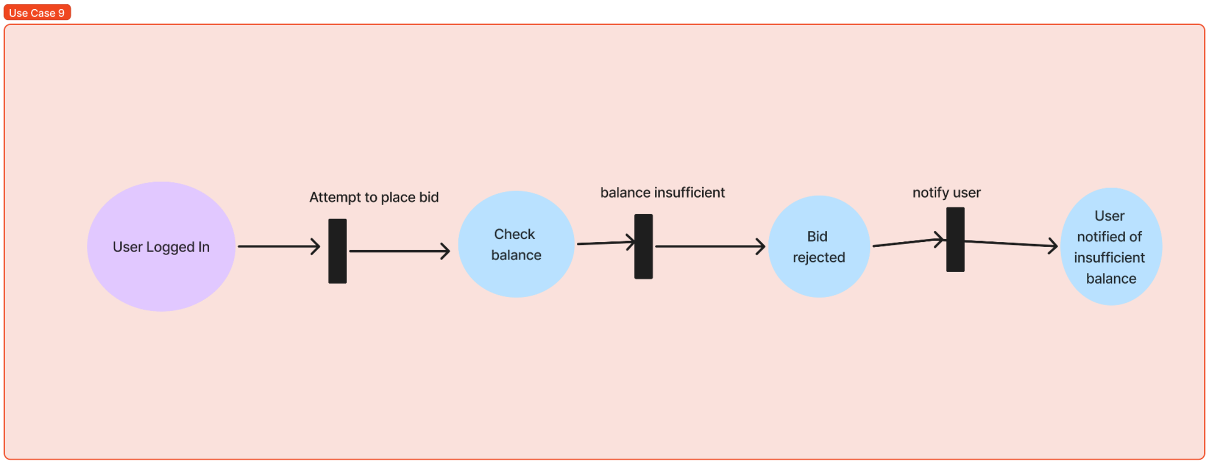
5. **Place Bid (User):**

- **Description:** Users can place bids on items. The system verifies the user's balance before allowing the bid.
- **Precondition:** The user must be logged in and have a sufficient account balance.
- **Postcondition:** The bid is submitted and linked to the item.



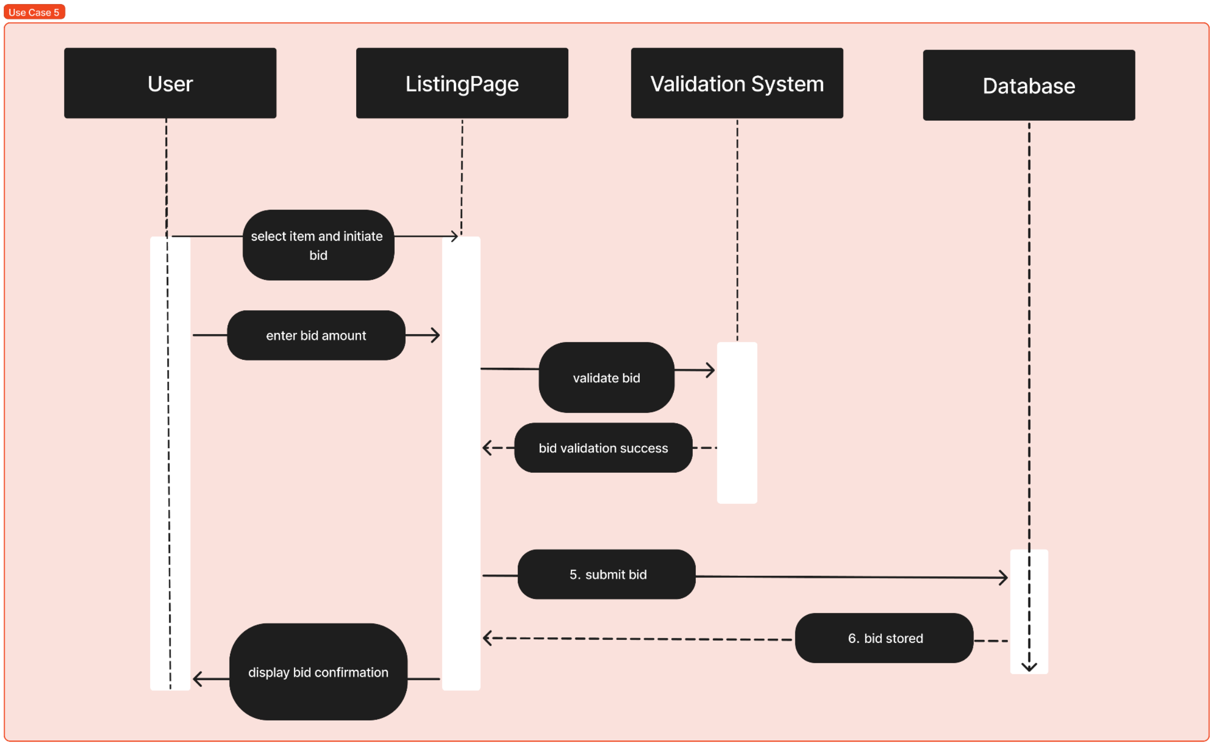
6. Exception:

The user tries to place a bid, but their account balance is insufficient to place the bid. This deviates from the normal flow where the user's balance is sufficient.



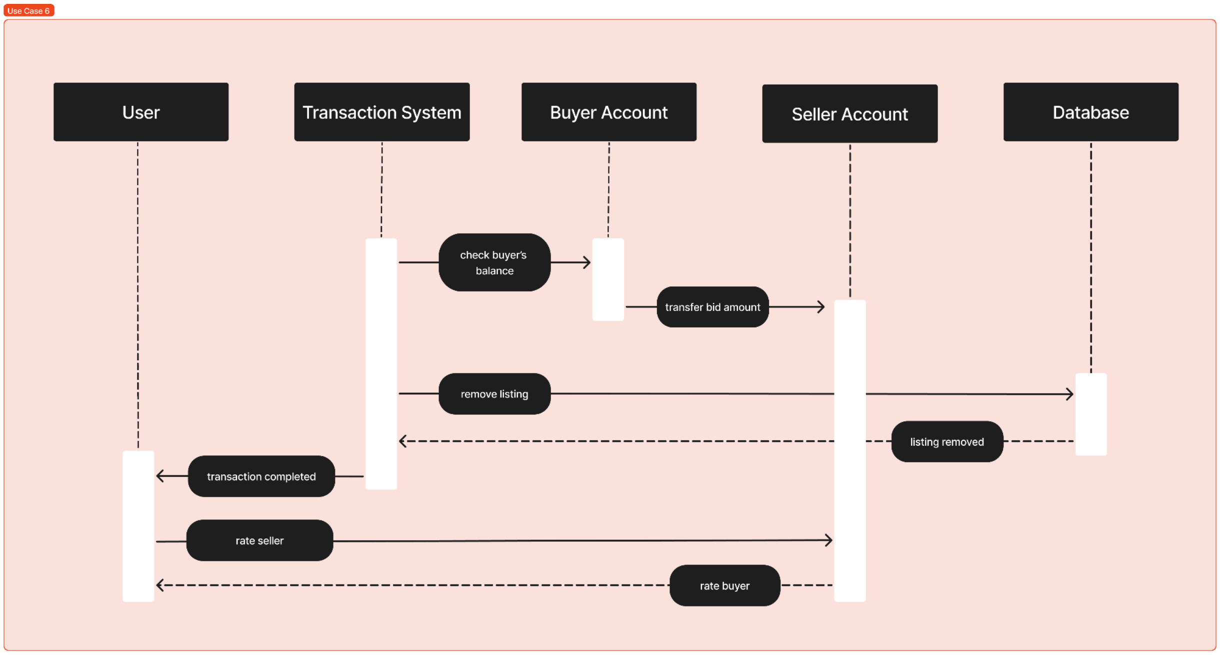
7. **Complete Transaction (User):**

- **Description:** Once a bid is accepted, the system automatically transfers the bid amount from the buyer's account to the seller's account, and the listing is removed.
- **Precondition:** The bid must be accepted, and the user must have sufficient funds.
- **Postcondition:** The transaction is completed, and both buyer and seller can rate each other.



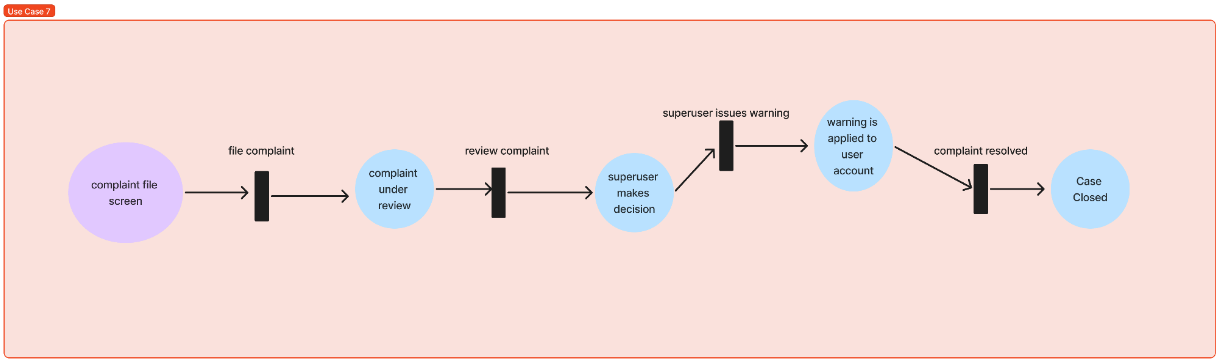
8. **Rate Transaction (User):**

- **Description:** After a transaction, both buyer and seller can rate each other anonymously on a scale of 1-5.
- **Precondition:** A completed transaction must exist between the two users.
- **Postcondition:** The rating is recorded and affects the user's profile rating.



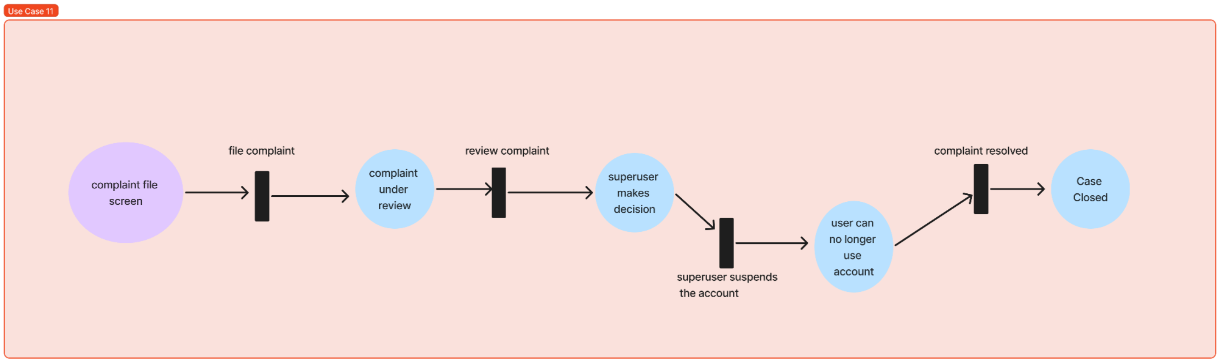
9. **Manage Complaints (Superuser):**

- **Description:** Superusers review complaints about users and can apply penalties or suspend accounts as necessary.
- **Precondition:** A complaint must be filed by a user.
- **Postcondition:** Superusers take action by issuing a warning.

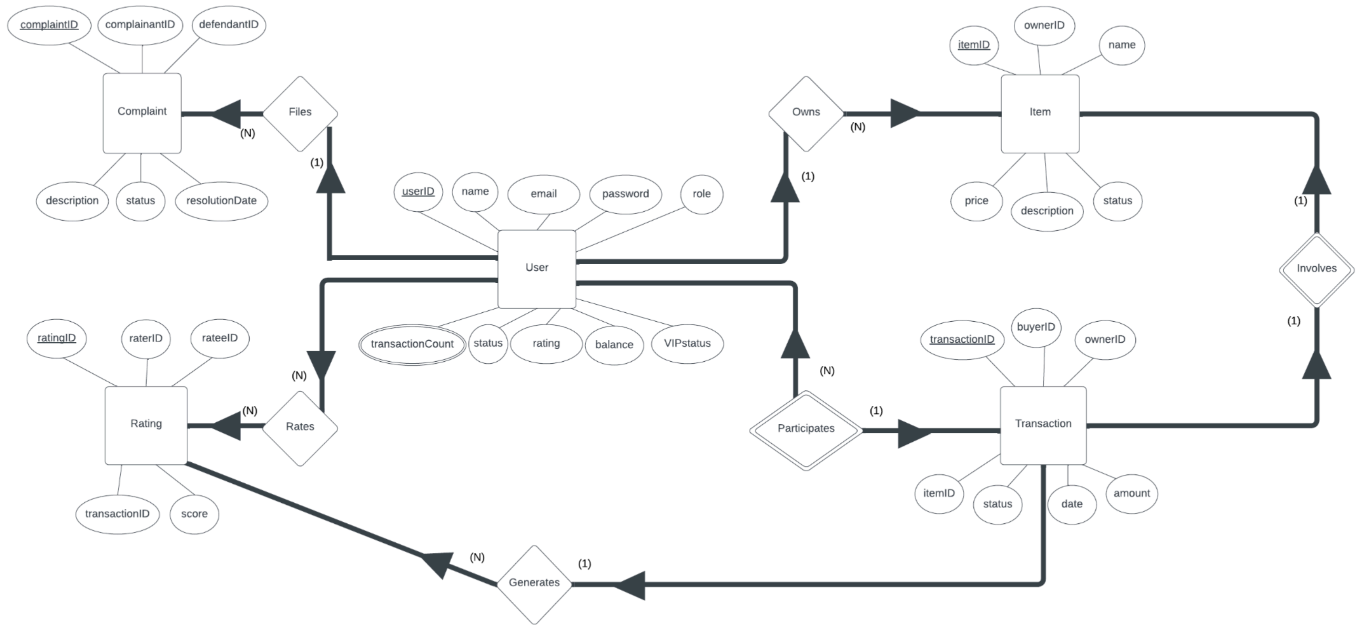


10. Exception:

Superusers take action by issuing a suspension, instead.



3. E-R diagram for the entire system



4. Detailed design:

for EVERY method use pseudo-code to delineate the input/output and main functionalities

Pseudocode required for each spec:

We have set up our DB Models in MongoDB so we'll have pseudocode for API that delivers each of the information here:

API: GET /listings

Purpose: Allows a Visitor (V) to browse listings provided by Users (U).

Input: Logged-in User (V or U or S)

1. Check if the user is logged in.
2. If not logged in, return a 401 Unauthorized response.
3. Retrieve all listings from the `Listings` table in MongoDB.
4. Return the listings with a 200 OK response.

API: POST /apply-user

Purpose: Allows a Visitor (V) to apply to be a User (U) which requires solving a random arithmetic CAPTCHA question.

Input: Visitor's response to the arithmetic question, logged-in Visitor (V)

1. Check if the user is logged in and is a Visitor.
2. If the user is not a Visitor, return a 403 Forbidden response.
3. Generate a random arithmetic question and display it.
4. Verify if the answer is correct. If incorrect, return a 400 Bad Request with "Incorrect answer. If correct, create an application record in `UserApplications`.
5. Notify Superusers (S) that there is a pending application for review.
6. Return a 200 OK response with "Application submitted successfully."

API: POST /approve-application

Purpose: Allows a Superuser (S) to approve a Visitor's application to become a User

Input: Application ID, Superuser (S)

1. Verify the logged-in user is a Superuser (S).
2. If not, return a 403 Forbidden response.
3. Retrieve the application from `UserApplications` using the Application ID.
4. If not found, return a 404 Not Found response.
5. Set application status to "Approved" and update the applicant's role in `User` table to "User".
6. Return a 200 OK response with "Application approved."

API: POST /account/deposit-withdraw

Purpose: Allows a User (U) to deposit or withdraw money.

Input: Amount, Operation Type (Deposit or Withdraw), Logged-in User (U)

1. Verify the logged-in user is a User.
2. If not, return a 403 Forbidden response.
3. Perform the deposit or withdrawal based on the operation type: Deposit: Add the amount to the user's balance. Withdraw: Ensure the user has enough balance, then deduct the amount.
4. Update the user's account balance in the `User` table.
5. Return a 200 OK response with an updated balance.

API: POST /listings/create

Purpose: Allows a User (U) to list items for sale/rent.

Input: Item details, Asking Price, Logged-in User (U)

1. Verify the logged-in user is a User.
2. If not, return a 403 Forbidden response.
3. Create a new listing with the item details and asking price.
4. Insert the listing into the `Listings` table with the owner's user ID.
5. Return a 201 Created response with "Listing created successfully."

API: POST /listings/bid

Purpose: Allows a User (U) to place a bid on an item they are interested in.

Input: Listing ID, Bid Amount, Logged-in User (U)

1. Verify the logged-in user is a User.
2. If not, return a 403 Forbidden response.
3. Ensure the user has adequate funds for the bid.
4. Retrieve the listing and check if bidding is still open.
5. Place the bid by saving it in the `Bids` table linked to the listing.
6. Return a 200 OK response with "Bid placed successfully."

API: POST /listings/accept-bid

Purpose: Allows the listing owner to accept a bid.

Input: Listing ID, Bid ID, Logged-in User (U as owner)

1. Verify the logged-in user is the owner of the listing.
2. If not, return a 403 Forbidden response.
3. Retrieve the bid and listing.
4. Transfer the bid amount from the buyer’s account to the owner’s account.
5. If the user is a VIP apply a 10% discount.
6. Remove the listing from active listings.
7. Return a 200 OK response with "Bid accepted, transaction completed."

API: POST /transactions/rate

Purpose: Allows the buyer/renter to rate the transaction.

Input: Transaction ID, Rating (1-5), Logged-in User (U)

1. Verify the logged-in user participated in the transaction.
2. If not, return a 403 Forbidden response.
3. Record the rating in the `Ratings` table, linked to the transaction.
4. Return a 200 OK response with "Rating submitted."

API: POST /users/check-ratings

Purpose: Checks ratings for a User (U) and suspends or fines them if necessary.

Input: Logged-in User (U)

1. Retrieve all ratings for the user.
2. Calculate average rating.
3. If the average rating is below 2: If suspended less than 3 times, apply a $50 fine.If suspended 3 times, deactivate the account.
4. Update the user's status in `User` table.

API: POST /users/check-vip

Purpose: Checks if a User (U) meets criteria to become a VIP.

Input: Logged-in User (U)

1. Check if user meets VIP criteria: Balance over $5,000. More than 5 transactions and No complaints
2. If criteria met, grant VIP status: Apply 10% discount on transactions.
3. If criteria are violated, revoke VIP status.
4. Update the user's role in `User` table.
5. Return the updated status with a 200 OK response.

5. System screens:

1. Home Page (home.js)

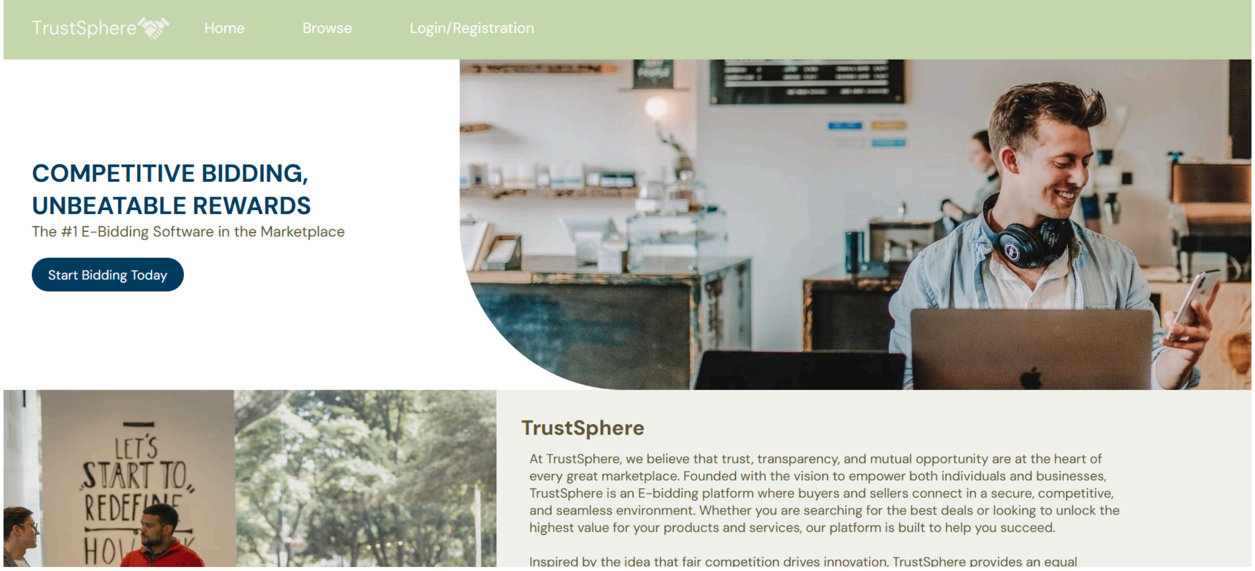
Purpose: The main landing page for both visitors and registered users, offering an overview of TrustSphere's features and a quick way to log in or register.

Features:

- **Login Button:** Directs users to the login page.
- **Register Button:** Takes visitors to the registration page to create a new account.
- **Overview Section:** Provides a brief explanation of TrustSphere's purpose, services, and how it benefits buyers and sellers.

Design:

- **Header:** Includes TrustSphere's logo, navigation menu, and links to Login/Register.
- **Main Section:** Highlights key platform features such as secure transactions, listings, and buyer/seller benefits.
- **Footer:** Links to Terms of Service, Privacy Policy, and Contact Us page.



COMPETITIVE BIDDING, UNBEATABLE REWARDS

The #1 E-Bidding Software in the Marketplace

TrustSphere

At TrustSphere, we believe that trust, transparency, and mutual opportunity are at the heart of every great marketplace. Founded with the vision to empower both individuals and businesses, TrustSphere is an E-bidding platform where buyers and sellers connect in a secure, competitive, and seamless environment. Whether you are searching for the best deals or looking to unlock the highest value for your products and services, our platform is built to help you succeed.

Inspired by the idea that fair competition drives innovation, TrustSphere provides an equal

2. User Login Page (user_login.js)

Purpose: Registered users log in to their accounts.

Features:

- **Username and Password Fields:** Secure input fields for credentials.
- **Login Button:** Authenticates user details and takes them to their dashboard.
- **Register Button:** Redirects visitors to the registration page if they don't have an account.

Design:

- **Simple Login Form:** Fields for username and password with a login button below.
- **Navigation Options:** Includes a link to the registration page for new users.

3. Visitor Registration Page (visitor_registration.js)

Purpose: Allows visitors to create a new account.

Features:

- **Username Field:** Users choose a unique username.
- **Password Field:** Secure password input.
- **Email Field:** Users enter their email for account verification and notifications.

- **Address Fields:** Users provide their address for identification and transaction purposes.
- **Register Button:** Submits the form to create a new account.
- **Return to Sign-In Button:** Navigates users back to the login page.

Design:

- **Interactive Form:** Structured input fields for username, password, email, and address with placeholders for clarity.
- **Clear Buttons:** Two action buttons for submitting the registration form or navigating back to the login page.

4. Add Listings Page (add_listings.js)

Purpose: Users can post items or services for sale, rent, or purchase.

Features:

- **Title and Description Fields:** Users provide the name and description of the listing.
- **Type Selection Dropdown:** Users can categorize the listing as “For Sale,” “For Rent,” or “Wanted.”
- **Price and Date Fields:** Users enter the price and listing date.
- **Submit Button:** Publishes the listing.

Design:

- **Form Layout:** Structured with clear labels for title, description, price, and type selection.
- **Preview Option:** Users can preview their listing before submission.

5. Deposit Page (deposit.js)

Purpose: Enables users to deposit funds into their TrustSphere account.

Features:

- **Account Balance Display:** Shows the user's current balance.
- **Deposit Amount Field:** Users input the amount to deposit.
- **Credit Card Information:** Required fields for cardholder name, card number, expiration date, and CVV.
- **Billing Address Fields:** Used to verify the transaction.
- **Deposit Button:** Processes the deposit and updates the balance.
- **Return Button:** Returns users to the previous page.

6. Withdraw Page (withdraw.js)

Purpose: Allows users to withdraw funds from their TrustSphere account.

Features:

- **Account Balance Display:** Shows available balance before withdrawal.
- **Withdraw Amount Field:** Users specify the amount to withdraw.
- **Credit Card Information:** Required for processing the withdrawal to an external account.
- **Billing Address Fields:** Used for transaction validation.
- **Withdraw Button:** Completes the withdrawal and updates the balance.
- **Return Button:** Allows users to go back to the previous page.

6. Memos of group meetings and possible concerns of teamwork

Date: 10/7/2024
To: Professor Jie Wei
From: Jezlea Ortega, Aprajita Srivastava, Nadia Ben Slima,, Suhana Lama, and Tahsina Khan
Subject: First Team Meeting to Discuss Roles, Technologies, and Project Management

---

The TrustSphere team met at approximately 2:15 - 3: 45 PM to discuss initial project setup: team roles, technologies to implement our website, weekly meeting times, action tracking, and report 1. Furthermore, the team was distributed into a frontend side and a backend team (backend inclusive of the database development. The frontend team will consist of Jezlea and Tashina, focusing on user interface and interaction. The backend team, comprised of Aprajita, Suhana, and Tahsina, will handle the backend architecture and functionality, alongside handling the database creation. Also, there was a selection of React, Node, and MongoDB for the project development. Furthermore, the team proceeded to discuss report 1 and distribute sections within the assignment: Tahsina assigned 1 - 1.2, Jezlea assigned 1.3 - 1.5, Aprajita assigned 2 and 3, Nadia assigned 2.1 and 3.1, and Suhana assigned 2.2, 3.2, and 4. After the distribution of tasks, the team proceeded to discuss project management. Tashina suggested the creation of an action tracker to monitor progress on the frontend and backend portion of each specification. Jezlea proceeded to create this tracker in Excel and share it with the entire team. Concluding the meeting, the team agreed to meet weekly on Mondays at 2 - 3:30 PM, with Tahsina and Jezlea agreeing to meet on 10/14 at 3:30 PM to discuss frontend design.

Date: 10/14/2024
To: Professor Jie Wei
From: Jezlea Ortega, Aprajita Srivastava, Nadia Ben Slima, and Tahsina Khan
Subject: Report 1 Progress and Project Development

The TrustSphere team met to discuss progress on report 1 and if there is any confusion regarding their assignments. Jezlea and Aprajita indicated they completed their assignments and they may be reviewed by the team if needed. Tahsina and Nadia discussed needing more time but ensured completion prior to the due date. There was no further progress on the project, as such the team discussed the goals for the week. Jezlea indicated she would begin the home page, log in, and register pages for the front end. The backend team discussed setting up the backend environment and Mongo database. As general project tasks and goals were discussed, the backend team dropped off the call and Tahsina and Jezlea remained to discuss frontend design. Tahsina and Jezlea created a general page design for the home page via Figma and picked a color palette to utilize throughout the web pages. The meeting concluded with Jezlea indicating she would start the page designs based on the color palette selected and the Figma design outlined.

Date: 10/21/2024
To: Professor Jie Wei
From: Jezlea Ortega, Aprajita Srivastava, and Suhana Lama
Subject: Backend Setup and Database Connection

The TrustSphere team met to discuss backend setup and database connection. Aprajita spearheaded the meeting by sharing her screen to step by step show Jezlea and Suhana the files and installations needed to start the server. There was difficulty with the backend setup that needed to be resolved, but ultimately the team was able to get the server started and receive a message indicating the database was connected. Given the lack of attendance, Jezlea wrapped up the meeting by indicating the home page was near completion and sent out a general message regarding goals for next week in the team chat. In the next week, Aprajita would set up the database for Suhana and Nadia to then start creating API points. Jezlea would complete the home page and proceed to work on account balance pages.

Date: 10/28/2024
To: Professor Jie Wei

From: Jezlea Ortega, Aprajita Srivastava, Nadia Ben Slima, Tahsina Khan, and Suhana Lama
Subject: Project Requirements Progress and Report II

The TrustSphere team met to discuss the project requirements progress. Unfortunately, Aprajita was only able to set up two tables in the database, and as such the team did not achieve their goals. However, Aprajita assured the team the database would be set up by 11/1. Additionally, the backend team scheduled a meeting for 11/1 to discuss in more detail specific tasks within each requirement for each individual to maintain ownership over. Additionally, the frontend team, specifically Jezlea, completed the home page and successfully began the creation of the withdraw and deposit account balance pages. Moreover, the team proceeded to allocate tasks for the next week. Tahsina agreed to create the visitor's page and work on the visitor's application to be a user page. Jezlea would continue to create the account balance pages, working additionally on a page for viewing account balance and selecting deposit or withdraw. Aprajita aimed to complete the login and register backend components, while Nadia and Suhana would be assigned tasks post-backend meetings. In the completion of our task assignment, we completed the meeting by discussing report II and allocating sections for each person: the introduction being shared amongst everyone, Tahsina assigned use cases, Suhana assigned the ER diagram, Aprajita working on detailed design, Nadia working on system screens, and Jezlea assigned memos and address of git repo.

Date: 11/4/2024
To: Professor Jie Wei
From: Jezlea Ortega, Aprajita Srivastava, Nadia Ben Slima, Tahsina Khan, and Suhana Lama
Subject: Project Requirements Progress and Report II

The TrustSphere team met to discuss the progress of the project requirements. Aprajita was able to create the login and register API points and indicated front end may connect these APIs. Additionally, Jezlea completed the account balance pages and add-listing form, indicating they are now dependent on API creation for completion. Tahsina was unable to complete her assigned tasks in time and as such will now complete them by 11/13. In terms of weekly assignments, Jezlea indicated she will attempt to complete the frontend pages for complaints being filed and browse listings. In regards to the backend, the following API creation assignments were distributed: Suhana will do get-listings, bid-listing, and approve-visitors, Nadia will do buy-listing, add-comment, and add-complaint, and Aprajita will complete account-balance, withdraw, and deposit.

Moreover, the team proceeded to discuss Report II progress. The first item on the report was distributed evenly to ensure each team member contributed to the collaboration class diagrams. Jezlea completed her portion today, while the others will work on theirs in the next week. Additionally, Jezlea completed the memos portion of the report. Finally, as no other progress was made the meeting wrapped up.

Date: 11/11/2024
To: Professor Jie Wei
From: Jezlea Ortega, Aprajita Srivastava, Nadia Ben Slima, Tahsina Khan, and Suhana Lama
Subject: Project Requirements Progress and Report II

The TrustSphere team met to discuss the progress of the project requirements and report II. In regards to the report, Jezlea and Aprajita finished their tasks. However, Suhana, Nadia, and Tahsina will need more time to finish up their tasks. The team is confident however the report will be completed prior to the deadline. Furthermore, the team proceeded to discuss their progress on weekly tasks. In the frontend team, Jezlea completed last weeks tasks and this week would work to finish the Superuser and User custom pages. This is inclusive of creating access to complaint forms, viewing listings, messages, etc. Tahsina was unable to complete her tasks from the prior week and as such will work this week on finishing. The backend team had trouble with task completion as Suhana ran into some issues with testing her API points. As such, she will spend the next two days working to rectify this issue. However, Aprajita successfully completed her tasks and Nadia's was near completion. In regards to backend assignments, API endpoints were distributed: Aprajita assigned API endpoints for /notify-users, /user-get-listing, /get-raffle, Suhana assigned API endpoints for /add-listing, /suspend-user, /approve-user, and Nadia API endpoints for /accept-bid, /rate-transactions, /check-vip.

Possible Concerns for Teamwork: Aprajita, Suhana, and Nadia are currently taking 335 which is an extremely difficult class requiring large amounts of time. The team is slightly concerned with the lack of progress made on requirements given backend will have a large amount of time needed towards 335 in a couple of weeks with their project.

7. Address of the git repo (github, gitlab, bitbucket, etc) of your team's work so far - put all materials including this report there
Github Repo Link: https://github.com/aprajita0/E-bidding_322Project