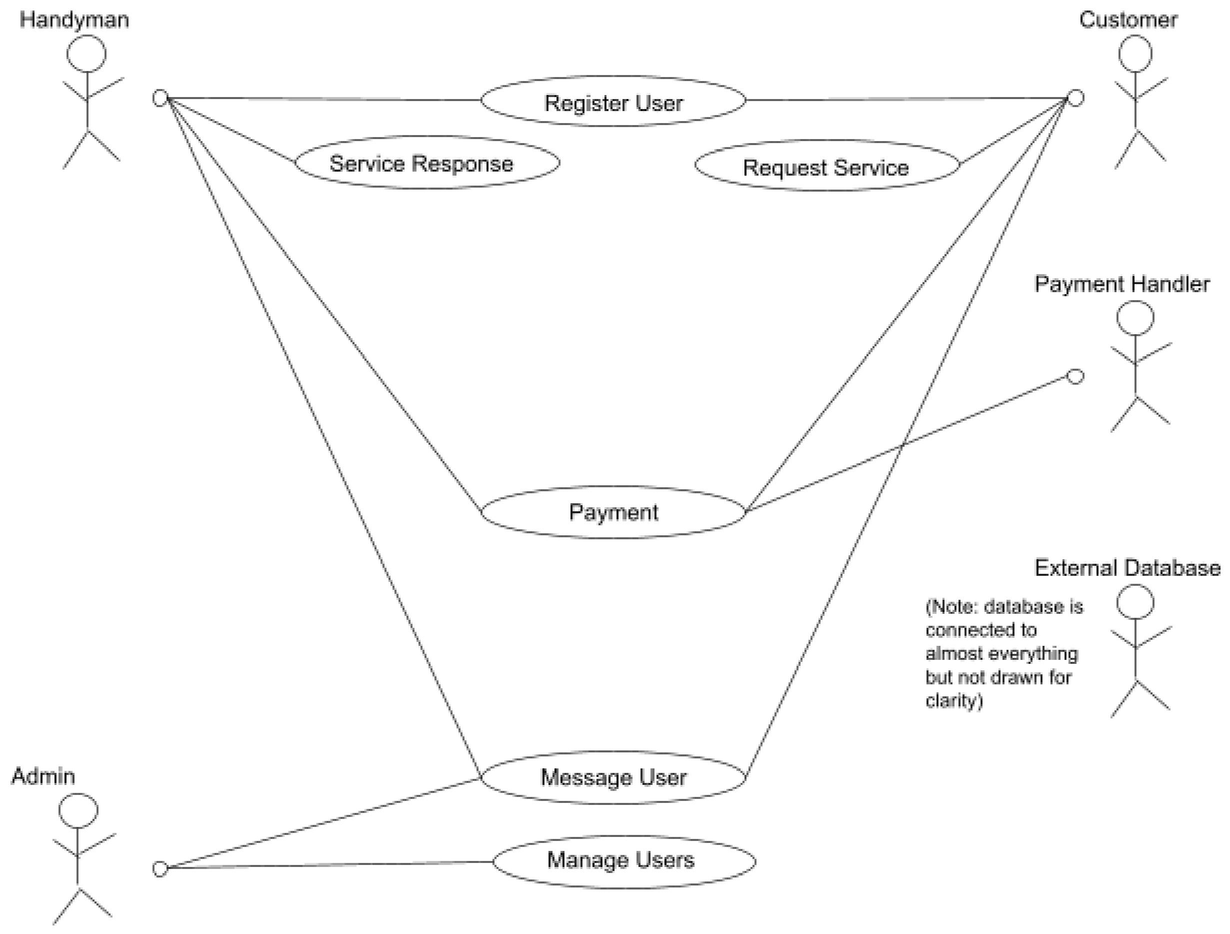
CSC 581 [PROJECT REPORT – 2]

I. PARTICIPANTS

Sajeel Mohammed Abdul
Cole Allen
Ananya Kakumanu
Shoaibuddin Mohammed
Malika Hafiza Pasha
Lokesh Purohit
Christian Quintero

II. UPDATED USE CASE DIAGRAM



III. <u>UPDATED USE CASES WITH BASIC FLOW</u>

1. Register User (Combination of sign in, sign out, edit account info, and create account)
   - Description: This use case describes the process of registering a user (***handyman*** or ***customer***) with the app. The details of the user must be saved to the ***external database*****,** and the registered user must be signed in to access the app's services.

   - Basic Flow
     1. The user is prompted to sign up or sign in
     2. The user selects "sign up" and is prompted to enter an email address. (A1)
     3. The existence of the email address is checked by the external database to verify its validity
     4. On acceptance of the email address, the user inputs their personal information, such as name, sex, age, address, payment information, and password
     5. The user is prompted to enter their password again to confirm their new password
     6. If the confirmation password matches the initial password, the user's account is created successfully
     7. User is then returned to the sign-in/sign-up menu
   - Alternate Flow
     i. <A1 – If the user selects "sign in", they are prompted for their email and password. If the latter matches the information saved in the database, the user is granted access to their account. If the user is signing in for the first time, they are directed to their profile and prompted to (optionally) fill it out. (A2) When done with the app, they may sign out by clicking the relevant button. A dialog box then appears with a brief snippet of the appointments pending with their timing, with a question asking confirmation of the decision to sign out? On selecting "YES", the user is signed out, taking them back to the sign in/sign up menu.>
     ii. <A2 – The user selects different fields like profile picture, name, address, age, payment information. On selecting a specific field, the user is allowed to update the information in that field. After the changes are made, the user selects the "SAVE" button at the bottom. On clicking on "SAVE", a dialog box appears to seek confirmation of the desire in making the change in information. On clicking "YES", the new information is fed to the external database, to replace the previous information, where it is saved in that user profile. The user is taken back to the previous screen (main edit profile page) where all the profile fields are displayed that can be chosen to edit information.>

2. Request Service (Combination of Create Service Request, View Handymen, Filter Handymen, and View Service Request, Close Service Ticket, Cancel Appointment)
    - Description: This use case describes the process of a ***customer*** requesting a service from a ***handyman***. The ***customer*** must first select a particular ***handyman***, then send a request to the latter; requests are saved to the ***external database***.

    - Basic Flow:
        1. A customer navigates to their dashboard(A1) and clicks the "Create Service Request" button.
        2. The customer inputs a description of the requested service and selects the desired appointment date and time. (A2)
        3. The system displays all the available handymen with their ratings and other information
        4. The customer selects a preferred handyman and writes a service request with a text explanation, optionally accompanied by images.
        5. Upon the customer clicking "Send", the service request is saved to the external database and sent to the selected handyman. Both the handyman and the customer can view the service request at any time. (A3)

    - Alternative Flow:
        i. <A1 – The user browses through handymen and finds one with the qualifications that they desire. The customer then starts a service request from that handyman's profile.>
        ii. <A2 – If the customer has preferences or does not see any handymen that they would like to choose, they can set filters for the list of displayed handymen to narrow down the results to best fit their criteria.>
        iii. <A3 – The customer or handyman may cancel a request if they desire.>

3. Service Response (Combination of Send Quote and Accept Service Quote; needs handyman response added as well, maybe also Review Appointment, Close Service Ticket, Cancel Appointment)
    - Description: This use case describes the process of the ***handyman*** viewing a service request from a ***customer*** and giving the ***customer*** a quote based on that service request, and the ***customer*** accepting the service quote and setting an appointment time.

- Basic Flow
    1. The handyman receives a service request, proceeds to check the description of the service required, appointment date and time requested, and location distance, and sends a service quotation.(A1)
    2. On the customer's acceptance of the quotation(A2), the system asks for payment confirmation.
    3. The customer fills in the payment information to complete the transaction setup(A3), in which the system confirms the service appointment to both the customer and the handyman.(A4)

- Alternative Flow:
    i. <A1 – The handyman may instead opt to deny the service request.>
    ii. <A2 – The customer may instead opt to deny the quotation or message the handyman to discuss pricing.>
    iii. <A3 – If the customer plans to instead pay by cash or check, they must note this here to continue without putting in online payment information; the handyman then has the option to confirm or deny payment method acceptability, and thus confirm the appointment.>
    iv. <A4 – The customer and the handyman both have the option to review appointment details at any time and cancel the appointment if unforeseen circumstances or changes in plans require the service to be postponed or canceled.>

4. Payment (Combines Edit Payment info and Confirm payment)
    - Description: This use case describes the process of the ***customer*** paying the ***handyman*** for their services. When creating or editing their accounts, the ***customer*** and the ***handyman*** both have the option to put in their payment information for current or future purchases or payments of services. After payment information is successfully updated, it gets updated in the ***external database***. When completing a service request, the ***payment handler*** will handle the transaction using the saved payment information of both parties.

    - Basic Flow
        1. Once a job is completed to the customer's satisfaction, the customer and handyman are both given a prompt to confirm payment. (A1)
        2. When the customer and handyman have both confirmed payment, stored payment information is passed along to the external payment handler; payment is then charged to the customer, deposited to the handyman, and confirmed as completed. (A2)

3. Once payment is confirmed as completed, the transaction is closed and logged in the database. Receipts are given to both users.
4. Users are returned to their main menus.

- Alternate Flow
  - i. <A1 – If a user has not entered any payment information yet, they are given the option to either enter and save payment details to their account, or skip to exchanging cash payment instead.>
  - ii. <A2 – If payment information is not correct, error messages are displayed to both users, and the offending user(s) is given the option to correct their financial information. If this is not possible, options are given to both users to exchange cash payment instead and confirm payment without using the third party payment system.>

5. Message User

- Description: This use case describes the process of an ***admin, customer*** or ***handyman*** sending a message to another user. Messages can be used to communicate about service requests, appointments, or other topics. The ***external database*** also holds these messages.

- Basic Flow

  1. A user (must be signed in) navigates to their inbox.
  2. The logged in user selects the recipient they want to communicate with (could be from a list of previous conversations or by searching for a user).
  3. User composes the message in the provided text box or editor.
  4. User clicks 'Send' or equivalent button to send the message to the corresponding recipient.
  5. System saves and sends the message. (A1)
  6. System updates the external database with the new message.
  7. The receiving user gets a notification about the new message. (A2)

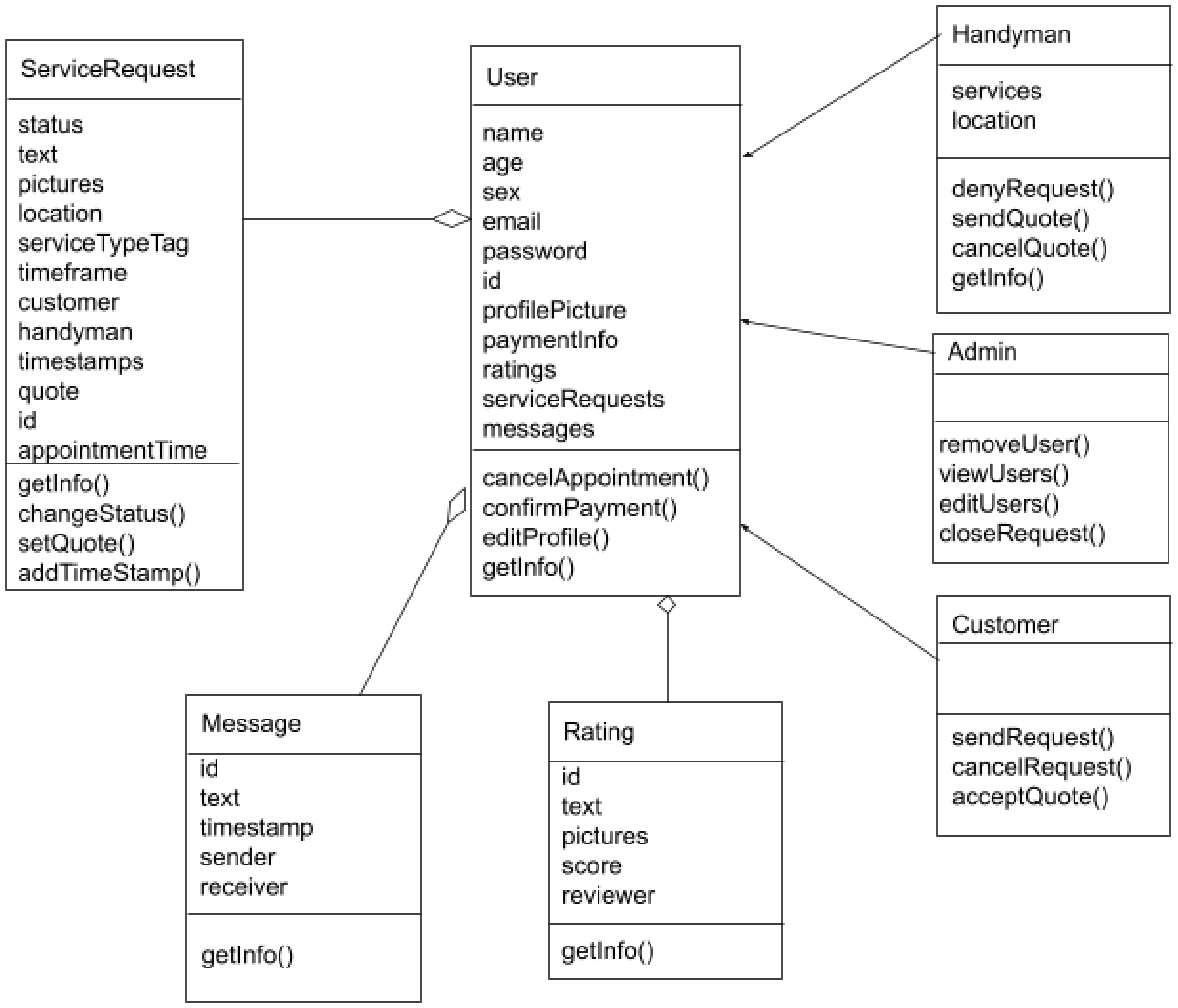
- Alternate Flow
  - i. <A1 – If the recipient has blocked the sender, the message will not be delivered. Instead, a notification will be sent to the sender telling them that the message cannot be delivered. Admins cannot be blocked.>
  - ii. <A2 – If the recipient has their notifications turned off, they will not get any notifications about new messages until they open the app.>

6. Manage User (derived from Remove users)
    - Description: This use case describes the process of an ***admin*** viewing and making privileged actions with regards to ***customers*** and ***handymen***.
    - Basic Flow
        1. An admin signs into the app via their own special portal using admin credentials.
        2. The admin is presented with a special menu (admin dashboard), where they can see admin notifications, messages, and a filterable list of users.
        3. Admin selects a particular user, either by searching or browsing.
        4. Admin selects the option “Remove User”
        5. Admin is prompted to confirm that they are sure they want to remove this user.
        6. Admin is prompted for their admin credentials.
        7. Admin is prompted for a reason as to remove the user; they have the option to upload images along with text.
        8. Admin is asked credentials a second time upon clicking “Confirm Removal”.
        9. All pending appointments with the given user are canceled and notifications sent to relevant users. The user is removed from the system. An email is sent to both admin and user as a record of removal. Finally, the removal reason and any associated images are stored in the database.

IV. <u>**LIST OF POSSIBLE CLASSES**</u>

- User: This class represents any user of the system, including customers, handymen, and admins.
- Handyman: This class represents a handyman who provides services to customers through the system.
- Admin: This class represents an administrator of the system, who has special privileges such as viewing and managing user accounts.
- Customer: This class represents a customer who requests services from handymen through the system.
- ServiceRequest: This class represents a request from a customer for a service from a handyman.
- Rating: This class represents ratings assigned to the user after every service request is completed.
- Message: This class represents a single message sent from one user to another.

V. CLASS DIAGRAM



Notes:

1. Admin and Customer intentionally have no unique attributes.
2. Database and payment methods are assumed to be handled by library methods and thus no classes are presented for them here.