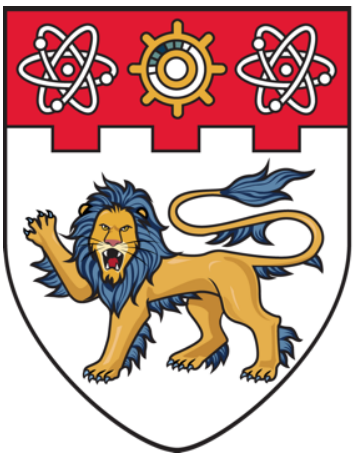


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ2006 Software Engineering Lab #3

SS6

Group Name:
ifandonlyif

Group Members:
Chua Kok Liang
Grace Mok Jie Qi
Jozua Heng Yi Jie
Nicholas Yeo Ming Jie
Royce Ang Jia Jie
Shahrin Chua Zong Da

Contents

Functional Requirements

1. System must be hosted on an Android OS device with Android 4.0 and above
2. System must have access to an Internet Connection
3. System must communicate with the backend server
4. System requires the user to login to use the system
    - 4.1. If the user does not have an account, the system must allow the user to create an account
        - 4.1.1. During the registration process, if an email has already been taken by another user, a popup must appear to tell the user the email is invalid until a unique email is entered
5. System must allow users to edit their account details such as username and profile picture
6. System must be able to retrieve information regarding all available one-for-one deals
    - 6.1. Information includes Name, Date, Vendor, Location, Category, and Terms and Conditions of deals
7. System must be able to display all retrieved one-for-one deals
    - 7.1. System must be able to display the deals according to their respective categories
    - 7.2. System must be able to allow the user to filter deals based on keywords in the name of the deal within their respective categories
8. Users must be able to select a deal for matching
    - 8.1. User must specify their preferred Locations of the deal
        - 8.1.1. User must select at least 1 Location from a list of given locations in Singapore by ticking checkboxes
    - 8.2. After selecting a deal, the system will start to find a match for the user
9. System must display a popup to inform users that the system is finding a match for the users
    - 9.1. The user can cancel the matching process by tapping the cancel button
        - 9.1.1. When the user taps on the cancel button, the popup is dismissed
    - 9.2. Once the user is matched with another user, the popup is dismissed by the system
10. System must match two users who select the same deal with at least one identical location choice
11. System must display a popup to inform both users that they have been successfully matched
    - 11.1. User can tap on the chat button to start chatting with the matched user
12. The system must create a chat room for the two users after one user taps on the chat button in the popup
13. System must allow the two users to send messages to one another
    - 13.1. System must notify a user when they receive a message from a match while they are outside of the system
        - 13.1.1. The notification is displayed in the notification bar

13.1.1.1. System must allow the user to tap on the notification to bring the user to the chat room

13.1.2. The notification contains the username of the user who sent the message and the content of the message

14. System must allow users to block another user that they have a chat room with
    - 14.1. System will close the chat room immediately for the user that tapped the block button
    - 14.2. System will never match the two users again once one user blocks the other
    - 14.3. System will no longer display the username of the blocked user in the Matches Page for the user that tapped the block button, and vice versa
15. System must allow users to delete a chat room with another user
    - 15.1. System will no longer display the username of the matched user in the Matches Page for the user that tapped the delete chat room button
    - 15.2. System will still display the username of the user that tapped the delete chat room button for the other user
    - 15.3. When the chat room is deleted, the messages are archived
        - 15.3.1. When these two users are matched together in the future, their chatroom will still contain the old messages

Non-Functional Requirements

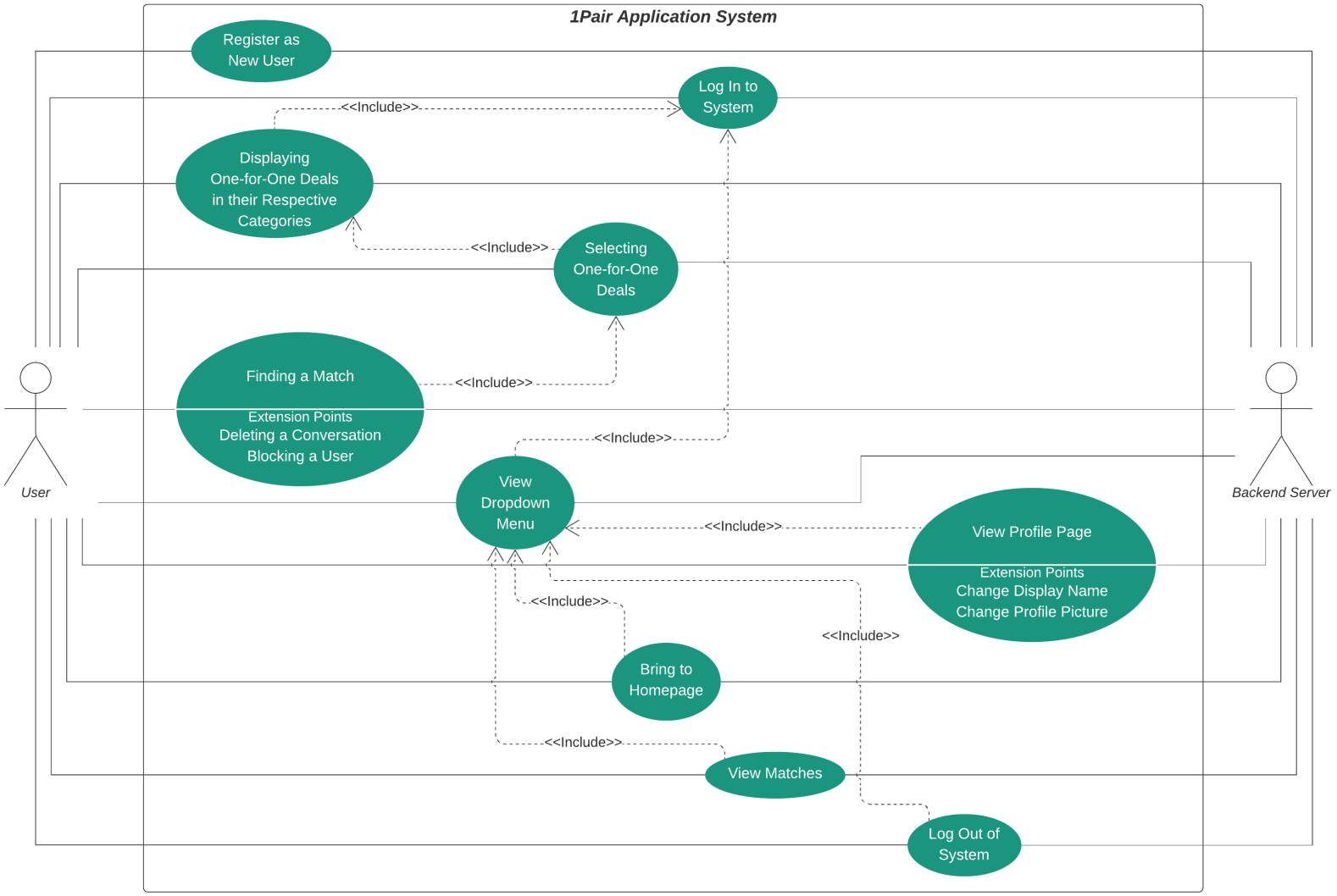
Usability	1. The registration process should be simple enough such that it should not take more than 10 minutes for the user 2. Login for new users should be immediately after registration 3. Main menu must be accessible from all pages 4. All pages must have a function that allows users to go back to previous pages
Reliability	1. Passwords must never be visible at any time during the registration and login process and thereafter unless toggled by users 2. Upon failing to update the account details, previous account details are not changed 3. The system must not be unavailable to users for more than 24 hours
Performance	1. After a system reboot or update, the full system functionality must be restored within 5 minutes 2. After an account is created, the details of the account should be recorded in the Backend Server within 30 seconds 3. All queries from the Backend Server should be completed within 30 seconds
Supportability	1. The system must be accessible to all android smartphone users above Android System 4.0 (IceCreamSandwich)

Data Dictionary

Term	Definition
User	A user refers to one who uses the application to find a match for one-for-one deals.
System	A system refers to the "1Pair" application on Android.
Account	An account refers to all the relevant information (such as profile picture, username, password, and email) and a unique identification number tagged to a registered user.
One-For-One deal	A one-for-one deal refers to a deal offered by a vendor in Singapore where users can get two items for the price of one.
Vendor	A vendor refers to a shop or franchise offering one-for-one deal(s).
Terms and Conditions	Terms and conditions of a one-for-one deal refers to the rules provided by the vendors regarding the one-for-one deal. The terms and conditions are to be abided by for the user to claim the one-for-one deal.
Deal Location	A deal location refers to the area where a specific one-for-one deal is available as stated in the deal's terms and conditions.
User Preferred Location	A user preferred location refers to an area selected by the user when selecting a deal.
Match	A match refers to a pair of users that are grouped together based on their preferred location for the same one-for-one deal.
Chat Room	A chat room refers to a platform where matched users can send messages to one another.
Waitlist	A waitlist refers to a dynamic queue where users are added to when they tap the "Find a match" button for a given deal.
Category	A category refers to a classification of the deals available in the system. The four categories are Food, Entertainment, Retail, and Others.

Archive	An archive refers to the chat history between matched users.
Registration	A registration refers to the process of setting up an account.
Backend Server	A backend server refers to a remote online database that stores information such as account details and deals.
Query	A query refers to a request to retrieve relevant data from the Backend server.

Complete Use Case Diagram



Complete Use Case Model

Use Case 1: Register as New User

Actor:	System, User, Backend Server
Description:	User registers as a new user in the system.
Entry-Conditions:	1. User has the system open and running. 2. System must be connected to the backend server.
Exit-Conditions:	1. System displays the Homepage.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. System displays the Landing Page. 2. User clicks the “Register” button. 3. User enters their preferred Display Name. 4. User enters their Email. 5. User enters their preferred Password which is at least 6 characters long. 6. System will display a popup with the text “Registering User, This may take a while”. 7. The user’s account information is sent to the backend server and saved there. 8. The user is automatically logged into the system. 9. System displays the Homepage.
Alternative Flows:	AF-S3: If the entered Email already exists in the server 1. The system will display a temporary message that says “Invalid Email/Password”. AF-S4: If the Password is not at least 6 characters long 1. The system will display a temporary message that says “Invalid Email/Password”.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 2: Log In to System

Actor:	System, User, Backend Server
Description:	User Logs In to the System
Entry-Conditions:	1. User has the system open and running. 2. System must be connected to the backend server.
Exit-Conditions:	3. System displays the Homepage.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. System displays the Landing Page. 2. User clicks the “Log In” button. 3. User enters their Email. 4. User enters their Password. 5. System will display a popup with the text “Logging In User, This may take a while”. 6. System displays the Homepage.
Alternative Flows:	AF-S2: If the entered Email does not exist in the server 1. The system will display a temporary message that says “Invalid Email/Password”. AF-S3: If the Password associated with the Email is invalid 1. The system will display a temporary message that says “Invalid Email/Password”.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 3: Displaying One-for-One Deals in their Respective Categories

Actor:	System, User, Backend Server
Description:	System displays all valid one-for-one deals for each specific category in the system.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	3. All valid one-for-one deals for the selected category are fully displayed.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. User logs in. 2. User selects the category he wants to view in the main page. 3. The system queries the backend server for the name, date, vendor, location, category, and terms and conditions of deals of all valid one-for-one deals for the deals belonging to the selected category. 4. The system displays the image and the name of all valid deals in the selected category in a ListView.
Alternative Flows:	AF-S4: If there are no valid one-for-one deals in the selected category 1. The system will display a blank page with a pop-up saying “Sorry there are no deals currently.”.
Exceptions:	-
Includes:	1. Use Case 2
Special Requirements:	-
Assumptions:	1. The order of displayed deals is in the order of how they are being retrieved. 2. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 4: Selecting One-for-One Deals

Actor:	System, User, Backend Server
Description:	User browses through a filtered list of displayed one-for-one deals based on the category selected and selects a desired deal.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server. 3. Deals must be displayed in the system.
Exit-Conditions:	4. User placed in waitlist of selected deal on the server.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. Displaying One-for-One Deals in their respective Categories (Use Case 3) has been invoked. 2. User selects the preferred one-for-one deal based on the displayed deals. 3. System will display name, date, vendor, location, category, and terms and conditions of the selected deal. 4. User clicks on the “Find a match” button. 5. System will display a popup with a scrollable list of all MRT stations, with each location having their own checkbox. 6. User selects up to 5 preferred locations by clicking each location’s checkbox. 7. System acknowledges the user’s selections and sends it to the backend server. 8. System displays a popup displaying the text “Finding you a match! This may take a while” to inform the user that he has been placed on the waitlist.
Alternative Flows:	AF-S6: If there are no preferred locations 1. The user exits the location selection popup. 2. The system returns to Step 3. AF-S8: If the user presses the “Cancel” button 1. The system informs the backend server the cancellation of the matching. 2. The popup closes and the system returns to Step 3.
Exceptions:	-
Includes:	1. Use Case 3
Special Requirements:	-
Assumptions:	2. The user knows which outlets the deal is available at. 3. The user knows which MRT stations are near the outlets. 4. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 5: Finding a Match

Actor:	System, User, Backend Server
Description:	The system matches two users who have selected the same deal and selected at least one identical location.
Entry-Conditions:	1. The waitlist on the server must contain at least 1 user. 2. A new user is placed on the waitlist.
Exit-Conditions:	1. User has been successfully matched with another user from the waitlist. 2. Both users that are matched are removed from the waitlist.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. Selecting One-for-One Deals (Use Case 4) has been invoked. 2. System receives the id of both users, the deal id and the location names that both users have in common from the server. 3. System notifies both users that they have been successfully matched through a popup. 4. The first user clicks on the “Chat” button in the popup to prompt the creation of the chatroom and enters the chatroom. 5. The second user clicks on the “Chat” button in the popup to enter the same chatroom.
Alternative Flows:	AF-S4: If there are no users in the filtered waitlist to be matched. 1. System will continue to display the popup in Use Case 4, Step 8 until user is successfully matched.
Exceptions:	-
Includes:	1. Use Case 4
Special Requirements:	-
Assumptions:	1. The user does not press the “Cancel” button (from Use Case 4 AF-S8) between Steps 1-6. 2. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 6: Deleting a Conversation

Actor:	System, User, Backend Server
Description:	Archives one user's side of the chatroom.
Entry-Conditions:	1. Two users have successfully matched and have entered the same chatroom.
Exit-Conditions:	2. When the user clicks on the "Delete Conversation" button.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. Finding a Match (Use Case 5) has been invoked. 2. Users send messages in the chatroom. 3. User clicks on the "Delete Conversation" button. 4. System closes the chatroom. 5. System notifies server of the deletion. 6. The chatroom will no longer be available for the user in the system.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use case 7: Blocking a User

Actor:	System, User, Backend Server
Description:	Allows a user to block a matched user.
Entry-Conditions:	1. Two users have successfully matched and have entered the same chatroom.
Exit-Conditions:	2. When the user clicks on the “Block” button.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. Finding a Match (Use Case 5) has been invoked. 2. Users send messages in the chatroom. 3. User clicks on the “Block” button. 4. System closes the chatroom for both users. 5. System notifies server of the deletion of the chatroom. 6. The chatroom will no longer be available for both users in the system. 7. Both users will never match with each other again.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 8: View Dropdown Menu

Actor:	System, User, Backend Server
Description:	System displays the Dropdown Menu.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	3. System displays the Dropdown Menu.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. User clicks the Dropdown Menu icon. 2. System displays the Dropdown Menu with the options “Profile”, “Homepage”, “Matches” and “Log Out”.
Alternative Flows:	-
Exceptions:	-
Includes:	1. Use Case 2
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 9: View Profile Page

Actor:	System, User, Backend Server
Description:	System displays user's profile page.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	3. System displays the Profile Page.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. View Dropdown Menu (Use Case 8) has been invoked. 2. User clicks on the "Profile" button. 3. System displays the Profile Page, which contains the display picture and username of the user.
Alternative Flows:	-
Exceptions:	-
Includes:	1. Use Case 8
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 10: Change Display Name

Actor:	System, User, Backend Server
Description:	User changes their Display Name.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	1. User has changed their Display Name. 2. System displays the Profile Page.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. View Profile Page (Use Case 9) has been invoked. 2. User clicks on the “Change Display Name” button. 3. System displays a page for the User to enter their new display name. 4. User enters their new display name and clicks the “Save” button. 5. System will display a popup with the text “Updating Display Name, This may take a while”. 6. System sends the updated Display Name to the server. 7. System displays the Profile Page again with the newly updated Display Name.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 11: Change Profile Picture

Actor:	System, User, Backend Server
Description:	User changes their Profile Picture.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	1. User has changed their Profile Picture. 2. System displays the Profile Page.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. View Profile Page (Use Case 9) has been invoked. 2. User clicks on the “Change Profile Picture” button. 3. System displays the User’s photo gallery. 4. User selects their preferred photo and clicks the “Crop” button. 5. System will display a popup with the text “Uploading Image… This may take a while”. 6. System sends the updated Profile Picture to the server. 7. System displays the Profile Page again with the newly updated Profile Picture.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 12: Bring to Homepage

Actor:	System, User, Backend Server
Description:	System displays user's profile page.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	3. System displays the Homepage.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. View Dropdown Menu (Use Case 8) has been invoked. 2. User clicks on the "Homepage" button. 3. System displays the Homepage.
Alternative Flows:	-
Exceptions:	-
Includes:	1. Use Case 8
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

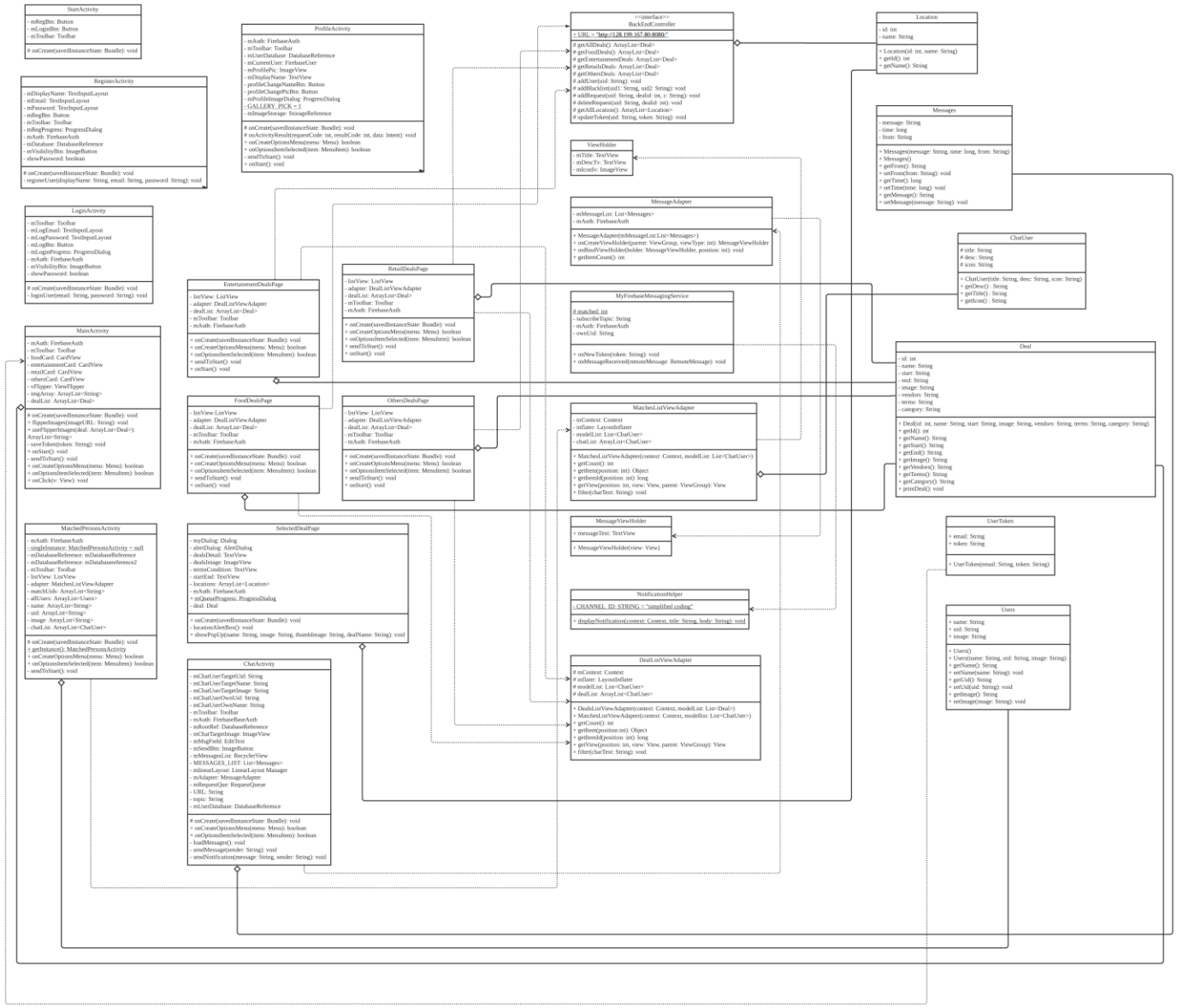
Use Case 13: View Matches

Actor:	System, User, Backend Server
Description:	System displays all chatrooms the User is in
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	3. System displays the specific chatroom the User wants to view.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. View Dropdown Menu (Use Case 8) has been invoked. 2. User clicks on the "Matches" button. 3. System displays a page with the Display Pictures of users who have matched with the user and the names of the users if the chatroom is not deleted. 4. User clicks on preferred chat to view the dialogue inside the chatroom.
Alternative Flows:	
Exceptions:	-
Includes:	1. Use Case 8
Special Requirements:	-
Assumptions:	1. System must have access to an Internet Connection.
Notes and Issues:	-

Use Case 14: Log Out of System

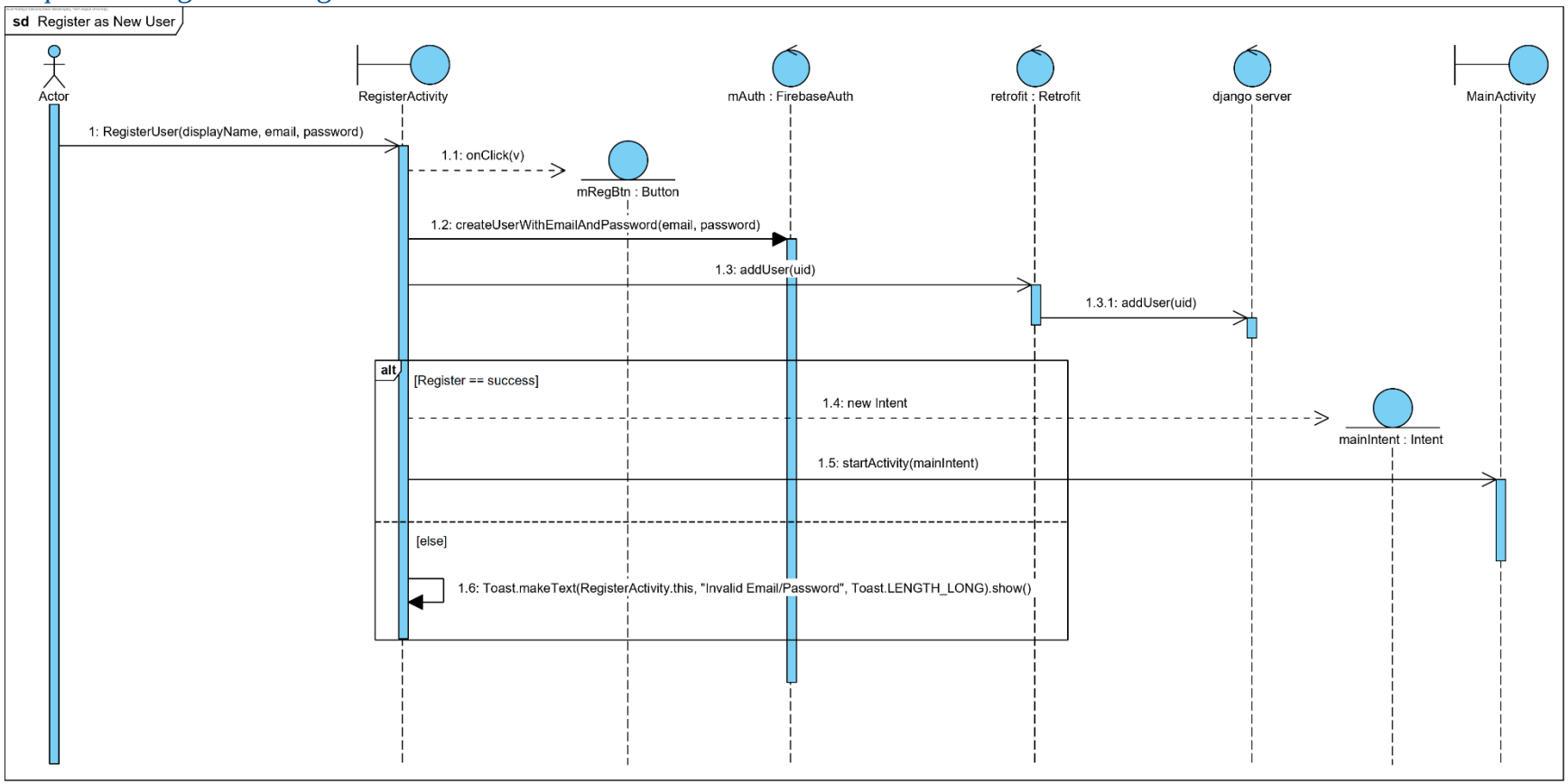
Actor:	System, User, Backend Server
Description:	System Logs the User out of the System.
Entry-Conditions:	1. User must be logged in. 2. System must be connected to the backend server.
Exit-Conditions:	3. System Logs the User out of the System.
Priority:	-
Frequency of Use:	-
Flow of Events:	1. View Dropdown Menu (Use Case 8) has been invoked. 2. User clicks on the “Log Out” button. 3. System displays the Landing Page.
Alternative Flows:	-
Exceptions:	-
Includes:	1. Use Case 8
Special Requirements:	-
Assumptions:	2. System must have access to an Internet Connection.
Notes and Issues:	-

Complete Class Diagram

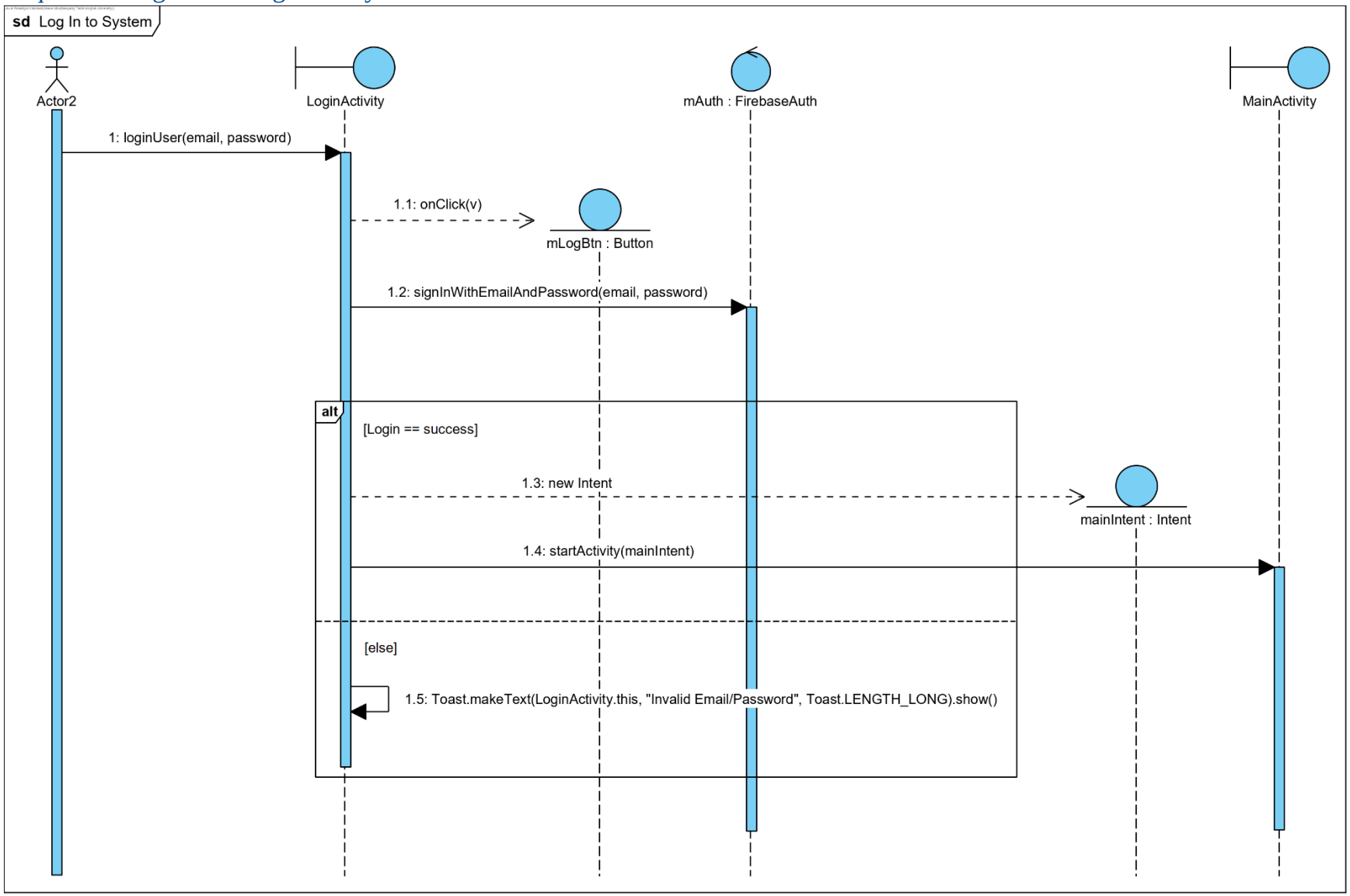


Complete Sequence Diagrams

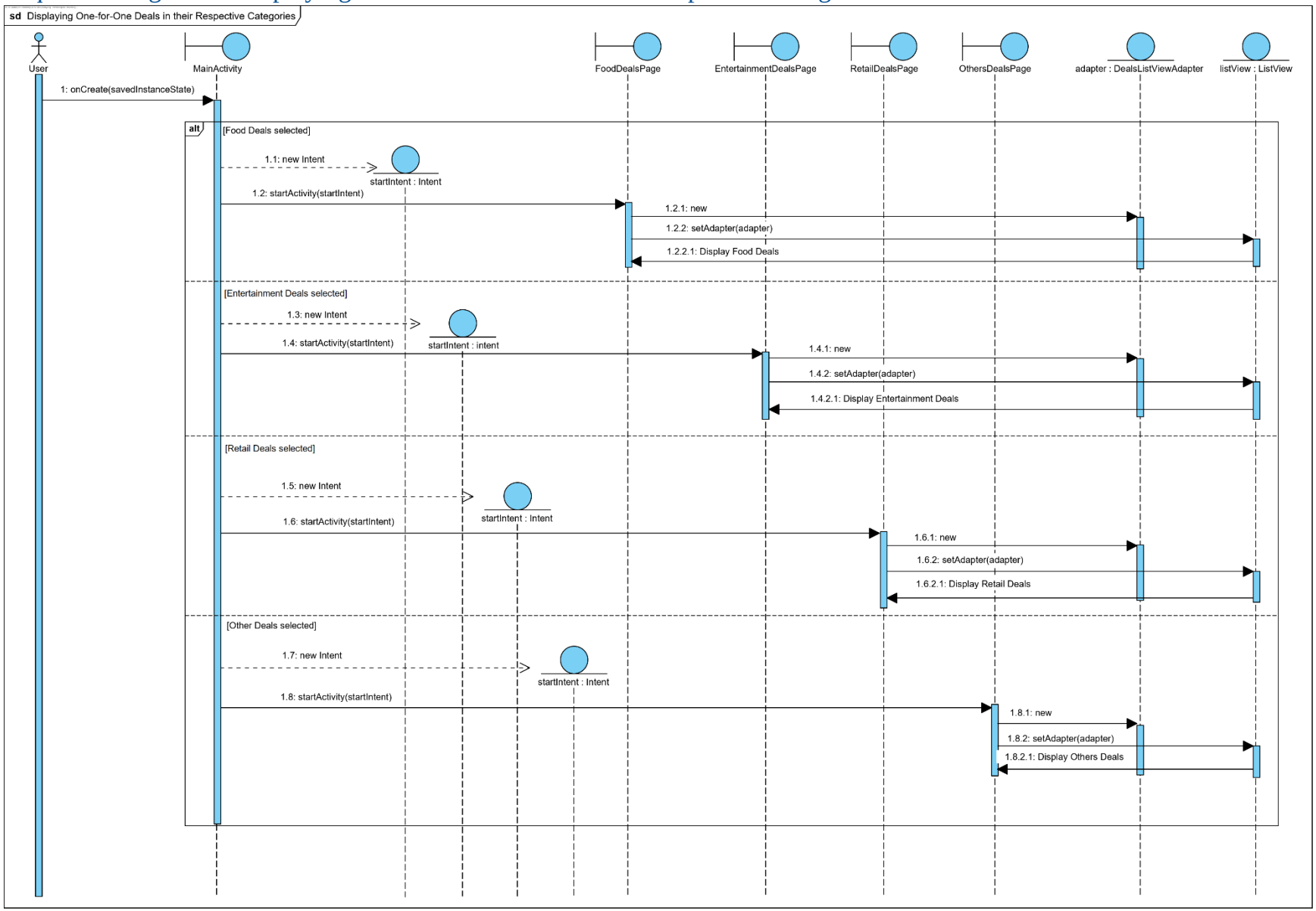
Sequence Diagram 1: Register as New User



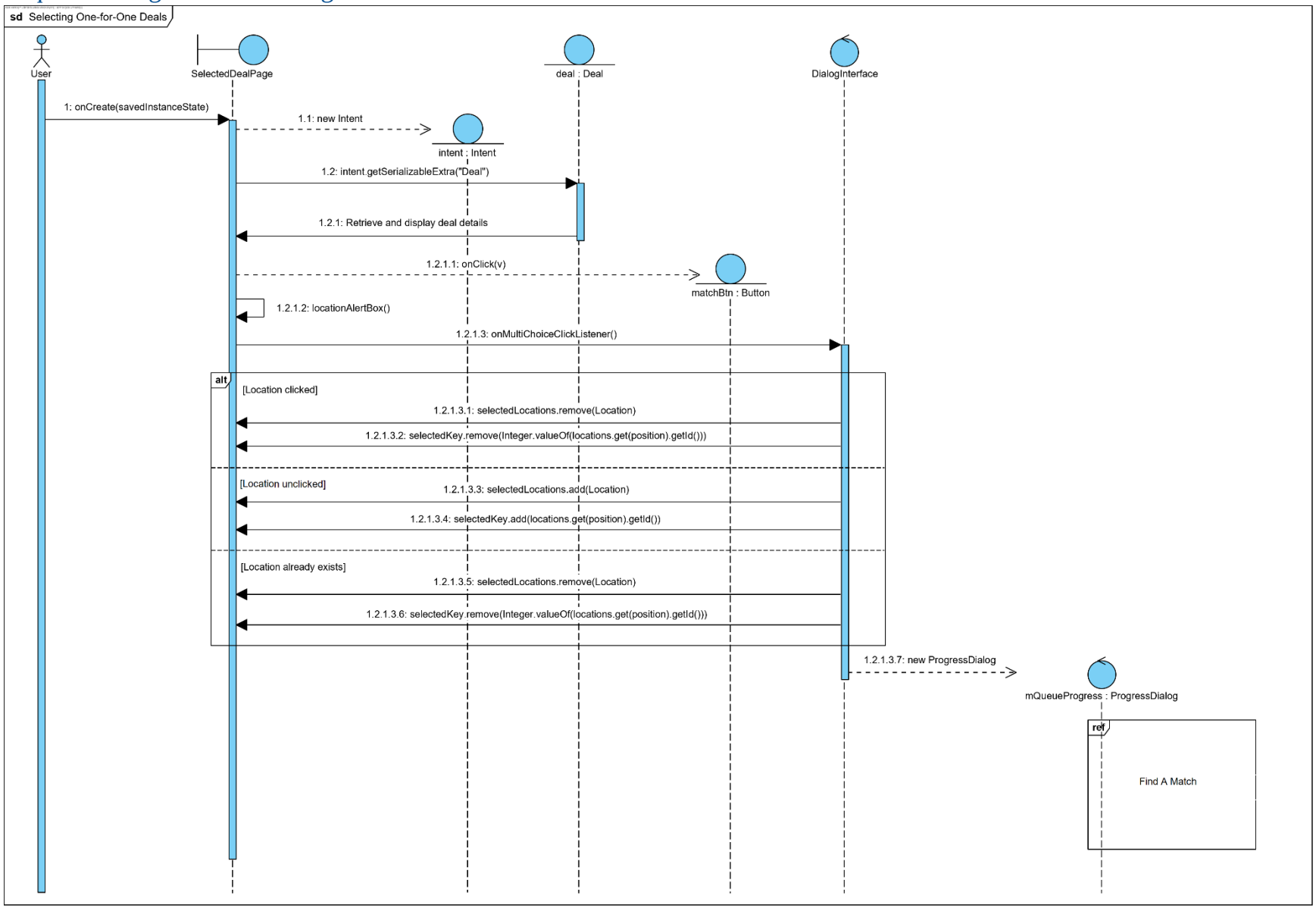
Sequence Diagram 2: Log In to System



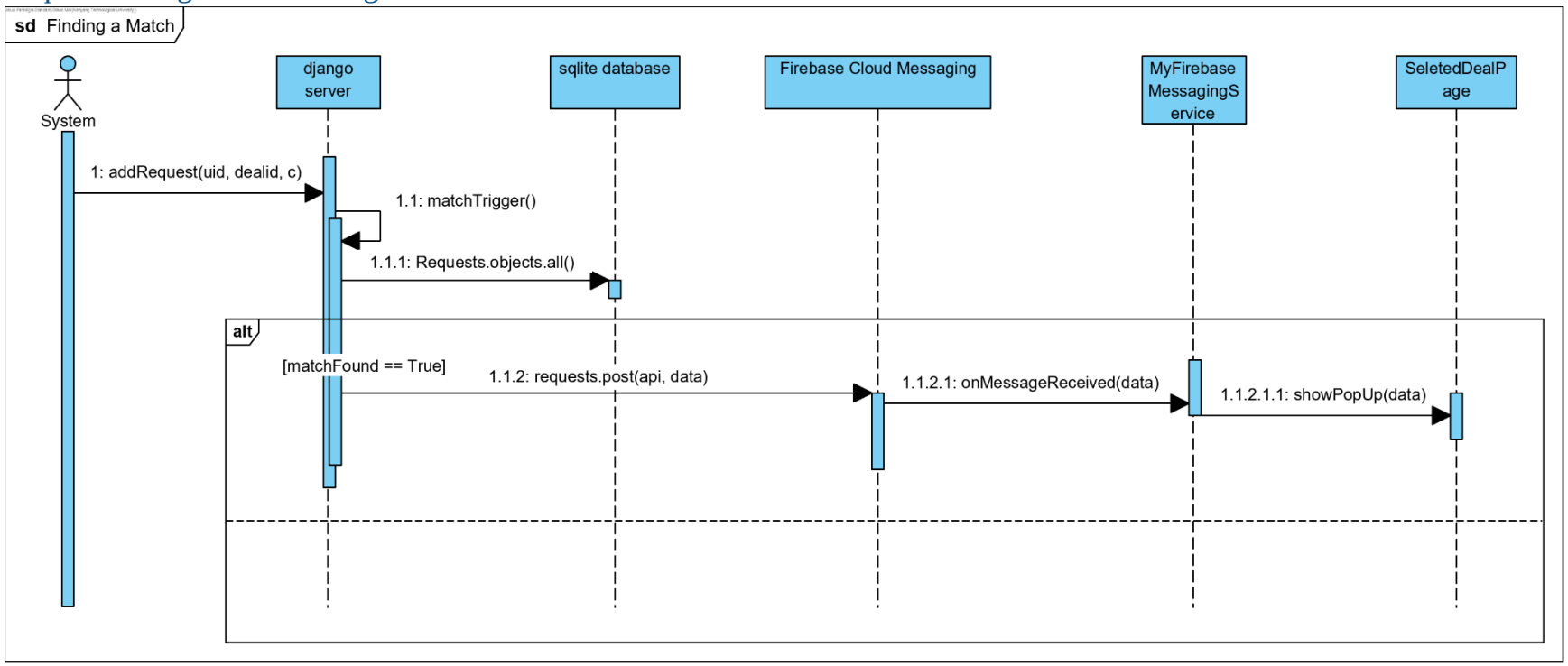
Sequence Diagram 3: Displaying One-for-One Deals in their Respective Categories



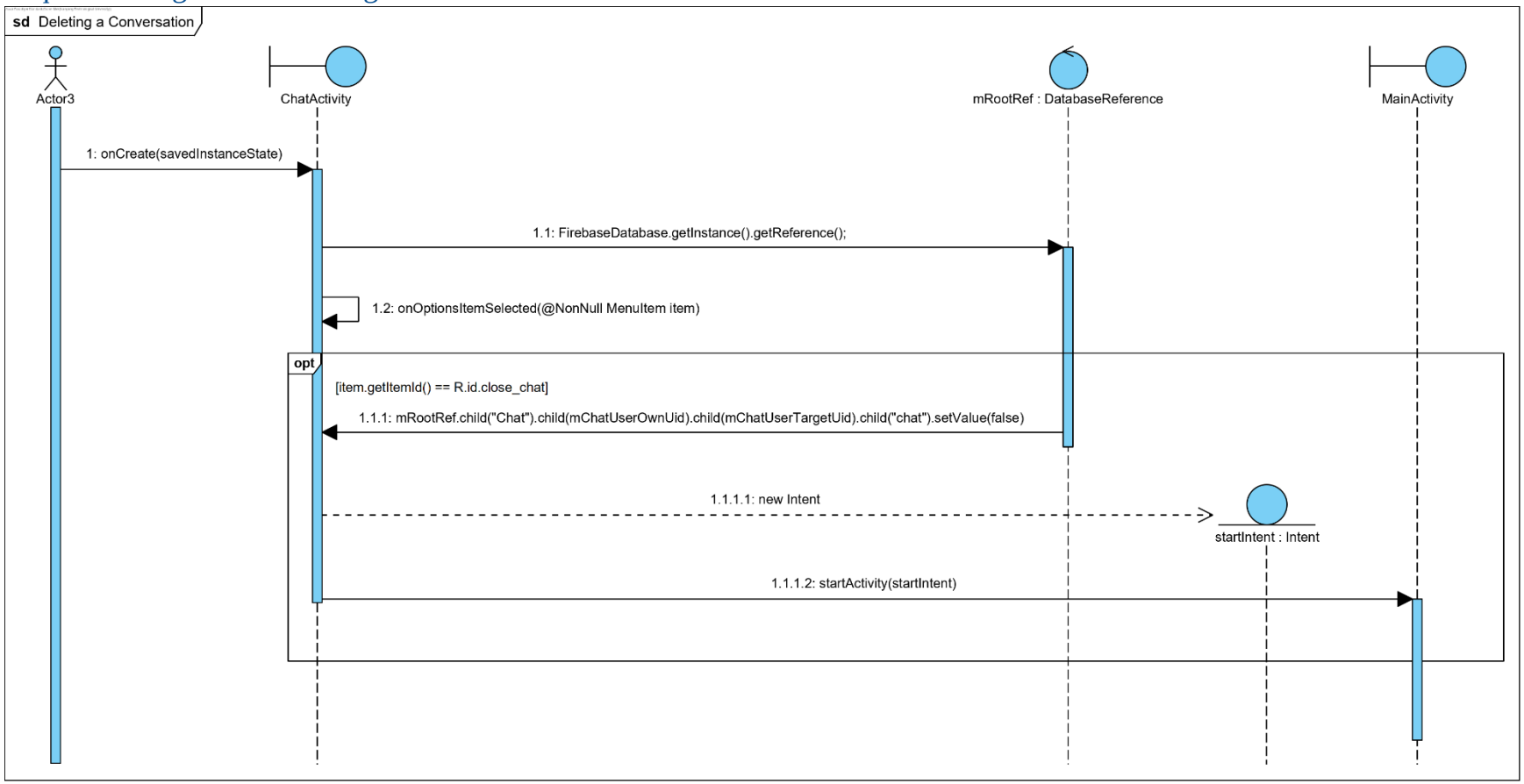
Sequence Diagram 4: Selecting One-for-One Deals



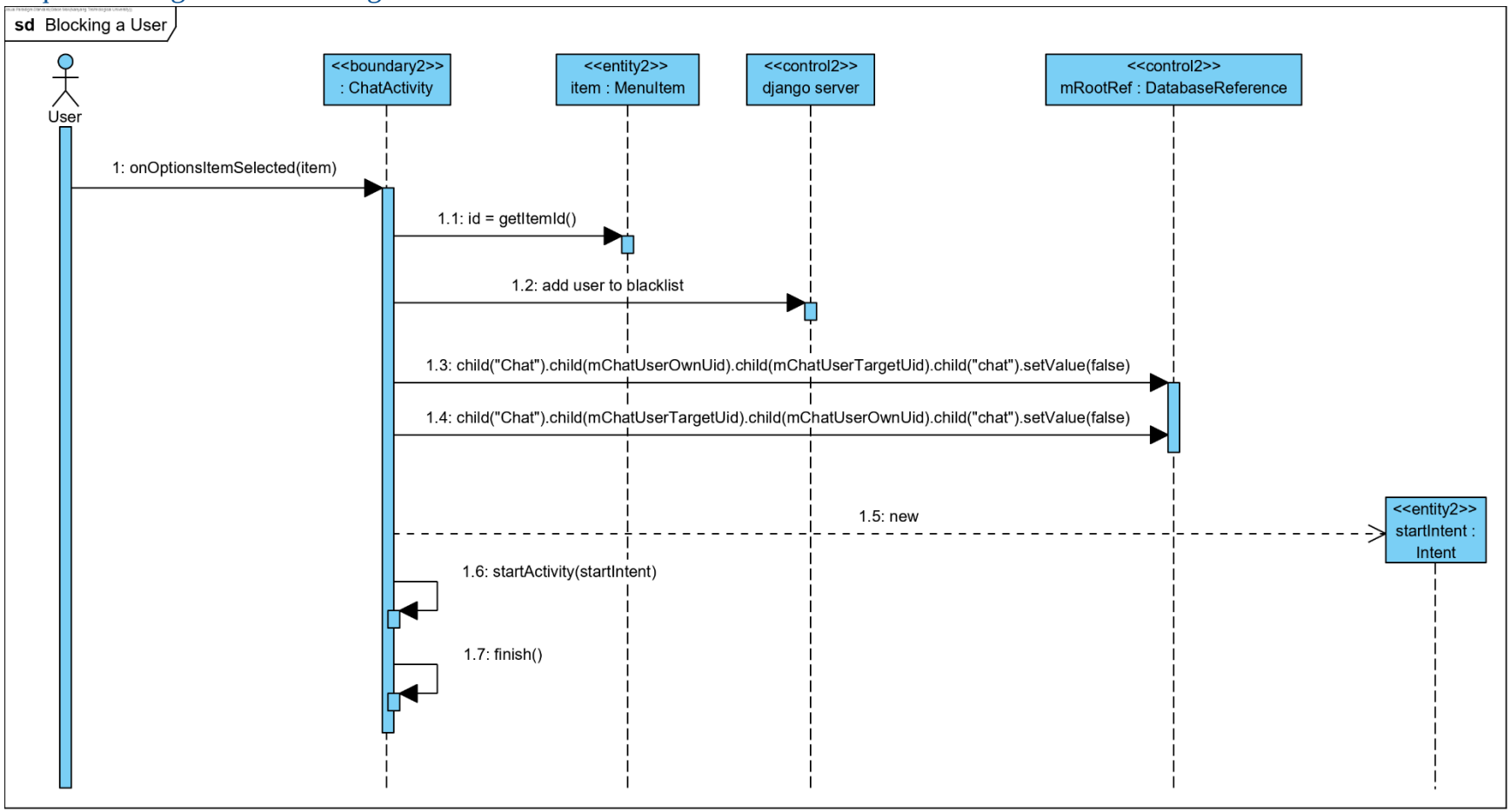
Sequence Diagram 5: Finding a Match



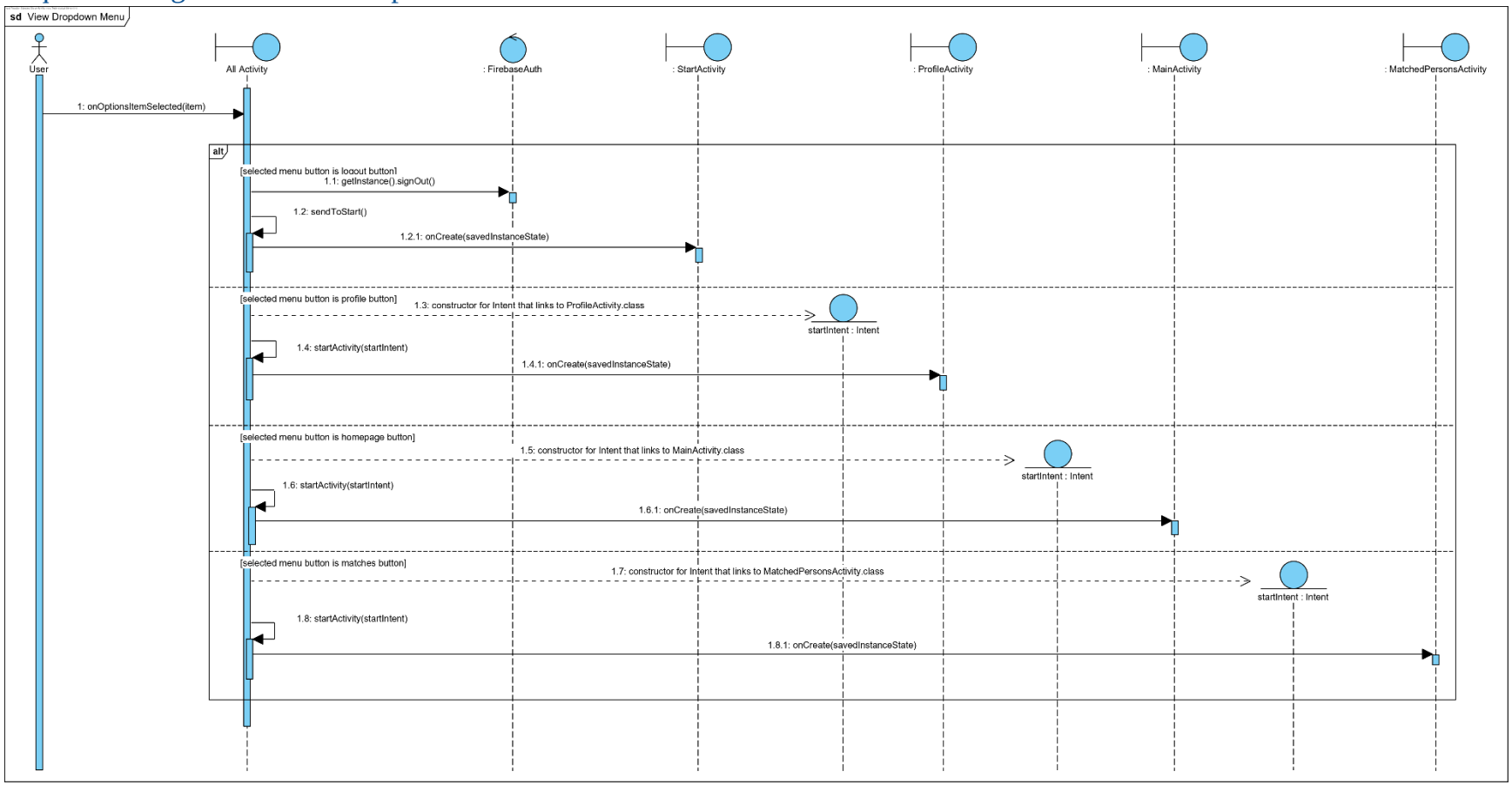
Sequence Diagram 6: Deleting a Conversation



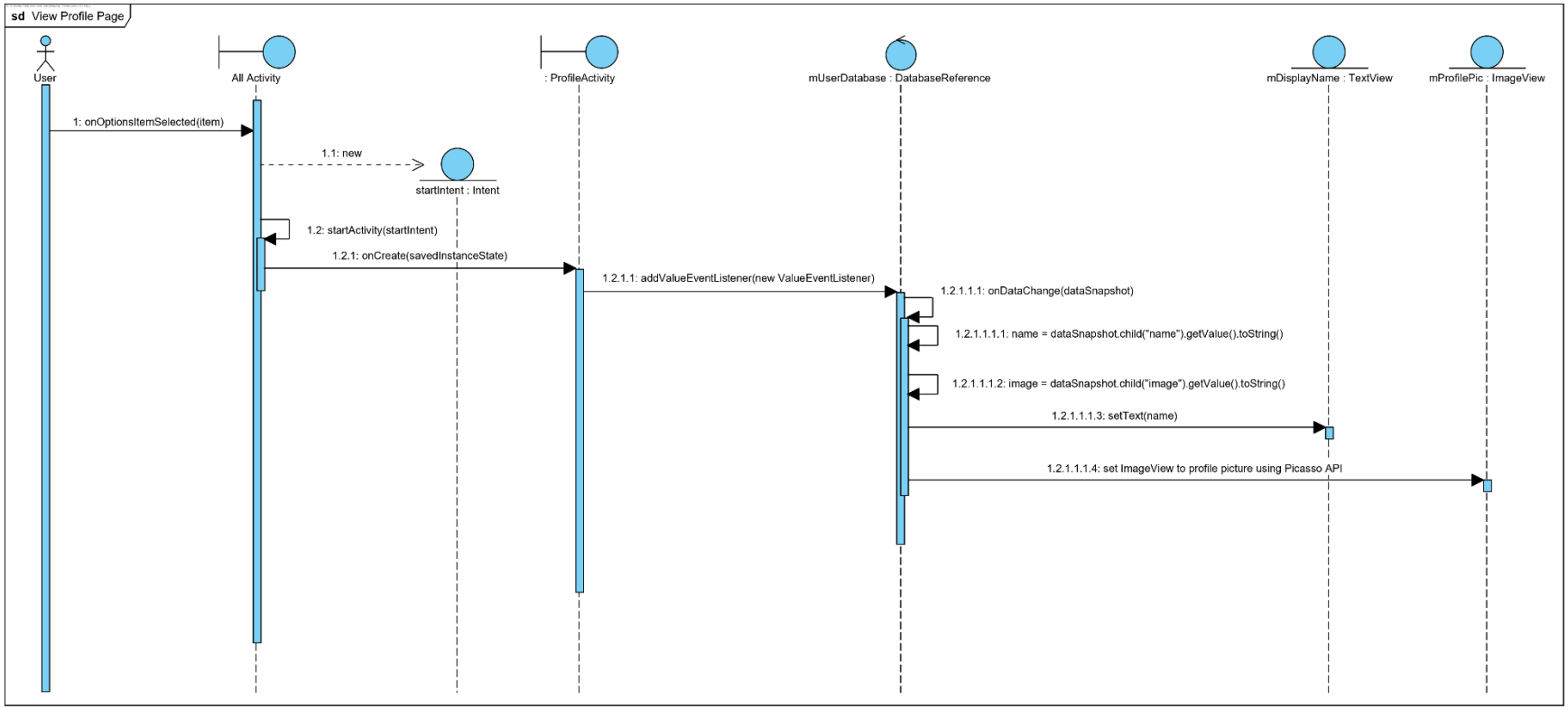
Sequence Diagram 7: Blocking a User



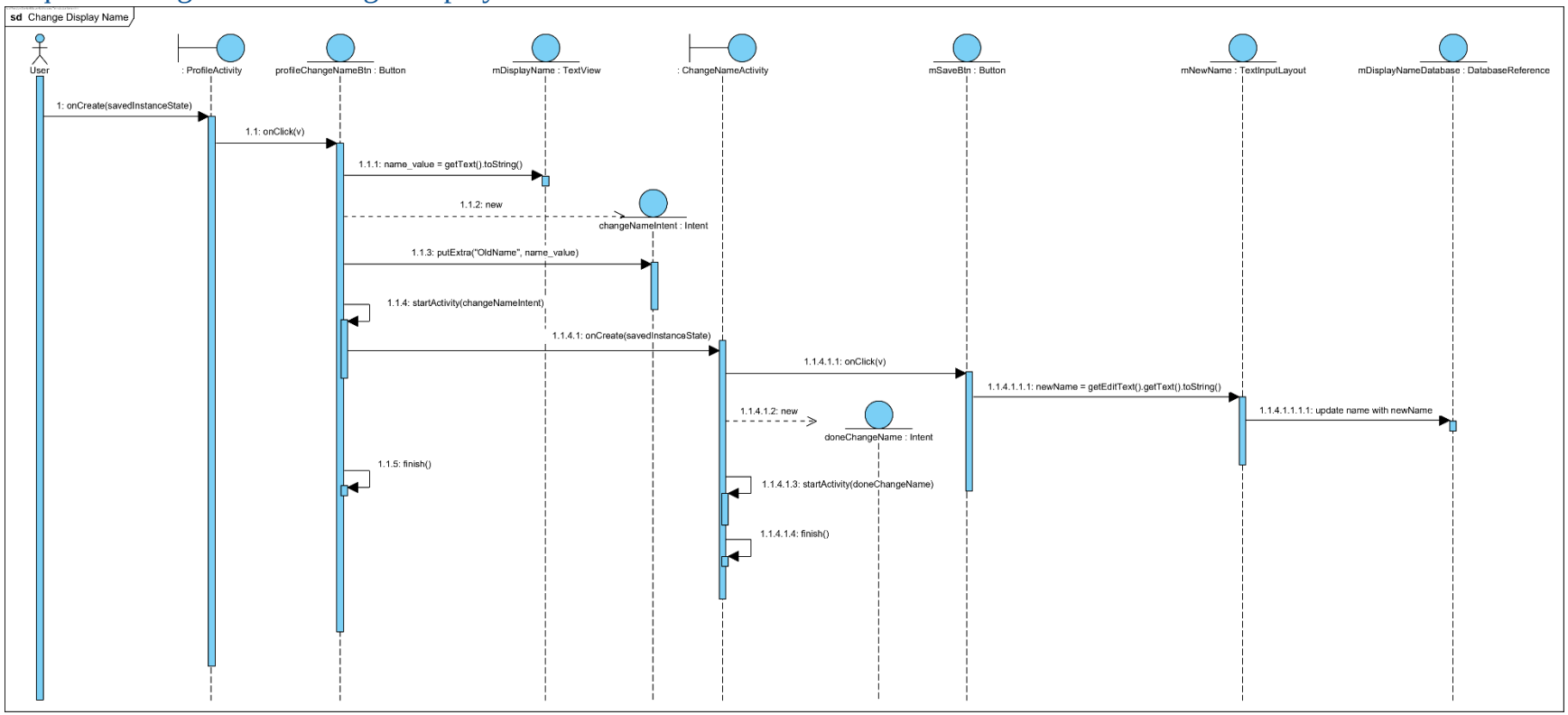
Sequence Diagram 8: View Dropdown Menu



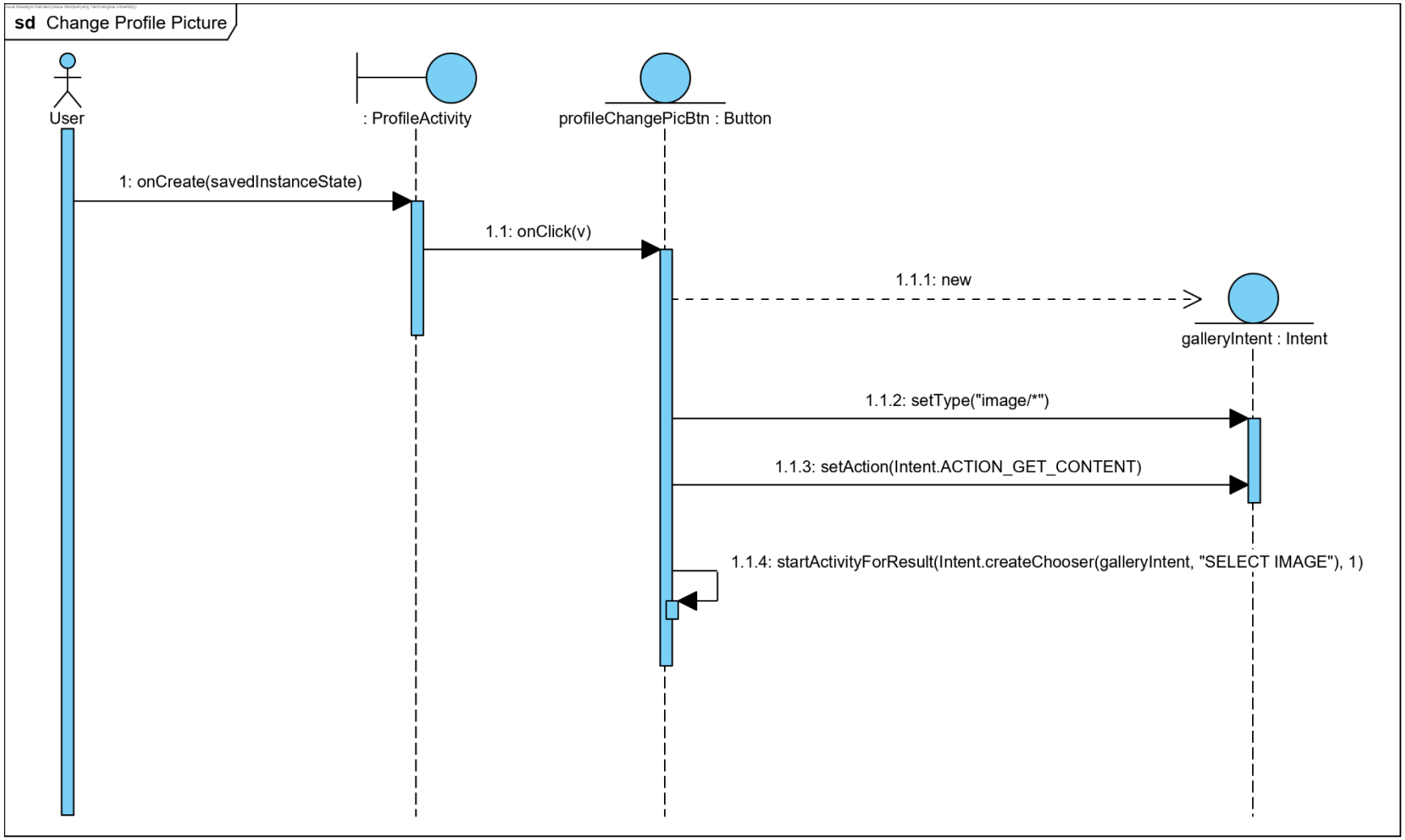
Sequence Diagram 9: View Profile Page



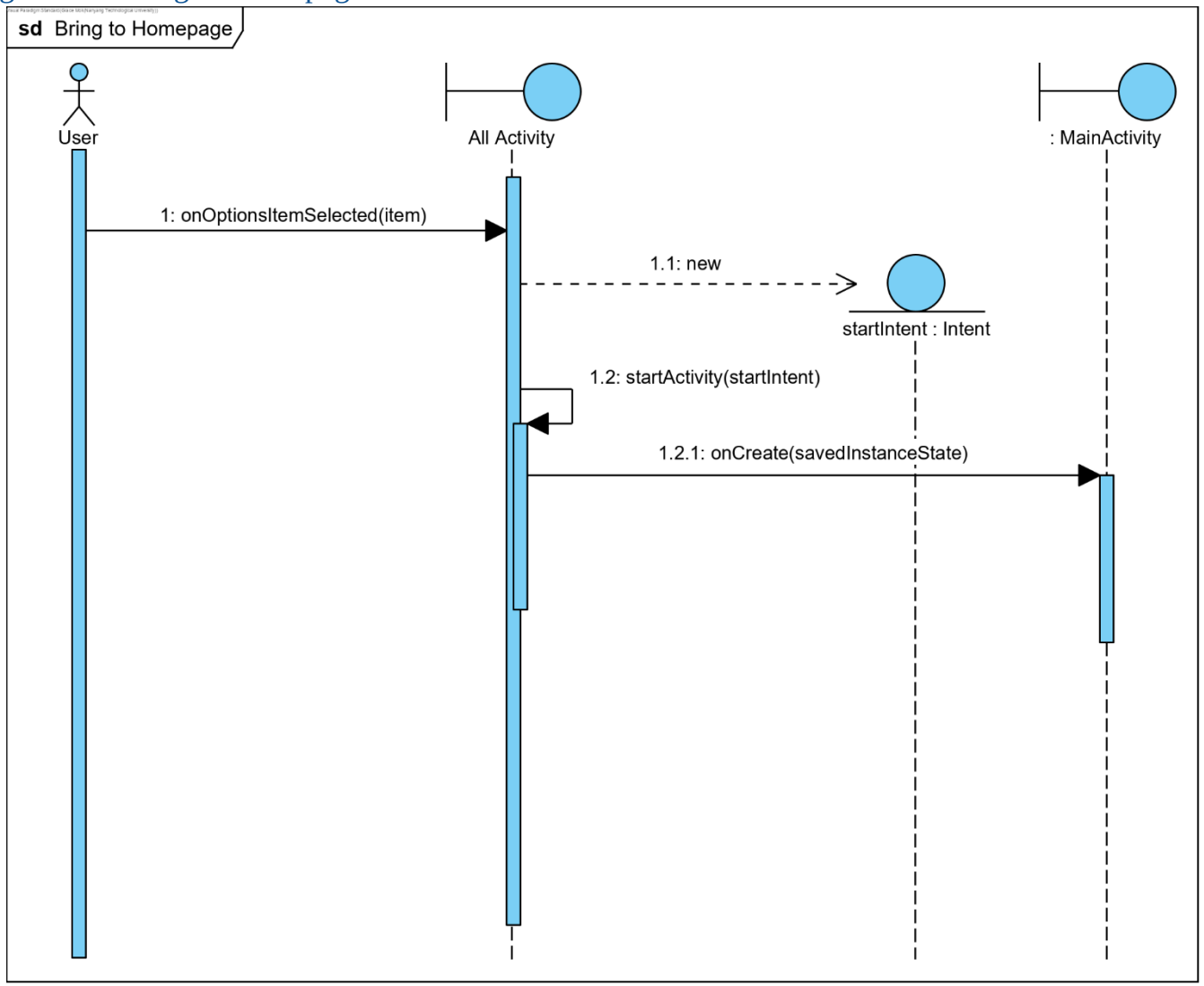
Sequence Diagram 10: Change Display Name



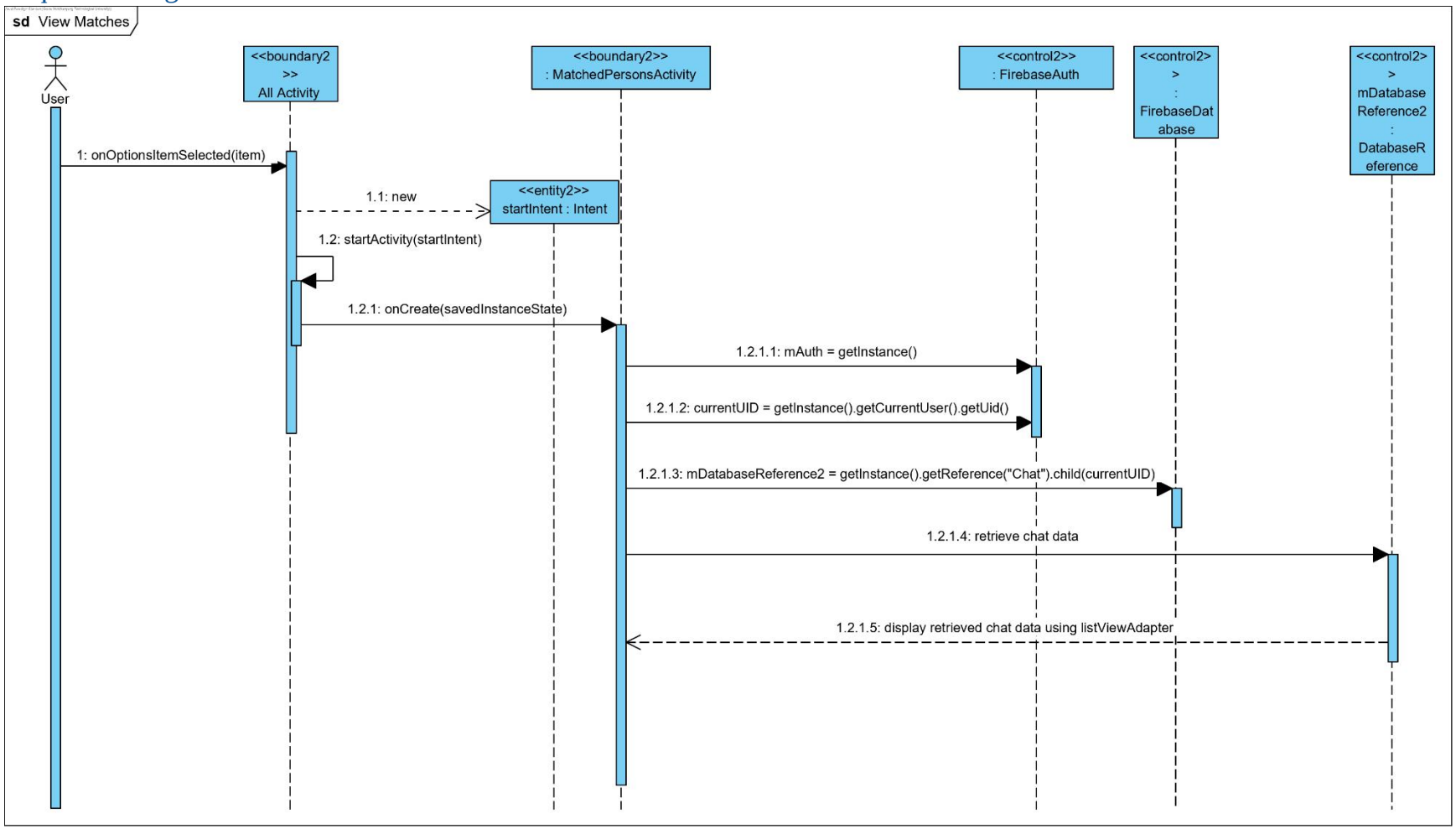
Sequence Diagram 11: Change Profile Picture



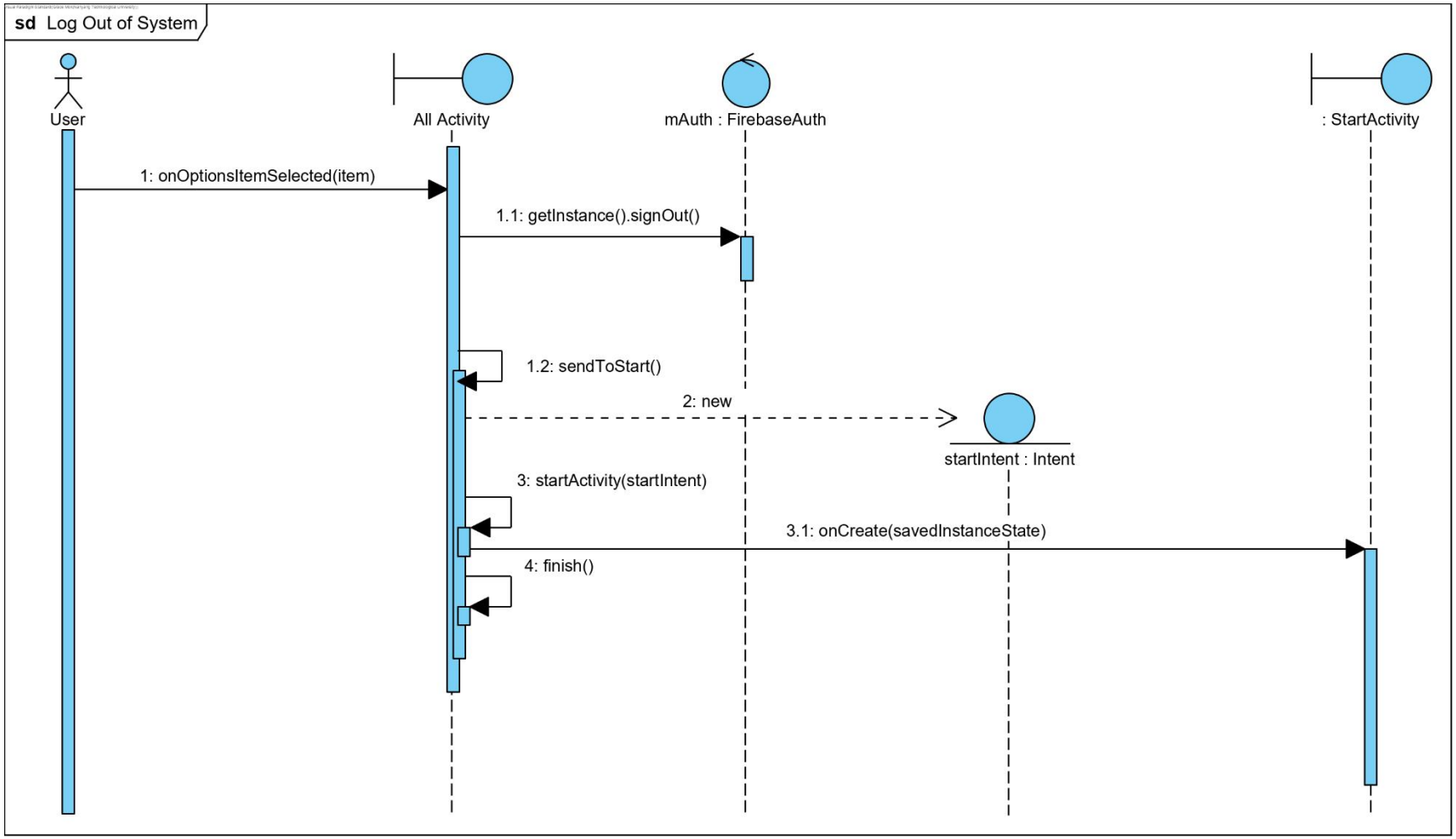
Sequence Diagram 12: Bring to Homepage



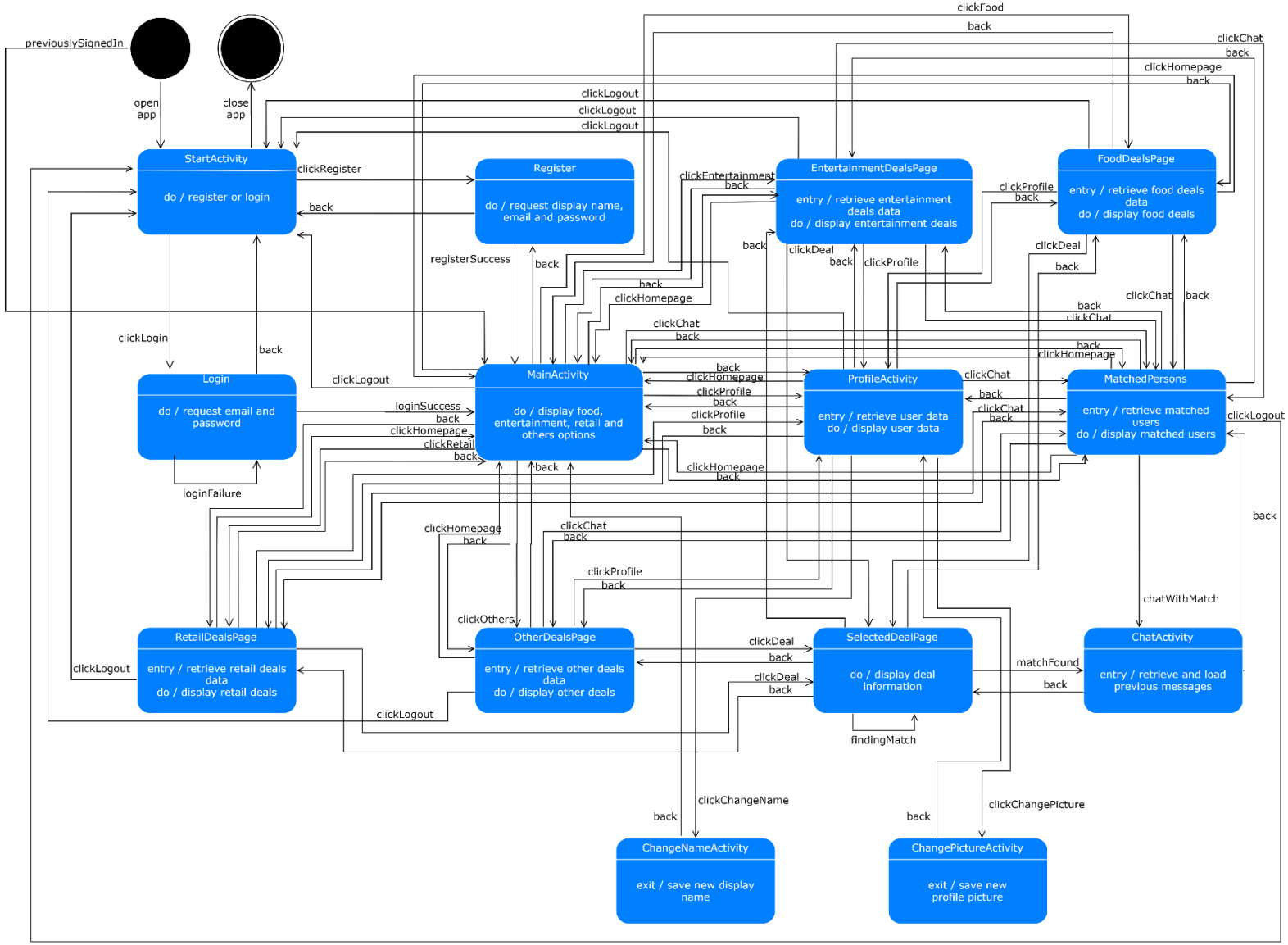
Sequence Diagram 13: View Matches



Sequence Diagram 14: Log Out of System

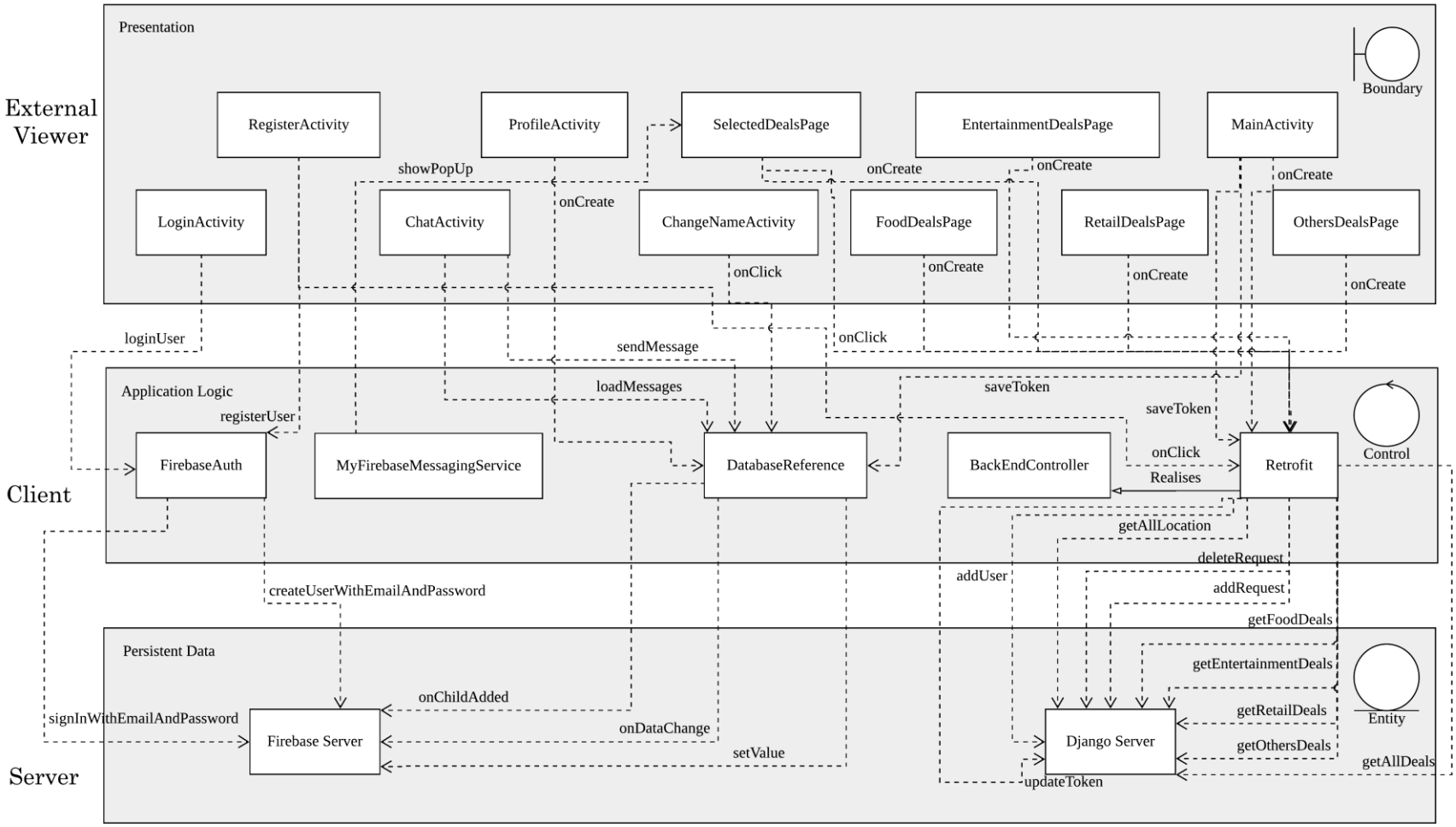


Complete Dialog Map



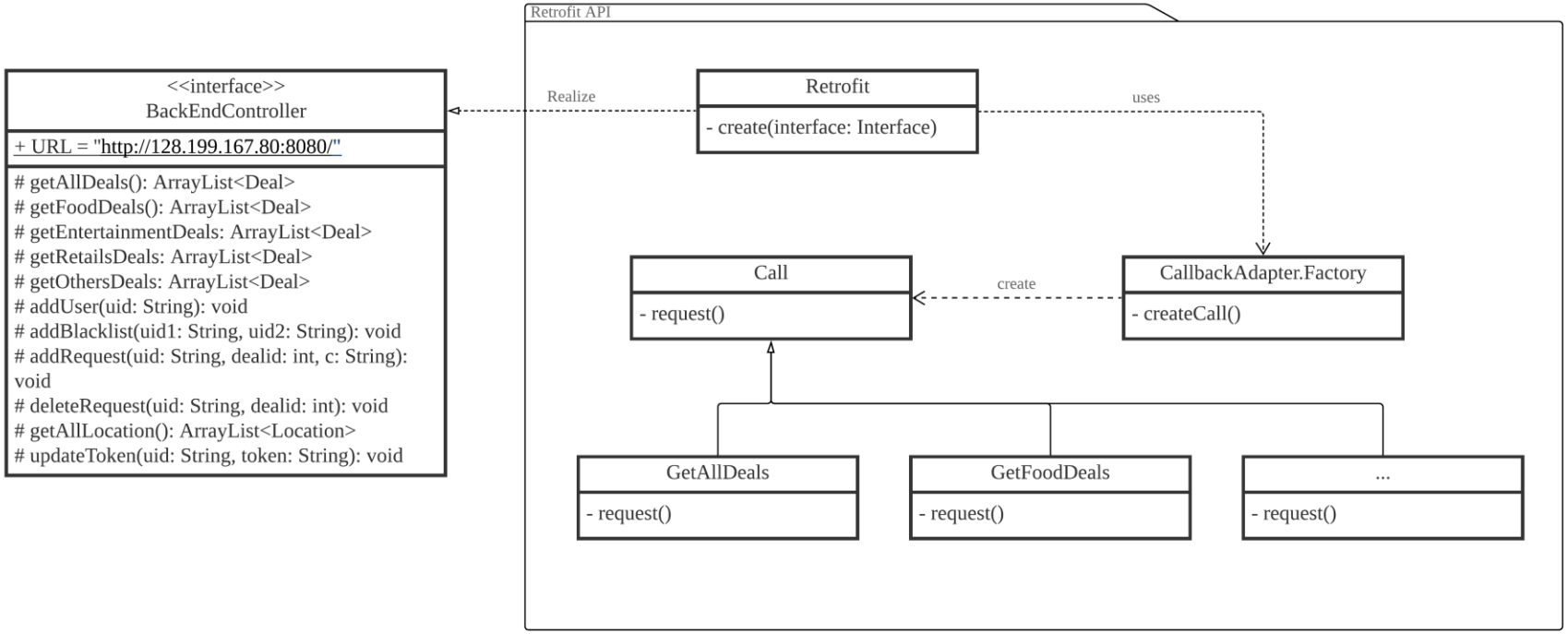
System Architecture

Three-tier Architectural Style + Client-Server Architectural Style



Design Issues and its Solutions

a. Data Access – Strategy and Factory Pattern



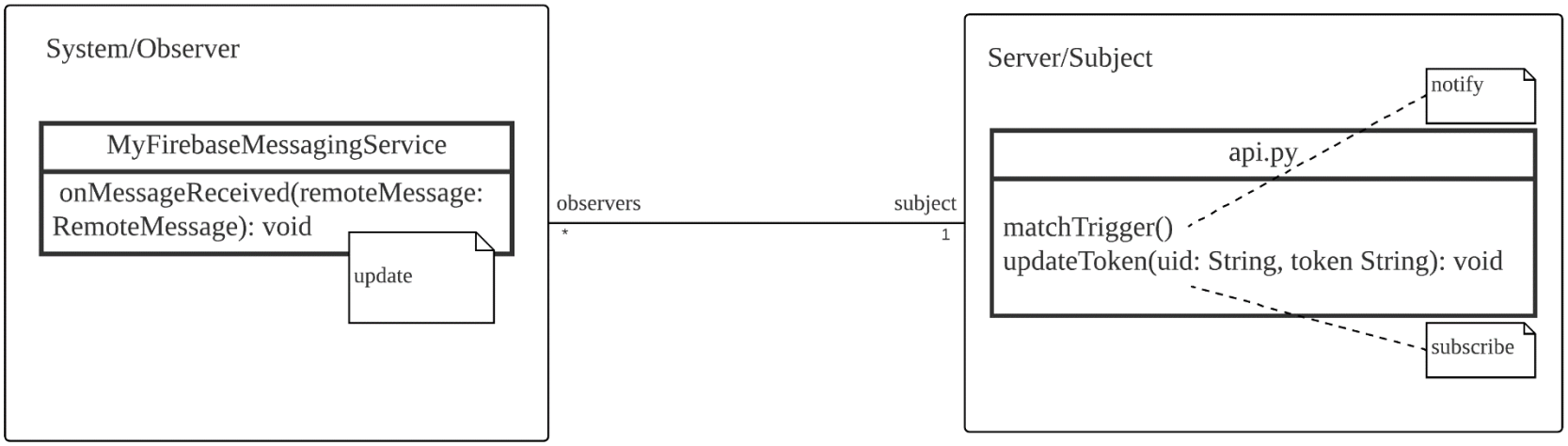
Problem:

If we want to add more API calls to our backend server, we need to change a lot of code. Also, as we scale up the types of API calls, there will be a lot of dependencies and tight coupling.

Solution:

We use a third-party API, Retrofit, to help us dynamically create the different types of API calls to our backend server based on an interface we created, BackEndController. This way, if we want to add more API calls, we simply add an abstract method in the BackEndController class. Also, there will only be one dependency to Retrofit throughout the entire system.

b. Observer Pattern



Problem:

Our system must update and show a popup whenever a match is received. However, we do not want to constantly check on the server for the result of the match.

Solution:

We apply the subscription and notification mechanism using Firebase Cloud Messaging services. This way, we can achieve loose coupling for the server and the system. This means that the Observer subsystem and the Client subsystem are relatively independent to each other and will have little impact on one another. We decided to use the push update as the system requires the full details of the matching result. Thus, it is faster as there is only a one-way communication. Observer is only responsible for receiving the notification from the server.