Pixell

Pixell

# Design Document

Sep 19, 2016

**Team 14**

**Jiapei Liang (Team Leader)**

**Yixuan Ding**

**Nanxin Jin**

**Heng Li**

**Yinuo Pan**

**Yiyang Zhou**

# Table of Contents

## Purpose

**S**tudents always want to save their money by finding ways to get used items and by trying to sell the items they no longer need. There are Facebook groups that sell used items, amazon, ebay, craigslist, and etc. However, trading used items is not a main focus on those platforms. What's more, none of them provide native experience for college students, who have both demand for used items and concern for trade safety. For amazon, ebay, and Craigslist, it takes a long period of time for items to be delivered due to the fact that items are sold from everywhere.

**W**ith our app, Pixell, it is possible that items can be in your hand at the same day you purchased it since you will only have transactions with students in your campus, and you can discuss with the seller further in details. In addition, by limiting users to college students with .edu email address, we have an easy way to verify user identification and deal with any fraud.

## Functional Requirements

### Login / Register

- As a user, I would like to register an account.
- As a Purdue student, I want to use my college email to sign up.
- As a user, I would like to edit my personal information.
- As a user, I would like to reset my password if I forget it.
- As a user, I would like to change my password.
- As a user, I would like to specify my campus when creating an account.
- As a developer, I would like to limit the registration with college email only in order to avoid frauds.
- As a user, I would like to be able to upload my own avatar.
- As a developer, I would like to expand the platform to universities other than just PUWL.

### Browsing

- As a buyer, I want to choose to only see seller's' post.
- As a seller, I want to choose to only see buyers' post.
- As a seller, I want to see what is the most needed items from buyers.
- As a buyer, I want to know how old the item is.
- As a user, I want to know how many people have viewed the posts.
- As a user, I want to see my own posts on my profile view.
- As a user, I want to see other users' posts on their profile views.
- As a user, I want to see other users' rating on their profile views.
- As a user, I want to see other users' rating on the posts they made.
- As a user, I want to see other users' profile by tapping their avatars.
- As a user, I want to know how many users are viewing the same item as I am.

**Posts**

- As a seller, I would like to make a post to sell my item.
- As a buyer, I would like to post my request to look for sellers selling certain items.
- As a seller, I want to post the description of my used items.
- As a seller, I want to post the picture of my used items.
- As a seller, I want to post the price of my used items.
- As a user, I want to tag items on the market.
- As a seller, I would like to specify the condition of the item I am selling. i.e. Like new, Very Good, Good, Acceptable, Broken.
- As a buyer, I would like to click into the post to see the details of the item.

**Search**

- As a buyer, I would like to search items by price range.
- As a buyer, I would like to search items by categories.
- As a buyer, I would like to save items to wish list, so that I can check them later.
- As a buyer, I would like to search items by conditions.

**Sort**

- As a buyer, I would like to have a search function.
- As a buyer, I would like to sort items by price.
- As a buyer, I would like to sort items by categories.
- As a buyer, I would like to sort items by posting time.
- As a buyer, I would like to sort items by seller's rate.
- As a buyer, I want to check statistic data of the item I interested in.

**Chat**

- As a user, I would like to have notification when start chatting.
- As a user, I would like to chat with seller or buyer privately.
- As a user, I would like to let buyer and seller know what item we are talking about when we start the chatting.
- As a seller, I would like to tap a button to confirm that the transaction is completed.
- As a seller, I would like to send pictures to the buyer who is chatting with me.
- As a seller, I would like to send texts to the buyer who is chatting with me.
- As a seller, I would like to send voice message to buyers.
- As a user, I would like to be able to leave messages to the one we are chatting with.
- As a buyer, I would like to be able to chat directly with the seller if he/she is online about the item.
- As a user, I would like see if another user is online.
- As a user, I would like to have the chatting history stored.

**Behavior Control**

- As a buyer, I would like to report bad seller / buyer.
- As a user, I would like to block user sending spams.
- As a developer, I would like to disable fraud account.
- As a user, I would like to sent a report when there is an invalid post.

**After Transaction**

- As a buyer, I would like to have my order list.
- As a seller, I want to see the rating and comment of buyers.
- As a seller, I would like to rate the buyer after transaction.
- As a buyer, I want to see the rating and comment of sellers.
- As a buyer, I would like to rate the seller after transaction.
- As a seller, I want buyers know when the item is available for sell.
- As a developer, I would like to receive user feedback.

## Non-Functional Requirements

### Security

- As a user, I would not like to share my personal information to public.
- As a developer, I will not provide online payment system to protect user personal information, user have to finish the transaction face to face.
- As a developer, I want users provide access to chat history when they report for spam.
- Use https to secure data transitions.

### Appearance (UI / UX)

- The UI must be simple and elegant, so that users can quickly figure out how to use the app.
- Add some features to serve disabled user. i.e. user can modify the font, user can use multi-finger position function to use the app.

### Performance

- As developer, we expect our program runs fast and have a good stability.
- The app must be able to run on a Android device.
- The response time should be fast, so the user don't need to wait too long.
- We should make sure the program will crash as less as possible.
- The system must handle 200 users at a time(if time allows).

# Design Outline

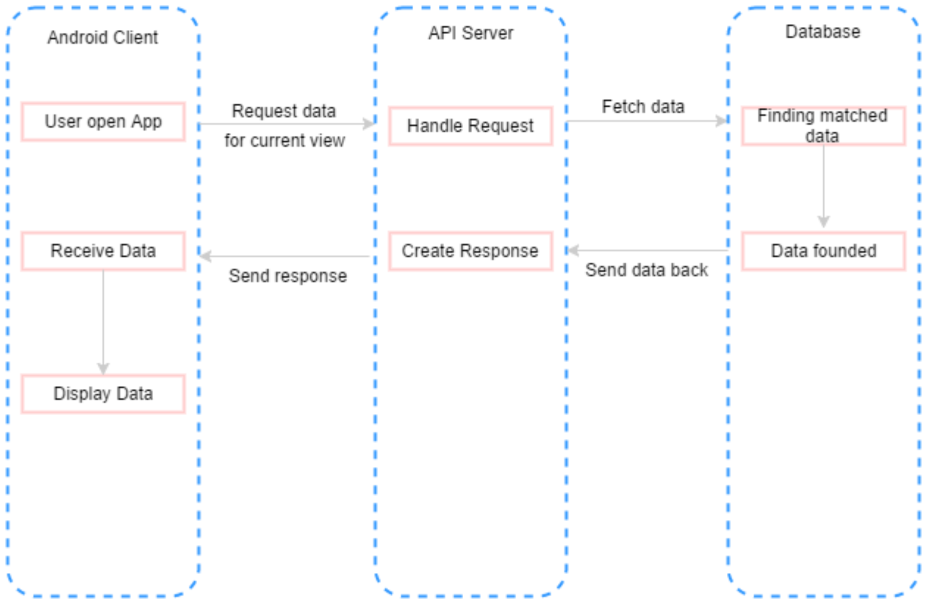
## Description of Data Classes and their Interactions

Our project is an android app that provides a platform for college students to trade used items. Our implementation will use a Client-Server model. The following are our components:

1. **Android Client**
   a. Sends request of processing data to server.
   b. Receives responses from server.
   c. Displays a sorted list of posts.

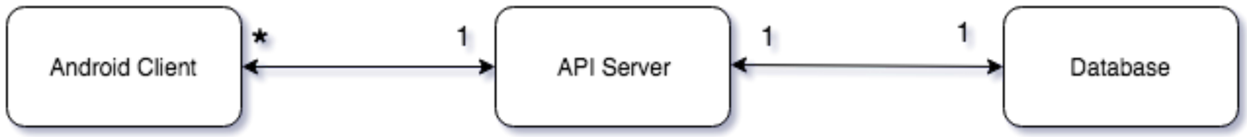
2. **API Server**
   a. Receives and processes requests from clients.
   b. Be able to retrieve/delete/replace data in database.
   c. Responses to clients' requests with data from database.
   d. Uses JSON token for the authentication system.

3. **Database**
   a. Stores information including users, posts, trade history, etc.
   b. Provides data to server when server asks for data.

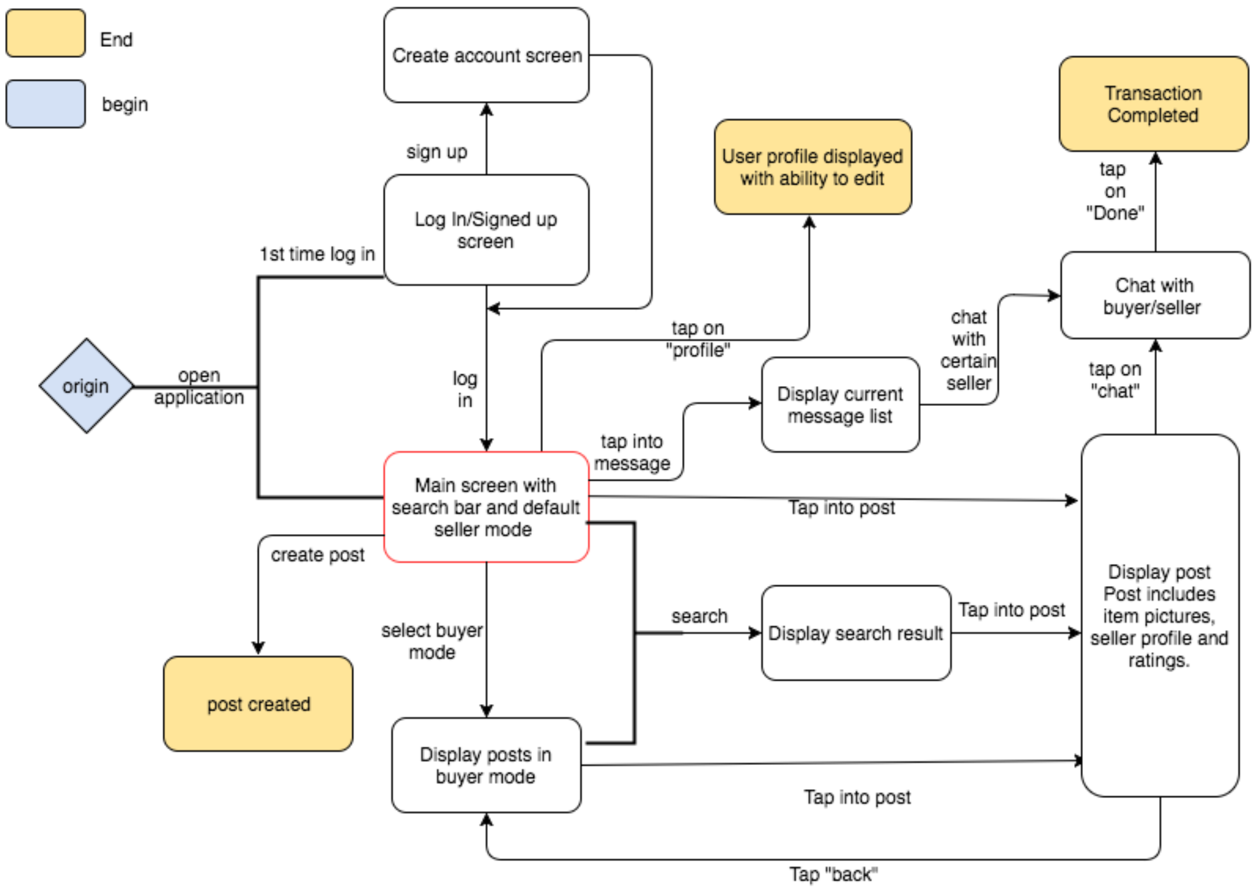


## High Level Overview of the System

The app (Android Client) and the API Server form a many-to-one client-server model. When the server receives a get-data request from the client, it will handle the request by accessing the database to retrieve data and then send the data to the client. When the client asks the server to store data, the server will handle the request by writing data into the database.



## Activity / State Diagram



## Design Issues

### Functional Issues

**1. Does users need to sign up to use our service?**

- Option 1:  Allow user to create their own account using any email address.
- Option 2:  Allow user to login with Facebook, Google, etc., without sign up.
- Option 3:  Allow user to sign up with Purdue email only.
- Decision:  We decided to limit the registration with Purdue email only, which is option 3. This is because our app are focusing on college students only(only PUWL students for now). Limiting the sign up email is helpful for our app to avoid frauds, since the university issued emails are unique to each student.

**2. Should we separate types of posts by buyer and seller?**

- Option 1: Put both buyer posts and seller posts together, with a status bar to show the type of posts.
- Option 2: Separate buyer posts and seller posts in two parts.
- Decision: We decided to separate types of posts by buyer and seller.  Because we believe this way is more convenience for user to search different types of posts. In this way, user can choose to search as a buyer or seller, and they won't mix up different types of posts.

**3. How many campus should we include in our app?**

- Option 1: Only Purdue campus in West Lafayette IN.
- Option 2: Including other campus.
- Decision:  We decided to include only Purdue campus in West Lafayette for now, because we don't have enough time in current semester. For the future uses, we will try to make our app easy to expand to other universities than just PUWL.

**4. What method should we use for user interactions?**

- Option 1: Comments on posts.
- Option 2: One to one instant messages.
- Decision: We decided to let user send one to one instant messages.  In this way, the buyers can bargain with sellers to get a good price eventually. Besides, one to one instant messages is more secure than comments, because it can protect user's personal information.

**5. What should be the base of the chatting between users?**

- Option 1: Connection based on User.
- Option 2: Connection based on Post.
- Decision: We decided to create new chat that base on posts. Because we are designing an app of college used item market, connection by posts is clearer and more convenience for both seller and buyer to discuss the price and payment method of specific item online.

**6. What kind of feedback should we use after the transaction?**

- Option 1: Send private message to seller or buyer by themselves.
- Option 2: Rate seller or buyer.
- Decision: We decided to use rating. Firstly, private message is invisible to other users. Because of this, other users are unable to know the reputation of specific seller or buyer. In this case, rating is more friendly to users. Also, rating is more directed and clearer than private messages, and it can be used to sort searching results.

**7. What should we do with posts when the transaction is done?**

- Option 1: Delete posts from database.
- Option 2: Do not show post, but keep it in the database.
- Decision: We chose not to show posts but keep it in the database after the transaction is done. Because we want to keep every transactions as history for users, but we don't want to show the items which have been sold.

**8. How should we decide when the transaction ends?**

- Option 1: When the buyer taps the "Delivered" button on the chatting view.
- Option 2: When both buyer and seller tap "Delivered" button on the chatting view.
- Decision: We chose to let both buyer and seller tap "Delivered" button. Because we believe the confirmation from both buyer and seller is helpful for our app to avoid frauds.

## Non-Functional Issues

**1. Where do we host our backend services?**

- Option 1: Heroku
- Option 2: Amazon Web Service
- Option 3: Digital Ocean
- Decision: We chose Heroku because it is free and it fits our needs, including speed and space.

**2. What language do we use to implement our backend services?**

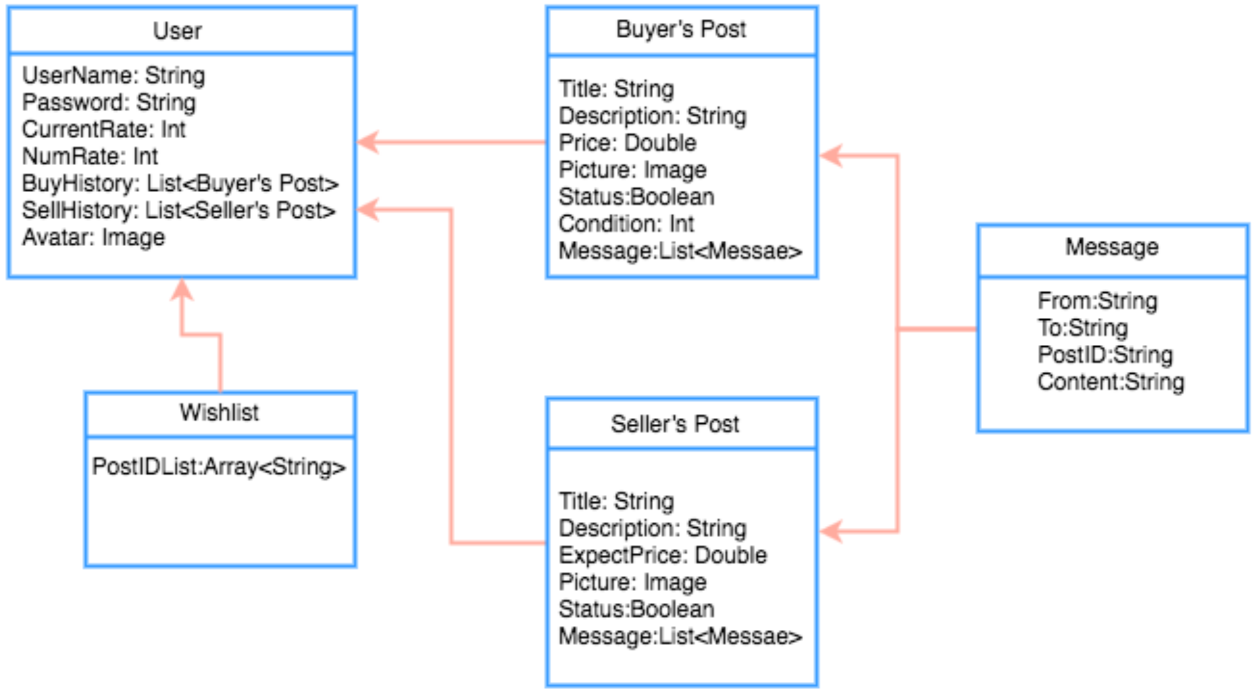
- Option 1: Javascript
- Option 2: C#
- Option 3: Python
- Decision: We chose Javascript because we are going to use Node.JS to build our backend, which uses Javascript, and since we are more familiar with Javascript than other languages.

**3. What type of database is most appropriate for our data?**

- Option 1: SQL
- Option 2: NoSQL (MongoDB)
- Decision: We chose NoSQL because it enables us to build applications faster, handle highly diverse data types, and manage applications more efficiently at scale.

# Design Detail

## Data Class Level-Design



## Description of Data Classes and their Interactions

Our classes are based on the types of objects which are required to process the functions. And those classes are the classes of object stored in the database.

User

- Represents all users of this application.
- Contains display name, password, rating and so on.
- History is a property of user, and it contains id of post created by users.
- History is all the completed posts from database, and displayed on user profile.
- Rating is a property of user, which displays the average rating of the user's posts.

Buyer's Post

- Represents all available and sold buyer's posts.
- Contains elements like name, picture, description and price.
- Post can be created by any user, and it can only be edited by its creator.
- Completed posts will not be displayed in default setting.
- Status is a property of post, and it represents if the post is available or sold.

Seller's Post

- Represent all valid and completed seller's posts.
- Contains elements like name, picture, description and price.
- Post can be created by any user, and it can only be edited by its creator.
- Completed posts will not be displayed in default setting.
- Status is a property of post, it represents if the post is valid or completed.

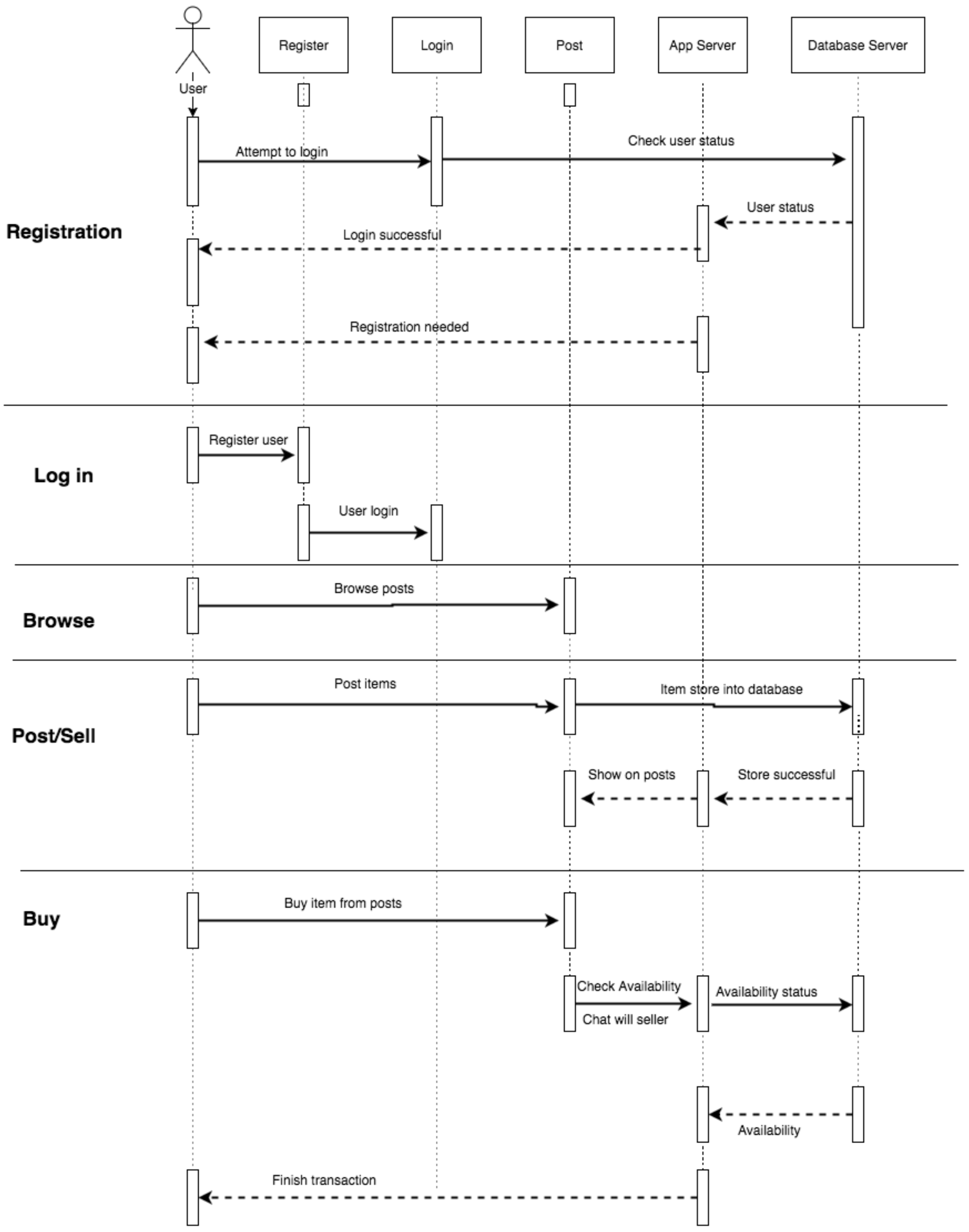
Wish list

- User can add a post to his or her wish list, it will add the id of that post into the wish list.
- When user enters his / her wish list page, the client will use the ids in wish list to retrieve the data from database, and display it.

Message

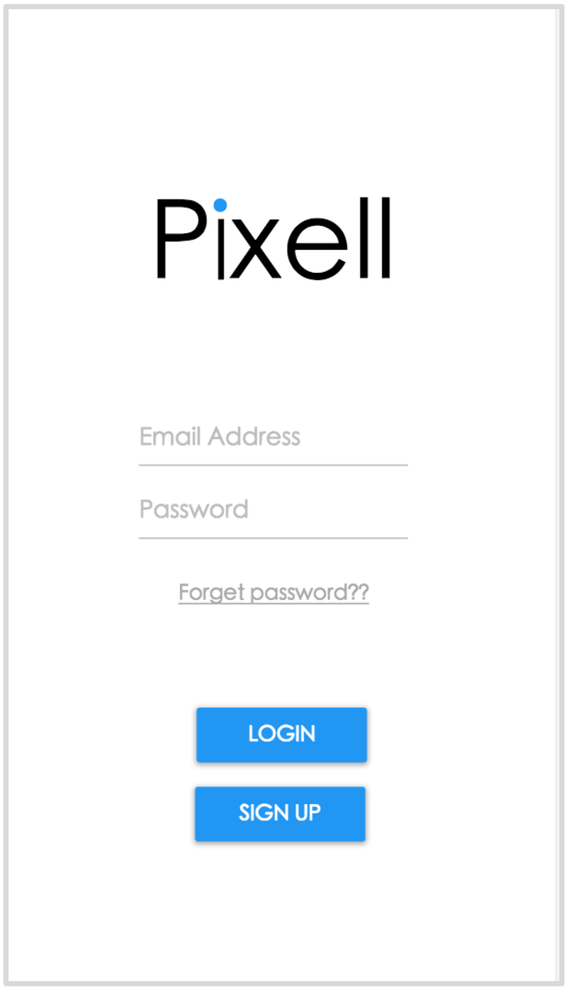
- Message represents the messages between sellers and buyers.
- The data of instant message is stored in user's phone.
- The unsaved messages are stored in database. They will be deleted after received.
- Users can talk to other users through the posts, not through users directly.

## Sequence Diagrams

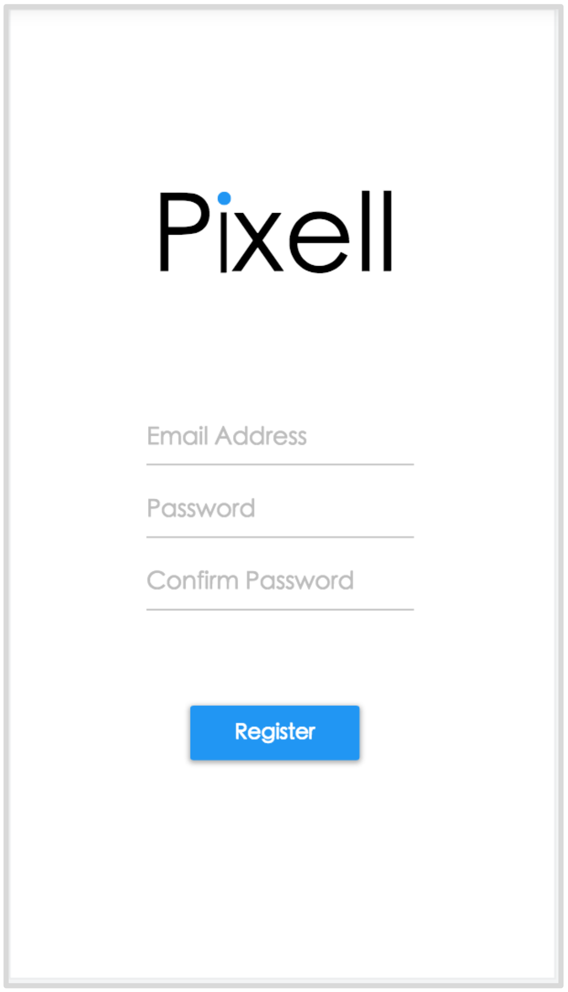


## UI Mockups

### Login View

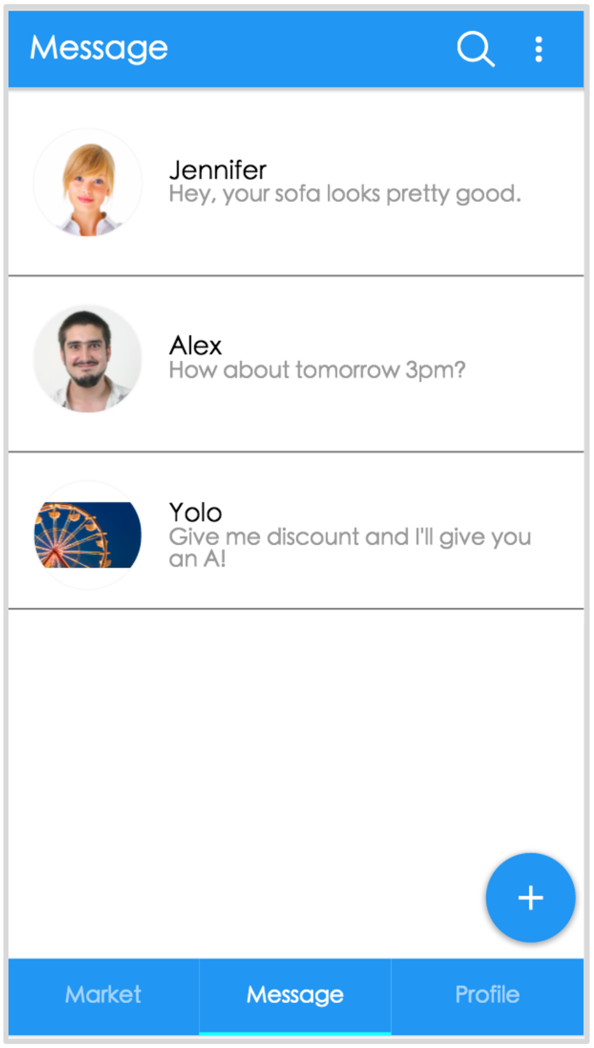


### Sign Up View



**Message View**

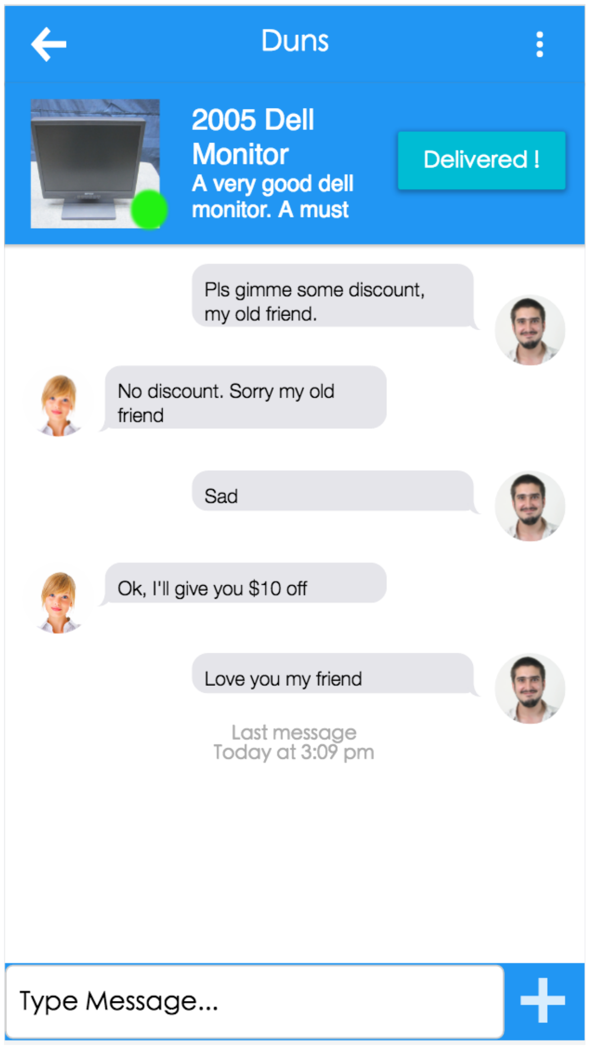
**Chatting View**



Jennifer
Hey, your sofa looks pretty good.

Alex
How about tomorrow 3pm?

Yolo
Give me discount and I'll give you an A!



2005 Dell Monitor
A very good dell monitor. A must

Pls gimme some discount, my old friend.

No discount. Sorry my old friend

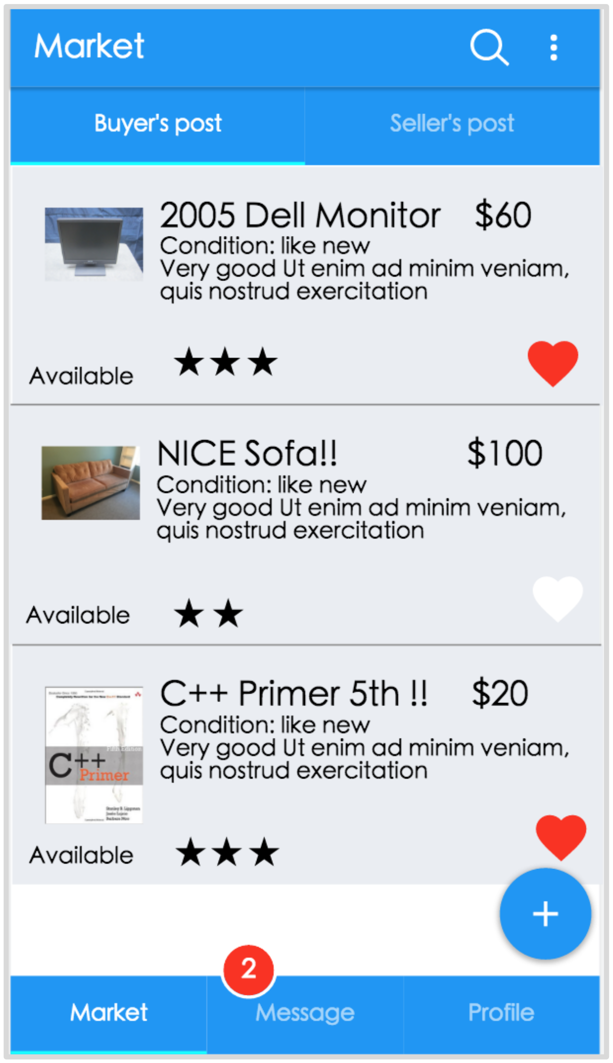
Sad

Ok, I'll give you $10 off

Love you my friend

Last message
Today at 3:09 pm

**Market View**



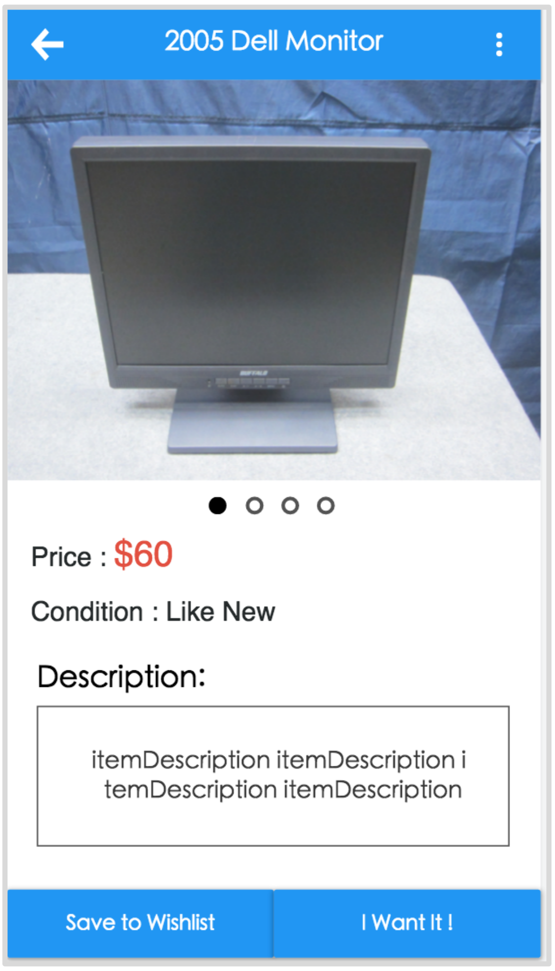
2005 Dell Monitor $60
Condition: like new
Very good Ut enim ad minim veniam, quis nostrud exercitation
Available ★★★

NICE Sofa!! $100
Condition: like new
Very good Ut enim ad minim veniam, quis nostrud exercitation
Available ★★

C++ Primer 5th !! $20
Condition: like new
Very good Ut enim ad minim veniam, quis nostrud exercitation
Available ★★★

**Make a Post View**

**Post Detail View**

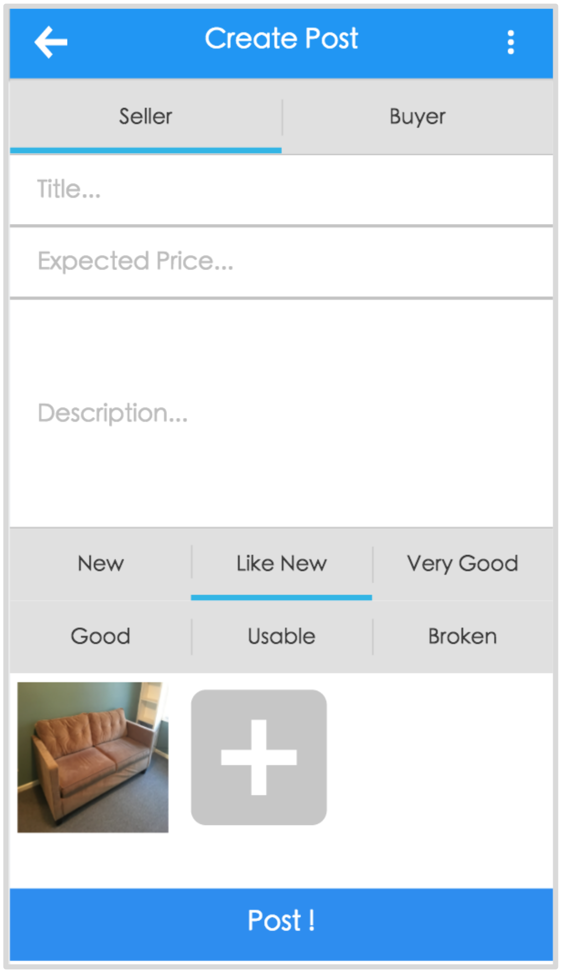


Price : $60

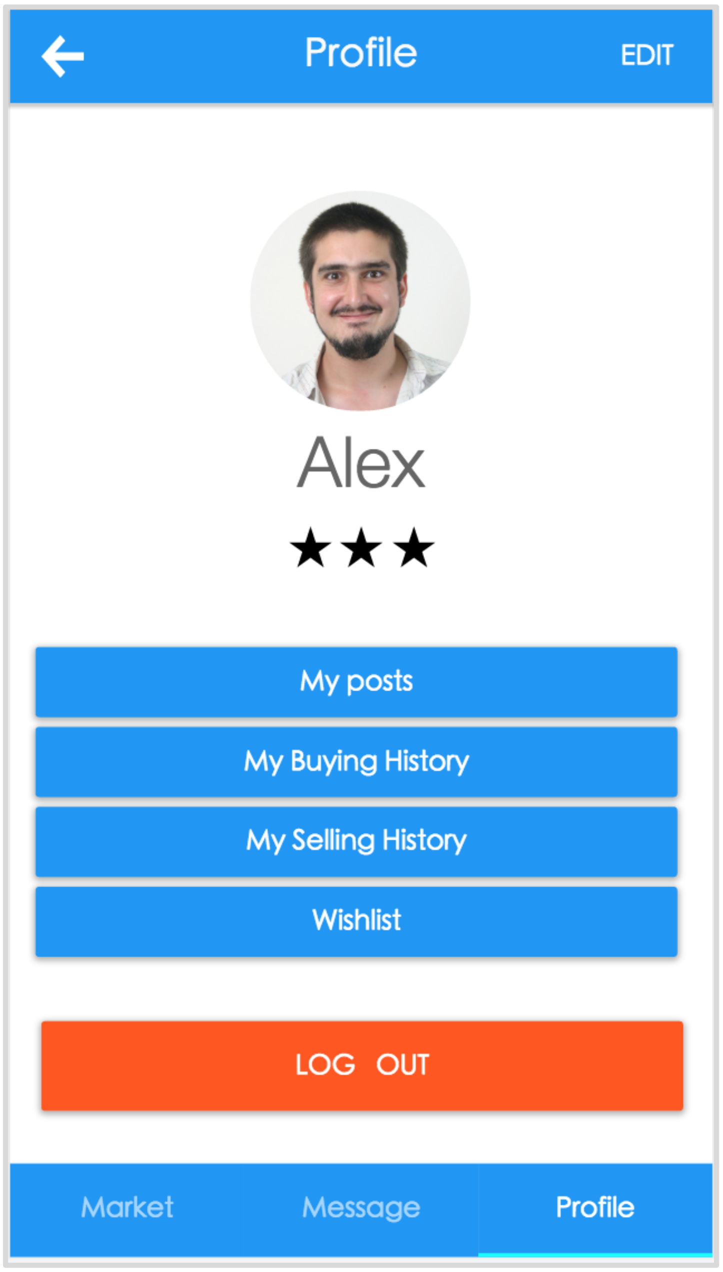
Condition : Like New

Description:

itemDescription itemDescription itemDescription itemDescription



**User Profile**



## API route

The API routes are the URIs provided by the API server, and they define the communication methods between clients and the server. Each URI is an action to interact with server or database.

| **Function** | **Route** | **HTTP Method** |
|---|---|---|
| Get posts | /posts | GET |
| Get user profile | /user/{id}/profile | GET |
| Edit user profile | /user/{id}/profile | PUT |
| Get user's buy history | /user/{id}/buy_history | GET |
| Get user's sell history | /user/{id}/sell_history | GET |
| Get user's wish list | /user/{id}/wishlist | GET |
| Get user's rating | /user/{id}/rating | GET |
| Rate a user | /user/{id}/rating | POST |
| Report a user | /user/{id}/report | POST |
| Get a post | /post/{id} | GET |
| Make a post | /post/{id} | POST |
| Edit a post | /post/{id} | PUT |
| Send a message | /post/{id}/message | POST |
| Report a post | /post/{id}/report | POST |