* All text responses are in JSON format.

|  |
| --- |
| **Function:** |
| **Register new user** |
| **Access Point:** |
| **“~/UserServices/registerNewUser”** |
| **Description:** |
| **The service is for registering new users into the database.** |
| **Parameters:** |
| **name: username**  **emailAddress**  **password**  **gender: 0 represents male, and 1 represents female**  **phone**  **profilePicture: profile picture data**  **addressLine1**  **addressLine2**  **city**  **state**  **zipcode**  **school\_name**  **school\_city** |
| **Response:** |
| 1. **“Successful”, “waiting for account activation.” This means the user information gets handled, and an email has been sent the user’s email inbox.** 2. **“Fail”, “The user has existed.” This means the user has already existed in database. No need to register. (ErrorCode: 1)** 3. **“Error”, “The user’s school or address information cannot be saved to DB.” This means some problems happen on hibernate when it tries to save school or street information to MySql database. Nothing will be saved by the server. (ErrorCode: 2)** 4. **“Error”, “The user image cannot be saved to server disk.” This means the image cannot be saved, but the information may have temporarily saved by the server. (ErrorCode: 3)** 5. **“Error”, “Unknown Error.” This means unknown error happens. (ErrorCode: 0)** |

|  |
| --- |
| **Function:** |
| **Resend the verification email** |
| **Access Point:** |
| **“~/UserServices/resendEmail”** |
| **Description:** |
| **The service is for resending the verification email to the user. The service uses session to record the temporary user information. However, this service does not work anymore if the program is terminated since the swift will delete the session cookie.** |
| **Parameters:** |
| **Nothing.** |
| **Response:** |
| 1. **“Successful”, “The email has been sent.” This means the email has been sent to the user’s email inbox.** 2. **“Fail”, “Please register the information first.” This means the user has never registered his/her information before. His/her session does not exist. (ErrorCode: 4)** 3. **“Error”, “The email cannot be sent to the destination.” Email cannot go to the destination’s email inbox. (ErrorCode: 5)** 4. **“Error”, “Unknown Error.” Same as above. (ErrorCode: 0)** |

|  |
| --- |
| **Function:** |
| **Request user information** |
| **Access Point:** |
| **“~/UserServices/requestUserInfo”** |
| **Description:** |
| **The service is for fetching the user information from the database.** |
| **Parameters:** |
| **Nothing. The service uses the session. User must login in order to use this service.** |
| **Response:** |
| 1. **email, name, phone, password, gender, addressLine1, addressLine2, city, state, zipcode, school\_name (all in json)** 2. **“Error”, “The user information cannot be fetched from database.” This means some problems happen on hibernate when it communicates with the database.** 3. **“Error”, “Unknown Error.” Same as above.** |

|  |
| --- |
| **Function:** |
| **Login the user** |
| **Access Point:** |
| **“~/UserServices/login”** |
| **Description:** |
| **This service is for login the user. A user’s session will be created at the server side.** |
| **Parameters:** |
| **emailAddress**  **password** |
| **Response:** |
| 1. **“Successful”, “The user has been authenticated.” This means the user has been logged in.** 2. **“Fail”, “The username and password are not correct.” This means information provided by the user is not right.** 3. **“Error”, “The server cannot communicate with the database.” This means some problems happen on hibernate when it communicates with the database.** 4. **“Error”, “Unknown Error.” The error that the server does know the cause.** |

|  |
| --- |
| **Function:** |
| **Post the item** |
| **Access Point:** |
| **“~/ProductServices/postItem”** |
| **Description:** |
| **This service is for posting the item information to the database (information only). If some photos need to be uploaded, please use the service below for uploading photos. A cookie which contains the item’s id is sent to the application. When you post item images, the cookie will be sent back to server automatically. The server can determine where to put images based on this cookie.** |
| **Parameters:** |
| **name: product’s name**  **description: the item description**  **price**  **condition: this should be the integer number**  **category: the item’s category** |
| **Response:** |
| ***All in json format.***  **“Successful”, “The information has been uploaded.” This means item information has been saved to the database.**  **“Fail”, “The user has not been verified.” This means the user is not verified through the email.**  **“Fail”, “The user has been signed out.” This means the user needs to login again.**  **“Fail”, “The user does not exist.” This means the user does not exist in the database.**  ***When errors happen, they both have “ErrorDetail”, “<detail>” to show the error detail.***  **“Error”, “Errors happen on interacting with the database.” This means some problems happen on interacting with the database.**  **“Error”, “Unknown Error.” The error cannot be detected by the database.** |

|  |
| --- |
| **Function:** |
| **Post the item image** |
| **Access Point:** |
| **“~/ProductServices/postItemImage”** |
| **Description:** |
| **This service is for uploading images. One image each time.** |
| **Parameters:** |
| **Be careful that this service accepts multipart form data.**  **imageIndex: the image index in the array (The server uses its index to mark the file name).**  **itemImage: the image data. The image is expected to be in jpeg format.** |
| **Response:** |
| **“Successful”, “The image has been uploaded.” This means the image is saved to the server.**  **“Fail”, “The product does not exist.” This means the product does not exist.**  **“Fail”, “The user has been signed out.” This means the user needs to login again.**  **“Fail”, “Please upload item basic info first.” This means item’s text information**  **“Error”, “Errors happen on interacting with the database.” This means some problems happen on interacting with the database.**  **"Error", "The image cannot be saved to server disk." This means the image cannot be written to the hard disk of the server.**  **“Error”, “Unknown Error.” The error that cannot be detected by the server.** |

|  |
| --- |
| **Function:** |
| **Get all categories** |
| **Access Point:** |
| **“~/CategoryServices/getAllCategories”** |
| **Description:** |
| **This services returns all categories in the database.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **“categories”, “{category: <categoryName>, category: <categoryName>}”** |

|  |
| --- |
| **Function:** |
| **Send forgot password email** |
| **Access Point:** |
| **“~/UserServices/forgotPassword”** |
| **Description:** |
| **This service sends an email to the user’s email inbox for changing the password.** |
| **Parameters:** |
| **emailAddress: form parameter** |
| **Response:** |
| **“Successful”, “The email has been sent.” An email will appear in the mail inbox.** |

|  |
| --- |
| **Function:** |
| **Set user’s prefer category** |
| **Access Point:** |
| **“~/UserServices/setUserPreferCategory”** |
| **Description:** |
| **This service is for setting the user prefer category. One category for each service.** |
| **Parameters:** |
| **category: the category in string (all lower case).** |
| **Response:** |
| **“Successful”, “The user’s prefer category has been set.”**  **“Fail”, “The user has been signed out.” This means the user’s basic information is not saved to the database successfully.** |

|  |
| --- |
| **Function:** |
| **Check if the email address has existed** |
| **Access Point:** |
| **“~/UserServices/isEmailExisted”** |
| **Description:** |
| **The service is for checking if the email address has existed in the database.** |
| **Parameters:** |
| **emailAddress** |
| **Response:** |
| **“True”, “The user has existed.”**  **“False”, “The user is not existed.”** |

|  |
| --- |
| **Function:** |
| **Check if the phone has existed** |
| **Access Point:** |
| **“~/UserServices/isPhoneExisted”** |
| **Description:** |
| **The service is for checking if the phone has existed in the database.** |
| **Parameters:** |
| **phone** |
| **Response:** |
| **“True”, “The user has existed.”**  **“False”, “The user is not existed.”** |

|  |
| --- |
| **Function:** |
| **Load items to item list page** |
| **Access Point:** |
| **“~/ProductServices/loadItems”** |
| **Description:** |
| **The request is the POST request. The service is for loading items to the item list page. 10 items will be returned per call. Be careful that this service only returns basic information of the item. If it is necessary to download cover images. Please use the service below.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **“items”, “{{itemId: <id>, name: <name>, price: <price>}, {itemId: <id>, name: <name>, price: <price>}, …}” The jsonArray contains all item json objects.**  **“Fail”, “No more items.” This means all items have been listed.**  **“Fail”, “The user has been signed out.” This means the session times out. The user needs to login again.**  **“Error”, “Unknown Error.” The error cannot be detedcted by the server.** |

|  |
| --- |
| **Function:** |
| **Load item cover image to the item list page.** |
| **Access Point:** |
| **“~/ProductServices/getItemCoverPhoto/{itemId}”** |
| **Description:** |
| **The request is the GET request. The service is for loading item cover photos to the item list page** |
| **Parameters:** |
| **itemId: the id of the item. This parameter is shown in the URL.** |
| **Response:** |
| **The image data. The format is jpeg.**  **If some problems happen, no image data gets downloaded.** |

|  |
| --- |
| **Function:** |
| **Update the user information on the database.** |
| **Access Point:** |
| **“~/UserServices/updateUser”** |
| **Description:** |
| **The service is for updating the user’s information** |
| **Parameters:** |
| **name**  **emailAddress**  **gender**  **phone**  **profilePicture: this is the image data**  **addressLine1**  **addressLine2**  **city**  **state**  **zipcode**  **school\_name**  **school\_city** |
| **Response:** |
| **“Successful”, “The info has been updated.”**  **“Fail”, “The another user is using the same email address or phone number.”**  **“Fail”, “The user does not exist.”**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **Check if the phone is existed in the database when update.** |
| **Access Point:** |
| **“~/UserServices/isPhoneExistedWhenUpdate”** |
| **Description:** |
| **The service is for checking if phone has existed in the database when update.** |
| **Parameters:** |
| **phone: the phone number** |
| **Response:** |
| **“True”, “The user has existed.”**  **“False”, “It is unique.”**  **“Fail”, “The user does not exist.”**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **Check if the email is existed in the database when update.** |
| **Access Point:** |
| **“~/UserServices/isEmailExistedWhenUpdate”** |
| **Description:** |
| **The service is for checking if email has existed in the database when update.** |
| **Parameters:** |
| **email: the email address** |
| **Response:** |
| **“True”, “The user has existed.”**  **“False”, “It is unique.”**  **“Fail”, “The user does not exist.”**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **Get the item information** |
| **Access Point:** |
| **“~/ProductServices/getItem/{itemId}”** |
| **Description:** |
| **The service for getting the item information.** |
| **Parameters:** |
| **itemId: the id the of the item. Be careful that this is the path parameter.** |
| **Response:** |
| **../../../../Desktop/Screen%20Shot%202017-12-26%20at%207.16.52%** |

|  |
| --- |
| **Function:** |
| **Get the item photo** |
| **Access Point:** |
| **“~/ProductServices/getItemImage/{itemId}/{itemImageIndex}”** |
| **Description:** |
| **The service for getting the item image. The service returns image data directly to the client.** |
| **Parameters:** |
| **itemId: the id the of the item. Be careful that this is the path parameter.**  **itemImageIndex: the item image index.** |
| **Response:** |
| **The image data if the image exists.**  **“nil” if the image does not exist.** |

|  |
| --- |
| **Function:** |
| **Log out the user** |
| **Access Point:** |
| **“~/UserServices/logout”** |
| **Description:** |
| **The service is for logging out the user.** |
| **Parameters:** |
| **None** |
| **Response:** |
| **“Successful”, “The user has been signed out”** |

|  |
| --- |
| **Function:** |
| **Loading items by the category** |
| **Access Point:** |
| **“~/ProductServices/loadItemByCategory/{category}”** |
| **Description:** |
| **The service is for loading items with a specific category** |
| **Parameters:** |
| **category: the category of the item** |
| **Response:** |
| **“Success”, “{{itemId: <id>, name: <name>, price: <price>, condition: <condition>}, {itemId: <id>, name: <name>, price: <price>, condition: <condition>}, …}” The jsonArray contains all item json objects.**  **“Fail”, “No more items.” This means all items have been listed.**  **“Fail”, “The user has been signed out.” This means the session times out. The user needs to login again.** |

|  |
| --- |
| **Function:** |
| **Send the request for purchasing the item.** |
| **Access Point:** |
| **“~/ProductServices/requestPurchasingItem”** |
| **Description:** |
| **The service is for sending the purchasing request to the seller** |
| **Parameters:** |
| **itemId: the id of the item**  **tradingAddressLine1**  **tradingAddressLine2**  **city**  **state**  **zipcode**  **time: the time in String format.**  **offerPrice: the offer of the request**  **note: the note that needs to be sent to the seller** |
| **Response:** |
| **“Success”, “The product is pending for you.”**  **“Fail”, “The product has been removed by the seller.”**  **“Fail”, “The user has not been verified.”**  **“Fail”, “You have sent the purchase request for the item.”**  **“Fail”, “You cannot purchase your own product.”**  **“Fail”, “The product has been sold.”** |

|  |
| --- |
| **Function:** |
| **Get the user profile image by id** |
| **Access Point:** |
| **“~/UserServices/getProfilePicById/{userId}”** |
| **Description:** |
| **The service is for retrieving the user’s profile image by the user id.** |
| **Parameters:** |
| **userId: path parameter** |
| **Response:** |
| **ImageData or nil.** |

|  |
| --- |
| **Function:** |
| **Confirm the purchase request.** |
| **Access Point:** |
| **“~/ProductServices/confirmPurchaseRequest”** |
| **Description:** |
| **The service is for confirming the purchase request.** |
| **Parameters:** |
| **Form parameter**  **requestId: the id of the request.** |
| **Response:** |
| **"Success", "The request is confirmed."**  **"Fail", "The user has not been verified."**  **"Fail", "The user has been signed out."**  **“Fail”, “The request was canceled by the buyer.”** |

|  |
| --- |
| **Function:** |
| **Get all pending items for the buyer.** |
| **Access Point:** |
| **“~/ProductServices/getBuyerPendingItems”** |
| **Description:** |
| **The service is for retrieving buyer pending items.** |
| **Parameters:** |
| **None** |
| **Response:** |
| **“Success”, “{{id, name, price}, {id, name, price}}”** |

|  |
| --- |
| **Function:** |
| **Get all pending items for the seller.** |
| **Access Point:** |
| **“~/ProductServices/getSellerPendingItems”** |
| **Description:** |
| **The service is for retrieving seller pending items.** |
| **Parameters:** |
| **None** |
| **Response:** |
| **“Success”, “{{id, name, price}, {id, name, price}}”** |

|  |
| --- |
| **Function:** |
| **Check if the user has logged in.** |
| **Access Point:** |
| **“~/UserServices/isLoggedIn”** |
| **Description:** |
| **The service is for checking if the user has logged in.** |
| **Parameters:** |
| **None** |
| **Response:** |
| **“isLoggedIn”, “True”**  **“isLoggedIn”, “False”** |

|  |
| --- |
| **Function:** |
| **Upload the device token to the remote location.** |
| **Access Point:** |
| **“~/UserServices/uploadDevice” (POST)** |
| **Description:** |
| **The service is for storing the device token into the database. The server can check if the device has already existed, so the duplicate device token is acceptable.** |
| **Parameters:** |
| **deviceID: the device token** |
| **Response:** |
| **“Success”, “The device has been uploaded.”**  **“Error”, “<Error Detail>”** |

|  |
| --- |
| **Function:** |
| **Binds the device with the user.** |
| **Access Point:** |
| **“~/UserServices/bindDevice” (POST)** |
| **Description:** |
| **The service is for binding the user with the device. This service should only be called when the user is logged in, and the login cookie is set. This also means the service should be called after login request completes.** |
| **Parameters:** |
| **deviceID: the device token** |
| **Response:** |
| **“Success”, “The device has been bind with the user.”**  **“Fail”, “The user has not logged in yet.”**  **“Error”, “<Error Detail>”** |

|  |
| --- |
| **Function:** |
| **Connect the client to APN.** |
| **Access Point:** |
| **“~/UserServices/connectAPN” (POST)** |
| **Description:** |
| **The service is for setting up the connection with APN for the user. This service should be called when the application just starts up. The connection should be persistent while the application is being used by the user.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **“Success”, “The connection is set up.”**  **“Error”, “<Error Detail>”** |

|  |
| --- |
| **Function:** |
| **Disconnect the client to APN.** |
| **Access Point:** |
| **“~/UserServices/disconnectAPN” (POST)** |
| **Description:** |
| **The service is for disconnecting the connection with APN for the user. This service should be called when the application terminates. The purpose for this service is for releasing resources of the server.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **“Success”, “The connection is disconnected.”**  **“Error”, “<Error Detail>”** |

|  |
| --- |
| **Function:** |
| **Get items bought by the user** |
| **Access Point:** |
| **“~/ProductServices/loadBoughtItems” (GET)** |
| **Description:** |
| **The service is for getting items bought by the user.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **../../../../Desktop/Screen%20Shot%202017-12-30%20at%204.37.45%**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **load items which are being sold by the user** |
| **Access Point:** |
| **“~/ProductServices/loadSellingItems” (GET)** |
| **Description:** |
| **The service is for loading items which are being sold by the user.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **../../../../Desktop/Screen%20Shot%202017-12-30%20at%204.36.56%**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **Get items which the user requests for** |
| **Access Point:** |
| **“~/ProductServices/loadRequestedItems” (GET)** |
| **Description:** |
| **The service is for getting items which are being requested by the user.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **../../../../Desktop/Screen%20Shot%202017-12-30%20at%204.37.45%**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **Load items information into the program.** |
| **Access Point:** |
| **“~/ProductServices/loadItems” (GET)** |
| **Description:** |
| **The service is for loading items for the item list page.** |
| **Parameters:** |
| **none** |
| **Response:** |
| **../../../../Desktop/Screen%20Shot%202017-12-30%20at%204.37.45%**  **“Fail”, “The user has been signed out.”** |

|  |
| --- |
| **Function:** |
| **Load items information into the program.** |
| **Access Point:** |
| **“~/UserServices/getProfilePic/{userId}” (GET)** |
| **Description:** |
| **The service is for loading profile image for a specific user.** |
| **Parameters:** |
| **userId: the id of the specific user** |
| **Response:** |
| **The image data.** |

|  |
| --- |
| **Function:** |
| **Refresh loading item histories** |
| **Access Point:** |
| **“~/ProductServices/refreshLoadingBoughtItems” (GET)**  **“~/ProductServices/refreshLoadingRequestedItems” (GET)**  **“~/ProductServices/refreshLoadingSellingItems” (GET)** |
| **Description:** |
| **The service is for resetting the loading status cookie.** |
| **Parameters:** |
| **None** |
| **Response:** |
| **“Success”, “The cookie has been reset.”** |

|  |
| --- |
| **Function:** |
| **Getting school logo** |
| **Access Point:** |
| **“~/UserServices/getSchoolLogo”** |
| **Description:** |
| **The school logo can be downloaded by calling this service** |
| **Parameters:** |
| **NONE** |
| **Response:** |
| **Image data.** |

|  |
| --- |
| **Function:** |
| **Load selling item cover photo** |
| **Access Point:** |
| **“~/ProductServices/loadSellingItemCoverPhoto”** |
| **Description:** |
| **The service is for loading the selling item cover photo** |
| **Parameters:** |
| **NONE** |
| **Response:** |
| **Image data.**  **Nil will be returned if no more image.** |

|  |
| --- |
| **Function:** |
| **Load bought item cover photo** |
| **Access Point:** |
| **“~/ProductServices/loadBoughtItemCoverPhoto”** |
| **Description:** |
| **The service is for loading the bought item cover photo** |
| **Parameters:** |
| **NONE** |
| **Response:** |
| **Image data.**  **Nil will be returned if no more image.** |

|  |
| --- |
| **Function:** |
| **Load requested item cover photo** |
| **Access Point:** |
| **“~/ProductServices/loadRequestedItemCoverPhoto”** |
| **Description:** |
| **The service is for loading the requested item cover photo** |
| **Parameters:** |
| **NONE** |
| **Response:** |
| **Image data.**  **Nil will be returned if no more image.** |