Types of requests required for basic functionality

* Login 🡪 server needs user data database and active AUTH key database
  + STORE AUTH KEY IN FILE
  + Register user
    - Upload user data to server
    - Create table entry for user data on server
    - Obtain AUTH key from server
  + Check user data (login)
    - Check user data against table entry for email
    - Send AUTH key if successful
  + Log out
    - Send active AUTH key to server and deactivate it
* Annota
  + Send picture to server with intent AND AUTH KEY
    - Receive appropriate response back
      * Intent will most likely be add to Annota
* To-Do List
  + Provide server with AUTH key and receive relevant To-Do Data
    - JSON or XML, not sure which is best to implement
      * Can Java parse JSON as easily as Python?
      * Look into Dynamic Layouts

GENERAL TODO

* Add settings page
  + Log out function
  + Language function
  + Colour theme function
  + Profile function
* Add user registration
* Finalize camera screen
  + Add picture button
    - When picture is taken, client is presented with three buttons on screen
      * Cancel 🡪 scrap picture
      * Crop 🡪 proceed to crop function as described below
      * Send 🡪 send as is, no cropping needed
  + From picture button allow the user to crop photo using zoom and draw approach (explained to client in Meeting 3 – OK, maybe add stock rectangular crop function)
  + Send picture to server which will save it into client specific “library”
    - User “library” consists of a SQL table to keep track of entries and allow for easy searching as well as a folder containing all uploaded images
    - Images will be named based on the time and date at which they were taken and the SQL database will also contain this exact time and date for referencing
    - SQL format as follows:
      * DateTime|Name|Content|Comments|Cat1|Cat2|Cat3
        + DateTime 🡪 DateTime string of when the reminder was created
        + Name 🡪 given by user
        + Content 🡪 annotated text content given by Google API
        + Comments 🡪 given by user to clarify importance
        + Cat1,Cat2,Cat3 🡪 categories, the app will offer 3 levels of categorization (OK)

For example: uOttawa 🡪 TA-ing 🡪 GNG2101

* + User library opens with swipe up from camera screen
    - Arrange annotations by date or category, probably won’t display images
    - Add search function to look for specific text within annotation, category, etc
* Add To-Do Screen
  + Each user will have their own notification SQL table with following format
  + Index|Reminder Name|Comments|Cat1|DateTime|Trigger
    - Index🡪 automatically incremented value, we don’t care much for it
    - Reminder Name 🡪 given to reminder by user
    - Comments 🡪 provided by user to specify task or help remember purpose
    - Cat1 🡪 category, most likely will not include categories for simplicity
    - DateTime 🡪 DateTime string of when the reminder was created
    - Trigger 🡪 trigger data to specify when/where the user will be notified of this task

Request Structure

* LOGIN 🡪 SEMICOLON IS NEW DILIMETER SINCE – IS AMBIGUOUS
  + Receives “LOGIN;email;password”
  + Responds with “uid;name”
* KEYCHECK 🡪 / is delimiter ONLY in SQL database uuid column, everything else is ;
  + App sends “KEYCHECK;auth\_key;name”
  + Server either confirms or denies auth\_key
  + Server uses name to look up associated auth keys
* LOGOUT
  + App sends “LOGOUT;auth\_key;name”
  + Server looks up uuids associated with name and subtracts auth\_key
* REGISTER
  + App sends “REGISTER;email;password;name”
  + Server returns “uid;name” for app to save in local app.key file
* TRANSCRIBE
  + App sends “TRANSCRIBE;auth\_key;name;image;drawing”
  + Both image and drawing are sent as bitmaps converted to strings
  + The server can either deny the request due to a bad auth key or it respond with “TRANSCRIBE\_OK;index;cat1;cat2;cat3” / “TRANSCRIBE\_EMPTY;index;cat1;cat2;cat3” to notify the app whether or not the Google API found text and provide the directory index of the annotation. The annotation will be saved regardless and the user will be allowed to enter details such as annotation name, comments, and categories. Cat1-3 are a list of pre-existing categories in the server so that the user can select one from a drop down menu instead of rewriting them every time. The category will be sent as a two dimensional array where ; is used to delimit cat1 from cat2 and cat3 whereas commas will be used within each cat list
* TRANSCRIBE\_INFO
  + Follow-up to TRANSCRIBE request, this is how the client provides the annotation name, comment, and other data. App sends “TRANSCRIBE\_INFO;index;name;comments;cat1;cat2;cat3”. Server will reply with either “REQUEST\_OK” or some error