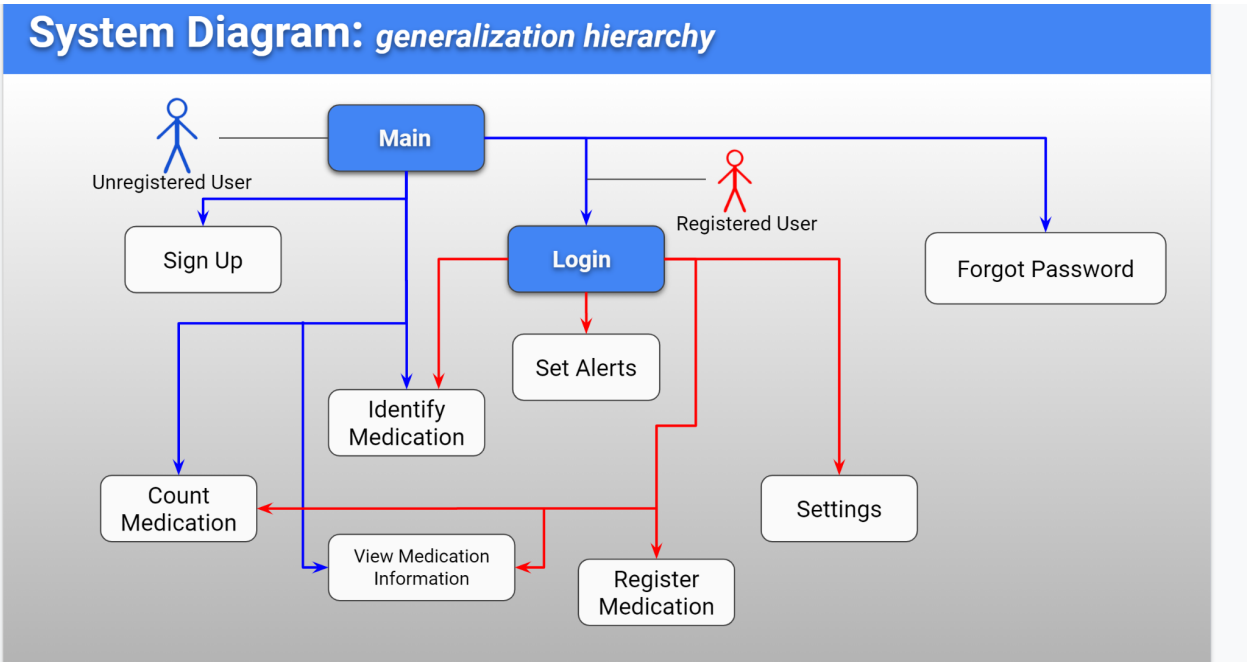
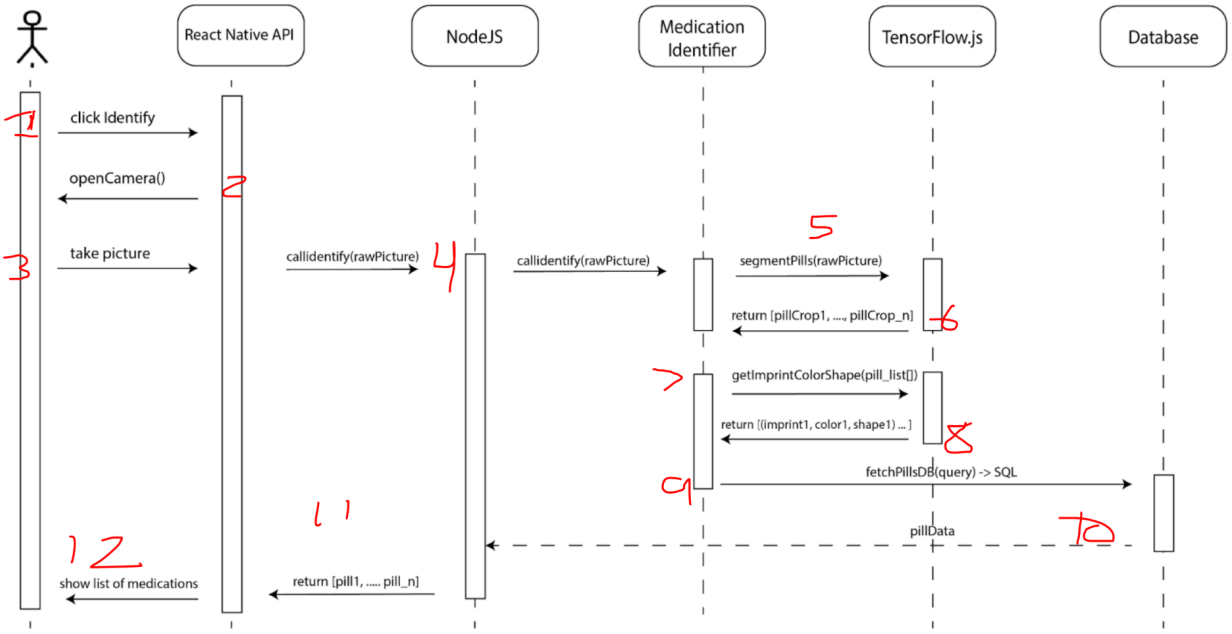


- This form of system diagram is called a generalization hierarchy
- All of the boxes colored in white are features based on our functional requirements
- The blue boxes represent entry points for registered and unregistered users
- The “Main” box represents the entry point of unregistered users (shown as the blue stick figure) and what features they have access to are shown as blue arrow paths.
- For example, unregistered users should be able to sign up for an account. They also can use the Count & Identify medication features for personal use, as well as view medication information.
- The “Login” Box represents the entry point of registered users (shown as the red stick figure) and what features they have access to are shown as red arrow paths.
- The features that are accessible by registered users are expected to require a long-term storage of information and retrieval of that information -- which is why they are indicated in red.
- For example, registered users can register their medication to their account. They can also change their account settings and set alerts.

Sequence Diagram (Identify Medication)



1. This is a sequence diagram that illustrates our functional requirement for Identify Medication. A user will initiate this process by pressing an “identify” button
2. Then, the front-end API will open the device camera
3. The user will position their medications as instructed, and then snap a picture which is immediately processed by the system.
4. The front-end then communicates with the back end (denoted here as NodeJS) by calling CallIdentify with rawPicture as a parameter, which refers to the raw picture represented as a 2 by 2 matrix of numbers
5. The backend then passes the information to a UTILITY CLASS called Medication Identifier. This class will interface with TensorFlow JS to complete the image classification tasks. It first makes a call to the segment_pills function
6. TensorFlow JS then returns an array of images, and each image corresponds to a single pill in the original image ; because we want to identify each individual pill
7. Now, Medication Identifier makes another call to tensorflow JS in order to classify the pill Imprint, pill color, and pill shape.
8. TensorFlow JS returns an array of 3-tuples, and each 3-tuple contains Imprint, color, and shape.
9. The final step is to query a Database that contains all of the registered medications in the United States, which are all hashed by Imprint, color, and shape. We are assuming everything here went fine and we are just trying to retrieve the medication information now.
10. After querying the database, the information is returned to the backend, and the backend will return an array of pill information objects to the front end API.

11. The front end API then represents this information in a user-friendly way, and now the process is done.