List of Features:

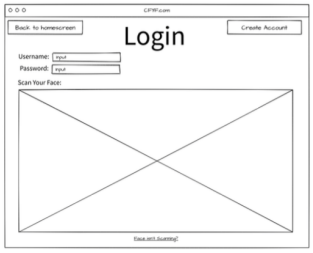
Features done:

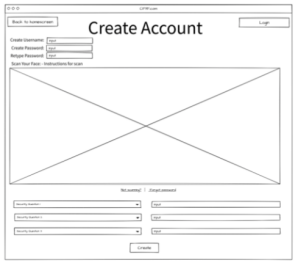
- Browser Camera Access
    - The application should be able to access the user's camera in order to run OpenCV software for facial recognition.
- OpenCV Processing
    - Efficient Facial Recognition using OpenCV
- Basic Text Editor
    - Once the user gets into the "login" page they will have access to a "private" text editor document to write and store whatever they would like.
- Homepage with feature list
    - The first thing the user sees is a homepage with options to sign in, create an account etc.
- Homepage and Camera Access Integration
    - The user should seamlessly transition between the website and the camera integration

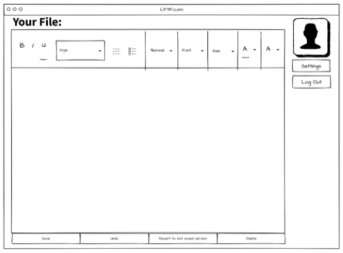
Features in progress in order of importance:

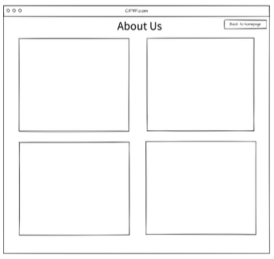
- In-Browser OpenCV Model training
    - The ability for an OpenCV Model to be trained in the browser that can effectively recognize someone's face
- Persistent OpenCV Models
    - The ability for a database to persist OpenCV Models
- Ability to create account
    - A new user that has not visited the website should be able to create a unique username, password and register their face with the OpenCV software
- Sign in to existing account
    - Once a user has created an account they can login by entering their username and password, then wait for the Two Factor Authentication face scan.
- User Data Deletion
    - A user should have the ability to, at any point, delete all data from the database. This includes account information, text document information and all facial recognition information. (account deletion)

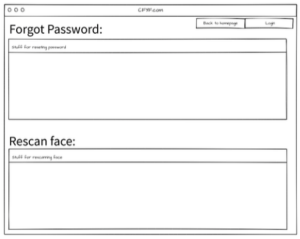
Front end design:

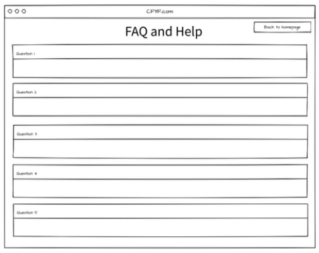




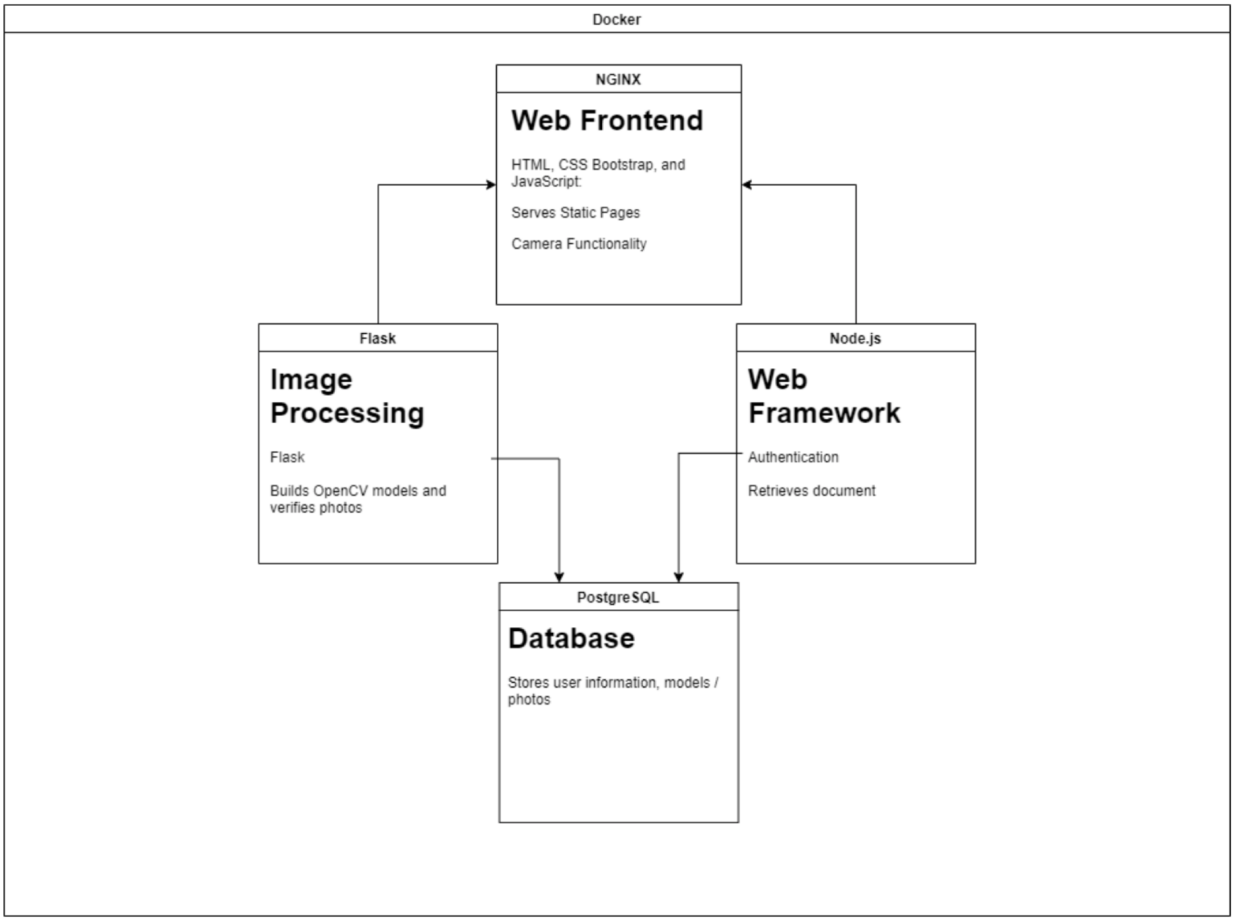








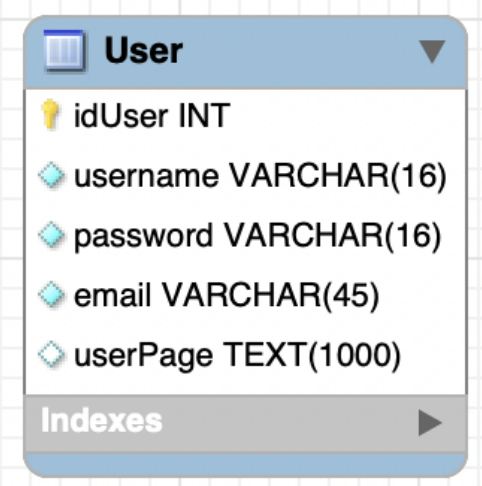
Architecture Diagram:



Web service design:

The only API we are using is TinyDocs, which is a platform that allows you to create an easy to use text editor into your html document. The API receives a key from us to setup the editor and then we can pull the text from the editor later using JS.

Database design:



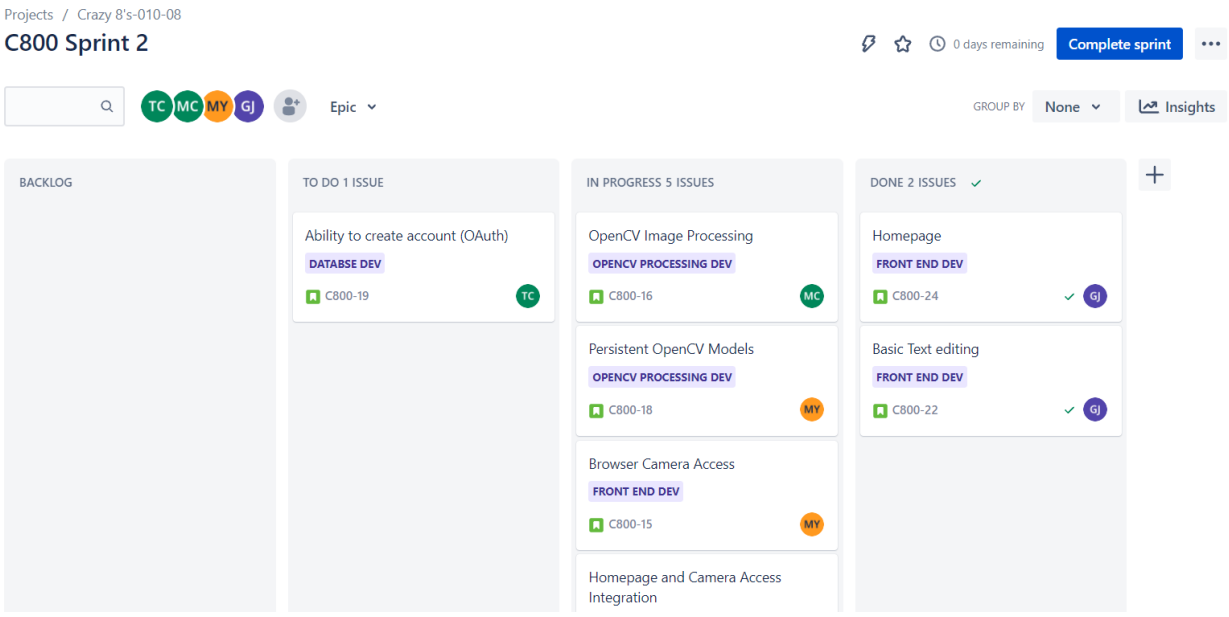
Our database will be quite simple with only 1 table and each user will have 1 row in this table. We will be storing this in a PostgreSQL database.

Challenges:

1. Integrating the front end features with the back end and image recognition.
2. Finding what settings best work for our use in facial recognition.
3. Integrating all apis and frameworks into one working application without them stepping on each other's toes.

Individual contributions:

- Jira Board pic:



- Contributions:
    - Greg: Front end design and implementation alongside the database design.

    - Link: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-08/commit/bb22e364bb5e8b86d99be0aee19f1cbe77ed7bee
- Matt: Accurate facial recognition program finished.
    - Link: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-08/commit/d6012a389ad3c897486194c86745eb7182b4e212
- Tyler: Designing front end html pages along with logo design and color scheme.
    - Link: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-08/commit/7e00a0806a614b8caaa6c8de0da0f62108b0fa7e
- Michael: Overall app integration with creating the Flask app, integrating the front end, and implementing the database.
    - Link: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-011-08/commit/8de60325ca02435b3670d4613285e64bebfed0b2