

**Sprint 2 Retrospective Review**

Team 18: Chunao Liu, Anurag Shah, Jenna Zhang, Yierpan Abuduwaili, Michelle He, Jingyuan Yang

**What Went Well?**

In general, we completed all objectives. Each UI component has the correct action handler to send requests to and receive responses from the server. The UI pages are able to communicate with each other and with the backend server smoothly, and most exceptions have been handled. With token-based authentication, the communication between the frontend and the backend is more secure. The basic functionalities of the app are completed.

* User Story #1

As a developper, I would like to build all models needed for our application and implement API views on backend to accept requests from frontend.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Implement database APIs to fetch, store, and delete images shared among a team. | 3 hrs | Jenna |
| 2 | Build a Code model to store code and implement associated APIs. | 3 hrs | Jenna |
| 3 | Implements APIs for all operations a team needs, such as add a team member, delete a team member, etc. | 4 hrs | Jenna |

**Completed:** Models and APIs for team operations are complete and function as expected.

* + User Story #2  
    As a user, I would like to have a way to reset my password.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Implement reset password functions on the back-end to receive reset password requests from the front-end and send an email with a reset password link to the given email address. | 5 hrs | Jenna |
| 2 | Integrate Django with AWS Simple Email Server to enable email sending functionality | 4 hrs | Jenna |
| 3 | Create an HTML page where users can enter and submit their new passwords. | 2 hrs | Jenna |
| 4 | Implement email resetting api on the frontend to enable the communication between the backend and the frontend. | 2 hrs | Jenna |

**Completed:** After the user clicks the ‘forgot password’ button and enters their email address, they will receive an email address to reset their password. They can change their password via the link in that email and then login with the new password instead of the old one.

* User Story #3

As a developper, I would like to enhance the security of our application by identifying a user, machine, or device before granting access to a resource, network, application.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Implement Django Authentication System to authenticate users and requests. | 4 hrs | Jenna |
| 2 | Implement token-based authentication | 4 hrs | Jenna |
| 3 | Accept token at frontend and pass token with each HTTP request. | 2 hrs | Jenna |

**Completed:** Token based authentication is implemented on the backend and the frontend. A token will be generated upon login and sent to the frontend from the backend. Each request sent from the frontend needs to include that token in the request header as a credential to pass the backend authentication.

* User Story #4

As a user, I would like to see whether creating an account is successful or not.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | The frontend sends the user information gained, including username, email address, and two passwords to backend | 2 hrs | Michelle |
| 2 | Build an API to the database that searches if a username is already there or not | 4 hrs | Michelle |
| 3 | Build an API to the database that searches if an email address is already there or not | 1 hr | Michelle |
| 4 | If either username or email address already exists in the database, send back an error code and display an error message that asks the user to reenter information | 2 hrs | Michelle |
| 5 | If all the verification is successful, add the user to the database, send back a success code, and display a success message | 4 hrs | Michelle |
| 6 | After displaying the success message for a few seconds, redirect to login page | 1 hr | Michelle |
| 7 | Debug and test functionality | 5 hrs | Michelle |

**Completed:** The server receives requests from the client. The backend API correctly checks whether the username or the email address are already in use by other users, and the server sends a response accordingly indicating whether registering an account is successful. The client receives the response and displays a corresponding error or success message.

* User Story #5

As a user, I want to send a typed code written in c and get feedback.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | The frontend sends the code in typen form to the backend | 1 hrs | Michelle |
| 2 | If compilation fails, send back an error code with the error message from compilation and display the error message to the user | 4 hrs | Michelle / Anurag |
| 3 | If compilation is successful, run the code. If it runs without any error, send back a success code and display the success message to the user | 3 hrs | Michelle / Anurag |
| 4 | If there is any runtime error, send back an error code with the error message from runtime and display the error message to the user | 3 hrs | Michelle / Anurag |
| 5 | Debug and test functionality | 5 hrs | Michelle / Anurag |

**Completed:** The server receives the code sent from the client. The backend API replies with a response depending on the compilation, and the program output, or consisting of errors in either step. The client correctly receives this response, and can display whether there are compile-time or runtime errors.

* + - User Story #6 (Library page)

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | display basic info of the code | 2 hrs | Jingyuan |
| 2 | display a small preview of the code | 2 hrs | Jingyuan |
| 3 | when user clicks on the tab they can see the run results and the whole code | 4 hrs | Jingyuan |
| 4 | testing and debug | 2 hrs | Jingyuan |

**Completed:** The user can see a list of all the pages in a list view. They can access each page by clicking on the arrow key and the enlarged view of the code is displayed.

* User Story #7 (group member page)

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | display basic info of each group member | 2 hrs | Jingyuan |
| 2 | display a small avatar | 2 hrs | Jingyuan |
| 3 | Let team leader invite other team member and add team members accordingly | 4 hrs | Jingyuan |
| 4 | testing and debug | 2 hrs | Jingyuan |

**Completed:** The user can see all the team members inside their group. The group leader has access to add or delete members to that group. A small avatar is displayed for each user, on the left side of the view.

* User Story #8 (Multithreading Compiler)  
  As a user, I would like to be able to process my images at the same time as other users.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Make Compiler pipeline multi thread compatible | 4 hrs | Anurag |

**Completed:** The compiler is multithreaded as a result of containerization with docker. It has been tested to work with multiple threads calling it in parallel on different sets of code.

* User Story #9 (OCR)

As a user, I would like the application to tell the difference between handwritten and typeform code images, process images at a slight tilt or skew, and provide outputs that indicate where my errors are located.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Complete integration of OCR and related pipeline into Django backend without errors | 4 hrs | Anurag |
| 2 | Create an image post-processing model to highlight syntax error lines in the output image (if any). | 6 hrs | Anurag |
| 3 | Segment the Handwritten and Typeform page data into words. | 5 hrs | Anurag |

**Completed:** The OCR pipeline is integrated into the Django backend. There are still some minor bugs in the pipeline itself, but the integration with Django has no errors. The image post-processing model works as intended and does not have errors under rotated or high/low contrast images, nor with images that have large white space regions in the middle or on margins.

* + User Story #10 (Camera)

As a user, I would like to take a picture or select a photo for backend processing and receive an output image that shows me the result in a separate window.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Take a picture and send the static picture to backend | 3 hrs | Yierpan |
| 2 | Build a API for image sending and receiving | 5 hrs | Yierpan |
| 3 | Receive the result back from backend and show the result | 6 hrs | Yierpan |
| 4 | The static page should be able to save the picture | 3 hrs | Yierpan |
| 5 | testing and debug | 3 hrs | Yierpan |

**Completed:** The user interface for the Camera page is complete, and layouts function as intended. It includes page redirects, a dropdown menu, a side and top bar, and a navigation bar. Pictures can be taken and sent to the backend. The image picker function is implemented, allowing users to select and upload pictures from their gallery instead of taking a new picture. The static image page displays the result received from the backend after its processing is complete.

* + User Story #11 (Account Page & Redirection bars)

As a User, I would like to see my user info and edit my information and I would like to navigate through pages.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Display user info and they should be editable if the backend database is complete or only thing left should be linking the data | 3 hrs | Yierpan |
| 2 | Implement sidebars and topbars to pages that require redirections | 6 hrs | Yierpan |
| 3 | Combine Pages and components with Jingyuan | 3 hrs | Yierpan |
| 4 | testing and debug | 2 hrs | Yierpan |

**Completed:** The user interface for the account page is complete. Users are able to see their account information, and edit it freely. This only applies for users that are logged in. The profile has various personal fields and an image avatar.

* User Story #12 (Scripting / Backend state transfer)

As a Developer, I would like the backend to have a fully functioning production pipeline. After receiving a compilation request, Django should activate the OCR script with the correct image and fetch the output code into the script that activates the virtual environment. After that, a script should fetch the pipe outcome from the virtual environment and send it to the OpenCV script. In the end, the outcome should be sent back to the user’s end.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Write a script that can pass and run the OCR given a picture and call it after the user uploads a picture | 3 hrs | Anurag |
| 2 | Write a script that can pass the OCR output (code) into a docker environment | 4 hrs | Anurag |
| 3 | Write a shell script or C program to create a pipe that can redirect the output of the docker environment into the system, including stack trace and terminal output | 8-9 hrs | Anurag |
| 4 | Write a script that can pass the stack trace to the OpenCV pipeline to highlight the error | 7 hrs | Anurag |
| 5 | Write a script that can pass the OpenCV output in a formatted way, save the result in the library and send it back to the user | 6 hrs | Anurag |

**Completed:** The pipeline is complete. There is a single facade design script that performs the entire OCR pipeline. That code is able to pass the OCR output (After post-processing) into a docker container, and its output redirected back to the python script (more details in user story 13). The stack trace is post-processed, and line numbers for errors are extracted and passed to the image post-processing filter. This image is then returned to the calling function (Django thread) as a Pillow Image that the backend can process.

* User Story #13 (Virtual Environment)

As a developer, I would like all users’ code to compile in a virtual environment. The backend system must remain the same no matter what code the user is running. If there is any runtime error, the backend will fetch the first line number that causes the problem and throws it into the OpenCV, which will return the original image the user sent with highlighted runtime errors.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Setup a Docker virtual environment for any incoming compilation request | 7 hrs | Anurag |
| 2 | When the OCR production is done or the user sends typed code, create a docker environment and run the code | 7 hrs | Anurag |
| 3 | If the program happens to have a runtime error, send the stack-track to the OpenCV pipeline to highlight any errors in the output | 7 hrs | Anurag |

**Completed:** A docker environment using an alpine linux based environment is created by the python script. This environment saves a piece of C code (input) to a file, compiles it, executes it, and then returns the stdout and stderr of that compilation. That error trace is then post-processed (unnecessary portions removed) and passed down the pipeline. This setup works multi-threaded and has a very low latency (<1.5 s). It is also isolated, so unsafe or malicious code sent to be executed will not interfere with the execution of the backend or of any other user’s code.

* User Story #14 (Image Transfer & Server security)

As a developer, I would like to make sure that all types of mobile Image package & request can be processed by our Backend. Since Android and Iphone have different ways to store and send their pictures, our backend API should be able to accept and store images in base64 encoding, uri and temporary file attachment. Also as a developer, I would also want to enhance my backend security by hiding any sensitive information, closing any unnecessary ports, etc.

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Further enhance the Image transfer API so that it supports multiple image types, such as base64, uri, etc. | 5 hrs | Chunao |
| 2 | Reconfigure the image output in form of uri so that front-end can easily download the image | 3 hrs | Chunao |
| 3 | Helping front-end to finalize the image transfer, ensure the transfer and database correctness | 4 hrs | Chunao |
| 4 | Using encoder to encode our backend security settings so that it won’t become a server vulnerability | 2 hrs | Chunao |

**Completed:** The server can now support image requests and transfers in the following formats: base64, uri, and raw image file data. This image transfer function is lossless and reliable. The front-end can also receive and display the image in an imageField.

* User Story #15 (Apache Server config)

As a developer, I would like to run my server with Apache server module so that I can have a continuous delivery with a much more general service size. We no longer need to specify port number when calling API, and future implementation of server features will be available with Apache server structure

| # | Description | Estimated Time | Owner |
| --- | --- | --- | --- |
| 1 | Download, construct and configure Apache Server on AWS EC2 | 5 hrs | Chunao |
| 2 | Redownload and Recompile wsgi.so so that they have a matching version of python executable | 3 hrs | Chunao |
| 3 | Unit testing to make sure the server is stable | 4 hrs | Chunao |

**Completed:** The server is now running in port 80 with the Apache server structure. It has all the features of a native django HTTP server, including django’s token-based authentication, and did not lose any functionality as a result of the transfer.

**What Didn’t Go Well?**

We managed to completely finish sprint objectives, so it was a large step up from Sprint 1 in general. There were a few areas that were not ideal. We were not able to meet as a group as often, in person or online, as we did the previous sprint. The sprint plan also had to be slightly changed, with a few tasks being moved around, some added, some removed, to better fit changing design ideas.

**How should you improve?**

The first way we can improve is by setting weekly meetings at the start of the week, to make sure that most, if not all, of the group can meet every week. The second way we can improve is to set up better estimates and to plan out design and task division better, to reflect different user stories that are similar in scope.