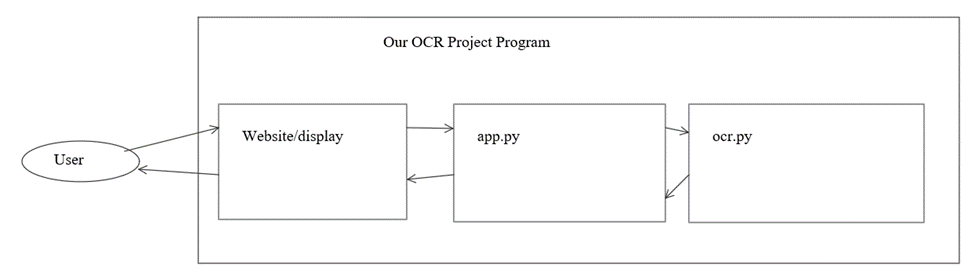
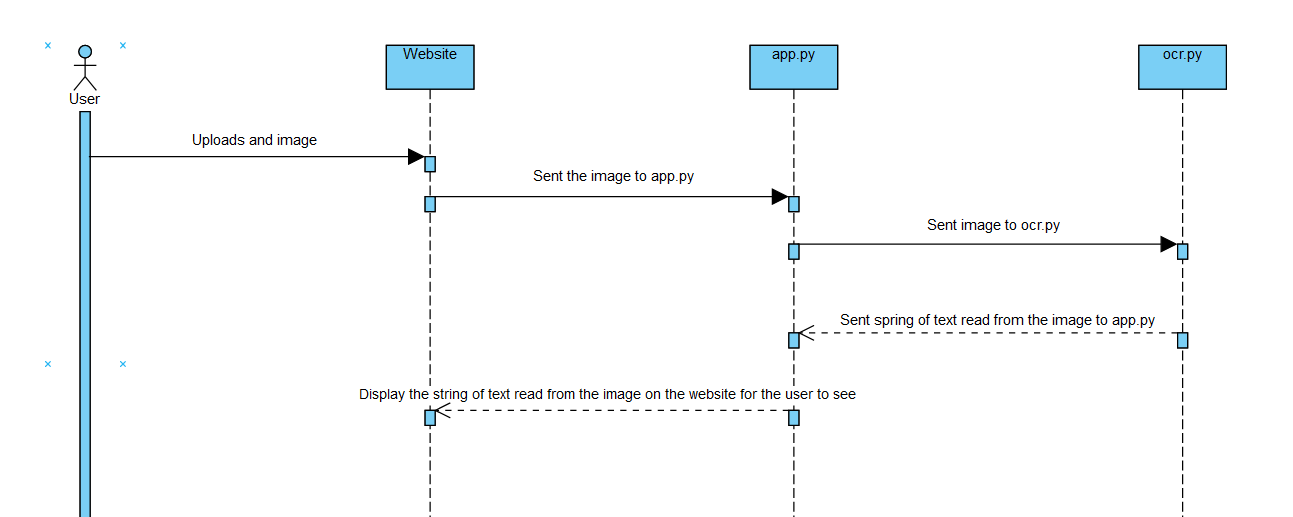
Design Documentation Project 3

1. Identify the Design Paradigm, explain
   1. Service oriented design -We choose service-oriented design because there are multiple functions, or services, that we want to provide for our user. The functions could be separately maintained and used. Our goal for project three is to make a website that can read words from an image file and transfer the words in the image to text that can be copy and pasted on the computer. We want our website to be able to read text and handwritings, in other words, we are making an OCR (Optical character recognition) program. We also plan on adding more features such as translation and pdf conversion to this program for project four, which are the other services that would be added. Nonetheless, our initial intensions are to automate away tedious tasks for our users while having a focus on photo recognition. Instead of having to a long time typing out the texts in an image, the user would easily obtain the text within seconds using our program. Our website supports the majority of picture file types such as pdf, png, jpg, jpeg, and gif. By having a website, it is easier for users with no computer science background to interact with our system. The website setup would just be consisting of two boxes. One box would show the uploaded image, while the other box would show the converted text. This satisfies the black box quality of a service component for our users. Since our users don’t have to be aware of out service’s inner working to use the services on our website. With the easy setup, our users should be able to navigate around our website with ease.
2. Describe Software Architecture of Our Prototype
   1. Our project has the pipes and filters architecture since the system goes through a series of process of transforming input data (image) through computational components (image manipulations) into output data (text). We have two main components in our program, the logic for our OCR, and the user interface of our OCR. The logic portions of our program are in a ocr.py file, and our user interface is in a app.py file. The program starts with the user uploading a file on our website. Then, app.py passes the image file to our ocr.py. The ocr.py would then process the image in a series of image manipulations to output a cleaned image that would be used to extract the text. The ocr.py would then return the extracted text from the image and passes text as a string back to app.py. In our final step, our app.py displays the text on our website for the user to see. If we add the aspect of translation and pdf conversion in project four, it would be an extra step after processing the output text from ocr.py. For translation, the image manipulation and cleaning process would be the same. After that, we would have to determine the language that is being translated. Then the text would be translated before getting display on the website screen. For PDF conversion, the process of transformation will also be the same. There will just be an extra component in data display. The text would just be converted into a PDF that user will be able to download while also being printed on the screen.
3. Use one or more UML to Design
   1. Use Case Diagram
      1. 
   2. Sequence Diagram
      1. 
4. Identify Design Patterns and Explain how to apply them
   1. Iterator (Behavior) – Iterator provides a way to access the element of a system sequentially without exposing its underlying representation. Our website/display contains an upload button and a convert button. Our users will have easy access to the components of our project without exposing our underlying representations to them. This quality satisfies an iterator design pattern.
   2. Observer or Publish/Subscribe (Behavior) – A observer or public/Subscribe design pattern defines one-to-many dependency between objects when a state change in one object results in all its dependents being notified and updated automatically. When a user uploads an image to our website, our left display box displays the image that is uploaded, and our right display box displays the text that is interpreted from the image. Since the two points mentioned above happens immediately and automatically when an image is uploaded and converted, the quality satisfies as an example of an observer or publish/subscribe design pattern.
   3. Adapter, Wrapper, or Translator – An adapter allows classes to work together that could not otherwise because of incompatible interface. We used Python for our backend logic of our program, and JavaScript/html for our front end. In order to connect our front end and back end we had to use a micro web framework called Flask. Flask allows our front end and back end to be able to work together, without it our front end and back end would not be able to work in the same system due to incompatible interface. Therefore, flask satisfies as an adapter.