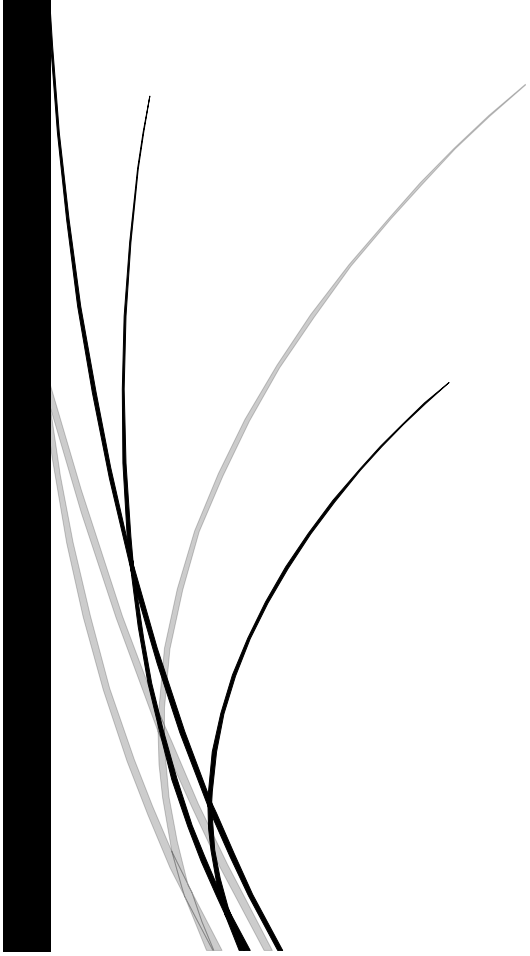
Assignment 3

Summer-2019

Team VAD

Vijay Kalavakolanu, Dong Jun Kim



Task 1(planning and scheduling):

See appendix for the planning and scheduling table.

task 2(Team communication):

See appendix for Github screenshot, link, and Slack link.

task 3(refined system):

Problem statement:

Our product is a book review app that lets you acquire the reviews of any book instantly by taking a picture of it. We use an image reading system to identify the book before searching platforms like amazon, Goodreads, and many other helpful sources. This makes the reviews precise due to the accuracy of the content we are searching for. This product is for all the book readers but specifically for high schools because they do a lot of summer reading assignments and it's helpful for them to choose the more interesting books. it saves time and is more effective because it shows the reviews for the exact book you are looking for and nothing else. A few alternatives would be searching for the book manually on the internet or libraries, which take more time and is less accurate. This project is worth the time and effort to develop because not only does it help the customer but in cases like high schools or even college, the students who must read the books that the institution chose wouldn't have to worry about reading monotonous books. To build this system we'll be using "TINEYE" which lets us search using just an image and its API allows us to implement a graphical user interface which lets the customers to use their own images to search for reviews on the book they are looking for. This project is interesting because we will be combining two different platforms into making something very unique. We are using a reverse image search engine and linking it to sources like amazon,

Goodreads, LibraryThing, and a few others. This saves the user from the need to search for the book manually, only thing they'll need to do is take a picture and let our software do the rest.

Requirements:

Functional requirements (User requirements):

- The user shall be able to login/ signup in the book review app.
- the user shall be able to upload a picture from camera roll or by taking a picture.
- the system shall be able to read the image and identify the book that the user is looking for.
- Once the system identifies which book the user is looking for, the system shall surf through the websites linked to the app to find all the book reviews related to that specific book.
- the system must contain libraries of all the sources used like Goodreads, LibraryThing, etc.
- the system shall offer different ways of listing out the results for the user (low-high ratings, latest-oldest ratings, most popular ratings, etc.)

Non-Functional requirements (System requirements):

- The system must take less than 10 seconds to load up all the results.
- the system must keep all the searches private.
- the system must be free to use.
- the app must contain an "upload image" button and a search button.
- the system must be able to run on smartphones.
- the system must have a high level of RAM in order to process the request at a high pace

Use-case system description:

Summary: The user takes a picture of the book they are willing to buy, and the system prints out a list of reviews from credible websites.

Basic course of events:

1. Once the user opens the application, he/she sees an option saying, "Choose a picture or upload a picture".
2. User selects the "upload a picture" option and gives permission for the application to use the camera.
3. User selects a picture or takes a picture and submits, completion of use case *upload a picture*
4. User allows the application to access the camera on their smartphone.
5. User takes a picture of the book and hits submit.
6. System prints all the results and lists them on the application to show the user all the reviews, completion use case *print results.*

Alternative Paths:

- If the picture is unclear in step 4, the application must produce an error message saying “picture unclear. please try again”.

Exception paths:

- If the user skips step 4(forgets to upload a picture), the system must remain on the main screen (no action is done).

Precondition:

- *upload a picture* use case must be completed.

Postcondition:

- The results containing all the book reviews must be printed out and sorted in the application for the user to examine.

See appendix for the revised use-case diagram.

task 4(System modeling):

4.1(architectural modeling):

A Layered model will be used as our architectural design model due to its simplicity and effectiveness. using this model ensures that each step of the process can be dealt with individually and that although they are a step by step process, we can independently test each of the layers.

The 4+1 model helps identify all the different views of the project and to cover all angles as to how this can help the user and make it as easy as possible for the developer. Logical view, development view, process view, and physical view are the angles we covered and related them to our use cases in order to make a relationship between them so once the making of the application begins there will be minimal amount of thinking and planning.

See appendix for the architectural design pattern diagram.

4.2(Behavioral modeling):

See appendix for the Sequence diagram.

Task 5 (Implementation):

```
┌──[x]─[dongjun@parrot]─[~]
└──╼ $python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'1.11.21'
>>>
```

```
 1 import unittest$
 2 $
 3 class TestStringMethods(unittest.TestCase):$
 4 $
 5 ^Iimage = 'pictureofbook.png'$
 6 $
 7 ^Idef verifyPic(self):$
 8 ^I^Iself.assertEqual(image[-4:],'.png')$
 9 $
10 if __name__ == '__main__':$
11 ^Iunittest.main()$
~
~
~
```

Task 6(Testing):

Identity Feature:

1. Identify Picture

Feature One, Identify Picture:
Partition Input: verifyPic(file_object = open("filename","r"))

- The user takes a picture from the file
    - Possible partitions are files ending with either .png, .jpg, .jpeg file signatures.

Test Specification:

- Filename: must be in a .png, .jpg, .jpeg format that is under 5 MB.

Test Case:

#1:

- Inputs:
  - pictureofbook.png
- Outputs:
  - Retrieves review information of book

#2:

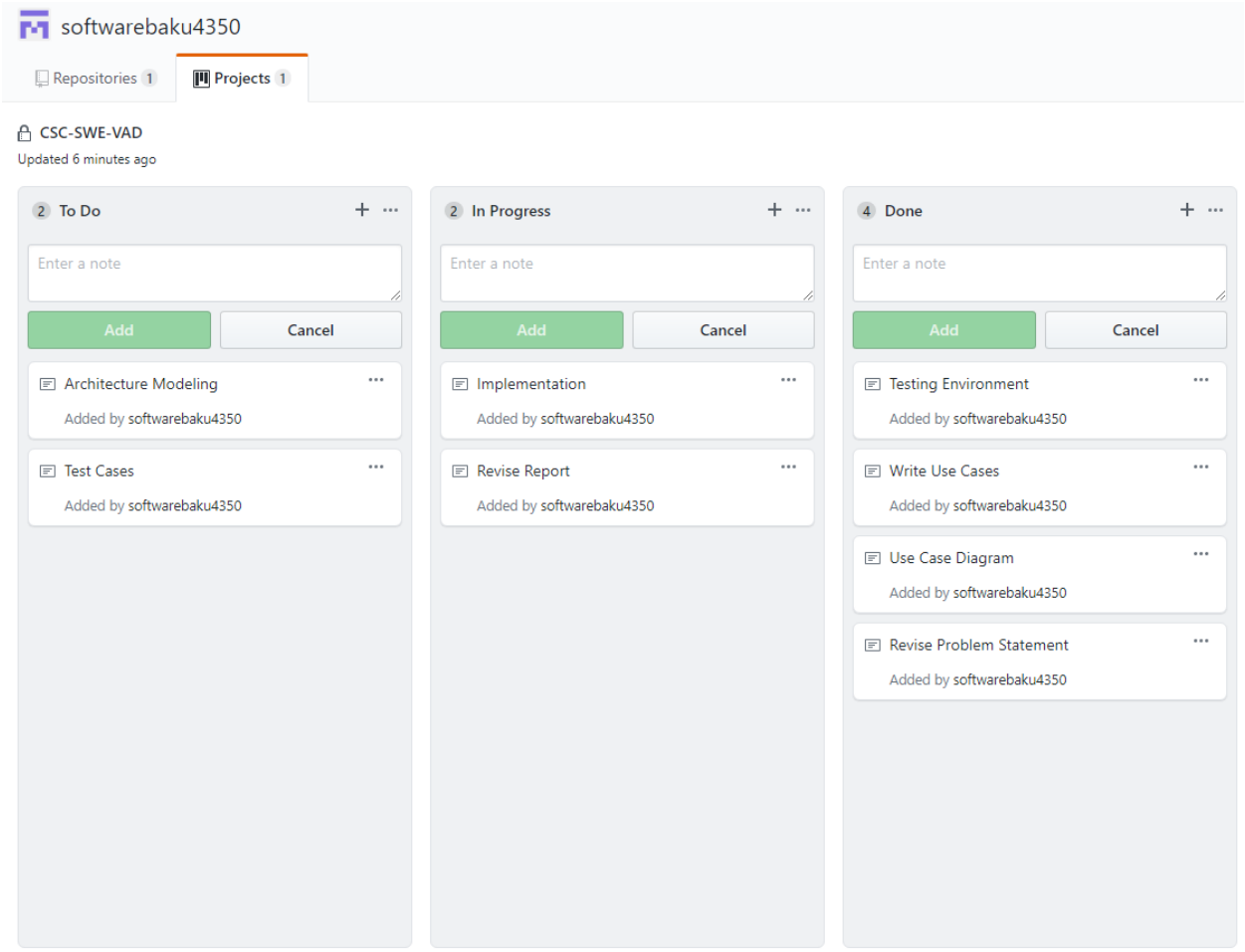
- Inputs:
  - filethatdoesntbelong.pdf
- Outputs:
  - File denied: file not recognized, try again with another file.

#3:

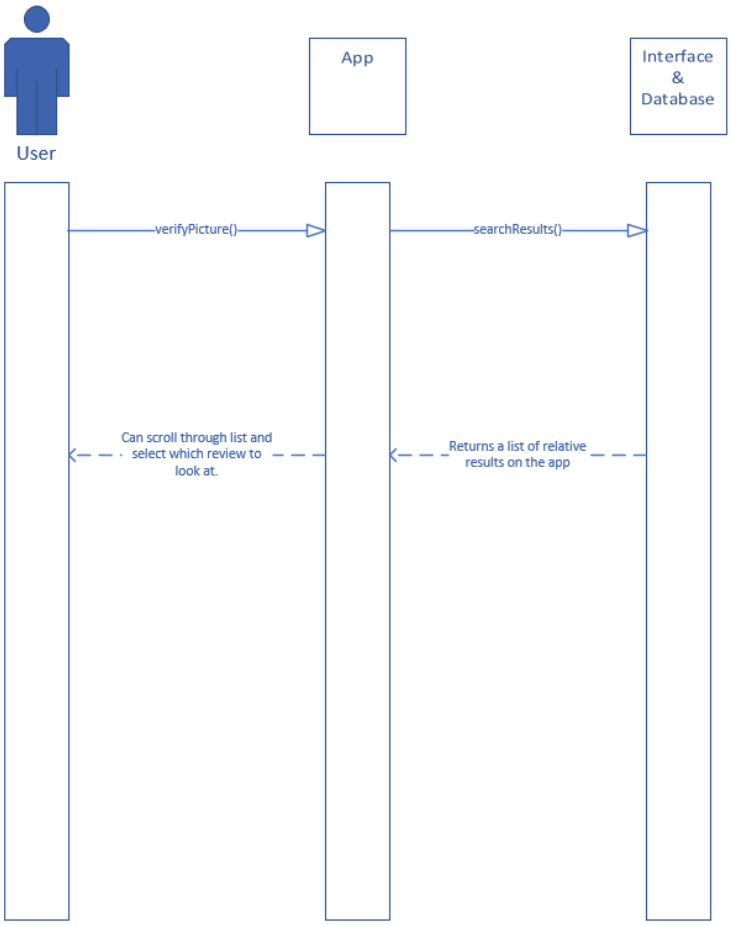
- Inputs:
  - averylargefile.png
- Outputs:
  - File denied: file exceeds 5 MB, try again with another file.

Appendix

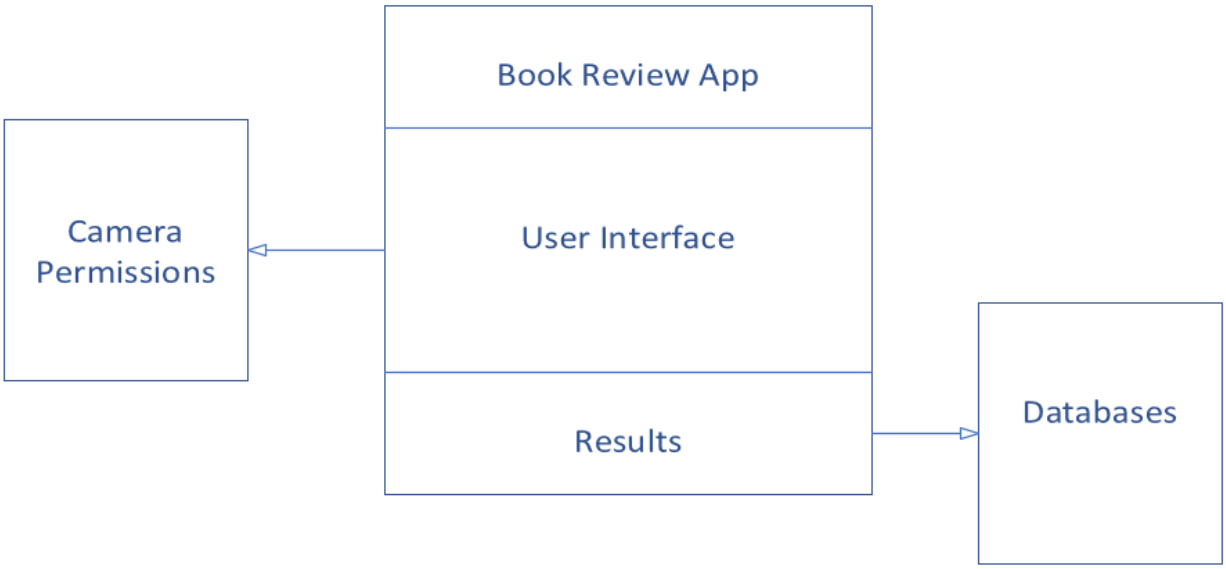
Github:



Process view (Sequence diagram):

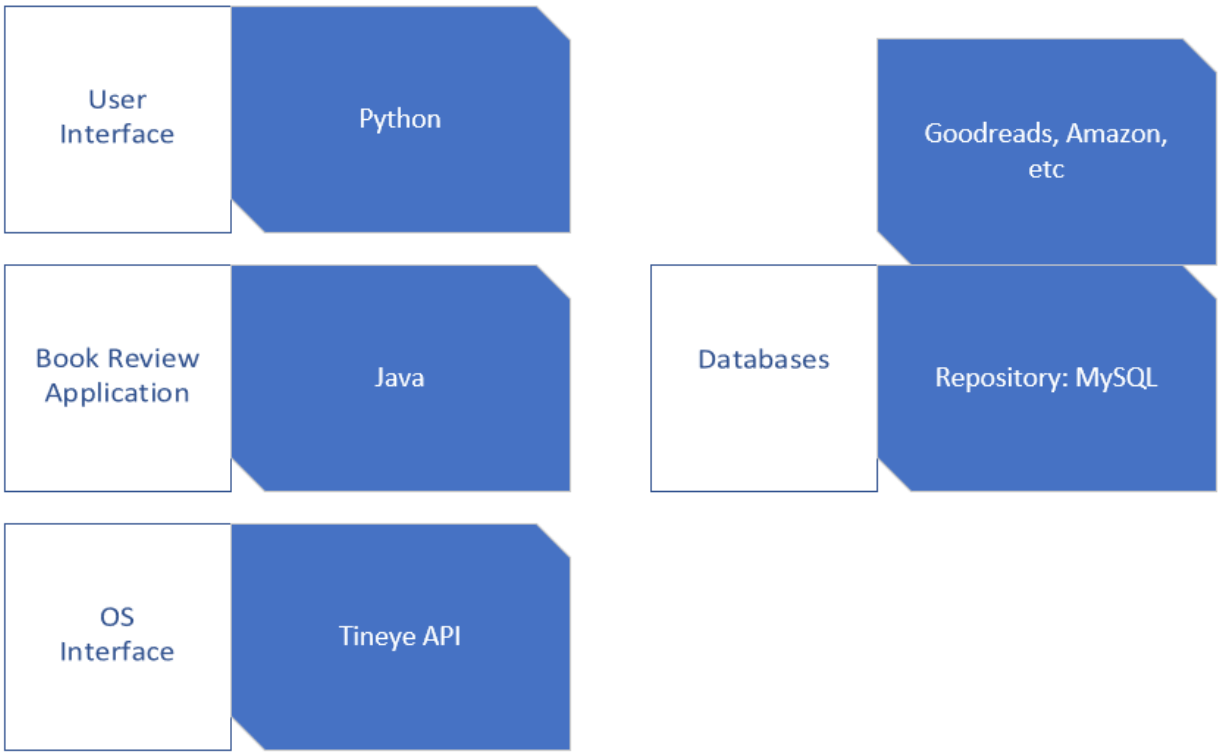


Physical view:

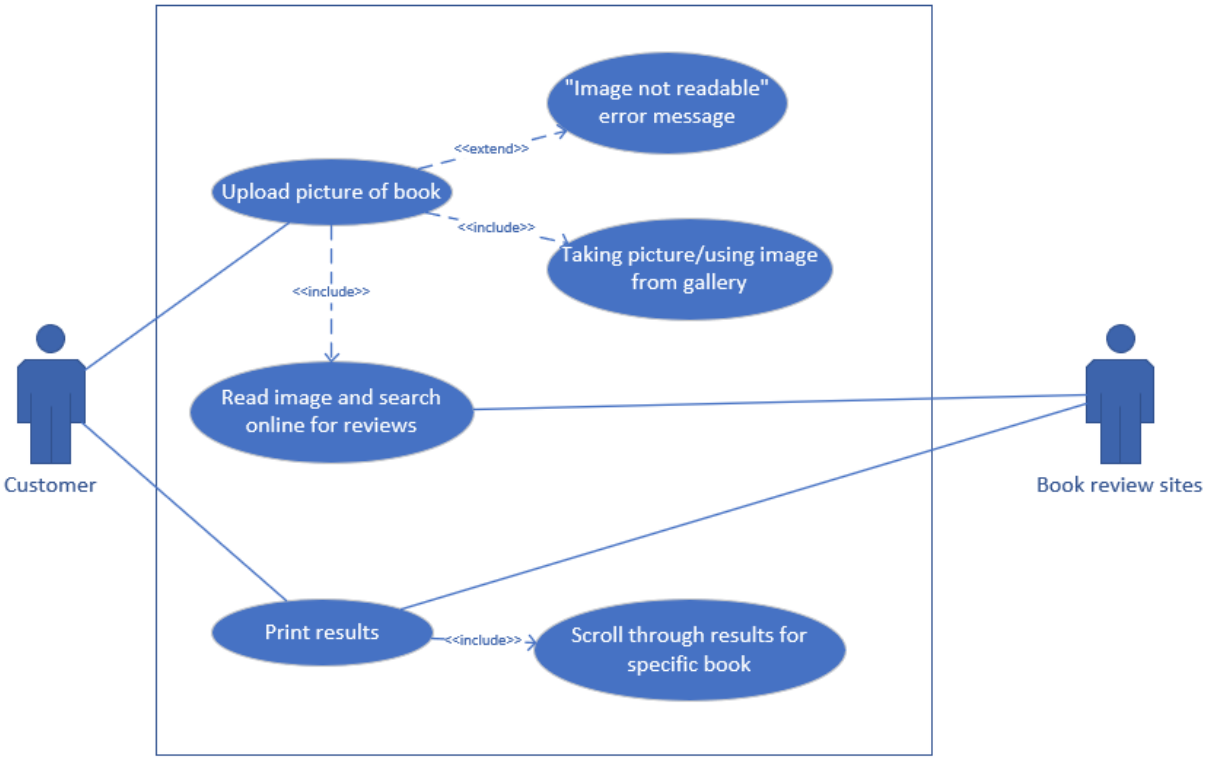


logical view:

Development view:



Revised use-case diagram:

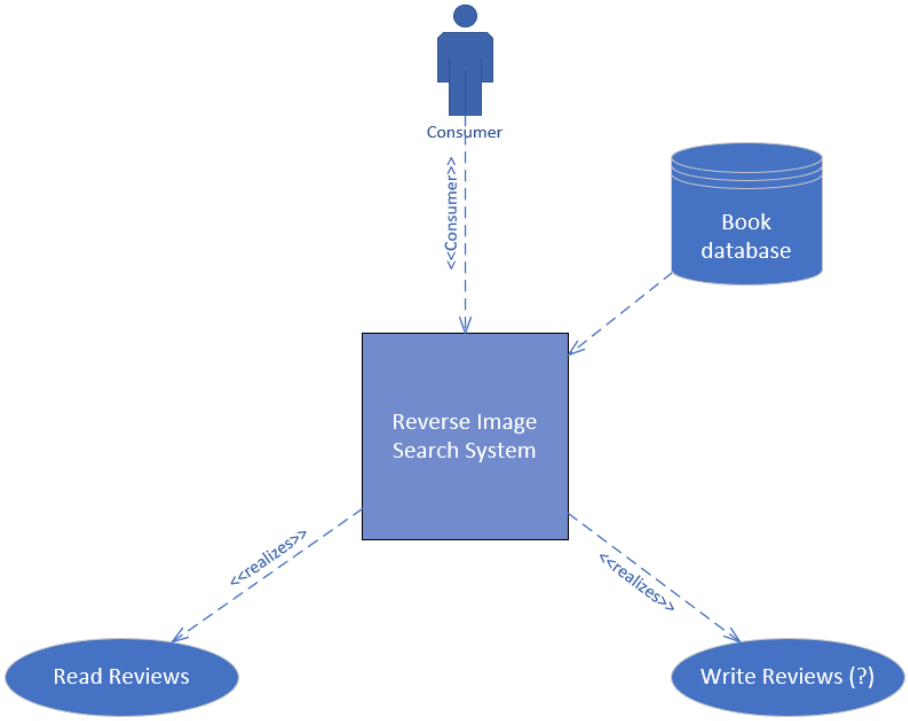


Implementation:

```
┌─[✗]─[dongjun@parrot]─[~]
└──╼ $python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'1.11.21'
>>>
```

```
 1 import unittest$
 2 $
 3 class TestStringMethods(unittest.TestCase):$
 4 $
 5 ^Iimage = 'pictureofbook.png'$
 6 $
 7 ^Idef verifyPic(self):$
 8 ^I^Iself.assertEqual(image[-4:],'.png')$
 9 $
10 if __name__ == '__main__':$
11 ^Iunittest.main()$
~
~
~
```

New context model:



Github: https://github.com/softwarebaku4350/CSC4350/tree/master/Assignment-3

Youtube: https://youtu.be/yJalrk7Iwmo

Slack: https://softwareengin-yfd3984.slack.com