Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

Second Prototype Testing Report

Reproducibility, Reusability, Readability (RE3)

Submitted to

Ana Trisovic
anatrisovic@g.harvard.edu

by

Team #5
RE3

Team Members

Andreas Francisco De Melo Oliveira andoliv@bu.edu
Ethan Hung ehung@bu.edu
Jyotsna Penumaka jyotsnap@bu.edu
Layan Bahaidarah layanb@bu.edu
Lukas Rosario lukasr@bu.edu

Summary of Equipment and Setup

Hardware:

- Computer/Laptop - *Thats is it!*

Software:

- Front End
    - React JS web application
    - Tailwind CSS
    - Firebase JavaScript SDK
- Backend:
    - Google Firebase
        - Firestore
        - Storage
        - Authentication
    - Python Scripts/Notebooks
    - Flask
    - Miniconda3
    - Docker
    - SocketIO

Set Up:

Since our project is just software based all of the set up required is to run npm start to load our web app. Also to run the server backend code for the readability page you can execute the following commands:

- Go to lb/readability_reproducibility_UI
- To open the project on a web browser, follow the following steps:
    - cd 21-05-Re3/re3-client
    - npm install
    - npm start

To run the flask server that lets a user execute the docker file :

- To start the flask server
    - cd 21-05-Re3/docker-app
    - python3 main.py

Detailed Measurements Taken:

The criteria for successful running and output is as follows:

1. The user was directed to the *Dashboard Page* when the project was opened in the web browser and after the user logged in.
2. From the *Dashboard Page*, the user navigated to the *Rating Page* and showed our readability survey from last semester.
3. After returning from the *Rating Page* back to the *Dashboard Page,* the user navigated to the *Readability Page* where the user was able to upload unseen code to our ML model and output a readability score
    a. The user obtained a red, yellow, and green readability score.
4. After returning from the *Readability Page* back to the *Dashboard Page,* the user navigated to the Reproducibility page.
5. The user was able to specify:
    a. R version of their choice
    b. select the code and data files related to their project
    c. specify the order in which we need to run the code files
    d. specify the author of the paper
    e. specify title of the research project
    f. specify keywords describing the research project
6. The above information must be successfully stored in our database (firebase firestore)
7. After submitting the above information the user is directed to a page where 2 sample R files are run inside a container.

a. The first file needs to output an import error (This is intentional). Our plan is to let the user know what additional information they need to provide as dependencies through these error logs.
b. The second file needs to run successfully with the RVersion specified by the user, in this case it would be 3.6.0.

8.

Conclusions:

Overall, prototype testing went well. The group demonstrated our current progress to Professor Pisano and Joseph Greene, and received helpful feedback for the future of our project. Following prototype testing, a few conclusions were made by the group. Firstly, the team realized that it would be extremely useful to add a profile page, which could display all the projects that the user has uploaded using our reproducibility service. This will require us to work more with google firestore. Another conclusion from testing was that it would also be useful to add a suggestion box to our readability page, which would suggest ways for the users to improve their score after they input an R code file with a low rating. Also things that are already part of the plan but have not been finished, include, moving the reproducibility functionality onto Google Cloud, running the containers for the reproducibility part remotely and expanding the level of specifications when making the docker containers. For the last part an example which our group will finish before the final demo is including the list of packages in the R file that need to be installed before the code is compiled.

Future Changes:

- Add information to the profile page:
    - The projects the user has uploaded using the reproducibility service.
    - The R code files and the ratings they scored.
- Add a suggestion box to the readability page that suggests ways to improve the readability score after obtaining a low readability score.
    - For example, warning the user if their average line length is too high.
- Move the reproducibility infrastructure to Google Cloud Build

- Run the container remotely. Currently, it is being run locally.
- Pull the files uploaded by the user into the container and use the specified R version, dependencies to run the project code files and output the resultant logs.

What should you expect?

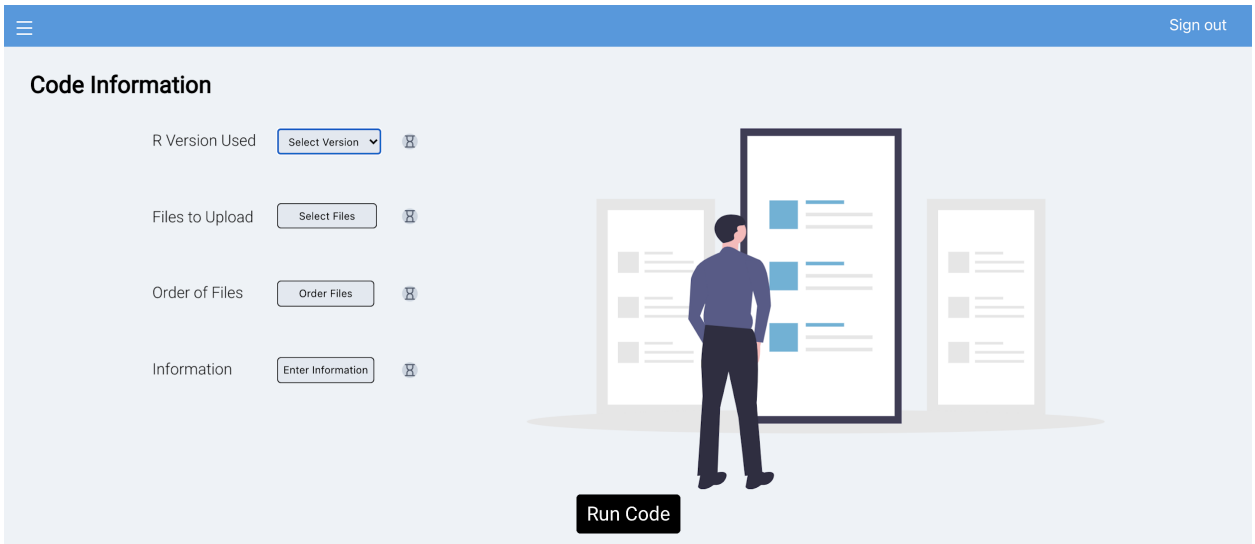

Figure 1 : Collect information about the research project dependencies to reproduce code

Figure 2 : Run the docker file using the specified R version and stream the output logs

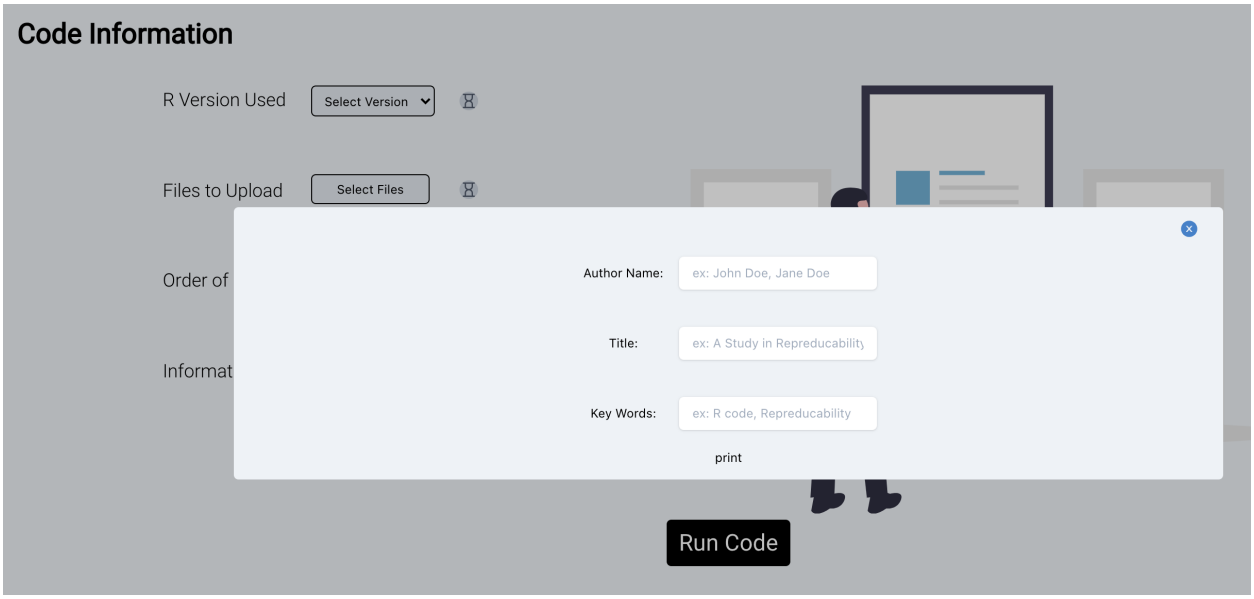

Figure 3: Collect information about the research project, the user plans to upload

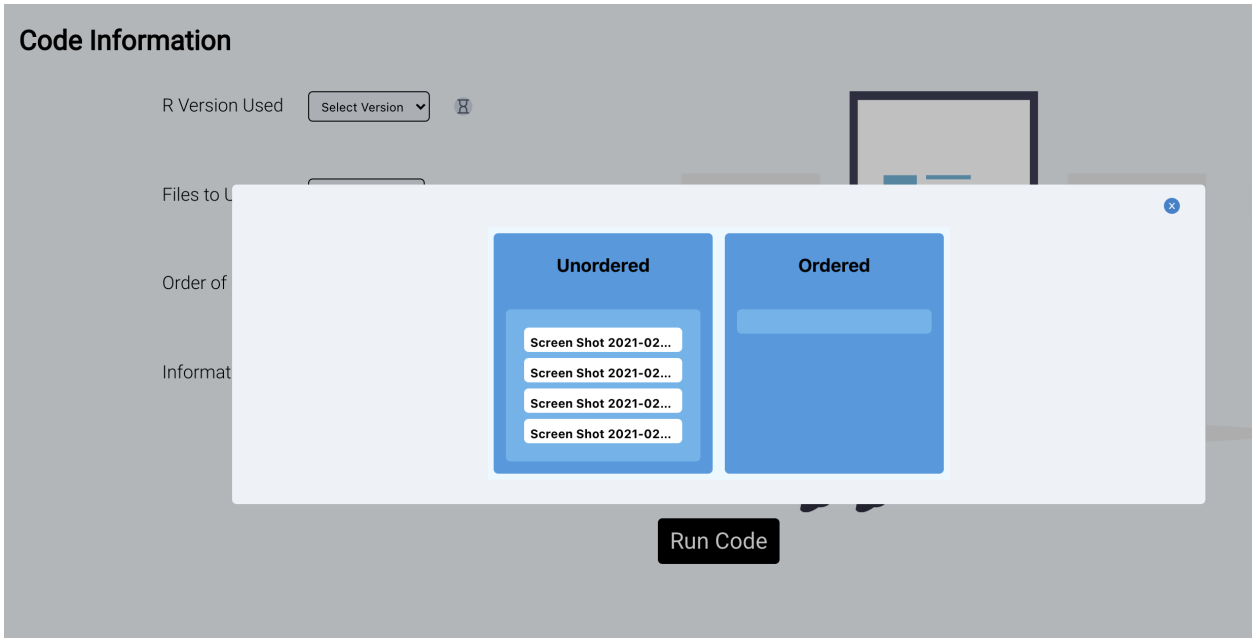

Figure 4: The user should be able to upload their files and order the files depending on how the user wants to run the files in the container.

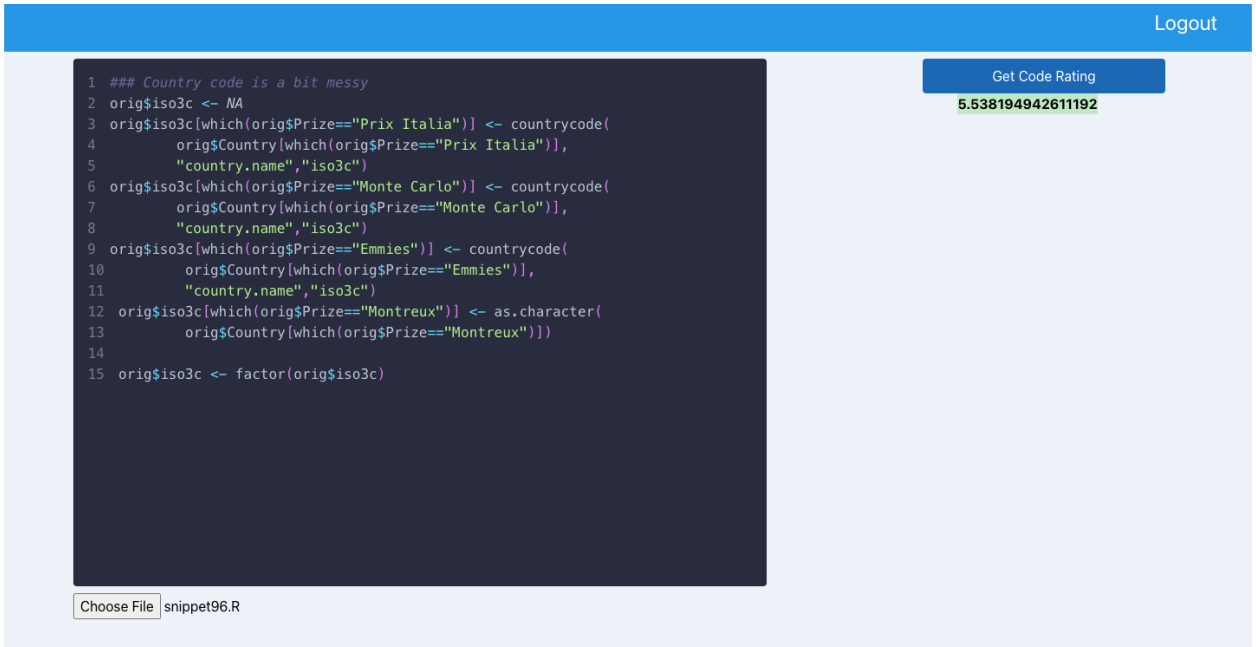

Figure 5: Upload file on the readability page and the code readability rating of such file

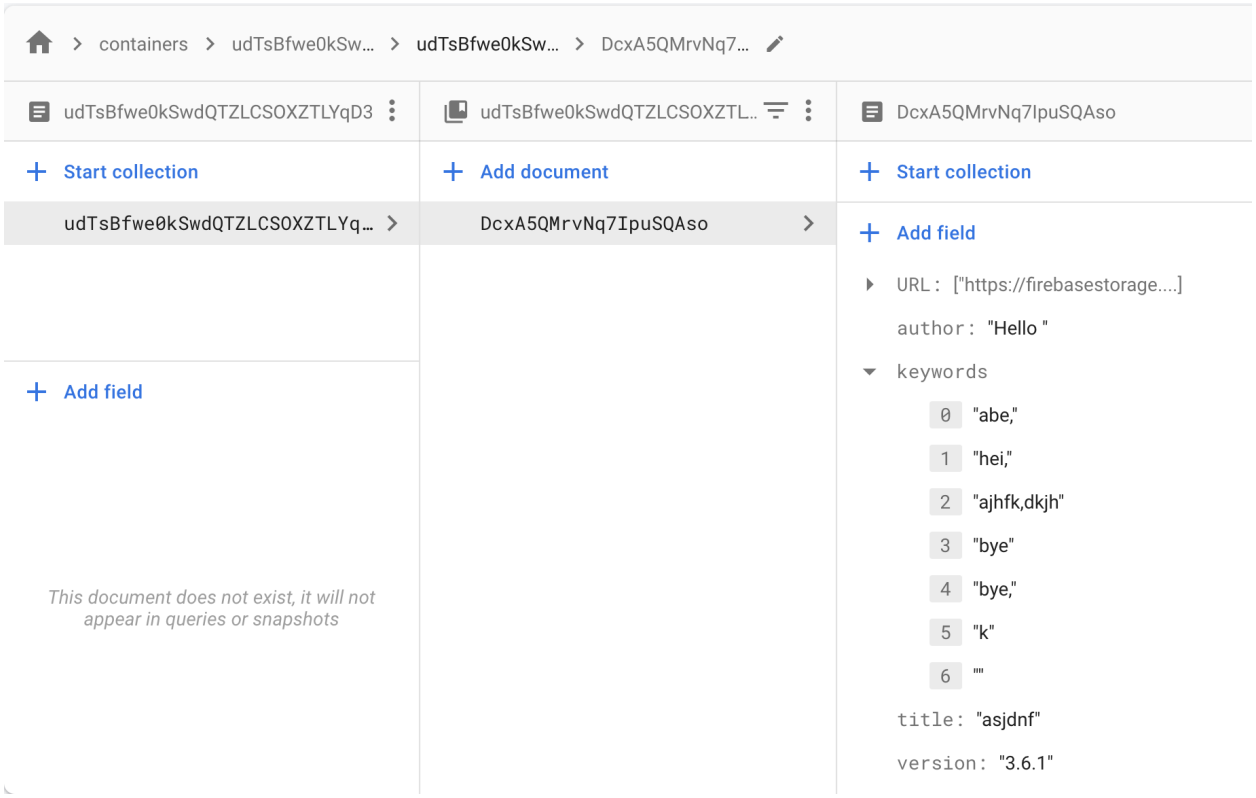

Figure 6: Upload the files details to Firestore, the details include the URL, author, keywords, title of the project, R version

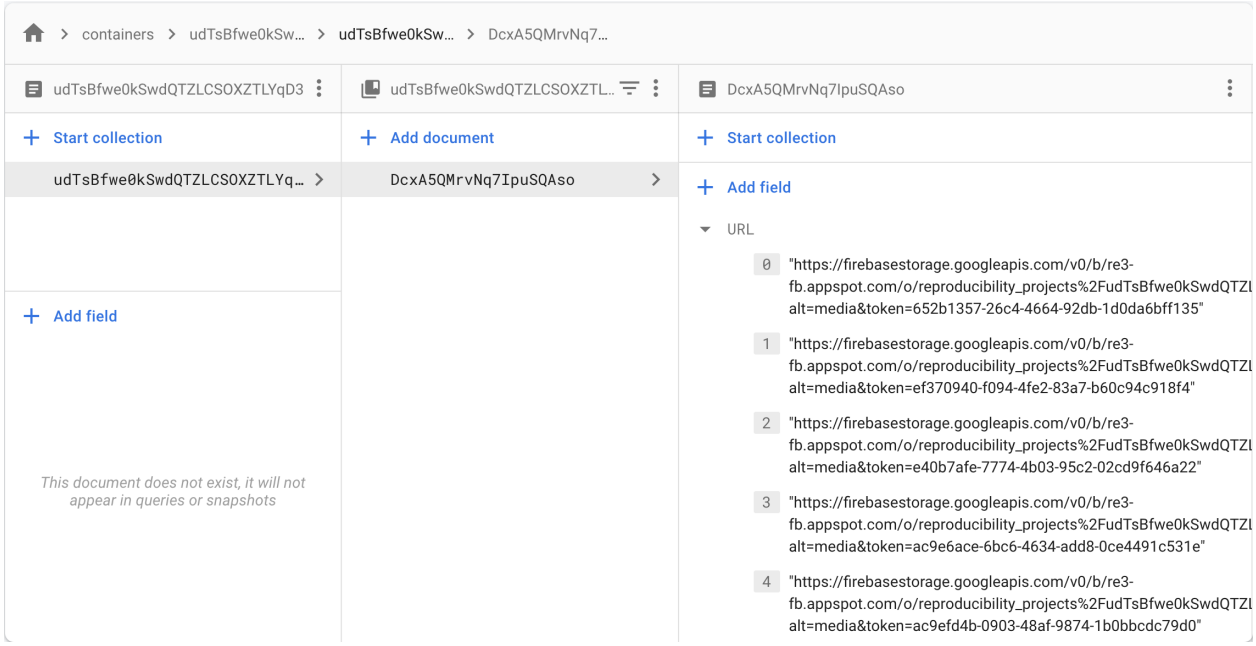

Figure 7: Upload the file projects to Firebase Storage and save the links in the firestore