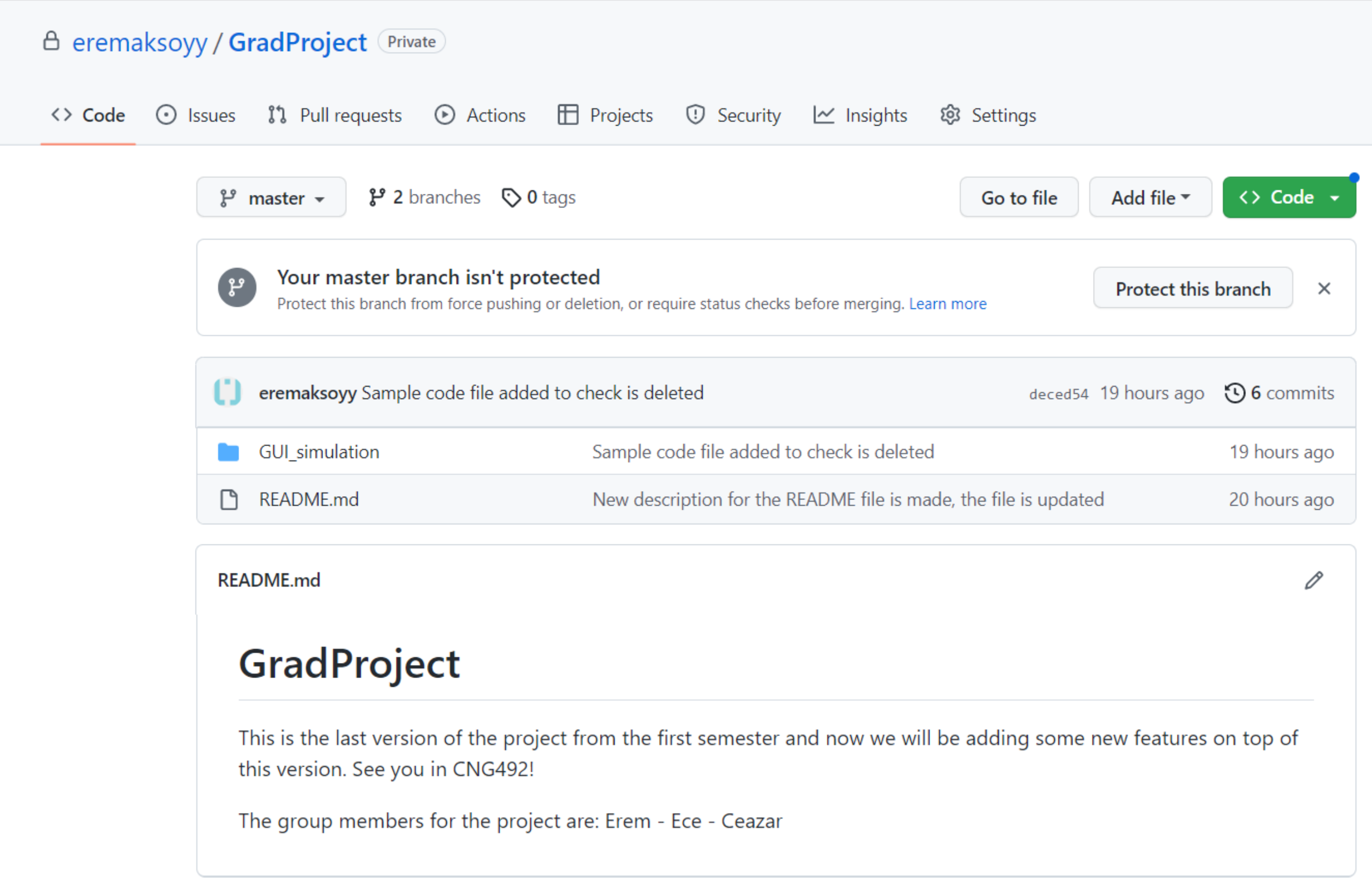
CNG 492 – COMPUTER ENGINEERING DESIGN II

CONFIGURATION MANAGEMENT PLAN

Fatma Erem Aksoy - 2315075
Ece Erseven – 2385383
Ceazar Hatokah - 2456119

Source Code

The source contains the main code and the version tested and working correctly. The source contains all of the calculations of the outputs related to queue and servers; in addition to that, it contains the front-end code which represents our GUI simulation, such as the input page, menu, help, etc. For our project, all the source code files will be saved in a folder inside a GitHub repository. These files will be shared and accessed by all team members; in addition to that, any changes or updates done to the source code will be committed by all team members. Commit is a change to the files containing a specific unique ID to those files, allowing users to check any changes done to a particular file. The repository we created and used to share our source code files can be seen in Figure 1 below.



Your master branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before merging. Learn more

eremaksoyy Sample code file added to check is deleted deced54 19 hours ago 6 commits

GUI_simulation	Sample code file added to check is deleted	19 hours ago
README.md	New description for the README file is made, the file is updated	20 hours ago

README.md

GradProject

This is the last version of the project from the first semester and now we will be adding some new features on top of this version. See you in CNG492!

The group members for the project are: Erem - Ece - Ceazar

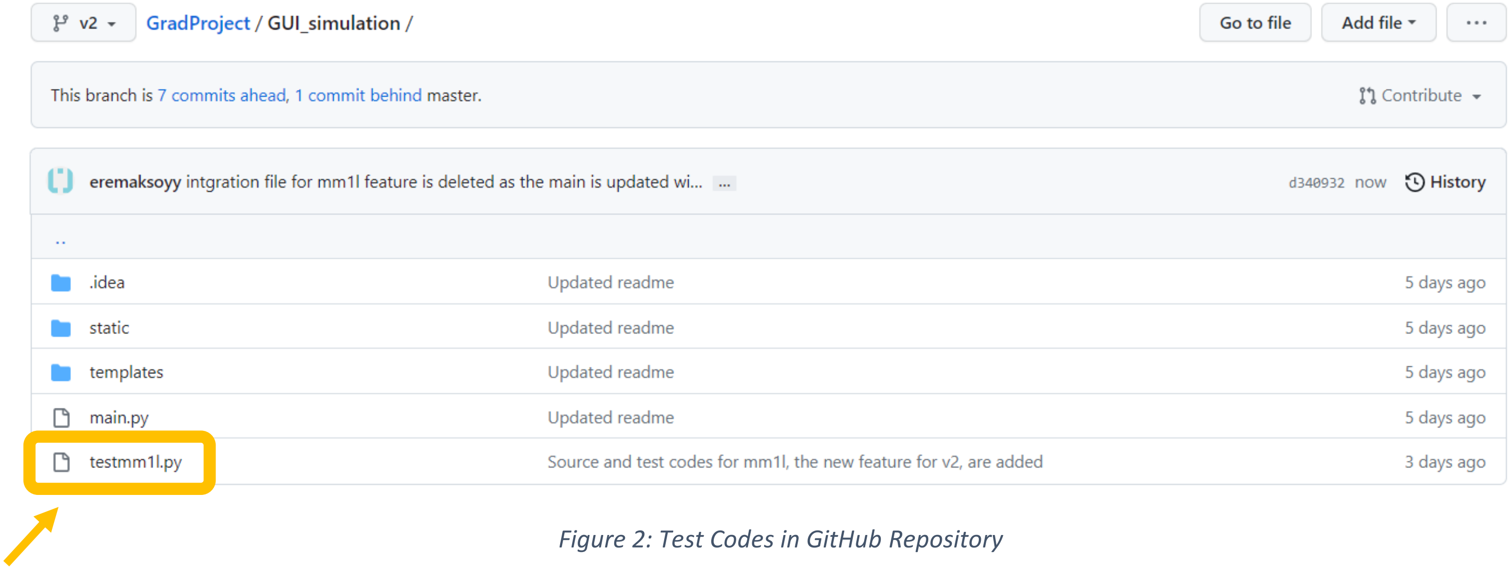
Figure 1: Source Codes in GitHub Repository

Tests

The current version of the test code for the project does the following, takes inputs (Arrival rate, Service rate, Queue size) and then calculates the given outputs:

- Traffic intensity of the queue = Arrival rate/ Service rate
- Average number of customers = traffic intensity / (1 – traffic intensity)
- Average time spent in the system.
- Number of customers in the queue

In our project, we will have some test codes to test if the results and calculations of the output are correct and accurate before using them in our project. For instance, when we have a new feature, mm1l having a limited number of entities unlike the previous version mm1 in this case, after writing the code and testing the output for the functionality we desire, we upload the code to GitHub stating its name that it is a test code file, testmm1l.py file in Figure 2. So that in case of any issue in the code, other members can also make changes before updating the main.py file and making the integrations with Flask. In GitHub, branching will be used for any test processes for our test codes before putting these codes into our source code and making the integrations. For every new feature we will add, we will test it and if the testing is successful, updated or new files of the new feature will be merged into the last version of the project. So we will have new versions' files of the project without interruption or causing any conflicts with the help of branching. Figure 2 below shows how we hold a test file in GitHub in order for all the members to see and try it, so in case of any error is spotted by any member, they can add a comment, and testing can be repeated.



This branch is 7 commits ahead, 1 commit behind master.

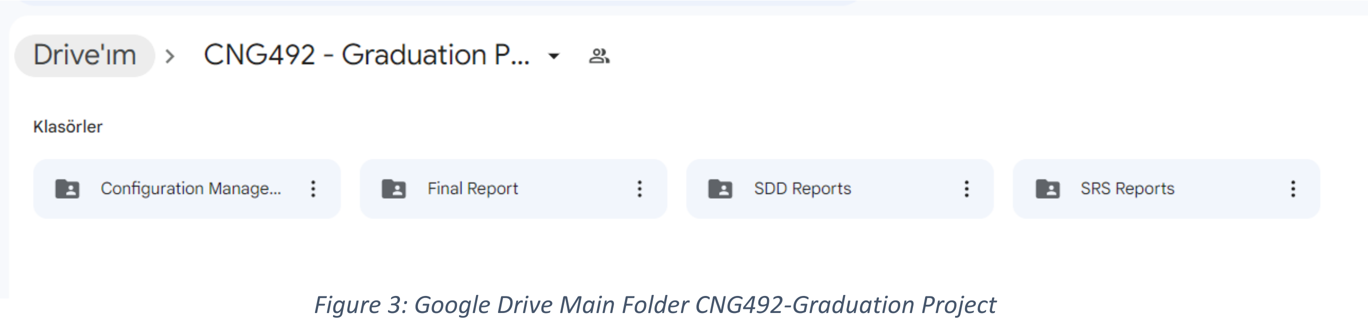
eremaksoyy intgration file for mm1l feature is deleted as the main is updated wi... d340932 now

..		
.idea	Updated readme	5 days ago
static	Updated readme	5 days ago
templates	Updated readme	5 days ago
main.py	Updated readme	5 days ago
testmm1l.py	Source and test codes for mm1l, the new feature for v2, are added	3 days ago

Figure 2: Test Codes in GitHub Repository

Document/Files

To keep track of all the project reports we have for the project, we use Google Drive. All the members have access to these documents with the link provided to them. The name of the folder that contains all the documents is CNG492 – Graduation Project, and this folder holds other sub-folders that specify by their names which files they hold in Figure 3. How one of these folders looks can be found in Figure 4 below.



Drive'ım > CNG492 - Graduation P...

Klasörler

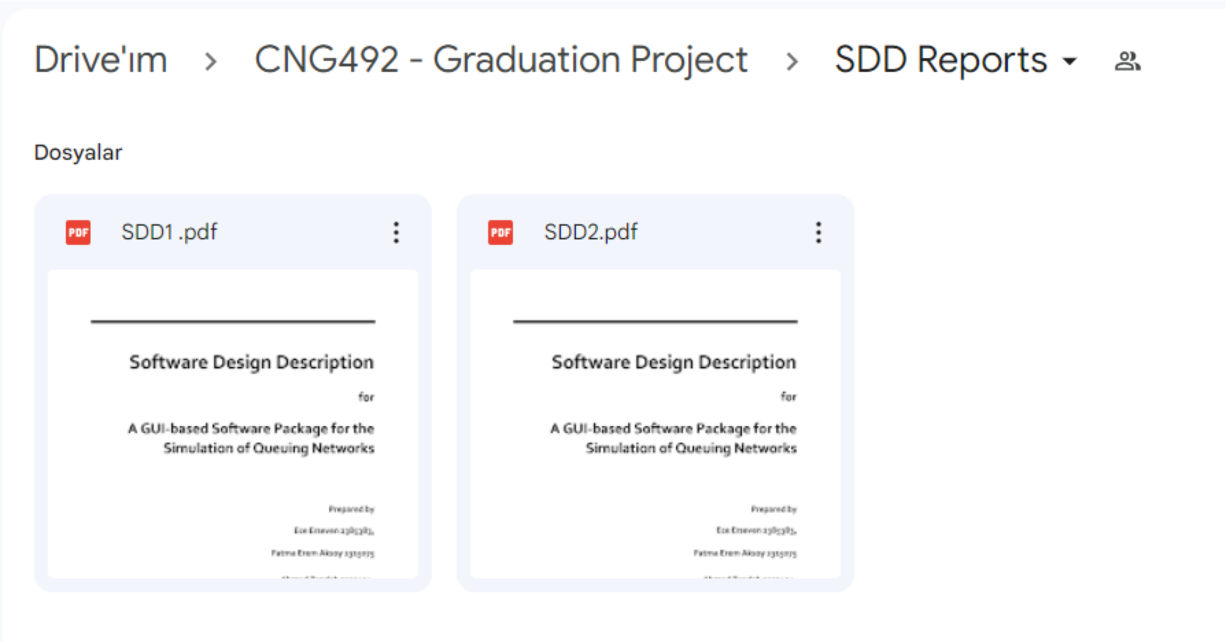
Configuration Manage...

Final Report

SDD Reports

SRS Reports

Figure 3: Google Drive Main Folder CNG492-Graduation Project



Drive'ım > CNG492 - Graduation Project > SDD Reports

Dosyalar

SDD1 .pdf

Software Design Description

for

A GUI-based Software Package for the Simulation of Queuing Networks

Prepared by

Ece Erseven [illegible]

Fatma Erem Aksoy [illegible]

SDD2.pdf

Software Design Description

for

A GUI-based Software Package for the Simulation of Queuing Networks

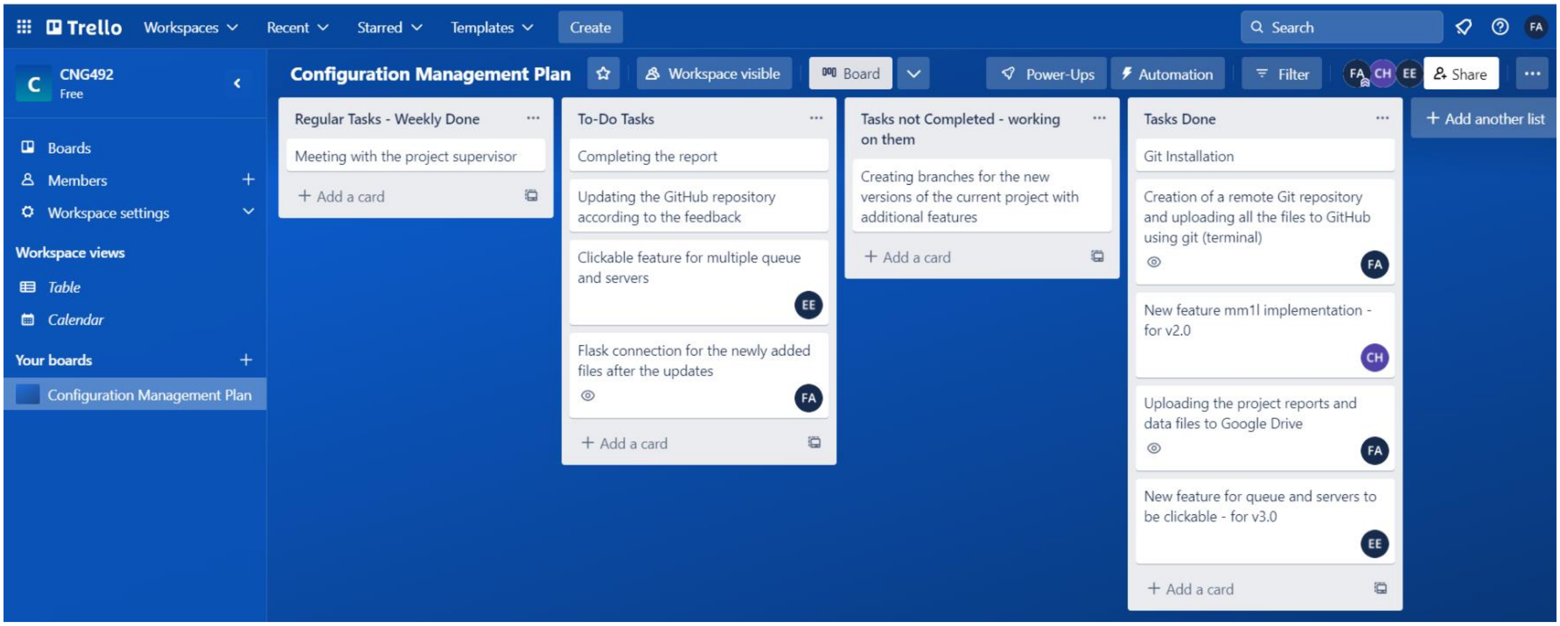
Prepared by

Ece Erseven [illegible]

Fatma Erem Aksoy [illegible]

Figure 4: Google Drive SDD Reports Folder Content

Figure 5 shows the platform Trello that we use to keep track of the tasks done, doing and to do for each documentation or critical feature that will be added to the project. It can be seen which member or members work on which task. We will add new boards as we go further with the tasks during the project, but the current version, Figure 5, is provided below:



Configuration Management Plan

Regular Tasks - Weekly Done
- Meeting with the project supervisor
- + Add a card

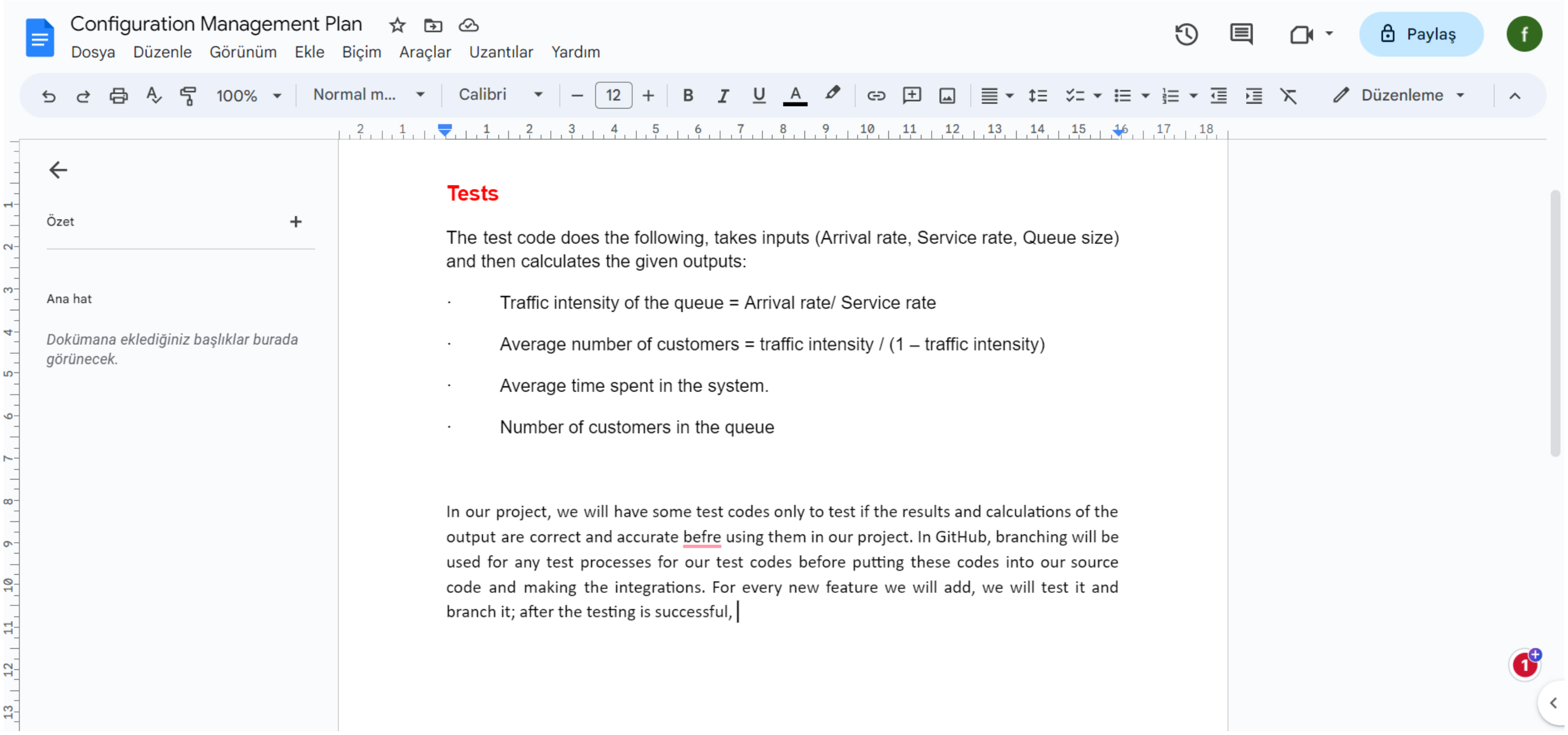
To-Do Tasks
- Completing the report
- Updating the GitHub repository according to the feedback
- Clickable feature for multiple queue and servers (EE)
- Flask connection for the newly added files after the updates (FA)
- + Add a card

Tasks not Completed - working on them
- Creating branches for the new versions of the current project with additional features
- + Add a card

Tasks Done
- Git Installation
- Creation of a remote Git repository and uploading all the files to GitHub using git (terminal) (FA)
- New feature mm1l implementation - for v2.0 (CH)
- Uploading the project reports and data files to Google Drive (FA)
- New feature for queue and servers to be clickable - for v3.0 (EE)
- + Add a card

Figure 5: Current Trello Board

We use Google Docs to write the reports. Every member is responsible for their part(s), and as Google Docs allows us to work on the document simultaneously, we do not face any problems or conflicts. Also, as it is accessed from the web, it allows us to use Grammarly while writing the report and fix our typos/wrong word usage as well. A sample screenshot from Google Docs for one of the sections in this report can be seen in Figure 6.


Configuration Management Plan
Dosya Düzenle Görünüm Ekle Biçim Araçlar Uzantılar Yardım
Paylaş
100%
Normal m...
Calibri
12
Düzenleme
Özet
Ana hat
Dokümana eklediğiniz başlıklar burada görünecek.

Tests

The test code does the following, takes inputs (Arrival rate, Service rate, Queue size) and then calculates the given outputs:

- Traffic intensity of the queue = Arrival rate/ Service rate
- Average number of customers = traffic intensity / (1 – traffic intensity)
- Average time spent in the system.
- Number of customers in the queue

In our project, we will have some test codes only to test if the results and calculations of the output are correct and accurate befre using them in our project. In GitHub, branching will be used for any test processes for our test codes before putting these codes into our source code and making the integrations. For every new feature we will add, we will test it and branch it; after the testing is successful,

Figure 6: Google Docs Platform

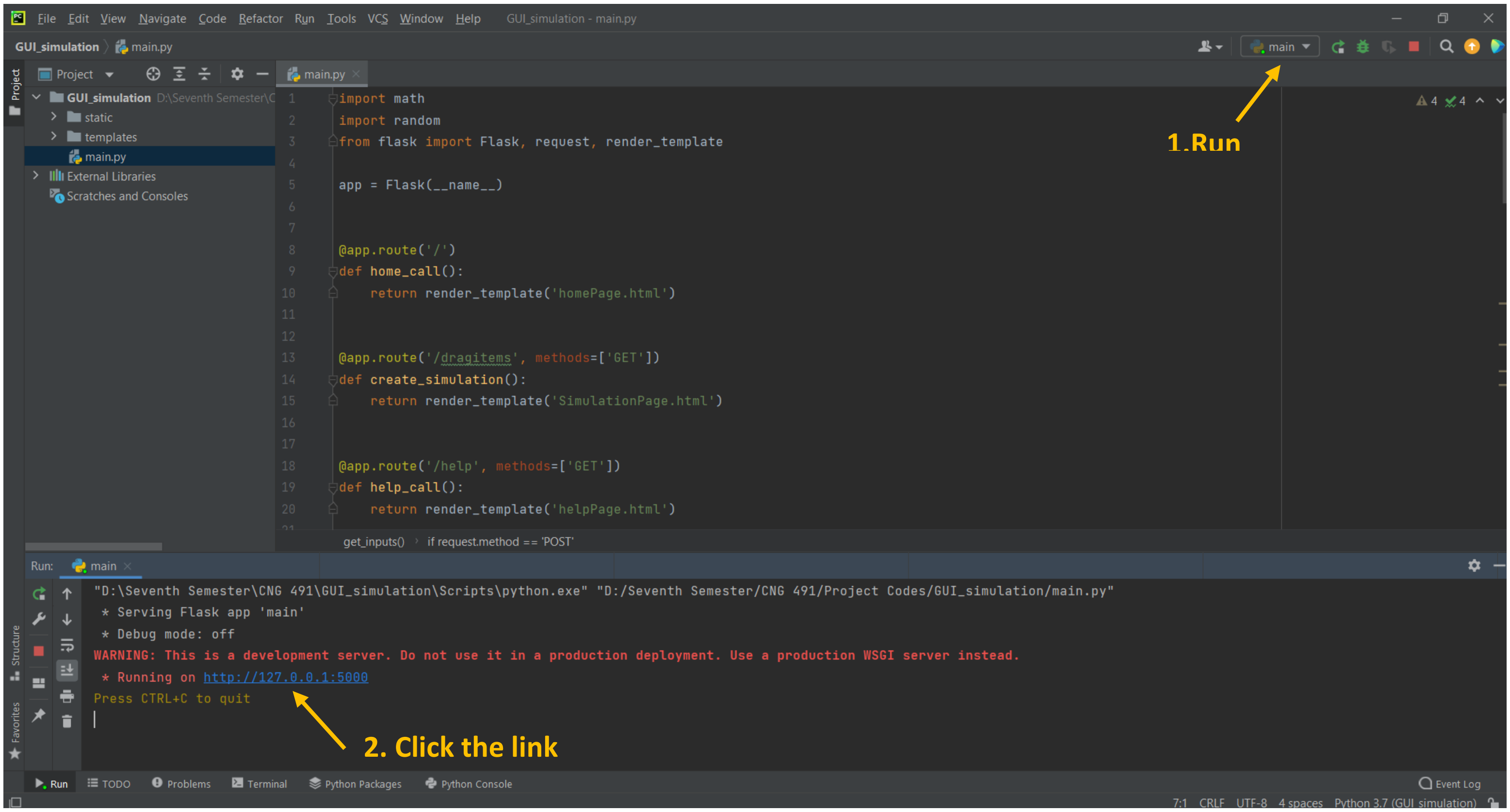
Build Scripts

The tasks we are working on to achieve this semester:

- M/M/1/L: Simple queue with a limited capacity size in code. Unlike M/M/1 that we implemented for version 1, this time we will have a limit for the number of entities for improved performance by reducing the traffic in the queue.
- M/M//C/L: With this functionality, we will be able to deal with multiple servers. This feature will be implemented to connect single queue and multiple servers.
- Queues with multiple servers connection implementation in code
- Drag and drop with multiple server-queue connection and adding new queue types to the simulation

As our project does not require parallel execution of different .py files simultaneously, it does not need any build scripts. However, how the code will be executed can be found below.

Figure 7 explains the steps to execute the code on PyCharm and open the Home page, Figure 8, of the simulation tool to create and run a simulation.



File Edit View Navigate Code Refactor Run Tools VCS Window Help GUI_simulation - main.py
GUI_simulation 〉 main.py
main
Project
GUI_simulation D:\Seventh Semester\C
static
templates
main.py
External Libraries
Scratches and Consoles
main.py
Run: main
Run TODO Problems Terminal Python Packages Python Console
Event Log
7:1 CRLF UTF-8 4 spaces Python 3.7 (GUI simulation)

```python
import math
import random
from flask import Flask, request, render_template

app = Flask(__name__)


@app.route('/')
def home_call():
    return render_template('homePage.html')


@app.route('/dragitems', methods=['GET'])
def create_simulation():
    return render_template('SimulationPage.html')


@app.route('/help', methods=['GET'])
def help_call():
    return render_template('helpPage.html')
```

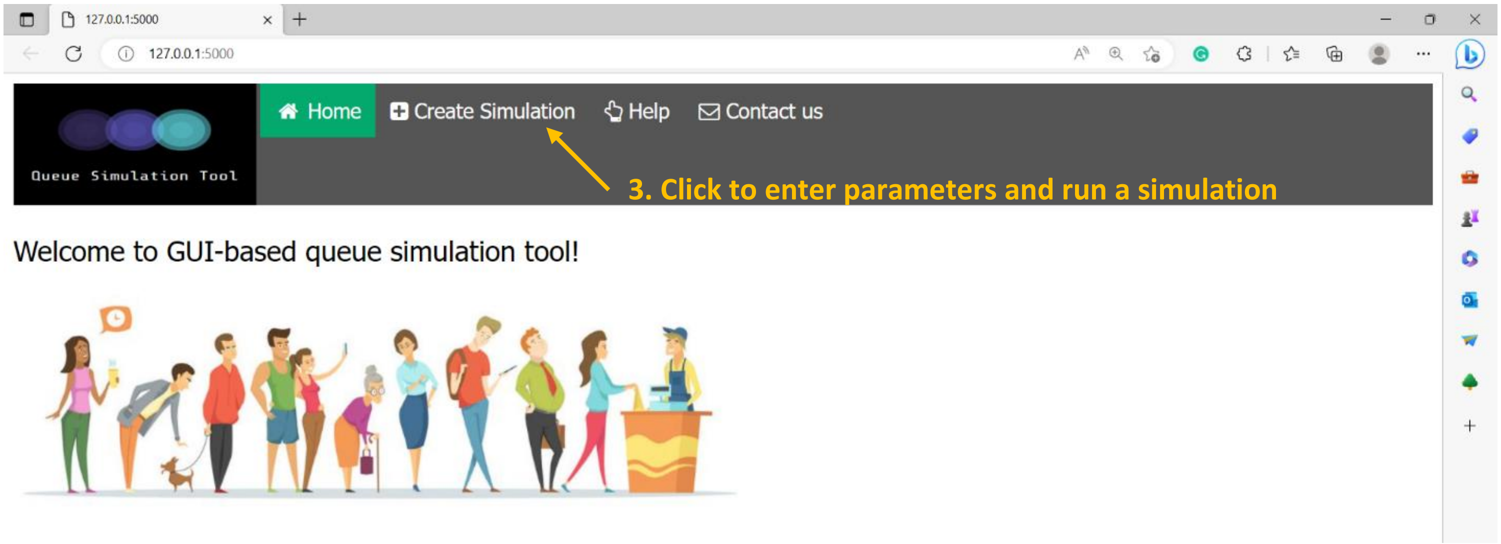
get_inputs() › if request.method == 'POST'

```
"D:\Seventh Semester\CNG 491\GUI_simulation\Scripts\python.exe" "D:/Seventh Semester/CNG 491/Project Codes/GUI_simulation/main.py"
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

1.Run

2. Click the link

Figure 7: How to Run the Code



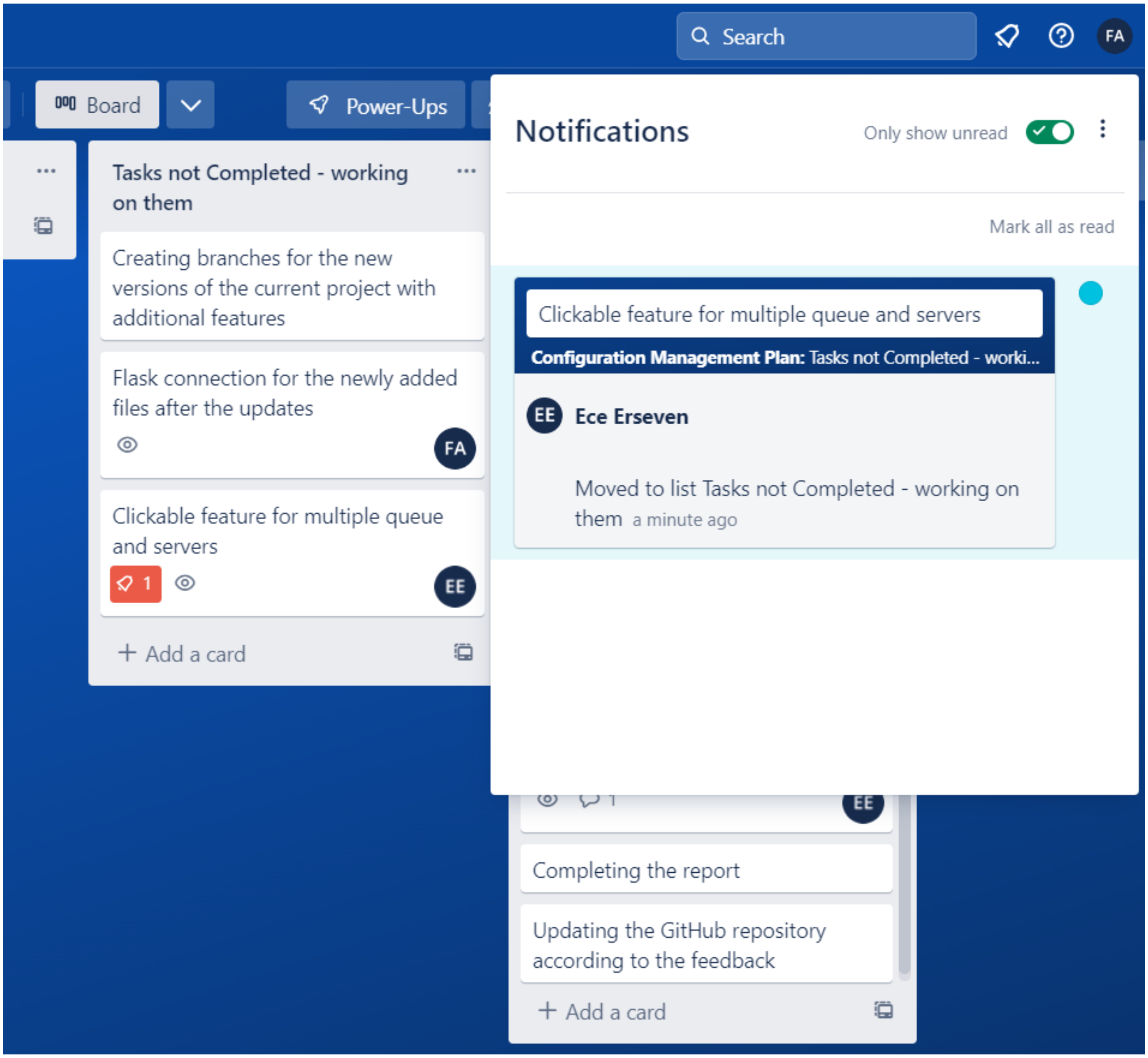
3. Click to enter parameters and run a simulation

Welcome to GUI-based queue simulation tool!

Figure 8: Home Page of the Tool

Reporting/Notification Scripts

To notify team members about the changes made by another team member or to communicate about any issue/task/meeting, we use WhatsApp group and Discord. For the tasks completed or changes made, we use Trello by assigning team members to any task on the list so that notification can be sent to other members in case of any update on the state of the task, see Figure 9 below.



Tasks not Completed - working on them

Creating branches for the new versions of the current project with additional features

Flask connection for the newly added files after the updates

Clickable feature for multiple queue and servers

Clickable feature for multiple queue and servers

Configuration Management Plan: Tasks not Completed - worki...

Ece Erseven

Moved to list Tasks not Completed - working on them a minute ago

Completing the report

Updating the GitHub repository according to the feedback

Figure 9: Trello Notification

Version Tree

We will use GitHub to keep track of the versions of the program and all the members will have an access to all the versions of the code files for the project. There are branches created for each new version and the code files are updated accordingly. The master branch is created to hold the v1 of the project which is the last version that we had last semester. V2 is created to add new features and updates for the 2nd version of the project that is essential for the multiple queue-multiple server structure. Only update was in the code files so an additional file, testmm1l.py, is added to the branch. Another branch called v3 is created to keep track of the changes made for the next new version which has a new clickable feature that will allow us to handle multiple queue-multiple server structure after using the drag and drop feature to connect them to each other. The update in this version is made in one of the existing HTML files and a new HTML file are added as well. As the last step, the integration for Flask is made in the main.py file as a result of merging all the new files.

Figure 10 shows the development of three versions of a system and the changed made in these versions are as specified, and to show these versions in Git repository, the tags will be used:

1. Version 1.4 has been developed to add a critical feature, drag and drop, to the first release of the system. Then the second system release, R1.1, happen to be ready for users to use the drag and drop feature for only single server and single queue case.
2. Version 2.0 and Version 2.1 have been developed to add new calculations and their graphs. As they both need testing multiple times with different intervals and it takes time, two versions are necessary to work on these features.
3. Version 3.0 has been developed to be able to use the drag and drop feature for multiple server and multiple queue cases.

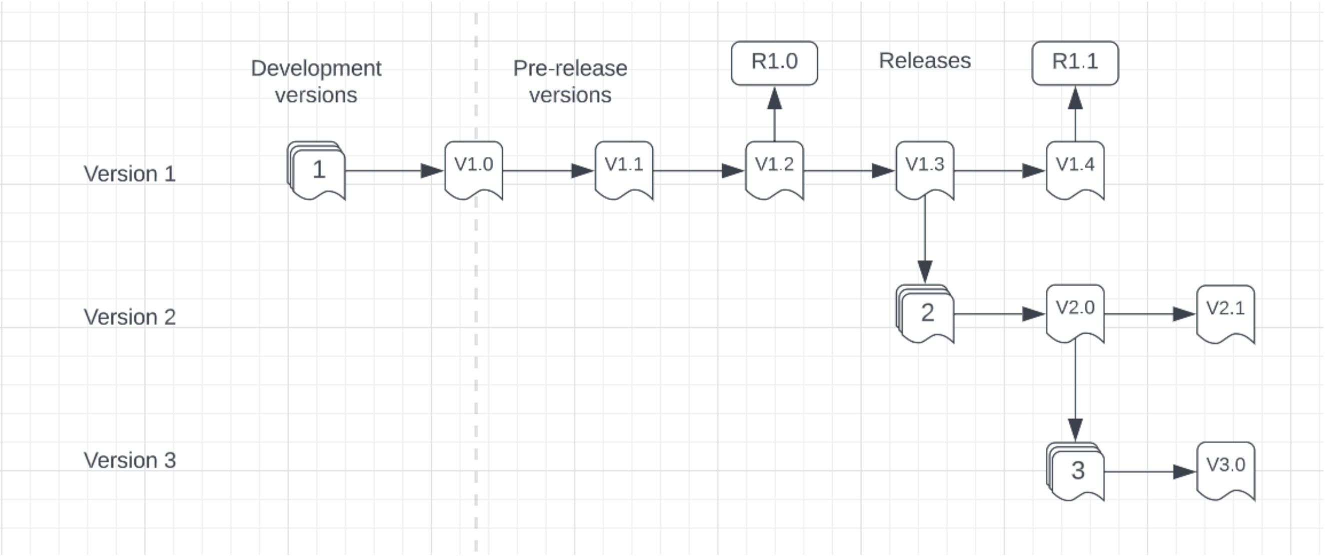
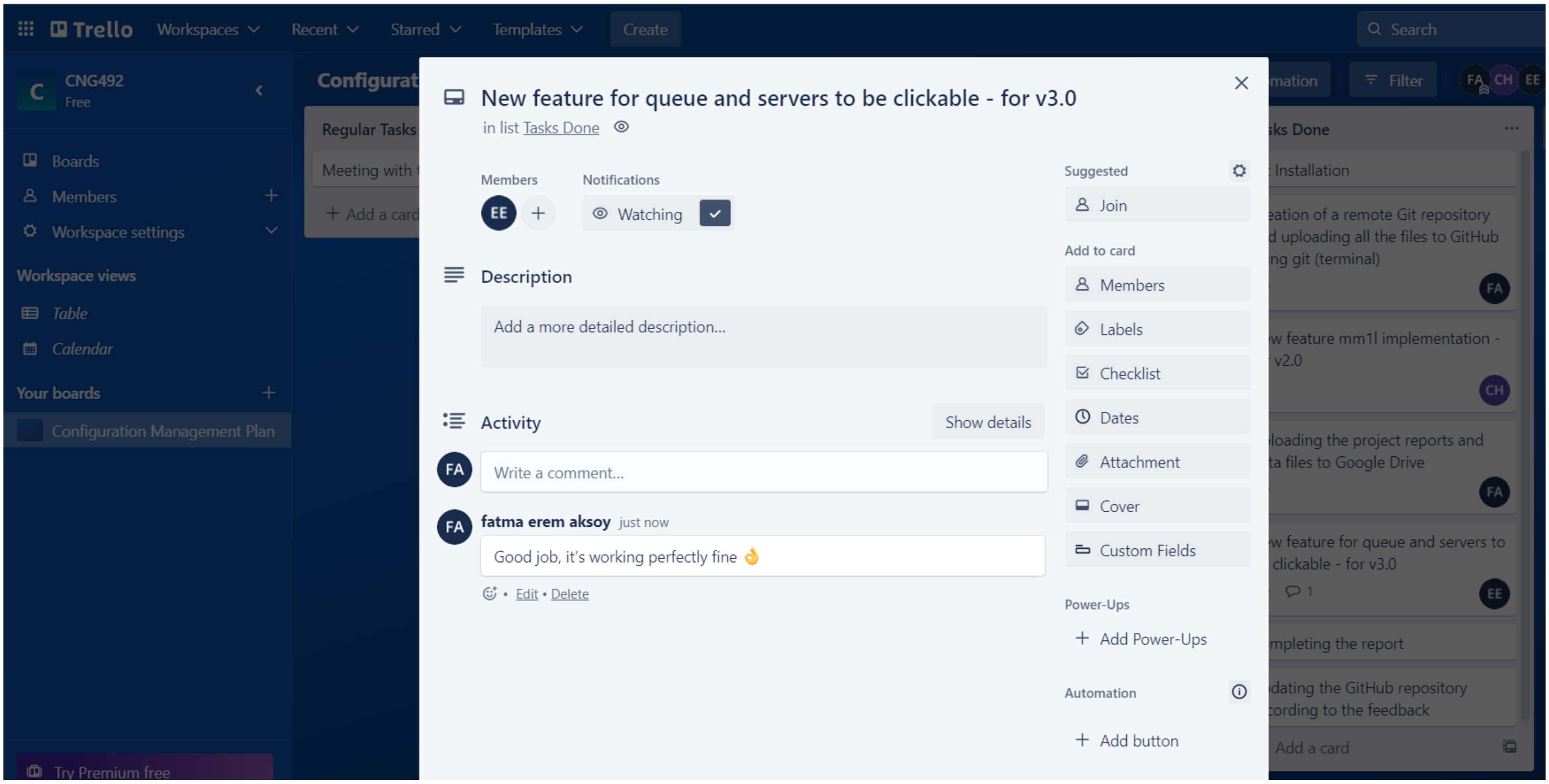


Figure 10: Current Version Tree

Log/History

For the tasks completed or the tasks that are being worked on, we use Trello as it saves the name of each member who makes any change or puts a comment on any task or other operations available. All the logs can be seen from there easily for the development process. Figure 11 shows what the comments look like on Trello.



New feature for queue and servers to be clickable - for v3.0

in list Tasks Done

Add a more detailed description...

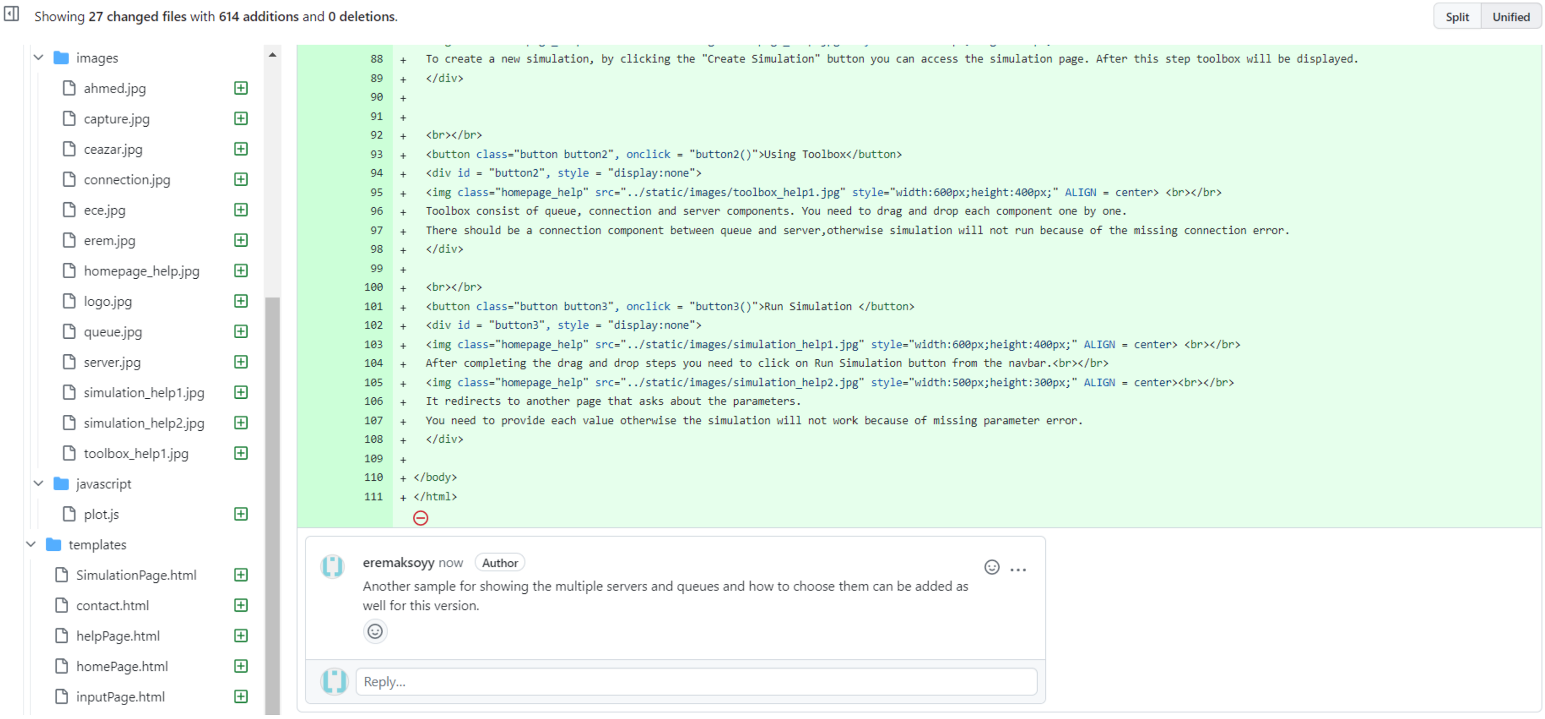
Write a comment...

fatma erem aksoy just now

Good job, it's working perfectly fine 👌

Figure 11: Leaving Comments on Trello

In terms of the system updates, adding new features or any type of version/system release, we use GitHub to keep all the files we use up-to date. All the changes made by any member can be checked from there and all the members can leave a comment to a change made by another member so that we can give feedback to each other for the next releases or to fix any bug/error before an upcoming release. Problems faced can also be solved by this way as all the members have access to every file available in the project. In Figure 12 below, a sample can be seen how our team uses GitHub to leave comments for a particular file.



Showing **27 changed files** with **614 additions** and **0 deletions**.

```
+   To create a new simulation, by clicking the "Create Simulation" button you can access the simulation page. After this step toolbox will be displayed.
+   </div>
+
+
+   <br></br>
+   <button class="button button2", onclick = "button2()">Using Toolbox</button>
+   <div id = "button2", style = "display:none">
+   <img class="homepage_help" src="../static/images/toolbox_help1.jpg" style="width:600px;height:400px;" ALIGN = center> <br></br>
+   Toolbox consist of queue, connection and server components. You need to drag and drop each component one by one.
+   There should be a connection component between queue and server,otherwise simulation will not run because of the missing connection error.
+   </div>
+
+   <br></br>
+   <button class="button button3", onclick = "button3()">Run Simulation </button>
+   <div id = "button3", style = "display:none">
+   <img class="homepage_help" src="../static/images/simulation_help1.jpg" style="width:600px;height:400px;" ALIGN = center> <br></br>
+   After completing the drag and drop steps you need to click on Run Simulation button from the navbar.<br></br>
+   <img class="homepage_help" src="../static/images/simulation_help2.jpg" style="width:500px;height:300px;" ALIGN = center><br></br>
+   It redirects to another page that asks about the parameters.
+   You need to provide each value otherwise the simulation will not work because of missing parameter error.
+   </div>
+
+ </body>
+ </html>
```

eremaksoyy now Author

Another sample for showing the multiple servers and queues and how to choose them can be added as well for this version.

Figure 12: Leaving a Comment on GitHub