Nicolas Grobelny

Optimal Camera Placement

CO3201 Interim Report

Contents

[Aims and Objectives 2](#_Toc88409175)

[Aims 2](#_Toc88409176)

[Objectives 3](#_Toc88409177)

[Planning and Timescales 4](#_Toc88409178)

[Methodology 4](#_Toc88409179)

[Scrum 4](#_Toc88409180)

[Why Scrum 4](#_Toc88409181)

[Timescales 5](#_Toc88409182)

[Semester One 5](#_Toc88409183)

[Semester Two 6](#_Toc88409184)

[Justification 6](#_Toc88409185)

[Specifications and Design 7](#_Toc88409186)

[Client Side 8](#_Toc88409187)

[Connection between server and user 8](#_Toc88409188)

[Server-Side Algorithms 9](#_Toc88409189)

[Prototype and Features 10](#_Toc88409190)

[Application Appearance and Functionality 10](#_Toc88409191)

[Features 12](#_Toc88409192)

[Done 12](#_Toc88409193)

[In Development 13](#_Toc88409194)

[Backlog/TODO 13](#_Toc88409195)

[Optional/Nice to Have 13](#_Toc88409196)

[Survey of Literature Review 15](#_Toc88409197)

[Optimal Camera Placement Problem 15](#_Toc88409198)

[Additional Algorithm Sources 16](#_Toc88409199)

[Additional Resources 17](#_Toc88409200)

[Methodology 17](#_Toc88409201)

[Front End 17](#_Toc88409202)

[Connection 17](#_Toc88409203)

[Backend 17](#_Toc88409204)

[Bibliography 18](#_Toc88409205)

# Aims and Objectives

## Aims

This project aims to determine the best solution/solutions to the Optimal Camera Placement Problem by conducting a technical review of the field. It will attempt to determine which algorithms are valid and whether studies can be reproduced.

Additionally, I would develop a web application using the results of the technical survey. The program should allow them to digitalise their surroundings, determine potential camera placements and get an optimal solution based on maximisation parameters and constraints.

In this way, the research would help security experts to apply the theoretical knowledge. Currently, the most common method to place cameras involves previous knowledge and the “rule of thumb”, making it inefficient.

## Objectives

The main objectives required to achieve the project are:

1. Stage One – Problem Research:
   * Conduct research into the Optimal Camera Placement Problem, maximising the number of potential algorithms that could be used to solve that problem.
2. Stage Two – Implementation Research:
   * Determine the best Frontend technology for this application, focusing on accessibility and simplicity (HTML5/CSS4/JavaScript).
   * Determine the best Backend technology, focusing on speed and single-page applications. (Python-Flask).
   * Determine the best connecting technology, focusing on simplicity and performance. (Web Sockets, more specifically Sockets.io).
3. Stage Three – Environment Implementation:
   * Implement a drawing environment on the client-side using JavaScript. The environment should resemble a simple drawing application such as Paint.
   * Implement a single page Backend allowing the user to fetch the web page and conduct updates using Sockets.
   * Implement the socket connection so that users can exchange information with the server efficiently and securely.
4. Stage Four – Algorithm Implementation:
   1. Pick a random algorithm from the algorithms researched during stage one.
   2. Determine its validity based on claimed time complexity and difficulty to implement. If the algorithm does not meet the criteria, disregard further steps.
   3. Implement and test the algorithm.
   4. Go to step “a” and continue till all algorithms have been considered.
5. Stage Five – Algorithm Comparison:
   1. Get sample data for the Optimal Camera Placement using sources such as GECCO OCP.
   2. Run each algorithm using sample data and collect results.
   3. Determine the best algorithm if possible. Otherwise, identify the situation in which the algorithm has the best performance.

Each stage must be completed in the given order. Inside each phase, unordered lists can be finished in any order, whereas subtasks from ordered sections need to be executed one after the other. That gives a clear structure and prevents implementing features in a way that requires refactoring.

# Planning and Timescales

## Methodology

### Scrum

Scrum is a project management framework based around Agile philosophy. It is based around a sprint – A short, time-boxed period during which a Scrum team attempts to complete a set amount of work. Sprints offer greater flexibility as project requirements can change without significantly impacting the deadline.

Another advantage of Scrum is its deliverability – At the end of each sprint, the stakeholder would see finished tasks and determine whether they are of acceptable quality.

### Why Scrum

The Optimal Camera Placement Application appears to be the first project of its type. It is near impossible to visualise all features that should be part of it. Currently, a list of required features tends to evolve naturally, be it from the client’s feedback or from the inaccessibility of the application itself. For example, the ability to display coordinates of the grid space using both grid space and real-world dimensions was not a planned feature, but testing of the application revealed that it is near impossible to draw accurate buildings without it.

As such, I require a framework that can quickly react to changes. Scrum performs well in this situation as its short sprint window (Two Weeks in this case) allows me to prototype requested features, show them to the user and change them if necessary.

## Timescales

### Semester One

#### Done

Stage One – Problem Research **(11th October – 20th October):**

* Conduct research into the Optimal Camera Placement Problem, maximising the number of potential algorithms that can solve that problem.

Stage Two – Implementation Research **(11th October – 20th October):**

* Determine the best Frontend technology for this application, focusing on accessibility and simplicity (HTML5/CSS4/JavaScript).
* Determine the best Backend technology, focusing on speed and single-page applications. (Python-Flask)
* Determine the best connecting technology, focusing on simplicity and performance. (Web Sockets, more specifically Sockets.io)

#### Started

Stage Three – Environment Implementation **(20th October – 1st December):**

* Implement a drawing environment on the client-side using JavaScript. The environment should resemble a simple drawing application such as Paint.
* Implement a single page Backend allowing the user to fetch the web page and conduct updates using Sockets. (Done)
* Implement the socket connection so that users can exchange information with the server efficiently and securely. (Done)

Stage Four – Algorithm Implementation **(3rd November – 21st February)**:

Each algorithm should take approximately a week to implement, with enough time left for adjustments or delays.

1. Pick a random algorithm from the algorithms researched during stage one.
2. Determine its validity based on claimed time complexity and difficulty to implement. If the algorithm does not meet the criteria, disregard further steps.
3. Implement and test the algorithm.
4. Go to step “a” and continue till all algorithms have been considered. **(Done 1 out of 10)**

### Semester Two

#### Not Started

Stage Five – Algorithm Comparison **(21st February – 7th April)**:

1. Get sample data for the Optimal Camera Placement using sources such as GECCO OCP.
2. Run each algorithm using sample data and collect results.
3. Determine the best algorithm if possible. Otherwise, identify the situation in which the algorithm has the best performance.

Stage Six – Testing, Improvements and User Suggestions **(7th April – 5th May)**:

* Ensure that test coverage is at least at 75 %, excluding network functions.
* Implement additional stakeholder suggestions (Such as optional features or quality of life improvements).
* Add accessibility features and ensure compliance with web standards.

### Justification

Whilst adding exact dates might allow me to better track whether the project is progressing at the correct rate, the nature of the Scrum sprint prevents me from specifying delivery dates within each epic.

Doing so would compromise quality as features will be rushed to meet the expected date. Another option is to ignore exact dates until the task is done, removing the reason for those times to be there in the first place.

Each epic has been chosen in a way to overestimate the amount of time needed to implement it. The project is progressing faster than expected, allowing to assign a significantly longer duration to Algorithm Implementation and Comparison. In case time proves to be insufficient to deliver the project, Stage Six is optional and can be removed permanently to ensure the quality of the deliverable.

# Specifications and Design

The Optimal Camera Placement Application (OCPA) is implemented as a web application.

Rather than using an MVC model, this application would be based around SPA (Single Page Application) model. Doing so decreases the load on the server as the user only sends and receives updates to currently presented content rather than a whole website. The client-side is responsible for most operations, using JavaScript to create a drawing environment that reacts to users’ inputs.

The SPA uses web sockets to communicate, allowing for constant, uninterrupted, and efficient exchange of information between the user and the server. Once the user finishes designing the environment to suit the real-life situation, they can then send the required data to the server to receive optimal camera positions.

The actual server-side would be based around Python web framework Flask. As the application requires no security, except potentially a simple login system, the data handlers do not require any complicated features. As such, Flask’s simplicity allows me to quickly implement working data handling and move on to more important features such as algorithms.

The algorithm section is the core of this project. It would consist of approximately ten algorithms, written (where possible) through the usage of C-based Python libraries such as NumPy. This approach ensures high performance whilst retaining the simplicity of Python code. As such, the project can be completed much quicker than the equivalent C application.

The actual algorithms will be based on the research conducted on the OCPP. The goal is to implement the most popular/viable algorithms used to solve this problem and compare them in different situations. Doing so should allow me to review these algorithms, determining which algorithms have the best performance and which should be avoided.

## Client Side

The client-side will resemble drawing software, not unlike Paint. To implement that, I will be using a combination of HTML and CSS for the interface following the Material Design style to create an accessible website. JavaScript canvas, in combination with mouse events, will be used to allow the user to draw.

As the canvas can cause performance issues, especially when frequently redrawing objects, all input events would be registered and applied first then the canvas would update. This process should prevent any input lag and simplify the implementation (As the project would guarantee that the updated point would eventually be drawn on the board).

JavaScript classes will be used to create an object-oriented environment that can be updated without connection to the server. This approach of limited communication has two advantages. The first one is a smaller load on the server, as it does not need to handle constant requests from possibly hundreds of users at the time. The second advantage is the ability to operate in areas with a bad connection. The data is stored locally. Therefore, no changes to it are made by the server until the server receives the data. If the user is disconnected from the server, they can reconnect later without data loss.

## Connection between server and user

The connection between server and user consists of two parts. Despite the application not using the MVC model, it still needs a Controller responsible for serving the main page. Fortunately, because the project does not use models, the controller only needs to deliver a view to the client side.

Whilst Flask only allows to handle one request at a given point, that is the only operation that is a single, synchronous process. It is unlikely to become a bottleneck for the performance as more concurrent workers can be added.

The second part of the system is a Web Socket JavaScript library called Socket.io. The Web Socket has won over MVC and AJAX due to reasons below:

MVC – Model-View-Controller requires several conversions and data transfers, especially when transferring model data to and from the client. Whilst implementing MVC would drastically simplify client-side code, the performance drawback is too great for this project.

AJAX – Asynchronous JavaScript and XML works similarly to the Web Socket technology as the user can send and receive data after the web page has loaded. The main difference is that whilst AJAX is more secure. Web Socket, however, has a significantly better performance as it does not need to establish a connection with the server every time it sends data. As such, it is a better choice in a situation where security is not a concern.

The Web Socket has been selected due to its advantages over other candidates. Whilst it is possible that there are other technologies out there that do outperform Web Sockets, the time needed to find them would outweigh potential benefits.

## Server-Side Algorithms

When the user requests new cameras to be placed, the client-side socket sends a request to the server containing the request and the Environment in the JSON form.

Upon arriving, the Python side would generate an Environment using NumPy data types (For efficiency) and create a Solver class.

The Solver class aims to contain all of the camera solving algorithms within it. That allows the development of helper functions, such as algorithm generating all coordinates between two grid points for visibility checker (Bresenham's line algorithm), reducing code redundancy.

This approach should be viable for as long as the algorithms stay relatively simple. If the actual class starts to become buggy and cumbersome, each algorithm can easily be converted into separate classes (Each inheriting from the parent with all of the helper functions) and adapt execution using Polymorphism. Whether this decision is necessary will be determined once several algorithms are implemented, but the project is developed with that in mind.

These Backend algorithms will be based around NumPy (C library in Python). Whilst other libraries such as TensorFlow or SciPy are likely to be used in Computational Intelligence algorithms, the NumPy will be the core of the project, with all data types and calculations being done in that library (Where possible).

Once the algorithm executes correctly, the program will return camera results to the user using a Web Socket by targeting a unique User Id. The User Id is generated automatically once the user connects to the socket.

# Prototype and Features

This section focuses on the current progress of the project. It would also include a feature/user story list.

## Application Appearance and Functionality

At the time of writing this report, the application looks as follows:

Chart, bar chart

Description automatically generated

Whilst the interface can be considered as ugly, it is functional allowing user to adjust size and draw basic schematics:

A screenshot of a computer

Description automatically generated with medium confidence

As well as sample the space automatically using the sampling slider on the left side:

Icon

Description automatically generated with medium confidence

Or even effectively place cameras:

A screenshot of a computer

Description automatically generated with medium confidence

Whilst not perfect, the program is achieving its purpose. The user can draw a basic schematic, sample it, and get semi-optimal camera placements. As the base working version has been developed, the project can now implement algorithms, adding additional quality-of-life features and interface improvements.

## Features

This section focuses on all features of this application. It is by no means a completed list as features would get added and subtracted based on development, testing and client’s feedback. Therefore, this list should be used as guidelines where each of the features will be considered but might not end up in the final version of the project.

In the context of user stories, the **User** means a security company employee responsible for determining camera positions.

This project also serves as a technological survey of the Optimal Camera Placement Problem. As such, it also includes user stories aimed at Scientists from that field.

### Done

#### Front-End

- As a User, I want to be able to draw a building, so that I can receive optimal camera positions.

- As a User, I want to be able to place obstacles, so that I can receive a realistic coverage of cameras.

- As a User, I want to be able to automatically sample my building, so that I can spend more time on drawing a more accurate layout.

- As a User, I want to be able to manually sample the drawn space, so that I can better protect crucial areas of the building.

- As a User, I want to be able to resize the environment, so that I can make plans for different situations, such outdoors areas.

#### Connection

- As a User, I want to be able to operate as much of the program offline, so that I can operate in areas without internet.

- As a User, I want to be able to easily reconnect to the program without losing data, so that I can operate even if the connection is temporarily disconnected.

- As a User, I want the program to transfer as little data as possible, so that I can operate in areas with poor/limited internet.

#### Back-End

- As a User, I want to be able to get optimal camera positions, so that I can secure the examined area.

- As a User, I want to be able to get camera positions fast, so that I can easily experiment with different parameters to suit my needs.

### In Development

#### Front-End

- As a User, I want to have a clear, accessible interface, so that I do not require extra training to know how to use the program.

- As a User, I want to be able to tell where each camera is looking and what samples it sees, so that I can validate results of the program.

#### Connection

- As a User, I want to know the status of the connection, so that I can know whether program is operational.

#### Back-End

- As a User, I want to be able to choose from variety of algorithms, so that I can get the most optimal result for my situation.

### Backlog/TODO

#### Front-End

- As a User, I want to have additional drawing tools such as Fill or bigger brush, so that I can draw the building plan more effectively.

- As a User, I want to be able to insert real-life dimensions, so I can know each camera position without needing to do conversions.

- As a User, I want to have an option to change camera parameters, such as cost, distance, or number of them, so that I can maintain constraints imposed onto me by the client.

- As a Scientist, I want to be able to insert environment data from the file, so that I can test algorithms without needing to redraw the environment.

#### Back-End

- As a User, I want to be able to have a way of determining the best algorithm for me, so that I can choose one without needing to know how they work.

- As a User, I only want to have a choice of optimal algorithms, so that I don’t waste time and resources using inefficient solutions.

- As a Scientist, I want to be able to see information about each iteration (Such as average score), so that I can compare algorithms easily.

### Optional/Nice to Have

#### Front-End

- As a User, I want to be able to save data to a file and share it with other people using the same program, so that I can easily exchange plans between people without losing data.

- As a User, I want to be able to use the program on a tablet or similar mobile device, so that I can use this program in any situation.

- As a User, I want an option to have multiple cameras in the same position, so that I can receive optimal solution in specific situations without losing performance.

#### Back-End

- As a User, I want to have access to more algorithms (More than ten), so that I can achieve even better efficiency when securing an area.

# Survey of Literature Review

The Literature Review aims to help me decide and justify sources to be relied on to make justified decisions about this project. The core of this review would focus on finding the origins of the Optimal Camera Placement Problem and stating what the problem is.

Once a clear definition of the OCPP is established, I will find several algorithms that can be implemented as part of this project. The quality of individual algorithms is not a concern, as the objective of this project is to conduct a literature review of existing technologies.

Lastly, a section of this review would be dedicated to justifying technologies used in this project, such as NumPy or Socket.IO.

## Optimal Camera Placement Problem

Optimal Camera Placement Problem is a more constrained Art Gallery Problem [1]. The Art Gallery Problem has originated in 1987 in [2] and is a computational geometry problem. Given a polygon with edges and vertices, determine the minimal number of guards that can cover the entire polygon. Assume unlimited depth-of-field and field-of-view [2, p. 1].

The problem is similar enough to its predecessor that one of the variations of this problem [2, pp. 258-265] has similar constraints to the OCPP (Limited FOV or DOF, but not both). As such, despite the [2] being considerably dated, it can still be used to learn about potential solutions to simpler problems, giving insight into the OCPP. Additionally, it serves as a reference guide to several studies such as [1] [3] [4] [5] that use it as a foundation block, borrowing terms and algorithms without always explaining them in a clear way.

Focusing on more recent studies, [1] will be one of the main sources for this project. The project explores potential ways of implementing the environment for the OCPP, accounting for elements such as blind spots when working in a 2D space. Additionally, it categorises algorithms by similarity, allowing me to determine whether I have implemented any algorithms working in effectively the same way.

However, there are drawbacks. The source does not share data used for testing purposes, merely sharing a document describing the data. Additionally, certain parameters and constraints are ignored (Such as different camera types or the need for different resolutions). Fortunately, that only affects the testing phase which is not as important as this project aims to do the same (In a reproducible way). Due to that, this source can still be used freely.

## Additional Algorithm Sources

To prevent potential bias from the [1], several other sources that describe potential algorithms will be used. Algorithms with the highest number of positive mentions would be considered first.

1. The source [3] is similarly a literature review but of a much smaller scope. It focuses on five types of algorithms, one of them being a brute-force algorithm. The main strengths are pseudo-code that helps with the implementation as it ensures similar implementation to the original algorithm and inclusion of the perfect solution. Through the usage of exhaustive search, the paper can compare examined algorithms to the best-case result, making the relative comparison more impactful. The main flaw is a small scale, examining only five algorithms.
2. The source [4] attempts to solve the optimisation problem, using its algorithm based around Vertex Colouring that does not appear in any other source. Whilst the algorithm itself has been poorly implemented, the paper lists a few different algorithms, such as Greedy Search or Rectangular algorithm, that could be implemented as part of this project. Additionally, the project makes a comprehensive list of real-world parameters and constraints, including items such as Vandalism where cameras seeing each other is preferable.
3. The source [5] is an attempt at solving the optimisation problem, in this case focusing on a singular Computation Intelligence algorithm. Whilst the algorithm itself has a very high time complexity (as stated by the author), the helpful part of this paper is in its parameter definition. It covers aspects such as spatial resolution (Ratio between pixel count as the size of the object. High resolution is required for Face Recognition) and different types of cameras. As such, the paper can be used when determining requirements and parameters but should not be used for anything else due to its higher-than-average time complexity.
4. The source [6] is an attempt at solving the OCPP, featuring several algorithms implemented by the author. The paper focuses a majority of it on modelling the environment, going through many potential solutions to the modelling such as sampling method or how to calculate visibility matrix. As such, the main strength of this source is its attention to detail regarding the environment. The main disadvantage is that paper spends a significant amount of time describing designed application, but the source code is not available, preventing reproducibility of results.
5. Lastly, the source [7] focuses on finding the optimal cameras positions when cameras are attached to a singular point. Whilst not as efficient as a multi-point environment, the raised concept is interesting. The possibility of placing multiple cameras in the same spot, whilst not necessarily optimal in every case, might allow for more optimal solutions in L-shaped areas. In most other cases, the added complexity would be a drawback, so adding this feature as an option is the best solution. The rest of the paper discusses technologies used to create a continuous image using multiple cameras, making it irrelevant.

## Additional Resources

This section contains additional sources that would be used to guide my decision process throughout the project. It will mainly include style guides, justification for technologies and wikis.

### Methodology

The methodology chosen for this project needs to be flexible and allow client to receive frequent prototypes. The Waterfall [8] was not selected due to its inflexibility.

From other options, Kanban [9] is effectively Scrum with constraints to prevent too many merges. As the project uses a single developer, Kanban is not needed. The XP [10] (Extreme Programming) methodology would have been a perfect fit for this project (due to alleviating risks of new technologies and constraint requirement changes). Unfortunately, the XP prioritises group work which is impossible in a one-person project.

In the end, Scrum [11] methodology has been chosen as it meets requirements for both flexibility and rapid prototyping, whilst still having small enough scope for a single-person project.

### Front End

#### Interface

One of the most important aspects of a user-friendly program is its interface. For this project, I would be following the newest version of Material Design [12]. This interface style is clear and concise, as well as familiar to a majority of users due to being used in all Google products. As such, I would follow it to design interface for this project.

#### Technologies

For the actual Front End, I would be using HTML [13], CSS [14], and JavaScript [15] without any additional Front End libraries. Whilst forcing me to write more code overall, this approach grants me a control over the code, allowing to optimise it specifically for this project. Other libraries such as SASS [16] and React.js [17] were considered but ultimately discarded.

The SASS and other CSS pre-compilers are useful in situations where multiple CSS files are present, or the individual files are particularly long. Unfortunately, the interface of this project aims to be simple. As such, the benefit of potentially writing CSS code faster using SASS is outweighed by more complicated workflow and file size of SASS files.

React.js could have been a useful library for this project if the optimisation was not one of the goals of it. Whilst React framework simplifies the exchange of information between client and the server, the overhead caused by functions managing that exchange causes it to be slow. As such, for this project it is better to write pure JavaScript code that can be optimised rather than depending on React.

### Connection

The Socket library [18] will be used as the main way of communicating between client and the server. The other possibilities such as MVC [19] and AJAX [15] were discarded due to reasons listed in the Specifications and Designs section.

### Backend

The most important thing about backend is the usage of Python [20] over other more optimised languages (Such as C# [21] or C++ [22]). The main reason for it is a library called NumPy [23]. It provides a similar performance to C and C++ (Due to it being written in C++) whilst still granting Python accessibility and ease of write. As such, it would be use as a base for algorithms.

# Bibliography

|  |  |
| --- | --- |
| [1] | M. B. J. L. L. I. Julien Kritter, “On the optimal placement of cameras for surveillance and the underlying set cover problem,” *Applied Soft Computing Journal,* vol. 74, pp. 133-153, 2019. |
| [2] | J. O'Rourke, Art Gallery Theorems and Algorithms, Oxford: Oxford University Press, 1987. |
| [3] | D. H. R. Y. a. S. C. J. Zhao, “Approximate techniques in solving optimal camera placement problems,” Barcelona, 2011. |
| [4] | M. P. a. J. Stahl, “The Camera Placement Problem - An art gallery problem variation,” 2008. |
| [5] | U. M. E. a. S. Sclaroff, “Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints,” in *In Proc. of OMNIVIS Workshop*, 2004, pp. 1-12. |
| [6] | H. E. a. R. Lienhart, “On the Optimal Placement of Multiple Visual Sensors,” 2006. |
| [7] | A. Watras, J.-J. Kim, H. Liu, Y. Hu and H. Jiang, “Optimal Camera Pose and Placement Configuration for Maximum Field-of-View Video Stitching,” *Sensors,* vol. 18, no. 2284, 2018. |
| [8] | K. P. W. Baca, “The Waterfall Model in Large-Scale Development,” in *International Conference on Product-Focused Software Process Improvement*, 2009. |
| [9] | Kanbanize, “Kanbanize,” Kanbanize, 2021. [Online]. Available: https://kanbanize.com/. [Accessed 21 11 2021]. |
| [10] | R. E. Jeffries, “What is Extreme Programming,” RonJeffries, 16 03 2011. [Online]. Available: https://ronjeffries.com/xprog/what-is-extreme-programming/. [Accessed 21 11 2021]. |
| [11] | K. S. a. J. Sutherland, “Scrum Guides,” 2017. [Online]. Available: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf. [Accessed 09 10 2021]. |
| [12] | Google, “Material Design 3,” Google, 15 11 2021. [Online]. Available: https://m3.material.io/. [Accessed 21 11 2021]. |
| [13] | W3 Schools, “W3 Schools HTML,” W3 Schools, 2021. [Online]. Available: https://www.w3schools.com/html/. [Accessed 21 11 2021]. |
| [14] | W3 Schools, “W3 Schools CSS,” W3 Schools, 2021. [Online]. Available: https://www.w3schools.com/css/. [Accessed 21 11 2021]. |
| [15] | Mozilla, “JavaScript Guide,” Mozilla, 24 08 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide. [Accessed 21 11 2021]. |
| [16] | The Sass Team, “SASS,” The Sass Team, 2021. [Online]. Available: https://sass-lang.com/documentation. [Accessed 21 11 2021]. |
| [17] | Facebook Open Source, “React JS,” Facebook Inc., 2021. [Online]. Available: https://reactjs.org/. [Accessed 21 11 2021]. |
| [18] | Socket.IO, “Socket.IO,” Socket.IO, 2021. [Online]. Available: https://socket.io/. [Accessed 21 11 2021]. |
| [19] | I. D. Miller, “Flask Diamond MVC,” 02 11 2017. [Online]. Available: https://flask-diamond.readthedocs.io/en/latest/model-view-controller/. [Accessed 21 11 2021]. |
| [20] | Python Software Foundation, “Python,” Python Software Foundation, 2021. [Online]. Available: https://www.python.org/doc/. [Accessed 21 11 2021]. |
| [21] | Microsoft, “A Tour of C#,” Microsoft, 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Accessed 21 11 2021]. |
| [22] | H. S. Bjarne Stroustrup, “C++ Core Guidelines,” Standard C++ Foundation, 19 08 2021. [Online]. Available: http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines. [Accessed 21 11 2021]. |
| [23] | NumPy, “NumPy,” NumPy, 2021. [Online]. Available: https://numpy.org/. [Accessed 21 11 2021]. |