GSoC'24 Project Proposal

FreeCAD : Improve the FreeCAD API Documentation

Anurag Singh

1. Basic Details

Name:

Anurag Singh

Contacts:

Email: prinicipalquantum30@gmail.com

GitHub Profile: Ovalelephant35

Matrix Username: Anurag Singh (Ovalelephant)

LinkedIn Profile: Anurag Singh

LeetCode : Ovalelephant35

Your First Language:

I have proficiency in English and elementary proficiency in French, although Hindi is my first language.

Location and Time zone:

Location: Jaipur, Rajasthan, India

Time zone: Indian Standard Time (UTC+5:30)

Communication:

- UTC 02:00 – UTC 07:00
- UTC 08:00 – UTC 15:00
- UTC 16:00 – UTC 19:30

I am quite Flexible with any time if it helps in better communication with developers and mentors, and I will be reachable anytime through my Mobile and Email.

Education Details:

Currently, I am in my third year of academic pursuit, enrolled in a double major program encompassing Computer Science and Physics at BITS Pilani.

My introduction to programming occurred during my second year of study. Since then, I have delved into diverse domains including Data Structures, Computer Architecture, System Design, Digital Electronics, and Object-Oriented Programming. Engaging in practical projects and coursework has afforded me a robust comprehension of these concepts, concurrently enhancing my aptitude for problem-solving.

2. Share Links, of your previous work on opensource projects.

For the past 7-8 months, I have actively engaged in contributing to open-source projects. Along this journey, I have acquired valuable knowledge spanning embedded systems, system design, object-oriented programming, machine learning, Arduino.

Additionally, I have tackled numerous algorithmic challenges on platforms such as Codeforces and LeetCode, nurturing a keen aptitude for programming and problem-solving in a broader context.

Projects/Repository/Pull Requests Links	**Description**
Verilog-Digital-Design	Designing Systems using Verilog.
Computational-Physics	Utilizing Python for Computational Physics.
Disco-Graph-Optimization-Project	Course Assignment – Bipartite Tough
JavaScript-Projects	Projects implemented in JavaScript.
ESG	ESG – Capability Build
Sugar Labs	Sugar Labs – Enhancing User experience
Chatbots	Chatbot with Self-Learning Capabilities.
Competitive-Programming-Solution	Challenges in Algorithmic Problem Solving.

I have made over **1100 contributions** on GitHub, including more than **40 pull requests** and involvement in over **30 issues**, both active and resolved. My contributions have extended across various organizations and projects.

For further details, please visit my profile at Ovalelephant35 on GitHub.

3. Project Details

During the last months, I've been an active member of the FreeCAD community, immersing myself in its ethos and methodologies. Throughout this period, I've gained substantial insights into the workings of FreeCAD, have contributed and raised some issue.

Here are the pull requests/Issue I have worked on for FreeCAD:

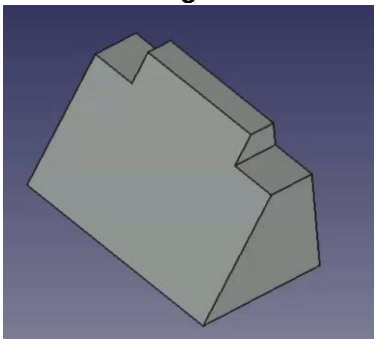
Pull Request Link/Issue	Description	Status
#12838	Added API documentation for Plate_surface, Rectangulartrimmedsurface, Shapefix,_edge in **PART** Workbench.	**Merged**
#13062	Added API documentation for ComplexGEOdata, Typepy, AreaPy, BuildSurfacePy in **CORE**, **PATH** and **PART** Workbench	**Merged**
#12547	Improve the FreeCAD API documentation	**Open**

How I got Interested in this Project:-

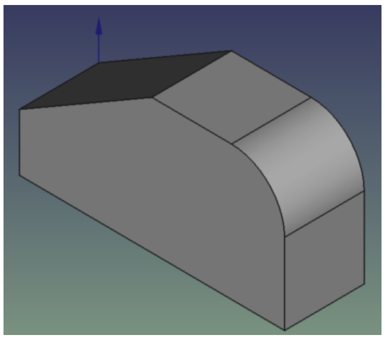
As I delved into my Engineering Drawing Course, I sought out open-source organizations offering Computer-Aided Design modelling software. Among them, FreeCAD stood out as an invaluable tool. Its versatility allowed me to craft a multitude of designs directly applicable to my studies. Even beyond the course, I've relied on FreeCAD for a wide range of 2D and 3D projects. Now, I'm keen to give back to the community that has provided me with such a powerful resource.

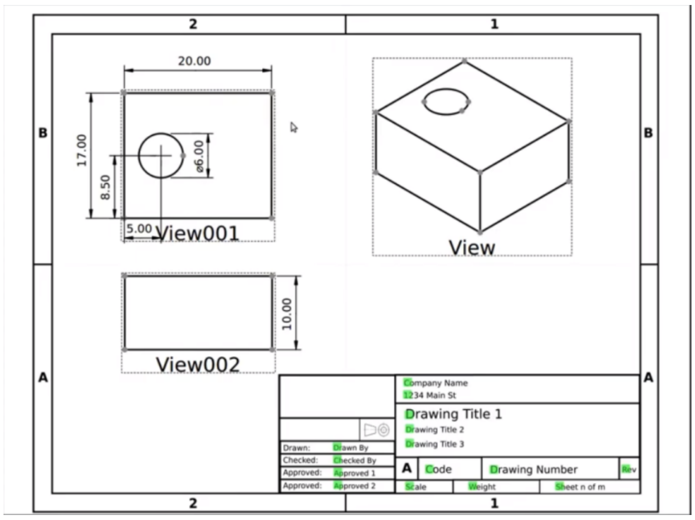
Here are some of the designs I created:

Basic Part Design

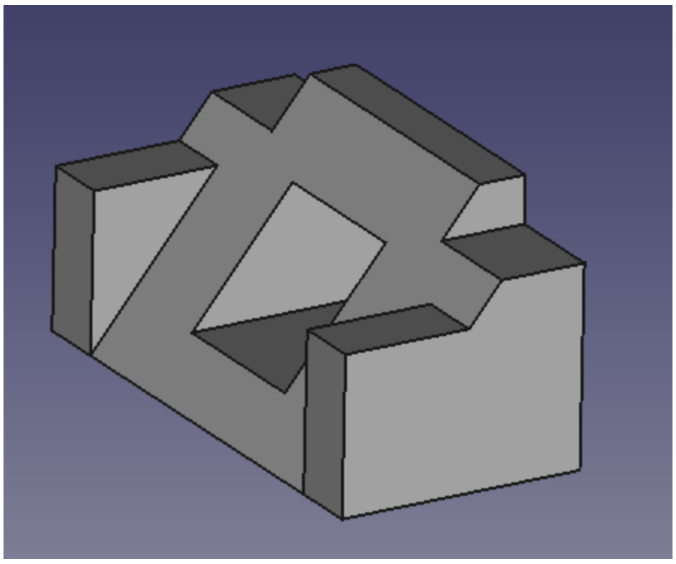

.

Basic Part Design - 2

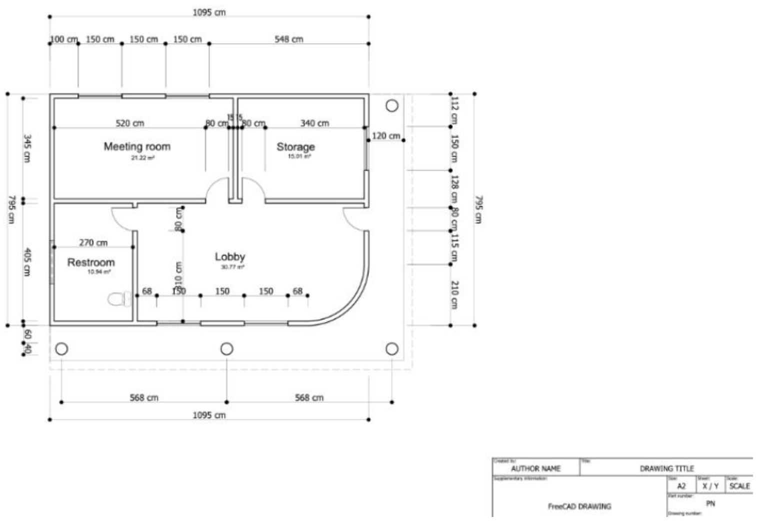




Perspective and Isometric View



Basic Part –3



Perspective Projection

What are You Making?

Outline :-

Work on the FreeCAD doxygen-generated documentation: Propose a better plan, document the modules better, make it clearer to read, etc.

Details :-

The API documentation of FreeCAD is generated with doxygen from the docstrings contained in the source code. It is hosted on https://github.com/FreeCAD/SourceDoc . It is currently not easily readable, the module's structure doesn't list classes and functions, python functionality is not well distinguishable from C++ functionality, and many other problems.

Expected Outcome :-

Identify problems and possible solutions, and propose changes to the docstrings and in-source doxygen instructions to build better docs, and possibly do some css work to produce a cleaner HTML result

There Are **three** Major Components of this Project :-

1. Assessment of Current Documentation :-

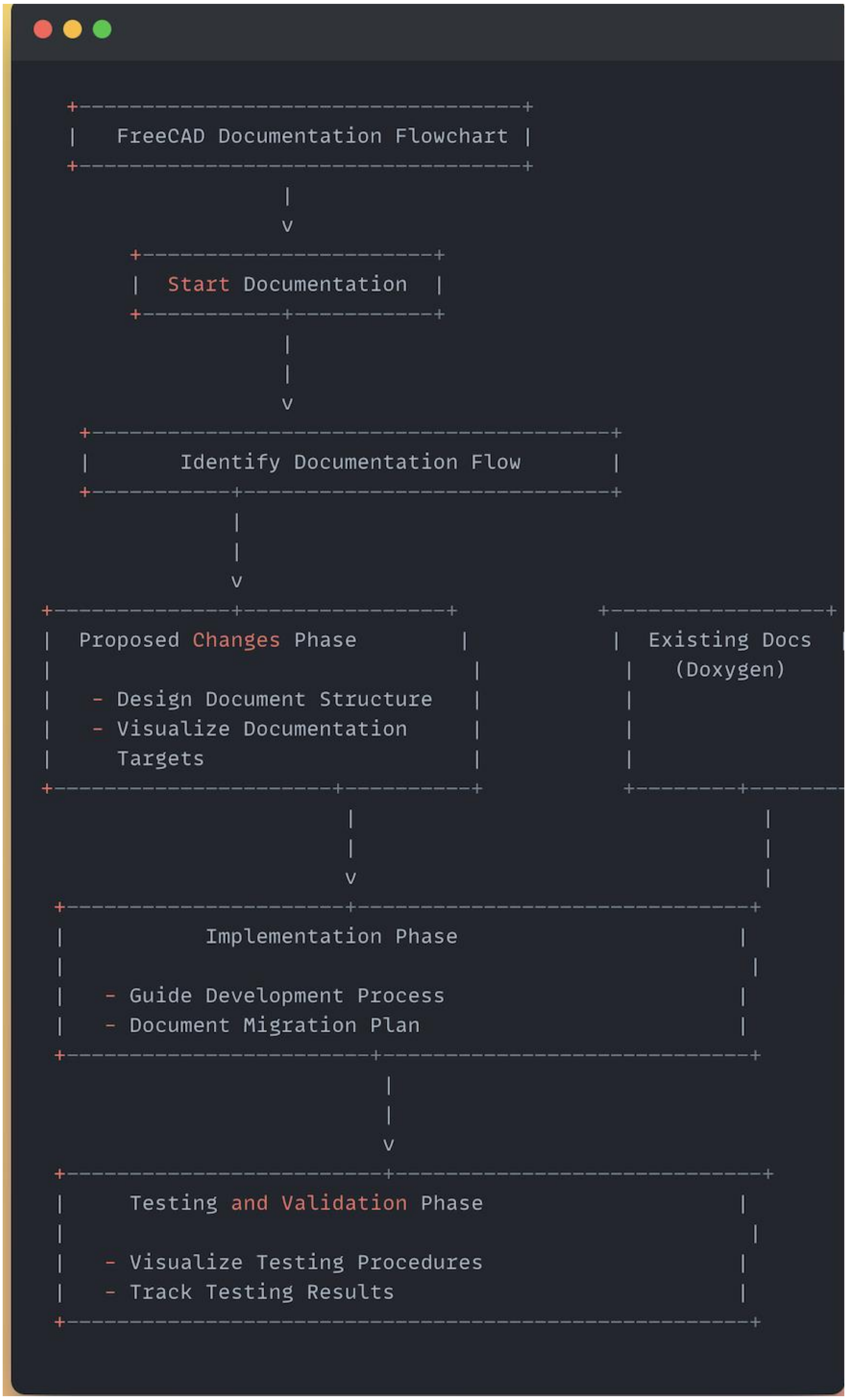
Review the existing FreeCAD API documentation on the SourceDoc GitHub repository.

2. Proposing Solution and Changes :-

Analyse potential solutions to address the identified problems.

3. Implementation and Testing :-

Implement the proposed changes in the FreeCAD source code, focusing on improving the documentation quality.



```
 +------------------------------------+
 |   FreeCAD Documentation Flowchart |
 +------------------------------------+
                  |
                  v
      +------------------------+
      |  Start Documentation   |
      +-----------+------------+
                  |
                  |
                  v
  +-------------------------------------------+
  |       Identify Documentation Flow         |
  +-----------+-------------------------------+
              |
              |
              v
+---------------+------------------+            +------------------+
|  Proposed Changes Phase          |            |  Existing Docs
|                                   |            |   (Doxygen)
|   - Design Document Structure     |            |
|   - Visualize Documentation       |            |
|     Targets                       |            |
+-----------------------+-----------+            +---------+--------
                        |                                  |
                        |                                  |
                        v                                  |
 +-----------------------+-----------------------------------+
 |           Implementation Phase                          |
 |                                                           |
 |   - Guide Development Process                           |
 |   - Document Migration Plan                             |
 +-------------------------+-------------------------------+
                           |
                           |
                           v
 +--------------------------+------------------------------+
 |     Testing and Validation Phase                        |
 |                                                           |
 |   - Visualize Testing Procedures                        |
 |   - Track Testing Results                               |
 +-------------------------------------------------------+
```

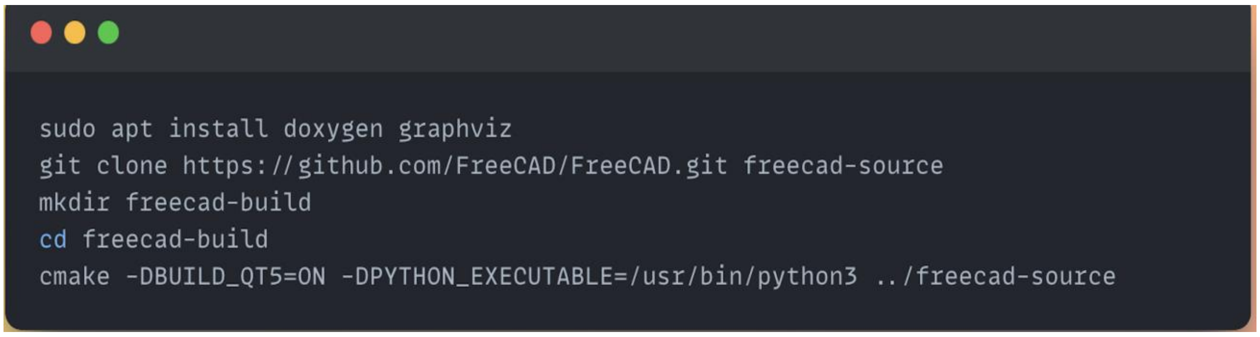
PART 1 :- ASSESSMENT OF CURRENT DOCUMENTATION :-

The documentation is presently housed within the FreeCAD source code repository. It is compiled using the following commands and stored in the designated location:

1. Storage Location: The documentation is stored within the FreeCAD/SourceCode repository.

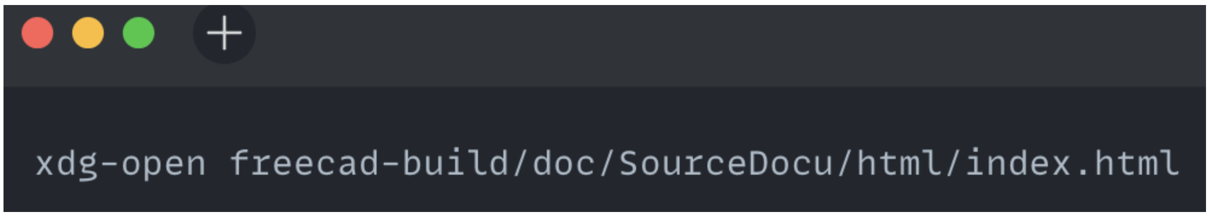
2. Compilation Process: Documentation compilation is initiated through specific commands.

Following are the commands to **Build Documentation:-**



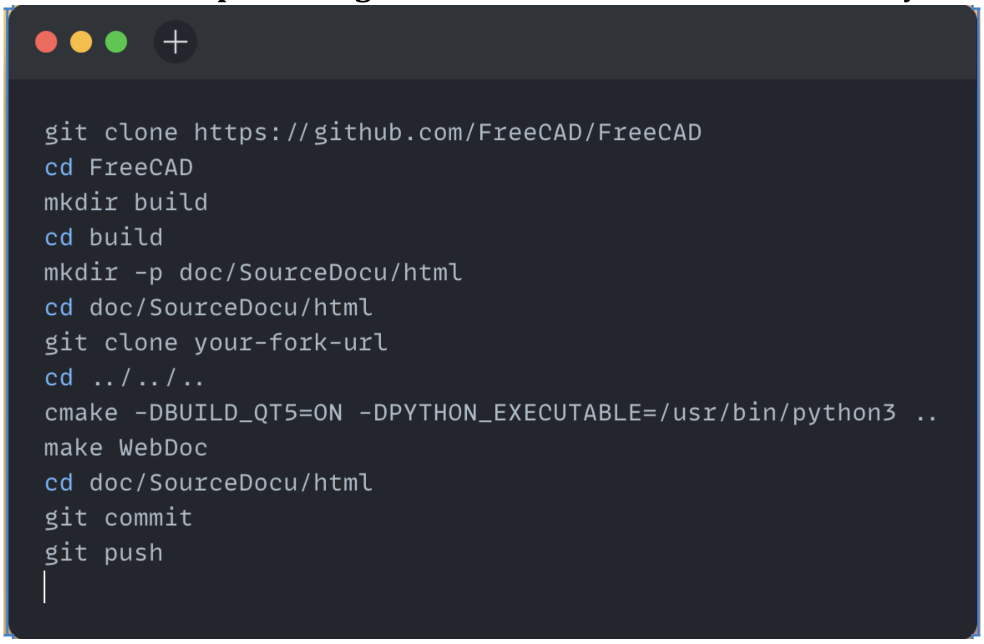
```
sudo apt install doxygen graphviz
git clone https://github.com/FreeCAD/FreeCAD.git freecad-source
mkdir freecad-build
cd freecad-build
cmake -DBUILD_QT5=ON -DPYTHON_EXECUTABLE=/usr/bin/python3 ../freecad-source
```

Then Pointing to index.html we can open a **web browser to view: -**



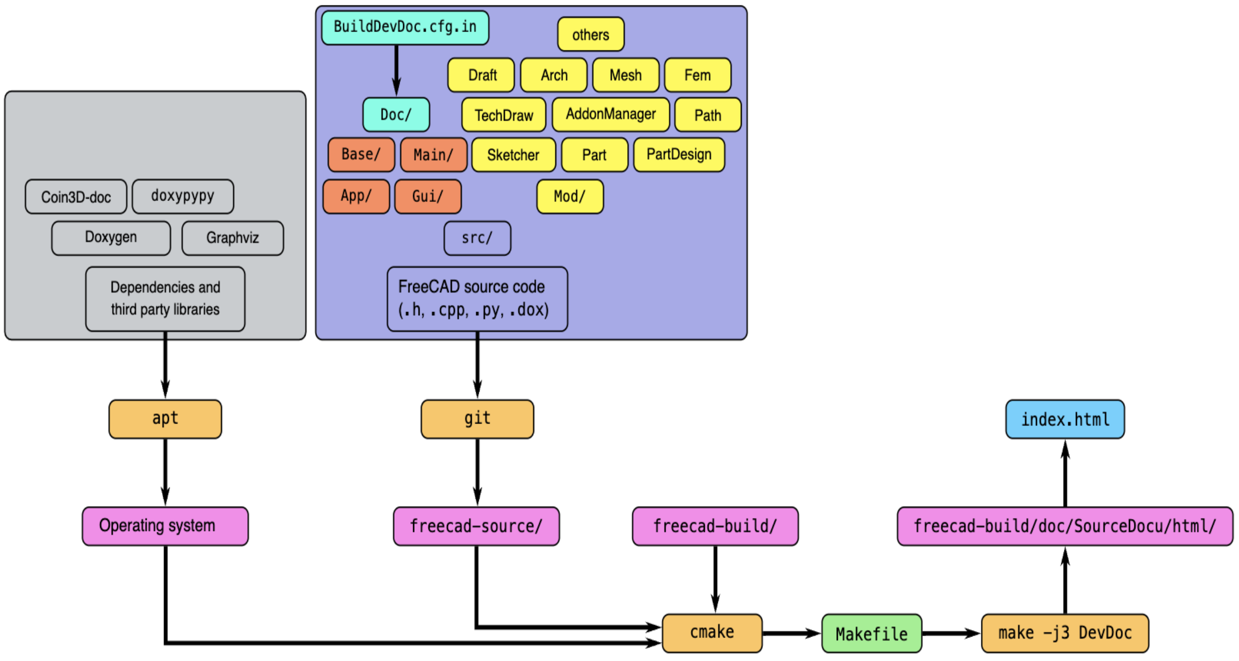
```
xdg-open freecad-build/doc/SourceDocu/html/index.html
```

To **Rebuild, Compare and generate** PR for better documentation system:-



```
git clone https://github.com/FreeCAD/FreeCAD
cd FreeCAD
mkdir build
cd build
mkdir -p doc/SourceDocu/html
cd doc/SourceDocu/html
git clone your-fork-url
cd ../../..
cmake -DBUILD_QT5=ON -DPYTHON_EXECUTABLE=/usr/bin/python3 ..
make WebDoc
cd doc/SourceDocu/html
git commit
git push
```

3. Current Flow to generate Document :-



4. Result Storage: Upon compilation, the generated documentation is stored in the predefined directory.

5. Review Of the Documentation :-

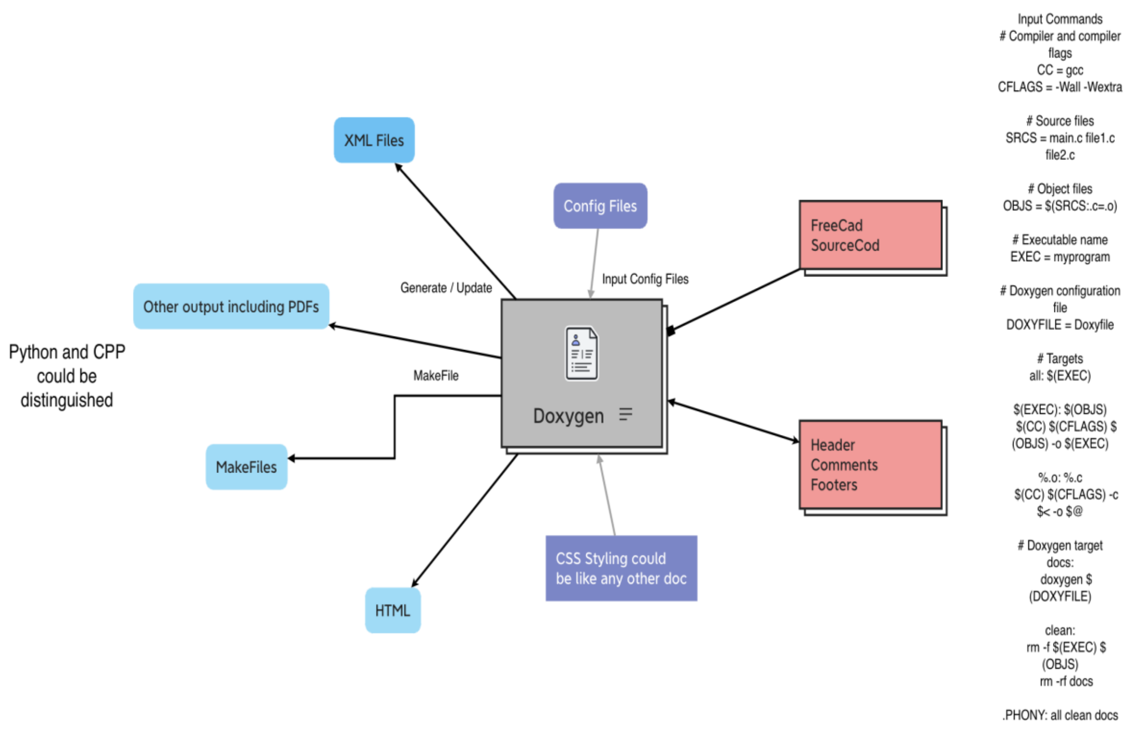
I thoroughly examined the current **FreeCAD API documentation** hosted on the SourceDoc GitHub repository.

I identified areas where Doxygen docstrings were underutilized or inconsistently applied throughout the codebase. I evaluated the clarity and organization of the documentation, focusing on the module structure and the distinction between C++ and Python functionality.

Documentation Issues:

1. Insufficient utilization and standardization of Doxygen docstrings.
2. Mixing of C++ and Python documentation, hindering usability for both types of users.
3. Lack of clear delineation between C++ and Python functionality.
4. Challenges faced by different user groups (experienced C++ developers vs. novice Python users).

PART- 2. PROPOSING SOLUTION AND CHANGES: -



```
Input Commands
# Compiler and compiler
flags
CC = gcc
CFLAGS = -Wall -Wextra

# Source files
SRCS = main.c file1.c
file2.c

# Object files
OBJS = $(SRCS:.c=.o)

# Executable name
EXEC = myprogram

# Doxygen configuration
file
DOXYFILE = Doxyfile

# Targets
all: $(EXEC)

$(EXEC): $(OBJS)
 $(CC) $(CFLAGS) $
(OBJS) -o $(EXEC)

%.o: %.c
 $(CC) $(CFLAGS) -c
$< -o $@

# Doxygen target
docs:
 doxygen $
(DOXYFILE)

clean:
 rm -f $(EXEC) $
(OBJS)
 rm -rf docs

.PHONY: all clean docs
```

How to Utilize Maximum From Doxygen and its structure

1. Standardizing and Enhancing Doxygen Docstrings:

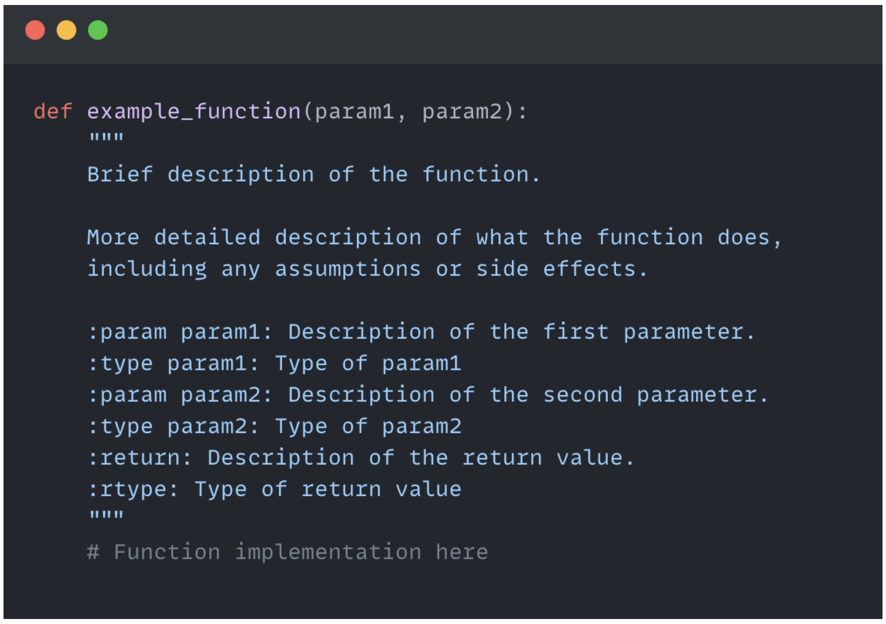
Rationale: Standardizing and enhancing Doxygen docstrings throughout the codebase will improve readability and clarity of the documentation. Consistent formatting and comprehensive descriptions will make it easier for developers to understand the functionality of each module, class, and function.

For Python Functions:

Detailed Description: More extensive explanation covering the function's behaviour, parameters, and return value.

Parameters: Description of each parameter, including its name, type, and usage.

Return Value: Description of the value returned by the function, including its type.



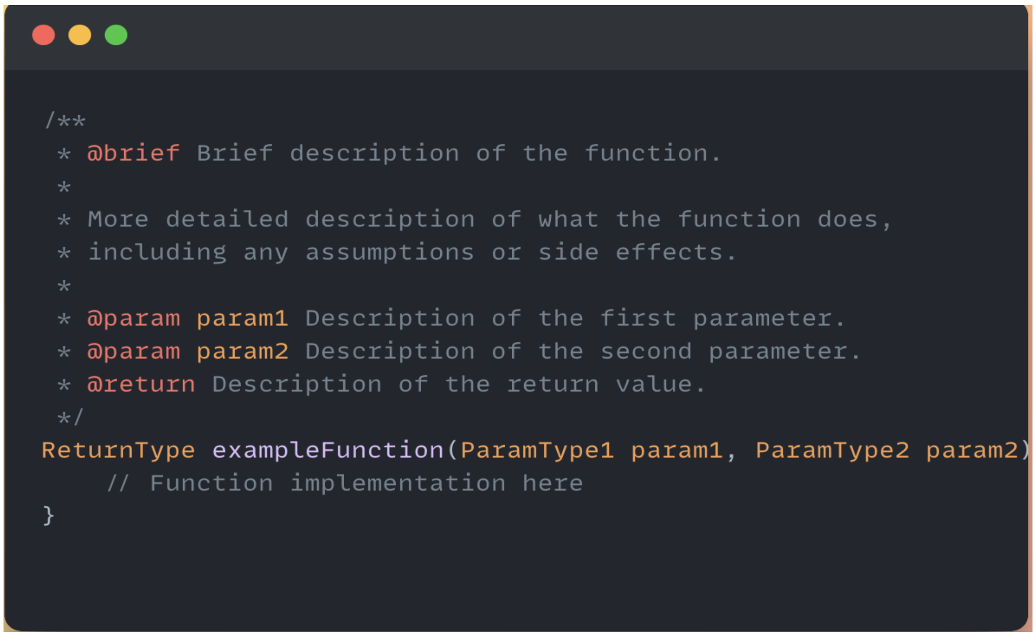
```python
def example_function(param1, param2):
    """
    Brief description of the function.

    More detailed description of what the function does,
    including any assumptions or side effects.

    :param param1: Description of the first parameter.
    :type param1: Type of param1
    :param param2: Description of the second parameter.
    :type param2: Type of param2
    :return: Description of the return value.
    :rtype: Type of return value
    """
    # Function implementation here
```

<u>For CPP Function:-</u>

<u>Detailed Description:</u> Additional details about the function's behavior, including any relevant information for callers.

<u>Parameters:</u> Description of each parameter, prefixed with @param, followed by its name and description.

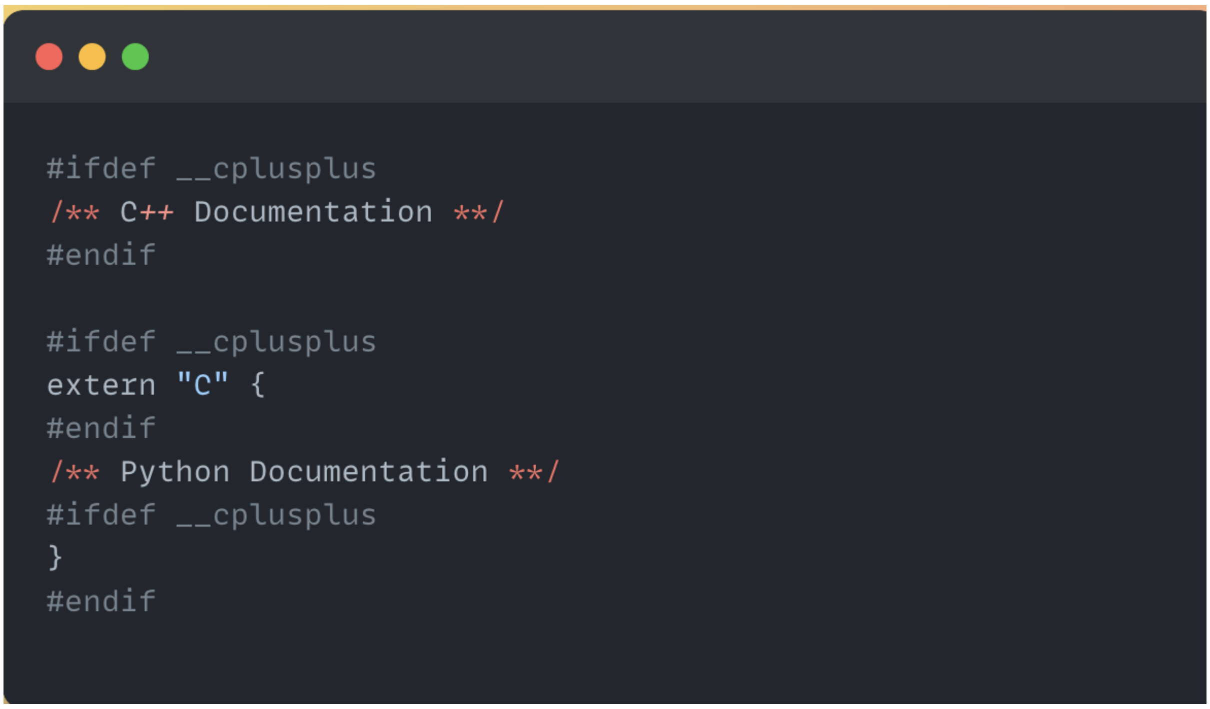
<u>Return Value:</u> Description of the value returned by the function, prefixed with @return



```cpp
/**
 * @brief Brief description of the function.
 *
 * More detailed description of what the function does,
 * including any assumptions or side effects.
 *
 * @param param1 Description of the first parameter.
 * @param param2 Description of the second parameter.
 * @return Description of the return value.
 */
ReturnType exampleFunction(ParamType1 param1, ParamType2 param2)
    // Function implementation here
}
```

<u>2. Implementing a System to Differentiate Between C++ and Python Documentation</u>

Rationale: Differentiating between C++ and Python documentation targets will cater to the distinct needs of each user group. By providing separate documentation targets, users can access information relevant to their programming language of choice without being overwhelmed by irrelevant details.



```
#ifdef __cplusplus
/** C++ Documentation **/
#endif

#ifdef __cplusplus
extern "C" {
#endif
/** Python Documentation **/
#ifdef __cplusplus
}
#endif
```

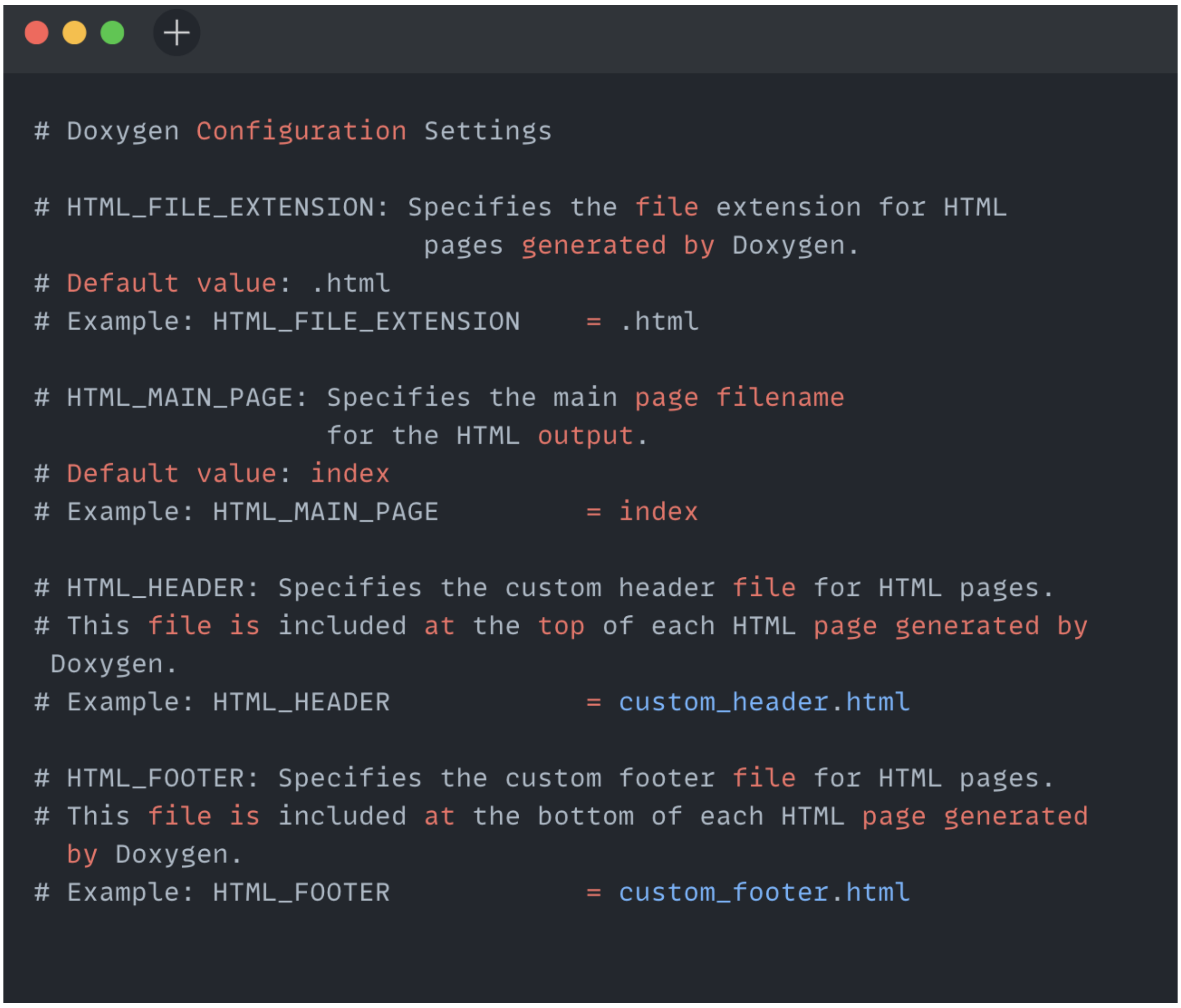
3. Exploring Doxygen Docsettings:

Rationale: Doxygen offers extensive configuration settings impacting documentation quality. Exploration and optimization of these settings are vital for enhancing FreeCAD documentation.

Configuration Exploration: Review default Doxygen settings and assess their suitability for FreeCAD. Identify key settings related to documentation structure, formatting, and integration.

Optimization Strategies: Develop optimization strategies for identified settings to improve readability and navigation. Test and validate optimizations to ensure effectiveness across different user scenarios.

Documentation of Findings: Document exploration findings and rationale behind optimization choices. Integrate optimized settings into proposed documentation improvements for FreeCAD.



```
# Doxygen Configuration Settings

# HTML_FILE_EXTENSION: Specifies the file extension for HTML
                        pages generated by Doxygen.
# Default value: .html
# Example: HTML_FILE_EXTENSION    = .html

# HTML_MAIN_PAGE: Specifies the main page filename
                  for the HTML output.
# Default value: index
# Example: HTML_MAIN_PAGE         = index

# HTML_HEADER: Specifies the custom header file for HTML pages.
# This file is included at the top of each HTML page generated by
 Doxygen.
# Example: HTML_HEADER            = custom_header.html

# HTML_FOOTER: Specifies the custom footer file for HTML pages.
# This file is included at the bottom of each HTML page generated
  by Doxygen.
# Example: HTML_FOOTER            = custom_footer.html
```

Explanation of the Snippet:-

HTML_FILE_EXTENSION: This setting allows customization of the file extension for HTML pages generated by Doxygen. By default, Doxygen generates HTML pages with the .html extension. However, this setting provides flexibility to change it to another extension if desired.

HTML_MAIN_PAGE: This setting specifies the filename of the main page for the HTML output. The default value is "index". This page serves as the entry point for the generated documentation and is typically the first page users see when accessing the documentation.

HTML_HEADER: This setting allows inclusion of a custom header file for HTML pages generated by Doxygen. The specified file will be included at the top of each HTML page, providing an opportunity to add custom branding, navigation, or other elements to the documentation layout.

HTML_FOOTER: Like HTML_HEADER, this setting specifies a custom footer file for HTML pages generated by Doxygen. The specified file will be included at the bottom of each HTML page, allowing for the addition of custom content such as copyright information, links, or acknowledgments.

PART - 3. Implementation and Testing :-

Key Documents Serving as Inspiration:-

https://wiki.freecad.org/The_FreeCAD_source_code - FreeCAD Source Code doc

https://doc.qt.io/qt-6/qobject.html - QT

https://www.coin3d.org/Coin/html/ - COIN

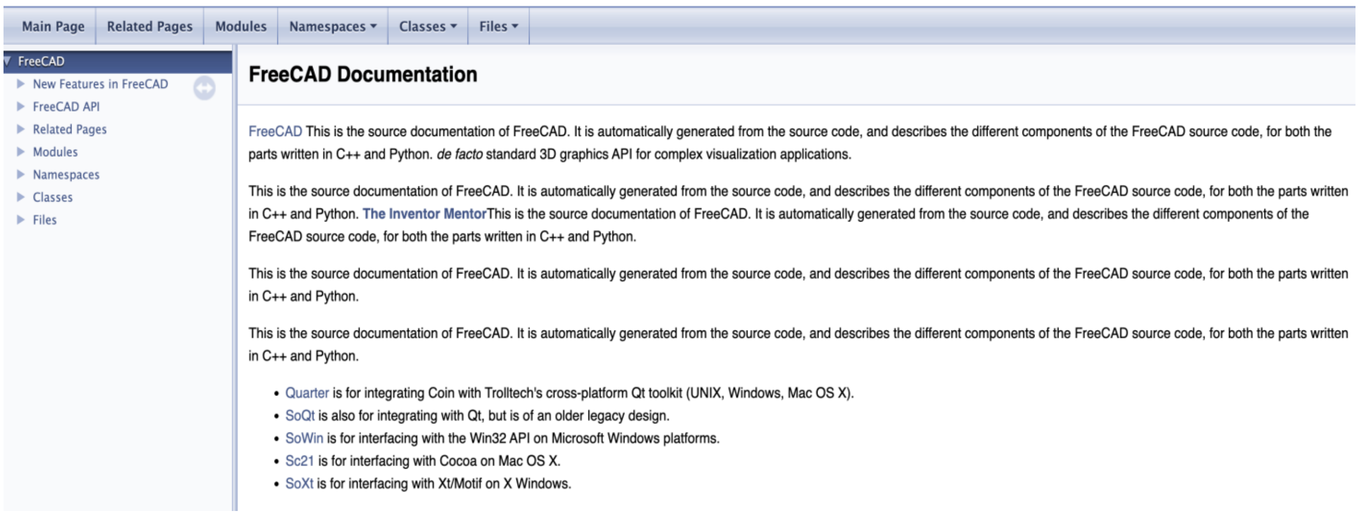
1. Implementation Plan:

Updating Doxygen Docstrings: Review existing docstrings and identify areas for improvement. Develop guidelines and templates for writing standardized docstrings. Assign tasks to developers for updating docstrings throughout the codebase.

Modifying Code Structure: Analyse the current code structure and identify areas requiring modification for better documentation organization. Restructure code as needed to align with proposed documentation improvements. Ensure changes are made in a modular and backward-compatible manner.

Integrating Necessary Tools or Scripts: Evaluate existing tools or scripts that can aid in documentation enhancement. Develop custom tools or scripts if necessary to automate repetitive tasks or enforce documentation standards. Integrate tools or scripts into the development workflow to streamline the documentation process.

Doxyfile Configurations and Custom Snippets: Customize the Doxyfile configuration to optimize Doxygen output for FreeCAD documentation. Define custom snippets or macros to enhance documentation readability and organization. Ensure consistency and maintainability of Doxyfile configurations across the project.



FreeCAD Documentation

FreeCAD This is the source documentation of FreeCAD. It is automatically generated from the source code, and describes the different components of the FreeCAD source code, for both the parts written in C++ and Python. *de facto* standard 3D graphics API for complex visualization applications.

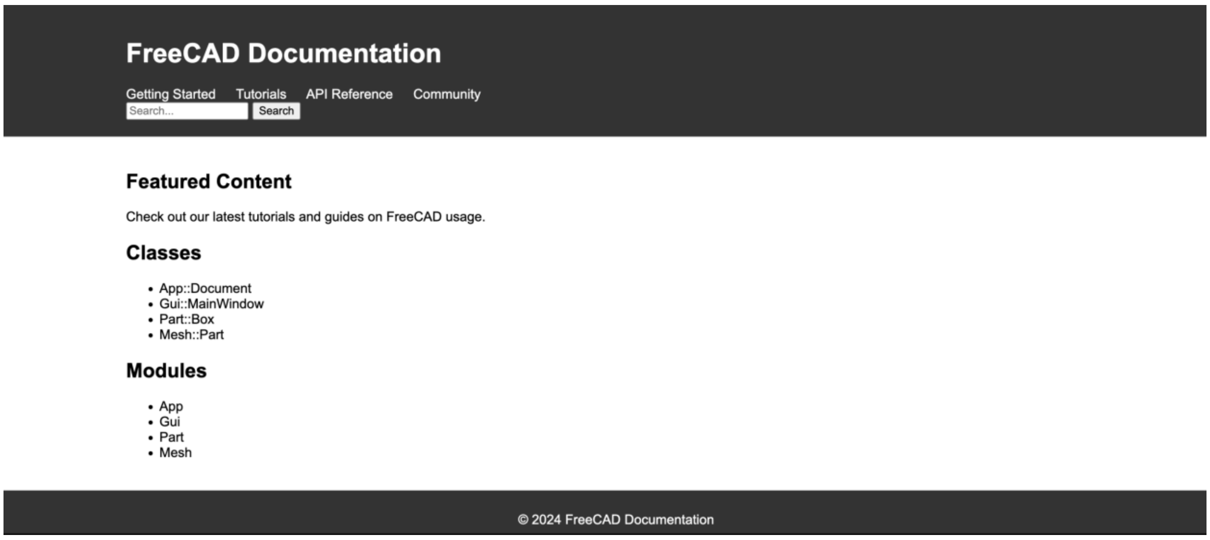
This is the source documentation of FreeCAD. It is automatically generated from the source code, and describes the different components of the FreeCAD source code, for both the parts written in C++ and Python. **The Inventor Mentor**This is the source documentation of FreeCAD. It is automatically generated from the source code, and describes the different components of the FreeCAD source code, for both the parts written in C++ and Python.

This is the source documentation of FreeCAD. It is automatically generated from the source code, and describes the different components of the FreeCAD source code, for both the parts written in C++ and Python.

This is the source documentation of FreeCAD. It is automatically generated from the source code, and describes the different components of the FreeCAD source code, for both the parts written in C++ and Python.

- Quarter is for integrating Coin with Trolltech's cross-platform Qt toolkit (UNIX, Windows, Mac OS X).
- SoQt is also for integrating with Qt, but is of an older legacy design.
- SoWin is for interfacing with the Win32 API on Microsoft Windows platforms.
- Sc21 is for interfacing with Cocoa on Mac OS X.
- SoXt is for interfacing with Xt/Motif on X Windows.

Sample MOCKUP –1


FreeCAD Documentation

Getting Started Tutorials API Reference Community

Search... Search

Featured Content

Check out our latest tutorials and guides on FreeCAD usage.

Classes

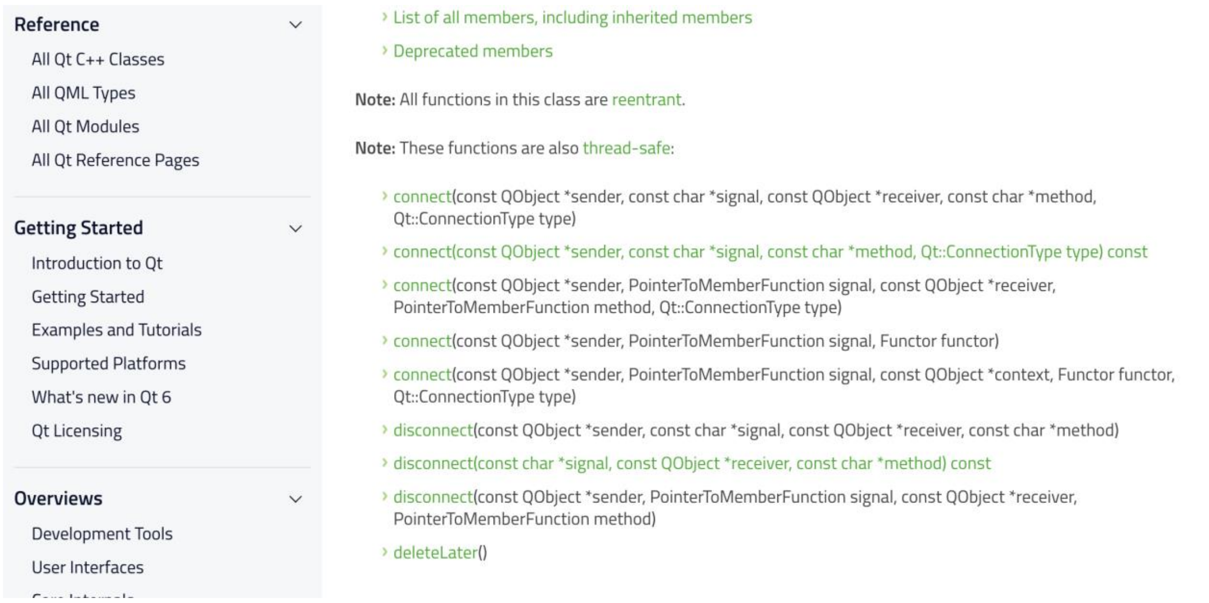
- App::Document
- Gui::MainWindow
- Part::Box
- Mesh::Part

Modules

- App
- Gui
- Part
- Mesh

© 2024 FreeCAD Documentation

Sample MOCKUP-2


Reference
- All Qt C++ Classes
- All QML Types
- All Qt Modules
- All Qt Reference Pages

Getting Started
- Introduction to Qt
- Getting Started
- Examples and Tutorials
- Supported Platforms
- What's new in Qt 6
- Qt Licensing

Overviews
- Development Tools
- User Interfaces

› List of all members, including inherited members

› Deprecated members

Note: All functions in this class are reentrant.

Note: These functions are also thread-safe:

› connect(const QObject *sender, const char *signal, const QObject *receiver, const char *method, Qt::ConnectionType type)

› connect(const QObject *sender, const char *signal, const char *method, Qt::ConnectionType type) const

› connect(const QObject *sender, PointerToMemberFunction signal, const QObject *receiver, PointerToMemberFunction method, Qt::ConnectionType type)

› connect(const QObject *sender, PointerToMemberFunction signal, Functor functor)

› connect(const QObject *sender, PointerToMemberFunction signal, const QObject *context, Functor functor, Qt::ConnectionType type)

› disconnect(const QObject *sender, const char *signal, const QObject *receiver, const char *method)

› disconnect(const char *signal, const QObject *receiver, const char *method) const

› disconnect(const QObject *sender, PointerToMemberFunction signal, const QObject *receiver, PointerToMemberFunction method)

› deleteLater()

QT-DOC MockUP-3

2. Testing and Validation:

Testing Methodology: Conduct unit testing to verify the accuracy of updated docstrings and code modifications. Perform integration testing to ensure that the revised documentation integrates seamlessly with existing code and other components. Utilize static analysis tools to identify any potential issues or inconsistencies in the documentation.

Testing Strategy for Revised Documentation: Develop test cases covering common usage scenarios for both C++ and Python users. Verify that the revised documentation effectively addresses the needs of both user groups, providing clear and relevant information. Solicit feedback from beta testers or user groups to gather insights on documentation usability and clarity.

Utilized Testing Frameworks or Tools: Employ testing frameworks such as Google Test for C++ code and pytest for Python code to automate testing processes. Use code coverage tools to assess the comprehensiveness of test coverage for the revised documentation. Document testing procedures and results to facilitate ongoing maintenance and future improvements.

3. Documentation Migration:

Migration Plan for Python API Documentation: Assess the feasibility of migrating Python API documentation to a markdown-based format. Develop a migration plan outlining the steps involved, including content conversion and formatting adjustments. Allocate resources and assign responsibilities for executing the migration plan effectively. Integration with Existing Markdown Documentation: Ensure seamless integration of migrated Python API documentation with existing markdown documentation for FreeCAD. Update navigation links and cross-references to reflect the new documentation structure. Test the integrated documentation to confirm accessibility and usability for end users.

What technologies (programming languages, etc.) will you be using?

Doxygen: To generate API reference documentation from annotated C++ source code.
C++: To understand and contribute to the FreeCAD source code.
HTML and XML: Output formats for the generated API documentation.
Markdown: Used for writing and formatting documentation content.
Python: Utilized for scripting, automation, and development of code examples.
Bash Scripting: Used for automation of system tasks and setup procedures.
Git: Employed for version control and collaboration on documentation repositories.

4.TimeLine:-

Break down the entire project into chunks and tell us what will you work on each week.

Weeks	Tasks to be Completed

Pre-GSOC (Apr – May)	• Investing additional time in exploring the FreeCAD codebase will lead to further optimization of the project plans.
Community Bonding Period (1st May – 26th May)	• Collaborate with mentors to optimize. • Resolve and Explore additional technologies and deepen my understanding of FreeCAD.
Week-1 (27th May – June 2nd)	• I will resolve all Missing and empty API documentation related issues. • Analyse existing Doxygen docstrings and code structure. Evaluate Doxygen configuration settings and identify potential optimizations.
Week-2 (June 3rd – 9th)	• Develop templates for standardized Doxygen docstrings.
Week-3 (June 10th –16th)	• Analyze codebase to identify areas requiring structural modifications for better documentation organization. • Implement changes to code structure while ensuring backward compatibility.
Week-4 (June 17th – 23rd)	• Review and refine code modifications to align with documentation enhancement goals.
Week-5 (June 24th – 30th)	• Evaluate existing tools or scripts for documentation enhancement. • Develop custom tools or scripts to automate tasks or enforce standards. • Integrate tools or scripts into the development workflow.
Week-6 (July 1st – 7th)	• Midterm Evaluation of all the work done till now.

	• Provide training to team members on usage of newly integrated tools or scripts.
Week-7 (July 8th – 14th) Midterm Evaluation	• Customize Doxyfile configuration to optimize output for FreeCAD documentation. • Define custom snippets or macros to enhance documentation readability.
Week-8 (July 15th –21st)	• Conduct testing to validate the effectiveness of Doxygen configuration changes.
Week-9 (July 22nd – 28th)	• Develop test cases covering common usage scenarios for both C++ and Python users. • Plan unit testing to verify accuracy of updated docstrings and code modifications.
Week-10 (July 29th – Aug 4th)	• Conduct unit testing to verify accuracy of updated docstrings and code modifications. • Perform integration testing to ensure seamless integration of revised documentation with existing codebase.
Week-11 (Aug 5th – Aug 11th)	• Solicit feedback from beta testers or user groups to assess documentation usability and clarity.
Week-12 (Aug 12th – Aug 18th)	• Conduct final testing to address any outstanding issues or concerns. • Review documentation enhancements and finalize documentation for release.
Final Evaluation	Final evaluation/testing.

5. Some Important Questions:-

How many hours will you spend each week on your project?

During my college summer vacation, commencing from May 15th to July 22nd, I anticipate being able to dedicate approximately 50-55 hours per week to GSoC-related activities. Prior to that period, I can allocate around 40-45 hours per week. With no other commitments during this time, I am fully prepared to devote most of my time to GSoC.

How will you report progress between evaluations?

I will maintain an active presence on GitHub by regularly submitting pull requests. Additionally, **I will actively engage on social media platforms such as LinkedIn, Instagram, and Twitter, where I will share updates on my progress, interact with the community.**

Discuss your post-GSoC plans. Will you continue contributing to Sugar Labs after GSOC ends?

My most important plan will be to helping in facilitate python API migration.

In my post-GSoC plan, I aim to enhance the API documentation for FreeCAD to make it more and more robust and informative. This involves thoroughly documenting each aspect of the API, every minute details and enhancing it. By providing clear and detailed documentation, developers will have a better understanding of how to utilize the FreeCAD API effectively in their projects.

THANKYOU!!!