

Personal Information

- Name: Bhavya Bansal
- Email: bhavyabansal4321@gmail.com
- CollegeMail : bhavyabansalcs20@bsacet.org
- Github: [Bhavya Bansal](#)
- LinkedIn : [Bhavya Bansal](#)
- Language : English
- Location: Mathura, UP, India
- Time Zone: GMT+5:30(New Delhi, India)
- University: Dr. A.P.J. Abdul Kalam Technical University (APJAKTU)
- College : B.S.A College of Engineering and Technology (BSACET)
- Major: Computer Science And Engineering

ABOUT ME

I am a diligent third-year undergraduate student pursuing a Bachelor of Technology in Computer Science from BSACET, Mathura. My proficiency in Python is the result of the [Python for Everybody Specialization](#), a renowned course offered by the University of Michigan and [IBM Data Science Professional Certificate](#), which has equipped me with essential programming skills. Additionally, I dedicatedly solved 4-5 problems from the [DSA cracker](#) by Luv Bubbor on a daily basis, which has augmented my proficiency in Python programming and C++.

My passion for programming motivated me to engage in competitive programming on prestigious platforms such as Leetcode, Codechef, and Greekforgreek, which enabled me to acquire a comprehensive understanding of diverse algorithms. On Codechef, I achieved the highest rating of 1621 (3 stars), and I have solved over 100 problems on [Leetcode](#).

My zeal for Learning led me to secure an internship as a Software Engineer at [Telaverge Communications](#) during my second year of BTech, where I gained invaluable experience working on real-time projects. Through this opportunity, I learned practical skills such as debugging large projects and finding online solutions to problems. Furthermore, I contributed various enhancements and features to their Core Regal Project, which deepened my understanding of both Python3 and Python2, as many Regal Rest APIs were implemented in Python2, leading to several bugs that I resolved during my tenure. Additionally, I successfully tackled various Regression issues by identifying and fixing their root cause.

I also proved my mettle by cracking the coding test of [Amazon ML Summer School '22](#), an achievement for which only 3000 students from all over India were selected([Acknowledgement](#)). This opportunity enabled me to learn various Machine Learning and Deep Learning algorithms in Python, which has further honed my expertise in programming.

Project Description

- Research and Analysis

- Reviewed the Sugar source code changes through issues ([Link](#)) that were made for porting to Python 3.
- Reviewed some Sugar source code changes through issues ([Link](#)) that were made for porting telepathy bindings.
- Read source code in sugar/src/main.py and other parts related, understand the core concepts of Sugar Desktop and Activities.
- Successfully installed the Sugar in my Virtual machine.
- Also finished setting up the Development Environment for Native users and Installed Required Dependencies.

```
bhavya@Bhavya:~/SugarLabsSetup$ ls  
sugar sugar-artwork sugar-datastore sugar-toolkit sugar-toolkit-gtk3
```

- Made myself familiar with the codebase of sugar by working and resolving issues on github.
- Accomplished qualification task posted by mentor, it's [here](#).
 - Experience With Python : As I mentioned, I am very much familiar with Python due to the time I spent at Telaverge Communications as a software developer. I have also created various projects in Python. In addition, I have completed plenty of courses on Coursera.
 - Experience With TelepathyGlib : Until now, I have developed a great understanding of this library by going through the official documentation of Telepathy-python and TelepathyGlib([Link](#)). I have also worked on the sugar codes and reviewed the issues of porting to TelepathyGlib. Additionally, I have read the entire documentation created by SugarLab for '[porting to python3](#)'. Moreover, I have gone through the Collabwrapper library and multiple use cases, as mentioned in the 'porting to python3' documentation.
 - Experience with Sugar Desktop and activities : To familiarize myself with the Sugar desktop environment, I first installed Sugar on my machine and then completed setting up its native development environment. From there, I delved into the codebase, debugged the issues that arose in order to gain deeper insights into the workings of the system.
- Requirements for Porting to Python3
 - Application Development : I have successfully completed various projects that demonstrate my proficiency in this field. One such project is the development of an advanced [Facial Recognition Based Attendance Management system](#) using Python. In this project, I used tkinter to create the GUI and connected my Python code with a SQL database locally to store student information. I also integrated automatic face recognition

using the OpenCV library, and marked attendance automatically once it recognized a face.

Link to the Presentation i have created to demonstrate in college : [Link](#)

Additionally, in my second year, I created a COVID-19 dashboard that involved combining Python with live Open Source API fetching data in JSON format to display a live worldwide COVID-19 database on the web by using the Django framework.

Link to the Presentation i have created to demonstrate in college : [Link](#)

I have experience in application development in Python, as demonstrated through my projects. I believe that the syntax may only change with different libraries, and with regards to Gtk3, I have already gone through various tutorials and gained a deeper understanding. Additionally, I have developed a clear understanding of collaborative feature implementation.

- Sugar activity development : I have reviewed the documentation for creating an activity in Sugar. Afterwards, I selected an activity and attempted to comprehend its flow by following the same documentation. I will be conducting a demo of the activity soon.
- Difference Between Python2 and Python3 : i have created a document([Link](#)) for my reference by following [Official documentation](#) of Changes in Python3 also i am well aware of this part where it might lag cause i have ported various API in Telaverge communication.

What am I making?

I will be porting Sugar and its associated Activities to Python 3. This will involve updating all the necessary dependencies, including telepathy-python, to ensure compatibility with the latest version of Python.

To further ensure the success of the Sugar and core Activities porting process, I will thoroughly review the source code changes since 0.112 that were made for porting to Python 3. This will allow me to identify any areas that may require special attention or additional modification to ensure the smooth functioning of the software.

To validate that the code changes have been implemented correctly, I will design tests and iterate until the tests have sufficient [coverage](#) for all identified code changes. This will ensure that the porting process has been successful and that the software continues to function reliably and efficiently.

In the event of any regressions in Sugar, the Toolkit, or the Datastore, I will work towards fixing them promptly to prevent any further issues from arising. This will ensure that the porting process is completed smoothly and without any unexpected disruptions.

I will work towards porting Telepathy bindings to TelepathyGLib to ensure compatibility with the latest version of Python. I will also port affected activities to the latest Sugargame or CollabWrapper library to ensure that the software remains up to date and fully functional.

In addition to porting the software to Python 3, I will also work towards fixing any problems that prevent the affected activities from being ported or used. This will ensure that the software is fully functional and can be used by users without any issues.

As an experienced developer with a strong background in software development and an in-depth understanding of the differences between Python 2 and 3, I am confident in my ability to successfully port Sugar and its Activities to Python 3. Through updating all dependencies, designing and implementing comprehensive tests, resolving issues and regressions, and porting activities to the latest Sugargame or CollabWrapper library, I will ensure that the software remains compatible, reliable, and fully functional. With my expertise and commitment to delivering high-quality work within the specified timeline, I am confident that I can complete this task successfully and meet all expectations.

Observations

- Port to Python3

While analyzing the Sugar source code for version 0.112 in the context of porting it to Python 3, it was discovered that the primary issues were related to the change in the default storage format of text from bytes to Unicode in Python 3. This required the use of the `encode()` method to convert Unicode to bytes and the `decode()` method to convert bytes to Unicode. The code also had to handle instances where both byte and Unicode strings were used, with some places using Unicode strings and others using byte strings created using `bytes()`, `b""`, or `encode()` method. Additionally, the long data type used in Python 2 for handling large numbers was replaced with the `int` data type in Python 3.

Porting also required changes in syntax and functionality such as replacing `iteritems()` with `.items()`, and `file()` with `open()` for file I/O. In Python 3, the `sort()` function's parameters were modified, the `cmp` parameter was removed, and only two optional parameters, `reverse` and `key`, were retained.

Dependencies in the Sugar code were updated for compatibility with Python 3, including renaming the `httplib` module to `http.client`, the `xmlrpclib` module to `xmlrpc.client`, and the `urlparse` module to `urllib.parse`. The `commands` module was not available in Python 3, and its

functionality was replaced with the subprocess module. The cPickle module was optimized and integrated into the pickle module for better performance.

In addition, the StringIO functionality in Python 2 was available through the io module in Python 3, and the http.server and socketserver modules replaced SimpleHTTPServer and SocketServer modules in Python 2, with additional features such as HTTP/2 support. Finally, in Python 2, the ConfigParser class was part of the ConfigParser module in the ConfigParser package, whereas in Python 3, the same class was a part of the configparser module.

Additionally, it was observed that the Webkit1 technology had several limitations, and many of its functions were either deprecated or removed. As a result, support for Webkit1 was removed, and instead, we ensured that the more modern and feature-rich Webkit2 technology was used."

- **Port to telepathyGlib**

I have thoroughly reviewed the porting to telepathyGlib and found that support for 'from telepathy.client import Connection, channel' is not available. Instead, I noticed that we establish a D-Bus connection to a remote service by first creating a connection to the D-Bus (using dbus.systembus or dbus.sessionbus), then getting a proxy object representing the remote D-Bus service using the 'get_object(service, path)' function. We then use the proxy object to call methods on the remote service or listen for signals.

To store information about the D-Bus connection, we created the 'self.connection' dictionary, which includes information about the interfaces supported by the remote object. We also have the '__conn_get_interfaces_reply_cb' and '__connection_ready_cb' methods, which are called when the connection is established and the remote object's interfaces are available. These methods allow for additional setup and initialization to be performed.

Also, we can no longer import constants directly from the telepathy module. Instead, we have created our constants such as CONNECTION_INTERFACE_REQUESTS, CHANNEL, CHANNEL_DISPATCHER, CONNECTION_HANDLE_TYPE_CONTACT, and others from the telepathyGlib module using the corresponding interfaces and constants provided by the Telepathy framework.

- **Activities**

I have gone through several activities present in Sugar and shortlisted a few. Specifically, I focused on activities that have not yet been ported to Python 3. Please let me know if there are

any priority activities that should be addressed first, or if there are any other changes that need to be added in my to do's list. I am committed to completing the 30 activities listed below during my time, and will work diligently to do so. However, in case of any unexpected scenarios, I will keep you informed about any changes in the timeline or completion status.

- Activities/Amazonas Tortuga
- Activities/Analyze
- Activities/Ayne
- Activities/Bounce
- Activities/Browse Gmail
- Activities/Colors
- Activities/CookieSearch
- Activities/Club othello
- Activities/Devtutor
- Activities/Erikos
- Activities/EditFont
- Activities/FollowmeButia
- Activities/Guido van robot
- Activities/I can read
- Activities/Jamedia tube
- Activities/Jamedia imagenes
- Activities/Jamtank
- Activities/Level
- Activities/Map
- Activities/Paths
- Activities/Pilas
- Activities/Pydebug
- Activities/Read SD Comics
- Activities/ShowNTell
- Activities/SocialCalc
- Activities/TamTam
- Activities/Teacher share
- Activities/Timeline
- Activities/TurtleArt/Extras
- Activities/TurtleConfusion

Besides porting these activities will work as per the things mentioned in the Project Description, Timelines and also as directed by the mentor.

How Will My Work Impact SugarLabs?

My contributions to SugarLabs can have a positive impact on the project and its community. Successfully porting Sugar and its Activities software to Python 3 will ensure that the software

remains up-to-date and compatible with the latest technologies, improving its reliability and functionality. As a result, users will have access to the latest features and enhancements, providing them with a more productive, robust and engaging experience.

Identifying and resolving any issues and regressions that may arise in SugarLabs can enhance the user experience for its users, which can lead to greater adoption and engagement with the project.

I can contribute to SugarLabs by porting activities to the latest libraries. This will help ensure that SugarLabs remains compatible with other software systems and libraries, providing users with a more seamless experience. My contributions can also inspire other developers to get involved in the SugarLabs community and contribute to its growth and success.

My work and contributions can have a significant impact on the project and its users. By contributing to SugarLabs, I can help make a positive impact on education and the lives of those who use this platform.

Contributions in SugarLabs:-


1. Initially, I worked on the issue "[where /usr/bin/sugar does not exit if window is closed#952](#)" I also had multiple discussions with James and Ibiam regarding the same issue.

The issue was observed on my machine where only the cursor was changing to 'sugar', and then it was crashing because the screen was returning as None.

To fix this, I added a validation check in `/src/jarabe/main.py` before changing the cursor. This check ensured that if the screen was coming as None, it safely exited from the system to prevent any further changes in the machine.

Link to discussion:

(<https://github.com/sugarlabs/sugar/issues/952#issuecomment-1482503787>)

 **bhavyabansal9068** commented 4 days ago • edited

@chimosky @quozl
Hello team,

Please guide me based on my analysis on this issue.
I have investigated the issue and have come to the conclusion that the problem lies in the `sugar/src/jarabe/main.py` file. Specifically, the `__start_window_manager()` function is being called internally and is changing the cursor of the virtual machine without performing verification.

```
def __start_window_manager():
    global _cursor_theme_settings, _cursor_theme

    _cursor_theme_settings = Gio.Settings.new('org.gnome.desktop.interface')
    _cursor_theme = _cursor_theme_settings.get_string('cursor-theme')
    _cursor_theme_settings.set_string('cursor-theme', 'sugar')

    __restart_window_manager()

    screen = Wnck.Screen.get_default()
    screen.connect('window-manager-changed', __window_manager_changed_cb)

    __check_for_window_manager(screen)
```

This is why no processes appear to be running even after closing the GNOME terminal but cursor is changed. Upon reviewing the logs, I also noticed that when we trigger Sugar from GNOME terminal, it fails because screen 0 is already in use. However, the root cause of this issue is the cursor change in the code.

Link to PR(Closed):

(<https://github.com/sugarlabs/sugar/pull/964>)

```
198     def __start_window_manager():
199         global _cursor_theme_settings, _cursor_theme
200
201 +     screen = Wnck.Screen.get_default()
202 +     if screen is None:
203 +         return sys.exit()
204 +
205         _cursor_theme_settings = Gio.Settings.new('org.gnome.desktop.interface')
206         _cursor_theme = _cursor_theme_settings.get_string('cursor-theme')
207         _cursor_theme_settings.set_string('cursor-theme', 'sugar')
208
209         __restart_window_manager()
210
211 -     screen = Wnck.Screen.get_default()
211         screen.connect('window-manager-changed', __window_manager_changed_cb)
212
213         __check_for_window_manager(screen)
214
```

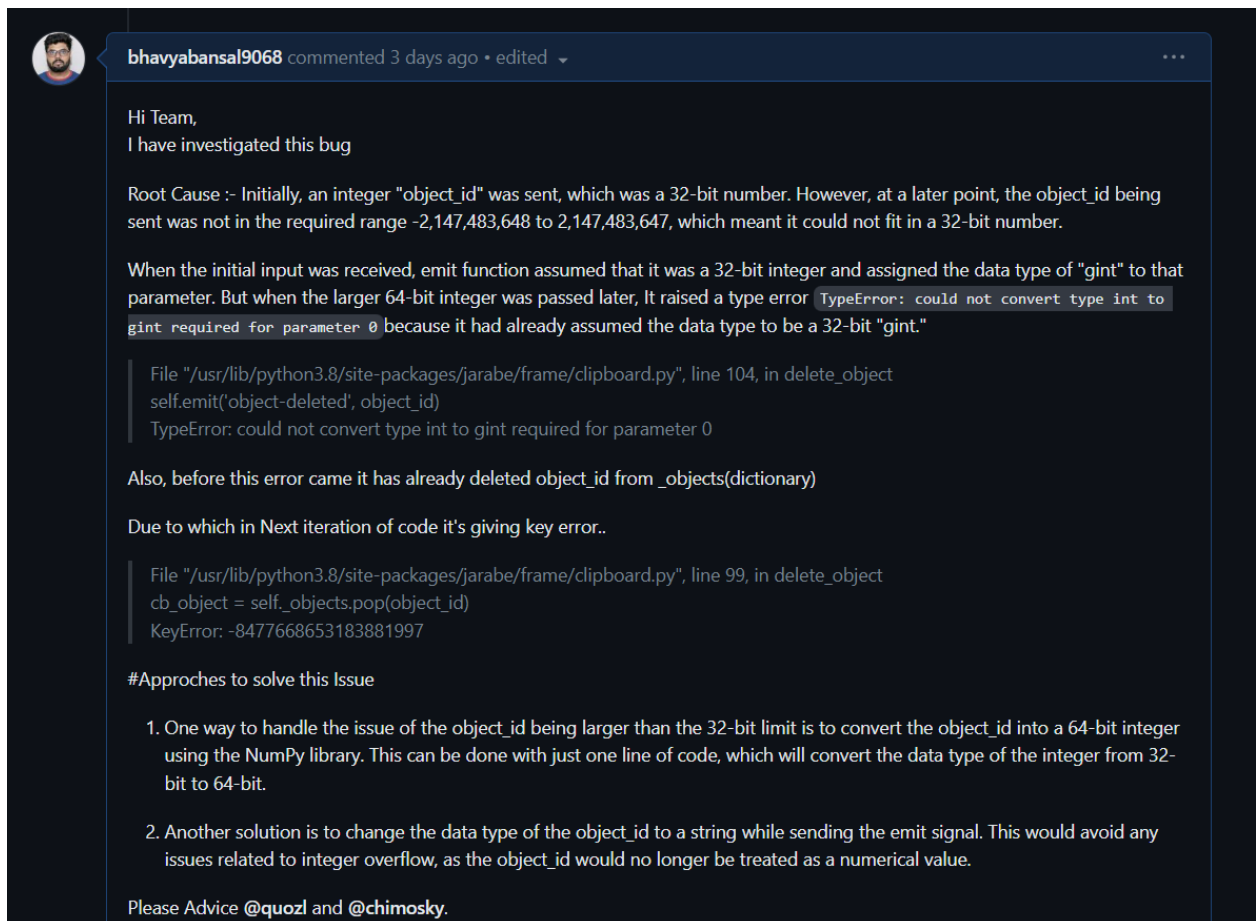

But later on, we came to the conclusion that there was a misunderstanding. This bug was for the X11 environment, not in Wayland. In Wayland, there is a plan to launch it from GNOME Terminal(will plan on it ones i finish porting python and it's dependencies).

2. Then i have taken the Issue "[Text clipping cannot be removed from Frame #856](#)") in this bug i have debugged the root cause reproduced the same in my machine and concluded that int datatype was used as an argument for the object-deleted signal, this meant that the Python interpreter was responsible for deciding whether to use int32 or int64 based on the type of input passed in.

I fixed the issue with minimal change by changing the datatype of the function parameter to int64 using GObject.

Link to Discussion:

(<https://github.com/sugarlabs/sugar/issues/856#issuecomment-1483868582>)



The screenshot shows a GitHub comment from user bhavyabansal9068, posted 3 days ago. The comment discusses a bug related to integer overflow in a Python script. It includes a detailed explanation of the root cause, a code snippet showing a TypeError, and two proposed solutions for handling the issue.

Comment by bhavyabansal9068:

Hi Team,
I have investigated this bug

Root Cause :- Initially, an integer "object_id" was sent, which was a 32-bit number. However, at a later point, the object_id being sent was not in the required range -2,147,483,648 to 2,147,483,647, which meant it could not fit in a 32-bit number.

When the initial input was received, emit function assumed that it was a 32-bit integer and assigned the data type of "gint" to that parameter. But when the larger 64-bit integer was passed later, It raised a type error `TypeError: could not convert type int to gint required for parameter 0` because it had already assumed the data type to be a 32-bit "gint."

```
File "/usr/lib/python3.8/site-packages/jarabe/frame/clipboard.py", line 104, in delete_object
self.emit('object-deleted', object_id)
TypeError: could not convert type int to gint required for parameter 0
```

Also, before this error came it has already deleted object_id from _objects(dictionary)

Due to which in Next iteration of code it's giving key error..

```
File "/usr/lib/python3.8/site-packages/jarabe/frame/clipboard.py", line 99, in delete_object
cb_object = self._objects.pop(object_id)
KeyError: -8477668653183881997
```

#Approches to solve this Issue

1. One way to handle the issue of the object_id being larger than the 32-bit limit is to convert the object_id into a 64-bit integer using the NumPy library. This can be done with just one line of code, which will convert the data type of the integer from 32-bit to 64-bit.
2. Another solution is to change the data type of the object_id to a string while sending the emit signal. This would avoid any issues related to integer overflow, as the object_id would no longer be treated as a numerical value.

Please Advice @quozl and @chimosky.

Link to PR(Merged) :

(<https://github.com/sugarlabs/sugar/pull/967>)

Fix text clipping cannot be removed from Frame Browse files

The int datatype was used as an argument for the object-deleted signal, this meant that the Python interpreter was responsible for deciding whether to use int32 or int64 based on the type of input passed in.

Initially, the interpreter was selecting int32, but when larger input values were used, a TypeError occurred because it could not convert int64 to int32 (gint).

Furthermore, the object_id was deleted when the object-deleted callback got executed leading to a KeyError.

This commit changes the datatype of the function parameter to int64, ensuring that the integer input is stored as int64 regardless of its range.


master

bhavya authored and chimosky committed 3 days ago 1 parent 900e92a commit b898aef

Showing 1 changed file with 1 addition and 1 deletion. Split Unified

```
src/jarabe/frame/clipboard.py
@@ -37,7 +37,7 @@ class Clipboard(GObject.GObject):
37 37     'object-added': (GObject.SignalFlags.RUN_FIRST, None,
38 38                     ([Object])),
39 39     'object-deleted': (GObject.SignalFlags.RUN_FIRST, None,
40 -                       ([int])),
40 +                       ([Object.TYPE_INT64])),
41 41     'object-selected': (GObject.SignalFlags.RUN_FIRST, None,
42 42                        ([int])),
43 43     'object-state-changed': (GObject.SignalFlags.RUN_FIRST, None,
```




This Issue is Successfully Closed by IbiAm:

 **chimosky** commented 19 hours ago Member ...


Tested, works as expected. Thanks!

Merged by hand, please review [b898aef](#) as I edited the commit message.


Also checkout our [contributing-doc](#) as it contains helpful info like working on a different branch before opening a PR.



  1  1

Fix Issue #856 with Emit Function Parameter Datatype during Object Deletion Signal. #967 Closed

 **chimosky** commented 19 hours ago Member ...

Closing as it's been fixed in [b898aef](#).



  **chimosky** closed this as completed 19 hours ago

Timeline

Overview

- Prior to the Community Bonding Period (Present - May 4)
 - Enhance knowledge on TelepathyGlib library
 - Acquire familiarity with Testing Sugar and Activities
 - Continue resolving issues in Sugar
 - Collaborate with Sugar Labs team to receive feedback on my activity and identify areas for improvement
- Community Bonding Period (May 4 - May 28)
 - Communicate regularly with my mentor to discuss my understanding and seek further guidance
 - Work on a list of Activities that needed to be Port.
 - Familiarize myself with the documentation related to porting to Python 3, Measuring Coverage by practicing them.
 - Begin implementing coding practices for porting Sugar to Python 3
 - Use this time as a buffer to discuss any outstanding issues or concerns with my mentor.
- Milestone 1 (May 29 - July 14)
 - Review Sugar source code changes since 0.112 that were made for porting to Python 3
 - Write the test cases and Iterate through them to maximize coverage.
 - Fix Regression issues in Sugar, the Toolkit, and the Datastore related to Porting.
 - Finish Porting the 50%(15) Activities to Python3(includes porting of all dependencies like Telepathy-Python) with testing.

(Would try to finish the maximum work by First milestone)

- Milestone 2 (July 15 - Aug 21)
 - Port all the remaining activities(15) to Python3.
 - Finish Complete testing.
 - Fix other remaining regressions issues.

- Add Proper Error Handling in the codebase.
- Make sure to Add proper doc string to all the functions.
- Finalize the codebase and finish the testing.

□ In-depth

- Week 1-2: (May 29 - Jun 11)
 - Review Sugar source code changes since 0.112 that were made for porting to Python 3
 - Start Designing Test Cases based on the changes identified in the codes.
 - Made myself familiar with collab wrapper library
 - Check for coverage based on the test cases.
 - Begin iterating on tests to achieve sufficient code coverage.
- Week 3-4: (Jun 12 - Jun 25)
 - Fix Regression issues in Sugar, the Toolkit, and the Datastore that might act as a blocker in the Porting process.
 - Initiate the coding for Porting to Python3(include all dependencies i.e., Telepathy-python, GTK, Collabwrapper).
 - Keep on iterating the test cases to improve them.
 - Discuss any doubts with my mentor.
- Week 5-6: (Jun 26 - Jul 09)
 - Test the sugar with the new changes for the porting i have did
 - Discuss and review them to mentor and would update according to the suggestions.
 - Review the PR for the changes.
- Week 7: (Jul 10 - Jul 14)
 - Submit mid term Evaluation and proceed further.
 - Receive Feedback based on my work and improve myself accordingly.
- Week 8-9: (Jul 15 - Jul 30)
 - Port remaining activities to Python3.
 - Test the sugar with the changes.
- Week 10-11: (Jul 31- Aug 13)

- Finalize any outstanding issues with porting activities to Python 3.
 - Might raise the PR after testing will discuss this with my mentor.
 - Beside this, improve the codebase by adding basic validation check.
 - This Validation check(Try and Except Block) would prevent further regression.
 - Also Add the doc strings to the function by discussing it with mentors.
- Week 12: (Aug 14 - Aug 21)
 - Update documentation and Finish testing the same.
 - Raise the Final PR
 - Submit final project deliverables

How Many Hours will you spend each week on your Project?

This project appears to be quite complex, and there is a possibility of uncovering various hidden elements during the implementation process. Therefore, I am committed to working to the best of my ability, and initially, I will be dedicating 6-8 hours a day and a total of 36-48 hours per week (including Saturdays) to this project. I understand that this project demands dedication, hard work, and consistent effort to meet the required standards. I assure you that I will give my best under the guidance of my mentors.

How will you report progress between evaluations ?

To report progress between evaluations, I will keep a regular check on the project milestones and deliver them on time. I will maintain proper documentation of the work completed, including code commits, pull requests, and resolved issues, which will be visible to my mentor. I will frequently communicate with my mentor through email, chat, or calls to update them on my progress and seek feedback on my work. In case of any issues or delays, I will inform my mentor as soon as possible and discuss strategies to resolve them. Moreover, to ensure consistency, I will send an end-of-day report to my mentor on a daily basis, as I used to do in Telaverge. This will ensure transparency and regular communication throughout the project, keeping my mentor informed about the progress and ensuring timely completion of the assigned tasks.

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

I am eager to make further contributions to Sugar Labs. I already have a few ideas in mind that I plan to work on, such as supporting Sugar from GNOME Terminal in Wayland, which was mentioned as a requirement by James ([Link](#)). I am committed to staying connected with Sugar Labs, as it provides a valuable opportunity for me to improve my skills. Additionally, I believe that my work will benefit organizations and schools that rely on Sugar Labs for high-quality education delivery.

Why me?

I am excited to apply for this project as I believe that my skills, experience, and enthusiasm make me a perfect fit. As a software developer who is passionate about Python, I was drawn to this project as it aligns perfectly with my technology stack.

After researching this project, I knew that I wanted to be a part of it and contribute to its success. As a beginner in the open source community, I am eager to learn and grow through this experience. If I get this opportunity, I am committed to take this project seriously and contribute to the community by maintaining it in the future.

My academic background in computer science and my cumulative percentile index of 8.4 demonstrates my proficiency in core foundational concepts. Furthermore, my experience in programming and Python, as well as my ability to independently solve and debug complex issues, make me an ideal candidate for this project.

I have previously worked on a project called [Regal](#), where I gained valuable experience in working in real-world situations and maintaining projects. I believe that this experience, combined with my technical skills, makes me the best choice for this project.

For me, this project holds significant value in terms of setting up a foundation for my career in open source. Therefore, I am fully committed to making the most of this opportunity and putting in all my efforts to excel within the given period. I am confident in myself, my skills, and my enthusiasm, and I am eager to contribute to the success of this project.

Thank you