Project Report Assignment 4 Group 2

Name: Toga

URL: https://github.com/beeware/toga

BeeWare Toga is a cross-platform GUI toolkit for Python that allows developers to create native desktop and mobile applications using a single codebase.

Onboarding experience

Did you choose a new project or continue on the previous one?
We chose a new repository because we thought that the previous one would be too difficult to actually resolve any issues on.

If you changed the project, how did your experience differ from before?
The new repo has a well documented page on how to contribute to the repo and how to set up the environment, which was quite easy. Some group members had some trouble running the testbed however. For some reason, it did not work on Ubuntu with a Miniconda environment. However, when running it in a Conda environment with PowerShell, it worked. Although the issues we had starting there was some issues with the environment thanks to the documentation it was easier to onboard

Effort spent

Date	Group member	Time (hours)	Activity
2025-02-24	Whole group	1.5	Found a repository and an issue. Set up the environment.
2025-02-26	Whole group	1.5	Familiarized ourselves with the repository structure, located the functions and tests, documented the onboarding experience, and logged the tests related to the issue. Began identifying some requirements.

2025-02-28	Whole group	1.5	Identified all the requirements, wrote the project plan and familiarized ourselves some more with the methods.
2025-02-28	Roger	1	Looked at some tests from previous issues and how they were tackled to form some ideas for testing our implementations.
2025-03-02	Jacob	2.5	Familiarized myself with the basic functionality of textinput.py and multilinetextinput.py, as well as how to use Apple's UIKit. Then implemented the ability to disable auto correction for iOS.
2025-03-03	Whole group	1	Found another issue
2025-03-03	Victoria	4	Looked through their test system, the backend tests, core tests and testbed to get a better understanding of how they construct their tests. unsuccessful in making the testbed run.
2025-03-04	Whole group	2	Went through our own code, and ran the briefcase tests
2025-03-04	Roger	4	Created test for iOS autocorrect implementation. This took very long as there were a lot of uncertainties about how tests were structured in the project.
2025-03-04	Joline	1	Familiarized with

			Androuid files where code whas suppose to be implemented and started with code structure
2025-03-04	Jacob	1	Implemented the option to disable spell checking and auto capitalization for IOS
2025-03-04	Victoria	6	Trying to make the testbed run again on Windows Ubuntu. Went through the beeware tutorial but it started to crash when trying to implement the code on Android. Wrote tests for text input in core and then moved the tests to the android directory instead.
2025-03-05	Joline	3	Completed code for Spellcheck and Autocorrect in multiline and textinput
2025-03-05	Whole group	0.5	Made up a plan on what we should do next
2025-03-05	Roger	1	Created tests for disable spell check, auto capitalization and inline prediction on iOS. These were a lot quicker to implement as opposed to the first time writing tests.
2025-03-05	Jacob	3	Fixed some minor errors with spell checking and auto capitalization for IOS. Implemented the option to turn off inline prediction. Started writing on the report.

2025-03-05	Victoria	1.5	The testbed can’t be run on Ubuntu so installed anaconda terminal on windows which ran the testbed without problems…
2025-03-05	Joline	2	Further look through of connected functions to and looking at another issue to maybe start
2025-03-06	Whole group	2	Last sync together before pull request for first issue. Plan for starting on a new issue even if unlikely we’ll finish in time. Realized we had misinterpreted the code structure.
2025-03-06	Roger	1.5	Moved tests, as we had them in testbed which was mainly GUIs rather than individual unit tests. Really though this is the culmination of the mistakes we made and the general structure for the final tests now that the group as a whole have figured out how the repository is laid out.
2025-03-06	Jacob	1.5	Fixed so that the methods for disable spellcheck and autocorrect follow the same format as for android.
2025-03-06	Victoria	0.5	After we figured out more about the general structure the tests had to be moved and rewritten.
2025-03-06	Joline	4 hours	Wrote Core functions to connect IOS and

			Android to core
2025-03-07	Whole group	1 hour	Last sync together before pull request for the issue.
2025-03-07	Roger	2 hours	Added UML and other small changes to report. Quick bug fix on tests.
2025-03-07	Jacob	2.5 hour	Report and wrote a change note.
2025-03-07	Victoria	1 hours	Report
2025-03-07	Joline	2 hours	Fixed errors in core as well as in android texinput so there were no errors in test suit

Overview of issue(s) and work done.

Title: Add API to control spell checkers and other platform-provided text input assistants
URL: https://github.com/beeware/toga/issues/2805

Summary
The request is for a way to disable spell check in Toga’s TextInput and MultilineTextInput, particularly for cases where red underlines and corrections are unnecessary or disruptive. A proposed solution is adding a spell_checking=False flag, with consideration for other input-related features like autocorrect and autocomplete.

Scope (functionality and code affected).
The issue affects text input fields in Toga’s *TextInput* and *MultilineTextInput* components on mobile platforms, Android and IOS to be more precise, where spell check is enabled by default. The affected code include the files “textinput.py” and “multilinetextinput.py” in IOS/src\toga_IOS/widgets for IOS, and “textinput.py” in android/src\toga_android/widgets for android. Since textInput is a parent class to multilineTextInput for android, only changes in “textInput.py” is necessary. Additional considerations may include disabling other input-related features like autocorrect and predictive text in these files as well.

Requirements for the new feature or requirements affected by functionality being refactored

1. Add new parameters to the TextInput and MultilineTextInput constructor.
    - Introduce new boolean variables which saves the current settings of autocorrect and spell check.
2. Add setters and getters to the TextInput and MultilineTextInput constructor
    - Implement setters which enables or disables autocorrection and spell check, and getters which returns the current setting.
3. Implement support for Android and IOS
    - Android: InputType has a couple of options that can be disabled.
    - IOS: spellCheckingType and autoCorrectionType are provided by IOS, amongst other options, to disable these settings.
4. Update tests
    - Add unit tests to verify that the setters effectively disables spell checking.

Code changes

ios/IOS/src\toga_IOS/libs/uikit.py
Added the classes “UITextAutocorrectionType”, “UITextSpellCheckingType”, “UITextAutocapitalizationType” and “UITextInlinePredictionType” which are enums that represents different input settings for IOS.

ios/IOS/src\toga_IOS/libs/textinput.py
Added getters and setters for autocorrect and spell checkers. Which turns off and on autocorrect and spell check. Also implemented setters for auto capitalization and inline prediction.

ios/IOS/src\toga_IOS/libs/multilinetextinput.py
Added getters and setters for autocorrect and spell checkers. Which turns off and on autocorrect and spell check. Also implemented setters for auto capitalization and inline prediction.

android/src/toga_android/widgets/textinput.py
Added getters and setters for autocorrect and spell checkers. Which turns off and on autocorrect and spell check.

android/src/toga_android/widgets/multilinetextinput.py
Added functions that inherited from textinput.py for autocorrect and spell check.

core/src/toga/widgets/textinput.py
Added property functions to be able to utilize the get and set functions for android and IOS. to be able to change the values for spellcheck and autocorrect in the widget

core/src/toga/widgets/multilinetextinput.py
Did the same things as we did for textinput.py

core/tests/widgets/test_textinput.py
The tests test_spellcheck() and test_autocorrect were implemented.

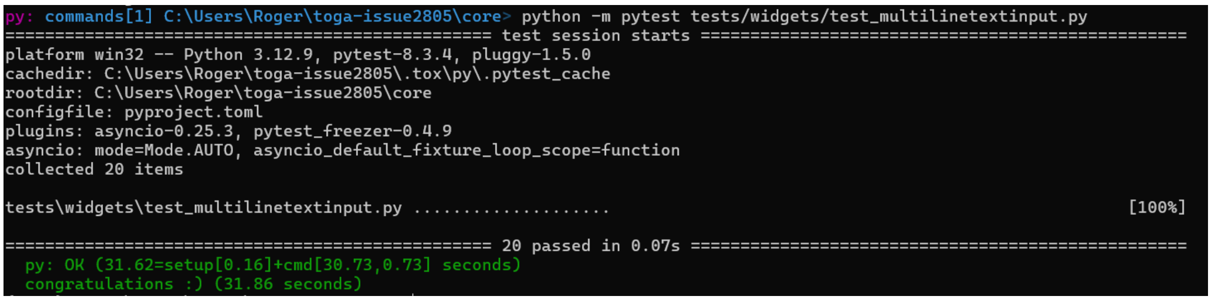
core/tests/widgets/test_multilinetextinput.py
The same tests were implemented in multiline text input.

Patch

github.com_beeware_toga_compare_main...Rocygel_toga-issue2805_main.png

Test results

Before:
multiline textinput:

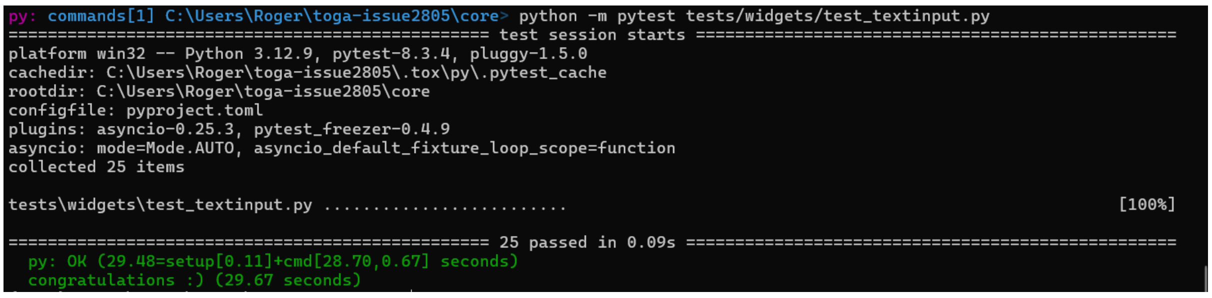


```
py: commands[1] C:\Users\Roger\toga-issue2805\core> python -m pytest tests/widgets/test_multilinetextinput.py
================================================= test session starts =================================================
platform win32 -- Python 3.12.9, pytest-8.3.4, pluggy-1.5.0
cachedir: C:\Users\Roger\toga-issue2805\.tox\py\.pytest_cache
rootdir: C:\Users\Roger\toga-issue2805\core
configfile: pyproject.toml
plugins: asyncio-0.25.3, pytest_freezer-0.4.9
asyncio: mode=Mode.AUTO, asyncio_default_fixture_loop_scope=function
collected 20 items

tests\widgets\test_multilinetextinput.py ....................                                                  [100%]

================================================= 20 passed in 0.07s ==================================================
  py: OK (31.62=setup[0.16]+cmd[30.73,0.73] seconds)
  congratulations :) (31.86 seconds)
```

textinput:

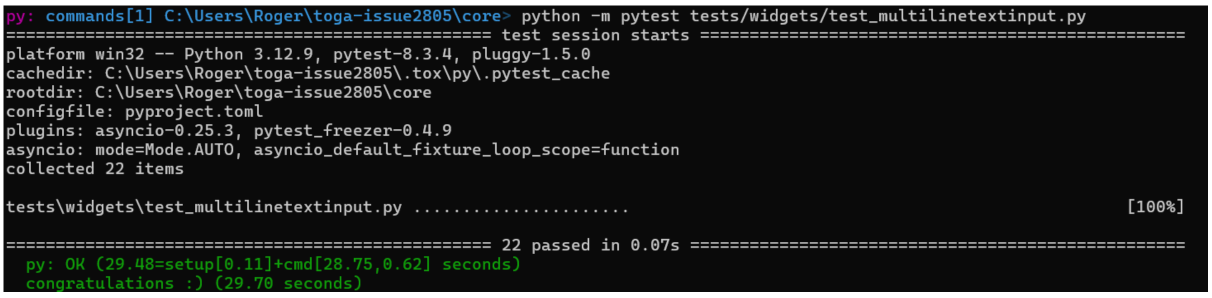


```
py: commands[1] C:\Users\Roger\toga-issue2805\core> python -m pytest tests/widgets/test_textinput.py
================================================= test session starts =================================================
platform win32 -- Python 3.12.9, pytest-8.3.4, pluggy-1.5.0
cachedir: C:\Users\Roger\toga-issue2805\.tox\py\.pytest_cache
rootdir: C:\Users\Roger\toga-issue2805\core
configfile: pyproject.toml
plugins: asyncio-0.25.3, pytest_freezer-0.4.9
asyncio: mode=Mode.AUTO, asyncio_default_fixture_loop_scope=function
collected 25 items

tests\widgets\test_textinput.py .........................                                                     [100%]

================================================= 25 passed in 0.09s ==================================================
  py: OK (29.48=setup[0.11]+cmd[28.70,0.67] seconds)
  congratulations :) (29.67 seconds)
```

After implementing our solution.
multiline textinput:

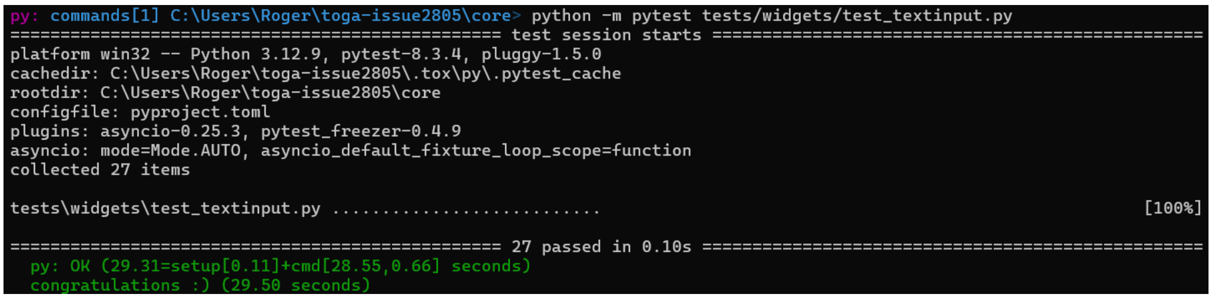


```
py: commands[1] C:\Users\Roger\toga-issue2805\core> python -m pytest tests/widgets/test_multilinetextinput.py
================================================= test session starts =================================================
platform win32 -- Python 3.12.9, pytest-8.3.4, pluggy-1.5.0
cachedir: C:\Users\Roger\toga-issue2805\.tox\py\.pytest_cache
rootdir: C:\Users\Roger\toga-issue2805\core
configfile: pyproject.toml
plugins: asyncio-0.25.3, pytest_freezer-0.4.9
asyncio: mode=Mode.AUTO, asyncio_default_fixture_loop_scope=function
collected 22 items

tests\widgets\test_multilinetextinput.py ......................                                                [100%]

================================================= 22 passed in 0.07s ==================================================
  py: OK (29.48=setup[0.11]+cmd[28.75,0.62] seconds)
  congratulations :) (29.70 seconds)
```

textinput:



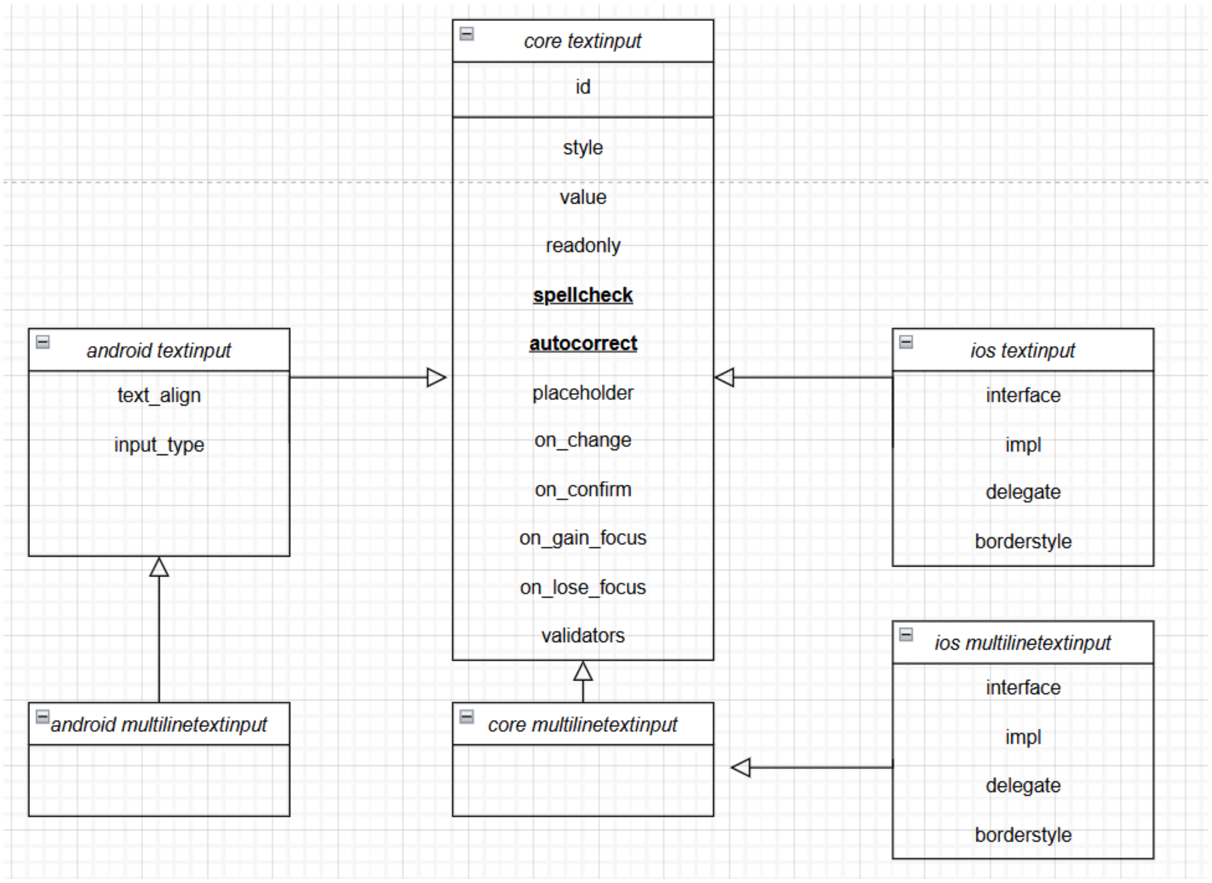
```
py: commands[1] C:\Users\Roger\toga-issue2805\core> python -m pytest tests/widgets/test_textinput.py
================================================ test session starts =================================================
platform win32 -- Python 3.12.9, pytest-8.3.4, pluggy-1.5.0
cachedir: C:\Users\Roger\toga-issue2805\.tox\py\.pytest_cache
rootdir: C:\Users\Roger\toga-issue2805\core
configfile: pyproject.toml
plugins: asyncio-0.25.3, pytest_freezer-0.4.9
asyncio: mode=Mode.AUTO, asyncio_default_fixture_loop_scope=function
collected 27 items

tests\widgets\test_textinput.py ...........................                                                     [100%]

================================================ 27 passed in 0.10s =================================================
  py: OK (29.31=setup[0.11]+cmd[28.55,0.66] seconds)
  congratulations :) (29.50 seconds)
```

UML class diagram and its description

Key changes/classes affected



There are two methods affected: textinput() and multilinetextinput() edited for iOS and Android. We added the spellcheck and autocorrect methods in the “class files” in core, iOS and Android. The multilinetextinput in core inherited essentially everything from the simple textinput one. Both Android and iOS textinput inherited from the core textinput. However, for Android the multilinetextinput inherited from its own textinput whereas iOS multiline was essentially a duplicate of the iOS textinput but inheriting from the multilinetextinput in core rather than the textinput one.

Overall experience

What are your main take-aways from this project? What did you learn?

Toga's structured testing allows core tests to run across multiple platforms. We wrote separate code for iOS and Android but faced integration issues when our naming conventions did not line up.

Onboarding and learning the code structure and so on takes longer time than first expected. Reading the documentation in detail is also helpful to speed up this process, which we at first did not do. So in the future we aim at trying to understand the project's structure and its documentation before attempting to solve one of its problems.

How did you grow as a team, using the Essence standard to evaluate yourself?

We have now reached the "Working Well" phase in the Essence standard. Switching repositories to Toga and tackling the challenge of disabling autocorrect and spell check naturally increased our collaboration. Regular group meetings—held nearly every day—helped us stay aligned on tasks, share progress, and resolve issues efficiently.

When someone encountered a problem and found a solution, they proactively shared it with the team to prevent others from getting stuck. Given the complexity of the repository structure, discussing it together was crucial in improving our understanding and workflow. These improvements in communication and teamwork have made us a more cohesive and efficient team.

Other things

In IOS we implemented the option to set auto capitalization and inline prediction, as well (we did not find that IOS supported these options). But we did not have the time to add this to the core and test it. But we had it in our pull request so someone else can continue with it.