**ASSIGNMENT COVER SHEET**

|  |  |  |
| --- | --- | --- |
| **Student’s name** | Lucas | Shaw |
| **Module name** | Software engineering and agile | |
| **Title of assignment** | Task 2 report Agile Overview | |
| **Complete Word Count in my assignment** | 550 | |
| **Date submitted** | 27 June 2025 | |

All work must be submitted by the due date. If an extension of time to submit work is required, a [Mitigating Circumstances Extension Form](https://canvas.qa.com/courses/1041) must be submitted.

<https://canvas.qa.com/courses/1041>

Has an extension been approved? Yes No If yes, please give the new submission date ….…/..…./…….

|  |
| --- |
| IMPORTANT: THIS STATEMENT MUST BE READ & SIGNED  **Academic Integrity Statement**  Academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework that is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly.   ​  **I declare that the work I am submitting is my work, is properly referenced and has not been submitted elsewhere.** |
| **Student Signature (Full Name): Lucas Shaw**  **Date: 27 June 2025** |

Table of Contents

[Collaborative development 3](#_Toc201910956)

[Class Based Views 3](#_Toc201910957)

[Test Driven Development (TDD) 4](#_Toc201910958)

[Appendix 5](#_Toc201910959)

# Collaborative development

The project uses Agile methods by focusing on incremental development using Git technology. As a sole developer, collaborative peer reviews weren’t relevant. But Git branches compartmentalise the functionality of the application. Allowing for multiple features to be worked on simultaneously in isolation, reduces conflicts when integrating (Gowda, P, 2022).

Git diff and pull requests are used for proofreading and visually displaying all changes. An example branch is add-bootstrap. Isolation of this design work separated styling and backend work, ensuring logic-based features would not be interfered with. This follows the Agile methodology by delivering adaptable functionality in isolation.

Branched workflows can showcase prototypes with isolation of functionality (Appendix, figure 1). Allowing faster feedback and approval of new concepts, reducing instances of wasted work when requirements change. Regular demonstrations can ensure alignment with end-user expectations (Atlassian, n.d). Technical implementation requires a workflow\_dispatch to run a specific branch of work (GitHub Docs, 2024).

A future workflow improvement is continuous integration (CI). A CI pipeline would automate tests on each commit. Even as a sole developer this practice forms smoother collaboration and increases the reliability of the project and development and the codebase grows (Crudu and MoldStud Research Team, 2024)*.*

# Class Based Views

Implementing Django’s class-based views (CBVs) facilitates reusable code, faster development and better organisation of functionality. This aligns with Agile development principles (Horilla Editorial Team, 2024).

CBV encapsulates logic for HTTP methods such as GET, POST, PATCH and DELETE requests, for create, read, update and delete (CRUD) requirements. This can be reused for faster reliable development (AlOmar et al., 2021).

CBVs enable better extensibility and maintainability to support an Agile adaptive workflow (Agarwal & Majumdar, 2013). For example, during late-stage development, non-admin user restriction for the user index page was implemented by simply adding SuperUserRequiredMixin (Appendix, figure 2) to the UserIndexView parameters. Another example, quickly adapting default class behaviour is simple. In this project, configuring bug date order only required overwriting BugListView’s get\_queryset (Appendix, figure 3) method (Django Software Foundation, 2025).

CBV reduces code bloat and improves readability due to the separation of concerns. The model, templates and URLs have specific roles defined in each Class. For example, BugListView inherits from Django’s ListView, to access all instances of the Bug model. No additional logic is needed.

# Test Driven Development (TDD)

Writing tests before code implementation ensures the business requirements are defined (James. C, 2024). Ideal in Agile workflows, as the test will redefine the business's requirements, so the implementation code must satisfy this before a production release is possible.

Tests first can also highlight missed specification issues and pre-existing coding oversights when integrating into the existing application. The user story may require a change to the code that breaches the scope of what is possible in a sprint.

TDD in this project was used to ensure normal users could not delete a bug. If the business required a change, the test would reflect this first. The test expectations ensure a minimal amount of application code, which prevents feature creep and focuses development. This speeds up adaptation by having to change less code (Conformiq, 2023).

Test expectations also perform a documentation role (Siniaalto & Abrahamsson, 2017). Useful for retrospectively understanding the code's purpose and for collaboration, speeding up the response to change. TDD’s red-green-refactor process supports iterative design and efficient coding.

# Appendix

figure 1

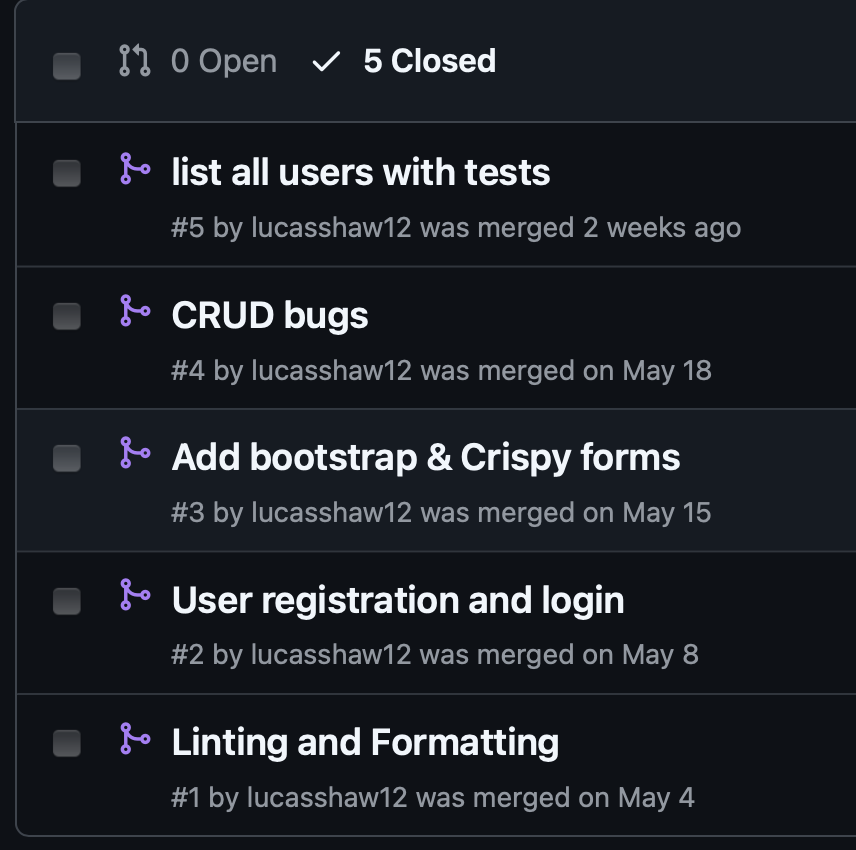
https://github.com/lucasshaw12/bug\_tracker/pulls?q=is:pr+is:closed

figure 2

https://github.com/lucasshaw12/bug\_tracker/blob/6fe6479cc2dae4dda4cb1525b99fd17e9604bc0a/accounts/mixins/custom\_mixins.py#L5

from django.contrib.auth.mixins import UserPassesTestMixin

from django.contrib.auth.views import redirect\_to\_login

class SuperUserRequiredMixin(UserPassesTestMixin):

def test\_func(self):

return self.request.user.is\_superuser

def handle\_no\_permission(self):

return redirect\_to\_login(self.request.get\_full\_path())

figure 3

https://github.com/lucasshaw12/bug\_tracker/blob/6fe6479cc2dae4dda4cb1525b99fd17e9604bc0a/bug\_tracker/views.py#L13

class BugListView(LoginRequiredMixin, ListView):

model = Bug

template\_name = "dashboard.html"

login\_url = reverse\_lazy("login")

def get\_queryset(self):

return Bug.objects.order\_by("date\_raised")

References

Ashwath Narayana Gowda, Priyanka Gowda. (2022). Git Branching and Release Strategies. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences. 10. 10.5281/zenodo.14221771. [Accessed 16 June 2025].

Atlassian. (n.d.) *Sprint Demo: What it is & How to Conduct One*. [online] Available at: Atlassian Agile Project Management website [atlassian.com+8atlassian.com+8community.atlassian.com+8](https://www.atlassian.com/agile/project-management/sprint-demo?utm_source=chatgpt.com) [Accessed 20 June 2025].

GitHub Docs, 2024. *Events that trigger workflows*. [online] GitHub. Available at: [https://docs.github.com/en/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows#workflow\_dispatch](https://docs.github.com/en/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows%23workflow_dispatch) [Accessed 22 Jun. 2025].

Crudu, A. & MoldStud Research Team, 2024. *The benefits of continuous integration and deployment in software development*. MoldStud. [online] 16 March. Available at: https://moldstud.com/articles/p-the-benefits-of-continuous-integration-and-deployment-in-software-development [Accessed 10 June 2025].

Horilla Editorial Team, 2024. *What Are Django ClassBased Views (CBVs) & its Advantages in 2024*. Horilla. [online] 4 July. Available at: <https://www.horilla.com/blogs/what-are-django-class-based-views-cbvs-and-its-advantages/> [Accessed 2 June 2025].

AlOmar, E.A., Wang, T., Raut, V., Mkaouer, M.W., Newman, C. & Ouni, A., 2021. *Refactoring for reuse: an empirical study*. Innovations in Systems and Software Engineering, 18(9), pp.1–31. [online] Available at: <https://researchgate.net/publication/357859607_Refactoring_for_reuse_an_empirical_study> [Accessed 4 June 2025].

Agarwal, M. & Majumdar, R., 2013. *Software maintainability and usability in agile environment*. International Journal of Computer Applications, 68(4), pp.30–36. [online] Available at: <https://research.ijcaonline.org/volume68/number4/pxc3886873.pdf> [Accessed 4 June 2025].

Django Software Foundation, 2025. *Using mixins with class-based views*. In *Django 5.2 documentation*. [online] Available at: <https://docs.djangoproject.com/en/5.2/topics/class-based-views/mixins/> [Accessed 4 June 2025].

James, C., 2024. *Impact of TestDriven Development (TDD) on Code Quality and Maintainability in Agile Teams*. [online] Available at: <https://researchgate.net/publication/385271226_Impact_of_Test-Driven_Development_TDD_on_Code_Quality_and_Maintainability_in_Agile_Teams> [Accessed 11 June 2025].

Conformiq, 2023. *The Six Benefits of TestDriven Development*. [online] 16 May. Available at: <https://www.conformiq.com/resources/blog-the-six-benefits-of-test-driven-development-05-16-2023> [Accessed 11 June 2025].

Siniaalto, M. & Abrahamsson, P., 2017. *A comparative case study on the impact of testdriven development on program design and test coverage*. arXiv preprint arXiv:1711.05082. [online] Available at: <https://arxiv.org/pdf/1711.05082> [Accessed 12 June 2025].