# Bug Tracker Web Application Report

This Django-based web application is built using Django, with the purpose of logging and managing bugs for a software development team, developed with an agile workflow.

## Problem Statement & Scope

Software development teams need to track and manage application bugs on a simple lightweight platform, which this application resolves by offering a simplified internal system. It supports the needs of developers, analysts, designers, and managers within an organization by providing the functionality to report, assignment and complete software application bugs.

The scope includes secure user registration, login, and logout. Bug creation, editing, completion, and closure workflows. Role-based access to restrict certain features. Dashboard views filtered by date.

## System Design

The project uses Django’s ORM for all functionality. The application uses two primary models (Appendix, figure 8):

CustomUser: Inherits from Django’s AbstractUser and includes additional fields such as user\_role, team\_name, and num\_bugs\_assigned.

Bug: Collects bug details including the title, description, expected vs actual behaviour, assigned user, complexity, severity, affected application and status flags.

A one-to-many relationship exists between CustomUser and Bug. Each user can be responsible for multiple bugs. A supporting ERD diagram shows this.

## Authentication, Access Control & Views

The application uses Django’s authentication which is configured further using custom views and forms. Token verification for signup and sessions prevents unauthorised access. Users are redirected to the dashboard after a successful login where they can manage their assigned bugs.

To protect sensitive views:

LoginRequiredMixin is applied to all bug-related views to ensure session-based access control.

SuperUserRequiredMixin (a custom mixin) further restricts deletion and access to the user list, even through direct URL input.

Navigation content dynamically changes based on user role. Superusers are shown additional links such as “Users” and “Delete Bug”, whereas normal users are restricted to basic create, read and update functionality. A non-interactive dropdown item in the navbar displays the user’s superuser status for transparency.

## Key Functional Features

Bug states include “Not started”, “In progress”, “Blocked”, “Under Peer Review”, “Fixed”, and “Closed without fix”, each with visual cues.

The Bootstrap framework is used for a screen responsive and consistent design appearence. With Crispy Forms form appearance and modification.

## Testing & Code Quality

The application includes full testing for login access, permissions and view rendering. Unit tests assert conditions like redirection for unauthorised users, visibility of form elements, all CRUD functionality and status transitions for bugs.

The codebase is formatted using Black for Python which conforms to PEP 8 standards. Djlint for HTML linting. A pre-commit hook ensures conformity to these standards before every commit, to maintain code quality and consistency across contributors.

## Conclusion

This bug tracker application combines class-based Django components, role-based access, and agile development practices into a functional and intuitive tool. It adheres well with agile development principles and supports scalability through its custom user model and component design. The application is thoroughly tested, easy to maintain, and user-friendly.

# Screenshot 2025-06-14 at 13.30.29.pngScreenshot 2025-06-14 at 13.30.22.pngAppendix

Figure 1 Delete bug confirmation screen

Figure 2 - Create bug page

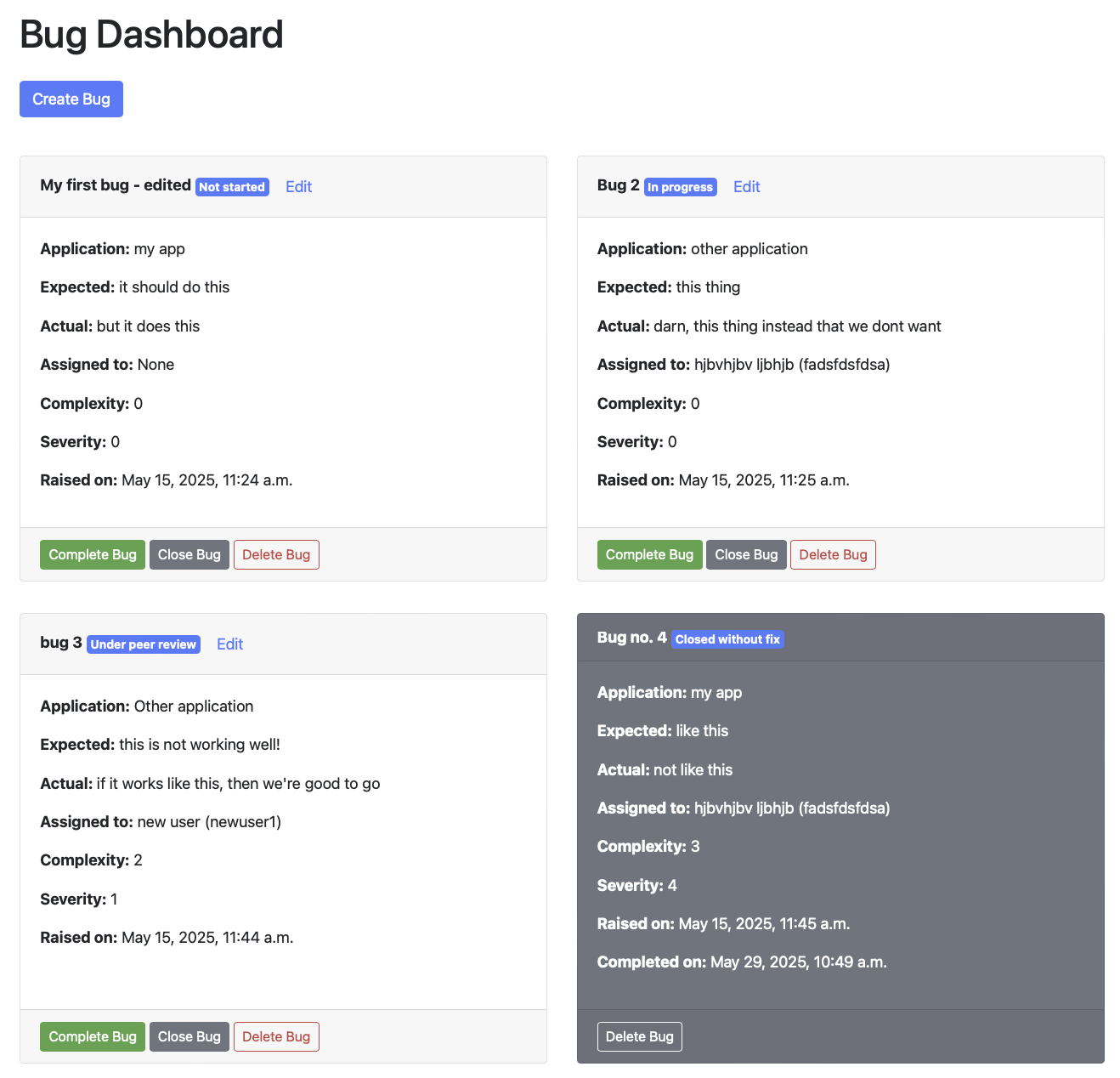
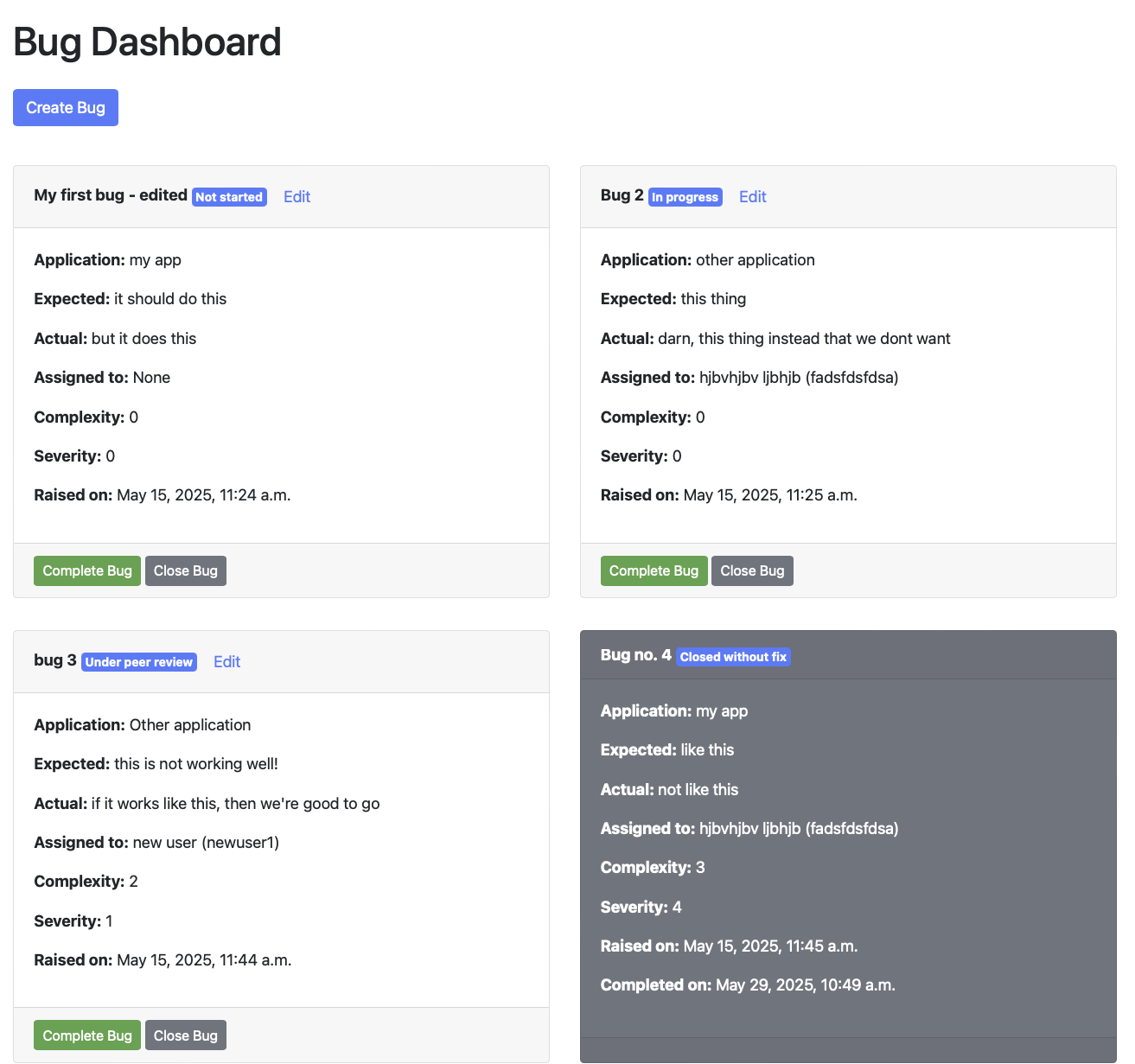


Figure 3 - Normal user dashboard

Figure 4 - Admin user dashboard

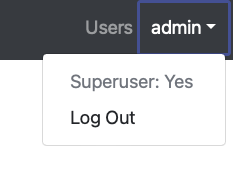
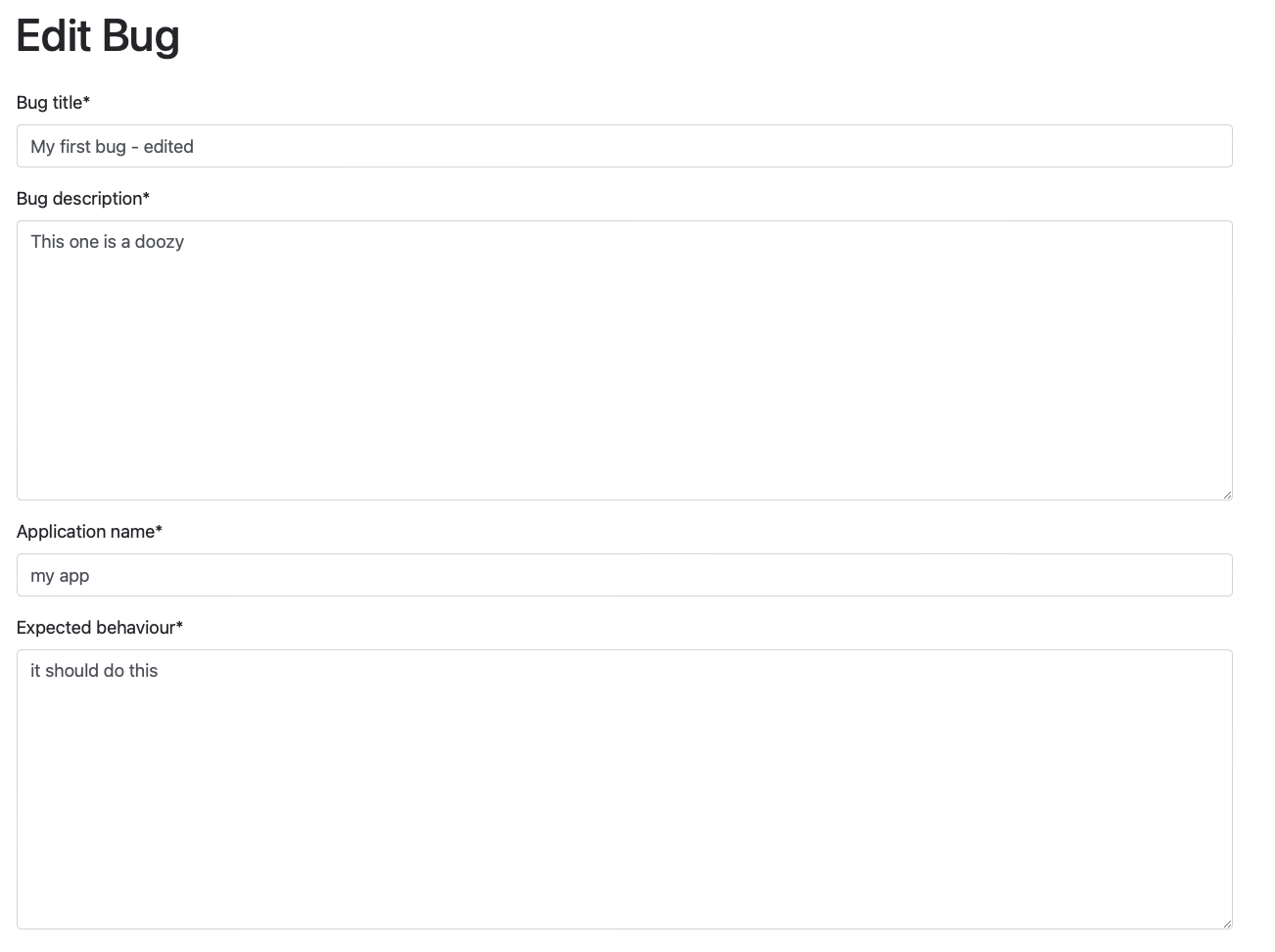


Figure 5- Edit bug screen

Figure 6 - Admin user navbar

A screenshot of a computer

Description automatically generated

Figure 7 Normal user navbar

A screenshot of a computer

Description automatically generated

Figure 8 Entity Relationship Diagram for CustomUser and Bug Models