# Bug Tracker Web Application Report

This project delivers a Django-based web application built using Python 3, aimed at tracking and managing software bugs in a team setting. Developed with an agile mindset, it incorporates secure authentication, dynamic UI behaviour, and an extensible data model suitable for use within software development teams of various disciplines.

## Problem Statement & Scope

Software teams frequently need lightweight tools to track and manage application bugs without the overhead of complex external platforms. This application addresses that gap by offering a simplified, internal bug tracking system. It supports the needs of developers, analysts, designers, and managers within an organisation, facilitating bug reporting, assignment, and completion tracking.

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 completion status and date.

## System Design

The project uses Django’s ORM for model definitions and CRUD operations. The application revolves around two primary models:

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

Bug: Captures key attributes of issues raised including title, description, expected vs actual behaviour, assigned user, complexity, severity, and status flags.

A one-to-many relationship exists between CustomUser and Bug, allowing each user to be responsible for multiple bugs. A supporting ERD (provided separately) outlines these entities and their relationships clearly.

## Authentication, Access Control & Views

The application incorporates Django’s built-in authentication mechanisms, extended via custom views and forms. Sign-up requires token verification to prevent unauthorised access. Once logged in, users are redirected to the dashboard 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 user management pages, preventing access even through direct URL manipulation.

Navigation dynamically adjusts based on user role. Superusers are shown additional links such as “Users” and “Delete Bug”, whereas standard users are restricted to basic CRUD 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”, “Fixed”, and “Closed without fix”, each with visual cues.

Bootstrap is used to ensure responsive design, mobile-friendliness, and professional UI styling.

Crispy Forms enhances the look and layout of user and bug forms, ensuring accessibility and consistency.

## Testing & Code Quality

The application includes extensive automated testing across key functionality such as login access, permissions, and view rendering. Unit tests assert conditions like redirection for unauthorised users, visibility of form elements, and state transitions for bugs.

The codebase is formatted using Black for Python and Djlint for HTML. A pre-commit hook ensures that code formatting standards are enforced before every commit, maintaining code quality and consistency across contributors.

## Conclusion

This bug tracker application combines clean architecture, role-based security, and modern development practices into a functional internal tool. It aligns well with agile development principles and supports scalability through its custom user model and modular design. The application is thoroughly tested, easy to maintain, and user-friendly—ready for adoption in a real-world development environment.

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

Figure 2 Delete bug confirmation screen

Figure 1 - Create bug page

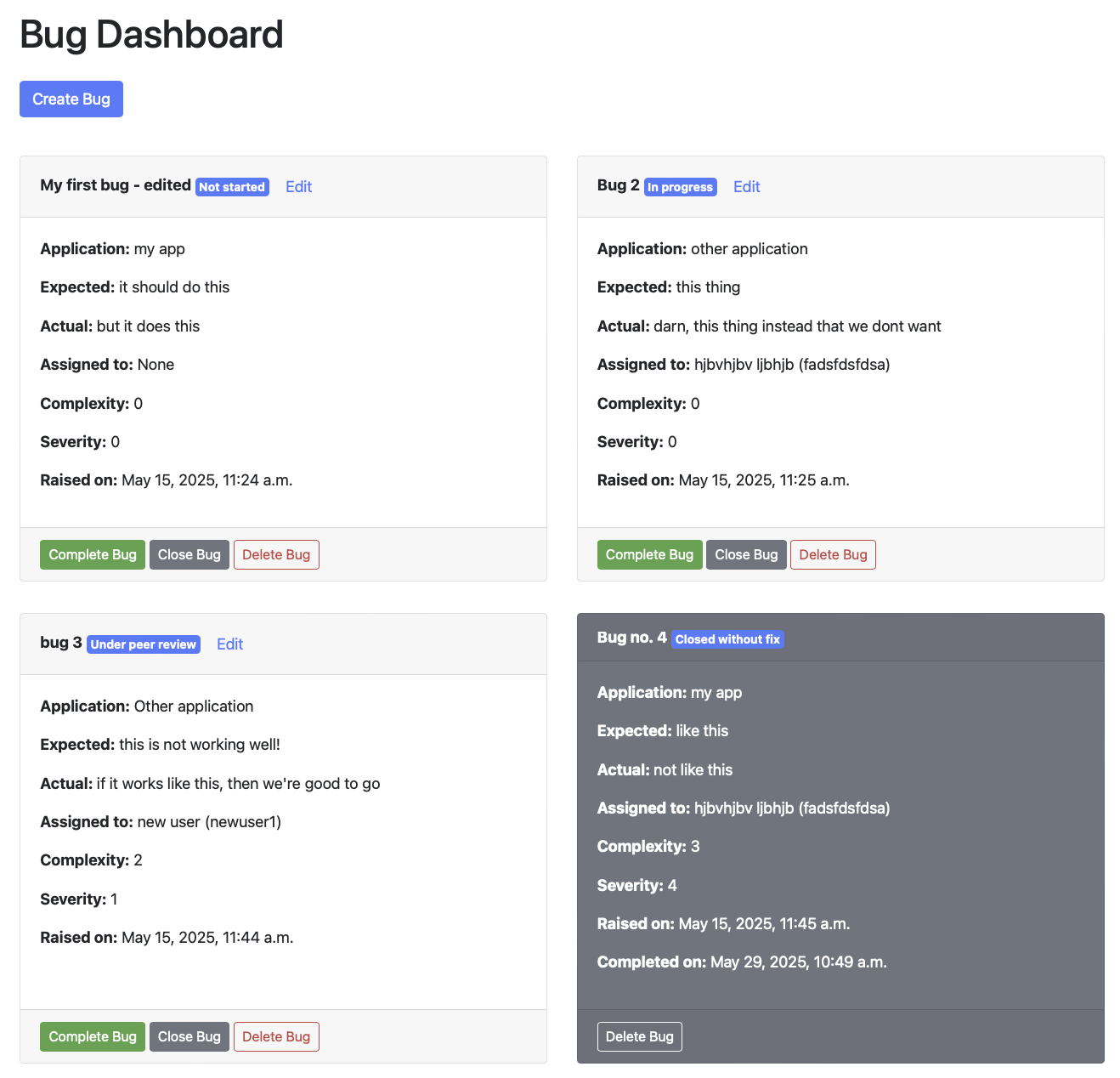
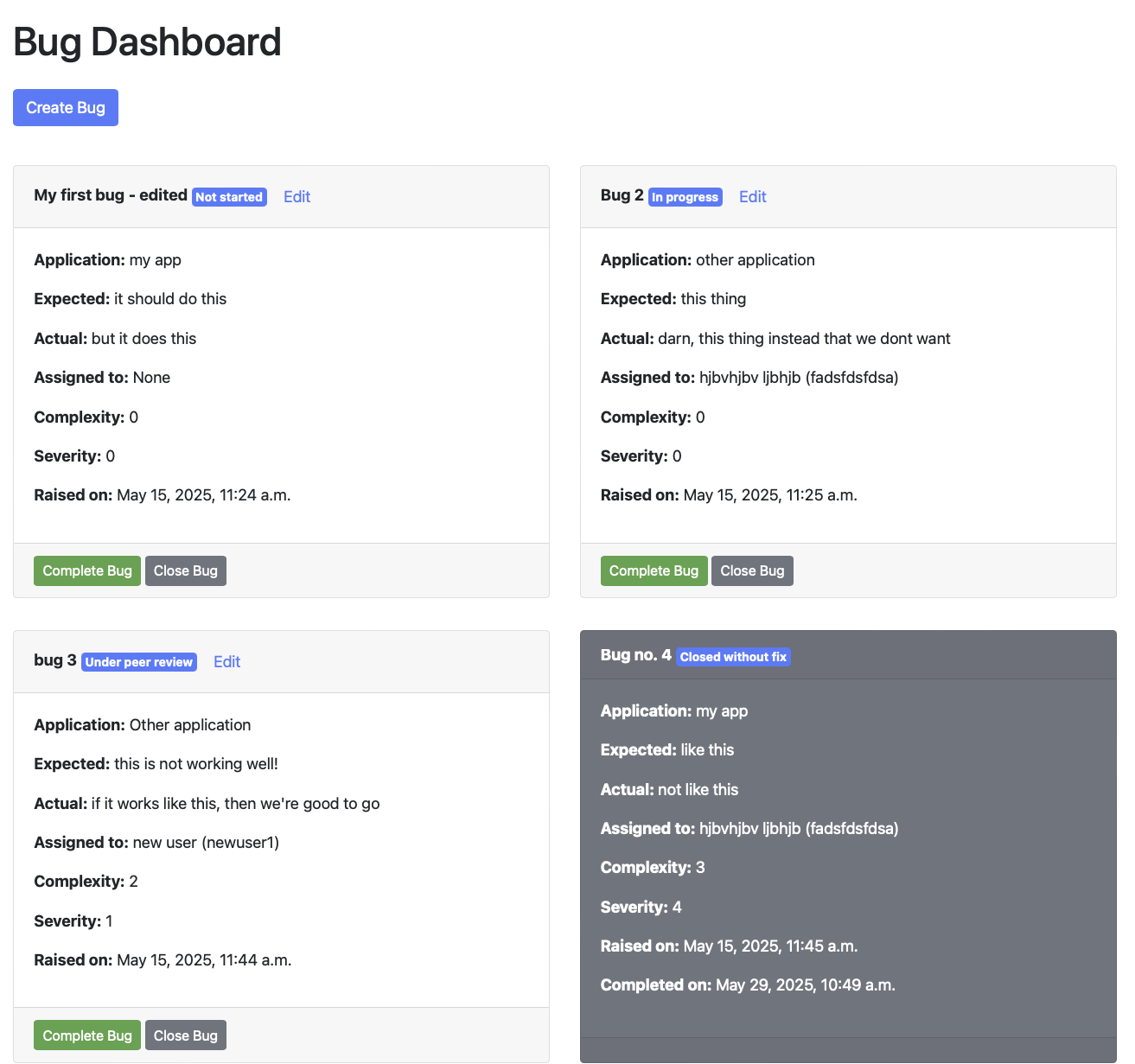


Figure 3 - Normal user dashboard

Figure 4 - Admin user dashboard

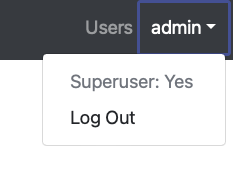
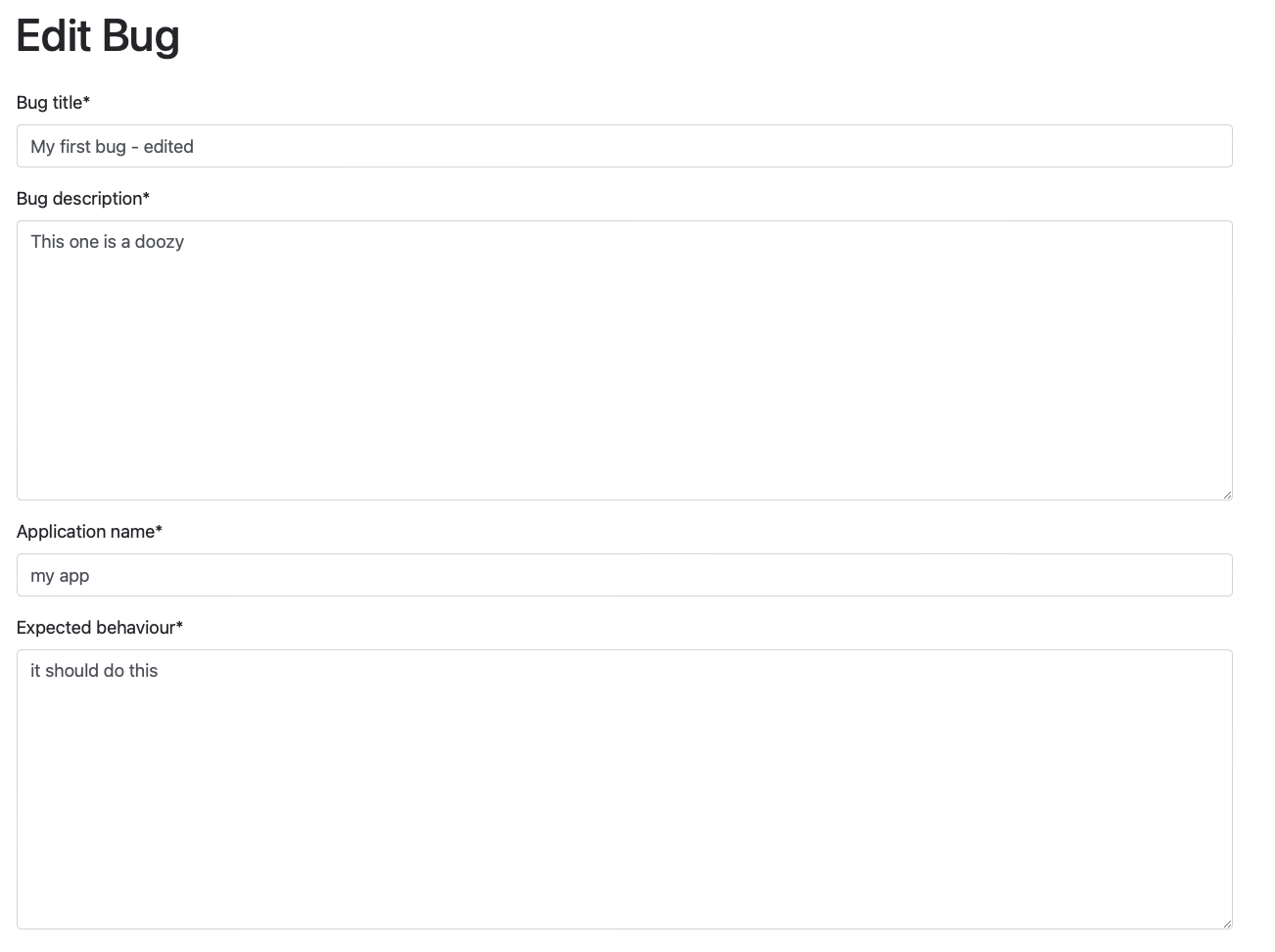


Figure 5- Edit bug screen

Figure 6 - Admin user navbar