# CSIT315: Group 4's Project Report

Team Members: Aidan Perez, Juan Rodriguez, Marc Puente, and Mengjia Yu
Date: 12/11/2025

---

# Process Model

*Evolutionary Process Model*

The team decided that the best process model for our project would be the Evolutionary Process. The model helped and allowed the development of bug tracker to be repeatedly refined through the different stages of the software development process multiple times before it was finalized and ready to be tested. The iterative process aligned well with the goal we had in mind for designing our bug-tracking system as it was based on continuous feedback and improved understanding of how we wanted to accomplish the requirements for the project.

# Requirement Gathering

*Brainstorming, Questionnaire*

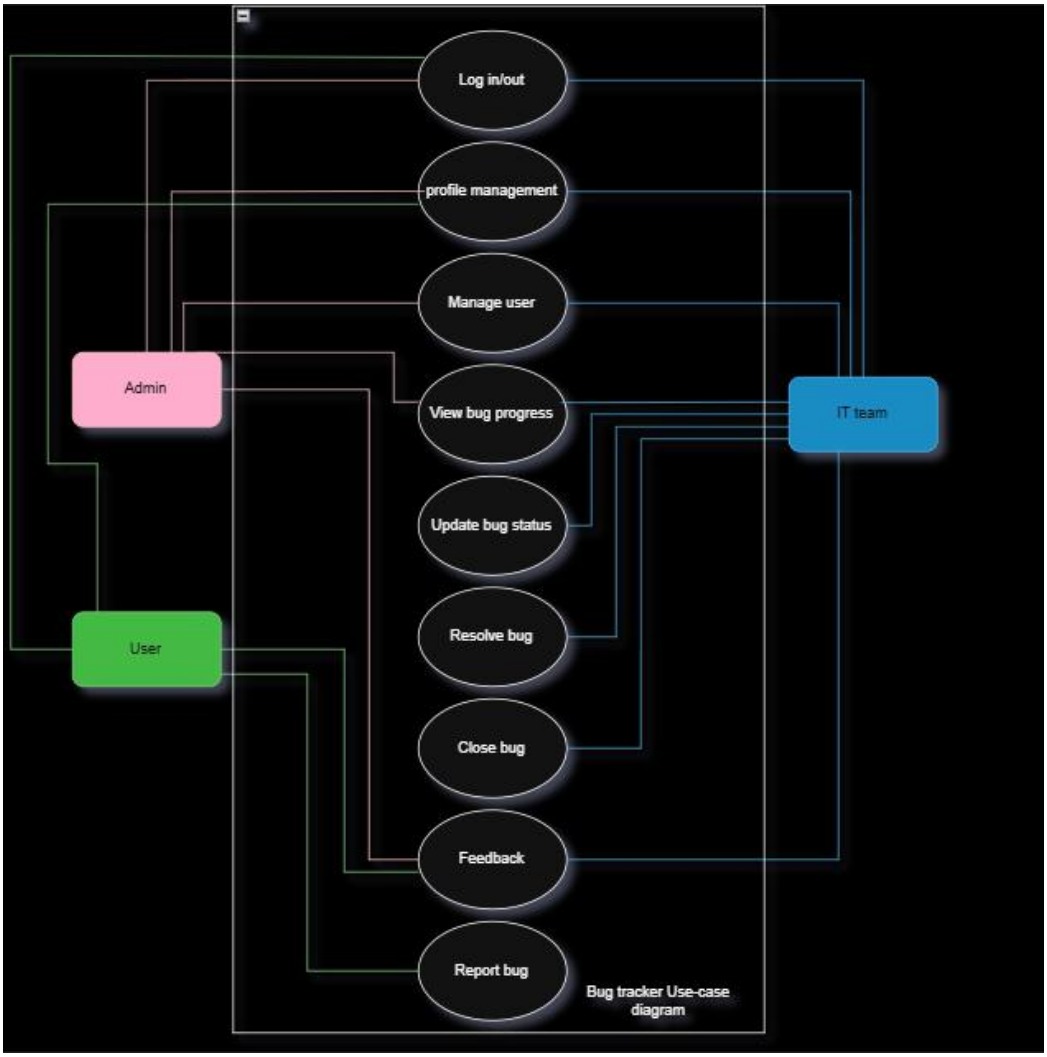
Initially, during the early stages of our research, the team brainstormed ideas of how to construct our bug tracker through shared ideas and thoughts during lab days. Although some of the ideas proved to stick with us throughout the entire process of our project, others felt like they lacked, weren't needed, or overcomplicated things. For that reason, we decided to conduct a survey using Google Forms. We distributed the survey to students at Montclair and professionals who already have working experience in the technology field, including contacts from companies such as FedEx and FanDuel. Students helped represent the main users for our systems, while professionals helped provide insight into real-life bug management. Our survey included questions such as:

- What is the most important action you need in a bug tracking system?
- What information should be included in a bug report to be useful?
- Would you like to provide feedback after an issue is resolved?

These responses directly contributed to system decisions like simple bug reporting, wanting status updates, and submitting feedback after bugs are resolved

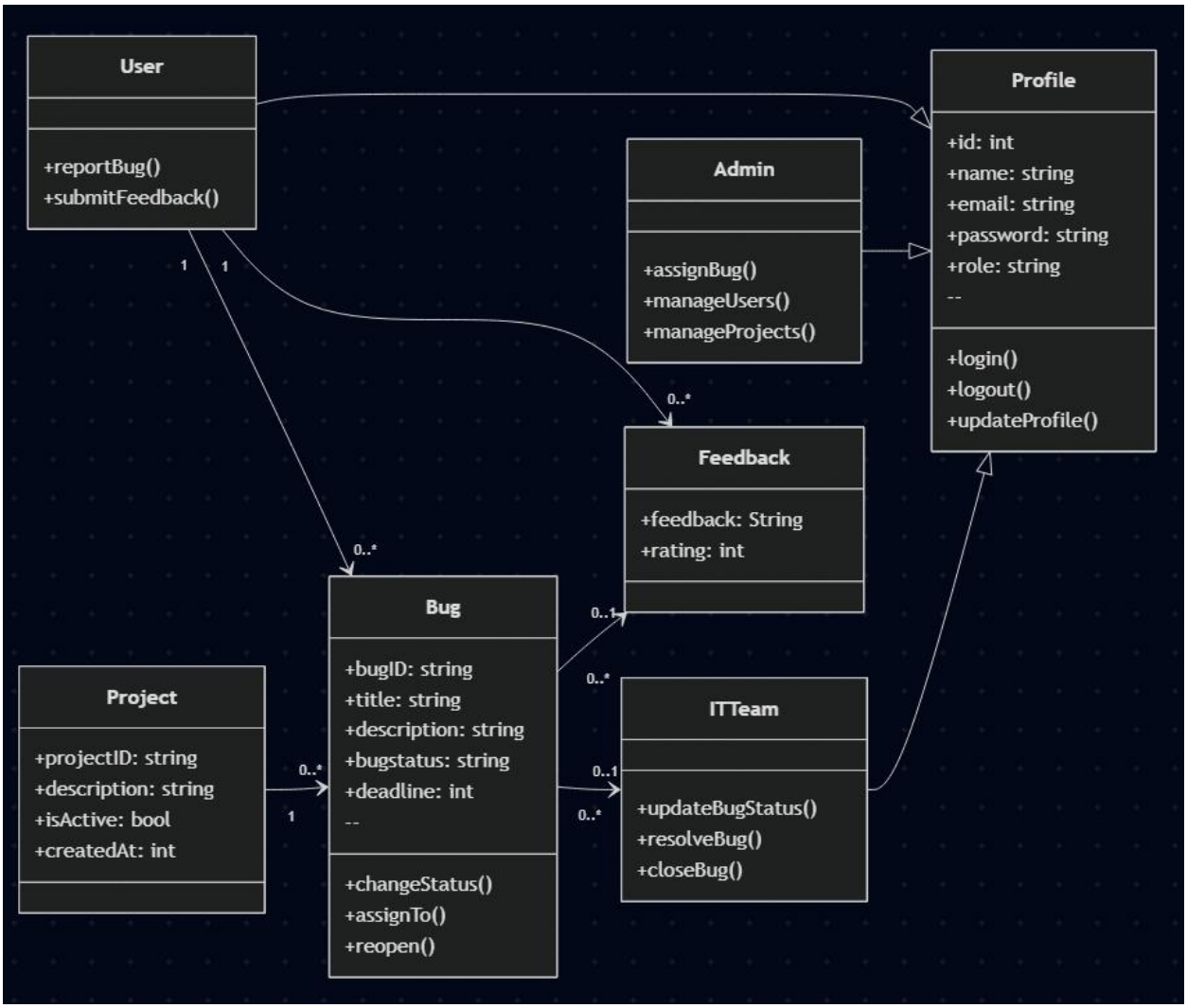
# Requirements Specification

*UML Diagrams*




**Admin**: Manages users/projects
**User**: Submit feedback and reports
**IT team**: Heart of the project in
In charge of handling bugs

*Class Diagram*

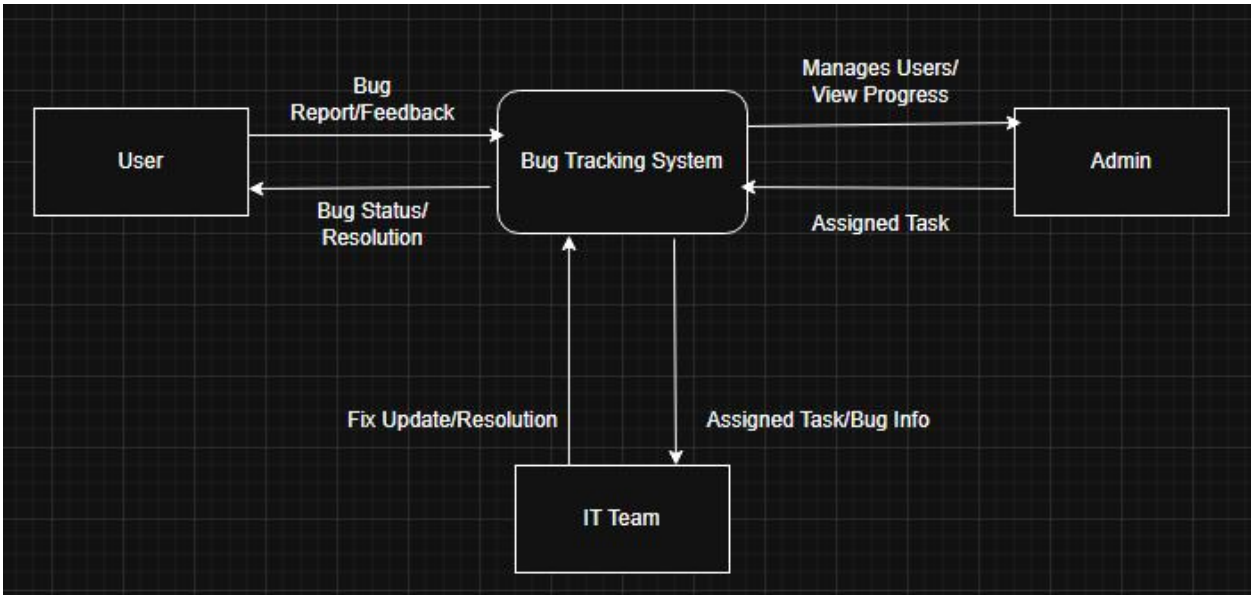



Brief Description: Shared parent class, Profile, helps provide the common functionality and attributes such as login, logout, and profile updates. The classes that inherent from Profile are:

- Admin - manages users, assigns bugs, and projects
- User - reports the bug and submits feedback after the issue is resolved
- ITTeam - resolves the issues, updates the status, and then closes the bug once it's complete

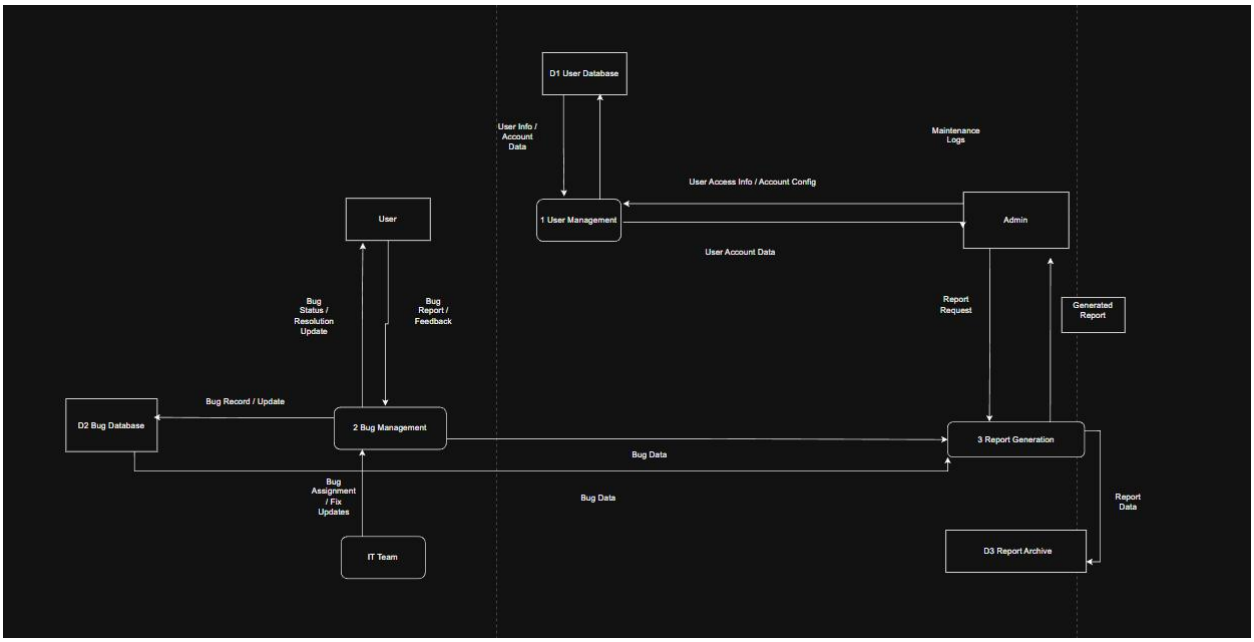
Bug Class contains all the essential information, and Project allows to group up as many bugs into one Project. The project helps organize each bug within specific scopes. Once the bug has been resolved, Users submit Feedback connected to that specific bug.

*DFD Level 0*



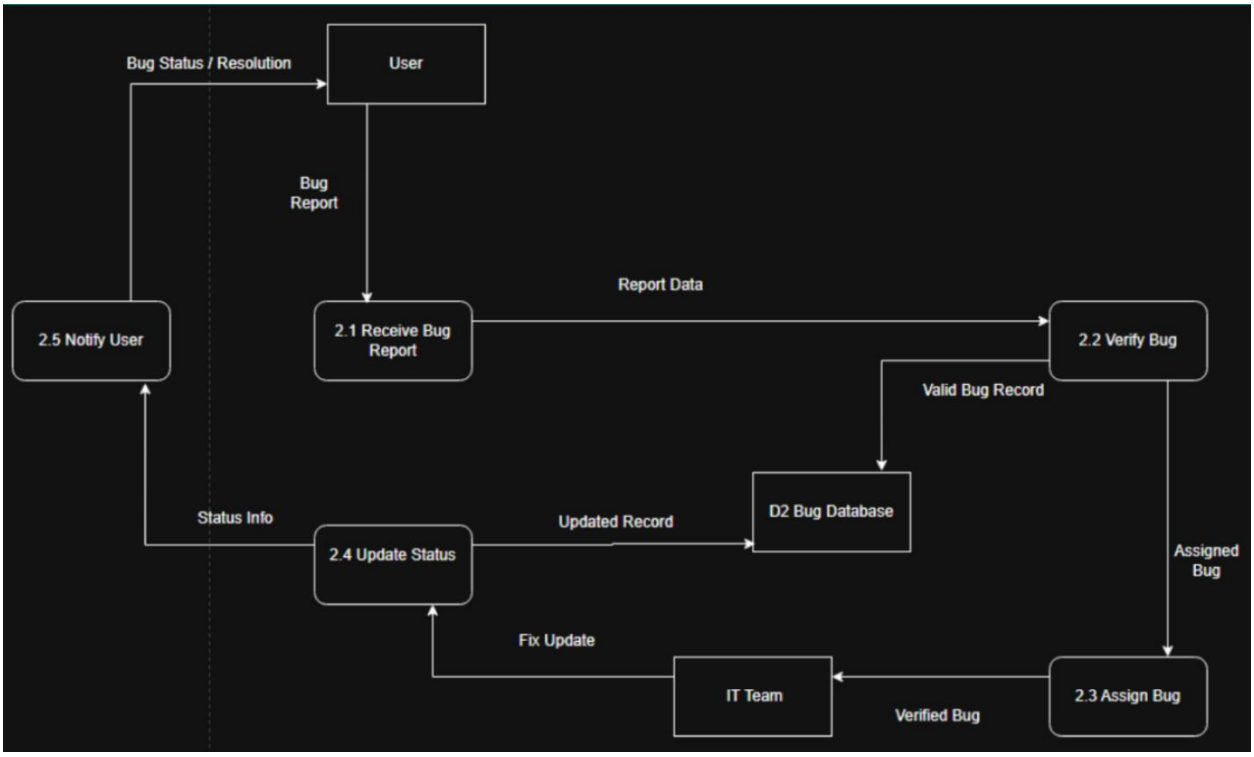

Brief Description: Flow between the actors and the Bug Tracking System. Highlights how users interact with the system to report bugs, how admins manages users while keeping track of the progress of the IT Team's work on bugs

DFD Level 1



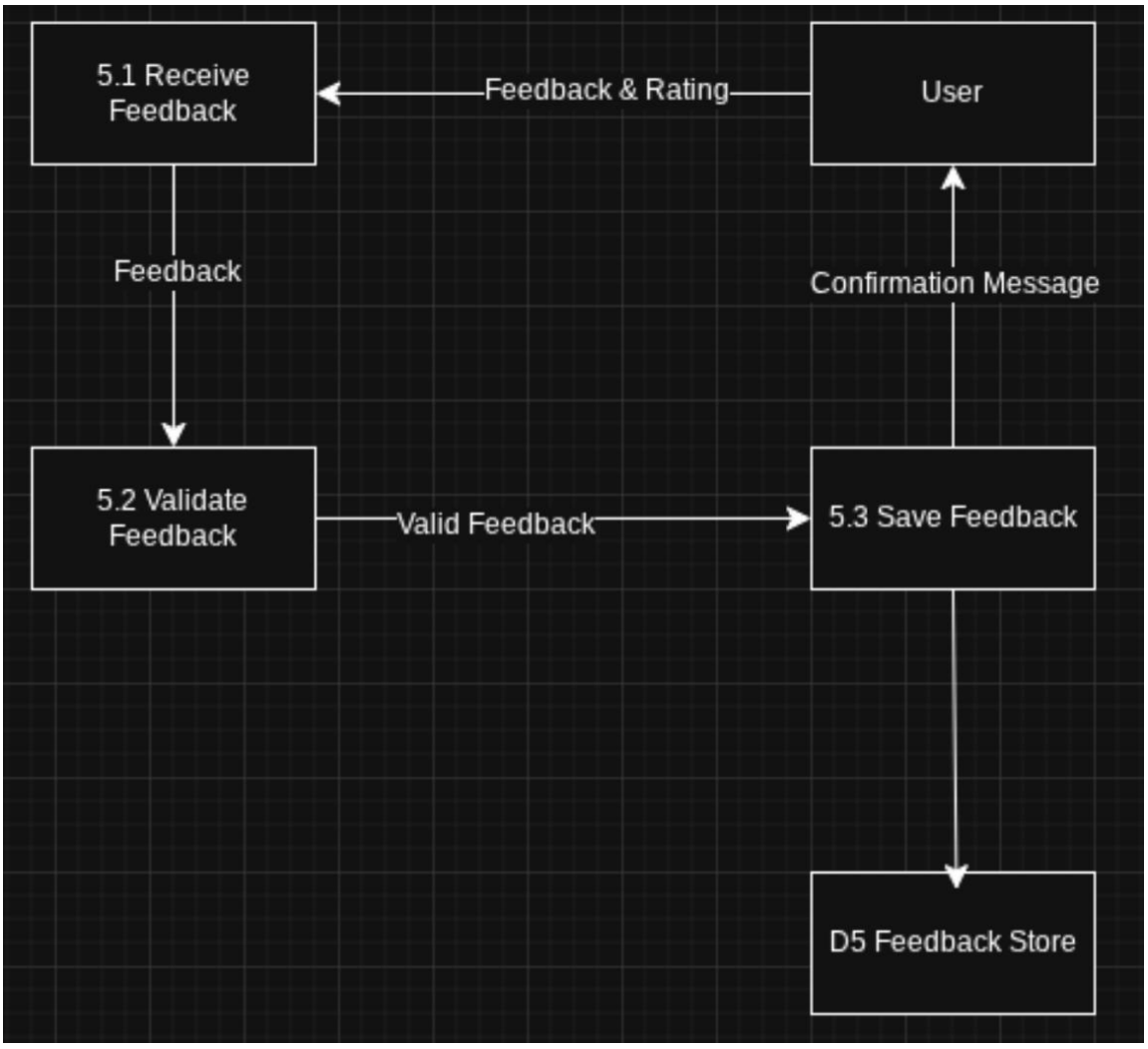
Brief description: The Level 1 diagram breaks the overall system into its four primary processes: User Management, Bug Management, Report Generation, and System Maintenance. User Management handles account creation, login, and access configuration by interacting with the User Database. Bug Management manages bug reporting, validation, database updates, and communication with the IT Team. Report Generation compiles system and bug data to produce reports requested by the Admin and stores them in the Report Archive. System Maintenance supports administrative tasks such as managing logs and updating system configurations, with records saved in the System Log File. Together, these processes illustrate how data flows throughout the system and how Users, Admins, and the IT Team interact with each subsystem.

*DFD 2 - Bug Management*



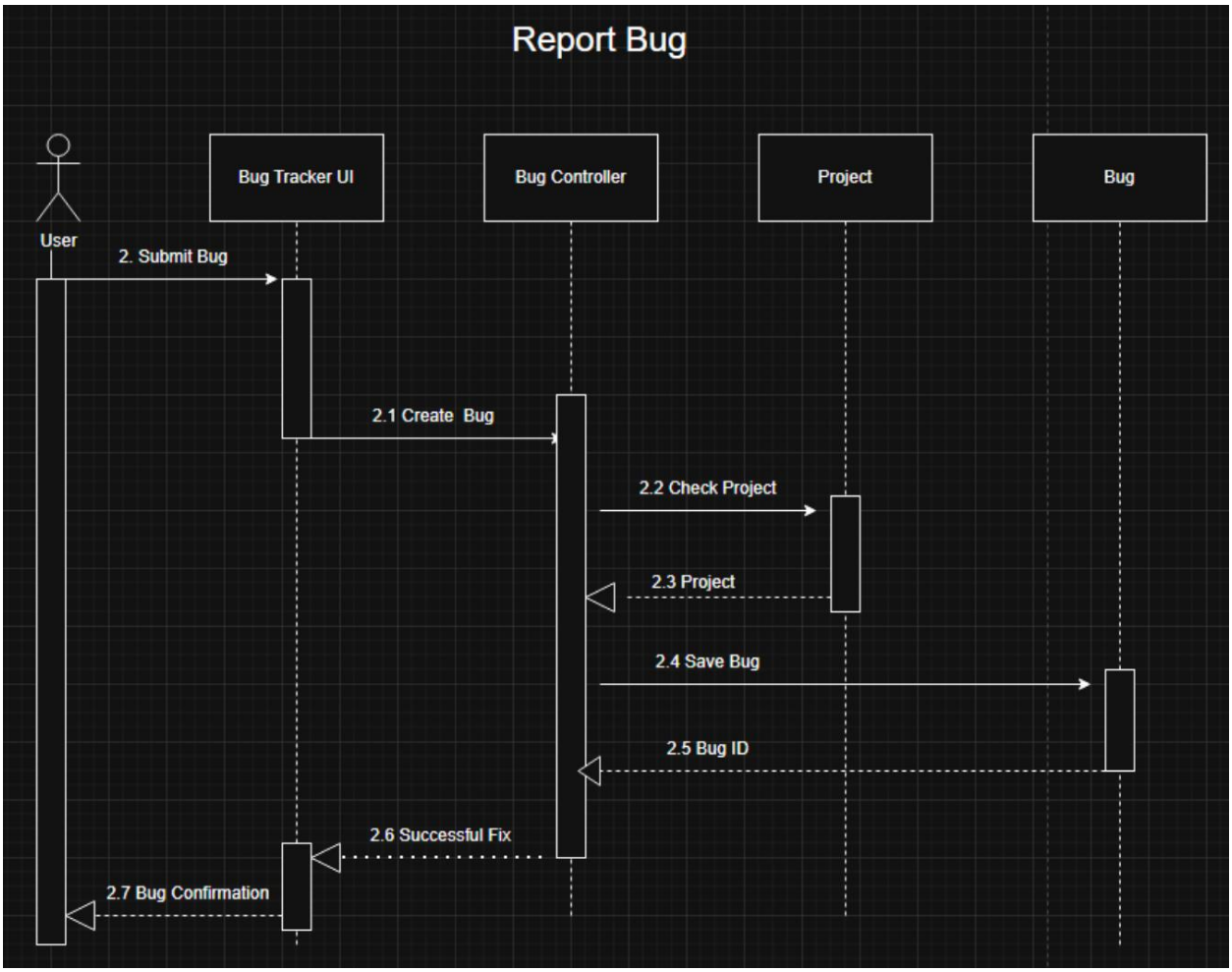

Brief description: Breaks down the bug reporting and resolution process. The User submits a bug, which is verified by the system and sent to the database. Once it's valid, the IT Team updates the status, and the system notifies the User of the resolution.

*DFD 2 - Submit Feedback*



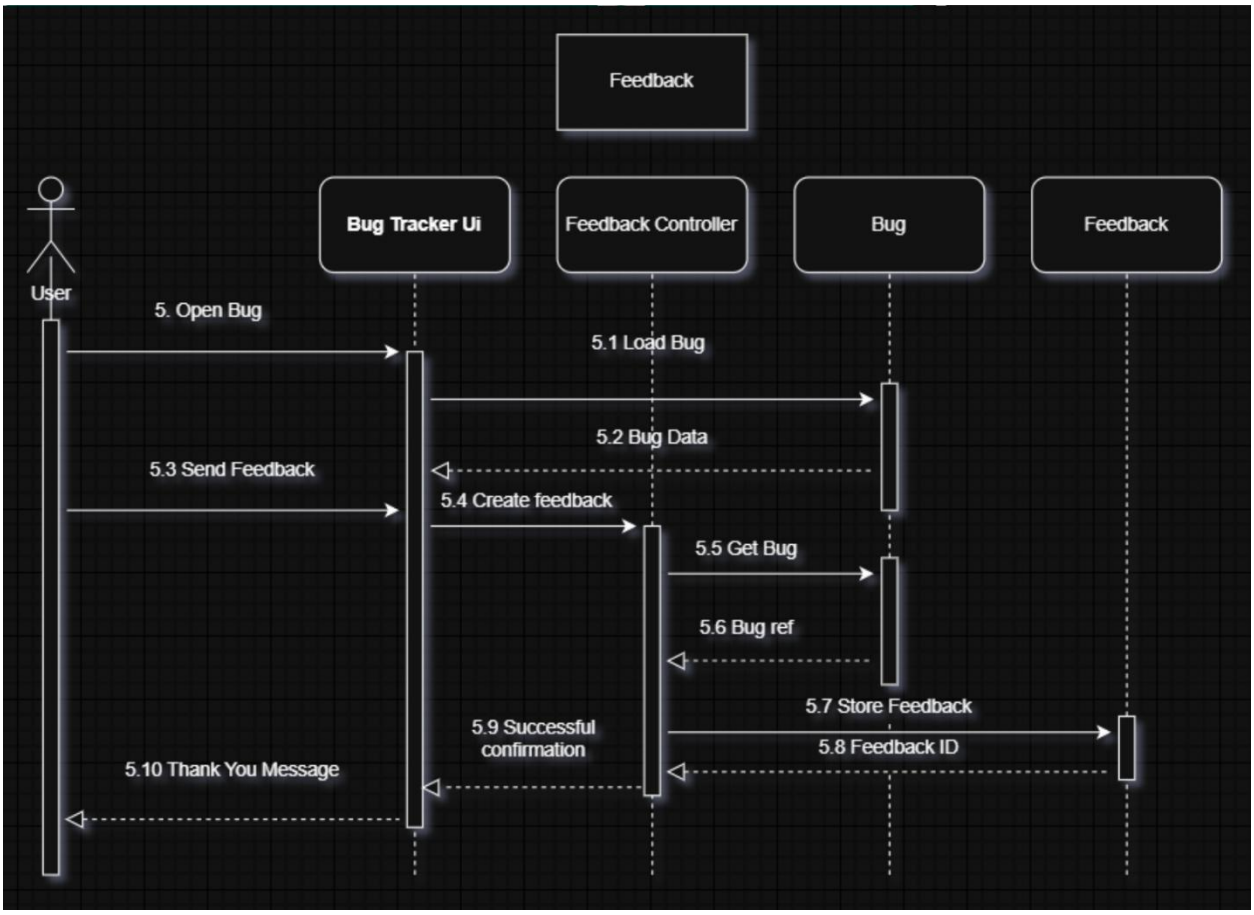

Brief description: Shows how the user feedback is handled. When a user submits a rating and feedback, the system receives and validates it. It is then saved into a database, and Users will be notified that their feedback was recorded.

*Sequence Diagram, Report Bug*

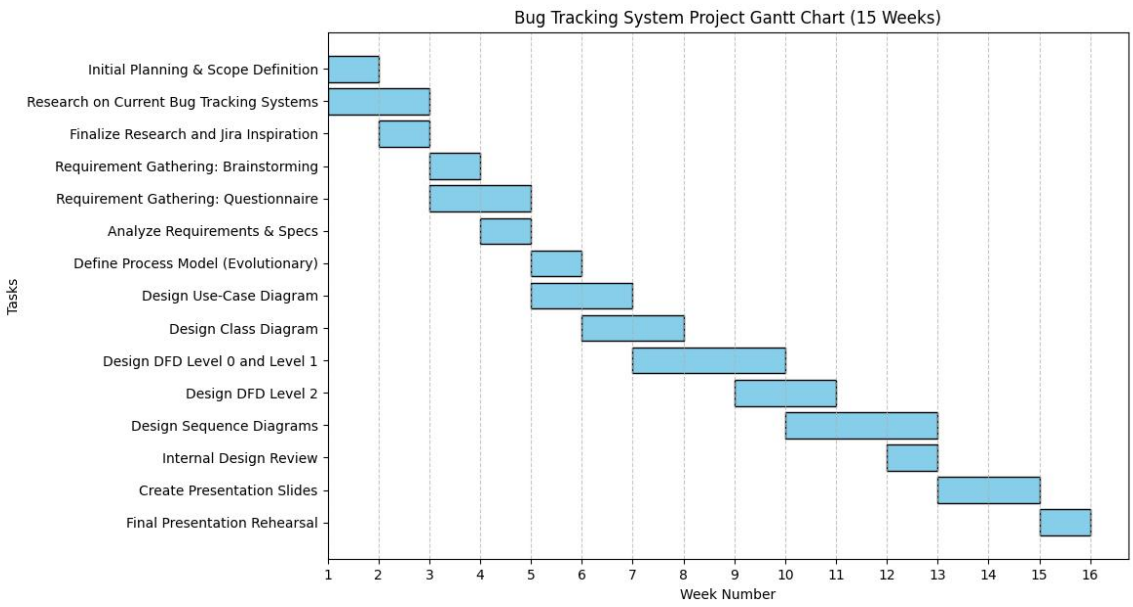



Brief Description: Shows how the process of a User reporting a new bug, with them submitting the details through the UI, which sends the request to the Bug Controller to create the bug. It then checks the related project and saves it into the system. Once it's been successfully stored and assigned an ID, the User receives confirmation that the bug has been submitted.

*Sequence Diagram, Feedback*




Brief Description: Shows how the user provides feedback. The User opens the bug and submits their feedback through the UI. The Feedback Controller retrieves the bug information and then stores the feedback in the system. Once the feedback is saved, the User receives a confirmation message for their submission.




# Limitations

1. Diagrams

- Initially, all of our diagrams were hand-drawn. Although the content on the Use-Case and Class Diagram weren't entirely wrong, it didn't meet the professional standards that the professor was looking for. Because of this, we were behind as all of our diagrams were then needed to be refined with changes made using websites like Mermaid Live and DrawIO which were completely new to all of us at the time.

2. Google Forms

- Responses from our outside sources took longer than anticipated. We had to wait around almost 2-3 weeks after our initial invitation was sent.

3. Communication

- It was difficult for us to stay in contact with one another. With our course work to worry about, most of us also had outside jobs over the weekends, which made it especially difficult to work with free time to do calls at any given time. It was only until a GroupMe chat was made that we were able to organize and stay on task without lacking behind.

4. Overcomplicated Diagrams

- Related to the diagrams, our initial Use-Case as Class Diagram had way too many actors and different positions in our Bug Tracker that just made the diagram way too

complicated and almost messy at times. This was fixed along with the remade diagrams on Mermaid Live and DrawIO