Group 4
Bug Tracking System Design

An effort by:

Aidan Perez, Juan Rodriguez,
Marc Puente, and Mengjia Yu

Project scope

In this presentation we hope to cover:

- Research on bug tracking system currently out in the market.
- Elicitation
- Requirement gathering
- Diagrams

Research

- Bugzilla: advanced reporting and custom fields
- Jira: workflow customization, kanban boards(using sprints)
- Redmine: Good at categorization(bug, feature, defect)

Jira pros	Bugzilla pros
Excellent for agile/scrum	Free to use
Highly visual	Custom fields
Rich integrations	Good for deep data analysis

Jira inspiration

- Organization: Well detailed info + deadlines.
- Reporting: great visualization and displays database status
- Workflow pattern: “To do - In progress - Testing - Done”

Requirement gathering

- Our team decided two methods for requirement gathering would work best.
  - Brainstorming
  - Questionnaire

Brainstorming

- Helped avoid any assumptions during early research
- Allowed us as a team to help decide who were going to be main our main actors in the future
- Supported the early grouping with the processes in our diagrams

Big takeaways:

- Defining roles
- Grouping tasks

Questionnaire

- Gathered students and other outside sources and asked them some questions:

- 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?

Requirements

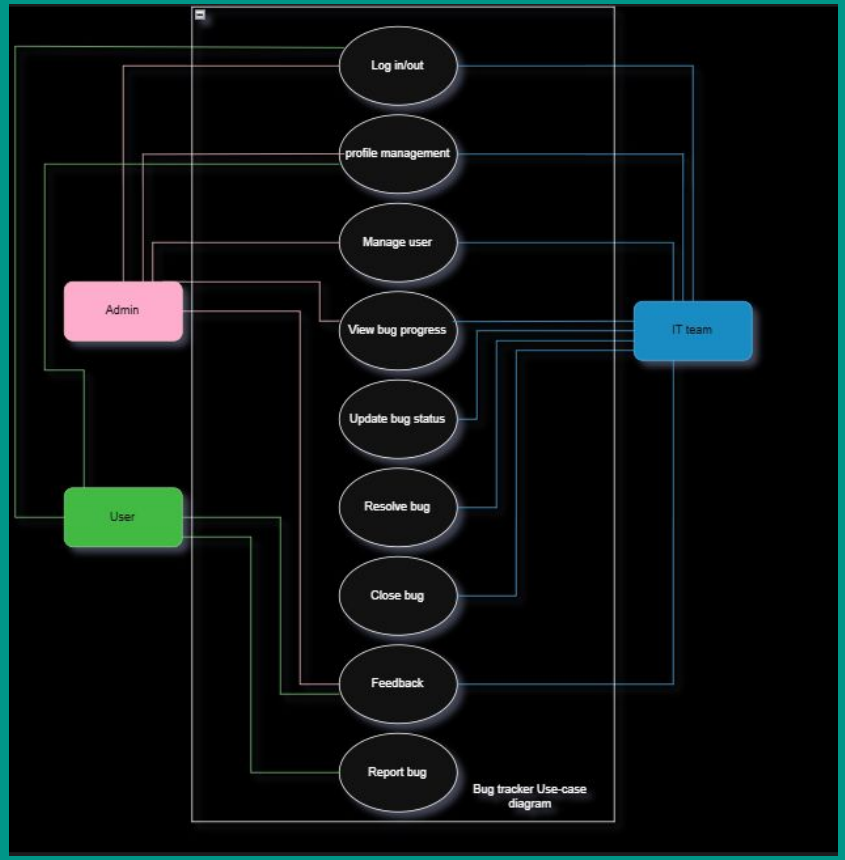
The requirements that we listed were:

- Functional
  - User Authorization
  - Create/Assign Bugs
  - Feedback
- Nonfunctional
  - User-friendly UI
  - Export capabilities (ex, .csv)

Process model: Evolutionary model

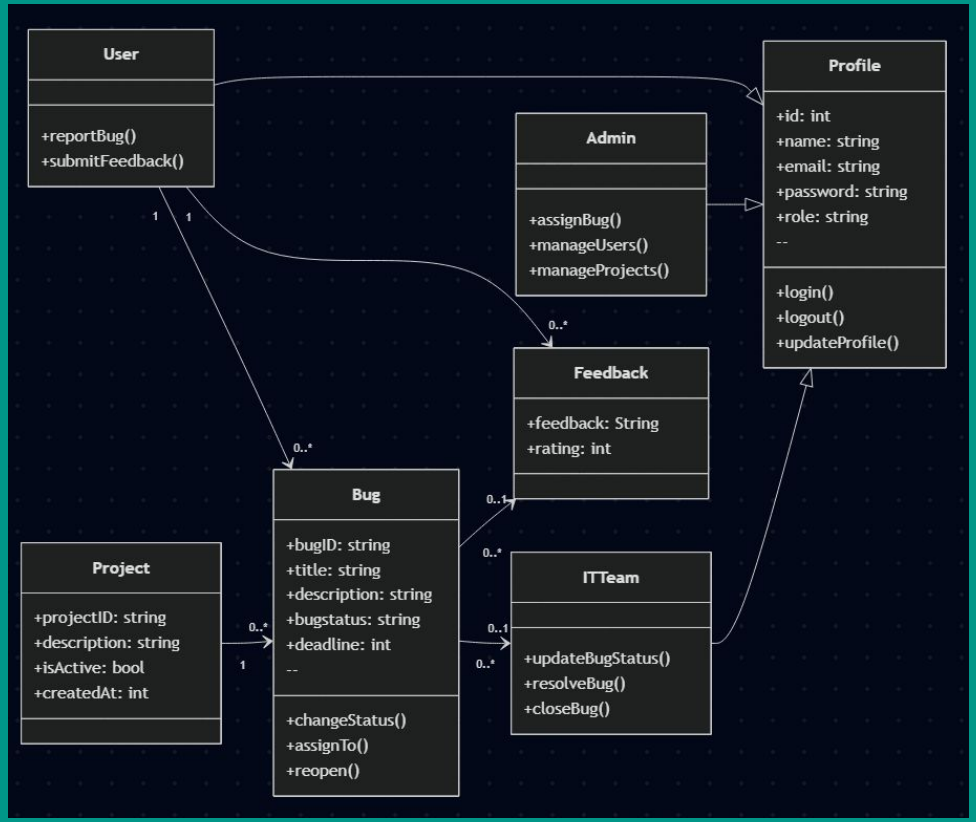
- Allows the team to repeat any process more than one time before moving on to the next. By using the iterative process flow

Use-Case



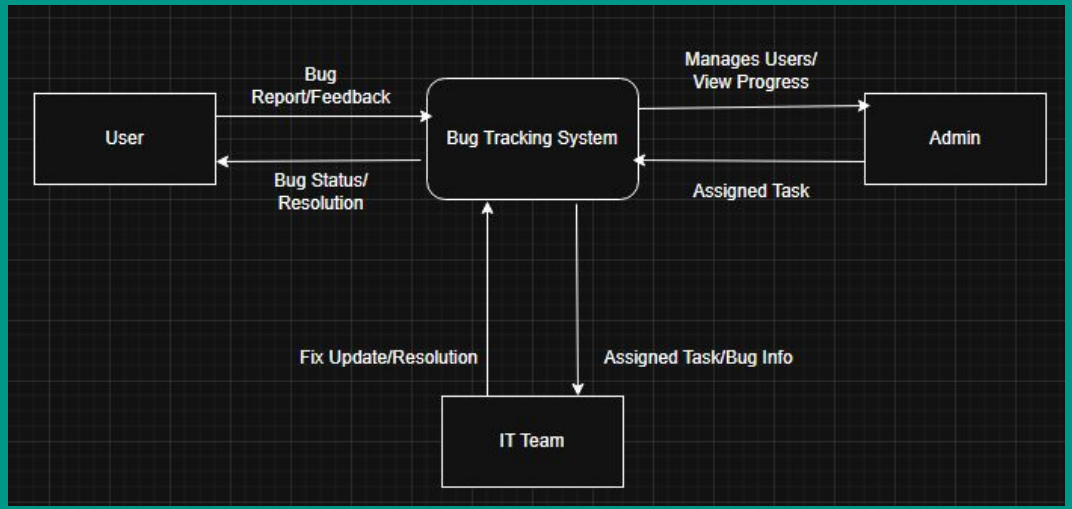
- **Admin**: Manages users/projects
- **User**: Submit feedback and reports
- **IT team**: Heart of the project in
  In charge of handling bugs

Class Diagram



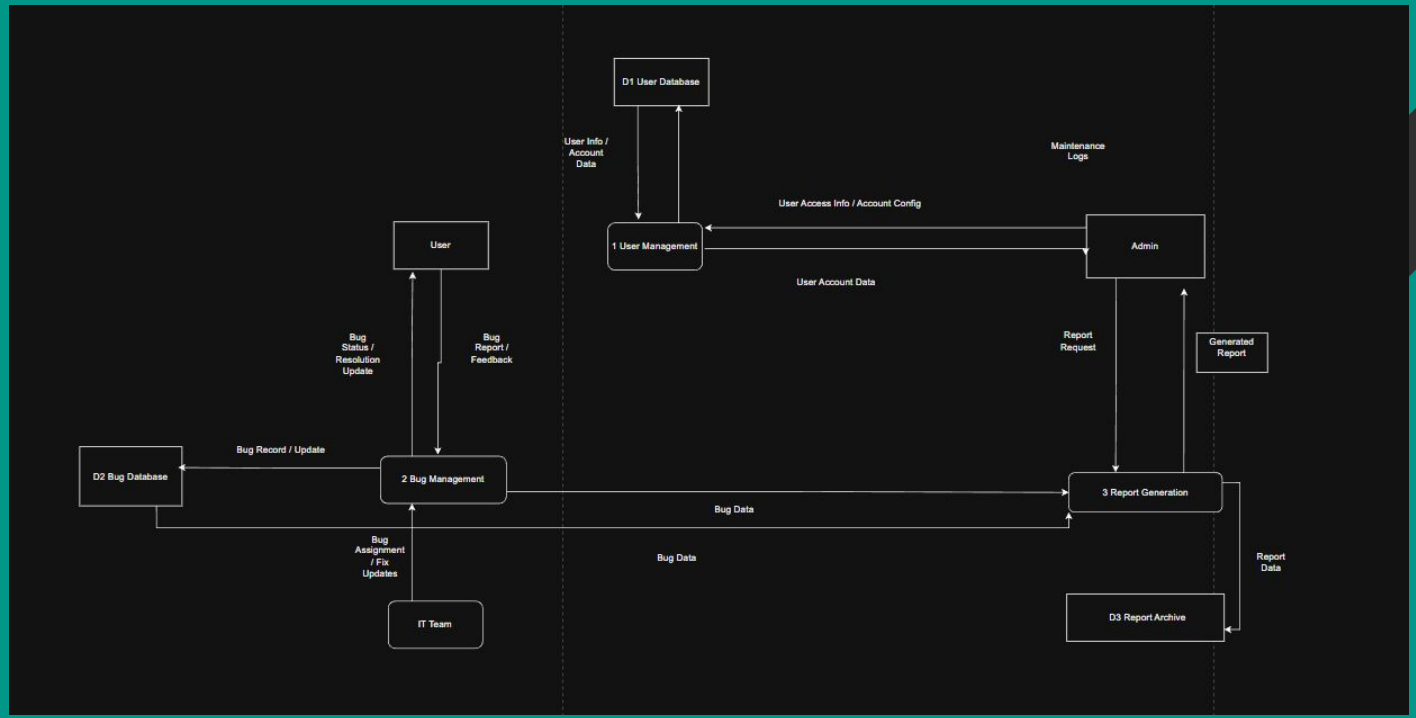
- **Profile**: acts as parent “Admin”, “User”, “ITTeam”
- This helps keep the authentication in a streamline
- Achieves higher cohesion to keep the classes single minded

DFD Level 0

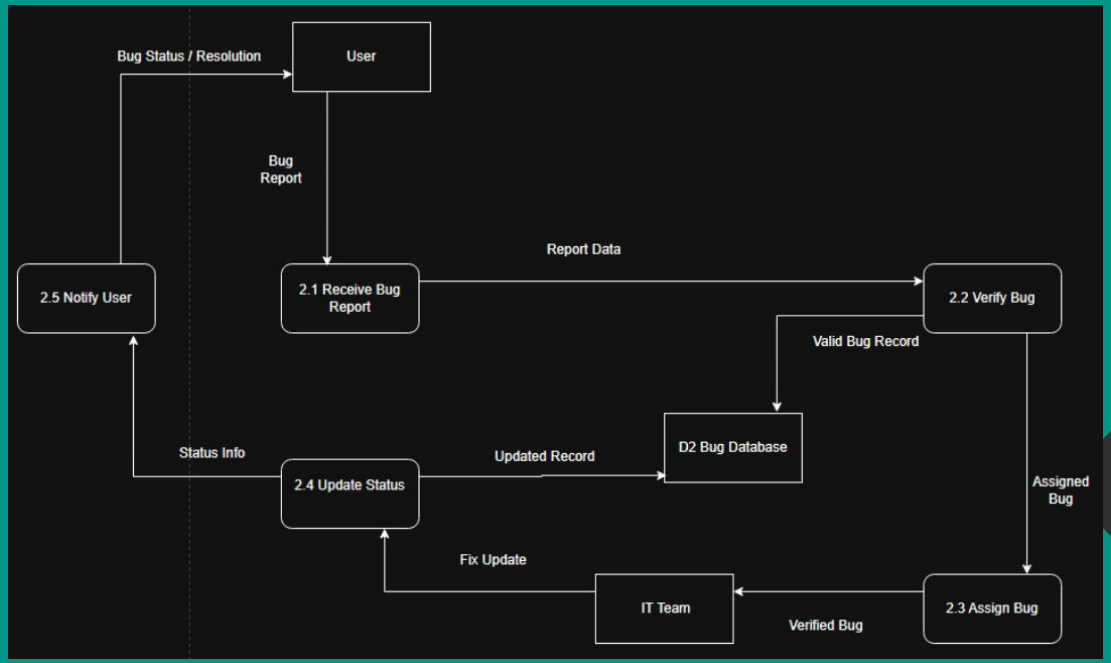


- External entities: User, It team, admin
- User: Submits bug reports and receives status updates
- It Team: Receives assigned tasks and resolves them.
- Admin: Oversees the submission and solving of bugs

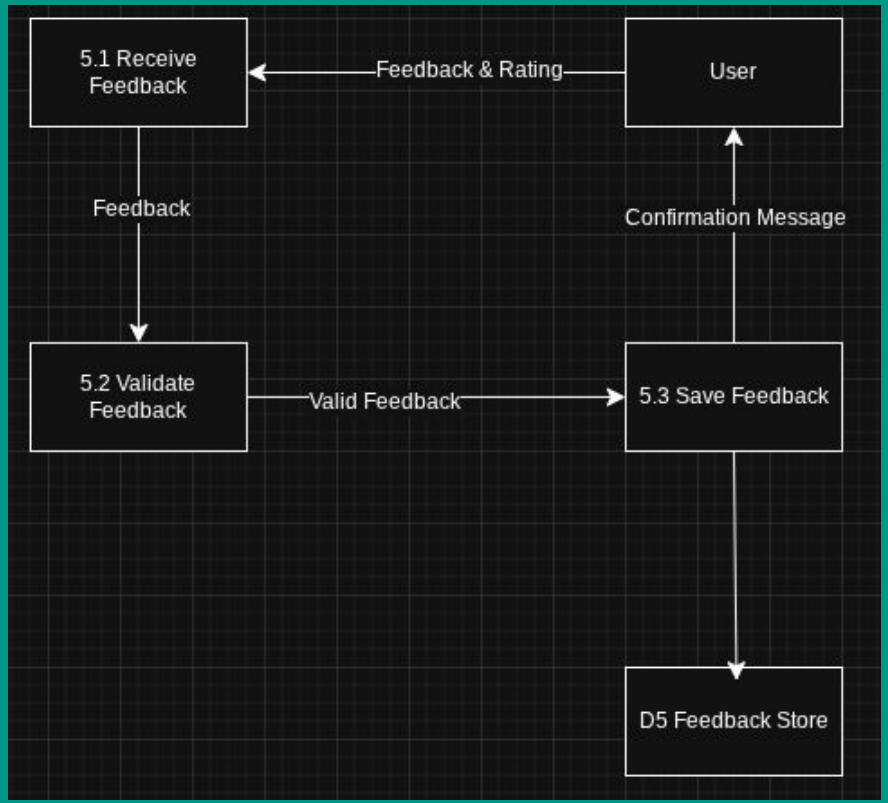
DFD Level 1


D1 User Database
User Info / Account Data
Maintenance Logs
User Access Info / Account Config
User
1 User Management
Admin
User Account Data
Bug Status / Resolution Update
Bug Report / Feedback
Report Request
Generated Report
Bug Record / Update
D2 Bug Database
2 Bug Management
3 Report Generation
Bug Data
Bug Assignment / Fix Updates
Bug Data
Report Data
IT Team
D3 Report Archive

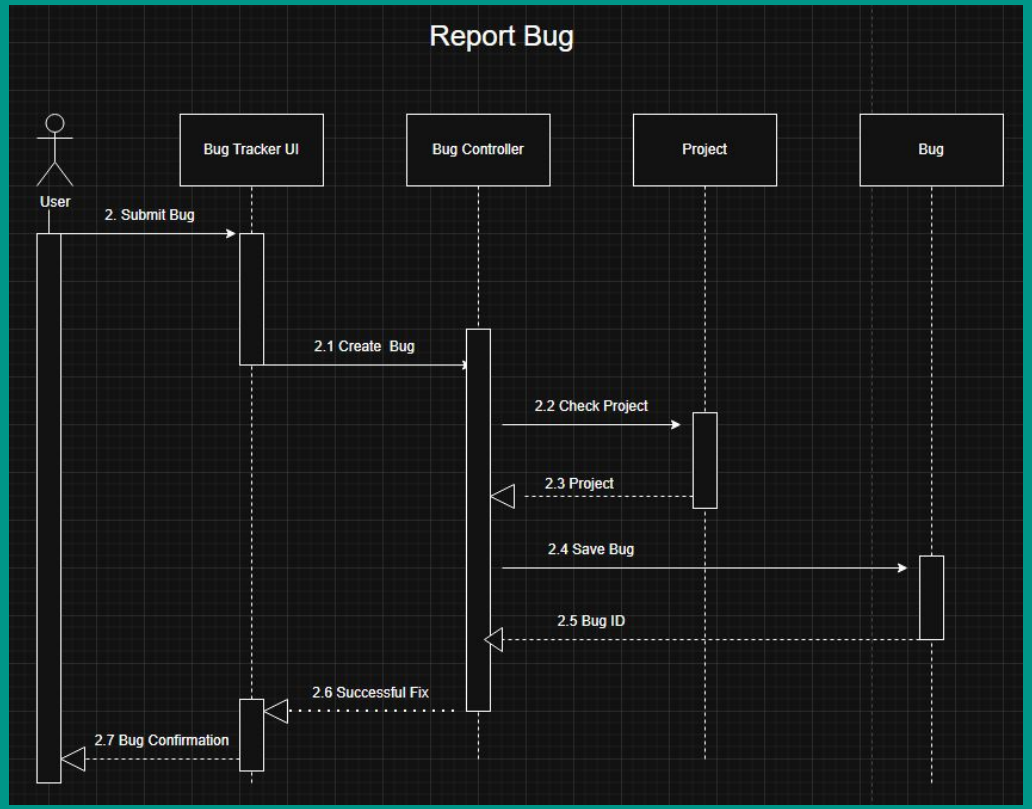
DFD Level 2 (Bug Management)



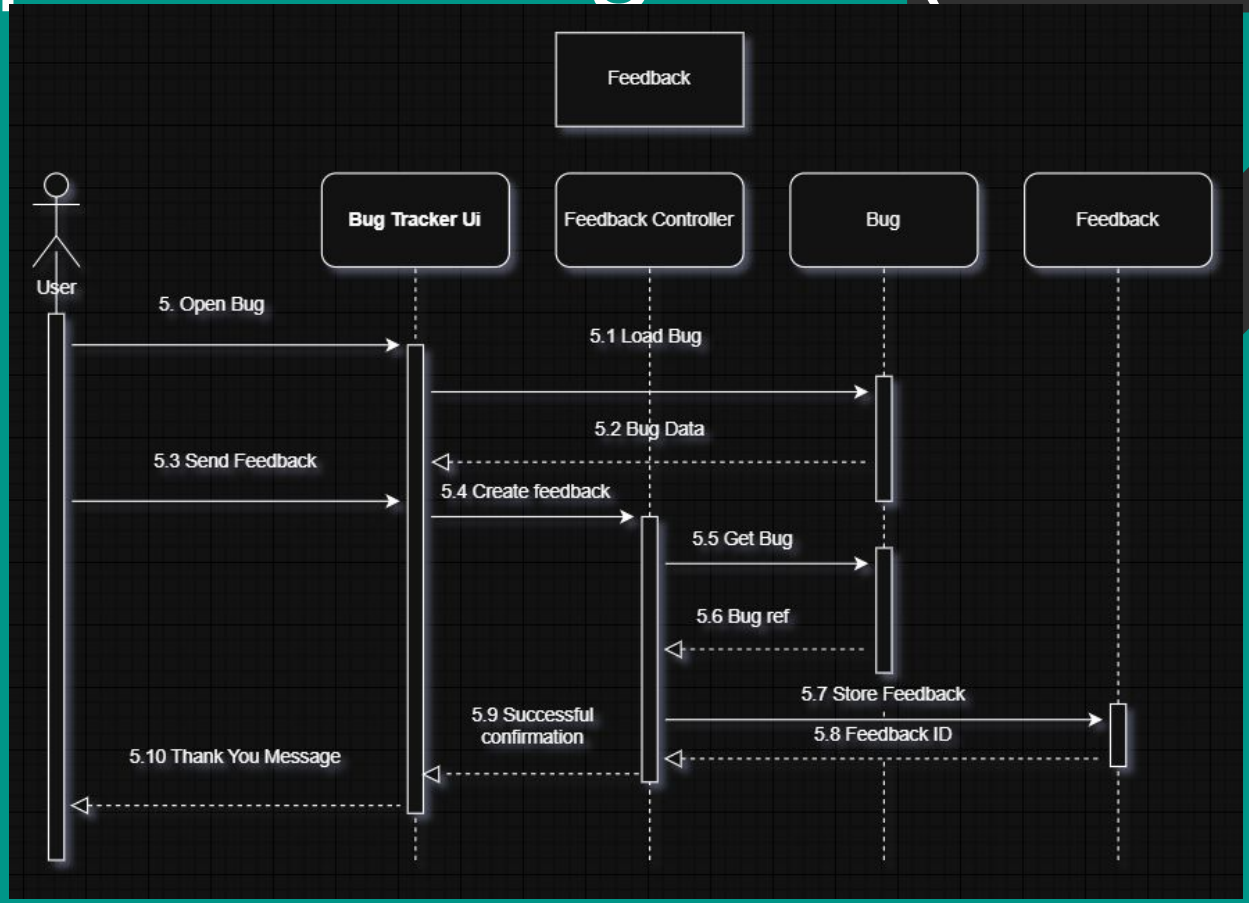
DFD Level 2 (Feedback)



Sequence Diagram (Report Bug)



Sequence Diagram (Feedback)



What we found difficult:

- Diagrams were hand-drawn; lacked behind after midterm
- Google Forms
- Communication
- Too many actors

Conclusion:

- Techniques for research
- How to design UML diagrams appropriately
- Hands on experience

Thank you for your time!