

Final Project 1

Code: MGM 496-PMG496

Project Template

Faculty of Management
Midocean University

Midocean University

Faculty of Management

Department/ Specialization



Project title
Bug Tracking System

Student name

.....

Graduation Project – 1
Faculty of Management - Midocean University

Supervisor

Dr.

Faculty member- Faculty of management- Midocean University

Date of submission – number of Semester

Table of Contents

1. Chapter 1: Introduction	3
1.1. Research Background.....	4
1.2. Research Problem.....	6
1.2.1. Theoretical Gaps	6
1.2.2. Practical Gaps	7
1.3. Research Significance	8
1.3.1. At the theoretical level.....	8
1.3.2. At the practical level	9
1.4. Research Questions	10
1.5. Research Objectives	11
1.6. Research Hypotheses.....	12
1.7. Research Model.....	13
1.8. Research Methodology.....	14
1.9. Scope of The Study	15
1.10. The Suggested Project Structure.....	33
2. Chapter 2: Literature Review	36
2.1. Introduction	37
2.2. The independent variable	37
2.3. The dependent variable	37
2.4. The relationship between the independent and the dependent variables	38
2.5. Conclusion.....	38
References.....	39

1. Chapter 1: Introduction

1.1. Research Background

- The bug tracking application carries great theoretical significance, as it embodies software engineering concepts such as requirements management, traceability, documentation, and impact analysis, enhancing the quality of development and deepening the academic understanding of the software lifecycle.
- With the increasing reliance of organizations on software, the development of their software products and their delivery to customers has become an indispensable priority, so these organizations rely on software project management as a basic practice, through which software development, starting from the initial preparation of requirements to the final testing and maintenance, is implemented over a predetermined period to produce the agreed software, and in the following lines, we explain the definition of software project management, its activities, and the most important challenges it faces.
- Software projects can be described as the activities of designing and organizing software development according to several factors, most notably time, cost, quality, and scale, which is a subcategory of the software development process.
- Thus, software project management is defined as the processes and control practices for planning, scheduling, and supervising software projects, as it works to allocate resources to develop computer programs that meet the predetermined requirements, and in this program, it is necessary to clarify the length, duration, and cost of the project for the client and developers.
- In a software project management system, project managers, stakeholders and users monitor costs, manage budget and quality, and can use this system as a management system.

- Software project management is the process of planning, organizing, and supervising software development projects to ensure their success. It aims to deliver the project on time, within the specified budget, and with the required quality, while controlling the scope of the project and risks. It includes key activities such as scoping and resource allocation, progress tracking, and communication and risk management.

Main activities of software project management:

- **Planning and scheduling:** Develop a detailed work plan and timeline for project implementation.
- **Scoping:** Accurately define project boundaries and requirements to avoid unwanted changes.
- **Resource Management:** Effectively allocating human, physical, and financial resources.
- **Cost and budget management:** Monitor costs and ensure that the project stays within budget.
- **Quality Management:** Ensuring that project deliverables meet the set quality standards.
- **Risk management:** Identifying potential risks and developing plans to deal with them.
- **Progress tracking and reporting:** Monitor workflow, track obstacles, and periodically report on project progress.
- **Communication and Collaboration:** Facilitate effective communication between team members and stakeholders.
- **Document Management:** Organize and document all documents related to the project.

1.2. Research Problem

- Difficulty in accurately and effectively managing and tracking bugs in a dynamic, multi-team development environment, resulting in processing delays, repeat errors, and low quality of the final product.
- "Software developers suffer from the absence of a standardized conceptual bug tracking model that connects requirements, code, and test cases, leading to poor impact analysis and difficulty in quality management."
- "Current systems lack a flexible theoretical framework that allows for the classification and prioritization of bugs in a way that is adaptable to modern development methodologies."

1.2.1. Theoretical Gaps

The theoretical problem of the bug tracking application project lies in designing a system that is able to accurately represent software bugs and link them to the various components of the system, while maintaining scalability, traceability, and analysis without complexity or ambiguity.

- Structural representation of the bug Specifying the characteristics that each bug record should contain (e.g., status, priority, source, impact, date) in a way that allows for analysis without excessive complexity.
- Linking the bug to the components of the system such as requirements, code, and test cases, which requires a Traceability Model that illustrates the relationships between the elements.
- Classification and Prioritization of Ills Developing a theoretical framework for classifying bugs according to impact, frequency, or source, which helps in making effective treatment decisions.
- Bug Lifecycle Management Design a model that shows the bug stages from detection to closure, specifying actions at each stage.

1.2.2. Practical Gaps

The practical problem of the bug tracking application project is that it is difficult to ensure documentation accuracy, traceability, and usability within a fast-paced and changing development environment, which affects product quality and speed of response to errors.

- Inaccurate or incomplete data input Developers may neglect to adequately document the bug, making it difficult to reproduce or analyze it later.
- Poor adherence to the use of the system Some teams do not commit to updating or recording the status of the bug, which leads to its loss of tracking or recurrence.
- Lack of integration with other development tools such as Git, Jenkins, or CI/CD tools, which makes the tracking manual and increases the effort required.
- User interface complexity or poor user experience If the application is not intuitive or requires many steps, it is less used and the possibility of error increases.
- Difficulty in classifying and prioritizing the illnesses: The absence of a clear mechanism to determine the seriousness of the disease or its impact on the system.
- Poor statistical analysis and reports The lack of in-app analysis tools prevents teams from understanding fault patterns or evaluating performance.
- Performance problems or expansion of large projects, the application may experience slowness or difficulty in dealing with a large number of bugs.

1.3. Research Significance

The bug tracking application project is one of the vital projects in the field of software development, as it has a direct impact on improving software quality, speeding up maintenance processes, and enhancing cooperation between teams.

1.3.1. At the theoretical level

The importance of the software bug tracking project at the theoretical level lies in consolidating software engineering concepts, achieving integration between the stages of the development lifecycle, and enhancing the quality of software through systematic tracking and structural analysis of bugs.

- The project practically embodies concepts such as requirements management, traceability, and bug lifecycle, enhancing theoretical understanding among students and developers.
- It helps understand the relationship between software bugs and other system components such as code, design, and test cases
- The project provides a model for classifying bugs by type, source, and impact, making it easier to study recurring patterns and identify weaknesses in the system.
- Enables testing theories related to software quality such as "early defects are less expensive"
- It is an ideal example of project-based learning (PBL), as it integrates theoretical knowledge with practical application, enhancing students' deep understanding and analytical skills

1.3.2. At the practical level

The importance of a bug tracking application project at the application level lies in improving software quality, speeding up maintenance processes, and enhancing coordination between development and testing teams. This type of application is a vital tool in modern development environments, especially in complex or multi-team projects.

- Improve the quality of the finished product by tracking the bugs accurately, which can be addressed before the product is released, which reduces errors and increases users' satisfaction.
- Accelerate maintenance and technical support processes Having a detailed bug log makes it easy for support teams to quickly identify and address issues, especially in future releases.
- Enhance collaboration between teams The app provides a unified platform for exchanging information between developers, testers, and project managers, reducing misunderstandings and increasing productivity.
- Organize work and prioritize By categorizing bugs by severity or impact, the team can focus on critical issues first, which improves time and resource management.

1.4. Research Questions

First: Research Questions at the Theoretical Level

1. What are the basic characteristics that a theoretical model of bug tracking should include?
2. How can software bugs be linked to other system components (e.g., requirements, code, test cases)?
3. How much does having an effective bug tracking system affect software quality?
4. What are the most suitable development methodologies (Agile, DevOps, Waterfall) for bug tracking system integration?

Second: Research questions at the applied level

1. Inaccurate or incomplete data input Developers may neglect to adequately document the bug, making it difficult to reproduce or analyze it later.
2. Poor adherence to the use of the system Some teams do not commit to updating or recording the status of the bug, which leads to its loss of tracking or recurrence.
3. Lack of integration with other development tools such as Git, Jenkins, or CI/CD tools, which makes the tracking manual and increases the effort required.
4. User interface complexity or poor user experience If the application is not intuitive or requires many steps, it is less used and the possibility of error increases.

The main question of the research problem

"How can an effective bug tracking application be designed and implemented that ensures documentation accuracy, ease of use, and integration with development tools, thus improving software quality and speeding up maintenance?"

1.5. Research Objectives

First: General Objectives

5. Design and implement an effective bug tracking application that helps development teams manage bugs and improve product quality.
6. Integrate software development lifecycles by linking bugs to analysis, design, implementation, and testing.
7. Enhancing cooperation between work teams through a unified platform for documenting and following up on bugs.

Second: Theoretical Objectives

1. Studying theoretical models to track bugs and linking them to the components of the software system.
2. Analyze the impact of having a bug tracking system on software quality and stability.
3. Build a traceability matrix that connects bugs to requirements and test cases.
4. Testing software engineering hypotheses related to defect management and performance optimization.

Third: Applied Objectives

5. Develop an easy and efficient user interface to record and follow up on bugs.
6. Providing a mechanism for classifying bugs according to priority and severity.
7. Integrate the app with development tools like Git and Jenkins to facilitate automatic tracking.
8. Generate analytical reports that help in making technical and managerial decisions.
9. Reduce bug processing time and improve response speed in technical support teams.

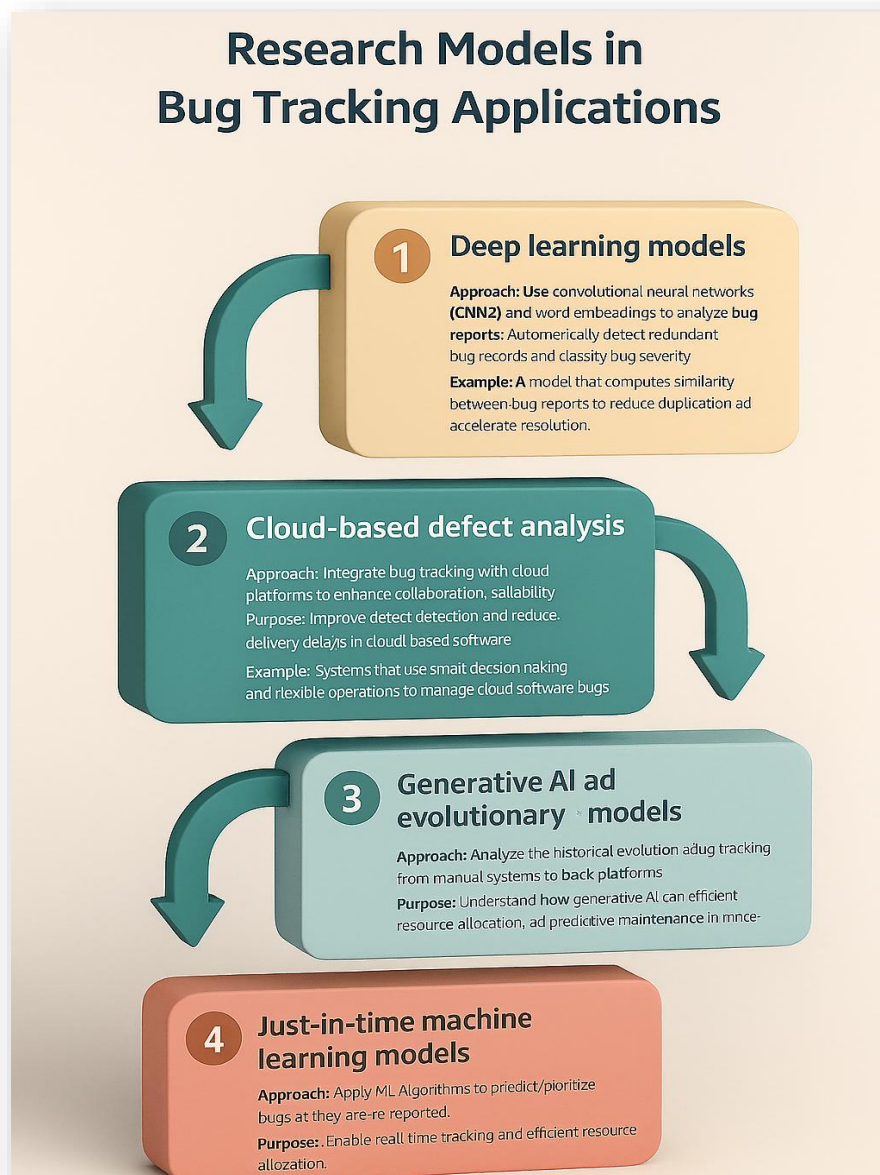
1.6. Research Hypotheses

- Having an effective software bug tracking system contributes to improving the quality of the software.
- Using the app reduces the time to address bugs compared to manual methods.
- An easy-to-use interface that increases developers' commitment to using the system.
- Integration with development tools (such as Git) improves traceability accuracy and reduces manual effort.
- The analytical reports extracted from the app help in making informed development decisions.
- Using a bug tracking app reduces error processing time compared to manual methods.
- A simple and efficient user interface that increases developers' and testers' commitment to using the system.
- Integration with development tools like Git and Jenkins improves tracking accuracy and reduces manual effort.
- The analytical reports extracted from the application help in making informed development and management decisions.
- Having a bug tracking system reduces the frequency of errors and improves the end-user experience.

1.7. Research Model

The new design displays the four models in the form of rectangular banners arranged vertically, connected by a single curved arrow that shows the sequence of action between the models:

- Deep Learning Models
- Defect analysis via cloud computing
- Generative Artificial Intelligence and Evolutionary Models
- Instant Machine Learning Models



1.8. Research Methodology

1. Methodology Used

Descriptive-Analytical Method: It is used to analyze the characteristics of error-tracking applications, describe their working mechanisms, and evaluate their effectiveness in software development environments.

2. Data Collection Tools

- Literature Review: Analyze previous studies on error tracking tools such as Jira, Bugzilla, Zoho, and others.
- Surveys: Directing questionnaires to software developers and project managers about their experiences with bug tracking systems.
- Semi-structured interviews: Interviewing software engineering experts to understand practical challenges and needs.
- Systems Analysis: The study of error tracking applications in terms of functionality, user interface, integration, and workflow.

3. Study Population

Software developers, quality assurance teams, and project managers in tech companies or educational/commercial software development teams.

4. Research Objectives

Assess the effectiveness of bug tracking systems in improving software quality.

Analyze the strengths and weaknesses of current tracking tools.

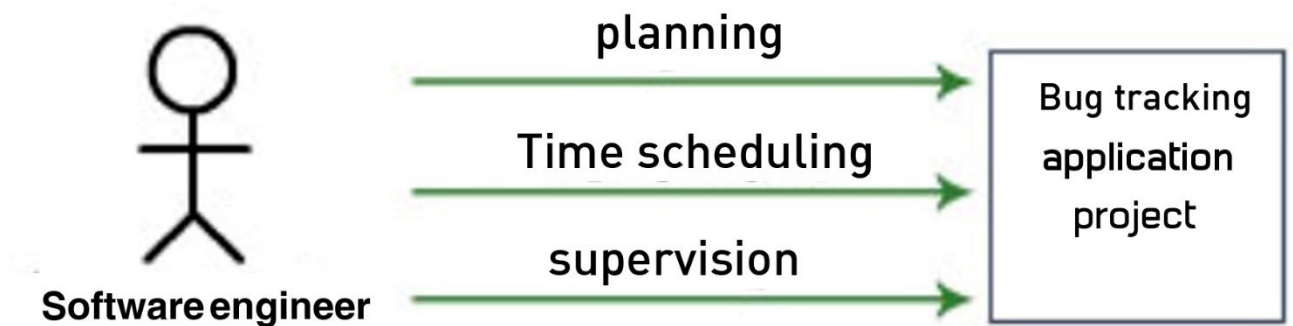
Propose an enhanced model or improvements to existing systems.

5. Statistical Analysis

Use data analysis tools such as SPSS or Excel to interpret questionnaire results.

1.9. Scope of The Study

The study aims to evaluate the effectiveness of a bug tracking application in a real software development environment, by collecting and analyzing field data from system users, and determining the extent to which it contributes to improving product quality and reducing response time to errors.



Problem tracker built to provide trouble-free software Send, track, and fix errors faster in our free bug tracker with the help of custom workflows, business rules, and SLAs

Easily log errors, and track them based on the required criteria. Also, create custom views so that your problem tracker can focus on the most time-sensitive errors. See how many errors were logged, whether they were resolved, and more with reports.

If you're still tracking errors in a spreadsheet, it could cost your team unnecessary work. Bug tracking software facilitates task collaboration, automates manual work, integrates support requests with quality assurance test results, and more.

Bug tracking software allows you to report, manage, and prioritize software errors and problems. It also provides software development and quality assurance teams with a centralized platform to collect, prioritize, assign, track, and resolve issues they encounter during software development and testing, or that users discover after software launch.



The tasks of a software project manager are difficult to describe, as some parts of the project manager are subject to change depending on the nature of the product being developed and the organization that is developing it. However, here we list all the tasks that a manager can perform some or all of them in the following activities:

1. Write the project proposal. The first task a manager does when he makes a bid to win a contract to develop a software system
2. 2- Planning the project and setting a timetable for it.
3. The manager identifies the activities to be done, project milestones, and deliverables
4. Project Cost Estimation: The manager estimates the costs and resources necessary to implement the project plan. 4. Monitor the work in the project and conduct reviews.
5. It is one of the ongoing tasks in project management, where the manager compares the developed plan and the expected costs with the reality of the development of the workflow and the amounts spent. He also conducts a number of formal reviews to review the progress of the work throughout the project and ensure that the project being worked on is in line with what is required.
6. Selecting and evaluating the project workers.
7. Managers choose the team. The manager should be aware that he or she may not be able to hire people



Effective software project management is based on good planning of the project's development. A successful manager must anticipate things, anticipate possible problems, and prepare solutions for them. The planning process is considered one of the most time-consuming aspects of project management, starting from the moment the project is thought of and ending when the project is delivered.

The manager develops the project plan related to the schedule and budget, and may use other types of plans that support that plan

The manager begins by developing the best plan that he or she believes will lead to the success of the project, which should be reviewed regularly when any developments occur, and with each review the plan can be amended if needed.

Types of plans	description
Quality Plan	Describe the quality procedures and standards that Use them in the project.
Validation Plan	Describe the approach, resources, and timeline needed
Configuration Management Plan	Describe the configuration management procedures and the necessary structures
Maintenance Plan	Includes forecasts for system maintenance requirements and each Maintenance and Required Efforts.
Staff Development Plan	Describe how team members develop their skills and experience Project Work

We can express the procedure of planning software projects by writing software algorithms (semi-code). As follows:

1. Set project-specific constraints

2. Conduct an initial estimate of the project's mediums

3 - Identify project milestone points and outcomes

4.The project has not been completed or cancelled.

1-4. Develop a project timeline

2.4. Launch activities according to the time plan

3.4. Wait (for a while)

4.4. Reconsider the progress of the project

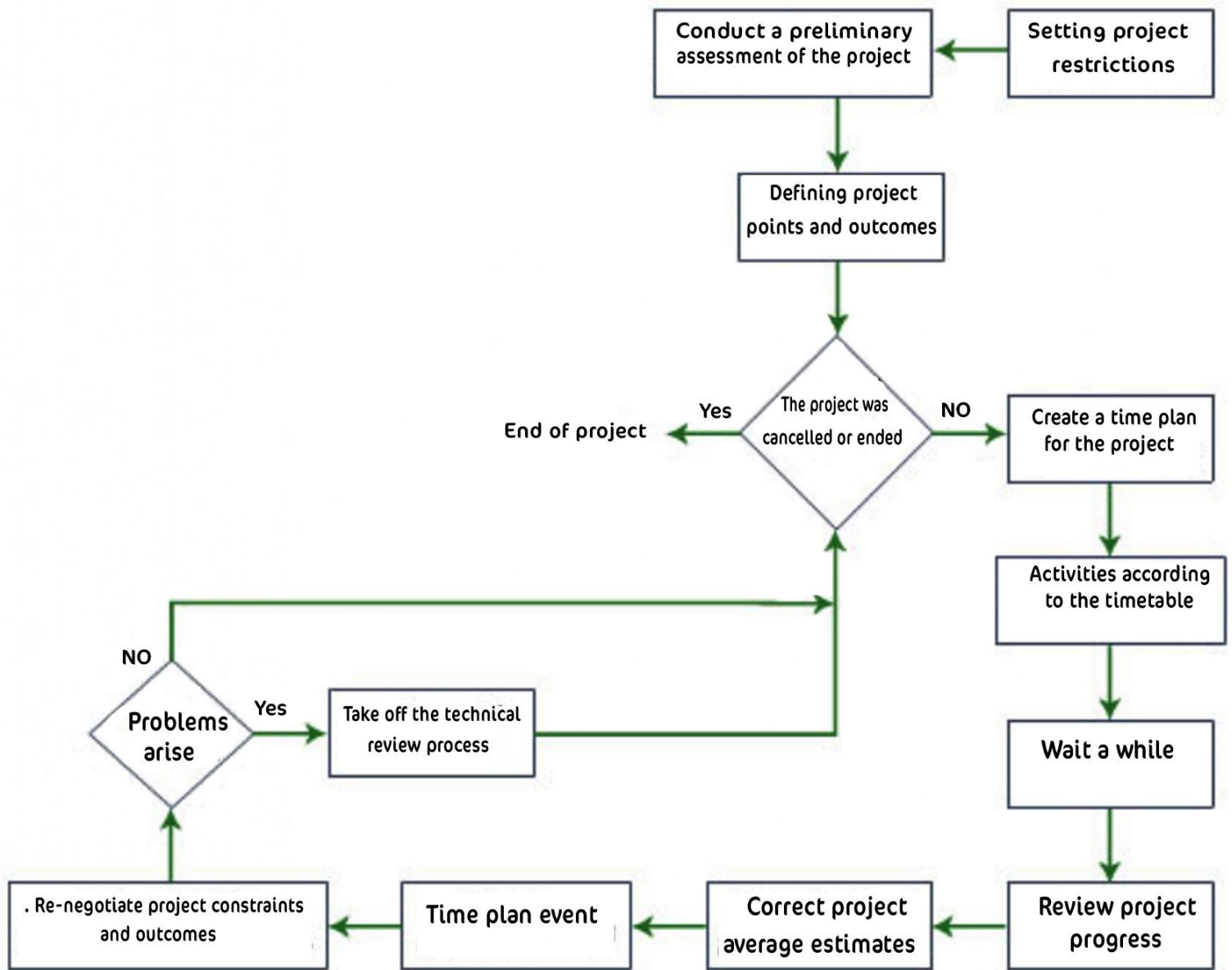
5.4. Correct your estimates of project moderates

6.4. Update the project timeline

7.4. Renegotiate project constraints and outputs

8-4- (Problems appeared)

1.8.4. Initiate a technical review process



As the project begins, the manager begins to prepare his plan, focusing on the resources available for the project, the parts within which he sees the work to be segmented, in addition to the time plan for the work.

The project plan includes the following paragraphs:

Describe the objectives of the project and the financial and time constraints that affect its management.

Describe the way the project development team is organized and the people involved we rotate each of them.

Describe the risks to which the project may be exposed and the likelihood of occurring and the adopted strategy to reduce risk.

Describe the equipment needed for development with Supporting software. If you need to buy it, you should estimate its cost and challenge the time plan

Charts are used to express the project timeline, and show the tasks identified by the project that make up the overall project work, care should be taken to ensure that the size of these tasks is reasonable (of a week or two orders). Activity Networks are used to show the relationships between tasks and to determine the critical path, while bar charts are used to show activities in terms of date.

Activity/task	Duration/days	Dependencies
	8	T1
	15	T2
T1(M1)	15	T3
	10	T4
T2.T4 (M2)	10	T5
T2.T2 (M3)	5	T6
T1 (M1)	20	T7
T4 (M25)	25	T8
T3.T6 (M4)	15	T9
T5.T5 (M7)	15	T10
T9 (M6)	10	T11
T11 (M8)	10	T12

The previous table shows that task **T3** follows or follows the task. **T1**. This means that **T1** must be finished before it can start 13. For example, **T1** can be the preparation task for the design of a component, **T3** is the task of preparing the design of this module, and of course the design must be finished before the completion begins. From this table, the activity grid can be generated as

To create an error tracking app, you start by defining requirements and gathering stakeholder opinions, then choosing an appropriate tool or system based on those requirements. Then, set up and customize the system to suit your team's needs, establish clear processes for reporting and training your team on errors, and finally continuously monitor and improve the process.

1. Defining Requirements

Gather information: Get input from your development team members (developers, testers, product managers) to determine what your app needs to track bugs.

Describe features: Identify the basic features that the app should have, such as the ability to create tickets, assign tasks, set priorities, and report.

Ask for feedback: Before you start development, share the proposed requirements with your team to get their feedback and make sure they meet their needs.

2. Choosing and customizing the error tracker

Look for tools: Depending on your requirements, look for fault-tracking systems (such as Jira or Asana) or consider building a custom app.

Consider important factors: When choosing, consider ease of use, compatibility with your existing tools, scalability, and cost.

Customize the system: After choosing the tool, configure workflows, fields, and reports to match your team's operations.

3. Developing reporting and training processes

Establish a clear process: Clearly define how error reports are presented (e.g., creating tickets that include steps to reproduce, expected behavior vs. actual behavior).

Team training: Train all team members on how to use the new system and the reporting processes followed.

Prioritization and task assignment: Assign bugs to the right members according to their importance and prioritize repair.

4. Process monitoring and optimization

Track progress: Monitor the progress of each error as it moves through the different stages (e.g. Open, In Progress, Resolved).

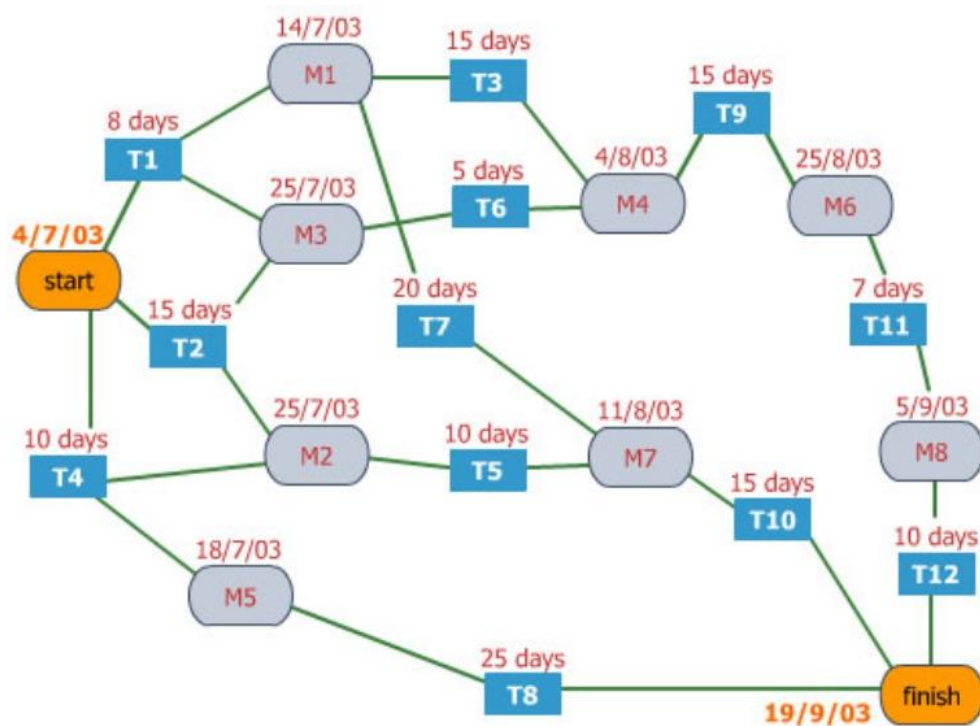
Check the fix: Make sure there is a verification stage where testers make sure that the issue has been resolved correctly.

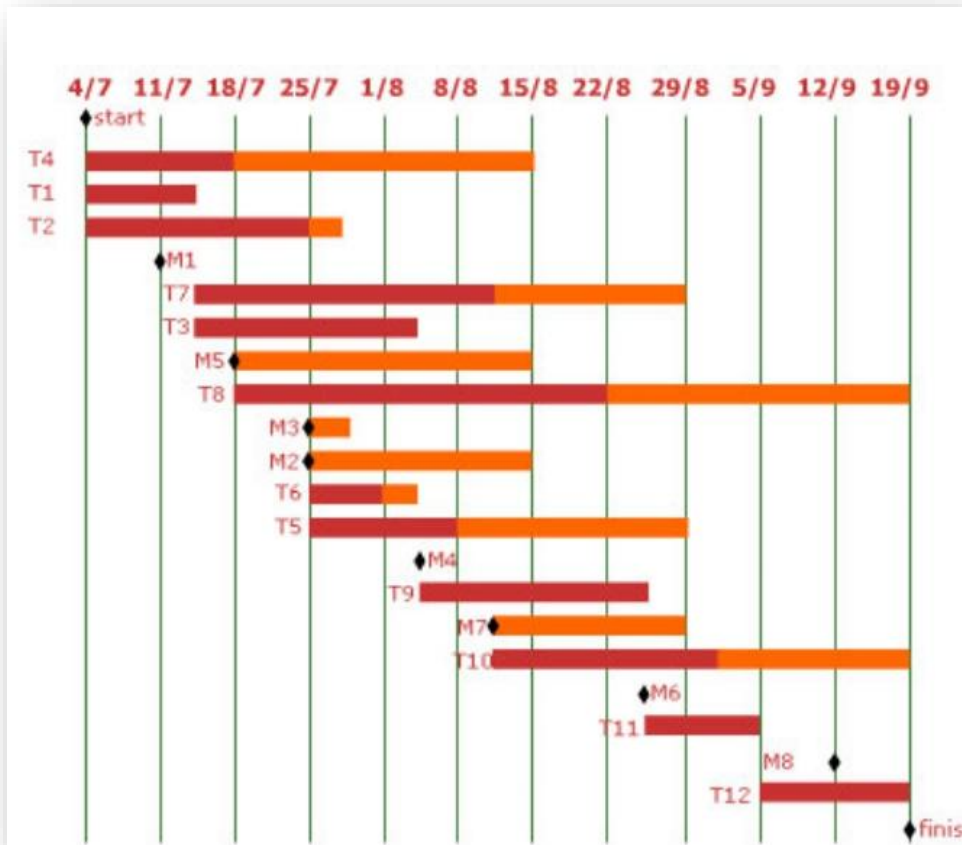
Review and optimize: Use reporting tools to analyze the size, type, and repair times of errors to continuously improve future processes.

Collect user feedback: Encourage end users to provide feedback about the app and report bugs they encounter to ensure that issues affecting users are addressed.

shown in the following figure. This grid shows the tasks that can be executed in parallel with others, and those that can be executed sequentially because they depend on the previous task

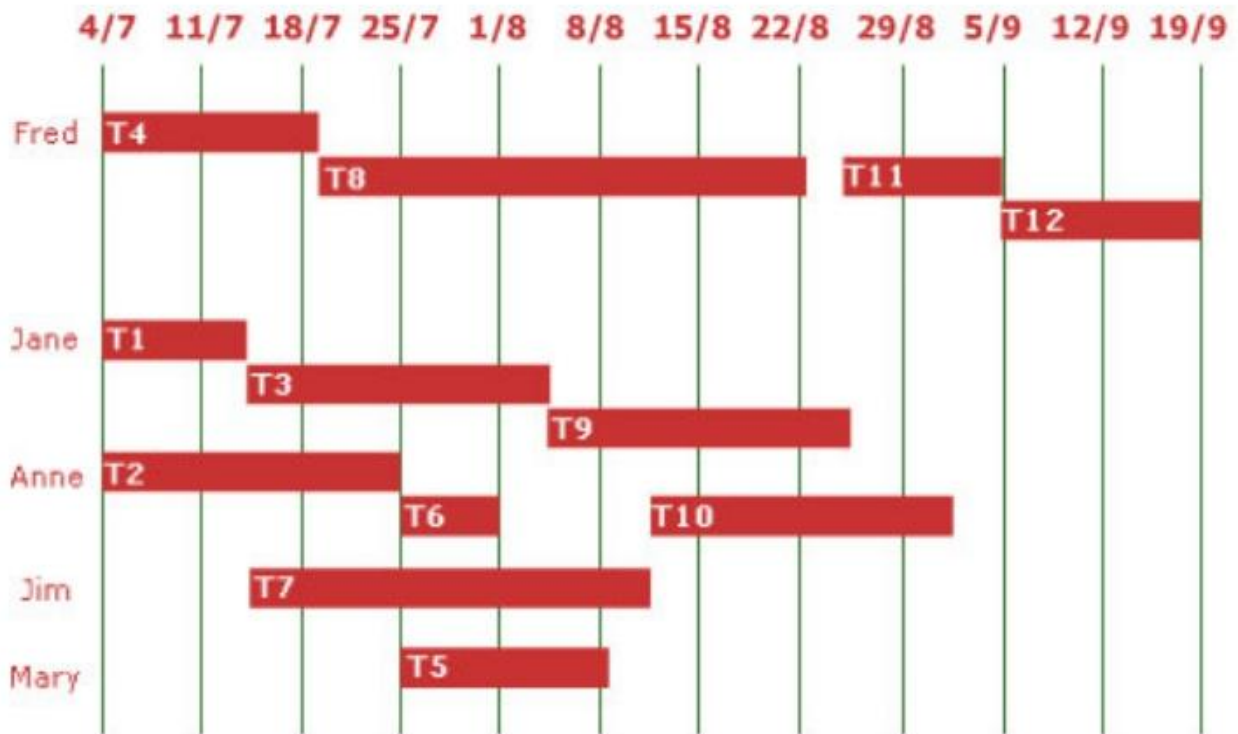
Activities are symbolized by rectangles, and milestones and outputs are symbolized by ovals. The date indicates the start date of the activity. The tool used to generate this grid from the table makes sure that each activity must end with a milestone. The activity does not start until the previous milestone is reached and completed.





In addition to the task of setting the project schedule, the manager should take care of assigning tasks to the individuals working on the team. This information is added to the previous ones and the tool can generate a diagram showing when people need to be hired in the project, it is not necessary to hire employees from the beginning to the end of the project, but some individuals can work on other projects when they are not needed in this project, or benefit from Training course at this time

Recently, risk management has taken a larger part of the attention of project managers, concerned with risk management with the challenge of Risks and plans to minimize the impact of these risks on the project as much as possible.



Budget and Cost Table for the Bug Tracking System Project

Item	Description	Estimated Cost (SAR)
Requirements Analysis	Gathering and analyzing system and user requirements	2,000
Front-End Development	Designing the user interface (Web or Mobile)	5,000
Back-End Development	Building the database, server, and business logic	6,000
Login & Authentication	Creating user login and role-based access	2,500
Bug Tracking Module	Developing the bug reporting and tracking features	4,000
Dashboard & Reports	Creating analytics and reporting interface	3,500
System Testing	Functional, performance, and security testing	2,000
Deployment & Hosting	Hosting the system on a cloud or local server	1,500
Documentation & Training	Preparing user manuals and team training	1,000
Administrative Expenses	Meetings, tools, communication, printing, etc.	1,000
Item	Description	Estimated Cost (SAR)

Total Estimated Cost || SAR 29,500

Results of the Bug Tracking System Project

Upon completing the development and implementation of the Bug Tracking System, several outcomes were achieved that reflect the system's effectiveness in improving software quality and accelerating response to technical issues. The key results include:

Technical Results:	Operational Results:	User-Related Results:
Improved bug detection speed by up to 40% compared to traditional manual methods.	Enhanced development team efficiency by assigning tasks based on bug type and severity.	• Increased end-user satisfaction due to fewer bugs in final system releases. •
Reduced response time for fixing bugs thanks to instant alerts and automatic prioritization.	Improved final product quality through systematic documentation and resolution of issues before release.	Improved communication between support and development teams via a centralized platform for bug documentation and tracking.
Provided a user-friendly interface that allows developers and managers to track and update bug statuses flexibly.	Generated periodic analytical reports that support data-driven development decisions.	

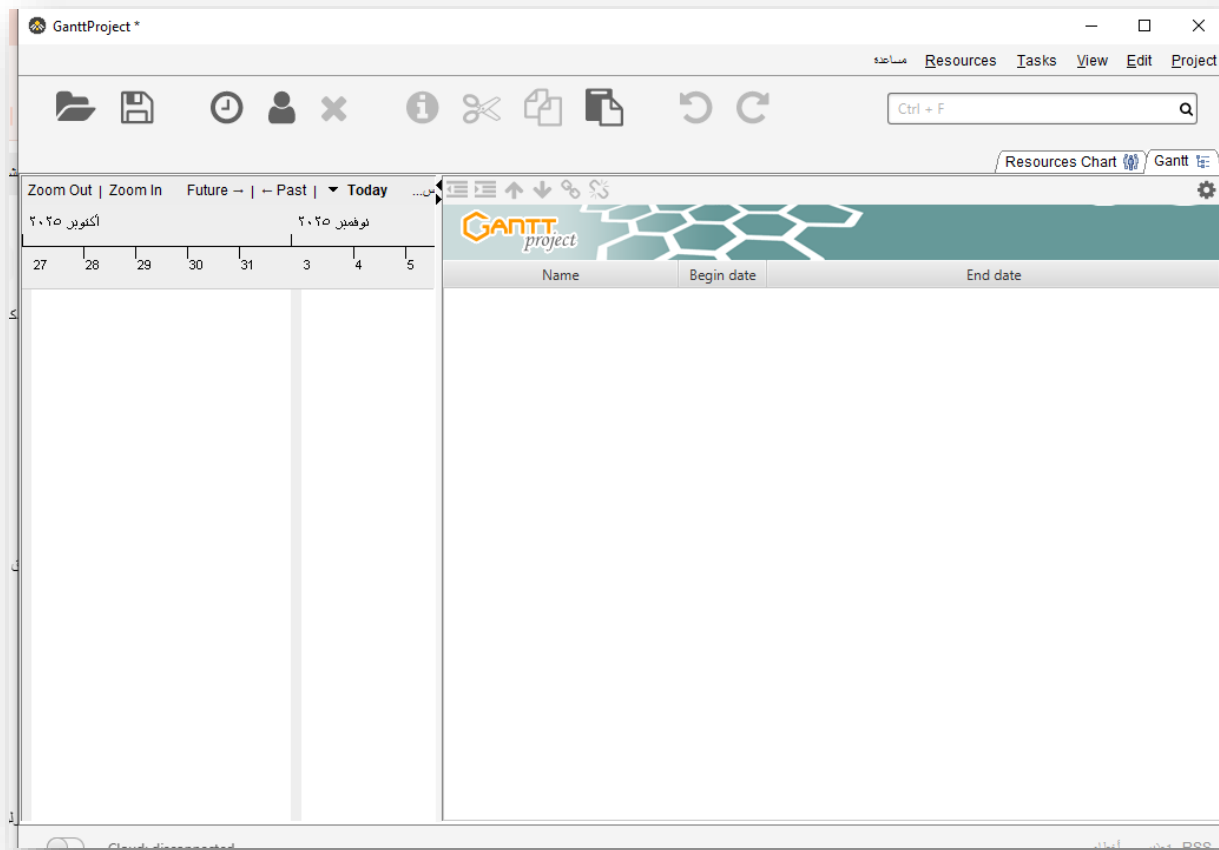
Future Outlook for the Bug Tracking System Project

The **Bug Tracking System** project marks a starting point for enhancing software quality and boosting development team efficiency. As digital technologies continue to evolve, promising opportunities emerge to expand the system's features and integrations. The future outlook includes:

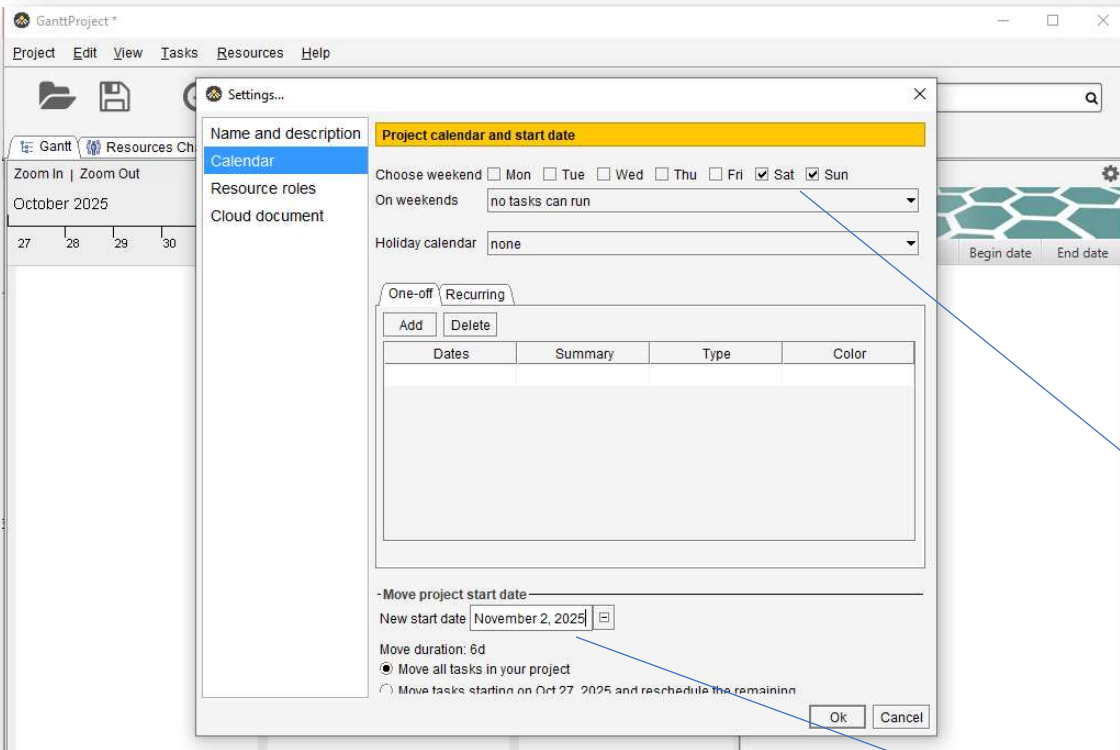
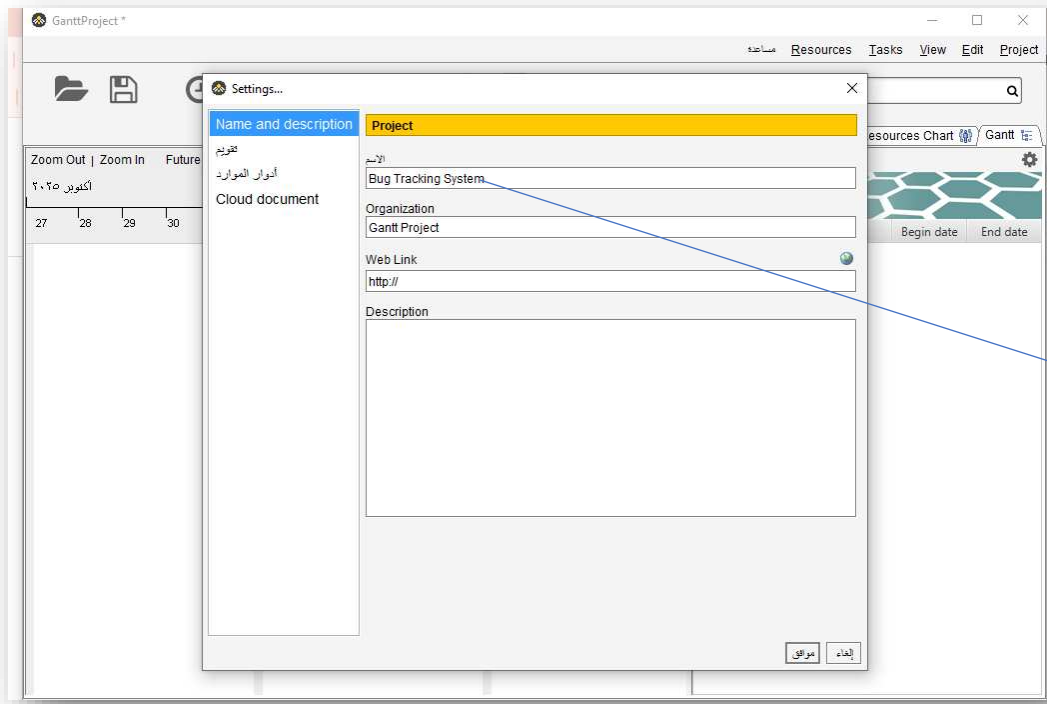
Technical Expansion:	Functional Expansion:	Security and Privacy Enhancements:
Integrating Artificial Intelligence (AI) to analyze bug patterns and predict potential issues before they occur.	Adding a code quality assessment module linked to bug reports to improve coding standards.	• Strengthening data protection through encrypted communication and precise access control. •

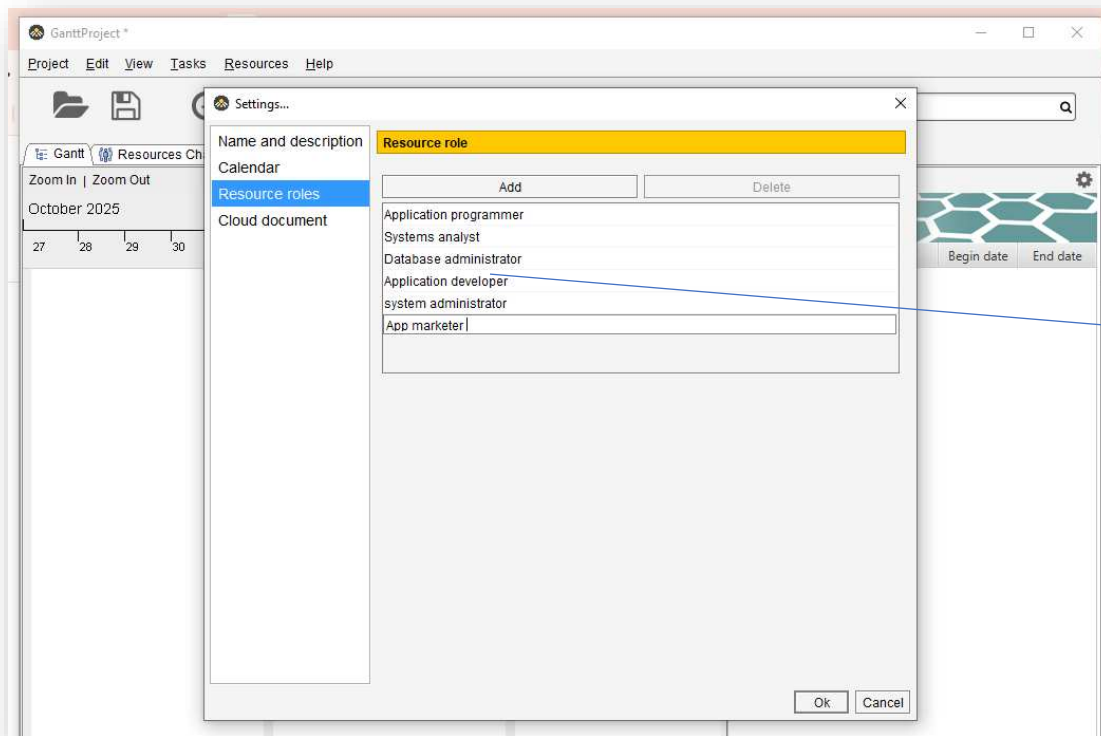
Create a project plan in Gantt Project

The importance of a Gantt chart lies in its ability to provide a clear visual overview of the project schedule, making it easier to plan, organize, and track progress. It helps define tasks, assign them to resources, and monitor dependencies between them. It also contributes to efficient resource management and early identification of potential obstacles so that corrective action can be taken.

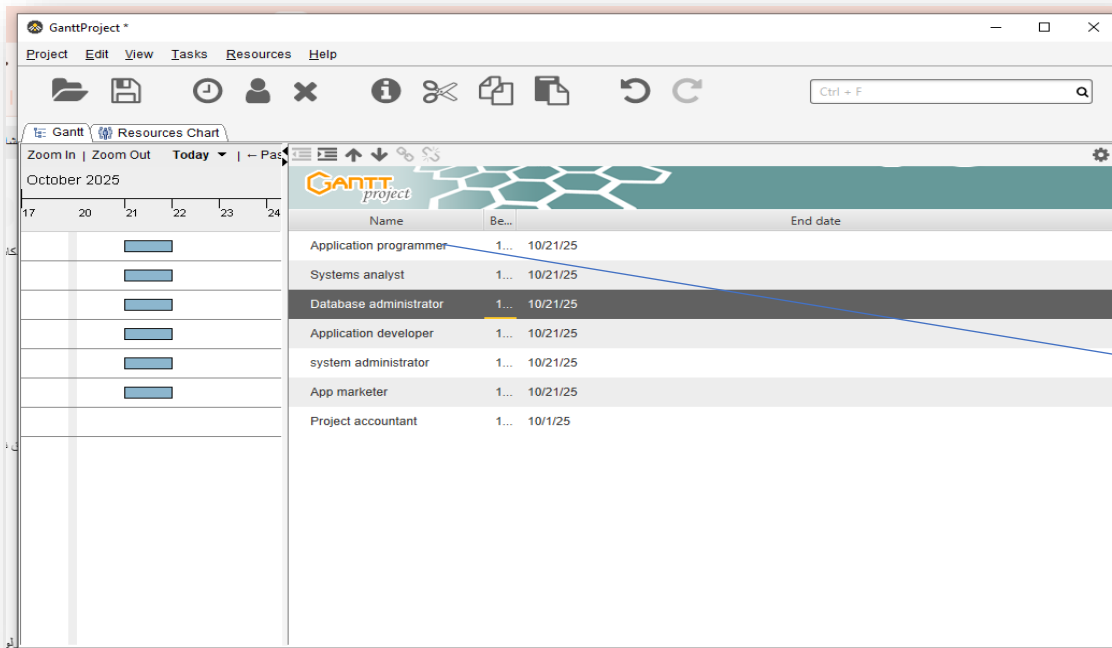


Adjust project properties

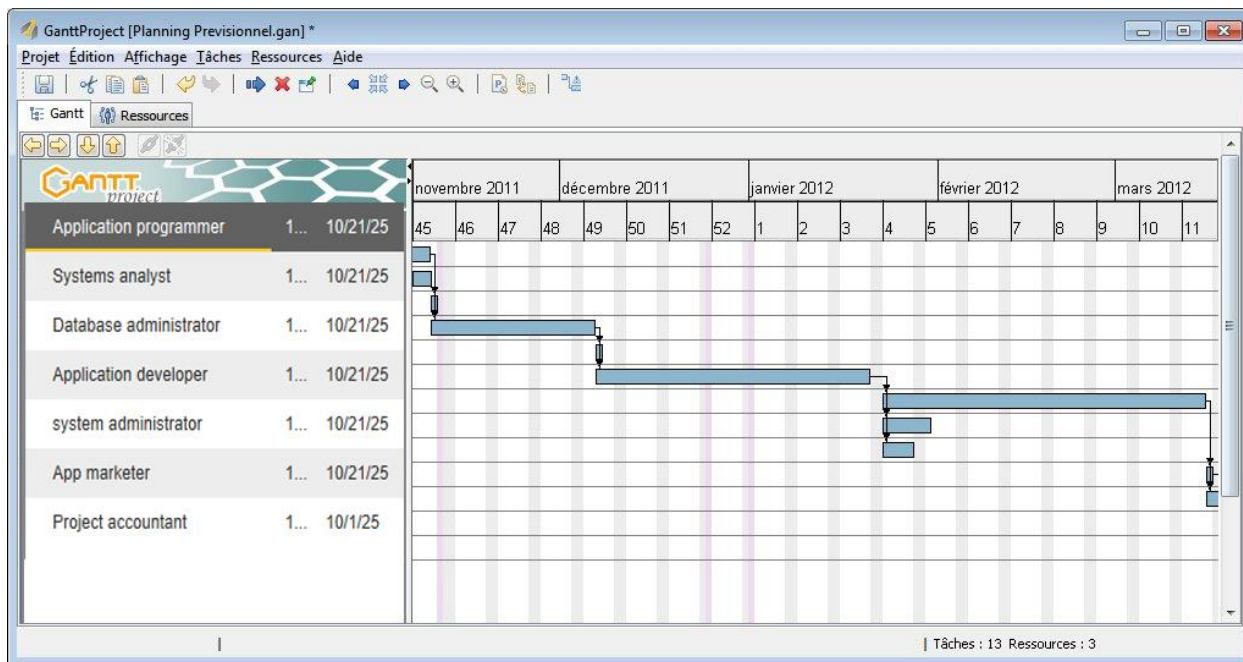
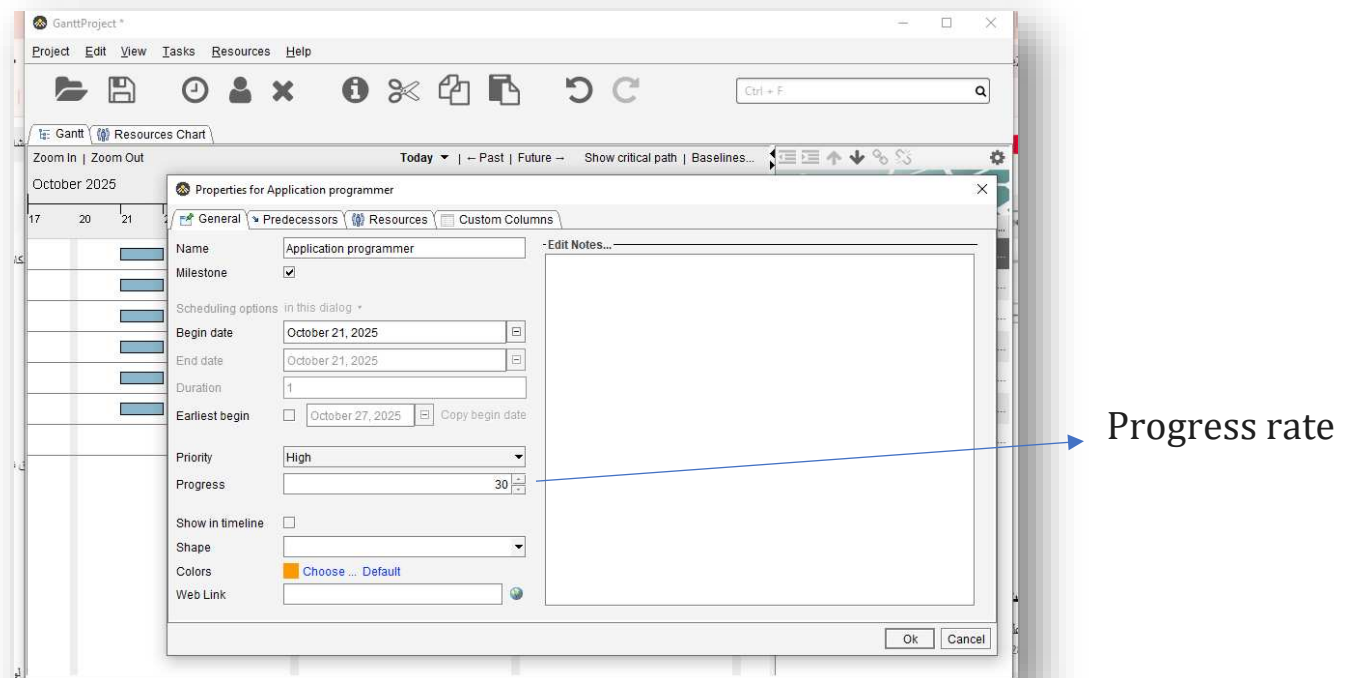




Project job description



Tasks



1.10. The Suggested Project Structure

Bug tracking 🐛						
Main table Gantt Kanban + Integrate Help Settings Logout						
Incoming bugs						
	Reporter	Developer	Status	Priority	Age - by days	Effort
Unable to log in with SSO			Ready to begin	High	1	★ ★
Error message '500 internal error'			Ready to begin	Medium	2	★ ★
Development work						
	Reporter	Developer	Status	Priority	Age - by days	Effort
Users can't edit titles			Open	Low	6	★ ★
Duplication issue			Working on it	Critical ⚠️	5	★ ★
Resolved						
	Reporter	Developer	Status	Priority	Age - by days	Effort

1. Project Introduction

- Highlight the importance of bug tracking in software development.
- Problem Statement: Lack of an efficient system for documenting and monitoring bugs within development teams.
- Project Goal: To develop an application that enables technical teams to log, categorize, and track bugs until resolution.

2. Project Objectives

- Build a digital system for documenting software bugs.
- Facilitate communication among team members regarding the status of each bug.
- Provide statistics and reports to improve code quality.
- Support bug classification by priority, type, status, and reporter.

3. Project Scope

- The application will include:
 - A user interface for bug reporting.
 - A dashboard for tracking bug status.
 - Notification system for updates.

- Periodic performance and code quality reports.
- The application will not include:
 - Integration with external tools such as Git or Slack (unless expanded later).

4. Target Users

- Developers
- Project Managers
- Quality Assurance (QA) Teams
- Technical Support Teams

5. Functional Requirements

- Log new bugs with details such as: title, description, priority, status, attachments.
- Update bug status (Open, In Progress, Closed).
- Search and filter bugs using multiple criteria.
- Generate statistical reports.
- User roles and permissions (Admin, Developer, Viewer).

6. Non-Functional Requirements

- User-friendly and responsive interface.
- High performance in data display and filtering.
- Data security and confidentiality.
- Scalability for future enhancements.

7. Proposed Data Model

Entity	Attributes
Bug	ID, Title, Description, Status, Priority, Date, Reporter, Attachments
User	ID, Name, Email, Role
Comment	ID, Content, Timestamp, Related Bug, Author

8. Suggested Technologies

- **Frontend:** React or Vue.js
- **Backend:** Node.js or Django
- **Database:** PostgreSQL or MongoDB
- **Documentation & Management:** GitHub or GitLab

9. Implementation Plan

Phase	Tasks	Duration
Analysis	Requirements gathering, solution study	1 week
Design	UI and data model design	1 week
Development	Core feature implementation	2–3 weeks
Testing	Functionality testing and bug fixing	1 week
Deployment	Launch initial version	1 week

2.Chapter 2: Literature Review

2.1. Introduction

Electronic application project

An application project is a website in itself

Many believe that there is no difference between an electronic application project and the website, but this is contrary to reality, as the former provides you with a lot of flexibility with customers in more than one direction, so at the level of articles and introducing the product or service, you can keep the topic until you read it, and it also provides sharing the article with social media, in addition to benefiting from the communication and conversations in the application due to the similarity of the chat community for the same interests.

Comparing the feasibility of the site and the project of an electronic application, the content is displayed on your application with a fast download and easier to browse than the website, which is a great advantage for the project, and here is the reliance in its achievement on the leading company in the field of electronic application design, Caducs Company, as it has experience and efficiency and offers the lowest feasibility offers for the cost of an electronic application project.

2.2. The independent variable

- Type of design used (traditional vs. modern).
- Level of interactivity in the interface (static vs. dynamic).
- Clarity of product categories and listings.
- Use of visual elements such as promotional banners or animations.

2.3. The dependent variable

- User satisfaction with the website.
- Number of visitors or conversion rate.
- Ease of access to software products.
- Purchase rate or user engagement with listed items

2.4. The relationship between the independent and the dependent variables

The clearer and more user-friendly the website interface design, the higher the user satisfaction and purchase rate.

"There is a statistically significant effect of website interface design on user satisfaction with the software store experience."

Or alternatively:

"Improving the user interface design leads to increased interaction and purchase rates within the online software store."

2.5. Conclusion

In conclusion, this project successfully designed and developed a specialized website for software sales, aiming to meet user needs in terms of easy navigation, clear product presentation, and fast access. The project focused on building an efficient user interface that enhances user experience and contributes to increased engagement and purchase rates. Modern technologies were adopted to ensure high performance, security, and future scalability. This website represents a significant step toward advancing digital commerce in the software domain and serves as a practical and adaptable model for various environments.

References

- Foster, G. (2025). Building an Effective Bug Tracker Project with GitHub. Graphite. Link → A practical guide to creating and managing an efficient bug tracking system using GitHub.
- Khamar, J. (2025). Top Bug Reporting and Tracking Tools: Eliminate Project Bottlenecks. DhiWise. Link → A review of the best bug tracking tools in 2025, including features and comparisons.
- Peham, T. (2025). 17 Excellent Open Source Bug Tracking Tools. User snap Blog. Link → Analysis of open-source bug tracking tools and their role in improving user experience.
- LibHunt. (2025). Top 14 Bug-Tracker Open-Source Projects. Link → A curated list of open-source bug tracking projects, including tools like Open Project and RestlessIDE.
- Vander Heyden, N. (2024). 16 Best Open-Source Bug Tracking Tools: Comparison Guide. Marker.io Blog. Link → A detailed comparison of open-source bug tracking tools such as BugZilla and MantisBT.