*Project Plan*

# Introduction

Welcome to our project aiming to revolutionize the used book industry through the fusion of traditional charm with modern digital convenience. Our primary objective is to create a user-friendly software platform that facilitates seamless exploration, discovery, and acquisition of books while fostering a vibrant community of book enthusiasts. Despite our ambitious vision, we acknowledge the constraints of budget and time that govern the management of this project. With limited resources, we are committed to delivering a high-quality solution that meets user needs within the specified timeframe.

# Project Organization

| **Position** | **Names** | **Description** |
| --- | --- | --- |
| Programmer | Aubrie Mclintrye | The programmer writes code for the program. |
| Programmer | Gautam Mehla | The programmer writes code for the program. |
| Tester | Michael Coughlin | The tester ensures that the program is free of errors by conducting repeated tests. |
| Designer | Perrin Brumfield | The designer ensures that the design is user-friendly and within the programmer’s capabilities. |
| Organizer | Leah Mattingly | The organizer helps the team stay organized and that all tasks are completed correctly and on time. |

# Risk Analysis

**Risk Identification**

| **Risk** | **Description** |
| --- | --- |
| Budget isn’t sufficient | The budget is too small for the project |
| Scope Creep | Additional features and requirements that are added |
| Overly Ambitious Scope | A challenging scope that can lead to project delays and even burnout. |
| Time Budget Insufficient | The schedule to build an application is too small |
| Missed Milestones | Weekly goals that are not met |
| Scalability | Application does not scale well |
| Bugs | Coding errors or logic flaws that impact the usability and user experience |
| Data Loss or Corruption | Accidental deletion or corruption that removes data and impacts the operation of the software |
| Poor performance | Hardware constraints and software bottlenecks that impact the performance of the software |
| Compatibility Issues | Incompatible between different platforms |
| Poor Documentation | Inadequate documentation that can lead to rework or delays in development tasks |
| Remote Work Problems | Time zone differences and digital communication software can exaggerate communication and collaboration challenges. |
| Lack of Technical Expertise | Inefficient expertise or experience can lead to delays or implementation errors |
| Burnout | Overwork and stress can result in decreased productivity, motivation, and quality of work |

**Risk Analysis**

| **Risk** | **Probability** | **Effect** |
| --- | --- | --- |
| Insufficient Budget | Moderate Likelihood | High - An insufficient budget would delay the project causing our team to fail. |
| Scope Creep | Moderate Likelihood | High - Scope creep can cause delays and misses to milestones. This could result in the team failing. |
| Overly Ambitious Scope | Moderate Likelihood | Moderate - If not reduced, an overly ambitious scope can cause milestone misses and project failure. |
| Time Budget Insufficient | High Likelihood | High - Insufficient time budget can increase the risk of turning the project in late or failing to finish the project. |
| Missed Milestones | Moderate Likelihood | Moderate - Missing milestones can cause delays. |
| Scalability | Low Likelihood | Low - Scalability issues can keep the application from growing. |
| Bugs | Moderate Likelihood | Moderate - Bugs and errors can affect usability and user experience. |
| Data Loss or Corruption | Low Likelihood | High - The loss of data or data corruption would reduce the application's overall functionality. |
| Poor performance | Low Likelihood | Moderate - The application's performance can affect the usability and user experience. |
| Compatibility Issues | Low Likelihood | Low - The compatibility of the application can affect usability for some users. |
| Poor Documentation | Moderate Likelihood | Moderate - Poor documentation can cause delays due to the effect it has on team members. |
| Remote Work Problems | Moderate Likelihood | Moderate - Remote work problems like different time zones can affect the efficiency and collaboration of the team. |
| Lack of Technical Expertise | Moderate Likelihood | Moderate - Lack of technical expertise can cause missed milestones and project delays. |
| Burnout | Moderate Likelihood | Moderate - Burnout could contribute to delays and could cause burnout in others by them picking up added tasks. |

**Risk Plans**

**Cost Risks** - Cost risks like *Insufficient Budget* will be mitigated by prioritizing requirements. Team members will focus on requirements of the highest priority first while maintaining the ability to reduce an overly ambitious scope.

**Schedule Risks** - Schedule risks like *Time Budget Insufficient* will be mitigated through the use of weekly standups and ongoing communication. The project will be monitored for adherence to the project plan and action will be taken if needed.

**Performance Risks -** Performance risks like *Scalability Issues* will be mitigated through the use of testing. Team members will provide feedback and respond accordingly.

**Operational Risks -** Operational risks like *Data Loss or Corruption* and *Bugs* will be mitigated through backup methods and testing. Team members will maintain good documentation and provide feedback.

**Technological Risks** - Technological risks like *Compatibility Issues* and *Poor Performance* will be mitigated through testing and quality assurance. The project will also respond to feedback from team members to reduce the risk.

**Communication Risks -** Communication risks like *Remote Work Issues* will be mitigated through consistent communication through Discord and Zoom. Team members will also participate in meetings to reduce the risks.

**Scope Creep Risks** - Scope Creep Risks like *Scope Creep* and *Overly Ambitious Scope* will be mitigated by defining clear objectives and consistent communication.

**Skills Resource Risks** - Resource risks like *Lack of Technical Expertise* and *Burnout* will be mitigated through knowledge sharing and managed workloads. Team members will be encouraged to reach out on issues that they are having.

# Hardware and Software Requirements

Hardware Requirements:

* Desktop or laptop

Software Requirements:

* Operating System: Windows 10 or later or MacOS
* Development Environment: Python for backend development, C# for frontend development
* Integrated Development Environment (IDE): Visual Studio for C# development, and IDEs like PyCharm or Visual Studio Code for Python development

Development Environment:

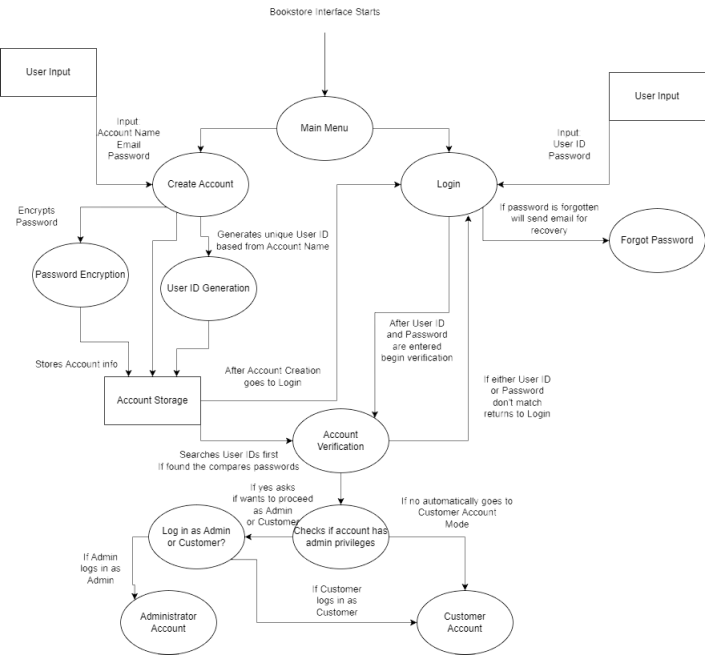
The development environment will consist of the following components:

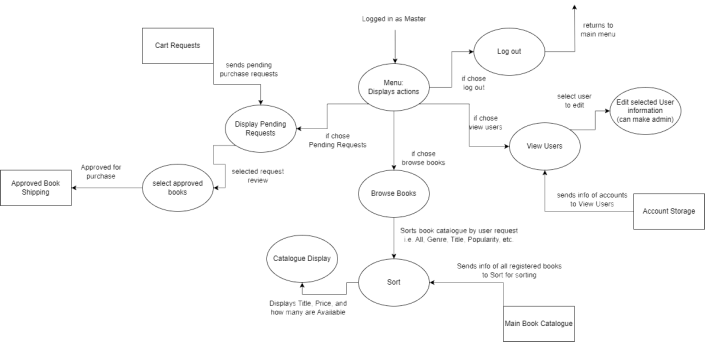
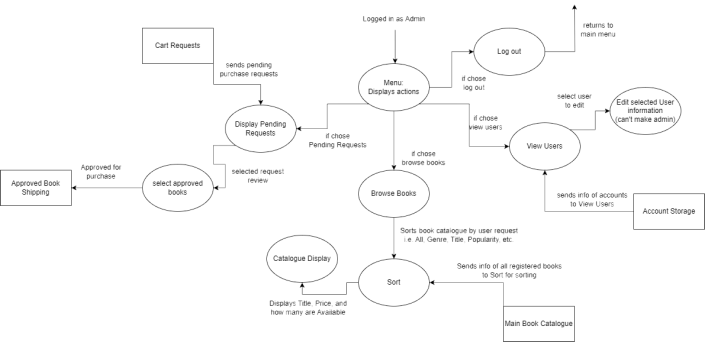
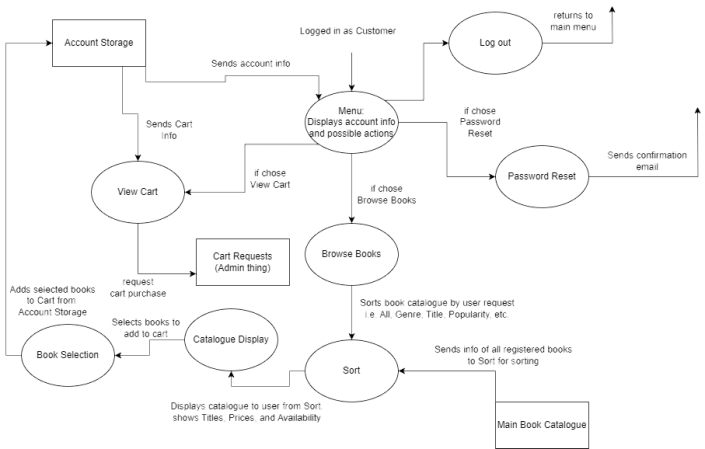
* Python: Utilized for backend development, including data processing, business logic implementation, and database interaction.
* C#: Employed for front-end development, including user interface design and implementation.
* Visual Studio: Used as the primary IDE for C# development, offering features such as code editing, debugging, and project management.

# Work Breakdown

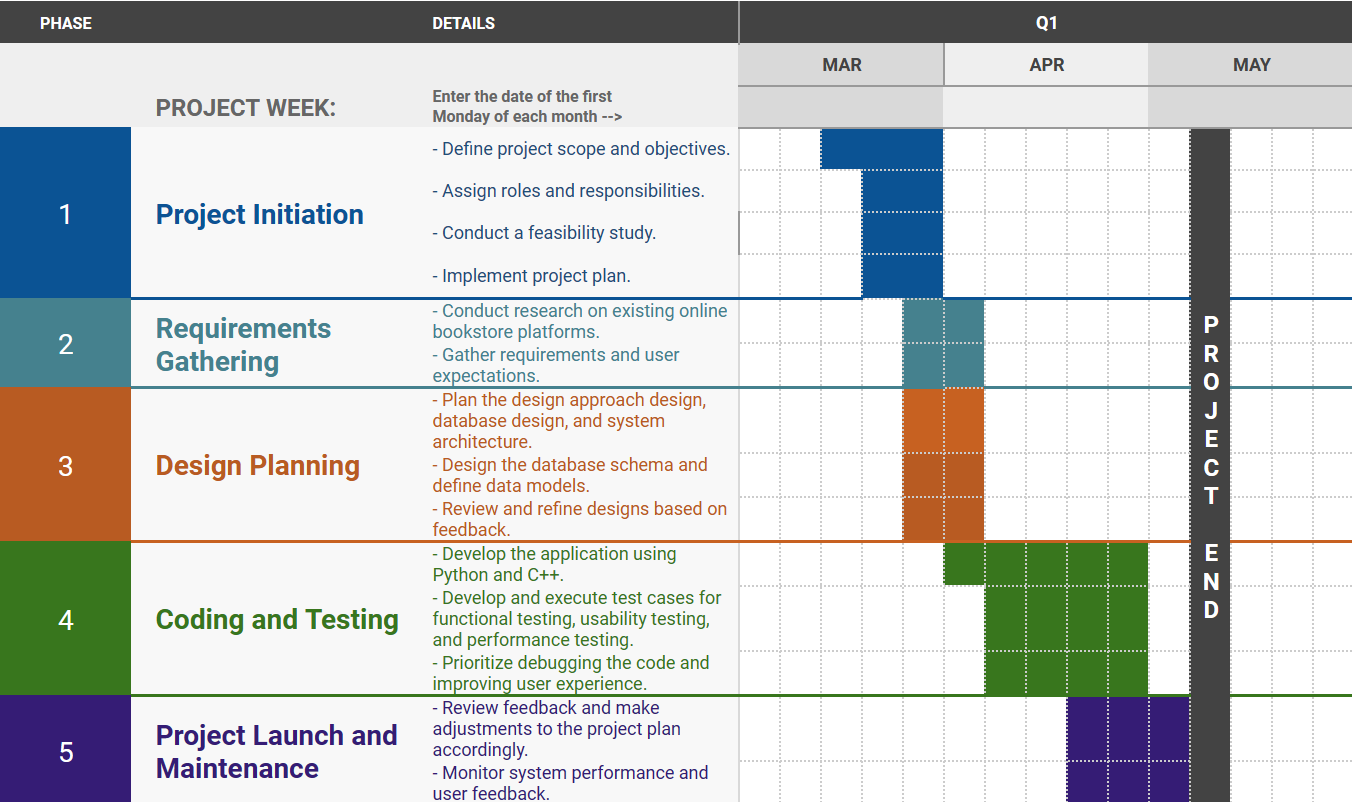
* Requirements Gathering:
  + 1.1 Conduct Stakeholder Interviews
  + 1.2 Document Functional Requirements
  + 1.3 Document Non-Functional Requirements
* System Design:
  + 2.1 Design Database Schema
  + 2.2 Design User Interface
  + 2.3 Define Software Architecture
* Development:
  + 3.1 Backend Development
    - 3.1.1 Implement Database Integration
    - 3.1.2 Develop Business Logic
    - 3.1.3 Create APIs
  + 3.2 Frontend Development
    - 3.2.1 Develop User Interface Components
    - 3.2.2 Implement Interaction with Backend Services
  + 3.3 Integration
    - 3.3.1 Integrate Backend and Frontend Components
    - 3.3.2 Ensure Communication between Components
* Testing:
  + 4.1 Unit Testing
    - 4.1.1 Test Backend Components
    - 4.1.2 Test Frontend Components
  + 4.2 Integration Testing
    - 4.2.1 Verify Interaction between Modules
  + 4.3 User Acceptance Testing
    - 4.3.1 Validate Stakeholder Expectations
* Deployment and Maintenance:
  + 5.1 Deployment
    - 5.1.1 Deploy Software to Production Environment
  + 5.2 Training
    - 5.2.1 Provide Staff Training on Software Usage
  + 5.3 Maintenance
    - 5.3.1 Establish Procedures for Ongoing Maintenance and Support
    - 5.3.2 Plan for Future Updates and Upgrades

# Process Flow Diagram





# Project Schedule



[Team 6 Project timeline](https://docs.google.com/spreadsheets/d/1BskpBdgo1avmooI69Ztv0GWbUlOleNH_mz60OPpL9S8/edit?usp=sharing)

Phase 1: Project Initiation

1. Define project scope and objectives.
2. Assign roles and responsibilities.
3. Conduct a feasibility study.
4. Implement project plan.

Phase 2: Requirements Gathering

1. Conduct research on existing online bookstore platforms.
2. Gather requirements and user expectations.

Phase 3: Design Planning

1. Plan the design approach design, database design, and system architecture.
2. Design the database schema and define data models.
3. Review and refine designs based on feedback.

Phase 4: Coding and Testing

1. Develop the application using Python and C++.
2. Develop and execute test cases for functional testing, usability testing, and performance testing.
3. Prioritize debugging the code and improving user experience.

Phase 5: Provide Software Solutions

1. Review feedback and make adjustments to the project plan accordingly.
2. Monitor system performance and user feedback.

Update software as needed.

# Monitoring and Reporting Mechanisms

#### Documentation Management:

All project documents, including requirements, design specifications, user manuals, and meeting minutes, will be shared and maintained on Google Drive. Google Drive offers collaborative document editing, version history, and easy accessibility, facilitating efficient document management and sharing.

#### Code Repository Management:

All project code will be maintained and shared through GitHub repositories. GitHub provides version control, collaboration features, and issue tracking, facilitating efficient code development and management.

Communication through Discord and Zoom will facilitate collaboration, feedback exchange, and decision-making.

# Appendix

| Activity  Number | Activity  Description | Estimated Time  (in days) | Dependencies |
| --- | --- | --- | --- |
| 1.1 | Conduct  Stakeholder Interviews | 3 | None |
| 1.2 | Document  Functional Requirements | 2 | 1.1 |
| 1.3 | Document  Non-Functional Requirements | 2 | 1.1 |
| 2.1 | Design  Database Schema | 3 | 1.2,  1.3 |
| 2.2 | Design  User Interface | 4 | 1.2,  1.3 |
| 2.3 | Define  Software Architecture | 3 | 1.2,  1.3 |
| 3.1 | Backend  Development | 12 | 2.1,  2.3 |
| 3.1.1 | Implement  Database Integration | 3 | 2.1 |
| 3.1.2 | Develop  Business Logic | 4 | 2.1 |
| 3.1.3 | Create  APIs | 3 | 2.1 |
| 3.2 | Frontend  Development | 8 | 2.2,  2.3 |
| 3.2.1 | Develop  User Interface Components | 4 | 2.2 |
| 3.2.2 | Implement  Interaction with Backend Services | 4 | 2.2 |
| 3.3 | Integration | 5 | 3.1,  3.2 |
| 3.3.1 | Integrate  Backend and Frontend Components | 3 | 3.1,  3.2 |
| 3.3.2 | Ensure  Communication between Components | 2 | 3.1,  3.2 |
| 4.1 | Unit  Testing | 4 | 3.1,  3.2 |
| 4.1.1 | Test  Backend Components | 2 | 3.1 |
| 4.1.2 | Test  Frontend Components | 2 | 3.2 |
| 4.2 | Integration  Testing | 3 | 3.3 |
| 4.2.1 | Verify  Interaction between Modules | 3 | 3.3 |
| 4.3 | User  Acceptance Testing | 4 | 4.1,  4.2 |
| 4.3.1 | Validate  Stakeholder Expectations | 4 | 4.1,  4.2 |
| 5.1 | Deployment | 3 | 3.3,  4.3 |
| 5.1.1 | Deploy  Software to Production Environment | 2 | 5.1 |
| 5.2 | Training | 2 | 5.1 |
| 5.2.1 | Provide  Staff Training on Software Usage | 2 | 5.1 |
| 5.3 | Maintenance | 5 | 5.1,  4.3 |
| 5.3.1 | Establish  Procedures for Ongoing Maintenance and Support | 3 | 5.3 |
| 5.3.2 | Plan  for Future Updates and Upgrades | 2 | 5.3 |