BOOKHIVE LIBRARY MANAGEMENT SYSTEM (LMS)

WWW Structures, Techniques and Standards
CSI 3140

Requirements
Phase 3

Team #33

Student Number	Name	Email
300163562	Lara Lim	llim067@uottawa.ca
300173889	Amani Farid	afari094@uottawa.ca
300184721	Samy Touabi	stoua026@uottawa.ca
300098986	Kian Zahrai	kzahr091@uottawa.ca

Date Submitted: July 23rd, 2023
Professor: Khalil Abuosba

Introduction

The term project for the WWW Structures, Techniques, & Standards class is to develop a system software to maintain a library management system. The system is designed to computerize the operations performed over the information about the members, book issues and returns and all other operations. This type of software is to be used by educational institutions' libraries. The software allows library members to browse, reserve, and loan books whereas library staff employ it to manage the acquisition, cataloging, and inventory of books. The administrator manages all library members accounts while also managing library staff accounts of the library branch.

The purpose of this document is to entail the selected project by expanding upon the project-specific deliverables with a focus on common design features, including modularity, reusability, architectural system design, and a standardized and secure design. The report is divided into two sections, with the first providing an overview to the project development methodologies and the business process designed to achieve the objectives of process automation, process enhancement, or process reengineering of the selected context. The second section entails any updates or progress reports that were needed to be addressed, providing any insight into what is happening within the project development.

Phase 3: The Development Process

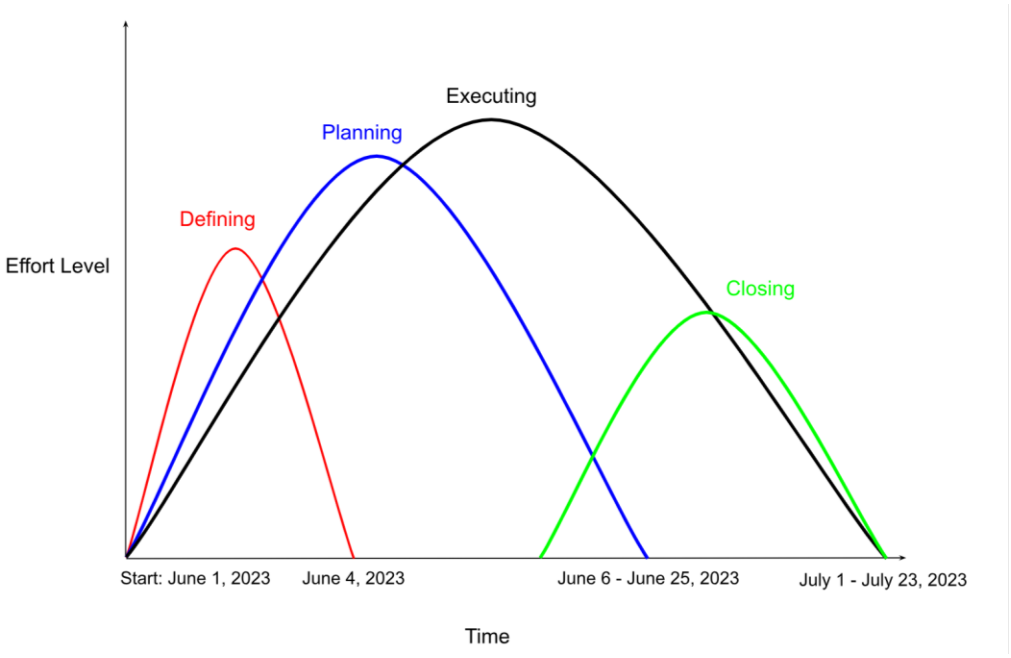
Each stage in the development process is crucial for the successful implementation of a multitier architecture project, especially for a library management system. It ensures that the system meets the requirements, is robust, and provides a seamless user experience. In addition, collaboration and communication among the development team, stakeholders, and end-users are crucial throughout the entire process to ensure the successful delivery of the library management system. After passing the primary stages which involve gathering requirements, analyzing ideas, and comparing designs, the team can start the development stage, integrating solutions in coordination, and setting up the roadmap to the final product. It is imperative the team will follow agile software development methodologies to enable rapid iterations, continuous improvement, and customer-centric product development. The development team will utilize modern programming languages, frameworks, and libraries to create intuitive and scalable software solution. Version control systems and collaboration tools will be implemented to facilitate efficient code management and seamless teamwork. The software products will be deployed through cloud-based platforms, enabling scalability, accessibility, and seamless updates. Regular maintenance and updates will be conducted to ensure optimal performance, security, and compatibility with evolving technologies.

With any system to be built, it is important to think of the system design from an architectural standpoint. A software architecture introduces constraints on implementation and restricts design choices. This reduces the complexity of a software system and prevents developers from making incorrect decisions. For instance, a Multi-tier Architecture is a software

architecture in which different software components, organized in tiers (layers), provide dedicated functionality. It is suitable to support enterprise-level Client-Server applications by providing solutions to scalability, security, fault tolerance, reusability, and maintainability. It helps developers to create flexible and reusable applications. To be specific, it is an architecture in which presentation, application processing, and data management functions are physically separated. The structure depicts a pattern in which the user interface (presentation tier), functional process logic (logic or application tier), computer data storage, and data access (data tier) are developed and maintained as independent modules, on separate platforms. Regarding this project, it is constructed to form a 3-tier architecture:

The **presentation** tier is the front end layer in the 3-tier system and consists of the user interface. This user interface is a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is to built on web technologies, including a combination of HTML/CSS and PHP, and communicate with others layers through API calls. The **logic** or **application** tier contains the functional business logic which drives the application's core capabilities. The logical tier is pulled out from the presentation tier and, as its own layer, it controls the application's functionality by performing detailed processing. For this software application, it is written in Python. Lastly, the **data** tier comprises of the database/data storage system and data access layer. This compartment of the application is to be supplemented by MySQL database system, for which its data is accessed by the application layer via the API calls.

Current projection of the development process:



Defining a development process involves breaking down the identified tasks from the business point of view, to build a new product from ideation all the way through launch. Having moved past the idea generation and definition stages, we move onto the product development stages, and with the help of an organized cross-departmental collaboration, the product becomes easier to manage and curate. At this stage, we first need to meet the **Executing** category of this process, supported with a list of status report, changes, qualities and forecasts. By incorporating these aspects into the execution phase of the project, our team can effectively monitor progress, manage changes, ensure quality, and forecast future needs. This helps in maintaining project control, meeting deadlines, and delivering a successful Library Management System.

Status Reports:

- Regularly provide status reports to track the progress of the project.
- Report on completed tasks, ongoing activities, and any potential delays or issues.
- Highlight key milestones achieved and upcoming milestones to stakeholders and project sponsors.
- Include information on resource utilization, budget expenditures, and overall project health.

Changes:

- Track and manage changes throughout the project lifecycle.
- Establish a change control process to evaluate and approve proposed changes.
- Document change requests, including the reason for the change, impact analysis, and potential risks.
- Assess the feasibility, cost, and time implications of each change.
- Obtain necessary approvals before implementing approved changes.

Quality:

- Implement a comprehensive quality assurance process to ensure the delivery of a high-quality LMS.
- Conduct thorough testing and quality checks at each phase of the development lifecycle.
- Define and adhere to coding standards, design guidelines, and best practices.
- Perform regular code reviews to identify and address any potential issues or code violations.
- Implement automated testing, unit testing, integration testing, and user acceptance testing to validate system functionality and performance.

Forecasts:

- Forecast resource requirements for each project phase, considering development, testing, documentation, and support needs.
- Estimate the effort and time required for future tasks and milestones.
- Consider the impact of potential risks, changes, and dependencies on the project timeline and resource allocation.

- Update forecasts based on actual progress and adjustments made during the project execution.
- Communicate forecasts to stakeholders to manage expectations and ensure alignment with project goals.

After meeting the **Executing** category, the next step is the **Closing** category. **Closing** in development processes refers to the completion of deliverables as per the objectives of time and cost. However, in addition to the assurance that all the work has been completed, it also argues for the assurance that all agreed upon project management processes have been executed and showcases the formal recognition of the completion of a project. It serves an important purpose for the organization and helps it avoid unfavorable and adverse scenarios. By completing these activities in the closing phase, the team ensures a smooth transition of the LMS to the library's operational environment. It facilitates user training, knowledge transfer, evaluation, and the capture of lessons learned to drive continuous improvement and future success.

Train Customer:

- Conduct training sessions for the library staff and administrators on how to use the LMS effectively.
- Provide comprehensive documentation, user manuals, and training materials.
- Train users on system functionalities, such as item management, user management, and reporting.
- Address any questions or concerns raised during the training sessions.
- Ensure users are confident and proficient in utilizing the LMS.

Transfer Documents:

- Hand over all project-related documents, including requirements, design specifications, test plans, and system documentation, to the library's designated stakeholders.
- Ensure the documents are organized, accessible, and easily understandable.
- Provide guidelines on how to access and maintain the documents for future reference.
- Ensure the transfer of ownership for any third-party software licenses or subscriptions used during the project.

Release Resources:

- Identify and release project resources, such as hardware, software licenses, and development tools, that are no longer required.
- Properly document and track the release of resources.
- Coordinate with relevant departments to handle the disposal or repurposing of any unused resources.

Evaluation:

- Conduct a post-implementation review to evaluate the success of the LMS project.
- Assess the achievement of project goals, adherence to the project plan, and overall user satisfaction.

- Gather feedback from library staff, administrators, and end users to identify areas of improvement.
- Evaluate the performance, stability, and scalability of the LMS in a live environment.
- Identify any lessons learned, challenges faced, and success factors to inform future projects.

Lessons Learned:

- Document lessons learned from the LMS project, capturing both successes and areas for improvement.
- Identify best practices, methodologies, and strategies that worked well during the project.
- Document challenges, risks, and issues encountered, along with their resolutions.
- Share the lessons learned with the project team and relevant stakeholders to facilitate knowledge sharing and continuous improvement.
- Incorporate the lessons learned into future projects to enhance project management practices.

By this time, our team has met all the stages of the application development process, which involved meeting the business process and development process objectives. With the business process providing the idea behind the application, the development process proved the actionable roadmap into the implementation of the application. these preparations and achievements, the next phase of development. To discuss the specific objectives and requirements regarding the project, the updated SRS document shall provide thorough details on the different aspects of the project.

I Project Overview

A. Project Scope Statement

The project is a Library Management System that seeks to streamline the key processes of a multi-branch library through system distribution and automation. The main components of the project are its item checkout, account management, inventory management, and book recommendation features. The project's item checkout, account management, and inventory management features will be improvements on current library processes through the unification of services and overall usability. The book recommendation feature is introduced as a new option which will increase interactions with libraries and reading material. An online payment system for late fees or lost items which would involve banking information is a possible part of the project. Its inclusion will be determined based on further evaluation of constraints and stakeholder feedback. In contrast, features outside of the scope of the project are considered peripheral and will not be developed. Book ordering mechanisms for acquiring the materials themselves are also considered outside of the scope. Services to facilitate software migration from external systems is not within the project scope.

The project timeline will adhere to the four-phase schedule with completion dates by June 6, July 3, July 22, and July 23 of 2023. The deliverables for the phases will address planning, implementation, demonstration, and formal documentation respectively. Agile will be used to facilitate timeliness. The cost associated with the system will be minimal to non-existent, as initial development will include economical hosting options for the database. The performance aspect of the system will align with that of other modern applications; specific details are provided in Section 5.1 of the accompanying Software Requirements Specification. Due to the minimal cost, the project's development will primarily balance its timing and performance requirements.

The software development of the Library Management System (LMS) follows an Object-Oriented Programming (OOP) paradigm, which promotes modularity and reusability of code. The system is designed with a multilayered architecture, specifically a Client/Server Architecture, to separate the presentation layer from the business logic and data access layers. This architectural design enhances scalability and maintainability. The development methodology adopted for this project is Agile/Scrum, allowing for iterative and incremental development, frequent feedback from stakeholders, and quick adaptation to changing requirements. Throughout the development process, standardized design principles and coding practices are followed to ensure consistency and code quality. Security is a top priority, and the system is planned to implement robust security measures, including user authentication, access control, and data encryption, to safeguard user information and protect against potential vulnerabilities. The goal LMS is to be a secure, user-friendly, and efficiently designed system that empowers users to seamlessly manage library resources while adhering to best software development practices.

B. Time, cost, performance trade-off assessment

When tackling any project development, it is important to strike a balance among time, cost, and performance based on project priorities, stakeholder expectations, and available resources. By conducting a trade-off assessment, project managers can identify the optimal balance that aligns with the project's goals and constraints. Regular monitoring and adjustment throughout the project can help manage trade-offs effectively and make informed decisions. As such, the following outlines a trade-off assessment between the variables of time, const and performance, to analyse the available resources and modify the development process accordingly:

Time vs. Cost:

If there is a strict deadline or time constraint for the project, allocating more resources and adopting an accelerated development approach may be necessary. This could increase costs due to additional manpower or expedited procurement of resources. Conversely, if there is flexibility in the project timeline, a longer development cycle may allow for a more cost-effective approach with better resource management and optimization.

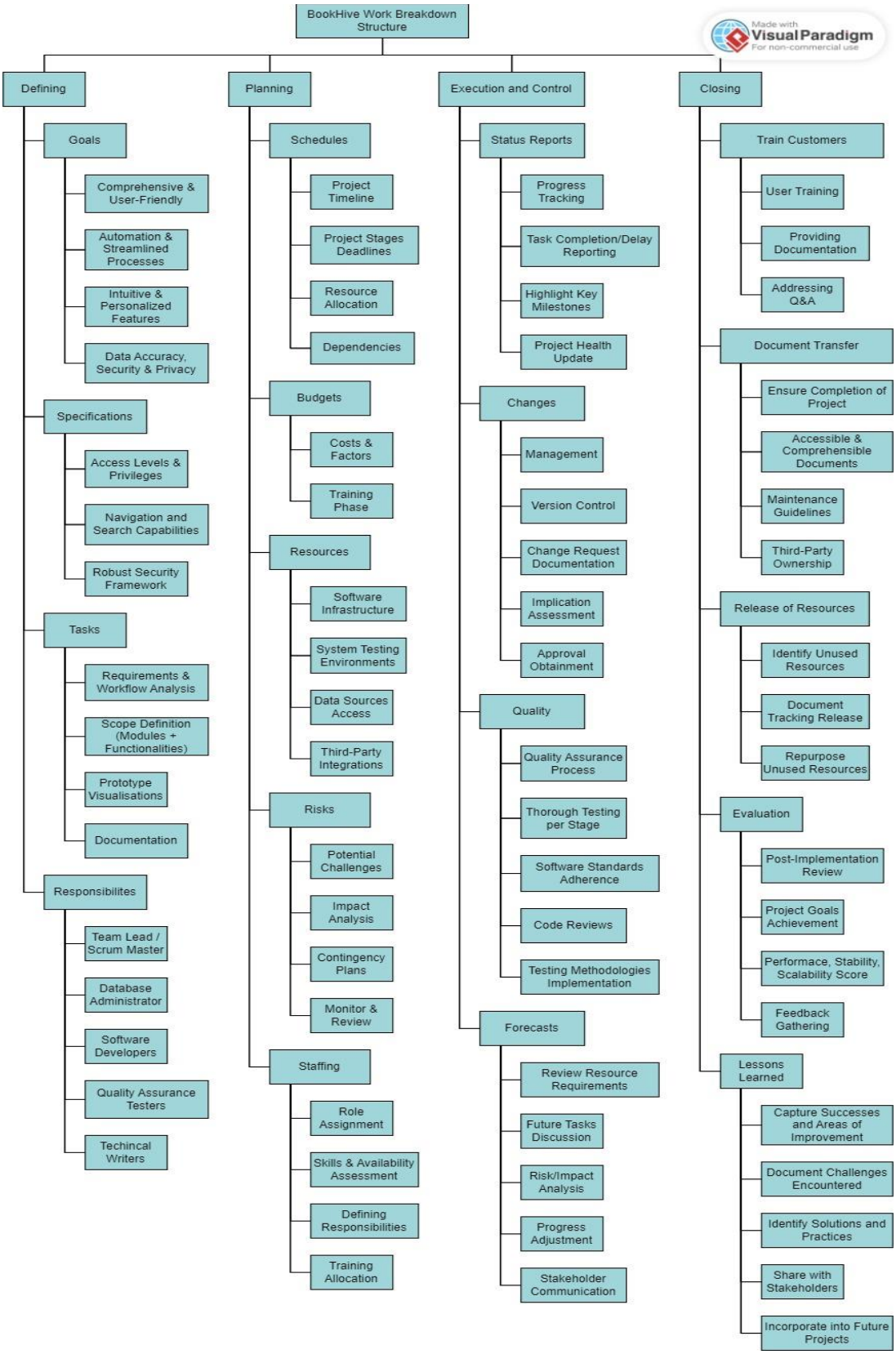
Time vs. Performance:

A shorter development timeline may prioritize meeting the project deadline, potentially leading to compromises in terms of system performance. This could mean a less optimized codebase or limited performance optimization. A longer development timeline provides more opportunities to focus on performance optimization, ensuring the LMS operates efficiently and can handle increased user loads.

Cost vs. Performance:

A higher budget allocation can allow for the use of advanced technologies, robust hardware infrastructure, and dedicated performance testing resources. This can result in a higher-performing system. A constrained budget may require trade-offs in terms of hardware capabilities or the extent of performance testing. This could lead to compromises in system performance, especially during peak usage periods.

C. WBS


BookHive Work Breakdown Structure
Made with Visual Paradigm For non-commercial use
Defining
Goals
Comprehensive & User-Friendly
Automation & Streamlined Processes
Intuitive & Personalized Features
Data Accuracy, Security & Privacy
Specifications
Access Levels & Privileges
Navigation and Search Capabilities
Robust Security Framework
Tasks
Requirements & Workflow Analysis
Scope Definition (Modules + Functionalities)
Prototype Visualisations
Documentation
Responsibilites
Team Lead / Scrum Master
Database Administrator
Software Developers
Quality Assurance Testers
Techincal Writers
Planning
Schedules
Project Timeline
Project Stages Deadlines
Resource Allocation
Dependencies
Budgets
Costs & Factors
Training Phase
Resources
Software Infrastructure
System Testing Environments
Data Sources Access
Third-Party Integrations
Risks
Potential Challenges
Impact Analysis
Contingency Plans
Monitor & Review
Staffing
Role Assignment
Skills & Availability Assessment
Defining Responsibilities
Training Allocation
Execution and Control
Status Reports
Progress Tracking
Task Completion/Delay Reporting
Highlight Key Milestones
Project Health Update
Changes
Management
Version Control
Change Request Documentation
Implication Assessment
Approval Obtainment
Quality
Quality Assurance Process
Thorough Testing per Stage
Software Standards Adherence
Code Reviews
Testing Methodologies Implementation
Forecasts
Review Resource Requirements
Future Tasks Discussion
Risk/Impact Analysis
Progress Adjustment
Stakeholder Communication
Closing
Train Customers
User Training
Providing Documentation
Addressing Q&A
Document Transfer
Ensure Completion of Project
Accessible & Comprehensible Documents
Maintenance Guidelines
Third-Party Ownership
Release of Resources
Identify Unused Resources
Document Tracking Release
Repurpose Unused Resources
Evaluation
Post-Implementation Review
Project Goals Achievement
Performace, Stability, Scalability Score
Feedback Gathering
Lessons Learned
Capture Successes and Areas of Improvement
Document Challenges Encountered
Identify Solutions and Practices
Share with Stakeholders
Incorporate into Future Projects

D. Preliminary schedule

The following schedule is based upon project-specific factors such as team size, resource availability, and project complexity, initialized on the 26th of May. The format of the schedule follows the Software Development Life Cycle, used to develop the software in a well-structured way and maintain quality by testing and validating user specifications before releasing the software to a live environment. As it is a process, we incorporated the Agile methodology within to drive the digital transformation of the product. Thus, having the schedule regularly reviewed and updated as the project progresses, allowed flexibility for changes and unforeseen circumstances. These are introduced in the later sections of this report.

1. Requirements Gathering and Analysis (0.5 weeks):
    - Define LMS Functional and Non-Functional Requirements
    - Conduct meetings with stakeholders to gather and analyze requirements.
    - Identify user stories, use cases, and business scenarios.
    - Document the scope of the project and create a requirements specification.

2. Design and Planning (1 weeks):
    - Design the system architecture and database schema.
    - Determine the high-level and low-level architecture (ie. Multitier Architecture)
    - Create wireframes and prototypes for user interface design.
    - Plan the project timeline, milestones, and deliverables.
    - Select the programming languages, frameworks, and technologies
    - Select database management system for storing the library data

3. Development (2 weeks):
    - Divide the system into multiple tiers or layers:
        - Implement the presentation layer: user interface
        - Implement the logic layer: core functionalities
        - Implement data access layer: database interaction via CRUD operations (Create, Read, Update, Delete)
    - Develop user registration, login, and authentication features.
    - Build inventory management, checkout, and return processes.
    - Implement reservation functionality, integrate the book recommendation feature.
    - Ensure compliance with security and data protection standards.

4. Testing and Quality Assurance (2 weeks)
    - Perform unit testing, integration testing, and system testing.
    - Conduct user acceptance testing to ensure the system meets requirements.
    - Perform performance testing and optimize system performance.
    - Administer Static and Dynamic Code Analysis and fix any bugs or issues.
    - Implement necessary changes based on user feedback and testing results.

- Develop robust error handling mechanisms to handle expected and unexpected errors in the system.

5. Documentation and Training (1 week):
    - Prepare user manuals, system documentation, and training materials.
    - Conduct training sessions for library staff and members.
    - Ensure users are familiar with system functionalities and processes.
    - Provide ongoing support and assistance to users as needed.

6. Deployment and Maintenance (0.5 weeks)
    - Prepare the production environment and deploy the LMS.
    - Take into account factors like server infrastructure, scalability, and backup strategies.
    - Perform final testing in the live environment.
    - Ensure data migration, if applicable, is successful.
    - Coordinate with the library staff to ensure a smooth transition to the new system.
    - Monitor the system's performance, apply necessary updates and patches, and provide ongoing maintenance and support to ensure its smooth operation.

E. Resource allocation

Resource allocation for the BookHive project involved distributing and assigning a variety of resources to ensure the successful execution of the project. The following resources were considered in the context of a software project

1. Human Resources:
    - Developer: Collaborate on the project as outlined by the team in planning and scrum meetings. Evenly share a variety of development tasks and consult team members to clarify if needed.
    - Scrum Master: Facilitate scrum meetings, guide flow of discussion and initiating events such as sprint planning.
2. Hardware and Infrastructure:
    - Web Servers: Allocate servers for hosting the web application, considering scalability and performance requirements.
    - Databases: Determine the database management system (DBMS) and allocate server resources for database storage and processing.

3. Software and Development Tools:
    - Programming Languages/Frameworks: Choose the appropriate programming languages and frameworks for web development.
    - Integrated Development Environment (IDE): Provide developers with the necessary tools and IDEs for coding.

- Version Control System: Use version control tools like Git to manage code changes and collaboration.
- Project Management Software: Utilize project management tools to track tasks, timelines, and progress.

4. Time Allocation:
   - Create a timeline for each phase of the project, including development, testing, and deployment.
   - Allocate sufficient time for each task and consider dependencies.
   - Identify and resolve scheduling conflicts.

5. Budget:
   - Allocate a budget to cover the costs of resources, software licenses, hardware, and other project-related expenses.

6. Communication and Collaboration:
   - Set up communication channels for team members to collaborate effectively.
   - Conduct regular meetings and ensure smooth communication flow.

7. Risk Management:
   - Identify potential risks that could affect resource availability or project progress.
   - Develop contingency plans to address unforeseen challenges.

8. Documentation:
   - Maintain proper documentation of project requirements, design decisions, and development progress.

Resource allocation was carefully planned to ensure the right resources were available at relevant times throughout the project's development. Resource monitoring and adjustment was also beneficial for optimizing resources.

F. Risk assessment/response

Performing risk assessments is an important step in being prepared for potential problems that can occur within any software project. During the risk assessment, if a potential risk is identified, a solution or plan of action is developed. Hence, regular risk assessments and monitoring throughout the project lifecycle are performed to help identify and address potential risks proactively. (A problem analyzed and planned early is a known quantity. Likewise, unanticipated problems can affect a project with no resolution plan.)

1. Technical Risks:

   Risk: Integration challenges with existing library systems or databases.

Response: Conduct a thorough assessment of existing systems, perform compatibility tests, and develop robust integration strategies. Involve stakeholders and subject matter experts during the planning phase.

Risk: Technical complexities in implementing specific features or functionalities.

Response: Assign experienced developers, conduct feasibility studies, and allocate additional time and resources for research and development. Break down complex tasks into smaller, manageable components.

Risk: Performance issues or scalability concerns as user load increases.

Response: Perform load testing and performance tuning throughout the development process. Optimize system architecture and database design to ensure scalability. Implement caching mechanisms and employ cloud infrastructure as needed.

2. Schedule and Resource Risks:

   Risk: Insufficient resources or staff availability during critical project phases.

   Response: Conduct resource planning and allocation early in the project. Identify potential resource constraints and proactively address them through resource management strategies, such as team augmentation or outsourcing.

   Risk: Delays in deliverables or milestones due to unforeseen circumstances.

   Response: Regularly monitor project progress and implement effective project management techniques to mitigate delays. Maintain open communication with stakeholders and manage expectations. Build contingency buffers into the schedule to accommodate unexpected challenges.

3. Security and Data Risks:

   Risk: Data breaches or unauthorized access to user information.

   Response: Implement robust security measures, such as encryption, user authentication, and access control mechanisms. Conduct regular security audits and vulnerability assessments. Adhere to industry best practices and compliance standards.

   Risk: Loss of data due to system failures or disasters.

   Response: Implement regular data backups and disaster recovery procedures. Store backups in secure locations and perform periodic data recovery drills. Implement redundant systems and infrastructure to minimize the risk of data loss.

4. User Adoption and User Experience Risks:

Risk: Resistance or reluctance from library staff or users to adopt the new system.

Response: Involve key stakeholders in the development process and seek their feedback and involvement. Conduct user acceptance testing and gather early user feedback to ensure the system meets their needs. Provide comprehensive training and support to facilitate a smooth transition.

Risk: Inadequate user experience leading to low system adoption or usability issues.

Response: Conduct user research, usability testing, and iterative design improvements to ensure an intuitive and user-friendly interface. Incorporate user feedback into the design and development process. Provide documentation, tutorials, and ongoing support to address user queries and concerns.

G. Finalized schedule

The finalized schedule is the preliminary schedule, augmented with rolled-up estimates and financial requirements. The finalized schedule and financial requirements will help in effective resource allocation, budget planning, and risk management throughout the development lifecycle of the LMS. For a larger-scaled production, it is important to consider additional financial aspects such as contingency funds for unforeseen circumstances, license renewals, and any ongoing subscription costs for services like cloud hosting.

1. Requirements Gathering and Analysis (0.5 weeks):
    - Rolled-up estimate: 0.5 weeks
    - Financial requirements: Analysis cost and stakeholder consultation expenses.

2. Design and Planning (1 week):
    - Rolled-up estimate: 1 week
    - Financial requirements: Design and planning costs, hiring external consultants if needed.

3. Development (2 weeks):
    - Rolled-up estimate: 2 weeks
    - Financial requirements: Development costs, salaries for developers, software licenses, and any external development services.

4. Testing and Quality Assurance (2 weeks):
    - Rolled-up estimate: 2 weeks
    - Financial requirements: Testing infrastructure, testing tools, salaries for testers.

5. Documentation and Training (1 week):
    - Rolled-up estimate: 1 week

- Financial requirements: Technical writers' salaries, training materials production cost.

6. Deployment (0.5 weeks):
    - Rolled-up estimate: 1 week
    - Financial requirements: Server hosting and setup costs, deployment costs.

7. Post-Deployment Maintenance and Support (Ongoing):
    - Rolled-up estimate: Ongoing
    - Financial requirements: Ongoing maintenance costs, technical support expenses.

Financial Rollup:

- Total Development Cost: Sum of analysis, design, development, testing, documentation, deployment, and post-deployment maintenance costs.
- Cash Flow: Plan for cash inflows and outflows, taking into account project expenses and expected revenue (if any) from the LMS.

In any work environment, cash flow requirements play a crucial role in funding the Library Management System (LMS) development project. Cash flow refers to the inflow and outflow of money within the project and is essential for managing the financial aspects of the project effectively.

Implications of Cash Flow Requirements for Funding:

Resource Allocation and Planning: Understanding the cash flow requirements allows the project manager and stakeholders to allocate resources appropriately. It helps in determining the budget for each phase of the project, including development, testing, and deployment. Proper resource planning ensures that necessary funds are available when needed, reducing the risk of delays or project disruptions due to insufficient funding.

Financial Stability and Timely Payments: Adequate cash flow ensures that salaries, vendor payments, and other project-related expenses can be met on time. Timely payments to team members and external consultants foster a positive working environment and maintain trust between all parties involved in the project.

Risk Management: An effective cash flow plan helps identify potential financial risks and contingencies. Having a financial cushion ensures that the project can withstand unexpected expenses or delays without jeopardizing its progress. Adequate funding also allows the team to explore alternative solutions when faced with challenges, reducing the impact of risks on project outcomes.

Investment Decisions: Accurate cash flow projections aid stakeholders in making informed investment decisions. It helps them assess the return on investment and evaluate the project's financial viability. Clear cash flow requirements provide a basis for justifying the project's

funding needs and securing additional investment if required

Working Capital Management: Cash flow requirements influence working capital management. It ensures that the project has sufficient funds to cover operational expenses, ongoing maintenance, and support after the project's completion. Effective working capital management minimizes the need for external funding during the project's lifecycle.

Budget Control and Monitoring: By tracking actual cash flow against the planned budget, project managers can exercise better budget control and monitoring. Deviations can be identified early, and corrective actions can be taken to ensure financial discipline and adherence to the approved funding plan.

Negotiating with Stakeholders: Clear cash flow requirements enable project managers to negotiate effectively with stakeholders, including sponsors and investors. Providing a transparent view of financial needs and justifying the allocation of funds builds confidence in the project's execution and enhances support from stakeholders.

Having a well-defined cash flow plan is essential for the successful funding and execution of the Library Management System development project. It allows for efficient resource management, timely payments, risk mitigation, and informed investment decisions. By considering the implications of cash flow requirements, project stakeholders can ensure that the project progresses smoothly, meeting its objectives within the allocated budget and timeline.

H. Managing the project

The management of the BookHive project consisted of several key activities and the attentive tracking of tasks throughout both planning and development. Meetings and team communication were essential. Sprint planning and scrum meetings were run according to the agile methodology to suit a fast software development environment. Meetings happened both in person and online based on the needs of the team. Notes from meetings were recorded and task lists were formulated. Team communication was regular to clarify points, plan meetings, and remain on track.

1. Project planning phase: Management during the project planning phase was facilitated by stakeholder analysis and attention to business needs. Discussing the problem and revisiting the project vision provided points to grow the project plan. Meetings and team communication were central at this stage, and the team discussed the business, user-oriented, financial, and technical goals. Following this, realistic timelines were created to complete the planned deliverables.
2. Software architecture establishment: This phase resulted in the technical foundations of the project. The team met often to dicuss potential frameworks, version control, software language options, hosting options, and code structure for the project. At this stage, agile methods were further integrated into the ongoing project.

3. Development and testing phase: Managing software development introduced further project management technologies in addition to agile methods, meetings, and team communication. The GitHub version control tool was key for collaboration on the code itself, but also tracking changes in a clean and streamlined way.

II. Project Updates

A. Scope Statement

Upon careful evaluation of the time constraints, we have embarked on a strategic revamping to refine and streamline the scope of our perhaps too ambitious Library Management System project. With a focus on delivering a high-fidelity prototype and workable solution within the allocated time frame, we have made prudent decisions to prioritize essential functionalities.

To begin, we have gotten rid of the fee system for overdue books. Originally, if a book was overdue, the user would accumulate a fee until that book was returned. Upon further evaluation, we decided to have a different system. Now, we have limited the number of books a user can have at the same time to 4. If a user wants more books, he has to return one or more of the books he has checked out. This is a much simpler implementation to ensure users return books and saved us a lot of time to focus on other aspects of the project.

Furthermore, we decided to merge the Librarian and Admin roles into a unified role (the Librarian). After careful analysis, we recognized the striking similarity between these roles, making their union a logical choice. This was due to the fact that the two roles were so similar that it didn’t make sense to distinguish them.

Lastly, our Library Management System is now geared towards one central location. This strategic move eliminates the complexity associated with managing multiple locations, fostering a cohesive user experience for users who grace our virtual library's digital halls.

In conclusion, these changes have paved the way for a high-fidelity prototype of our Library Management System within a reasonable time frame, all the while maintaining our core functionalities for decent user experience.

B. Status report

A status report is a great means to convey where the project's at in terms of project health, risks and progress. This is done by having a mechanism in place that can respond in time to stakeholders or sponsors, who want project updates on a regular basis. This status report provides an overview of the Library Management System development project's current progress and health. The development team has successfully completed key milestones and is on track to deliver the planned functionalities within the allocated budget and schedule. The project manager is closely monitoring potential risks and issues to ensure timely mitigation. Recommendations include proactive risk management and effective communication with stakeholders to ensure a successful project outcome.

1. Project Information:
    - Project Name: BookHive Library Management System
    - Report Date: July 23, 2023
    - Project Managers: Lara Lim, Amani Farid Samy Touabi, Kian Zahrai
    - Sponsors and Stakeholders: CSI3140 – Professor Khalil Abousba
    - Reporting Period: Final Project Status Report

2. Project Status Summary:
    - Key Accomplishments:
        - Completed the requirements gathering phase and finalized the project scope.
        - Designed the system architecture and established the database schema.
        - Developed the user authentication and authorization modules.
        - Implemented the book search functionality and availability status display.
    - Completed Work:
        - Requirements gathering and analysis.
        - System design and architecture planning.
        - User authentication and authorization implementation.
        - Book search functionality development.
        - Integration of book checkout and return features.
        - Implementing user profile management.
        - Finalizing the book recommendation feature.
        - Conducted End-to-End testing.
    - Planned Work:
        - Conducting thorough testing and quality assurance.
        - Implement security protocols.
        - Implementation of automation methods and tools (i.e. Dev/Ops)
    - Project Milestones:
        - Completion of system design and database setup.
        - Successful implementation of user authentication.
        - Deployment of system functionalities.

- Project Deliverables:
  - Diagrams: Activity, Class, Component, Sequence, Entity-Relationship, Dataflow, Project Network, RACI Matrix, Risk Register
  - Software Requirements Specifications
  - Source Code, Unit tests
  - Static & Dynamic Code Analysis Report

- Action Items:
  - Complete integration security of protocols.
  - Conduct user testing and gather feedback for improvements.
  - Work on finalizing the user profile management system.

3. Project Health:
   - Project Budget Overview:
     - The project is currently within the allocated budget, with no significant deviations.

   - Project Schedule Overview:
     - The project priorities have been changed but are progressing as per the planned schedule.

   - Quality Overview:
     - The development team follows coding standards and conducts regular code reviews to ensure code quality.

   - Scope Overview:
     - The project scope remains unchanged, and all key features are being developed as planned.

4. Risk Management Overview:
   - Project Risks & Issues:
     - Limited availability of external consultants for specialized functionalities.
     - Potential delays due to unforeseen technical challenges.
   - Roadblocks:
     - None reported at the moment, but continuous risk monitoring is in place.

5. Conclusions / Recommendations:

- The project is progressing well, meeting key milestones, and adhering to the planned schedule and budget.
- To address the risk of limited availability of external consultants, consider expanding the internal team's expertise or exploring alternative solutions.
- Conduct regular meetings with stakeholders to ensure alignment with project goals and expectations.

C. Problem solving

Our project aimed to develop an advanced book recommendation and borrowing system which would cater to both students and librarians. Throughout the development process, we encountered several challenges that required us to adapt our initial plans and come up with alternative solutions. Originally, our project involved two separate roles for inventory management: admin and librarian, each with distinct responsibilities. However, during the development phase, we realized that maintaining two roles would lead to complexity in managing user access and permissions. To simplify the system, we decided to merge these roles into a single user category: "Librarian." This modification allowed us to streamline user management and improve the overall user experience.

Furthermore, the initial plan involved the implementation of a fine system to encourage users to return books promptly. However, as we delved deeper into the project, we recognized that creating and managing the fine system could be time-consuming and might deter users from utilizing the platform. To address this, we opted for a more straightforward approach by limiting the number of books a user can check out simultaneously. By setting a checkout limit, users are encouraged to return borrowed books promptly before requesting additional ones, thus ensuring a fair distribution of resources and facilitating efficient book circulation.

Also, one of the major challenges we encountered was finding a suitable and cost-effective hosting platform for our web application. Commercial hosting services were often expensive or lacked the necessary resources for our project's requirements. As a solution, we decided to host the website on our own machines using Ngrok. Ngrok is a tunneling service that allows us to expose our local server to the internet securely. By leveraging Ngrok, we were able to create a public URL that enabled remote access to our application hosted locally. This approach not only provided us with complete control over our server and data but also allowed us to test and iterate the project in a development environment before making it publicly accessible.

Initially, our approach to the development included the utilization of automation and DevOps methods and tools. We planned to leverage automation to streamline repetitive tasks, improve efficiency, and enhance the overall development process. By implementing DevOps practices, we aimed to foster collaboration between development and operations teams, ensuring faster delivery, continuous integration, and continuous deployment. These approaches were envisioned to support our project's scalability and maintainability, enabling us to adapt to changing requirements seamlessly.

However, as the project progressed and the severity and scale of the LMS became more apparent, we reevaluated our approach to automation and DevOps. It became evident that the complexity of the project, coupled with its large-scale implementation, required a more robust and sophisticated approach. While automation and DevOps practices are valuable for many software development projects, the LMS's unique requirements demanded a tailored solution.

Considering the project's magnitude and the need for high security, data integrity, and

seamless integration with existing systems, we opted to adopt a more customized and specialized approach to development. This decision allowed us to focus on specific aspects of the project that required individualized attention, while also ensuring that the LMS met stringent security and performance standards.

Although we set aside the initial plan for broad automation and DevOps adoption, our revised approach has proven to be more effective in addressing the specific challenges and complexities of the LMS. By tailoring our development methods to the project's requirements, we have been able to ensure a robust, secure, and scalable system that aligns precisely with the needs of our stakeholders and users.

In conclusion, our project faced various challenges during the development process. However, by adapting our plans and finding innovative solutions, we successfully streamlined user roles, implemented a checkout limit, and hosted our web application on our own machines using Ngrok. These modifications not only simplified the project but also improved the overall user experience. As we move forward, we will continue to evaluate and enhance the platform to ensure BookHive remains a user-friendly and accessible resource for book enthusiasts and readers.

D. Revised project estimates

The following schedule is an updated version of the preliminary schedule, where based upon project timelines and available resources, revision was necessary to be made. The format of the schedule still follows the Software Development Life Cycle, used to develop the software in a well-structured way and maintain quality by testing and validating user specifications before releasing the software to a live environment. And, as it is a process, we incorporated the Agile methodology within to drive the digital transformation of the product. Thus, having the schedule regularly reviewed and updated as the project progresses, allowed flexibility for changes and unforeseen circumstances. These are introduced in the later sections of this report.

1. Requirements Gathering and Analysis (0.5 weeks):
    - Define LMS Functional and Non-Functional Requirements
    - Conduct meetings with stakeholders to gather and analyze requirements.
    - Identify user stories, use cases, and business scenarios.
    - Document the scope of the project and create a requirements specification.

2. Design and Planning (1 weeks):
    - Design the system architecture and database schema.
    - Determine the high-level and low-level architecture (ie. Multitier Architecture)
    - Create wireframes and prototypes for user interface design.
    - Plan the project timeline, milestones, and deliverables.
    - Select the programming languages, frameworks, and technologies
    - Select database management system for storing the library data

3. Development (3 weeks):
    - Divide the system into multiple tiers or layers:

- Implement the presentation layer: user interface
- Implement the logic layer: core functionalities
- Implement data access layer: database interaction via CRUD operations (Create, Read, Update, Delete)

- Develop user registration, login, and authentication features.
- Build inventory management, checkout, and return processes.
- Implement reservation functionality, integrate the book recommendation feature.
- Ensure compliance with security and data protection standards.

4. Testing and Quality Assurance (1.5 weeks)
   - Perform unit testing, integration testing, and system testing.
   - Conduct user acceptance testing to ensure the system meets requirements.
   - Perform performance testing and optimize system performance.
   - Administer Static and Dynamic Code Analysis and fix any bugs or issues.
   - Implement necessary changes based on user feedback and testing results.
   - Develop robust error handling mechanisms to handle expected and unexpected errors in the system.

5. Documentation and Training (0.5 weeks):
   - Prepare user manuals, system documentation, and training materials.
   - Conduct training sessions for library staff and members.
   - Ensure users are familiar with system functionalities and processes.
   - Provide ongoing support and assistance to users as needed.

6. Deployment and Maintenance (0.5 weeks)
   - Prepare the production environment and deploy the LMS.
   - Take into account factors like server infrastructure, scalability, and backup strategies.
   - Perform final testing in the live environment.
   - Ensure data migration, if applicable, is successful.
   - Coordinate with the library staff to ensure a smooth transition to the new system.
   - Monitor the system's performance, apply necessary updates and patches, and provide ongoing maintenance and support to ensure its smooth operation.

E. Management issue

The team's proactive approach, effective communication, and timely problem-solving have helped to address potential management issues and ensure the successful development of the Library Management System. Through careful planning, risk management, and collaboration, the team was able to resolve challenges and maintain project progress in line with the project's goals and objectives. During the development of the Library Management System, the team encountered and proactively addressed some management issues to ensure the project's successful completion:

1. Resource Constraints:
    - Problem Description: The team faced resource constraints, particularly in terms of specialized skills required for certain functionalities.
    - Response: To address this issue, the project manager identified knowledge gaps and skill requirements early on. The team arranged additional training sessions and workshops to upskill existing members. In cases where specific expertise was not available in-house, the team collaborated with external consultants to ensure the successful implementation of critical functionalities.
2. Unforeseen Technical Issues:
    - Problem Description: The team encountered unexpected technical challenges during the implementation of certain functionalities.
    - Response: The team quickly identified and categorized the technical issues. Subject matter experts collaborated with developers to devise effective solutions and workarounds. Regular code reviews and testing helped identify and fix bugs promptly. Continuous monitoring and troubleshooting reduced the impact of technical issues on project timelines and overall progress.

F. Summary

As mentioned, modularizing different tiers of the application gave our development teams the ability to develop and enhance the product with greater speed than developing a singular code base. This is because a specific layer can be upgraded with minimal impact on the other layers. This helped improve the development efficiency by allowing the teams (or individuals) to focus on their core competencies. Our development team has separate developers who specialize in front- end, server back-end, and data back-end development, and by modularizing these parts of an application, we no longer have to rely on full stack developers and can better utilize the specialties of each team.

Scalability is another great advantage of a 3-layer architecture. By separating out the different layers, we scaled each layer independently depending on the need at any given time. For example, during testing, when receiving many web requests but not many requests which affect our application layer, we scaled our web servers without touching or affecting the application servers. Similarly, when we were receiving many large application requests from only a handful of web users, we scaled out your application and data layers to meet those requests without contacting our web server. This allows us to load balance each layer independently,

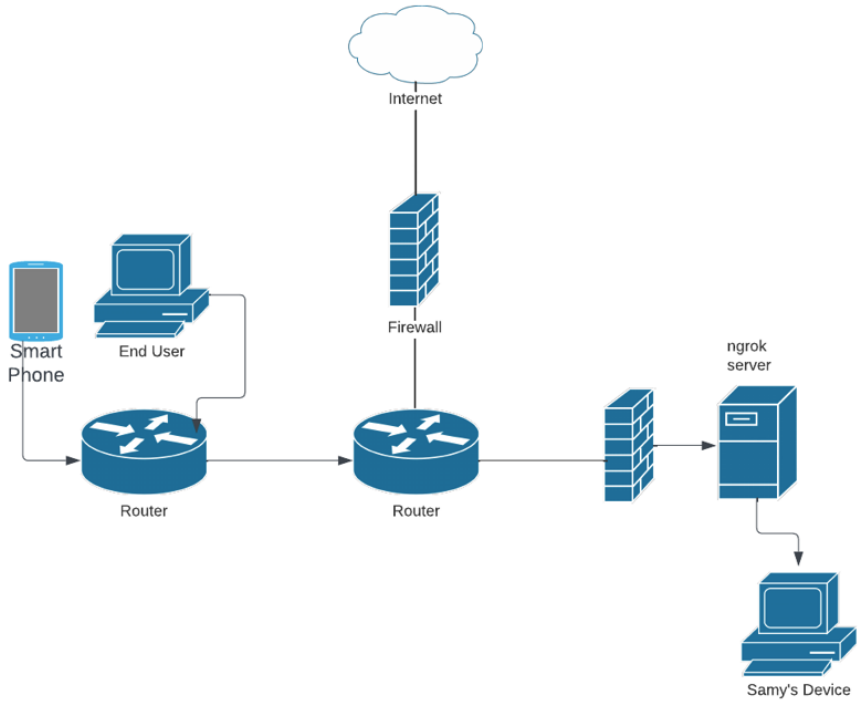
improving overall performance with minimal resources.

Additionally, the independence created from modularizing the different tiers gave us many deployment options. For example, we tested upon having our web server hosted in a public or private cloud while our application and data layers were hosted onsite. Although, due to the timing of the project progress, we had decided have our application and data layers hosted in the cloud while our web server be locally hosted, which led to a better showcasing of our application overall, demonstrating our product's features in full effect.

By having disparate layers, we can also increase reliability and availability by hosting different parts of our application on different servers and utilizing cached results. In contrast, with a full-stack system, we would have to worry about a server going down and greatly affecting performance throughout our entire system. Thus, with this 3-layer application, the increased independence created when physically separating different parts of an application minimizes performance issues when a server goes down.

Project Network Diagram

A project network diagram is a graph that displays the order in which a project's activities are to be completed. Derived from the work breakdown structure, the terminal elements of a project are organized sequentially based on the relationship among them.



Risk Register

A risk register is a document that is used as a risk management tool to identify potential setbacks

within a project. This process aims to collectively identify, analyze, and solve risks before they become problems.

Risk Description	Impact Description	Impact Level 1 (low) to 5 (high)	Probability Level 1 (low) to 5 (high)	Priority Level Impact x Probability
Incomplete requirement specification	Additional time taken deliberating on previously necessary requirements	3	1	3
Ambiguous requirements	Revisitation of already defined requirements interfering with development	2	2	4
Unrealistic schedule	Reevaluation of task planning interfering with development	3	2	6
Limited technologies within budget	Fewer options for long term scalable solutions	1	5	5
Project team member unavailability	Project progress slows	4	2	8
Vulnerability in deployed project	Susceptibility to malicious intent	4	3	12
Database security breach	User information compromised	5	3	15
Hosting platform down	Application unavailable to users	5	2	10
Version control issues	Inconsistent or lost development progress and logs	4	1	4

RACI Matrix

A RACI matrix is a simple, effective means of defining project roles and responsibilities, providing a comprehensive chart of who is responsible, accountable, consulted, and informed every step of the way.

Project Activity/Deliverable	Amani	Kian	Lara	Samy
Meeting planning	A	A	A	A
Scrum meeting management	C	R	C	C
Deliverable 1				
Data flow diagram iteration 1	I	I	I	R
Project scope	I	I	R	I
Project development methodology	I	R	I	I
External interface requirements	R	I	I	I
SRS iteration 1	R	R	R	R
Deliverable 2				
Diagrams review meeting	A	A	A	A
Application Development meeting	A	A	A	A
SRS revision and updates	R	R	R	R
Data flow diagram iteration 2	C	C	C	R
ER diagram	C	C	C	R
Sequence diagram	R	C	C	C
Component diagram	C	C	R	C
Activity diagram	C	R	C	C
Class diagram	R	C	R	C
Database integration	R	I	I	R
PHP implementation	I	I	I	R
Librarian functionality	I	I	R	I
Student functionality	I	R	I	I
Python NLTK implementation	R	I	I	I
Application hosting	C	C	R	R
Project plan report	R	R	R	R
Video demonstration	R	R	R	R

Description of Team and Roles

Task	Sub-task(s)	Assigned
Context Data Flow Diagrams	- Create a diagram to represent the flow of data in the library management system. - Level 0, Level 1, Level 2	Samy
ER Diagram	- Create a diagram to represent the relationships between different entities or object within the LMS	Samy
Sequence Diagram	- Create a diagram to represent the processes and objects involved and the sequence of messages exchanged between the processes and objects to carry out the functionality in the LMS.	Amani
Component Diagram	- Create a diagram to represent the components that are wired together to form larger components or software systems in the LMS.	Lara
Activity Diagram	- Create a diagram to represent the workflows of stepwise activities and actions performed by the LMS.	Kian
Class Diagram	- Create a diagram to represent the structure of the LMS by showing the system's classes, their attributes, operations, and the relationships among objects.	Amani, Lara
Diagrams Review meeting	- Review constructed diagrams and implement feedback - Discussion on programming, deployment and assigning preliminary tasks	Lara, Amani, Samy, Kian
Application Development Meeting	- Separating application into tiers (presentation, logic, data) - Agree upon Programming Languages - Discuss Deployment options	Lara, Amani, Samy, Kian
Implement CSS Layout	- Incorporate stylings	Amani, Lara
Implement PHP pages	- Make login, student, and librarian PHP pages	Samy
Implement Librarian functionalities	- Add Book - Remove Book - Create Account	Lara,
Implement Student Functionalities	- Search & Filter - Recommended Books - Change profile settings	Samy, Kian,
Implement Python NLTK	- Configure python library for sentimental analysis	Amani
Project Plan Report	- Deliverables Write-up	Lara, Amani, Samy, Kian
SRS	- Update document contents	Lara, Amani, Samy, Kian
Project Demonstration Video	- Record project presentation, demonstrate product, explain solution	Lara, Amani, Samy, Kian
Static Code Analysis Report	- Perform static code analysis	Amani, Kian
Dynamic Code Analysis Report	- Perform dynamic code analysis	Samy
RACI Matrix	- Create a matrix to represent project roles and responsibilities.	Lara

Risk Register	- Create a table to identify potential setbacks within the project.	Lara
Project Network Diagram	- Create a diagram to represent the order in which the project's activities are to be completed	Lara

References

Gruss. (2019). *Text Classification With Python*. *YouTube*. Retrieved July 22, 2023, from https://www.youtube.com/watch?v=EfEW3_RLnGA&t=34s.