

Project Title

GYM HUB

(A Gym Management System)

Team Number: 03

Team Members:

Akhilesh Singh (1002106742)

Alma Babydasan (1002168004)

Gayathri Reddy Dendi (1002182864)

Chandra Vamsi Krishna Alla (1002170054)

Table Of Contents

Sr. No	Contents
1.	Project Initiation <ul style="list-style-type: none">• Abstract• Project Proposal• Objectives• Scope of the Project• Project Timeline• Cost and Time Estimation
2.	Project-Planning <ul style="list-style-type: none">• WBS• Project Schedule- Gantt Chart• Burndown Chart
3.	Risk Management <ul style="list-style-type: none">• Risk Identification• Risk Assessment• Risk Mitigation
4.	Project Quality Assurance
5.	Design <ul style="list-style-type: none">• ER Diagram• Sequence Diagram
6.	Implementation <ul style="list-style-type: none">• Hardware/Software Requirements• Development environment and tools Used
7.	Testing <ul style="list-style-type: none">• Manual Testing• Unit Testing• Integration Testing• System Testing• User Acceptance Testing
8.	Maintenance <ul style="list-style-type: none">• Ongoing maintenance plan• Managing updates, patches and bug fixes• Future Maintenance Strategies
9.	Roles and Responsibilities
10.	Conclusion
11.	References

Part 1: Project Initiation

Project Proposal : Write a brief project proposal that outlines the client's requirements, objectives, and scope of the project. Include a project timeline.

Abstract:

The “GymHub” is a web-based application which helps the user to keep track of their health, view various exercises and select membership plans much more efficiently by providing features such as tracking heart rate, calories burned, cardio workouts and viewing exercises .

Objectives:

The primary objective of GymHub is to provide users with a streamlined and efficient platform for tracking and managing their health. The application aims to provide a solution that enables users to easily view their heart rate, amount of calories burned, and select different membership suitable to them. The admin can also add members, trainers manage different gym branches.

Key Outcomes and Contributions:

- Gym Management System Development: Create a comprehensive system for tracking user health and efficient management system for admins.
- User Account Creation: Enable secure user signup, including new user registration and user login.
- Interactive Dashboard for Users: Creating a user-friendly dashboard for efficient health tracking.
- Interactive Dashboard for Admin: Creating a well-organized dashboard for admins.

Project Proposal:

Gym Hub is designed to assist in the daily operations of a fitness center by streamlining and automating them. It provides necessary capabilities for managing gym memberships, trainers, payments as well as any prior payments. The main aim of this software is to enhance the management of gym. The following are some of the key features of our project:

- Users:

- User Registration

- User Dashboard which includes user details, available features such as viewing workouts, membership details and about us.

- Select membership through myplan.

- View various workouts.

- Admins:

- Admin Registration

- Admin Dashboard which includes admin details, adding new members, removing members(delete or edit user information).

- Adding and viewing different gyms.
- Adding and viewing trainers available.

Project Objectives:

Our website is designed to enhance the efficiency and user experience of gym operations for both administrators and members by focusing on the following goals:

- Simplifying administrative duties through a straightforward interface that allows for easy management of member data, attendance, and gym operations.
- Boost member involvement by offering features like diverse workout plans to encourage active participation in their fitness journey.
- Maintain the precision of member data by providing administrative capabilities for easy registration, modification, or removal of member profiles, while giving members easy access to their own information.
- Improve communication between administrators and members by incorporating sections such as the home screen, amenities, membership details, about us, and contact information.
- Craft an intuitive and user-friendly website interface that promotes easy navigation and interaction for both administrators and members.
- Digitize gym management tasks like attendance tracking, member registration, and profile management to enhance operational efficiency.
- Create a central repository for all vital gym-related information, providing members with a complete view of the gym's services.
- Ensure that the Gym Management System is compatible across various devices, facilitating seamless interaction for members and administrators on computers, tablets, or mobile phones.
- Implement strong security measures to safeguard sensitive member information, ensuring the confidentiality and privacy of personal data.

Requirements:

Functional Requirements:

1. **Admin Registration:** Members should be able to register using personal information such as name, email, and contact number. The registration process must include setting up a secure password.
2. **User Registration:** Similar to the admin, the user should be able to register using name, email and contact number.
3. **Admin Login:** Admin should be able to login using email and password.
4. **User Login:** User should be able to login using email and password.
5. **Adding or deleting a member for admin:** Admin should be able to add a new member, edit his/her info, and delete a member completely.
6. **Adding and viewing gyms across various locations:** Admin should be able to add and view gyms across different locations.
7. **Adding and viewing trainers:** Admin should be able to add and view gym trainers.

Non-functional Requirements:

1. **Usability:** The system should be easy to navigate for all user types, with a clear and consistent interface design that reduces learning time and enhances user satisfaction.
2. **Performance:** The system should handle multiple simultaneous users without degradation of performance, ensuring quick response times for all operations, especially during peak usage times.

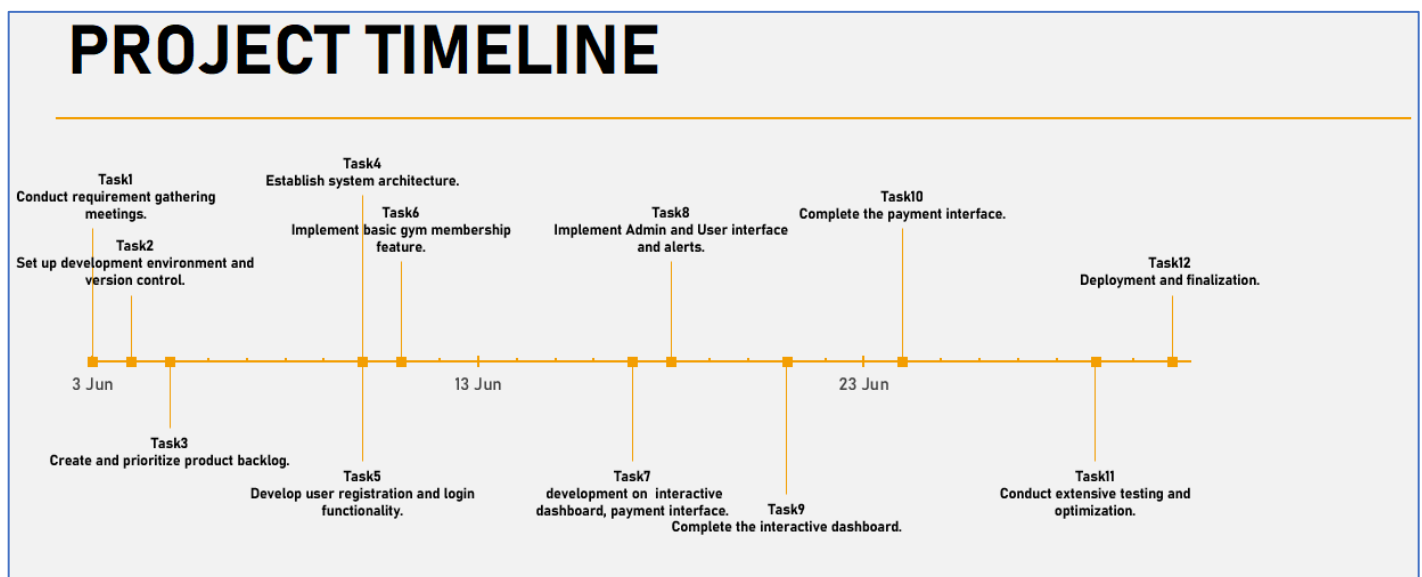
3. **Security:** Implement strong security protocols for data protection, including secure transmission of sensitive information, encrypted storage of personal data, and regular security audits to identify and mitigate vulnerabilities.
4. **Scalability:** The system architecture should support scaling to accommodate an increasing number of users, classes, and data without significant changes to the system.
5. **Reliability:** The system should be reliable, with minimal downtime, and have the ability to recover quickly from any failures, ensuring continuous availability for users.
6. **Maintainability:** The codebase should be well-documented and structured to facilitate easy updates, bug fixes, and feature additions without affecting system stability.

Project Scope:

The scope of the GymHub project includes creating an integrated software platform to streamline fitness center operations. Essential features include managing user accounts, providing detailed user and administrator dashboards, and processing financial transactions. The system is designed to boost operational effectiveness by simplifying the management of gym memberships, trainer schedules, and payments. It will also secure user access and offer a user-friendly interface for both gym members and staff, thereby enhancing the overall administrative efficiency and user interaction with the facility.

Project Timeline:

Below image represents the timeline for our project, which gives an understanding of various tasks and their start time.



Cost and time estimation: Calculate cost using COCOMO, LOC, FP using online software and tools.

Cost/Time Estimation:

Project Cost:

Lines of Code (LOC) Estimation:

Given:

- Total Lines of Code (LOC): 4,000
- Number of Developers: 2
- Project Completion: 4 weeks (20 working days)
- Average productivity rate: 200 lines of code per person-day

Effort Estimation:

- Total Effort in Person-Days = Total LOC / (Productivity Rate × Number of Developers)
- Total Effort in Person-Days = 4,000 LOC / (200 LOC/person-day × 2 developers)
- Total Effort in Person-Days = 10 person-days

Timeline Estimation:

- Total Effort in Person-Days / Number of Developers = Total Days
- 10 person-days / 2 developers = 5 working days

Cost Estimation:

- Cost per person-month: \$5,000 (approximate average cost for a mid-level developer)
- Total Effort in Person-Months = Total Effort in Person-Days / 20 days per month
- Total Effort in Person-Months = 10 person-days / 20 days per month = 0.5 person months
- Cost = Total Effort in Person-Months × Cost per Person-Month × Number of Developers
- Cost = 0.5 person-months × \$5,000 per person-month × 2 developers
- Cost = \$5,000

Function Point (FP) Estimation:

Assumptions:

- Programming language: React.js (frontend) and Node.js (backend)
- Average Value Conversion (AVC) for modern web frameworks: 25 (assuming high level language)
- Unadjusted Function Point (UFP) count: 150 (assuming average complexity)
- Average rating for the 14 General System Characteristics (GSCs): 3 (assuming average complexity)

Adjusted Function Point (AFP) Calculation:

- $AFP = \text{Count Total} \times [0.65 + 0.01 \times \Sigma(F_i)]$
- $AFP = 150 \times [0.65 + 0.01 \times (14 \times 3)]$
- $AFP = 150 \times [0.65 + 0.42]$
- $AFP = 150 \times 1.07$
- $AFP = 160.5$

Lines of Code (LOC) Estimation from FP:

- $LOC = AVC \times \text{Number of Function Points}$
- $LOC = 25 \times 160.5$
- $LOC = 4,012.5 \approx 4,013$

Effort Estimation:

- Industry average productivity rate: 8 function points per person-month
- Total Effort in Person-Months = $AFP / (\text{Productivity Rate} \times \text{Number of Developers})$
- Total Effort in Person-Months = $160.5 \text{ FP} / (8 \text{ FP per person-month} \times 2 \text{ developers})$
- Total Effort in Person-Months = 10.03 person-months

Timeline Estimation:

- Total Effort in Person-Months = 10.03 person-months
- Project Completion: 4 weeks (0.8 person-months)

Required Number of Developers:

- Required Number of Developers = Total Effort in Person-Months / Desired Timeline (in person-months)
- Required Number of Developers = $10.03 \text{ person-months} / 0.8 \text{ person-months}$
- Required Number of Developers = 12.54 developers \approx 13 developers

Cost Estimation (FP):

- Cost per person-month: \$5,000 (approximate average cost for a mid-level developer)
- Cost = Total Effort in Person-Months × Cost per Person-Month × Number of Developers
- Cost = $10.03 \text{ person-months} \times \$5,000 \text{ per person-month} \times 13 \text{ developers}$
- Cost = \$651,950

Based on the Lines of Code (LOC) estimation, with 2 developers and a 4-week timeline, the estimated cost for the project is \$5,000.

However, the Function Point (FP) estimation suggests that you would need an unrealistically large team of 13 developers to complete the project within 4 weeks, and the estimated cost would be around \$651,950, which seems highly inflated for a project with an estimated 4,000 LOC. Again, the significant discrepancy between the LOC and FP estimations highlights the importance of carefully considering the assumptions and factors involved in each method, as well as validating

the estimates against historical data or expert judgment.

COCOMO Estimation:

COCOMO II - Constructive Cost Model

Software Size

Sizing MethodSource Lines of Code

SLOC

% Design Modified

% Code Modified

% Integration Required

Assessment and Assimilation (0% - 8%)

Software Understanding (0% - 50%)

Unfamiliarity (0-1)

New

4000

Reused

40

0

0

5

5

Modified

10

10

15

5

5

50

0

Software Scale Drivers

Precedentedness

Very High

Architecture / Risk Resolution

Very High

Process Maturity

High

Development Flexibility

Low

Team Cohesion

Very High

Software Cost Drivers

Product

Required Software Reliability

Nominal

Personnel

Analyst Capability

High

Platform

Time Constraint

Very High

Data Base Size

High

Programmer Capability

High

Storage Constraint

Very High

Product Complexity

Nominal

Personnel Continuity

Low

Platform Volatility

Low

Developed for Reusability

Low

Application Experience

Nominal

Project

Use of Software Tools

Very High

Documentation Match to Lifecycle Needs

Nominal

Platform Experience

High

Multisite Development

High

Language and Toolset Experience

High

Required Development Schedule

Low

Maintenance

Off

Software Labor Rates

Cost per Person-Month (Dollars)

500

Results

Software Development (Elaboration and Construction)

Effort = 10.2 Person-months
Schedule = 6.0 Months
Cost = \$5087

Staffing Profile

Your project is too small to display a staffing profile due to truncation.

Total Equivalent Size = 4004 SLOC
Effort Adjustment Factor (EAF) = 0.84

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.6	0.8	0.8	\$305
Elaboration	2.4	2.3	1.1	\$1221
Construction	7.7	3.8	2.0	\$3866
Transition	1.2	0.8	1.6	\$610

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.3	0.8	0.2
Environment/CM	0.1	0.2	0.4	0.1
Requirements	0.2	0.4	0.6	0.0
Design	0.1	0.9	1.2	0.0
Implementation	0.0	0.3	2.6	0.2
Assessment	0.0	0.2	1.9	0.3
Deployment	0.0	0.1	0.2	0.4

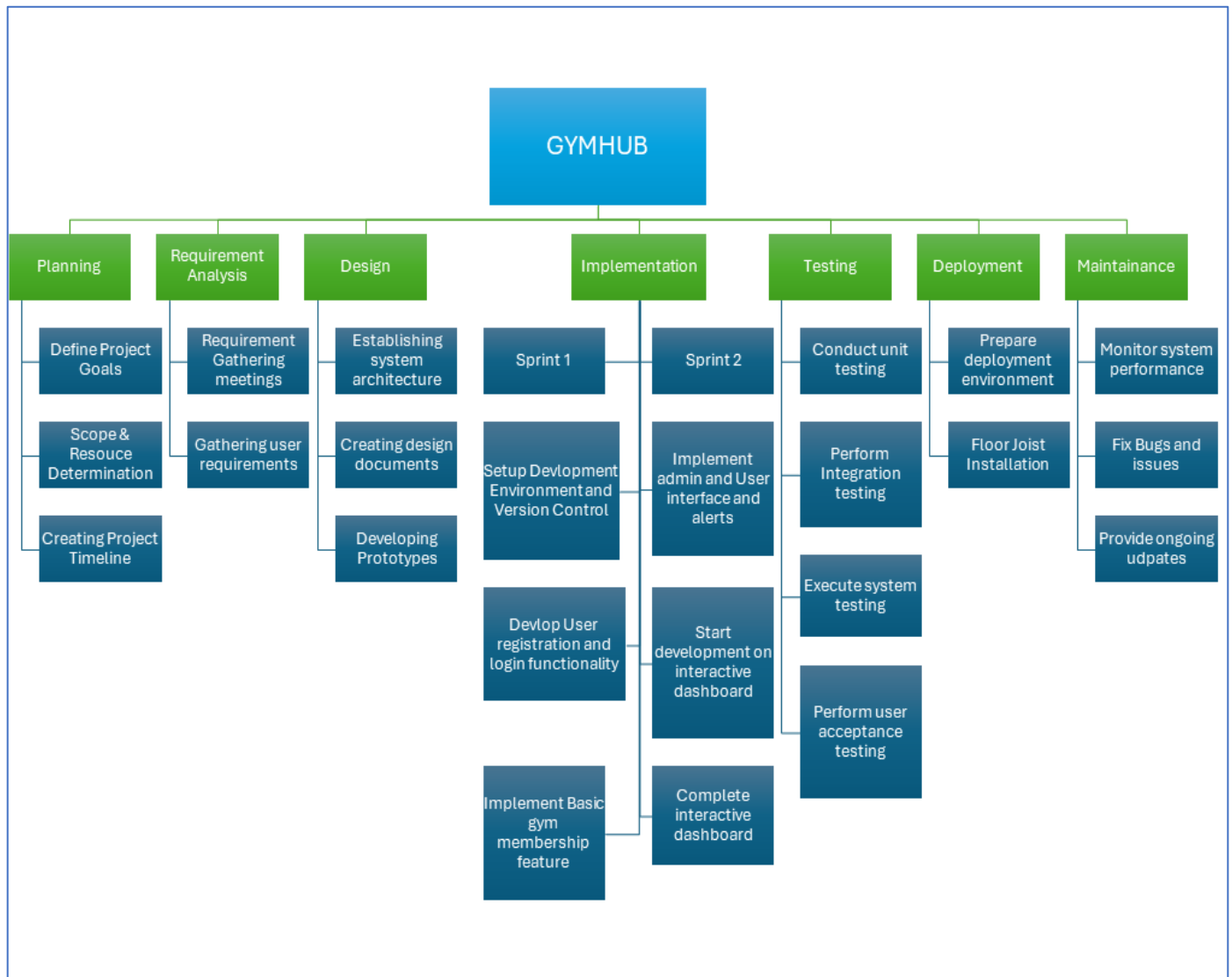
After considering LOC, FP estimation and COCOMO, the COCOMO estimation appears more realistic and aligned with the project's scope and timeline. Nevertheless, it's advisable to monitor the project's progress closely and make necessary adjustments to ensure successful delivery within the planned timeline and budget.

Part 2: Project Planning

Create a WBS for the project. Include at least three levels of decomposition. (5 marks):

Work Breakdown Structure:

The Work Breakdown Structure (WBS) is divided into various levels , with each level representing a set of tasks contributing to the development of the GymHub.

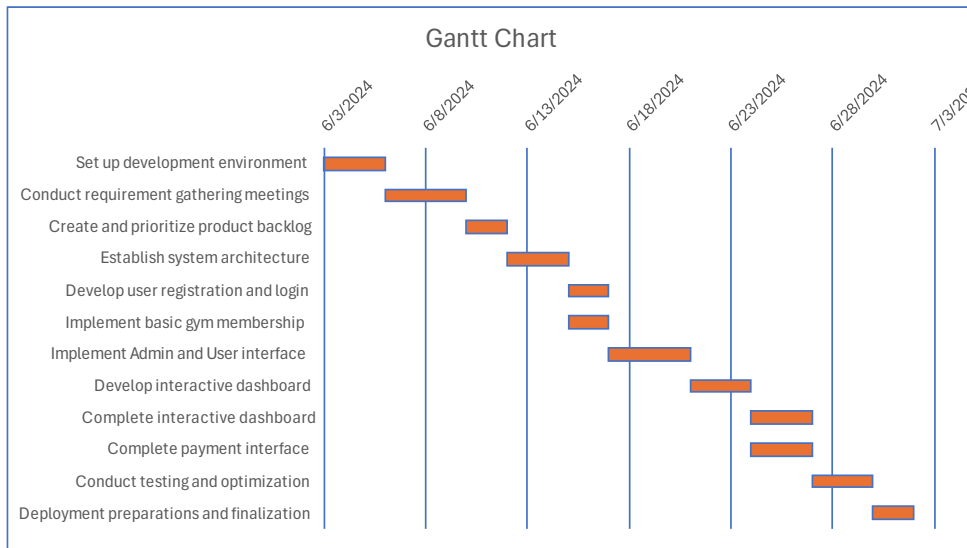


Project Schedule : Develop a project schedule using a Gantt chart or another appropriate tool. Identify critical path activities and milestones.

Gantt Chart:

The Gantt chart below provides a detailed schedule for the Expense Management System project, outlining each phase. It shows a well-organized timeline with specific start and end dates for each task, demonstrating careful planning.

Gantt chart for complete project –



Burndown Chart:

The burndown chart below illustrates the comparison between planned and actual work progression over time. It's evident that the team consistently maintained pace, aligning with the anticipated effort across the project's duration.

● Weeks 1-2 (Sprint 1):

- Task 1: Set up development environment and version control.
- Task 2: Conduct requirement gathering meetings.
- Task 3: Create and prioritize product backlog.
- Task 4: Establish system architecture.
- Task 5: Develop user registration and login functionality.
- Task 6: Implement basic gym membership feature.

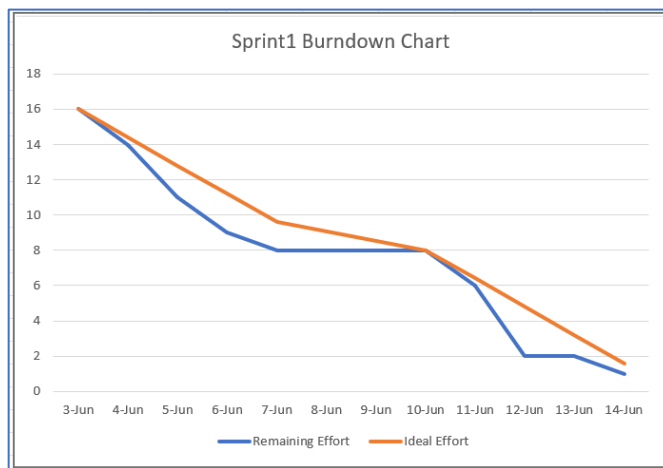
● Weeks 3-4 (Sprint 2):

- Task 7: Implement Admin and User interface and alerts.
- Task 8: Start development on the interactive dashboard and payment interface.
- Task 9: Complete the interactive dashboard.
- Task 10: Complete the payment interface.
- Task 11: Conduct extensive testing and optimization.

- Task 12: Deployment preparations and finalization.

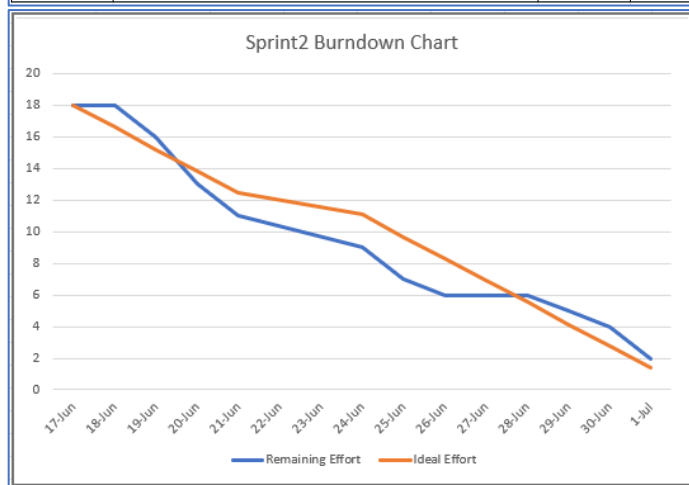
Sprint1 burndown chart:

BURNDOWN CHART													
BackLogID	Task	Initial Estimate Day 0	3-Jun Day1	4-Jun Day2	5-Jun Day3	6-Jun Day4	7-Jun Day5	10-Jun Day6	11-Jun Day7	12-Jun Day8	13-Jun Day9	14-Jun Day10	
1	Set up development environment and version control	3	1	1	1								
2	Conduct requirement gathering meetings.	4	1	1		1			1				
3	Create and prioritize product backlog	2		1	1								
4	Establish system architecture	3						1	1		1		
5	Develop user registration and login functionality	2						1	1				
6	Implement basic gym membership feature	2							1			1	
	Remaining Effort	16	14	11	9	8	8	6	2	2	1	0	
	Ideal Effort	16	14.4	12.8	11.2	9.6	8	6.4	4.8	3.2	1.6	0	



Sprint 2 burndown chart:

BackLogID	Task	Initial Estimate Day 0	17-Jun Day1	18-Jun Day2	19-Jun Day3	20-Jun Day4	21-Jun Day5	24-Jun Day6	25-Jun Day7	26-Jun Day8	27-Jun Day9	28-Jun Day10	29-Jun Day11	30-Jun Day12	1-Jul Day13
1	Implement Admin and User interface and alerts	4		1	1	1	1								
2	Start development on the interactive dashboard and payment interface	3			1	1	1								
3	Complete the interactive dashboard	3		1	1							1			
4	Complete the payment interface	2						1	1						
5	Conduct extensive testing and optimization	4						1					1	1	1
6	Deployment and finalization	2												1	1
	Remaining Effort	18	18	16	13	11	9	7	6	6	6	5	4	2	0
	Ideal Effort	18	16.6154	15.2308	13.8462	12.4615	11.0769	9.69231	8.30769	6.92308	5.53846	4.15385	2.76923	1.3846154	0



Part 3: Risk Management

Risk Identification: Identify at least five potential risks associated with the project, and categorize them as technical, organizational, or external risks.

Risks Involved:

Technical Risks:

1. Integration Task Risk:
 - a. Some tasks may take longer than usual leading to slower integration of two parts of the project.
2. Software Issues:
 - a. Software issues (like lower disk space), and internet connectivity issues.
3. Quality Risk:
 - a. Limited testing leading to bugs in the system.
4. Finishing Touches:
 - a. Not using standard coding rules may lead to more time for beautification/formatting of the code.

Organizational Risks:

1. Team Availability:
 - a. All team members are only available from Friday to Sunday, which might delay completion and slow development.
2. Team Health:
 - a. Team health issues.
3. Poor Communication:
 - a. Miscommunication among team members leads to poor communication.

External Risks:

1. Requirement Changes:
 - a. Some requirements might change within the timeframe allotted for the project.

Risk Assessment: Assess the identified risks in terms of their likelihood and impact. Create a risk matrix.

Risk Assessment Table:

	Risk	Probability (%)	Impact (\$)	Total (\$)
1	Requirement Changes	20	1000	200
2	Team Availability	30	500	150
3	Integration Task Risk	10	2700	270
4	Software Issues	5	700	35
5	Team Health	20	200	40
6	Quality Risk	20	800	160
7	Poor Communication	30	600	180
8	Finishing Touches	10	200	20

Risk Matrix:

Likelihood	High Impact (H)	Medium Impact (M)	Low Impact (L)
High (H)	H	H	M
Medium (M)	H	M	L
Low (L)	M	L	L

Risk Mitigation Plan: Develop a risk mitigation plan for the high-priority risks. Describe specific actions to minimize or eliminate these risks.

Technical Risk Mitigations:

1. Testing after every sprint and maintaining communication among team members:
- a. Testing after every sprint ensures that any bugs or issues are identified early, and maintaining communication helps to address them quickly.
2. Make sure backup is kept after every commit:
- a. Regular backups help prevent data loss due to software issues or integration problems.
3. Implementation needs to be completed faster, so we have enough time for thorough testing. Also, keep testing after every requirement is completed:
- a. Prioritizing implementation allows for sufficient testing time, ensuring quality.
4. Using standard coding practices, so later beautification work can be avoided:
- a. Adhering to standard coding practices reduces the need for extensive code reformatting later.

Organizational Risk Mitigations:

1. **Regular meetings to be held:**
 - a. Regular meetings facilitate ongoing communication and coordination among team members, addressing issues like availability and health.
2. **Prioritize bigger tasks first and schedule extra hours to finish the tasks taken up:**
 - a. Prioritizing tasks and scheduling extra hours helps manage team availability and ensures timely project progress.
3. **Having meetings and communicating early so the tasks can be divided among others:**
 - a. Early communication and task division help in managing workload and preventing burnout.
4. **Regular meetings and proper documentation of tasks and code:**
 - a. Proper documentation and regular updates keep everyone informed and reduce miscommunication.

External Risk Mitigations:

1. **Prioritize bigger tasks first and also schedule extra hours to finish the tasks taken up:**
 - a. Prioritizing tasks and scheduling extra hours can help manage changes in requirements within the project timeframe.

Part 4: Project Quality Assurance

Quality Assurance: Discuss how you would ensure the quality of the deliverables throughout the project lifecycle.

1. Requirements Gathering and Analysis:

- **Clear Requirements:** Make sure the requirements are clearly defined and agreed upon by all stakeholders.
- **Validation:** Check with stakeholders to ensure the requirements meet their needs and expectations.

2. Design Phase:

- **Scalable Architecture:** Create a modular and scalable system architecture for easy maintenance and future upgrades.
- **Prototyping:** Develop prototypes to get early feedback and check alignment with user expectations.
- **Design Reviews:** Hold regular design meetings to ensure the system meets requirements and follows best practices.

3. Development Phase:

- **Code Reviews:** Regularly review code to fix bugs, improve quality, and ensure adherence to coding standards.
- **Unit Testing:** Conduct tests on individual components to confirm their functionality.
- **Integration Testing:** Test the interaction between different modules to ensure they work together properly.
- **CI/CD:** Set up automated processes for building, testing, and deploying to ensure consistent and reliable updates.

4. Testing Phase:

- **Functional Testing:** Test to verify the system meets the required functions.
- **User Acceptance Testing (UAT):** Have end-users test the system to ensure it meets their needs.
- **Regression Testing:** Check that new changes don't negatively impact existing functionalities.
- **Performance Testing:** Assess system performance under different conditions to ensure it is responsive and scalable.

5. Deployment and Maintenance:

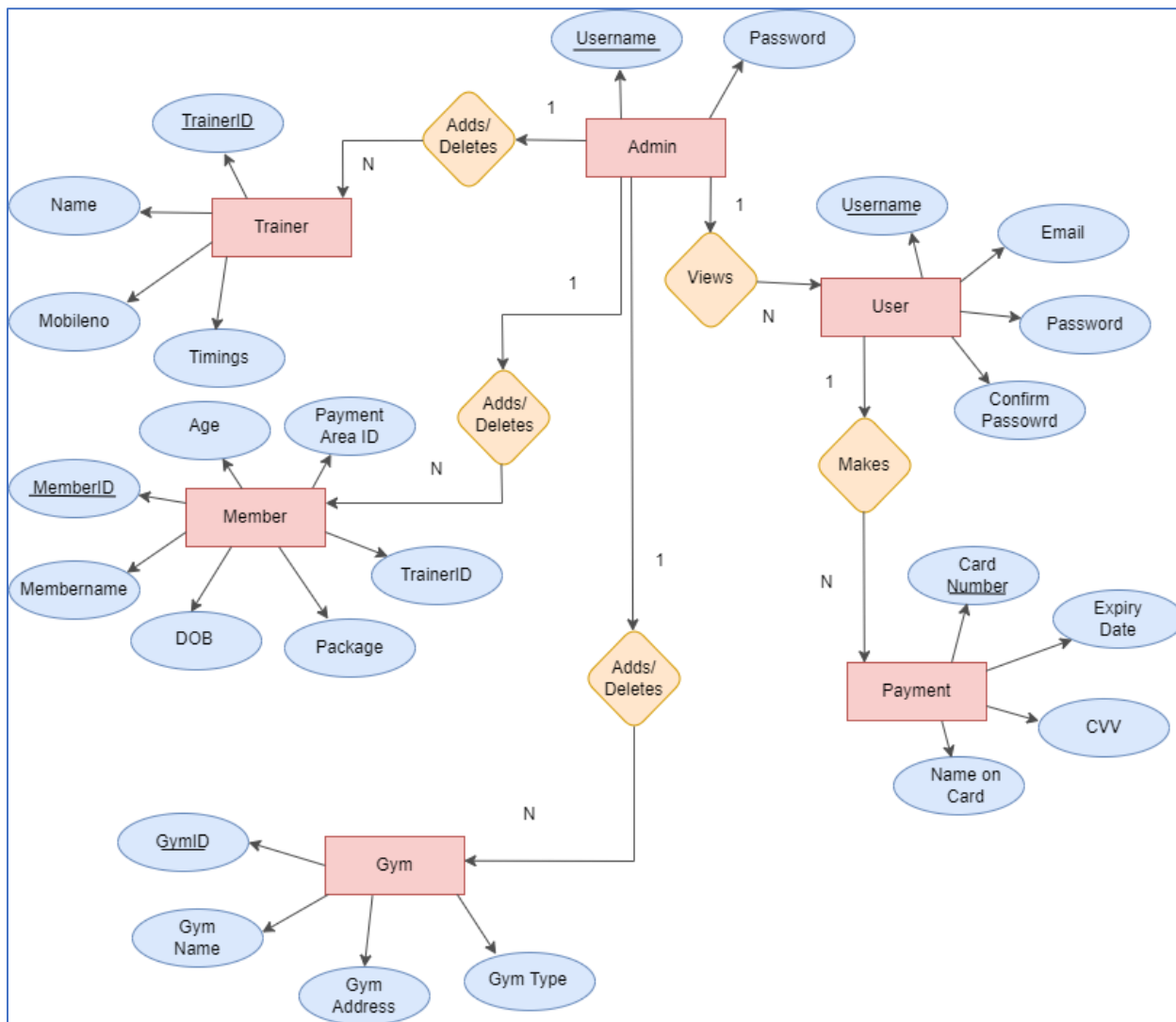
- **Staging Environment:** Use a staging environment for final checks before going live.
- **Monitoring and Logging:** Implement systems to monitor performance and log activities to help with troubleshooting.
- **Feedback Mechanism:** Create a way to collect user and stakeholder feedback for ongoing improvements.
- **Regular Updates and Maintenance:** Continuously update and maintain the system to fix bugs, address security issues, and enhance performance.

Part 5: Design

Include ER, UML, ...diagram.

ER Diagram:

The following Entity-Relationship diagram has six main entities namely ADMIN, USER, MEMBER, TRAINER, GYM and PAYMENT, with one-to-many relationship from ADMIN to TRAINER, MEMBER and GYM. It means that an admin can add/delete many members, trainers and gyms.



Entities:

USER Entity: Represents users of the system.

- Attributes: Username, email, password, confirm password.

ADMIN Entity: Represents admins of the system.

- Attributes: username, password.

MEMBERS Entity: Represents users that have memberships.

- Attributes: memberID, membername, age, DOB, package, trainerID, payment areaID.

TRAINER Entity: Represents gym trainers available for the users of the system.

- Attributes: trainerID, name, mobilenno, timings.

GYM Entity: Represents various gyms available for the users of the system.

- Attributes: gymID, gym name, gym type, gym address.

RELATIONSHIPS:

ADMIN to MEMBER: one-to-many relationship(1:N), indicating that admins can add or delete many members who have membership.

ADMIN to TRAINER: one-to-many relationship(1:N), indicating that admins can add or delete many trainers.

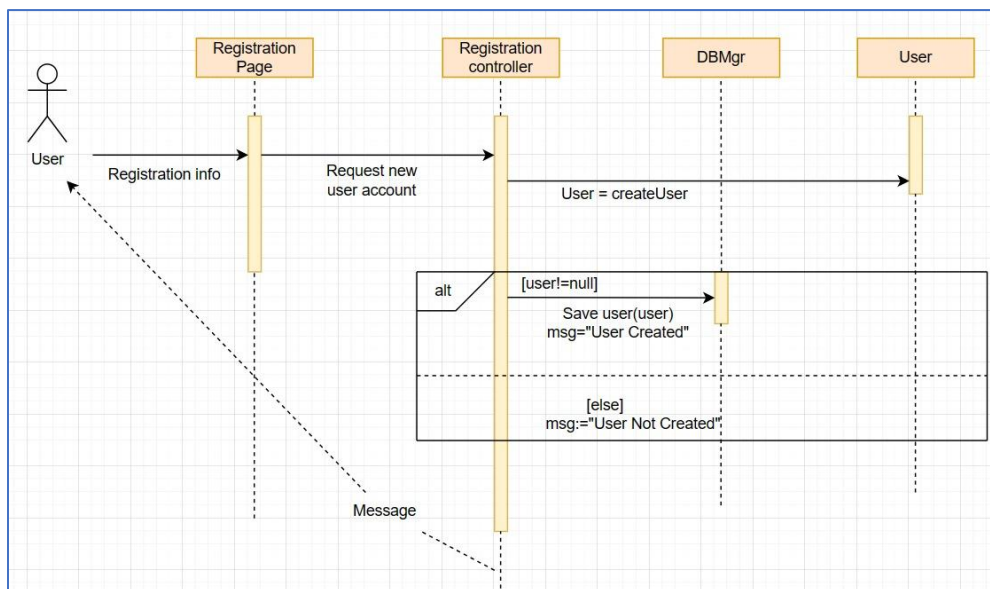
ADMIN to GYM: one-to-many relationship(1:N), indicating that admins can add or delete various gyms.

ADMIN to USER: one-to-many relationship(1:N), indicating that admins can view and manage many users who do not have a membership.

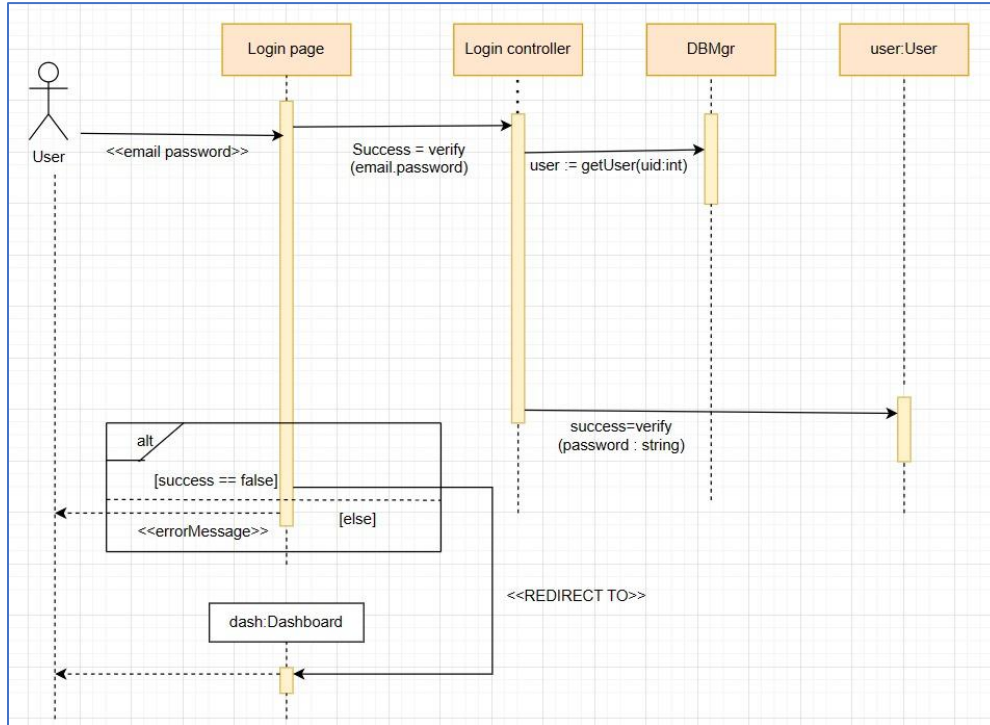
USER to PAYMENT: indicates that a single user can make multiple payments.

Sequence Diagrams:

Sign Up/Register:



Login:



Design Approach (SDLC Model):

Scrum (a subset of agile) Model is the most suitable model for our project since the requirements might change over time and are not fully defined at the start of the project. Since this model supports iterative development process, it allows flexible and rapid adaptation to changing needs. This is essential for dynamic environment like gym where changing requirements like new membership plans or fitness tracking features and user feedback can influence software functionality.

Part 6: Implementation

Describe any kind of tools, framework, libraries, .. to be used in your project.

For implementing our project, we've selected a variety of tools and technologies to enhance efficiency, collaboration, and smooth development across different stages. These tools cover everything from coding and database handling to interface design and system testing.

Development Environment and Tools:

Text Editors / Integrated Development Environments (IDEs):

- **Visual Studio Code:** A streamlined yet powerful editor for coding, offering features like code highlighting, error debugging, and integration with Git.
- **PHPStorm:** A specialized IDE for PHP, providing enhanced features for code assistance and improving developer efficiency.
- **Notepad++:** A flexible text editor that supports multiple programming languages with features like code highlighting, making it a straightforward and effective environment for coding.

Web Servers:

- **Apache HTTP Server:** A reliable and widely used open-source web server that excels in delivering dynamic web content.

Database Management:

- **phpMyAdmin:** An easy-to-use web interface for managing MySQL databases, helping with tasks such as database setup, queries, and maintenance.
- **MySQL Workbench:** Comprehensive tool for database design and management, which includes capabilities for optimizing database performance and administration.

Local Development Environment:

- **XAMPP:** A versatile web server package that combines Apache, MySQL, PHP, and Perl, ideal for local testing and debugging of web applications.

Front-End Frameworks and Libraries:

- **Bootstrap:** A widely adopted framework for creating responsive and attractive web interfaces, simplifying the development of consistent and mobile-adaptive UI components.
- **CSS:** Used to style and format the visual aspects of web pages, ensuring they are visually cohesive and appealing.

Web Browsers:

- **Google Chrome, Mozilla Firefox, Microsoft Edge:** These popular browsers are crucial for testing the Gym Management System in various settings to confirm its functionality and user experience are consistent across platforms.

Part 7: Testing

Explain and discuss both manual testing and automated testing. Result from unit test, integration test, system testing and user acceptance testing along with the automation result.

System testing: System testing for GymHub served as an important stage in validating the application's functionality. The test scenarios consist of user login, user signup and admin login. By testing the integrated components, the system's frontend, backend, and database interactions were meticulously observed to confirm smooth operation of the application.

Manual Testing: In GymHub, manual testing involved QA staff and sometimes actual users working directly with the app. This hands-on method helped catch subtle problems that automated tests might miss, particularly those affecting the user experience. Testers looked at the app's design, how smoothly it worked, and any usability issues to ensure it was easy to use. The feedback from this testing was crucial for understanding how well GymHub worked in real situations, leading to higher user satisfaction and better overall performance.

Automation Testing: We have used Selenium to perform automation testing. Selenium is a popular tool for automation testing designed to simplify a broad range of testing needs, from unit testing, integration testing to system testing.

The three test cases performed are:

Test 1: User Login

Test 2: User Signup

Test 3: Admin Login

Unit Testing:

Test Case #	Functionality	I/P and Action	Expected Result	Actual Result	Status
UT001	User Signup	Adding all the details with valid inputs and clicks on SignUp.	Smooth transition to the Login page.	Successfully moving to Login page.	Pass
UT002	User Login	Giving valid credentials and clicks on Login.	A successful Login message is displayed and redirected to Particular User's Dashboard.	A successful Login message is displayed and redirected to Particular User's Dashboard.	Pass
UT003	Verify an Error msg is displayed for invalid or empty field During user signUp	User click on with an empty fields. Should give error for input validation indication fields req.	User should gets an error msg for field/s req.	User gets a Msg to fill Empty field/s	Pass

UT004	Error msg for login for invalid Credentials	User enters invalid Credentials at Login time.	User should not be able to login he/she will get a Msg on the screen saying Invalid Credentials	User Gets a Msg of error saying Invalid Credentials	Pass
UT005	Success msg for new user registration	User entering all valid data at time of Signup	User should get the Msg for successful Registration.	User gets the Msg for successful Registration and also receives an email.	Pass

Integration Testing:

Test Case #	Functionality	I/P and Action	Expected Result	Actual Result	Status
IT001	User Signup	User signs up and we check if it goes to user login and it contains the word "GymHub".	User should signup successfully and it goes to login page which contains the word "GymHub"	Successfully navigating to login page and login page contains "GymHub"	Pass
IT002	User Login	User logs in and we check if it goes to user dashboard and it contains the word "Gym Hub".	User should login successfully and it goes to user dashboard which contains the word "Gym Hub"	Successfully navigating to user dashboard page and the dashboard contains "Gym Hub"	Pass
IT003	Admin Login	Admin logs in and we check if it navigates to Admin dashboard with word "GymHub".	Admin should log in successfully and we check if it navigates to Admin dashboard with word "GymHub".	Admin successfully logs in and is able to navigate to Admin dashboard with word "GymHub".	Pass

User Acceptance Testing:

Test Case #	Functionality	I/P and Action	Expected Result	Actual Result	Status
AT001	Test User Login and Registration	User will give inputs and login/ register	On successful login/registration user is redirected to Dashboard	On successful login/registration user is redirected to Dashboard	Pass
AT002	Navigation	User Will be navigating through these pages and using the functionalities	User should be able to navigates where they clicks.	User should be able to navigates where they clicks.	Pass
AT003	Dashboard updating on profile, check exercises and apply for a membership.	User checking links to exercises, updating profile and applying for membership and logging out successfully.	On updating profile dashboard is updated, videos are streamed and log out successfully.	Successfully updated profile dashboard, streamed desired exercise videos and logged out successfully.	Pass

Automated testing:

Login selenium

Unit:

```
PS C:\Users\dendi\myproject> python test_app_t.py

DevTools listening on ws://127.0.0.1:53264/devtools/browser/c56df22c-8937-4129-af75-1b503dcb5de5
.
-----
Ran 1 test in 8.995s

OK
```

Integration:

```
PS C:\Users\dendi\myproject> python test_app_int_log.py

DevTools listening on ws://127.0.0.1:54253/devtools/browser/a2a6ac9e-8c39-4e29-9442-e4d80302fc60
.
-----
Ran 1 test in 9.347s

OK
```

System:

```

PS C:\Users\dendi\myproject> python test_app_sys_login.py

DevTools listening on ws://127.0.0.1:54998/devtools/browser/f32838a5-7a4e-47bd-9e67-8e5f10e40372
[22404:15272:0702/143505.086:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[22404:15272:0702/143506.372:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[22404:15272:0702/143508.839:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[22404:15272:0702/143509.972:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
.
-----
Ran 1 test in 17.089s

OK

```

Signup selenium

Unit:

```

PS C:\Users\dendi\myproject> python test_app_sign.py

DevTools listening on ws://127.0.0.1:53647/devtools/browser/11bd09bb-6cdb-47ac-9f9d-cf6ec39fc216
.
-----
Ran 1 test in 7.972s

OK

```

Integration:

```

PS C:\Users\dendi\myproject> python test_app_int_sign.py

DevTools listening on ws://127.0.0.1:54377/devtools/browser/ab218f40-2522-459a-b3dd-d306bd32d4b7
.
-----
Ran 1 test in 8.357s

OK

```

System:

```

PS C:\Users\dendi\myproject> python test_app_sys_sign.py

DevTools listening on ws://127.0.0.1:55260/devtools/browser/6dc60ae4-9b6b-49bf-8c12-6baad052ed8d
[19312:35052:0702/144715.311:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[19312:35052:0702/144717.291:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[19312:35052:0702/144718.388:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[19312:35052:0702/144719.594:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
[19312:35052:0702/144721.841:ERROR:interface_endpoint_client.cc(722)] Message 0 rejected by interface blink.mojom.WidgetHost
.
-----
Ran 1 test in 17.637s

OK

```

Admin login selenium

Unit:

```
PS C:\Users\dendi\myproject> python .\test_app_admin.py

DevTools listening on ws://127.0.0.1:53920/devtools/browser/ee3e3c02-30d5-4d61-9b44-16933c59b7b2
.
-----
Ran 1 test in 7.936s

OK
```

Integration:

```
PS C:\Users\dendi\myproject> python test_app_int_admin.py

DevTools listening on ws://127.0.0.1:54538/devtools/browser/d799c777-6713-4925-83af-2f9e8270f91d
.
-----
Ran 1 test in 8.262s

OK
```

System:

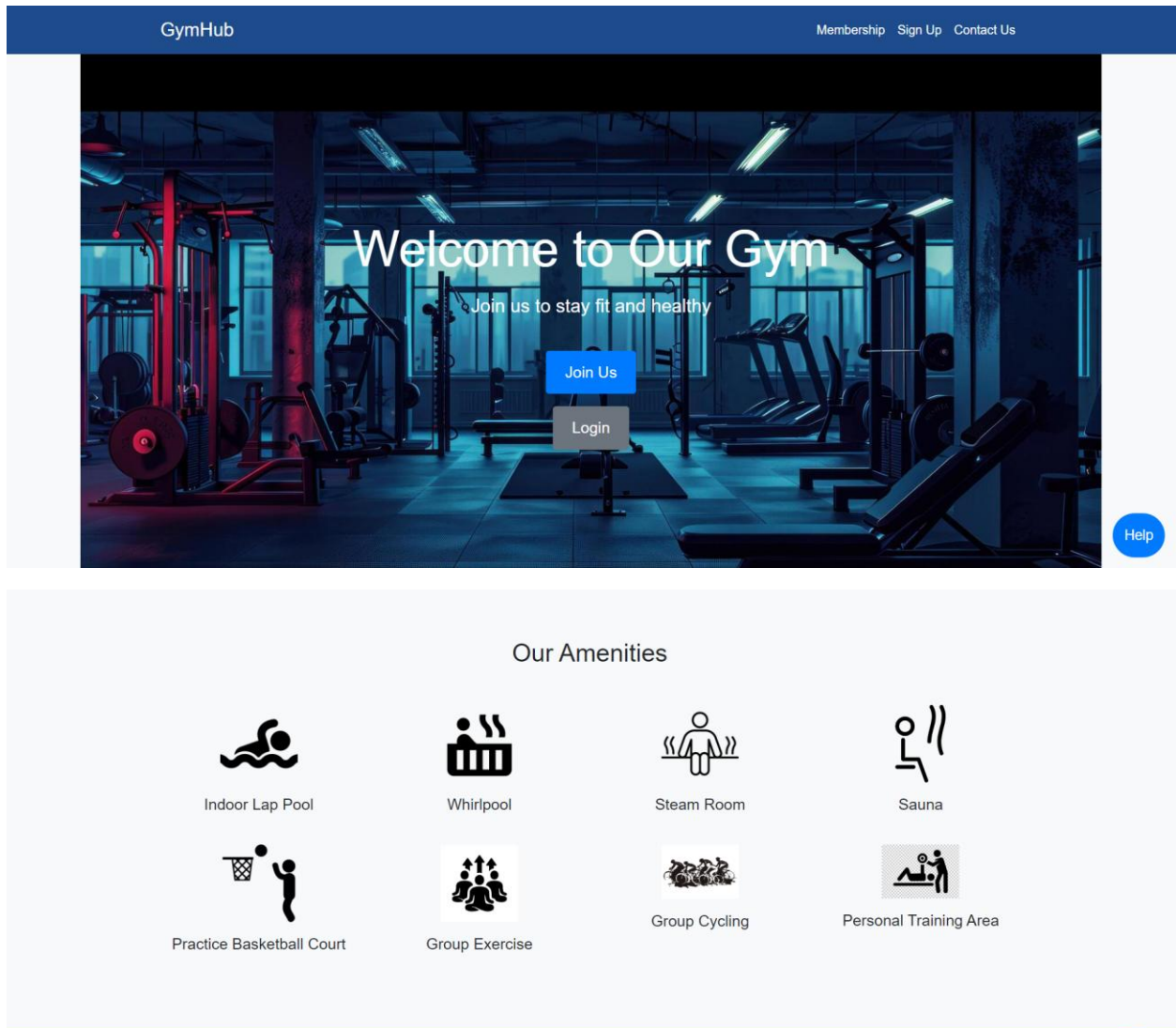
```
PS C:\Users\dendi\myproject> python test_app_sys_admin.py

DevTools listening on ws://127.0.0.1:56019/devtools/browser/174b8922-b6ba-4944-a509-861499fad20d
.
-----
Ran 1 test in 8.707s

OK
```

Implementation Screenshots:

User:



Membership Pricing

Plan	Monthly	Monthly Commitment	Yearly
Platinum	\$36.99 per month	\$31.99 per month	\$23.99 per month
National	\$31.99 per month	-	\$19.99 per month
Silver	\$14.99 per month	\$9.99 per month	-

Why Choose GymHub?



State-of-the-art Facilities



Personalized Training Programs



Community Atmosphere

Help

Advantages Joining GymHub

Discover the benefits that make GymHub the right choice for your fitness journey:

State-of-the-art equipment
Professional trainers
Wide range of fitness classes

- Personalized fitness plans
- Community-driven environment
- Flexible membership options

COMPANY

About Us

RESOURCES

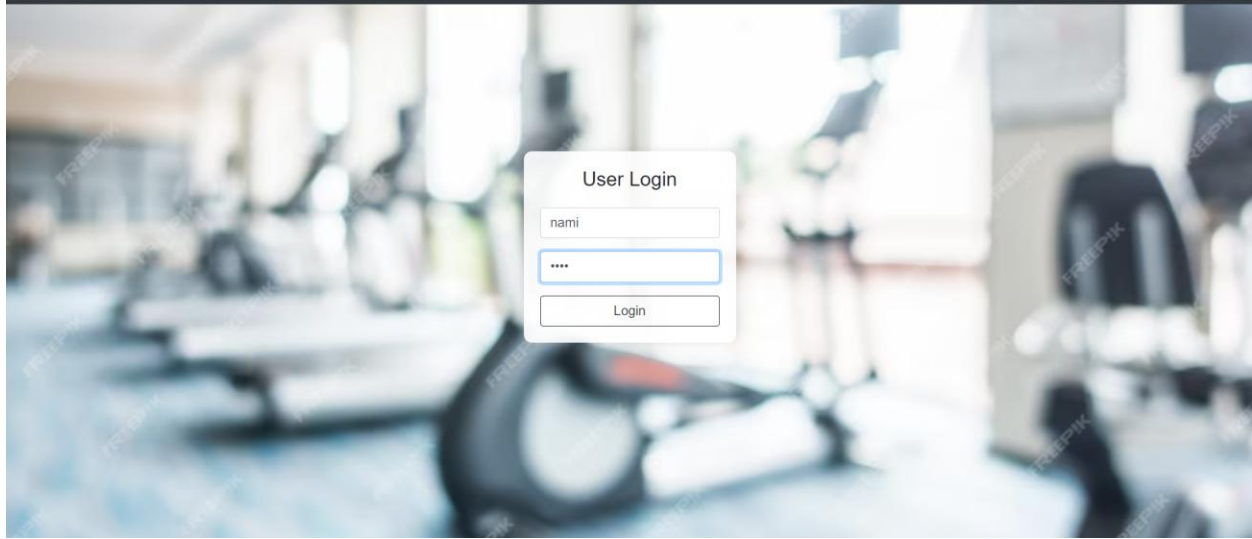
Contact Us

Help

Welcome Gym Hub

User Signup

Sign Up



Profile Picture

nami

Dashboard

Exercises

Update Profile

My Plan

Logout

Gym Hub

Welcome to Fitness, nami!

Heart Rate

72 Bpm

Calories Burned


120 Kcal


Cardio Full Training

01 hrs

Personal Training

Personal Trainers create safe, effective, and efficient exercise programs. Each session consists of approximately one hour of exercise prescription/programming based on each client's individual needs and goals. We have experts to guide you in the fields of General Health and Wellbeing, Cardiovascular Fitness, Hypertrophy, and much more.







GymHub Centre

WORKOUT ROUTINES

30 EXERCISES THAT SHOULD BE IN YOUR WORKOUT ROUTINE

Though there's no definitive list of exercises, these moves should get consideration.



Back: T-Bar Row

As the favorite back exercise of Arnold Schwarzenegger himself, the T-bar row transcends all mere mortal workouts simply because of the Austrian Oak's blessing. Here's how to do it.



Back: Deadlift

The bread and butter of countless gym routines, this move, if done properly, will predominantly engage your back and legs, while building overall strength for your entire body. If you want to master the deadlift,



Back: "The Bird-Dog" aka One-Arm, One-Leg Plank

Sure, you may look a little funny doing it, but this variation on the plank challenges you to keep your back flat and stable. It goes to show that not all essential exercises are



Platinum Plan

\$36.99 per month

[Subscribe](#)

National Plan

\$31.99 per month

[Subscribe](#)

Silver Plan

\$14.99 per month

[Subscribe](#)

Admin:

GymHub Admin

Admin Login

USERNAME

PASSWORD

Login

User LoginAdmin Sign Up

GymHub

log out

GYM

ADD GYM

VIEW GYMS

DELETE GYM

MEMBERS

TRAINERS

ADD GYM

GYM ID

GYM NAME

GYM ADDRESS

GYM TYPE

ADD

GymHub

log out

GYM

MEMBERS

ADD MEMBER

VIEW MEMBERS

DELETE MEMBER

TRAINERS

ADD MEMBER

MEMBER ID

MEMBER NAME

AGE

DOB

PACKAGE

MOBILE NO

PAYMENT AREA ID

GymHub

log out

GYM

MEMBERS

TRAINERS

ADD TRAINER

VIEW TRAINERS

DELETE TRAINER

SEARCH TRAINER

ENTER TRAINER NAME OR TRAINER ID

TRAINER ID	NAME	TIME	MOBILE NO
T1	George	5:00 AM	9999999999
T2	Tanveer	9:00 AM	8888888888
T3	Wong Lee	11:00 AM	7777777777
T4	Kiran Das	1:00 PM	6666666666
T5	Harry Styles	3:00 PM	5555665566
T6	James Corden	5:00 PM	6677667766

Part 8: Maintenance

Describe list of improvements that you can add to your project in future.

Routine Maintenance Tasks:

- Performance and Security Audits: Regularly check and improve system performance and security.
- Data Backups: Frequently back up the database to protect data.
- Log Management: Consistently clear logs and temporary files to avoid storage overflow and keep the system efficient.

Preventive Maintenance:

- System Health Checks: Routinely monitor server, database, application, and user interface performance.
- Database Maintenance: Regularly optimize the database for faster responses and efficient storage.
- Code Quality Assurance: Use tools to check the code for errors or inefficiencies.
- Continuous Integration: Automate testing to ensure all new code meets quality standards.

Update and Patch Management:

- Regular Updates: Update all software components and libraries to the newest versions.
- Security Patching: Apply security patches swiftly to fix vulnerabilities, especially urgent ones for critical issues.

Monitoring and Alerts:

- Performance Monitoring: Use tools to continuously monitor system performance and spot any irregularities.
- Alert Systems: Set up alerts for important budget thresholds to manage finances better.

Corrective Maintenance:

- Issue Resolution: Quickly fix any reported system bugs or problems.
- Post-Incident Analysis: Review what caused issues and learn from them to improve the system.
- Performance Optimization: Regularly enhance the system's performance, focusing on processing and data retrieval speeds.

Management of Updates, Patches, and Bug Fixes:

- Version Control: Use a robust system to manage changes and enable easy reversions if needed.
- Staging Environment: Test new updates thoroughly in a staging environment before full deployment.
- Release Schedule: Keep a regular update schedule for routine patches and have a system for urgent updates.

Scalability and Futureproofing:

- Scalability Planning: Design the system to easily handle more users or increased activity.
- Modular Design: Keep the system flexible to update easily or add new features.
- Technological Updates: Stay current with new technologies and trends to keep the system innovative.

- Continuous User Feedback: Regularly gather and use user feedback to continually refine and improve the system.

Future Improvements:

- Payment Gateway can be added by integrating API.
- Admin and User Dashboard pages can be improved by adding a few more attributed making it more personal for the user.
- Timings of various exercises can also be updated on the website.
- Mobile Optimization: Developing a dedicated mobile app or enhancing the mobile. responsiveness of the web interface to improve accessibility and user engagement.
- Feature Expansion: Adding features such as personalized workout plans, user progress tracking, and interactive elements like forums or chat services for better user interaction.
- Advanced Analytics: Integrating more advanced analytics features for admins to help track gym.

Part 9: Roles and Responsibilities

SPRINT		MEMBER	ROLES and RESPONSIBILITY
Sprint (WEEK 1 - 2)	1	Akhilesh Singh	<p>Scrum Master. Responsible for developing the Data Dictionary aligned with project specifications.</p> <p>Developed UI pages for User, Admin and integrated each of them with styling.</p>
Sprint (WEEK 1 - 2)	1	Chandra Vamsi Krishna Alla	<p>Tasked with designing the database architecture and creating tables. Implemented authentication and authorization functionalities.</p> <p>Scrum Master. Reviewed and revised the Data Dictionary as required by evolving project needs.</p>
Sprint (WEEK 1 - 2)	1	Gayathri Reddy Dendi	<p>Handled the presentation of data including names, descriptions, categories, and pricing. Conducted data verification and validation.</p> <p>Checking of product backlog and creating database connections for all functionalities involved.</p>
Sprint (WEEK 1 - 2)	1	Alma Babydasan	<p>Developed the structural framework for project operations.</p> <p>Created User and Admin interfaces after completion of login page creation for User and Admin.</p>
Sprint (WEEK 3 - 4)	2	Alma Babydasan	<p>ScrumMaster: Completed the integration of interactive dashboard with the login page and added some more valuable contents to help user decide on the Gym membership.</p> <p>Conducted Integration testing for user login, admin login and user sign up pages.</p>
Sprint (WEEK 3 - 4)	2	Gayathri Reddy Dendi	<p>ScrumMaster: Created payment page after selecting the membership deal and</p>

		<p>integrated with interactive dashboard page. In database also payment records updated after completing transaction.</p> <p>Conducted manual testing and unit testing for user login, admin login and user sign up page. Created all the diagrams needed for final document.</p>
Sprint (WEEK 3 - 4)	2	<p>Chandra Vamsi Krishna Alla</p> <p>Integrating User and Admin interfaces with database and establishing connection between all of functions.</p> <p>Conducted testing of the MySQL database functionality and also tested the connections between the pages and the database with updating values.</p>
Sprint (WEEK 3 - 4)	2	<p>Akhilesh Singh</p> <p>Developed an Interactive dashboard for User's where they can check their Plan, different types of exercises, update profile picture, etc.</p> <p>Conducted Integration testing for the user login, admin login and user signup page. Conducted manual and user acceptance testing to validate product features and user interfaces.</p>

Part 10: Conclusion

GymHub has successfully implemented a comprehensive gym management system by integrating a variety of sophisticated tools and technologies. The use of powerful IDEs like Visual Studio Code and PHPStorm has made coding more efficient, increasing productivity considerably. With its ability to highlight code in various languages, Notepad++ helped with quick edits and fixing problems, making it a very useful tool for GymHub.

Apache HTTP Server played an important role in making sure web content is delivered consistently, which is very important to keep a good user experience. Along with this, strong database management systems such as phpMyAdmin and MySQL Workbench were used for managing detailed data functions from user registration to complicated inquiries and maintenance tasks. XAMPP gave a steady and adaptable base for examining and correcting errors in the application, making sure all functions were working properly prior to being deployed. For the front end, Bootstrap and CSS greatly helped in creating an interface that is flexible to various devices (responsive) while also looking good visually. They adjusted smoothly on different web browsers like Google Chrome, Mozilla Firefox guaranteeing users quality.

The application has many features, starting from simple admin and user account handling to more complex tasks like managing gym locations and trainers. This broad functionality paired with an easy-to-use interface puts GymHub at the forefront of gym management solutions.

Overall, GymHub's effective integration of multiple tools and technologies has not only streamlined gym management processes but also enhanced the overall user experience, making it a robust and reliable system.

Part 11: References

Text Editors / Integrated Development Environments (IDEs):

- **Visual Studio Code:** A streamlined yet powerful editor for coding, offering features like code highlighting, error debugging, and integration with Git. (<https://visualstudio.microsoft.com/>)
- **PHPStorm:** A specialized IDE for PHP, providing enhanced features for code assistance and improving developer efficiency. (<https://www.jetbrains.com/phpstorm/>)
- **Notepad++:** A flexible text editor that supports multiple programming languages with features like code highlighting, making it a straightforward and effective environment for coding. (<https://notepad-plus-plus.org/>)

Web Servers:

- **Apache HTTP Server:** A reliable and widely used open-source web server that excels in delivering dynamic web content. (<https://httpd.apache.org/>)

Database Management:

- **phpMyAdmin:** An easy-to-use web interface for managing MySQL databases, helping with tasks such as database setup, queries, and maintenance. (<https://www.phpmyadmin.net/>)
- **MySQL Workbench:** Comprehensive tool for database design and management, which includes capabilities for optimizing database performance and administration. (<https://www.mysql.com/products/workbench/>)

Local Development Environment:

- **XAMPP:** A versatile web server package that combines Apache, MySQL, PHP, and Perl, ideal for local testing and debugging of web applications. (<https://www.apachefriends.org/download.html>)

Front-End Frameworks and Libraries:

- **Bootstrap:** A widely adopted framework for creating responsive and attractive web interfaces, simplifying the development of consistent and mobile-adaptive UI components. (<https://getbootstrap.com/>)
- **CSS:** Used to style and format the visual aspects of web pages, ensuring they are visually cohesive and appealing.

Web Browsers:

- **Google Chrome, Mozilla Firefox, Microsoft Edge:** These popular browsers are crucial for testing the Gym Management System in various settings to confirm its functionality and user experience are consistent across platforms. (<https://www.google.com/chrome/>)