A · P · U

ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Title	:	**Responsive Web Design & Development Group Assignment - Report**
Intake Code	:	**UCDF2309ICT(SE)**
Module Code	:	**AAPP012-4-2-RWDD**
Level of Study	:	**Diploma Semester 4**
Project Title	:	**Goalify**
Module Lecturer	:	**Mr. Daniel Mago Vistro** **Mr. Mohamad Firdaus Che Abdul Rani**
Hand Out Date	:	**6 January 2025**
Submission Date	:	**24 January 2025**
Group No.	:	**Group 1**

Group Members:

NO	Name	Student ID
1	Lim Chee Xuan	TP 075916
2	Lum Han Xun	TP 076160
3	Ng Yvonne	TP 076390
4	Paureen Tan Nie Nie	TP 075914
5	Phang Shea Wen	TP 075813

Table of Contents

1.0 Introduction and Project Plan

In the ever-evolving world, increasing productivity could be a vital strategy to address different kinds of challenges, such as striving for competitive advantages, improving quality of life and promoting economic growth. According to Josh Howarth, 71% of business leaders are pressured by executives to foster employee productivity. About 80-95% of college students suffer from procrastination in their academics (Jude, n.d.). These statistics indicate that low productivity and lack of time management skills might affect the overall outcomes of the given tasks or projects. Our team has proposed a web application that allows the community to enhance their productivity.

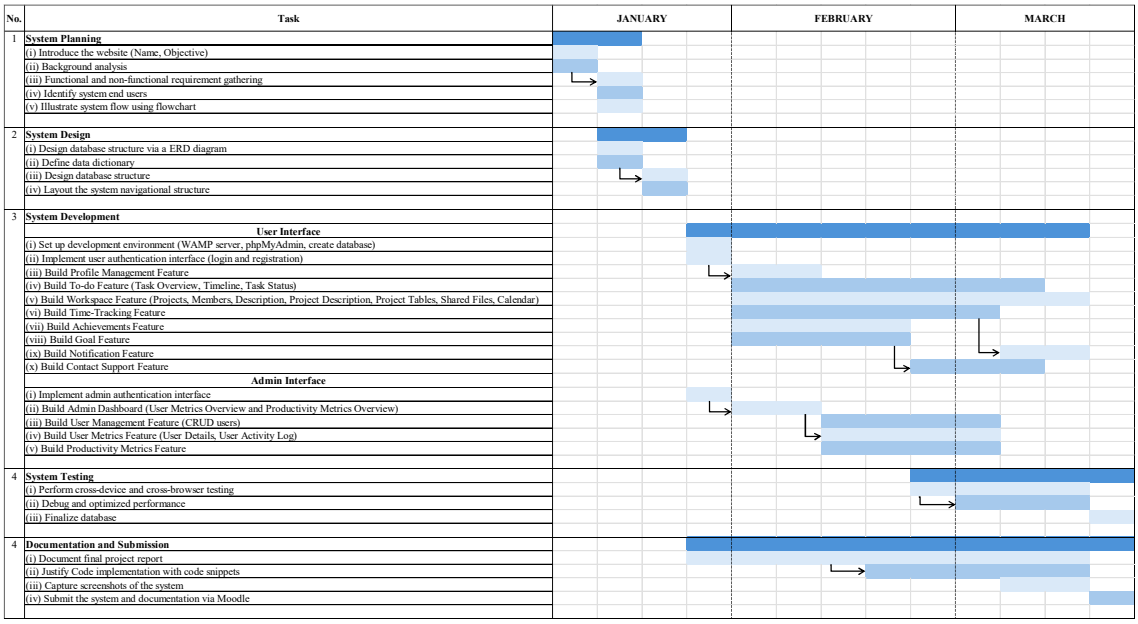


Figure 1.1 Goalify Project Plan

1.1 Title and Target Users of the Web Application

The title of the web application is “Goalify” as it clearly emphasizes that the main role of the application is to achieve goals effectively, efficiently and productively. It is simple and easy to understand to reduce the effort on guessing its general functions because people can immediately associate the application as a tool to manage tasks’ completion time, streamline the workflow and collaboration. “Goalify” is said to be responsive as it’s interface size in different types of devices such as desktops, tablets and mobile phones is in the same ratio.

The target users of “Goalify” include anyone seeking better productivity and needing an application to help them manage tasks and deadlines. For instance, professionals and entrepreneurs can use this tool to organize tasks, manage time and make informed decisions. Students and educators can benefit from handling assignments, tracking study hours and setting academic goals. Remote workers can improve collaboration, enhance effective communication and streamline workflows by utilizing the collaboration features. With these useful features, “Goalify” caters to the requirements of users aiming to stay organized and efficient.

1.2 Objectives of the Project

The objective of creating “Goalify” is to **streamline task management** in daily scenarios, as it provides various useful features to effectively organize, prioritize and track tasks. Users are able to focus on high-prioritize tasks and track the progress of projects with real-time updates, therefore improving tasks planning. With clearer tasks management, “Goalify” not only boosts overall efficiency and performance, but also reduces task-related stress.

“Goalify” also tends to **improve time utilization** by providing an intuitive and visualized task scheduling system. Users can gain a clearer understanding of their daily workload and prepare accordingly, thereby reducing time wastage. By utilizing the time-tracking features, users can monitor the time spent on various tasks and projects. Therefore, users can identify and analyze their inefficiency and time management habits, which are helpful in aiding them to practice better time management skills.

Furthermore, “Goalify” can **enhance collaboration**. Users can assign their tasks among team members and share relevant files easily to complete a project. Team leaders are able to set deadlines and monitor each member’s progress in real-time to ensure overall productivity. The visibility of the progression increases transparency and avoids unnecessary arguments, providing a cohesive work environment with clear accountability. By using “Goalify”, the scheduled projects will be accomplished on time with clearer task division.

2.0 Background Analysis and Requirement Gathering

2.1 Web Application Context

"Goalify" offers various features designed to streamline workflow and increase efficiency. The **task management** system allows users to create, monitor and prioritize tasks through features such as reminders, categories and deadlines. Users can also define long-term and short-term goals, receive reminders and track their progress with the **goal setting** features. Therefore, users can stay motivated and ensure that they stay on track. The **time tracking** tool allows users to monitor the time spent on various projects and tasks, which provides users insight into their time management habit and work efficiency. Additionally, the **collaboration** features play a vital role in enhancing teamwork, as it provides users with the ability to work together on shared tasks and projects, with additional tools like file sharing, comments and real-time updates. Last, the **productivity analytics** feature delivers data-driven insights that help users to identify trends and improve overall performance.

2.2 End-user

- User
- Administrator

2.3 End-user Functional Requirements

<table>
<tr><td colspan="2">Student can be able to:</td></tr>
<tr><td>

Create a new account or log in to existing account.
Access to the to-do list

View their task in to-do list
Add their task in to-do list
View task timeline of the current day
View task status

Access to the workspace feature

Create a workspace
View the workspace’s details (Projects, members and descriptions)

</td><td>

View their achievements
Track their time when doing a task
View or add their goals

Short-term goal
Long-term goal

Access to the inbox to get information
Access to the help section to get assistance
Access to their profile

Edit their personal information
Change their account password
Log out

</td></tr>
<tr><td colspan="2">Administrator can be able to:</td></tr>
<tr><td colspan="2">

Log in to their admin account.
Access admin’s dashboard to manage backend system.

</td></tr>
</table>

2.4 Non-Functional Requirements

- The web application shall be accessible across different devices.
- The web application shall protect user privacy.

3.0 Design

3.1 Flowchart

3.1.1 User

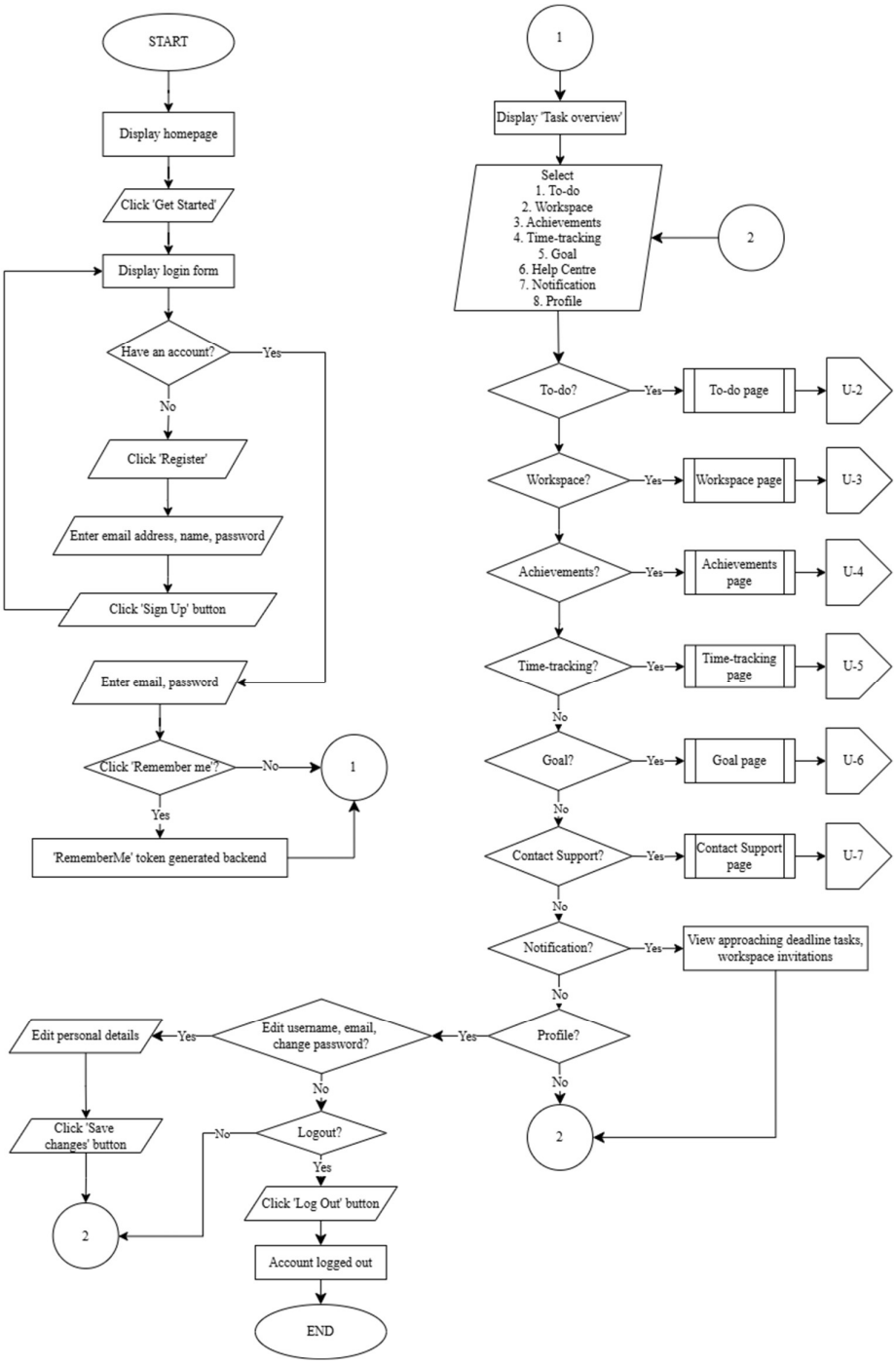


Figure 3.1.1 User

3.1.1.1 Task Management

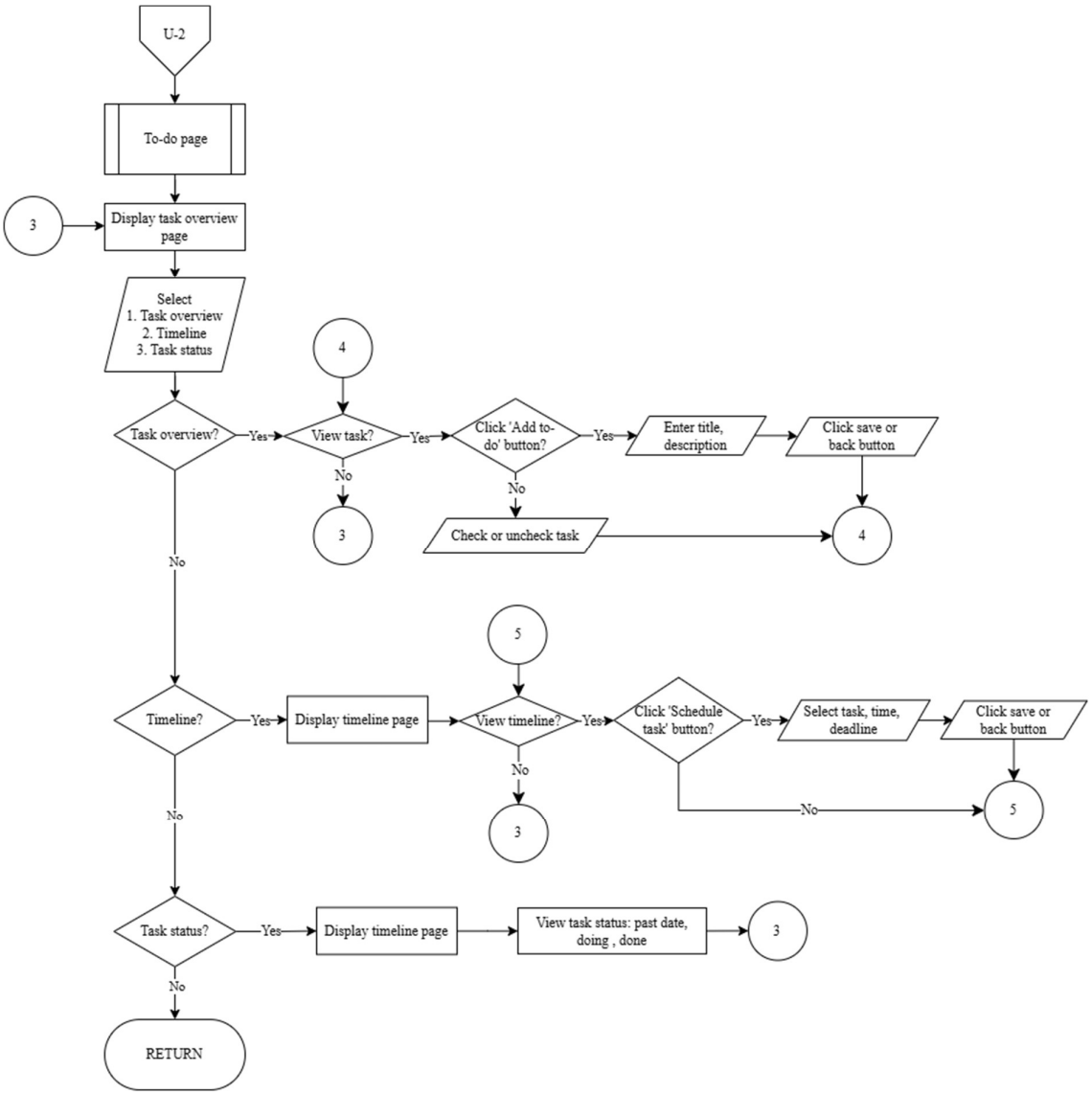


Figure 3.1.1.1 Task Management

3.1.1.2 Collaboration Features

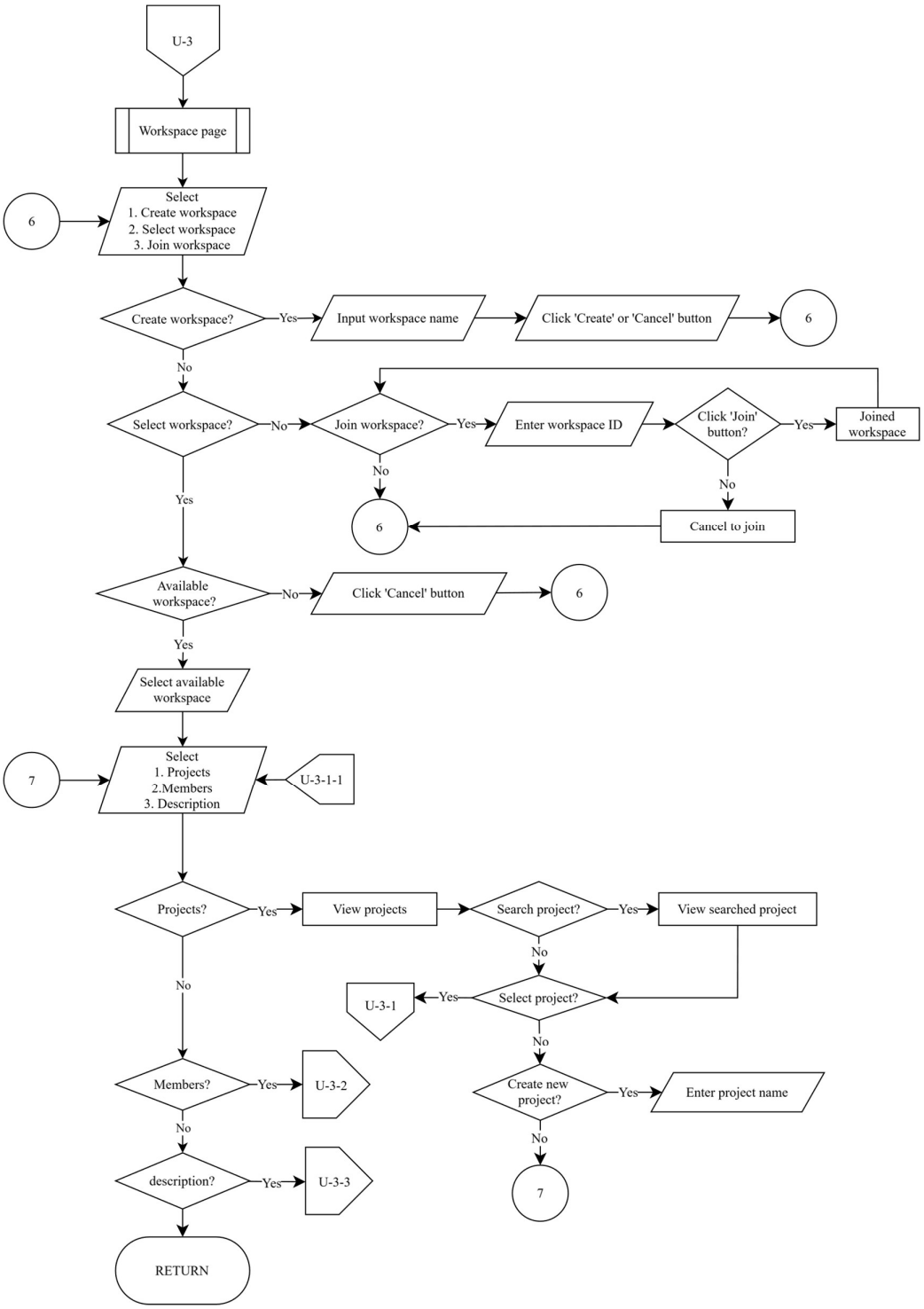


Figure 3.1.1.2.1 Collaboration Features (1)

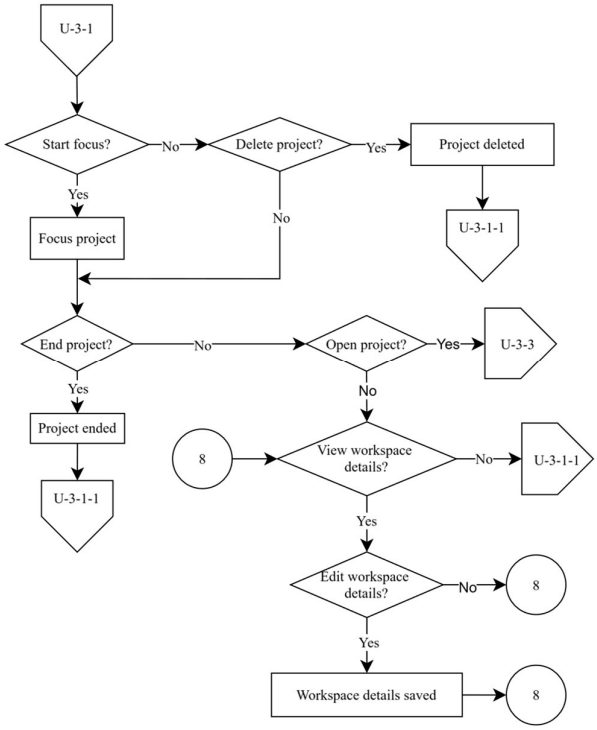


Figure 3.1.1.2.2 Collaboration Features (2)

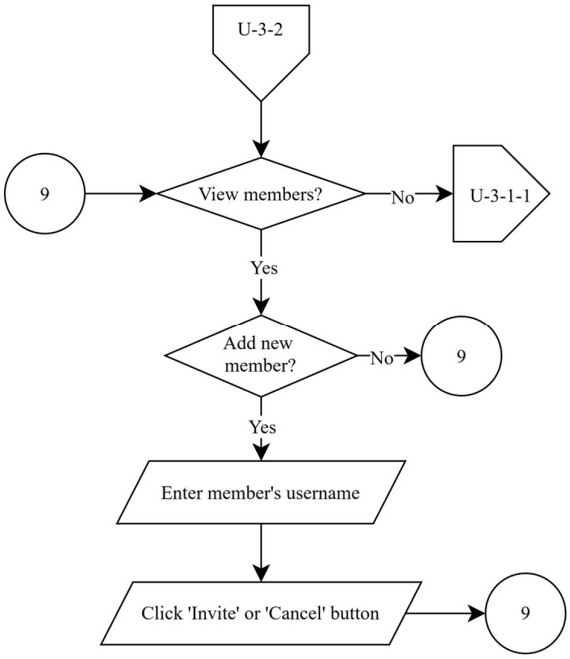


Figure 3.1.1.2.3 Collaboration Features (3)

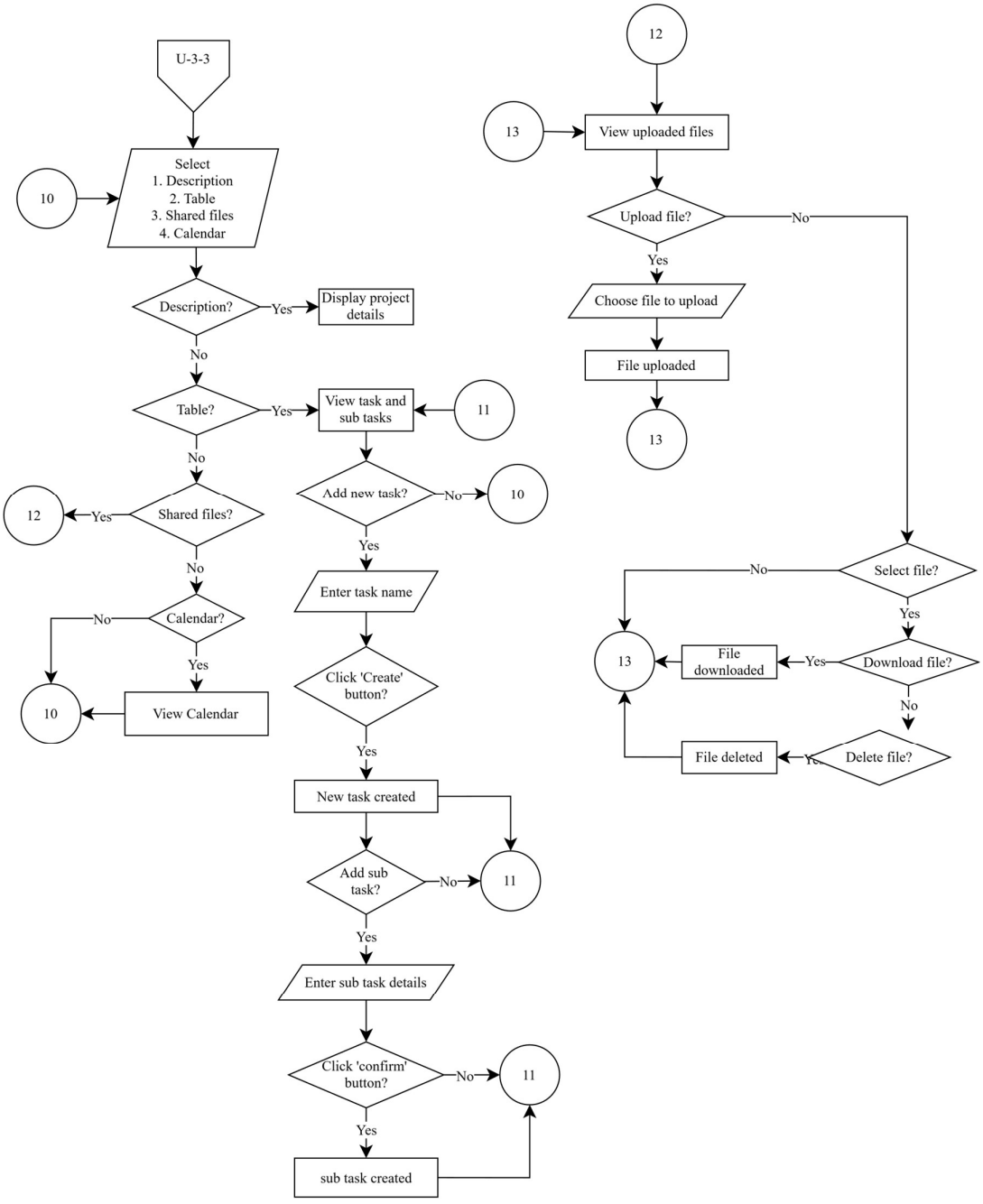


Figure 3.1.1.2.4 Collaboration Features (4)

3.1.1.3 Productivity Analytics

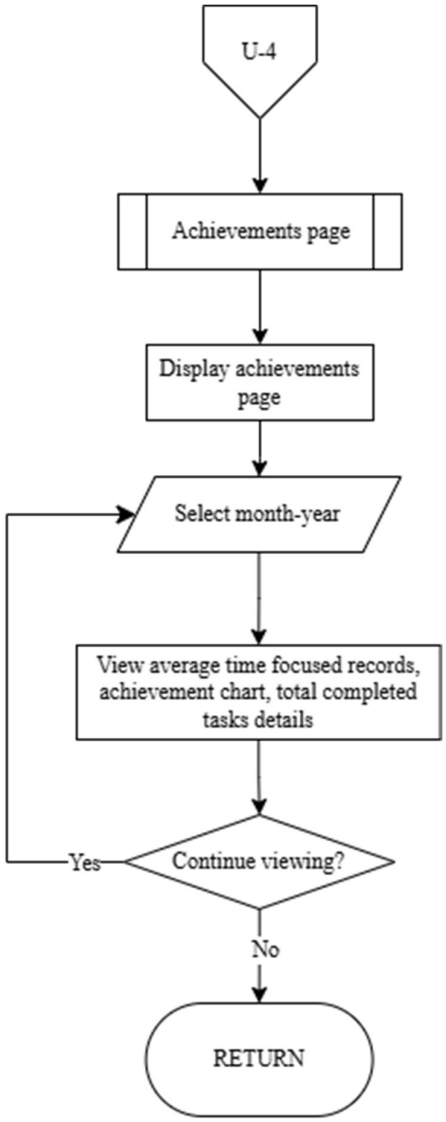


Figure 3.1.1.3 Productivity Analytics

3.1.1.4 Time-tracking

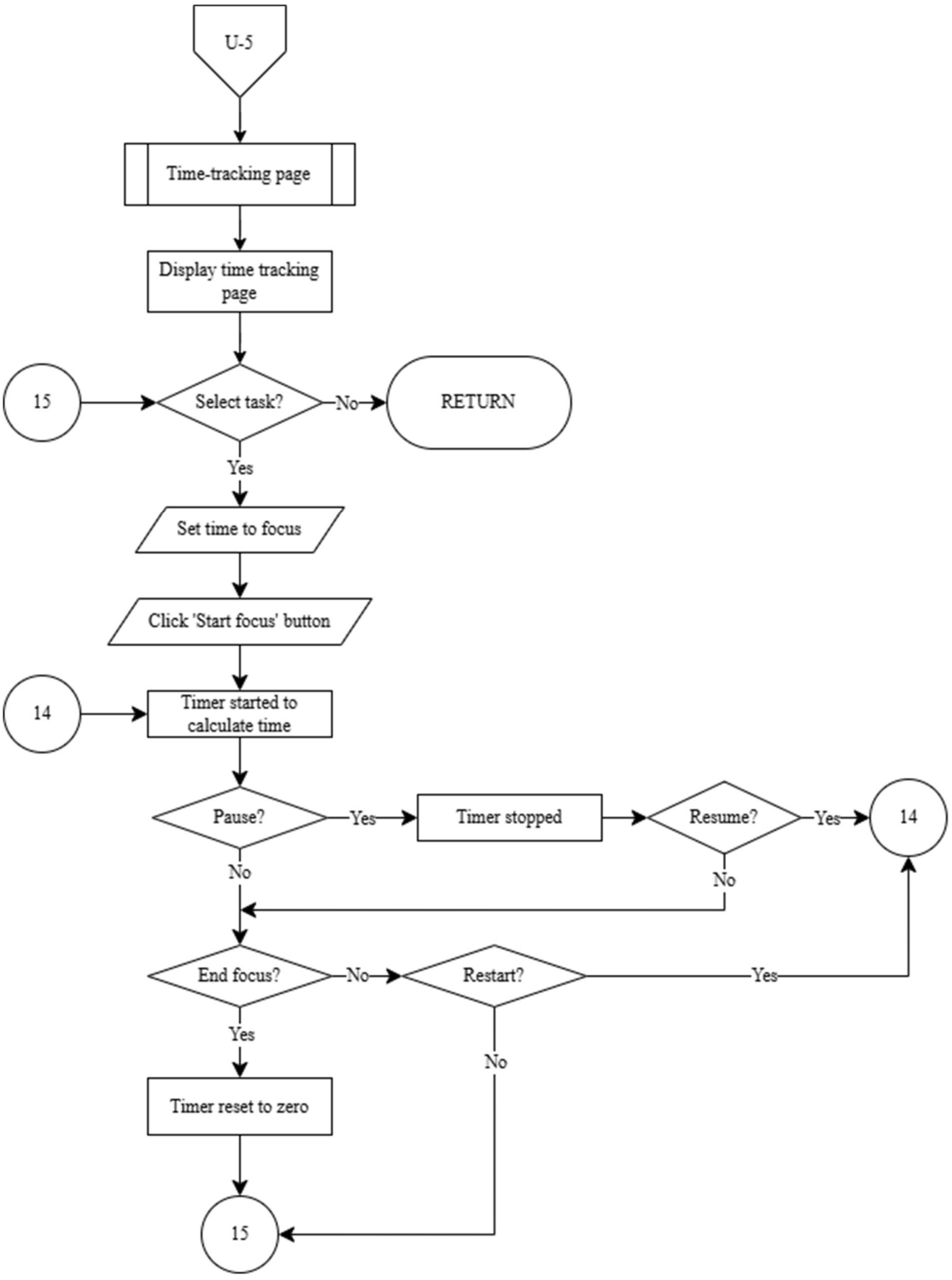


Figure 3.1.1.4 Time-tracking

3.1.1.5 Goal Setting

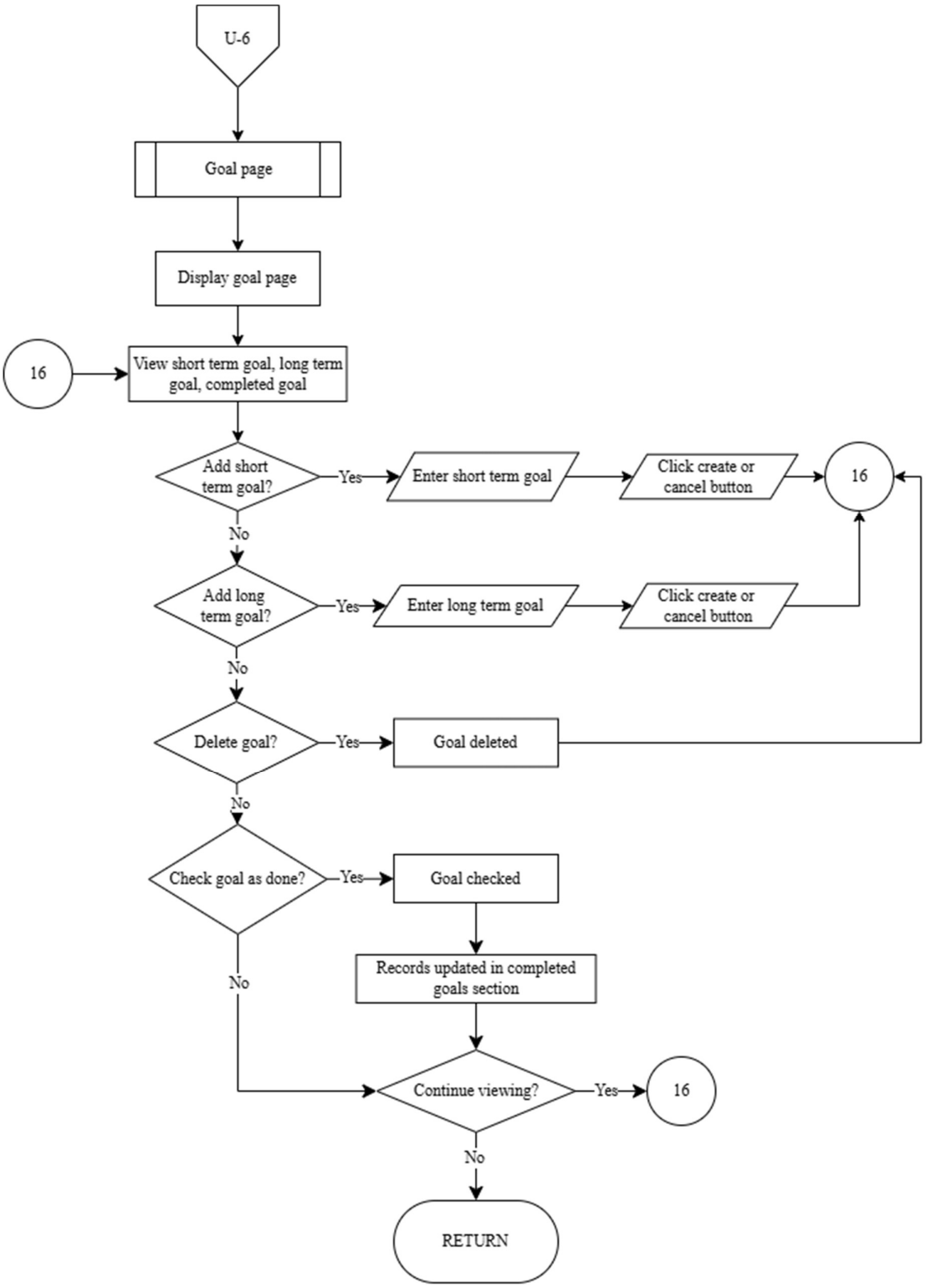


Figure 3.1.1.5 Goal Setting

3.1.2 Admin

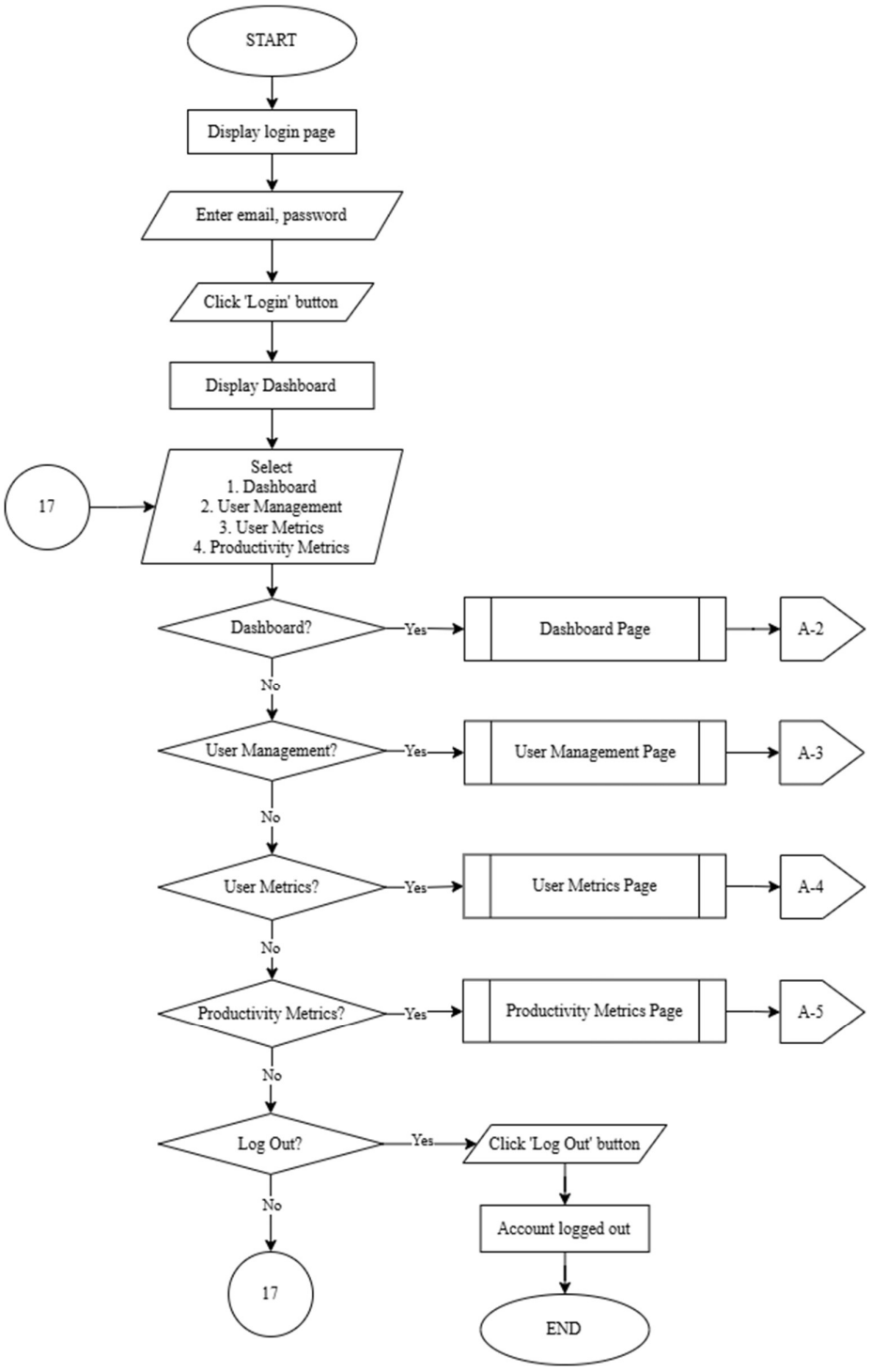


Figure 3.1.2 Admin

3.1.2.1 Dashboard

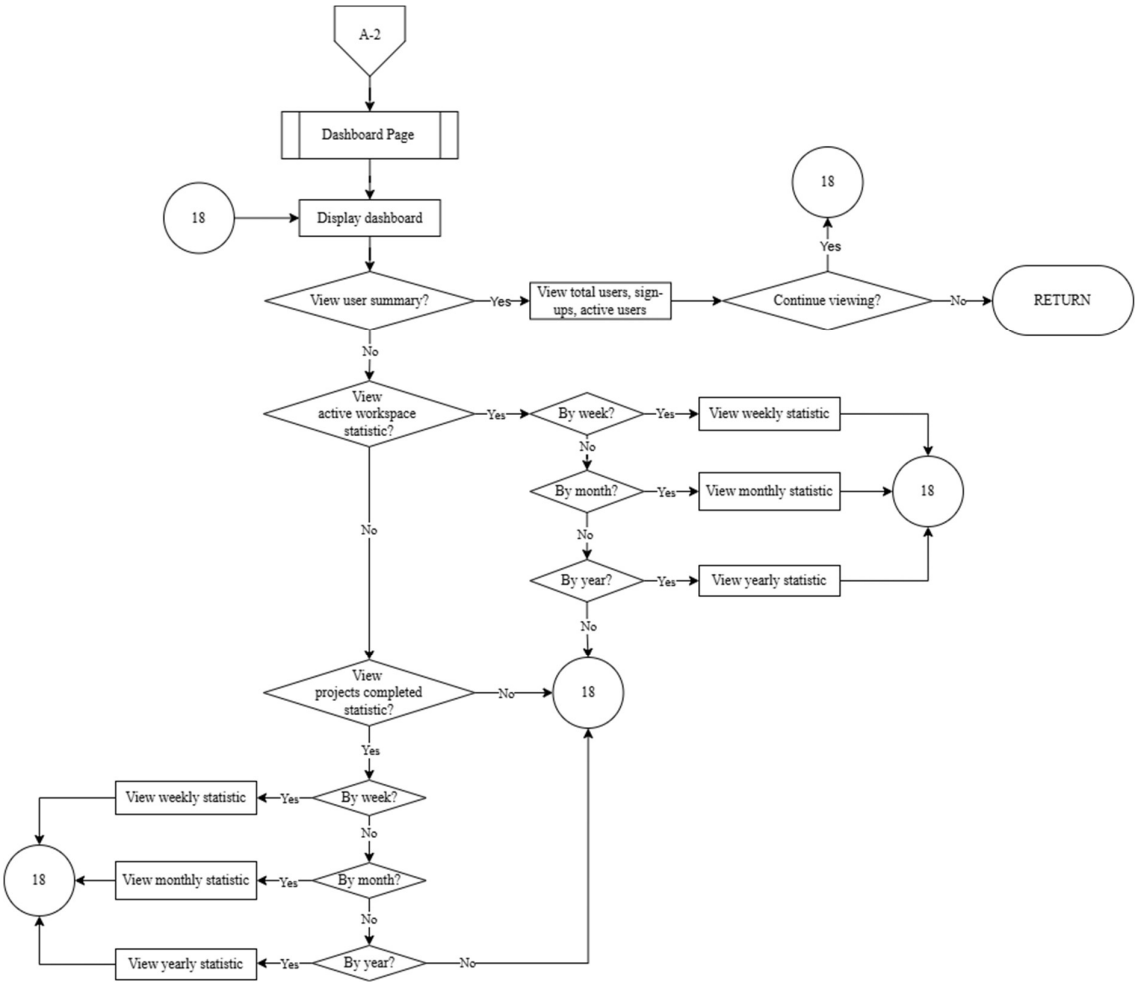


Figure 3.1.2.1 Dashboard

3.1.2.2 User Management

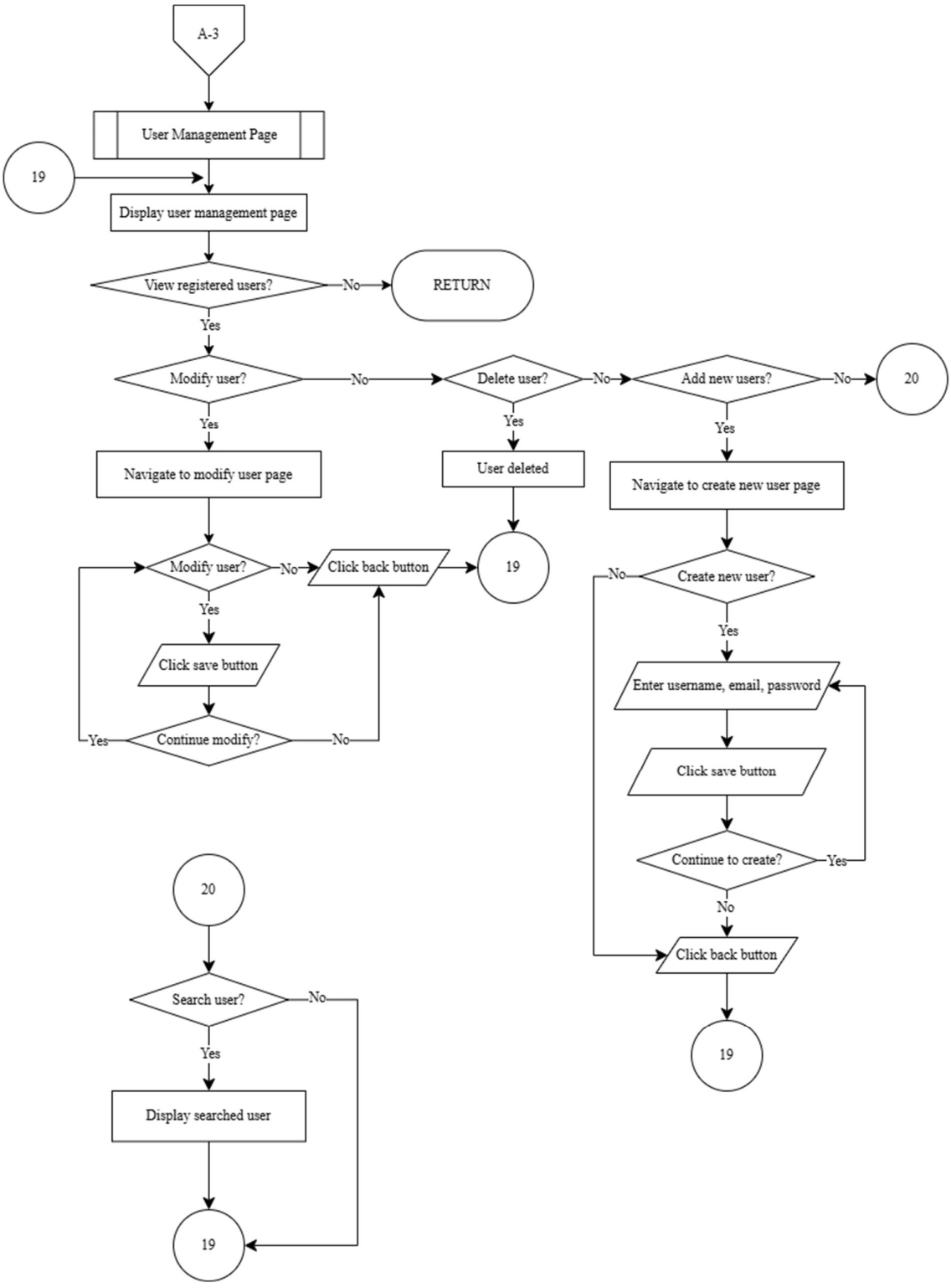


Figure 3.1.2.2 User Management

3.1.2.3 User Metrics

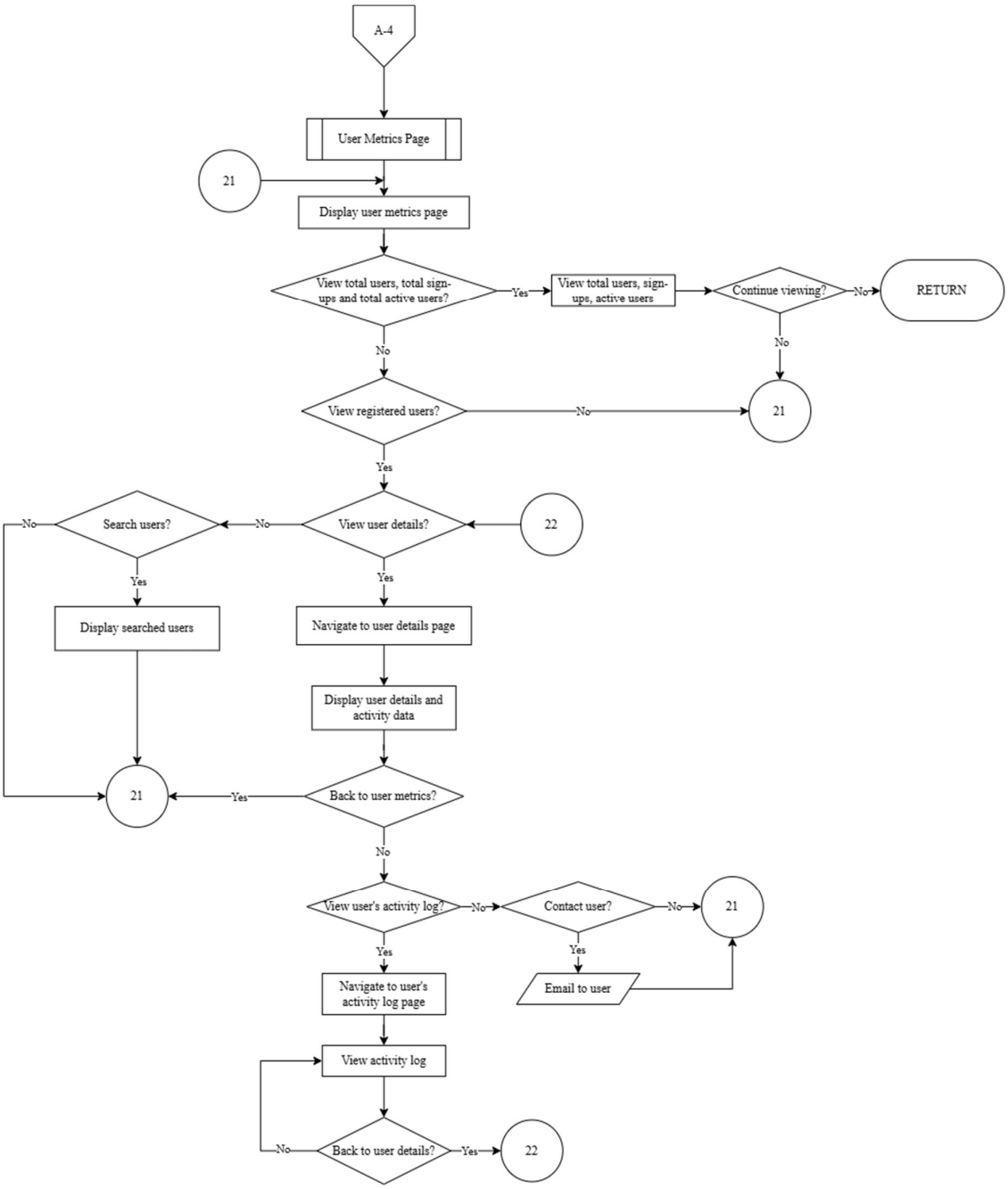


Figure 3.1.2.3 User Metrics

3.1.2.4 Productivity Metrics

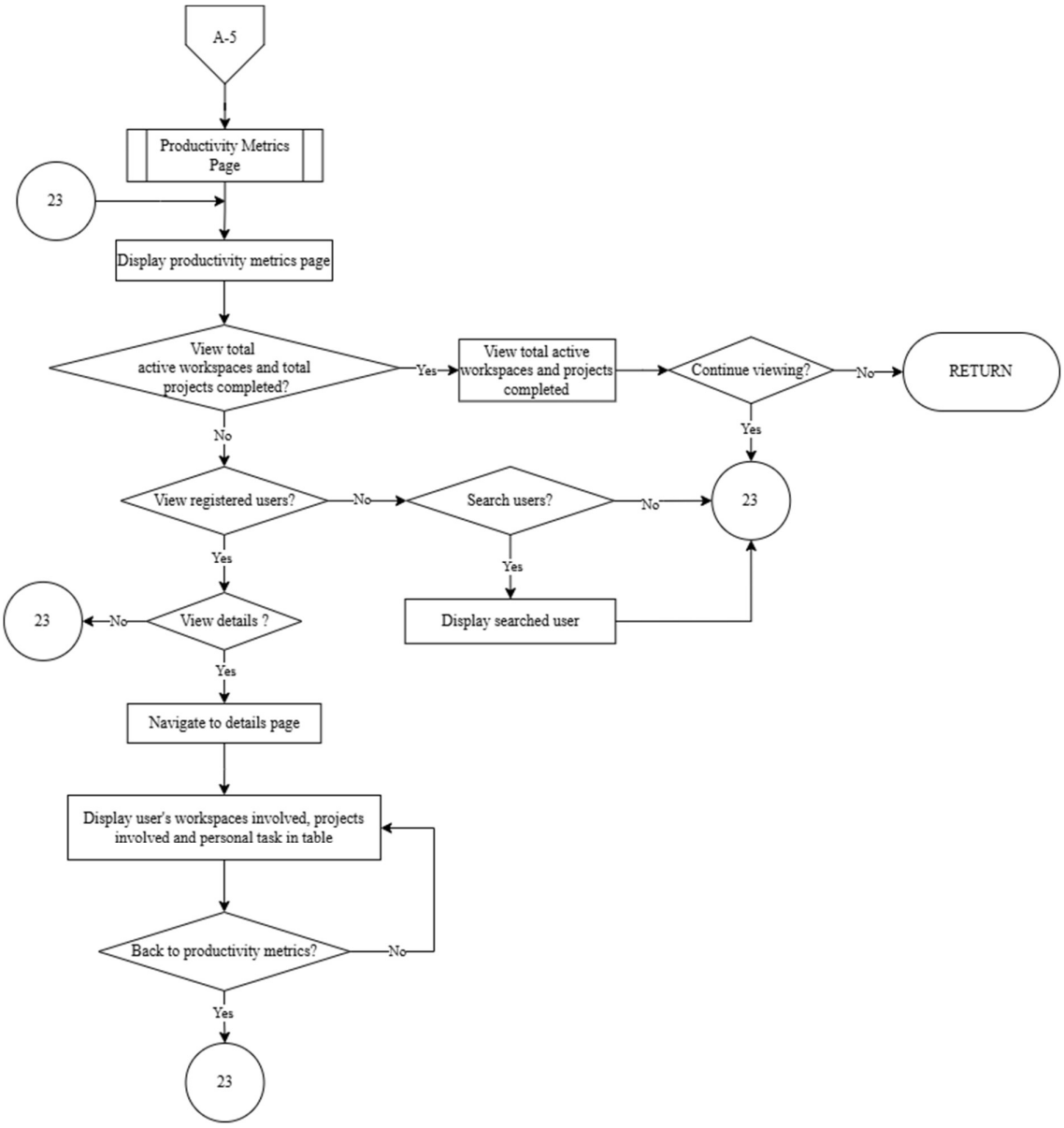


Figure 3.1.2.4 Productivity Metrics

3.2 ERD Diagrams

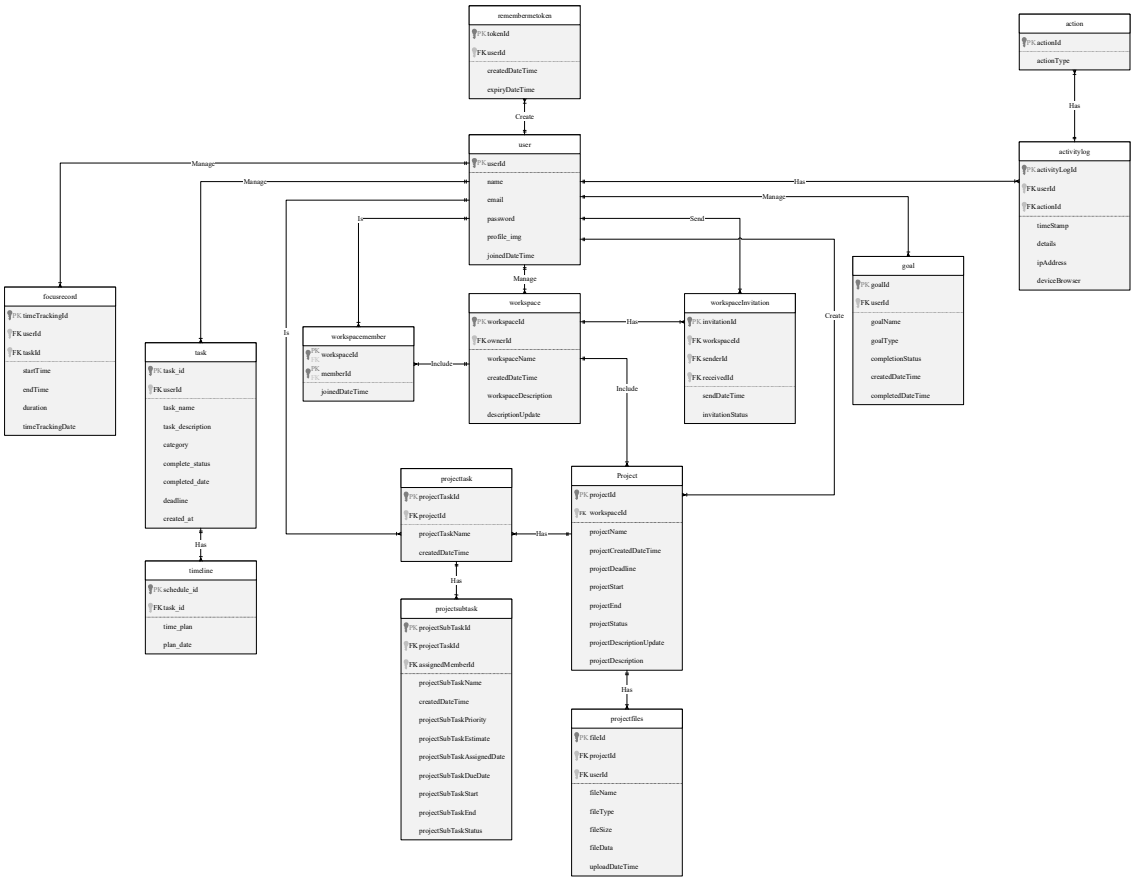


Figure 4.0 ERD Diagram

3.2.1 Data Dictionary

Entity	Attribute	Data Type	Length	Description	Example Input
user	userId	varchar	10	Primary key, a unique identifier for each user	#USR08a1d
	name	varchar	50	Name of user	Mr Meow
	email	varchar	50	Email of user	meowmeow@gmail.com
	password	varchar	100	Hashed password for users to login	$2y$10$5gWikkU1.eBufroTYQBtn..W5fVoV0VRkApAYhXQ2NMMbie43zqES
	profile_img	varchar	255	Profile image path of users	uploads/hello_kitty.jpg
	joinedDateTime	timestamp	N/A	Date and time when users registered	2025-03-17 10:46:26
task	task_id	varchar	10	Primary key, a unique identifier for each to-do task	#TASK0191B
	task_name	text	65535	Name of to-do task	Lunch break
	task_description	text	65535	Brief description for each to-do task	Have a coffee and small walk
	category	enum (	N/A	Task category	plan ahead

		‘urgent’, ‘plan ahead’, ‘handle fast’, ‘on hold’)			
	complete_status	enum (‘past_date’, ‘doing’, ‘done’)	N/A	Complete status of each to-do task	doing
	completed_date	date	N/A	Date when a to-do task is completed	2025-01-06
	deadline	date	N/A	Due date of task	2025-03-22
	created_at	timestamp	N/A	Date and time of each task is created	2025-01-24 12:00:00
	userId	varchar	10	Foreign key, references userId in user	#USR6ed46
timeline	schedule_id	varchar	10	Primary key, a unique identifier for each scheduled task	#TIME12D3F
	time_plan	time	N/A	Scheduled time for each task to complete	09:02:00
	plan_date	date	N/A	Date that users schedule the task	14:45:00

	task_id	varchar	10	Foreign key, references task_id in task	#TASK13105
workspace	workspaceId	varchar	10	Primary key, a unique identifier for each workspace	#WSPCe2af4
	workspaceName	varchar	20	Name of workspace	RWDD Group 1
	createdDateTime	timestamp	N/A	Date and time of workspace created	2025-01-06 22:00:21
	ownerId	varchar	10	Foreign key, references userId in user	#USR6ed46
	workspaceDescription	text	65535	Brief introduction of workspace	A workspace to work together in RWDD
	descriptionUpdate	timestamp	N/A	Date and time of most recent description update	2025-01-07 14:02:52
workspacemembers	workspaceId	varchar	10	Foreign key, references workspaceId in workspace	#WSPCe2af4
	memberId	varchar	10	Foreign key, references userId in user	#USR6ed46

	joinedDateTime	timestamp	N/A	Date and time of joining workspace	2025-01-07 16:50:13
workspaceinvitation	invitationId	int	N/A	Primary key, a unique identifier for each invitation	1
	workspaceId	varchar	10	Foreign key, references workspaceId in workspace	#WSPCcd618
	senderId	varchar	10	Foreign key, references userId in user	#USR4dca5
	receiveId	varchar	10	Foreign key, references userId in user	#USRf081j
	sendDateTime	timestamp	N/A	Date and time the invitation sent	2025-03-02 13:45:11
	invitationStatus	enum(‘accept’, ‘reject’)	N/A	User response status to the invitation	accept
project	projectId	varchar	10	Primary key, a unique identifier for each project	#PRJ3f2bb
	projectName	varchar	50	Name of project	RWDD Productivity Website
	projectCreatedDateTime	timestamp	N/A	Date and time of project created	2025-01-10 13:50:14

	projectDeadline	timestamp	N/A	Deadline set for project	2025-01-10 13:50:14
	projectStart	timestamp	N/A	Date and time of project initiation	2025-03-30 00:00:00
	projectEnd	timestamp	N/A	Date and time of project completed	2025-02-29 16:05:44
	projectStatus	enum (‘pending’, ‘in progress’, ‘complete d’, ‘’)	N/A	Completion status of project	in progress
	workspaceId	varchar	10	Foreign key, references workspaceId in workspace	#WSPCe2af4
	projectDescripti onUpdate	timestamp	N/A	Date and time of project description be updated	2025-02-29 16:05:44
	projectDescripti on	text	65535	Brief introduction of the project	Create a website to track the productivity
projecttask	projectTaskId	varchar	10	Primary key, a unique identifier for each project task	#P-T4c6bb

	projectTaskNam e	varchar	100	Title name of project task	Interface design of website
	createdDateTim e	timestamp	N/A	Date and time the project task created	2025-02-29 16:05:44
	projectId	varchar	10	Foreign key, references projectId in project	#PJR3f2bb
projectsubtask	projectSubTaskI d	varchar	10	Primary key, a unique identifier for each project sub task	#P-ST53556
	projectSubTask Name	varchar	100	Title name of project sub task	Top navigation bar design
	createdDateTim e	timestamp	N/A	Date and time a project sub task created	2025-03-20 08:19:47
	assignedMembe rId	varchar	10	Foreign key, references userId in user	#USR6ed46
	projectSubTask Priority	enum (‘high’, ‘medium’, ‘low’, ‘’)	N/A	Priority level of project task	high
	projectSubTask Estimate	int	N/A	Time estimated for project task completion	4

	projectSubTaskAssignedDate	timestamp	N/A	Date and time of project sub task assigned	2025-03-12 08:08:00
	projectSubTaskDueDate	timestamp	N/A	Date and time of project sub task due	2025-06-10 14:30:30
	projectSubTaskStart	timestamp	N/A	Date and time of project task initiation	2025-03-12 08:08:00
	projectSubTaskEnd	timestamp	N/A	Date and time of project task completion	2025-04-12 08:10:00
	projectSubTaskStatus	enum (‘pending’, ‘in progress’, ‘completed’, ‘’)	N/A	Completion status of project task	completed
	projectTaskId	varchar	10	Foreign key, references ProjectTaskId in projecttask	#P-T4c6bb
projecfiles	fileId	int	N/A	Primary key, a unique identifier for each file uploaded	1
	fileName	varchar	255	Name of file uploaded	RWDD_Proposal.pdf
	fileType	varchar	500	Path of file	RWDD_Group_1/
	FileSize	int	N/A	Size of file in bytes	16560

	fileData	longblob	N/A	Size of file in binary	[BLOB – 16.2KiB]
	projectId	varchar	10	Foreign key, references ProjectId in project	#PRJ3f2bb
	uplooadDateTime	timestamp	N/A	Date and time of file uploaded	2025-03-20 15:22:11
	userId	varchar	10	Foreign key, references userId in user	#USR6ed46
focusrecord	timeTrackingId	varchar	10	Primary key, a unique identifier for each time tracking activity	#TID0b88B
	userID	varchar	10	Foreign key, references userId in user	#USR6ed46
	taskId	varchar	10	Foreign key, references taskId in task	#TASK13105
	startTime	timestamp	N/A	Date and time of starting time tracker	2025-03-20 13:52:10
	endTime	timestamp	N/A	Date and time of stopping time tracker	2025-03-20 15:00:42
	duration	int	N/A	Time-spend from start focus to end	3600

				focus in second	
	timeTrackingDate	date	N/A	Date of tracking activity	2025-01-18
goal	goalId	varchar	10	Primary key, a unique identifier for each goal	#GL2f1d5
	goalName	text	N/A	Title name of goal	Improve cooking skills
	goalType	enum (‘short-term’, ‘long-term’)	N/A	Long-term or short-term goals	short-term
	completionStatus	enum (‘incomplete’, ‘completed’)	N/A	Completion status of goal	completed
	createdDateTime	timestamp	N/A	Date and time of goal created	2025-03-13 08:12:45
	userId	varchar	10	Foreign key, references userID in user	#USR6ed46
	completedDateTime	timestamp	N/A	Date and time of goal completed	2025-04-13 08:12:45
action	actionId	varchar	10	Primary key, a unique identifier for	A001

				each user action	
	actionType	varchar	50	Type of system action performed	Created a Workspace
activitylog	activityLogId	int	N/A	Primary key, a unique identifier for each user activity	2
	userID	varchar	10	Foreign key, references userId in user	#USR6ed46
	timestamp	timestamp	N/A	Date and time of activity	2025-01-06 22:00:21
	actionId	varchar	10	Foreign key, references actionId in action	A001
	details	varchar	800	Specific description of activity happened	Create a new workspace named “RWDD Group 1”
	ipAddress	varchar	20	IP address of user device	192.168.1.2
	deviceBrowser	varchar	50	Type of device browser	Chrome on Windows 10
remembermetoken	tokenId	binary	32	Primary key, a unique identifier for each	4ad0ff1f959dcdc8f005cae7b359efc0990591b518d3ed

				rememberMe token	bb1e585779e520433f
	userId	varchar	10	Foreign key, references userId in user	#USR6ed46
	createdDateTime	datetime	N/A	Date and time of token created	2025-03-20 23:39:43
	expiryDateTime	datetime	N/A	Expiry date of the token	2025-04-19 23:39:43

3.3 Wireframe

Figure 5.1, Figure 5.2 & Figure 5.3 Register Page, User Home Page & Login Page

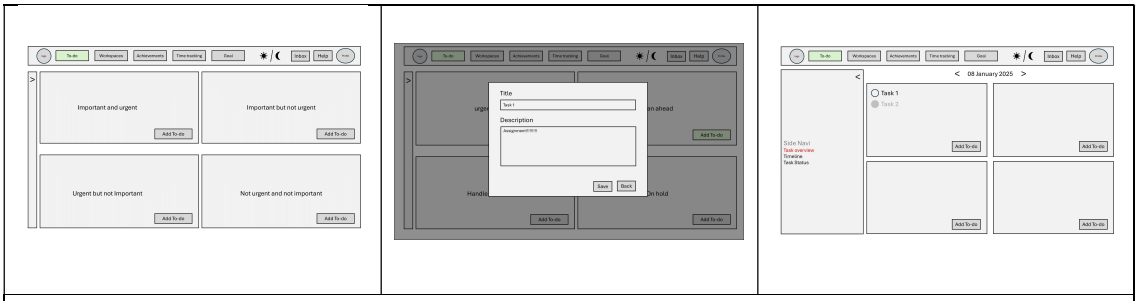


Figure 5.4, Figure 5.5 & Figure 5.6 To-Do Pages

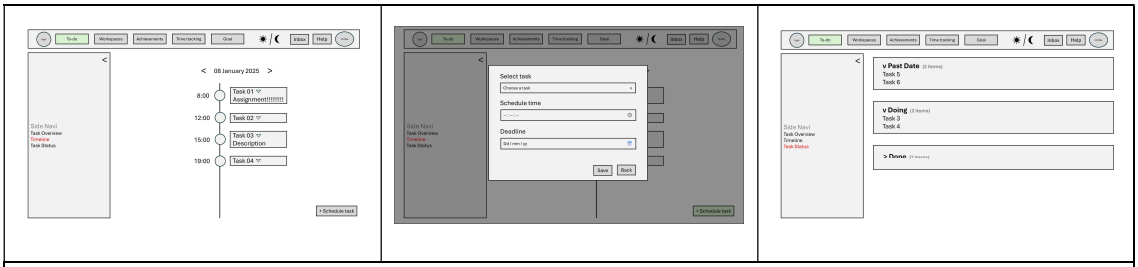


Figure 5.7, Figure 5.8 & Figure 5.9 To-Do Pages

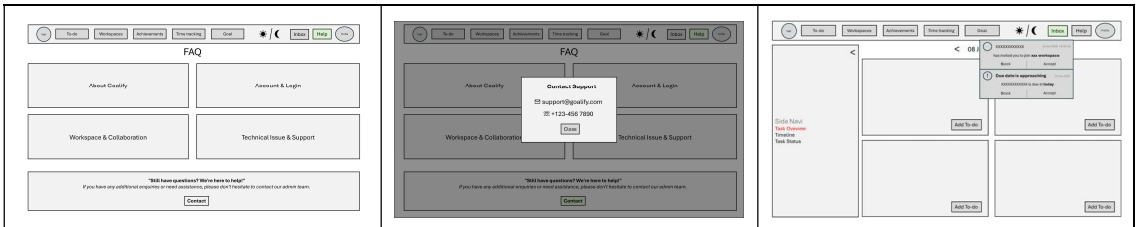


Figure 5.10, Figure 5.11 Help Pages || Figure 5.12 Inbox Page

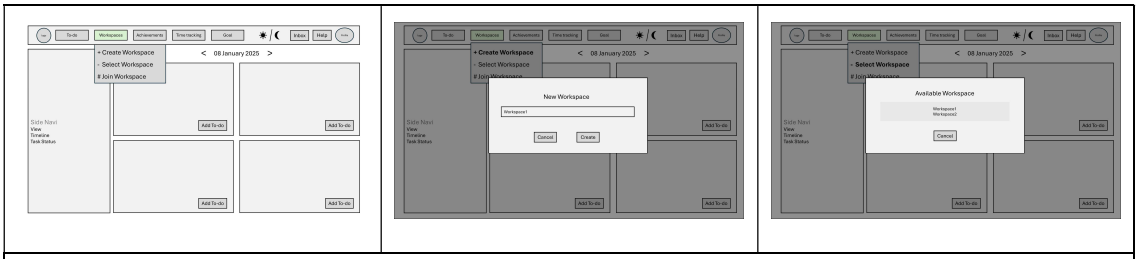


Figure 5.13, Figure 5.14 & Figure 5.15 Workspaces Pages

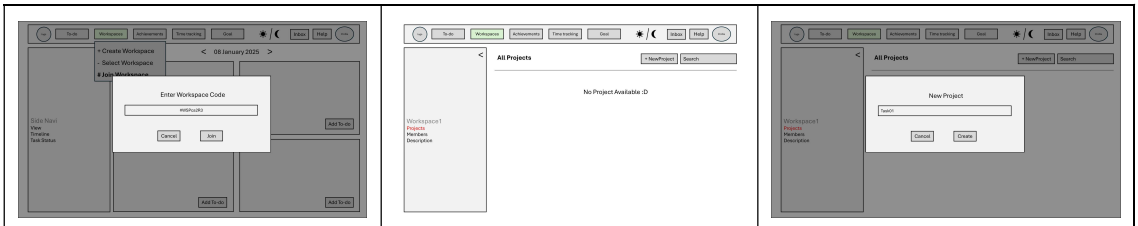


Figure 5.16, Figure 5.17 & Figure 5.18 Workspaces Pages

Figure 5.19, Figure 5.20 & Figure 5.21 Workspaces Pages

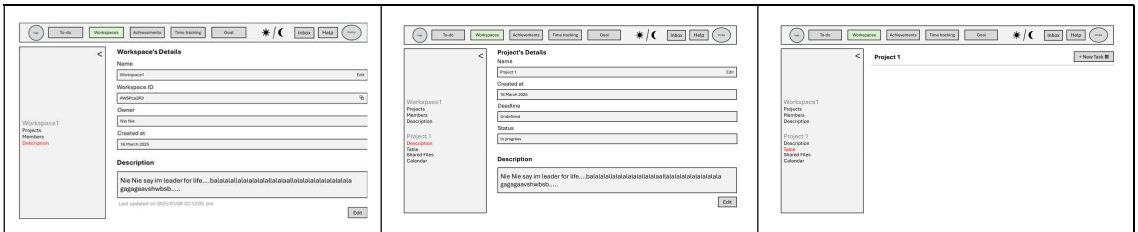


Figure 5.22, Figure 5.23 & Figure 5.24 Workspaces Pages

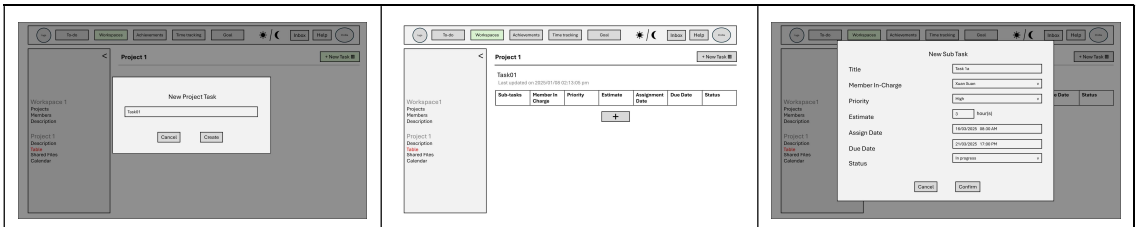


Figure 5.25, Figure 5.26 & Figure 5.27 Workspaces Pages

Figure 5.28, Figure 5.29 & Figure 5.30 Workspaces Pages

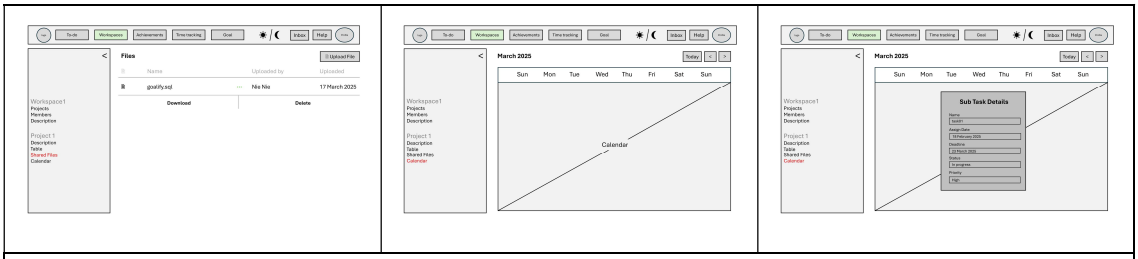


Figure 5.31, Figure 5.32 & Figure 5.33 Workspaces Pages

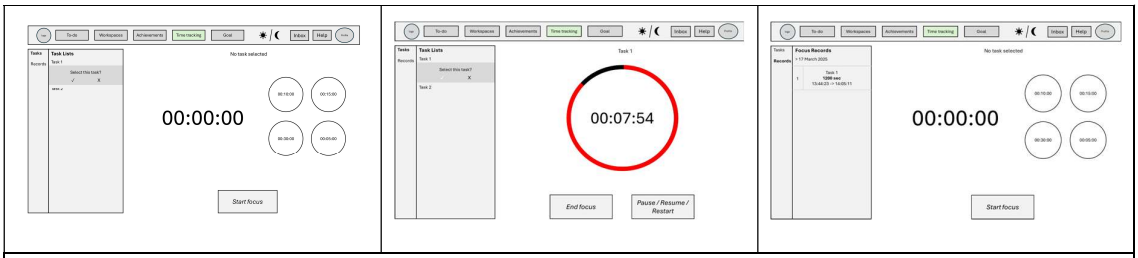
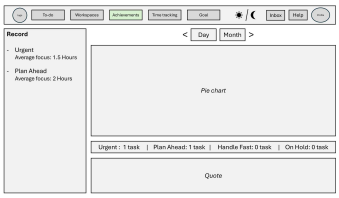


Figure 5.34 & Figure 5.35, Figure 5.36 Time-Tracking Page



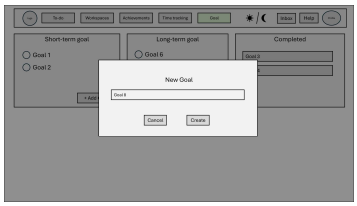


Figure 5.37 Achievement Page || Figure 5.38 & Figure 5.39 Goal Pages

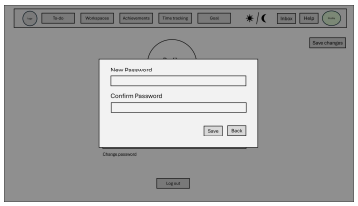
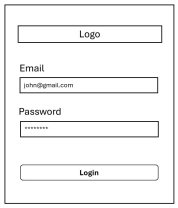


Figure 5.40 & Figure 5.41 Profile Pages



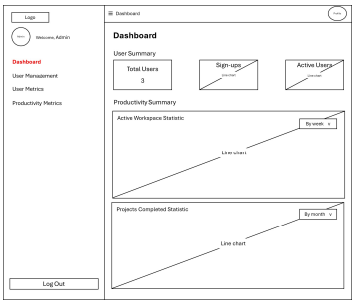


Figure 5.42 Admin Login Page & Figure 5.43 Admin Dashboard Page

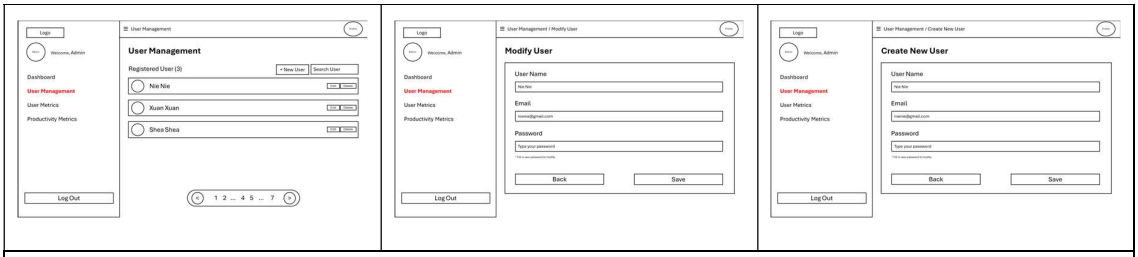


Figure 5.44, Figure 5.45 & Figure 5.46 User Management Pages

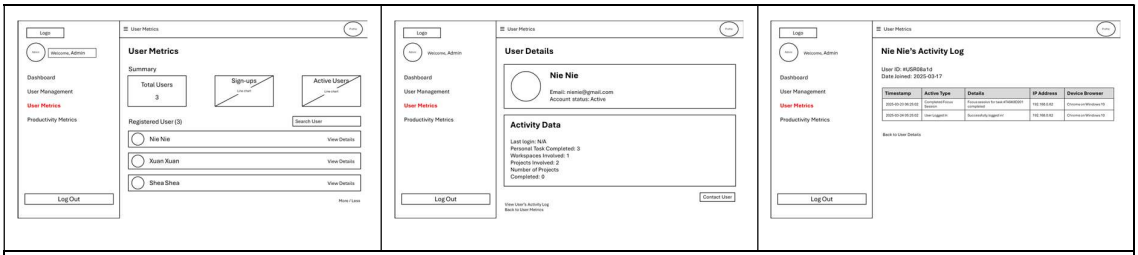
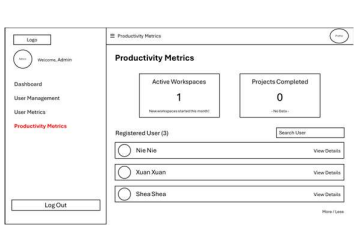


Figure 5.47, Figure 5.48 & Figure 5.49 User Metrics Pages



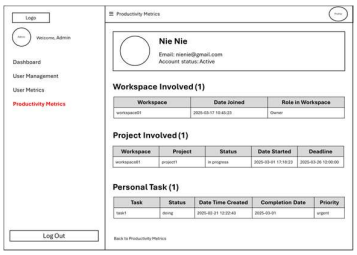


Figure 5.50 & Figure 5.51 Productivity Metrics Pages

3.4 Navigational Structure

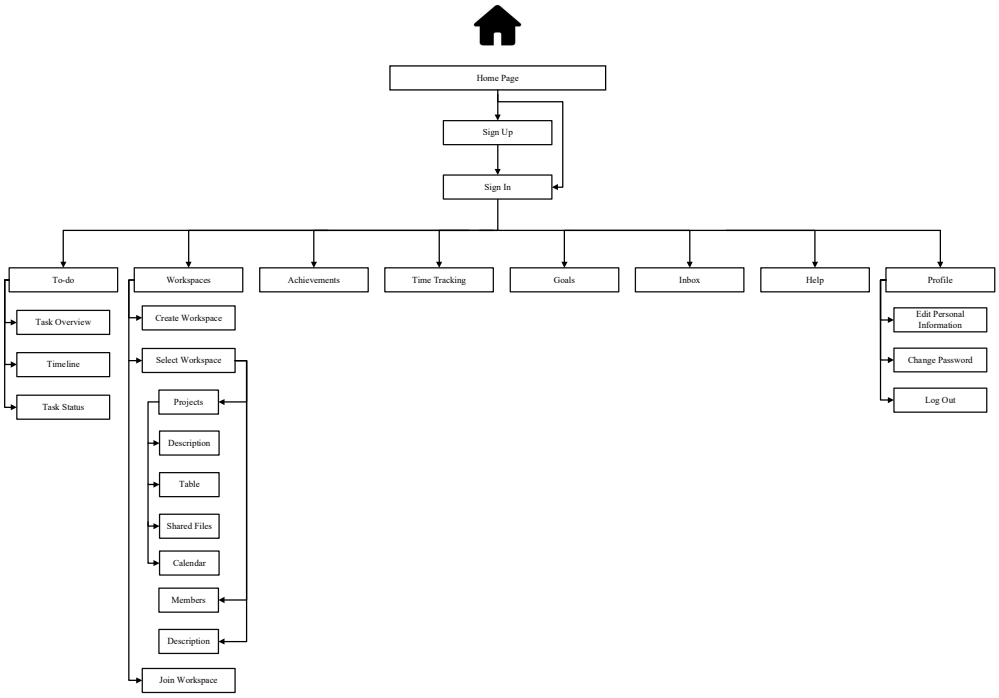


Figure 6.1 User's Navigational Structure

Firstly, the website starts with basic **user authentication**, allowing users to either sign in via their existing account or sign up for a new account. After users successfully log into their account, they can access the website.

The proposed website consists of certain task management features. A **to-do feature** is provided, allowing users to view or add tasks to a personalized to-do list, view task timeline and task status. A **workspace area** is dedicated within the website, where users can create workspace for multi-user collaboration. They may also view the details of workspaces available, including projects, members and description.

Other than that, an **achievements** section is also introduced to allow users to track their achievements over a specified period. Additionally, a **time tracking feature** is implemented, enabling users to track their time while focusing on a specific task. Users can also set their long-term and short-term goals under the **goal section**.

An inbox is included for pushing notifications such as task deadline reminders and workspace invites. Furthermore, **a help section** is available for users to get assistance in resolving problems faced while using the website. Lastly, users may **click on their profile** to edit personal information, modify their account password and log out of the website.

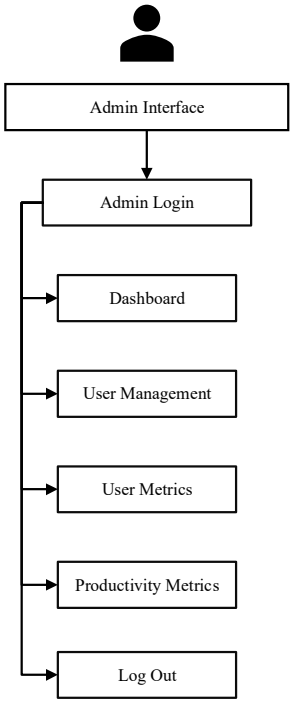


Figure 6.2 Admin's Navigational Structure

System Admin may **login** via the **Admin Login Interface**, providing dedicated admin authentication to access the system administration page, managing basic website operational functions including viewing the dashboard, performing user management, monitoring user metrics, overseeing user productivity metrics and logging out of the system.

4.0 Implementation

To develop the 'Goalify' productivity enhancement web application, the following development tools and technologies are applied:

- Visual Studio Code: A code editor used for managing, organizing and writing HTML, CSS, JavaScript and PHP scripts.
- WAMP server: A web development platform used for setting up a local web server environment on local computers for deploying the 'Goalify' web application. It includes an Apache server, MySQL database, MariaDB and PhpMyAdmin for database management.
- MySQL: A relational database management system (RDBMS) used to store and manage data manipulated within the application (*What Is MySQL?*, 2023).
- Google Chrome or Microsoft Edge: Browser used to debug and test the front-end interface of the 'Goalify' web application.

The following sections will explain the implementation concepts and steps in building the 'Goalify' web application, encompassing setting up the system environment, defining the database schema, clarifying the implementation of each key features and justifying the responsiveness of the web application.

4.1 System Environment and Database Configuration

To set up a local web server environment for the 'Goalify' web application locally, the WAMP server is installed. WampServer is an acronym for Windows, Apache, MySQL and PHP, providing a centralized approach for running a local development environment. The primary components include Apache (a web server processing PHP scripts and serves web pages), MySQL and PHP. By installing WAMP server, the application can be tested in a realistic web environment before actual deployment.

After installing the WAMP server, the configuration of the MySQL is conducted for data management. By using PhpMyAdmin, a software tool that allows web-based administration of the MySQL database, a database for the 'Goalify' application is created. The setup involves:

1. Through http://localhost, access the PhpMyAdmin 5.2.1 under 'Your Aliases'.
2. Create the database and name it as 'goalify' to store user accounts, tasks, workspaces, etc.

To ensure consistency across different installations, the database structure was exported as a 'sql' file, allowing easy database restoring by importing the file into PhpMyAdmin. To import the database:

1. Login into PhpMyAdmin and open the created database.
2. Navigate to the 'Import' tab, click 'Choose File', select the provided 'goalify.sql' file and click 'Go' to import.
3. Verify that all tables and data have been successfully imported.

By importing the predefined database structure, it allows the application to function as intended with all the required tables, relationships and data.

4.2 Implementation Justification

The ‘Goalify’ web application offers core features that enhance user productivity and collaboration. The following sections outline the key user and admin features, together with the implementation details for each feature.

4.2.1 User Features

4.2.1.1 User Authentication

- PHANG SHEA WEN-

Goalify - userHandler.php

```php
function login($pdo) {
    $email = trim($_POST['login-email']);
    $userEmail = filter_var($email, FILTER_SANITIZE_EMAIL); // Remove illegal characters
    $userPwd = $_POST['login-pwd'];
    if (isset($_POST['remember-me'])) {
        $rememberMe = $_POST['remember-me'];
    } else {
        $rememberMe = null;
    }
    $stmt = $pdo -> prepare('SELECT * FROM user WHERE email = :email');
    // not necessarily to manually check for SQL statement errors for pdo like in mysqli
    $stmt -> execute([':email' => $userEmail]);
    $user = $stmt -> fetch();
    if ($user && password_verify($userPwd, $user['password'])) {
        createUserSession($user['userId'],'registered-user', $pdo, '/Goalify/Pages/user/todo/task-overview/task-overview.php', isset($rememberMe));
        insertActivityLog($pdo, $user['userId'], 'A001', "Successfully logged in !");
        return "Login Successfully";
    } elseif (!$user) {
        $loginErrorMsg = 'Email not found'; // echo in js, use js to insert error message
        return $loginErrorMsg;
    } elseif (!password_verify($userPwd, $user['password'])) {
        $loginErrorMsg = 'Invalid password';
        return $loginErrorMsg;
    }
}
```

Figure 4.2.1.1.1 User Login

Goalify - session.php

```php
function createCookie(string $userId, PDO $pdo) {
    // raw binary data 32 bytes (256 bits), 1 byte = 8 bits
    // bin2hex converts to hexadecimal characters, 64 characters, 1 byte = 2 hex char
    $token = random_bytes(32); // store raw binary data in database
    $expiry = time() + (30*60*60*24);
    $expiryDateTime = date('Y-m-d H:i:s', $expiry);  // Current date and time in DATETIME format
    $tokenStmt = $pdo -> prepare('INSERT INTO rememberMeToken(tokenId, userId, createdDateTime, expiryDateTime) VALUES (:tokenid, :userid, :created, :expiry);');
    $tokenStmt -> execute([':tokenid' => $token, ':userid' => $userId, ':created' => date('Y-m-d H:i:s', time()), ':expiry' => $expiryDateTime]);
    $cookieData = ['userId' => $userId, 'tokenId' => bin2hex($token)]; // bind2hex converts binary data to hexadecimal in lowercase
    setcookie("rememberMe", serialize($cookieData), $expiry, "/", "", false, true);
}
```

Figure 4.2.1.1.2 Create Cookie Session

Goalify - session.php

```php
function checkCookie(PDO $pdo) {
    if (isset($_COOKIE['rememberMe'])) {
        $cookieData = unserialize($_COOKIE['rememberMe']);
        $tokenId = $cookieData['tokenId'];
        $userId = $cookieData['userId'];
        $tokenStmt = $pdo -> prepare('SELECT HEX(tokenId) AS tokenId, expiryDateTime FROM rememberMeToken WHERE userId = :userId'); // HEX converts binary data to hexadecimal in upper case
        $tokenStmt -> execute([':userId' => $userId]);
        $storedToken = $tokenStmt -> fetch();
        // strtotime -> converts datetime in SQL to Unix Time stamp
        if ($storedToken) {
            if ((strtolower($storedToken['tokenId']) == $tokenId) && (strtotime($storedToken['expiryDateTime']) > time())){
                // var_dump($_COOKIE['rememberMe']);
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

Figure 4.2.1.1.4 Check Cookie Session

Aspect	Description	Implementation
Form Handling	Users enter email and password in a login form.	HTML, CSS, JavaScript (Form Validation)
Email and Password verification	Checks if the entered email exists in the database and verifies if the entered password matches with the hashed password stored in the database. If login credentials are invalid, show error message.	• MySQL: Use prepared statements (`SELECT query`) to retrieve all registered user email. • PHP: '`password_verify()`' to compare entered password with the stored hashed password. • HTML: show error message passed from the back-end.
Session Handling	Creates a session upon successful login to store essential user data such as the user ID, role of the user, login time and last activity time. The session data may be updated in the following programs for further data manipulation. A session timeout is set to 1 hour of inactivity for security purposes.	• PHP: use '`session_start()`' to start session and '`$_SESSION`' to store session data.
Cookie Creation	Creates a 'remember me' cookie if the user checks the 'Remember Me' option upon successful login. The expiry duration of the cookie session is 30 days. The cookie stores the cookie token ID and user ID as serialized data, with an expiration of 30 days.	• PHP: use '`setcookie()`' to create cookie and '`$_COOKIE['rememberMe]`' to store cookie session data.

Security Measures	Prevent SQL injection and cross-scripting (XSS) during login	• PHP: use prepared statements to avoid SQL injection and input sanitization for security.

Goalify - userHandler.php

```php
function register($pdo, $registerName, $registerEmail, $registerPwd) {
    $strongPwd = validatePasswordStrength($registerPwd);
    if ($strongPwd) {
        $hashedPwd = password_hash($registerPwd, PASSWORD_BCRYPT);
        $stmt = $pdo -> prepare('SELECT * FROM user WHERE email = :email');
        $stmt -> execute([':email' => $registerEmail]);
        $result = $stmt -> fetch();
        if ($result) {
            $registerErrorMsg = 'This account is already created';
            return $registerErrorMsg;
        } else {
            $userId = "#USR" . strtolower(substr(md5(uniqid()), 0, 5));
            $stmt = $pdo -> prepare('INSERT INTO user(userId, name, email, password) VALUES (:id, :name, :email, :pwd)');
            $stmt -> execute([':id' => $userId, ':name' => $registerName, ':email' => $registerEmail, ':pwd' => $hashedPwd]);
            return "Registered Successfully";
            // header("location: $page");
        }
    } else {
        $registerErrorMsg = 'Weak password';
        return $registerErrorMsg;
    }
}
```

Figure 4.2.1.1.2 User Registration

Aspect	Description	Implementation
Form Handling	Users enter name, email and password in a registration form.	HTML, CSS, JavaScript (Form Validation)
Password Hashing	Passwords are hashed before being stored in the database for security purposes.	• PHP: use '`password_hash()`' to encrypt the entered password.
User Data Storage	Insert new user record into the database after validation.	• MySQL: use prepared statements (`INSERT query`) to insert new user record into the database.
Email Duplicate Check	Checks if entered email exists in the database, avoid duplicate email registration.	• MySQL: using prepared statements (`SELECT query`) to retrieve all registered user email.

Validation and Error Handling	Enforce a strong password policy, requiring special characters, uppercase or lowercase letters and digits.	• JavaScript: validates the strength of the password before submission.
Security Measures	Prevent SQL injection and cross-scripting (XSS) during login	• PHP: use prepared statements to avoid SQL injection and input sanitization for security.

4.2.1.2 Task Management

- NG YVONNE-

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    header('Content-Type: text/plain');
    if (isset($_POST["task-name"])) {
        $task_name = trim($_POST["task-name"]);
        $task_description = trim($_POST["task-description"] ?? "");
        $task_category = trim($_POST["category"] ?? "");
        $task_complete_status = trim($_POST["complete_status"] ?? "");
        $user_id = $_SESSION["user_id"] ?? "";
        $task_id = generateTaskId();
        error_log($task_id);
        ob_flush();
        flush();
    }

    try {
        $stmt = $pdo->prepare("INSERT INTO task (task_id, task_name, task_description, category, complete_status, userId) VALUES (:task_id, :task_name, :task_description
        $stmt->bindParam(":task_id", $task_id);
        $stmt->bindParam(":task_name", $task_name);
        $stmt->bindParam(":task_description", $task_description);
        $stmt->bindParam(":category", $task_category);
        $stmt->bindParam(":complete_status", $task_complete_status);
        $stmt->bindParam(":user_id", $user_id);

        if ($stmt->execute()) {
            echo "Task saved successfully!";
            insertActivityLog($pdo, $user_id, "A039", "Added task '$task_name'");
            exit;
        } else {
            echo "Failed to save task.";
            exit;
        }
    }catch (Exception $e) {
        echo "Error: " . $e->getMessage();
    }
```

Figure 4.2.1.2.1 Save task

Aspect	Description	Implementation
Task Category	Categorized into 4 main groups: Urgent, Plan Ahead, Handle Fast, On Hold.	• HTML: use 'data-category' to represent each category JavaScript: target the closest 'div' that has '.category' class and get the attribute 'data-

		category' to correctly categorize the tasks that are added to be displayed in the front-end later. • CSS: use ' :empty::before ' to set the content of 'div' as 'Add Task…' when no task is added. Word-wrap, overflow-wrap and work-break property are used to prevent the content displayed exceed the 'div'.
Add to-do task	Each task category has an 'Add to do' button for users to add task. An overlay form will appear for users to fill out the name of task and description when user click on the button. Users cannot add tasks for the previous date.	• HTML: use button tag to create 'Add to do' button • JavaScript: date After clicking the button, use backticks format for '`createOverlay()`' to create a dynamic HTML form only if the chosen date is not previous date.
'Back' and 'Save' button	The overlay will be removed after clicking these two buttons. 'Back' button will not affect anything while 'Submit' button will submit the overlay form to save the data into database.	• JavaScript: use backticks format to dynamically create buttons for 'back' and 'Save'. Overlay is removed using '`removeOverlay()`' after clicking the buttons.
Form submission	When users click on the 'Save' button, the input details will be posted to the server. The completed status of each task is considered as 'doing' initially.	• JavaScript: create a new `FormData` object and appends the necessary input details, using '`POST`'

		method to send to the server. The fetch function is used to connect with PHP that handle the form submission.
Task handler	After the form is posted, it will be received by the server. PHP will carry out a series of processes to get the posted input, bind with parameters and finally save to database.	• PHP: use `isset($_POST[""])` to check whether the data is posted correctly. • MySQL: use prepared statement (`INSERT query`) to insert new task record into the database.
Generate unique id for tasks	A unique ID for each task will be generated upon posted.	• PHP: use '`generateTaskId()`' to generate a random MD5 hash unique ID in 32-character hexadecimal string, extracts the first 5 characters, converts them to uppercase and concatenate with prefix '`#TASK`'.

```php
$current_date = date("Y-m-d");

$updateStmt = $pdo->prepare("UPDATE task SET complete_status = 'past_date' WHERE userId = :user_id
                        AND deadline IS NOT NULL
                        AND (deadline < :current_date OR deadline < created_at)
                        AND TRIM(complete_status) != 'done'");
$updateStmt->execute([
    ":user_id" => $user_id,
    "current_date" => $current_date
]);

if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $result = $pdo->prepare("SELECT * FROM task WHERE userId = :user_id AND (deadline >= :current_date OR deadline IS NULL) ORDER BY created_at ASC");
    $result->execute([
        ":user_id" => $user_id,
        "current_date" => $current_date
    ]);

    $row = $result->fetchAll(PDO::FETCH_ASSOC);

    echo json_encode($row);
}
```

Figure 4.2.1.2.2 Fetch task to display

Aspect	Description	Implementation
Display task record according to date	The matched tasks will be displayed according to the user's selected date.	• JavaScript: use '`loadTaskByDate()`' to fetch data from database and display according to date chosen by users. • PHP: use '`$_SERVER["REQUEST_METHOD"] == "GET"` ' works with MySQL prepared statement (`SELECT query`) to retrieve desired task records from database.
Add a checkbox to each task	Users can mark their tasks as done or undone with the checkboxes.	• JavaScript: use querySelector to select the target 'div', dynamically create a checkbox-typed input element inside each div and a span element for respective text. It also used 'click' `eventListener` to style the checked and unchecked task by using CSS-in-JS approach.
Update task status	When users have checked the task, the complete status will change from 'doing' to 'done'. If they unchecked, the status would return to the original state, 'doing'.	• JavaScript: use new '`FormData()`' to append the changed status and some details to be posted to the server. • MySQL: use prepared statement (`UPDATE query`) to update the complete status.
Format date and time	Format the date and time before displaying it on the front-end or	• JavaScript: use '`toLocaleDate`

	posting to back-end for compatibility and easy reading.	`String("en-CA")` ' to compare date with new `Date().toISOString().split("T")[0]` is used format the date and save to database.
Date picker	The date picker is placed at the top for users to choose their desired date by directly selecting the date from the dropdown calendar or using arrow beside the date.	• JavaScript: use date picker library, '`Flatpickr`', to enable users to choose the date dynamically.

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    header("Content-Type: application/json");

    try {
        if (isset($_GET["date"])){
            $date = isset($_GET["date"]) ? $_GET["date"] : date("Y-m-d");


            $stmt = $pdo->prepare("SELECT timeline.time_plan, task.task_name
                                   FROM timeline
                                   LEFT JOIN task ON timeline.task_id = task.task_id
                                   WHERE timeline.plan_date = :date AND task.userId = :user_id
                                   ORDER BY timeline.time_plan ASC");


            $stmt->bindParam(":date", $date, PDO::PARAM_STR);
            $stmt->bindParam(":user_id", $user_id, PDO::PARAM_STR);
            $stmt->execute();

            $tasks = $stmt->fetchAll(PDO::FETCH_ASSOC);

            echo json_encode($tasks);
            exit;
        }


        if (isset($_GET['unscheduled']) && $_GET['unscheduled'] === "true") {
            $stmt = $pdo->prepare("SELECT task_id, task_name FROM task WHERE complete_status = 'doing' AND userId = :user_id ORDER BY created_at ASC");

            $stmt->bindParam(":user_id", $user_id, PDO::PARAM_STR);
            $stmt->execute();

            $tasks = $stmt->fetchAll(PDO::FETCH_ASSOC);
            echo json_encode($tasks);
            exit;
        }

    } catch (Exception $e) {
        echo json_encode(["error" => "Error fetching tasks: " . $e->getMessage()]);
    }
```

Figure 4.2.1.2.3 Fetch task to display in timeline

Aspect	Description	Implementation
Display tasks of the selected	Users can view their scheduled tasks in the form of timeline based on their	• JavaScript: use '`loadTaskByDate()`' to

date in timeline	scheduled date. The description for each task will only be shown when the dropdown arrow is trigger.	fetch task name, scheduled date and its description to display on screen. The '`hasTask`' flag is set to determine whether the message 'No task scheduled…' should be displayed when the timeline is empty. `querySelectorAll()` is used to select all dropdown arrows and set an `eventListener` for each arrow. • PHP: use prepared statement (`SELECT query`) to retrieve task name and task description from database.
Date picker	The date picker is placed at the top for users to choose their desired date by directly selecting the date from the dropdown calendar or using arrow beside the date.	• JavaScript: use date picker library, '`Flatpickr`', to enable users to choose the date dynamically.
Design of timeline	All displayed tasks are aligned with the time, a representative dot and their descriptions.	• HTML: use '`tbody`' tag to place all timeline content in it. • JavaScript: use '`td`' tag for each scheduled task record, which first column contains the

		scheduled time, second column contains task name with task description hidden under dropdown arrow and a representative dot in between two columns. CSS-in-JS is used to style the gradient color of dots. '`startColor`' and '`endColor`' is determined and `Math.round` is used to calculate the ratio.

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    header("Content-Type: application/json");

    if (isset($_POST["task_id"], $_POST["task_name"], $_POST["time_plan"], $_POST["deadline"], $_POST["plan_date"])) {
        $task_id = trim($_POST["task_id"]);
        $task_name = trim($_POST["task_name"]);
        $task_time_plan = trim($_POST["time_plan"]);
        $task_deadline = trim($_POST["deadline"]);
        $task_plan_date = date("Y-m-d", strtotime($_POST["plan_date"] ?? date("Y-m-d")));
        $user_id = $_SESSION["user_id"] ?? "";
        $schedule_id = generateTimeId();
    } else {
        echo json_encode(["error" => "All fields are required."]);
        exit;
    }

    try {
        $stmt1 = $pdo->prepare("INSERT INTO timeline (schedule_id, task_id, time_plan, plan_date) VALUES (:schedule_id, :task_id, :time_plan, :plan_date)");

        $stmt2 = $pdo->prepare("UPDATE task
                                SET deadline = :deadline
                                WHERE task_id = :task_id AND userId = :user_id");


        $stmt1->bindParam(":schedule_id", $schedule_id, PDO::PARAM_STR);
        $stmt1->bindParam(":task_id", $task_id, PDO::PARAM_STR);
        $stmt1->bindParam(":time_plan", $task_time_plan);
        $stmt1->bindParam(":plan_date", $task_plan_date);

        $stmt2->bindParam(":deadline", $task_deadline);
        $stmt2->bindParam(":task_id", $task_id, PDO::PARAM_STR);
        $stmt2->bindParam(":user_id", $user_id, PDO::PARAM_STR);

        if ($stmt1->execute() && $stmt2->execute()) {
            insertActivityLog($pdo, $user_id, "A038", "Updated deadline for '$task_name'");
            insertActivityLog($pdo, $user_id, "A047", "Scheduled time for '$task_name'");
            insertActivityLog($pdo, $user_id, "A012", "Task '$task_name' added to timeline");
            echo json_encode(["message" => "Task updated successfully!"]);
            exit;
        } else {
            echo json_encode(["error" => "Failed to update task."]);
```

Figure 4.2.1.2.4 Save scheduled task

Aspect	Description	Implementation
Scheduling task	Schedule task by clicking the 'Schedule task' button and choosing task from the dropdown list, select the time and deadline to complete the process.	• JavaScript: use '`createOverlay()`' to create an overlay form when button '`Schedule task`' is clicked. '`removeOverlay()`' is used to remove the overlay, then display the scheduled task in the timeline. • CSS: input of type time and date are implemented with '`::webkit-calendar-picker-indicator`' to enable user selection.
Deadline	Set the minimum limit of date for dateline to current date	• JavaScript: use `getElementById` to target the deadline and set its attribute '`min`' to current date.
Save the scheduled task	Click the 'Save' button to send the input to the server.	• JavaScript: set all input into variable and post to the server by using new `FormDate()`. • PHP: use `generateTimeId()` to generate unique ID for each scheduled task. • MySQL: use prepared statement (`INSERT query`) to insert

		scheduled task details with the ID generated into '`timeline`' table. At the same time, use prepared statement (`UPDATE query`) to update the deadline in '`task`' table.

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    header("Content-Type: application/json");

    $result = $pdo->prepare("SELECT task_id, task_name, deadline, complete_status FROM task WHERE userId = :user_id ORDER BY deadline ASC");
    $result->bindParam(":user_id", $user_id, PDO::PARAM_STR);
    $result->execute();
    $tasks = $result->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($tasks);
    exit;

}
```

Figure 4.2.1.2.5 Task Status

Aspect	Description	Implementation
Task status category	There are 3 main groups to categorize the tasks: past date, doing and done. 'Past date' is the task that has passed the deadline and has not been done. 'Doing' is the task that is doing and 'Done' is the task that has been checked.	• HTML: use '`div`' tag to differentiate '`Past date`', '`Doing`' and '`Done`'. • JavaScript: Each container collapses initially until the dropdown arrow is triggered. `querySelectorAll()` is used to select all dropdown arrow and add `eventListener` for each arrow to perform the rotate-down function.
Display tasks and their deadlines	The tasks are displayed like a list in each group and separate by a horizontal line.	• JavaScript: fetch the task from database and set the fetched date format by using

based on groups		'toLocaleDate String("en-GB")'. update_count() is used to calculate the total number of tasks for each category and display alongside task name. • MySQL: use prepared statement (SELECT query) to select records from database.

4.2.1.3 Collaboration Features

- PHANG SHEA WEN-

Goalify - createWorkspace.php

```php
<?php
session_start();
require '../../../includes/db.php';
require '../../workspaces/workspace.php';
require '../../../includes/logging.php';

$userId = $_SESSION['user_id'];
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset($_POST['new-workspace-name'])) {
        $newWorkspaceName = $_POST['new-workspace-name'];
        $newWorkspaceId = generateId('workspace');
        if (insertNewWorkspace($pdo, $userId, $newWorkspaceId, $newWorkspaceName)) {
            $getWorkspaceName = getWorkspaceDetails($pdo, $newWorkspaceId, 'workspaceName');
            insertActivityLog($pdo, $_SESSION['user_id'], 'A014', "Workspace '$getWorkspaceName' is created !");
            echo json_encode(['success' => "Workspace '$getWorkspaceName' is created !"]);
        } else {
            echo json_encode(['fail' => "Workspace '$getWorkspaceName' is not created :("]);
        }
    }
}
exit();
?>
```

Figure 4.2.1.3.1.1 Create Workspace

Goalify - loadWorkspace.php

```php
<?php
session_start();
require '../../../includes/db.php';
require '../../workspaces/workspace.php';

$userId = $_SESSION['user_id'];

$availableWorkspaces = getWorkspaces($pdo, $userId);
echo json_encode(['availableWorkspaces' => $availableWorkspaces]);
?>
```

Figure 4.2.1.3.1.2 Load Available Workspaces

Goalify - joinWorkspace.php

```php
<?php
session_start();
require '../../../includes/db.php';
require '../../workspaces/workspace.php';
require '../../../includes/logging.php';

$userId = $_SESSION['user_id'];
error_log($userId);
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset($_POST['join-workspace-id'])) {
        $joinWorkspaceId = $_POST['join-workspace-id'];
        $result = insertNewWorkspaceMember($pdo, $userId, $joinWorkspaceId);
        $getWorkspaceName = getWorkspaceDetails($pdo, $joinWorkspaceId, 'workspaceName');
        if ($result === 'joined') {
            echo json_encode(['success' => "Previously joined Workspace '$getWorkspaceName' !"]);
        } else if ($result === true) {
            insertActivityLog($pdo, $_SESSION['user_id'], 'A019', "Successfully joined Workspace '$getWorkspaceName' !");
            echo json_encode(['success' => "Successfully joined Workspace '$getWorkspaceName' !"]);
        } else {
            echo json_encode(['fail' => "Failed to join Workspace '$getWorkspaceName' :("]);
        }
    }
}
exit();
?>
```

Figure 4.2.1.3.1.3 Join Workspace

Aspect	Description	Implementation
Workspace Creation and Management	Handles workspace creation, retrieval and management for users. It allows users to create new workspaces and fetch existing ones.	• PHP: create workspace using workspace name submitted via a create workspace form, select loaded workspaces, and join workspaces using workspace ID. • SQL: Uses `SELECT` and `INSERT INTO` queries. Utilizes prepared statements for security.
Workspace Invitations	Handles invitations to workspaces for collaboration.	• SQL: Uses prepare insert statement `INSERT INTO workspaceinvitation` to store invites. Invitation will be sent to the receiver's inbox.

Goalify - projectDescription.php

```php
if (isset($_SESSION['workspace_id']) && isset($_SESSION['project_id'])) {
    $userId = $_SESSION['user_id'];
    $workspaceId = $_SESSION['workspace_id'];
    $projectId = $_SESSION['project_id'];
    $owner = false;
    $descriptionFromDb = null;
    $stmt = $pdo -> prepare('SELECT 1
                            FROM `workspace`
                            WHERE `ownerId` = :userId AND `workspaceId` = :workspaceId;');
    $stmt -> execute([':userId' => $userId, ':workspaceId' => $workspaceId]);
    $result = $stmt -> fetchAll();
    if ($result) {
        $owner = true;
    }
    $projectName = getProjectDetails($pdo, $projectId, 'projectName');
    $projectCreatedDateTime = getProjectDetails($pdo, $projectId, 'projectCreatedDateTime');
    $projectDeadline = getProjectDetails($pdo, $projectId, 'projectDeadline');
    $projectStatus = getProjectDetails($pdo, $projectId, 'projectStatus');
    $projectDescription = getProjectDetails($pdo, $projectId, 'projectDescription');
    $projectDescriptionUpdate = getProjectDetails($pdo, $projectId, 'projectDescriptionUpdate');
}
```

Figure 4.2.1.3.3.1 Fetch Project Details

Goalify - projectDescriptionUpdate.php

```php
<?php
    ob_clean(); // Clean the output buffer
    header('Content-Type: application/json'); // response in json format
    session_start();

    require '../../../../includes/db.php';
    require '../../workspace.php';
    require '../../../../includes/logging.php';

    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        if (isset($_POST['description']) && isset($_POST['projectId'])) {
            $description = $_POST['description'];
            $projectId = $_POST['projectId'];
        } elseif (isset($_POST['project-name']) && isset($_POST['projectId'])) {
            $projectName = $_POST['project-name'];
            $projectId = $_POST['projectId'];
        }
        if (isset($_POST['project-deadline'])) {
            $projectId = $_SESSION['project_id'];
            $projectDdl = $_POST['project-deadline'];
        }

    }

    if (isset($description) && isset($projectId)) {
        // update workspace description
        updateProject($pdo, 'projectDescription', $description, $projectId);

        // get description update timestamp
        $descriptionUpdate = getProjectDetails($pdo, $projectId, 'projectDescriptionUpdate');

        if ($descriptionUpdate) {
            $projectName = getProjectDetails($pdo, $projectId, 'projectName');
            insertActivityLog($pdo, $_SESSION['user_id'], 'A023', "Project '$projectName''s description is updated!");
            echo json_encode(['descriptionUpdate' => $descriptionUpdate]);
        } else {
            // Error: No update found, return error message
            echo json_encode(['error' => 'Failed to retrieve project description update']);
        }
    } elseif (isset($projectName) && isset($projectId)) {
        // update workspace name
        updateProject($pdo, 'projectName', $projectName, $projectId);

        // get updated workspace name
        $newProjectName = getProjectDetails($pdo, $projectId, 'projectName');

        if ($newProjectName) {
            insertActivityLog($pdo, $_SESSION['user_id'], 'A045', "Project Name is updated to '$newProjectName'!");
            echo json_encode(['projectName' => $newProjectName]);
        } else {
            // Error: No update found, return error message
            echo json_encode(['error' => 'Failed to retrieve project name']);
        }
    } else if (isset($projectDdl)) {
        updateProject($pdo, 'projectDeadline', $projectDdl, $projectId);
        $newProjectDeadline = getProjectDetails($pdo, $projectId, 'projectDeadline');
        if ($newProjectDeadline) {
            $projectName = getProjectDetails($pdo, $projectId, 'projectName');
            insertActivityLog($pdo, $_SESSION['user_id'], 'A024', "Project deadline for '$projectName' is set to '$newProjectDeadline'");
            echo json_encode(['success' => "Project deadline is set to '$newProjectDeadline'", "projectDdl" => $newProjectDeadline]);
        } else {
            echo json_encode(['fail' => "Project deadline is not updated"]);
        }
    } else {
        // Error: Missing workspace details
        echo json_encode(['error' => 'Missing project details']);
    }
    exit();
?>
```

Figure 4.2.1.3.3.3.2 Update Project Details

Aspect	Description	Implementation
Fetching Projects	Retrieves a list of projects associated with a user.	• PHP: Uses `SELECT * FROM project WHERE userId = ?` to fetch project details securely.
Loading Projects	Fetches and displays project data. Loads ongoing projects (started), pending projects (created but not started), and all projects.	• PHP: Uses `SELECT * FROM project` to retrieve all projects. • JS: Uses `fetch('PHP/loadProjects.php')` to dynamically load project details. Generates project elements dynamically.
Project Handling	Manage project creation, deletion and updates.	• PHP: Uses `INSERT INTO project` for new projects. Deletes projects with `DELETE FROM project WHERE projectId = ?`. Updates details with `UPDATE project SET ... WHERE projectId = ?`.
Form Handling	When users click the 'Add Project' button, a pop-up form appears for entering the project name.	• JS: `projects.js` uses `createOverlay()` to generate a dynamic HTML form.
Workspace Description Retrieval	Fetches workspace descriptions stored in the database.	• PHP: Uses `SELECT workspaceDescription FROM workspace WHERE workspaceId = ?` to fetch data securely.
Updating Workspace Name and Description	Allow users to update workspace descriptions. Check whether the logged in user is the owner or member of the workspace, allowing only the owner to edit workspace name and description.	• PHP: Handles form submission via `$_POST`. Updates workspace descriptions using `UPDATE worskspace SET workspaceDescription = ? WHERE workspaceId = ?`.

Managing Workspace Members	Fetches members within a workspace.	• PHP: Uses `SELECT * FROM members WHERE workspaceId = ?`. • JS: List the workspace members showing their profile picture and name.

Goalify - loadProjectTask.php

```php
<?php
session_start();
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/includes/db.php';
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/user/workspaces/workspace.php';
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/includes/userFunction.php';

if (isset($_SESSION['project_id'])) {
    $userId = $_SESSION['user_id'];
    $workspaceId = $_SESSION['workspace_id'];
    $selectedProjectId = $_SESSION['project_id'];
}

$workspaceName = getWorkspaceDetails($pdo, $workspaceId, 'workspaceName');
$projectName = getProjectDetails($pdo, $selectedProjectId, 'projectName');
$projectDeadline = getProjectDetails($pdo, $selectedProjectId, 'projectDeadline');
$owner = getOwner($pdo, $workspaceId);
$allMembers = getMembers($pdo, $workspaceId);

// array_udiff -> does not reset indexing (array keys), array_values -> help resets the indexing after removing certain elements from the array
$members = array_values(array_udiff($allMembers, $owner, function ($a, $b) {
    return $a['userId'] <=> $b['userId']; // compare by 'userId'
}));

$ownerMember = [...$owner, ...$members];

// select all project tasks
$allProjectTasks = getProjectTasks($pdo, $selectedProjectId);
$i = 0;
foreach ($allProjectTasks as $projectTask) {
    // select all project sub tasks
    // append new key 'sub-tasks', with array of sub-tasks as its value
    $allProjectSubTasks = getProjectSubTasks($pdo, $projectTask['projectTaskId']);
    $j = 0;
    foreach ($allProjectSubTasks as $projectSubTask) {
        if ($projectSubTask['assignedMemberId'] !== null) {
            $assignedMemberName = getUserName($pdo, $projectSubTask['assignedMemberId']);
            $allProjectSubTasks[$j]['assignedMemberName'] = $assignedMemberName;
        } else {
            $allProjectSubTasks[$j]['assignedMemberName'] = null;
        }
        $j++;
    }
    $allProjectTasks[$i]['sub-tasks']= $allProjectSubTasks;
    $i++;
}

echo json_encode(['selectedProjectId' => $selectedProjectId, 'projectName' => $projectName, 'allProjectTasks' => $allProjectTasks, 'ownerMembers' => $ownerMember, 'projectDeadline' => $projectDeadline])
?>
```

Table 4.2.1.3.5 Load Project Tasks

Goalify - taskUpdate.php

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Set Time Zone
    if (isset($_POST['time-zone'])) {
        $timezone = $_POST['time-zone'];
        // $SESSION['time-zone'] = $timezone;
        date_default_timezone_set($timezone);
    } else {
        error_log("No time zone is received");
    }

    // Add New Project Task
    if (isset($_POST['new-project-task-name']) && isset($_POST['project-id'])) {
        addProjectTask();
    }
    // Modify Project Task Name
    if (isset($_POST['project-task-id']) && isset($_POST['project-task-name'])) {
        updateProjectTaskName();
    }
    // Delete Project Task
    if (isset($_POST['delete-project-task-id'])) {
        removeProjectTask();
    }

    // Add New Project Sub-Task
    if (isset($_POST['project-task-id']) && isset($_POST['sub-task-name'])) {
        addProjectSubTask();
    }
    // Modify Project Sub-Task Name
    if (isset($_POST['project-sub-task-name']) && isset($_POST['project-sub-task-id'])) {
        error_log($_POST['project-sub-task-name']);
        updateProjectSubTaskName();
    }
    // Delete Project Sub-Task
    if (isset($_POST['delete-project-sub-task-id'])) {
        removeProjectSubTask();
    }

    // Update Project Sub Task Assigned Member
    if (isset($_POST['project-sub-task-id']) && isset($_POST['assigned-member'])) {
        updateProjectSubTaskMember();
```

```
    }

    // Update Project Sub Task Priority
    if (isset($_POST['project-sub-task-id']) && isset($_POST['priority'])) {
        updateProjectSubTaskPriority();
    }

    // Update Project Sub Task Estimation
    if (isset($_POST['project-sub-task-id']) && isset($_POST['sub-task-estimation'])) {
        updateProjectSubTaskEstimate();
    }

    // Update Project Sub Task Assigned Date
    if (isset($_POST['project-sub-task-id']) && isset($_POST['assigned-date'])) {
        updateProjectSubTaskAssignedDate();
    }

    // Update Project Sub Task Due Date
    if (isset($_POST['project-sub-task-id']) && isset($_POST['due-date'])) {
        updateProjectSubTaskDueDate();
    }

    // Update Project Sub Task Status
    if (isset($_POST['project-sub-task-id']) && isset($_POST['status'])) {
        updateProjectSubTaskStatus();
    }

    // Get Project Deadline
    if (isset($_POST['projectDetails'])) {
        if ($_POST['projectDetails'] == 'projectDeadline') {
            $projectId = $_SESSION['project_id'];
            $deadline = getProjectDetails($pdo, $projectId, 'projectDeadline');
            if ($deadline) {
                echo json_encode(['success' => 'Project deadline is fetched', 'projectDeadline' => $deadline]);
            } else {
                echo json_encode(['success' => 'Failed to fetch project deadline']);
            }
        }
    }
}
```

Table 4.2.1.3.6 Update Project Sub-Tasks

Aspect	Description	Implementation
Project Description Retrieval	Fetches project descriptions stored in the database.	• PHP: Uses `SELECT projectDescription FROM project WHERE projectId = ?` to fetch data securely.
Updating Project Descriptions	Allow users to update project descriptions.	• PHP: Handles form submission via `$_POST`. Updates project descriptions using `UPDATE project SET projectDescription = ? WHERE projectId = ?`.
Fetching Projects Task and Project Sub-Task	Retrieves a list of projects with its sub-tasks associated with the selected project.	• PHP: Uses `SELECT * FROM projectTask WHERE projectId = ?` to fetch all project tasks securely and uses `SELECT * FROM projectSubTask WHERE projectId = ?` to fetch project details

Loading Project Tasks and Sub-Tasks	Fetches and displays project tasks and project sub-tasks in tabular form.	• JS: Uses `fetch('PHP/loadProjectTask.php')` to dynamically load project tasks and sub-tasks details. Dynamically creating project task tables with sub-task as rows.
Project Task and Sub-Task Handling	Manages project tasks and sub-task creation, deletion and updates. Each sub-task consist of details such as name, assigned member, priority, assigned date, due date, and progress.	• PHP: Uses `INSERT` statements for new project tasks and sub tasks via a form using JS. Deletes projects using `DELETE` statements, and updates details using `UPDATE` statements. • JS: uses `createOverlay()` to create dynamic HTML 'add new project task' form and 'add new project sub-task' form.
Form Handling	When users click the New Project Task' button or 'New Project Sub-Task' button, a pop-up form appears for entering the project name.	• JS: `projects.js` uses `createOverlay()` to generate a dynamic HTML form.

Goalify - sharedFiles.php

```php
<?php
session_start();
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/includes/db.php';
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/includes/session.php';
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/user/workspaces/workspace.php';

if (!isset($_SESSION['user_id'])) {
    $path = '/Goalify/Pages/user/login & register/login.php';
    header("location: $path");
    exit();
}

sessionTimeOut('/Goalify/Pages/user/login & register/login.php');
if (isset($_SESSION['project_id'])) {
    $userId = $_SESSION['user_id'];
    $workspaceId = $_SESSION['workspace_id'];
    $selectedProjectId = $_SESSION['project_id'];
}

$workspaceName = getWorkspaceDetails($pdo, $workspaceId, 'workspaceName');
$projectName = getProjectDetails($pdo, $selectedProjectId, 'projectName');
?>
```

Figure 4.2.1.3.4.1 Fetch Shared Files

Goalify - fileHandler.php

```php
function uploadFile() {
    global $pdo;
    $projectId = $_SESSION['project_id'];
    $fileName = $_FILES['new-file']['name'];
    $fileType = $_FILES['new-file']['type'];
    $fileSize = $_FILES['new-file']['size'];
    $userId = $_SESSION['user_id'];
    $fileData = file_get_contents($_FILES['new-file']['tmp_name']); // returns raw binary data
    if ($fileSize > 0) {
        $projectName = getProjectDetails($pdo, $projectId, 'projectName');
        if(insertNewProjectFile($pdo, $userId, $projectId, $fileName, $fileType, $fileSize, $fileData)) {
            insertActivityLog($pdo, $_SESSION['user_id'], 'A035', "File {$fileName} is uploaded to Project '$projectName'!");
            echo json_encode(['success' => "File {$fileName} is uploaded"]);
        } else {
            echo json_encode(['fail' => "File {$fileName} is not uploaded"]);
        }
    } else {
        echo json_encode(['fail' => "File {$fileName} is not uploaded"]);
    }
}

function deleteFile() {
    global $pdo;
    $projectId = $_SESSION['project_id'];
    $fileId = $_POST['deleteFileId'];
    $fileName = getProjectFileDetails($pdo, $fileId, 'fileName');
    if (deleteProjectFile($pdo, $fileId, $projectId)) {
        $projectName = getProjectDetails($pdo, $projectId, 'projectName');
        insertActivityLog($pdo, $_SESSION['user_id'], 'A037', "File {$fileName} is deleted from Project '$projectName'!");
        echo json_encode(['success' => "File '{$fileName}' is deleted"]);
    } else {
        echo json_encode(['fail' => "File '{$fileName}' is not deleted"]);
    }
}
```

Figure 4.2.1.3.4.2 Upload and Delete Files

Goalify - downloadFile.php

```php
function downloadFile() {
    global $pdo;
    $projectId = $_SESSION['project_id'];
    $fileId = $_GET['downloadFileId'];
    $fileName = getProjectFileDetails($pdo, $fileId, 'fileName');
    if ($fileName) {
        $fileType = getProjectFileDetails($pdo, $fileId, 'fileType');
        $fileData = getProjectFileDetails($pdo, $fileId, 'fileData');
        header("Content-Type: " . $fileType);
        header("Content-Disposition: attachment; filename=\"" . $fileName . "\"");
        header("Content-Length: " . strlen($fileData));
        echo $fileData;
    } else {
        echo "File '{$fileId}' is not found";
    }
}
```

Figure 4.2.1.3.4.3 Download Files

Aspect	Description	Implementation
Display Shared Files	Retrieves and displays files shared among workspace members.	• PHP: Uses `SELECT * FROM projectfiles WHERE workspaceId = ?`. Prevents unauthorized access with prepared statements.
File Download and delete	Handles file downloads and deleted for shared files.	• PHP: allow user to download file using '`echo $fileData;`', the raw binary data of the file.

		• JS: Triggers downloads when event listener added captures click event from the download button; and trigger deleted when event listener added captures click event from the delete button.
File Upload Handling	Manages file uploads for workspace collaboration.	• PHP: insert name, file type, file size and raw binary data of file into the database using `INSERT` statement. • JS: Uses `createOverlay()` for an upload form. Send files to `PHP/fileHandler.php` using `fetch()`.
Loading Project-Related Files	Fetches project-related files for display and download.	• PHP: Uses `SELECT * FROM projectfiles WHERE projectId = ?`. Ensures secure retrieval. • JS: Fetches files via `fetch('PHP/loadProject Files.php')`. Dynamically updates the UI.

```php
<?php
session_start();
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/includes/db.php';
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/user/workspaces/workspace.php';
require $_SERVER['DOCUMENT_ROOT'].'/Goalify/Pages/includes/userFunction.php';


header('Content-Type: application/json');

if (isset($_SESSION['project_id'])) {
    $projectId = $_SESSION['project_id'];
    $allProjectTasks = getProjectTasks($pdo, $projectId);

    $allSubTasks = []; // Store all sub-tasks
    $taskColors = []; // Store colors for each task

    foreach ($allProjectTasks as $projectTask) {
        $taskId = $projectTask['projectTaskId'];

        // Generate or get a color for this task
        if (!isset($taskColors[$taskId])) {
            $taskColors[$taskId] = sprintf("#%06X", mt_rand(0, 0xFFFFFF)); // Random Hex Color
        }

        // Fetch sub-tasks for this task
        $subTasks = getProjectSubTasks($pdo, $taskId);
        foreach ($subTasks as $subTask) {
            if ($subTask['assignedMemberId'] !== null) {
                $subTask['assignedMemberName'] = getUserName($pdo, $subTask['assignedMemberId']);
            } else {
                $subTask['assignedMemberName'] = null;
            }

            // Assign the same color as the parent task
            $subTask['color'] = $taskColors[$taskId];

            $allSubTasks[] = $subTask; // Add sub-task to the list
        }
    }

    echo json_encode(['success' => 'All sub-tasks fetched', 'allSubTasks' => $allSubTasks]);
} else {
    echo json_encode(['fail' => 'Sub-tasks not fetched, project ID undefined']);
}
exit();
?>
```

Table 4.2.1.3.2 Calendar and Event Management

Aspect	Description	Implementation
Event Calendar	Displays and manages events scheduled by users.	• PHP: Fetches events from `fetchEvents.php`. • JS: Uses AJAX to fetch and update event data dynamically. Integrates with the front-end calendar.
Fetching Scheduled Events	Retrieves project sub-task data from the database to display in the calendar.	• PHP: Uses `SELECT * FROM projectSubTask WHERE projectId = ?` to fetch project sub-tasks. Encodes data in JSON format. Uses prepared statements.

4.2.1.4 Productivity Analytics

- PAUREEN TAN NIE NIE-

Goalify - achievementHandler.php

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    if (!isset($_GET['userId'])) {
        echo json_encode(["error" => "User ID missing"]);
        exit();
    }
    $userId = $_GET['userId'];
    $month = isset($_GET["month"]) ? (int)$_GET["month"] : date("m");
    $year = isset($_GET["year"]) ? (int)$_GET["year"] : date("Y");
    $response = [];

    // ---------------------------------- Fetch Focus Time Data (Record Section) ----------------------------------
    $sqlFocus =
        "SELECT t.category,
            COUNT(DISTINCT f.taskId) AS totalTasks,
            SUM(f.duration) AS totalTimeSpent,
            SUM(f.duration) / COUNT(DISTINCT f.taskId) AS avgTimeSpent
        FROM focusrecord AS f
        JOIN task t ON f.taskId = t.task_id
        WHERE MONTH(f.timeTrackingDate) = :month
        AND YEAR(f.timeTrackingDate) = :year
        AND f.userId = :userId
        GROUP BY t.category";

    $stmtFocus = $pdo->prepare($sqlFocus);
    $stmtFocus->bindParam(':month', $month, PDO::PARAM_INT);
    $stmtFocus->bindParam(':year', $year, PDO::PARAM_INT);
    $stmtFocus->bindParam(':userId', $userId, PDO::PARAM_STR);
    $stmtFocus->execute();
    $response["focusData"] = $stmtFocus->fetchAll(PDO::FETCH_ASSOC);

    // ---------------------------------- Fetch Completed Task Count (Chart Section) ----------------------------------
    $sqlCompleted =
        "SELECT t.category, COUNT(*) AS totalCompletedTasks
        FROM task t
        WHERE t.complete_status = 'done'
        AND MONTH(t.completed_date) = :month
        AND YEAR(t.completed_date) = :year
        AND t.userId = :userId
        GROUP BY t.category
        ORDER BY totalCompletedTasks DESC";

    $stmtCompleted = $pdo->prepare($sqlCompleted);
    $stmtCompleted->bindParam(':month', $month, PDO::PARAM_INT);
    $stmtCompleted->bindParam(':year', $year, PDO::PARAM_INT);
    $stmtCompleted->bindParam(':userId', $userId, PDO::PARAM_STR);
    $stmtCompleted->execute();
    $response["completedTasks"] = $stmtCompleted->fetchAll(PDO::FETCH_ASSOC);

    echo json_encode($response);
}
```

Figure 4.2.1.4.1 Fetch Focus Time Data and Completed Task Count

Goalify - achievementHandler.php

```php
// ---------------------------------- Fetch Quote ----------------------------------
if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["quote"])) {
    $quoteApiUrl = "https://zenquotes.io/api/random";
    $quoteData = @file_get_contents($quoteApiUrl);
    $decodedData = json_decode($quoteData, true);
    if (json_last_error() === JSON_ERROR_NONE && is_array($decodedData) && isset($decodedData[0])) {
        // Remove the 'h' field from the API response
        unset($decodedData[0]['h']);
        header("Content-Type: application/json");
        echo json_encode($decodedData[0]); // Send only the first quote without 'h'
    } else {
        error_log("Invalid API response: " . $quoteData);
        echo json_encode(["error" => "Invalid API response"]);
    }
    exit();
}
```

Figure 4.2.1.4.2 Fetch Quote

Aspect	Description	Implementation
User Authentication	Ensures only logged-in users can access the achievements page.	• PHP: Checks if `$_SESSION["user_id"]` exists, otherwise redirects to login. Uses `sessionTime Out()` to log out inactive users.
Retrieve Motivational Quote	Fetches a daily motivational quote from an external API. If the fetch fails, a default message is shown.	• PHP: Calls https://zenquotes.io/api/random , removes unnecessary fields and returns JSON. If the API fails, logs an error and returns a default response. • JS: Uses `fetch()` to retrieve data from `achievementHandler.php?quote=true`, parses JSON response and updates `.quote` element in the DOM. If data is missing or an error occurs, a fallback message is displayed.
Retrieve Focus Records	Fetches time spent on different tasks for the selected month/year.	• PHP: Queries `focusrecord` and `task` tables, calculates total tasks, time spent and average time per task. Data is grouped by task category.
Retrieve Completed Tasks	Fetches the number of tasks marked as "done" per category.	• PHP: Queries `task` table to count completed tasks filtered by `month`, `year` and `user_id`.
Combine Achievements Data	Merges focus records and completed tasks into a single dataset.	• PHP: Stores results from both queries in an associative array grouped by `category`. If a category exists in focus records

		but not in completed tasks (or vice versa), initialize missing values to zero.
Fetch Data	Dynamically updates the achievements data on the page.	• JS: Uses `fetch()` to retrieve achievements data from `achievementHandler.php`.
Visualize Data Using Pie Chart	Represents completed tasks in a graphical format.	• JS: Calculates task completion percentages, creates an SVG pie chart dynamically and updates tooltip information on hover.
Task Category Breakdown	Displays the count of completed tasks under different urgency levels.	• JS: Loops through predefined categories (`Urgent`, `Plan Ahead`, `Handle Fast`, `On Hold`) and updates the UI accordingly.

4.2.1.5 Time-Tracking

- LUM HAN XUN-

```php
// insert focus record
if (isset($_GET['insertFocusRecord'])) {
    $timeTrackingId = generateTimeTrackingId();
    $userId = $_SESSION["user_id"];
    $taskId = $_POST["taskId"];
    $startTime = $_POST["startTime"];
    $endTime = $_POST["endTime"];
    $duration = $_POST["duration"];
    $timeTrackingDate = $_POST["timeTrackingDate"];

    try {
        $stmt = $pdo->prepare(query: 'INSERT INTO `focusrecord`(`timeTrackingId`, `userId`, `taskId`, `startTime` , `endTime`, `duration`, `timeTrackingDate`) VALUES
                                    (:timeTrackingId, :userId, :taskId, :startTime, :endTime, :duration, :timeTrackingDate);');
        $stmt->execute(params: [':timeTrackingId' => $timeTrackingId, ':userId' => $userId, ':taskId' => $taskId, ':startTime' => $startTime, ':endTime' => $endTime,
                                ':duration' => $duration, ':timeTrackingDate' => $timeTrackingDate]);
    } catch (PDOException $pdoE) {
        error_log(message: "PDO error inserting new focus record: " . $pdoE->getMessage());
    }
}

// insert activity log
if (isset($_GET['insertActivityLog'])) {
    $userId = $_SESSION["user_id"];
    $actionId = $_POST["actionId"];
    $details = $_POST["details"];

    insertActivityLog(pdo: $pdo, userId: $userId, actionId: $actionId, details: $details);
}
```

Figure 4.2.1.5.1 Update focus record and activity log

```php
// generate time tracking id
1 reference
function generateTimeTrackingId(): string
{
    return "#TID" . strtoupper(string: substr(string: md5(string: uniqid()), offset: 0, length: 5));
}

// format the unit of duration to hr, min, sec
1 reference
function formatDuration($second): string
{
    $hr = floor(num: $second / 3600);
    $min = floor(num: ($second % 3600) / 60);
    $sec = $second % 60;

    $result = [];
    if ($hr > 0) {
        $result[] = "$hr hr";
    }
    if ($min > 0) {
        $result[] = "$min min";
    }
    if ($sec > 0) {
        $result[] = "$sec sec";
    }

    return implode(separator: " ", array: $result);
}
```

Figure 4.2.1.5.2 Generate time tracking ID and format duration

```php
// fetch task from database
if (isset($_GET['getTasks'])) {

    global $pdo;
    if (!$pdo) {
        echo json_encode(value: ["error" => "Database connection failed"]);
        exit;
    }

    $userId = $_SESSION["user_id"];

    $stmt = $pdo->prepare(query: 'SELECT userId, task_id, task_name, complete_status
                                  FROM task
                                  WHERE userId = :userId
                                  AND complete_status = "doing"');
    $stmt->execute(params: [':userId' => $userId]);
    $tasks = $stmt->fetchAll(PDO::FETCH_ASSOC);

    echo json_encode(value: $tasks);
    exit;
}

// get focus records from database
if (isset($_GET['getRecords'])) {

    global $pdo;
    if (!$pdo) {
        echo json_encode(value: ["error" => "Database connection failed"]);
        exit;
    }

    $userId = $_SESSION["user_id"];

    $stmt = $pdo->prepare(query: 'SELECT taskId, task_name, startTime, endTime, duration
                                  FROM focusrecord
                                  LEFT JOIN task
                                  ON task.task_id = focusrecord.taskId
                                  WHERE focusrecord.userId = :userId
                                  AND timeTrackingDate >= CURRENT_DATE
                                  AND timeTrackingDate < CURRENT_DATE + INTERVAL 1 DAY
                                  ORDER BY duration DESC');
    $stmt->execute(params: [':userId' => $userId]);
    $records = $stmt->fetchAll(PDO::FETCH_ASSOC);

    foreach ($records as &$record) {
        $record['startTime'] = date(format: "H:i:s", timestamp: strtotime(datetime: $record['startTime']));
        $record['endTime'] = date(format: "H:i:s", timestamp: strtotime(datetime: $record['endTime']));
        $record['duration'] = formatDuration(second: $record['duration']);
    }
    unset($record);

    echo json_encode(value: $records);
    exit;
}
```

Figure 4.2.1.5.3 Fetch tasks and focus records

Aspect	Description	Implementation
User Authentication	Ensures only logged-in users can access the time-tracking page.	• PHP: Checks if `$_SESSION["user_id"]` exists, otherwise redirects to login. Uses `sessionTimeOut()` to log out inactive users.
Generate Time Tracking ID	Creates a unique identifier for each focus session.	• PHP: Uses `md5(uniqid())` to generate a unique string and returns an ID in the format `#TIDXXXXX`.
Insert Focus Record	Saves user focus session details into the database.	• PHP: Uses `INSERT INTO focusrecord` to store `timeTrackingId`, `userId`, `taskId`, `startTime`, `endTime`, `duration` and `timeTrackingDate`.
Retrieve Task List	Fetches active tasks assigned to the logged-in user.	• PHP: Queries `task` table for tasks where `complete_status = "doing"`, then returns the data as JSON.
Fetch Focus Records	Retrieves the user's focus session history for today.	• PHP: Queries `focusrecord` for today's date and joins with task table to include `task` names. Orders result by `duration DESC`.
Format Duration for Display	Converts total seconds into readable format (hrs, mins, secs).	• PHP: Uses `floor()` to calculate hours, minutes and seconds, then formats the output as `"X hr Y min Z sec"`.
Display Task List in UI	Lists available tasks dynamically in the UI.	• JS: Uses `fetch('timeTrackingHandler.php?getTasks=1')` to retrieve

		tasks and update `<ul class="task">`.
Show Focus Records Table	Displays focus session details in a sortable table.	• JS: Uses `fetch('timeTrackingHandler.php?getRecords=true')` to get focus records, then formats `startTime`, `endTime` and `duration`.
Start, Pause and Resume Timer	Controls the countdown timer for focus sessions.	• JS: Uses `setInterval()` to decrease time, `clearInterval()` to pause and `Date.now()` to adjust remaining time upon resume.
End Focus Session	Saves session data to the database when a session is completed.	• JS & PHP: Sends task ID, start time, end time, duration to `timeTrackingHandler.php` using a `POST` request.

4.2.1.6 Goal Setting

- LIM CHEE XUAN-

Goalify - loadGoal.php

```php
function getShortTermGoals($pdo, $userId) {
    try {
        $stmt = $pdo -> prepare('SELECT * FROM `goal` WHERE `userId` = :userId AND `goalType` = :goalType AND `completionStatus` = :completionStatus;');
        $stmt -> execute([':userId' => $userId, ':goalType' => 'short-term', ':completionStatus' => 'incomplete']);
        $result = $stmt -> fetchAll();
        return $result;
    } catch (PDOException $pdoE) {
        error_log("PDO error getting all short-term goals: ".$pdoE -> getMessage());
        return null;
    }
}

function getLongTermGoals($pdo, $userId) {
    try {
        $stmt = $pdo -> prepare('SELECT * FROM `goal` WHERE `userId` = :userId AND `goalType` = :goalType AND `completionStatus` = :completionStatus;');
        $stmt -> execute([':userId' => $userId, ':goalType' => 'long-term', ':completionStatus' => 'incomplete']);
        $result = $stmt -> fetchAll();
        return $result;
    } catch (PDOException $pdoE) {
        error_log("PDO error getting all short-term goals: ".$pdoE -> getMessage());
        return null;
    }
}

function getCompletedGoals($pdo, $userId) {
    try {
        $stmt = $pdo -> prepare('SELECT * FROM `goal` WHERE `userId` = :userId AND `completionStatus` = :completionStatus;');
        $stmt -> execute([':userId' => $userId, ':completionStatus' => 'completed']);
        $result = $stmt -> fetchAll();
        return $result;
    } catch (PDOException $pdoE) {
        error_log("PDO error getting all short-term goals: ".$pdoE -> getMessage());
        return null;
    }
}
```

Figure 4.2.1.6.1.1 Fetch Goals

Figure 4.2.1.6.1.2 Update Goals

Aspect	Description	Implementation
User Authentication	Ensures only logged-in users can access the time-tracking page.	• PHP: Checks if `$_SESSION["user_id"]` exists, otherwise redirects to login. Uses `sessionTimeOut()` to log out inactive users.
Retrieve Short-Term Goals	Displays active short-term goals for the user.	• PHP &JS: Uses `fetch('loadGoal.php')` to retrieve `goalType = 'short-term'` and updates `.short-term-list`.
Retrieve Long-Term Goals	Displays active long-term goals for the user.	• PHP &JS: Uses `fetch('loadGoal.php')` to retrieve `goalType = 'long-term'` and updates `.long-term-list`.
Retrieve Completed Goals	Displays all completed goals.	• PHP &JS: Uses `fetch('loadGoal.php')` to retrieve `completionStatus =`

		'completed' and updates .completed-list.
Add Short-Term Goal	Allows users to create new short-term goals.	• JS: Listens for button click on #add-short-term-goal, opens a modal and sends a POST request to goalHandler.php.
Add Long-Term Goal	Allows users to create new long-term goals.	• JS: Listens for button click on #add-long-term-goal, opens a modal and sends a POST request to goalHandler.php.
Mark Goal as Completed	Updates goal status to "completed" when checked.	• JS& PHP: Listens for click event on the check button, sends a POST request to goalHandler.php and updates the UI dynamically.
Delete Goal	Removes a goal permanently.	• JS& PHP: Listens for click event on the delete button, sends a POST request to goalHandler.php and removes the goal from the UI.
Render Goals in UI	Dynamically updates the goal list sections.	• JS: Uses fetch('loadGoal.php') to retrieve goal data and insert elements into .short-term-list, .long-term-list and .completed-list.

4.2.1.7 Light and Dark Theme

- PHANG SHEA WEN-

```
document.addEventListener('DOMContentLoaded', () => {
    const themeMode = document.querySelector('.material-symbols-rounded.theme_mode');

    // Dark and Light Mode
    // local storage -> local browser storage, specific to the same domain, tied to browser profile
    function themeModeListener() {
        if (themeMode) {
            themeMode.addEventListener('click', () => {
                const updateTheme = document.body.getAttribute('data-theme') === 'dark' ? 'light' : 'dark';
                document.body.setAttribute('data-theme', updateTheme);
                const updateIcon = updateTheme === 'dark'? 'light_mode' : 'dark_mode';
                themeMode.textContent = updateIcon;
                if (updateTheme === 'dark') {
                    localStorage.setItem('theme', 'dark');
                } else {
                    localStorage.setItem('theme', 'light')
                }

            })
        }
    }

    function setTheme () {
        const storedTheme = localStorage.getItem('theme');
        if (storedTheme) {
            document.body.setAttribute('data-theme', storedTheme);
            const updateIcon = storedTheme === 'dark' ? 'light_mode' : 'dark_mode';
            themeMode.textContent = updateIcon;
        } else {
            document.body.setAttribute('data-theme', 'light');
            themeMode.textContent = 'dark_mode';
        }
    }

    // Initialize on Page Load
    setTheme();
    themeModeListener();

    // Listen to Screen Size Changes
    mediaQuery.addEventListener('change', (e) => { // e is event object
        setTheme();
    });
})
```

Figure 4.2.1.7.1 Toggle Light and Dark Theme

Aspect	Description	Implementation
Toggle light theme	When the button of the theme mode icon is toggled to light mode, the theme of the website will also change to light mode. Light mode is the default mode for the website.	• JavaScript: '`themeMode`' variable is created to select the theme mode icon and add a click `eventListener()` to it. '`themeModeListener()`' is then added to it to trigger '`data-theme`' attribute to be updated and use `localStorage.setItem()` to set the theme to light mode.

Toggle dark theme	When the button of the theme mode icon is toggled to dark mode, the theme of the website will change to dark mode.	• JavaScript: the click `eventListener()` is triggered and set the theme to dark mode if the current mode is in light mode. • CSS: `[dark-theme='dark']` is used to group and set the color of specific elements in dark theme color when theme mode icon is triggered. The CSS styles that out of the group are considered as default (light mode).

4.2.1.8 Notification

- LIM CHEE XUAN-

-LUM HAN XUN-

undefined - notificationHandler.php

```php
// fetch tasks that are going to due
    $stmtTask = $pdo->prepare('SELECT userId, task_name, complete_status, deadline, DATEDIFF(deadline, NOW()) AS daysLeft
                               FROM task
                               WHERE userId = :userId
                               AND complete_status = "doing"
                               AND DATEDIFF(deadline, NOW()) <= 3');
    $stmtTask->execute([':userId' => $userId]);
    $dueTasks = $stmtTask->fetchAll(PDO::FETCH_ASSOC);

    // fetch project sub task that are going to due
    $stmtProject = $pdo->prepare('SELECT w.workspaceName, p.projectName, pst.projectSubTaskName, pst.assignedMemberId,
                                  pst.projectSubTaskStatus, DATEDIFF(pst.projectSubTaskDueDate, NOW()) AS daysLeft
                                  FROM projectsubtask pst
                                  JOIN projecttask pt ON pt.projectTaskId = pst.projectTaskId
                                  JOIN project p ON pt.projectId = p.projectId
                                  JOIN workspace w ON p.workspaceId = w.workspaceId
                                  WHERE pst.assignedMemberId = :userId
                                  AND projectSubTaskStatus = "in progress"
                                  AND DATEDIFF(pst.projectSubTaskDueDate,NOW()) <= 3');
    $stmtProject->execute([':userId' => $userId]);
    $dueProjectSubTask = $stmtProject->fetchAll(PDO::FETCH_ASSOC);

    // fetch workspace invitation
    $stmtWorkspaceInvitation = $pdo->prepare('SELECT w.workspaceName, wi.invitationId, wi.senderId, wi.receivedId, wi.sendDateTime, wi.invitationStatus, u.name, u.profile_img
                                              FROM workspaceinvitation wi
                                              JOIN user u ON u.userId = wi.senderId
                                              JOIN workspace w ON w.workspaceId = wi.workspaceId
                                              WHERE wi.receivedId = :userId
                                              AND wi.invitationStatus IS NULL');
    $stmtWorkspaceInvitation->execute([':userId' => $userId]);
    $workspaceInvitation = $stmtWorkspaceInvitation->fetchAll(PDO::FETCH_ASSOC);

    $result = [
        "task" => $dueTasks ?: null,
        "project" => $dueProjectSubTask ?: null,
        "invitation" => $workspaceInvitation ?: null,
    ];

    echo json_encode($result);
    exit;
```

Figure 4.2.1.8.1 Fetch Data for Users

Aspect	Description	Implementation
Display Invitation Notifications	Shows notifications when a user is invited to a workspace or project.	• HTML: Uses `<div class="invitation">` to wrap each invitation. Includes inviter details, workspace/project name and response buttons (Block, Accept).
Display Task Warnings	Shows task due date reminders to users.	• HTML: Uses `<div class="task-warning">` to wrap the notification. Contains task due date, description and action button.
Fetch Task Due Notifications	Retrieves tasks that are due within three days for the logged-in user.	• PHP: Queries the `task` table for tasks where `complete_status = "doing"` and the deadline is within 3 days using `DATEDIFF(deadline, NOW()) <= 3`.

4.2.1.9 Help Centre

- PAUREEN TAN NIE NIE-

Goalify - contactSupport.php

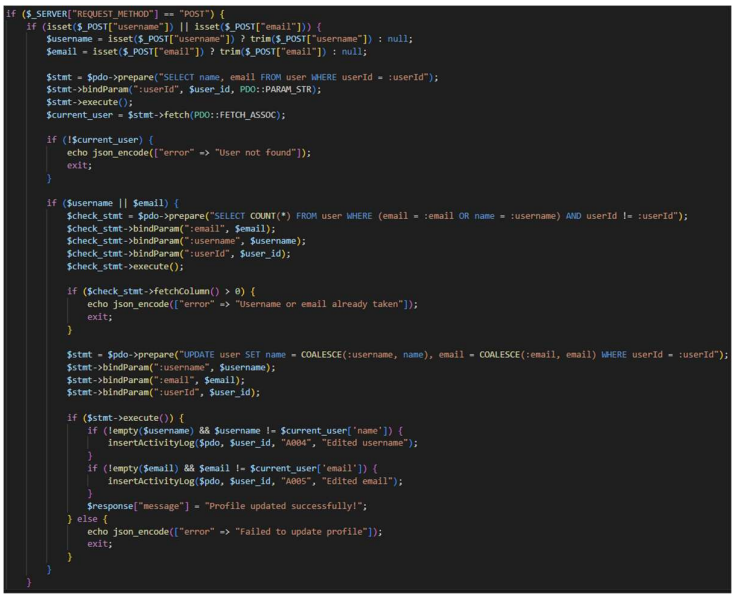
```
<script>
    document.addEventListener('DOMContentLoaded', () => {
        const contactButton = document.querySelector('.contact-button');
        contactButton.addEventListener('click', () => {
            if (!document.body.contains(document.querySelector('#overlay'))) {
                createOverlay(
                    'contact-support',
                    `
                    <p><strong>Contact Support</strong></p>
                    <p><span class="material-symbols-outlined">mail</span><a href="mailto:support@goalify.com">admin@goalify.com</a></p>
                    <p><span class="material-symbols-outlined">call</span><a href="tel:+1234567890"> +123 456 7890</a></p>
                    <div class="button">
                        <button type="button" id="close-contact">Close</button>
                    </div>
                    `
                );
                document.querySelector('#close-contact').addEventListener('click', () => {
                    removeOverlay('contact-support');
                });
            }
        });
    });
</script>
```

Figure 4.2.1.9.1 Contact Support Overlay

Aspect	Description	Implementation
Open Contact Support Overlay	Displays a pop-up overlay with support contact details when the "Contact" button is clicked.	• JS: Uses `createOverlay()` to generate an overlay containing email and phone contact details. The `removeOverlay()` function is used to close it.
Display FAQ Section	Lists common questions and answers regarding Goalify.	• HTML: Uses `<div class="faq-section ">` to organize questions under different topics such as about goalify, account & login, workspace & collaboration and technical issues & support.
Offering Additional Help	Encourages users to reach out if they need more assistance.	• HTML: Uses `<div class="furtherHelp">` section with a "Contact" button that opens the support overlay.

4.2.1.10 Profile Management

- NG YVONNE-



```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["username"]) || isset($_POST["email"])) {
        $username = isset($_POST["username"]) ? trim($_POST["username"]) : null;
        $email = isset($_POST["email"]) ? trim($_POST["email"]) : null;

        $stmt = $pdo->prepare("SELECT name, email FROM user WHERE userId = :userId");
        $stmt->bindParam(":userId", $user_id, PDO::PARAM_STR);
        $stmt->execute();
        $current_user = $stmt->fetch(PDO::FETCH_ASSOC);

        if (!$current_user) {
            echo json_encode(["error" => "User not found"]);
            exit;
        }

        if ($username || $email) {
            $check_stmt = $pdo->prepare("SELECT COUNT(*) FROM user WHERE (email = :email OR name = :username) AND userId != :userId");
            $check_stmt->bindParam(":email", $email);
            $check_stmt->bindParam(":username", $username);
            $check_stmt->bindParam(":userId", $user_id);
            $check_stmt->execute();

            if ($check_stmt->fetchColumn() > 0) {
                echo json_encode(["error" => "Username or email already taken"]);
                exit;
            }

            $stmt = $pdo->prepare("UPDATE user SET name = COALESCE(:username, name), email = COALESCE(:email, email) WHERE userId = :userId");
            $stmt->bindParam(":username", $username);
            $stmt->bindParam(":email", $email);
            $stmt->bindParam(":userId", $user_id);

            if ($stmt->execute()) {
                if (!empty($username) && $username != $current_user['name']) {
                    insertActivityLog($pdo, $user_id, "A004", "Edited username");
                }
                if (!empty($email) && $email != $current_user['email']) {
                    insertActivityLog($pdo, $user_id, "A005", "Edited email");
                }
                $response["message"] = "Profile updated successfully!";
            } else {
                echo json_encode(["error" => "Failed to update profile"]);
                exit;
            }
        }
    }
```

Figure 4.2.1.10.1 Change username and email

```php
if (!empty($_FILES["profile_pic"]["name"])) {
    $target_dir = "uploads/";

    if (!is_dir($target_dir)) {
        mkdir($target_dir, 0777, true);
    }

    $profile_img = $target_dir . basename($_FILES["profile_pic"]["name"]);

    if (move_uploaded_file($_FILES["profile_pic"]["tmp_name"], $profile_img)) {
        $stmt = $pdo->prepare("UPDATE user SET profile_img = :profile_img WHERE userId = :userId");
        $stmt->bindParam(":profile_img", $profile_img);
        $stmt->bindParam(":userId", $user_id);

        if ($stmt->execute()) {
            $response["profile_img"] = $profile_img;
            insertActivityLog($pdo, $user_id, "A003", "Updated profile image");
        } else {
            echo json_encode(["error" => "Failed to update profile picture"]);
            exit;
        }
    } else {
        echo json_encode(["error" => "File upload failed"]);
        exit;
    }
}
```

Figure 4.2.1.10.2 Update profile picture

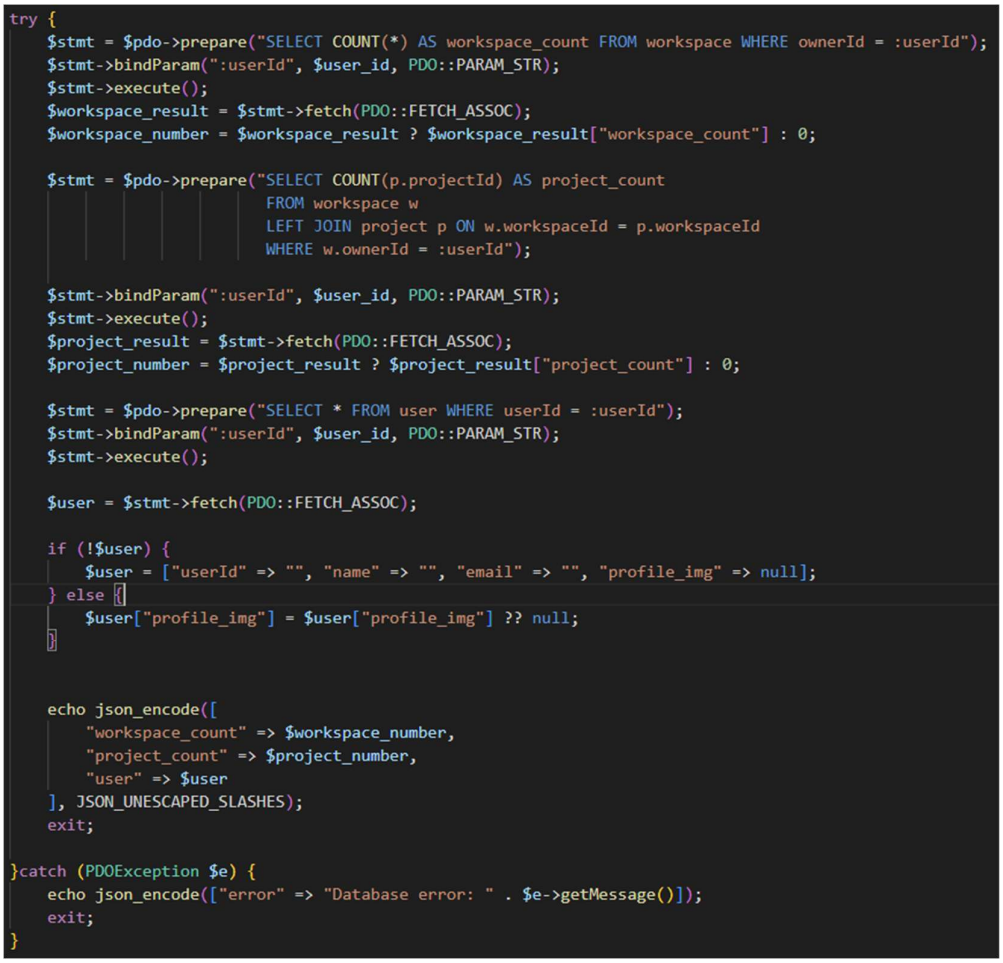
```php
if (isset($_POST["new_password"])) {
    $new_password = trim($_POST["new_password"]);
    $confirm_password = trim($_POST["confirm_password"]);

    if ($new_password !== $confirm_password) {
        echo json_encode(["error" => "Passwords do not match"]);
        exit;
    }

    $hashed_password = password_hash($new_password, PASSWORD_BCRYPT);

    $stmt = $pdo->prepare("UPDATE user SET password = :new_password WHERE userId = :userId");
    $stmt->bindParam(":new_password", $hashed_password);
    $stmt->bindParam(":userId", $user_id);

    if ($stmt->execute()) {
        $response["success"] = true;
        $response["message"] = "Password updated successfully!";
        insertActivityLog($pdo, $user_id, "A006", "Changed password");
    } else {
        echo json_encode(["error" => "Failed to update password"]);
        exit;
    }
}
echo json_encode($response);
exit;
```

Figure 4.2.1.10.3 Change password

Aspect	Description	Implementation
Change profile picture	Users can change their profile pictures by uploading images. The accepted image types are jpg, jpeg, png and svg+xml. Once the profile picture is changed, the profile icon on the top right of the page will also be changed.	• HTML: a main '`div`' container holds an img element, an '`edit icon`' and an input element. Input type is set to file to allow users to upload images. • JavaScript: the edit icon has added a '`click`' `eventListener` to determine the file that has uploaded. • PHP: the images will be store in '`uploads`' directory. '`mkdir($target_dir, 0777, true)`' is used to create a directory with permissions of full read, write and execute access if '`uploads`' does not exists.
Change username and email	Users can change their username and email after logging in. Username and email cannot be empty or invalid.	• HTML: use input type text and email to allow users input. The input fields are set to required. • JavaScript: a predefined email pattern is used to validate the email entered. '`querySelector()`' and '`getElementById()`'

		are used to retrieve the input of users and prepare to be posted to the server using '`new FormData()`'. • CSS: When user focuses on the input filed of username, text-decoration property is triggered.
Change password	Users can change their password by clicking the link below the email input field. Fields for new password and confirm password is required to fill out.	• HTML: use '`a`' tag and set `href` to '`#`' to create a link on the same page for users to change their passwords. • JavaScript: `createOverlay()` is used to open an overlay form to change password. An if statement is used to ensure the 'new password' is the same with the 'confirm password'.
Save changes	'Save changes' button at the top right is clicked to save the changes into database.	• JavaScript: all modifications will be passed into a form and posted to the server. '`removeOverlay()`' is called once the changes are saved successfully.

		• MySQL: use prepared statement (`UPDATE query`) to update the data in '`user`' table. '`password_hash()`' is used to hash the password to '`BCRYPT`' before saving into database.



```php
try {
    $stmt = $pdo->prepare("SELECT COUNT(*) AS workspace_count FROM workspace WHERE ownerId = :userId");
    $stmt->bindParam(":userId", $user_id, PDO::PARAM_STR);
    $stmt->execute();
    $workspace_result = $stmt->fetch(PDO::FETCH_ASSOC);
    $workspace_number = $workspace_result ? $workspace_result["workspace_count"] : 0;

    $stmt = $pdo->prepare("SELECT COUNT(p.projectId) AS project_count
                           FROM workspace w
                           LEFT JOIN project p ON w.workspaceId = p.workspaceId
                           WHERE w.ownerId = :userId");

    $stmt->bindParam(":userId", $user_id, PDO::PARAM_STR);
    $stmt->execute();
    $project_result = $stmt->fetch(PDO::FETCH_ASSOC);
    $project_number = $project_result ? $project_result["project_count"] : 0;

    $stmt = $pdo->prepare("SELECT * FROM user WHERE userId = :userId");
    $stmt->bindParam(":userId", $user_id, PDO::PARAM_STR);
    $stmt->execute();

    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$user) {
        $user = ["userId" => "", "name" => "", "email" => "", "profile_img" => null];
    } else {
        $user["profile_img"] = $user["profile_img"] ?? null;
    }


    echo json_encode([
        "workspace_count" => $workspace_number,
        "project_count" => $project_number,
        "user" => $user
    ], JSON_UNESCAPED_SLASHES);
    exit;

}catch (PDOException $e) {
    echo json_encode(["error" => "Database error: " . $e->getMessage()]);
    exit;
}
```

Figure 4.2.1.10.4 Display profile details

Aspect	Description	Implementation
Display number of workspaces and projects	The current number of workspaces and projects the users enrolled in will be displayed below the username.	• JavaScript: use '`getElementById()`' to select the container of workspace and project. '`fetch()`' is used to request '`GET`' method from server to get data from database. • MySQL: use prepared statement (`SELECT query`) to retrieve the number of workspace and projects from database. Since the workspaces and projects is in different table, '`LEFT JOIN`' is used to combine two table and retrieve the data together.
Display all profile information on page load	Information of the profile will be displayed on page loaf if users do not log out.	• JavaScript: determines which value of the elements the data will display and fetch data from database. • MySQL: use prepared statement (`SELECT query`) to retrieve all data from database for a specific user.

```php
<?php
require_once 'logging.php';
include 'db.php';
session_start();
$userId = $_SESSION['user_id'];
insertActivityLog($pdo, $userId, 'A002', "Successfully logged out !");
session_unset();
session_destroy();
header("Location: /Goalify/Pages/user/login%20&%20register/login.php");
exit();
?>
```

Figure 4.2.1.10.5 Log Out

Aspect	Description	Implementation
Log out	Users will be logged out when the 'Log out' button is clicked.	• HTML: create a button element for log out function. • JavaScript: select the button by using '`getElementById()`' and add a '`click`' `eventListener`. When the button is clicked, the page will return to the homepage. '`window.location.href`' is used to link with the php file of log out to perform `session_destroy()`. • PHP: use `session_unset()` to clear all session variables, delete all session ID and data using '`session_destroy()`'.

4.2.2 Administrator Features

4.2.2.1 Administrator Login

- PAUREEN TAN NIE NIE-

Goalify - adminHandler.php

```php
function login($pdo) {
    $email = trim($_POST['login-email']);
    $userEmail = filter_var($email, FILTER_SANITIZE_EMAIL); // Remove illegal characters
    $userPwd = $_POST['login-pwd'];
    $adminEmail = $_ENV['ADMIN_EMAIL'] ?? ''; // fetch data from env
    $adminPassword = $_ENV['ADMIN_PASSWORD'] ?? ''; // fetch data from env
    if ($userEmail === $adminEmail && $userPwd === $adminPassword) {
        createUserSession('admin', 'admin', $pdo, '/Goalify/Pages/admin/dashboard/dashboard.php', false);
        return "Login Successfully";
    } elseif ($userEmail !== $adminEmail) {
        $loginErrorMsg = 'Email not found'; // echo in js, use js to insert error message
        return $loginErrorMsg;
    } else {
        $loginErrorMsg = 'Invalid password';
        return $loginErrorMsg;
    }
}
```

Figure 4.2.2.1 Administrator Login

Aspect	Description	Implementation
User Authentication	Verifies user credentials before granting access.	• PHP: `login.php` processes data using `$_POST`, checks credentials against the env file using `$_ENV[]` and verifies email and password.
Session Management	Maintains user login state across pages.	• PHP: Uses `session_start()` to create a session and store user data upon successful login.
Error Handling	Provides feedback for invalid login attempts.	• PHP: Returns error messages if the email or password is incorrect.

4.2.2.2 Administrator Dashboard

- LUM HAN XUN-

Goalify - dashboardHandler.php

```php
if (isset($_GET['getMetrics'])) {
    $response = [];
    // total users
    $result = $pdo->query("SELECT COUNT(*) AS total FROM user");
    $response['totalUsers'] = $result->fetchColumn();
    // sign-ups (monthly)
    $signupQuery = $pdo->query("SELECT MONTHNAME(joinedDateTime) AS month, COUNT(*) AS count FROM user GROUP BY MONTH(joinedDateTime) ORDER BY joinedDateTime ASC");
    $months = [];
    $signups = [];
    while ($row = $signupQuery->fetch(PDO::FETCH_ASSOC)) {
        $months[] = $row['month'];
        $signups[] = (int) $row['count'];
    }
    $response['months'] = $months;
    $response['signups'] = $signups;
    // active users (weekly)
    $activeUsersQuery = $pdo->query("SELECT
        DATE_FORMAT(DATE_SUB(timestamp, INTERVAL WEEKDAY(timestamp) DAY), '%m-%d') AS week_start,
        COUNT(DISTINCT userId) AS count
        FROM activitylog
        WHERE actionId = 'A001'
        GROUP BY YEAR(timestamp), WEEK(timestamp)
        ORDER BY MIN(timestamp) ASC
    ");
    $weeks = [];
    $activeUsers = [];
    while ($row = $activeUsersQuery->fetch(PDO::FETCH_ASSOC)) {
        $weeks[] = $row['week_start'] ?? 'No Data';
        $activeUsers[] = (int) ($row['count'] ?? 0);
    }
    $response['weeks'] = $weeks;
    $response['activeUsers'] = $activeUsers;
    echo json_encode($response);
}
```

Figure 4.2.2.2.1 Fetch Summary – Total Users, Sign-Ups, Active Users

- PAUREEN TAN NIE NIE-

Goalify - dashboardHandler.php

```php
if (isset($_GET['workspace'])) {
    $filter = $_GET['workspace'];

    if ($filter == 'week') {
        $stmt = $pdo->prepare('SELECT DATE(createdDateTime) AS date,
        COUNT(*) AS totalWorkspace
        FROM workspace
        WHERE YEARWEEK(createdDateTime, 1) = YEARWEEK(CURDATE(), 1)
        GROUP BY DATE(createdDateTime)
        ORDER BY date ASC');

    } elseif ($filter == 'month') {
        $stmt = $pdo->prepare("SELECT DATE(DATE_SUB(createdDateTime, INTERVAL WEEKDAY(createdDateTime) DAY)) AS week,
        COUNT(*) AS totalWorkspace
        FROM workspace
        WHERE createdDateTime >= DATE_FORMAT(CURDATE(), '%Y-%m-01')
        GROUP BY week
        ORDER BY week ASC");

    } elseif ($filter == 'year') {
        $stmt = $pdo->prepare("SELECT DATE_FORMAT(createdDateTime, '%Y-%m') AS month,
        COUNT(*) AS totalWorkspace
        FROM workspace
        WHERE createdDateTime >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
        GROUP BY DATE_FORMAT(createdDateTime, '%M %Y')
        ORDER BY STR_TO_DATE(month, '%M %Y') ASC");
    }

    $stmt->execute();
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

    if(!$result) $result = [];

    echo json_encode($result);
    exit();
}
```

Figure 4.2.2.2.2 Fetch Active Workspace Statistics (Week/Month/Year)

Goalify - dashboardHandler.php

```php
if (isset($_GET['project'])) {
    $filter = $_GET['project'];

    if ($filter == 'week') {
        $stmt = $pdo->prepare('SELECT DATE(projectCreatedDateTime) AS date,
        COUNT(*) AS totalProject
        FROM project
        WHERE YEARWEEK(projectCreatedDateTime, 1) = YEARWEEK(CURDATE(), 1)
        GROUP BY DATE(projectCreatedDateTime)
        ORDER BY date ASC');

    } elseif ($filter == 'month') {
        $stmt = $pdo->prepare("SELECT DATE(DATE_SUB(projectCreatedDateTime, INTERVAL WEEKDAY(projectCreatedDateTime) DAY)) AS week,
        COUNT(*) AS totalProject
        FROM project
        WHERE projectCreatedDateTime >= DATE_FORMAT(CURDATE(), '%Y-%m-01')
        GROUP BY week
        ORDER BY week ASC");

    } elseif ($filter == 'year') {
        $stmt = $pdo->prepare("SELECT DATE_FORMAT(projectCreatedDateTime, '%Y-%m') AS month,
        COUNT(*) AS totalProject
        FROM project
        WHERE projectCreatedDateTime >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
        GROUP BY DATE_FORMAT(projectCreatedDateTime, '%M %Y')
        ORDER BY STR_TO_DATE(month, '%M %Y') ASC");
    }

    $stmt->execute();
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

    if(!$result) $result = [];

    echo json_encode($result);
    exit();
}

// for user metrics
if (isset($_GET['getUser'])) {
    try {
        $stmt = $pdo->prepare('SELECT userId AS id, name, profile_img FROM user');
        $stmt->execute();
        $users = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($users);
    } catch (PDOException $e) {
        echo json_encode([]);
    }
    exit;
}
```

Figure 4.2.2.2.3 Fetch Projects Completed Statistics (Week/Month/Year)

Aspect	Description	Implementation
Workspace and Project Filters	Allow admins to filter workspaces and projects by time range (week, month, year).	• JS: `dashboard.js` listens for changes in filter selection and fetches relevant data using `fetch()`.
Fetching Workspace Data	Retrieves workspace count based on selected time range.	• PHP: `dashboardHandler.php` processes the request and queries the database `using SELECT COUNT(*) FROM workspaces`. • JS: Uses `fetch()` to request data and updates the chart dynamically.

Fetching Project Data	Retrieves project count based on selected time range.	• PHP: `dashboardHandler.php` processes the request and queries the database using `SELECT COUNT(*) FROM projects`. • JS: Uses `fetch()` to request data and updates the chart dynamically.
Chart Display	Visualizes workspace, project and user activity data in bar and line charts.	• JS: `dashboard.js` uses `Chart.js` to create and update charts dynamically.
Fetching Total Users	Retrieves the total number of registered users.	• PHP: `dashboardHandler.php` queries `SELECT COUNT(*) AS total FROM users` and returns the count as JSON. • JS: Uses `fetch()` to update the UI dynamically.
Fetching Sign-Up and Active User Metrics	Retrieves sign-up counts per month and active users per week.	• PHP: `dashboardHandler.php` processes database queries for sign-ups `(SELECT MONTHNAME(joinedDateTime) AS month, COUNT(*) AS count FROM user ...)` and active users `(SELECT DATE_FORMAT(DATE_SUB(timestamp, INTERVAL WEEKDAY(timestamp) DAY, '%m-%d'),COUNT(DISTINCT userId) FROM activitylog WHERE actionId='A001' ...)`. • JS: Uses `fetch()` to populate bar and line charts dynamically.

4.2.2.3 User Management

- PHANG SHEA WEN-

Goalify - userHandler.php

```php
} else if (isset($_POST['create-user-email']) && isset($_POST['create-user-password'])) {
    $registerName = $_POST['create-user-name'];
    $registerEmail = $_POST['create-user-email'];
    $registerPwd = $_POST['create-user-password'];
    $msg = register($pdo, $registerName, $registerEmail, $registerPwd);
    if ($msg === "Registered Successfully") {
        echo json_encode(['register' => true, 'msg' => $msg]);
        exit();
    } else {
        echo json_encode(['register' => false, 'msg' => $msg]);
        exit();
    }
```

Figure 4.2.2.4.1 Create User

Goalify - userHandler.php

```php
} else if (isset($_POST['user-id']) && isset($_POST['modify-user-name']) && isset($_POST['modify-user-email'])) {
    $userId = $_POST['user-id'];
    $oldUserEmail = getUserEmail($pdo, $userId);
    error_log($oldUserEmail);
    $newUsrName = $_POST['modify-user-name'];
    $newUsrEmail = $_POST['modify-user-email'];
    if ($newUsrName !== '' && $newUsrEmail !== '') {
        if (modifyUserName($pdo, $userId, $newUsrName)) {
            $update = true;
            if ($newUsrEmail !== $oldUserEmail) {
                $msg = modifyUserEmail($pdo, $userId, $newUsrEmail);
                if ($msg === "Email Updated Successfully") {
                    $update = true;
                    if (isset($_POST['modify-user-password']) && $_POST['modify-user-password'] !== '') {
                        $newPwd =  $_POST['modify-user-password'];
                        error_log($newPwd);
                        $msg = modifyUserPwd($pdo, $userId, $newPwd);
                        if ($msg === "Password Updated Successfully") {
                            $update = true;
                            $msg = "User is updated!";
                        } else {
                            $update = false;
                        }
                    } else if ($_POST['modify-user-password'] == '') {
                        $update = true;
                        $msg = "User is updated!";
                    }
                } else {
                    $update = false;
                }
            } else {
                if (isset($_POST['modify-user-password']) && $_POST['modify-user-password'] !== '') {
                    $newPwd =  $_POST['modify-user-password'];
                    $msg = modifyUserPwd($pdo, $userId, $newPwd);
                    if ($msg === "Password Updated Successfully") {
                        $update = true;
                        $msg = "User is updated!";
                    } else {
                        $update = false;
                    }
                } else if ($_POST['modify-user-password'] == '') {
                    $update = true;
                    $msg = "User is updated!";
                }
            }
            echo json_encode(['update' => $update, 'msg' => $msg, 'userName' => getUserName($pdo, $userId), 'userEmail' => getUserEmail($pdo, $userId)]);
        }
    } else {
        echo json_encode(['update' => false, 'msg' => 'Name and Email cannot be left blank']);
    }
    exit();
}
```

Figure 4.2.2.4.2 Modify User

Aspect	Description	Implementation
Loading Users	Retrieves a list of users from the database for display.	• PHP: `loadUsers.php` queries `SELECT * FROM users`, encodes

		the data in JSON and returns it to be displayed in the admin panel.
Creating a New User	Allow admins to add a new user to the system.	• PHP: `createUser.php` handles form submission, validates input, hashes passwords using `password_hash()` and inserts user data into the database using `INSERT INTO users`.
Modifying User Data	Update user details like name, email, or role.	• PHP: `modifyUser.php` processes form data and executes `UPDATE users SET ... WHERE userId = ?` to modify existing user records.
Deleting a User	Removes a user from the system.	• PHP: `deleteUser.php` processes a deletion request using `DELETE FROM users WHERE userId = ?`, ensuring secure deletion. • JS: `action.js` listens for clicks on the delete button, fetches `deleteUser.php` via `fetch()` and reloads the page upon success.

4.2.2.4 User Metrics

- PAUREEN TAN NIE NIE-

Goalify - userMetricsHandler.php

```php
// user details
if (isset($_GET['getUserDetails']) && isset($_GET['userId'])) {
    $userId = $_GET['userId'];
    try {
        // fetch user details
        $stmt = $pdo->prepare('SELECT * FROM user WHERE userId = ?');
        $stmt->execute([$userId]);
        $user = $stmt->fetch(PDO::FETCH_ASSOC);
        if (!$user) {
            echo json_encode(['error' => 'User not found']);
            exit;
        }
        // fetch last login
        $lastLoginStmt = $pdo->prepare('SELECT timestamp FROM activitylog WHERE userId = ? AND actionId = "A001" ORDER BY timestamp DESC LIMIT 1');
        $lastLoginStmt->execute([$userId]);
        $lastLogin = $lastLoginStmt->fetchColumn();
        // fetch tasks completed
        $tasksCompletedStmt = $pdo->prepare('SELECT COUNT(*) FROM task WHERE userId = ? AND complete_status = "done"');
        $tasksCompletedStmt->execute([$userId]);
        $tasksCompleted = $tasksCompletedStmt->fetchColumn();
        // fetch workspaces involved
        $workspacesInvolvedStmt = $pdo->prepare('
            SELECT COUNT(DISTINCT w.workspaceId)
            FROM (
                SELECT workspaceId FROM workspace WHERE ownerId = ?
                UNION
                SELECT workspaceId FROM workspacemember WHERE memberId = ?
            ) AS w
        ');
        $workspacesInvolvedStmt->execute([$userId, $userId]);
        $workspacesInvolved = $workspacesInvolvedStmt->fetchColumn();
        // fetch projects involved
        $projectsInvolvedStmt = $pdo->prepare('
            SELECT COUNT(DISTINCT p.projectId)
            FROM project p
            JOIN projectTask pt ON p.projectId = pt.projectId
            JOIN projectSubTask pst ON pt.projectTaskId = pst.projectTaskId
            WHERE pst.assignedMemberId = ?
        ');
        $projectsInvolvedStmt->execute([$userId]);
        $projectsInvolved = $projectsInvolvedStmt->fetchColumn();
        // fetch projects completed
        $projectsCompletedStmt = $pdo->prepare('
            SELECT COUNT(DISTINCT p.projectId)
            FROM project p
            JOIN projectTask pt ON p.projectId = pt.projectId
            JOIN projectSubTask pst ON pt.projectTaskId = pst.projectTaskId
            WHERE pst.assignedMemberId = ? AND p.projectStatus = "completed"
        ');
        $projectsCompletedStmt->execute([$userId]);
        $projectsCompleted = $projectsCompletedStmt->fetchColumn();
        $result = [
            "user" => [$user],
            "activity" => [
                "lastLogin" => $lastLogin,
                "tasksCompleted" => $tasksCompleted,
                "workspacesInvolved" => $workspacesInvolved,
                "projectsInvolved" => $projectsInvolved,
                "projectsCompleted" => $projectsCompleted,
            ],
        ];
        echo json_encode($result);
    } catch (PDOException $e) {
        error_log("Database error: " . $e->getMessage());
        echo json_encode(['error' => 'Database error']);
    }
    exit;
}
```

Figure 4.2.2.5.1 Fetch User Metrics Details

Goalify - userMetricsHandler.php

```php
// contact
if (isset($_GET['getUserEmail']) && isset($_GET['userId'])) {
    $userId = $_GET['userId'];
    $stmt = $pdo->prepare('SELECT email FROM user WHERE userId = ?');
    $stmt->execute([$userId]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);
    if ($user) {
        echo json_encode(["email" => $user['email']]);
    } else {
        echo json_encode(["error" => "User email not found"]);
    }
    exit;
}
```

Figure 4.2.2.5.2 Fetch User's Contact Details

Goalify - userMetricsHandler.php

```php
// user activity
if (isset($_GET['getUserActivity']) && isset($_GET['id'])) {
    include_once '../../includes/db.php';
    global $pdo;
    $userId = $_GET['id'];
    if (!$pdo) {
        echo json_encode(["error" => "Database connection failed"]);
        exit;
    }
    try {
        $queryUser = "SELECT userId, name, DATE(joinedDateTime) AS joinedDate FROM user WHERE userId = :userId";
        $stmtUser = $pdo->prepare($queryUser);
        $stmtUser->execute([':userId' => $userId]);
        $user = $stmtUser->fetch(PDO::FETCH_ASSOC);
        if (!$user) {
            echo json_encode(["error" => "No user found for userId: $userId"]);
            exit;
        }
        $queryActivity =
            "SELECT timestamp, actionType, details, ipAddress, deviceBrowser
            FROM activitylog as al
            JOIN action as a
            ON al.actionId = a.actionId
            WHERE userId = :userId
            ORDER BY timestamp DESC";
        $stmtActivity = $pdo->prepare($queryActivity);
        $stmtActivity->execute([':userId' => $userId]);
        $activity = $stmtActivity->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode([
            "user" => $user,
            "activity" => $activity ?: []
        ]);
    } catch (PDOException $e) {
        echo json_encode(["error" => "Database query failed: " . $e->getMessage()]);
    }
    exit;
}
```

Figure 4.2.2.5.3 Fetch User's Activity Details

Aspect	Description	Implementation
User List & Search	Displays a list of registered users with a search function.	• JS: `userMetrics.js` fetches user data from `userMetricsHandler.php` and updates the user list dynamically.
Fetching User Activity	Retrieves a user's activity log, including login times and interactions.	• PHP: `userMetricsHandler.php` queries user activity logs from the database. • JS: `userActivity.js` fetches and updates the activity table dynamically.
Fetching User Details	Retrieves detailed user information, such as email, status and projects involved.	• PHP: `userMetricsHandler.php` processes user details requests. • JS: `userDetails.js` updates the UI with fetched user data.
Fetching Total Users	Retrieves the total number of registered users.	• PHP: `userMetricsHandler.php` queries `SELECT COUNT(*) AS total FROM users` and returns the count as JSON. • JS: Uses `fetch()` to update the UI dynamically.
Fetching Sign-Up and Active User Metrics	Retrieves sign-up counts per month and active users per week.	• PHP: `userMetricsHandler.php` processes database queries for sign-ups `(SELECT MONTHNAME(joinedDateTime) AS month, COUNT(*) AS count FROM user ...)` and active users `(SELECT DATE_FORMAT(DATE_SUB(timestamp, INTERVAL WEEKDAY(timestamp) DAY, '%m-%d'),COUNT(DISTINCT userId) FROM activitylog WHERE actionId='A001' ...).` • JS: Uses `fetch()` to populate bar and line charts dynamically.

4.2.2.5 Productivity Management

- LUM HAN XUN-

Goalify - productivityHandler.php

```php
if (isset($_GET['getUserDetails'])) {

    global $pdo;
    if (!$pdo) {
        echo json_encode(["error" => "Database connection failed"]);
        exit;
    }

    if (!isset($_GET['userId'])) {
        echo "User ID not provided.";
        exit;
    }

    $userId = $_GET['userId'];

    $stmtUser = $pdo->prepare('SELECT userId, name, profile_img, email
                               FROM user
                               WHERE userId = :userId');
    $stmtUser->execute([':userId' => $userId]);
    $user = $stmtUser->fetchAll(PDO::FETCH_ASSOC);

    $stmtWorkspace =  $pdo->prepare('SELECT w.workspaceId, w.workspaceName, "Owner" AS role, w.createdDateTime AS dateTime
                                     FROM workspace w
                                     WHERE w.ownerId = :userId

                                     UNION

                                     SELECT wm.workspaceId, w.workspaceName, "Member" AS role, wm.joinedDateTime AS dateTime
                                     FROM workspacemember wm
                                     JOIN workspace w
                                     ON wm.workspaceId = w.workspaceId
                                     WHERE wm.memberId = :userId');
    $stmtWorkspace->execute([':userId' => $userId]);
    $workspace = $stmtWorkspace->fetchAll(PDO::FETCH_ASSOC);

    $stmtProject =  $pdo->prepare('SELECT DISTINCT w.workspaceName, p.projectName, p.projectStart, p.projectStatus, p.projectDeadline
                                     FROM project p
                                     JOIN workspace w ON p.workspaceId = w.workspaceId
                                     WHERE p.projectId IN (
                                         SELECT pt.projectId
                                         FROM projectTask pt
                                         JOIN projectSubTask pst ON pt.projectTaskId = pst.projectTaskId
                                         WHERE pst.assignedMemberId = :userId
                                     )');
    $stmtProject->execute([':userId' => $userId]);
    $project = $stmtProject->fetchAll(PDO::FETCH_ASSOC);

    $stmtTask =  $pdo->prepare('SELECT task_name, category, created_at, completed_date, complete_status
                                FROM task
                                WHERE userId = :userId');
    $stmtTask->execute([':userId' => $userId]);
    $task = $stmtTask->fetchAll(PDO::FETCH_ASSOC);

    if (!$user) {
        echo "User not found.";
        exit;
    }

    $result = [
        "user"=> $user ?: null,
        "workspace"=> $workspace ?: null,
        "project"=> $project ?: null,
        "task"=> $task ?: null
    ];

    echo json_encode($result);
    exit;
}
```

Figure 4.2.2.3 Fetch User's Productivity Details

Aspect	Description	Implementation
User Search & Display	Allows admins to search for users and view a summary of registered users.	• JS: `productivity.js` listens for search input using `document.querySelector()` and sends a

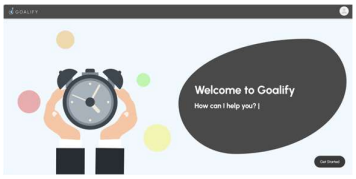
		request to `productivity Handler.php` using `fetch()`. The results dynamically update the user list.
Fetching User List	Retrieves the list of registered users for display.	• PHP: `productivityHandler.php` queries the database using `SELECT * FROM` users, encodes the data in JSON and sends it back to `productivity.js`, which then processes and displays the users.
Fetching Workspace Overview	Retrieves total workspaces and calculates growth rate.	• PHP: `productivityHandler.php` calculates the total number of workspaces using `COUNT(*) FROM workspaces`, determines growth percentage and returns it. • JS: `productivity.js` fetches this data and updates the UI dynamically.
Fetching Project Overview	Retrieves total projects and calculates growth rate.	• PHP: `productivityHandler.php` queries the database for total projects and growth rate, returning results as JSON. • JS: `productivity.js` updates the UI dynamically by modifying `innerHTML` values.
User Productivity Details	Displays detailed information about a selected user's workspace, projects and tasks.	• JS: `productivity_details.js` extracts the user ID from the URL using `URLSearchParams`, fetches data from `productivity`

		`Handler.php` and dynamically updates the HTML tables.
Displaying Users	Shows a list of users based on search or pagination.	• JS: `productivity.js` clears existing user elements, creates div elements dynamically using `document.createElement()` and attaches event listeners for viewing user details.
Fetching Registered Users	Retrieves all registered users.	• PHP: `productivityHandler.php` queries users table with `SELECT * FROM users`, then returns data as JSON. • JS: `productivity.js` processes this response and updates the UI.
Fetching Workspace Statistics	Retrieves workspace statistics and displays growth rates.	• PHP: `productivityHandler.php` calculates statistics based on current and previous month's data, returning percentage changes. • JS: `productivity.js` fetches this data and modifies the UI accordingly.
Fetching Project Statistics	Retrieves project statistics and displays growth rates.	• PHP: `productivityHandler.php` retrieves project statistics from the database and returns a JSON response. • JS: `productivity.js` fetches the data and displays it dynamically.

Fetching User Details	Retrieves specific user details such as workspaces, projects and tasks.	• JS: `productivity_details.js` sends a request to `productivityHandler.php` with the user ID, which then queries the database and returns structured JSON data to be displayed.

4.3 Key Features

4.3.1 User Web Page



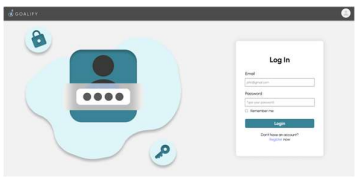
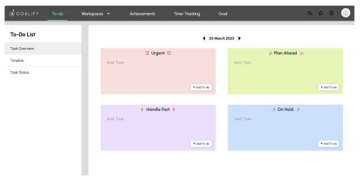
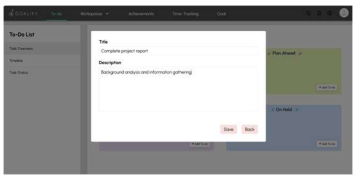


Figure 4.3.1.1, Figure 4.3.1.2 & Figure 4.3.1.3 Register Page, User Home Page & Login Page





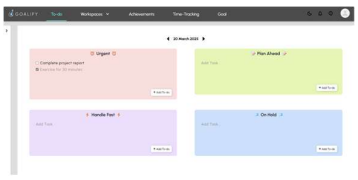
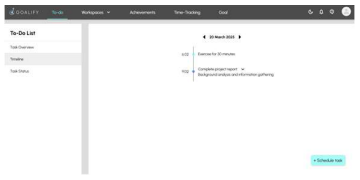
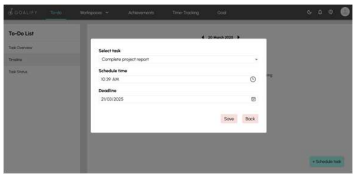


Figure 4.3.1.4, Figure 4.3.1.5 & Figure 4.3.1.6 To-Do Pages





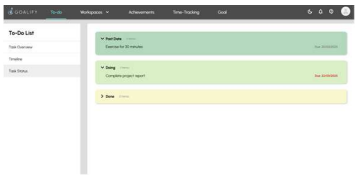
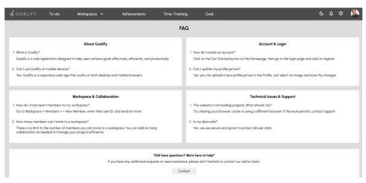


Figure 4.3.1.7, Figure 4.3.1.8 & Figure 4.3.1.9 To-Do Pages



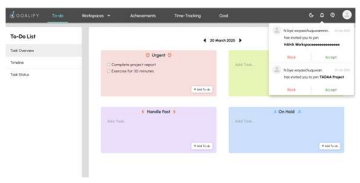


Figure 4.3.1.10 & Figure 4.3.1.11 Help Pages || Figure 4.3.1.12 Inbox Page

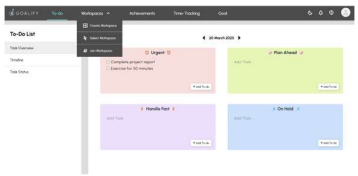
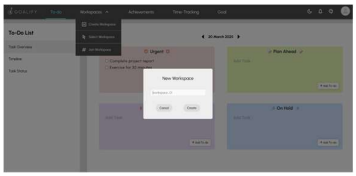
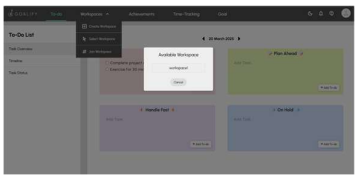




Figure 4.3.1.13, Figure 4.3.1.14 & Figure 4.3.1.15 Workspaces Pages

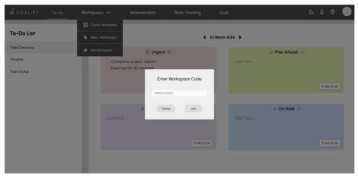
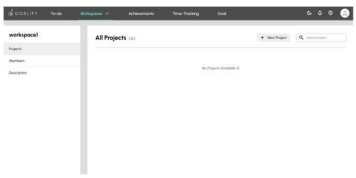
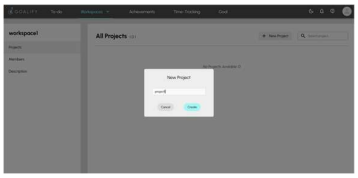




Figure 4.3.1.16, Figure 4.3.1.17 & Figure 4.3.1.18 Workspaces Pages

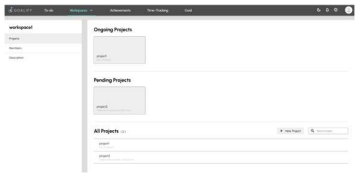


Figure 4.3.1.19, Figure 4.3.1.20 & Figure 4.3.1.21 Workspaces Pages

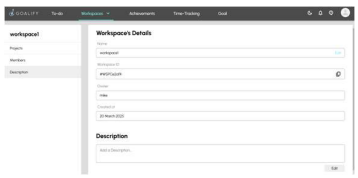
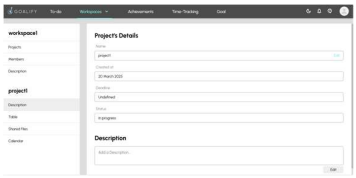
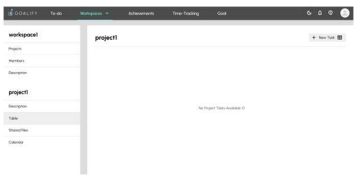




Figure 4.3.1.22, Figure 4.3.1.23 & Figure 4.3.1.24 Workspaces Pages

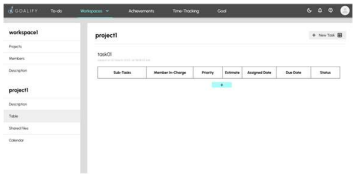
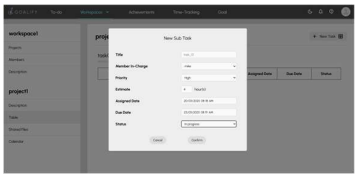




Figure 4.3.1.25, Figure 4.3.1.26 & Figure 4.3.1.27 Workspaces Pages

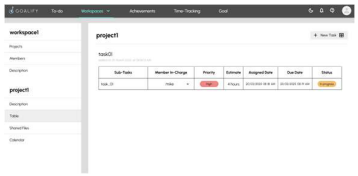
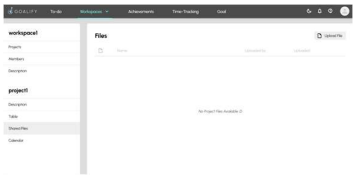




Figure 4.3.1.28, Figure 4.3.1.29 & Figure 4.3.1.30 Workspaces Pages

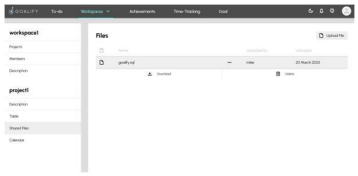
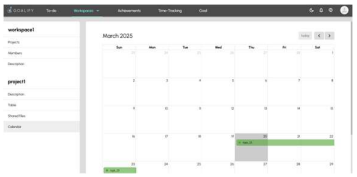
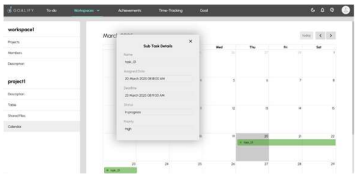
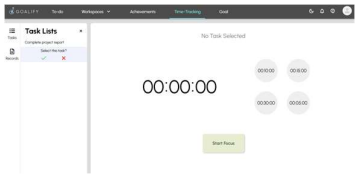
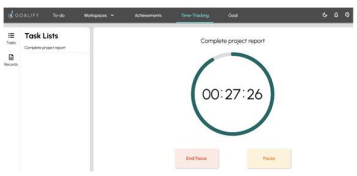




Figure 4.3.1.31, Figure 4.3.1.32 & Figure 4.3.1.33 Workspaces Pages





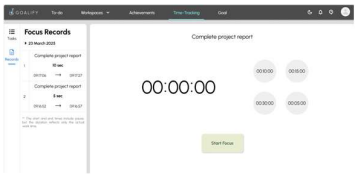


Figure 4.3.1.34, Figure 4.3.1.35 & Figure 4.3.1.36 Time-Tracking Pages

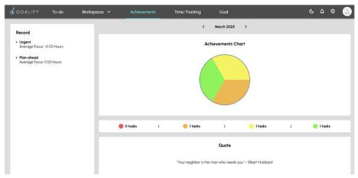
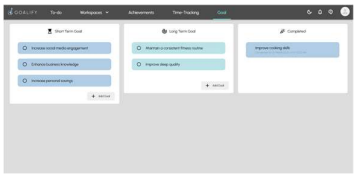
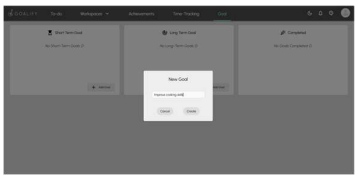




Figure 4.3.1.37 Achievement Page || Figure 4.3.1.38 & Figure 4.3.1.39 Goal Pages

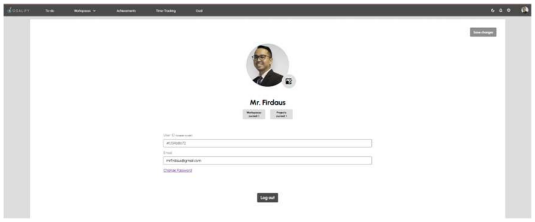



Figure 4.3.1.40 & Figure 4.3.1.41 Profile Page

4.3.2 Administrator Web Page

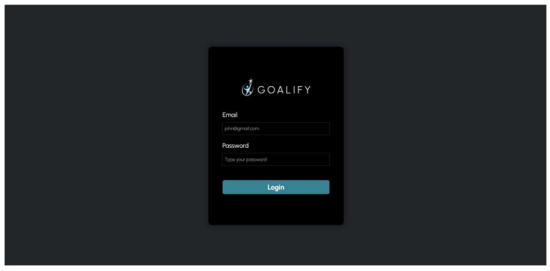
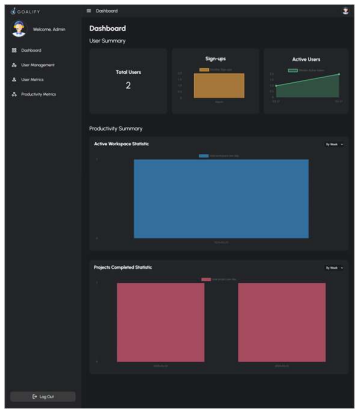



Figure 4.3.2.1 Admin Login Page & Figure 4.3.2.2 Admin Dashboard Page

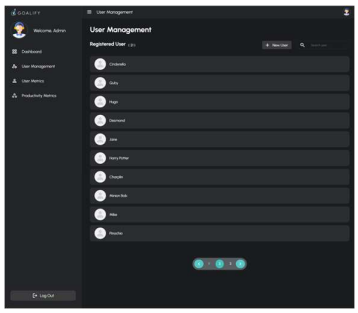
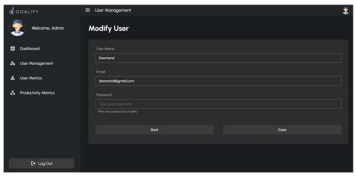
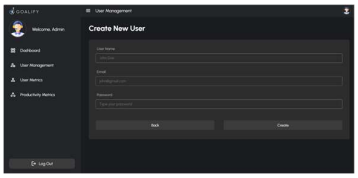
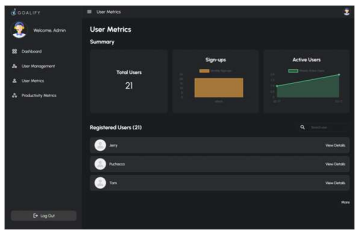
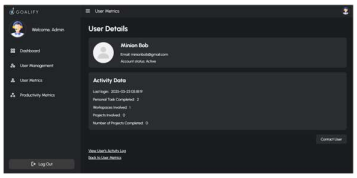




Figure 4.3.4.3, Figure 4.3.4.4 & Figure 4.3.4.5 User Management Pages





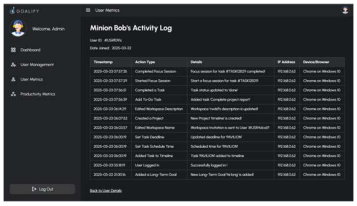
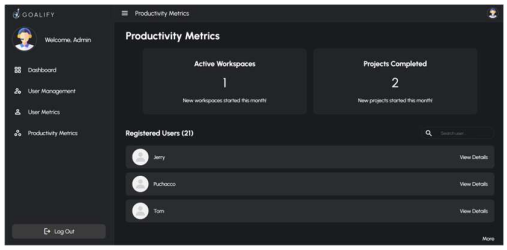


Figure 4.3.4.6, Figure 4.3.4.7 & Figure 4.3.4.8 User Metrics Pages



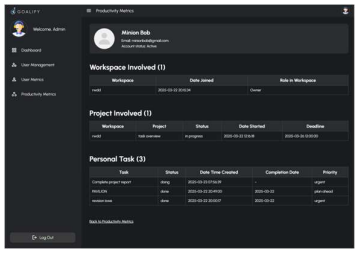


Figure 4.3.4.9 & Figure 4.3.4.10 Productivity Metrics Pages

5.0 Conclusion

'Goalify' is designated to deliver an optimally efficient, user-centric, and collaborative productivity tool, aiding users in managing their personal day-to-day tasks, workspaces, and projects seamlessly. With highly customizable project and task scheduling features that also comes with additional collaboration features, this website comprehensively enhances user productivity while promising an intuitive navigating experience. Apart from task management and collaboration, this website simplifies productivity tracking with a monthly task completion and focus analysis, allowing users to easily monitor their progress and evaluate their monthly productivity level. Users may also define goals via the goal setting feature, categorizing user goals into short-term and long-term goals, providing a clear

vision and motivating users to complete both goals that are to be emphasized in the present and in the long run. Additionally, real-time reminders on task deadlines are also implemented, ensuring users are always ahead of the deadlines.

'Goalify' also provides an administrative platform for managing system-level operational functions. Currently, this system has a single system administrator with predefined credentials that cannot be altered. The administrator may perform CRUD (Create, Read, Update, Delete) operations on system users, oversee user metrics, and monitor user productivity statistics. By focusing on usability, efficiency, and control, the administrator acts as the central hub of the platform, ensuring sustainable alignment with the system objectives.

The integration of PHP, MySQL, JavaScript, and CSS successfully delivered a dynamic, responsive, and efficient productivity website. Further refinements to be considered in order to accommodate future expansion are listed below:

(I) Multiple Administrators and Role-Based Access Control (RBAC)
Introducing a hierarchical administrator system, where more than one admin can manage system operations and health, enforcing role-based permission levels for different system users, maintaining structured system management.

(II) Advanced Reporting and Analytics
Implementing automated reporting systems, detailing the project progress, team performance, and workspace activity, with graphical illustrations as visual aid and exportable data providing notable insights for calculated decision-making.

(III) User-Admin Communication
Enhancing the existing support system by integrating an in-platform communication channel, allowing users to submit queries and request assistance, enabling administrators to respond effectively and address user pain points in real time.

(IV) Real-Time Chat in Workspaces
Integrating instant messaging within workspaces, allowing team members to communicate seamlessly, improving collaboration experiences, information sharing, and task coordination without needing external communication tools.

(V) Cloud Storage Integration for File Management
Migrating file sharing from local database to cloud storage such as Firebase, One Drive, or Google Drive, seamlessly handling large files, reducing database load, and elevating system performance.

(VI) Enhanced Security Measures

Strengthen system security by implementing multi-factor authentication or biometric authentication, role-based access restrictions, and more comprehensive audit logs, tracking all system users' actions, security events, and system changes.

6.0 References

(2025). Youtu.be. https://youtu.be/d2ve7xQNco8?si=YL8QlFS_BDfOlAzp

Decode a user agent. (2020, August 16). SitePoint Forums | Web Development & Design Community. https://www.sitepoint.com/community/t/decode-a-user-agent/356311/3

GeeksforGeeks. (2020, March 2). *How to Get Local IP Address of System using PHP ?* GeeksforGeeks. https://www.geeksforgeeks.org/how-to-get-local-ip-address-of-system-using-php/

GeeksforGeeks. (2024, September 20). *PHP determining client IP Address*. GeeksforGeeks. https://www.geeksforgeeks.org/php-determining-client-ip-address/

GeeksforGeeks. (2024, September 23). *How to identify server IP address in PHP?* GeeksforGeeks. https://www.geeksforgeeks.org/how-to-identify-server-ip-address-in-php/

Get the client IP address using PHP. (n.d.). Stack Overflow. https://stackoverflow.com/questions/15699101/get-the-client-ip-address-using-php

Gilbertson, D. (2023, January 28). A simple pie chart in SVG - David Gilbertson - Medium. *Medium*. https://david-gilbertson.medium.com/a-simple-pie-chart-in-svg-dbdd653b6936

Howarth, J. (2024). 51 Employee productivity & engagement stats (2024). *Exploding Topics*. https://explodingtopics.com/blog/employee-productivity-stats

Jude. (n.d.). 20+ Student productivity statistics and trends in 2024. *NotionYou*. https://www.judkin.com/blog/student-productivity-statistics-and-trends?srsltid=AfmBOor-bTt381Qh4qHf3tOMLWJRTd-zpyHbIYaOo6iUK00Jif46VGxV

Laurence Svekis. (2024, August 26). *Build a custom tooltip in JavaScript | Taught by Laurence Svekis* [Video]. YouTube. https://www.youtube.com/watch?v=fDYc5r_TUCo

PHP: Hypertext Preprocessor. (n.d.). https://www.php.net/manual/en/reserved.variables.server.php

Step-by-step guide | Chart.js. (n.d.). https://www.chartjs.org/docs/latest/getting-started/usage.html

svg arc, how to determine sweep and larg-arc flags given start end & via point. (n.d.). Stack Overflow. https://stackoverflow.com/questions/21816286/svg-arc-how-to-determine-sweep-and-larg-arc-flags-given-start-end-via-point

User-Agent - HTTP | MDN. (2025, March 13). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/User-Agent

W3Schools.com. (n.d.). https://www.w3schools.com/Bootstrap/bootstrap_ref_js_tooltip.asp

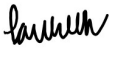
What is MySQL? (2023). Oracle.com. https://www.oracle.com/my/mysql/what-is-mysql/

ZenQuotes.Io. (2023, March 12). *ZenQuotes Documentation*. ZenQuotes.io Docs. https://docs.zenquotes.io/zenquotes-documentation/#call-quotes

ZenQuotes.Io. (n.d.). *Zen Quotes API | Daily Inspirational Quotes*. ZenQuotes.io. https://zenquotes.io/#:~:text=Fetch%20Random%20Quotes,%5Bkey%5D%20%3D%20optional

7.0 Workload Matrix

Work Breakdown	Lim Chee Xuan	Lum Han Xun	Ng Yvonne	Paureen Tan Nie Nie	Phang Shea Wen	Total

Introduction						
• Title and Target Users of the Web Application	20 %	20 %	20 %	20 %	20 %	100 %
• Objectives of the Project	20 %	20 %	20 %	20 %	20 %	100 %
Background Analysis and Requirement Gathering						
• Web Application Context	20 %	20 %	20 %	20 %	20 %	100 %
• End-user	20 %	20 %	20 %	20 %	20 %	100 %
• End-user Functional Requirements	20 %	20 %	20 %	20 %	20 %	100 %
• Non-functional Requirements	20 %	20 %	20 %	20 %	20 %	100 %
Flowchart						
• User	20 %	20 %	20 %	20 %	20 %	100 %
• Admin	20 %	20 %	20 %	20 %	20 %	100 %
ERD Diagrams	20 %	20 %	20 %	20 %	20 %	100 %
• Data Dictionary	20 %	20 %	20 %	20 %	20 %	100 %
Wireframe	20 %	20 %	20 %	20 %	20 %	100 %
Navigational Structure	20 %	20 %	20 %	20 %	20 %	100 %
Signature						

Group Number : Group 1

Group Leader Name : Phang Shea Wen

Member Name and Student ID (including Group Leader)	Task Completed				Overall Contribution %
	Documentation	**Web App (Client-side)**	**Web (Server Side)**	**Database**	
Lim Chee Xuan TP075916	• Wireframe • Explain code snippets	Frontend of • User register and login Frontend and backend of • Goal • Notification	-	Design table • goal	25%
Lum Han Xun TP076160	• Introduction • Background Analysis • Flowchart • Explain code snippets	Frontend and backend of • Time-tracking • Notification	Frontend and backend of • Dashboard • Productivity Metrics	Design table • focusrecord • workspaceinvitation	25%
Ng Yvonne TP076390	• Introduction • Background Analysis • Flowchart • Explain code snippets	Frontend and backend of • To-do • User Profile	-	Design table • task • timeline	25%
Paureen Tan Nie Nie TP075914	• Requirement Gathering • ERD • Data Dictionary • Navigational Structure • Explain code snippets	Frontend and backend of achievements • Achievements • Contact support	Frontend and backend of • Admin Login • Dashboard • User Metrics	Design table • action • activitylog	25%
Phang Shea Wen TP075813	• Project Plan • Requirement Gathering • ERD • Data Dictionary	Frontend and backend of • User interface (top, side and bottom bar) • User register and login	Frontend and backend of • Admin interface (top, side and bottom bar) • User Management	Design table • project • projectfiles • projectsubtask • projecttask • remembermetoken • user • workspace	25%

	• Navigational Structure • Wireframe • Conclusion • Explain code snippets	• Workspaces • Dark theme and light theme		• workspacember	
				Total	100%