Technology Park Malaysia

AAPP006-4-2

Software Development Project Final Year Report

Team: Group E

Project Title: Projectify

Intake: UCDF2208ICT(SE)

Group Member		
No.	***Name***	***TP Number***
1	CHOK JUNG SHING	TP076074
2	LEONG SENG KHUAN	TP070856
3	TAN PO YEH	TP070780
4	TONG JIA CHUEN	TP070484

Table of Content

1.0 Acknowledgement

First and foremost, we would like to thank our Lecturer, Mr. Justin Gilbert A/L Alexius Sil Vester, who helped us learn a lot about this project. His vast knowledge, extensive experience, and comments enabled us to successfully accomplish our project, "Projectify". Without his help, we would not have been possible to finish our final semester project. We could not have asked for any senior in our studies. In other words, without the contributions of each individual in this project, we would not have been a success in accomplishing our project. Moreover, I would like to thank our University "Asia Pacific University of Technology & Innovation (APU)" for letting us explore such project in the final semester.

2.0 Abstract

Presentation Title: Final Year Project Management System

Research Focus: Moodle

Presentation Type: Poster Presentation

Abstract:

The primary objective of this project was to enhance the features and functionality of a Final Year Project Management system by drawing inspiration from Moodle, a renowned learning management platform. Moodle has been widely adopted in educational institutions due to its strong capabilities, and yet it still has certain limitations when it comes to managing complex student projects and facilitating seamless collaboration. By referencing Moodle, we mainly focus on developing a comprehensive platform that integrates with project management features and discussion forums features. With these features, students can benefit from streamlined workflows, efficient task tracking, and enhanced communication channels that are critical to successful project outcomes. Moreover, designing and implementing effective systems that support students' learning experiences is also a part of our project. We aim to create a user-friendly interface that simplifies the complex processes associated with project management, making it easier for the students to focus on their academic work. The platform will also include tools for supervisors to monitor progress, provide feedback, and facilitate regular communication, ensuring that students receive the guidance and support they need.

In summary, "Projectify" aims to revolutionize the way students and educators manage their schoolwork or projects by providing a comprehensive, interactive, user-centric platform. By

solving the challenges of traditional project management systems. Our projects aim to set new standards in educational technology and promote better communication, collaboration, and productivity in academic environments.

3.0 Workload Matrix

Name	Task	Signature
Name: Tan Po Yeh **TP Number: TP070780**	6.0 System Hierarchy Chart 7.0 Context Diagram 8.0 Data Flow Diagram Level 0 9.3 Data Flow Diagram Level 1 - Tan Po Yeh 10.3 Data Dictionary - Tan Po Yeh 11.0 Entity Relationship Diagram 13.1 Unit Testing 14.0 Significant Code Sources 17.0 Appendix	
Tong Jia Chuen **TP Number: TP070484**	4.0 Introduction 15.0 Conclusion and Limitations 9.4 Data Flow Diagram Level 1 – Tong Jia Chuen 10.4 Data Dictionary - Tong Jia Chuen	
Chok Jung Shing **TP Number: TP070674**	5.0 Project Plan 9.1 Data Flow Diagram Level 1 – Chok Jung Shing 10.1 Data Dictionary – Chok Jung Shing 12.0 Screens Design, Report Design and User Manual	
Leong Seng Khuan **TP Number: TP070856**	1.0 Acknowledgement 2.0 Abstract 3.0 Workload Matrix 9.2 Data Flow Diagram Level 1- Leong Sheng Khuan 10.2 Data Dictionary – Leong Seng Khuan 13.2 User Acceptance Testing	

4.0 Introduction

4.1 Project Background

Our app, “Projectify” is a user-friendly app is designed to be better than the current platforms like Moodle which have been widely used among universities for Study purposes. We designed “Projectify” with more interactive features and innovative usage to improvise and revolutionize how students and lecture manage their learning tools. Moreover, this app “Projectify” can provide virtual classrooms where students can engage in collaborative topics or seek help with their assignments. With this real-life chat system being built in our app student can access to more than hundred interactive lectures, students and even outsiders. To add on, student can benefit by using the real-life chat system, with our more than hundred interactive people, the student can boost their interpersonal skills and overall academic. Additionally, our app “Projectify” is designed to make student more convenience by simplifying the organization and tracking of student assignments, tasks and deadlines. Likewise, it can efficiently help students by managing their academic workload and making sure they’re staying on the right track achieving their goals. In the nutshell, our app simplified the complexity of online academic resources in university such as Moodle

4.2 Problem Context

1. The task of the administrators are complex:

The project that we are doing is related to educational organizations, there is no doubt it is used with a lot of different activities such as course subjects, student enrolment and other else which makes it time-consuming and challenging for our administrators.

1. Challenges in monitoring student progress:

Since a large sum of students will be using our app “Projectify”, managing the students’ performance of data throughout different course programs and subjects can be tough and challenging, which will be difficult to identify areas for improvement.

2. Restricted partnership and interaction:

With our lack of information systems and communication channels, it can make it hard for the administrators, instructors, and students to work together, therefore affecting our resource allocation and decision-making.

3. Reduced student satisfaction and engagement:

If our app is not designed or properly manage in the system, it can end up not completed and unsatisfied learning experience for students. Which in results of affecting their overall academic.

4. Larger administrative expenses:

With time being consuming, additional staff is needed and to manage the assets will be hard to handle that could lead to a bad and poor system. This could potentially cause more expenses.

4.3 Proposed Solution

1. Supervisor Oversight:

Our "Projectify" is designed to ease supervisors with tracking down the progress of multiple case, offering easy inputs and it allows easy of use to communicate with our real-life chat system. This will allow the supervisors to manage and support students if they face difficulties.

2.Dashboard:

With our "Projectify", we will be managing all the final-year projects, which has a dashboard for each student that includes the project status and other assignment deadlines. This allows students to easily monitor their own assignment/project progress with efficient.

3.Communication and Collaboration:

With our integrated chat box. It functions to facilitate the easy communication and collaboration between students, supervisors, and any assignments for scheduling a meet physically for exchanging idea, sharing their thoughts among friends and resolving past year

questions through the web. This will benefit the students and will be able to provide regular updates and coordinate task for their assignments.

4.Integration and Customization:

With the flexible customization features in Projectify, it provides student with their innovative thinking to customize and organize their profile or course material to their own liking.

4.4 Project Objectives

Our main goal of this object is to make it efficient for the administrative progress by focusing on the four goal which is, “creating a user-friendly interface”, “introducing technology for automation”, “integrating data analytics”, and “improving collaboration capabilities”. We hope to reach the objectives as it can give many benefits to the use of reducing the decision-making and improving stakeholder communication. Other than that, we want to achieve and hope to overcome any obstacles that we faced, such as restricted cooperation and difficult work. Not to mention, we really hope that we could create our administrative program structure that satisfied all needs of students, teachers, and administrators that is efficient and effective. With all of these being achieve, it can result in better educational and satisfaction among users.

4.5 Project Scope

Objectives:

Our app “Projectify” Platform manages Final Year Projects that allows student’s projects/assignment for submission and to track down each the progress for both students and lectures can monitor them. Its aims are to improve the student habits, skills, and productivity.

1.Primary Goals:

- To Centralized Students submission and collaboration among lecture, outsider and student.
- To improve communication through chat systems or discussion forums.
- To be able to provide lecture with tools to monitor and review students’ projects
- Increase accountability and transparency by allowing students to track progress
- To simplified admin tasks.

2.Scope:

"Projectify" features include:

- Discussion Forums
- Integrated calendar
- Dashboard system
- User profiles
- Real-Life chat system
- Progress tracking
- Admin and channel management

3. Deliverables:

- Fully functional, user-friendly web platform.
- Efficient project management tools.
- Technical assistance for dependability and skills.

5. Milestones:

- Development Kick-off: Week 3-5
- Completion of System Design: Week 6-8
- Implementation of Core Features (Discussion Forums, Calendar, Dashboard): Week 8
- User Profile and Admin Management System Integration: Week 9
- Implementation of Channel Management and Progress Tracking: Week 10
- Final Testing on the System and Bug Fixing: Week 11
- Deployment Time and User Training: Week 12

5.Technical needs:

- By using programing languages such as (JavaScript, HTML, CSS, PHP).
- Safeguard authentication and authorization
- Flexible database (SQL/NoSQL).

- Responsive design for all devices
- API integration for calendar and notifications.
- Real-time chat using WebSocket's or similar functions.

6. Possible Limitations or Challenges:

- **Resources:** Limited development, testing, and maintenance staff.
- **Limited Time**: Tight deadlines may limit features, there for making user experience unsatisfied.
- **Security**: Requires robust data protection.
- **User training:** Needs additional resources for user training to adapt with the system interface and functionality.
- **Scalability**: System performance may be strained by rapid growth.
- **Expectations**: Managing diverse stakeholder expectations.

7.Conclusion:

"Projectify" will boost Final Year Project (FYP) administration with a comprehensive and user-friendly platform that improves communication and productivity among students and teachers.

5.0 Project Plan

5.1 System Development Methodology

For developing "Projectify", our Final Year Project Management System, we have chosen the Agile Development Methodology. This methodology is particularly suitable for our project because it can allow the team to respond quickly to provide feedback and make changes to the requirements. It also allows for flexibility, continuous improvements, and it is able to adapt to changing requirements after every small change (Laoyan, 2024). Below will be the explanation of why Agile is used and considered most appropriate methodology for our project.

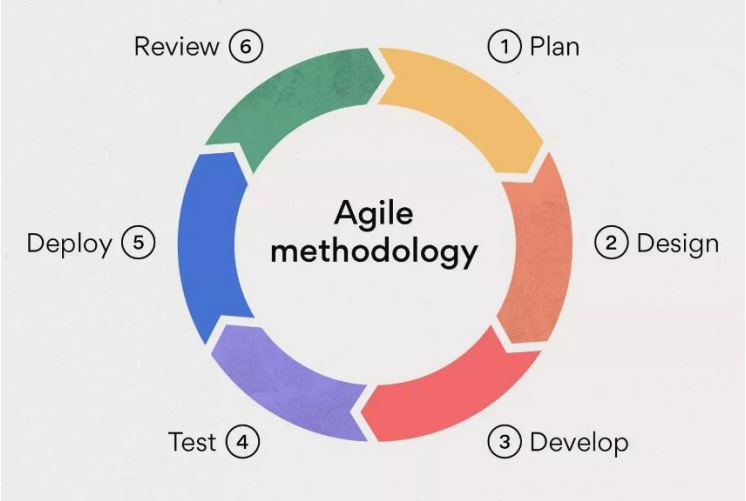


Figure 5.1 1

Figure 5.1 Agile Methodology

Reasons & Benefits of Agile Development Methodology:

1. Flexibilities and Adaptability:
   It is crucial for "Projectify" to use this method as it allows us to adapt to changing requirements easily throughout the whole development process. Feedback given from students and lecturers may provide good future improvements and any adjustments to the functionalities (Students, 2022).

2. Additional Delivery:
   Breaking the project into smaller parts or you can call it manageable chunks (sprints) is a better process for our development. As it allows us to deliver parts in better ways and also to detect any issues during the development process.

3. Improvement and continuous feedback:
   This methodology will benefit with regular feedback from stakeholders (students, lecturers, and administrators). It gives nonstop feedback that helps to make necessary improvements and adjustments, improving the system's usability and also functionalities.

4. Effective Communication:

Agile encourages effective communication among all team members and stakeholders. For "Projectify", this means that developers, students, lecturers, and administrators can have regular meetings to discuss progress, issues, and upcoming tasks.

5. Customer's Approach:
   The methodology aims to deliver perfect system to the users. By giving them the best experience and friendly-user software. Engages with students and lecturers with the system that meets their needs.

Disadvantages of Agile Development Methodology:

Although there are lots of benefits using this methodology, there are also disadvantages to it. More time and commitment are used using this methodology, because it requires lots of time for the stakeholders to give feedback and time consuming

Implementation of "Projectify":

Plan

1. Project Planning
   - Defining project vision and objectives.
   - Identify the stakeholders
   - Create a high-level roadmap

2. Phase Planning
   - Divide project in phases
   - Define the project goals
   - Prioritize project tasks

Design

1. Phase Planning (Continued)

- Define detailed user requirements and design specifications for each phase.

Develop

1. Phase Execution
    - Start to develop and implement the system features
    - Integration and testing part

Test

1. Phase Execution (Continued)
    - Continuous integration & testing
    - Ensuring code changes are tested correctly

2. Phase Review
    - Showcase the system to the stakeholders
    - Demonstrate new features and improvements made during the phases
    - Collect feedback from stakeholders for any changes

Deploy

1. Release and Deployment
    - Plan regular releases and deploy new features for improvements.
    - Ensure users have access to latest functionality and features.

Review

1. Evaluation Performance

- Conduct regular meetings to evaluate the team's performance.
- Identify and focus on improvements.

2. Monitoring and Maintenance
    - Always monitor the system's performance and its stability.
    - Use monitoring tools to detect any possible issues.
    - Schedule maintenance and regular updates to fix bugs and ensure system's security

3. Documentation and Training
    - Create detailed documentation that includes the system information.
    - Provide training classes to stakeholders so that they know how to use it effectively.

5.2 Project Gantt Chart

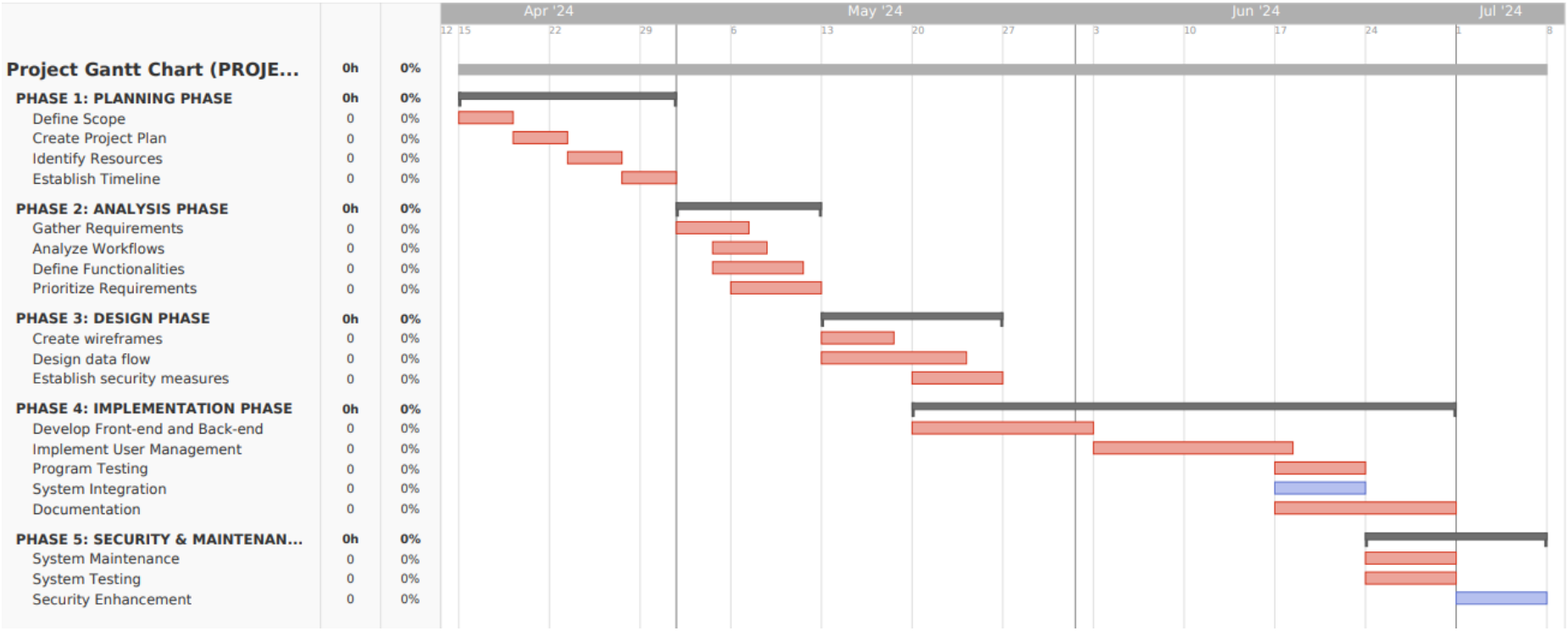
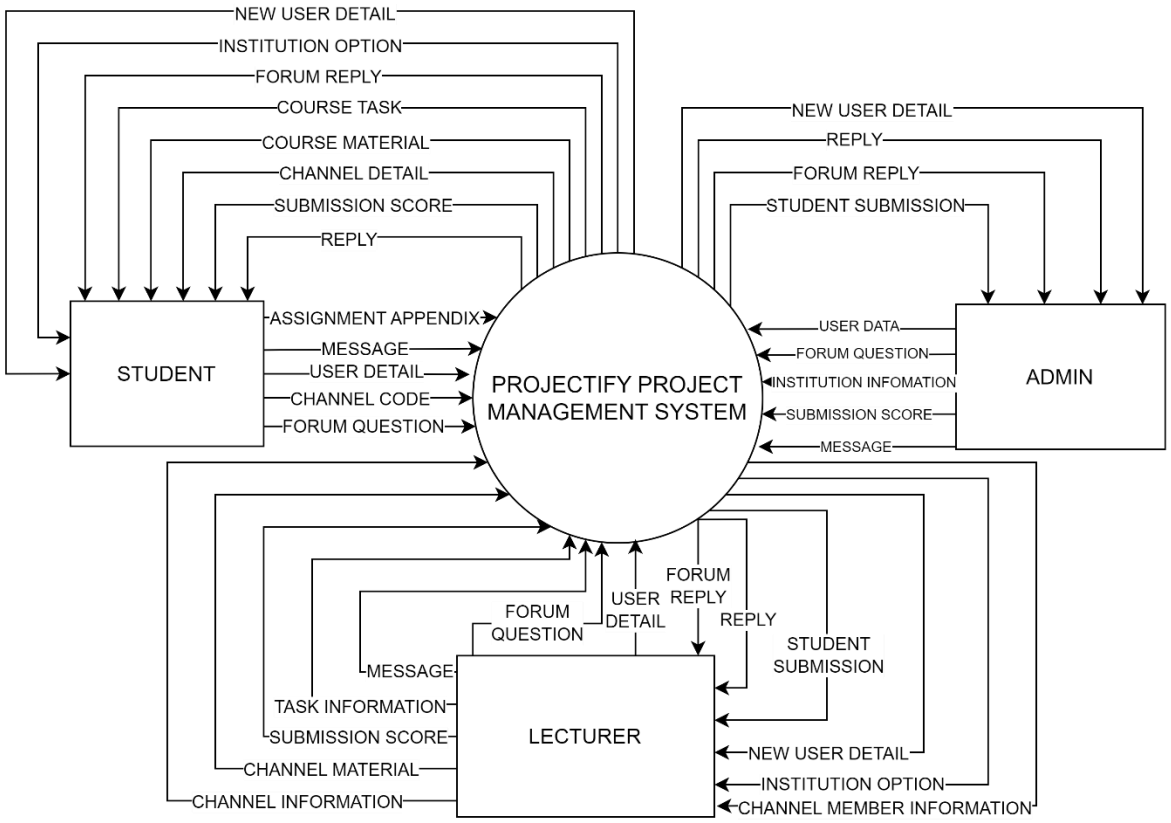


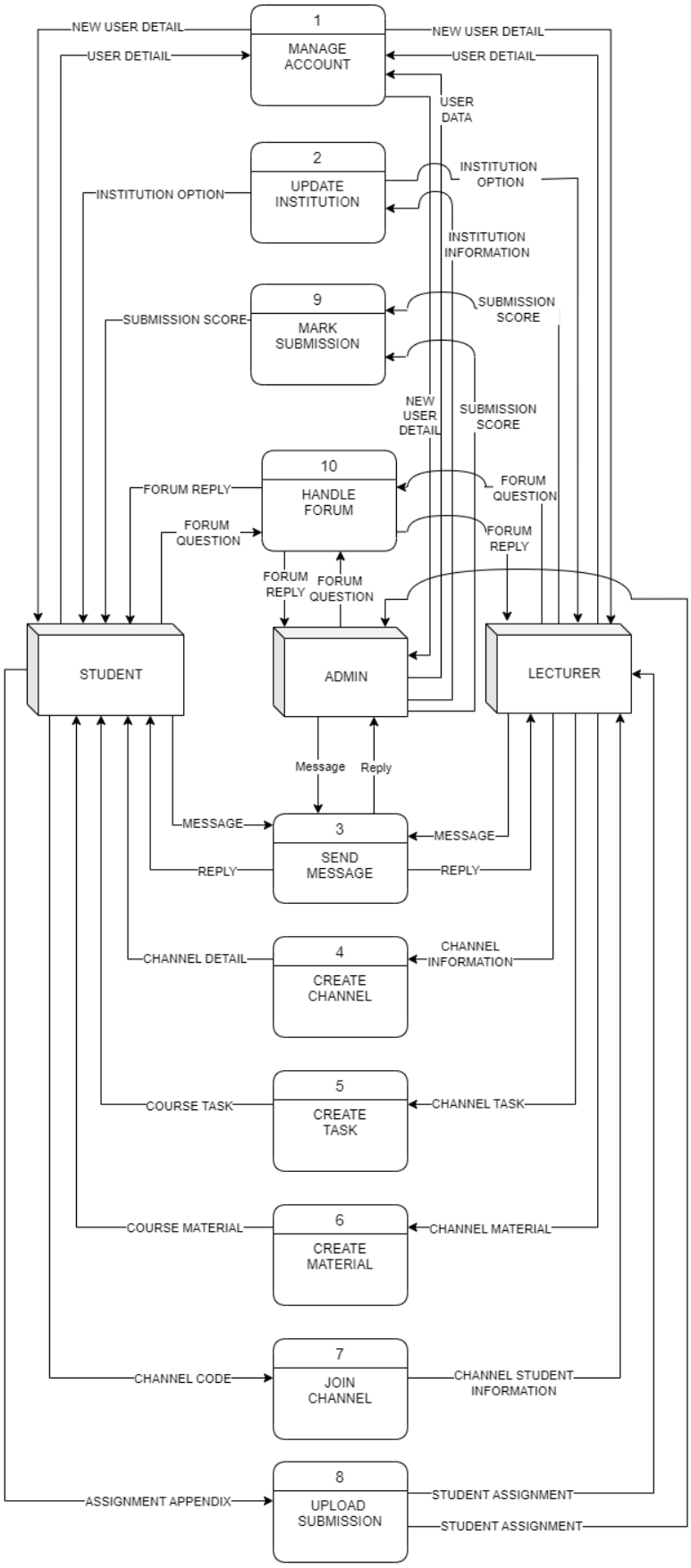
Figure 5.2.1 Project Gantt Chart

6.0 System Hierarchy Chart

7.0 Context Diagram



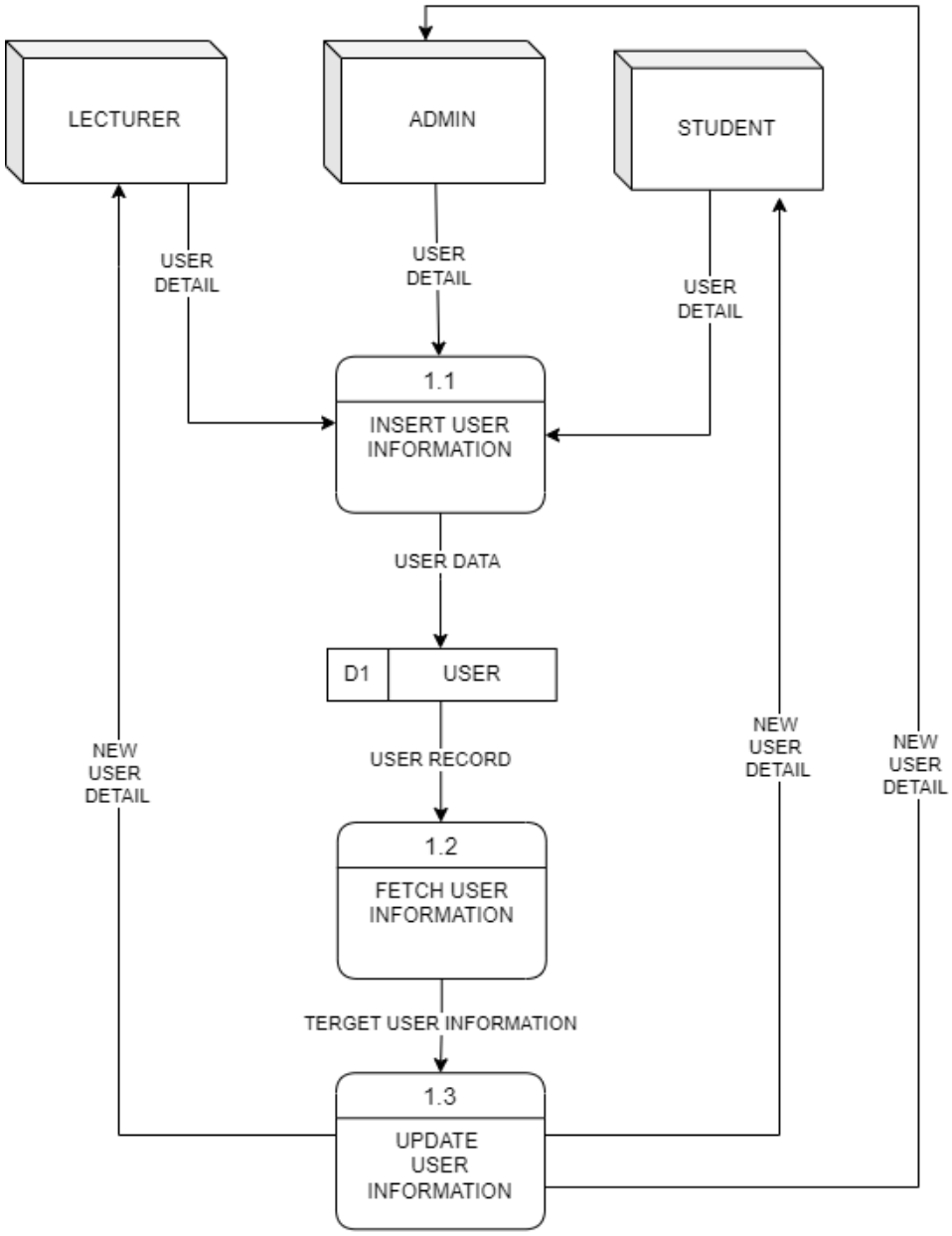
8.0 Data Flow Diagram Level 0



9.0 Data Flow Diagram Level 1

9.1 Data Flow Diagram Level 1 - Chok Jung Shing

9.1.1 User Manage Account Information

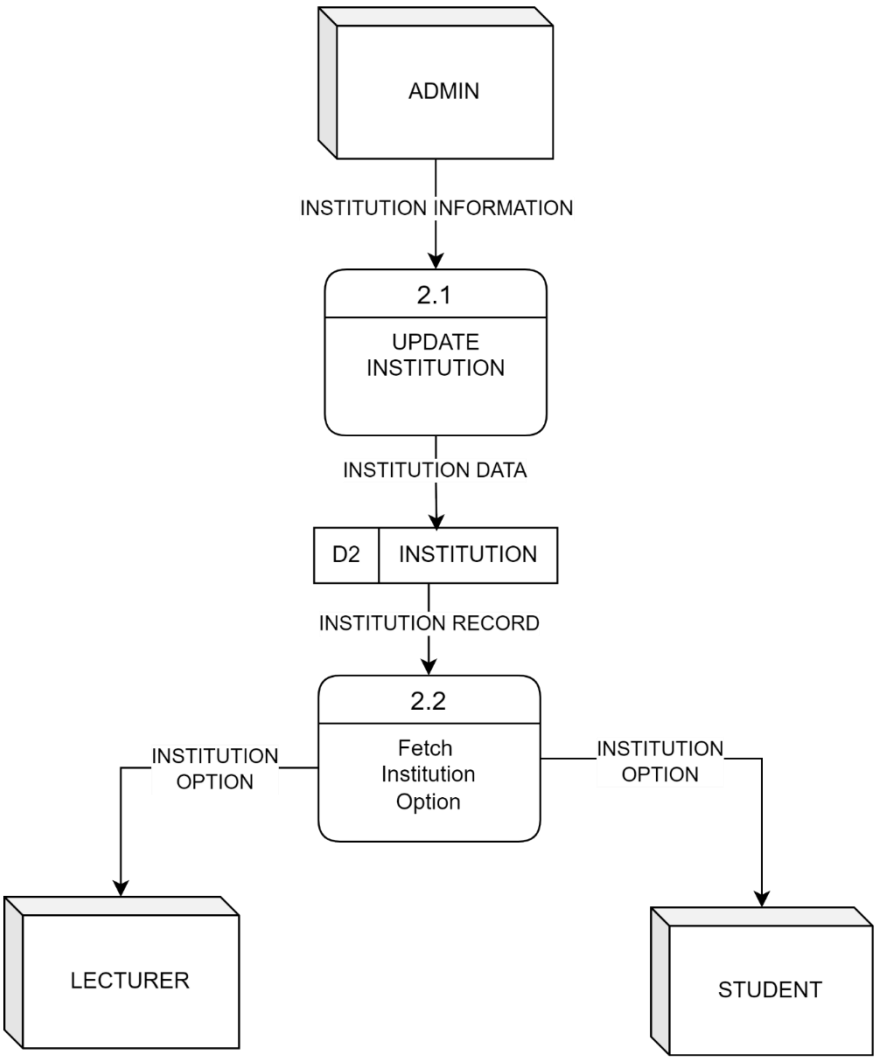


This Data Flow Diagram (DFD) Level 1 contains three processes including 1.1 INSERT USER INFORMATION, 1.2 FETCH USER INFORMATION, 1.3 UPDATE USER INFORMATION. The diagram involves three external entities: Student, Admin, and Lecturer. Students, Admin and Lecturers insert their new user information to the system, which then processes this user information in several steps.

The 1.1 INSERT USER INFORMATION process inserts new user information from any of the three entities. This data, along with the user information, is then stored in the data store D1 USER. Then the 1.2 FETCH USER INFORMATION process fetch user record from D1 USER to get user information. The 1.3 UPDATE USER INFORMATION process is responsible for updating their user information.

In summary, this Level 1 DFD illustrates the message processing workflow, from inserting user detail for new user information and storing inside the user database. Then, extracting relevant information from the target and updating their user information, forwarding it to the user. The D1 USER data store holds the user information and related data throughout this process.

9.1.2 Update Institution



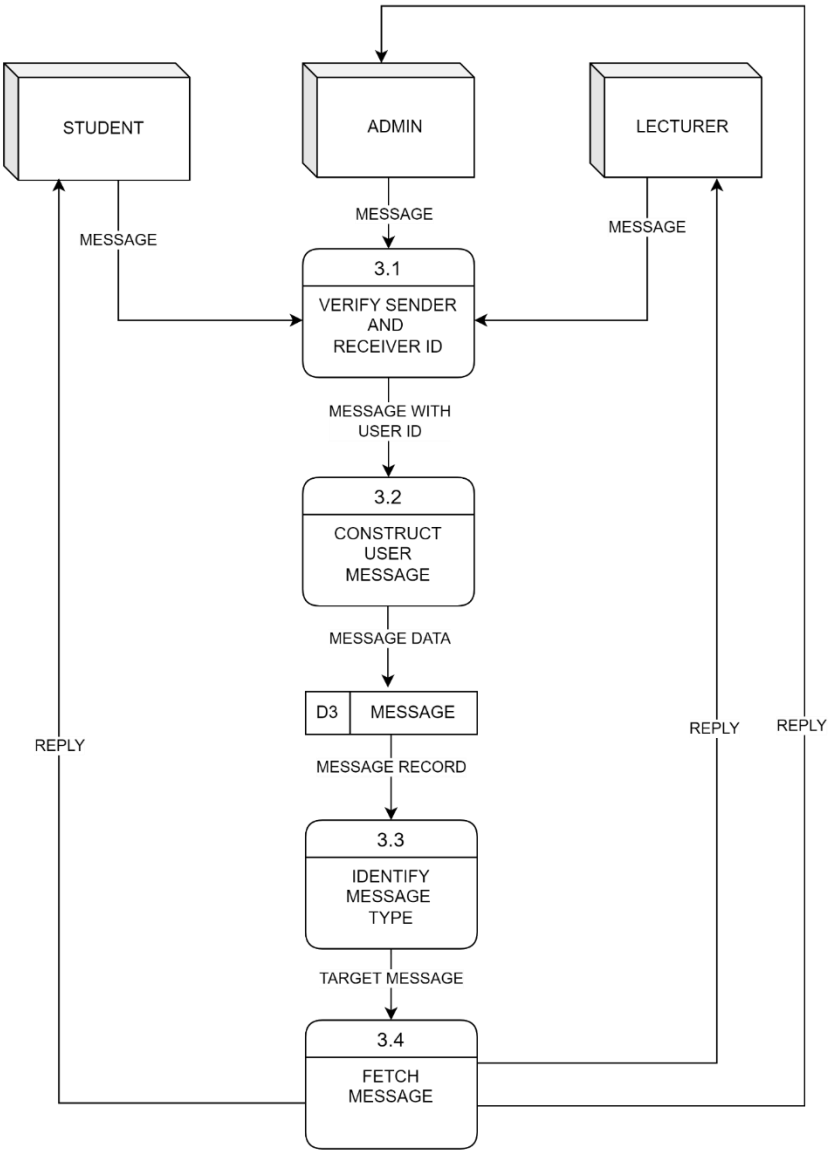
This Data Flow Diagram (DFD) Level 1 contains two processes, which include 2.1 UPDATE INSTITUTION, 2.2 FETCH INSTITUTION OPTION. The diagram involves three external entities: Student, Admin, and Lecturer. Admin send institution information to the system, which then processes this information in several steps.

The 2.1 UPDATE INSTITUTION process admin insert new institution information and update. This data will then be stored in the data store D2 INTITUTION. The 2.2 FETCH INSTITUTION OPTION will then fetch updated institution information and give an option for Lecturers and Students.

In summary, this Level 1 DFD illustrates admin updating institution processing workflow, from inserting new institution information, extracting relevant information, forwarding the option to the recipient. The D2 INTITUTION data store holds the new institution information and related data throughout this process.

9.2 Data Flow Diagram Level 1- Leong Sheng Khuan

9.2.1 Send Message



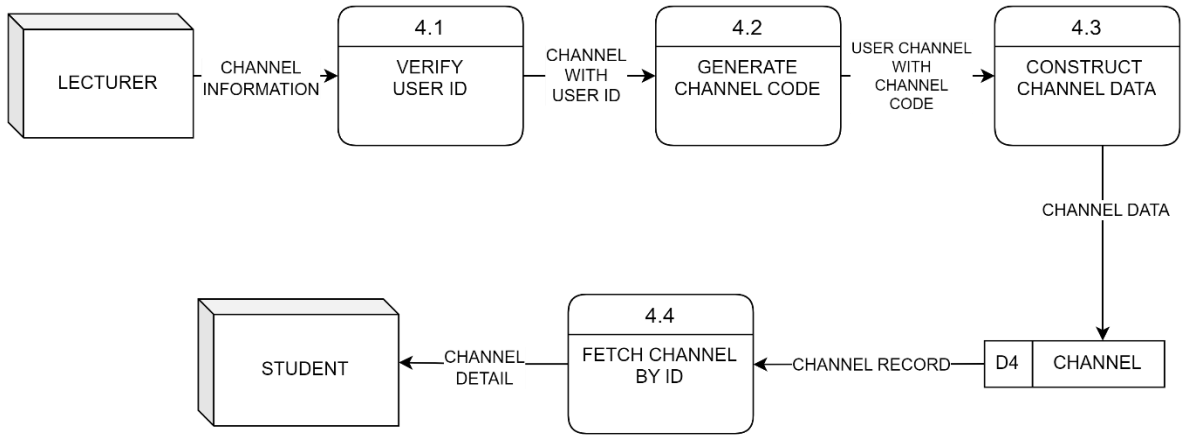
This Data Flow Diagram (DFD) Level 1 contains four processes: 3.1 VERIFY SENDER AND RECEIVER ID, 3.2 CONSRUCT USER MESSAGE, 3.3 IDENTIFY MESSAGE TYPE, and 3.4 FETCH MESSAGE. The diagram involves three external entities: Student, Admin, and Lecturer. Students, Admin, and Lecturers send messages to the system, which then processes these messages in several steps.

The 3.1 VERIFY SENDER AND RECEIVER ID process verifies sender messages and receiver ID from any of the three entities. The 3.2 CONSRUCT USER MESSAGE process combines the user ID, sender message, and receiver ID. This data, along with the message

content, is then stored in the data store D3 MESSAGE. The 3.3 3 IDENTIFY MESSAGE TYPE process is responsible for sending the message record by filtering the message type such as, discussion forum or chat, from the data store, ensuring all message data is appropriately sent to the receiver. Lastly, the 3.4 FETCH MESSAGE process forwards the message to the target recipient.

In summary, this Level 1 DFD illustrates the message processing workflow, from verifying sender message by receiver ID, extracting relevant user information and storing the message data, filtering the message type, forwarding it to the recipient. The D3 MESSAGE data store holds the messages and related data throughout this process.

9.2.1 Create Channel



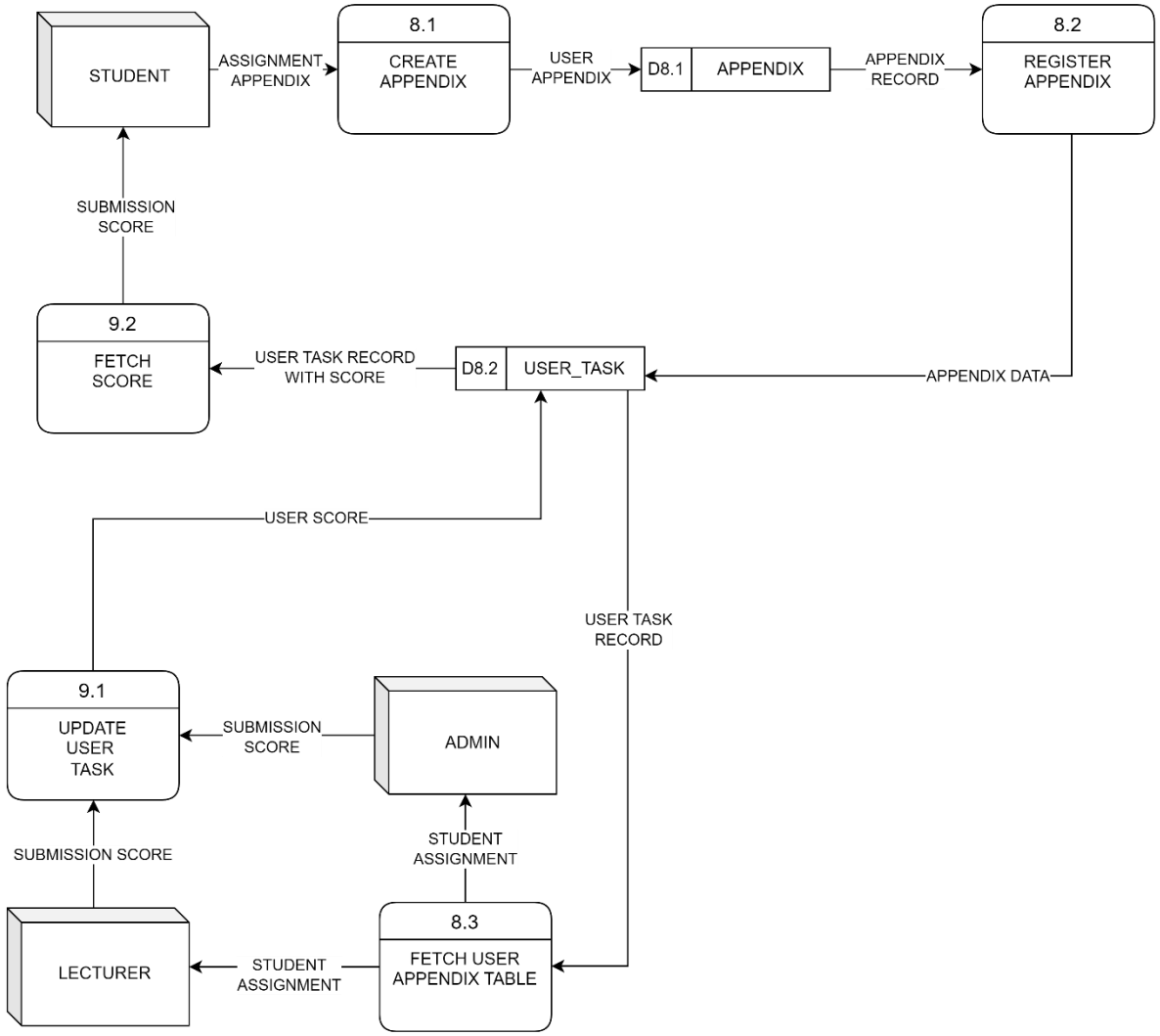
This Data Flow Diagram (DFD) Level 1 contains four processes: 4.1 VERIFY USER ID, 4.2 GENERATE CHANNEL CODE, 4.3 CONTRUCT CHANNEL DATA, and 4.4 FETCH CHANNEL BY ID. The diagram involves two external entities: Lecturer and Student. Lecturers provide channel information to the system, and students receive channel details from the system.

The 4.1 VERIFY USER ID process handles incoming channel information from the Lecturer by verifying user ID. The 4.2 GENERATE CHANNEL CODE process associates the received channel information with the user's ID. The 4.3 CONTRUCT CHANNEL DATA process is responsible for saving the channel record into the data store, ensuring all channel data is appropriately logged. This data, along with the channel code, is then stored in the data store D4 CHANNEL. Finally, the 4.4 FETCH CHANNEL BY ID process provides the channel details to the student.

In summary, this Level 1 DFD illustrates the channel information processing workflow, from receiving channel information, associating it with the user ID creating channel, construct channel data and store into the channel data, providing the channel details to students. The D4 CHANNEL data store holds the channel records and related data throughout this process.

9.3 Data Flow Diagram Level 1 - Tan Po Yeh

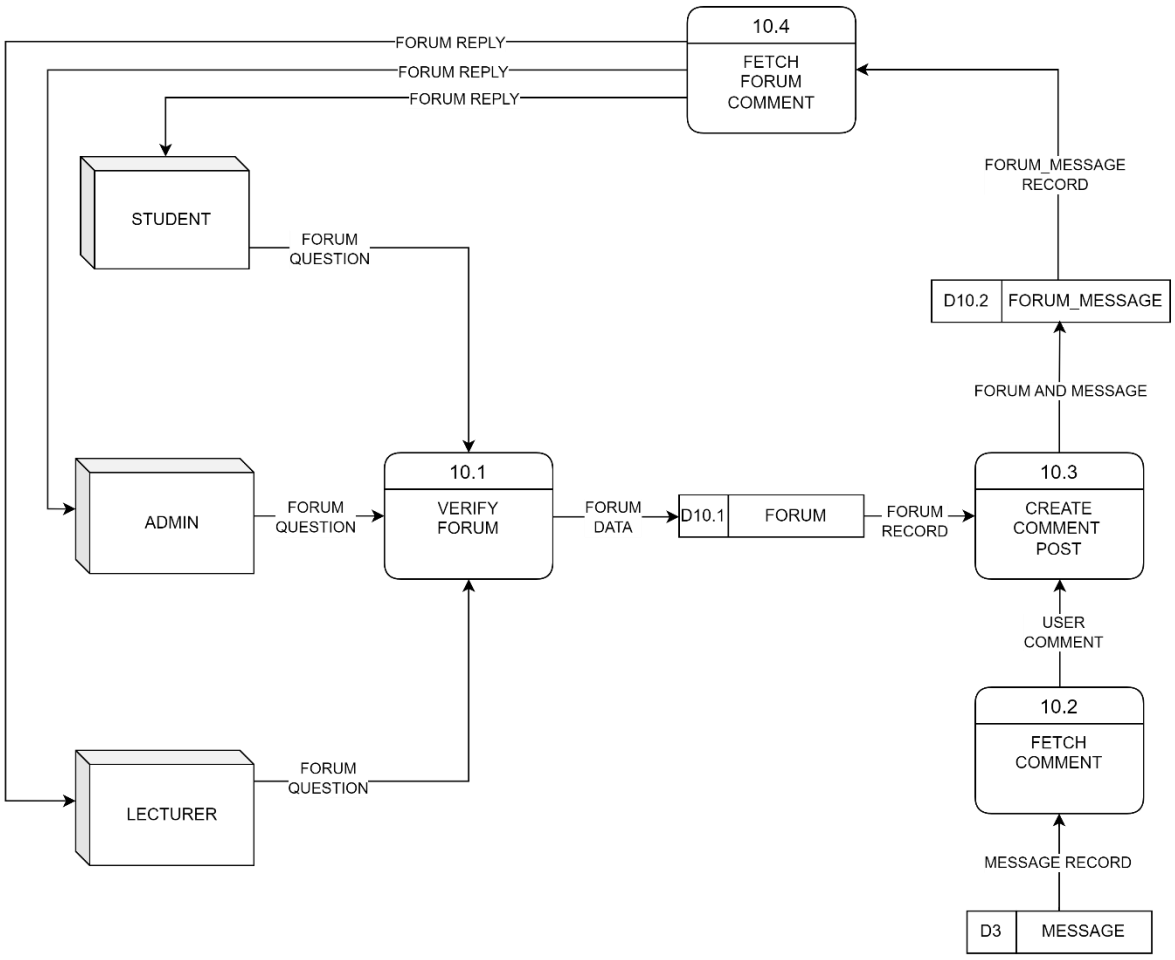
9.3.1 Upload Submission and Mark Submission



This DFD diagram level 1 contains two processes which are the 8.0 UPLOAD SUBMISSION and 9.0 MARK SUBMISSION. The diagram contains 3 external entities, the student uploads their assignment appendix and receives the submission score, lecturer and admin receive the student assignment and give the submission score to the appendix. For the data store, the D8.1 Appendix stores the student appendix while D8.2 User_Task registers the student appendix ID with the task ID and also the score.

In summary, this Level 1 DFD illustrates how the student assignment appendix is stored and how the assignment appendix is obtained and marked by the admin and lecturer. The Appendix data store holds student submissions and the User_Task data store links submissions with tasks and records scores.

9.3.2 Handle Forum

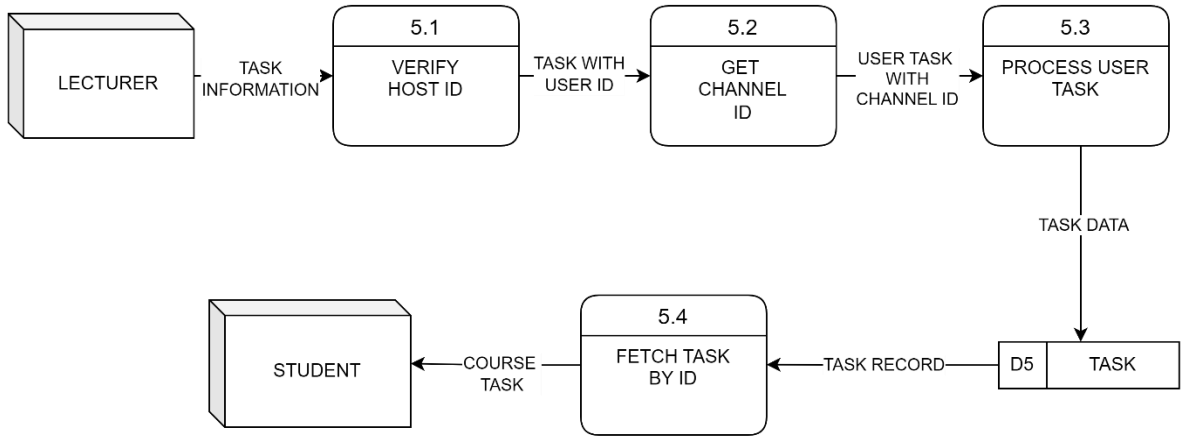


This DFD diagram level 1 contains one process from DFD diagram level 0 which is the 10.0 HANDLE FORUM. The diagram also contains 3 external entities, students, lecturers and admin can create forum posts to ask questions and receive post comments from the system. There are three data stores involved in this section. D3 Message stores the post comments from the users, D10.1 Forum stores the forum data created by the users and D10.2 Forum_Message records the post comment to the corresponding forum record.

In summary, this diagram shows how the users' forum post is stored and how the forum reply is obtained and fetched to the users. The Forum data store holds user forum post data, Message data store saves the comment post and Forum_Message table saves the forum post with the comment post.

9.4 Data Flow Diagram Level 1 - Tong Jia Chuen

9.4.1 Create Task



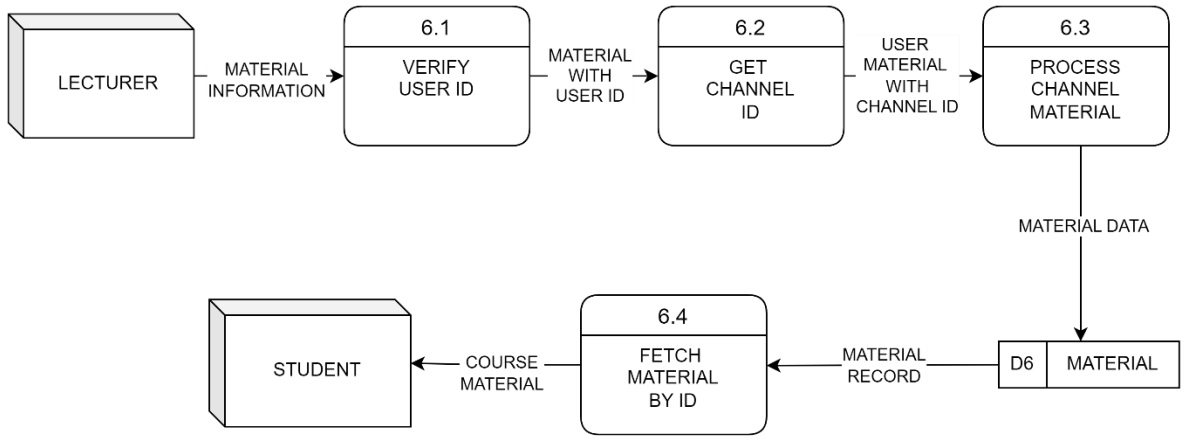
The Data Flow Diagram (DFD) level 1 shows the procedures in managing tasks. The DFD identifies two main external entities: Lecturer and the Student. The lecturer will start the workflow by providing task Information and the students interact with the system to retrieve the task. The Data Flow Diagram (DFD) level 1 has four main processes that play a crucial role in task management system.

Frist of all is **5.1 Verify Host ID**, this process takes the send information by the lecturer as input and outputs with identified user ID. The second process is **5.2 Get Channel ID**. This process receives the task with a user ID and gets the corresponding channel ID. Thirdly, is **5.3 Process User Task** this process is responsible for managing the user task with the channel ID. In addition to that the **5.3 Process User Task** collect Task data will be stored in a data store**, D5 Task,** which plays a role in maintaining the task data.

Finally, the **5.4 Fetch Task by ID** process allows students to retrieve tasks. The students give their course task information, which then serves as an input for this process. After that, the process then proceeds to access the stored task data to fetch the task records as output.

In the nutshell, this DFD level 1 diagram presents a simple view or the overall process of task management system. The diagram itself highlights the process that is involved in verifying, processing and retrieving tasks. The process is started by lecturers who input task information, which is afterwards verified, linked to the correct channel, processed as task data and saved in a data store.

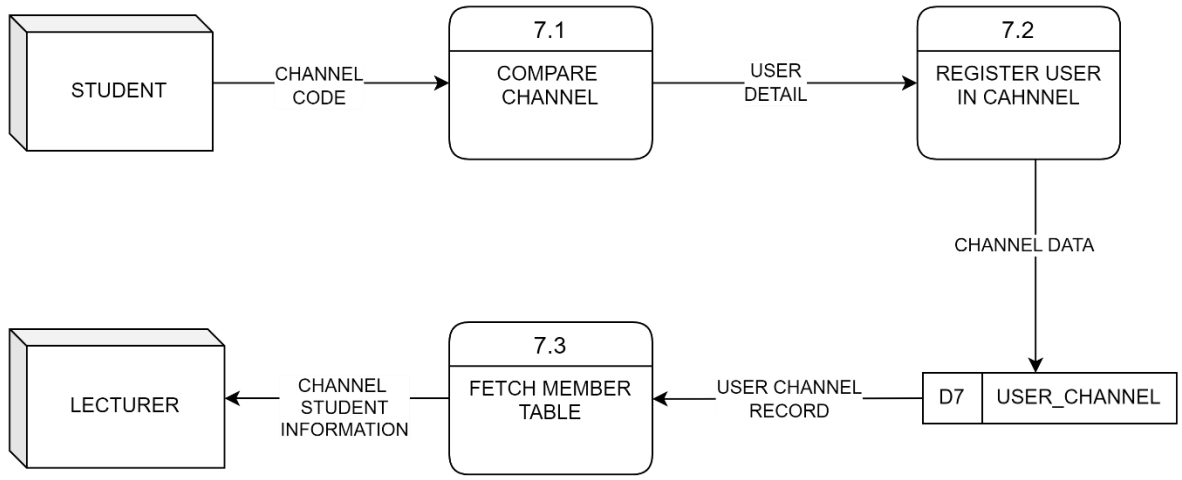
9.4.2 Create Material



The Data Flow Diagram (DFD) level 1 shows "Create Material" process. The DFD identifies two main external entities: Lecturer and the Student The lecturer will start the workflow by providing Material Information and the students interacts with the system to retrieve the material, that is important for their coursework and study . The Data Flow Diagram(DFD) level 1 has four main processes that play a crucial role in task management system .

Frist of all is **6.1 Verify User ID** , this process ensures that the lecturer is verify for further authentication before proceeding to send out material information to the student. The second process is **6.2 Get Channel ID**. This process receives the material with a User Material with Channel ID . Thirdly, is **6.3 Process Channel Material** process takes the User Material with Channel ID and changes it into a Mate. In addition to that ,its collected Material data will be store in a data store**, D6 Material**. Finally ,is the 6.4 Fetch Material by ID process allows students to retrieve educational materials. . The students give their course material information, which is used to fetch the corresponding material record

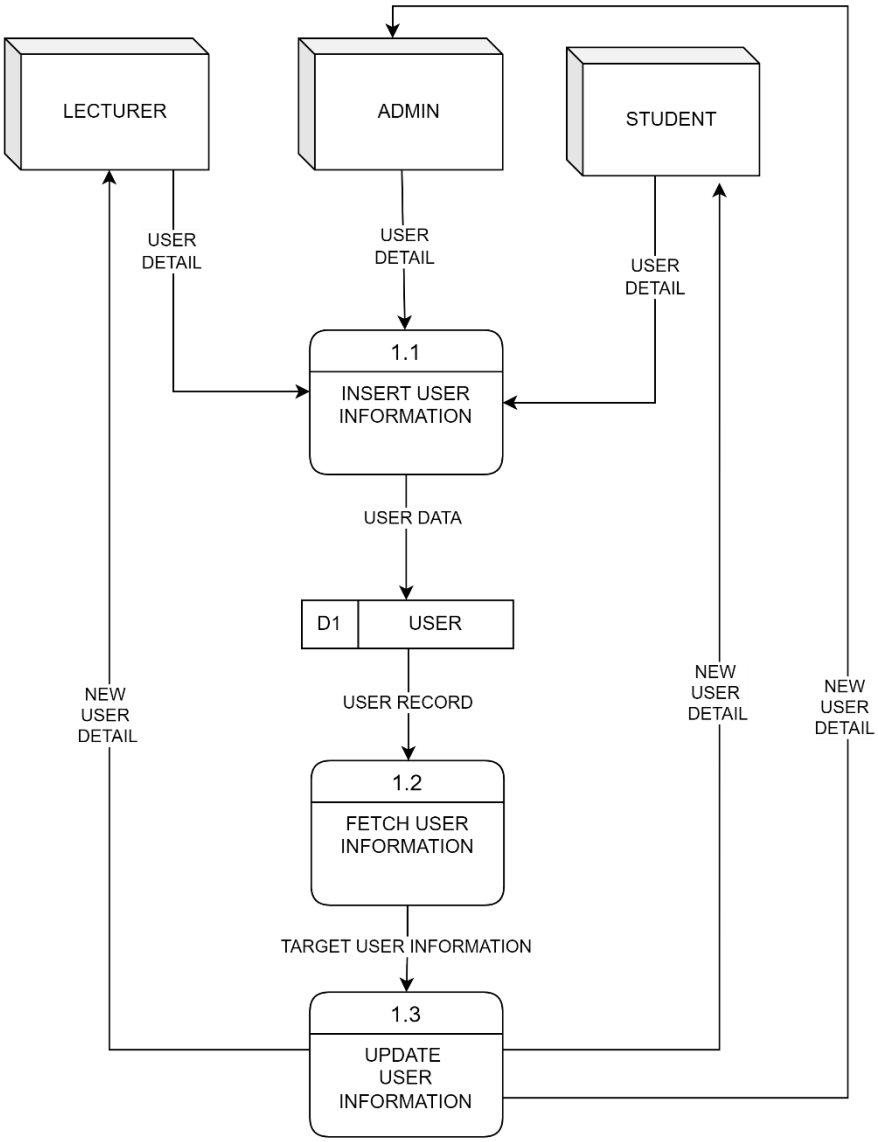
9.4.3 Join Channel



10.0 Data Dictionary

10.1 Data Dictionary – Chok Jung Shing

10.1.1 Manage Their Own Account



External Entity
Name: Student **Description:** Student enters new details **Input Data Flows:** New user detail **Output Data Flows:** user detail
Name: Admin **Description:** Admin sends new user detail

Input Data Flows: New user detail

Output Data Flows: user detail

Name: Lecturer

Description: Lecturer enters new details

Input Data Flows: New user detail

Output Data Flows: user detail

Data Process

Name: 1.1 INSERT USER INFORMATION

Description: Insert new user information from a user of (Student Lecturer)

Input Data Flows: New user detail, New user detail, New user detail

Output Data Flows: User's Data

Process:

```
<?php
   include "dbConn.php";
   global $connection;

   $userID = $_SESSION['userid'];
   $userType = $_SESSION['usertype'];

   // Use a prepared statement to prevent SQL injection
   $stmt = $connection->prepare("SELECT * FROM `User`u INNER JOIN `Image`i ON
u.`ImageID` = i.`ImageID` WHERE u.`UserID` = ?");
   $stmt->bind_param("s", $userID);
   $stmt->execute();
   $result = $stmt->get_result();

   // Check if the query was successful
   if ($result) {
      $row = $result->fetch_assoc();
      if ($row) {
         $userID = htmlspecialchars($row['UserID'], ENT_QUOTES, 'UTF-8');
         $userName = htmlspecialchars($row['Name'], ENT_QUOTES, 'UTF-8');
         $userIC = htmlspecialchars($row['IC'], ENT_QUOTES, 'UTF-8');
```

```
        $password = htmlspecialchars($row['Password'], ENT_QUOTES, 'UTF-8');
        $studEmail = htmlspecialchars($row['Email'], ENT_QUOTES, 'UTF-8');
        $userAge = htmlspecialchars($row['Age'], ENT_QUOTES, 'UTF-8');
        $gender = htmlspecialchars($row['Gender'], ENT_QUOTES, 'UTF-8');
        $imagePath = htmlspecialchars($row['Filepath'], ENT_QUOTES, 'UTF-8');
        $imageID = htmlspecialchars($row['ImageID'], ENT_QUOTES, 'UTF-8');

        echo '
        <h1>Settings</h1>
        <div class="setting1">
          <div class="setting2">
            <p>Name</p>
            <input type="text" name="userName" placeholder="' . $userName .
'"></input>
          </div>
          <div class="setting2">
            <p>IC Number</p>
            <input type="text" name="userIC" placeholder="' . $userIC . '"></input>
          </div>
          <div class="setting2">
            <p>Password</p>
            <input type="text" name="password" placeholder="' . $password . '"></input>
          </div>
          <div class="setting2">
            <p>Email</p>
            <input type="text" name="studEmail" placeholder="' . $studEmail .
'"></input>
          </div>
          <div class="setting2">
            <p>Age</p>
            <input type="text" name="userAge" placeholder="' . $userAge . '"></input>
          </div>
          <div class="setting2">
            <p>Gender</p>
```

```
                <select id="gender" name="gender">
                    <option value="" placeholder="' . $gender . '" disabled selected>Gender: ' .
$gender . '</option>
                    <option value="male">Male</option>
                    <option value="female">Female</option>
                    <option value="non-binary">Non-binary</option>
                    <option value="prefer-not-to-say">Prefer not to say</option>
                </select>
            </div>
            <div class="setting3">
                <p>Image</p>
                <img src="'.$imagePath.'" style="width: 100px; height: auto; border-radius:
10px; margin-bottom: 3%"></img>
                <input type="hidden" class="hidden_input" name="imageID"
value="'.$imageID.'"></input>
                <input type="file" name="imagePath"></input>
            </div>
        </div>
            <input type="submit" name="btnUpdate" value="Update">
        ';
    } else {
        echo "No user found with the given ID.";
    }
} else {
    echo "Error executing query: " . $connection->error;
}

$stmt->close();
$connection->close();
?>
```

Name: 1.2 FETCH USER INFORMATION

Description: verify sender message from a user of (Student, Admin, Lecturer) and receiver.

Input Data Flows: User Record

Output Data Flows: Target User's Information

Process:

```
FUNCTION fetch student information ()
INCLUDE "dbConn.php";
   global $connection;

   SET userID = $_SESSION["userid"];

   SET fetchQuery1 = "SELECT * FROM `User` WHERE `UserID` = '$userID'";
   SET result1 = mysqli_query($connection, $fetchQuery1);
   IF ($result1 && $row = mysqli_fetch_assoc($result1)) {
      SETuserImageID = htmlspecialchars($row['ImageID'], ENT_QUOTES, 'UTF-8');
   }

   SET fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = '$userImageID'";
   SET result2 = mysqli_query($connection, $fetchQuery2);
   IF ($result2 && $row = mysqli_fetch_assoc($result2)) {
      SET userImagePath = htmlspecialchars($row['Filepath'], ENT_QUOTES, 'UTF-8');
   }
END FUNCTION

FUNCTION fetch lecturer information ()
INCLUDE "dbConn.php";
   global $connection;

   SET userID = $_SESSION["userid"];

   SET fetchQuery1 = "SELECT * FROM `User` WHERE `UserID` = '$userID'";
   SET result1 = mysqli_query($connection, $fetchQuery1);
   IF ($result1 && $row = mysqli_fetch_assoc($result1)) {
```

```
        SET userImageID = htmlspecialchars($row['ImageID'], ENT_QUOTES, 'UTF-8');
    }

    SET fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = '$userImageID'";
    SET result2 = mysqli_query($connection, $fetchQuery2);
    IF ($result2 && $row = mysqli_fetch_assoc($result2)) {
        SET userImagePath = htmlspecialchars($row['Filepath'], ENT_QUOTES, 'UTF-8');
    }
END FUNCTION

FUNCTION fetch admin information ()
INCLUDE "dbConn.php";
    global $connection;

    SET userID = $_SESSION["userid"];

    SET fetchQuery1 = "SELECT * FROM `User` WHERE `UserID` = '$userID'";
    SET result1 = mysqli_query($connection, $fetchQuery1);
    IF ($result1 && $row = mysqli_fetch_assoc($result1)) {
        SET userImageID = htmlspecialchars($row['ImageID'], ENT_QUOTES, 'UTF-8');
    }

    SET fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = '$userImageID'";
    SET result2 = mysqli_query($connection, $fetchQuery2);
    IF ($result2 && $row = mysqli_fetch_assoc($result2)) {
        SET userImagePath = htmlspecialchars($row['Filepath'], ENT_QUOTES, 'UTF-8');
    }
END FUNCTION
```

Name: 1.3 UPDATE USER INFORMATION

Description: verify sender message from a user of (Student, Admin, Lecturer) and receiver.

Input Data Flows: Target User Information

Output Data Flows: New User Detail

Process:

```
SESSION_START();
REQUIRE_ONCE(__DIR__ . '/../dbConn.php');
global $connection;

IF (isset($_POST['btnUpdate'])) {
   SET userID = $_SESSION['userid'];

   SET updateFields = [];
   SET params = [];
   SET types = "";

   IF (!empty($_POST['userName'])) {
      SET updateFields[] = "Name = ?";
      SET params[] = $_POST['userName'];
      SET types .= "s";
   }
   IF (!empty($_POST['userIC'])) {
      SET updateFields[] = "IC = ?";
      SET params[] = $_POST['userIC'];
      SET types .= "s";
   }
   IF (!empty($_POST['password'])) {
      SET updateFields[] = "Password = ?";
      SET params[] = $_POST['password'];
      SET types .= "s";
   }
   IF (!empty($_POST['studEmail'])) {
      SET updateFields[] = "Email = ?";
```

```
    SET params[] = $_POST['studEmail'];
    SET types .= "s";
}
IF (!empty($_POST['userAge'])) {
    SET updateFields[] = "Age = ?";
    SET params[] = $_POST['userAge'];
    SET types .= "i";
}
IF (!empty($_POST['gender'])) {
    SET updateFields[] = "Gender = ?";
    SET params[] = $_POST['gender'];
    SET types .= "s";
}

// Handle image upload
IF (!empty($_FILES['imagePath']['name']) && !empty($_POST['imageID'])) {
    SET imageID = $_POST['imageID'];
    SET file = $_FILES['imagePath'];

    // Check if file is uploaded successfully
    IF ($file["error"] == UPLOAD_ERR_OK) {
        SET allowedFileTypes = ['image/jpeg', 'image/png', 'image/gif'];
        SET maxFileSize = 2 * 1024 * 1024; // 2 MB

        // Get file information
        SET filename = basename($file["name"]);
        SET tempname = $file["tmp_name"];
        SET filetype = $file["type"];
        SET filesize = $file["size"];

        // Check if the file type is allowed
        IF (!in_array($filetype, $allowedFileTypes)) {
            ECHO '<script>';
            ECHO 'window.alert("Error: Only JPG, PNG, and GIF files are allowed.");';
```

```
            ECHO 'window.location.href = "/projectify/setting.php";';
            ECHO '</script>';
            EXIT;
        }

        // Check if the file size is within the limit
        IF ($filesize > $maxFileSize) {
            ECHO '<script>';
            ECHO 'window.alert("Error: File size exceeds the 2 MB limit.");';
            ECHO 'window.location.href = "/projectify/setting.php";';
            ECHO '</script>';
            exit;
        }

        // Ensure the upload directory exists
        SET upload_dir = __DIR__ . "/../userdata/";
        IF (!is_dir($upload_dir)) {
            mkdir($upload_dir, 0755, true);
        }

        // Generate a unique name for the file to avoid overwriting
        SET uniqueFilename = uniqid() . "_" . $filename;
        SET folder = $upload_dir . $uniqueFilename;
        SET formattedfilename = "userdata/" . $uniqueFilename;

        IF (move_uploaded_file($tempname, $folder)) {
            // Update image table
            SET updateImageQuery = "UPDATE `Image` SET `Filename` = ?, `Filepath` = ?
WHERE `ImageID` = ?";
            SET stmtImage = $connection->prepare($updateImageQuery);
            SET stmtImage->bind_param("ssi", $filename, $formattedfilename, $imageID);
            SET stmtImage->execute();

           IF ($stmtImage->affected_rows > 0) {
```

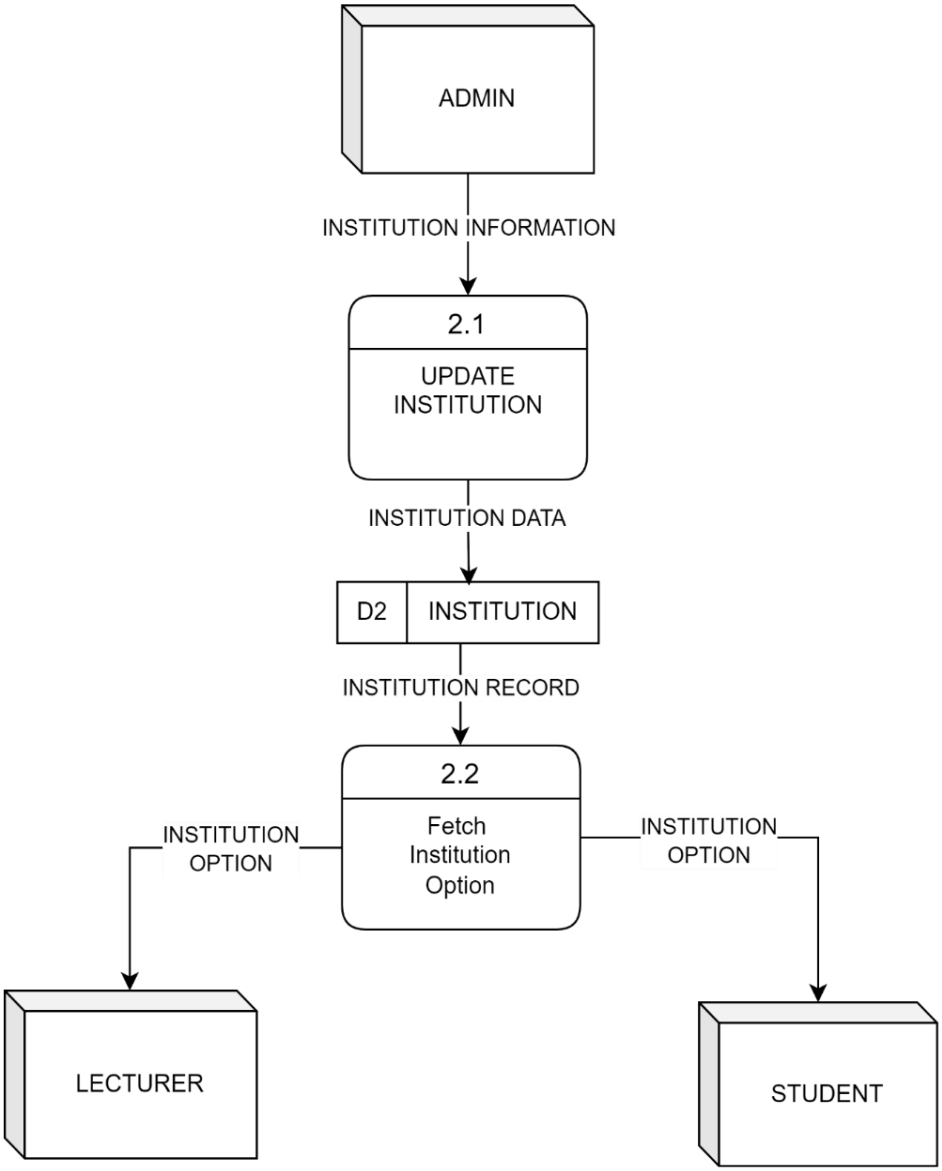
```
            SET updateFields[] = "ImageID = ?";
            SET params[] = $imageID;
            SET types .= "i";
          } ELSE{
            ECHO '<script>';
            ECHO 'window.alert("Error updating image data in the database.");';
            ECHO 'window.location.href = "/projectify/setting.php";';
            ECHO '</script>';
            EXIT;
          }

          $stmtImage->close();
        } ELSE {
          ECHO '<script>';
          ECHO 'window.alert("Error moving uploaded file.");';
          ECHO 'window.location.href = "/projectify/setting.php";';
          ECHO '</script>';
          EXIT;
        }
      } ELSE {
        ECHO '<script>';
        ECHO 'window.alert("Error uploading file.");';
        ECHO 'window.location.href = "/projectify/setting.php";';
        ECHO '</script>';
        EXIT;
      }
    }
```

Data Store
Name: D1 USER
Description: Store new user information
Input Data Flows: User data
Output Data Flows: User Record

Data Structure: User ID, Name, IC, Password, Email, Age, Email, Age, Gender, Image, ImageID

Data Flow
Name: INSERTING NEW USER INFORMATION **Description:** Insert user information **Origin:** Student, Admin, Lecturer external entities **Destination:** 1.1 INSERT USER INFORMATION **Data Structure:** User ID, Name, IC, Password, Email, Age, Email, Age, Gender, Image. ImageID
Name: STORING NEW USER INFORMATION **Description:** storing user information **Origin:** 1.1 INSERT USER INFORMATION **Destination:** D1 USER **Data Structure:** User ID, Name, IC, Password, Email, Age, Email, Age, Gender, Image. ImageID
Name: FETCHING NEW USER INFORMATION **Description:** fetching user information **Origin:** D1 USER **Destination:** 1.2 FETCH USER INFORMATION **Data Structure:** User ID, Name, IC, Password, Email, Age, Email, Age, Gender, Image. ImageID
Name: UPDATE USER INFORMATION **Description:** updating targeted user information **Origin:** 1.2 FETCH USER INFORMATION **Destination:** 1.3 UPDATE USER INFORMATION **Data Structure:** User ID, Name, IC, Password, Email, Age, Email, Age, Gender, Image, ImageID
Name: UPDATED USER INFORMATION **Description:** n**ew** user information **Origin:** 1.3 UPDATE USER INFORMATION **Destination:** Student, Admin, Lecturer external entities **Data Structure:** User ID, Name, IC, Password, Email, Age, Email, Age, Gender, Image, ImageID

10.1.2 Update Institution



External Entity
Name: Admin
Description: Admin sends new institution information
Input Data Flows: Information about institution
Output Data Flows: institution information

Name: Student

Description: Student receive institution option.

Input Data Flows: Institution option

Output Data Flows: None

Name: Lecturer

Description: Lecturer receive institution option.

Input Data Flows: Institution option

Output Data Flows: None

Data Process

Name: 2.1 UPDATE INSTITUTION

Description: Updates the Institution with information to the database

Input Data Flows: Institution information

Output Data Flows: Institution Data

Process:

```
   REQUIRE_ONCE(__DIR__ . '/../dbConn.php');
   global $connection;

   IF (isset($_POST['name']) && isset($_POST['address']) && isset($_POST['location']))
{
      SET name = mysqli_real_escape_string($connection, $_POST['name']);
      SET address = mysqli_real_escape_string($connection, $_POST['address']);
      SET location = mysqli_real_escape_string($connection, $_POST['location']);

      IF (!empty($name) && !empty($address) && !empty($location)) {
         SET query = "INSERT INTO `institution`(`Name`, `Address`, `Location`)
VALUES ('$name','$address','$location')";

         IF ($result = mysqli_query($connection, $query)) {
            ECHO '<script>';
            ECHO 'window.alert("Upload Successful!");';
            ECHO 'window.location.href = "/projectify/institution.php";';
            ECHO '</script>';
         } ELSE {
```

```
            ECHO '<script>';
            ECHO 'window.alert("Error uploading the record.");';
            ECHO 'window.location.href = "/projectify/institution.php";';
            ECHO '</script>';
        }
    } ELSE {
        ECHO '<script>';
        ECHO 'window.alert("Error: ID must not be empty.");';
        ECHO 'window.location.href = "/projectify/institution.php";';
        ECHO '</script>';
    }
} ELSE {
    ECHO '<script>';
    ECHO 'window.alert("Error: Required data not received.");';
    ECHO 'window.location.href = "/projectify/institution.php";';
    ECHO '</script>';
}
```

Name: 2.2 FETCH INSTITUION OPTION

Description: Retrieve the Institution data from the updated database

Input Data Flows: Institution record

Output Data Flows: Institution option

Process:

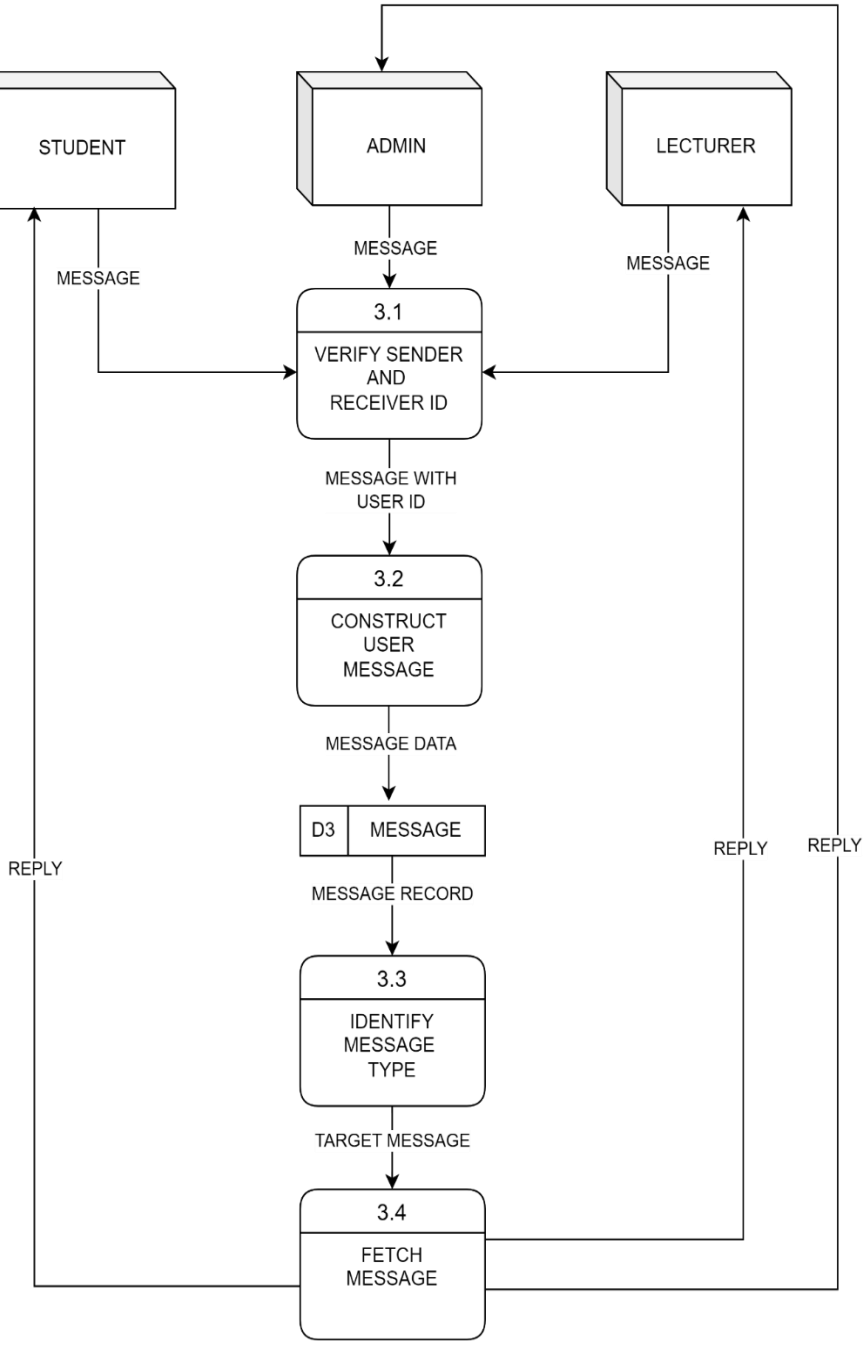
```
SET fetchQuery = "
  SELECT *
  FROM Institution
  ORDER BY InstitutionID ASC;
  ";

  SET result = mysqli_query($connection, $fetchQuery);
```

Data Store
Name: D2 INSTITUTION **Description:** Store institution data. **Input Data Flows:** Institution Data

Output Data Flows: Institution Record **Data Structure:** Institution Name, Institution Address, Institution Location
Data Flow
Name: ADDING INSTITUTION INFORMATION **Description:** Adding new institution information **Origin:** Admin external entities **Destination:** 2.1 UPDATE INSTITUTION **Data Structure:** Institution Name, Institution Address, Institution Location
Name: STORING INSTITUTION INFORMATION **Description:** storing new institution information **Origin:** 2.1 UPDATE INSTITUTION **Destination:** D2 INSTITUTION **Data Structure:** Institution Name, Institution Address, Institution Location
Name: GIVING INSTITUTION INFORMATION OPTION **Description:** giving new institution information option **Origin:** D2 INSTITUTION **Destination:** 2.2 FETCH INSTITUTION OPTION **Data Structure:** Institution Name, Institution Address, Institution Location
Name: INSTITUTION INFORMATION OPTION **Description:** institution information option **Origin:** 2.2 FETCH INSTITUTION OPTION **Destination:** Students and Lecturer external entities **Data Structure:** Institution Name, Institution Address, Institution Location

10.2 Data Dictionary – Leong Seng Khuan

10.2.1 Send Message



External Entity
Name: Student **Description:** Student sends a message. **Input Data Flows:** Reply **Output Data Flows:** Message

Name: Admin

Description: Admin sends a message.

Input Data Flows: Reply

Output Data Flows: Message

Name: Lecturer

Description: Lecturer sends a message.

Input Data Flows: Reply

Output Data Flows: Message

Data Process

Name: 3.1 VERIFY SENDER AND RECEIVER ID

Description: verify sender message from a user of (Student, Admin, Lecturer) and receiver.

Input Data Flows: Message

Output Data Flows: Message including the sender's ID, receiver's ID, and message content.

Process:

```
// Handle form submission
        SET ('#userForm').submit(function(event) {
            event.preventDefault();

            VAR formData = $(this).serialize();

            $.ajax({
                url: 'php/sendMessage.php',
                method: 'POST',
                data: formData,
                success: function(response) {
                    console.log(response);

                    // Clear the input field after successful submission
                    SET ('input[name="message"]').val('');

                    // Refresh the message container
```

```
                VAR selectedUserID = $('#messageUser').val();
                FETCHMESSAGES(selectedUserID);
            },
            error: function(xhr, status, error) {
                console.error(error);
            }
        });
    });

    // Handle change event for messageUser select element
    SET ('#messageUser').change(function() {
        VAR selectedImage = $(this).find('option:selected').data('image');
        SET ('#userImage').attr('src', selectedImage);

        VAR selectedUserID = $(this).val();
        FETCHMESSAGES(selectedUserID);
    });
// Fetch messages function
    FUNCTION fetchMessages(selectedUserID) {
        $.ajax({
            url: 'php/fetchMessage.php',
            method: 'POST',
            data: { selectedUserID: selectedUserID },
            success: function(response) {
                $('#messageContainer').html(response);
            },
            error: function(xhr, status, error) {
                console.error(error);
            }
        });
    }
});
```

Name: 3.2 CONSTRUCT USER MESSAGE

Description: Identify the user based on user ID from the message.

Input Data Flows: Message with User ID

Output Data Flows: Message Data

Process:

```
CONSTRUCT USER MESSAGE
   SESSION_START();
   REQUIRE_ONCE(__DIR__ . '/../dbConn.php');
   global $connection;

   IF(isset($_POST['messageUser'])) {
      SET currentUserID = $_SESSION['userid'];
      SET receiverID = isset($_POST['messageUser']) ?
mysqli_real_escape_string($connection, $_POST['messageUser']) : "";
      SET messageContent = isset($_POST['message']) ?
mysqli_real_escape_string($connection, $_POST['message']) : "";

      // Check if receiver exists
      SET checkReceiverQuery = "SELECT * FROM `User` WHERE `UserID` =
'$receiverID'";
      SET receiverResult = mysqli_query($connection, $checkReceiverQuery);

      IF (mysqli_num_rows($receiverResult) > 0) {
         // Insert message
         SET insertMessageQuery = "
            INSERT INTO Message (Content, Timestamp, SenderID, ReceiverID)
            VALUES ('$messageContent', NOW(), '$currentUserID', '$receiverID')
         ";

         IF (!mysqli_query($connection, $insertMessageQuery)) {
            ECHO "Error: " . $insertMessageQuery . "<br>" . mysqli_error($connection);
         } ELSE {
            ECHO "Message sent successfully.";
         }
      } ELSE {
         ECHO "Error: Receiver does not exist.";
```

```
    }
  }
?>
```

Name: 3.3 IDENTITY MESSAGE TYPE

Description: Identity the message type and store the message data into the database.

Input Data Flows: Message Data

Output Data Flows: Message Record

Process:

IDENTIFY MESSAGE TYPE

```
// Initialize a variable to store the HTML
    SET messagesHTML = "";
WHILE ($row = mysqli_fetch_assoc($result)) {
    SET messageContent = htmlspecialchars($row['Content'], ENT_QUOTES, 'UTF-
8');
    SET timestamp = htmlspecialchars($row['Timestamp'], ENT_QUOTES, 'UTF-8');
    SET senderID = $row['SenderID'];

    // Check if the message is outgoing or incoming
    IF ($senderID == $currentUserID) {
      // Outgoing message
      SET messageHTML = '
      <div class="outgoing-chats">
        <div class="outgoing-msg">
          <div class="outgoing-chats-msg">
            <p class="multi-msg">' . $messageContent . '</p>
            <span class="time">' . $timestamp . '</span>
          </div>
        </div>
      </div>';
    } ELSEIF ($senderID == $chatPartnerID) {
      // Incoming message
      SET messageHTML = '
      <div class="received-chats">
        <div class="received-msg">
```

```
                <div class="received-msg-inbox">
                  <p>' . $messageContent . '</p>
                  <span class="time">' . $timestamp . '</span>
                </div>
              </div>
            </div>';
        }

        // Append the constructed message HTML to the messagesHTML variable
        SET messagesHTML .= $messageHTML;
    }

    // Echo the messages HTML
    ECHO $messagesHTML;
  } ELSE {
    ECHO "Session variables not set.";
  }
?>      'messages': messages
```

Name: 3.4 FETCH MESSAGE

Description: fetch message and reply to the user.

Input Data Flows: Target Message

Output Data Flows: Reply

Process:

```
<?php
  SESSION_START();
  REQUIRE_ONCE(__DIR__ . '/../dbConn.php');
  global $connection;

  // Check if session variables are set
  IF (isset($_SESSION['userid']) && isset($_POST['selectedUserID'])) {
    SET currentUserID = $_SESSION['userid'];
    SET chatPartnerID = $_POST['selectedUserID'];
```

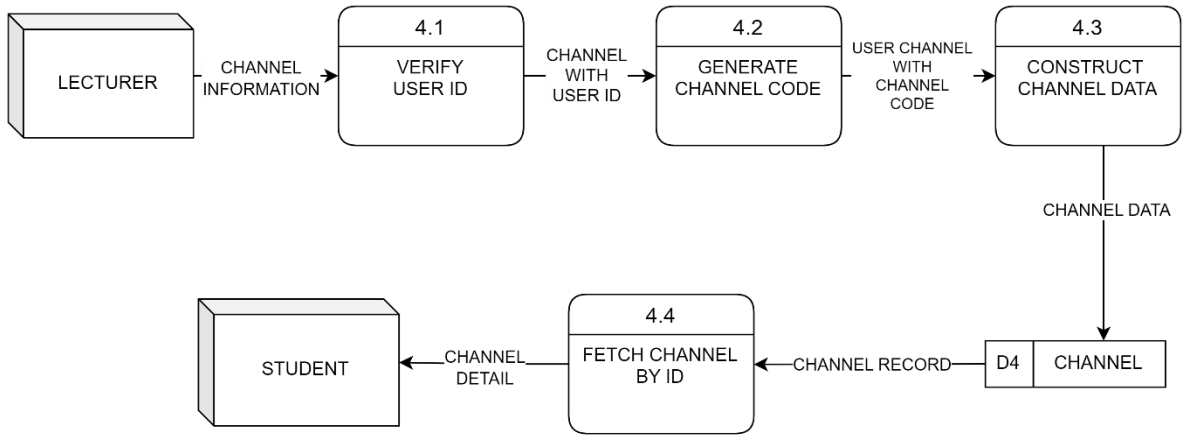
```
    // Prepare the query with placeholders
    SET fetchMessagesQuery = "
        SELECT *
        FROM Message
        WHERE (SenderID = ? AND ReceiverID = ?)
        OR (SenderID = ? AND ReceiverID = ?)
        ORDER BY Timestamp ASC
    ";

    // Prepare and bind the statement
    SET statement = mysqli_prepare($connection, $fetchMessagesQuery);
    mysqli_stmt_bind_param($statement,
                           "iiii",
                           $currentUserID,
                           $chatPartnerID,
                           $chatPartnerID,
                           $currentUserID
                           );
    // Execute the statement
    mysqli_stmt_execute($statement);

    // Get the result
    SET result = mysqli_stmt_get_result($statement);

  }

  RETURN reply
END FETCH MESSAGE
```

Data Store
Name: D3 MESSAGE **Description:** Store messages with user data. **Input Data Flows:** Message Data

Output Data Flows: Message Record **Data Structure:** Content, Timestamp, SenderID, ReceiverID

Data Flow
Name: MESSAGE **Description:** Message content sent by the user. **Origin:** Student, Admin, Lecturer external entities **Destination:** 3.1 RECEIVE MESSAGE **Data Structure:** MessageContent, Timestamp
Name: MESSAGE WITH USER ID **Description:** Message content with user ID appended. **Origin:** 3.1 RECEIVE MESSAGE **Destination:** 3.2 IDENTIFY USER **Data Structure:** UserID, MessageContent, Timestamp
Name: MESSAGE DATA **Description:** Detailed message data. **Origin:** 3.2 IDENTIFY USER **Destination:** D3 MESSAGE **Data Structure:** MessageID, UserID, MessageContent, Timestamp
Name: MESSAGE RECORD **Description:** Stored message record. **Origin:** D3 MESSAGE **Destination:** 3.3 STORE MESSAGE **Data Structure:** MessageID, UserID, MessageContent, Timestamp
Name: TARGET MESSAGE **Description:** Specific message to be replied to. **Origin:** 3.3 STORE MESSAGE **Destination:** 3.4 PROCESS REPLY **Data Structure:** MessageID, UserID, MessageContent, Timestamp
Name: REPLY **Description:** Reply to the user's message. **Origin:** 3.4 PROCESS REPLY **Destination:** Student, Admin, Lecturer external entities **Data Structure:** MessageID, ReplyContent, Timestamp

10.2.2 Create Channel



External Entity
Name: Lecturer **Description:** Provides channel information. **Input Data Flows:** Information **Output Data Flows:** Channel Information
Name: Student **Description:** Requests channel details. **Input Data Flows:** Channel Detail **Output Data Flows:** None

Data Process

Name: 4.1 VERIFY USER ID
Description: Verifies the user ID of the lecturer.1
Input data flows: Channel Information
Output data flows: Channel with User ID
Process:
// Include database connection
INCLUDE "dbConn.php";
global $connection;
// Initialize variables
SET channelName = isset($_POST['txttitle']) ? mysqli_real_escape_string($connection, $_POST['txttitle']) : "";

```
SET channelDescription = isset($_POST['txtdescription']) ?
mysqli_real_escape_string($connection, $_POST['txtdescription']) : "";
SET userID = $_SESSION["userid"];
SET createChannel = "display: none";
```

Name: 4.2 GENERATE CHANNEL CODE

Description: Generates a unique channel code for the user.

Input data flows: Channel with User ID

Output data flows: User Channel with Channel Code

Process:

```
// Check if the 'Create Channel' button is pressed
IF (isset($_POST['btnCreateChannel'])) {
   SET createChannel = "";
} ELSEIF (isset($_POST['btnCreateChannel2'])) {
   // Generate a unique channel code
   SET channelCode = getUniqueChannelCode($connection);
      SET totalOfMember = $row['total'];
   }
}
```

Name: 4.3 CONSTRUCT CHANNEL DATA

Description: Constructs the complete channel data

Input data flows: User Channel with Channel Code

Output data flows: Channel Data

Process:

```
// Insert new channel into the database
   SET insertQuery1 = "INSERT INTO `channel`(`ChannelCode`, `Title`, `Description`,
`OwnerID`) VALUES ('$channelCode', '$channelName','$channelDescription', '$userID')";
   IF (mysqli_query($connection, $insertQuery1)) {
      SET channelID = mysqli_insert_id($connection);
      SET tresult = saveImage($_FILES["channelprofile"], 'ChanID', $channelID,
$connection, 'Channel');
      IF ($result === "Image updated successfully.") {
         echo '<script>window.alert("Create Successful!"); window.location.href =
"/projectify/lecturerchannel.php";</script>';
      } ELSE {
```

```
        ECHO '<script>window.alert("Error: ' . $result . '");</script>';
      }
   } ELSE {
      ECHO '<script>window.alert("Error inserting channel: ' . mysqli_error($connection) .
'");</script>';
   }
   SET createChannel = "display: none";
}
```

Name: 4.4 FETCH CHANNEL BY ID

Description: Fetches channel details by ID for the student.

Input data flows: Channel Record

Output data flows: Channel Detail

Process:

```
// Include database connection
INCLUDE "dbConn.php";
global $connection, $channelID, $channelName, $imagePath;

// Fetch channel details
SET fetchQuery1 = "SELECT * FROM `Channel` WHERE `ChanID` = '$channelID'";
SET result1 = mysqli_query($connection, $fetchQuery1);
IF ($result1 && $row = mysqli_fetch_assoc($result1)) {
   SET channelID = htmlspecialchars($row['ChanID'], ENT_QUOTES, 'UTF-8');
   SET channelCode = htmlspecialchars($row['ChannelCode'], ENT_QUOTES, 'UTF-8');
   SET channelName = htmlspecialchars($row['Title'], ENT_QUOTES, 'UTF-8');
   SET imageID = htmlspecialchars($row['ImageID'], ENT_QUOTES, 'UTF-8');
   SET ownerID = htmlspecialchars($row['OwnerID'], ENT_QUOTES, 'UTF-8');

   SET fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = '$imageID'";
   SET result2 = mysqli_query($connection, $fetchQuery2);
   IF ($result2 && $row = mysqli_fetch_assoc($result2)) {
      SET imagePath = htmlspecialchars($row['Filepath'], ENT_QUOTES, 'UTF-8');
   }

   SET fetchQuery3 = "SELECT * FROM `User` WHERE `UserID` = '$ownerID'";
```

```
SET result3 = mysqli_query($connection, $fetchQuery3);
IF ($result3 && $row = mysqli_fetch_assoc($result3)) {
    SET userName = htmlspecialchars($row['Name'], ENT_QUOTES, 'UTF-8');
}
SET query = "SELECT COUNT(*) as total FROM `User_Channel` WHERE `ChanID`
= '$channelID'";
SET result = mysqli_query($connection, $query);
IF ($result) {
    SET row = mysqli_fetch_assoc($result);
```

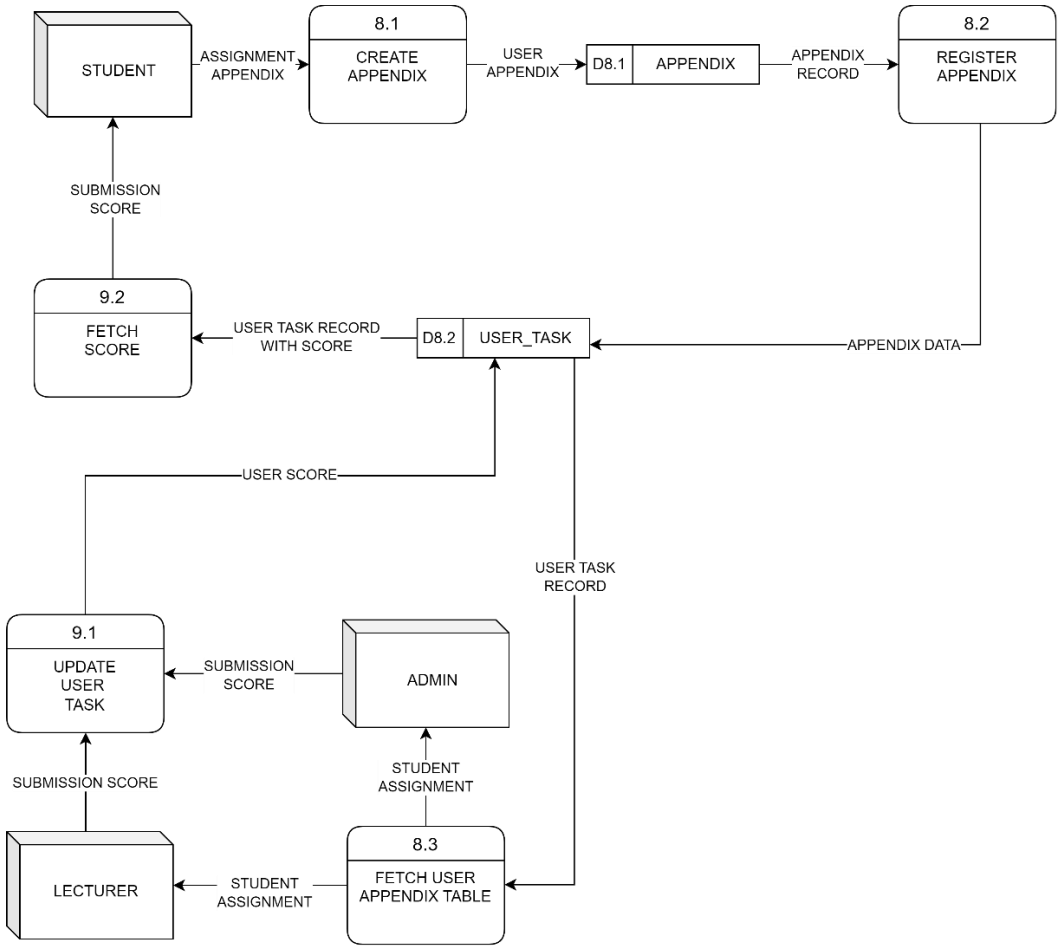
Data Store
Name: D4 CHANNEL Description: Stores channel data. Input data flows: Channel Record Output data flows: Channel Detail Data Structure: ChannelID, UserID, ChannelInfo, ChannelCode

Data Flow
Name: CHANNEL INFORMATION **Description:** Information about the channel provided by the lecturer. **Origin:** Lecturer **Destination:** 4.1 VERIFY USER ID **Data Structure:** ChannelInfo
Name: CHANNEL WITH USER ID **Description:** Channel information with appended user ID. **Origin:** 4.1 VERIFY USER ID **Destination:** 4.2 GENERATE CHANNEL CODE **Data Structure:** ChannelInfo, UserID
Name: USER CHANNEL WITH CHANNEL CODE **Description:** User channel information with a unique channel code. **Origin:** 4.2 GENERATE CHANNEL CODE **Destination:** 4.3 CONSTRUCT CHANNEL DATA **Data Structure:** ChannelInfo, UserID , ChannelCode

Name: CHANNEL DATA **Description:** Complete channel data. **Origin:** 4.3 CONSTRUCT CHANNEL DATA **Destination:** D4 CHANNEL **Data Structure:** ChannelID, UserID, ChannelInfo, ChannelCode
Name: CHANNEL RECORD **Description:** Stored channel record. **Origin:** D4 CHANNEL **Destination:** 4.4 FETCH CHANNEL BY ID **Data Structure:** ChannelID, UserID, ChannelInfo, ChannelCode
Name: CHANNEL DETAIL **Description:** Detailed channel information fetched for the student. **Origin:** 4.4 FETCH CHANNEL BY ID **Destination:** Student **Data Structure:** ChannelID, UserID, ChannelInfo, ChannelCode

10.3 Data Dictionary - Tan Po Yeh

10.3.1 Upload Submission and Mark Submission



External Entity
Name: STUDENT Description: STUDENT upload their submission for grading Input Data Flows: ASSIGNMENT APPENDIX Output Data Flows: SUBMISSION SCORE
Name: LECTURER Description: LECTURER get the STUDENT APPENDIX and give score to the submission Input Data Flows: STUDENT ASSIGNMENT Output Data Flows: SUBMISSION SCORE
Name: ADMIN Description: ADMIN can alter STUDENT APPENDIX score Input Data Flows: STUDENT ASSIGNMENT Output Data Flows: SUBMISSION SCORE

Data Process

Name: 8.1 CREATE APPENDIX

Description: Collect user input and construct user appendix

Input data flows: ASSIGNMENT APPENDIX

Output data flows: USER APPENDIX

Process:

```
CREATE APPENDIX
   INCLUDE "dbConn.php"
   GLOBAL connection

   SET materialTitle TO ""
   SET materialDescription TO ""

   IF POST CONTAINS 'txttitle'
      SET materialTitle TO ESCAPE_STRING(connection, POST['txttitle'])

   IF POST CONTAINS 'txtdescription'
      SET materialDescription TO ESCAPE_STRING(connection, POST['txtdescription'])

   SET createMaterial TO "display: none"

   IF POST CONTAINS 'btnCreateMaterial'
```

```
    SET createMaterial TO ""
  ELSE IF POST CONTAINS 'btnCreateMaterial2'
    SET insertQuery1 TO "INSERT INTO `material`(`Title`, `Description`, `ChanID`, `Timestamp`) VALUES ('materialTitle',
'materialDescription' , 'channelID', NOW())"

    IF EXECUTE_QUERY(connection, insertQuery1)
      SET materialID TO GET_LAST_INSERT_ID(connection)

      SET result TO SAVE_DOCUMENT(POST["material"], 'MaterialID', materialID, connection, 'Material')

      IF result IS EQUAL TO "File uploaded successfully."
        ECHO '<script>'
        ECHO 'window.alert("Create Successful!");'
        ECHO 'window.location.href = "/projectify/lecturermaterial.php?channelID=' + channelID + '";'
        ECHO '</script>'
      ELSE
        ECHO '<script>'
        ECHO 'window.alert("Error: ' + result + '");'
        ECHO '</script>'
      END IF
    ELSE
      ECHO '<script>'
```

```
        ECHO 'window.alert("Error inserting channel: ' + GET_DB_ERROR(connection) + '");'
        ECHO '</script>'
      END IF

      SET createMaterial TO "display: none"
   END IF
END CREATE APPENDIX

FUNCTION ESCAPE_STRING(connection, value)
   RETURN mysqli_real_escape_string(connection, value)
END FUNCTION

FUNCTION EXECUTE_QUERY(connection, query)
   RETURN mysqli_query(connection, query)
END FUNCTION

FUNCTION GET_LAST_INSERT_ID(connection)
   RETURN mysqli_insert_id(connection)
END FUNCTION

FUNCTION GET_DB_ERROR(connection)
   RETURN mysqli_error(connection)
```

```
END FUNCTION

FUNCTION SAVE_DOCUMENT(file, columnname, materialID, connection, tableName)
   // Check if file is uploaded successfully
   IF file["error"] IS EQUAL TO UPLOAD_ERR_OK
      // Get file information
      SET file_name TO file["name"]
      SET file_type TO file["type"]
      SET file_size TO file["size"]
      SET file_tmp_name TO file["tmp_name"]

      // Move uploaded file to desired location
      SET upload_dir TO "userdata/"
      SET destination TO upload_dir + file_name

      IF MOVE_UPLOADED_FILE(file_tmp_name, destination)
         // Insert file information into database
         SET updateQuery TO "UPDATE `tableName` SET `Filename` = 'file_name', `Filetype` = 'file_type', `Filesize` = 'file_size',
`Filecontent` = 'destination' WHERE `columnname` = 'materialID'"

         IF EXECUTE_QUERY(connection, updateQuery)
            RETURN "File uploaded successfully."
```

```
        ELSE
            RETURN "Error: " + GET_DB_ERROR(connection)
        END IF
    ELSE
        RETURN "Error uploading file."
    END IF
  ELSE
    RETURN "Please select a file to upload."
  END IF
END FUNCTION

FUNCTION MOVE_UPLOADED_FILE(tempName, destination)
  // Implementation to move uploaded file
  RETURN move_uploaded_file(tempName, destination)
END FUNCTION

FUNCTION EXECUTE_QUERY(connection, query)
  RETURN mysqli_query(connection, query)
END FUNCTION

FUNCTION GET_DB_ERROR(connection)
  RETURN mysqli_error(connection)
```

```
END FUNCTION
```

Name: 8.2 REGISTER APPENDIX

Description: Register user appendix to user_task database

Input data flows: APPENDIX RECORD

Output data flows: APPENDIX DATA

Process:

```
INCLUDE DATABASE CONNECTION FILE
INCLUDE "dbConn.php"

GLOBAL connection, icon

// Get channelID from url parameters
SET channelID TO GET['channelID']

// Get userID from session
SET userID TO SESSION["userid"]

// Fetch tasks from database
SET fetchQuery1 TO "SELECT * FROM `Task` WHERE `ChanID` = 'channelID' ORDER BY `Deadline` ASC"
SET result1 TO EXECUTE_QUERY(connection, fetchQuery1)

WHILE row IN FETCH_ASSOC(result1)
```

```
SET title TO ESCAPE_HTML(row['Title'])
SET description TO ESCAPE_HTML(row['Description'])
SET deadline TO ESCAPE_HTML(row['Deadline'])
SET taskID TO row['TaskID']

// Check if user has submitted the task
SET fetchQuery2 TO "
SELECT ut.*, a.*
FROM `User_Task` ut
JOIN `Appendix` a ON ut.AppendixID = a.AppendixID
WHERE ut.TaskID = 'taskID' AND a.UserID = 'userID'"
SET result2 TO EXECUTE_QUERY(connection, fetchQuery2)
SET row2 TO FETCH_ASSOC(result2)

SET status TO (row2 IS NOT NULL) ? ESCAPE_HTML(row2['Status']) : ""
SET appendixID TO (row2 IS NOT NULL) ? ESCAPE_HTML(row2['AppendixID']) : ""
SET filename TO (row2 IS NOT NULL) ? ESCAPE_HTML(row2['Filename']) : ""
SET filetype TO (row2 IS NOT NULL) ? ESCAPE_HTML(row2['Filetype']) : ""

// Determine the appropriate icon based on file type
SET icon TO ""
SWITCH filetype
```

```
    CASE 'application/pdf':
        SET icon TO 'pdf.png'
    CASE 'application/msword':
    CASE 'application/vnd.openxmlformats-officedocument.wordprocessingml.document':
    CASE 'application/vnd.openxmlformats-officedocument.word':
        SET icon TO 'word.png'
    DEFAULT:
        SET icon TO 'empty.png'
END SWITCH

IF status IS EQUAL TO "Submitted"
    ECHO "
    <div class='material2'>
        <h1>" + title + "</h1>
        <p>" + description + "</p>
        <p>Deadline: " + deadline + "</p>
        <div class='material-container'>
            <div class='material-panel' style='margin-right: 1%;'>
                <img src='image/" + icon + "' alt='" + filetype + "'>
                <span>" + filename + "</span>
            </div>
            <form action='' method='POST' enctype='multipart/form-data' style='text-decoration: none;'>
```

```
            <input type='hidden' name='channelID' value='" + channelID + "'>
            <input type='hidden' name='taskID' value='" + taskID + "'>
            <input type='hidden' name='oldAppendixID' value='" + appendixID + "'>
            <div style='display: flex; flex-direction: column;'>
               <input type='file' name='updateFile' id='fileInput" + taskID + "' required>
               <input type='submit' name='btnUpdateFile' id='submitBtn" + taskID + "' style='background-color: darkgray; color: white;'>
            </div>
         </form>
      </div>
   </div>"
ELSE
   ECHO "
   <div class='material2'>
      <h1>" + title + "</h1>
      <p>" + description + "</p>
      <p>Deadline: " + deadline + "</p>
      <div class='material-container'>
         <div class='material-panel' style='margin-right: 1%;'>
            <img style='max-height: 55px; width: auto;' src='image/" + icon + "' alt='" + filetype + "'>
            <span style='font-size: 12px;'>No Submission</span>
         </div>
         <div>
```

```
                <form action='' method='POST' enctype='multipart/form-data' style='text-decoration: none;'>
                    <input type='hidden' name='taskID' value='" + taskID + "'>
                    <input type='hidden' name='userID' value='" + userID + "'>
                    <div style='display: flex; flex-direction: column;'>
                        <input type='file' name='uploadFile' accept='.pdf,.doc,.docx,.xls,.xlsx,.ppt,.pptx' required>
                        <input type='submit' name='btnUploadFile' id='submitBtn" + taskID + "'>
                    </div>
                </form>
            </div>
        </div>
    </div>"
  END IF
END WHILE

// Handle file update
IF POST CONTAINS 'btnUpdateFile'
  SET taskID TO ESCAPE_STRING(connection, POST['taskID'])
  SET oldAppendixID TO ESCAPE_STRING(connection, POST['oldAppendixID'])

  SET query1 TO "INSERT INTO Appendix(`Timestamp`, `UserID`) VALUES (Now(),'userID')"
  IF EXECUTE_QUERY(connection, query1)
    SET appendixID TO GET_LAST_INSERT_ID(connection)
```

```
    CALL saveDocument(POST['updateFile'], 'AppendixID', appendixID, connection, 'Appendix')

    SET query2 TO "UPDATE user_task SET AppendixID = 'appendixID' WHERE TaskID = 'taskID' AND AppendixID =
'oldAppendixID'"
    IF EXECUTE_QUERY(connection, query2)
      ECHO "<script>
        window.alert('Update Successful!');
        window.location.href = '/projectify/studentassignment.php?channelID=' + channelID;
        </script>"
    ELSE
      ECHO "<script>
        window.alert('Error inserting into user_task: ' + GET_DB_ERROR(connection));
        window.location.href = '/projectify/studentassignment.php?channelID=' + channelID;
        </script>"
  ELSE
    ECHO "<script>
      window.alert('Error inserting into user_task: ' + GET_DB_ERROR(connection));
      window.location.href = '/projectify/studentassignment.php?channelID=' + channelID;
      </script>"
  EXIT

// Handle file upload
```

```
IF POST CONTAINS 'btnUploadFile'
   SET taskID TO ESCAPE_STRING(connection, POST['taskID'])
   SET userID TO ESCAPE_STRING(connection, POST['userID'])

   SET query1 TO "INSERT INTO Appendix(`Timestamp`, `UserID`) VALUES (Now(),'userID')"
   IF EXECUTE_QUERY(connection, query1)
      SET appendixID TO GET_LAST_INSERT_ID(connection)
      CALL saveDocument(POST['uploadFile'], 'AppendixID', appendixID, connection, 'Appendix')

      SET query2 TO "INSERT INTO user_task (TaskID, AppendixID, Status) VALUES ('taskID', 'appendixID', 'Submitted')"
      IF EXECUTE_QUERY(connection, query2)
         ECHO "<script>
            window.alert('Upload Successful!');
            window.location.href = '/projectify/studentassignment.php?channelID=' + channelID;
            </script>"
      ELSE
         ECHO "<script>
            window.alert('Error inserting into user_task: ' + GET_DB_ERROR(connection));
            window.location.href = '/projectify/studentassignment.php?channelID=' + channelID;
            </script>"
   ELSE
      ECHO "<script>
```

```
        window.alert('Error inserting into user_task: ' + GET_DB_ERROR(connection));
        window.location.href = '/projectify/studentassignment.php?channelID=' + channelID;
        </script>"
    EXIT
```

Name: 8.3 FETCH USER APPENDIX TABLE

Description: Fetch user_task table to form table for admin and lecturer

Input data flows: USER TASK RECORD

Output data flows: STUDENT ASSIGNMENT

Process:

```
// Fetch and display student data

SET userID TO SESSION["userid"]

SET fetchQuery TO "
    SELECT
    u.`UserID` AS StudentID,
    i.`Filepath` AS StudentImage,
    u.`Name` AS StudentName,
    u.`IC` AS StudentIC,
    u.`Email` AS StudentEmail,
    it.`Name` AS StudentInstitution
    FROM `User` u
```

```
    INNER JOIN `User_Institution` ui ON ui.`UserID` = u.`UserID`
    INNER JOIN `Institution` it ON it.`InstitutionID` = ui.`InstitutionID`
    INNER JOIN `Image` i ON i.`ImageID` = u.`ImageID`"

EXECUTE fetchQuery
SET result TO EXECUTE_QUERY(connection, fetchQuery)

// Check if query was successful
IF result IS NOT NULL
    ECHO "
      <table class='tasktable'>
        <tr>
          <th>Student Image</th>
          <th>Student Name</th>
          <th>Student IC</th>
          <th>Student Email</th>
          <th>Institution</th>
          <th>Status</th>
        </tr>
    "
    WHILE row IN FETCH_ASSOC(result)
        SET studID TO ESCAPE_HTML(row['StudentID'])
```

```
SET studImage TO ESCAPE_HTML(row['StudentImage'])
SET studName TO ESCAPE_HTML(row['StudentName'])
SET studIC TO ESCAPE_HTML(row['StudentIC'])
SET studEmail TO ESCAPE_HTML(row['StudentEmail'])
SET studInstitution TO ESCAPE_HTML(row['StudentInstitution'])

ECHO "
<tr>
 <td><img src='" + studImage + "'></img></td>
 <td>" + studName + "</td>
 <td>" + studIC + "</td>
 <td>" + studEmail + "</td>
 <td>" + studInstitution + "</td>
 <td>
  <form action='php/deleteMember2.php' method='POST'>
   <div class='columndisplay'>
    <input type='hidden' name='studID' value='" + studID + "' />
    <input type='hidden' name='chanID' value='" + channelID + "' />
    <input type='submit' name='btnDelete' value='Delete' />
   </div>
  </form>
 </td>
```

```
        </tr>"
    END WHILE

    ECHO "</table>"
END IF

FUNCTION EXECUTE_QUERY(connection, query)
    RETURN mysqli_query(connection, query)
END FUNCTION

FUNCTION FETCH_ASSOC(result)
    RETURN mysqli_fetch_assoc(result)
END FUNCTION

FUNCTION ESCAPE_HTML(value)
    RETURN htmlspecialchars(value, ENT_QUOTES, 'UTF-8')
END FUNCTION
```

Name: 9.1 UPDATE USER TASK

Description: Update score to user submission

Input data flows: SUBMISSION SCORE

Output data flows: USER SCORE

Process:

```
REQUIRE_ONCE(__DIR__ . '/../dbConn.php')
GLOBAL connection

// If appendixID and score are available
IF POST CONTAINS 'appendixID' AND POST CONTAINS 'score'
   SET appendixID TO ESCAPE_STRING(connection, POST['appendixID'])
   SET score TO ESCAPE_STRING(connection, POST['score'])

   // Check if appendixID and score are not empty
   IF appendixID IS NOT EMPTY AND score IS NOT EMPTY
      SET query TO "UPDATE `User_Task` SET `Score` = 'score' WHERE `AppendixID` = 'appendixID'"

      EXECUTE QUERY
      IF EXECUTE_QUERY(connection, query)
         ECHO "<script>
            window.alert('Update Successful!');
            window.location.href = '/projectify/report.php';
            </script>"
      ELSE
         ECHO "<script>
            window.alert('Error updating the score.');
            window.location.href = '/projectify/report.php';
```

```
            </script>"
    ELSE
        // Output error if appendixID or score is empty
        ECHO "<script>
            window.alert('Error: Appendix ID and score must not be empty.');
            window.location.href = '/projectify/report.php';
            </script>"
ELSE
    // Output error if required data is missing in post
    ECHO "<script>
        window.alert('Error: Required data not received.');
        window.location.href = '/projectify/report.php';
        </script>"
```

Name: 9.2 FETCH SCORE

Description: Fetch student submission score to student

Input data flows: USER TASK RECORD WITH SCORE

Output data flows: SUBMISSION SCORE

Process:

```
SET fetchQuery TO "
    SELECT
    i.`Filepath` AS ChannelImage,
    c.`Title` AS ChannelName,
```

```
    c.`ChannelCode` AS ChannelCode,
    t.`Title` AS TaskTitle,
    a.`AppendixID` AS AppendixID,
    a.`Filename` AS Submission,
    ut.`Status` AS Status,
    ut.`Score` AS Score
    FROM `User_Channel` uc
    INNER JOIN `Task` t ON uc.ChanID = t.ChanID
    INNER JOIN `User_Task` ut ON ut.`TaskID` = t.`TaskID`
    INNER JOIN `Appendix` a ON a.`AppendixID` = ut.`AppendixID`
    INNER JOIN `Channel` c ON c.ChanID = uc.ChanID
    INNER JOIN `Image` i ON i.`ImageID` = c.`ImageID`
    WHERE a.`UserID` = 'userID'
    AND ut.`Score` IS NOT NULL;"

EXECUTE QUERY
SET result TO EXECUTE_QUERY(connection, fetchQuery)

IF result IS NOT EMPTY
    OUTPUT "<table class='tasktable'>
            <tr>
                <th>Channel</th>
```

```
            <th>Task Title</th>
            <th>Appendix ID</th>
            <th>Appendix Name</th>
            <th>Status</th>
            <th>Score</th>
          </tr>"

WHILE THERE ARE ROWS IN result
   // Retrive data from each row
   SET channelImage TO ESCAPE_SPECIAL_CHARS(result['ChannelImage'])
   SET channelName TO ESCAPE_SPECIAL_CHARS(result['ChannelName'])
   SET channelCode TO ESCAPE_SPECIAL_CHARS(result['ChannelCode'])
   SET taskTitle TO ESCAPE_SPECIAL_CHARS(result['TaskTitle'])
   SET appendixID TO ESCAPE_SPECIAL_CHARS(result['AppendixID'])
   SET submission TO ESCAPE_SPECIAL_CHARS(result['Submission'])
   SET status TO ESCAPE_SPECIAL_CHARS(result['Status'])
   SET score TO ESCAPE_SPECIAL_CHARS(result['Score'])

   // Output row in table format
   OUTPUT "<tr>
           <td class='column1'>
              <img class='channel' src='" + channelImage + "'></img>
```

```
                <div class='detail'>
                    <h1>" + channelName + "</h1>
                    <p>" + channelCode + "</p>
                </div>
            </td>
            <td>" + taskTitle + "</td>
            <td>" + appendixID + "</td>
            <td>" + submission + "</td>
            <td>" + status + "</td>
            <td>" + score + "</td>
        </tr>"

    END WHILE

    OUTPUT "</table>"

ELSE
    // Output error message when query failed
    OUTPUT "Error: Failed to fetch data."
```

Data Store
Name: D8.1 APPENDIX Description: APPENDIX store STUDENT assignment appendix information Input data flows: USER APPENDIX Output data flows: APPENDIX RECORD Data Structure: Appendix = AppendixID + Title + Filename + Filetype + Filesize + Filecontent + Timestamp + UserID
Name: D8.2 USER_TASK Description: USER_TASK assigns user appendix with task identifier Input data flows: APPENDIX DATA, USER SCORE Output data flows: USER TASK RECORD, USER TASK RECORD WITH SCORE Data Structure: User_Task = TaskID + AppendixID + (Status) + (Score)

D8.1 APPENDIX:

Field Name	Field Type	Field Length	Constraint	Description
AppendixID	Alphanumeric	11	Primary Key, Auto Increment, Not Null	Unique identifier for each appendix record.
Filename	Varchar	200	Not Null	Name of the file
Filetype	Varchar	50	Not Null	Type of the file (e.g., PDF, DOCX)
Filesize	Int	11	Not Null	Size of the file in bytes
Filecontent	Blob		Not Null	Binary content of the file
Timestamp	Timestamp		Not Null	Upload time of the appendix
UserID	Alphanumeric	11	Foreign Key	Identifier of the user who uploaded the file

D8.2 USER_TASK:

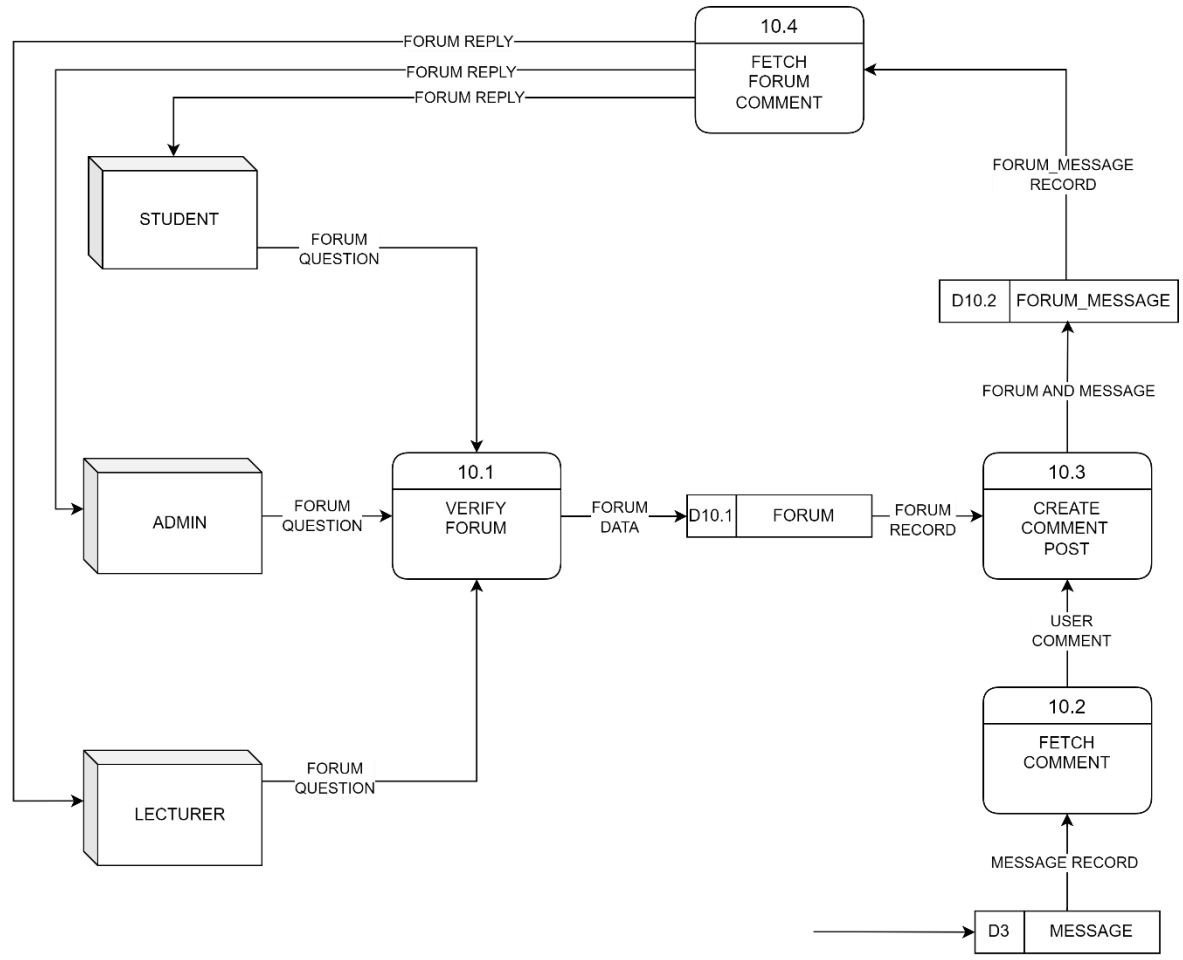
Field Name	Field Type	Field Length	Constraint	Description
TaskID	Alphanumeric	11	Primary Key, Foreign Key, Not Null	Unique identifier for each task record.
AppendixID	Alphanumeric	11	Primary Key, Foreign Key, Not Null	Unique identifier for each appendix record.
Status	Varchar	50	Not Null	Status of the task appendix (e.g., submitted, completed).
Score	Varchar	255		Optional score or rating related to the task appendix.

Data Flow
Name: ASSIGNMENT APPENDIX Description: STUDENT submission appendix information Origin: STUDENT external entity Destination: 8.1 CREATE APPENDIX Data Structure: ASSIGNMENT APPENDIX = Title + Filename + Filetype + Filesize + Filecontent + Timestamp
Name: USER APPENDIX Description: Assign user id to appendix Origin: 8.1 CREATE APPENDIX Destination: D8.1 APPENDIX Data Structure: USER APPENDIX = Title + Filename + Filetype + Filesize + Filecontent + Timestamp + UserID
Name: APPENDIX RECORD Description: Fetch the appendix record from the database Origin: D8.1 APPENDIX

Destination: 8.2 REGISTER APPENDIX Data Structure: APPENDIX RECORD = AppendixID + Title + Filename + Filetype + Filesize + Filecontent + Timestamp + UserID
Name: APPENDIX DATA Description: Fetch the selected submission appendix ID to register in the USER_TASK Origin: 8.2 REGISTER APPENDIX Destination: D8.2 USER_TASK Data Structure: APPENDIX DATA = AppendixID
Name: USER TASK RECORD Description: Fetch the user_task record to form a table Origin: D8.2 USER_TASK Destination: 8.3 FETCH USER APPENDIX TABLE Data Structure: USER TASK RECORD = TaskID + AppendixID + (Status) + (Score)
Name: STUDENT ASSIGNMENT Description: Fetch the user appendix table to the admin Origin: 8.3 FETCH USER APPENDIX TABLE Destination: ADMIN external entity Data Structure: STUDENT ASSIGNMENT = Channel + Student Name + Student IC + Assignment + Submission + Submission Date + Due Date + (Score)
Name: STUDENT ASSIGNMENT Description: Fetch the user appendix table to the lecturer Origin: 8.3 FETCH USER APPENDIX TABLE Destination: LECTURER external entity Data Structure: STUDENT ASSIGNMENT = Channel + Student Name + Student IC + Assignment + Submission + Submission Date + Due Date + (Score)
Name: SUBMISSION SCORE Description: Student appendix score Origin: ADMIN external entity Destination: 9.1 UPDATE USER TASK Data Structure: SUBMISSION SCORE = Appendix ID + Score
Name: SUBMISSION SCORE Description: Student appendix score Origin: LECTURER external entity Destination: 9.1 UPDATE USER TASK

Data Structure: SUBMISSION SCORE = Appendix ID + Score
Name: USER SCORE Description: Appendix score Origin: 9.1 UPDATE USER TASK Destination: D8.2 USER_TASK Data Structure: USER SCORE = Score
Name: USER TASK RECORD WITH SCORE Description: Fetch the user appendix score from the database Origin: D8.2 USER_TASK Destination: 9.2 FETCH SCORE Data Structure: USER TASK RECORD WITH SCORE = TaskID + AppendixID + Status + Score
Name: SUBMISSION SCORE Description: Fetch the appendix score to the user Origin: 9.2 FETCH SCORE Destination: STUDENT external entity Data Structure: SUBMISSION SCORE = Channel + Task Title + Appendix ID + Appendix Name + Status + Score

10.3.2 Handle Forum



External Entity
Name: STUDENT Description: STUDENT create forum and receive reply Input Data Flows: FORUM REPLY Output Data Flows: FORUM QUESTION
Name: LECTURER Description: LECTURER creates forum and receives reply Input Data Flows: FORUM REPLY Output Data Flows: FORUM QUESTION
Name: ADMIN Description: ADMIN creates forum and receives reply Input Data Flows: FORUM REPLY Output Data Flows: FORUM QUESTION

Data Process

Name: 10.1 VERIFY FORUM

Description: Collect and verify user forum content

Input data flows: FORUM QUESTION

Output data flows: FORUM DATA

Process:

```
// Include database connection and function files
INCLUDE "dbConn.php"
INCLUDE "php/function.php"
GLOBAL connection

// Initialize variables
SET createChannel TO "display: none"
SET userID TO SESSION["userid"]

// Check which form is submitted
IF POST CONTAINS 'createPost'
   SET createChannel TO ""
ELSE IF POST CONTAINS 'createPost2'
   SET postProfile TO $_FILES['postprofile']
   SET postTitle TO ESCAPE_STRING(connection, POST['txttitle'])
```

```
SET postDescription TO ESCAPE_STRING(connection, POST['txtdescription'])

SET insertQuery1 TO "INSERT INTO `Forum`(`Topic`, `Content`, `Timestamp`, `HostID`) VALUES ('postTitle', 'postDescription',
NOW(), 'userID')"

EXECUTE INSERT QUERY
IF EXECUTE_QUERY(connection, insertQuery1)
   SET forumID TO LAST_INSERT_ID(connection)

   // Handle image upload
   IF postProfile IS NOT NULL
      SET imageID TO INSERT_IMAGE(postProfile, connection)

      // Check if image upload successful\
      IF IS_NUMERIC(imageID)
         SET updateQuery TO "UPDATE `Forum` SET `ImageID` = 'imageID' WHERE `ForumID` = 'forumID'"

         EXECUTE UPDATE QUERY
         IF EXECUTE_QUERY(connection, updateQuery)
            OUTPUT "<script>
                  window.alert('Create Successful!');
                  window.location.href = '/projectify/discussionforum.php';
```

```
                </script>"
            ELSE
                OUTPUT "<script>
                    window.alert('Error updating forum with image: ERROR_MESSAGE');
                    window.location.href = '/projectify/discussionforum.php';
                    </script>"
        ELSE
            OUTPUT "<script>
                window.alert('Error uploading image: imageID');
                window.location.href = '/projectify/discussionforum.php';
                </script>"
    ELSE
        OUTPUT "<script>
            window.alert('Create Successful!');
            window.location.href = '/projectify/discussionforum.php';
            </script>"
ELSE
    OUTPUT "<script>
        window.alert('Error inserting forum: ERROR_MESSAGE');
        window.location.href = '/projectify/discussionforum.php';
        </script>"
```

```
    SET createChannel TO "display: none"

END IF
```

Name: 10.2 FETCH COMMENT

Description: Fetch comment from message to forum

Input data flows: MESSAGE RECORD

Output data flows: USER COMMENT

Process:

```
// Initialize variables
SET forums = []

FOR EACH forum IN forums
    SET forumID = forum['ForumID']

    // Fetch message from current query
    SET fetchMessagesQuery TO "
        SELECT
            m.`Content` AS Message,
            m.`Timestamp` AS MessageTimestamp,
            u.`Name` AS Repliername
```

```
    FROM `Message` m
    INNER JOIN `Forum_Message` fm ON fm.`MessageID` = m.`MessageID`
    INNER JOIN `User` u ON u.`UserID` = m.`SenderID`
    WHERE fm.`ForumID` = 'forumID'
"

SET messagesResult = EXECUTE_QUERY(connection, fetchMessagesQuery)

// Initialize messages
SET messages = []

// Check if query was successful
IF messagesResult IS NOT NULL
    WHILE THERE ARE MORE ROWS IN messagesResult
        SET message = FETCH_NEXT_ROW(messagesResult)

        ADD message TO messages
    END WHILE

SET forum['Messages'] = messages

ADD forum TO forums
```

```
END FOR EACH

// Display forum and their message
FOR EACH forum IN forums
    OUTPUT forum['ForumID']
    OUTPUT forum['Topics']
    OUTPUT forum['Content']
    OUTPUT forum['Image']
    OUTPUT forum['Timestamp']
    OUTPUT forum['HostName']
    OUTPUT forum['HostID']

    // Check if forum['Messages'] is empty
    IF forum['Messages'] IS EMPTY
        OUTPUT "No Replies Yet"
        OUTPUT "Be the first to reply to this forum!"
    ELSE
        FOR EACH message IN forum['Messages']
            OUTPUT message['Message']
            OUTPUT message['MessageTimestamp']
            OUTPUT message['Repliername']
        END FOR EACH
```

```
    END IF

    OUTPUT COMMENT FORM FOR forum['ForumID'] AND forum['HostID']
END FOR EACH
```

Name: 10.3 CREATE COMMENT POST

Description: Identify and integrate comment posts to the forum_message database

Input data flows: USER COMMENT

Output data flows: FORUM AND MESSAGE

Process:

```
// Get posted data
SET comment TO POST['comment']
SET forumID TO POST['forumID']
SET hostID TO POST['hostID']
SET insertCommentQuery TO "
    INSERT INTO `Message` (`Content`, `Timestamp`, `SenderID`)
    VALUES ('comment', NOW(), 'hostID')
"

IF EXECUTE_QUERY(connection, insertCommentQuery)
    REDIRECT TO "/projectify/discussionforum.php"
ELSE
```

```
    OUTPUT "Error inserting comment: ERROR_MESSAGE"
    REDIRECT TO "/projectify/discussionforum.php"
END IF
```

Name: 10.4 FETCH FORUM COMMENT

Description: Fetch forum message to the target post and user

Input data flows: FORUM AND MESSAGE

Output data flows: FORUM REPLY

Process:

```
// Initialize variable
SET forums = []

FOR EACH forum IN forums
    SET forumID = forum['ForumID']
    SET fetchMessagesQuery TO "
        SELECT
            m.`Content` AS Message,
            m.`Timestamp` AS MessageTimestamp,
            u.`Name` AS Repliername
        FROM `Message` m
        INNER JOIN `Forum_Message` fm ON fm.`MessageID` = m.`MessageID`
        INNER JOIN `User` u ON u.`UserID` = m.`SenderID`
        WHERE fm.`ForumID` = 'forumID'
```

```
    "

    SET messagesResult = EXECUTE_QUERY(connection, fetchMessagesQuery)
    SET messages = []
    IF messagesResult IS NOT NULL
        // LOOP THROUGH EACH MESSAGE RESULT
        WHILE THERE ARE MORE ROWS IN messagesResult
            // FETCH MESSAGE DETAILS
            SET message = FETCH_NEXT_ROW(messagesResult)

            // ADD MESSAGE TO messages LIST
            ADD message TO messages
        END WHILE
    SET forum['Messages'] = messages
    ADD forum TO forums
END FOR EACH

FOR EACH forum IN forums
    // OUTPUT FORUM DETAILS
    OUTPUT forum['ForumID']
    OUTPUT forum['Topics']
    OUTPUT forum['Content']
```

```
    OUTPUT forum['Image']
    OUTPUT forum['Timestamp']
    OUTPUT forum['HostName']
    OUTPUT forum['HostID']

    // CHECK IF forum['Messages'] IS EMPTY
    IF forum['Messages'] IS EMPTY
        // OUTPUT NO REPLIES YET MESSAGE
        OUTPUT "No Replies Yet"
        OUTPUT "Be the first to reply to this forum!"
    ELSE
        FOR EACH message IN forum['Messages']
            // OUTPUT MESSAGE DETAILS
            OUTPUT message['Message']
            OUTPUT message['MessageTimestamp']
            OUTPUT message['Repliername']
        END FOR EACH
    END IF
END FOR EACH
```

Data Store
Name: D10.1 FORUM Description: Store user-created forum information Input data flows: FORUM DATA Output data flows: FORUM RECORD Data Structure: Forum = ForumID + Topics + Content + Timestamp + HostID + ImageID
Name: D10.2 FORUM_MESSAGE Description: Assign forum comments with the corresponding forum database table Input data flows: FORUM AND MESSAGE Output data flows: FORUM_MESSAGE RECORD Data Structure: Forum_Message = ForumID + MessageID + Status
Name: D3 MESSAGE Description: Store forum comment Input data flows: MESSAGE DATA Output data flows: MESSAGE RECORD Data Structure: Message = MessageID + Content + Timestamp + ReceiverID + SenderID

D10.1 FORUM:

Field Name	Field Type	Field Length	Constraint	Description
ForumID	Alphanumeric	11	Primary Key, Auto Increment, Not Null	Unique identifier for each forum record
Topics	Varchar	50	Not Null	Topics of the forum
Content	Text		Not Null	Content/Question of the forum
Timestamp	Timestamp		Not Null	Upload time of the forum
ImageID	Alphanumeric	11	Foreign Key	Identifier of the image for the forum

OwnerID	Alphanumeric	11	Foreign Key, Not Null	Identifier of the user who host the forum

D10.2 FORUM_MESSAGE:

Field Name	Field Type	Field Length	Constraint	Description
ForumID	Alphanumeric	11	Primary Key, Foreign Key, Not Null	Unique identifier for each forum.
MessageID	Alphanumeric	11	Primary Key, Foreign Key, Not Null	Unique identifier for each message.
Status	Varchar	50		Status of the message (e.g., active, archived).

D3 MESSAGE:

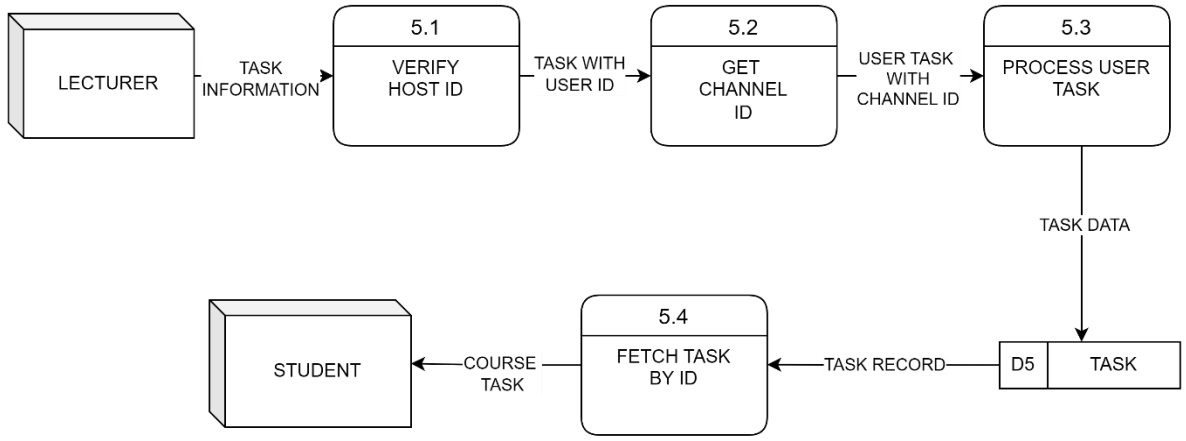
Field Name	Field Type	Field Length	Constraint	Description
MessageID	Alphanumeric	11	Primary Key, Auto Increment, Not Null	Unique identifier for each message record
Content	Text		Not Null	Content of the message
Timestamp	Timestamp		Not Null	Upload time of the message
SenderID	Alphanumeric	11	Foreign Key	Identifier of the user who send this message
ReceiverID	Alphanumeric	11	Foreign Key	Identifier of the user who receive this message

Data Flow

Name: FORUM QUESTION Description: Forum content created by the user Origin: STUDENT external entity, ADMIN external entity, LECTURER external entity Destination: 10.1 VERIFY FORUM Data Structure: FORUM QUESTION = Topics + Content + Timestamp
Name: FORUM DATA Description: User forum data Origin: STUDENT external entity, ADMIN external entity, LECTURER external entity Destination: 10.1 VERIFY FORUM Data Structure: FORUM DATA = ForumID + Topics + Content + Timestamp + HostID + ImageID
Name: FORUM RECORD Description: User record from database Origin: D10.1 FORUM Destination: 10.3 CREATE COMMENT POST Data Structure: FORUM RECORD = ForumID + Topics + Content + Timestamp + HostID + ImageID
Name: MESSAGE RECORD Description: Message comment from database Origin: D3 MESSAGE Destination: 10.2 FETCH COMMENT Data Structure: MESSAGE RECORD = MessageID + Content + Timestamp + SenderID
Name: USER COMMENT Description: Fetch forum message comment Origin: 10.2 FETCH COMMENT Destination: 10.3 CREATE COMMENT POST Data Structure: USER COMMENT = MessageID + SenderID
Name: FORUM AND MESSAGE Description: Identify and integrate the message to the corresponding forum_message database table Origin: 10.3 CREATE COMMENT POST Destination: D10.2 FORUM_MESSAGE Data Structure: FORUM AND MESSAGE = MessageID
Name: FORUM_MESSAGE RECORD

Description: Fetch forum message record to the user Origin: D10.2 FORUM_MESSAGE Destination: 10.4 FETCH FORUM COMMENT Data Structure: FORUM_MESSAGE RECORD = ForumID + MessageID + Status
Name: FORUM REPLY Description: Fetch forum reply to the user Origin: 10.4 FETCH FORUM COMMENT Destination: STUDENT external entity, ADMIN external entity, LECTURER external entity Data Structure: FORUM REPLY = ForumID + MessageID + Content + Timestamp + SenderID + Status

10.4 Data Dictionary - Tong Jia Chuen



External Entity
Name: Lecturer **Description:** The lecturer send task information **Input Data Flows:** none **Output Data Flows:** Task Information
Name: Student **Description:** Retrieve Tasks **Input Data Flows:** Course Task **Output Data Flows:** none

Data Process
Name: 5.1 VERIFY HOST ID Description: Verifies the lecturer's identity Input Data Flows: Task information Output Data Flows: Task with user ID Process: // Retrieve user ID from session SET userID = SESSION["userid"]

```
// Fetch user information
SET fetchQuery1 = "SELECT * FROM `User` WHERE `UserID` = userID"
SET result1 = EXECUTE_QUERY(connection, fetchQuery1)

IF result1 IS NOT NULL AND HAS_ROWS(result1)
   SET row = FETCH_ASSOC(result1)
   SET userImageID = HTML_SPECIALCHARS(row['ImageID'], ENT_QUOTES, 'UTF-
8')

// Fetch user image information
SET fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = userImageID"
SET result2 = EXECUTE_QUERY(connection, fetchQuery2)

IF result2 IS NOT NULL AND HAS_ROWS(result2)
   SET row = FETCH_ASSOC(result2)
   SET userImagePath = HTML_SPECIALCHARS(row['Filepath'], ENT_QUOTES, 'UTF-
8')

// Fetch number of pending tasks
SET fetchQuery3 = "
SELECT COUNT(*) AS numberPending
FROM `User_Channel` uc
INNER JOIN `Task` t ON uc.ChanID = t.ChanID
INNER JOIN `User_Task` ut ON ut.`TaskID` = t.`TaskID`
INNER JOIN `Appendix` a ON a.`AppendixID` = ut.`AppendixID`
INNER JOIN `Channel` c ON c.`ChanID` = t.`ChanID`
WHERE c.`OwnerID` = userID AND ut.`Score` IS NOT NULL AND t.`TaskID` IS NOT
NULL
"
SET result3 = EXECUTE_QUERY(connection, fetchQuery3)

IF result3 IS NOT NULL
   SET row = FETCH_ASSOC(result3)
   SET totalOfPending = row['numberPending']
```

```
// Fetch number of channels owned by user
SET fetchQuery4 = "
SELECT COUNT(*) AS channelNumber
FROM `Channel`
WHERE `OwnerID` = userID
"
SET result4 = EXECUTE_QUERY(connection, fetchQuery4)

IF result4 IS NOT NULL
   SET row = FETCH_ASSOC(result4)
   SET totalOfChannel = row['channelNumber']

// Fetch number of task submissions pending evaluation
SET fetchQuery5 = "
SELECT COUNT(*) AS numberSubmission
FROM `User_Channel` uc
INNER JOIN `Task` t ON uc.ChanID = t.ChanID
INNER JOIN `User_Task` ut ON ut.`TaskID` = t.`TaskID`
INNER JOIN `Appendix` a ON a.`AppendixID` = ut.`AppendixID`
INNER JOIN `Channel` c ON c.`ChanID` = t.`ChanID`
WHERE c.`OwnerID` = userID AND ut.`Score` IS NULL AND t.`TaskID` IS NOT NULL
"
SET result5 = EXECUTE_QUERY(connection, fetchQuery5)

IF result5 IS NOT NULL
   SET row = FETCH_ASSOC(result5)
   SET totalOfMark = row['numberSubmission']

// Retrieve task information from POST data
SET taskTitle = IF IS_SET($_POST['txttitle']) THEN ESCAPE_STRING(connection,
$_POST['txttitle']) ELSE ""
SET taskDescription = IF IS_SET($_POST['txtdescription']) THEN
ESCAPE_STRING(connection, $_POST['txtdescription']) ELSE ""
```

```
SET taskDeadline = IF IS_SET($_POST['txtdeadline']) THEN
ESCAPE_STRING(connection, $_POST['txtdeadline']) ELSE ""

// Initialize task creation display state
SET createTask = "display: none"

// Check if the create assignment button is pressed
IF IS_SET($_POST['btnCreateAssignment'])
   SET createTask = ""

// Check if the create assignment confirmation button is pressed
ELSE IF IS_SET($_POST['btnCreateAssignment2'])
   // Insert new task into the database
   SET insertQuery1 = "INSERT INTO `task`(`Title`, `Description`, `Deadline`, `ChanID`)
VALUES (taskTitle, taskDescription, taskDeadline, channelID)"
   IF EXECUTE_QUERY(connection, insertQuery1)
      OUTPUT "Create Successful!"
      REDIRECT "/projectify/lecturerassignment.php?channelID=" + channelID
   ELSE
      OUTPUT "Error inserting task: " + GET_ERROR(connection)
   SET createTask = "display: none"
```

Name: 5.2 GET CHANNEL ID

Description: Retrieves the channel ID link with the verified user task

Input Data Flows: Task User ID

Output Data Flows: User Task With Channel ID

Process:

```
START_SESSION
SET channelID = GET_PARAMETER('channelID')
INCLUDE "php/function.php"
INCLUDE "php/createTask.php"
INCLUDE "php/fetchChannel.php"

HTML_DOCUMENT_START
```

```
 HTML_ELEMENT(lang="en")
  HEAD_ELEMENT
   META_ELEMENT(charset="UTF-8")
   META_ELEMENT(name="viewport", content="width=device-width, initial-
scale=1.0")
   LINK_ELEMENT(rel="stylesheet", href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.2/css/all.min.css")
   LINK_ELEMENT(rel="stylesheet", href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css")

SCRIPT_ELEMENT(src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js")
   TITLE_ELEMENT "Channel"
   LINK_ELEMENT(rel="stylesheet", href="css/channel.css")
   LINK_ELEMENT(rel="stylesheet", href="css/assignment.css")
   SCRIPT_ELEMENT(src="javascript/function.js", defer)
   SCRIPT_CONTENT
    IF (WINDOW.HISTORY.REPLACE_STATE EXISTS)
     CALL WINDOW.HISTORY.REPLACE_STATE(null, null,
WINDOW.LOCATION.HREF)
   SCRIPT_END
  HEAD_END
  BODY_ELEMENT
   DIV_ELEMENT(class="main")
    DIV_ELEMENT(class="choice")
     BUTTON_ELEMENT(style="cursor: pointer; border:none; background-color:
transparent;", onclick="redirectTo('lecturerchannel.php')", id="backbutton")
      IMG_ELEMENT(class="back", src="image/back.png")
     BUTTON_END
     DIV_ELEMENT(class="title")
      IMG_ELEMENT(src="image/whitelogo.png", alt="")
      H1_ELEMENT "Projectify"
     DIV_END
     DIV_ELEMENT(class="photo")
      IMG_ELEMENT(class="profile-frame", src="<?php echo $imagePath; ?>")
```

```
        DIV_END

        P_ELEMENT(class="course") "<?php echo $channelName; ?>"
        P_END
        P_ELEMENT(class="info") "Lecturer: <?php echo $userName; ?>"
        P_END
        P_ELEMENT(class="info") "Channel Code: <?php echo $channelCode; ?>"
        P_END
        P_ELEMENT(class="info") "Student: <?php echo $totalOfMember ?>"
        P_END

        DIV_ELEMENT(class="line")
        DIV_END

        A_ELEMENT(href="lecturermaterial.php?channelID=<?php echo $channelID; ?>")
          IMG_ELEMENT(src="image/controlpanel.png", alt="Icon")
          "Material"
        A_END
        A_ELEMENT(href="lecturerassignment.php?channelID=<?php echo
$channelID; ?>")
          IMG_ELEMENT(src="image/classroom.png", alt="Icon")
          "Assignment"
        A_END
        A_ELEMENT(href="lecturermember.php?channelID=<?php echo $channelID; ?>")
          IMG_ELEMENT(src="image/member.png", alt="Icon")
          "Member"
        A_END
      DIV_END

      DIV_ELEMENT(class="panel")
        DIV_ELEMENT(class="material1")
          FORM_ELEMENT(action="", method="POST")
            BUTTON_ELEMENT(name="btnCreateAssignment", style="cursor: pointer;
border:none; background-color: white;")
```

```
            IMG_ELEMENT(class="export", src="image/export.png")
          BUTTON_END
          INCLUDE "php/fetchTask.php"
        FORM_END
      DIV_END
    DIV_END

    DIV_ELEMENT(style="<?php echo $createTask; ?>",
class="createMaterialContainer")
      A_ELEMENT(href="javascript:history.go(-1);", class="back-icon")
       I_ELEMENT(class="fas fa-arrow-left")
       I_END
      A_END
      FORM_ELEMENT(action="", method="POST", enctype="multipart/form-data")
       H1_ELEMENT "Create Assignment"
       SPAN_ELEMENT "Post Task For Evaluation"
       INPUT_ELEMENT(name="txttitle", placeholder="Task Title", required)
       TEXTAREA_ELEMENT(name="txtdescription", placeholder="Task Description",
required)
       INPUT_ELEMENT(type="date", name="txtdeadline", placeholder="Task
Deadline", required)
       BUTTON_ELEMENT(name="btnCreateAssignment2") "Upload Material"
       BUTTON_END
      FORM_END
    DIV_END

  DIV_END
 BODY_END
HTML_DOCUMENT_END
```

Name: 5.3 PROCESS USER TASK

Description: Processes the user task based on the provided channel ID

Input Data Flows: User Task with Channel ID

Output Data Flows: Task Data

Process:

```
 // Include database connection
INCLUDE "dbConn.php"

// Initialize variables for task creation
SET taskTitle = ""
SET taskDescription = ""
SET taskDeadline = ""
SET createTask = "display: none"

// Sanitize and assign task title, description, and deadline from POST data
IF IS_SET($_POST['txttitle'])
   SET taskTitle = ESCAPE_STRING(connection, $_POST['txttitle'])

IF IS_SET($_POST['txtdescription'])
   SET taskDescription = ESCAPE_STRING(connection, $_POST['txtdescription'])

IF IS_SET($_POST['txtdeadline'])
   SET taskDeadline = ESCAPE_STRING(connection, $_POST['txtdeadline'])

// Check button actions for task creation
IF IS_SET($_POST['btnCreateAssignment'])
   SET createTask = ""

ELSE IF IS_SET($_POST['btnCreateAssignment2'])
   // Insert new task into the database
   SET insertQuery1 = "INSERT INTO `task`(`Title`, `Description`, `Deadline`, `ChanID`)
VALUES (taskTitle, taskDescription, taskDeadline, channelID)"

   // Execute the insert query
```

```
IF EXECUTE_QUERY(connection, insertQuery1)
   OUTPUT "Create Successful!"
   REDIRECT "/projectify/lecturerassignment.php?channelID=" + channelID

ELSE
   OUTPUT "Error inserting task: " + GET_ERROR(connection)

SET createTask = "display: none"
```

Name: 5.4 FECTH TASK BY ID

Description: Retrieves the task record based on the ID provided by the student.

Input Data Flows: Task Record

Output Data Flows: Course Task

Process:

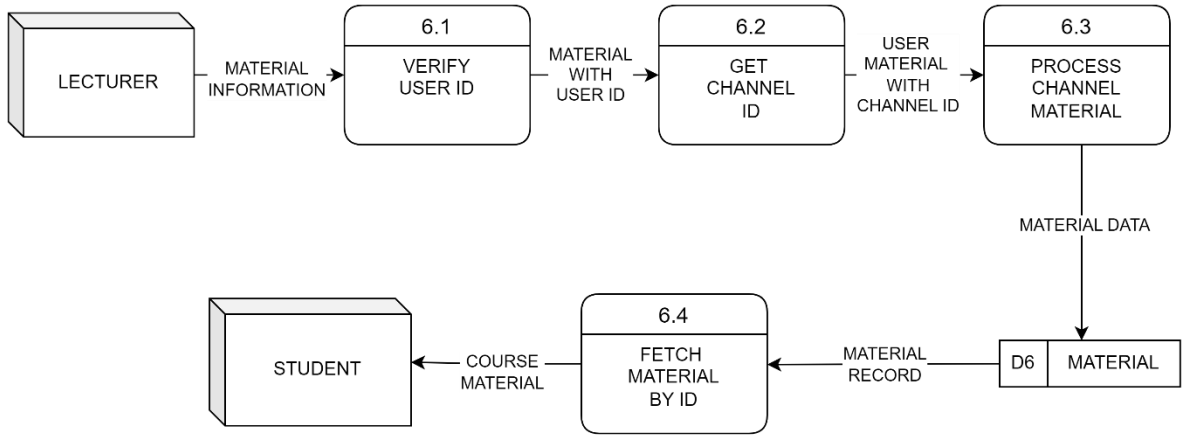
```
// Include database connection file
INCLUDE "dbConn.php"

// Initialize fetch query to select tasks based on channel ID, ordered by deadline
fetchQuery = "SELECT * FROM `Task` WHERE `ChanID` = channelID ORDER BY
`Deadline` ASC"

// Execute fetch query
result = EXECUTE_QUERY(connection, fetchQuery)

// Process each fetched material
WHILE row = FETCH_ASSOC(result)
   // Retrieve and sanitize material details
   title = HTML_SPECIALCHARS(row['Title'], ENT_QUOTES, 'UTF-8')
   description = HTML_SPECIALCHARS(row['Description'], ENT_QUOTES, 'UTF-8')
   deadline = HTML_SPECIALCHARS(row['Deadline'], ENT_QUOTES, 'UTF-8')

   // Output HTML for each material
```

```
OUTPUT '
<div class="material2">
   <h1>' + title + '</h1>
   <p>' + description + '</p>
   <p>Deadline: ' + deadline + '</p>
   <div class="material-container">
      <div class="material-panel" onclick="displayMessage1()">
         <img src="image/upload.png"><img>
      </div>
   </div>
</div>
'
```

Data Store
Name: D5 Task **Description:** Store Data Task **Input Data Flows:** Task Data **Output Data Flows:** Task Record **Data Structure:** TaskID + Title + Description + Deadline + ChanID + OwnerID

Data Flow
Name: Task Information **Description:** Information about the task provided by the lecturer **Origin:** Lecture external entities **Destination:** 5.1 Verify Host ID **Data Structure:** Title + Description + Deadline
Name: Task with user ID **Description:** Task information tagged with the verified user ID. **Origin:** 5.1 Verify Host ID **Destination:** 5.2 Get Channel ID **Data Structure:** TaskID + Title + Description + Deadline + UserID

Name: User Task with Channel ID **Description:** Task information associated with a specific channel ID. **Origin:** 5.2 Get Channel ID **Destination:** 5.3 Process User Task **Data Structure:** TaskID + Title + Description + Deadline + UserID + ChanID
Name: Task Data **Description:** Processed task information ready for storage. **Origin:** 5.3 Process User Task **Destination:** D5 Task **Data Structure:** TaskID + Title + Description + Deadline + ChanID + OwnerID
Name: Task Record **Description:** The specific task record retrieved for the student. **Origin:** D5 Task **Destination:** Student **Data Structure:** TaskID + Title + Description + Deadline + ChanID + OwnerID



External Entity
Name: Lecturer **Description:** Lecturer provide material information **Input Data Flows:** none **Output Data Flows:** Material information
Name: Student **Description:** Retrieve Material

Input Data Flows: Course Material Information

Output Data Flows: none

Data Process

Name: 6.1 Verify User ID

Description: Verifies the lecturer's identity

Input Data Flows: Material Information

Output Data Flows: Material with user ID

Process:

```
// Include database connection
INCLUDE "dbConn.php"

// Initialize variables
SET materialTitle = ""
SET materialDescription = ""
SET createMaterial = "display: none"

// Sanitize and assign material title and description from POST data
IF IS_SET($_POST['txttitle'])
   SET materialTitle = ESCAPE_STRING(connection, $_POST['txttitle'])

IF IS_SET($_POST['txtdescription'])
   SET materialDescription = ESCAPE_STRING(connection, $_POST['txtdescription'])

// Check button actions
IF IS_SET($_POST['btnCreateMaterial'])
   SET createMaterial = ""

ELSE IF IS_SET($_POST['btnCreateMaterial2'])
   // Insert new material into the database
   SET insertQuery1 = "INSERT INTO `material`(`Title`, `Description`, `ChanID`,
`Timestamp`) VALUES (materialTitle, materialDescription, channelID, NOW())"

   // Execute the insert query
```

```
    IF EXECUTE_QUERY(connection, insertQuery1)
        SET materialID = GET_LAST_INSERT_ID(connection)

        // Save uploaded document associated with the material
        SET result = SAVE_DOCUMENT($_FILES["material"], 'MaterialID', materialID,
connection, 'Material')

        // Check if document upload was successful
        IF result EQUALS "File uploaded successfully."
            OUTPUT "Create Successful!"
            REDIRECT "/projectify/lecturermaterial.php?channelID=" + channelID
        ELSE
            OUTPUT "Error: " + result

    ELSE
        OUTPUT "Error inserting channel: " + GET_ERROR(connection)

    SET createChannel = "display: none"

    // Include database connection
    INCLUDE "dbConn.php"

    // Retrieve user ID from session
    SET userID = $_SESSION["userid"]

    // Fetch user information
    SET fetchQuery1 = "SELECT * FROM `User` WHERE `UserID` = userID"
    SET result1 = EXECUTE_QUERY(connection, fetchQuery1)

    // Initialize variables
    SET userImageID = ""
    SET userImagePath = ""
    SET totalOfPending = 0
    SET totalOfChannel = 0
```

```
    SET totalOfMark = 0

    // Process user information query result
    IF result1 IS NOT NULL AND HAS_ROWS(result1)
        SET row = FETCH_ASSOC(result1)
        SET userImageID = HTML_SPECIALCHARS(row['ImageID'], ENT_QUOTES, 'UTF-8')

        // Fetch user image information
        SET fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = userImageID"
        SET result2 = EXECUTE_QUERY(connection, fetchQuery2)

        // Process user image information query result
        IF result2 IS NOT NULL AND HAS_ROWS(result2)
            SET row = FETCH_ASSOC(result2)
            SET userImagePath = HTML_SPECIALCHARS(row['Filepath'], ENT_QUOTES,
'UTF-8')

    // Fetch number of pending tasks
    SET fetchQuery3 = "
    SELECT COUNT(*) AS numberPending
    FROM `User_Channel` uc
    INNER JOIN `Task` t ON uc.ChanID = t.ChanID
    INNER JOIN `User_Task` ut ON ut.`TaskID` = t.`TaskID`
    INNER JOIN `Appendix` a ON a.`AppendixID` = ut.`AppendixID`
    INNER JOIN `Channel` c ON c.`ChanID` = t.`ChanID`
    WHERE c.`OwnerID` = userID AND ut.`Score` IS NOT NULL AND t.`TaskID` IS NOT
NULL
    "
    SET result3 = EXECUTE_QUERY(connection, fetchQuery3)

    // Process number of pending tasks query result
    IF result3 IS NOT NULL
        SET row = FETCH_ASSOC(result3)
        SET totalOfPending = row['numberPending']
```

```
// Fetch number of channels owned by user
SET fetchQuery4 = "
SELECT COUNT(*) AS channelNumber
FROM `Channel`
WHERE `OwnerID` = userID
"
SET result4 = EXECUTE_QUERY(connection, fetchQuery4)

// Process number of channels query result
IF result4 IS NOT NULL
   SET row = FETCH_ASSOC(result4)
   SET totalOfChannel = row['channelNumber']

// Fetch number of task submissions pending evaluation
SET fetchQuery5 = "
SELECT COUNT(*) AS numberSubmission
FROM `User_Channel` uc
INNER JOIN `Task` t ON uc.ChanID = t.ChanID
INNER JOIN `User_Task` ut ON ut.`TaskID` = t.`TaskID`
INNER JOIN `Appendix` a ON a.`AppendixID` = ut.`AppendixID`
INNER JOIN `Channel` c ON c.`ChanID` = t.`ChanID`
WHERE c.`OwnerID` = userID AND ut.`Score` IS NULL AND t.`TaskID` IS NOT NULL
"
SET result5 = EXECUTE_QUERY(connection, fetchQuery5)

// Process number of task submissions query result
IF result5 IS NOT NULL
   SET row = FETCH_ASSOC(result5)
   SET totalOfMark = row['numberSubmission']
```

Name: 6.2 Get Channel ID

Description: Retrieves the channel ID link with the verified user material

Input Data Flows: Material with user ID

Output Data Flows: User material with channel ID

Process:

```
  // Start session to manage user data
START_SESSION()

// Get channelID from URL parameter
channelID = GET_PARAMETER('channelID')

// Include necessary PHP files for functions and data retrieval
INCLUDE "php/function.php"
INCLUDE "php/createMaterial.php"
INCLUDE "php/fetchChannel.php"
INCLUDE "php/fetchDocument.php"

// HTML starts here
DISPLAY_HTML_START()

// HTML head section with meta tags, stylesheets, and scripts
HTML_HEAD_START()
   SET CHARSET_UTF8()
   SET VIEWPORT_WIDTH_DEVICE()
   ADD_EXTERNAL_STYLESHEET("https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.2/css/all.min.css")
   ADD_EXTERNAL_STYLESHEET("https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css")

ADD_EXTERNAL_SCRIPT("https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js")
   SET_TITLE("Channel")
   ADD_INTERNAL_STYLESHEET("css/channel.css")
   ADD_INTERNAL_STYLESHEET("css/lecturermaterial.css")
   ADD_INTERNAL_SCRIPT("javascript/function.js", DEFERRED)
```

```
HTML_HEAD_END()

// HTML body section
HTML_BODY_START()
   // Main container for the page
   DIV_START(class="main")

      // Sidebar/Choice Section
      DIV_START(class="choice")
         // Back button to redirect to lecturerchannel.php
         BUTTON_ONCLICK_REDIRECT("lecturerchannel.php", ID="backbutton")
            IMG(class="back", SRC="image/back.png")
         BUTTON_END()

         // Projectify title and logo
         DIV(class="title")
            IMG(SRC="image/whitelogo.png")
            H1("Projectify")
         DIV_END()

         // User profile photo
         DIV(class="photo")
            IMG(class="profile-frame", SRC=imagePath)
         DIV_END()

         // Channel information
         P(class="course", TEXT=channelName)
         P(class="info", TEXT="Lecturer: " + userName)
         P(class="info", TEXT="Channel Code: " + channelCode)
         P(class="info", TEXT="Student: " + totalOfMember)

         // Divider line
         DIV(class="line")
```

```
    // Navigation links
    A_HREF("lecturermaterial.php?channelID=" + channelID)
      IMG(SRC="image/controlpanel.png")
      TEXT("Material")
    A_END()

    A_HREF("lecturerassignment.php?channelID=" + channelID)
      IMG(SRC="image/classroom.png")
      TEXT("Assignment")
    A_END()

    A_HREF("lecturermember.php?channelID=" + channelID)
      IMG(SRC="image/member.png")
      TEXT("Member")
    A_END()
  DIV_END()

  // Main panel for displaying materials and other content
  DIV_START(class="panel")
    // Form and button to create material
    DIV_START(class="material1")
      FORM(action="", method="POST")
        BUTTON(name="btnCreateMaterial")
          IMG(class="export", SRC="image/export.png")
        BUTTON_END()
        // Include PHP script to fetch and display documents/materials
        INCLUDE "php/fetchDocument.php"
      FORM_END()
    DIV_END()
  DIV_END()

  // Container for creating material (initially hidden)
  DIV_STYLE_DISPLAY(createMaterial)
    A_HREF("javascript:history.go(-1);", class="back-icon")
```

```
            I(class="fas fa-arrow-left")
        A_END()
        // Form to upload material
        FORM(action="", method="POST", enctype="multipart/form-data")
            H1("Upload Course Material")
            SPAN("Design and Label Your Material")
            INPUT(type="file", name="material", required)
            INPUT(name="txttitle", placeholder="Material Title", required)
            TEXTAREA(name="txtdescription", placeholder="Material Description", required)
            BUTTON(name="btnCreateMaterial2")
                TEXT("Upload Material")
        FORM_END()

    DIV_END()
HTML_BODY_END()

// End of HTML document
HTML_END()
```

Name: 6.3 Process Channel Material

Description: Processes handles the core process of the material the user task based on the provided channel ID

Input Data Flows: User Material with Channel ID

Output Data Flows: Material Data

Process:

```
 // Include database connection file
INCLUDE "dbConn.php"

// Initialize global connection variable
global $connection

// Retrieve material title from POST data, escape for database
materialTitle = IF IS_SET($_POST['txttitle']) THEN ESCAPE_STRING(connection,
$_POST['txttitle']) ELSE ""
```

```
// Retrieve material description from POST data, escape for database
materialDescription = IF IS_SET($_POST['txtdescription']) THEN
ESCAPE_STRING(connection, $_POST['txtdescription']) ELSE ""

// Initialize variable to control material creation form visibility
createMaterial = "display: none"

// Check if the create material button is clicked
IF IS_SET($_POST['btnCreateMaterial'])
   SET createMaterial = ""

// Check if the form for uploading material is submitted
ELSE IF IS_SET($_POST['btnCreateMaterial2'])
   // Construct query to insert new material into database
   insertQuery1 = "INSERT INTO `material`(`Title`, `Description`, `ChanID`, `Timestamp`)
VALUES (materialTitle, materialDescription, channelID, NOW())"

   // Execute query to insert material data
   IF EXECUTE_QUERY(connection, insertQuery1)
      // Get the auto-generated ID of the inserted material
      materialID = GET_INSERT_ID(connection)

      // Call function to save uploaded document
      result = saveDocument($_FILES["material"], 'MaterialID', materialID, connection,
'Material')

      // Check if document upload was successful
      IF result EQUALS "File uploaded successfully."
         // Display success message and redirect to material page
         OUTPUT "Create Successful!"
         REDIRECT "/projectify/lecturermaterial.php?channelID=" + channelID
      ELSE
         // Display error message if document upload failed
```

```
        OUTPUT "Error: " + result
    ELSE
        // Display error message if material insertion into database fails
        OUTPUT "Error inserting material: " + GET_ERROR(connection)

        // Hide material creation form after error
        SET createMaterial = "display: none"
ENDIF
```

Name: 6.4 Fetch Material by ID

Description: Retrieves the material record based on the course material provided by the student.

Input Data Flows: Material Record

Output Data Flows: Course Material

Process:

```
// Include database connection file
INCLUDE "dbConn.php"

// Initialize global variables
channelID, channelName, imagePath = EMPTY_STRING

// Fetch channel details based on channel ID
fetchQuery1 = "SELECT * FROM `Channel` WHERE `ChanID` = channelID"
result1 = EXECUTE_QUERY(connection, fetchQuery1)

// Check if channel details are fetched successfully
IF result1 IS NOT NULL AND HAS_ROWS(result1)
    // Retrieve channel information
    row = FETCH_ASSOC(result1)
    channelID = HTML_SPECIALCHARS(row['ChanID'], ENT_QUOTES, 'UTF-8')
    channelCode = HTML_SPECIALCHARS(row['ChannelCode'], ENT_QUOTES, 'UTF-8')
    channelName = HTML_SPECIALCHARS(row['Title'], ENT_QUOTES, 'UTF-8')
    imageID = HTML_SPECIALCHARS(row['ImageID'], ENT_QUOTES, 'UTF-8')
    ownerID = HTML_SPECIALCHARS(row['OwnerID'], ENT_QUOTES, 'UTF-8')
```

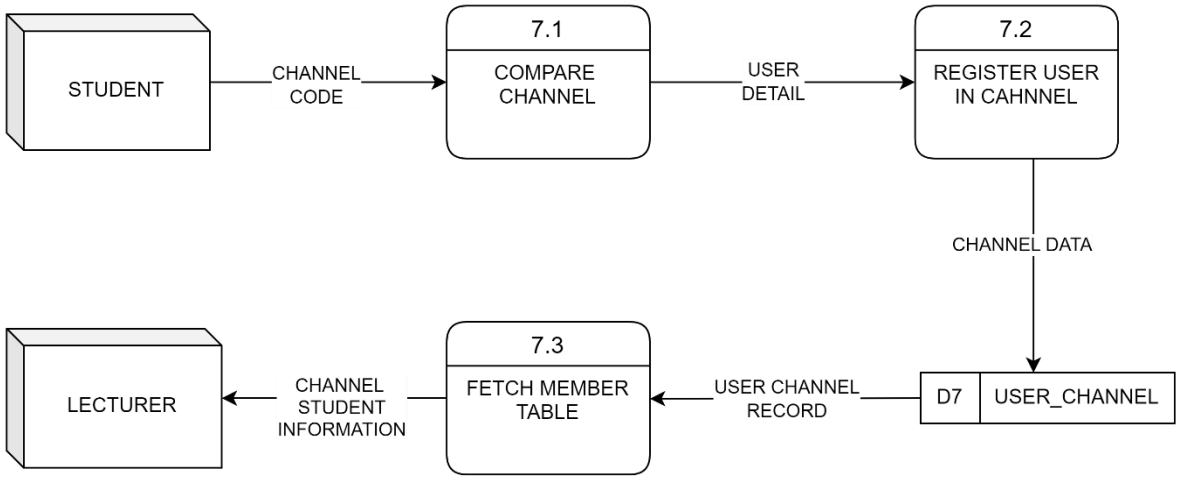
```
    // Fetch image path associated with the channel
    fetchQuery2 = "SELECT * FROM `Image` WHERE `ImageID` = imageID"
    result2 = EXECUTE_QUERY(connection, fetchQuery2)

    // Check if image path is fetched successfully
    IF result2 IS NOT NULL AND HAS_ROWS(result2)
        row = FETCH_ASSOC(result2)
        imagePath = HTML_SPECIALCHARS(row['Filepath'], ENT_QUOTES, 'UTF-8')

    // Fetch owner's name associated with the channel
    fetchQuery3 = "SELECT * FROM `User` WHERE `UserID` = ownerID"
    result3 = EXECUTE_QUERY(connection, fetchQuery3)

    // Check if owner's name is fetched successfully
    IF result3 IS NOT NULL AND HAS_ROWS(result3)
        row = FETCH_ASSOC(result3)
        userName = HTML_SPECIALCHARS(row['Name'], ENT_QUOTES, 'UTF-8')

    // Query to count total members in the channel
    query = "SELECT COUNT(*) as total FROM `User_Channel` WHERE `ChanID` =
channelID"
    result = EXECUTE_QUERY(connection, query)

    // Check if total members count is fetched successfully
    IF result IS NOT NULL
        row = FETCH_ASSOC(result)
        totalOfMember = row['total']
```

Data Store
Name: D6 Material **Description:** Store Data Material **Input Data Flows:** Material Data **Output Data Flows:** Material Record

dData Structure:
Title+Description+Filename+Filetype+Filesize+Filecontent+Timestamp+ChanID

Data Flow
Name: Material Information **Description:** Information about the material provided by the lecturer **Origin:** Lecture external entities **Destination:** 6.1 Verify Host ID **Data Structure:** Title + Description + Deadline
Name: Material with user ID **Description:** Material information tagged with the verified user ID. **Origin:** 6.1 Verify Host ID **Destination:** 6.2 Get Channel ID **Data Structure:** Channe ID+ MemberID
Name: User Material with Channel ID **Description:** Material information associated with a specific channel ID. **Origin:** 6.2 Get Channel ID **Destination:** 6.3 Process User Task **Data Structure:** TaskID + AppendixID
Name: Material Data **Description:** Processed material information ready for storage. **Origin:** 6.3 Process Channel Material **Destination:** D6 Material **Data Structure:** Title + Description + Filename + Filetype + Filesize + Filecontent + Timestamp + ChanID
Name: Material Record **Description:** Material Information provided by the student. **Origin:** D6 Material **Destination:** 6.4 **Data Structure:** TaskID + AppendixID + Status + Score
Name: Course Material **Description:** Material Information provided by the student. **Origin:** 6.4 Fetch Material by ID

Destination: Student **Data Structure:** Title + Description + Filename + Filetype + Filesize + Filecontent + Timestamp + ChanID



External Entity
Name: Student **Description:** **Input Data Flows:** none **Output Data Flows:** Channel code
Name: Lecture **Description:** Retrieve Student information **Input Data Flows:** Channel Student information **Output Data Flows:** none

Data Process

Name: 7.1 Compare Channel

Description:

Input Data Flows: Channel Code

Output Data Flows: User Detail

Process:

```
BEGIN
   INCLUDE "dbConn.php"
   DECLARE $channelCode, $userID, $createChannel
   INITIALIZE $createChannel = "display: none"

   // Check if the form for joining a channel was submitted
   IF FORM_SUBMITTED AND btnJoinChannel CLICKED THEN
      SET $createChannel = "" // Show join channel form
   ELSE IF FORM_SUBMITTED AND btnJoinChannel2 CLICKED THEN
      // Retrieve channel code from the form input
      $channelCode = ESCAPE_SPECIAL_CHARACTERS($_POST['txtcode'])

      // Retrieve user ID from session
      $userID = GET_SESSION_VALUE("userid")

      // Construct query to fetch channel details based on channel code
      $fetchQuery1 = "SELECT * FROM Channel WHERE ChannelCode =
'$channelCode'"

      // Execute the fetch query
      EXECUTE_QUERY($fetchQuery1, $result1)

      // Check if the query was successful and fetch the channel details
      IF $result1 IS SUCCESSFUL THEN
         IF ROWS_FETCHED($result1) > 0 THEN
            FETCH_ASSOCIATED_ROW($result1, $row)

            // Extract channel ID from fetched data
            $chanID = ESCAPE_SPECIAL_CHARACTERS($row['ChanID'])
```

```
            // Check if the user is already a member of the channel
            $checkQuery = "SELECT * FROM User_Channel WHERE ChanID = '$chanID'
AND MemberID = '$userID'"

            // Execute the membership check query
            EXECUTE_QUERY($checkQuery, $checkResult)

            // Check if the user is already a member
            IF $checkResult IS SUCCESSFUL AND ROWS_FETCHED($checkResult) > 0
THEN
               DISPLAY_ALERT("You are already a member of this channel.")
            ELSE
               // Insert user into the channel as a member
               $insertQuery = "INSERT INTO User_Channel(ChanID, MemberID)
VALUES ('$chanID','$userID')"

               // Execute the insertion query
               EXECUTE_QUERY($insertQuery, $insertResult)

               // Check if insertion was successful
               IF $insertResult IS SUCCESSFUL THEN
                  DISPLAY_ALERT("Joined channel successfully!")
                  REDIRECT_TO("/projectify/")
               ELSE
                  DISPLAY_ALERT("Error joining channel: " + GET_MYSQL_ERROR())
               END IF
            END IF
         ELSE
            DISPLAY_ALERT("Channel not found. Please check the channel code.")
         END IF
      ELSE
         DISPLAY_ALERT("Error: " + GET_MYSQL_ERROR())
      END IF
```

```
    END IF
END
```

Name: 7.2 Register User in Channel

Description: The user detail will be register in the user channel

Input Data Flows: User Detail

Output Data Flows: Channel Data

Process:

```
// Include database connection
INCLUDE "dbConn.php"

// Retrieve channel code from POST data or set to empty string if not set
channelCode = IF IS_SET($_POST['txtcode']) THEN ESCAPE_STRING(connection,
$_POST['txtcode']) ELSE ""

// Retrieve user ID from session
userID = $_SESSION["userid"]

// Initialize display state for create channel form
createChannel = "display: none"

// Check if 'Join Channel' button is clicked
IF IS_SET($_POST['btnJoinChannel'])
    createChannel = ""

// Check if 'Join Channel 2' button is clicked
ELSE IF IS_SET($_POST['btnJoinChannel2'])
    // Fetch channel details based on provided channel code
    fetchQuery1 = "SELECT * FROM Channel WHERE ChannelCode = channelCode"
    result1 = EXECUTE_QUERY(connection, fetchQuery1)
```

```
   // Check if query was successful and fetch the channel ID
   IF result1 NOT NULL AND HAS_ROWS(result1)
      row = FETCH_ASSOC(result1)
      chanID = HTML_SPECIALCHARS(row['ChanID'], ENT_QUOTES, 'UTF-8')

      // Insert user into the channel
      insertQuery1 = "INSERT INTO user_channel (ChanID, MemberID) VALUES
(chanID, userID)"

      // Execute the insert query
      IF EXECUTE_QUERY(connection, insertQuery1)
         OUTPUT "Join Successful!"
         REDIRECT "/projectify/studentchannel.php"
      ELSE
         OUTPUT "Error inserting task: " + GET_ERROR(connection)

      // Hide create channel form
      createChannel = "display: none"
   ENDIF
ENDIF
```

Name: 7.3 Fetch Member Table

Description: Fetch Member table to get member information

Input Data Flows: User Channel Record

Output Data Flows: Channel Student Information

Process:

```
// Retrieve channel ID from GET parameter and user ID from session
channelID = GET['channelID']
userID = SESSION['userid']

// Construct SQL query to fetch student details
fetchQuery = "
SELECT
```

```
    ui.`UserID` as StudentID,
    i.`Filepath` as StudentImage,
    u.`Name` as StudentName,
    u.`IC` as StudentIC,
    u.`Email` as StudentEmail,
    it.`Name` as StudentInstitution
FROM `User_Channel` uc
INNER JOIN `Channel` c ON c.`ChanID` = uc.`ChanID`
INNER JOIN `User` u ON uc.`MemberID` = u.`UserID`
INNER JOIN `User_Institution` ui ON ui.`UserID` = u.`UserID`
INNER JOIN `Institution` it ON it.`InstitutionID` = ui.`InstitutionID`
INNER JOIN `Image` i ON i.`ImageID` = u.`ImageID`
WHERE c.`OwnerID` = userID AND c.`ChanID` = channelID
"

// Execute query to fetch results from database
result = EXECUTE_QUERY(connection, fetchQuery)

// Check if query was successful
IF result IS NOT NULL
    // Output table headers
    OUTPUT '
        <table class="tasktable">
            <tr>
                <th>Student Image</th>
                <th>Student Name</th>
                <th>Student IC</th>
                <th>Student Email</th>
                <th>Institution</th>
                <th>Status</th>
            </tr>
    '

    // Iterate through result set and output each row
```

```
    WHILE row = FETCH_ASSOC(result)
        // Sanitize and retrieve values
        studID = HTML_SPECIALCHARS(row['StudentID'], ENT_QUOTES, 'UTF-8')
        studImage = HTML_SPECIALCHARS(row['StudentImage'], ENT_QUOTES, 'UTF-
8')
        studName = HTML_SPECIALCHARS(row['StudentName'], ENT_QUOTES, 'UTF-
8')
        studIC = HTML_SPECIALCHARS(row['StudentIC'], ENT_QUOTES, 'UTF-8')
        studEmail = HTML_SPECIALCHARS(row['StudentEmail'], ENT_QUOTES, 'UTF-
8')
        studInstitution = HTML_SPECIALCHARS(row['StudentInstitution'],
ENT_QUOTES, 'UTF-8')

        // Output table row with student details and delete button
        OUTPUT '
            <tr>
                <td><img src="' + studImage + '"></td>
                <td>' + studName + '</td>
                <td>' + studIC + '</td>
                <td>' + studEmail + '</td>
                <td>' + studInstitution + '</td>
                <td>
                    <form action="php/deleteMember.php" method="POST">
                        <div class="columndisplay">
                            <input type="hidden" name="studID" value="' + studID + '" />
                            <input type="hidden" name="chanID" value="' + channelID + '" />
                            <input type="submit" name="btnDelete" value="Delete" />
                        </div>
                    </form>
                </td>
            </tr>
        '

    // Close the table
```

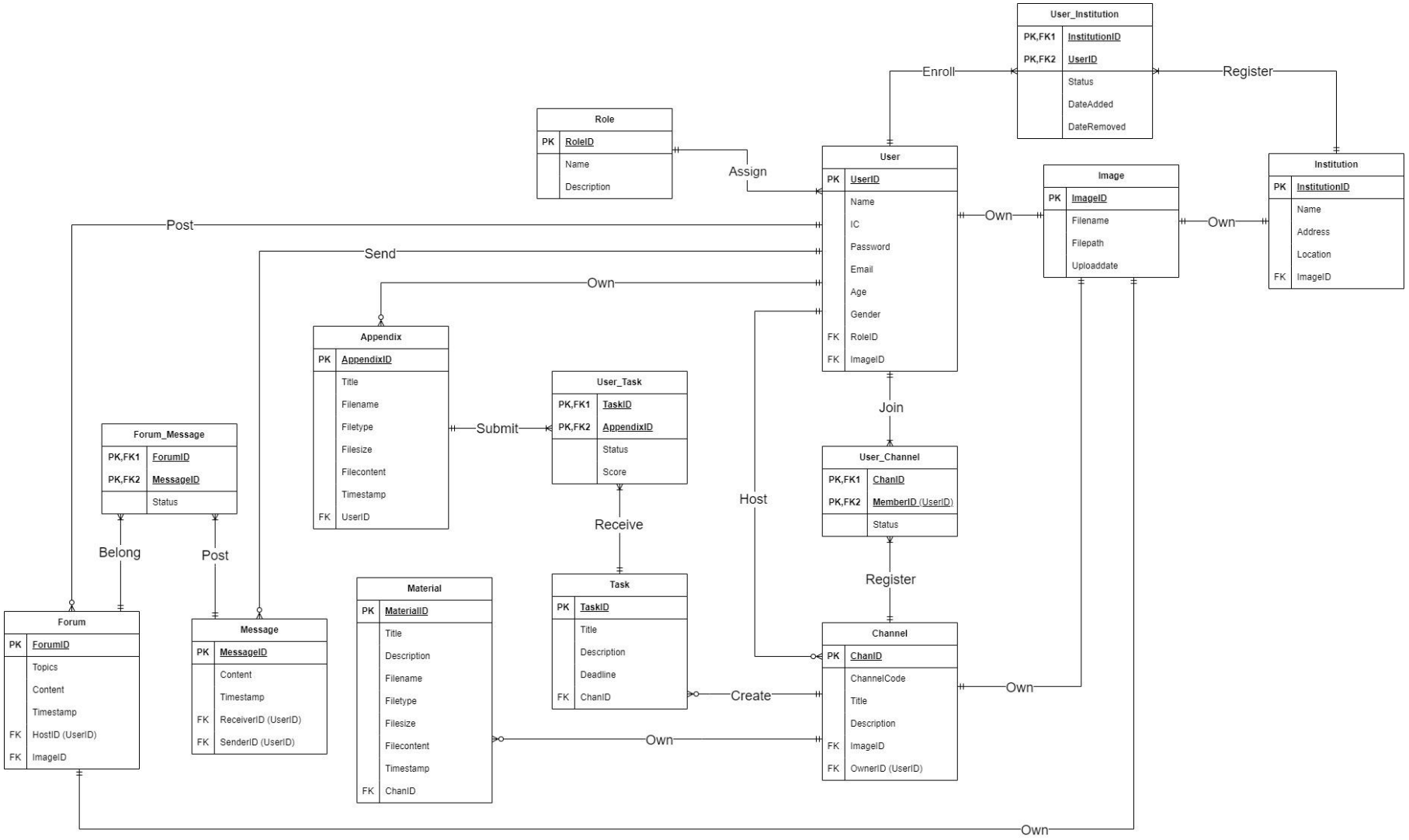
```
OUTPUT '</table>'
```

Data Store
Name: D7 User Channel **Description:** Store Channel Data **Input Data Flows:** Channel Data **Output Data Flows:** User Channel Record **Data Structure:** ChanID + ChannelCode + Title + Description + ImageID + OwnerID

Data Flow
Name: Channel Code **Description:** A code will be provided by the student to join a channel. **Origin:** Student external entities **Destination:** 7.1 Compare Channel **Data Structure:** ChannelCode
Name: User Detail **Description:** User information **Origin:** 7.1 Compare Channel **Destination:** 7.2 Register User In Channel **Data Structure:** UserID
Name: Channel Data **Description:** Data relevant to the channel and on how user register in it **Origin:** 7.2 Register User In Channel **Destination:** D7 User Channel **Data Structure:** ChanID + ChannelCode + Title + Description + ImageID + OwnerID
Name: User Channel Record

Description: Record of user channel relationship **Origin:** D7 User Channel **Destination:** 7.3 Fetch Member Table **Data Structure:** ChanID + MemberID + Status
Name: Channel Student Information **Description:** Contain student information **Origin:** 7.3 Fetch Member Table **Destination:** Lecturer **Data Structure:** UserID + Filepath + Name + IC + Email + Institution

11.0 Entity Relationship Diagram



12.0 Screens Design, Report Design and User Manual

12.1 User Register Manual

This is the main page our Projectify system. New user of Projectify system will need to register an account using their email address. First, user will need to click the log in and sign in button located on the top right corner of the main page.

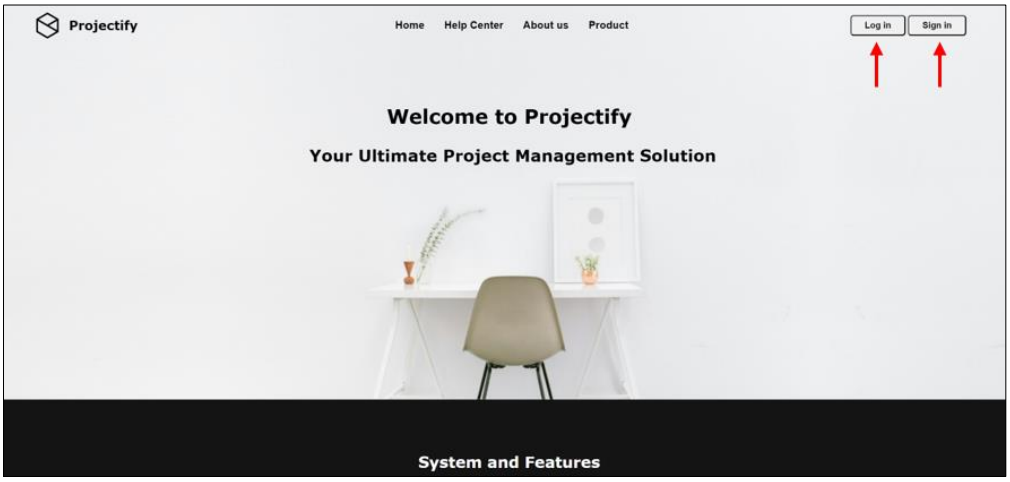


Figure 12.1.1 – Home Page

Then, user will be redirected to the login page, there will be 2 sections, one is “Sign In” and another one is “Sign Up”. But for new users, they would have to Sign Up and register for an account. The “Sign Up” section will be on the right side of the panel.

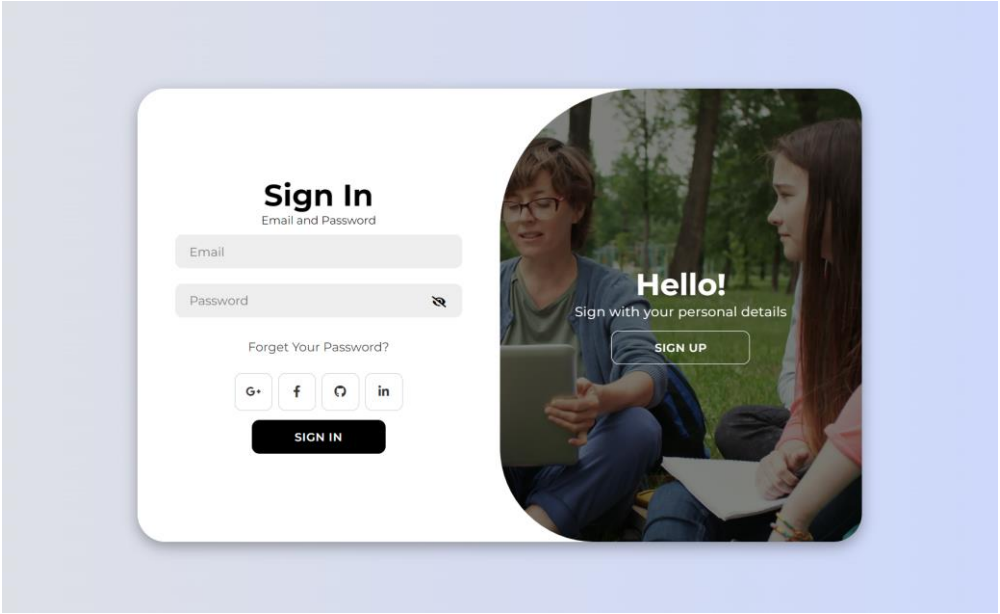


Figure 12.1.2 – Sign In Page

On the “Sign Up” page, they are required to fill in the username, email address and the password. Then, click the black "SIGN UP" button to proceed.

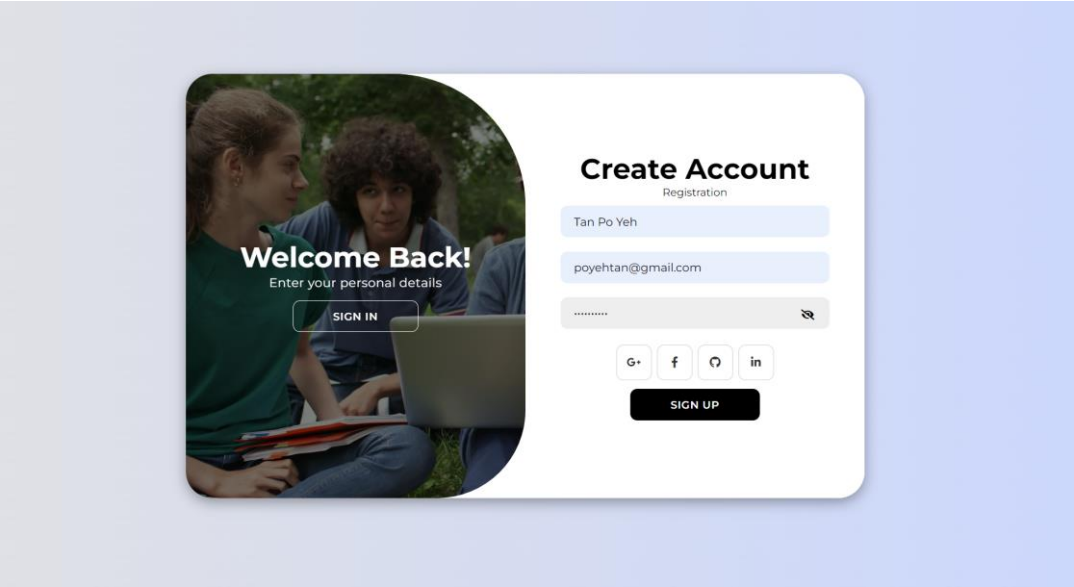


Figure 12.1.3 – Sign Up Page

Next, user is required to provide all information such as “Profile Picture”, “IC Number”, “Age”, and etc. After that, user can click “Sign Up” to complete the registration. The arrow on the top left corner also allows users to return back to the page for refilling their personal information.

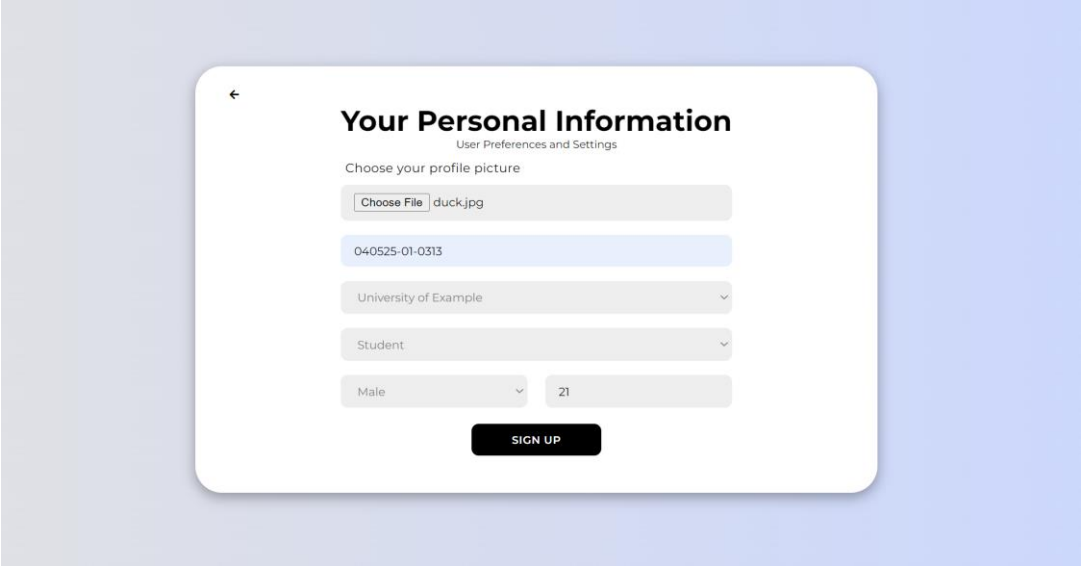


Figure 12.1.4 – Sign Up Page

Once it is successful, the system will display a success message to the user just like the picture shown below.

Figure 12.1.5

12.2 User Login Manual

Let's say already have an account. They can login into the system by locating at the top right "Sign In" of the main page. Then, it will redirect user to the "Sign In" section.

Figure 12.2.1 – Header

User is required to enter their email address and password to access his own registered account.

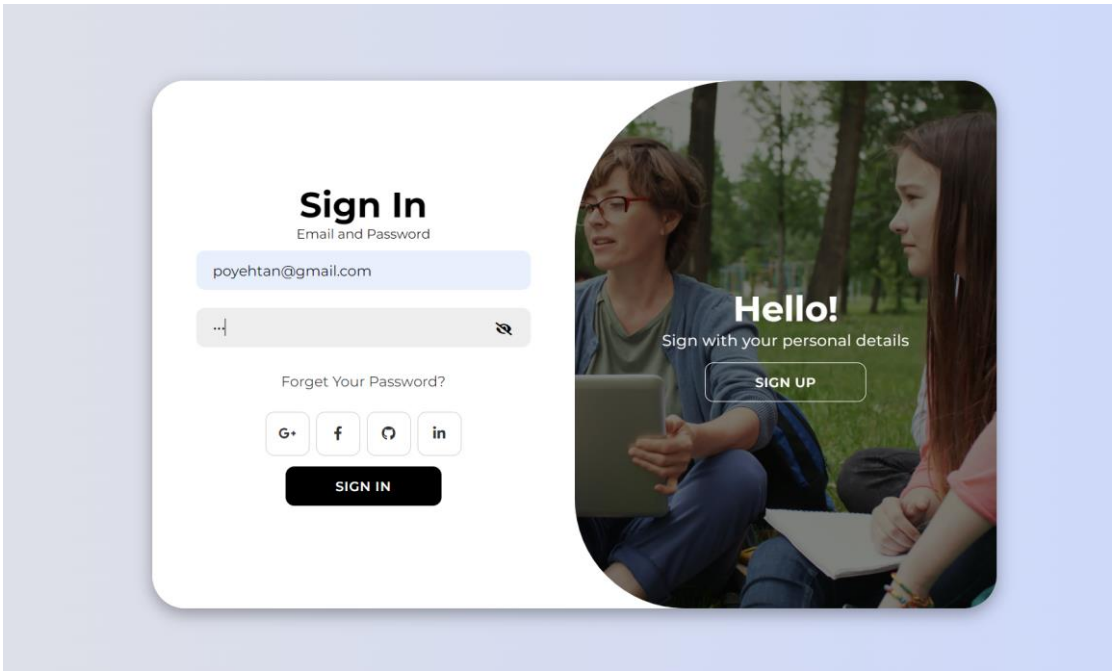


Figure 12.2.2 – Sign In Panel

Once success, system will display success message to the user and direct user to the page according to their role assigned in the database.

Figure 12.2.3 – Header

12.3 Student Manual

12.3.1 Student Channels Manual

Student dashboard that shows the logged-in student's joined channel. To join a channel, the student will have to click on the top right corner or the page that has a "+" icon.

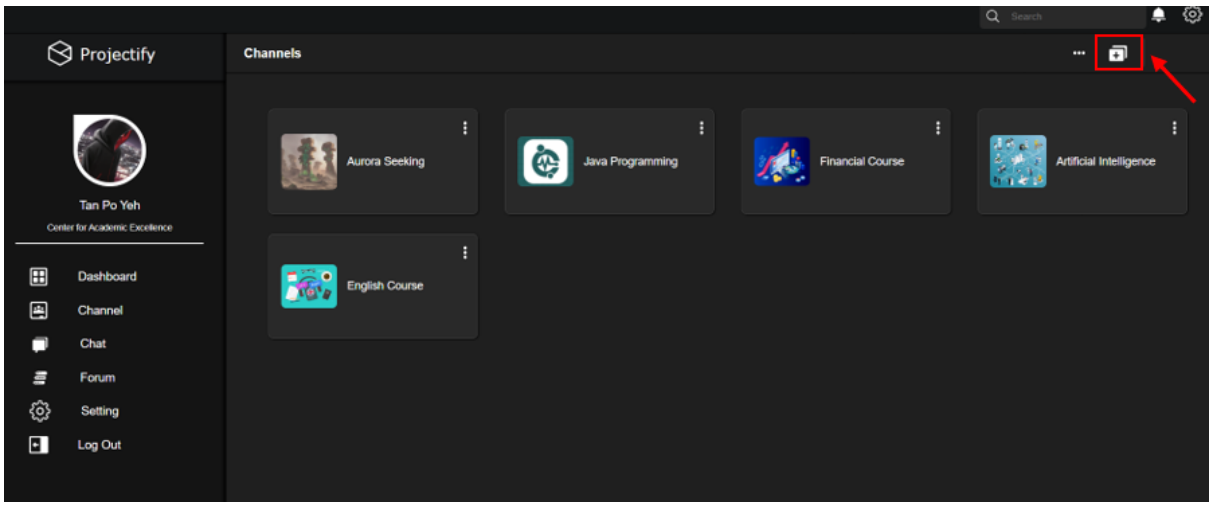


Figure 12.3.1 – Student Manual

A pop-up panel will appear for students to enter the channel's code that has provided by their lecturer. A message will appear after the join is successful.

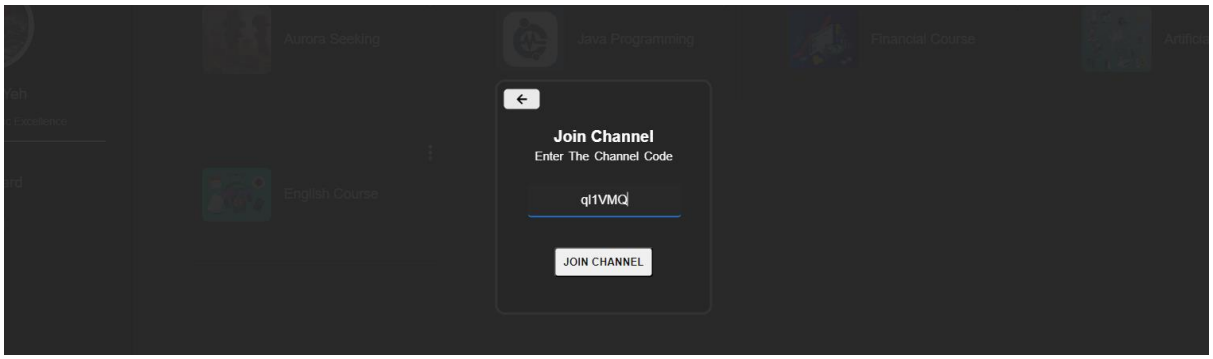


Figure 12.3.2 – Student Manual

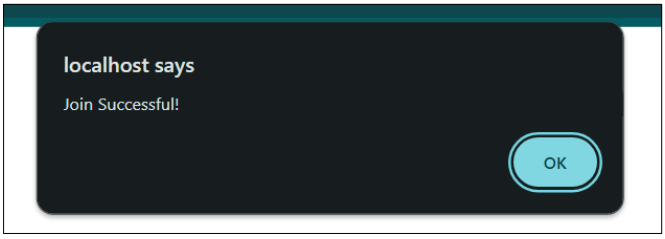


Figure 12.3.3 – Student Manual

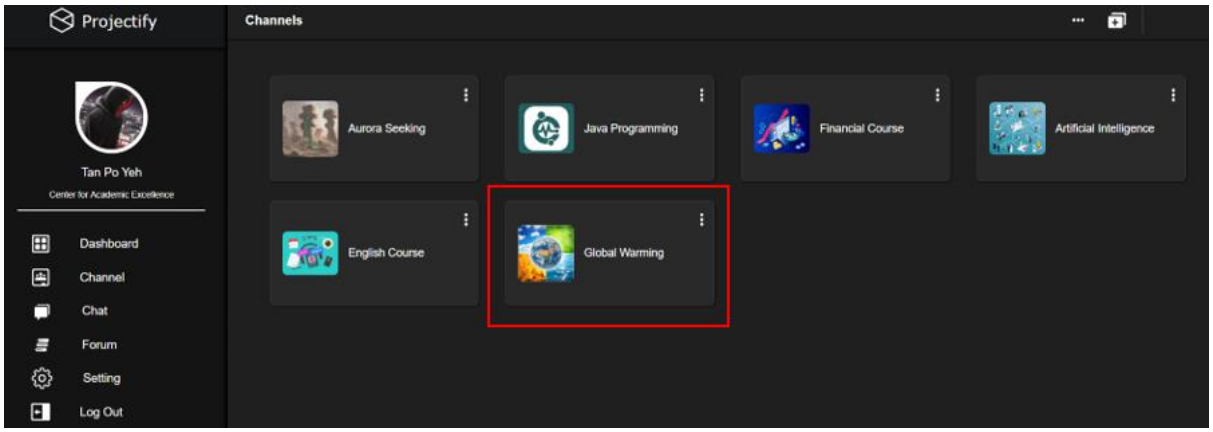


Figure 12.3.4 – Student Manual

12.3.2 Student Material Manual & Student Assignment Manual

After clicking into the one of the channels, there will be another 2 sections which is Material and Assignment. There is also a back button on the top left corner that allows user to return back to the previous page.

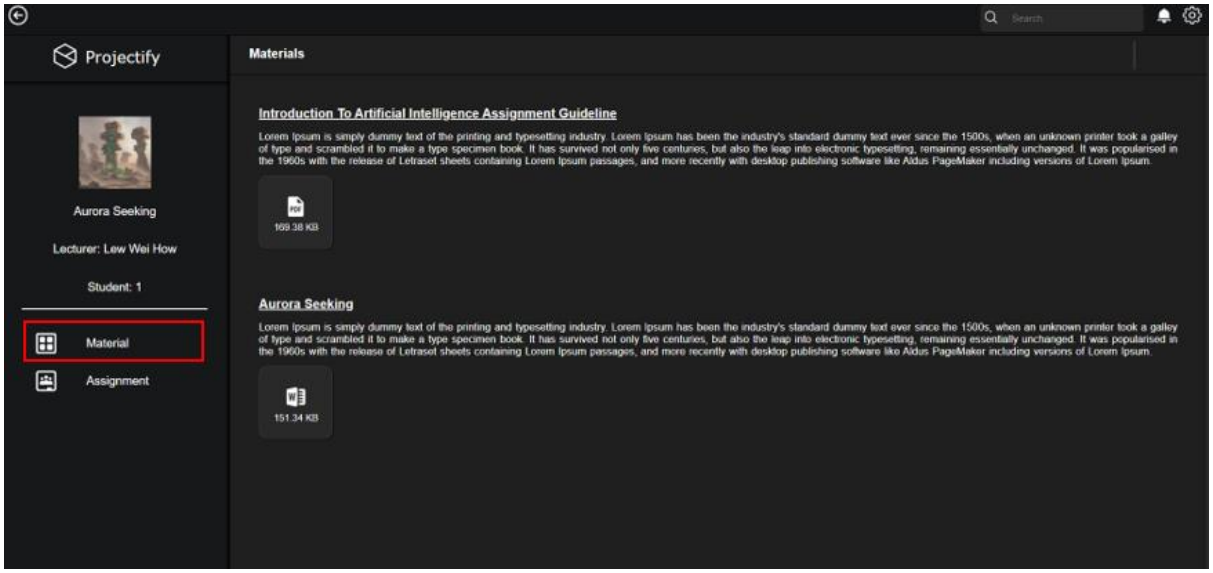


Figure 12.3.5 – Student Manual

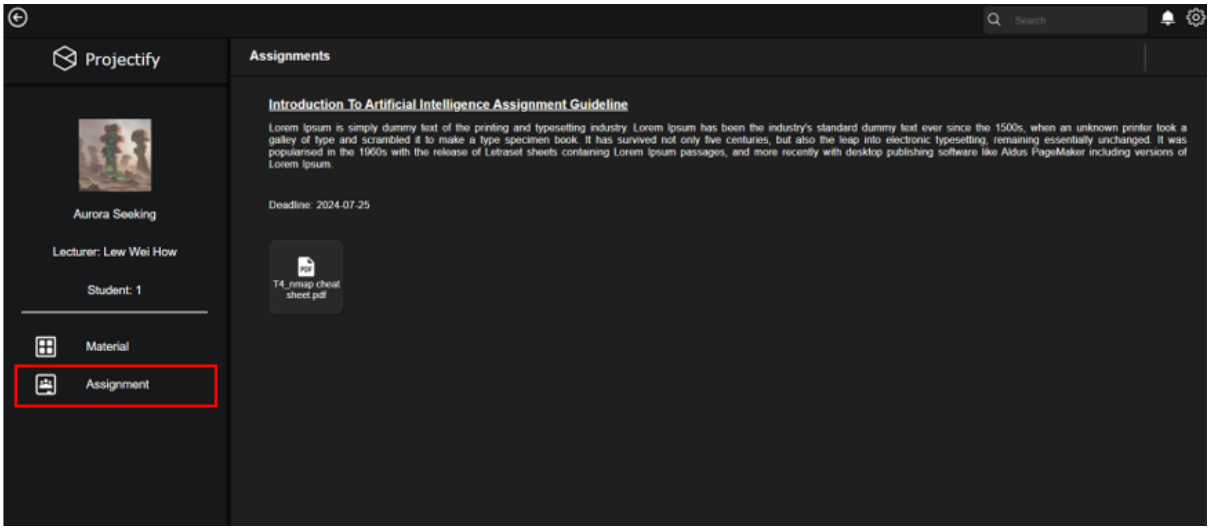


Figure 12.3.6 – Student Manual

12.4 Lecturer Manual

12.4.1 Lecturer Add Channels Manual

Lecturers can click the "+" icon on the top right corner of the panel to create a channel for students. After clicking, there will be a pop-up panel appear on the screen for lecturers to create a channel.

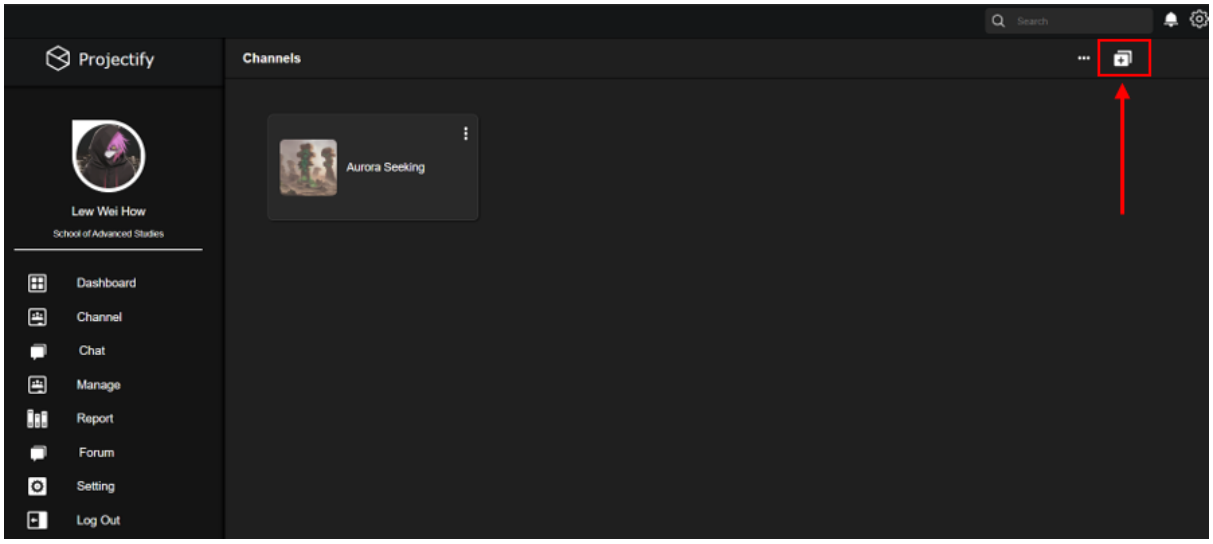


Figure 12.4.1 – Lecturer Manual

There are 3 fields that requires the lecturer to fill in for creating a channel which is “Choose File” to upload the channel banner or image, “Channel’s Name”, and “Channel’s Description”. After that, lecturer will have to click on the “create” button below the fields to create the channel. Other than that, there will be a button on the top left corner of the panel that allows lecturer to return back to the previous page.

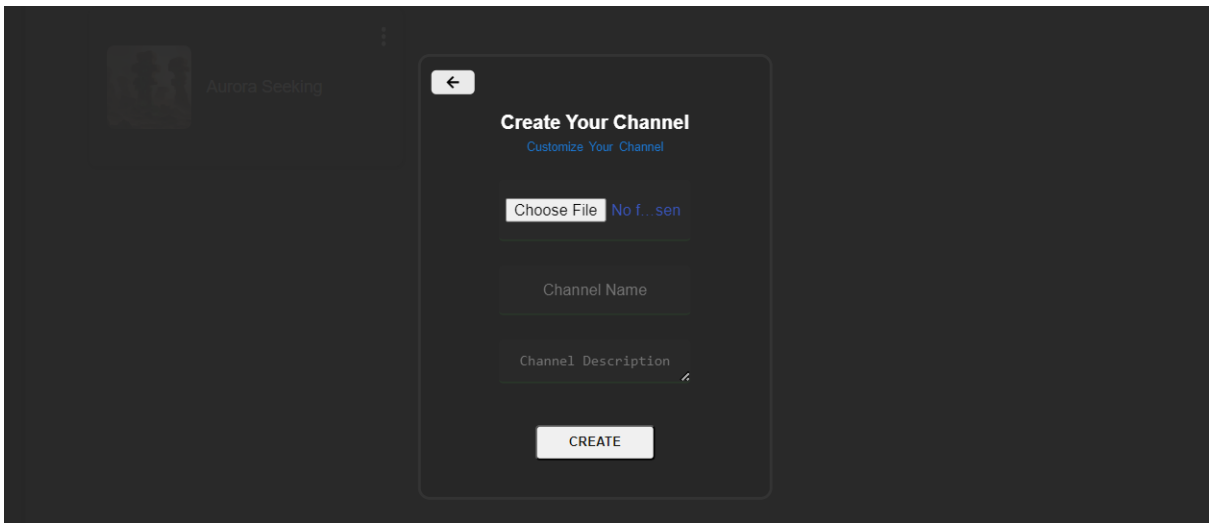


Figure 12.4.2 – Lecturer Manual

A message will appear if lecturers have successfully created the channel.

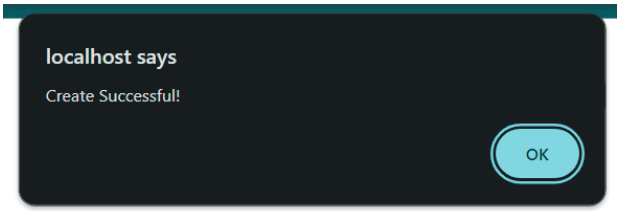


Figure 12.4.3 – Lecturer Manual

As you can see that the channel has been created.

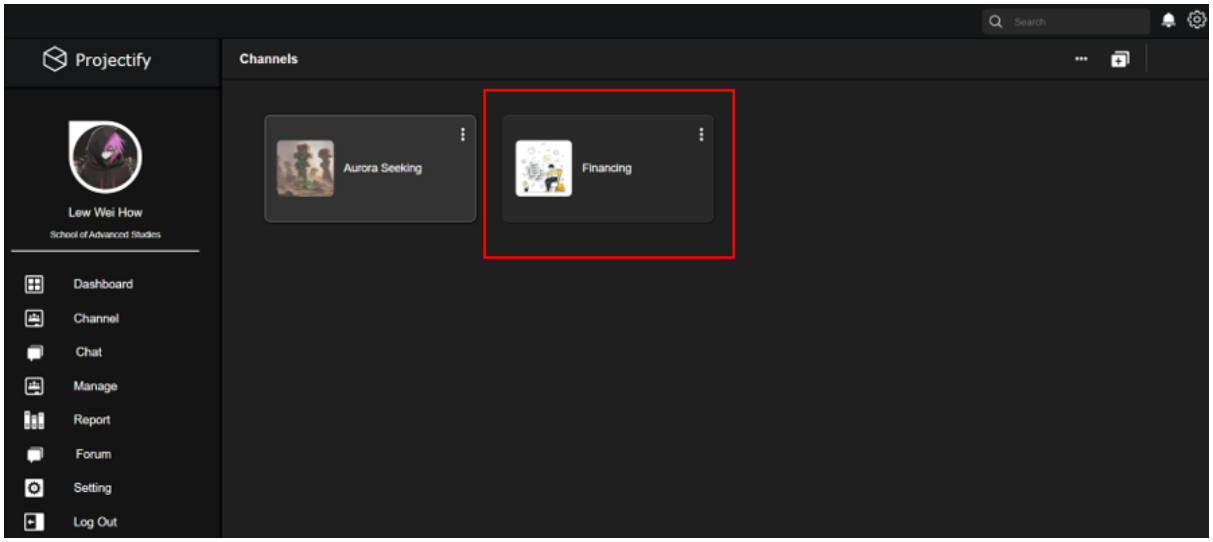


Figure 12.4.4 – Lecturer Manual

12.4.2 Lecturer Manage Materials

Lecturers would have to click on one of the channels if they wanted to add materials for the channel.

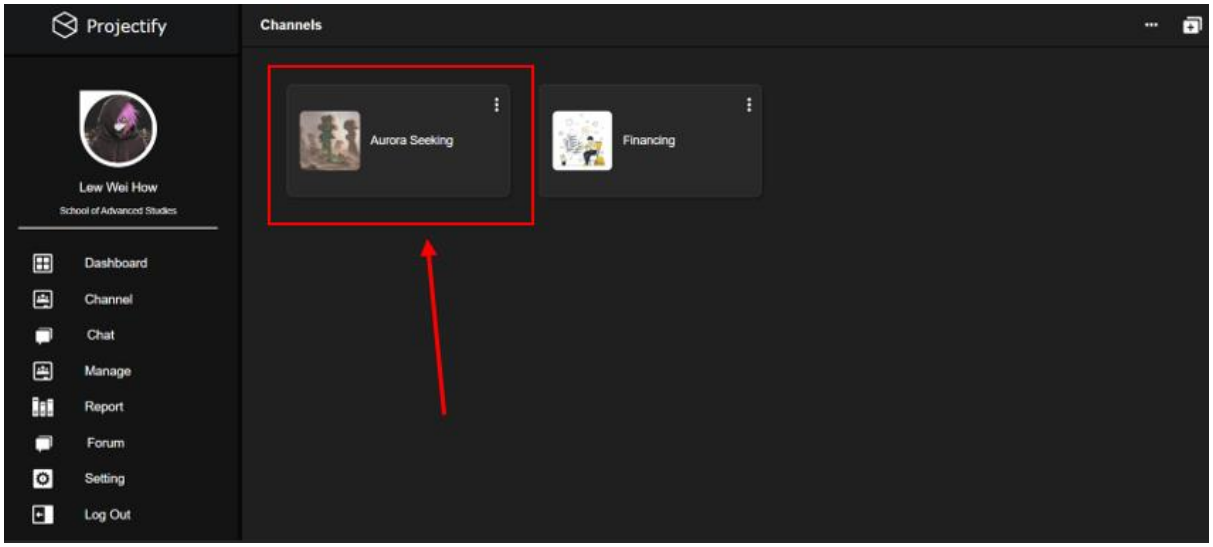


Figure 12.4.5 – Lecturer Manual

After that, there is a button(icon) at the top right corner of the material's panel. Lecturers will have to click at that panel to create/add the course materials for the channel they have selected.

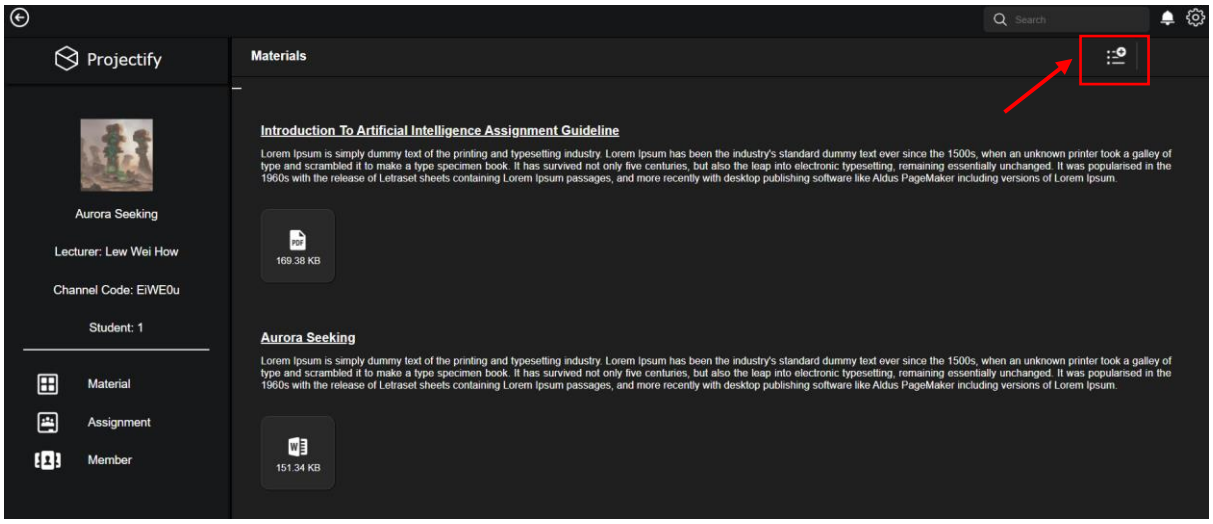


Figure 12.4.6 – Lecturer Manual

A panel will pop up after clicking of the button(icon). This panel requires lecturer to fill in all fields to upload a course material which is "File" for course file, "Material Title", "Material Description". After that, lecturer will have to click on the button below the panel to upload it.

There will also be a back button at the top left corner of the panel to redirect the lecturer if they want to go back to the previous section.

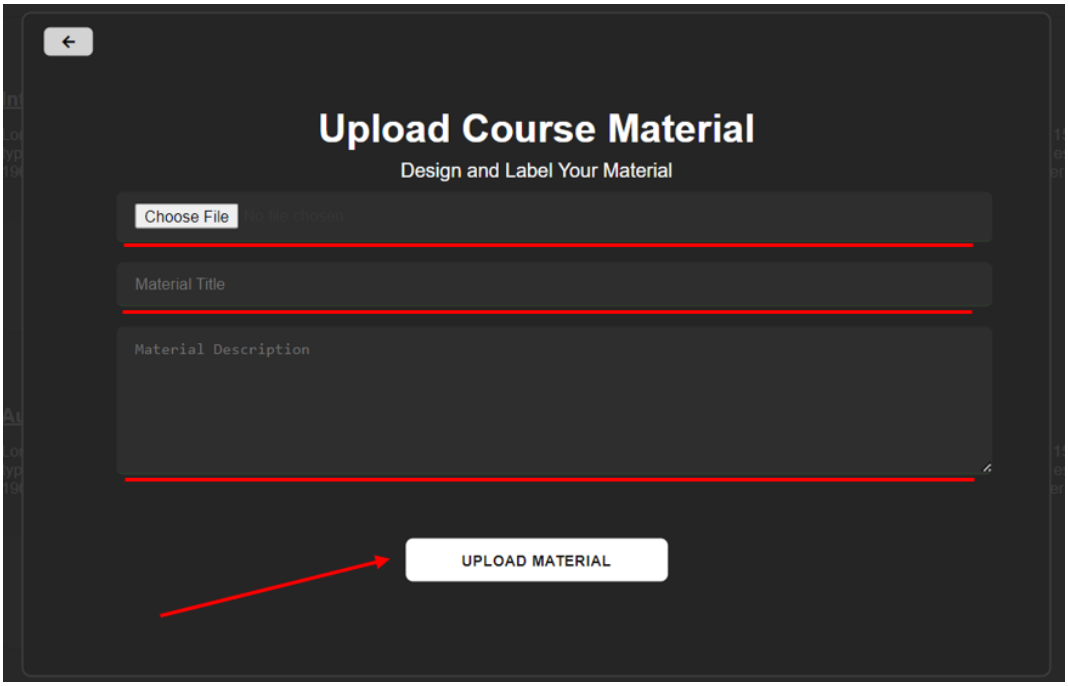


Figure 12.4.7 – Lecturer Manual

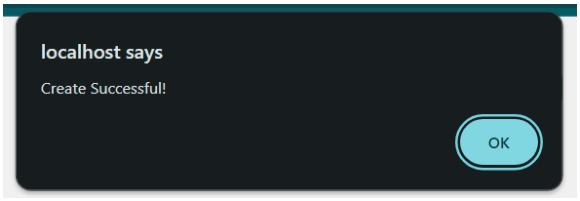


Figure 12.4.8 – Lecturer Manual

After receiving a message that says, “Create Successful”, the below page shows that a new course material is added into the channel. A new file will be also display for students to download later on.

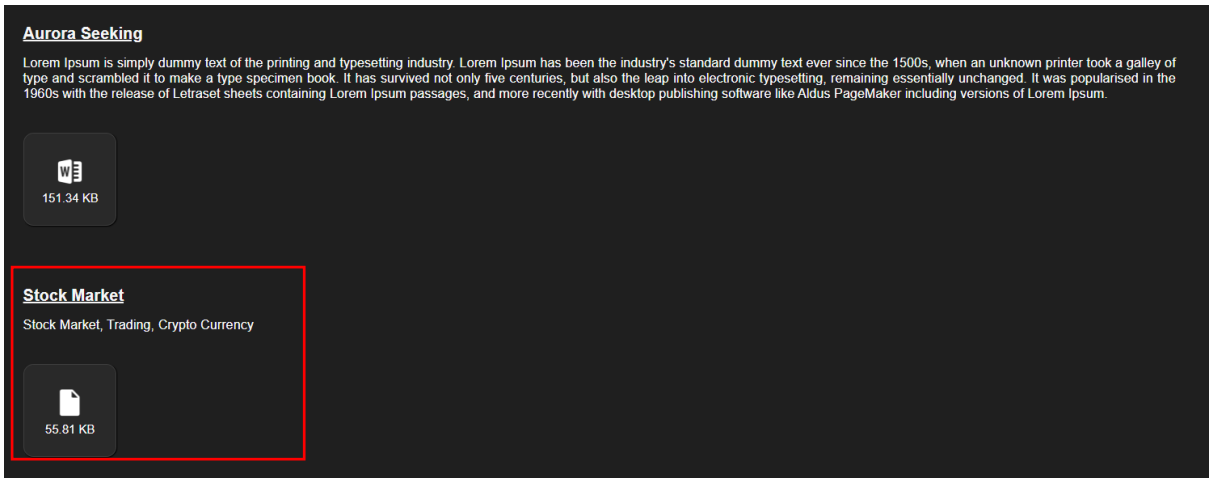


Figure 12.4.9 – Lecturer Manual

12.4.3 Lecturer Manage Assignments Manual

Lecturer switches to Assignment section in the channel page. At the top right corner of the panel, there is a button(icon) for lecturer to add/upload assignments.

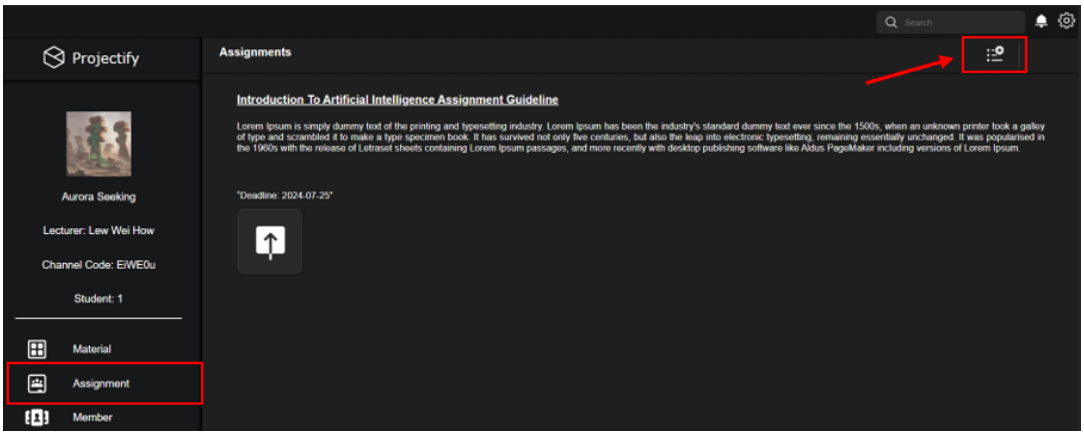


Figure 12.4.10 – Lecturer Manual

After clicking into the button(icon) a pop-up panel will appear that requires lecturers to fill in all fields to create an assignment. “Task Title”, “Task Description”, “dd/mm/yy”, and after all that lecturer will have to click the button below “Upload Material” to create the assignment.

Task Title: System Analysis & Design

Task Description: Complete all course and subjects to pass this module

Chapter 1 – Introduction to System Analysis

dd/mm/yyyy: 29/06/2024

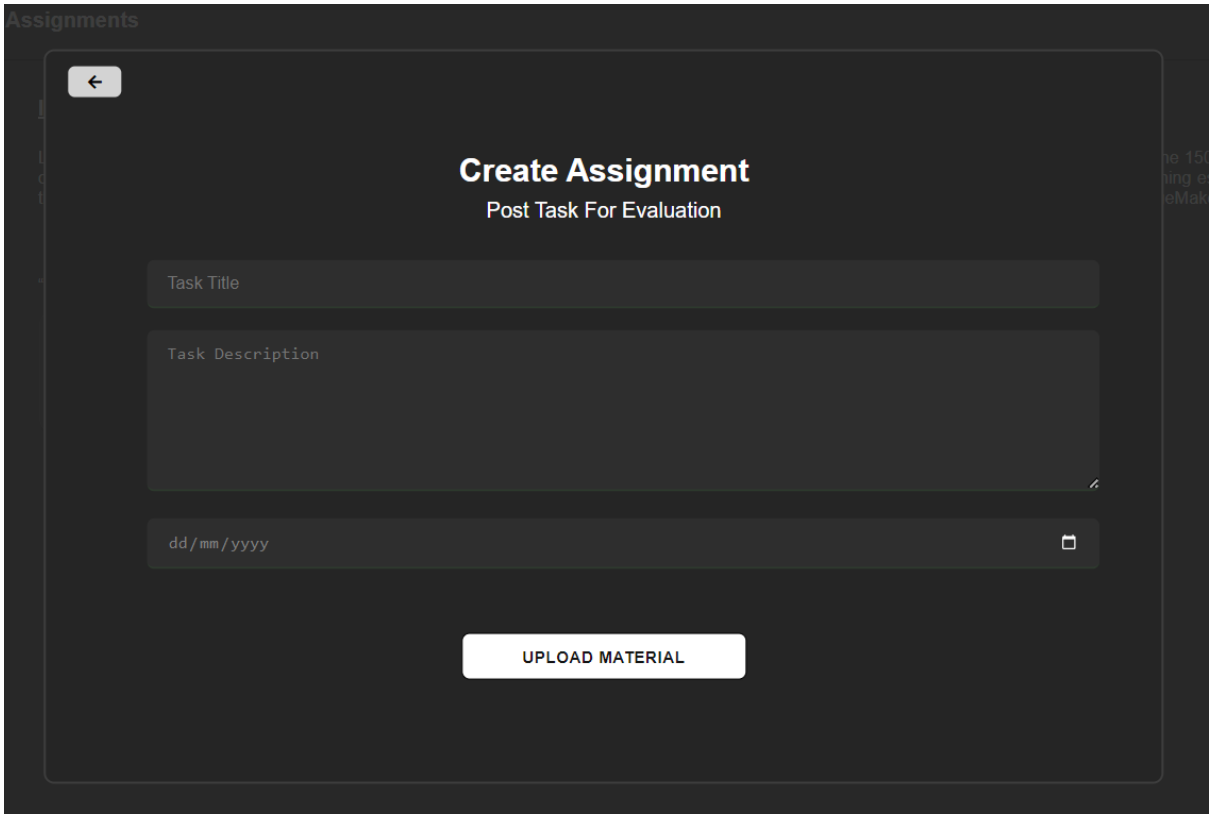


Figure 12.4.11 – Lecturer Manual

Click “OK” after uploading the materials.

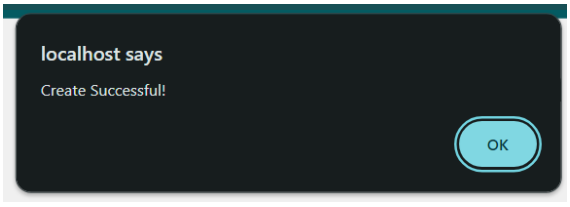


Figure 12.4.12 – Lecturer Manual

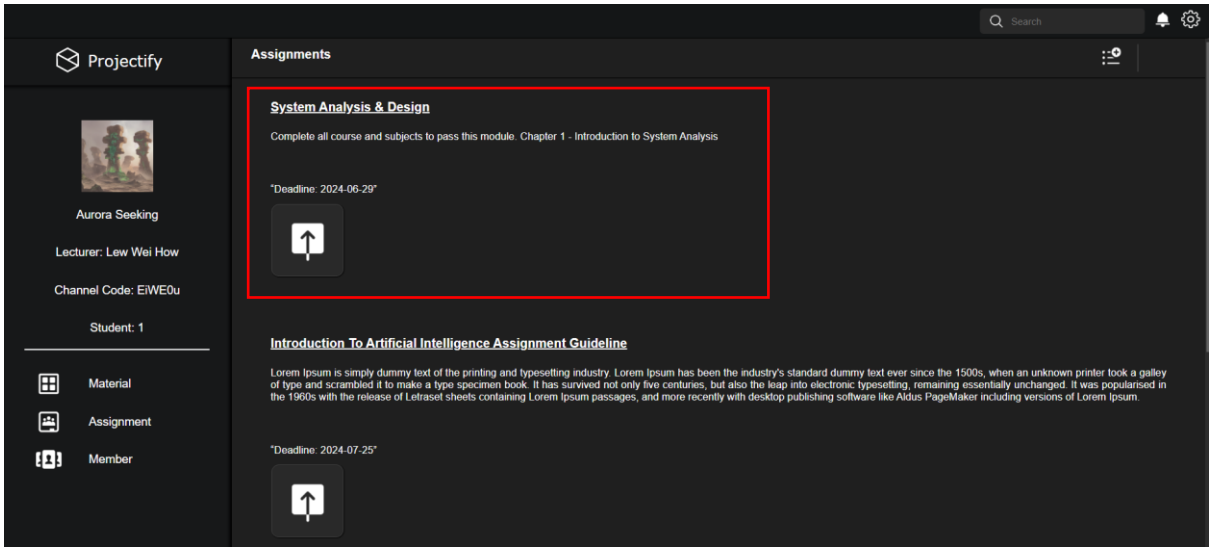


Figure 12.4.13 – Lecturer Manual

12.4.4 Lecturer Manage Member Manual

Lecturer can delete members that have joined the channel. Scroll to the right side of the table row, there is a column named “status”, and lecturer can click onto the “delete” button to remove the member from the channel.

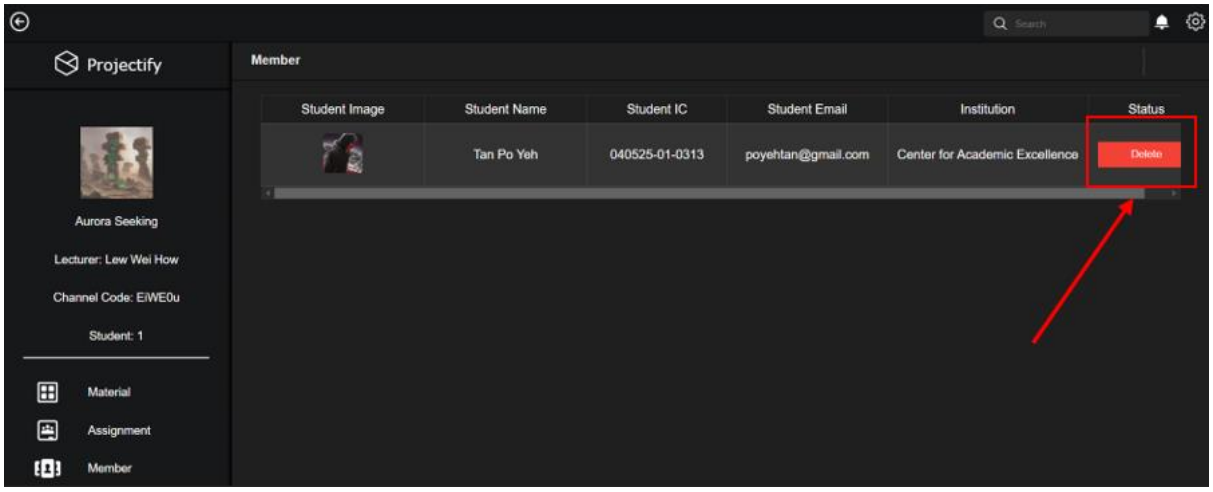


Figure 12.4.14 – Lecturer Manual

After clicking delete, a message will display “Deletion Successful”.

Figure 12.4.15 – Lecturer Manual

The member will then be removed from the lecturer manage member manual.

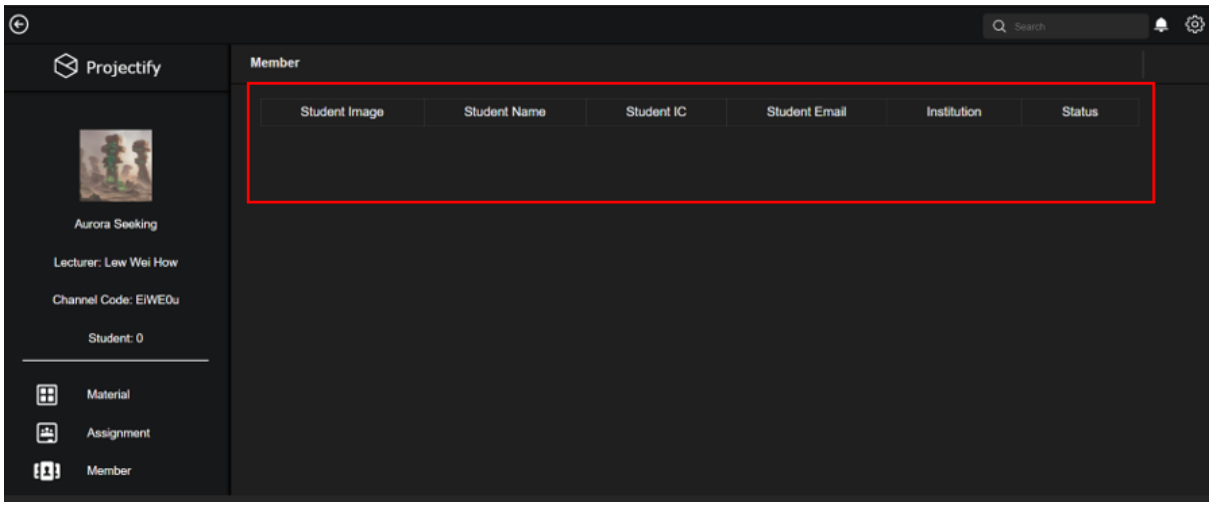


Figure 12.4.16 – Lecturer Manual

12.4.5 Lecture Manage Channels Manual 2.0

Lecturer can manage the channels by deleting them. Each channel will be displayed and lecturer can delete them at the "status" column. At the right side of the panel, lecturer could choose to delete the channels if they decided that it's not updated.

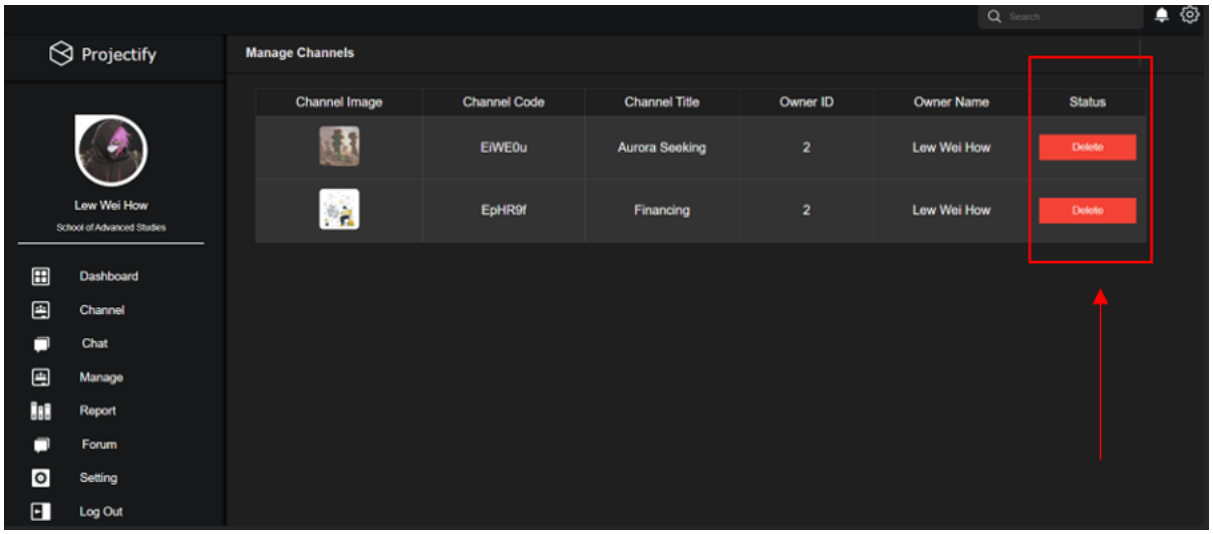


Figure 12.4.17 – Lecturer Manual

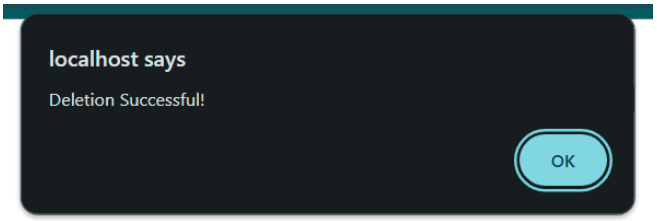


Figure 12.4.18 – Lecturer Manual

After the lecturer chooses which channel to delete, a message will appear and display "Deletion Successful". And returning back to the page, the channel will be deleted and not shown

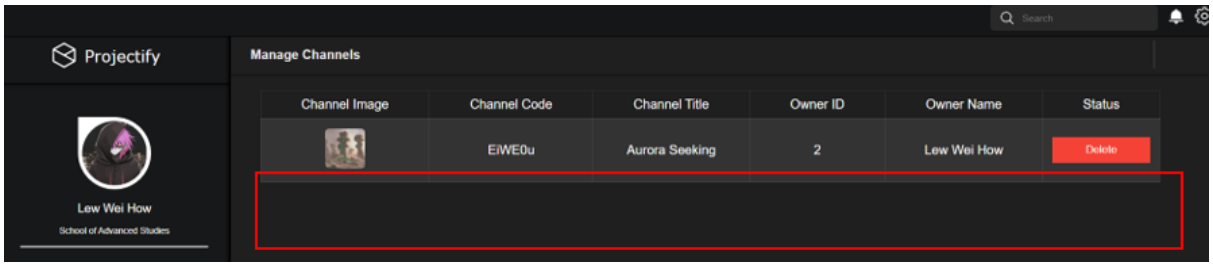


Figure 12.4.19 – Lecturer Manual

12.4.6 Lecturer Manage Report Manual

Lecturer can mark student's report submission at the "Report" Page. Lecturer will have to scroll to the right to grade the student's submission.

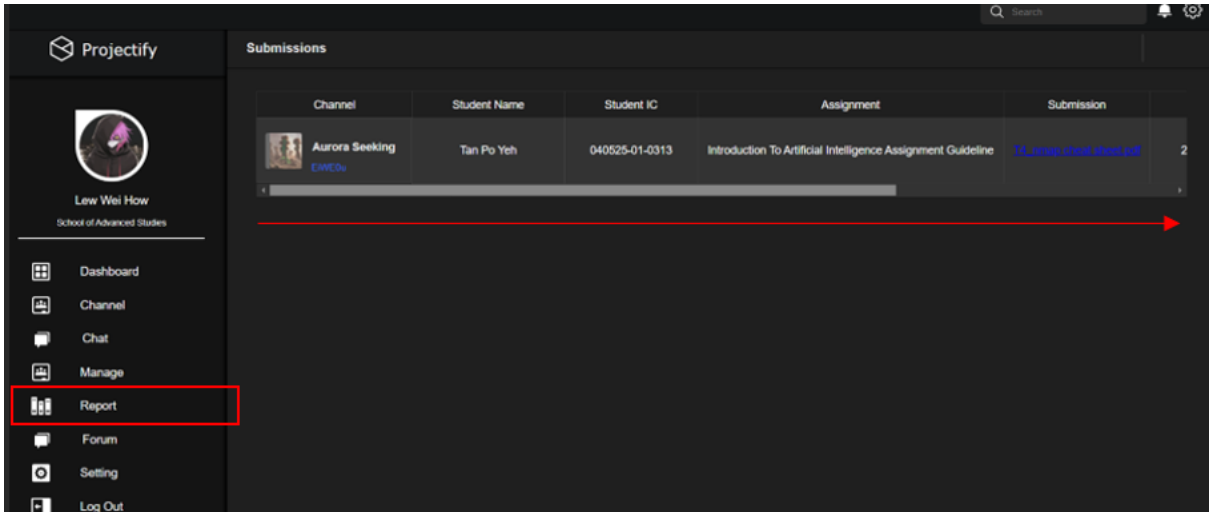


Figure 12.4.20 – Lecturer Manual

After scrolling to the right, there will be a grading "score" column that allows lecturer to grade the student's report mark. After entering the student's mark, there will be a button below the input field that the lecturer will have to click on it to "update" the scores.

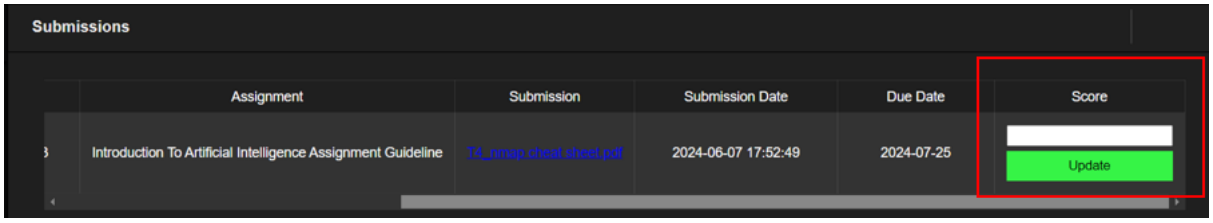


Figure 12.4.21 – Lecturer Manual

Figure 12.4.22 – Lecturer Manual

A message will be displayed after clicking on the "update" button saying that the grading is successful. The graded score from the lecturer will be displayed at the "Students Overview Page" so as the other information.

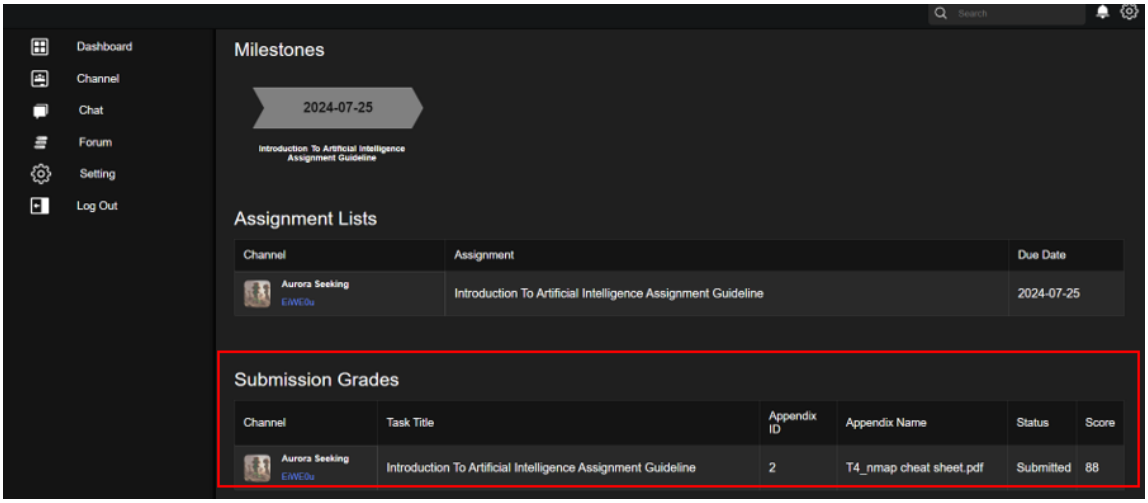


Figure 12.4.23 – Lecturer Manual

12.5 Admin Manual

12.5.1 Admin Manage User Manual

Admin can manage the users in the system. At the right of the manage user panel, there will be a column “status” that allows admin to ban/unban user from the system.

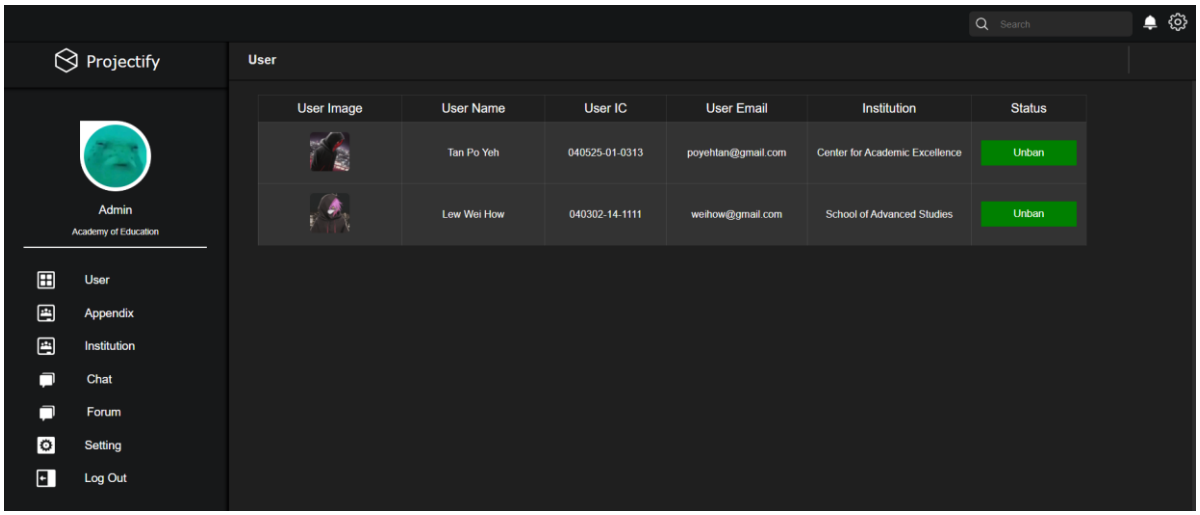


Figure 12.5.1 – Admin Manual

The “status” is all green which they currently have access to the system. Let’s say the admin decided to ban the user “Lew Wai How” from the system, admin can click the button to ban the user. As you can see, the user status is currently green which is “Unban”.

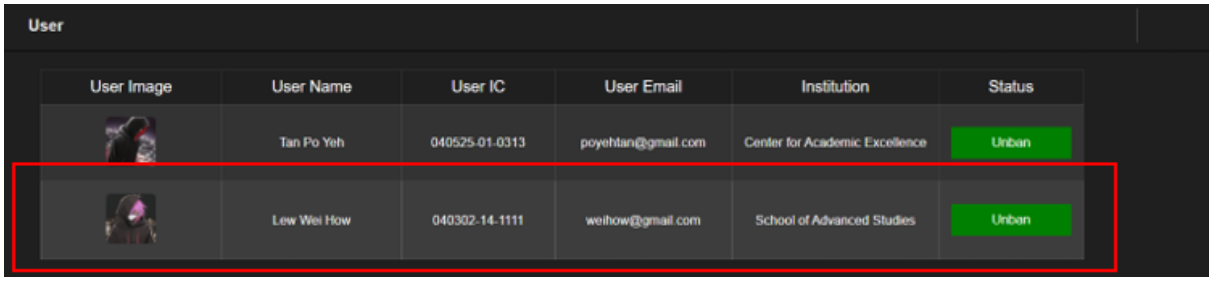


Figure 12.5.2 – Admin Manual

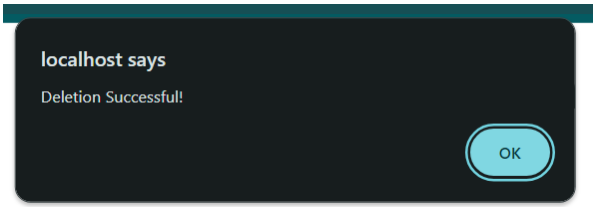


Figure 12.5.3 – Admin Manual

As you can see now, the user has been “banned” from the system. The status button colour will be changed to red also. If the admin decided to unban the user, simply press back the “Banned” button to unban the user. The status will then return back to green, and user will have access back to the system.

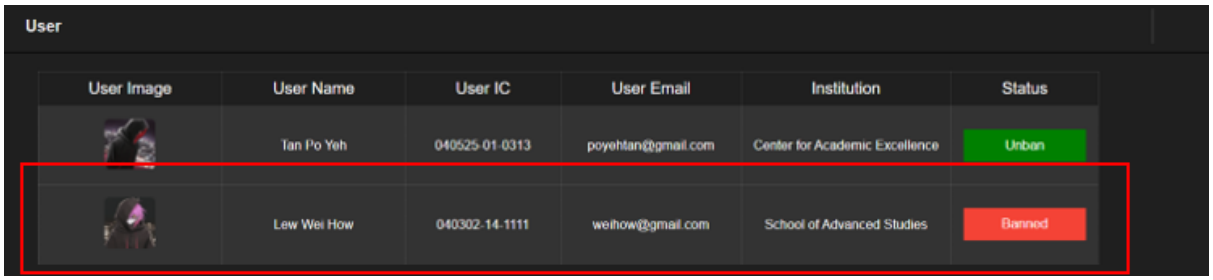


Figure 12.5.4 – Admin Manual

12.5.2 Admin Manage Appendix Manual

Admin can also grade the student’s report, scroll right of the appendix panel.

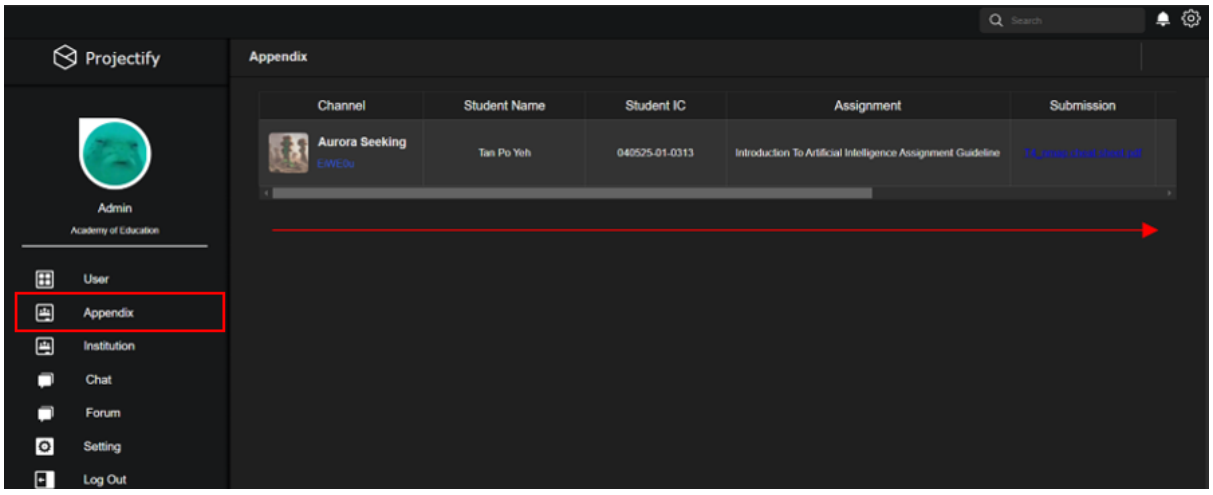


Figure 12.5.5 – Admin Manual

There will be a column “score” for admin to change the marks of the student. Let’s say the admin wants to reduce the student’s mark from 88 to 85. Then after done click the “update button to update the database.

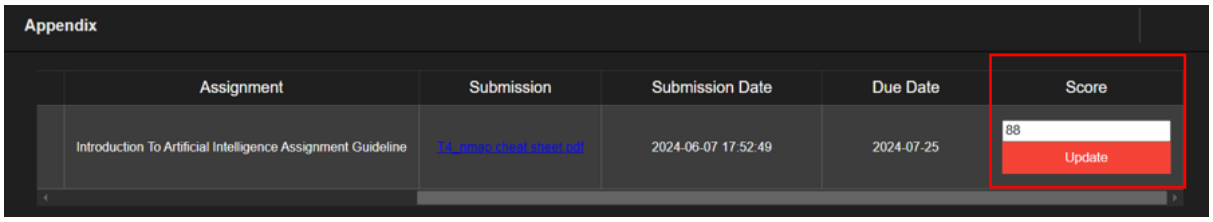


Figure 12.5.6 – Admin Manual

Figure 12.5.7 – Admin Manual

A message will be displayed that says “Update Successful” after clicking the update button. Then the scores will update as well in the “Student Overview Page”.

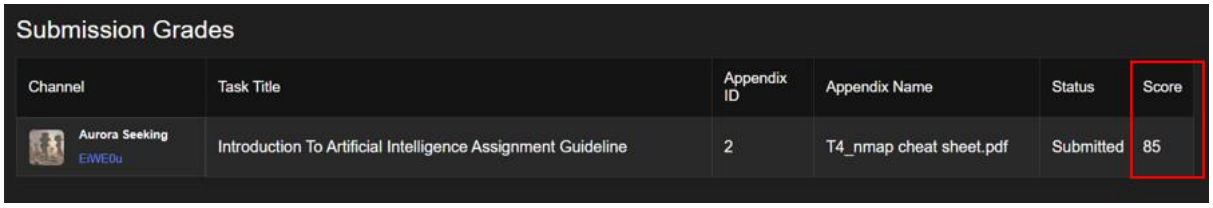


Figure 12.5.8 – Admin Manual

12.5.3 Admin Manage Institution Manual

Admin can delete the institution in the Institution Page. At the right side of the institution panel there will be a column named “Tool” that allows admin to delete/remove the Institution that has in the database.

Let’s assume the doctor wants to delete Institution ID = 2.

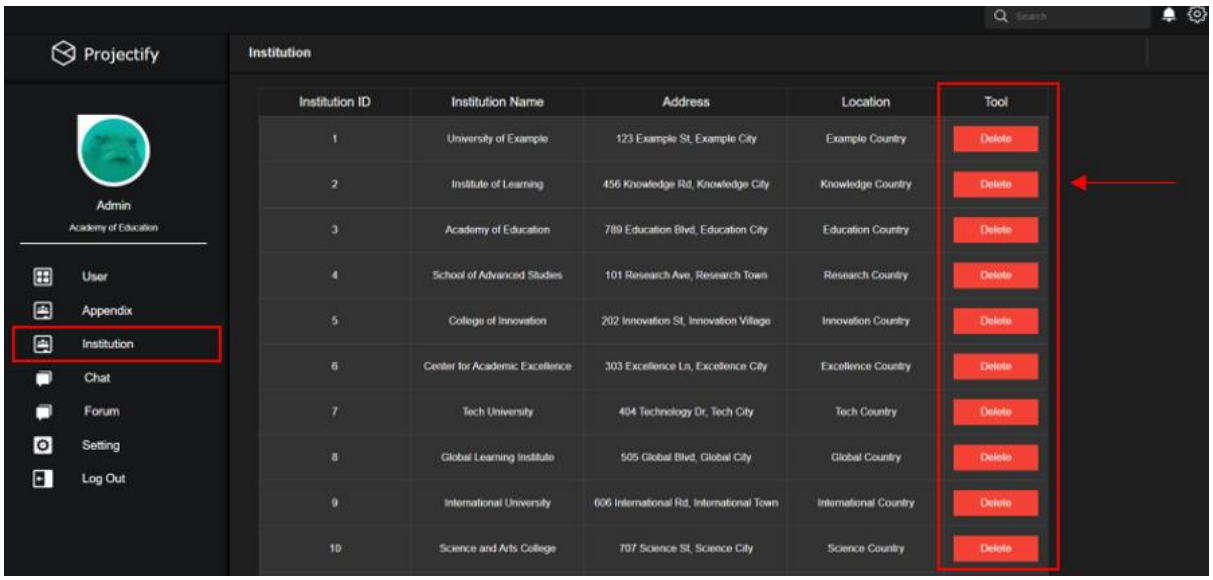


Figure 12.5.9 – Admin Manual

A message will be displayed on top of the page saying that the deletion is successful. Then the selected Institution will be deleted from the page

Figure 12.5.10 – Admin Manual

12.6 Common User Manual

12.6.1 Users Chat Manual

This is the users chat page. At the right side, it will be the chatting panel for any user to have conversations between students, admin, and lecturers. As for now, I will be using the student's account to chat with the admin. There will be a list of users that the student can choose to start a conversation with.

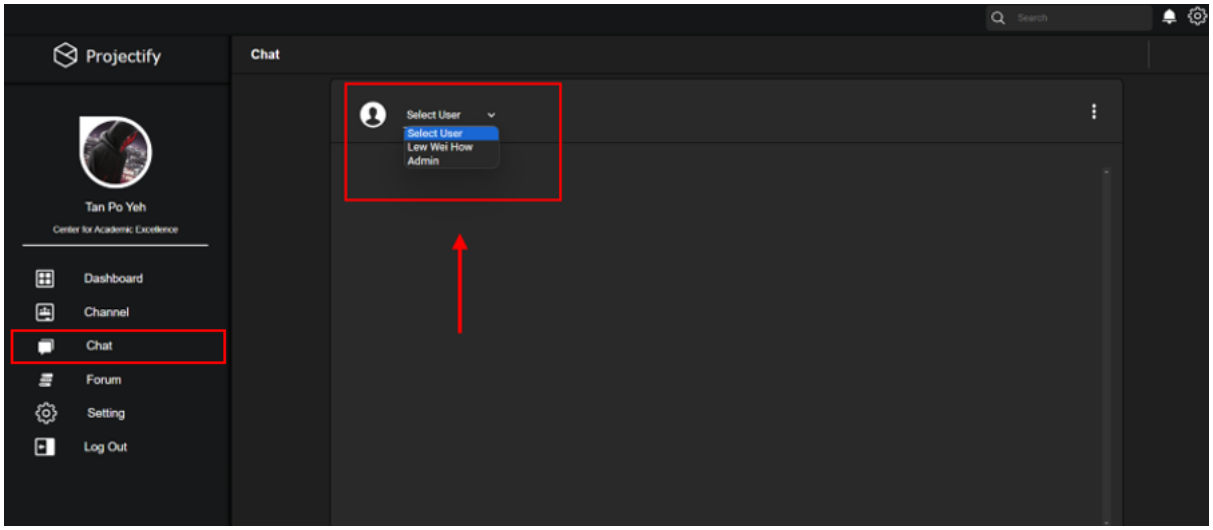


Figure 12.6.1 – Common User Manual

After selecting which user the student wants to communicate with, the right-side panel will display it and also has chat histories of the conversation of students with other users.

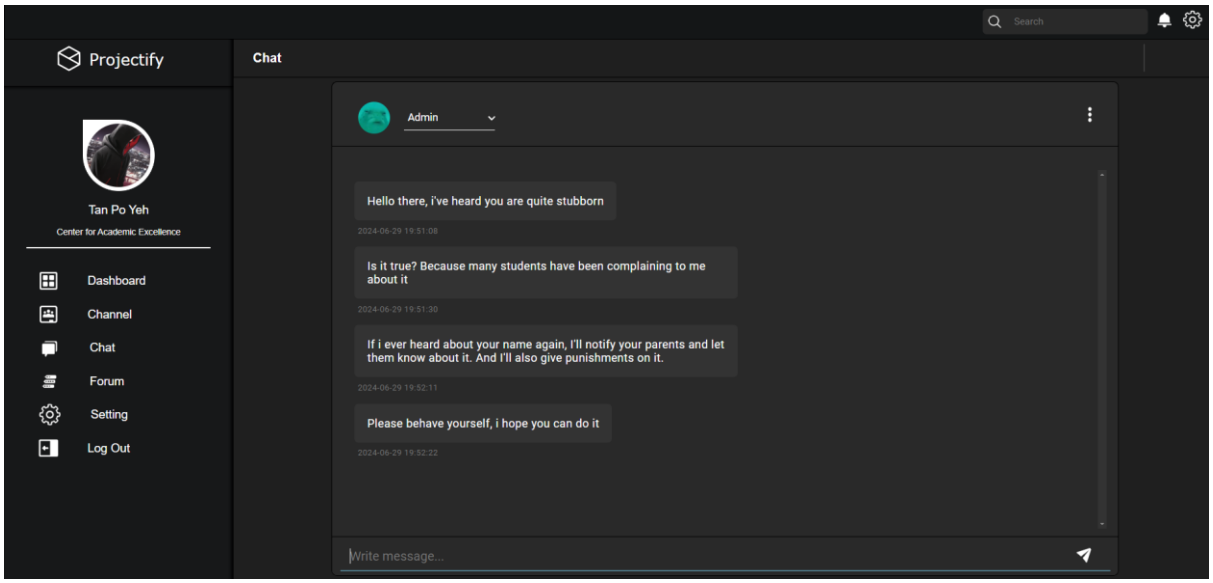


Figure 12.6.2 – Common User Manual

Student can start a conversation by typing what they want at the bottom of the panel. There will be a “send icon” of the bottom right corner which is for students to send the conversation once they are done typing.

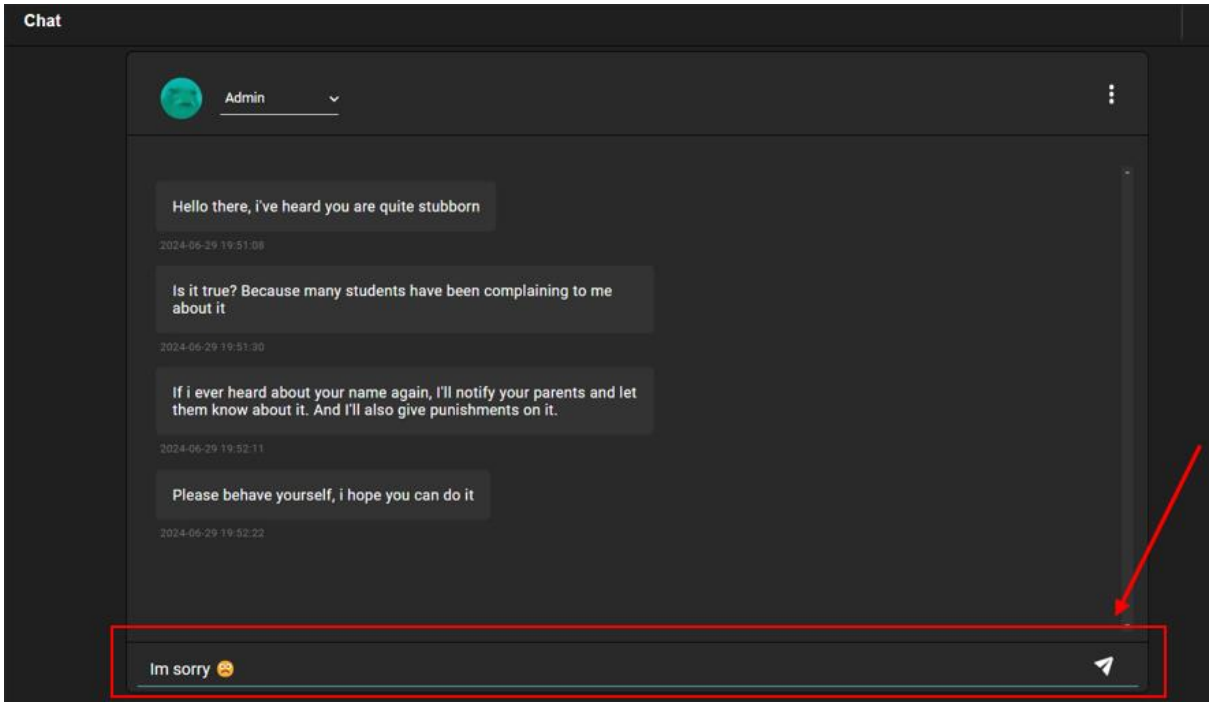


Figure 12.6.3 – Common User Manual

After clicking on the “send icon”, the text will be sent to the user. And the chat display will be in the panel as well. The user will be able to see it from there and reply to them.

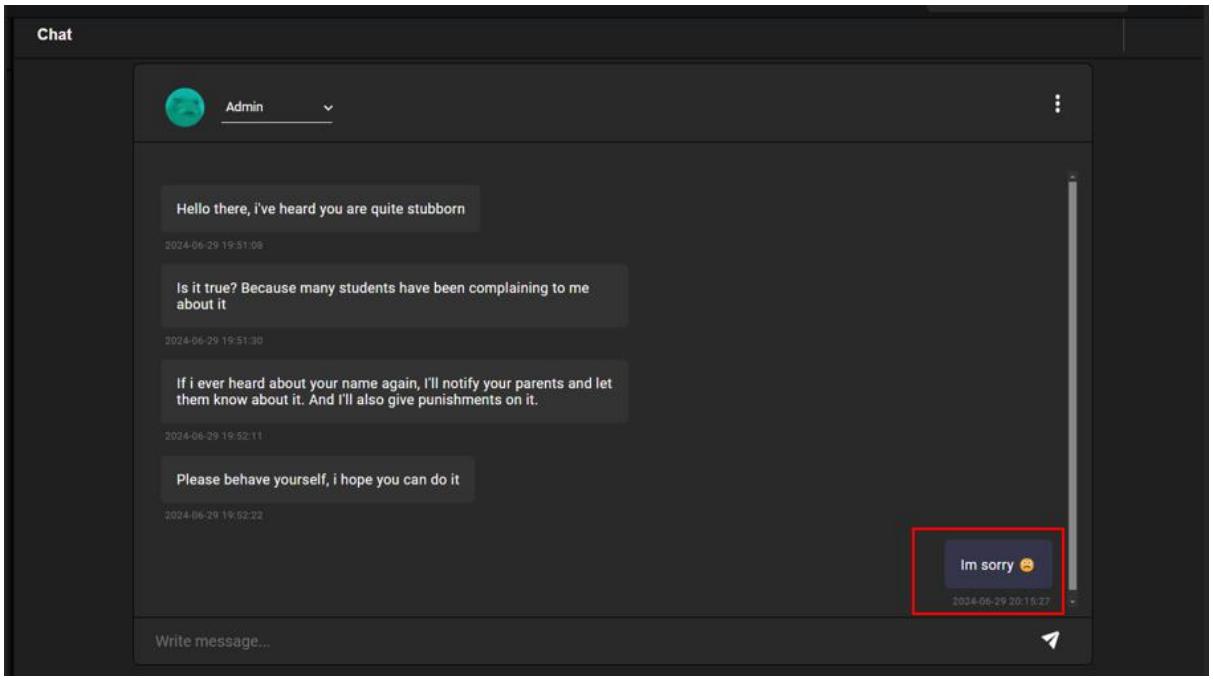


Figure 12.6.4 – Common User Manual

This will be displaying at the users chat box as well.

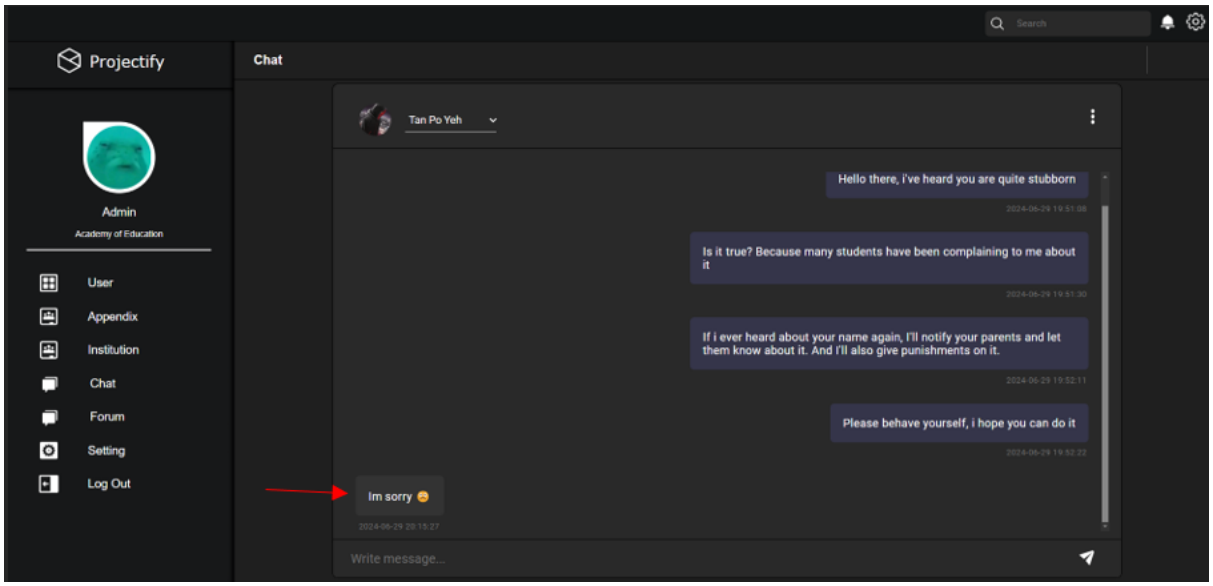


Figure 12.6.5 – Common User Manual

12.6.2 Users Forum Manual

This is the student's forum page, which can discuss anything about the subject etc. The right side will be the panel for student forums. If students want to "Create Post", "View Your Post", "View All Post", there will be a tool on the top of the panel called "Tools" for students to perform those.

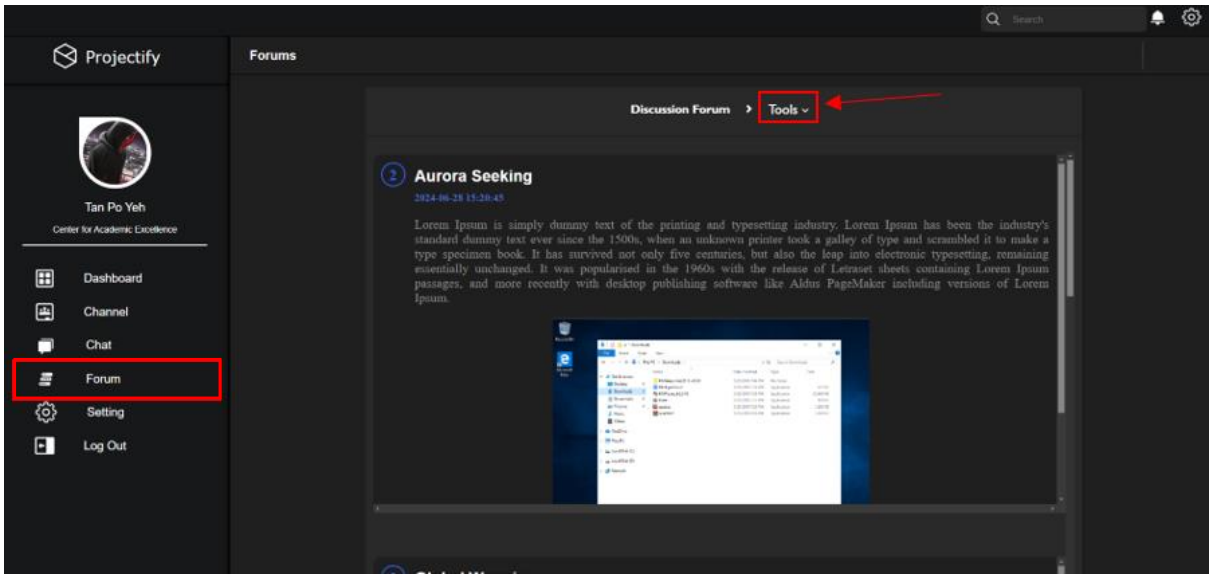


Figure 12.6.6 – Common User Manual

This will be what it looks like after clicking on the "Tools" button. Each category will display below of what the students have choose.

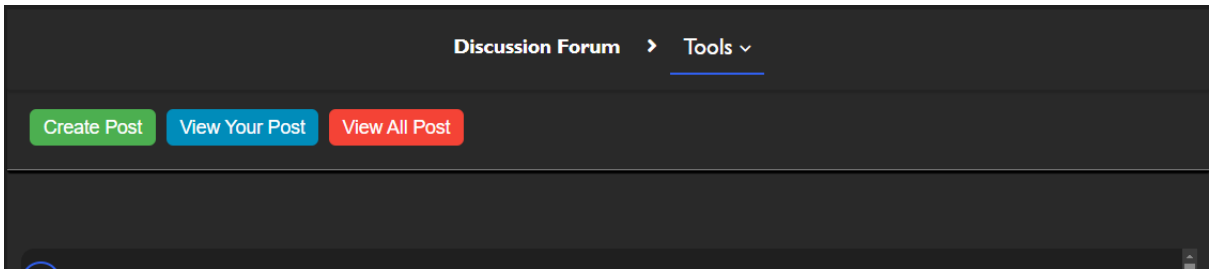


Figure 12.6.7 – Common User Manual

12.6.3 Users Forum Create Post Manual

To create a new post in the forums, user will have to click on the green "Create Post" button.

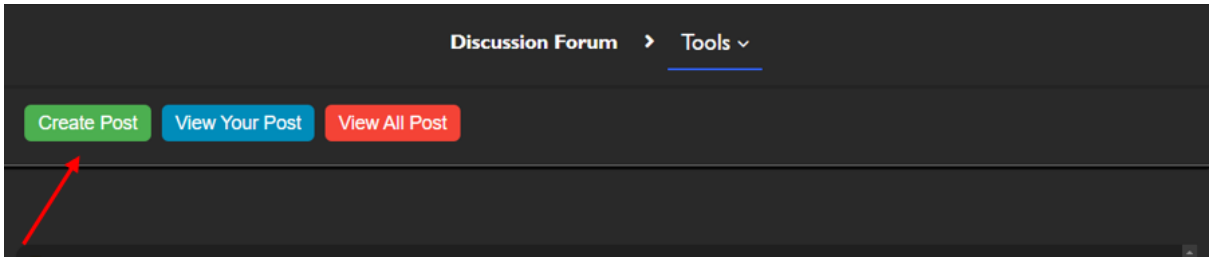


Figure 12.6.8 – Common User Manual

After clicking on the button, there will be a pop-up panel for users to fill up in order to make a post in forums. Upload the image post, “Post Question Title”, “Post Question Content”.

Let’s assume that the user wants to add a new forum with the content of:

File: *_____*

Post Question Title: My cute dogs

Post Content Title: Please rate my pet, out of 1 to 10. Other than that, anyone that described the best of my pet will receive a reward from me.

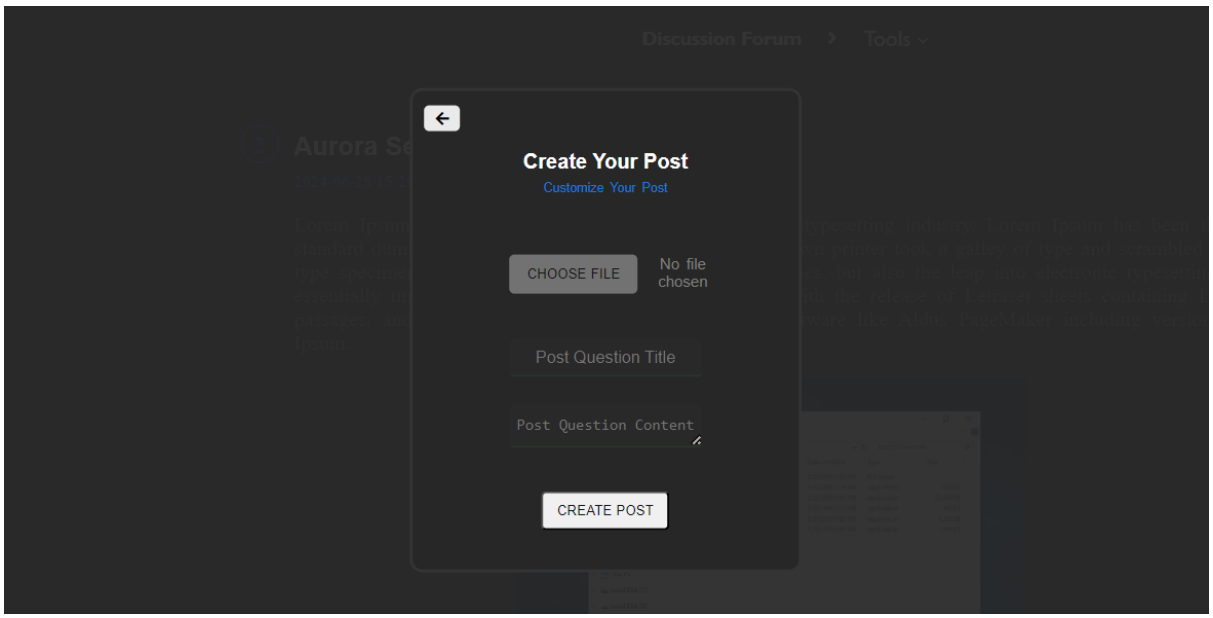


Figure 12.6.9 – Common User Manual

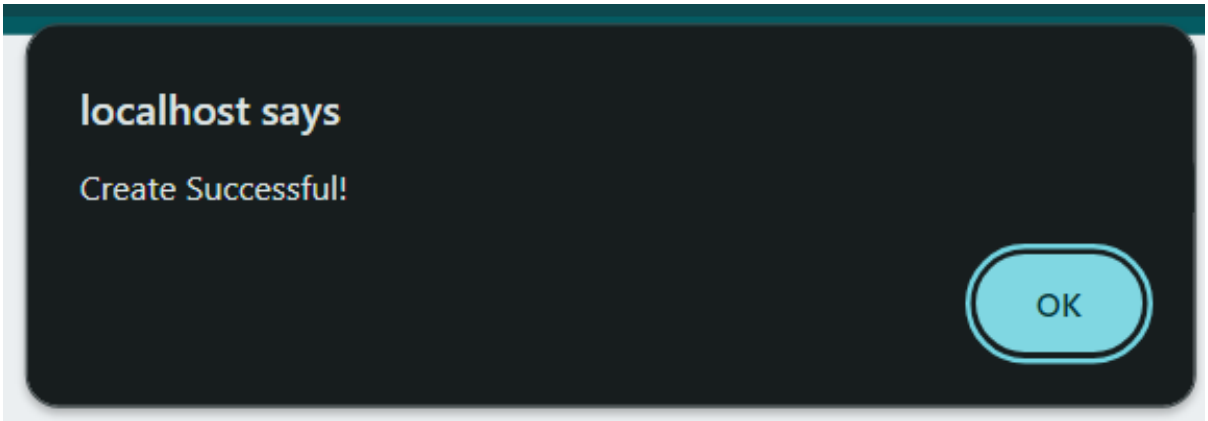


Figure 12.6.10 – Common User Manual

A message will appear after successfully create a forum post. Then, post will be shown at the bottom page of the forum's panel.

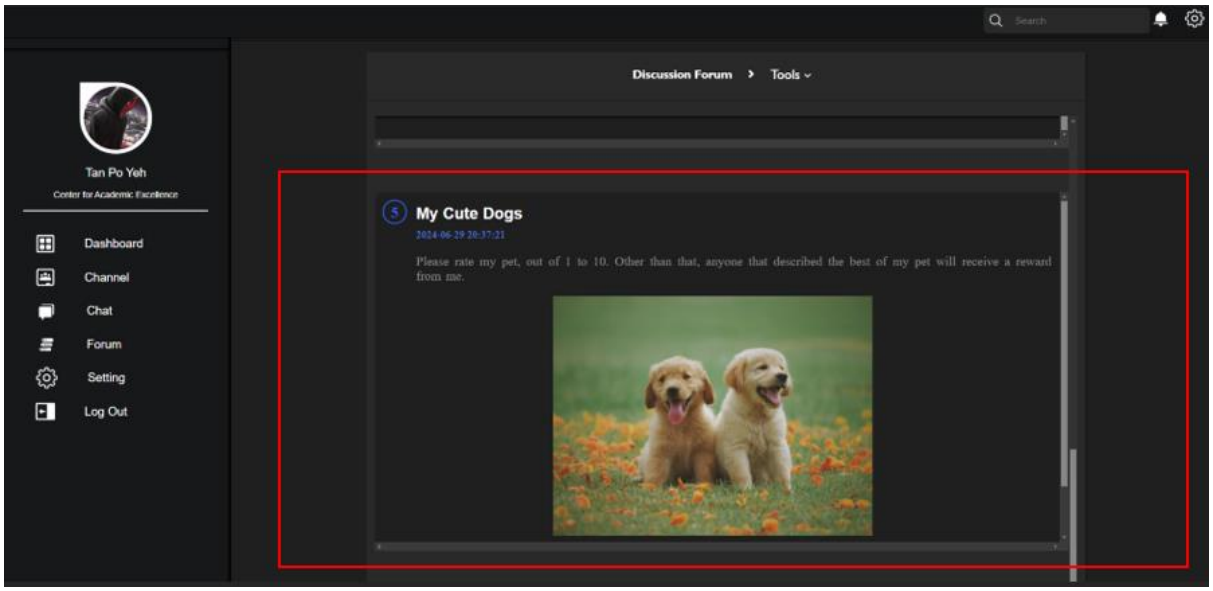


Figure 12.6.11 – Common User Manual

12.6.4 Users View Your Post & View All Post Manual

To view user's own post, click on the blue "View Your Post" button to see the old post that the users have post before.

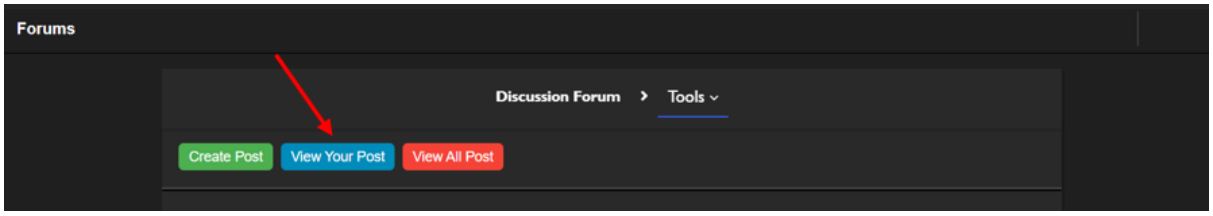


Figure 12.6.12 – Common User Manual

This is the page where users, can see the old posts that they have in the forums page. The dog post appears below because the user has posted an old forum before.

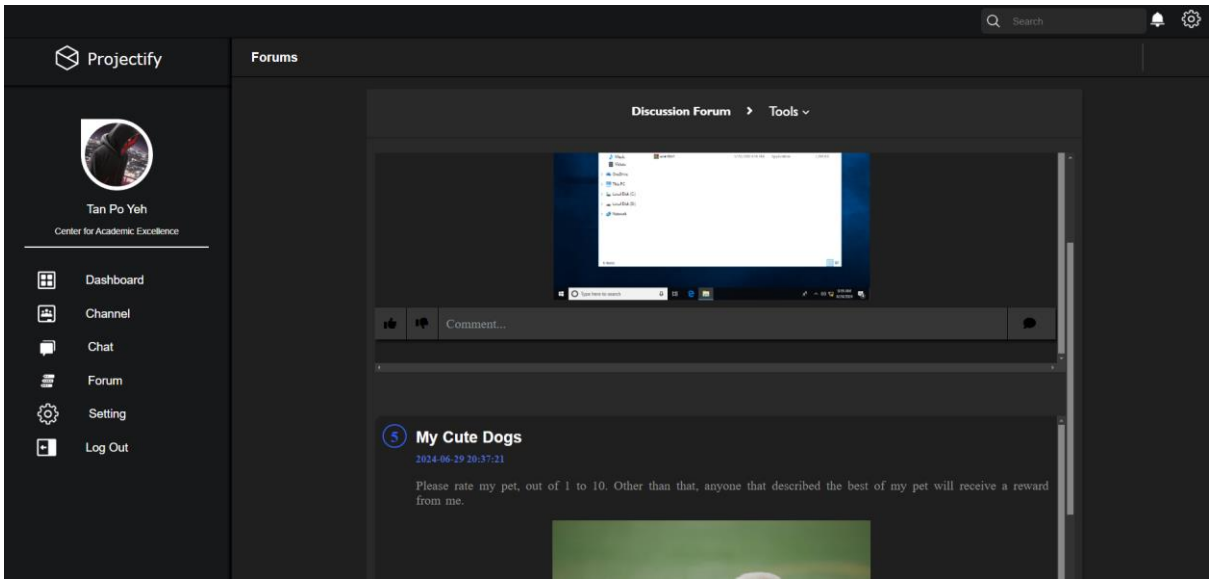


Figure 12.6.13 – Common User Manual

To view all the forum post, users will have to click on this red "View All Post" button.

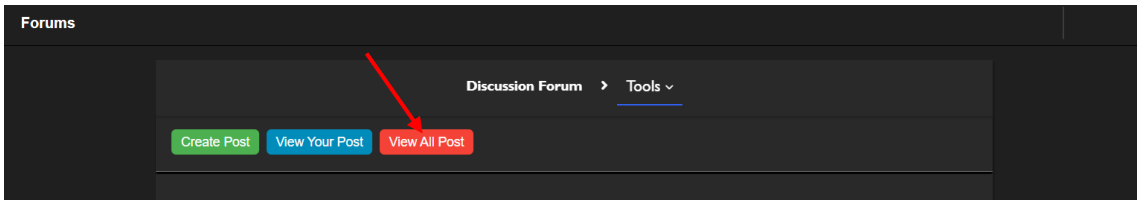


Figure 12.6.14 – Common User Manual

User can view all the post after clicking the button. Other user's forum post will be displayed.

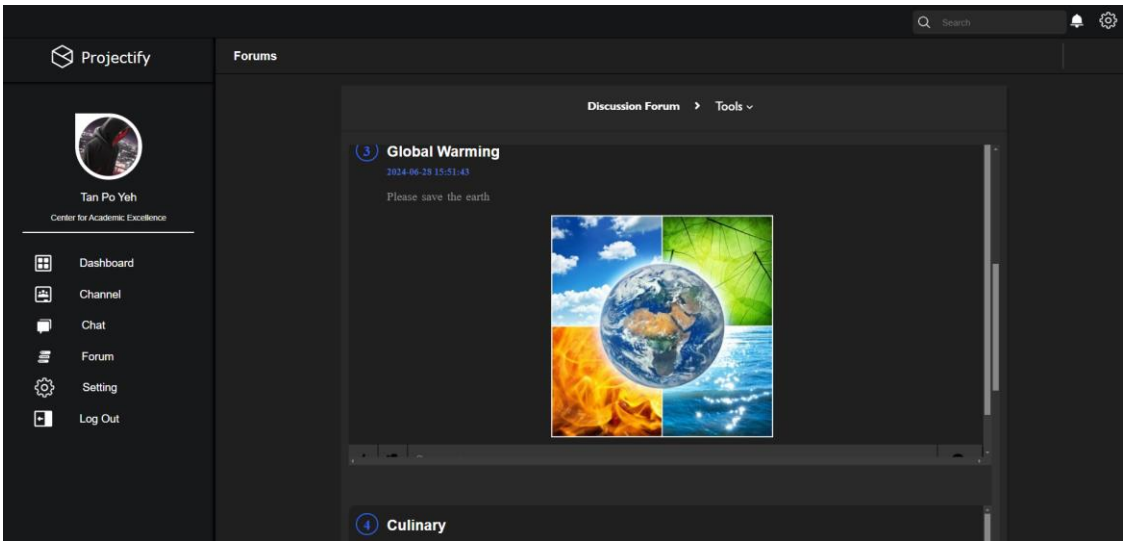


Figure 12.6.15 – Common User Manual

12.6.5 Users Forum Post Comments Manual

At every post below, there will be a comment section for any user to comment on it. By clicking the comment under the image There will be a comment section appear

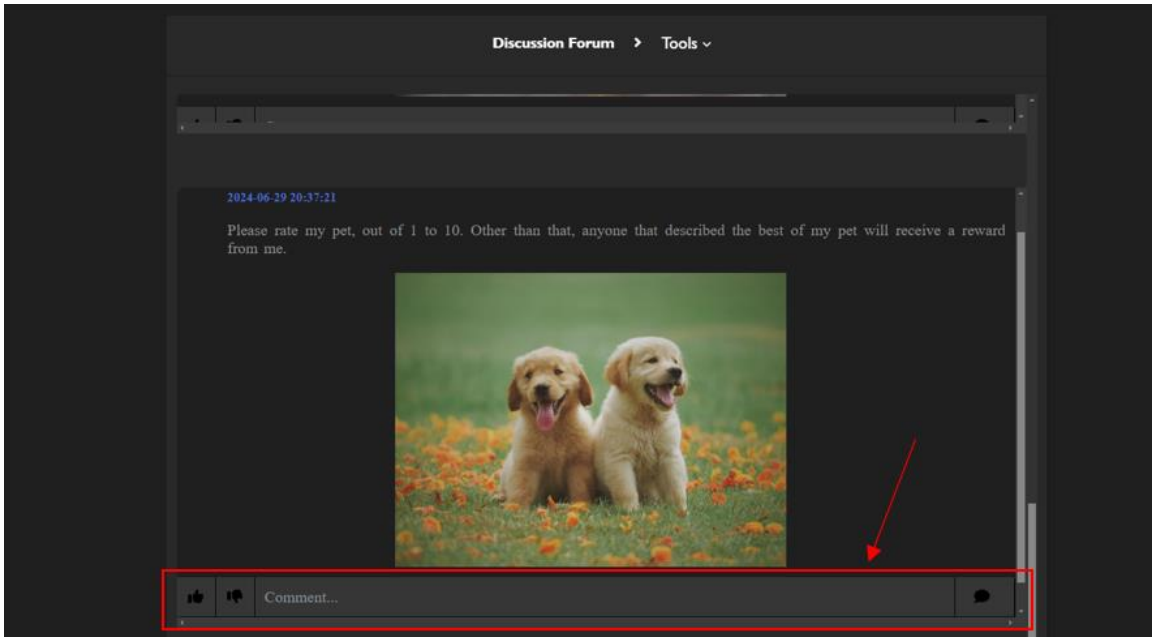


Figure 12.6.16 – Common User Manual

The comment section allows user to comment on it, if the user is done typing, they can click on the “Confirm Button” at the bottom right corner to send the message. If the user decided not to send, there will also be a “Cancel” button. Any other comments will also appear on top of the comment bar.

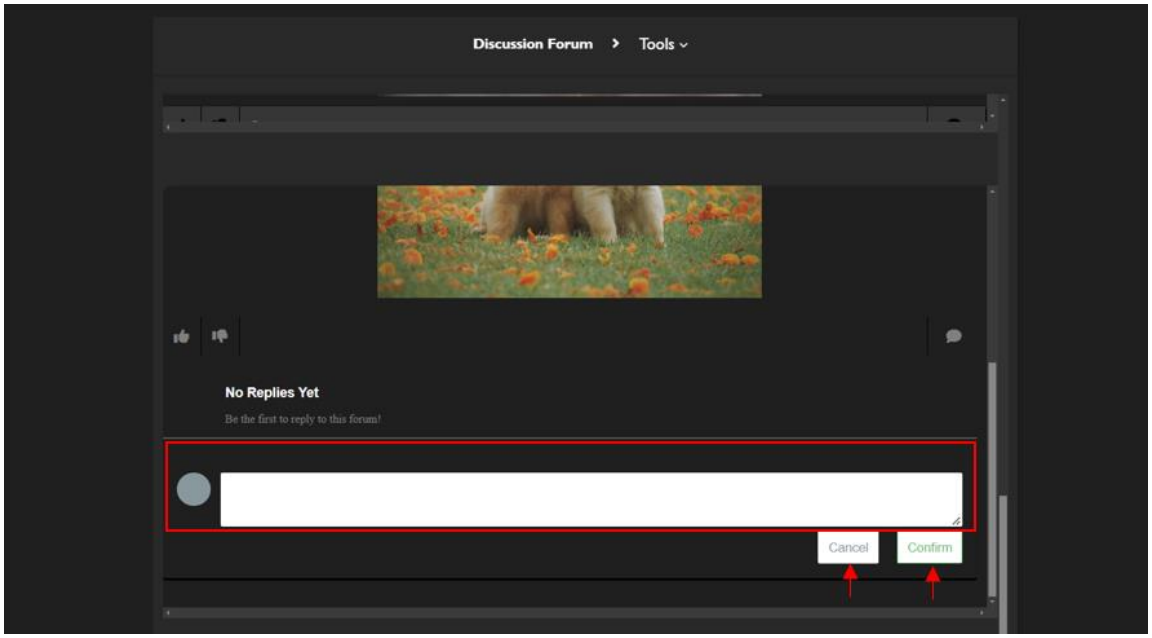


Figure 12.6.17 – Common User Manual

Assume the user typed:

Text: This dog is so cute! Definitely a 10, It just warms my heart.

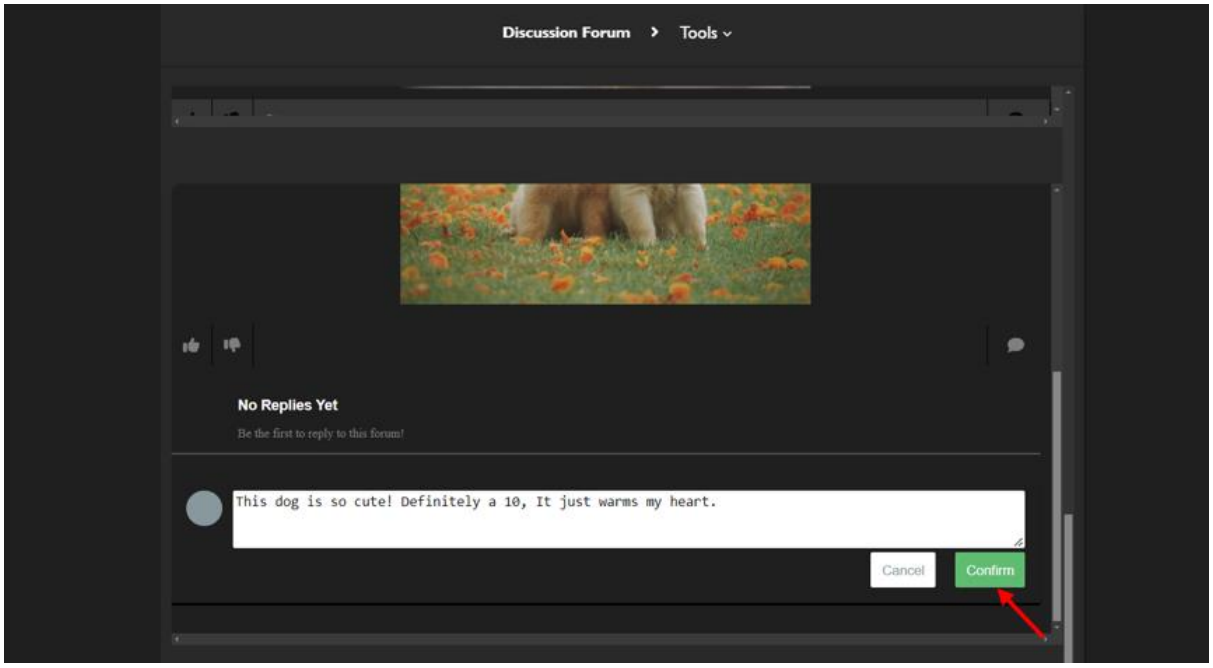


Figure 12.6.18 – Common User Manual

After clicking " Confirm", a message will appear that confirmed that student's message is sent. Now, going back to the forum post comment section. Student can see back the messages.

Figure 12.6.19 – Common User Manual

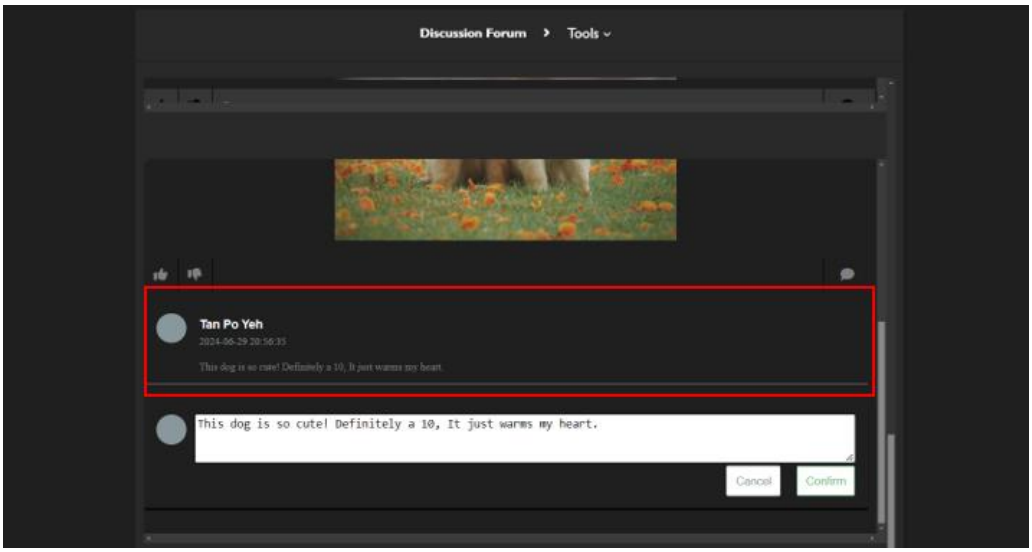


Figure 12.6.20 – Common User Manual

12.6.6 Users Setting Manual

This page allows user to edit their own profile. Each of the fields allows user to modify their own data/information. As the current age is 20, and the gender is male.

Let's assume that I used the student's account. I want to modify:

Age: 21

Gender: Female

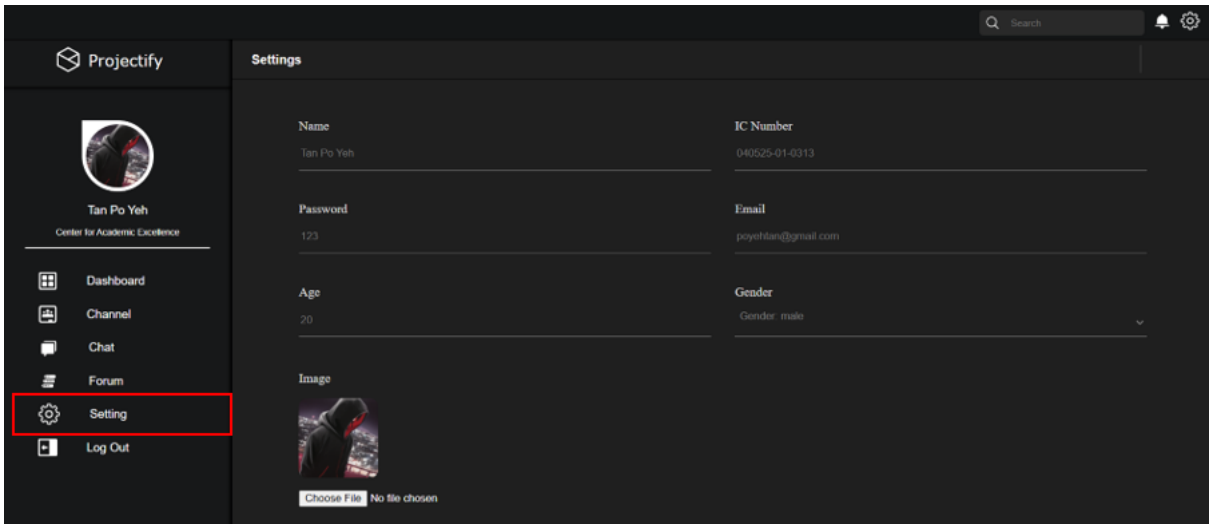


Figure 12.6.21 – Common User Manual

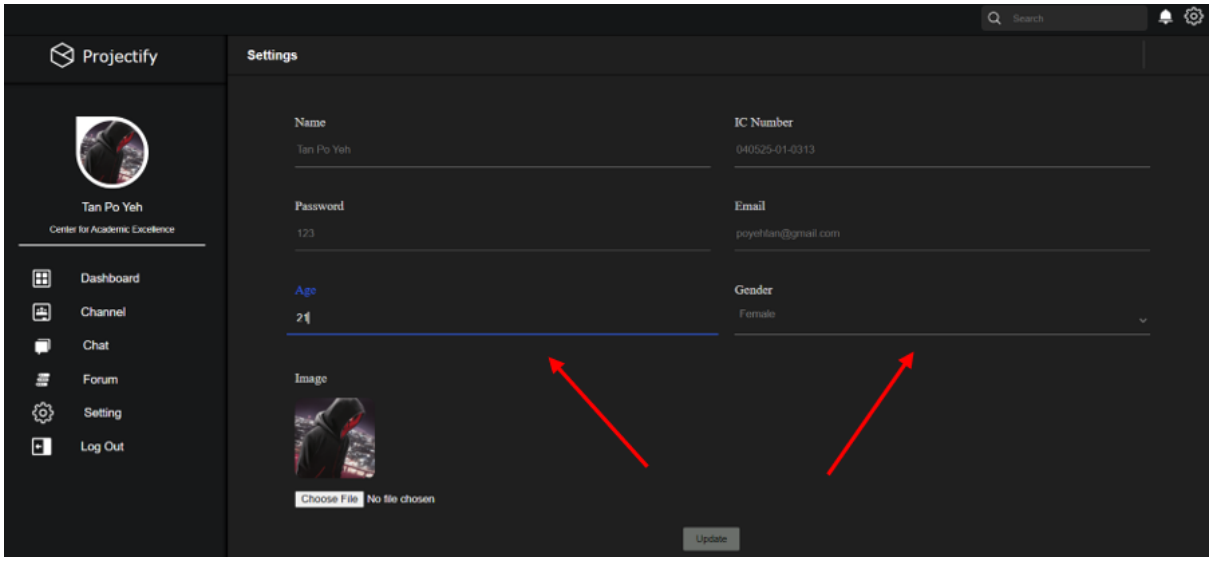


Figure 12.6.22 – Common User Manual

After filling up the information that the user wants to change, there is a button below the panel "Update" for user to click on it to save the information and update it in the database.

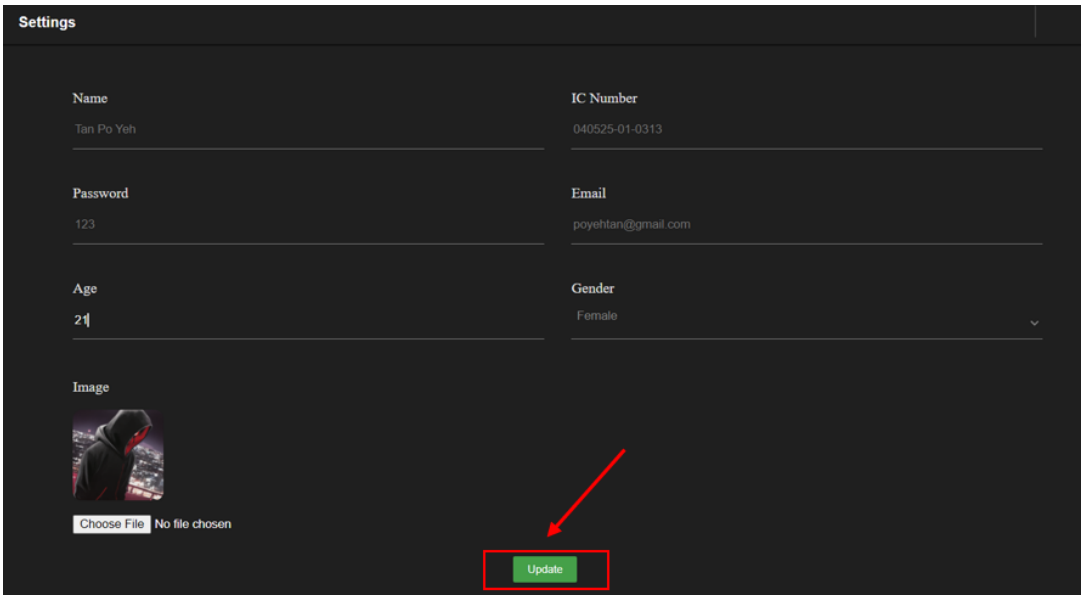


Figure 12.6.23 – Common User Manual

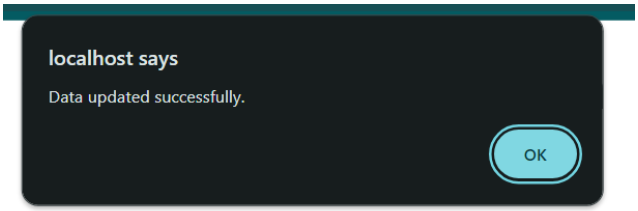


Figure 12.6.24 – Common User Manual

A message will be displayed saying that the data has been updated successfully.

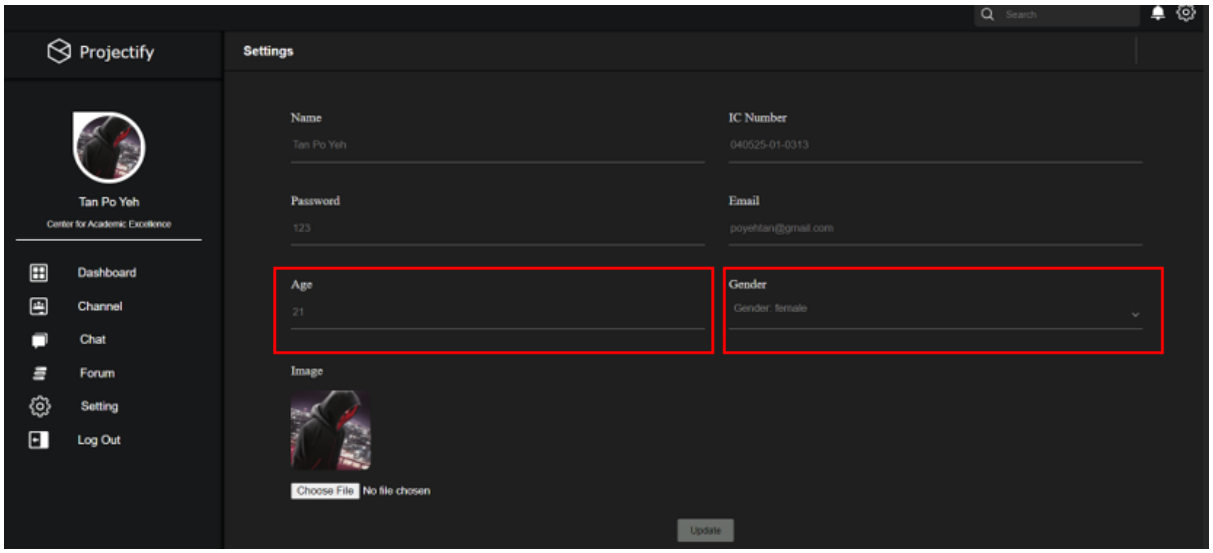


Figure 12.6.25 – Common User Manual

Assume that the user also wants to change his profile picture.

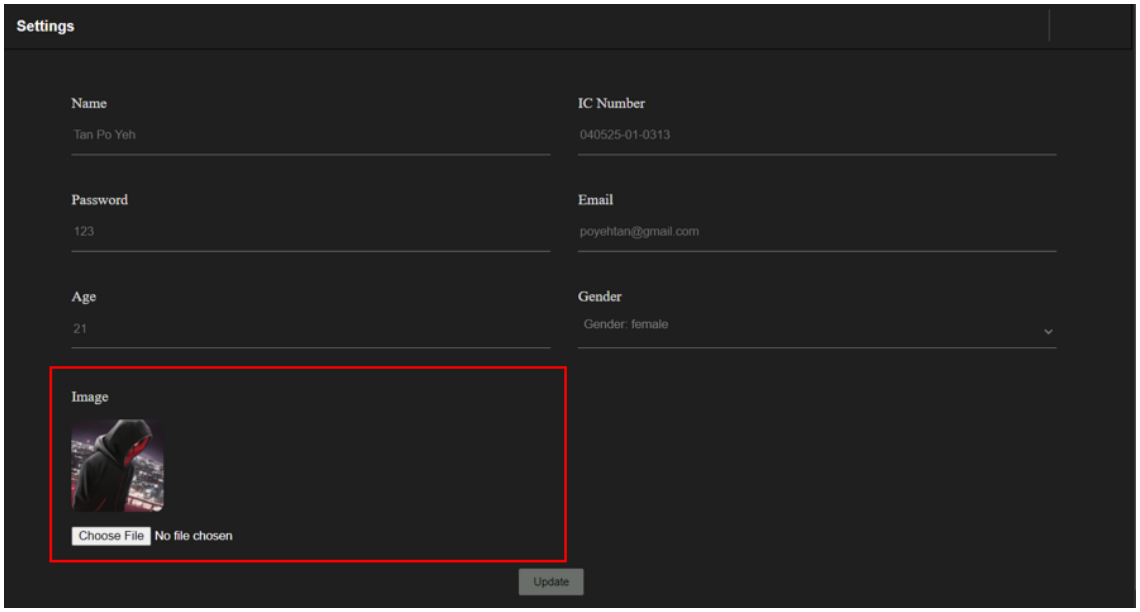


Figure 12.6.26 – Common User Manual

Click on the “update” button after done choosing the profile picture.

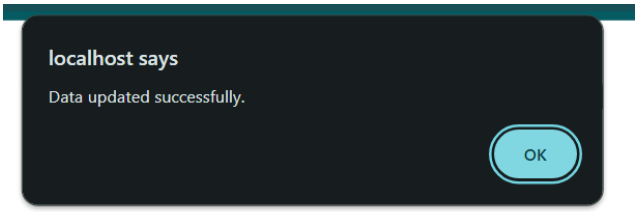


Figure 12.6.27 – Common User Manual

After refreshing, the profile picture will be updated for the user.

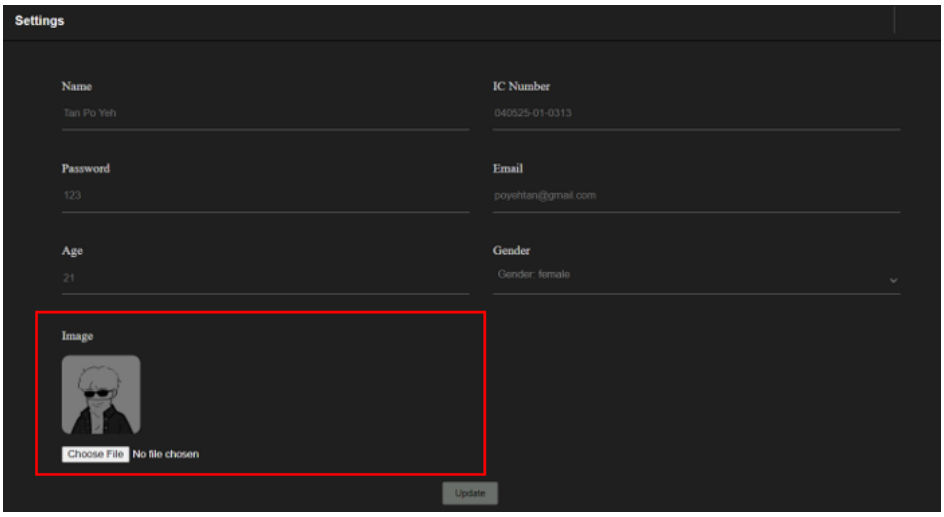


Figure 12.6.28 – Common User Manual

12.6.7 User Log Out Manual

I'll be using back the student's account to demonstrate how to log out for the account. The last section of the dashboard, there is a section button called "Log Out" where user will have to click on that to log out.

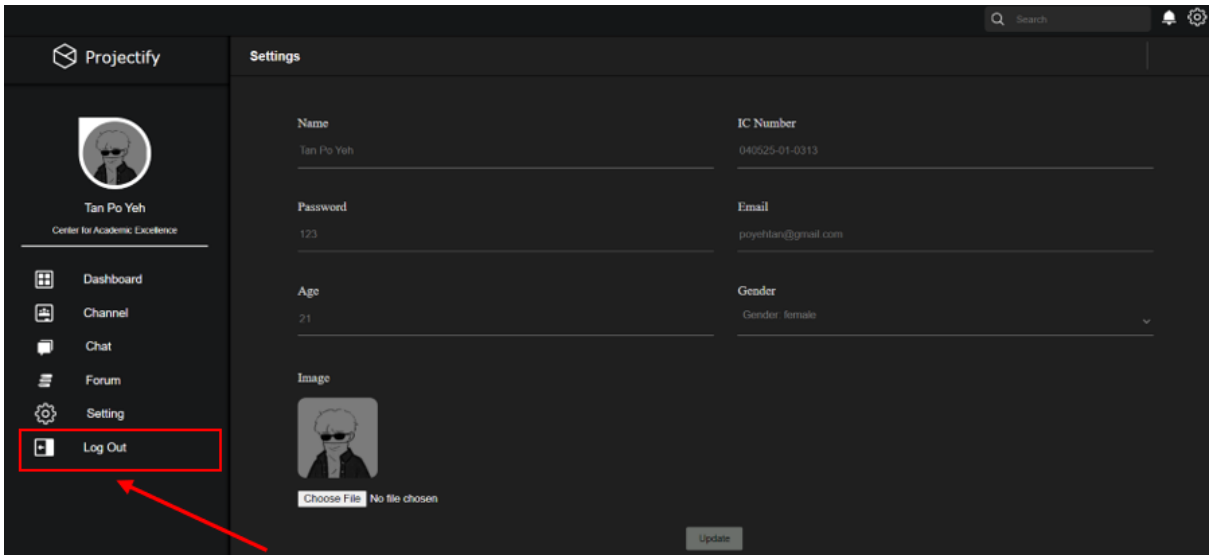


Figure 12.6.29 – Common User Manual

There will be a message display on top of the page, requesting the user to confirm on logging out the account. If yes, then click on the "OK" button. And then user will be redirected back to the home page of our system.

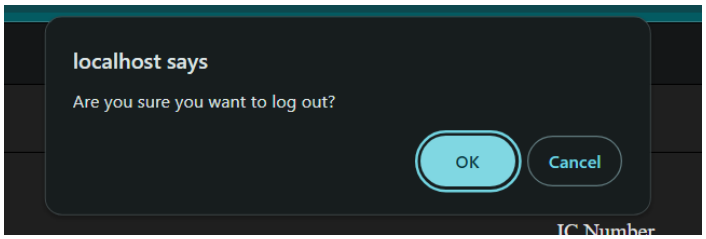


Figure 12.6.30 – Common User Manual

13.0 Test Plan

13.1 Unit Testing

13.1.1 User Authentication System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
T001	Sign-Up Functionality	User will have to access to sign in page.	User located at the sign in page.	1. Users have to click on the sign-up button 2. User will be redirected to the page sign up.	N/A	User access to sign-up page	The system redirect user to the sign-up page	Pass
		Testing the sign-up programme logic	User email does not register the account yet	In the sign in page: 1. Users click on the button to sign in. 2. User then enter their email, name, password and all the required information 3. User then click on the register button	Correct Input Data 1. Email: poyehtan@gmail.com 2. Name: Tan Po Yeh 3. IC: 040525141111 4. Password: 123 5. Age: 20 6. Gender: Male 7. Role: Student 8. Image: profile.png	The system will display a success message to the user. The user data will be updated to the database. Then, system will lead user to the Projectify system page	The system displays success message to the system, user data are stored and user is redirected to Projectify system page	Pass
				In the sign in page:	Incorrect Input Data 1. Email:	The system will display notice	The system displays	Pass

				1. Users click on the sign-in button 2. User then enter their email, name, password and all the required information 3. User then click on the register button	max@gmail.com 2. Name: Max 3. IC: 032013942104 4. Password: 5. Age: 20 6. Gender: Male 7. Role: Student 8. Image: note.docx	message for the user and decline the register request.	notice message and decline the register request.	
			User email has register before	In the sign in page: 1. Users click on the sign-in button 2. User then enter their email, name, password and all the required information 3. User then click on the register button	<u>Incorrect Input Data</u> 1. Email: poyehtan@gmail.com 2. Name: Tan Po Yeh 3. IC: 040525141111 4. Password: 123 5. Age: 20 6. Gender: Male 7. Role: Student 8. Image: profile.png	The system will display alert user for the repeated email and decline the register request.	The system displays alert user for the repeated email and decline the register request.	Pass
T002	Sign-In Functionality	User can access the sign-in page	User in the main page	1. Users click on the sign-in button	N/A	User access to the sign-in page	The system redirect user to the sign-in page	Pass
		Testing the sign-in	User have register account before	In the sign in page:	<u>Correct Input Data</u> 1. Email: poyehtan@gmail.com	The system will display success message to the	The system displays success	Pass

		programme logic		1. User then enter their email and password correctly 2. User then click on the sign-in button	2. Password: 123	grant access to the user	message to the system and grant access to the next page	
				In the sign in page: 1. User then enter their email and password incorrectly 2. User then click on the sign-in button	Incorrect Input Data 1. Email: poyeh@gmail.com 2. Password: tpy@kdisj?	The system will display error message and lead user to sign-in page again	The system displays error message and redirect user to sign-in page again	Pass

13.1.2 Chat System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
T003	Access Chat Functionality	User can access chat page	User is logged in	1. User clicks on the chat button	N/A	User accesses chat page	The system redirects user to the chat page	Pass
T004	Change chat partner	Users can select the chat partner	User is logged in	1. Users select the target chat partner	N/A	The system fetches the target user chat with the user	The system fetches the target user chat with the user	Pass

T005	Send Message	Test sending a message in the chat	User is on the chat page	1. User types a message 2. User clicks send button	Correct Input Data Message: Hello, World!	The message is sent and displayed in the chat window	The message is sent and displayed in the chat window	Pass
T006		Test sending an invalid message	User is on the chat page	1. User types an empty message 2. User clicks send button	Incorrect Input Data Message: ""	The system does not send the message	The system does not send the message	Pass

13.1.3 Forum System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
T007	Access Forum Page	User can access forum page	User is logged in	1. User clicks on the forum button	N/A	User accesses forum page	The system redirects user to the forum page	Pass
T008	View All Forum	User can view all the posted forum	User is logged in	1. User clicks on the "Tools" 2. User clicks on the "View All Post" button	N/A	System fetch all the post to the page	The System fetch all the post to the page	Pass
T009	View User Own Post	User can view their own posted forum	User is logged in	1. User clicks on the "Tools"	N/A	System fetch user posted forum to the page	The system fetch user posted forum to the page	Pass

				2. User clicks on the "View Own Post" button				
T010	Create Post	Test creating a forum post	User is on the forum page	1. User clicks on create post 2. User fills in post details 3. User submits the post	Correct Input Data Title: New Post Photo: Question.png Description: This is a new post.	The post is created and displayed in the forum	The post is created and displayed in the forum	Pass
		Test creating a forum post with errors	User is on the forum page	1. User clicks on create post 2. User leaves title empty 3. User submits the post	Incorrect Input Data Title: "" Photo: Question.docs Description: This is a new post.	The system displays an error message and does not create the post	The system displays an error message and does not create the post	Pass

13.1.4 Channel System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
T011	Create Channel	Test creating a new channel with valid information	User is logged in	1. User clicks on create channel 2. User fills in channel details 3. User submits the form	Channel Name: New Channel Photo: Banner.png Description: This is a new channel.	The channel is created and displayed in the user's channels	The channel is created and displayed in the user's channels	Pass

		Test creating a new channel with invalid information	User is logged in	1. User clicks on create channel 2. User fills in channel details 3. User submits the form	Channel Name: New Channel Photo: Banner.docx Description: This is a new channel.	System display error messages to the user and redirect to create channel page	The system display error messages to the user and redirect to create channel page	Pass
T012	Join Channel	Test joining an existing channel	User is logged in	1. User enters channel code 2. User clicks join button	Channel Code: 1dsh45	The user joins the channel and it is added to their channel list	The user joins the channel and it is added to their channel list	Pass
		Test joining a non-existent channel	User is logged in	1. User enters invalid channel code 2. User clicks join button	Channel Code: 000000	The system displays an error message and does not join the channel	The system displays an error message and does not join the channel	Pass

13.1.5 Material System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
T012	Create Material	Test uploading new material	User is on the material page	1. User clicks on upload material 2. User selects a file 3. User submits the form	Material Name: New Material File: document.pdf Description: This is a new material.	The material is uploaded and displayed in the material list	The material is uploaded and displayed in the material list	Pass

		Test uploading invalid material	User is on the material page	1. User clicks on upload material 2. User selects an invalid file 3. User submits the form	Material Name: New Material File: document.exe Description: This is a new material.	The system displays an error message and does not upload the material	The system displays an error message and does not upload the material	Pass
T013	Access Material	User can access the material list	User is logged in	1. User clicks on material button	N/A	System fetch material to the page	The system fetch material to the page	Pass
T014	Download Material	User can download the material	User is logged in	1. Users click on the material download button	N/A	System commands the user device will download the material	The user device downloads the material	Pass

13.1.6 Task System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
T015	Create Task	Test creating a new task	User is logged in	1. User clicks on create task 2. User fills in task details 3. User submits the form	Task title: "New Task" Description: "This is a new task." Deadline: "2024-12-31"	The task is created and displayed in the task list	The task is created and displayed in the task list	Pass

		Test creating a task with errors	User is logged in	1. User clicks on create task 2. User leaves title empty 3. User submits the form	Title: "" Description: "This is a new task." Deadline: "2024-12-31"	The system displays an error message and does not create the task	The system displays an error message and does not create the task	Pass
T016	Access Task List	Test accessing to the task list	User is logged in	1. User clicks on task button	N/A	User accesses to the task page and list	The system redirects user to the task page and list	Pass
T017	Upload Assignment to Task	Test uploading their assignment submission to the task with valid information	User is on the material list	1. Users click on the task submission button 2. Users choose the file from the device	File: document.pdf	Users upload the submission and system display success message to the user	The user uploads the submission and system display success message to the user	Pass
		Test uploading their assignment submission to the task with invalid information	User is on the material list	1. Users click on the task submission button 2. Users choose the file from the device	File: document.exe	System display error	The system display error	Pass

13.2 User Acceptance Testing

Name: Cikgu Salasiah **Signature:** Salahsiah

Question	Rating ✓ /✗				
	Worst	Bad	Neutral	Great	Excellent
Is it to sign in to the system?				✓	
Is it user friendly?					✓
How easily for lecturer managing student?				✓	
How easily for lecturer uploading material?			✓		
It is good to have chat system?					✓
It is good to have discussion forum?				✓	
How easily for student to manage their task?					✓
It is easy to use in our system?				✓	
Is it to sign up to the system?					✓

Name: Cikgu Lee Zi Hao **Signature:** Zihao

Question	Rating ✓ /✗				
	Worst	Bad	Neutral	Great	Excellent
Is it to sign in to the system?			✓		
Is it user friendly?				✓	
How easily for lecturer managing student?					✓
How easily for lecturer uploading material?					✓
It is good to have chat system?					✓
It is good to have discussion forum?					✓

How easily for student to manage their task?				✓	
It is easy to use in our system?				✓	
Is it to sign up to the system?					✓

13.2.1 User Authentication System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T001	Sign-Up Functionality	User is accessible to the "Sign Up" page	User located at the "Sign Up" page.	1. Users have to click on the button to "Sign Up" 2. It will redirect user to the "Sign Up" panel	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao				✓	
		Testing the sign-up programme logic	User email does not register the account yet	In the sign-in page: 1. Users enter their email and password 2. Alert will display "Invalid credentials!" 3. Users then click on to "Sign Up" 4. It will redirect user to "Sign Up" 5. User will then enter their email, name, password	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao					✓

				and all the required information for registration 6. User then click on the register button 7. Alert will display "Register Success!"							
			User email has register before	In the sign-up page: 1. Users then enter their email, name, password and all the required information for registration 2. Users click on the register button 3. Alert will display "Email Already Exists!"	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓
T002	Sign-In Functionality	User can access the sign-in page	User in the main page	1. Users enter the sign-up button to register 2. User will enter “Sign Up” page.	Cikgu Salasiah	Salahsiah					✓

		Testing the sign-in programme logic	User have register account before	In the sign in page: 1. Users enter their email and password correctly 2. Users click on the sign-in button 3. Alert will display "Login Success!"	Cikgu Lee Zi Hao					✓	

13.2.2 Chat System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T003	Access Chat Functionality	User can access chat page	User is logged in	1. User clicks on the chat button	Cikgu Salasiah						✓
					Cikgu Lee Zi Hao					✓	
T004	Change chat partner	Users can select the chat partner	User is logged in	1. Users select the target chat partner	Cikgu Salasiah						✓
					Cikgu Lee Zi Hao					✓	

T005	Send Message	Test sending a message in the chat	User is on the chat page	1. User types a message 2. User clicks send button 3 Alert will display "Message sent successfully."	Cikgu Salasiah	Salahsiah					
					Cikgu Lee Zi Hao	Zihao			✓		
T006		Test sending an invalid message	User is on the chat page	1. User types of an empty message 2. User clicks send button 3. Alert will display "Error: Receiver does not exist."	Cikgu Salasiah	Salahsiah			✓		
					Cikgu Lee Zi Hao	Zihao				✓	

13.2.3 Forum System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent

T007	Access Forum Page	User can access forum page	User is logged in	1. User clicks on the forum button	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao					✓
T008	View All Forum	User can view all the posted forum	User is logged in	1. Users click on the button. 2. Users click on the "View All Post".	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao				✓	
T009	View User Own Post	User can view their own posted forum	User is logged in	1. Users click on the button. 2. Users click on the button "View Own Post"	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao					✓
T010		Test creating a forum post	User is on the forum page	1. User clicks on create post 2. User fills in post details 3. User submits the post 4. Alert will display "Create Successful!"	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					

	Create Post	Test creating a forum post with image	User is on the forum page	1. User clicks on create post 2. User fills in post details 3. User submits the post 4. Alert will display "Error uploading image"	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao					✓
		Test creating a forum post with errors	User is on the forum page	1. User clicks on create post 2. User leaves title empty 3. User submits the post 4. Alert will display "Error inserting forum!" 5. Return to create post	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao				✓	

13.2.4 Channel System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent

T011	Create Channel	Test creating a new channel with valid information	Lecturer is logged in	1. User clicks on create channel 2. User fills in channel details 3. User submits the form 4. Alert will display "Create Successful!"	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	
		Test creating a new channel with invalid information	Lecturer is logged in	1. User clicks on create channel 2. User fills in channel details 3. User submits the form 4. Alert will display "Error inserting channel!"	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao				✓	
T012	Join Channel	Test joining an existing channel	Students is logged in	1. User enters channel code 2. User clicks join button	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao					

		Test joining a non-existent channel	Students is logged in	1. User enters invalid channel code 2. User clicks join button 3. Alert will display "Join Successful!"	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓
T013	Delete Channel	Ensure that a valid channelID is deleted successfully.	Lecturer is logged in	1. Set up a valid channelID 2. Submit a POST request with channelID set to the valid ID. 3. Verify that the deletion is successful by checking the database. 4. Verify that the successfully delete alert message is displayed.	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	

		Testing that the system handles deletion attempts for non-existing IDs.	A channelID that does not exist in the database.	1. Set up an invalid channelID 2. Submit a POST request with channelID set to the invalid ID. 3. Verify that an error alert message is display "Error deleting the record.".	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	
		Ensure that the system handles empty ID submissions.	Lecturer is logged in	1. Submit a POST request with an empty channelID. 2. Verify that an error alert message is displayed indicating that the ID must not be empty.	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao					✓
		Ensure that the system handles missing channelID POST parameter.	Lecturer is logged in	1. Submit a POST request without the channelID parameter. 2. Verify that an error alert message is displayed	Cikgu Salasiah	Salahsiah			✓		
					Cikgu Lee Zi Hao	Zihao					✓

				indicating that required data was not received.							

13.2.5 Material System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T014	Create Material	Test uploading new material	User is on the material page	1. User clicks on upload material 2. User selects a file 3. User submits the form 4. Alert will display "File uploaded successfully."	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓
		Test uploading invalid material	User is on the material page	1. User clicks on upload material 2. User selects an invalid file 3. User submits the form 4. Alert will display "Error: File	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓

				uploaded successfully."							
T015	Access Material	User can access the material list	User is logged in	1. User clicks on material button	Cikgu Salasiah	Salahsiah			✓		
					Cikgu Lee Zi Hao	Zihao				✓	
T016	Download Material	User can download the material	User is logged in	1. Users click on the material download button	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	

13.2.6 Task System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T017	Create Task	Test creating a new task	User is logged in	1. User clicks on create task 2. User fills in task details 3. User submits the form	Cikgu Salasiah	Salahsiah			✓		
					Cikgu Lee Zi Hao	Zihao				✓	

				4. Alert will display "Create Successful!"							
		Test creating a task with errors	User is logged in	1. User clicks on create task 2. User leaves title empty 3. User submits the form 4. Alert will display "Error inserting task!"	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao			✓		
T018	Access Task List	Test accessing to the task list	User is logged in	1. User clicks on task button	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao			✓		
T019	Upload Assignment to Task	Test uploading their assignment submission to the task with valid information	User is on the material list	1. Users click on the task submission button 2. Users choose the file from the device	Cikgu Salasiah	Salahsiah					✓

				3. Alert will display "Create Successful!"	Cikgu Lee Zi Hao	Zikhao			✓		
		Test uploading their assignment submission to the task with invalid information	User is on the material list	1. Users click on the task submission button 2. Users choose the file from the device 3. Alert will display "Error inserting task!"	Cikgu Salasiah	Salahsiah			✓		
					Cikgu Lee Zi Hao	Zikhao			✓		

13.2.7 Update User Information System

			Precondition	Steps	Test by	Sign	Rating ✓ /✗

Test ID	Test Unit	Test Objective					Worst	Bad	Neutral	Good	Excellent
T020	Update User Information Functionality	User can access the user setting page	Users is logged in	1. Users click on the setting button	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	
		Testing the users update new information with image for profile	Users is logged in	In update info page: 1. Users enter their new username, userIC, password, email, age, gender and image 2. User clicks submit button 3. Alert will display "Data updated successfully."	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓

		Test users update new information with image more than storage space	Users is logged in	In update info page: 1. Users enter their new username, userIC, password, email, age and gender and image 2. User clicks submit button 3. Alert will display "Error: File size exceeds the 2 MB limit." 4. Alert will display "Error uploading file."	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	
		Test users update new information with invalid image type	Users is logged in	In update info page: 1. Users enter their new username, userIC, password, email, age and gender and image 2. User clicks submit button 3. Alert will display "Error: Only JPG,	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓

				PNG, and GIF files are allowed." 4. Alert will display "Error uploading file."							
		Test users click submit button with no user information change	Users is logged in	In update info page: 1. User clicks submit button 2. Alert will display "No fields to update."	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓

13.2.8 Institution System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T021	Institution Functionality	Admin can access to institution page	Admin is logged in	1. Admin click on the “Institution” button 2. Admin will be redirected to institution page.							✓

T022	Add Institution Functionality	Testing admin to add new institution	Admin is logged in	In institution page: 1. Admin enter new Institution ID, Institution Name, Address, Location 2. Admin clicks submit button 3. Alert will display "Upload Successful!"	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	
		Testing admin to didn't add institution id	Admin is logged in	In institution page: 1. Admin enter new Institution ID, Institution Name, Address, Location 2. Admin clicks submit button 3. Alert will display "Error: ID must not be empty!"	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	
		Testing admin to add invalid institution	Admin is logged in	In institution page: 1. Admin enter new Institution ID,	Cikgu Salasiah	Salahsiah					✓

				Institution Name, Address, Location 2. Admin clicks submit button 3. Alert will display "Error: Required data not received."	Cikgu Lee Zi Hao	Zihao					✓
T022	Delete Institution	Ensure that a valid institutionID is deleted successfully.	Admin is logged in	In institution page: 1. Set up a valid institution ID 2. Submit a POST request with instID set to the valid ID. 3. Verify that the deletion is successful by checking the database. 4. Verify that the success alert message is displayed.	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao				✓	

		Testing that the system handles deletion attempts for non-existing IDs.	An institution ID that does not exist in the database.	1. Set up an invalid institution ID 2. Submit a POST request with instID set to the invalid ID. 3. Verify that an error alert message is display "Error deleting the record.".	Cikgu Salasiah	Salahsiah			✓		
					Cikgu Lee Zi Hao	Zihao					✓
		Ensure that the system handles empty ID submissions.	Admin is logged in	1. Submit a POST request with an empty instID. 2. Verify that an error alert message is displayed indicating that the ID must not be empty.	Cikgu Salasiah	Salahsiah				✓	
					Cikgu Lee Zi Hao	Zihao					✓
		Ensure that the system handles missing instID POST parameter.	Admin is logged in	1. Submit a POST request without the instID parameter. 2. Verify that an error alert message is displayed indicating that	Cikgu Salasiah	Salahsiah					✓
					Cikgu Lee Zi Hao	Zihao				✓	

				required data was not received.							

13.2.9 Appendix System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T023	Appendix Functionality	Admin can access to appendix page	Admin is logged in	1. Admin click on the “adminappendix” button 2. Admin will be redirected to appendix page.	Cikgu Salasiah	*Salahsiah*			✓		
T024	View Appendix	Testing admin to view appendix information	Admin is logged in	In appendix page 1. Admin can view table of ChannelImage, ChannelName, ChannelCode, StudentName, StudentIC, TaskTitle,	Cikgu Salasiah	*Salahsiah*				✓	

				Submission, SubmissionDate, Deadline, Score, AppendixID	Cikgu Lee Zi Hao	Zihao				✓	
T025	Update student score in appendix page	Ensure that a valid AppendixID and score is updated successfully.	Admin is logged in	1. Set up a valid AppendixID and Score 2. Submit a POST request with AppendixID and Score set to the valid ID. 3. Verify that the update is successful by checking the database. 4. Verify that the "Update successful"	Cikgu Salasiah	Salahsiah					✓

				alert message is displayed.							
		Testing that the system handles update attempts for non-existing IDs.	A AppendixID and Score that does not exist in the database.	1. Set up an invalid AppendixID and Score 2. Submit a POST request with AppendixID and score set to the invalid ID. 3. Verify that an error alert message is display "Error updating the record.".	Cikgu Lee Zi Hao	Zihao			✓		
		Ensure that the system handles empty ID submissions.	Admin is logged in	1. Submit a POST request with an empty AppendixID and Score. 2. An alert will display "Error: Appendix ID and score must not be empty.".	Cikgu Salasiah	Salahsiah					✓

		Ensure that the system handles missing AppendixID and Score POST parameter.	Admin is logged in	1. Submit a POST request without the AppendixID and Score parameter. 2. Verify that an error alert message is displayed indicating that required data was not received.	Cikgu Lee Zi Hao	Zihao					✓

13.2.10 Delete Member System

Test ID	Test Unit	Test Objective	Precondition	Steps	Test by	Sign	Rating ✓ /✗				
							Worst	Bad	Neutral	Good	Excellent
T026	Delete Member	Ensure that a valid MemberID is deleted successfully.	Admin is logged in	1. Set up a valid MemberID 2. Submit a POST request with MemberID set to the valid ID. 3. Verify that the deletion is successful by checking the database. 4. Verify that the “Deletion successful”	Cikgu Salasiah	Salahsiah				✓	

				alert message is displayed.							
		Testing that the system handles deletion attempts for non-existing IDs.	A MemberID that does not exist in the database.	1. Set up an invalid MemberID 2. Submit a POST request with MemberID set to the invalid ID. 3. Verify that an error alert message is display "Error deleting the record.".	Cikgu Lee Zi Hao	Zrihao			✓		
		Ensure that the system handles empty ID submissions.	Admin is logged in	1. Submit a POST request with an empty MemberID. 2. Verify that an error alert message is displayed indicating that the UserID must not be empty.	Cikgu Salasiah	Salahsiah			✓		
		Ensure that the system handles missing MemberID POST parameter.	Admin is logged in	1. Submit a POST request without the MemberID parameter. 2. Verify that an error alert message is displayed indicating	Cikgu Lee Zi Hao	Zrihao				✓	

				that required data was not received.							
T027	Delete Member from Channel	Ensure that a valid MemberID is deleted successfully.	Lecturer is logged in	1. Set up a valid MemberID 2. Submit a POST request with MemberID set to the valid ID. 3. Verify that the deletion is successful by checking the database. 4. Verify that the “Deletion successful” alert message is displayed.	Cikgu Salasiah	Salahsiah					✓
		Testing that the system handles deletion attempts for non-existing IDs.	A MemberID that does not exist in the database.	1. Set up an invalid MemberID 2. Submit a POST request with MemberID set to the invalid ID. 3. Verify that an error alert message is display "Error deleting the record.".	Cikgu Lee Zi Hao	Zihao					✓

		Ensure that the system handles empty ID submissions.	Lecturer is logged in	1. Submit a POST request with an empty MemberID. 2. Verify that an error alert message is displayed indicating that the UserID must not be empty.	Cikgu Salasiah	Salahsiah			✓		
		Ensure that the system handles missing MemberID POST parameter.	Lecturer is logged in	1. Submit a POST request without the MemberID parameter. 2. Verify that an error alert message is displayed indicating that required data was not received.	Cikgu Lee Zi Hao	Zihao			✓		

14.0 Significant Source Codes

In Projectify, most of the back-end system uses Hypertext Preprocessor (PHP) to operate. In this section, we will be showing and explaining the core source codes in this system. Systems like authentication system, chat system, forum system, channel system, material system and task system will be introduced. Below are the source codes:

14.1 User Authentication

```php
<?php
    session_start();
    include "dbConn.php";

    global $connection, $name, $email, $password, $age, $gender, $userID, $institutionID;
    $name = isset($_POST['txtname']) ? $_POST['txtname'] : "";
    $email = isset($_POST['txtemail']) ? $_POST['txtemail'] : "";
    $password = isset($_POST['txtpassword']) ? $_POST['txtpassword'] : "";

    $ic = isset($_POST['txtic']) ? $_POST['txtic'] : "";
    $age = isset($_POST['txtage']) ? $_POST['txtage'] : "";
    $gender = isset($_POST['txtgender']) ? $_POST['txtgender'] : "";
    $role = isset($_POST['txtrole']) ? $_POST['txtrole'] : "";
    $institutionID = isset($_POST['txtinstitution']) ? $_POST['txtinstitution'] : "";

    $loginEmail = isset($_POST['txtemail2']) ? $_POST['txtemail2'] : "";
    $loginPassword = isset($_POST['txtpassword2']) ? $_POST['txtpassword2'] : "";

    $signUp1ContainerStyle = "";
    $signInContainerStyle = "";
    $toggleContainerStyle = "";
    $signUp2ContainerStyle = "display: none";
```

The "authentication.php" above is responsible for handling the user sign-in and sign-up processes. First, the system collects the user information such as the name, email, password, age, gender, userID, and institutionID from the HTML form. The system then initializes the containers like signUp1ContainerStyle, signInContainerStyle, toggleContainerStyle, and signUp2ContainerStyle, which are used to control the display of different parts of the sign-up and sign-in interface.

```php
    if (isset($_POST['btnSignUp1'])) {
        $query = "SELECT * FROM `User` WHERE Email = '$email'";
        $result = mysqli_query($connection, $query);
        if ($result) {
            // Check if email already exists
            if (mysqli_num_rows($result) > 0) {
                $error_message = "Email Already Exists!";
            } else {
                if (!empty($name) && !empty($email) && !empty($password)) {
                    $_SESSION['temp1'] = $name;
                    $_SESSION['temp2'] = $email;
                    $_SESSION['temp3'] = $password;

                    // Set styles for display
                    $signUp1ContainerStyle = "display: none";
                    $signInContainerStyle = "display: none";
                    $toggleContainerStyle = "display: none";
                    $signUp2ContainerStyle = "";
                } else {
                    $error_message = "Please fill in all the fields.";
                }
            }
        } else {
            $error_message = "Database error occurred.";
        }

        if (isset($error_message)) {
            echo '<script>alert("' . $error_message . '");</script>';
        }
```

When the first "Sign Up" button (btnSignUp1) is pressed, the system will collect the username, email and password to a session variable. Then, the PHP reverses the initialized containers so the hidden container can be shown. Assume that the email is registered by the user before in the database, an alert system will notify the user that the account was registered before.

If the second "Sign Up" button (btnSignUp2) is triggered, the system will collect the remaining user registered information like age, gender, userID, and institutionID. The system verifies the data and inserts the user information into the database. If there are any errors in inserting the query, the system will alert the user and redirect them to the “Sign In” page/panel.

```
54      }elseif(isset($_POST['btnSignUp2'])){
55
56          $registerName = $_SESSION['temp1'];
57          $registerEmail = $_SESSION['temp2'];
58          $registerPassword = $_SESSION['temp3'];
59
60          // Insert new user into the database
61          $insertQuery1 = "INSERT INTO `User` (`Name`, `IC`, `Email`, `Password`, `Age`, `Gender`, `RoleID`) VALUES ('
    $registerName', '$ic', '$registerEmail', '$registerPassword','$age','$gender', '$role')";
62          if(mysqli_query($connection, $insertQuery1)){
63              $selectQuery = "SELECT UserID FROM `User` WHERE `Email` = '$registerEmail'";
64              $result = mysqli_query($connection, $selectQuery);
65              if ($result && $row = mysqli_fetch_assoc($result)) {
66                  $userID = $row['UserID'];
67                  $_SESSION['temp4'] = $userID;
68              }
69          }
70
71          $userID = $_SESSION['temp4'];
72          $insertQuery2 = "INSERT INTO `User_Institution` (`InstitutionID`, `UserID`, `Status`, `DateAdded`) VALUES ('
    $institutionID', '$userID', 'Pending', CURRENT_TIMESTAMP)";
73          if(mysqli_query($connection, $insertQuery2)){
74              saveImage($_FILES["profile"],'UserID',$userID, $connection,'User');
75              echo '<script>';
76              echo 'window.alert("Register Success!");';
77              echo 'window.location.href = "/projectify/signin.php";';
78              echo '</script>';
79          }
80
81          unset($_SESSION['temp1']);
82          unset($_SESSION['temp2']);
83          unset($_SESSION['temp3']);
84          unset($_SESSION['temp4']);
```

The button that we created "Sign In" (btnSignIn). It queries the User table for all records and iterates through them to find a matching email and password. If a match is found, it sets the user type (admin, student, or lecturer) in the session based on the RoleID. It also retrieves and stores the user's institution information in the session. If login is successful, a JavaScript alert indicates success, and the user is redirected to their respective dashboard page (admin, lecturer, or student).

```php
    }elseif(isset($_POST['btnSignIn'])){
        $query = "SELECT * FROM `User`";
        $result = mysqli_query($connection, $query);
        $matchFound = false;
        while ($row = mysqli_fetch_assoc($result)) {
            $useridData = htmlspecialchars($row['UserID'], ENT_QUOTES, 'UTF-8');
            $emailData = htmlspecialchars($row['Email'], ENT_QUOTES, 'UTF-8');
            $usernameData = htmlspecialchars($row['Name'], ENT_QUOTES, 'UTF-8');
            $passwordData = htmlspecialchars($row['Password'], ENT_QUOTES, 'UTF-8');
            $roleIndex = $row['RoleID'];

            if ($loginEmail === $emailData && $loginPassword === $passwordData) {
                $matchFound = true;
                if($roleIndex == 1){
                    $_SESSION['usertype'] = "admin";
                } elseif($roleIndex == 2){
                    $_SESSION['usertype'] = "student";
                } elseif($roleIndex == 3){
                    $_SESSION['usertype'] = "lecturer";
                }

                $_SESSION['userid'] = $useridData;
                $_SESSION['email'] = $emailData;
                $_SESSION['username'] = $usernameData;
                $_SESSION['password'] = $passwordData;

                $selectQuery2 = "SELECT * FROM `User_Institution` WHERE UserID = '$useridData'";
                $result2 = mysqli_query($connection, $selectQuery2);
                if($result2){
                    if ($row = mysqli_fetch_assoc($result2)) {
                        $institutionID = htmlspecialchars($row['InstitutionID'], ENT_QUOTES, 'UTF-8');
                    }
                }

                $selectQuery3 = "SELECT * FROM `Institution` WHERE InstitutionID = '$institutionID'";
                $result3 = mysqli_query($connection, $selectQuery3);
                if($result3){
                    if ($row = mysqli_fetch_assoc($result3)) {
                        $_SESSION['institutionname'] = htmlspecialchars($row['Name'], ENT_QUOTES, 'UTF-8');
                    }
                }

                echo '<script>';
                echo 'window.alert("Login Success!");';
                echo '</script>';

                if($_SESSION['usertype'] == "admin"){
                    echo '<script>';
                    echo 'window.location.href = "/projectify/admin.php";';
                    echo '</script>';
                } elseif($_SESSION['usertype'] == "lecturer"){
                    echo '<script>';
                    echo 'window.location.href = "/projectify/lecturer.php";';
                    echo '</script>';
```

The saveImage function updates an image in the database for a specific user. It checks if a file is uploaded, validates its type and size, and generates a unique filename. If valid, it inserts the image details into the Image table and moves the file to a target directory. Upon successful file upload, it retrieves the ImageID of the newly inserted record and updates the user's record in the specified table with this ImageID. If any errors occur during this process, it returns the corresponding error message.

php\function.php

```php
    //Update Image To Database
    function saveImage($file, $columnname, $userID, $connection, $tableName) {
        // Define allowed file types and maximum file size (in bytes)
        $allowedFileTypes = ['image/jpeg', 'image/png', 'image/gif'];
        $maxFileSize = 2 * 1024 * 1024; // 2 MB

        if (!empty($file["name"])) {
            $filename = $file["name"];
            $tempname = $file["tmp_name"];
            $filetype = $file["type"];
            $filesize = $file["size"];

            // Check if the file type is allowed
            if (!in_array($filetype, $allowedFileTypes)) {
                return "Error: Only JPG, PNG, and GIF files are allowed.";
            }

            // Check if the file size is within the limit
            if ($filesize > $maxFileSize) {
                return "Error: File size exceeds the 2 MB limit.";
            }

            // Generate a unique name for the file to avoid overwriting
            $uniqueFilename = uniqid() . "_" . basename($filename);
            $folder = "./userdata/" . $uniqueFilename;
            $formattedfilename = "userdata/" . $uniqueFilename;

            // Execute the query that needs to run before the file is moved
            $query = "INSERT INTO `Image` (`Filename`, `Filepath`, `Uploaddate`) VALUES ('Profile', '$formattedfilename', CURRENT_TIMESTAMP)";

            if (mysqli_query($connection, $query)) {
                // Move the uploaded file to the target folder
                if (move_uploaded_file($tempname, $folder)) {
                    // Fetch the last inserted ImageID using LAST_INSERT_ID()
                    $imageID = mysqli_insert_id($connection);

                    if ($imageID) {
                        $updateQuery = "UPDATE `$tableName` SET `ImageID` = '$imageID' WHERE `$columnname` = $userID";
                        if (mysqli_query($connection, $updateQuery)) {
                            return "Image updated successfully.";
                        } else {
                            return "Error updating table $tableName: " . mysqli_error($connection);
                        }
                    } else {
                        return "Error fetching image ID: " . mysqli_error($connection);
                    }
                } else {
                    return "Error: Failed to upload file.";
                }
            } else {
                return "Error executing initial query: " . mysqli_error($connection);
            }
        } else {
            return "No file uploaded.";
        }
    }
```

14.2 Chat System

The chat system is constructed by three main PHP, which is the "chat.php", "sendMessage.php" and "fetchMessage.php". The "fetchMessage.php" will fetch the message depending on the chat partner selection. It first verifies session variables for the current user ID and a selected chat partner ID from a POST request. Then, the system prepares an SQL query to fetch messages based on sender and receiver IDs and orders by timestamp. The message fetch will then be constructed, between outgoing and incoming messages based on sender IDs. Finally, it outputs the formatted HTML of messages or indicates if session variables are not set.

php\fetchMessage.php

```php
<?php
    session_start();
    require_once(__DIR__ . '/../dbConn.php');
    global $connection;

    // Check if session variables are set
    if (isset($_SESSION['userid']) && isset($_POST['selectedUserID'])) {
        $currentUserID = $_SESSION['userid'];
        $chatPartnerID = $_POST['selectedUserID'];

        // Prepare the query with placeholders
        $fetchMessagesQuery = "
            SELECT *
            FROM Message
            WHERE (SenderID = ? AND ReceiverID = ?)
            OR (SenderID = ? AND ReceiverID = ?)
            ORDER BY Timestamp ASC
        ";

        // Prepare and bind the statement
        $statement = mysqli_prepare($connection, $fetchMessagesQuery);
        mysqli_stmt_bind_param($statement, "iiii", $currentUserID, $chatPartnerID, $chatPartnerID, $currentUserID);

        // Execute the statement
        mysqli_stmt_execute($statement);

        // Get the result
        $result = mysqli_stmt_get_result($statement);

        // Initialize a variable to store the HTML
        $messagesHTML = "";

        while ($row = mysqli_fetch_assoc($result)) {
            $messageContent = htmlspecialchars($row['Content'], ENT_QUOTES, 'UTF-8');
            $timestamp = htmlspecialchars($row['Timestamp'], ENT_QUOTES, 'UTF-8');
            $senderID = $row['SenderID'];

            // Check if the message is outgoing or incoming
            if ($senderID == $currentUserID) {
                // Outgoing message
                $messageHTML = '
                <div class="outgoing-chats">
                    <div class="outgoing-msg">
                        <div class="outgoing-chats-msg">
                            <p class="multi-msg">' . $messageContent . '</p>
                            <span class="time">' . $timestamp . '</span>
                        </div>
                    </div>
                </div>';
            } elseif ($senderID == $chatPartnerID) {
                // Incoming message
                $messageHTML = '
                <div class="received-chats">
                    <div class="received-msg">
                        <div class="received-msg-inbox">
                            <p>' . $messageContent . '</p>
                            <span class="time">' . $timestamp . '</span>
                        </div>
                    </div>
                </div>';
            }

            // Append the constructed message HTML to the messagesHTML variable
            $messagesHTML .= $messageHTML;
        }

        // Echo the messages HTML
        echo $messagesHTML;
    } else {
        echo "Session variables not set.";
    }
?>
```

The second PHP, "sendMessage.php" is used to handle the sending message action from the user. This PHP script will check if a form has been submitted with a 'messageUser' parameter set. If true, it retrieves the current user's ID from the session and sanitizes inputs for receiver ID and message content from the form submission. The script then verifies if the receiver ID exists in the database. If the receiver exists, it inserts the message content along with sender and receiver IDs into our system's database. And if the receiver could not be found in our database, it will display an error panel.

php\sendMessage.php

```php
<?php
    session_start();
    require_once(__DIR__ . '/../dbConn.php');
    global $connection;

    if(isset($_POST['messageUser'])) {
        $currentUserID = $_SESSION['userid'];
        $receiverID = isset($_POST['messageUser']) ? mysqli_real_escape_string($connection, $_POST['messageUser']) : "";
        $messageContent = isset($_POST['message']) ? mysqli_real_escape_string($connection, $_POST['message']) : "";

        // Check if receiver exists
        $checkReceiverQuery = "SELECT * FROM `User` WHERE `UserID` = '$receiverID'";
        $receiverResult = mysqli_query($connection, $checkReceiverQuery);

        if (mysqli_num_rows($receiverResult) > 0) {
            // Insert message
            $insertMessageQuery = "
                INSERT INTO Message (Content, Timestamp, SenderID, ReceiverID)
                VALUES ('$messageContent', NOW(), '$currentUserID', '$receiverID')
            ";

            if (!mysqli_query($connection, $insertMessageQuery)) {
                echo "Error: " . $insertMessageQuery . "<br>" . mysqli_error($connection);
            } else {
                echo "Message sent successfully.";
            }
        } else {
            echo "Error: Receiver does not exist.";
        }
    }
?>
```

Our PHP is controlled by an Ajax JavaScript so the message can appear in real time. Whenever the drop-down input with the ID of '#messageUser' is changed, the system will trigger the fetchMessages.php to get the message record of the user with the selected message user. If a user sends a message, Ajax will trigger the sendMessage.php. The JavaScript also prevent the page from refreshing every time a message is sent so the drop-down input variable will remain at same.

chat.php

```
21      <script>
22          $(document).ready(function() {
23              // Handle form submission
24              $('#userForm').submit(function(event) {
25                  event.preventDefault();
26
27                  var formData = $(this).serialize();
28
29                  $.ajax({
30                      url: 'php/sendMessage.php',
31                      method: 'POST',
32                      data: formData,
33                      success: function(response) {
34                          console.log(response);
35
36                          // Clear the input field after successful submission
37                          $('input[name="message"]').val('');
38
39                          // Refresh the message container
40                          var selectedUserID = $('#messageUser').val();
41                          fetchMessages(selectedUserID);
42                      },
43                      error: function(xhr, status, error) {
44                          console.error(error);
45                      }
46                  });
47              });
48
49              // Handle change event for messageUser select element
50              $('#messageUser').change(function() {
51                  var selectedImage = $(this).find('option:selected').data('image');
52                  $('#userImage').attr('src', selectedImage);
53
54                  var selectedUserID = $(this).val();
55                  fetchMessages(selectedUserID);
56              });
57
58              // Fetch messages function
59              function fetchMessages(selectedUserID) {
60                  $.ajax({
61                      url: 'php/fetchMessage.php',
62                      method: 'POST',
63                      data: { selectedUserID: selectedUserID },
64                      success: function(response) {
65                          $('#messageContainer').html(response);
66                      },
67                      error: function(xhr, status, error) {
68                          console.error(error);
69                      }
70                  });
71              }
72          });
73
74
75          if ( window.history.replaceState ) {
76              window.history.replaceState(null, null, window.location.href);
77          }
78      </script>
```

14.3 Forum System

php\fetchForum.php

```php
<?php
    global $connection;
    $userID = $_SESSION['userid'];

    if(isset($_POST["viewOwn"])){
        // Fetch user forums
        $fetchForumsQuery = "
        SELECT
            f.`ForumID` AS ForumID,
            f.`Topic` AS Topics,
            f.`Content` AS Content,
            i.`Filepath` AS Image,
            f.`Timestamp` AS Timestamp,
            u.`Name` AS HostName,
            u.`UserID` AS HostID
        FROM `Forum` f
        LEFT JOIN `Image` i ON i.`ImageID` = f.`ImageID`
        LEFT JOIN `User` u ON u.`UserID` = f.`HostID`
        WHERE u.`UserID` = $userID ;
        ";
    }elseif(isset($_POST["viewAll"])){
        // Fetch all forums
        $fetchForumsQuery = "
        SELECT
            f.`ForumID` AS ForumID,
            f.`Topic` AS Topics,
            f.`Content` AS Content,
            i.`Filepath` AS Image,
            f.`Timestamp` AS Timestamp,
            u.`Name` AS HostName,
            u.`UserID` AS HostID
        FROM `Forum` f
        LEFT JOIN `Image` i ON i.`ImageID` = f.`ImageID`
        LEFT JOIN `User` u ON u.`UserID` = f.`HostID`
        ";
    }else{
        // Fetch all forums
        $fetchForumsQuery = "
        SELECT
            f.`ForumID` AS ForumID,
            f.`Topic` AS Topics,
            f.`Content` AS Content,
            i.`Filepath` AS Image,
            f.`Timestamp` AS Timestamp,
            u.`Name` AS HostName,
            u.`UserID` AS HostID
        FROM `Forum` f
        LEFT JOIN `Image` i ON i.`ImageID` = f.`ImageID`
        LEFT JOIN `User` u ON u.`UserID` = f.`HostID`
        ";
    }

    $forumsResult = mysqli_query($connection, $fetchForumsQuery);

    $forums = [];
    if ($forumsResult) {
        while ($forum = mysqli_fetch_assoc($forumsResult)) {
            $forumID = $forum['ForumID'];
            // Fetch messages for each forum
            $fetchMessagesQuery = "
            SELECT
                m.`Content` AS Message,
                m.`Timestamp` AS MessageTimestamp,
                u.`Name` AS Repliername
            FROM `Message` m
            INNER JOIN `Forum_Message` fm ON fm.`MessageID` = m.`MessageID`
            INNER JOIN `User` u ON u.`UserID` = m.`SenderID`
            WHERE fm.`ForumID` = '$forumID'
            ";
            $messagesResult = mysqli_query($connection, $fetchMessagesQuery);

            $messages = [];
            if ($messagesResult) {
                while ($message = mysqli_fetch_assoc($messagesResult)) {
                    $messages[] = $message;
                }
            }

            $forum['Messages'] = $messages;
            $forums[] = $forum;
        }
    }

    // Display forums and their messages
    foreach ($forums as $forum) {
        $forumID = htmlspecialchars($forum['ForumID'], ENT_QUOTES, 'UTF-8');
        $topics = htmlspecialchars($forum['Topics'], ENT_QUOTES, 'UTF-8');
```

The forum system is available for all the actors in the system(admin, lecturer and student). The forum system uses the "fetchForum.php" and "createForum.php".

The "fetchForum.php" is responsible for managing the display of forums based on the user selection. The code fetches the forums created by the logged-in user or all forums. Each forum's details, such as ID, topic, content, image, timestamp, hostname, and host ID are retrieved from the database. For each forum, it also fetches any responding messages. The forums and their messages are then displayed in a structured format, including options for users to post new comments. The script handles both scenarios where forums have replies and where they do not, providing an interactive interface for forum discussions.

The forum will select the query depending on the button click. User can view their own forum or all forums. The system will fetch the query based on the button pressed.

```
        $content = htmlspecialchars($forum['Content'], ENT_QUOTES, 'UTF-8');
        $image = htmlspecialchars($forum['Image'], ENT_QUOTES, 'UTF-8');
        $timestamp = htmlspecialchars($forum['Timestamp'], ENT_QUOTES, 'UTF-8');
        $hostName = htmlspecialchars($forum['HostName'], ENT_QUOTES, 'UTF-8');
        $hostID = htmlspecialchars($forum['HostID'], ENT_QUOTES, 'UTF-8');

        echo '
        <div class="response">
        <div class="response__number">'.$forumID.'</div>
        <h1 class="response__title">'.$topics.'</h1>
        <h1 class="response__title">'.$timestamp.'</h1>
        <p class="response__content">'.$content.'</p>
        <img class="response__image" src="'.$image.'"></img>

        <div class="post-group">
            <div class="post">
            <h3 class="post__author"></h3>
            <h4 class="post__timestamp"></h4>
            <p class="post__body"></p>

            <div class="post__actions">
                <div class="button button--approve">
                <i class="fa fa-thumbs-o-up"></i><i class="fa fa-thumbs-up solid"></i>
                </div>
                <div class="button button--deny">
                <i class="fa fa-thumbs-o-down"></i><i class="fa fa-thumbs-down solid"></i>
                </div>

                <div class="button button--fill comment-trigger">
                <span>Comment...</span>
                </div>

                <div class="button button--flag">
                <i class="fa fa-comment-o"></i><i class="fa fa-comment solid"></i>
                </div>
        ';

        if (empty($forum['Messages'])) {
            echo '
            <div class="post__comments">
                <div class="comment-group">
                    <div class="post">
                        <h3 class="post__author">No Replies Yet</h3>
                        <p class="post__body">Be the first to reply to this forum!</p>
                    </div>
                </div>
                <form action="php/sendComment.php" method="POST" enctype="multipart/form-data">
                    <div class="comment-form">
                        <div class="comment-form__avatar"></div>
                        <textarea name="comment" required></textarea>
                        <div class="comment-form__actions">
                            <input type="hidden" name="forumID" value="'.$forumID.'"></input>
                            <input type="hidden" name="hostID" value="'.$hostID.'"></input>
                            <input type="button" value="Cancel" class="button button--light cancel" onclick="handleCancel()"><
/input>
                            <input type="submit" value="Confirm" class="button button--confirm"></input>
                        </div>
                    </div>
                </form>
            </div>
            ';
        } else {
            foreach ($forum['Messages'] as $message) {
                $messageContent = htmlspecialchars($message['Message'], ENT_QUOTES, 'UTF-8');
                $messageTimestamp = htmlspecialchars($message['MessageTimestamp'], ENT_QUOTES, 'UTF-8');
                $replier = htmlspecialchars($message['Repliername'], ENT_QUOTES, 'UTF-8');

            echo '
            <div class="post__comments">

                <div class="comment-group">
                    <div class="post">
                        <div class="post__avatar comment__avatar"></div>
                        <h3 class="post__author">'.$replier.'</h3>
                        <h4 class="post__timestamp">'.$messageTimestamp.'</h4>
                        <p class="post__body">'.$messageContent.'</p>
                    </div>
                </div>';
            }
            echo'
            <form action="php/sendComment.php" method="POST" enctype="multipart/form-data">
                <div class="comment-form">
                    <div class="comment-form__avatar"></div>
                    <textarea name="comment" required></textarea>
                    <div class="comment-form__actions">
                        <input type="hidden" name="forumID" value="'.$forumID.'"></input>
                        <input type="hidden" name="hostID" value="'.$hostID.'"></input>
                        <input type="button" value="Cancel" class="button button--light cancel" onclick="handleCancel()"><
/input>
                        <input type="submit" value="Confirm" class="button button--confirm"></input>
```

The next "createForum.php" handles the creation of the new forum post. Whenever a user clicks the 'createPost' button (createPost), it reveals the post creation form. This form will reveal the hidden post for creating a forum post. Upon the 'createPost2' submission, it retrieves and sanitizes the form data and image. Then, the system inserts the post's title, description, timestamp, and user ID into the 'Forum' table. If an image is uploaded, it processes and inserts it, updating the forum entry with the image ID. If there is any error along the process, the user will be redirected to the discussion forum page.

php\createForum.php

```php
<?php
    include "dbConn.php";
    include "php/function.php";
    global $connection;

    $createChannel = "display: none";
    $userID = $_SESSION["userid"];

    if (isset($_POST['createPost'])) {
        $createChannel = "";
    } elseif (isset($_POST['createPost2'])) {
        $postProfile = isset($_FILES['postprofile']) ? $_FILES['postprofile'] : null;
        $postTitle = isset($_POST['txttitle']) ? mysqli_real_escape_string($connection, $_POST['txttitle']) : "";
        $postDescription = isset($_POST['txtdescription']) ? mysqli_real_escape_string($connection, $_POST['txtdescription']) : "
";

        // Insert into Forum table
        $insertQuery1 = "INSERT INTO `Forum`(`Topic`, `Content`, `Timestamp`, `HostID`) VALUES ('$postTitle', '$postDescription',
NOW(), '$userID')";

        if (mysqli_query($connection, $insertQuery1)) {
            $forumID = mysqli_insert_id($connection);

            // Handle the image upload
            if ($postProfile) {
                $imageID = insertImage($postProfile, $connection);

                if (is_numeric($imageID)) {
                    // Update the Forum table with the ImageID
                    $updateQuery = "UPDATE `Forum` SET `ImageID` = '$imageID' WHERE `ForumID` = '$forumID'";
                    if (mysqli_query($connection, $updateQuery)) {
                        echo '<script>';
                        echo 'window.alert("Create Successful!");';
                        echo 'window.location.href = "/projectify/discussionforum.php";';
                        echo '</script>';
                    } else {
                        echo '<script>';
                        echo 'window.alert("Error updating forum with image: ' . mysqli_error($connection) . '");';
                        echo 'window.location.href = "/projectify/discussionforum.php";';
                        echo '</script>';
                    }
                } else {
                    echo '<script>';
                    echo 'window.alert("Error uploading image: ' . $imageID . '");';
                    echo 'window.location.href = "/projectify/discussionforum.php";';
                    echo '</script>';
                }
            } else {
                echo '<script>';
                echo 'window.alert("Create Successful!");';
                echo 'window.location.href = "/projectify/discussionforum.php";';
                echo '</script>';
            }
        } else {
            echo '<script>';
            echo 'window.alert("Error inserting forum: ' . mysqli_error($connection) . '");';
            echo 'window.location.href = "/projectify/discussionforum.php";';
            echo '</script>';
        }

        $createChannel = "display: none";
    }
?>
```

14.4 Channel System

The channel system for lecturer and student is different in some components. Lecturers are able to create forum channels and students can only able to join channel code using the channel code the lecturer provided. Below are the "createChannel.php"

The "createChannel.php" manages the creation of a new channel. When the 'btnCreateChannel' button is clicked, it displays the channel creation form. When the second 'btnCreateChannel2' button, it retrieves and sanitizes the form data and generates a unique channel code. It then inserts the channel's code, title, description, and owner ID into the 'channel' table. An image can be uploaded and linked to the channel. If the operations are successful, it displays a success message and redirects the user to the lecturer channel page.

php\createChannel.php

```php
<?php
    include "dbConn.php";
    include "php/function.php";
    global $connection;

    $channelName = isset($_POST['txttitle']) ? mysqli_real_escape_string($connection, $_POST['txttitle']) : "";
    $channelDescription = isset($_POST['txtdescription']) ? mysqli_real_escape_string($connection, $_POST['txtdescription']) : ""
;
    $userID = $_SESSION["userid"];

    $createChannel = "display: none";

    if (isset($_POST['btnCreateChannel'])) {
        $createChannel = "";
    } elseif (isset($_POST['btnCreateChannel2'])) {

        // Insert new channel into the database
        $channelCode = getUniqueChannelCode($connection);

        $insertQuery1 = "INSERT INTO `channel`(`ChannelCode`, `Title`, `Description`,`OwnerID`) VALUES ('$channelCode', '
$channelName', '$channelDescription','$userID')";

        if (mysqli_query($connection, $insertQuery1)) {
            $channelID = mysqli_insert_id($connection);
            $result = saveImage($_FILES["channelprofile"], 'ChanID', $channelID, $connection, 'Channel');
            if ($result === "Image updated successfully.") {
                echo '<script>';
                echo 'window.alert("Create Successful!");';
                echo 'window.location.href = "/projectify/lecturerchannel.php";';
                echo '</script>';
            } else {
                echo '<script>';
                echo 'window.alert("Error: ' . $result . '");';
                echo '</script>';
            }
        } else {
            echo '<script>';
            echo 'window.alert("Error inserting channel: ' . mysqli_error($connection) . '");';
            echo '</script>';
        }
        $createChannel = "display: none";
    }
?>
```

For the student, When the 'btnJoinChannel' button is clicked, it displays the join channel form. When the second button 'btnJoinChannel2' is pressed, it retrieves and sanitizes the channel code the user key in. It will check whether the data is in the database or not. If it is found, it will input the user's ID and channel's ID into our 'user_channel' table, linking the user to the channel. If error, the user is redirected to the student channel page if the join operation is successful.

php\joinChannel.php

```php
<?php
    include "dbConn.php";
    global $connection;

    $channelCode = isset($_POST['txtcode']) ? mysqli_real_escape_string($connection, $_POST['txtcode']) : "";

    $userID = $_SESSION["userid"];
    $createChannel = "display: none";

    if (isset($_POST['btnJoinChannel'])) {
        $createChannel = "";
    } elseif (isset($_POST['btnJoinChannel2'])) {
        $fetchQuery1 = "SELECT * FROM `Channel` WHERE `ChannelCode` = '$channelCode'";
        $result1 = mysqli_query($connection, $fetchQuery1);
        if ($result1 && $row = mysqli_fetch_assoc($result1)) {
            $chanID = htmlspecialchars($row['ChanID'], ENT_QUOTES, 'UTF-8');
        }

        // Corrected the INSERT query by removing the extra comma
        $insertQuery1 = "INSERT INTO `user_channel` (`ChanID`, `MemberID`) VALUES ('$chanID', '$userID')";

        if (mysqli_query($connection, $insertQuery1)) {
            echo '<script>';
            echo 'window.alert("Join Successful!");';
            echo 'window.location.href = "/projectify/studentchannel.php";';
            echo '</script>';
        } else {
            echo '<script>';
            echo 'window.alert("Error inserting task: ' . mysqli_error($connection) . '");';
            echo '</script>';
        }
        $createChannel = "display: none";
    }
?>
```

14.5 Material System

The material system can be used only by the lecturer and student. The lecturer can create and download material in the channel while the student can only download the material from the channel. We will show the code for handling the material creation below this.

php\createMaterial.php

```php
<?php
    include "dbConn.php";
    global $connection;

    $materialTitle = isset($_POST['txttitle']) ? mysqli_real_escape_string($connection, $_POST['txttitle']) : "";
    $materialDescription = isset($_POST['txtdescription']) ? mysqli_real_escape_string($connection, $_POST['txtdescription']) : "
";

    $createMaterial = "display: none";

    if (isset($_POST['btnCreateMaterial'])) {
        $createMaterial = "";
    } elseif (isset($_POST['btnCreateMaterial2'])) {
        // Insert new material into the database
        $insertQuery1 = "INSERT INTO `material`(`Title`, `Description`, `ChanID`, `Timestamp`) VALUES ('$materialTitle','
$materialDescription','$channelID', NOW())";
        if (mysqli_query($connection, $insertQuery1)) {
            $materialID = mysqli_insert_id($connection);

            $result = saveDocument($_FILES["material"], 'MaterialID', $materialID, $connection,'Material');

            if ($result === "File uploaded successfully.") {
                echo '<script>';
                echo 'window.alert("Create Successful!");';
                echo 'window.location.href = "/projectify/lecturermaterial.php?channelID=' . $channelID . '";';
                echo '</script>';
            } else {
                echo '<script>';
                echo 'window.alert("Error: ' . $result . '");';
                echo '</script>';
            }
        } else {
            echo '<script>';
            echo 'window.alert("Error inserting channel: ' . mysqli_error($connection) . '");';
            echo '</script>';
        }
        $createChannel = "display: none";
    }
?>
```

Our script for PHP will manage the creation of new educational materials. Let's say if the 'btnCreateMaterial' button has entered, it will show the material creation form. Upon form submission with 'btnCreateMaterial2', it sanitizes and inserts the material title and description into the database, associating it with a specific channel ID. It then attempts to upload an associated file using the saveDocument function. If it works, a message will appear saying that its successful and the user is will be redirected to our materials page. If there are errors, appropriate error messages are shown.

The saveDocument() function handles the uploading and saving of a document to the server and its metadata to the database. It checks for the file that has successfully uploaded, and then it will move it to a specified directory that we created, after that it updates the database with the correct information such as file's name, type, size, and path. If the upload was successful to the database it will display a message saying it successfully updated. If not, an error message will be displayed at the user's page.

php\function.php

```
113     //Update Document To Database
114     function saveDocument($file, $columnname, $materialID ,$connection, $tableName) {
115         // Check if file is uploaded successfully
116         if ($file["error"] == UPLOAD_ERR_OK) {
117             // Get file information
118             $file_name = $file["name"];
119             $file_type = $file["type"];
120             $file_size = $file["size"];
121             $file_tmp_name = $file["tmp_name"];
122 
123             // Move uploaded file to desired location
124             $upload_dir = "userdata/";
125             $destination = $upload_dir . $file_name;
126             if (move_uploaded_file($file_tmp_name, $destination)) {
127                 // Insert file information into database
128                 $updateQuery = "UPDATE `$tableName` SET `Filename` = '$file_name', `Filetype` = '$file_type', `Filesize` = '
    $file_size', `Filecontent` = '$destination' WHERE `$columnname` = '$materialID'";
129 
130                 if (mysqli_query($connection, $updateQuery)) {
131                     return "File uploaded successfully.";
132                 } else {
133                     return "Error: " . mysqli_error($conn);
134                 }
135             } else {
136                 return "Error uploading file.";
137             }
138         } else {
139             return "Please select a file to upload.";
140         }
141     }
```

And for the next code, "fetchDocument.php" handles the display of the material according to the channel the user is in. It retrieves the materials' title, description, filename, file type, and size, determines the appropriate icon based on the file type, formats the file size for readability, and generates HTML to display each material with a download link, icon, and formatted file size.

php\fetchDocument.php

```php
<?php
    include "dbConn.php";

    // Fetch materials from the database
    $fetchQuery = "SELECT * FROM `Material` WHERE `ChanID` = '$channelID' ORDER BY `Timestamp` ASC";
    $result = mysqli_query($connection, $fetchQuery);

    // Process each material
    while ($row = mysqli_fetch_assoc($result)) {
        $title = htmlspecialchars($row['Title'], ENT_QUOTES, 'UTF-8');
        $description = htmlspecialchars($row['Description'], ENT_QUOTES, 'UTF-8');
        $filename = htmlspecialchars($row['Filename'], ENT_QUOTES, 'UTF-8');
        $filetype = htmlspecialchars($row['Filetype'], ENT_QUOTES, 'UTF-8');
        $filesize = $row['Filesize'];

        // Determine the appropriate icon based on file type
        $icon = '';
        switch ($filetype) {
            case 'application/pdf':
                $icon = 'pdf.png';
                break;
            case 'application/msword':
            case 'application/vnd.openxmlformats-officedocument.wordprocessingml.document':
            case 'application/vnd.openxmlformats-officedocument.word':
                $icon = 'word.png';
                break;
            default:
                $icon = 'file.png';
                break;
        }

        // Format file size for better readability
        $filesizeUnits = array('B', 'KB', 'MB', 'GB', 'TB');
        $size = $filesize; // Rename $filesize to $size
        $unitIndex = 0;
        while ($size >= 1024 && $unitIndex < count($filesizeUnits) - 1) {
            $size /= 1024;
            $unitIndex++;
        }
        $formattedFileSize = round($size, 2) . ' ' . $filesizeUnits[$unitIndex];

        // Output HTML for each material
        echo '
        <div class="material2">
            <h1>'.$title.'</h1>
            <p>'.$description.'</p>
            <div class="material-container">
                <div class="material-panel">
                    <a href="download.php?filename='.$filename.'">
                        <img src="image/'.$icon.'" alt="'.$filetype.'">
                    </a>
                    <span>'.$formattedFileSize.'</span>
                </div>
            </div>
        </div>
        ';
    }
?>
```

If the document container is pressed, the system will allow the user to download the material to the device. Below is the code that is responsible for downloading the document.

download.php

```php
<?php
    // Ensure the filename is provided
    if (isset($_GET['filename'])) {
        // Fetch the filename from the URL parameter
        $filename = $_GET['filename'];

        // Define the file path
        $filepath = 'userdata/' . $filename;

        // Check if the file exists
        if (file_exists($filepath)) {
            // Set the appropriate headers for file download
            header('Content-Description: File Transfer');
            header('Content-Type: application/octet-stream');
            header('Content-Disposition: attachment; filename="' . basename($filepath) . '"');
            header('Expires: 0');
            header('Cache-Control: must-revalidate');
            header('Pragma: public');
            header('Content-Length: ' . filesize($filepath));
            // Read the file and output its contents
            readfile($filepath);
            // Exit the script
            exit;
        } else {
            // If the file doesn't exist, output an error message
            echo 'File not found.';
        }
    } else {
        // If the filename parameter is not provided, output an error message
        echo 'Filename parameter is missing.';
    }
?>
```

The PHP script handles file downloads. It checks if a filename is provided via a URL parameter and constructs the file path. Then, the code sets the appropriate headers to facilitate the file download. If the file exists, it reads and outputs the file content for download; otherwise, it displays an error message indicating that the file is not found.

14.6 Task System

For the task system, the lecturer can create course tasks for the student while the student can upload the course submission(appendix). The "createTask.php", are used for handling this system.

The "createTask.php" script manages the creation of tasks in a database. When the "btnCreateAssignment" button is pressed, it retrieves the task title, description, and deadline from POST parameters and inserts the lecturer key in the task into the database. Then, the system displays a success or error message. If successful, it redirects back to the lecturer's assignment page.

php\createTask.php

```php
<?php
    include "dbConn.php";
    global $connection;

    $taskTitle = isset($_POST['txttitle']) ? mysqli_real_escape_string($connection, $_POST['txttitle']) : "";
    $taskDescription = isset($_POST['txtdescription']) ? mysqli_real_escape_string($connection, $_POST['txtdescription']) : "";
    $taskDeadline = isset($_POST['txtdeadline']) ? mysqli_real_escape_string($connection, $_POST['txtdeadline']) : "";

    $createTask = "display: none";

    if (isset($_POST['btnCreateAssignment'])) {
        $createTask = "";
    } elseif (isset($_POST['btnCreateAssignment2'])) {
        // Insert new task into the database
        $insertQuery1 = "INSERT INTO `task`(`Title`, `Description`, `Deadline`, `ChanID`) VALUES ('$taskTitle','$taskDescription'
, '$taskDeadline', '$channelID')";
        if (mysqli_query($connection, $insertQuery1)) {
            echo '<script>';
            echo 'window.alert("Create Successful!");';
            echo 'window.location.href = "/projectify/lecturerassignment.php?channelID=' . $channelID . '";';
            echo '</script>';
        } else {
            echo '<script>';
            echo 'window.alert("Error inserting task: ' . mysqli_error($connection) . '");';
            echo '</script>';
        }
        $createTask = "display: none";
    }
?>
```

For students, the PHP that handles the submission upload is the "fetchTask2.php". The code begins by including a database connection and initializing variables based on user session data and GET parameters. Tasks associated with a specific channel are fetched from the database and processed in a loop. For each task, it checks if the user has submitted any files related to the task. Depending on the submission status ("Submitted" or not), the script generates HTML to display task details and either a file upload form for submission or a view of the submitted file with options to update.

The script handles file uploads through forms and updates the database accordingly using SQL queries. It distinguishes file types (e.g., PDF, Word documents) to display appropriate icons. Upon successful updates or uploads, the script triggers alerts and redirects the user to the relevant page. Error handling is also implemented to manage database operations and file uploads, ensuring smooth user interaction throughout the assignment management process.

php\fetchTask2.php

```php
<?php
    include "dbConn.php";
    global $connection, $icon;

    $channelID = $_GET['channelID'];
    $userID = $_SESSION["userid"];

    // Fetch tasks from the database
    $fetchQuery1 = "SELECT * FROM `Task` WHERE `ChanID` = '$channelID' ORDER BY `Deadline` ASC";
    $result1 = mysqli_query($connection, $fetchQuery1);

    // Process each task
    while ($row = mysqli_fetch_assoc($result1)) {
        $title = htmlspecialchars($row['Title'], ENT_QUOTES, 'UTF-8');
        $description = htmlspecialchars($row['Description'], ENT_QUOTES, 'UTF-8');
        $deadline = htmlspecialchars($row['Deadline'], ENT_QUOTES, 'UTF-8');

        $taskID = $row['TaskID'];

        // Check if the user has submitted the task
        $fetchQuery2 = "
        SELECT ut.*, a.*
        FROM `User_Task` ut
        JOIN `Appendix` a ON ut.AppendixID = a.AppendixID
        WHERE ut.TaskID = '$taskID' AND a.UserID = '$userID'
        ";
        $result2 = mysqli_query($connection, $fetchQuery2);
        $row2 = mysqli_fetch_assoc($result2);
        $status = ($row2) ? htmlspecialchars($row2['Status'], ENT_QUOTES, 'UTF-8') : '';
        $appendixID = ($row2) ? htmlspecialchars($row2['AppendixID'], ENT_QUOTES, 'UTF-8') : '';
        $filename = ($row2) ? htmlspecialchars($row2['Filename'], ENT_QUOTES, 'UTF-8') : '';
        $filetype = ($row2) ? htmlspecialchars($row2['Filetype'], ENT_QUOTES, 'UTF-8') : '';

        // Determine the appropriate icon based on file type
        $icon = '';
        switch ($filetype) {
            case 'application/pdf':
                $icon = 'pdf.png';
                break;
            case 'application/msword':
            case 'application/vnd.openxmlformats-officedocument.wordprocessingml.document':
            case 'application/vnd.openxmlformats-officedocument.word':
                $icon = 'word.png';
                break;
            default:
                $icon = 'empty.png';
                break;
        }

        // Determine the appropriate HTML based on the status
        if ($status == "Submitted") {
            echo '
            <div class="material2">
                <h1>'.$title.'</h1>
                <p>'.$description.'</p>
                <p>Deadline: '.$deadline.'</p>
                <div class="material-container">
                    <div class="material-panel" style="margin-right: 1%;">
                        <img src="image/'.$icon.'" alt="'.$filetype.'">
                        <span>'.$filename.'</span>
                    </div>
                    <form action="" method="POST" enctype="multipart/form-data" style="text-decoration: none;">
                        <input type="hidden" name="channelID" value="'.$channelID.'">
                        <input type="hidden" name="taskID" value="'.$taskID.'">
                        <input type="hidden" name="oldAppendixID" value="'.$appendixID.'">
                        <div style="display: flex; flex-direction: column;">
                            <input type="file" name="updateFile" id="fileInput'.$taskID.'" required>
                            <input type="submit" name="btnUpdateFile" id="submitBtn'.$taskID.'" style="background-color:
darkgray; color: white;"
                            style="
                            ></input>
                        </div>
                    </form>
                </div>
            </div>
            ';
        } else {
            echo '
            <div class="material2">
                <h1>'.$title.'</h1>
                <p>'.$description.'</p>
                <p>Deadline: '.$deadline.'</p>
                <div class="material-container">
                    <div class="material-panel" style="margin-right: 1%;">
                        <img style="max-height: 55px; width: auto;" src="image/'.$icon.'" alt="'.$filetype.'">
                        <span style="font-size: 12px;">No Submission</span>
                    </div>
```

```
                    <div>
                        <form action="" method="POST" enctype="multipart/form-data" style="text-decoration: none;">
                            <input type="hidden" name="taskID" value="'.$taskID.'">
                            <input type="hidden" name="userID" value="'.$userID.'">
                            <div style="display: flex; flex-direction: column;">
                                <input type="file" name="uploadFile" accept=".pdf,.doc,.docx,.xls,.xlsx,.ppt,.pptx" required>
                                <input type="submit" name="btnUploadFile" id="submitBtn'.$taskID.'"></input>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
            ';
        }
    }

    if (isset($_POST['btnUpdateFile'])) {
        $taskID = isset($_POST['taskID']) ? mysqli_real_escape_string($connection, $_POST['taskID']) : "";
        $oldAppendixID = isset($_POST['oldAppendixID']) ? mysqli_real_escape_string($connection, $_POST['oldAppendixID']) : "";

        $query1 = "INSERT INTO Appendix(`Timestamp`, `UserID`) VALUES (Now(),'$userID')";
            if (mysqli_query($connection, $query1)) {
                $appendixID = mysqli_insert_id($connection);
                saveDocument($_FILES["updateFile"], 'AppendixID', $appendixID, $connection,'Appendix');

                $query2 = "UPDATE user_task SET AppendixID = '$appendixID' WHERE TaskID = '$taskID' AND AppendixID = '
$oldAppendixID'";
                if (mysqli_query($connection, $query2)) {
                    echo '<script>';
                    echo 'window.alert("Update Successful!");';
                    echo 'window.location.href = "/projectify/studentassignment.php?channelID=' . $channelID . '";';
                    echo '</script>';
                }
            } else {
            echo '<script>';
            echo 'window.alert("Error inserting into user_task: ' . mysqli_error($connection) . '");';
            echo 'window.location.href = "/projectify/studentassignment.php?channelID=' . $channelID . '";';
            echo '</script>';
        }
        exit();
    }

    if (isset($_POST['btnUploadFile'])) {
        $taskID = isset($_POST['taskID']) ? mysqli_real_escape_string($connection, $_POST['taskID']) : "";
        $userID = isset($_POST['userID']) ? mysqli_real_escape_string($connection, $_POST['userID']) : "";

        $query1 = "INSERT INTO Appendix(`Timestamp`, `UserID`) VALUES (Now(),'$userID')";
            if (mysqli_query($connection, $query1)) {
                $appendixID = mysqli_insert_id($connection);
                saveDocument($_FILES["uploadFile"], 'AppendixID', $appendixID, $connection,'Appendix');

                $query2 = "INSERT INTO user_task (TaskID, AppendixID, Status) VALUES ('$taskID', '$appendixID', 'Submitted')";
                if (mysqli_query($connection, $query2)) {
                    echo '<script>';
                    echo 'window.alert("Upload Successful!");';
                    echo 'window.location.href = "/projectify/studentassignment.php?channelID=' . $channelID . '";';
                    echo '</script>';
                }
            } else {
            echo '<script>';
            echo 'window.alert("Error inserting into user_task: ' . mysqli_error($connection) . '");';
            echo 'window.location.href = "/projectify/studentassignment.php?channelID=' . $channelID . '";';
            echo '</script>';
        }
        exit();
    }
?>
```

15.0 Conclusion

This section will be covering our state practical expectation on what we have issued, and we will be outlining the observed issues which our system is unable to address and also addressing the necessary needs in future enhancements during the project of our new system "Projectify".

Assumptions

- **User Adaptation:** Users such as student, lecture and administrator are expected to adapt rapidly to our new system which is "Projectify " due to its simple layout and intensive user training session.
- **Resource Availability:** We expected that enough resources including developers, testers, and maintenance staff, throughout the project to ensure it is completed on time and with good quality.
- **System Integration:** We believe our new system "Projectify" will integrate seamlessly with our university, such as educational resources, student information lecture information and etc.
- **Data Security:** It is expected that our new system "Projectify" will have a strong data protection measure to safeguard user information and ensures that our system comply with necessary privacy rules that is set.
- **Scalability:** We believe our new system "Projectify" will rapidly adapt to fast user growth, the system should maintain user load without having to face substantial performance concerns.

Limitations

- **Additional Feature:** Our new current system is lacking an advanced data search feature for the chat and forum system. This makes it, challenging for users such as student and lecture to quickly locate specific information or past discussions.
- **File Management:** The channel system only allow lecturers to append one file only at a time in the channel system, which can cause difficulty to share various information and document efficiently .
- **Admin Database Access:** The administrator has limited access to the database, limiting their access to perform certain activities and manging certain data effectively.
- **Scalability:** The fast-paced user growth may affect our system performance, that potentially need regularly monitoring and upgrades

- **User Training:** More materials like education resources are needed ensuring all users can effectively learn and use our new system's features.
- **Lack of staff:** Restricted development, testing and maintenance staff may influence the system's reliability and quality, therefore making user in unsatisfactory manner.

Enhancements

- **Advanced Additional Features:** Implementing an advanced search into the chat and forum systems will allow users to rapidly locate documents and find relevant information
- **Improvised File Management:** Allowing lectures to append various files to the channel system would boost resource sharing and relevant information to the student
- **Extend Admin Access:** Allowing administrators more comprehensive database access will boost data management and support enable various administrative tasks
- **Scalability Solutions:** Utilizing load management and server optimization can help control system efficiency through periods of rapidly user expansion.
- **Enhanced User Training:** Establishing better resources to user training will allow users such as student and lectures to adapt more easily to the new system "Projectify".
- **Security Enhancements:** Regularly establishing data protection procedures will ensure the long-term security of user information and complying with our privacy rules.

In a nutshell "Projectify" hopes to transform the management of Final Year Projects by offering a straightforward interface that improves communication and productivity for students and lectures. Although there are limitations and challenges, overcoming them with future developments will ensure long-term success and boost users' satisfaction.

16.0 References

Data flow diagram examples, symbols, types, and tips. (2019). Retrieved from LucidChart: https://www.lucidchart.com/blog/data-flow-diagram-tutorial

Laoyan, S. (2024, February 2). *What is Agile methodology? (A beginner's guide)*. Retrieved from asana: https://asana.com/resources/agile-methodology

Mughal, A. (2024). *HTML Code For Discussion Forum*. Retrieved from Code Him: https://www.codehim.com/html5-css3/html-code-for-discussion-forum/

Nava, G. (2021). *How to Create a Simple Web-Based Chat Application*. Retrieved from Envato Tuts+: https://code.tutsplus.com/how-to-create-a-simple-web-based-chat-application--net-5931t

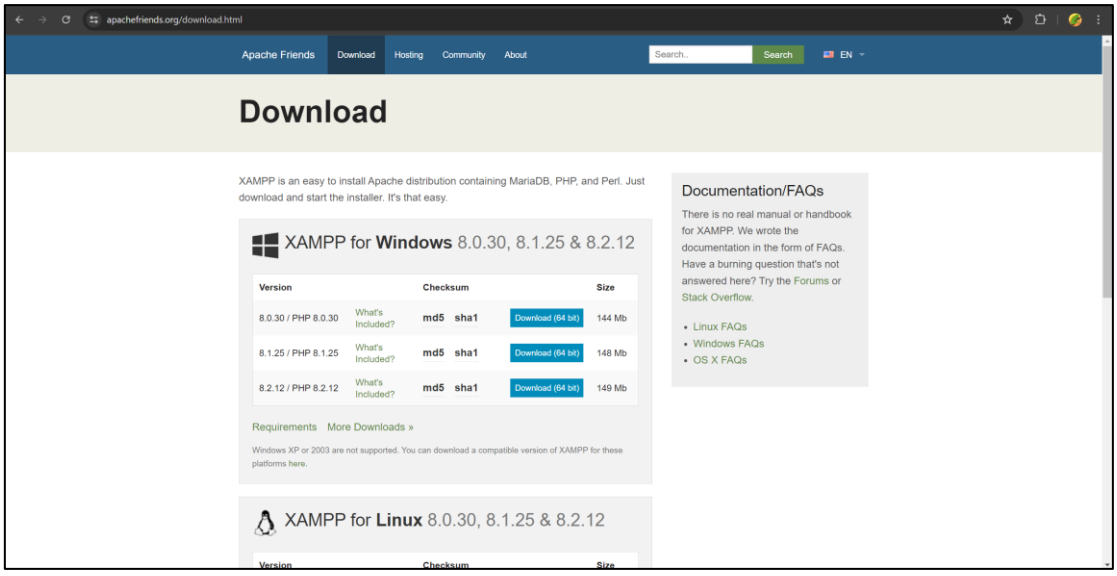
Students, M. (2022, February 11). *Agile Methodology: Advantages and Disadvantages*. Retrieved from University Of Minnesota : https://ccaps.umn.edu/story/agile-methodology-advantages-and-disadvantages

Verma, R. (2024, April 18). *Building a Chat Interface using HTML and CSS - CSS Projects*. Retrieved from Scaler Topics: https://www.scaler.com/topics/chat-interface-project-css/

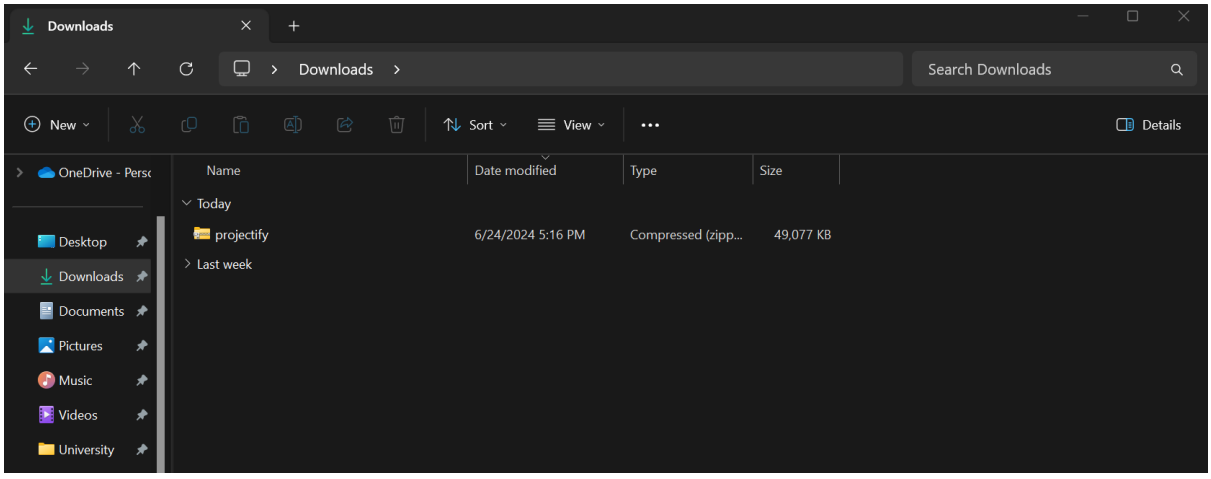
17.0 Appendix

Installation Guide

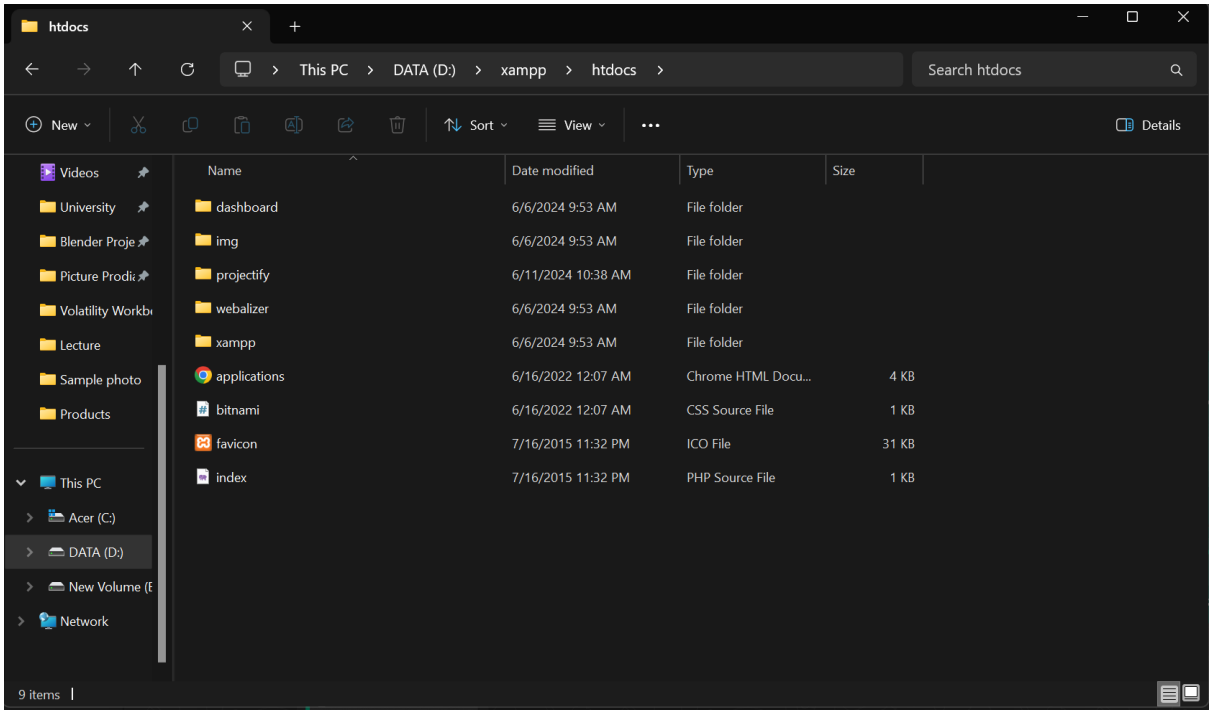
In order to execute the Projectify system, user will need to download Xampp, a server for web-based system provided by Apache Friends. To download the application, go to the website (https://www.apachefriends.org/download.html) and download the suitable version and type for the pc.



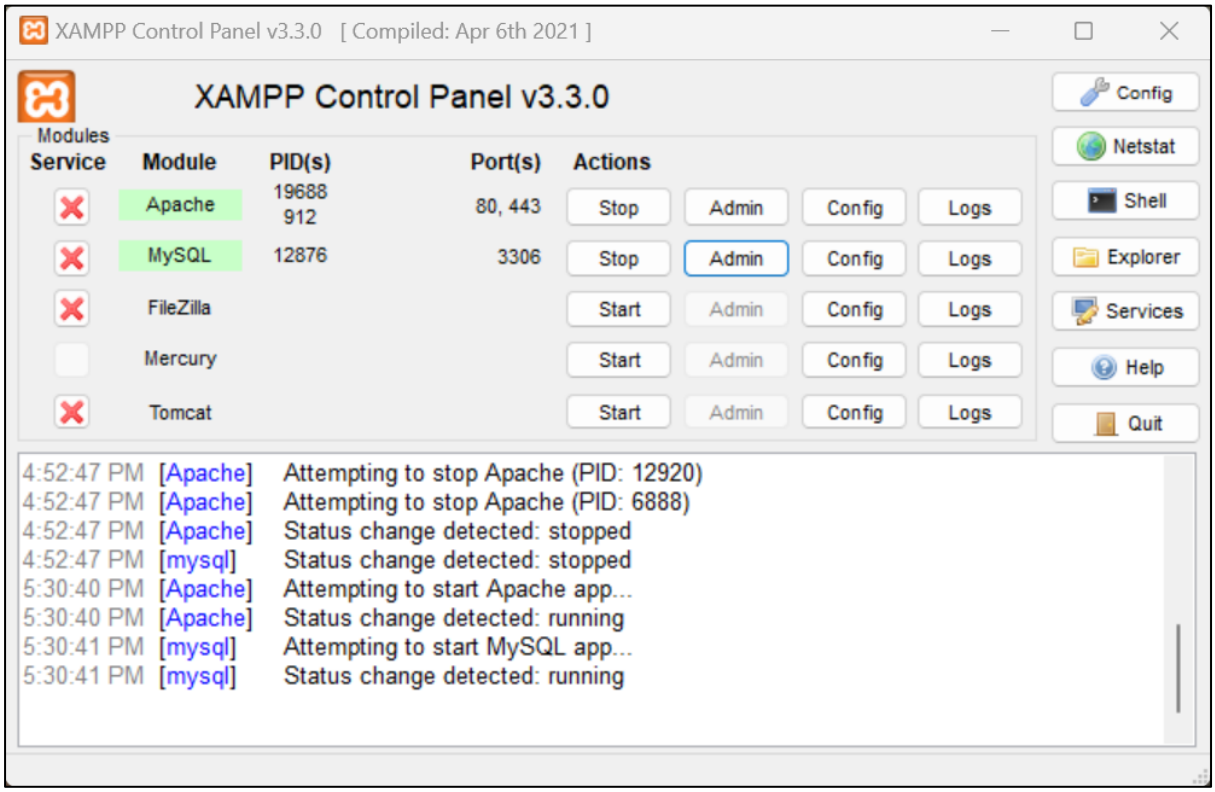
Then, follow the installation instruction of the Xampp to setup the server for the Projectify System. Once the Xampp installation is done, we will proceed the next step by download the Projectify.zip from the source like the picture shown below.



After all the installation is done, you will need to extract the Projectify.zip to the ../xampp/htdocs just like the figure shown below.

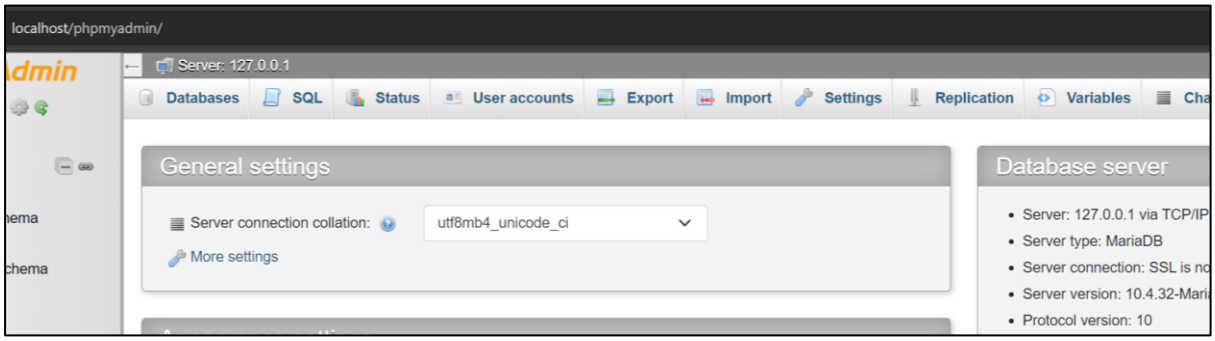


After that, run the Xampp applications and suit on the Apache and MySQL module. You will able to see the green status once the module is on.

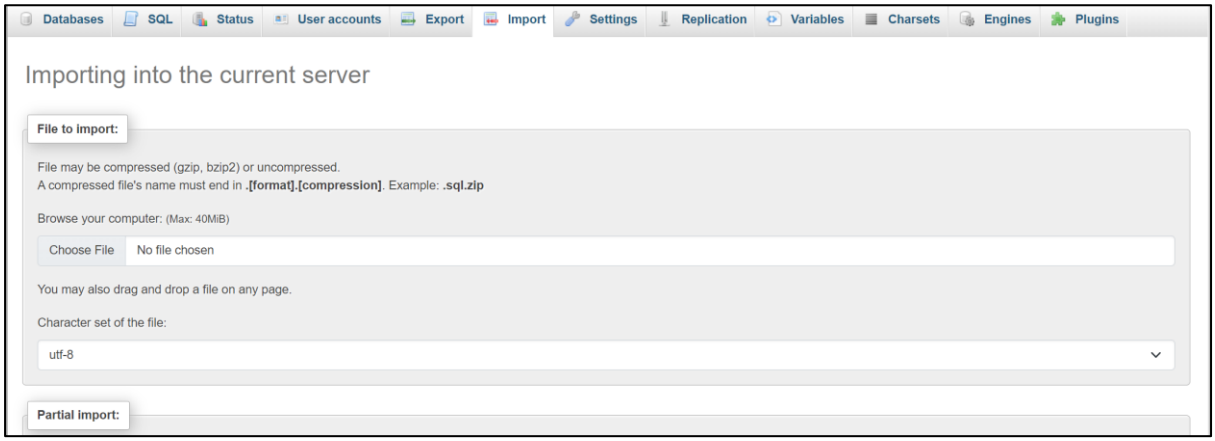


Then, click the "Admin" button that located on the same column with "MySQL". The system will redirect user to the database of the system.

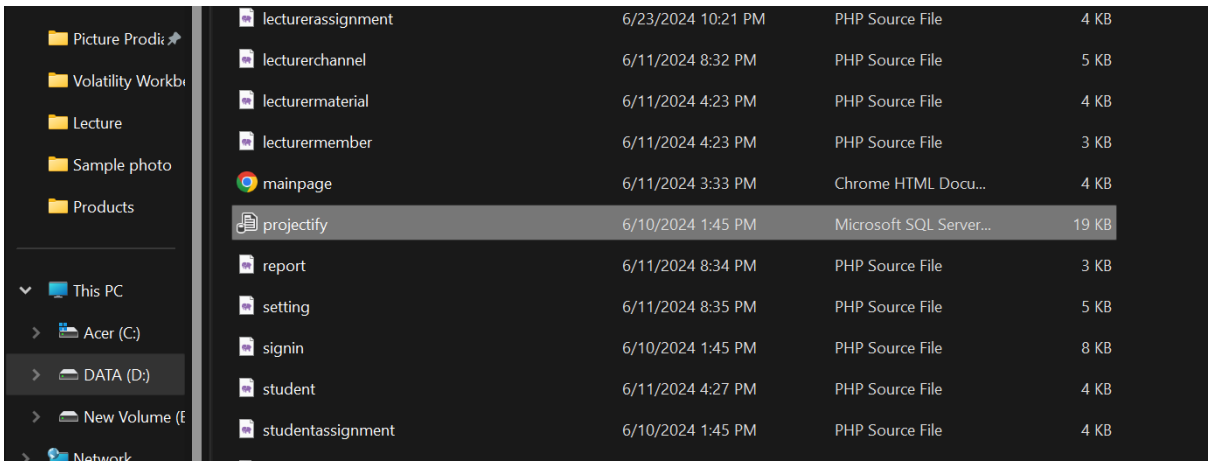
Within it, click the "Import" button located on the top middle section of phpmyadmin.



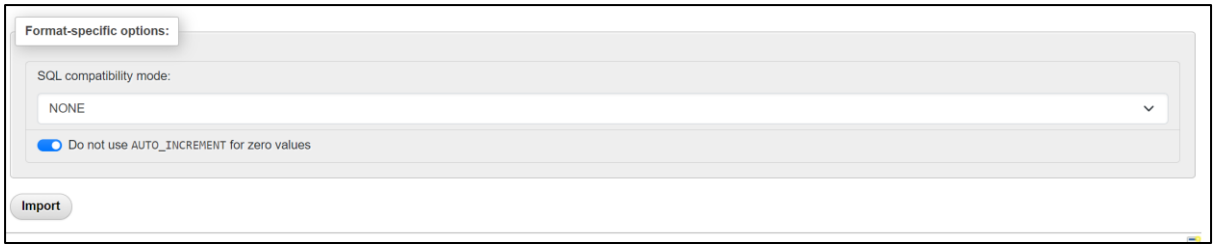
Then, click the "Choose File" button on the first panel.



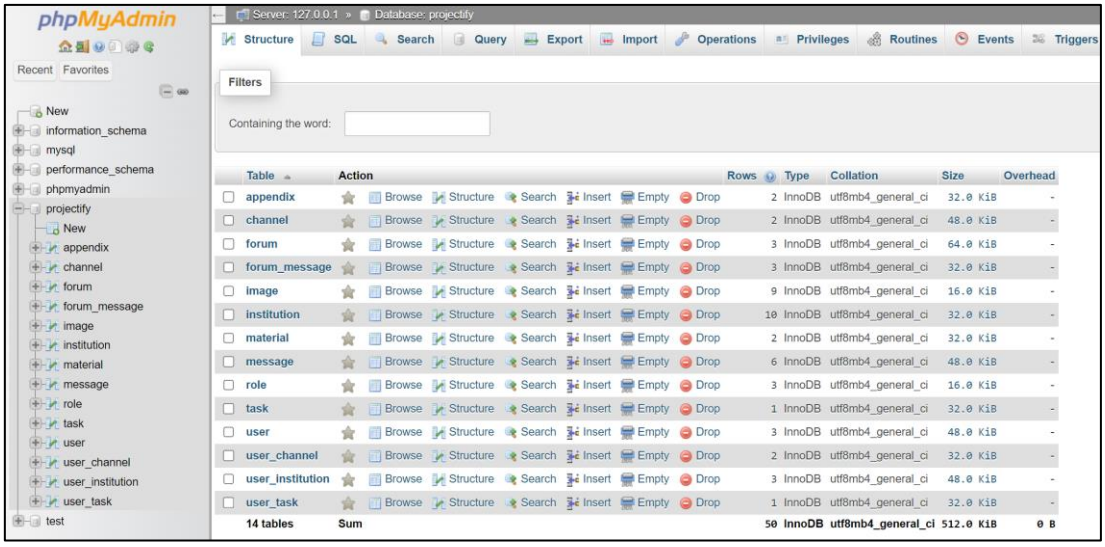
Choose the projectify.sql file from file we extracted. The projectify.sql should be show like the picture show below.



Lastly, scroll down to the bottom page and click the "Import" button.



If everything is setup correctly, user will able to see a new database called "projectify" is created on the left section of the panel.



Through following the instruction, we mention above, we are able to access the Projectify website by typing localhost/Projectify/mainpage.html in a web browser like the picture show below