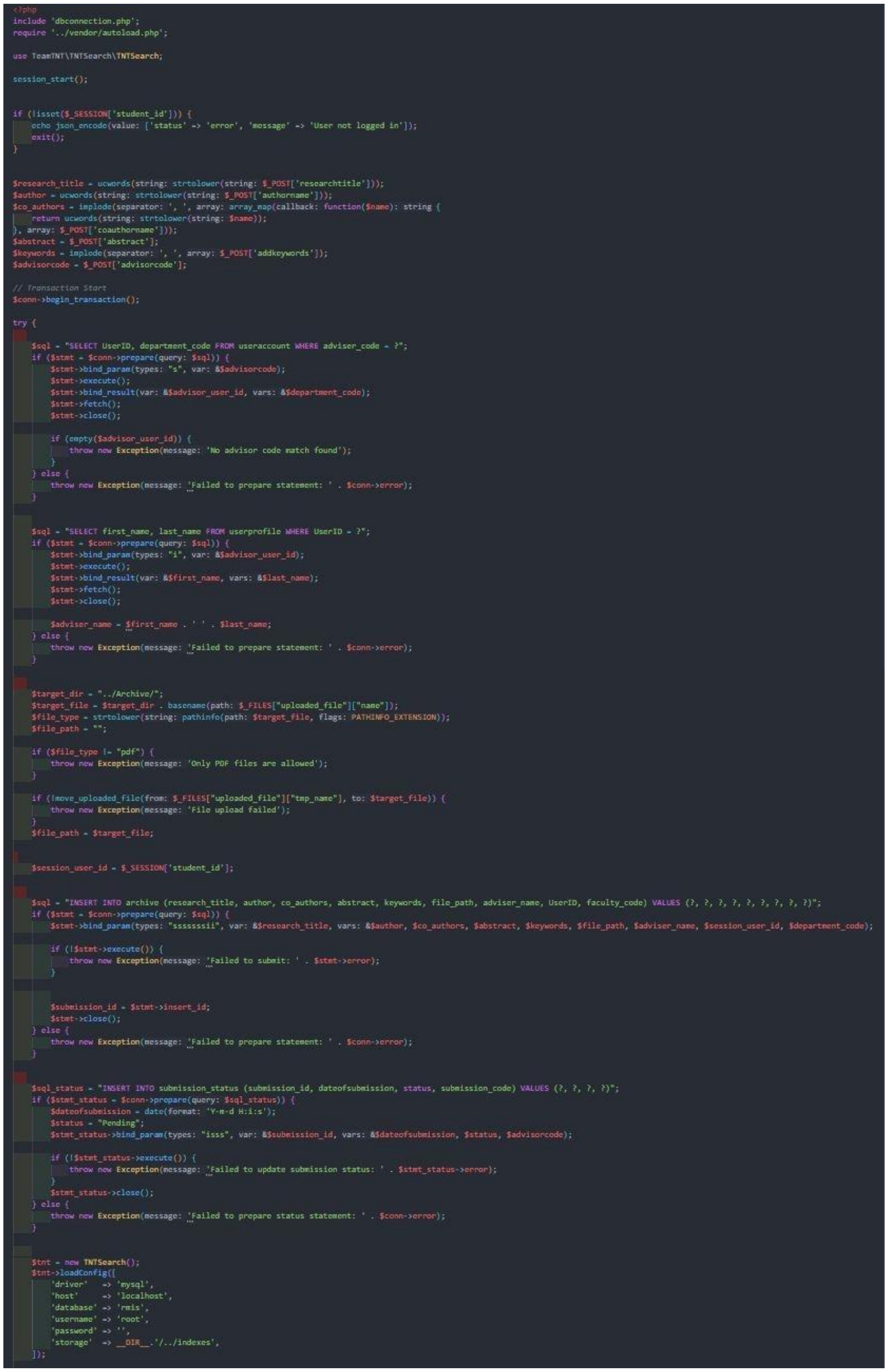
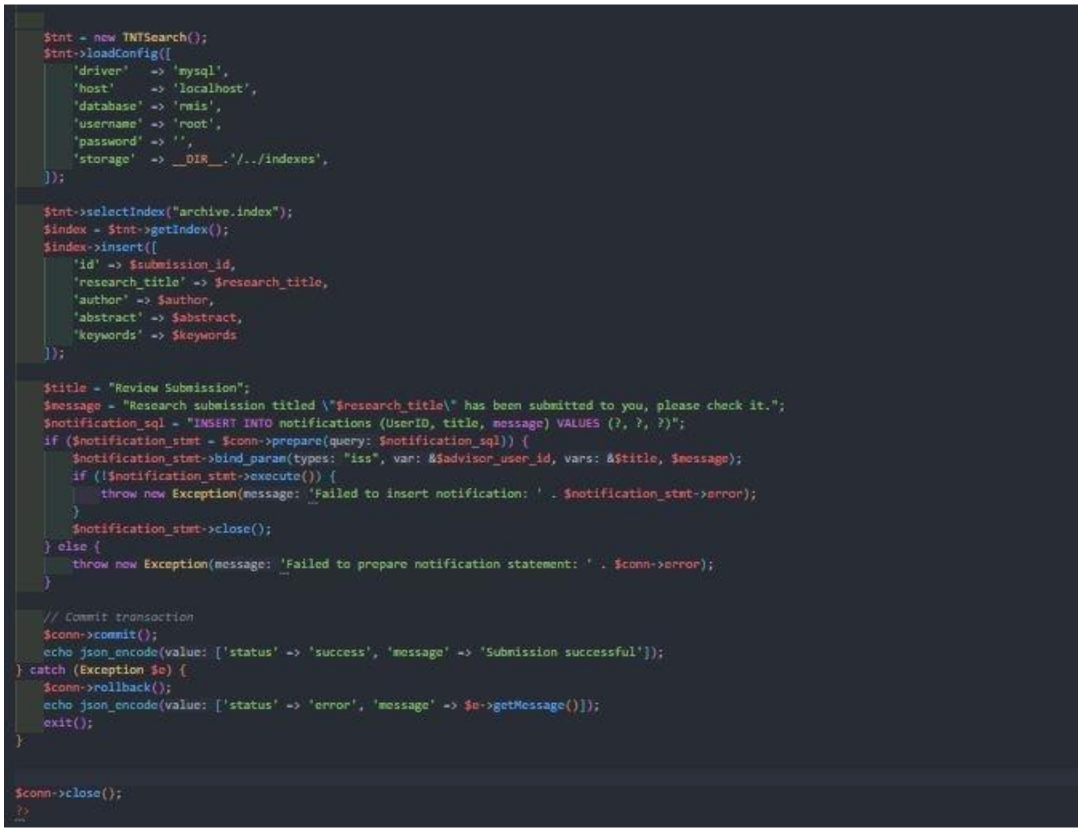
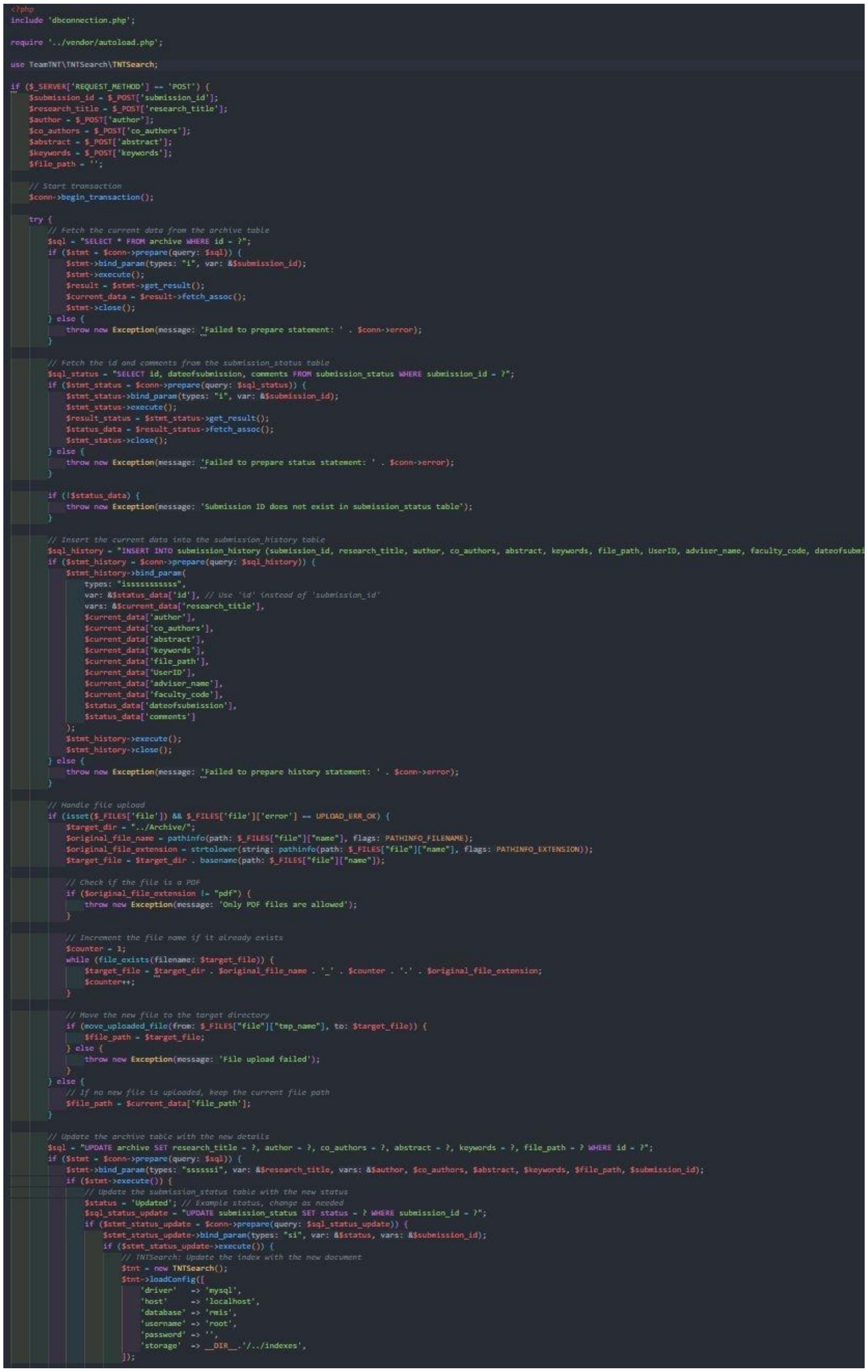
SOURCE CODE



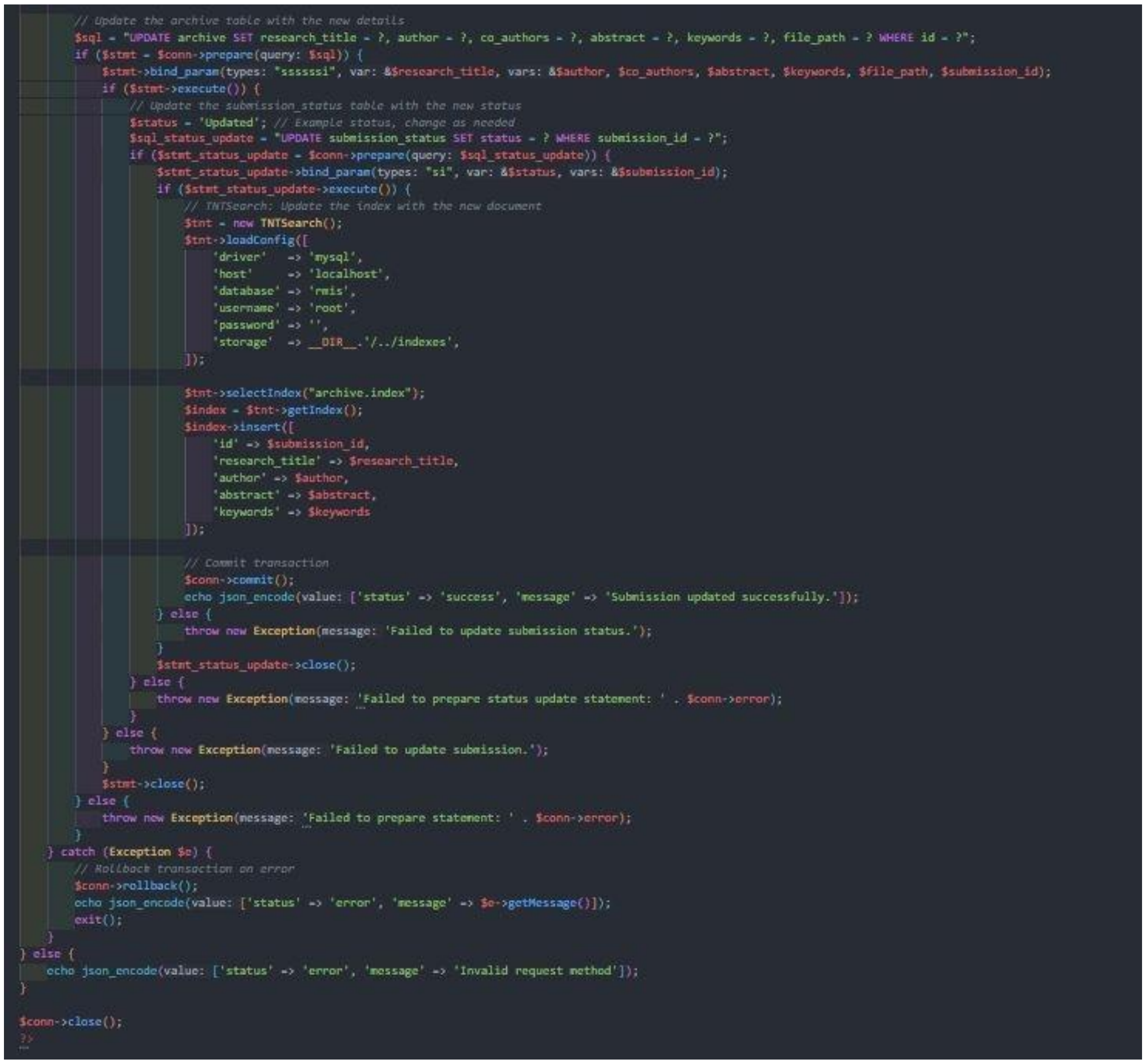
```php
<?php
include 'dbconnection.php';
require '../vendor/autoload.php';

use TeamTNT\TNTSearch\TNTSearch;

session_start();


if (!isset($_SESSION['student_id'])) {
    echo json_encode(value: ['status' => 'error', 'message' => 'User not logged in']);
    exit();
}


$research_title = ucwords(string: strtolower(string: $_POST['researchtitle']));
$author = ucwords(string: strtolower(string: $_POST['authorname']));
$co_authors = implode(separator: ', ', array: array_map(callback: function($name): string {
    return ucwords(string: strtolower(string: $name));
}, array: $_POST['coauthorname']));
$abstract = $_POST['abstract'];
$keywords = implode(separator: ', ', array: $_POST['addkeywords']);
$advisorcode = $_POST['advisorcode'];

// Transaction Start
$conn->begin_transaction();

try {

    $sql = "SELECT UserID, department_code FROM useraccount WHERE adviser_code = ?";
    if ($stmt = $conn->prepare(query: $sql)) {
        $stmt->bind_param(types: "s", var: &$advisorcode);
        $stmt->execute();
        $stmt->bind_result(var: &$advisor_user_id, vars: &$department_code);
        $stmt->fetch();
        $stmt->close();

        if (empty($advisor_user_id)) {
            throw new Exception(message: 'No advisor code match found');
        }
    } else {
        throw new Exception(message: 'Failed to prepare statement: ' . $conn->error);
    }


    $sql = "SELECT first_name, last_name FROM userprofile WHERE UserID = ?";
    if ($stmt = $conn->prepare(query: $sql)) {
        $stmt->bind_param(types: "i", var: &$advisor_user_id);
        $stmt->execute();
        $stmt->bind_result(var: &$first_name, vars: &$last_name);
        $stmt->fetch();
        $stmt->close();

        $adviser_name = $first_name . ' ' . $last_name;
    } else {
        throw new Exception(message: 'Failed to prepare statement: ' . $conn->error);
    }


    $target_dir = "../Archive/";
    $target_file = $target_dir . basename(path: $_FILES["uploaded_file"]["name"]);
    $file_type = strtolower(string: pathinfo(path: $target_file, flags: PATHINFO_EXTENSION));
    $file_path = "";

    if ($file_type != "pdf") {
        throw new Exception(message: 'Only PDF files are allowed');
    }

    if (!move_uploaded_file(from: $_FILES["uploaded_file"]["tmp_name"], to: $target_file)) {
        throw new Exception(message: 'File upload failed');
    }
    $file_path = $target_file;


    $session_user_id = $_SESSION['student_id'];


    $sql = "INSERT INTO archive (research_title, author, co_authors, abstract, keywords, file_path, adviser_name, UserID, faculty_code) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
    if ($stmt = $conn->prepare(query: $sql)) {
        $stmt->bind_param(types: "sssssssii", var: &$research_title, vars: &$author, $co_authors, $abstract, $keywords, $file_path, $adviser_name, $session_user_id, $department_code);

        if (!$stmt->execute()) {
            throw new Exception(message: 'Failed to submit: ' . $stmt->error);
        }


        $submission_id = $stmt->insert_id;
        $stmt->close();
    } else {
        throw new Exception(message: 'Failed to prepare statement: ' . $conn->error);
    }


    $sql_status = "INSERT INTO submission_status (submission_id, dateofsubmission, status, submission_code) VALUES (?, ?, ?, ?)";
    if ($stmt_status = $conn->prepare(query: $sql_status)) {
        $dateofsubmission = date(format: 'Y-m-d H:i:s');
        $status = "Pending";
        $stmt_status->bind_param(types: "isss", var: &$submission_id, vars: &$dateofsubmission, $status, $advisorcode);

        if (!$stmt_status->execute()) {
            throw new Exception(message: 'Failed to update submission status: ' . $stmt_status->error);
        }
        $stmt_status->close();
    } else {
        throw new Exception(message: 'Failed to prepare status statement: ' . $conn->error);
    }


    $tnt = new TNTSearch();
    $tnt->loadConfig([
        'driver'   => 'mysql',
        'host'     => 'localhost',
        'database' => 'rmis',
        'username' => 'root',
        'password' => '',
        'storage'  => __DIR__.'/../indexes',
    ]);
```



```php

    $tnt = new TNTSearch();
    $tnt->loadConfig([
        'driver'    => 'mysql',
        'host'      => 'localhost',
        'database' => 'rmis',
        'username' => 'root',
        'password' => '',
        'storage'  => __DIR__.'/../indexes',
    ]);

    $tnt->selectIndex("archive.index");
    $index = $tnt->getIndex();
    $index->insert([
        'id' => $submission_id,
        'research_title' => $research_title,
        'author' => $author,
        'abstract' => $abstract,
        'keywords' => $keywords
    ]);

    $title = "Review Submission";
    $message = "Research submission titled \"$research_title\" has been submitted to you, please check it.";
    $notification_sql = "INSERT INTO notifications (UserID, title, message) VALUES (?, ?, ?)";
    if ($notification_stmt = $conn->prepare(query: $notification_sql)) {
        $notification_stmt->bind_param(types: "iss", var: &$advisor_user_id, vars: &$title, $message);
        if (!$notification_stmt->execute()) {
            throw new Exception(message: 'Failed to insert notification: ' . $notification_stmt->error);
        }
        $notification_stmt->close();
    } else {
        throw new Exception(message: 'Failed to prepare notification statement: ' . $conn->error);
    }

    // Commit transaction
    $conn->commit();
    echo json_encode(value: ['status' => 'success', 'message' => 'Submission successful']);
} catch (Exception $e) {
    $conn->rollback();
    echo json_encode(value: ['status' => 'error', 'message' => $e->getMessage()]);
    exit();
}


$conn->close();
?>
```



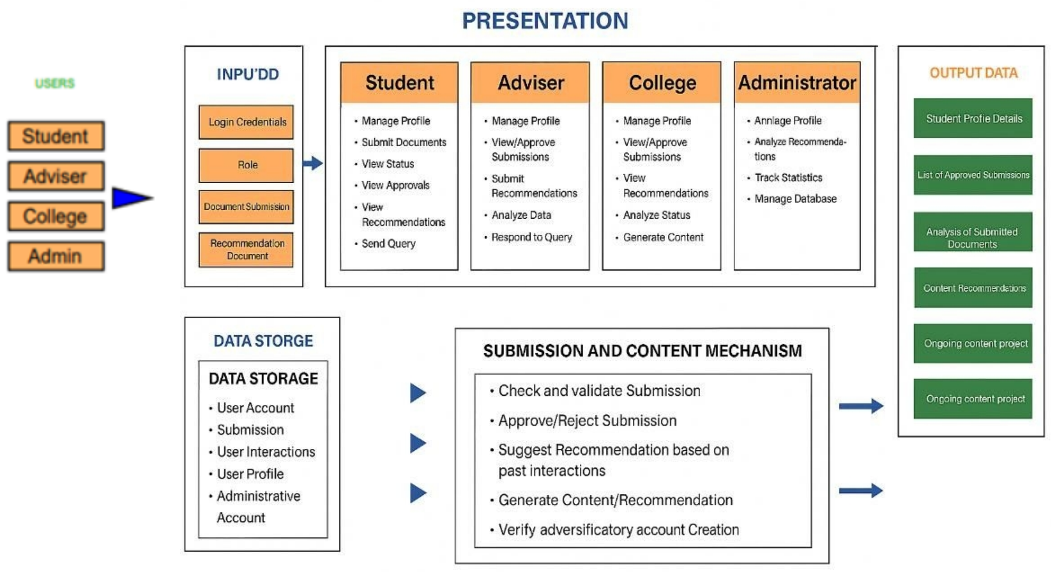
```php
<?php
include 'dbconnection.php';

require '../vendor/autoload.php';

use TeamTNT\TNTSearch\TNTSearch;

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $submission_id = $_POST['submission_id'];
    $research_title = $_POST['research_title'];
    $author = $_POST['author'];
    $co_authors = $_POST['co_authors'];
    $abstract = $_POST['abstract'];
    $keywords = $_POST['keywords'];
    $file_path = '';

    // Start transaction
    $conn->begin_transaction();

    try {
        // Fetch the current data from the archive table
        $sql = "SELECT * FROM archive WHERE id = ?";
        if ($stmt = $conn->prepare(query: $sql)) {
            $stmt->bind_param(types: "i", var: &$submission_id);
            $stmt->execute();
            $result = $stmt->get_result();
            $current_data = $result->fetch_assoc();
            $stmt->close();
        } else {
            throw new Exception(message: 'Failed to prepare statement: ' . $conn->error);
        }

        // Fetch the id and comments from the submission_status table
        $sql_status = "SELECT id, dateofsubmission, comments FROM submission_status WHERE submission_id = ?";
        if ($stmt_status = $conn->prepare(query: $sql_status)) {
            $stmt_status->bind_param(types: "i", var: &$submission_id);
            $stmt_status->execute();
            $result_status = $stmt_status->get_result();
            $status_data = $result_status->fetch_assoc();
            $stmt_status->close();
        } else {
            throw new Exception(message: 'Failed to prepare status statement: ' . $conn->error);
        }

        if (!$status_data) {
            throw new Exception(message: 'Submission ID does not exist in submission_status table');
        }

        // Insert the current data into the submission_history table
        $sql_history = "INSERT INTO submission_history (submission_id, research_title, author, co_authors, abstract, keywords, file_path, UserID, adviser_name, faculty_code, dateofsubmi
        if ($stmt_history = $conn->prepare(query: $sql_history)) {
            $stmt_history->bind_param(
                types: "isssssssssss",
                var: &$status_data['id'], // Use 'id' instead of 'submission_id'
                vars: &$current_data['research_title'],
                $current_data['author'],
                $current_data['co_authors'],
                $current_data['abstract'],
                $current_data['keywords'],
                $current_data['file_path'],
                $current_data['UserID'],
                $current_data['adviser_name'],
                $current_data['faculty_code'],
                $status_data['dateofsubmission'],
                $status_data['comments']
            );
            $stmt_history->execute();
            $stmt_history->close();
        } else {
            throw new Exception(message: 'Failed to prepare history statement: ' . $conn->error);
        }

        // Handle file upload
        if (isset($_FILES['file']) && $_FILES['file']['error'] == UPLOAD_ERR_OK) {
            $target_dir = "../Archive/";
            $original_file_name = pathinfo(path: $_FILES["file"]["name"], flags: PATHINFO_FILENAME);
            $original_file_extension = strtolower(string: pathinfo(path: $_FILES["file"]["name"], flags: PATHINFO_EXTENSION));
            $target_file = $target_dir . basename(path: $_FILES["file"]["name"]);

            // Check if the file is a PDF
            if ($original_file_extension != "pdf") {
                throw new Exception(message: 'Only PDF files are allowed');
            }

            // Increment the file name if it already exists
            $counter = 1;
            while (file_exists(filename: $target_file)) {
                $target_file = $target_dir . $original_file_name . '_' . $counter . '.' . $original_file_extension;
                $counter++;
            }

            // Move the new file to the target directory
            if (move_uploaded_file(from: $_FILES["file"]["tmp_name"], to: $target_file)) {
                $file_path = $target_file;
            } else {
                throw new Exception(message: 'File upload failed');
            }
        } else {
            // If no new file is uploaded, keep the current file path
            $file_path = $current_data['file_path'];
        }

        // Update the archive table with the new details
        $sql = "UPDATE archive SET research_title = ?, author = ?, co_authors = ?, abstract = ?, keywords = ?, file_path = ? WHERE id = ?";
        if ($stmt = $conn->prepare(query: $sql)) {
            $stmt->bind_param(types: "ssssssi", var: &$research_title, vars: &$author, $co_authors, $abstract, $keywords, $file_path, $submission_id);
            if ($stmt->execute()) {
                // Update the submission_status table with the new status
                $status = 'Updated'; // Example status, change as needed
                $sql_status_update = "UPDATE submission_status SET status = ? WHERE submission_id = ?";
                if ($stmt_status_update = $conn->prepare(query: $sql_status_update)) {
                    $stmt_status_update->bind_param(types: "si", var: &$status, vars: &$submission_id);
                    if ($stmt_status_update->execute()) {
                        // TNTSearch: Update the index with the new document
                        $tnt = new TNTSearch();
                        $tnt->loadConfig([
                            'driver'   => 'mysql',
                            'host'     => 'localhost',
                            'database' => 'rmis',
                            'username' => 'root',
                            'password' => '',
                            'storage'  => __DIR__.'/../indexes',
                        ]);
```



```php
        // Update the archive table with the new details
        $sql = "UPDATE archive SET research_title = ?, author = ?, co_authors = ?, abstract = ?, keywords = ?, file_path = ? WHERE id = ?";
        if ($stmt = $conn->prepare(query: $sql)) {
            $stmt->bind_param(types: "ssssssi", var: &$research_title, vars: &$author, $co_authors, $abstract, $keywords, $file_path, $submission_id);
            if ($stmt->execute()) {
                // Update the submission_status table with the new status
                $status = 'Updated'; // Example status, change as needed
                $sql_status_update = "UPDATE submission_status SET status = ? WHERE submission_id = ?";
                if ($stmt_status_update = $conn->prepare(query: $sql_status_update)) {
                    $stmt_status_update->bind_param(types: "si", var: &$status, vars: &$submission_id);
                    if ($stmt_status_update->execute()) {
                        // TNTSearch: Update the index with the new document
                        $tnt = new TNTSearch();
                        $tnt->loadConfig([
                            'driver'   => 'mysql',
                            'host'     => 'localhost',
                            'database' => 'rmis',
                            'username' => 'root',
                            'password' => '',
                            'storage'  => __DIR__.'/../indexes',
                        ]);

                        $tnt->selectIndex("archive.index");
                        $index = $tnt->getIndex();
                        $index->insert([
                            'id' => $submission_id,
                            'research_title' => $research_title,
                            'author' => $author,
                            'abstract' => $abstract,
                            'keywords' => $keywords
                        ]);

                        // Commit transaction
                        $conn->commit();
                        echo json_encode(value: ['status' => 'success', 'message' => 'Submission updated successfully.']);
                    } else {
                        throw new Exception(message: 'Failed to update submission status.');
                    }
                    $stmt_status_update->close();
                } else {
                    throw new Exception(message: 'Failed to prepare status update statement: ' . $conn->error);
                }
            } else {
                throw new Exception(message: 'Failed to update submission.');
            }
            $stmt->close();
        } else {
            throw new Exception(message: 'Failed to prepare statement: ' . $conn->error);
        }
    } catch (Exception $e) {
        // Rollback transaction on error
        $conn->rollback();
        echo json_encode(value: ['status' => 'error', 'message' => $e->getMessage()]);
        exit();
    }
} else {
    echo json_encode(value: ['status' => 'error', 'message' => 'Invalid request method']);
}

$conn->close();
?>
```

System Architecture and Design

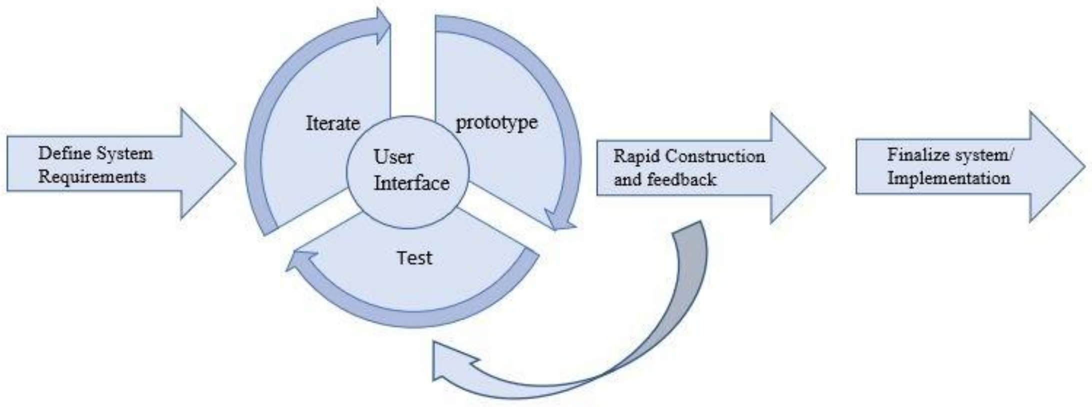
I. Conceptual Framework



Document Management Evaluation and Archiving Information System for WMSU Research Papers shall deal with four distinct groups of users: researchers, advisers, colleges and administrators. All activities between the different sets of users happen in a web browser; thus, an active Internet connection will be required for all operations. The system processes information, and data provided by users through HTTP requests, facilitating updating and interactions. Its architecture ensures the secure transmission and storage of data, and access controls have been ensured to protect information of the user.

It serves as the front-end interface for managing and evaluating detailed research. Through this interface, researchers can submit their papers, while reviewers can review and administrators administrate the entire process. The other important function Handled by this interface is the publication of research. It will process submissions and peer 18 reviews before publishing research papers. This functionality supports the researcher by providing accurate data insight that will enable him/her to make informed decisions, hence improving academic output. It is built on the backbone of a number of modern technologies, all of which provide strong and scalable solutions. The web application provides a fast and friendly user interface made with HTML, CSS, and JavaScript for all users. On the server side, Php will be used, providing efficient handling over all the operations of the server side. MySQL will be employed as the database management system to reliably store and fetch all data. The tech stack makes sure that the application is functional, efficient, secure, and ready for scaling in times of a huge user base.

II. RAD Methodology


Define System Requirements
Iterate
prototype
User Interface
Test
Rapid Construction and feedback
Finalize system/ Implementation

Using Rapid Application Development (RAD) methodology, WMSU's Document Management Evaluation and Archiving Information System was designed and developed for an efficient and user-oriented environment. RAD promotes cooperation and active collaboration with users as the developers continually develop the product following rapid iterations. Thus, RAD is divided into four distinct phases that together yield a working and user- oriented platform. The process started with the system Requirements Definition phase, which involved intensive consultation with WMSU administrators, faculty, and students to identify the core functionalities that should be implemented in the system, including secure submission, archiving, and retrieval of research papers as well as rolespecific access controls for administrators, faculty, and students. Great effort was made to ensure the platform would be user friendly, mobile compatible, and could handle a high data volume.

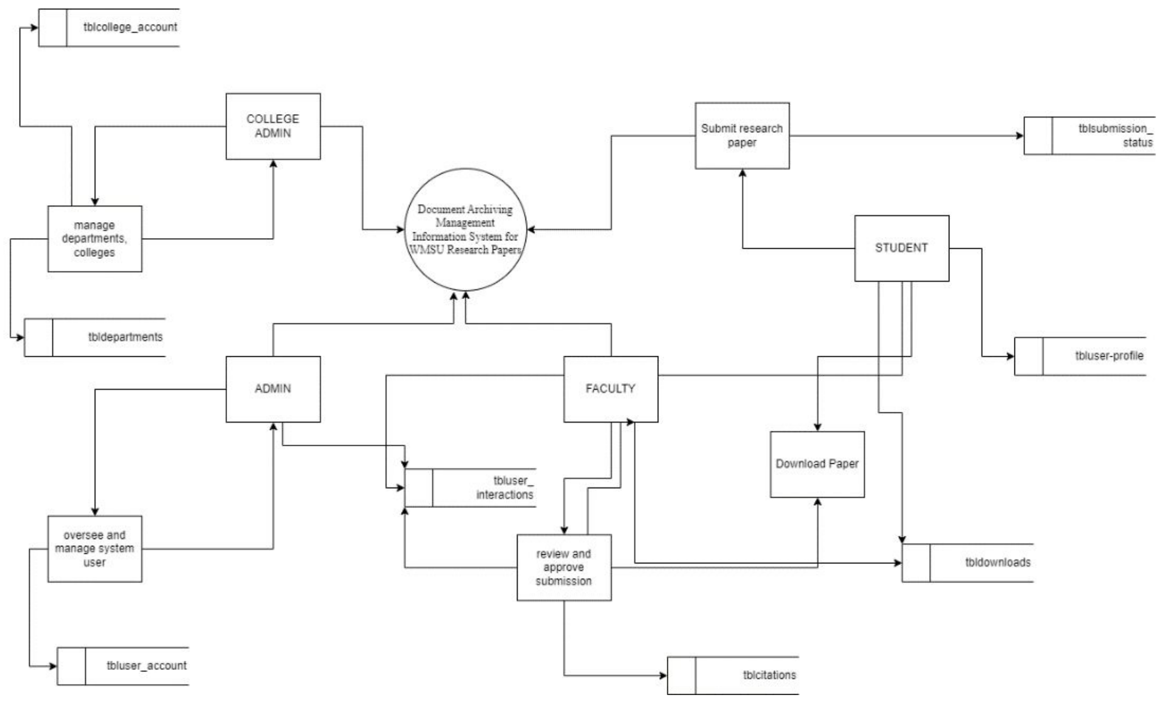
The phase outcome was a clear and well detailed list of system requirements from which further stages were based. In this User Design stage, initial prototypes were established and developed to help visualize both the system's interface as well as its workflows. Some key prototypes developed included the submission portal, document review dashboards, and administrative tools. Prototypes were shared with stakeholders for early feedback to ensure that designs met user expectations. These prototypes were also used for usability, navigation, and workflow efficiency by stakeholders. During this phase, feedback resulted in several iterations of design, where elements such as layout, accessibility, and ease of use were further refined. By the end of this phase, a complete blueprint of the system's design was ready, incorporating the needs both functional and aesthetical of its users.

The Rapid Construction and Feedback phase focused on transforming the prototypes into a working system. Development occurred incrementally as there was an effort to introduce basic functionalities, such as secure submission of documents, mechanisms for feedback, user authentication, and data encryption. The architecture of the system was designed using technologies such as HTML, CSS, JavaScript, PHP, and MySQL. This ensured that the system was responsive, efficient, and scalable. Beta testing was conducted regularly during this phase to allow real users to interact with the system and provide feedback. This loop of feedback was critical in identifying issues, such as minor bugs, performance bottlenecks, and usability challenges, and resolving them.

Continuous testing and refinement during this stage ensured that every iteration of the system was better than the last, leading to a finished and functional product.

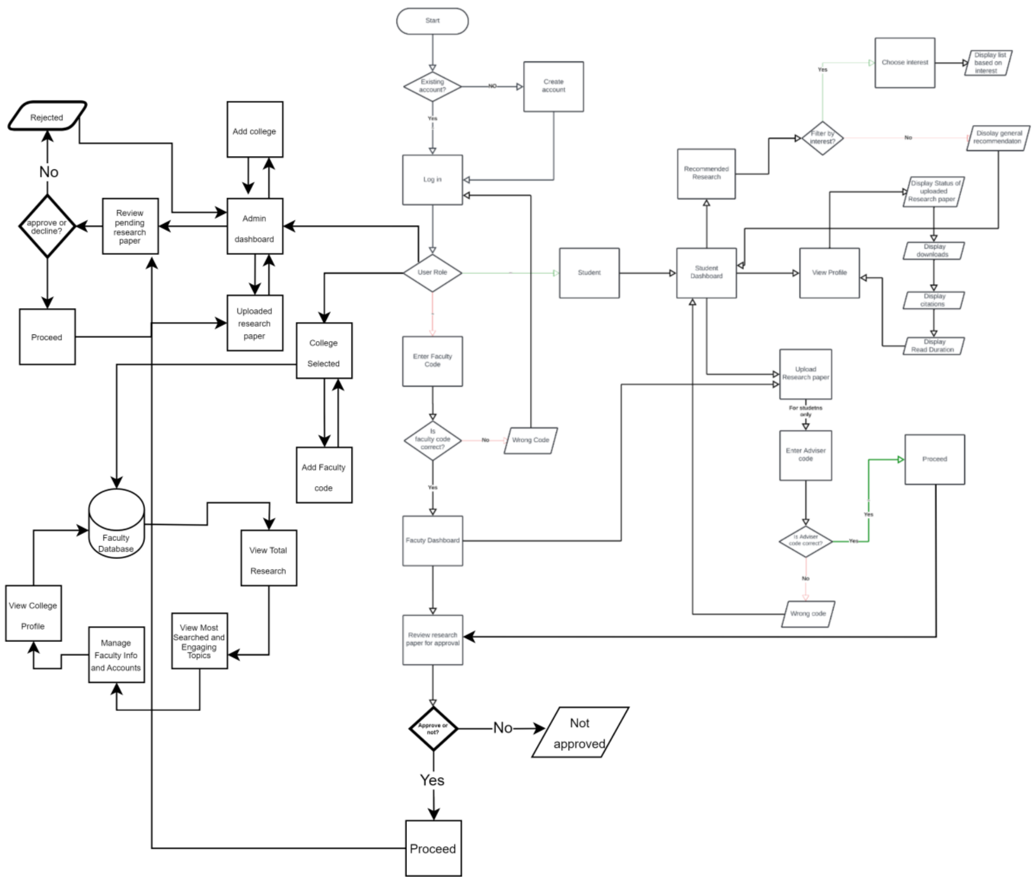
The final phase is implementation. This is when the system is finalized and prepared for deployment. All the feedback from beta testing was incorporated to address any outstanding issues and optimize system performance. Thorough end-to-end testing ensured that the system would be working flawlessly across many different scenarios, from a high user load to complex workflows. It was then deployed to ensure that administrators, faculty members, and other users knew how to navigate and effectively make use of the system with comprehensive training sessions. For post-deployment, the support and monitoring for prompt concerns were established in a way to smooth transition into the new system. Therefore, by following the RAD approach for system development, the developmental process was streamlined while upholding a focus on both user needs and continuous improvements. The collaborative and iterative approach allowed the development team to create a robust, scalable and, most importantly, user-friendly system that would meet the demands of WMSU at each stage, yet flexible for future enhancements.

III. Data Flow Diagram



In the Data Flow Diagram of the Document Archiving Management Information System, the tblcollege_account table is related to the process of the College Admin. The data in the tblcollege_account table pertains to the accounts in the colleges that make up the university; thus the college admin performs the process of updating this data. Since the college admin communicates with the system to keep track of details pertaining to the college account, the system captures this information as it is processed and stored in the database with the table name being tblcollege_account. The tbl downloads table is associated with the Download Paper process, thus linked both to the Faculty and to the Students. This table keeps a record of all research paper downloads from the archive. Information entered each download occasion is stored in the tbldownloads table , so the system knows which papers are being accessed. The tbluser_interactions logs activities of the users within the system, such as submissions, reviews, or downloads. Because Faculty, Students, and Admins each interact with the system differently-be it in submitting research, reviews of papers, or downloading various materials-these interactions are logged and saved in tbluser_interactions . Therefore, how the users engage with the system will be recorded and tracked to monitor the use of the system by the administrators.

IV. Flowchart



All the users go through a Start or Log In/Create Account process. Depending on the choice of account creation and log in, the system shall determine what role the particular user is to assume, that is, Student, Faculty, Admin, or College. At login time a student can upload or submit research paper, which will then be stored in the Research Database. Alternatively, the student will also upload information that is personal and this shall be kept in the Student Profile Database. The student may also change the password, where the modification will be effects in the Account Database. On login, the student shall be required to input an adviser code that will subsequently be validated with the Faculty Database. Students can view research paper recommendations whose data is drawn from the Research Database. Students can set account role and details, saved in the Student Profile Database. Students can also view their profile, drawn from the Student Profile Database, and examine the status of their research, comprising information such as downloads, citations, and recent publications, sourced from the Research Database.

A faculty member, as the students, undergoes a similar process when logging in. A faculty may submit or upload research papers that will be saved in the Research Database; faculty may also upload personal information so that this will be saved in the Faculty Profile Database. The account database gets updated with the new password every time a faculty member changes. Most likely, the most important faculty functions have to do with the reviewing of the research papers of students. The faculty can then approve or reject submissions that will update the research status in the Research Database. The faculty can input faculty code validated against the Faculty Database. Can view the recommended research papers which data is fetched from the Research Database. Faculty can also set account role and details, can view profile which data is fetched from the Faculty Profile Database. Other information available to the instructors are recently accepted articles, which can be obtained from the Research Database. Admins have unique capabilities within the system. After logging in, they can choose a year to filter data from the Usage

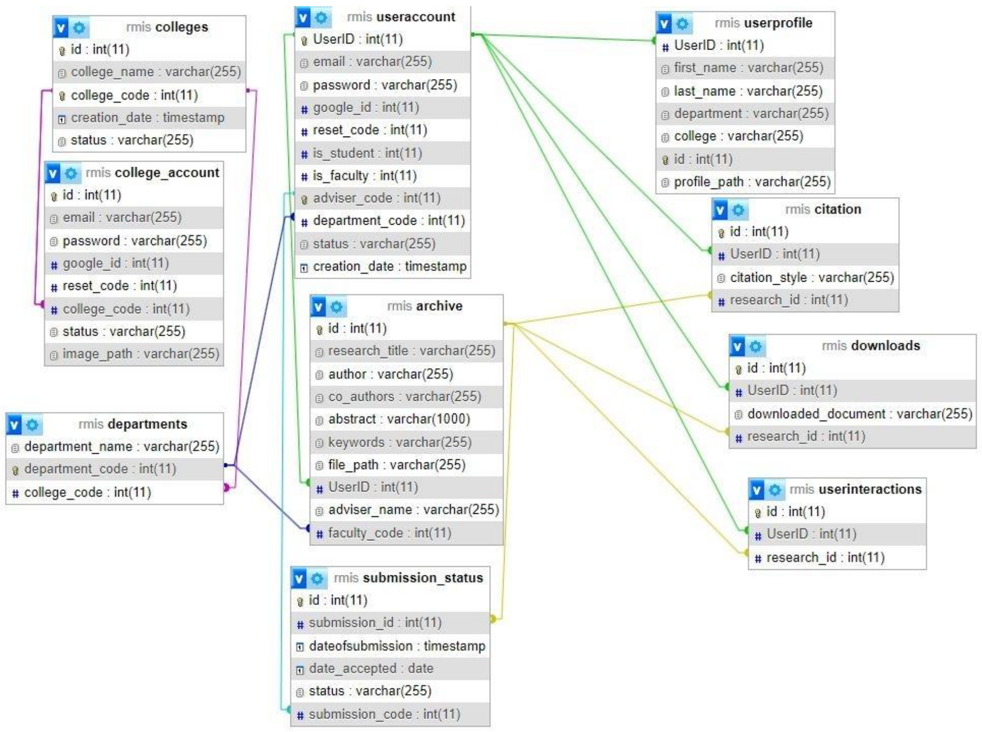
Statistics Database. Admins have the ability to deactivate college accounts, with updates saved in the College Database. Admin can also add college codes to the College Database and view statistics such as the total number of colleges, departments, and research activity, which is sourced from the College Database and Research Database. Admins are also able to view usage metrics, such as the most searched topics (from Search Logs), the college with the most views, citations, and downloads (from the Usage Statistics Database), and trending topics (from Search Logs).

Admins can manage college codes and accounts, updating, adding, or deleting

records in the College Database. For college administrators, after logging in, they are able to deactivate faculty accounts, with updates saved in the Faculty Database, and add faculty codes, which are inserted into the Faculty Database. The college admins can view total research statistics, including submissions, approvals, and pending research, with data pulled from the Research Database. College administrators also have access to view popular topics, based on search logs, and can manage faculty information and accounts, updating the Faculty Database. And can view their college profile, with data sourced from the College Profile Database. Some databases manage the system operations. For example, the Research Database manages research paper, submission status, citations, downloads, and trending topics. The Student Profile Database corresponds to handling student details, and Faculty Profile Database manages their profile too. Finally, Account Database manages log in information and set up for every account. The College Database stores college-specific data, and the Usage Statistics Database captures statistics on views, downloads, and engagement. Lastly, Search Logs track searches and trending topics.

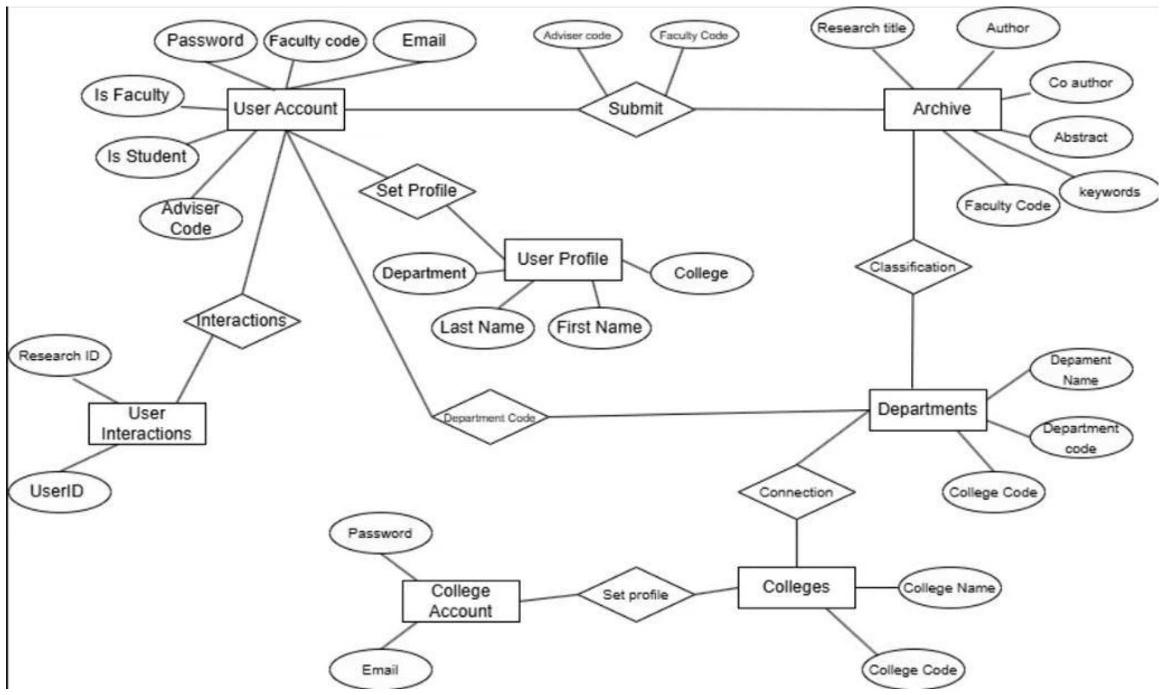
DATABASE SCHEMA

I. Database Design



All tables are interconnected to each other to ensure that all data is related to each other as user account table is connected to user profile as dedicated table to set or update a profile information such as first name, last name, department, college. Also user account table is connected to archive table, archive table is the main table to hold the documents and archived it. It is connected to user account to ensure the details and meta data of the uploader of the document is connected as clear to be fetched also user account table is connected to other table such as citation, downloads and user interactions this is crucial to ensure that all the data activity of the user is properly stored in the database to use it in data analytics for the system to analyze and provide an overview.

II. Entity Relationship Diagram



In the entity relationship diagram the user account will submit a document and stored it in archive entity showing the relation that it include the faculty code of the user account to be stored in the faculty code of archive entity which then connected to the departments table department code, in this way the archive can have a classification of which department it is came from, and then the departments entity have a connection to colleges entity via college code attribute same name for both table. And college's entity is connected to college account entity college's entity serves as profile table for college account where profile information of college is saved in the attribute's college name.