ARMAlytics

TeknoSpace

Software Design Document

Version <02.00>

Revision History

Date	Version	Description	Author
<07/02/2024>	<01.00>	SDD 1.0	Group - ARMAlytics
<07/23/2024>	<02.00>	SDD 2.0	Group - ARMAlytics

Table of Contents

1. INTRODUCTION 4

   1.1 Purpose 4

   1.2 Scope 4

   1.3 Overview 4

   1.4 Reference Material 4

   1.5 Definitions and Acronyms 5

2. SYSTEM OVERVIEW 5

3. SYSTEM ARCHITECTURE 8

   3.1 Architectural Design 8

   3.2 Decomposition Description 12

   3.3 Design Rationale 17

4. DATA DESIGN 17

   4.1 Data Description 17

   4.2 Data Dictionary 20

5. COMPONENT DESIGN 23

6. HUMAN INTERFACE DESIGN 27

   6.1 Overview of User Interface 27

   6.2 Screen Images 29

   6.3 Screen Objects and Actions 35

7. REQUIREMENTS MATRIX 38

8. APPENDICES 39

Software Design Document

1. Introduction

1.1 Purpose

The Software Design Document (SDD) outlines the architecture and system design of TeknoSpace, a platform created for educational purposes. It specifies how TeknoSpace works, its features, interfaces, and general structure. This document is critical for project managers, software developers, and other stakeholders who are involved in the system's implementation and comprehension.

1.2 Scope

TeknoSpace is a web-based platform designed specifically for educational purposes at the Cebu Institute of Technology - University (CIT-U). The scope of TeknoSpace encompasses providing students with a centralized hub to access and view posts, announcements, and updates from CIT-U's various academic and administrative departments. The platform ensures that students receive daily updates with the latest information coming from CIT-U. The primary goal is to enhance accessibility, allowing students to quickly and easily access relevant information. By facilitating effective communication between CIT-U administration, faculty, and students, TeknoSpace aims to streamline the delivery of information, ensuring students are consistently informed of academic activities and administrative changes. The platform features include a centralized school updates portal, an interactive lost and found section, and robust Student Engagement tools. These features collectively contribute to making TeknoSpace an essential tool for improving communication and engagement within the CIT-U community.

1.3 Overview

This Software Design Document (SDD) for TeknoSpace serves to comprehensively detail the system's design and architecture. It acts as a guide for developers, outlining the technical specifications and design decisions essential for implementing the TeknoSpace platform effectively. This Software Design Document (SDD) for TeknoSpace begins with an introduction, system overview, system architecture, data design , component design and human interface design(UI).

1.4 Reference Material

Software Design Document. (n.d.). Bellevue College. Retrieved June 30, 2024, from https://www.bellevuecollege.edu/wp-content/uploads/sites/135/2019/04/SDD_RoadTrip.pdf

Burke, R., Vahedian, F., Mobasher, B. (2014). Hybrid Recommendation in Heterogeneous Networks. In: Dimitrova, V., Kuflik, T., Chin, D., Ricci, F., Dolog, P., Houben, GJ. (eds) User Modeling, Adaptation, and Personalization. UMAP 2014. Lecture Notes in Computer Science, vol 8538. Springer, Cham. https://doi.org/10.1007/978-3-319-08786-3_5

1.5 Definitions and Acronyms

Acronym	Definition
SDD	Software Design Document
CIT-U	Cebu Institute of Technology- University
UI	User Interface

2. System overview

The project is a website designed to help students at CIT-U stay updated with school announcements, maintenance and lost & found items inside the school. The system acts as a social platform similar to Facebook but is specifically tailored for Teknoys. It provides functionalities for user registration, login, posting announcements (Faculty and Admin), commenting on announcements, and liking announcements.

1. User Registration and Login:
    - Students & Faculty can create accounts by providing their full name, university email, password, course, id number, user type (Faculty or Student- Verified through their ID picture).
    - Registered users can log in to the system using their email or school id and password.
2. Posting Announcements:
    - Faculty and Admin can post announcements that include a caption and optional attachments.
    - Announcements and Maintenance are time stamped and have indication whether it is important or not.
    - Lost items posted on the Lost&Found tab have indication on its post header whether it is still lost or has already found.

3. Commenting on Announcements:

    - Users can comment on any posted announcement.
    - Comments include the content and are time stamped, with each comment linked to a specific announcement and user.

4. Liking Announcements:

    - Users can like any announcement.
    - Likes are recorded and associated with both the announcement and the user who liked it.

The website is intended to serve the university's student community, providing a stationary, centralized platform for them to have communication and updates. It aims to enhance student engagement and ensure that important information is easily accessible. The system is built using a combination of MySQL for the database, PHP for backend functionality and VSCode for the frontend using HTML, CSS and JavaScript.

Design

1. Frontend (VSCode)

    - HTML/CSS: Structures and styles the user interface.
    - JavaScript: Provides interactivity and handles client-side operations like validation of requests.
        - AJAX: Facilitates asynchronous communication between the client and server without reloading the entire page. Allows for dynamic updates to the web page based on server responses.
        - JSON: Used for formatting data to be sent to and received from the server. It is a lightweight data-interchange format that JavaScript can easily parse and generate.

2. Backend

    - PHP: Handles server-side operations including data processing, logics, and interactions with the database.

3. Database

    - MySQL: Stores user profiles, announcements, comments and likes. Ensuring data integrity and efficient querying.

System Components and Their Functions:

1. User Management:
   - Registration: Validates and stores new user data
   - Login: Authenticates users and manages sessions.
2. Announcement Management
   - Post Announcement: Handles the creation of new announcements
   - View Announcements: Retrieves and displays the information of specific announcements
3. Interaction Management
   - Post Comment: Manages the addition of comments to announcements
   - Like Announcements: Tracks likes on announcements
4. Profile Management
   - User Information: Displays the correct information of the logged-in user.
   - Profile Customization: Users can change their profile photo and cover photo.
   - Post Management: Users can edit and delete their own posts. They can also change information, such as marking the post as IMPORTANT or FOUND.

The idea for the project originated from the need for a stationary, not cluttered, efficient, and organized way for university students to stay informed about school-related updates. Even modern social media posts would be hidden from social media like Facebook as the algorithm of the website would be from the recent likes of students, thus being uninformed students. This system aims to remove that issue by providing a dedicated platform for announcements and student interactions.

By elevating the familiar social media-like interface, the system encourages active participation and ensures students do not miss important updates, maintenance, and lost & found items around the campus. The use of web technologies like HTML, CSS, JavaScript, PHP, and MySQL allows for a robust and scalable solution that can be easily adapted to different universities or educational institutions.

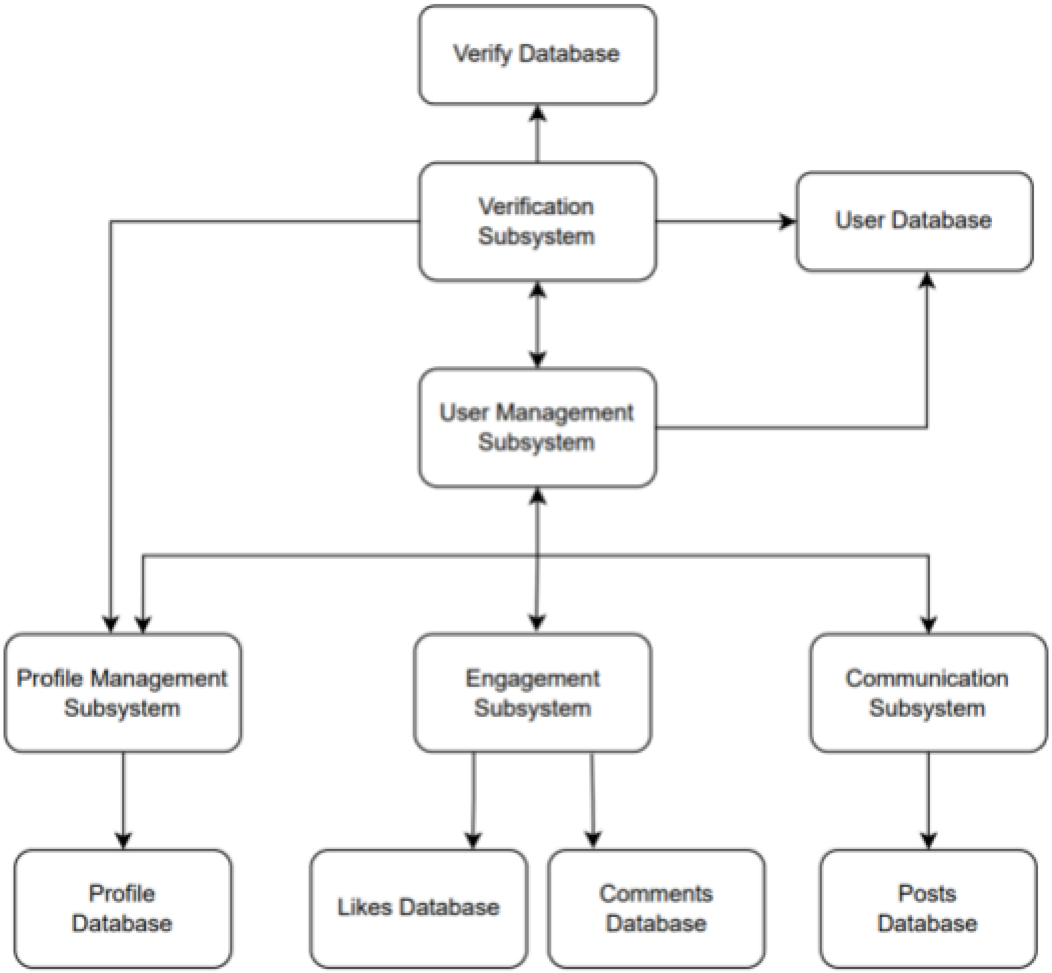
3. System Architecture

3.1 Architectural Design

The architecture and system design of TeknoSpace encompass its features, interfaces, and overall structure, ensuring effective implementation and operation. This design aims to facilitate efficient user management, communication and engagement within the system.

High-Level Subsystem

This high-level overview outlines the primary subsystems and data repositories that constitute the core structure of the platform. Each subsystem has distinct responsibilities and collaborates with others to deliver a cohesive and integrated user experience.



High-Level Subsystems

1. **User Management Subsystem**
   - **Responsibilities:**
     - Handle user authentication (login/logout)
     - Manage user roles (students, administrators, faculty)
     - Profile management
     - Verify user credentials during the signup process
   - **Collaboration:**
     - Works with the Verification Subsystem to authenticate and verify users during signup.
     - Works with the Communication Subsystem to authenticate users for posting updates and interacting with announcements.
     - Provides user information to the Engagement Subsystem for personalized engagement experiences.
     - Collaborates with the Profile Management Subsystem to ensure accurate profile management.
2. **Verification Subsystem**
   - Responsibilities:
     - Store signup data temporarily in the Verify Database to ensure it's from the university.
     - Transfer verified user data to the Users Database.
   - Collaboration:
     - Interacts with the User Management Subsystem to handle the verification process.
     - Upon verification, collaborates with the Profile Management Subsystem to create user profiles.

3. **Profile Management Subsystem**
   - Responsibilities:
     - Manage user profiles, including profile pictures and cover photos.
     - Store verified user profiles in the Profile Database.
   - Collaboration:
     - Receives verified user data from the Verification Subsystem.
     - Uses authentication data from the User Management Subsystem for profile management.
     - Provides user profile information to the Engagement and Communication Subsystems.
4. **Communication Subsystem**
   - **Responsibilities:**
     - Centralized platform for posting acknowledgments and announcements.
     - Enable faculty and admin to edit or delete their own posts, and to mark them as important or normal posts for the maintenance and school updates sections.
     - Allows admin and faculty users to post about lost items.
     - Allows the admin or faculty to mark the post as found in the lost and found section.
   - **Collaboration:**
     - Interacts with the Communication Subsystem to post updates about important announcements and found items.
     - Uses authentication services from the Profile Management Subsystem to ensure only authenticated users can post or respond.
     - Notifies users via the Communication Subsystem about updates on lost items by posting it as found.

5. **Engagement Subsystem**
   - **Responsibilities:**
     - Facilitate discussions and feedback on posts
     - Promote student interaction and community engagement
     - Enable all users to respond with information about found items posting
     - Allow students to edit or delete their own comments on their profile page.
   - **Collaboration:**
     - Uses user profiles from the User Management Subsystem to track and manage interactions.
     - Receives updates from the Communication Subsystem to provide context for discussions.
     - Uses authentication services from the Profile Management Subsystem to ensure only authenticated users can post or respond.

Data Repositories

1. **User Database**
   - Stores user credentials, profiles, and roles
   - Used by the User Management Subsystem
2. **Verify Database**
   - Stores signup data temporarily to ensure it's from the university.
   - Used by the Verification Subsystem.
3. **Profile Database**
   - Stores all user profiles, including profile pictures, cover photos, and userId.
   - Used by the Profile Management Subsystem.

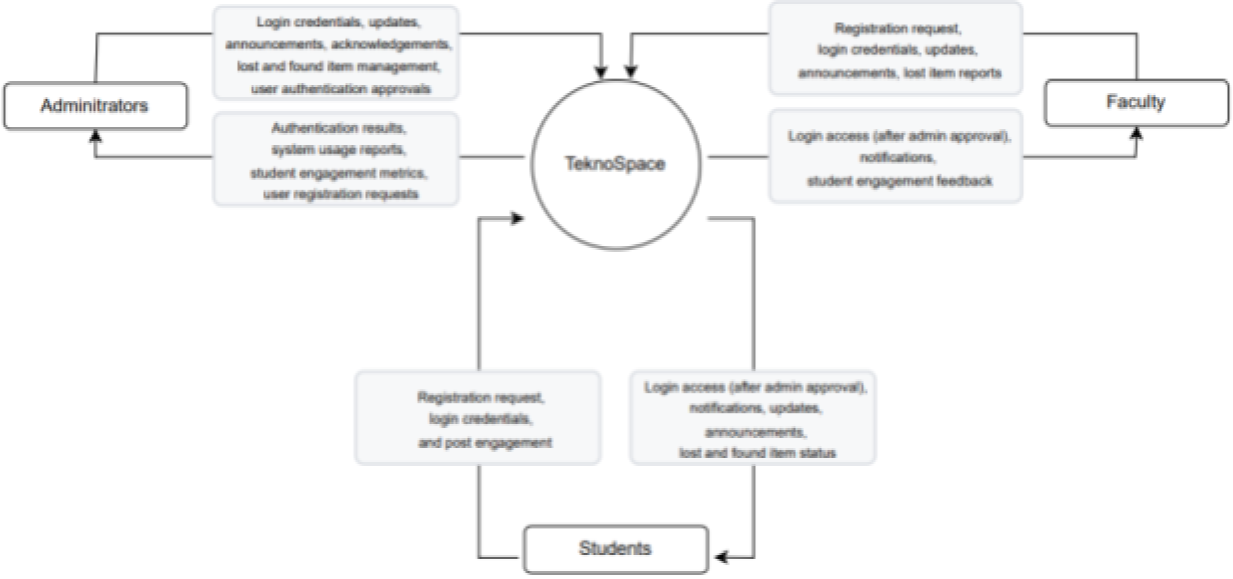
4. **Posts Database**
   - ○ Stores announcements, acknowledgments, lost and found, and other communications.
   - ○ Stores records of lost and found items.
   - ○ Used by the Engagement, Communication, and Lost and Found Subsystems
5. **Comments Database**
   - ○ Stores comments on posts
   - ○ Used by the Engagement Subsystem
6. **Like Database**
   - ○ Stores the number of likes.
   - ○ Used by the Engagement Subsystem
7. **Profile Database**
   - ○ Stores the individual's profile.

3.2 Decomposition Description

To provide a decomposition of the subsystems in the architectural design of the university announcements web application, a functional description using data flow diagrams (DFD) and structural decomposition diagrams is used. This approach helps in visualizing how data flows through the system and how the system is structurally decomposed into various subsystems.

Top-Level Data Flow Diagram (DFD)

The top-level DFD represents the entire system as a single process and shows the interactions between external entities and the system.


Login credentials, updates,
announcements, acknowledgements,
lost and found item management,
user authentication approvals
Registration request,
login credentials, updates,
announcements, lost item reports
Adminitrators
Faculty
Authentication results,
system usage reports,
student engagement metrics,
user registration requests
TeknoSpace
Login access (after admin approval),
notifications,
student engagement feedback
Registration request,
login credentials,
and post engagement
Login access (after admin approval),
notifications, updates,
announcements,
lost and found item status
Students

Students

- To System: Registration request, login credentials, lost item reports, found item information, discussion posts, feedback
- From System: Login access (after admin approval) updates, announcements, lost and found item status

Faculty

- To System: Registration request, login credentials, updates, announcements, lost item reports
- From System: Login access (after admin approval) student engagement feedback

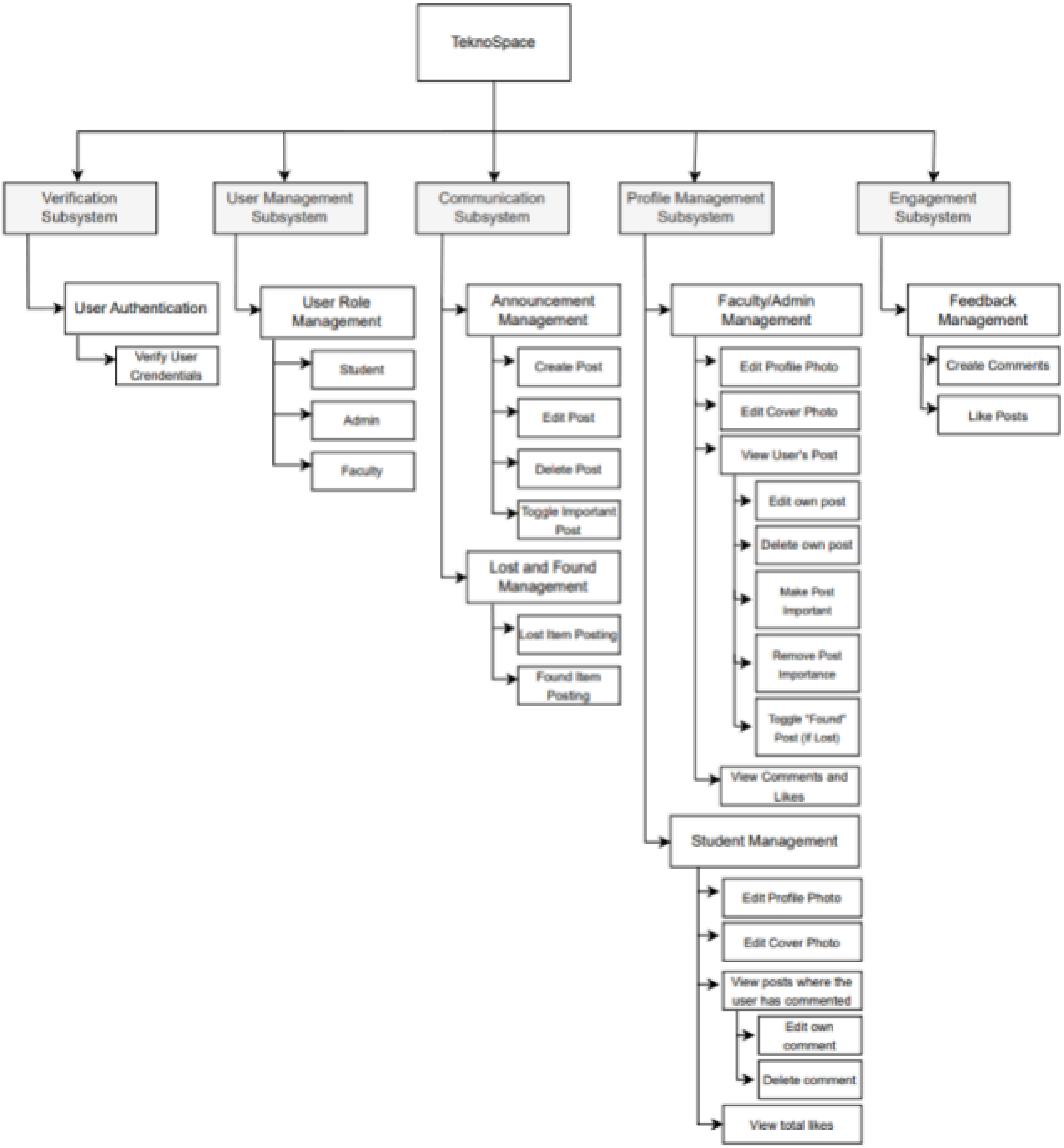
Administrators

- To System: Login credentials, updates, announcements, acknowledgements, lost and found item management, user authentication approvals
- From System: Authentication results, system usage reports, student engagement metrics, user registration requests

This functional description and decomposition of the TeknoSpace system into subsystems highlight the data flows, structural organization, and responsibilities of each subsystem. This structure ensures that each part of the system is modular, maintainable, and capable of efficient interaction with other parts to provide a seamless user experience for university announcements and engagements.

Structural Decomposition Diagrams

The structural decomposition diagram provides a high-level overview of the TeknoSpace system, illustrating the interactions and responsibilities of its various subsystems. The diagram helps visualize how the system components work together to deliver a seamless user experience.



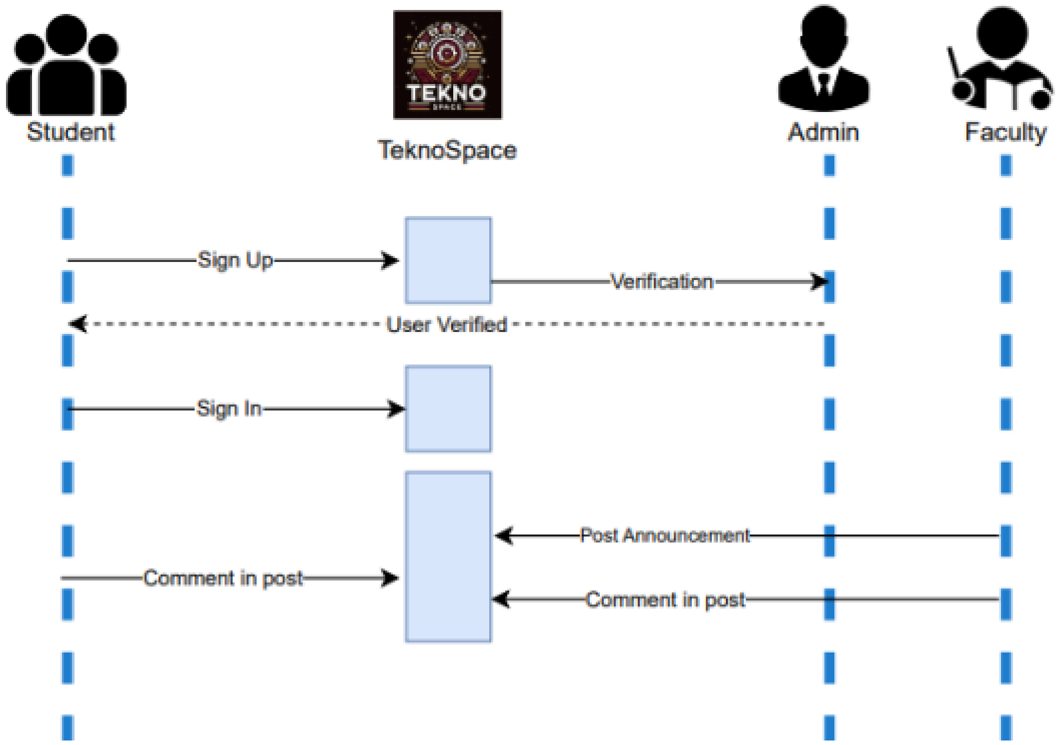
Subsystem Model

TeknoSpace is an integrated digital ecosystem for universities, facilitating communication, user management, and engagement among students, faculty, and administrators. The platform is built on five main subsystems:

1. Verification Subsystem: Ensures secure user registration and authentication.
2. User Management Subsystem: Handles user accounts, roles, and access control.
3. Communication Subsystem: Enables announcements, posts, and a lost-and-found feature.
4. Profile Management Subsystem: Allows users to manage their profiles and personal information.
5. Engagement Subsystem: Facilitates interaction through comments, likes, and feedback.

These subsystems work together to create a cohesive platform that streamlines administrative tasks, enhances communication, and fosters a sense of community within the university. TeknoSpace provides features such as user authentication, role-based access, announcement management, profile customization, and interactive engagement tools.

Sequence Diagram



The sequence diagram illustrates the core functionalities of TeknoSpace, a university-oriented social platform designed to facilitate communication between students, faculty, and administrators. The system implements a structured approach to user engagement, beginning with a student sign-up process that requires administrative verification. Once verified, students gain access to the platform, allowing them to sign in and participate in discussions. TeknoSpace distinguishes itself from conventional social media platforms by restricting students' ability to create posts; instead, they are limited to commenting on announcements made by faculty members. This deliberate constraint fosters a more controlled and academically focused environment. The diagram also depicts the ability of administrators to engage in discussions by commenting on posts, thus promoting a multi-tiered interaction model within the university community. By centralizing content creation with faculty and enabling student participation through comments, TeknoSpace maintains a curated flow of information while still allowing for student feedback and engagement.

3.3 Design Rationale

The chosen architecture emphasizes modularity and scalability by dividing the system into distinct subsystems (User Management, Communication, Lost and Found and Engagement), each with specific responsibilities. This modular approach allows for independent scaling, maintainability, and parallel development. Centralized user management ensures consistent authentication and authorization across the platform, reducing data redundancy and simplifying future enhancements. Separating the Communication and Lost and Found subsystems allows for focused development and flexibility for future expansions, while a dedicated Additionally, Engagement Subsystem emphasizes the importance of user interaction and community-building features. Using separate databases for different data types optimizes storage, enhances security, and allows independent scaling. While this modular design introduces complexity, it provides long-term flexibility, maintainability, and clear separation of concerns, making it a robust choice for a comprehensive university communication platform.

4. Data Design

4.1 Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

Information Domain Transformation into Data Structures

1. User’s Profiles

- Attributes: Id, userType, firstName, middleName, lastName, idNumber, course, email, password

  Each user profile can be represented as an object or a record in the database.

2. Posts:

- Attributes: Id, username, content(title and description), audience, profile_image, image_path, created_at

Announcements or posts are stored at a separate database with each student able to see these posts but strictly belonging to the faculty.

3. Comments:

- Attributes: commentID, postID, postType, userID, content, profile_image, image_path, ,created_at

Comments are linked to posts and the faculty(userType) profile.

4. Likes:

- Attributes: likesID, postID, userID, liked_at, totalLikes

Likes are linked to both posts and user profiles.

Storage, Processing, and Organization of Major Data Entities

1. User Photos
    - User photos are stored in the database's table called profile. This table holds the cover photos and profile photos associated with each user.
    - Processing: When a user uploads or updates their cover photo or profile photo, the images are saved in the profile table. Each photo is linked to the user's unique user ID. The photos are retrieved when displaying user profiles, allowing for the appropriate images to be shown with the user's information.
    - Organization: Each record in the profile table represents a single user's photo
2. User Profiles
    - Storage- Profiles are stored in the database's table called users
    - Processing: When a new user sign up to the website, their information is validated and then inserted into the users table. The user's information is retrieved when logging in, viewing profiles, and commenting.
    - Organization: Each record in the users table represents a single user.
3. Pending Verification Users
    - Storage: The verify table in the database stores information about users who have signed up but are awaiting admin verification. This table tracks users who need confirmation of their enrollment or affiliation with CIT-U

before they are fully registered.
- Processing: When a user signs up, their information is initially stored in the verify table. The admin reviews and verifies whether the user is officially enrolled or affiliated with CIT-U. Upon successful verification, the user's data is transferred from the verify table to the users table, where all verified users are stored.
- Organization: Each record in the verify table represents a single user awaiting verification.

4. Posts:

   - Storage: Posts are stored in the database’s table called posts

   - Processing: When a faculty member post an announcement, the data is inserted into the posts table. Announcements are retrieved when displaying the news feed or a specific announcement (postType).

   - Organization: Each record in the posts table represents a single announcement

5. Comments:

   - Storage: Comments are stored in the database’s table called comments

   - Processing: When a user comments on a post, the data is inserted into the comments table. Comments are retrieved when displaying an announcement with its comments.

   - Organization: Each record in the comments table represents a single comment, linked to a post and a user.

6. Likes:

   - Storage: Likes are stored in the database’s table called likes.

   - Processing: When a user likes an announcement, the data is inserted into the likes table. Likes are counted and displayed for each announcement.

   - Organization: Each record in the likes table represents a single like, linked to an announcement and a user.

Databases and Data Storage Items

Relational Database Management System (RDBMS): MySQL

Tables:

- (comments): Stores comments on announcements

- (likes): Stores likes on announcements

- (posts): Stores posts and announcements
- (profile): Stores user's profile photo and cover photo
- (users): Stores verified users
- (verify): Stores unverified users

4.2 Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

System Entities (Major Data) - Alphabetically Listed

1. comments
    - Type: Table
    - Description: Stores comments on announcements.
    - Attributes:
        - Id (INT, PRIMARY KEY, AUTO_INCREMENT): Unique identifier for each comment.
        - postId (INT, FOREIGN KEY): Identifier for the announcement being commented on.
        - userId (INT, FOREIGN KEY): Identifier for the user who posted the comment.
        - comment (VARCHAR(200)): Content of the comment.
        - created_at (TIMESTAMP): Date and Time the comment was posted.
        - updated_comment_at (TIMESTAMP): Date and Time the comment was updated
2. likes
    - Type: Table
    - Description: Stores likes on announcements.

- Attributes:
    - id (INT, PRIMARY KEY, AUTO_INCREMENT): Unique identifier for each like.
    - postID (INT, FOREIGN KEY): Identifier for the announcement being liked.
    - userID (INT, FOREIGN KEY): Identifier for the user who liked the announcement.

3. posts
    - Type: Table
    - Description: Stores announcements posted by users.
    - Attributes:
        - id (INT, PRIMARY KEY, AUTO_INCREMENT): Unique identifier for each announcement.
        - userId (INT, FOREIGN KEY): Identifier for the user who posted the announcement.
        - username (VARCHAR(255)): Username of the user posting..
        - content (TEXT): Content of the post.
        - posttype (VARCHAR(250)): Indicates the post type.
        - profile_image (VARCHAR(255)): File path or URL to the picture of the user posting.
        - image_path (VARCHAR(255)): File path or URL to the post's picture
        - is_important (tinyint(1)): Indicator for important posts.
        - updated_important_at(TIMESTAMP): Date and Time the important posts is updated.
        - status (VARCHAR(5)): Indicates the status of Lost and Found Items
        - updated_status_at (TIMESTAMP): Date and Time of the status of a post
        - created_at (TIMESTAMP): Date and Time the post was created.

4. profile
    - Type: Table
    - Description: Stores profile photos and cover photos.
    - Attributes:
        - userId (INT, PRIMARY KEY, AUTO_INCREMENT): Unique identifier for each profile.
        - profile_pic (VARCHAR(255)): Image path of the profile photo.
        - cover_photo (VARCHAR(255)): Image path of the cover photo.
5. users
    - Type: Table
    - Description: Stores user profiles.
    - Attributes:
        - Id (INT, PRIMARY KEY, AUTO_INCREMENT): Unique identifier for each user.
        - userType (VARCHAR(100)): Indicates the role or type of user
        - firstName, middleName, lastName (VARCHAR(100)): Name of the user.
        - idNumber (VARCHAR(100)): Indicates the university ID number of the student
        - course (VARCHAR(100)): Indicates the course of the user
        - email (VARCHAR(100), UNIQUE): Email address of the user.
        - password (VARCHAR(255)): Hashed password for the user.
        - reset_token_hash(VARCHAR(64)): Hashed token for resetting user password
        - reset_token_expires_at (DATETIME): Expiration Date and Time of hash token
6. verify
    - Type: Table
    - Description: Stores user profiles.
    - Attributes:

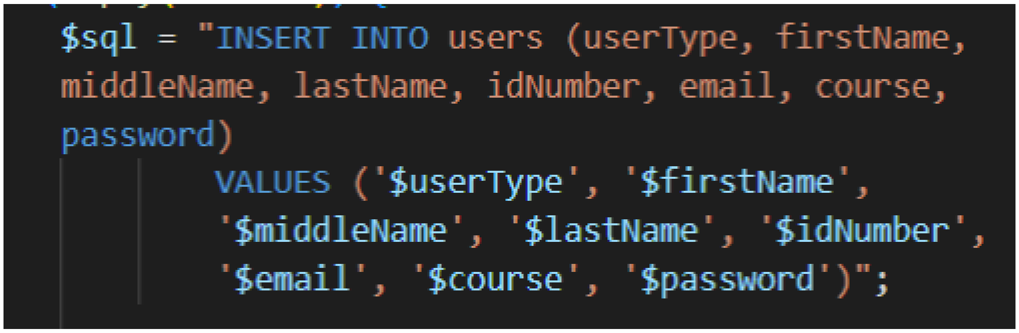
- Id (INT, PRIMARY KEY, AUTO_INCREMENT): Unique identifier for each user.
- userType (VARCHAR(100)): Indicates the role or type of user
- firstName, middleName, lastName (VARCHAR(100)): Name of the user.
- idNumber (VARCHAR(100)): Indicates the university ID number of the student
- course (VARCHAR(100)): Indicates the course of the user
- email (VARCHAR(100), UNIQUE): Email address of the user.
- password (VARCHAR(255)): Hashed password for the user.
- reset_token_hash(VARCHAR(64)): Hashed token for resetting user password
- reset_token_expires_at (DATETIME): Expiration Date and Time of hash token

5. Component Design

User Management Component

1. **Function: Register User**

    - Pseudocode:



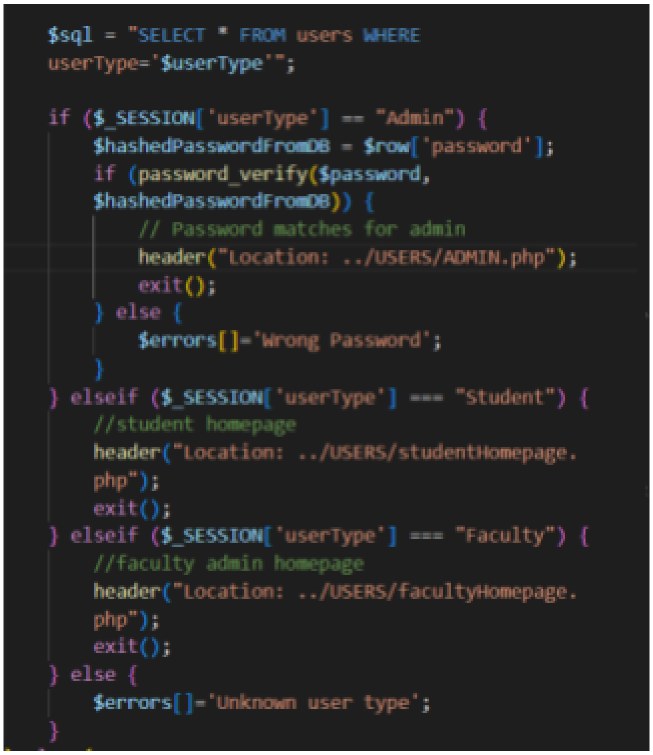
```
$sql = "INSERT INTO users (userType, firstName,
middleName, lastName, idNumber, email, course,
password)
        VALUES ('$userType', '$firstName',
        '$middleName', '$lastName', '$idNumber',
        '$email', '$course', '$password')";
```

- Local Data:
    - userType:VARCHAR,
    - firstName:VARCHAR,
    - middleName:VARCHAR,

- lastName:VARCHAR,
- idNumber:VARCHAR,
- email:VARCHAR,
- course:VARCHAR,
- password:VARCHAR

2. Function: Log In User

- Pseudocode:



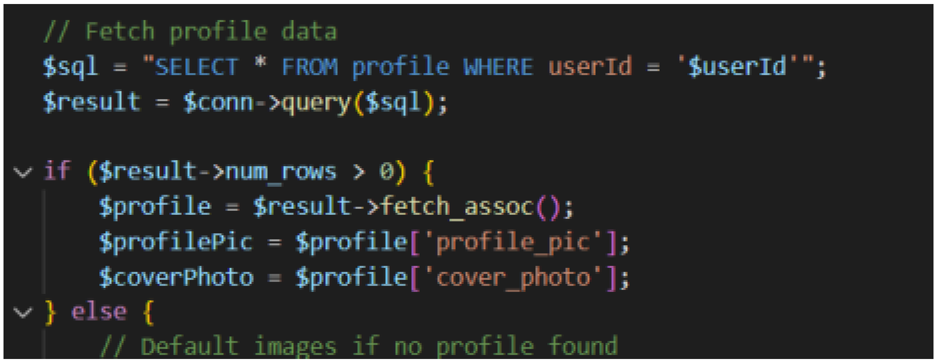
```
$sql = "SELECT * FROM users WHERE
userType='$userType'";

if ($_SESSION['userType'] == "Admin") {
    $hashedPasswordFromDB = $row['password'];
    if (password_verify($password,
    $hashedPasswordFromDB)) {
        // Password matches for admin
        header("Location: ../USERS/ADMIN.php");
        exit();
    } else {
        $errors[]='Wrong Password';
    }
} elseif ($_SESSION['userType'] === "Student") {
    //student homepage
    header("Location: ../USERS/studentHomepage.
    php");
    exit();
} elseif ($_SESSION['userType'] === "Faculty") {
    //faculty admin homepage
    header("Location: ../USERS/facultyHomepage.
    php");
    exit();
} else {
    $errors[]='Unknown user type';
}
```

- Local Data:
    - userType:VARCHAR

3. Function: Update Profile

- Pseudocode:



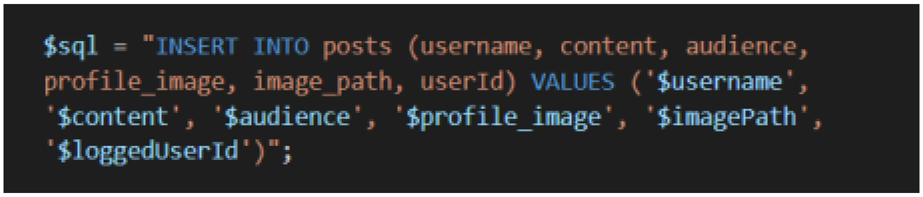
```
// Fetch profile data
$sql = "SELECT * FROM profile WHERE userId = '$userId'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $profile = $result->fetch_assoc();
    $profilePic = $profile['profile_pic'];
    $coverPhoto = $profile['cover_photo'];
} else {
    // Default images if no profile found
```

- Local Data: userId:INT

4. **Function: Create Post**

- Pseudocode:

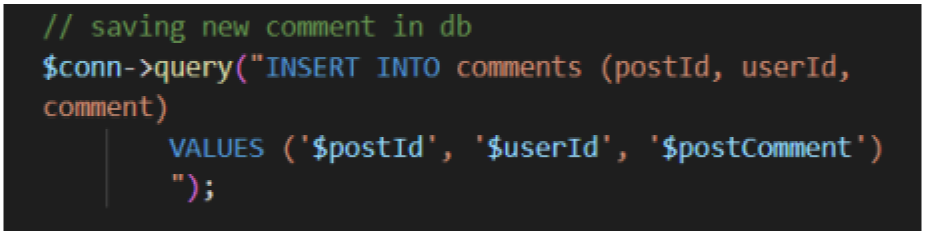


```
$sql = "INSERT INTO posts (username, content, audience,
profile_image, image_path, userId) VALUES ('$username',
'$content', '$audience', '$profile_image', '$imagePath',
'$loggedUserId')";
```

- Local Data:
  - username:VARCHAR
  - content:VARCHAR
  - audience:VARCHAR
  - profile_image:VARCHAR
  - image_path:VARCHAR
  - userId:INT

5. **Function: Add Comment**

- Pseudocode:

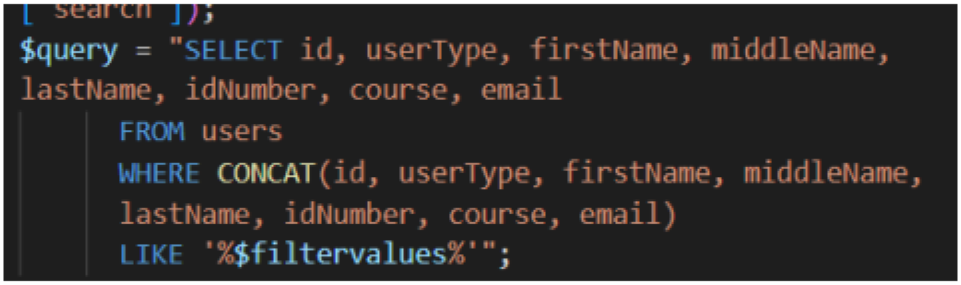


```
// saving new comment in db
$conn->query("INSERT INTO comments (postId, userId,
comment)
         VALUES ('$postId', '$userId', '$postComment')
         ");
```

- Local Data:
  - postId:INT
  - userId:INT
  - comment:VARCHAR

6. **Function: Search All Accounts**

- Pseudocode:



```
['search']);
$query = "SELECT id, userType, firstName, middleName,
lastName, idNumber, course, email
        FROM users
        WHERE CONCAT(id, userType, firstName, middleName,
        lastName, idNumber, course, email)
        LIKE '%$filtervalues%'";
```

- Local Data:
    - id:INT
    - userType:VARCHAR
    - firstName:VARCHAR
    - middleName:VARCHAR
    - lastName:VARCHAR
    - course:VARCHAR
    - email:VARCHAR

6. Human Interface Design

6.1 Overview of User Interface

User Registration and Login

1. **Registration:**
   - **Functionality:** Users can create an account by providing a username, email, and password.
   - **Usage:** Users visit the registration page, fill in the required information, and submit the form.
   - **Feedback:** If the registration is successful, the user is redirected to the login page with a success message. If there are errors (e.g., missing information, username/email already in use), appropriate error messages are displayed.
2. **Login:**
   - **Functionality:** Registered users can log in by providing their username/email and password.
   - **Usage:** Users enter their credentials on the login page and submit the form.
   - **Feedback:** If the login is successful, the user is redirected to their profile page or the homepage. If the login fails (e.g., incorrect password), an error message is displayed.

Profile Management

1. **Viewing Profile:**
   - **Functionality:** Users can view their profile, which includes their username, email, and a list of their posts.
   - **Usage:** After logging in, users can navigate to their profile page via a link in the navigation menu.
   - **Feedback:** The profile page displays the user's information and posts.

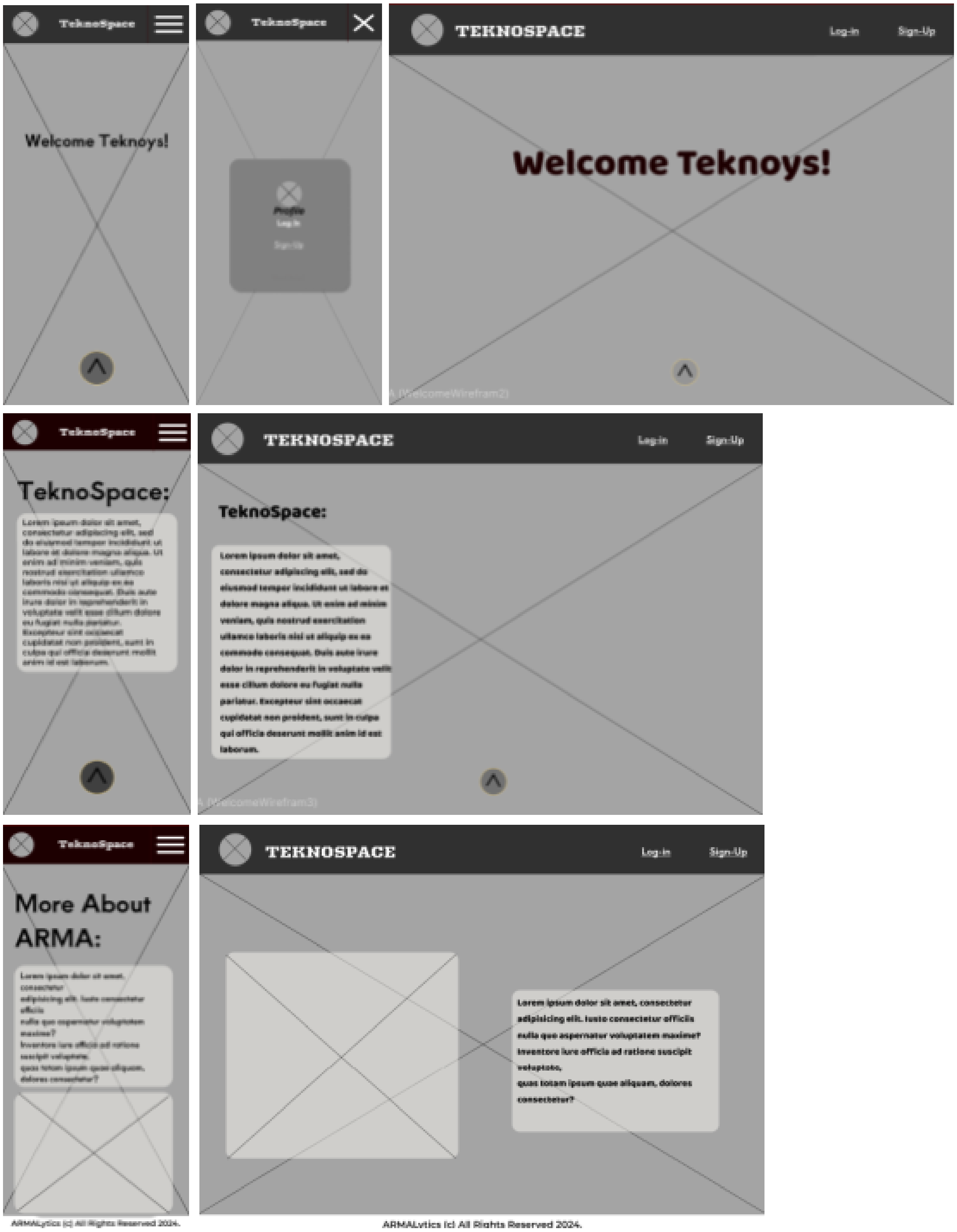
Posting Content and Interaction

1. **Creating a Post:**
   - **Functionality:** Users can create new posts with a title and content.
   - **Usage:** On the homepage or profile page, users click a "New Post" button, fill in the title and content, and submit the form.
   - **Feedback:** If the post is successfully created, it is displayed on the homepage and the user's profile page. If there are errors (e.g., missing title or content), appropriate error messages are shown.

2. **Commenting on Posts:**
   - ○ **Functionality:** Users can comment on posts made by other users.
   - ○ **Usage:** On the post details page, users enter a comment in the comment box and submit it.
   - ○ **Feedback:** If the comment is successfully added, it appears below the post. If there are errors (e.g., empty comment), appropriate error messages are shown.
3. **Liking Posts:**
   - ○ **Functionality:** Users can like posts to show their appreciation.
   - ○ **Usage:** On the post details page or homepage, users click a "Like" button next to a post.
   - ○ **Feedback:** If the like is successfully added, the like count for the post increases. If there are errors (e.g., user has already liked the post), appropriate error messages are shown.
4. **Edit Post**
   - ○ **Functionality:** Users can edit their own posts to update the title or content.
   - ○ **Usage:** On the post details page or profile page, users click the "Edit Post" button, make changes to the title or content, and submit the form.
   - ○ **Feedback:** If the post is successfully updated, the changes are reflected on the homepage and the user's profile page. If there are errors (e.g., missing title or content), appropriate error messages are shown.
5. **Delete Post**
   - ○ **Functionality:** Users can delete their own posts.
   - ○ **Usage:** On the post details page or profile page, users click the "Delete Post" button and confirm the deletion.
   - ○ **Feedback:** If the post is successfully deleted, it is removed from the homepage and the user's profile page. If there are errors (e.g., unable to delete the post), appropriate error messages are shown.

6.2 Screen Images

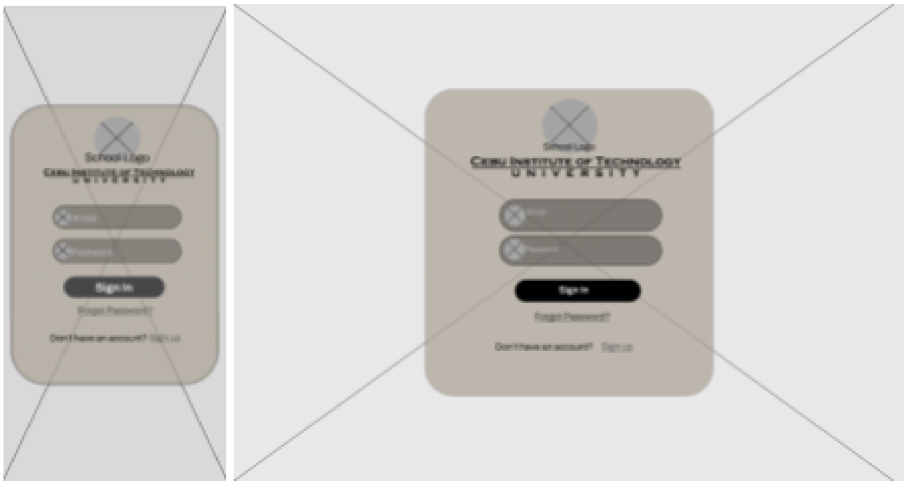
Welcome Page



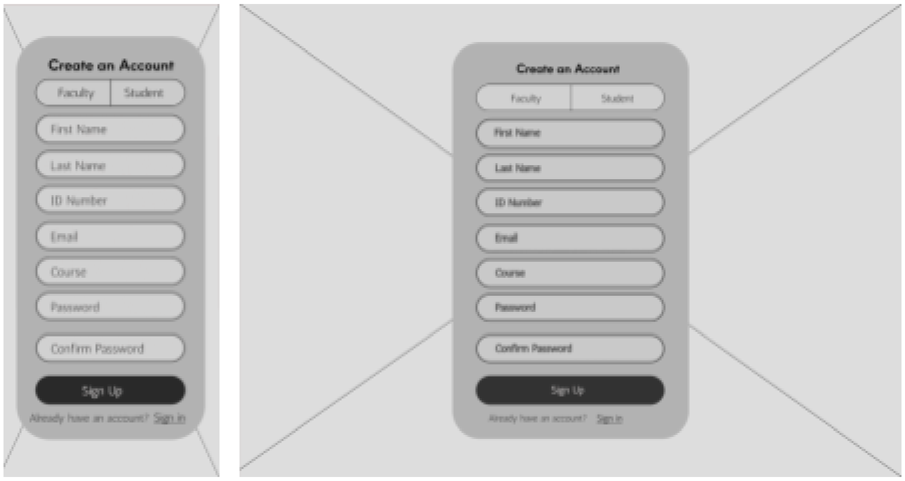
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iusto consectetur officiis nulla quo aspernatur voluptatem maxime? Inventore iure officia ad ratione suscipit voluptates, quas totam ipsum quae aliquam, dolores consectetur?

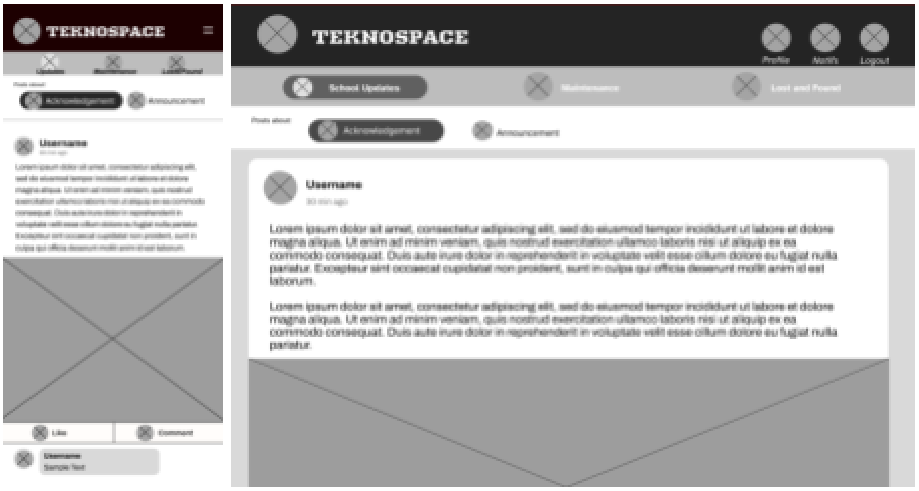
Sign In



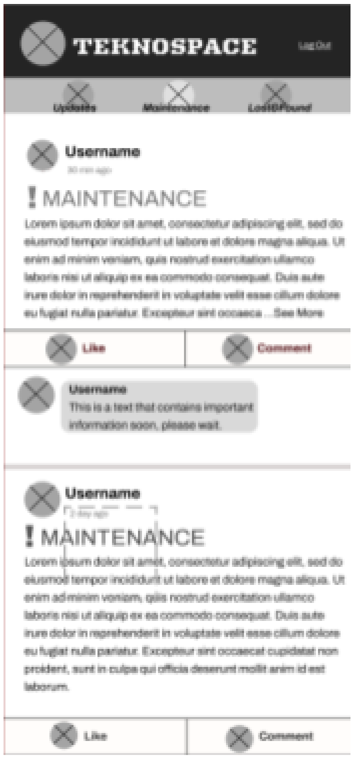
Sign up

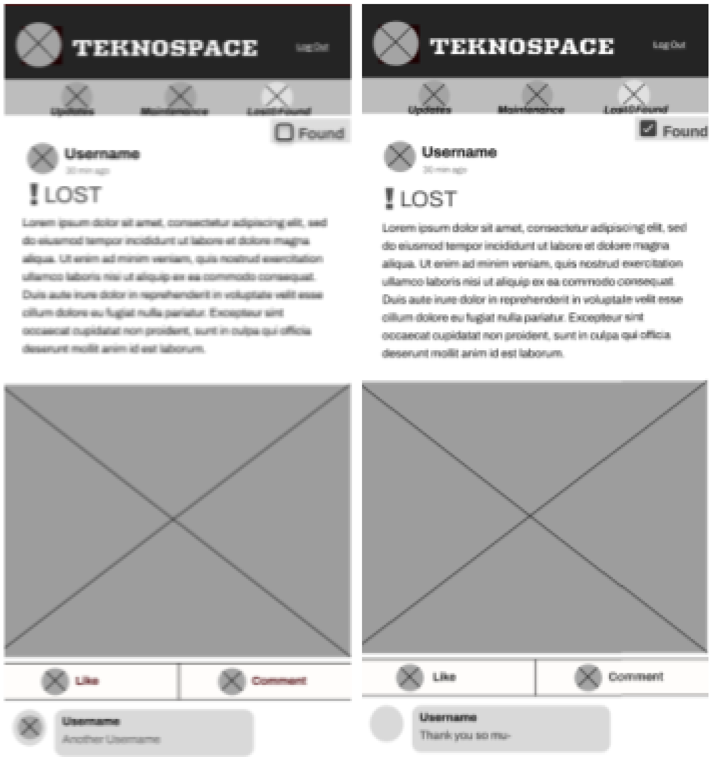


Student Homepage (School Updates)



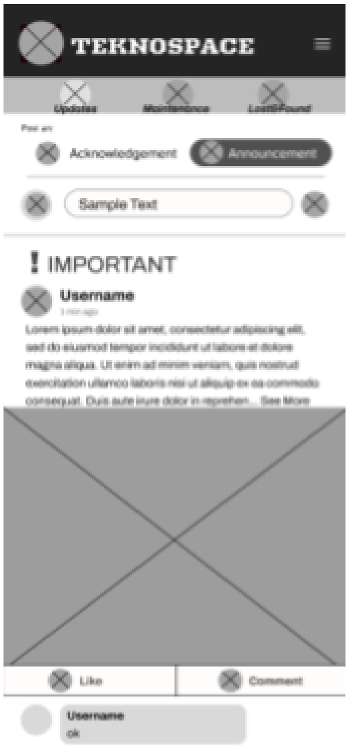
Maintenance **Lost and Found**





Faculty Homepage

News Feed



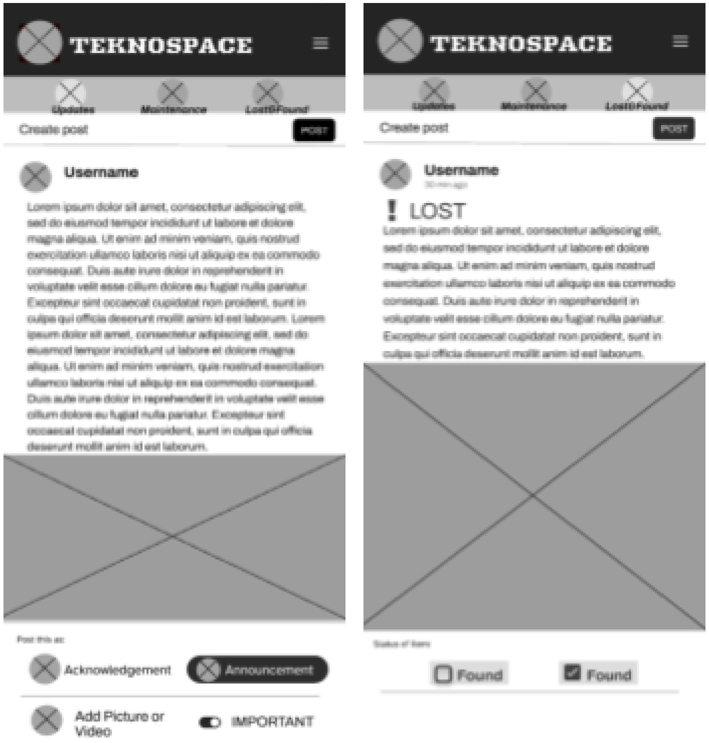
! IMPORTANT

Username

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehen... See More

Username
ok

Create Post (School Updates and Lost & Found)



Username

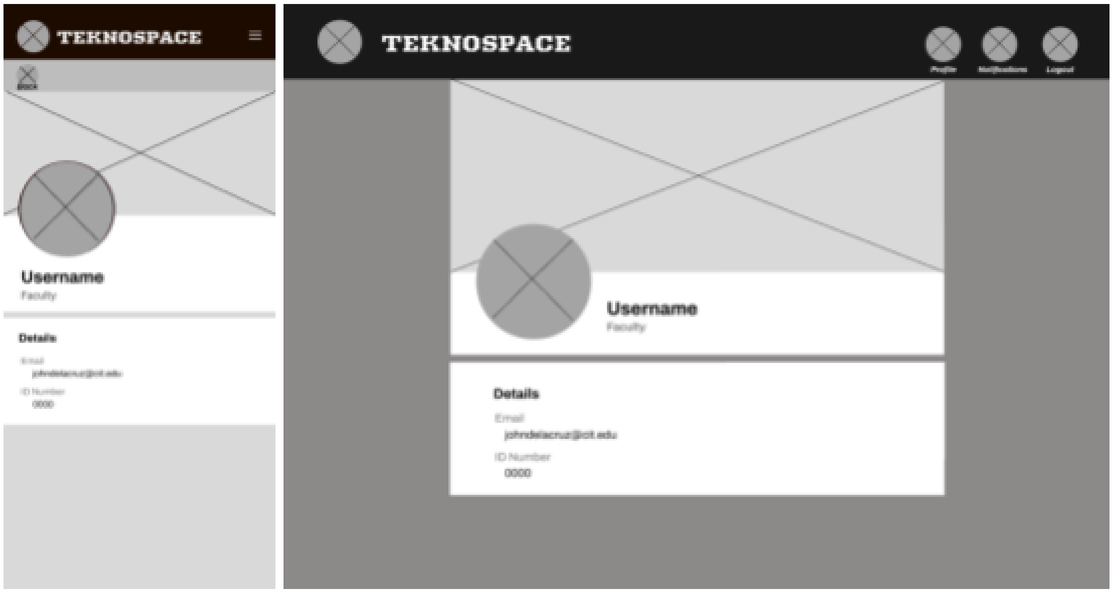
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Username

! LOST

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Profile Page



Username
Faculty

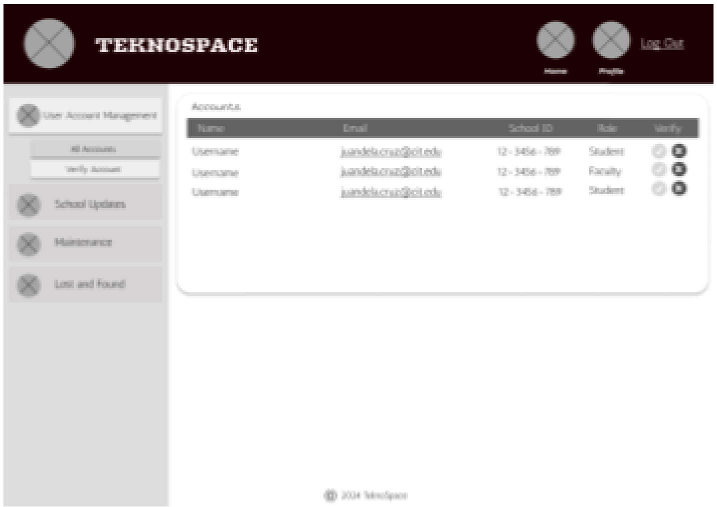
Details
Email
johndelacruz@cit.edu
ID Number
0000

Username
Faculty

Details
Email
johndelacruz@cit.edu
ID Number
0000

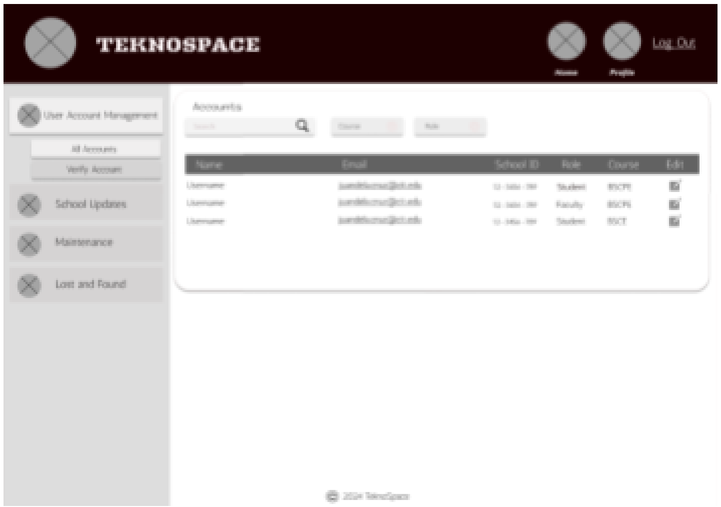
Admin Homepage

- **Verify, All and Edit Accounts**



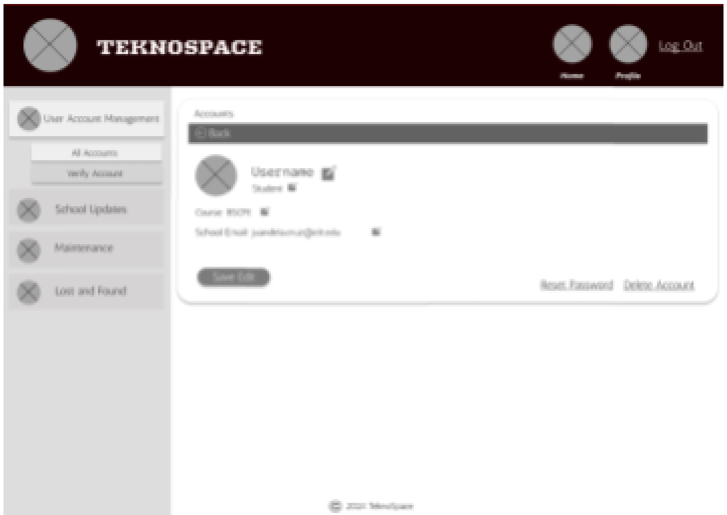
Accounts

Name	Email	School ID	Role	Verify
Username	juandela.cruz@cit.edu	12 - 3456 - 789	Student	✓ ☒
Username	juandela.cruz@cit.edu	12 - 3456 - 789	Faculty	✓ ☒
Username	juandela.cruz@cit.edu	12 - 3456 - 789	Student	✓ ☒



Accounts

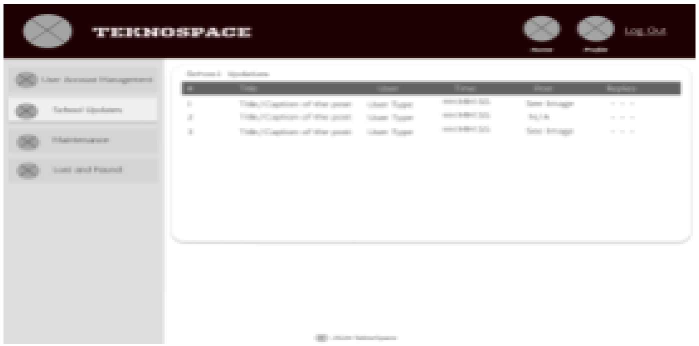
Name	Email	School ID	Role	Course	Edit
Username	juandelacruz@cit.edu	12 - 3456 - 789	Student	BSCPE	✎
Username	juandelacruz@cit.edu	12 - 3456 - 789	Faculty	BSCPE	✎
Username	juandelacruz@cit.edu	12 - 3456 - 789	Student	BSCE	✎



Course: BSCPE

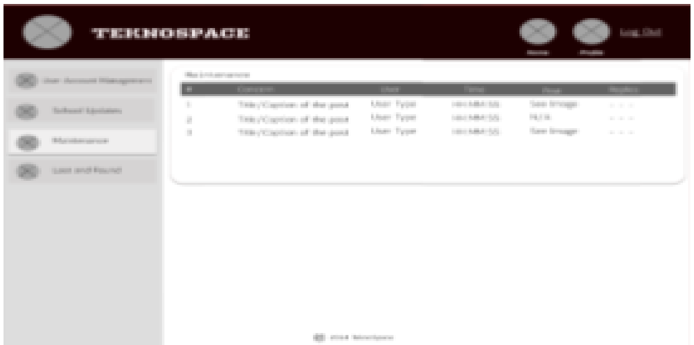
School Email: juandelacruz@cit.edu

- **School Updates, Maintenance, and Lost & Found**



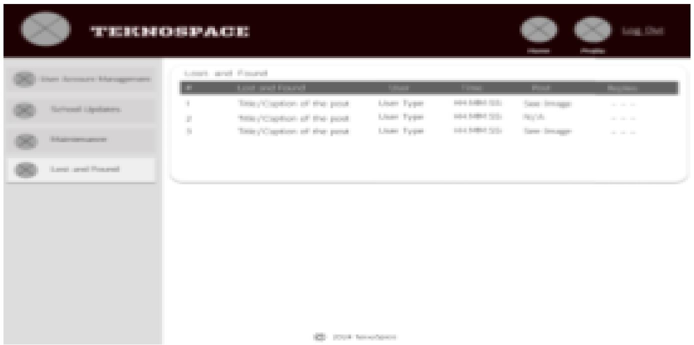
School Updates

#	Title	User	Time	Post	Replies
1	Title/Caption of the post	User Type	HH:MM:SS	See Image	- - -
2	Title/Caption of the post	User Type	HH:MM:SS	N/A	- - -
3	Title/Caption of the post	User Type	HH:MM:SS	See Image	- - -



Maintenance

#	Concern	User	Time	Post	Replies
1	Title/Caption of the post	User Type	HH:MM:SS	See Image	- - -
2	Title/Caption of the post	User Type	HH:MM:SS	N/A	- - -
3	Title/Caption of the post	User Type	HH:MM:SS	See Image	- - -



Lost and Found

#	Lost and Found	User	Time	Post	Replies
1	Title/Caption of the post	User Type	HH:MM:SS	See Image	- - -
2	Title/Caption of the post	User Type	HH:MM:SS	N/A	- - -
3	Title/Caption of the post	User Type	HH:MM:SS	See Image	- - -

6.3 Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

Welcome Page

1. **About Us Button:**
   - **Action:** Loads a part of the page that provides comprehensive information about the platform, its purpose, mission, and the history behind its development. The page may include text, images, and videos to explain these aspects.
2. **The Team Button:**
   - **Action:** Opens a part of the page that introduces the team members who developed and manage the platform, displaying their names, roles, and contact information.
3. **Services Button:**
   - **Action:** Details the various services and features offered by the platform. This includes sections on announcements, maintenance updates, and lost and found posts, each with a description and examples.
4. **Sign In Button:**
   - **Action:** Redirects the user to the Sign In page where users can enter their login credentials.
5. **Sign Up Button:**
   - **Action:** Redirects the user to the Sign Up page where new users can register for an account.

Sign In Page

1. **Login Form:**
   - **Fields:** Email or ID Number, Password
   - **Action:** Users enter their credentials and click the "Sign In" button to log in. The form submits the entered credentials to the server for authentication. If the credentials are correct, the user is redirected to their homepage. If incorrect, an error message is displayed, prompting the user to try again.
2. **Forgot Password Link:**
   - **Action:** Redirects users to a password reset page. Navigates to a page where users can enter their email address to receive password reset instructions.
3. **Sign Up Link:**
   - **Action:** Redirects new users to the Sign Up page to create an account.

Sign Up Page

1. **Registration Form:**

   - **Fields:** User Type (dropdown), First Name, Middle Name, Last Name, Email, Course, ID Number, Password, Confirm Password

   - **Action:** Users fill in the required fields and click the "Register" button to create a new account. If the registration is successful, the user is redirected to the login page with a success message. If there are validation errors, appropriate error messages are displayed, and the user is prompted to correct the issues.

Student Homepage

1. **News Feed:**

   - **Categories:** School Updates, Maintenance, Lost and Found

   - **Posts:** Title, Content, Attachments

   - **Actions:**

     - **Like Post:** Clicking the like button increments the like count for the post.

     - **Comment on Post:** Clicking the comment button opens a comment form where users can add their comments. The comment is added below the post.

Faculty Homepage

1. **News Feed:**

   - **Categories:** School Updates, Maintenance, Lost and Found

   - **Posts:** Title, Content, Attachments

   - **Actions:**

     - **Like Post:** Clicking the like button increments the like count for the post.

     - **Comment on Post:** Clicking the comment button opens a comment form where users can add their comments.

2. **Create Post Button:**

   - **Action:** Opens a form for faculty to create new posts.

3. **Create Post Form:**

   - **Fields:** Title, Content, Attachments (optional)

   - **Action:** Faculty fill in the fields and click the "Post" button to submit a new post. The form collects the post data and submits it to the server. If the post is successfully created, it appears in the relevant category of the news feed.

Admin Homepage

- **Manage Account Button:**
    - **Action:** A modal will pop up with two buttons to choose from: "All Accounts" and "Verify Accounts."
- **All Accounts Page:**
    - **Actions:**
        - **Remove User:** Administrators select a user and confirm deletion. The user account is removed from the system.
        - **Update User Information:** Opens a form to edit a user's information. Administrators modify the user's details and save the changes.
- **Verify Accounts Page:**
    - **Actions:**
        - **Verify Users:** This is to verify whether the new account is a faculty or student of the school or not.
        - **Remove Unverified User:** Administrators can remove unverified users who are not part of the official list of faculty and students.

Profile Page

- **Profile Information:**
    - Name, Email Address, ID Number, user type
- **Profile Customization:**
    - **Action:** Users can change their profile photo and cover photo.
- **Post Management:**
    - **Action:**
        - **Edit Post:** Faculty/Admin can edit their own posts to update the title or content.
        - **Delete Post:** Faculty/Admin can delete their own posts.
        - **Mark Important/ Remove Important:** Faculty/Admin can change the status of the post whether it is important or not.
        - **Found:** Faculty/Admin can change the status of a lost item post if the item has been found.

7. Requirements Matrix

Functional Requirements:

- FR1: User Registration (FACULTY & STUDENT)
- FR2: User Login (FACULTY & STUDENT)
- FR3: Post Announcement (FACULTY)
- FR4: Post Comment (FACULTY & STUDENT)
- FR5: Like Announcement (FACULTY & STUDENT)

Functional Requirement	**Description**	**System Component**	**Data Structure**
FR1	User Registration	registerUser (PHP)	Users (Table)
FR2	User Login	loginUser (PHP)	Users (Table)
FR3	Post Announcement	postAnnouncement (PHP)	Announcements (Table)
FR4	Post Comment	postComment (PHP)	Comments (Table)
FR5	Like Announcement	likeAnnouncement (PHP)	Likes (Table)

1. FR1: User Registration
   - Description: Allow new users to create an account.
   - System Component: registerUser function in PHP. Data
   - Structure: Users table in MySQL.
2. FR2: User Login Description:
   - Allow existing users to log in.
   - System Component: loginUser function in PHP.
   - Data Structure: Users table in MySQL.
3. FR3: Post Announcement
   - Description: Allow users to post announcements.

- System Component: postAnnouncement function in PHP.
- Data Structure: Announcements table in MySQL.

4. FR4: Post Comment
    - Description: Allow users to comment on announcements.
    - System Component: postComment function in PHP.
    - Data Structure: Comments table in MySQL.
5. FR5: Like Announcement Description:
    - Allow users to like announcements.
    - System Component: likeAnnouncement function in PHP.
    - Data Structure: Likes table in MySQL.

8. Appendices

What are HTML, CSS, & Javascript? (n.d.). CIAT. Retrieved June 30, 2024, from https://www.ciat.edu/blog/html-css-javascript/

Software Design Document. (n.d.). Bellevue College. Retrieved June 30, 2024, from https://www.bellevuecollege.edu/wp-content/uploads/sites/135/2019/04/SDD_RoadTrip.pdf