

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования «Пермский национальный исследовательский политехнический университет»
(ФГАОУ ВО «ПНИПУ»)
ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ
КАФЕДРА ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И АВТОМАТИЗИРОВАННЫЕ
СИСТЕМЫ

Отчет о практической работе на тему:
“Разработка платформы для обмена ассетами и инди-играми”

Выполнил:

Студент группы: РИС-22-26

Прядени И.А.

Проверил:

Кузнецов Д.Б.

г. Пермь - 2024

СОДЕРЖАНИЕ

Введение.....	3
1 Анализ предметной области.....	4
2 Проектирование системы.....	5
2.1 Выбор технологий.....	5
2.2 Проектирование базы данных.....	5
2.3 Пользовательский интерфейс.....	7
Выводы.....	7
3 Разработка веб-приложения.....	8
3.1 Подготовка веб-сервера.....	8
3.2 Разработка фронтенда и бэкенда.....	9
3.3 Проверка работоспособности веб-приложения.....	10
Выводы.....	12
Заключение.....	13
ПРИЛОЖЕНИЕ А. Запросы для создания таблиц базы данных.....	14
ПРИЛОЖЕНИЕ Б. Исходный код веб-приложения.....	17

Введение

Целью практической работы является разработка платформы для обмена ассетами и инди-играми.

Задачи:

- Анализ предметной области.
- Проектирование платформы обмена ассетами и инди-играми.
- Разработка веб-приложения платформы.

1 Анализ предметной области

Инди-игра – компьютерная игра, созданная одним разработчиком или небольшим коллективом без финансовой поддержки издателя компьютерных игр. Разработка подобных продуктов может затягиваться на многие месяцы и годы. Однако, не имея творческих ограничений, инди-разработчики часто создают инновационные и креативные проекты.

Публикация в свою очередь может вызвать определенные проблемы, так как разработка осуществляется без помощи издателя, обратная связь и количество играющих пользователей существенно уменьшается.

Ассет – цифровой объект, который представляет часть игрового контента. Часто при создании прототипов и основных систем разработка останавливается из-за недостатка тех или иных ассетов. Поэтому наличие платформы, облегчающей создание ассетов и получение обратной связи поможет ускорить процесс разработки.

Платформа должна обеспечить следующие функции:

- Регистрация и аутентификация пользователей: Возможность создавать учетные записи, входить и выходить из системы.
- Загрузка ассетов и игр: Пользователи должны иметь возможность загружать файлы, добавлять описания и изображения.
- Отзывы и рейтинги: Возможность оставлять отзывы и ставить оценки для контента.
- Управление контентом: Редактирование и удаление собственных ассетов и игр.
- Наличие форума: Зарегистрированные пользователи имеют доступ к форуму и обсуждения тем.

2 Проектирование системы

2.1 Выбор технологий

Операционной системой, являющейся основной веб-сервера, была выбрана система Linux и дистрибутив Arch Linux, предоставляющий больше возможностей настройки и гибкость. В качестве программного обеспечения, обрабатывающего входящие запросы и обслуживающего веб-контент, был выбран Apache HTTP Server, широко используемый веб-сервер с открытым исходным кодом, совместимый с различными операционными системами.

Для хранения данных была выбрана СУБД MariaDB, предоставляющая стабильность, производительность и высокую надежность.

Для реализации серверной логики динамического веб-сайта был выбран язык программирования PHP, который можно встроить в HTML и совместим со многими веб-приложениями.

2.2 Проектирование базы данных

Проектирование структуры базы данных включает создание основных таблиц и установление связей между ними для обеспечения целостности и согласованности данных. Основные таблицы и связи включают:

- **users:**
 - `id` (PRIMARY KEY) – уникальный идентификатор пользователя.
 - `username` – имя пользователя
 - `email` – электронная почта пользователя.
 - `password` – хешированный пароль пользователя.
 - `created_at` – дата и время создания учетной записи.
- **assets:**
 - `id` (PRIMARY KEY) – уникальный идентификатор ассета.
 - `user_id` (FOREIGN KEY) – идентификатор пользователя, загрузившего ассет.
 - `title` – название ассета.
 - `description` – описание ассета.
 - `file_path` – путь к файлу ассета.

- `image_path` – путь к изображению ассета.
- `created_at` – дата и время загрузки ассета.
- **asset_coments:**
 - `id` (PRIMARY KEY) - уникальный идентификатор комментария.
 - `asset_id` (FOREIGN KEY) - идентификатор ассета, к которому относится комментарий.
 - `user_id` (FOREIGN KEY) - идентификатор пользователя, оставившего комментарий.
 - `comment` - текст комментария.
 - `rating` - оценка ассета.
 - `created_at` - дата и время оставления комментария.
- **downloads:**
 - `id` (PRIMARY KEY) – уникальный идентификатор загрузки.
 - `user_id` (FOREIGN KEY) – идентификатор пользователя, скачавшего ассет.
 - `asset_id` (FOREIGN KEY) – идентификатор скачанного ассета.
 - `download_time` - дата и время скачивания ассета.
- **forum_topics:**
 - `id` (PRIMARY KEY) уникальный идентификатор темы форума.
 - `title` - название темы.
 - `user_id` (FOREIGN KEY) – идентификатор пользователя, создавшего тему.
 - `created_at` - время и дата создания темы.
- **forum_messages:**
 - `id` (PRIMARY KEY) – уникальный идентификатор сообщения.
 - `topic_id` (FOREIGN KEY) - идентификатор темы, к которой относится сообщение.
 - `user_id` (FOREIGN KEY) - идентификатор пользователя, оставившего сообщение.
 - `message` - текст сообщения.

- `created_at` - дата и время оставленного сообщения.

2.3 Пользовательский интерфейс

Проектирование пользовательского интерфейса направлено на создание удобного и интуитивно понятного взаимодействия с пользователем.

Основные элементы интерфейса включают:

- **Навигационная панель:**
 - Ссылки на разделы сайта (Assets, Forum, Upload, Register/Login или Logout).
 - Отображение имени пользователя с ссылкой на его профиль.
- **Главная страница:**
 - Список популярных и новых ассетов.
 - Возможность фильтрации и поиска контента.
- **Страница пользователя:**
 - Отображение профиля пользователя, его загруженных ассетов и игр.
 - Возможность редактирования и удаления собственных ассетов.
- **Страница ассета:**
 - Детальная информация об ассете.
 - Кнопка для скачивания.
 - Раздел для комментариев и оценок.
- **Форум:**
 - Список тем форума.
 - Возможность создания новых тем.
 - Отображение сообщений в теме с возможностью добавления новых сообщений.

Выводы

Этап проектирования системы платформы для обмена ассетами и инди-играми позволил сформировать четкое и структурированное представление о будущей системе.

3 Разработка веб-приложения

3.1 Подготовка веб-сервера

Для успешной реализации проекта и его последующего функционирования на веб-сервере, необходимо правильно настроить окружение. В данном случае используется Arch Linux и Apache HTTP Server. Этот этап включает в себя следующие шаги:

1. Установка Apache HTTP Server:

1. Обновление системы:

```
sudo pacman -Syu
```

2. Установка Apache HTTP Server:

```
sudo pacman -S apache
```

2. Настройка Apache:

1. Редактирование файла конфигурации Apache:

```
sudo nano /etc/http/conf/httpd.conf
```

2. Измените параметров для базовой настройки:

```
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User http
Group http
ServerAdmin you@example.com
DocumentRoot "/srv/http"
<Directory "/srv/http">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

3. Запуск и настройка автозапуска Apache

1. Запуск Apache HTTP Server:

```
sudo systemctl start httpd
```

2. Настройка автозапуска Apache при загрузке системы:

```
sudo systemctl enable httpd
```

4. Установка PHP:

1. Установка PHP и необходимых модулей:

```
sudo pacman -S php php-apache
```


2. Настройка Apache для работы с PHP:

```
LoadModule php_module modules/libphp.so
AddHandler php-script .php
Include conf/extra/php_module.conf
```

5. Установка и настройка базы данных MySQL/MariaDB

1. Установка MariaDB:

```
sudo pacman -S mariadb
```

2. Инициализация базы данных:

```
sudo mariadb-install-db --user=mysql --basedir=/usr --datadir=/var/lib/mysql
```

3. Запуск и настройка MariaDB:

```
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

4. Создание базы данных и пользователя проекта:

```
CREATE DATABASE game_assets_db;
CREATE USER 'game_user'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON game_assets_db.* TO 'game_user'@'localhost';
FLUSH PRIVILEGES;
```

3.2 Разработка фронтенда и бэкенда

1. Настройка серверной части и базы данных:

Создание базы данных реализовано с помощью запросов CREATE TABLE, представленных в *Приложении А*.

Подключение к базе данных выполнено с помощью использования PHP расширения mysqli в файле “config.php” и последующим включением его в необходимые страницы.

2. Аутентификация и регистрация пользователей:

Созданы страницы для регистрации (register.php) и входа (login.php) пользователей. Реализована логика хэширования паролей и сохранения информации о пользователях в сессиях.

3. Управление контентом:

Реализованы функции загрузки ассетов (upload.php), их редактирования (edit_asset.php) и удаления (delete_asset.php).

Созданы страницы для отображения списка всех ассетов и детальной информации об отдельных ассетах с возможностью скачивания (index.php, asset.php).

4. Комментарии и рейтинг:

Реализована возможность добавления комментариев и рейтингов к ассетам (asset.php), а также отображение всех комментариев к конкретному ассету.

5. Форум:

Созданы страницы для форума, где пользователи могут создавать новые темы и оставлять сообщения в существующих темах (forum.php, topic.php).

6. Стилизация интерфейса:

CSS использован для стилизации элементов интерфейса, таких как навигационная панель, кнопки, формы и т.д.

Стили CSS сохраняются в отдельном файле style.css, который подключается ко всем страницам приложения.

7. Обработка форм и динамическое отображение данных:

Формы для регистрации, входа, добавления и редактирования ассетов, создания комментариев и форумных сообщений созданы с использованием HTML и обработаны на стороне сервера с использованием PHP.

Данные, такие как список ассетов и комментарии, динамически выводятся на страницы с использованием PHP и MySQL-запросов.

3.3 Проверка работоспособности веб-приложения

При входе на сайт, пользователя встречает страница с ассетами (рис. 1). Здесь он может скачивать ассеты или игры, зарегистрироваться или зайти в ранее созданный аккаунт.

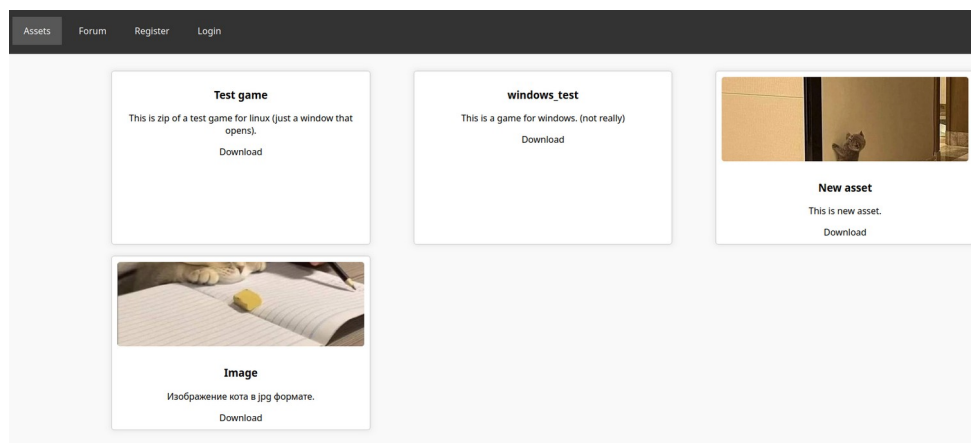
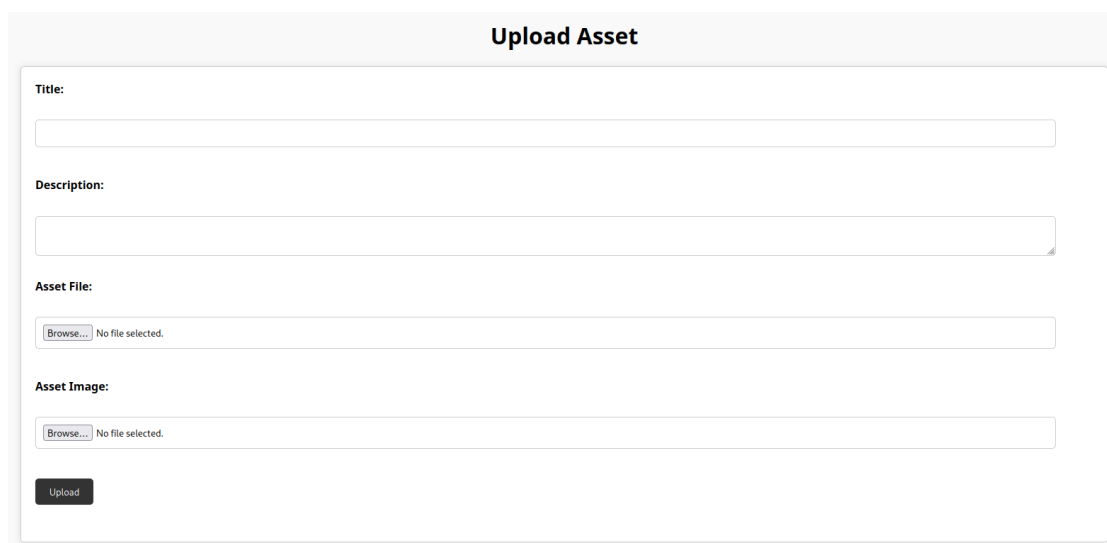


Рис. 1 – Страница ассетов

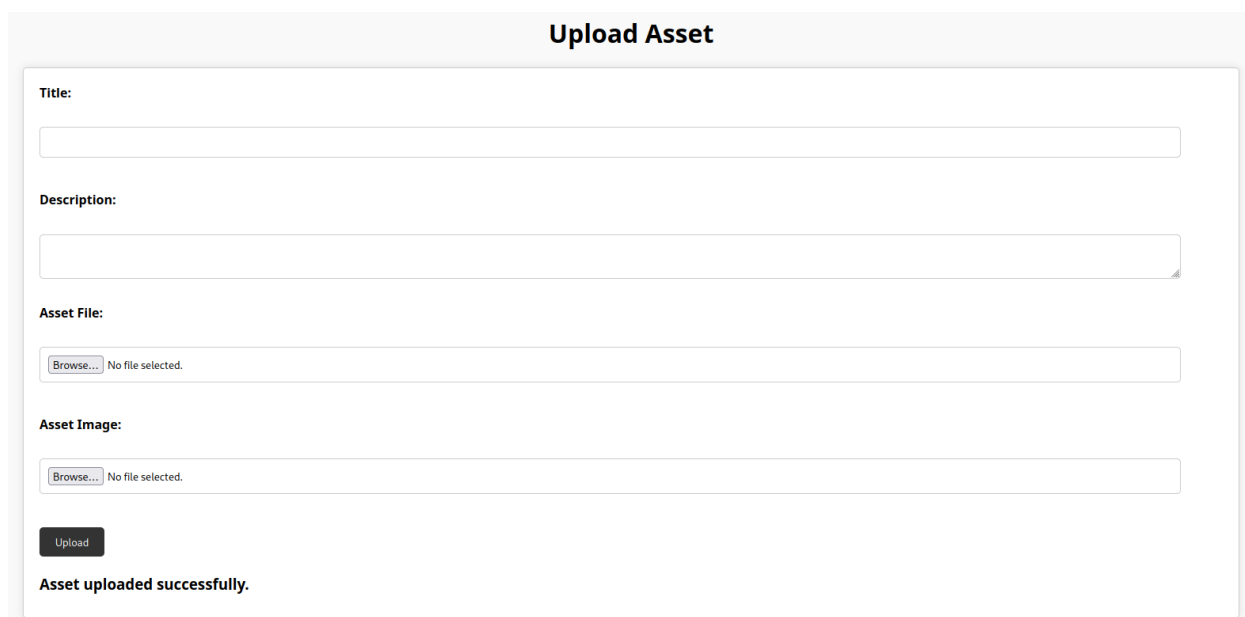
Если пользователь имеет учетную запись, появляется возможность загрузки (рис. 2) ассета или игры.



The screenshot shows a web form titled "Upload Asset". It contains four input fields: "Title:" (a single-line text box), "Description:" (a multi-line text area), "Asset File:" (a file selection box with a "Browse..." button and the text "No file selected."), and "Asset Image:" (another file selection box with a "Browse..." button and the text "No file selected."). At the bottom of the form is a dark "Upload" button.

Рис. 2 – Страница загрузки нового ассета

В случае успешной загрузки появляется соответствующее сообщение (рис. 3). То же происходит в случае ошибки.



This screenshot shows the same "Upload Asset" form as in Figure 2, but with an additional message at the bottom: "Asset uploaded successfully." The form fields and the "Upload" button remain visible above this message.

Рис. 3 – Успешная загрузка ассета

Также если пользователь зарегистрирован, появляется возможность зайти на страницу управления профилем (рис. 4). На данной странице показана основная информация аккаунта а также все ассеты и игры, загруженные текущим пользователем. Страница содержит возможность скачивания, изменения или удаления ассетов пользователя.

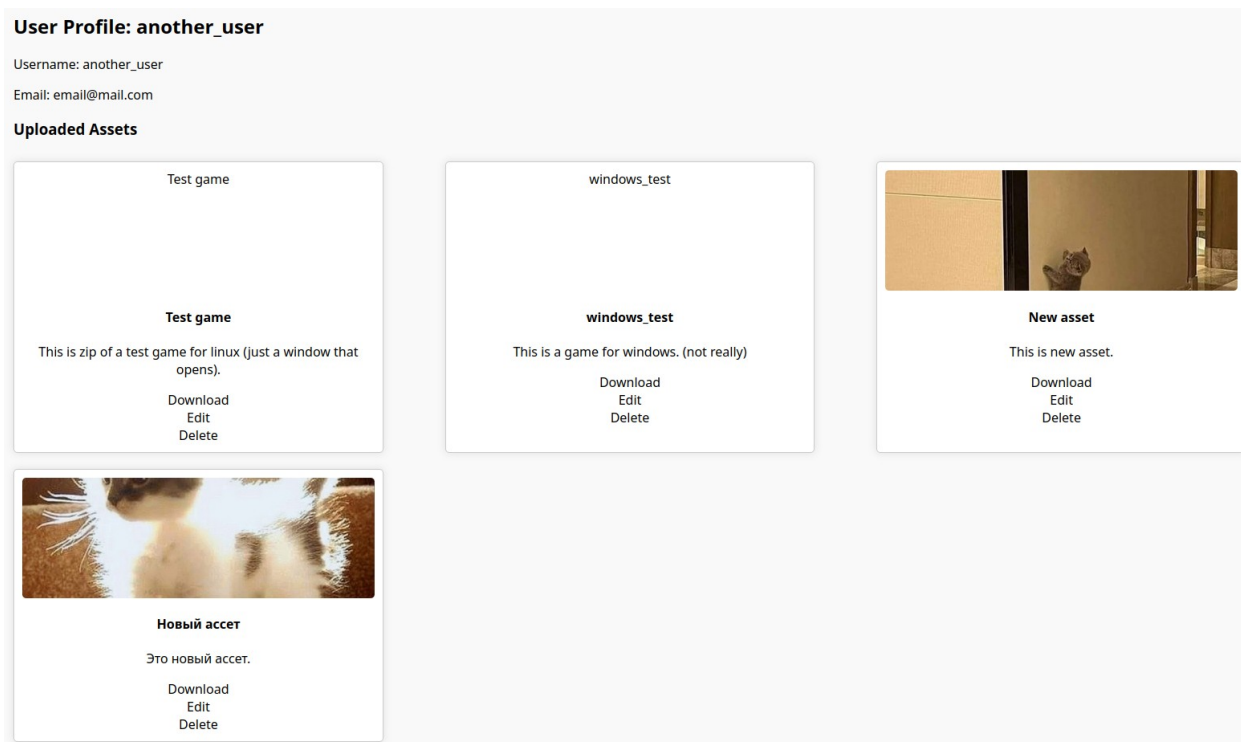


Рис. 4 – Страница пользователя

Выводы

В результате разработки бэкенда и фронтенда была создана веб-платформа для загрузки и скачивания инди-игр и ассетов, а также взаимодействия пользователей через комментарии и форумные темы. Исходный код вышеперечисленных файлов представлен в *Приложении Б*. Также была проведена проверка работоспособности веб-приложения.

Заключение

В ходе данной работы была разработана платформа для обмена ассетами и инди-играми. Были определены требования, спроектирована структура базы данных, выбраны подходящие технологии и реализовано веб-приложение. Проведена проверка работоспособности веб-приложения, что подтвердило корректность его работы.

Перспективы развития платформы включают:

- Дальнейшее развитие и доработка веб-приложения;
- Добавление возможности организации геймджемов;
- Введение платежных систем для покупки ассетов и инди-игр.

Практическое применение данной платформы имеет значительный потенциал для инди-разработчиков, облегчая обмен ассетами, получение обратной связи и сотрудничество в рамках разработки игр.

ПРИЛОЖЕНИЕ А. Запросы для создания таблиц базы данных.

```
CREATE TABLE `asset_comments` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `asset_id` int(11) NOT NULL,  
  `user_id` int(11) NOT NULL,  
  `comment` text NOT NULL,  
  `rating` int(11) NOT NULL,  
  `created_at` timestamp NULL DEFAULT current_timestamp(),  
  PRIMARY KEY (`id`),  
  KEY `asset_id` (`asset_id`),  
  KEY `user_id` (`user_id`),  
  CONSTRAINT `asset_comments_ibfk_1` FOREIGN KEY (`asset_id`)  
REFERENCES `assets` (`id`),  
  CONSTRAINT `asset_comments_ibfk_2` FOREIGN KEY (`user_id`)  
REFERENCES `users` (`id`));
```

```
CREATE TABLE `assets` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `user_id` int(11) NOT NULL,  
  `title` varchar(100) NOT NULL,  
  `description` text DEFAULT NULL,  
  `file_path` varchar(255) NOT NULL,  
  `image_path` varchar(255) DEFAULT NULL,  
  `created_at` timestamp NULL DEFAULT current_timestamp(),  
  PRIMARY KEY (`id`),  
  KEY `user_id` (`user_id`),  
  CONSTRAINT `assets_ibfk_1` FOREIGN KEY (`user_id`)  
REFERENCES `users` (`id`));
```

```
CREATE TABLE `downloads` (  

```

```
`id` int(11) NOT NULL AUTO_INCREMENT,  
`user_id` int(11) NOT NULL,  
`asset_id` int(11) NOT NULL,  
`download_time` timestamp NULL DEFAULT current_timestamp(),  
PRIMARY KEY (`id`),  
KEY `user_id` (`user_id`),  
KEY `asset_id` (`asset_id`),  
    CONSTRAINT `downloads_ibfk_1` FOREIGN KEY (`user_id`)  
REFERENCES `users` (`id`),  
    CONSTRAINT `downloads_ibfk_2` FOREIGN KEY (`asset_id`)  
REFERENCES `assets` (`id`));
```

```
CREATE TABLE `forum_messages` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `topic_id` int(11) NOT NULL,  
    `user_id` int(11) NOT NULL,  
    `message` text NOT NULL,  
    `created_at` timestamp NULL DEFAULT current_timestamp(),  
    PRIMARY KEY (`id`),  
    KEY `topic_id` (`topic_id`),  
    CONSTRAINT `forum_messages_ibfk_1` FOREIGN KEY (`topic_id`)  
REFERENCES `forum_topics` (`id`));
```

```
CREATE TABLE `forum_topics` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `title` varchar(255) NOT NULL,  
    `user_id` int(11) NOT NULL,  
    `created_at` timestamp NULL DEFAULT current_timestamp(),  
    PRIMARY KEY (`id`));
```

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(50) NOT NULL,  
  `email` varchar(100) NOT NULL,  
  `password` varchar(255) NOT NULL,  
  `created_at` timestamp NULL DEFAULT current_timestamp(),  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `email` (`email`));
```


ПРИЛОЖЕНИЕ Б. Исходный код веб-приложения

“asset.php”:

```
<?php
require('config.php');
session_start();
require('include/functions.php');

$asset_id = isset($_GET['id']) ? (int)$_GET['id'] : 0;

$stmt = $conn->prepare("SELECT title, description, file_path, user_id
FROM assets WHERE id = ?");
$stmt->bind_param("i", $asset_id);
$stmt->execute();
$stmt->bind_result($title, $description, $file_path, $user_id);
$stmt->fetch();
$stmt->close();

$author_username = getUsername($conn, $user_id);

if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['comment'])) {
    $comment = $_POST['comment'];
    $rating = $_POST['rating'];
    $user_id = $_SESSION['user_id'];

    $stmt = $conn->prepare("INSERT INTO asset_comments (asset_id,
user_id, comment, rating) VALUES (?, ?, ?, ?)");
    $stmt->bind_param("iisi", $asset_id, $user_id, $comment, $rating);

    if ($stmt->execute()) {
```

```

$stmt->close();
header("Location: asset.php?id=$asset_id");
exit();
} else {
    echo "Error adding comment: " . $stmt->error;
}
}

```

```

$comments_result = $conn->query("SELECT ac.comment, ac.rating,
u.username, ac.created_at FROM asset_comments ac JOIN users u ON ac.user_id
= u.id WHERE ac.asset_id = $asset_id ORDER BY ac.created_at ASC");

```

```

$user_comment_stmt = $conn->prepare("SELECT id FROM
asset_comments WHERE asset_id = ? AND user_id = ?");

```

```

$user_comment_stmt->bind_param("ii", $asset_id, $_SESSION['user_id']);

```

```

$user_comment_stmt->execute();

```

```

$user_comment_stmt->store_result();

```

```

$user_has_commented = $user_comment_stmt->num_rows > 0;

```

```

$user_comment_stmt->close();

```

```

?>

```

```

<!DOCTYPE html>

```

```

<html>

```

```

<head>

```

```

<title><?php echo htmlspecialchars($title); ?></title>

```

```

<link rel="stylesheet" type="text/css" href="style.css">

```

```

</head>

```

```

<body>

```

```

<?php generateNavBar($conn) ?>

```

```

<div class="container">

```

```

<div class="asset-detail">

```

```

<h2><?php echo htmlspecialchars($title); ?></h2>
<p><?php echo htmlspecialchars($description); ?></p>
<p>Uploaded by <?php echo htmlspecialchars($author_username); ?
></p>

<a href="<?php echo $file_path; ?>" download>Download</a>

<!-- Список комментариев -->
<div class="comments">
    <h3>Comments</h3>
    <?php if ($comments_result->num_rows > 0): ?>
        <?php while ($row = $comments_result->fetch_assoc()): ?>
            <div class="comment">
                <p><?php echo htmlspecialchars($row['comment']);
?></p>

                <p>Rating: <?php echo $row['rating']; ?>/5</p>
                <p>Posted by <?php echo
htmlspecialchars($row['username']); ?> on <?php echo $row['created_at']; ?></p>
            </div>
        <?php endwhile; ?>
    <?php else: ?>
        <p>No comments available.</p>
    <?php endif; ?>
</div>

<?php if (isLoggedIn() && !$user_has_commented): ?>
    <div class="form-container">
        <h3>Add a new comment</h3>
        <form method="post">
            <textarea name="comment" rows="4" cols="50"
required></textarea><br>

```

```

        <label for="rating">Rating:</label>
        <select name="rating" id="rating" required>
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5</option>
        </select><br><br>
        <button type="submit">Post Comment</button>
    </form>
</div>

<?php elseif (isLoggedIn()): ?>
    <p>You have already commented on this asset.</p>
<?php else: ?>
    <p><a href="login.php">Login</a> to post comments.</p>
<?php endif; ?>
</div>
</div>
</body>
</html>

```

“config.php”:

```

<?php
$servername = "localhost";
$username = "pryazha";
$password = "89f519gh";
$dbname = "indie_platform";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {

```

```
        die("Connection failed: " . $conn->connect_error);
    }
?>
```

“delete_asset.php”:

```
<?php
require('config.php');
session_start();
require_once('include/functions.php');

if (!isLoggedIn()) {
    header("Location: login.php");
    exit();
}

if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id']))
{
    $asset_id = $_GET['id'];

    $stmt = $conn->prepare("SELECT title, file_path, image_path FROM
assets WHERE id = ?");
    $stmt->bind_param("i", $asset_id);
    $stmt->execute();
    $stmt->bind_result($title, $file_path, $image_path);
    $stmt->fetch();
    $stmt->close();
} elseif ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['asset_id'])) {
    $asset_id = $_POST['asset_id'];
    $file_path = $_POST['file_path'];
    $image_path = $_POST['image_path'];
```

```

$conn->begin_transaction();

$stmt_delete_comments = $conn->prepare("DELETE FROM
asset_comments WHERE asset_id = ?");
$stmt_delete_comments->bind_param("i", $asset_id);
$stmt_delete_comments->execute();
$stmt_delete_comments->close();

$stmt_delete_downloads = $conn->prepare("DELETE FROM downloads
WHERE asset_id = ?");
$stmt_delete_downloads->bind_param("i", $asset_id);
$stmt_delete_downloads->execute();
$stmt_delete_downloads->close();

$stmt_delete_asset = $conn->prepare("DELETE FROM assets WHERE
id = ?");
$stmt_delete_asset->bind_param("i", $asset_id);
if ($stmt_delete_asset->execute()) {
    $stmt_delete_asset->close();

    if (file_exists($file_path)) {
        unlink($file_path);
    }

    if ($image_path && file_exists($image_path)) {
        unlink($image_path);
    }

    $conn->commit();

```

```

        header("Location: user.php");
        exit();
    } else {
        $conn->rollback();
        $error_message = "Failed to delete asset.";
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Delete Asset</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <?php generateNavBar($conn); ?>

    <div class="container">
        <div class="header">
            <h2>Delete Asset</h2>
        </div>

        <div class="form-container">
            <?php if ($_SERVER["REQUEST_METHOD"] == "GET" &&
isset($_GET['id'])): ?>
                <p>Are you sure you want to delete the asset "<strong><?php
echo htmlspecialchars($title); ?></strong>"?</p>

                <form action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">

```

```

        <input type="hidden" name="asset_id" value="<?php echo
$asset_id; ?>">

        <input type="hidden" name="file_path" value="<?php echo
htmlspecialchars($file_path); ?>">

        <input type="hidden" name="image_path" value="<?php echo
htmlspecialchars($image_path); ?>">

        <button type="submit">Delete</button>

        <a href="user.php">Cancel</a>

    </form>

<?php endif; ?>

<?php if (!empty($error_message)): ?>
    <p class="error"><?php echo htmlspecialchars($error_message); ?
></p>

<?php endif; ?>
</div>
</div>
</body>
</html>

```

“edit_asset.php”:

```

<?php
require('config.php');
session_start();
require_once('include/functions.php');

if (!isLoggedIn()) {
    header("Location: login.php");
    exit();
}

```



```

        if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id']))
        {
            $asset_id = $_GET['id'];

            $stmt = $conn->prepare("SELECT title, description FROM assets
WHERE id = ?");

            $stmt->bind_param("i", $asset_id);
            $stmt->execute();
            $stmt->bind_result($title, $description);
            $stmt->fetch();
            $stmt->close();
        } elseif ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['asset_id'])) {
            $asset_id = $_POST['asset_id'];
            $title = $_POST['title'];
            $description = $_POST['description'];

            $stmt = $conn->prepare("UPDATE assets SET title = ?, description = ?
WHERE id = ?");

            $stmt->bind_param("ssi", $title, $description, $asset_id);
            if ($stmt->execute()) {
                $stmt->close();
                header("Location: user.php");
                exit();
            } else {
                $error_message = "Failed to update asset.";
            }
        }
    }
?>

<!DOCTYPE html>

```

```
<html lang="en">
<head>
    <title>Edit Asset</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <?php generateNavBar($conn); ?>

    <div class="container">
        <div class="header">
            <h2>Edit Asset</h2>
        </div>

        <div class="form-container">
            <form method="post">
                <input type="hidden" name="asset_id" value="<?php echo
$asset_id; ?>">
                <label for="title">Title:</label>
                <input type="text" id="title" name="title" value="<?php echo
htmlspecialchars($title); ?>" required>

                <label for="description">Description:</label>
                <textarea id="description" name="description" rows="4"
required><?php echo htmlspecialchars($description); ?></textarea>

                <button type="submit">Update Asset</button>
            </form>
            <?php
            if (!empty($error_message)) {
```

```

        echo '<p class="error">' . htmlspecialchars($error_message) .
'</p>';
    }
    ?>
</div>
</div>
</body>
</html>

```

“forum.php”:

```

<?php
require('config.php');
session_start();
require('include/functions.php');

if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['title'] && isset($_POST['message'])) {
    $title = $_POST['title'];
    $message = $_POST['message'];
    $user_id = $_SESSION['user_id'];

    $stmt = $conn->prepare("INSERT INTO forum_topics (title, user_id)
VALUES (?, ?)");
    $stmt->bind_param("si", $title, $user_id);

    if ($stmt->execute()) {
        $new_topic_id = $stmt->insert_id;
        $stmt->close();

        $stmt = $conn->prepare("INSERT INTO forum_messages (topic_id,
user_id, message) VALUES (?, ?, ?)");

```

```

$stmt->bind_param("iis", $new_topic_id, $user_id, $message);

if ($stmt->execute()) {
    $stmt->close();
    header("Location: forum.php");
    exit();
} else {
    echo "Error adding message: " . $stmt->error;
}
} else {
    echo "Error adding topic: " . $stmt->error;
}
}

// Получаем список тем
$topics_result = $conn->query("SELECT ft.id, ft.title, u.username,
ft.created_at FROM forum_topics ft JOIN users u ON ft.user_id = u.id ORDER
BY ft.created_at DESC");

?>
<!DOCTYPE html>
<html>
<head>
    <title>Forum</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <?php generateNavBar($conn) ?>
    <div class="container">
        <div class="forum">

```

```

<!-- Форма для создания новой темы -->
<?php if (isLoggedIn()): ?>
    <div class="form-container">
        <h2>Create New Topic</h2>
        <form method="post">
            Title: <input type="text" name="title" required><br>
            Message: <textarea name="message" rows="4" cols="50"
required></textarea><br>
            <button type="submit">Create Topic</button>
        </form>
    </div>
<?php else: ?>
    <p><a href="login.php">Login</a> to create new topics.</p>
<?php endif; ?>

<!-- Список тем форума -->
<h2>Forum Topics</h2>
<ul class="topics">
    <?php if ($topics_result->num_rows > 0): ?>
        <?php while ($row = $topics_result->fetch_assoc()): ?>
            <li>
                <a href="topic.php?id=<?php echo $row['id']; ?>"><?php
echo htmlspecialchars($row['title']); ?></a>
                <p>Posted by <?php echo
htmlspecialchars($row['username']); ?> on <?php echo $row['created_at']; ?></p>
            </li>
        <?php endwhile; ?>
    <?php else: ?>
        <li>No topics available.</li>
    <?php endif; ?>

```

```
</ul>
</div>
</div>
```

```
</body>
```

```
</html>
```

“index.php”:

```
<?php
```

```
require('config.php');
```

```
session_start();
```

```
require_once('include/functions.php');
```

```

    $assets_result = $conn->query("SELECT id, title, description, file_path,
image_path FROM assets");
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Indie Game and Asset Platform</title>
```

```
    <link rel="stylesheet" type="text/css" href="style.css">
```

```
</head>
```

```
<body>
```

```
    <?php generateNavBar($conn) ?>
```

```
    <div class="container">
```

```
        <div class="assets" id="assets">
```

```
            <?php if ($assets_result->num_rows > 0): ?>
```

```
                <?php while ($row = $assets_result->fetch_assoc()): ?>
```

```
                    <div class="tile">
```

```
                        <?php if ($row['image_path']): ?>
```

```

        
        <?php endif; ?>
        <h3><a href="asset.php?id=<?php echo $row['id']; ?>"><?php
echo htmlspecialchars($row['title']); ?></a></h3>
        <p class="asset-description"><?php echo
htmlspecialchars($row['description']); ?></p>
        <a href="<?php echo $row['file_path']; ?>"
download>Download</a>
    </div>
    <?php endwhile; ?>
    <?php else: ?>
        <p>No assets available.</p>
    <?php endif; ?>
</div>
</div>
</body>
</html>

```

“login.php”:

```

<?php
require('config.php');
session_start();
require_once('include/functions.php');

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = $_POST['email'];
    $password = $_POST['password'];

    $stmt = $conn->prepare("SELECT id, password FROM users WHERE
email = ?");

```

```

$stmt->bind_param("s", $email);
$stmt->execute();
$stmt->store_result();
$stmt->bind_result($id, $hashed_password);
$stmt->fetch();

        if ($stmt->num_rows > 0 && password_verify($password,
$hashed_password)) {
            $_SESSION['user_id'] = $id;
            header("Location: index.php");
            exit();
        } else {
            $status = "Invalid email or password.";
        }
    }
?>

<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <?php generateNavBar($conn) ?>
    <div class="container">
        <div class="header">
            <h1>Login</h1>
        </div>
        <div class="form-container">
            <form method="post">

```



```
<label for="email">Email:</label>
```

```
<input type="email" name="email" required><br>
```

```
<label for="password">Password:</label>
```

```
<input type="password" name="password" required><br>
```

```
<button type="submit">Login</button>
```

```
</form><br>
```

```
<h3><?php if (isset($status)) { echo $status; } ?></h3>
```

```
</div>
```

```
</div>
```

```
</body>
```

```
</html>
```

“logout.php”:

```
<?php
```

```
session_start();
```

```
session_unset();
```

```
session_destroy();
```

```
header("Location: index.php");
```

```
exit();
```

```
?>
```

“register.php”:

```
<?php
```

```
require('config.php');
```

```
session_start();
```

```
require_once('include/functions.php');
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['username']) && isset($_POST['email']) &&
isset($_POST['password'])) {
```

```
$username = $_POST['username'];  
$email = $_POST['email'];  
$password = password_hash($_POST['password'],  
PASSWORD_DEFAULT);
```

```
$stmt = $conn->prepare("INSERT INTO users (username, email,  
password) VALUES (?, ?, ?)");
```

```
$stmt->bind_param("sss", $username, $email, $password);
```

```
if ($stmt->execute()) {  
    $_SESSION['user_id'] = $stmt->insert_id;  
    header("Location: index.php");  
    exit();  
}
```

```
else {  
    $status = "Error: " . $stmt->error;  
}
```

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

```
}
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Register</title>
```

```
<link rel="stylesheet" type="text/css" href="style.css">
```

```
</head>
```

```
<body>
```

```
<?php generateNavBar($conn) ?>
```

```
<div class="container">
```

```
<div class="header">
```

```

        <h1>Register</h1>
    </div>
    <div class="form-container">
        <form action="register.php" method="post">
            <label for="username">Username:</label>
                <input type="text" name="username" id="username"
required><br>

            <label for="email">Email:</label>
                <input type="email" name="email" id="email" required><br>

            <label for="password">Password:</label>
                <input type="password" name="password" id="password"
required><br>

            <button type="submit">Register</button>
        </form><br>
        <h3><?php if (isset($status)) { echo $status; } ?></h3>
    </div>
</div>
</body>
</html>

“style.css”:
/* style.css */

body {
    font-family: Arial, sans-serif;
    background-color: #f9f9f9;
    margin: 0;
    padding: 0;

```

```
}
```

```
.nav {  
    background-color: #333;  
    overflow: hidden;  
    padding: 1em;  
}
```

```
.nav ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}
```

```
.nav ul li {  
    float: left;  
    margin-right: 10px;  
}
```

```
.nav ul li a {  
    color: white;  
    text-decoration: none;  
    padding: 14px 20px;  
    display: block;  
}
```

```
.nav ul li a:hover {  
    background-color: #575757;  
}
```

```
.user-info {  
    float: right;  
    color: white;  
}
```

```
.container {  
    width: 80%;  
    margin: 0 auto;  
    padding-top: 20px;  
}
```

```
.assets {  
    display: flex;  
    flex-wrap: wrap;  
    justify-content: space-between;  
}
```

```
.tile {  
    background-color: white;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    padding: 10px;  
    margin: 10px 0;  
    width: 30%;  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
    text-align: center;  
    box-sizing: border-box;  
}
```

```
.tile:hover {
```

```
background-color: #f0f0f0;  
}
```

```
.tile a {  
  color: black;  
  text-decoration: none;  
  display: block;  
}
```

```
.asset-description {  
  display: -webkit-box;  
  -webkit-box-orient: vertical;  
  -webkit-line-clamp: 3;  
  overflow: hidden;  
  text-overflow: ellipsis;  
  white-space: normal;  
}
```

```
.asset-image {  
  width: 100%;  
  height: 150px;  
  object-fit: cover;  
  border-radius: 5px;  
  margin-bottom: 10px;  
  overflow: hidden;  
}
```

```
.asset-image img {  
  width: 100%;  
  height: 100%;
```

```
    object-fit: cover;
}
```

```
.header {
    text-align: center;
    margin-bottom: 20px;
}
```

```
.form-container {
    background-color: white;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

```
.form-container form label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
}
```

```
.form-container form input,
.form-container form textarea {
    width: 95%;
    padding: 10px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
```

```
.form-container form button {  
    background-color: #333;  
    color: white;  
    padding: 10px 20px;  
    border: none;  
    border-radius: 5px;  
    cursor: pointer;  
}
```

```
.form-container form button:hover {  
    background-color: #575757;  
}
```

```
.comments-section {  
    background-color: white;  
    padding: 20px;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    margin-top: 20px;  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
}
```

```
.comments-section h3 {  
    margin-top: 0;  
}
```

```
.comment {  
    border-bottom: 1px solid #ccc;  
    padding: 10px 0;
```



```
}
```

```
.comment:last-child {  
    border-bottom: none;  
}
```

```
.topic {  
    background-color: #fff;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    padding: 20px;  
    margin-top: 20px;  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
}
```

```
.topic h2 {  
    margin-top: 0;  
}
```

```
.messages {  
    margin-top: 20px;  
}
```

```
.message {  
    margin-bottom: 20px;  
    padding: 10px;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    background-color: #f9f9f9;  
}
```

```
.message p {  
    margin: 0;  
}
```

```
.form-container {  
    background-color: #fff;  
    padding: 20px;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
    margin-top: 20px;  
}
```

```
.form-container h3 {  
    margin-top: 0;  
    margin-bottom: 10px;  
}
```

```
.form-container form textarea {  
    width: 95%;  
    padding: 10px;  
    margin-bottom: 10px;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
}
```

```
.form-container form button {  
    background-color: #333;  
    color: white;
```

```
padding: 10px 20px;
border: none;
border-radius: 5px;
cursor: pointer;
}
```

```
.form-container form button:hover {
    background-color: #575757;
}
```

“topic.php”:

```
<?php
require('config.php');
session_start();
require_once('include/functions.php');
```

```
$topic_id = isset($_GET['id']) ? (int)$_GET['id'] : 0;
```

```
$topic_stmt = $conn->prepare("SELECT ft.title, u.username, ft.created_at
FROM forum_topics ft JOIN users u ON ft.user_id = u.id WHERE ft.id = ?");
$topic_stmt->bind_param("i", $topic_id);
$topic_stmt->execute();
$topic_stmt->bind_result($topic_title, $topic_username, $topic_created_at);
$topic_stmt->fetch();
$topic_stmt->close();
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['message'])) {
    $message = $_POST['message'];
    $user_id = $_SESSION['user_id'];
```

```
$stmt = $conn->prepare("INSERT INTO forum_messages (topic_id,
user_id, message) VALUES (?, ?, ?)");
```

```
$stmt->bind_param("iis", $topic_id, $user_id, $message);
```

```
if ($stmt->execute()) {
```

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

```
    header("Location: topic.php?id=$topic_id");
```

```
    exit();
```

```
} else {
```

```
    echo "Error adding message: " . $stmt->error;
```

```
}
```

```
}
```

```
$messages_result = $conn->query("SELECT fm.message, u.username,
fm.created_at FROM forum_messages fm JOIN users u ON fm.user_id = u.id
WHERE fm.topic_id = $topic_id ORDER BY fm.created_at ASC");
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title><?php echo htmlspecialchars($topic_title); ?> - Forum</title>
```

```
    <link rel="stylesheet" type="text/css" href="style.css">
```

```
</head>
```

```
<body>
```

```
    <?php generateNavBar($conn) ?>
```

```
    <div class="container">
```

```
        <div class="topic">
```

```
            <h2><?php echo htmlspecialchars($topic_title); ?></h2>
```

```
            <p>Posted by <?php echo htmlspecialchars($topic_username); ?> on
```

```
<?php echo $topic_created_at; ?></p>
```

```

<div class="messages">
    <?php if ($messages_result->num_rows > 0): ?>
        <?php while ($row = $messages_result->fetch_assoc()): ?>
            <div class="message">
                <p><?php echo htmlspecialchars($row['message']); ?></p>
                <p>Posted by <?php echo
htmlspecialchars($row['username']); ?> on <?php echo $row['created_at']; ?></p>
            </div>
        <?php endwhile; ?>
    <?php else: ?>
        <p>No messages available.</p>
    <?php endif; ?>
</div>

<?php if (isLoggedIn()): ?>
    <div class="form-container">
        <h3>Add a new message</h3>
        <form method="post">
            <textarea name="message" rows="4" cols="50"
required></textarea><br>
            <button type="submit">Post Message</button>
        </form>
    </div>
<?php else: ?>
    <p><a href="login.php">Login</a> to post messages.</p>
<?php endif; ?>
</div>
</div>
</body>

```

```
</html>
```

“upload.php”:

```
<?php
```

```
require 'config.php';
```

```
session_start();
```

```
require('include/functions.php');
```

```
if (!isset($_SESSION['user_id'])) {  
    header("Location: login.php");  
    exit();  
}
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST" &&  
isset($_POST['title']) && isset($_POST['description']) && isset($_FILES['file'])  
&& isset($_FILES['image'])) {  
    $title = $_POST['title'];  
    $description = $_POST['description'];  
    $file = $_FILES['file'];  
    $image = $_FILES['image'];  
    $user_id = $_SESSION['user_id'];  
  
    $file_path = 'uploads/' . basename($file['name']);  
    if (move_uploaded_file($file['tmp_name'], $file_path)) {  
        $image_path = 'uploads/' . basename($image['name']);  
        if (move_uploaded_file($image['tmp_name'], $image_path)) {  
            $stmt = $conn->prepare("INSERT INTO assets (title, description,  
file_path, image_path, user_id) VALUES (?, ?, ?, ?, ?)");  
            $stmt->bind_param("ssssi", $title, $description, $file_path,  
$image_path, $user_id);
```

```

        if ($stmt->execute()) {
            $status = "Asset uploaded successfully.";
        } else {
            $status = "Error: " . $stmt->error;
        }
        $stmt->close();
    } else {
        $status = "Failed to upload image.";
    }
} else {
    $status = "Failed to upload file.";
}
}
?>

```

```

<!DOCTYPE html>
<html>
<head>
    <title>Upload Asset</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <?php generateNavBar($conn) ?>
    <div class="container">
        <div class="header">
            <h1>Upload Asset</h1>
        </div>
        <div class="form-container">
            <form action="upload.php" method="post"
enctype="multipart/form-data">

```

```

<label for="title">Title:</label><br>
<input type="text" name="title" id="title" required><br><br>

<label for="description">Description:</label><br>
<textarea name="description" id="description"
required></textarea><br><br>

<label for="file">Asset File:</label><br>
<input type="file" name="file" id="file" required><br><br>

<label for="image">Asset Image:</label><br>
<input type="file" name="image" id="image" required><br><br>

<button type="submit">Upload</button>
</form><br>
<h3><?php if (isset($status)) { echo $status; } ?></h3>
</div>
</div>
</body>
</html>

```

“user.php”:

```

<?php
require('config.php');
session_start();
require_once('include/functions.php');

if (!isLoggedIn()) {
    header("Location: login.php");
    exit();
}

```



```

$user_id = $_SESSION['user_id'];
$user_info = getUserInfo($conn, $user_id);
$username = $user_info['username'];

$stmt = $conn->prepare("SELECT id, title, description, file_path,
image_path FROM assets WHERE user_id = ?");
$stmt->bind_param("i", $user_id);
$stmt->execute();
$result = $stmt->get_result();
$stmt->close();

?>
<!DOCTYPE html>
<html>
<head>
    <title>User Profile</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <?php generateNavBar($conn); ?>
    <div class="container">
        <div class="profile">
            <h2>User Profile: <?php echo htmlspecialchars($username); ?
></h2>

            <div class="user-details">
                <p>Username: <?php echo htmlspecialchars($username);
?></p>

```

```

        <p>Email: <?php echo htmlspecialchars($user_info['email']); ?
></p>

</div>

<h3>Uploaded Assets</h3>
<div class="assets">
    <?php if ($assets_result->num_rows > 0): ?>
    <?php while ($row = $assets_result->fetch_assoc()): ?>
    <div class="tile">
        <div class="asset-image">
            ">
        </div>
        <h4><?php echo htmlspecialchars($row['title']); ?></h4>
        <p class="asset-description"><?php echo
htmlspecialchars($row['description']); ?></p>
        <a href="<?php echo $row['file_path']; ?>"
download>Download</a>
        <a href="edit_asset.php?id=<?php echo $row['id']; ?
>">Edit</a>
        <a href="delete_asset.php?id=<?php echo $row['id']; ?
>">Delete</a>
    </div>
    <?php endwhile; ?>
    <?php else: ?>
    <p>No assets uploaded.</p>
    <?php endif; ?>
</div>
</div>
</div>

```

```
</body>
```

```
</html>
```

“function.php”:

```
<?php
```

```
function isLoggedIn() {
```

```
    return isset($_SESSION['user_id']);
```

```
}
```

```
function getUsername($conn, $user_id) {
```

```
    $stmt = $conn->prepare("SELECT username FROM users WHERE id  
= ?");
```

```
    $stmt->bind_param("i", $user_id);
```

```
    $stmt->execute();
```

```
    $stmt->bind_result($username);
```

```
    $stmt->fetch();
```

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

```
    return $username;
```

```
}
```

```
function getUserInfo($conn, $user_id) {
```

```
    $stmt = $conn->prepare("SELECT username, email, created_at FROM  
users WHERE id = ?");
```

```
    $stmt->bind_param("i", $user_id);
```

```
    $stmt->execute();
```

```
    $stmt->bind_result($username, $email, $created_at);
```

```
    $stmt->fetch();
```

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

```
    return ['username' => $username,
```

```
        'email' => $email,
```

```
        'created_at' => $created_at];
```

```
}
```

```
function generateNavBar($conn) {  
    echo '<nav class="nav">';  
    echo '<ul>';  
    echo '<li><a href="index.php">Assets</a></li>';  
    echo '<li><a href="forum.php">Forum</a></li>';  
    if (isLoggedIn()) {  
        echo '<li><a href="upload.php">Upload</a></li>';  
        echo '<li><a href="logout.php">Logout</a></li>';  
    } else {  
        echo '<li><a href="register.php">Register</a></li>';  
        echo '<li><a href="login.php">Login</a></li>';  
    }  
    if (isLoggedIn()) {  
        echo '<div class="user-info">';  
        echo '<li><a href="user.php">' . htmlspecialchars(getUsername($conn,  
$_SESSION['user_id'])) . '</a></li>';  
        echo '</div>';  
    }  
    echo '</ul>';  
    echo '</nav>';  
}  
?>
```