Sabir Hussein

Student ID: 00623376

CHC163

Client Server Systems Assignment 2

22/04/25

CRN - 50249

URL - http://localhost:8000//index.php

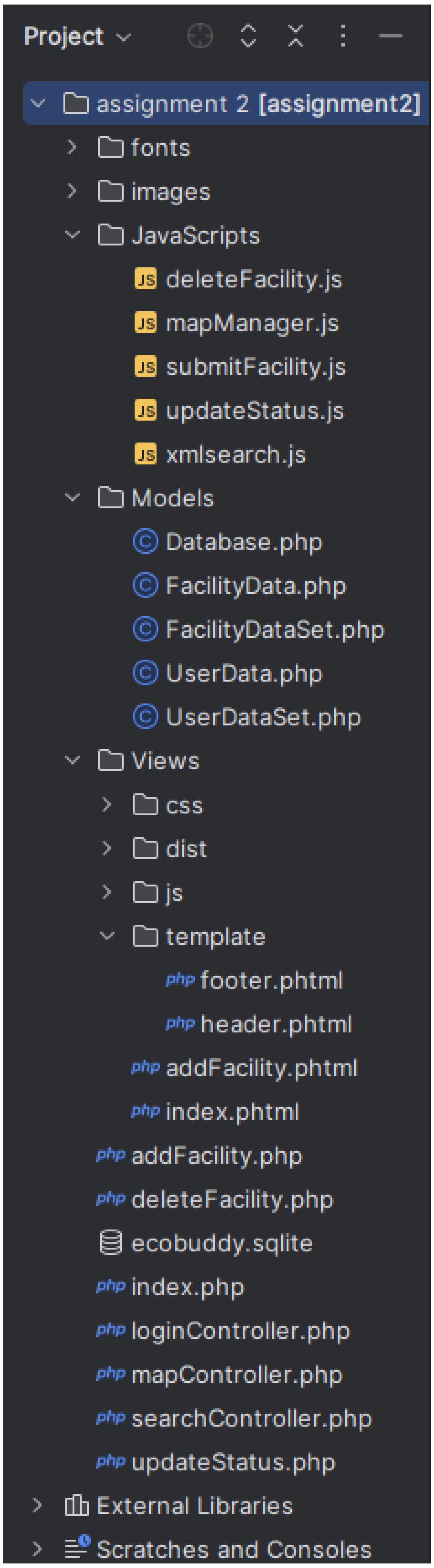
User	**Manager**	**User**
User ID	2	1
Username	Admin	Lee
Password	654321	123456
User Type	1	2

@00623376 Sabir Hussein Client Server Systems Assignment 2

Mark range %	**100-80%** Outstanding to Excellent *(60-48 marks)*	**79-60%** Very Good to Good *(47-36 marks)*	**59-40%** Fair to Adequate *(35-24 marks)*	**39-20%** Unsatisfactory to Poor *(23-12 marks)*	**19-0%** Very poor to Extremely poor *(11-0 marks)*
	"An excellent system ready to deliver to the customer"	**"A system with useful features"**	**"A minimum viable system with basic features"**	**"Incomplete features"**	**"Non- functioning system"**
Mapping **required feature** 60 marks	**Industry ready** application with **excellent performance** and **efficient data usage**. • Excellent OO code structure including reusable **classes**, design pattern(s) in both JavaScript and PHP as appropriate. ***(up to 10 marks)*** • Extensive use of **HTTPRequest** on **three or more** features of the solution UI to improve efficiency and performance using AJAX. ***(up to 10 marks)*** • Excellent and **secure input** validation, and demonstration of **security protection** such as URL tokens. ***(up to 10 marks)*** • **JSON or XML** data formats used for AJAX. PHP DB classes modified/extended to produce JSON/XML as necessary. ***(up to 10 marks)*** • **Sophisticated** map system with AJAX driven **real-time activity** and **updates**. Current user geo located. Outstanding user experience. Responsive layout. ***(up to 10 marks)*** • Excellent and consistently commented code. ***(up to 10 marks)***	• Reusable JavaScript **functions** or **classes added** to perform robust input validation and displaying data to the users **dynamically**. • At least **two** AJAX type data transactions to acquire data from the PHP backend. • **JSON or XML** data formats used for AJAX. • **Useful** map system functioning using **AJAX**. Current user geo located. Very good user experience. Responsive layout • Good comments evident throughout.	• Some JavaScript added to perform input validation and/or displaying data to the users dynamically. • At least **one** AJAX type data transactions to acquire data from the PHP backend. • Plain text data format used for AJAX. • **Basic** map/list system functioning using AJAX. Acceptable user experience. • Some useful code comments evident. • Data is stored in an SQLIte file.	• Some JavaScript added to perform basic input validation and/or displaying location data to the users from PHP backend but non-functioning or incomplete/has issues and unsatisfactory system. Poor user experience. • **Significant amounts of copied code. Poorly presented document. Use of AI tools.** • Minimal code comments. • Use of prohibited tools such as jQuery, mySQL server.	• Little or no JavaScript or extra features added to your semester one work. • **Significant amounts of copied code. Poorly presented document. Use of AI tools.** • Little or no user interface or experience. • No code comments. • Use of prohibited tools such as jQuery, , mySQL server.

Mark range %	**100-80%** Outstanding to Excellent *(40-32 marks)*	**79-60%** Very Good to Good *(31-24 marks)*	**59-40%** Fair to Adequate *(23-16)*	**39-20%** Unsatisfactory to Poor *(15-8 marks)*	**19-0%** Very poor to Extremely poor *(7-0 marks)*
Chosen feature 40 marks	**Industry ready** feature with **excellent performance** and **efficient data usage**. • **Excellent** OO code structure including reusable **classes**, design pattern, in both JavaScript and PHP as appropriate. ***(up to 8 marks)*** • Extensive use of **HTTPRequest** on different features of the feature to improve sophistication, efficiency and performance using AJAX. ***(up to 8 marks)*** • Excellent and **secure input** validation, and demonstration of **security protection** such as URL tokens. ***(up to 8 marks)*** • JSON or XML data formats used for AJAX. PHP DB classes modified/extended to produce JSON/XML as necessary ***(up to 8 marks)*** • Excellent and consistently commented code ***(up to 8 marks)***	• Reusable JavaScript **functions** or classes added to perform robust input validation and displaying data to the users **dynamically**. • At least **two** AJAX type data transactions for your chosen feature. • JSON or XML data formats used for AJAX. • Good comments evident throughout.	• Some JavaScript added to perform robust input validation and/or displaying data to the users dynamically. • At least **one** AJAX type data transaction for the chosen extra feature. • Plain text data format used for AJAX. • Some useful code comments evident. • Data is stored in an SQLIte file.	• Some JavaScript added to perform basic input validation or displaying data to the users but non-functioning, incomplete and unsatisfactory extra feature. • **Significant amounts of copied code. Poorly presented document. Use of AI tools.** • Minimal code comments. • Use of prohibited tools such as jQuery, mySQL server.	• Little or no JavaScript or extra features added to your semester one work. • **Significant amounts of copied code. Poorly presented document. Use of AI tools.** • No code comments. • Use of prohibited tools such as jQuery , mySQL server.

Project File Structure



```
Project
assignment 2 [assignment2]
  fonts
  images
  JavaScripts
    deleteFacility.js
    mapManager.js
    submitFacility.js
    updateStatus.js
    xmlsearch.js
  Models
    Database.php
    FacilityData.php
    FacilityDataSet.php
    UserData.php
    UserDataSet.php
  Views
    css
    dist
    js
    template
      footer.phtml
      header.phtml
    addFacility.phtml
    index.phtml
  addFacility.php
  deleteFacility.php
  ecobuddy.sqlite
  index.php
  loginController.php
  mapController.php
  searchController.php
  updateStatus.php
External Libraries
Scratches and Consoles
```

Model Files

FacilityData.php

```
<?php

/*
 * Class Facility Data
 *
 * Represents data for each facility
 */
class FacilityData
{
    protected $_id, $_title,
        $_category, $_description,
        $_houseNumber, $_streetName,
        $_county, $_town, $_postcode,
        $_lng, $_lat, $_contributor, $_status; // class for each facility

    public function __construct($dbRow) { // initialises the variables to
the corresponding row in the database
        $this->_id = $dbRow['id'];
        $this->_title = $dbRow['title'];
        $this->_category = $dbRow['category'];
        $this->_description = $dbRow['description'];
        $this->_houseNumber = $dbRow['houseNumber'];
        $this->_streetName = $dbRow['streetName'];
        $this->_county = $dbRow['county'];
        $this->_town = $dbRow['town'];
        $this->_postcode = $dbRow['postcode'];
        $this->_lng = $dbRow['lng'];
        $this->_lat = $dbRow['lat'];
        $this->_contributor = $dbRow['contributor'];
        $this->_status = $dbRow['status'];
    }
    // accessors
    public function getID() { // id accessor
        return $this->_id;
    }

    public function getTitle() { // title accessor
        return $this->_title;
    }

    public function getCategory() { // category accessor
        return $this->_category;
    }

    public function getDescription() { // description accessor
        return $this->_description;
    }
    public function getHouseNumber() { // house number accessor
        return $this->_houseNumber;
    }
    public function getStreetName() { // street name accessor
        return $this->_streetName;
    }
    public function getCounty() { // county accessor
        return $this->_county;
    }
    public function getTown() { // town accessor
        return $this->_town;
    }

    public function getPostCode() { // post code accessor
        return $this->_postcode;
    }
    public function getLng() { // longitude accessor
        return $this->_lng;
    }

    public function getLat() { // latitude accessor
        return $this->_lat;
    }

    public function getContributor() { // contributor accessor
        return $this->_contributor;
    }

    public function getStatus() { // status accessor
        return $this->_status;
    }

}
```

FacilityDataSet.php

```
<?php

require_once ('Database.php');          // Connects to the shared PDO
instance
require_once('FacilityData.php');      // Facility model to represent each
facility row as an object

class FacilityDataSet
{
    protected $_dbHandle, $_dbInstance;

    // Constructor: Set up the database connection using the Singleton
Database class
    public function __construct()
    {
        $this->_dbInstance = Database::getInstance();              // Get
shared DB instance
        $this->_dbHandle = $this->_dbInstance->getDbConnection();  // Get
the PDO handle
    }

    /**
     * Fetch all facilities as an array of FacilityData objects.
     * Used where full object properties are needed
     */
    public function fetchAllFacilities()
    {
        $sqlQuery = 'SELECT * FROM ecoFacilities';
        $statement = $this->_dbHandle->prepare($sqlQuery);
        $statement->execute();

        $dataSet = [];
        while ($row = $statement->fetch()) {
            $dataSet[] = new FacilityData($row); // Wrap each row in a
FacilityData object
        }
        return $dataSet;
    }

    /**
     * Search for facilities using keyword filtering on various fields.
     * Used by the live search suggestion and results display.
     */
    public function searchFacilities(string $query): array {
        $sql = "SELECT * FROM ecoFacilities
                WHERE title LIKE :q
                   OR category LIKE :q
                   OR description LIKE :q
                   OR streetName LIKE :q
                   OR county LIKE :q
                   OR town LIKE :q
                   OR postcode LIKE :q
                ORDER BY title ASC";

        $statement = $this->_dbHandle->prepare($sql);
        $statement->bindValue(':q', '%' . $query . '%');
        $statement->execute();

        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    /**
     * Get all facility records joined with status names.
     * Used for rendering map markers with human-readable status.
     */
    public function getAllFacilities(): array {
        $statement = $this->_dbHandle->query("
            SELECT f.id, f.title, f.description, f.town, f.lat, f.lng,
s.statusComment AS status
            FROM ecoFacilities f
            JOIN ecoFacilityStatus s ON f.status = s.id
        ");
        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    /**
     * Fetch all users to use as a contributor lookup map
     */
    public function getAllUsers(): array {
        $statement = $this->_dbHandle->query("SELECT id, username FROM
ecoUser");
        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    /**
     * Return all statuses as an associative map
     */
    public function getAllStatuses(): array {
        $statement = $this->_dbHandle->query("SELECT id, statusComment FROM
ecoFacilityStatus");

        $statusMap = [];
        while ($row = $statement->fetch(PDO::FETCH_ASSOC)) {
            $statusMap[$row['id']] = $row['statusComment'];
```

```php
        }

        return $statusMap;
    }

    /**
     * Insert a new facility into the database.
     * Used when managers add new records from the form.
     */
    public function insertFacility(array $data): bool {
        $statement = $this->_dbHandle->prepare("
            INSERT INTO ecoFacilities
            (title, category, description, houseNumber, streetName, county,
town, postcode, lng, lat, contributor, status)
            VALUES
            (:title, :category, :description, :houseNumber, :streetName,
:county, :town, :postcode, :lng, :lat, :contributor, :status)
        ");

        return $statement->execute([
            ':title' => $data['title'],
            ':category' => $data['category'],
            ':description' => $data['description'],
            ':houseNumber' => $data['houseNumber'],
            ':streetName' => $data['streetName'],
            ':county' => $data['county'],
            ':town' => $data['town'],
            ':postcode' => $data['postcode'],
            ':lng' => $data['lng'],
            ':lat' => $data['lat'],
            ':contributor' => $data['contributor'],
            ':status' => $data['status']
        ]);
    }

    /**
     *
     * Edit the status of a facility (used via AJAX for managers).
     */
    public function updateStatus($id, $status): bool {
        $statement = $this->_dbHandle->prepare("UPDATE ecoFacilities SET
status = :status WHERE id = :id");
        return $statement->execute([':status' => $status, ':id' => $id]);
    }

    /**
     * Delete a facility by its ID (used via AJAX for managers).
     */
    public function deleteFacility($id): bool {
        $statement = $this->_dbHandle->prepare("DELETE FROM ecoFacilities
WHERE id = :id");
        return $statement->execute([':id' => $id]);
    }

    /**
     * Get a limited set of facilities for pagination display.
     * Used on the homepage with 20-per-page.
     */
    public function fetchPaginatedFacilities(int $limit, int $offset):
array {
        $statement = $this->_dbHandle->prepare("SELECT * FROM ecoFacilities
ORDER BY id LIMIT :lim OFFSET :off");
        $statement->bindValue(':lim', $limit, PDO::PARAM_INT);
        $statement->bindValue(':off', $offset, PDO::PARAM_INT);
        $statement->execute();

        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    /**
     * Get all facility categories (used to populate dropdowns)
     */
    public function fetchCategories(): array {
        $statement = $this->_dbHandle->prepare("SELECT id, name FROM
ecoCategories ORDER BY name ASC");
        $statement->execute();
        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    /**
     * Get all facility status options (used in the dropdown when
adding/editing)
     */
    public function fetchStatuses(): array {
        $statement = $this->_dbHandle->prepare("SELECT id, statusComment
FROM ecoFacilityStatus ORDER BY id ASC");
        $statement->execute();
        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    /**
     * Count all facilities — used to calculate total number of pages
     */
    public function countFacilities(): int {
        return (int) $this->_dbHandle->query("SELECT COUNT(*) FROM
ecoFacilities")->fetchColumn();
    }
}
```

UserDataSet.php

```
<?php
require_once ('Database.php');   // Connects to the database class
require_once ('UserData.php');   // Loads the user model class

class UserDataSet {
    protected $_dbHandle, $_dbInstance;

    // Constructor: Sets up the database connection using Singleton pattern
    public function __construct() {
        $this->_dbInstance = Database::getInstance();                 //
Reuse the single DB connection
        $this->_dbHandle = $this->_dbInstance->getDbConnection();    //
Actual PDO connection handle
    }

    /**
     * Validates login credentials
     * If valid, saves user info to the session
     * Uses JOIN to fetch readable user type (Manager/User)
     */
    public function checkUserCredentials($username, $password) {
        // Query selects user and joins their type from ecoUsertypes
        $sql = "
        SELECT u.id, u.username, u.password, t.name AS userType
        FROM ecoUser u
        JOIN ecoUsertypes t ON u.userType = t.id
        WHERE u.username = :username
        ";

        // Prepare and bind the username securely
        $statement = $this->_dbHandle->prepare($sql);
        $statement->bindParam(':username', $username);
        $statement->execute();


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

        // Compare the submitted password with stored password
        if ($user && $password === $user['password']) {
            // Save important session info for use across the app
            $_SESSION['userID'] = $user['id'];
            $_SESSION['username'] = $user['username'];
            $_SESSION['userType'] = $user['userType'];
            return true;
        }

        // If login fails, return false
        return false;
    }
}
```

JavaScript

mapManager.js (Map Function)

```
class EcoMapManager {
    constructor(mapElementId = "ecoMap") {
        // Get the DOM element where the map should be displayed
        this.mapElement = document.getElementById(mapElementId);

        // Create a map to hold markers keyed by facility ID
        this.markerMap = new Map();

        // Exit early if the map element is not found
        if (!this.mapElement) return;

        // Initialize the map and load components
        this.initMap();         // Set up the base map
        this.loadTileLayer();   // Add OpenStreetMap tile layer
        this.centerOnUser();
        this.loadMarkers();     // Fetch and place facility markers
    }

    initMap() {
        // Initialize the Leaflet map centered over Manchester
        this.map = L.map(this.mapElement).setView([53.48, -2.24], 12);
    }

    loadTileLayer() {
        // Load OpenStreetMap tiles and add attribution
        L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
            attribution: '&copy; OpenStreetMap contributors'
        }).addTo(this.map);
    }

    centerOnUser() {
        // Check if the browser supports geolocation
        if (!navigator.geolocation) return;

        // Attempt to get the user's location
        navigator.geolocation.getCurrentPosition(
            (pos) => {
                const userLatLng = [pos.coords.latitude,
pos.coords.longitude];

                // Center the map on user's location
                this.map.setView(userLatLng, 14);

                // Add a marker to show the user's location
                L.marker(userLatLng, {
                    icon: L.icon({
                        iconUrl: "https://cdn-icons-
png.flaticon.com/512/684/684908.png",
                        iconSize: [32, 32],
                        iconAnchor: [16, 32],
                        popupAnchor: [0, -32]
                    })
                }).addTo(this.map).bindPopup("You are here").openPopup();
            },
            (err) => {
                // Handle cases where location permission is denied
                console.warn("User denied geolocation", err);
            }
        );
    }

    async loadMarkers() {
        try {
            // Fetch all facility data as JSON from the server
            const res = await fetch("mapController.php");
            const facilities = await res.json();

            // Loop through each facility and add a marker
            facilities.forEach(f => {
                const marker = L.marker([f.lat, f.lng]).addTo(this.map)
                    .bindPopup(`
                        <strong>${f.title}</strong><br>
                        ${f.description}<br>
                        ${f.town}<br>
                        <em>Status: ${f.status}</em>
                    `);

                // Store the marker with its ID for future reference
                this.markerMap.set(f.id, marker);
            });
        } catch (err) {
            // Log an error if markers fail to load
            console.error("Failed to load map markers:", err);
        }
    }
}
```

```
// Wait for the DOM to finish loading before running map logic
document.addEventListener("DOMContentLoaded", () => {
    // Create an instance of the map manager
    const ecoMap = new EcoMapManager();
    window.ecoMap = ecoMap; // Expose for access in other scripts

    // Add click event to each row that should focus the map
    document.querySelectorAll(".map-focus-row").forEach(row => {
        row.addEventListener("click", (e) => {
            // Don't trigger scroll if clicking inside a select or button
            if (e.target.closest('select') || e.target.closest('button'))
return;

            const id = row.dataset.id;
            const marker = ecoMap.markerMap.get(parseInt(id));

            if (marker) {
                // Zoom into the clicked facility marker
                ecoMap.map.setView(marker.getLatLng(), 16);
                marker.openPopup();

                // Scroll the page to the map view smoothly
                document.getElementById("ecoMap").scrollIntoView({
behavior: "smooth", block: "center" });
            }
        });
    });
});
```

xmlsearch.js (Search Function)

```
class SearchSuggestion {
    constructor(inputId, suggestionBoxId, outputId, searchURL) {
        this.input = document.getElementById(inputId);
        this.suggestionBox = document.getElementById(suggestionBoxId);
        this.output = document.getElementById(outputId);
        this.searchURL = searchURL;

        this.seen = new Set(); // To prevent duplicate suggestions
        this.init(); // Set up search event handling
    }

    /**
     * Bind the keyup event on the search input.
     */
    init() {
        this.input.addEventListener("keyup", () => this.onSearchInput());
    }

    /**
     * Handle input changes: clear if empty or send AJAX request.
     */
    onSearchInput() {
        const str = this.input.value.trim();

        if (str.length === 0) {
            this.clearSuggestions();      // Hide suggestions if input is
empty
            this.output.innerHTML = "";  // Clear result display
            return;
        }

        const xhr = new XMLHttpRequest();
        xhr.onreadystatechange = () => {
            if (xhr.readyState === 4 && xhr.status === 200) {
                this.renderResults(xhr.responseText); // Parse and display
XML results
            }
        };

        xhr.open("GET", `${this.searchURL}?q=${encodeURIComponent(str)}`,
true);
        xhr.send();
    }

    /**
     * Parse the XML and delegate rendering to both suggestions and cards.
     * @param {string} xmlString - The XML response as string.
     */
    renderResults(xmlString) {
        const parser = new DOMParser();
        const xmlDoc = parser.parseFromString(xmlString, "text/xml");
        const facilities = xmlDoc.getElementsByTagName("facility");

        this.renderSuggestions(facilities);
        this.renderCards(facilities);
    }

    /**
     * Render clickable search suggestion items under the input.
     * @param {HTMLCollection} facilities - XML elements representing
search results.
     */
    renderSuggestions(facilities) {
        this.suggestionBox.innerHTML = "";
        this.seen.clear();

        Array.from(facilities).forEach(f => {
            const title = f.getElementsByTagName("title")[0].textContent;
            const category =
f.getElementsByTagName("category")[0].textContent;
            const town = f.getElementsByTagName("town")?.[0]?.textContent
?? "";

            // Avoid duplicate suggestion values
            [title, category, town].forEach(value => {
                const lower = value.toLowerCase();
                if (value && !this.seen.has(lower)) {
                    const li = document.createElement("li");
                    li.className = "list-group-item list-group-item-
action";
                    li.innerHTML = `<a
href="?search=${encodeURIComponent(value)}" class="text-decoration-none d-
block">${value}</a>`;
                    this.suggestionBox.appendChild(li);
                    this.seen.add(lower);
                }
            });
        });
        // Hide the suggestion box if no results found
        this.suggestionBox.classList.toggle("d-none", facilities.length ===
0);
    }

    /**
     * Display full result cards beneath the search bar.
     * @param {HTMLCollection} facilities - XML facility elements.
     */
    renderCards(facilities) {
        this.output.innerHTML = "";

        Array.from(facilities).forEach(f => {
            const title = f.getElementsByTagName("title")[0].textContent;
            const category =
f.getElementsByTagName("category")[0].textContent;
            const town = f.getElementsByTagName("town")?.[0]?.textContent
?? "";

            const desc =
f.getElementsByTagName("description")[0].textContent;

            const col = document.createElement("div");
            col.className = "col-md-6 mb-3";
            col.innerHTML = `
        <a href="?search=${encodeURIComponent(title)}" class="text-
decoration-none">
          <div class="card shadow-sm">
            <div class="card-body">
              <h5 class="card-title">${title}</h5>
              <h6 class="card-subtitle text-muted mb-2">${category} —
${town}</h6>
              <p class="card-text">${desc}</p>
            </div>
          </div>
        </a>
      `;
            this.output.appendChild(col);
        });
    }

    /**
     * Clear and hide the suggestion dropdown box.
     */
    clearSuggestions() {
        this.suggestionBox.innerHTML = "";
        this.suggestionBox.classList.add("d-none");
    }
}
```

updateStatus.js (Update Function)

```
class StatusUpdater {
    constructor(dropdownClass, buttonClass, endpoint) {
        // Class name for all dropdowns used to select a new status
        this.dropdownClass = dropdownClass;

        // Class name for the update buttons
        this.buttonClass = buttonClass;

        // Server-side endpoint that processes the update
        this.endpoint = endpoint;

        // Setup event listener for update buttons
        this.attachListeners();
    }

    // Listens for clicks on any update-status buttons and sends the new
status
    attachListeners() {
        document.addEventListener("click", (e) => {
            // Only handle clicks on elements with the update button class
            if (!e.target.classList.contains(this.buttonClass)) return;

            // Get the facility ID from the clicked button
            const id = e.target.dataset.id;

            // Find the corresponding dropdown for that facility
            const select =
document.querySelector(`.${this.dropdownClass}[data-id='${id}']`);
            if (!select) return;

            // Grab the selected status from the dropdown
            const newStatus = select.value;

            // Send this status to the backend for update
            this.updateStatus(id, newStatus);
        });
    }

    // Handles sending the update request to the server
    updateStatus(id, status) {
        fetch(this.endpoint, {
            method: "POST",
            headers: {
                "Content-Type": "application/x-www-form-urlencoded"
            },
            // Send both facility ID and new status value
            body:
`id=${encodeURIComponent(id)}&status=${encodeURIComponent(status)}`
        })
            .then(res => res.text()) // Parse server response as plain text
            .then(msg => {
                alert(msg); // Show confirmation or failure message

                if (typeof ecoMap?.loadMarkers === "function") {
                    ecoMap.loadMarkers(); // Refresh markers to reflect new
status
                }
            })
            .catch(() => alert("Failed to update status.")); // Fallback
error message
    }
}

// Wait until the page is fully loaded before attaching event handlers
document.addEventListener("DOMContentLoaded", () => {
    // Create an instance of the class using your existing DOM structure
    new StatusUpdater("status-dropdown", "update-status-btn",
"updateStatus.php");
});
```

deleteFacility.js (Delete Button)

```
class DeleteFacility {
    constructor(triggerClass, endpoint, onSuccess = null) {
        this.triggerClass = triggerClass;
        this.endpoint = endpoint;
        this.onSuccess = onSuccess;

        this.init(); // Start listening for delete events
    }

    /**
     * Attach a global click event listener that responds to elements with
the delete trigger class.
     */
    init() {
        document.addEventListener("click", (e) => {
            // Check if the clicked element is a delete button
            if (e.target && e.target.classList.contains(this.triggerClass))
{
                const id = e.target.dataset.id; // Get the facility ID from
the data attribute

                // Ask the user for confirmation before deleting
                if (!confirm("Are you sure you want to delete this
facility?")) return;

                // Send the delete request
                this.sendDelete(id);
            }
        });
    }

    /**
     * Send a POST request to delete the facility by ID.
     * @param {string|number} id - The ID of the facility to delete.
     */
    sendDelete(id) {
        fetch(this.endpoint, {
            method: "POST",
            headers: {
                "Content-Type": "application/x-www-form-urlencoded"
            },
            body: "id=" + encodeURIComponent(id) // Send the facility ID in
the request body
        })
            .then(res => res.text()) // Parse response as plain text
            .then(msg => {
                alert(msg); // Show the result to the user

                // Call the success callback if provided
                if (typeof this.onSuccess === "function") this.onSuccess();
            })
            .catch(() => alert("Failed to delete")); // Handle any errors
during the request
    }
}
```

submitFacility.js (Submit for Add Facility)

```
class FormHandler {
    constructor(formId, endpoint, messageId) {
        this.form = document.getElementById(formId);        // Get the form
element by ID
        this.messageBox = document.getElementById(messageId); // Get the
message box element
        this.endpoint = endpoint;                            // URL to submit
form data to

        // Only initialize if form element is found
        if (this.form) {
            this.init();
        }
    }

    /**
     * Bind the submit event to the form and prevent default browser
behavior.
     */
    init() {
        this.form.addEventListener("submit", (e) => {
            e.preventDefault(); // Prevent the page from reloading
            this.submitForm();  // Trigger AJAX form submission
        });
    }

    /**
     * Gather form data and send it via XMLHttpRequest to the server.
     */
    submitForm() {
        const formData = new FormData(this.form);      // Collect form
fields and values
        const xhr = new XMLHttpRequest();              // Create a new AJAX
request
        xhr.open("POST", this.endpoint, true);         // Prepare POST
request to the endpoint

        xhr.onload = () => {
            const success = xhr.status === 200;        // Check if the
response is OK
            const className = success ? "success" : "danger"; // Bootstrap
alert class
            this.messageBox.innerHTML = `<div class="alert alert-
${className}">${xhr.responseText}</div>`; // Show response

            if (success) this.form.reset();            // Reset form if it
was successful
        };

        xhr.send(formData); // Send the collected form data
    }
}
```

Views and Controller

header.phtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Basic HTML setup for responsive design -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge"> <!-- Ensures IE
uses latest engine -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Mobile responsive -->

    <!-- Bootstrap CSS and Custom CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.c
ss" rel="stylesheet" crossorigin="anonymous">
    <link href="/Views/css/my-style.css" rel="stylesheet"> <!-- Custom site
styles -->

    <!-- Dynamic page title based on the $view object -->
    <title>ecoBuddy - <?php echo $view->pageTitle; ?></title>
</head>

<!-- Uses Bootstrap flexbox layout to ensure footer sticks and content
grows -->
<body class="d-flex flex-column h-100">

<!-- NAVIGATION HEADER -->
<header>
    <!-- Responsive, fixed top Bootstrap navbar -->
    <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
        <div class="container-fluid">
            <!-- Brand/logo link -->
            <a class="navbar-brand" href="#">ecoBuddy</a>

            <!-- Collapsible navbar toggle for mobile view -->
            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarCollapse" aria-
controls="navbarCollapse" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>

            <!-- Navbar items -->
            <div class="collapse navbar-collapse" id="navbarCollapse">
                <ul class="navbar-nav me-auto mb-2 mb-md-0">
                    <!-- Home link always visible -->
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page"
href="/index.php">Home</a>
                    </li>

                    <!-- Show Add Facility only for logged-in Manager users
-->
                    <?php if (isset($_SESSION['userType']) &&
$_SESSION['userType'] === 'Manager'): ?>
                        <li class="nav-item">
                            <a class="nav-link active" aria-current="page"
href="/addFacility.php">Add Facility</a>
```

```
                        </li>
                    <?php endif; ?>
                </ul>
            </div>
        </div>
    </nav>
</header>

<!-- MAIN PAGE CONTENT -->
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <!-- Site branding & title area -->
            <div id="title" class="col-xs-12">
                <img src="/images/new_uos_logo.jpg" alt="Salford
University" />
                <h1><?php echo $view->pageTitle ?> </h1>
            </div>
        </div>

        <!-- LOGIN FORM if not logged in -->
        <?php if (!isset($_SESSION['username'])): ?>
            <form method="post" action="loginController.php" class="d-flex
gap-2 align-items-center">
                <input type="text" name="username" placeholder="Username"
required class="form-control form-control-sm">
                <input type="password" name="password"
placeholder="Password" required class="form-control form-control-sm">
                <button type="submit" name="loginbutton" class="btn btn-
primary btn-sm">Login</button>
            </form>

            <!-- LOGOUT + user info if logged in -->
        <?php else: ?>
            <span class="me-2">
                Welcome, <?= htmlspecialchars($_SESSION['username']) ?>
(<?= $_SESSION['userType'] ?>)
            </span>
            <form method="post" action="loginController.php" class="d-
inline">
                <button type="submit" name="logoutbutton" class="btn btn-
danger btn-sm">Logout</button>
            </form>
        <?php endif; ?>

        <!-- ERROR ALERT if login failed -->
        <?php if (isset($_GET['error']) && $_GET['error'] == 1): ?>
            <div class="alert alert-danger alert-dismissible fade show my-
2" role="alert">
                Invalid username or password.
                <button type="button" class="btn-close" data-bs-
dismiss="alert"></button>
            </div>
        <?php endif; ?>
```

addFacility.phtml

```
<?php require_once('template/header.phtml'); ?> <!-- Includes the main site
header and nav -->

<!-- Page Container -->
<div class="container mt-4">
    <h2>Add New Facility</h2> <!-- Page Heading -->

    <!-- Facility Submission Form -->
    <form id="addFacilityForm" class="col-lg-8">

        <!-- Title Input -->
        <div>
            <label for="title" class="form-label">Title</label>
            <input type="text" id="title" name="title" class="form-control"
required>
        </div>

        <!-- Category Dropdown  -->
        <div>
            <label for="category" class="form-label">Category</label>
            <select name="category" class="form-select" required>
                <option disabled selected>-- Select Category --</option>
                <?php foreach ($view->categories as $cat): ?>
                    <option value="<?= htmlspecialchars($cat['id']) ?>">
                        <?= htmlspecialchars($cat['name']) ?>
                    </option>
                <?php endforeach; ?>
            </select>
        </div>

        <!-- Description Input -->
        <div>
            <label for="description" class="form-label">Description</label>
            <input type="text" id="description" name="description"
class="form-control" required>
        </div>

        <!-- House Number -->
        <div>
            <label for="houseNumber" class="form-label">House
Number</label>
            <input type="number" id="houseNumber" name="houseNumber"
class="form-control" required>
        </div>

        <!-- Street Name -->
        <div>
            <label for="streetName" class="form-label">Street Name</label>
            <input type="text" id="streetName" name="streetName"
class="form-control" required>
        </div>

        <!-- County Input (letters only) -->
        <div>
            <label for="county" class="form-label">County</label>
            <input name="county" type="text" pattern="[A-Za-z\s]+"
title="Letters and spaces only" class="form-control" required>
        </div>
        <!-- Town Input (letters only) -->
        <div>
            <label for="town" class="form-label">Town</label>
            <input name="town" type="text" pattern="[A-Za-z\s]+"
title="Letters and spaces only" class="form-control" required>
        </div>

        <!-- Postcode Input  -->
        <div>
            <label for="postcode" class="form-label">Postcode</label>
            <input name="postcode" type="text" id="postcode" class="form-
control" required>
        </div>

        <!-- Longitude Input -->
        <div>
            <label for="lng" class="form-label">Longitude</label>
            <input name="lng" type="number" id="lng" class="form-control"
step="0.000001" min="-180" max="180" required>
        </div>

        <!-- Latitude Input -->
        <div>
            <label for="lat" class="form-label">Latitude</label>
            <input name="lat" type="number" id="lat" class="form-control"
step="0.000001" min="-90" max="90" required>
        </div>

        <!-- Status Dropdown -->
        <div>
            <label for="status" class="form-label">Status</label>
            <select name="status" class="form-select" required>
                <option disabled selected value="">-- Select Status --
</option>
                <?php foreach ($view->statuses as $status): ?>
                    <option value="<?= htmlspecialchars($status['id']) ?>">
                        <?= htmlspecialchars($status['statusComment']) ?>
                    </option>
                <?php endforeach; ?>
            </select>
        </div>

        <!-- Submit Button -->
        <button type="submit" class="btn btn-primary mt-3">Add
Facility</button>

        <!-- Area to show success/failure message -->
        <div id="addFacilityMsg" class="mt-3"></div>

        <!—Submit Script -->
        <script src="../JavaScripts/submitFacility.js"></script>
        <script>
            document.addEventListener("DOMContentLoaded", () => {
                new FormHandler("addFacilityForm", "addFacility.php",
"addFacilityMsg");
            });
        </script>
    </form>
</div>
```

```
<?php require_once('template/footer.phtml'); ?> <!-- Include footer
template -->
```

addFacility.php

```
<?php
// Start the session if it's not already active (for login and user info)
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

// Load the database model for interacting with facility records
require_once('Models/FacilityDataSet.php');

// Load the login logic — handles session-based login checks
require_once('loginController.php');

// Create the database handler for facilities
$model = new FacilityDataSet();

// Prepare the $view object to pass data to the form view
$view = new stdClass();
$view->pageTitle = 'Add Facility';


// --------------------
// Handle form submission
// --------------------
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // These are the minimum required fields from the form
    $required = ['title', 'description', 'town', 'postcode', 'lat', 'lng',
'status', 'category'];

    // Loop over all required fields to check if any are missing
    foreach ($required as $field) {
        if (empty($_POST[$field])) {
            // If any field is missing, stop and return an error
            http_response_code(400);
            echo "Missing field: $field";
            exit;
        }
    }

    // Prepare data array to be inserted into the database
    $data = [
        'title' => $_POST['title'],
        'category' => $_POST['category'],
        'description' => $_POST['description'],
        'houseNumber' => $_POST['houseNumber'] ?? '',
        'streetName' => $_POST['streetName'] ?? '',
        'county' => $_POST['county'] ?? '',
        'town' => $_POST['town'],
        'postcode' => $_POST['postcode'],
        'lng' => $_POST['lng'],
        'lat' => $_POST['lat'],
        'contributor' => $_SESSION['userID'] ?? 1,
        'status' => $_POST['status']
    ];

    if ($model->insertFacility($data)) {
        echo "Facility added successfully!";
    } else {
        http_response_code(500);
        echo "Failed to add facility.";
    }

    exit; // Prevent the form from showing again after a POST
}

// Load category and status options from DB for the form dropdowns
$view->categories = $model->fetchCategories();
$view->statuses = $model->fetchStatuses();

// Show the HTML form
require('Views/addFacility.phtml');
```

index.phtml

```
<?php require('template/header.phtml') ?> <!-- Load header and navigation -
->

<!-- Main container  -->
<div class="container-fluid px-5 mx-auto" style="max-width: 1800px;">

    <!-- If search, show the reset and current search term -->
    <?php if ($showReset): ?>
        <div class="d-flex justify-content-between align-items-center mb-
3">
            <h5 class="mb-0">Showing results for: <strong><?=
htmlspecialchars($searchTerm) ?></strong></h5>
            <a href="/index.php" class="btn btn-outline-secondary">Reset
Search</a>
        </div>
    <?php endif; ?>

    <!-- Live Search Bar -->
    <h3>Search</h3>
    <div class="position-relative mb-3" style="z-index: 999;">
        <input type="text" id="xmlSearch" class="form-control"
placeholder="Search facilities...">
        <ul id="searchSuggestions" class="list-group position-absolute w-
100 d-none"></ul> <!-- Suggestions dropdown -->
    </div>

    <!-- Dynamic search results rendered here -->
    <div class="row" id="xmlOutput"></div>

    <!-- Scripts: Live Search (XML), Delete Handler, Status Updater-->
    <script src="/JavaScripts/xmlsearch.js"></script>
    <script>
        document.addEventListener("DOMContentLoaded", function () {
            new SearchSuggestion("xmlSearch", "searchSuggestions",
"xmlOutput", "searchController.php");
        });
    </script>

    <script src="/JavaScripts/deleteFacility.js"></script>
    <script>
        document.addEventListener("DOMContentLoaded", () => {
            new DeleteFacility("delete-btn", "deleteFacility.php", () => {
                if (typeof loadFacilityTable === "function")
loadFacilityTable();
                if (typeof loadMapMarkers === "function") loadMapMarkers();
            });
        });
    </script>

    <script src="/JavaScripts/updateStatus.js"></script>

    <!-- Leaflet Map Display -->
    <h3>ecoFacility Map View</h3>
    <div id="ecoMap" style="height: 500px;"></div>
    <script src="/JavaScripts/mapManager.js"></script>

    <!-- Load Leaflet.js CSS & JS for map support -->
    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css">
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>

    <!-- Welcome message and debug information -->
    <h2>Welcome to ecoBuddy </h2>
    <div class="row">
        <div class="col-5">
            <p><?php echo $view->dbMessage; ?></p>
        </div>
        <div class="col-7">
            <p>Current script <?php echo $_SERVER["PHP_SELF"]; ?></p>
        </div>
    </div>

    <!-- Facility Table Display -->
    <div class="table-responsive">
        <table class="table table-striped table-hover">
            <thead class="table-dark">
            <tr>
                <th>Title</th>
                <th>Category</th>
                <th>Description</th>
                <th>House Number</th>
                <th>Street Name</th>
                <th>County</th>
                <th>Town</th>
                <th>Post Code</th>
                <th>Longitude</th>
                <th>Latitude</th>
                <th>Contributor</th>
                <th>Status</th>
                <?php if (isset($_SESSION['userType']) &&
$_SESSION['userType'] === 'Manager'): ?>
                    <th>Update Status</th>
                <?php endif; ?>
                <?php if (isset($_SESSION['userType']) &&
$_SESSION['userType'] === 'Manager'): ?>
                    <th class="text-center" style="width:
90px;">Delete</th>
                <?php endif; ?>
            </tr>
            </thead>
            <tbody>

            <!-- Loop through facility records and display each one -->
            <?php foreach ($facilities as $FacilityData): ?>
                <tr class="map-focus-row" data-id="<?= $FacilityData['id']
?>">
                    <td><?= htmlspecialchars($FacilityData['title'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['category'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['description'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['houseNumber'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['streetName'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['county'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['town']) ?></td>
                    <td><?= htmlspecialchars($FacilityData['postcode'])
?></td>
                    <td><?= htmlspecialchars($FacilityData['lng']) ?></td>
                    <td><?= htmlspecialchars($FacilityData['lat']) ?></td>
                    <td><?=
htmlspecialchars($userMap[$FacilityData['contributor']] ?? 'Unknown')
?></td>
                    <td><?=
htmlspecialchars($statusMap[$FacilityData['status']] ?? 'Unknown') ?></td>
                    <?php if (isset($_SESSION['userType']) &&
$_SESSION['userType'] === 'Manager'): ?>
                        <!-- Edit button only for Manager users -->
                        <td class="text-center">
                            <select class="form-select form-select-sm
status-dropdown" data-id="<?= $FacilityData['id'] ?>">
                                <?php foreach ($statusMap as $sid =>
$label): ?>
                                    <option value="<?= $sid ?>" <?= $sid ==
$FacilityData['status'] ? 'selected' : '' ?>>
                                        <?= htmlspecialchars($label) ?>
                                    </option>
                                <?php endforeach; ?>
                            </select>
                            <button class="btn btn-sm btn-success mt-1
update-status-btn" data-id="<?= $FacilityData['id'] ?>">
                                Update
                            </button>
                        </td>
                    <?php endif; ?>
                    <!-- Delete button only for Manager users -->
                    <?php if (isset($_SESSION['userType']) &&
$_SESSION['userType'] === 'Manager'): ?>
                        <td class="text-center">
                            <button class="btn btn-sm btn-danger delete-
btn" data-id="<?= $FacilityData['id'] ?>">
                                Delete
                            </button>
                        </td>
                    <?php endif; ?>
                </tr>
            <?php endforeach; ?>
            </tbody>
        </table>
    </div>
    <!-- Pagination Bar -->
    <nav class="mt-4">
        <ul class="pagination justify-content-center">
            <?php if ($page > 1): ?>
                <li class="page-item"><a class="page-link" href="?page=<?=
$page - 1 ?>">Previous</a></li>
            <?php endif; ?>
            <?php for ($p = 1; $p <= $totalPages; $p++): ?>
                <li class="page-item <?= $p == $page ? 'active' : '' ?>">
                    <a class="page-link" href="?page=<?= $p ?>"><?= $p
?></a>
                </li>
            <?php endfor; ?>
            <?php if ($page < $totalPages): ?>
                <li class="page-item"><a class="page-link" href="?page=<?=
$page + 1 ?>">Next</a></li>
            <?php endif; ?>
        </ul>
    </nav>

    <!-- Load footer -->
    <?php require('template/footer.phtml') ?>
</div>
```

index.php

```
<?php
// Start session if not already started
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

// Load model class to access facilities
require_once('Models/FacilityDataSet.php');

// Create view object to hold variables used in the template
$view = new stdClass();
$view->pageTitle = 'Homepage';
$view->loginError = false;

// Load login logic to check session & credentials
require_once('loginController.php');

// Instantiate Facility dataset object to run queries
$ecoFacilities = new FacilityDataSet();

// Handle pagination values
$page = isset($_GET['page']) ? max(1, (int)$_GET['page']) : 1;
$limit = 20;
$offset = ($page - 1) * $limit;

// If a search term is provided, override paginated fetch
$searchTerm = $_GET['search'] ?? null;
$facilities = [];
$showReset = false;

if ($searchTerm) {
    $facilities = $ecoFacilities->searchFacilities($searchTerm);
    $total = count($facilities); // simple count of search result
    $showReset = true;
} else {
    $facilities = $ecoFacilities->fetchPaginatedFacilities($limit,
$offset);
    $total = $ecoFacilities->countFacilities();
}

$totalPages = ceil($total / $limit);

// Prepare mapping of user IDs to usernames
$userMap = [];
foreach ($ecoFacilities->getAllUsers() as $u) {
    $userMap[$u['id']] = $u['username'];
}

// Prepare mapping of status IDs to status strings
$statusMap = $ecoFacilities->getAllStatuses();

$view->FacilityDataSet = $ecoFacilities->fetchAllFacilities();

// Provide result count message
if (count($view->FacilityDataSet) == 0) {
    $view->dbMessage = "No results";
} else {
    $view->dbMessage = count($view->FacilityDataSet) . " result(s)";
}

// Load main view template
require_once('Views/index.phtml');
```

deleteFacility.php

```
<?php
// Start a session so we can access the logged-in user's role
session_start();

// Include the class that handles database actions for facilities
require_once('Models/FacilityDataSet.php');


// Only managers should be able to delete data
if (isset($_SESSION['userType']) && $_SESSION['userType'] === 'User') {
    // Regular users are not authorized to delete records
    http_response_code(403);
    echo "Unauthorized";
    exit;
}

// Check if a facility ID was submitted
$id = $_POST['id'] ?? null;

if (!$id) {
    // If no ID is provided, return a client error
    http_response_code(400); // Bad Request
    echo "No facility ID provided";
    exit;
}

$dataset = new FacilityDataSet();

// Run the deletion query and return the result
```

```
if ($dataset->deleteFacility($id)) {
    echo "Facility deleted successfully!";
} else {
    // If something goes wrong in the database
    http_response_code(500); // Internal Server Error
    echo "Deletion failed.";
}
```

loginController.php

```
<?php
// Start a session if one hasn't already been started
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

// Load the UserDataSet class which handles login-related database
operations
require_once('Models/UserDataSet.php');

// Create a user model instance to check login credentials
$users = new UserDataSet();

// Handle Login Request
if (isset($_POST["loginbutton"])) {
    // Get the submitted username and password,
    $username = $_POST["username"] ?? '';
    $password = $_POST["password"] ?? '';

    // Use the model to validate the credentials
    $isValidLogin = $users->checkUserCredentials($username, $password);

    if ($isValidLogin) {
        // If credentials are correct, go to homepage
        header("Location: index.php");
        exit;
    } else {
        // If credentials are wrong, reload homepage with an error message
        header("Location: index.php?error=1");
        exit;
    }
}

// Handle Logout Request
if (isset($_POST["logoutbutton"])) {
    // Destroy all session data (logs user out)
    session_destroy();

    // Send user back to homepage
    header("Location: index.php");
    exit;
}
```

mapController.php

```php
<?php
// Load the class responsible for database access to ecoFacilities
require_once("Models/FacilityDataSet.php");

// Tell the browser the response is JSON
header('Content-Type: application/json');

// Instantiate the model to interact with the database
$model = new FacilityDataSet();

// Retrieve all facilities including status joined via SQL
$facilities = $model->getAllFacilities();

// Return the dataset as a JSON-encoded string
echo json_encode($facilities);
```

searchController.php

```php
<?php
// Include the database interaction class for facilities
require_once('Models/FacilityDataSet.php');

// This script returns an XML response
header("Content-Type: text/xml");

// Read the user's search input from the URL trim it and make it lowercase
for easier comparison
$searchTerm = strtolower(trim($_GET['q'] ?? ""));

// Start an empty <facilities> XML document to build our response
$xmlOutput = new SimpleXMLElement("<facilities></facilities>");

// Only run a search if the user typed something
if ($searchTerm !== "") {
    // Create the model to access database methods
    $model = new FacilityDataSet();

    // Call the search method which looks for matches across several fields
(title, category, town, etc.)
    $results = $model->searchFacilities($searchTerm);

    // Loop through all matching results and convert each one into a
<facility> block inside our XML
    foreach ($results as $facilityData) {
        $facility = $xmlOutput->addChild("facility");

        // Add each property as a child element inside <facility>
        $facility->addChild("id", $facilityData['id']);
        $facility->addChild("title",
htmlspecialchars($facilityData['title']));
        $facility->addChild("category",
htmlspecialchars($facilityData['category']));
        $facility->addChild("town",
htmlspecialchars($facilityData['town']));
        $facility->addChild("description",
htmlspecialchars($facilityData['description']));
    }
}
```

```
// Convert the XML object into a string and send it to the browser
echo $xmlOutput->asXML();
```

updateStatus.php

```php
<?php
session_start(); // Start or resume the session to check user role

require_once('Models/FacilityDataSet.php'); // Include access to database
methods

// Only allow access if logged-in user is a Manager
if (!isset($_SESSION['userType']) || $_SESSION['userType'] !== 'Manager') {
    http_response_code(403);
    echo "Unauthorized";      // Message for unauthorized users
    exit;
}

// Get the facility ID and new status value from the AJAX POST request
$id = $_POST['id'] ?? null;
$status = $_POST['status'] ?? null;

// Make sure both values are present before proceeding
if (!$id || !$status) {
    http_response_code(400); // Bad request
    echo "Missing data.";
    exit;
}

// Use the model to update the facility's status in the database
$model = new FacilityDataSet();
if ($model->updateStatus($id, $status)) {
    echo "Status updated successfully!";
} else {
    http_response_code(500); // Server error if update fails
    echo "Update failed.";
}
```

CSS

my-style.css

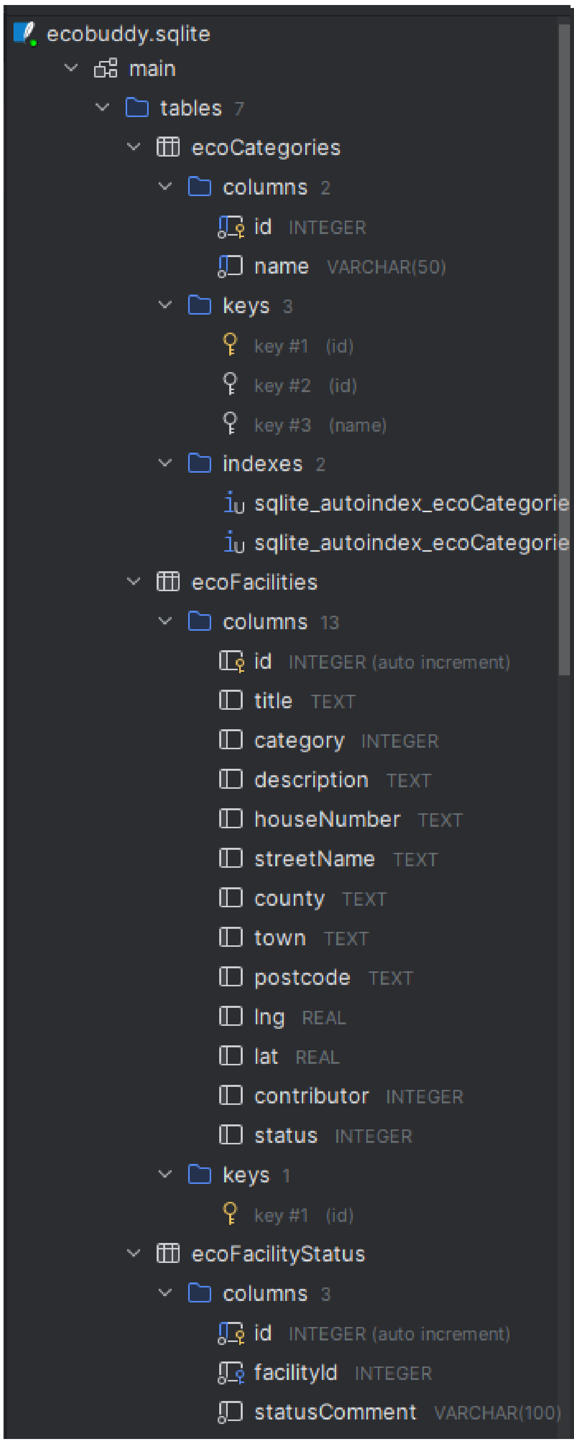
```css
/* =============================
   GLOBAL LAYOUT & BODY
   ============================= */
body {
    background-color: #f4f7fa; /* Light background for the whole site */
    font-family: 'Segoe UI', sans-serif;
    margin: 0;
    padding-top: 70px; /* Offsets content for fixed headers */
}

.container {
    max-width: 1800px;
    margin: auto;
    padding: 2rem;
}
```

```css
.container-fluid {
    padding: 2rem;
}


/* ==============================
   NAVBAR & HEADER STYLING
   ============================== */
header, .navbar {
    background-color: #0d6efd; /* Bootstrap primary blue */
    color: white;
    padding: 0.25rem 1rem;
    margin-bottom: 1.5rem;
    min-height: 50px;
}

.navbar a,
.navbar-brand {
    color: white;
    font-weight: bold;
    text-decoration: none;
}

.navbar a:hover {
    color: #f4f7fa;
}


/* ==============================
   CARDS - USED FOR SEARCH RESULTS
   ============================== */
.card {
    background-color: #ffffff;
    border: none;
    border-radius: 10px;
    box-shadow: 0 0.5rem 1rem rgba(0,0,0,0.08);
    transition: transform 0.2s ease;
}

.card:hover {
    transform: scale(1.01);
    box-shadow: 0 0.75rem 1.5rem rgba(0,0,0,0.1);
}


/* ==============================
   TABLE STYLING - TEAL THEME
   ============================== */
.table {
    background-color: #ffffff;
    border-radius: 8px;
    overflow: hidden;
    border-collapse: collapse;
    width: 100%;
}

.table thead {
    background-color: #006d77; /* Teal header */
    color: #ffffff;
}
```

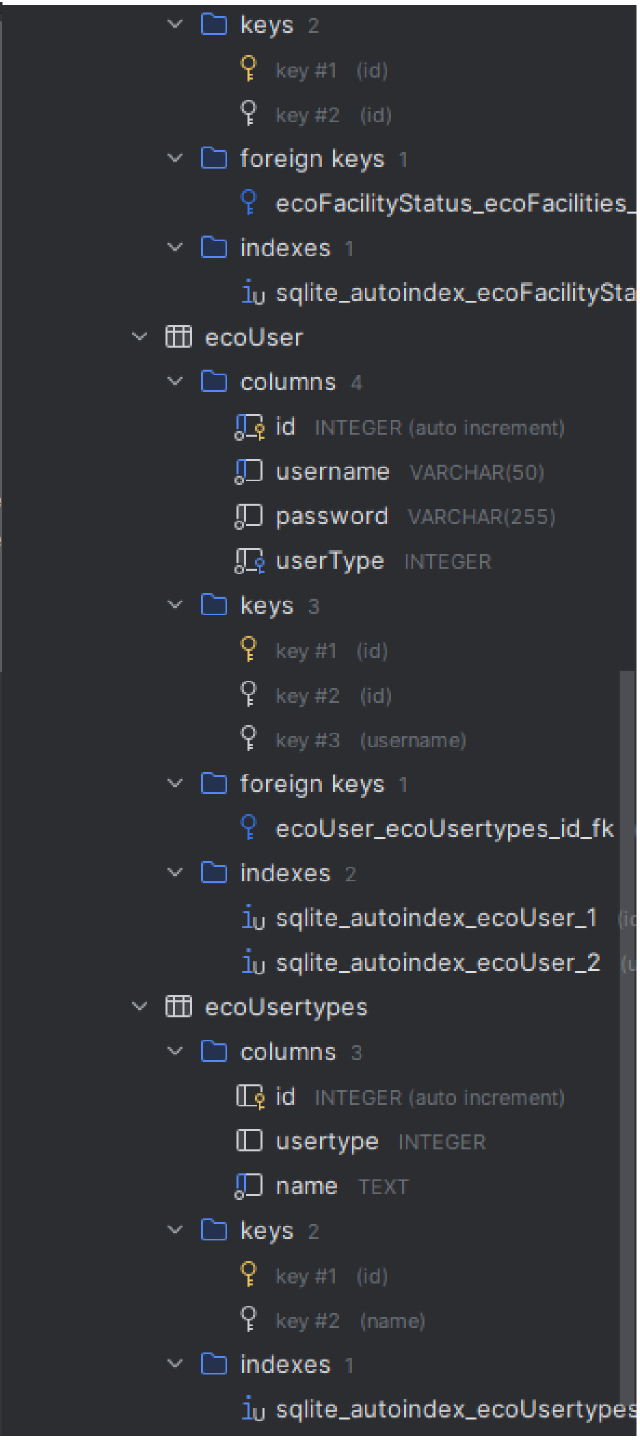
```css
.table thead th {
    padding: 0.75rem;
    font-weight: 600;
    border-bottom: 1px solid #dee2e6;
}

.table tbody td {
    padding: 0.75rem;
    vertical-align: middle;
    border-bottom: 1px solid #e3e8ea;
}

.table-striped tbody tr:nth-child(odd) {
    background-color: #e0f7f1;
}

.table-hover tbody tr:hover {
    background-color: #c0ebdf;
    cursor: pointer;
}

.table tbody td:first-child {
    font-weight: 500;
    color: #00524f;
}

.table thead th:first-child {
    border-top-left-radius: 8px;
}

.table thead th:last-child {
    border-top-right-radius: 8px;
}

.table-responsive {
    margin-bottom: 1rem;
    border-radius: 8px;
    overflow-x: auto;
    -webkit-overflow-scrolling: touch;
}


/* =============================
   BUTTONS
   ============================= */
.btn {
    border-radius: 30px;
    font-weight: 500;
}

.btn-outline-success:hover {
    background-color: #198754;
    color: white;
}

.btn-outline-danger:hover {
    background-color: #dc3545;
    color: white;
}
```

```css
/* =============================
   ALERT MESSAGES
   ============================= */
.alert {
    margin-top: 1rem;
}


/* =============================
   LOGIN FORM
   ============================= */
form.d-flex.gap-2 input {
    min-width: 160px;
}


/* =============================
   TYPOGRAPHY
   ============================= */
h1, h2, h3, a {
    color: #003366;
}


/* =============================
   MAP DISPLAY (LEAFLET)
   ============================= */
#map {
    height: 500px;
    width: 100%;
    border-radius: 10px;
    margin-top: 2rem;
}


/* =============================
   HOVERABLE TABLE ROWS (LINK-LIKE)
   ============================= */
.map-focus-row {
    cursor: pointer;
    transition: background-color 0.2s, color 0.2s;
}

.map-focus-row:hover {
    background-color: #f0f8ff; /* Light blue background */
    color: #0d6efd;            /* Bootstrap blue text */
    text-decoration: underline;
}


/* =============================
   SEARCH SUGGESTION DROPDOWN
   ============================= */
#searchSuggestions {
    max-height: 250px;
    overflow-y: auto;
    border: 1px solid #ccc;
    border-top: none;
    background-color: white;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    font-size: 16px;
}

#searchSuggestions li a {
    padding: 10px 15px;
    color: #333;
}

#searchSuggestions li a:hover {
    background-color: #f1f1f1;
    color: #0d6efd;
}

/* ==============================
   Status Update DROPDOWN
   ============================== */

.status-dropdown {
    width: 100%;
    font-size: 0.875rem;
    padding: 0.25rem 0.5rem;
}

.update-status-btn {
    width: 100%;
    padding: 0.3rem;
    font-size: 0.75rem;
    background-color: #0dcaf0;
    border: none;
    color: white;
    border-radius: 0.25rem;
    transition: background-color 0.3s;
}

.update-status-btn:hover {
    background-color: #0bb5d4;
}
```

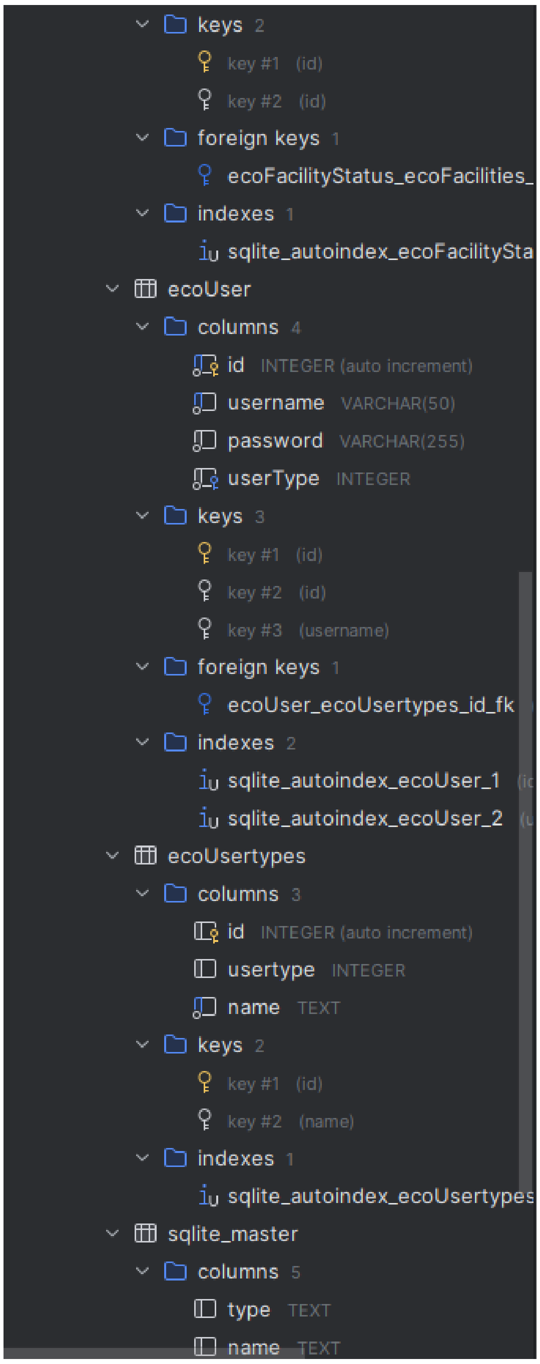
Database Screenshots



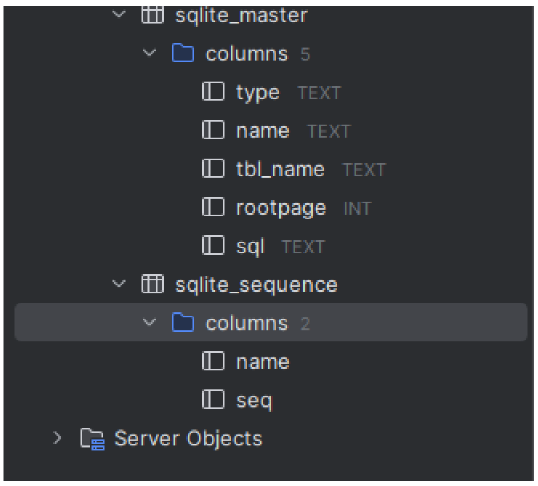
```
ecobuddy.sqlite
  main
    tables 7
      ecoCategories
        columns 2
          id INTEGER
          name VARCHAR(50)
        keys 3
          key #1 (id)
          key #2 (id)
          key #3 (name)
        indexes 2
          sqlite_autoindex_ecoCategorie
          sqlite_autoindex_ecoCategorie
      ecoFacilities
        columns 13
          id INTEGER (auto increment)
          title TEXT
          category INTEGER
          description TEXT
          houseNumber TEXT
          streetName TEXT
          county TEXT
          town TEXT
          postcode TEXT
          lng REAL
          lat REAL
          contributor INTEGER
          status INTEGER
        keys 1
          key #1 (id)
      ecoFacilityStatus
        columns 3
          id INTEGER (auto increment)
          facilityId INTEGER
          statusComment VARCHAR(100)
```



```
        keys 2
          key #1 (id)
          key #2 (id)
        foreign keys 1
          ecoFacilityStatus_ecoFacilities_
        indexes 1
          sqlite_autoindex_ecoFacilitySta
      ecoUser
        columns 4
          id INTEGER (auto increment)
          username VARCHAR(50)
          password VARCHAR(255)
          userType INTEGER
        keys 3
          key #1 (id)
          key #2 (id)
          key #3 (username)
        foreign keys 1
          ecoUser_ecoUsertypes_id_fk
        indexes 2
          sqlite_autoindex_ecoUser_1
          sqlite_autoindex_ecoUser_2
      ecoUsertypes
        columns 3
          id INTEGER (auto increment)
          usertype INTEGER
          name TEXT
        keys 2
          key #1 (id)
          key #2 (name)
        indexes 1
          sqlite_autoindex_ecoUsertypes
```

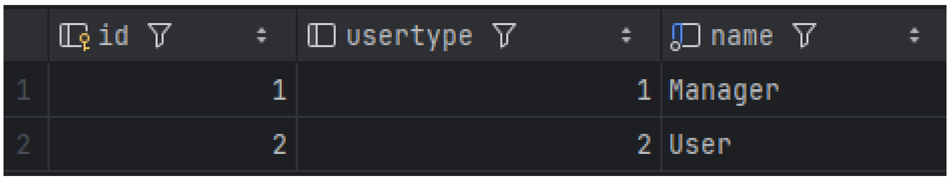


```
    keys 2
      key #1 (id)
      key #2 (id)
    foreign keys 1
      ecoFacilityStatus_ecoFacilities_
    indexes 1
      sqlite_autoindex_ecoFacilitySta
  ecoUser
    columns 4
      id INTEGER (auto increment)
      username VARCHAR(50)
      password VARCHAR(255)
      userType INTEGER
    keys 3
      key #1 (id)
      key #2 (id)
      key #3 (username)
    foreign keys 1
      ecoUser_ecoUsertypes_id_fk
    indexes 2
      sqlite_autoindex_ecoUser_1
      sqlite_autoindex_ecoUser_2
  ecoUsertypes
    columns 3
      id INTEGER (auto increment)
      usertype INTEGER
      name TEXT
    keys 2
      key #1 (id)
      key #2 (name)
    indexes 1
      sqlite_autoindex_ecoUsertypes
  sqlite_master
    columns 5
      type TEXT
      name TEXT
```



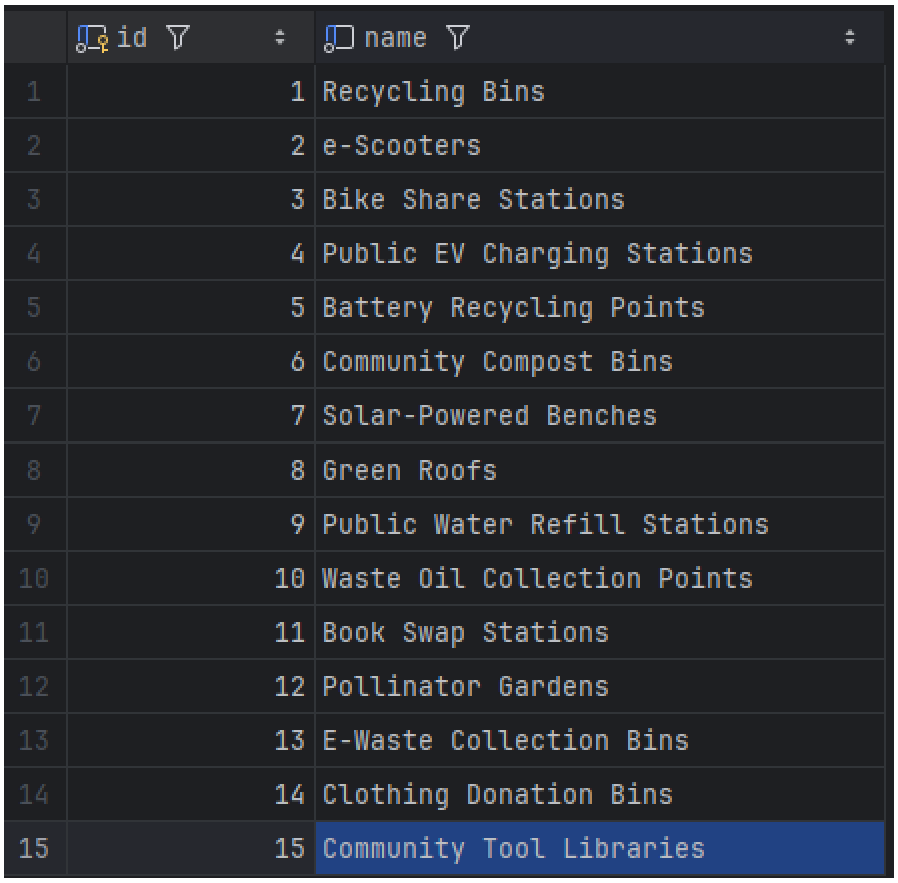
```
  sqlite_master
    columns 5
      type TEXT
      name TEXT
      tbl_name TEXT
      rootpage INT
      sql TEXT
  sqlite_sequence
    columns 2
      name
      seq
Server Objects
```

Database Screenshot for ecoUsertypes



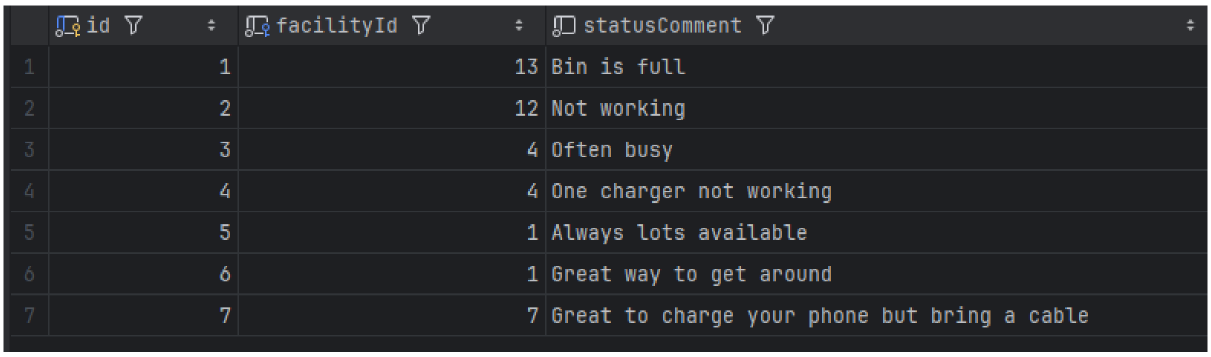
	id	usertype	name
1	1	1	Manager
2	2	2	User

Database Screenshot for ecoCategories



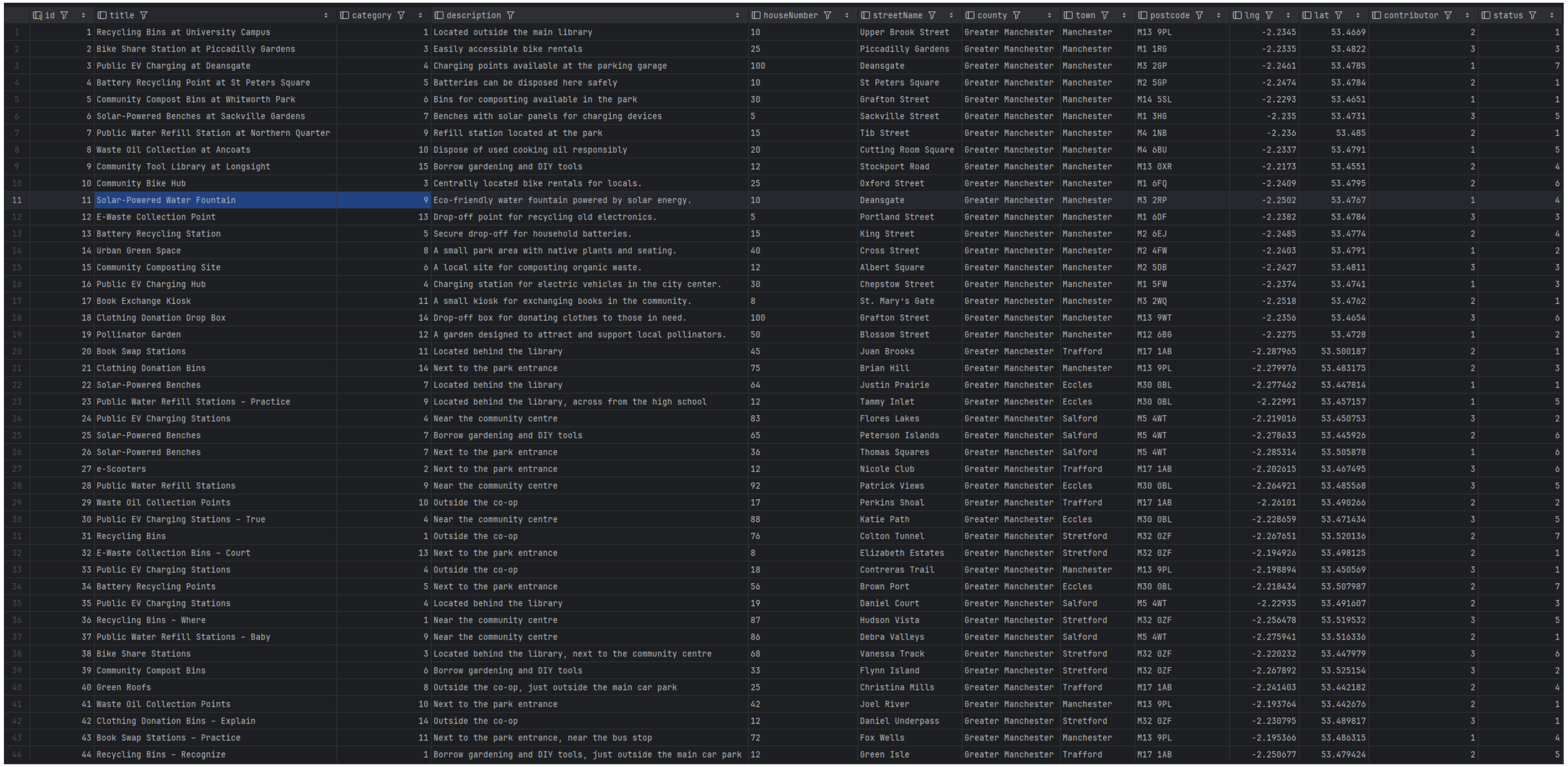
	id	name
1	1	Recycling Bins
2	2	e-Scooters
3	3	Bike Share Stations
4	4	Public EV Charging Stations
5	5	Battery Recycling Points
6	6	Community Compost Bins
7	7	Solar-Powered Benches
8	8	Green Roofs
9	9	Public Water Refill Stations
10	10	Waste Oil Collection Points
11	11	Book Swap Stations
12	12	Pollinator Gardens
13	13	E-Waste Collection Bins
14	14	Clothing Donation Bins
15	15	Community Tool Libraries

Database Screenshot for ecoFacilityStatus



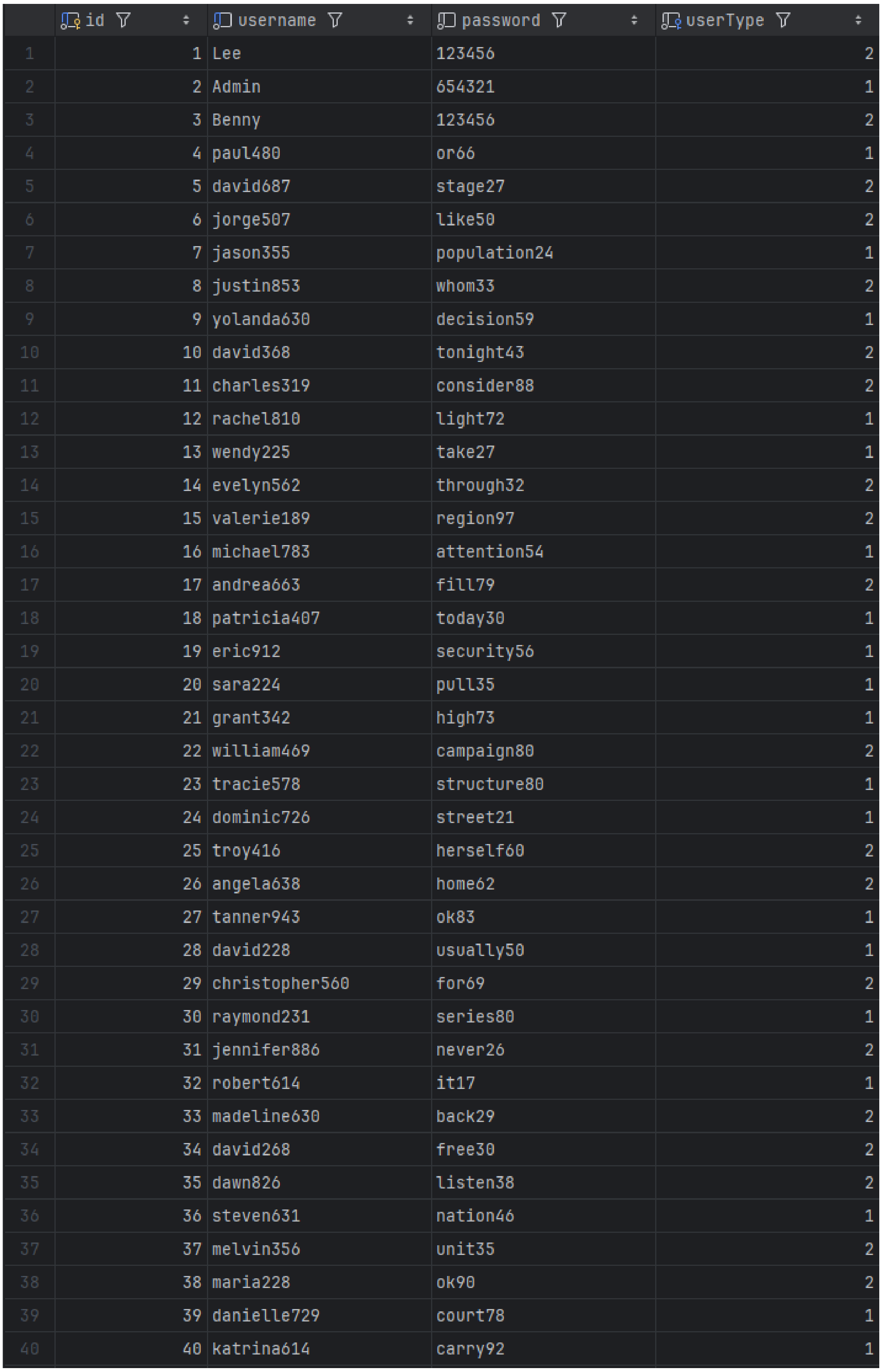
	id	facilityId	statusComment
1	1	13	Bin is full
2	2	12	Not working
3	3	4	Often busy
4	4	4	One charger not working
5	5	1	Always lots available
6	6	1	Great way to get around
7	7	7	Great to charge your phone but bring a cable

atabase Screenshot for ecoFacilities (476 Records)



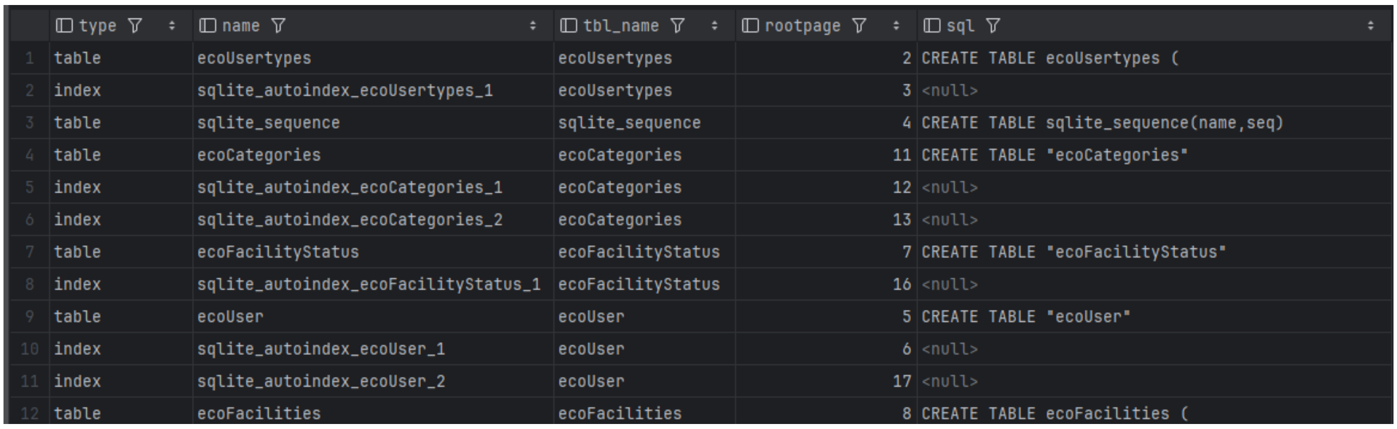
id	title	category	description	houseNumber	streetName	county	town	postcode	lng	lat	contributor	status
1	Recycling Bins at University Campus	1	Located outside the main library	10	Upper Brook Street	Greater Manchester	Manchester	M13 9PL	-2.2345	53.4669	2	1
2	Bike Share Station at Piccadilly Gardens	3	Easily accessible bike rentals	25	Piccadilly Gardens	Greater Manchester	Manchester	M1 1RG	-2.2335	53.4822	3	3
3	Public EV Charging at Deansgate	4	Charging points available at the parking garage	100	Deansgate	Greater Manchester	Manchester	M3 2GP	-2.2461	53.4785	1	7
4	Battery Recycling Point at St Peters Square	5	Batteries can be disposed here safely	10	St Peters Square	Greater Manchester	Manchester	M2 5GP	-2.2474	53.4784	2	1
5	Community Compost Bins at Whitworth Park	6	Bins for composting available in the park	30	Grafton Street	Greater Manchester	Manchester	M14 5SL	-2.2293	53.4651	1	1
6	Solar-Powered Benches at Sackville Gardens	7	Benches with solar panels for charging devices	5	Sackville Street	Greater Manchester	Manchester	M1 3HG	-2.235	53.4731	3	5
7	Public Water Refill Station at Northern Quarter	9	Refill station located at the park	15	Tib Street	Greater Manchester	Manchester	M4 1NB	-2.236	53.485	2	1
8	Waste Oil Collection at Ancoats	10	Dispose of used cooking oil responsibly	20	Cutting Room Square	Greater Manchester	Manchester	M4 6BU	-2.2337	53.4791	1	5
9	Community Tool Library at Longsight	15	Borrow gardening and DIY tools	12	Stockport Road	Greater Manchester	Manchester	M13 0XR	-2.2173	53.4551	2	4
10	Community Bike Hub	3	Centrally located bike rentals for locals.	25	Oxford Street	Greater Manchester	Manchester	M1 6FQ	-2.2409	53.4795	2	6
11	Solar-Powered Water Fountain	9	Eco-friendly water fountain powered by solar energy.	10	Deansgate	Greater Manchester	Manchester	M3 2RP	-2.2502	53.4767	1	4
12	E-Waste Collection Point	13	Drop-off point for recycling old electronics.	5	Portland Street	Greater Manchester	Manchester	M1 6DF	-2.2382	53.4784	3	3
13	Battery Recycling Station	5	Secure drop-off for household batteries.	15	King Street	Greater Manchester	Manchester	M2 6EJ	-2.2485	53.4774	2	4
14	Urban Green Space	8	A small park area with native plants and seating.	40	Cross Street	Greater Manchester	Manchester	M2 4FW	-2.2403	53.4791	1	2
15	Community Composting Site	6	A local site for composting organic waste.	12	Albert Square	Greater Manchester	Manchester	M2 5DB	-2.2427	53.4811	3	3
16	Public EV Charging Hub	4	Charging station for electric vehicles in the city center.	30	Chepstow Street	Greater Manchester	Manchester	M1 5FW	-2.2374	53.4741	1	3
17	Book Exchange Kiosk	11	A small kiosk for exchanging books in the community.	8	St. Mary's Gate	Greater Manchester	Manchester	M3 2WQ	-2.2518	53.4762	2	1
18	Clothing Donation Drop Box	14	Drop-off box for donating clothes to those in need.	100	Grafton Street	Greater Manchester	Manchester	M13 9WT	-2.2356	53.4654	3	6
19	Pollinator Garden	12	A garden designed to attract and support local pollinators.	50	Blossom Street	Greater Manchester	Manchester	M12 6BG	-2.2275	53.4728	1	2
20	Book Swap Stations	11	Located behind the library	45	Juan Brooks	Greater Manchester	Trafford	M17 1AB	-2.287965	53.500187	2	1
21	Clothing Donation Bins	14	Next to the park entrance	75	Brian Hill	Greater Manchester	Manchester	M13 9PL	-2.279976	53.483175	2	3
22	Solar-Powered Benches	7	Located behind the library	64	Justin Prairie	Greater Manchester	Eccles	M30 0BL	-2.277462	53.447814	1	1
23	Public Water Refill Stations - Practice	9	Located behind the library, across from the high school	12	Tammy Inlet	Greater Manchester	Eccles	M30 0BL	-2.22991	53.457157	1	5
24	Public EV Charging Stations	4	Near the community centre	83	Flores Lakes	Greater Manchester	Salford	M5 4WT	-2.219016	53.450753	3	2
25	Solar-Powered Benches	7	Borrow gardening and DIY tools	65	Peterson Islands	Greater Manchester	Salford	M5 4WT	-2.278633	53.445926	2	6
26	Solar-Powered Benches	7	Next to the park entrance	36	Thomas Squares	Greater Manchester	Salford	M5 4WT	-2.285314	53.505878	1	6
27	e-Scooters	2	Next to the park entrance	12	Nicole Club	Greater Manchester	Trafford	M17 1AB	-2.202615	53.467495	3	6
28	Public Water Refill Stations	9	Near the community centre	92	Patrick Views	Greater Manchester	Eccles	M30 0BL	-2.264921	53.485568	3	5
29	Waste Oil Collection Points	10	Outside the co-op	17	Perkins Shoal	Greater Manchester	Trafford	M17 1AB	-2.26101	53.490266	2	2
30	Public EV Charging Stations - True	4	Near the community centre	88	Katie Path	Greater Manchester	Eccles	M30 0BL	-2.228659	53.471434	3	5
31	Recycling Bins	1	Outside the co-op	76	Colton Tunnel	Greater Manchester	Stretford	M32 0ZF	-2.267651	53.520136	2	7
32	E-Waste Collection Bins - Court	13	Next to the park entrance	8	Elizabeth Estates	Greater Manchester	Stretford	M32 0ZF	-2.194926	53.498125	2	1
33	Public EV Charging Stations	4	Outside the co-op	18	Contreras Trail	Greater Manchester	Manchester	M13 9PL	-2.198894	53.450569	3	1
34	Battery Recycling Points	5	Next to the park entrance	56	Brown Port	Greater Manchester	Eccles	M30 0BL	-2.218434	53.507987	2	7
35	Public EV Charging Stations	4	Located behind the library	19	Daniel Court	Greater Manchester	Salford	M5 4WT	-2.22935	53.491607	2	3
36	Recycling Bins - Where	1	Near the community centre	87	Hudson Vista	Greater Manchester	Stretford	M32 0ZF	-2.256478	53.519532	3	5
37	Public Water Refill Stations - Baby	9	Near the community centre	86	Debra Valleys	Greater Manchester	Salford	M5 4WT	-2.275941	53.516336	2	1
38	Bike Share Stations	3	Located behind the library, next to the community centre	68	Vanessa Track	Greater Manchester	Stretford	M32 0ZF	-2.220232	53.447979	3	6
39	Community Compost Bins	6	Borrow gardening and DIY tools	33	Flynn Island	Greater Manchester	Stretford	M32 0ZF	-2.267892	53.525154	3	2
40	Green Roofs	8	Outside the co-op, just outside the main car park	25	Christina Mills	Greater Manchester	Trafford	M17 1AB	-2.241403	53.442182	2	4
41	Waste Oil Collection Points	10	Next to the park entrance	42	Joel River	Greater Manchester	Manchester	M13 9PL	-2.193764	53.442676	2	1
42	Clothing Donation Bins - Explain	14	Outside the co-op	12	Daniel Underpass	Greater Manchester	Stretford	M32 0ZF	-2.230795	53.489817	3	1
43	Book Swap Stations - Practice	11	Next to the park entrance, near the bus stop	72	Fox Wells	Greater Manchester	Manchester	M13 9PL	-2.195366	53.486315	1	4
44	Recycling Bins - Recognize	1	Borrow gardening and DIY tools, just outside the main car park	12	Green Isle	Greater Manchester	Trafford	M17 1AB	-2.250677	53.479424	2	5

Database Screenshot for ecoUsers (400 Users)



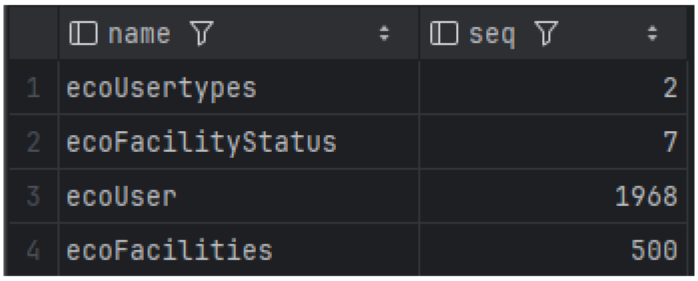
	id	username	password	userType
1	1	Lee	123456	2
2	2	Admin	654321	1
3	3	Benny	123456	2
4	4	paul480	or66	1
5	5	david687	stage27	2
6	6	jorge507	like50	2
7	7	jason355	population24	1
8	8	justin853	whom33	2
9	9	yolanda630	decision59	1
10	10	david368	tonight43	2
11	11	charles319	consider88	2
12	12	rachel810	light72	1
13	13	wendy225	take27	1
14	14	evelyn562	through32	2
15	15	valerie189	region97	2
16	16	michael783	attention54	1
17	17	andrea663	fill79	2
18	18	patricia407	today30	1
19	19	eric912	security56	1
20	20	sara224	pull35	1
21	21	grant342	high73	1
22	22	william469	campaign80	2
23	23	tracie578	structure80	1
24	24	dominic726	street21	1
25	25	troy416	herself60	2
26	26	angela638	home62	2
27	27	tanner943	ok83	1
28	28	david228	usually50	1
29	29	christopher560	for69	2
30	30	raymond231	series80	1
31	31	jennifer886	never26	2
32	32	robert614	it17	1
33	33	madeline630	back29	2
34	34	david268	free30	2
35	35	dawn826	listen38	2
36	36	steven631	nation46	1
37	37	melvin356	unit35	2
38	38	maria228	ok90	2
39	39	danielle729	court78	1
40	40	katrina614	carry92	1

Database Screenshot for sqlite_master



	type	name	tbl_name	rootpage	sql
1	table	ecoUsertypes	ecoUsertypes	2	CREATE TABLE ecoUsertypes (
2	index	sqlite_autoindex_ecoUsertypes_1	ecoUsertypes	3	<null>
3	table	sqlite_sequence	sqlite_sequence	4	CREATE TABLE sqlite_sequence(name,seq)
4	table	ecoCategories	ecoCategories	11	CREATE TABLE "ecoCategories"
5	index	sqlite_autoindex_ecoCategories_1	ecoCategories	12	<null>
6	index	sqlite_autoindex_ecoCategories_2	ecoCategories	13	<null>
7	table	ecoFacilityStatus	ecoFacilityStatus	7	CREATE TABLE "ecoFacilityStatus"
8	index	sqlite_autoindex_ecoFacilityStatus_1	ecoFacilityStatus	16	<null>
9	table	ecoUser	ecoUser	5	CREATE TABLE "ecoUser"
10	index	sqlite_autoindex_ecoUser_1	ecoUser	6	<null>
11	index	sqlite_autoindex_ecoUser_2	ecoUser	17	<null>
12	table	ecoFacilities	ecoFacilities	8	CREATE TABLE ecoFacilities (

Database Screenshot for sqlite_sequence



	name	seq
1	ecoUsertypes	2
2	ecoFacilityStatus	7
3	ecoUser	1968
4	ecoFacilities	500