

Water Resources Utilization Section (WRUS) Portal

System Documentation

September 11, 2025

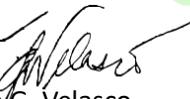


"Nurturing Cities for a Sustainable Future"

Approved by:

Carmen Hizelle G. Velasco

Chief, Water Resources Utilization Section



Prepared by:

Jsonon Kym V. Carabeo, CCNA

Land Management Inspector





Table of Contents

I.	Background of the Study	5
II.	Introduction	6
III.	Definition of Terms	7
IV.	Objectives	10
V.	Scopes and Limitations	11
VI.	SWOT Analysis	16
VII.	Gantt Chart	17
VIII.	Interface Overview	18
IX.	System Architecture	57
X.	Technologies Use	58
XI.	Database Structure	59
XII.	Data Flow Diagram	61
XIII.	Project File Structure	79
XIV.	Photo Documentation	85
XV.	Source Code	86
	XV.A HTML	86
	XV.B CSS	153
	XV.C Javascript	205
	XV.C.1 Authentication Scripts	205
	XV.C.2 Component Scripts	208
	XV.C.3 Data Scripts	218
	XV.C.4 Module Scripts	241
	XV.C.4.a Consumable Module	241
	XV.C.4.b Dashboard Module	256
	XV.C.4.c MEMIS Upload Module	278
	XV.C.4.d ICS Module	283
	XV.C.4.e Map Module	295
	XV.C.4.f PDF Scripts	309
	XV.C.4.g Permit Module	324
	XV.C.4.h Analytics Module	341
	XV.C.4.i Image Upload Scripts	352
	XV.C.4.j User Management Module	357

XV.C.4.k Utility Scripts	367
XV.C.4.l Mobile Inventory Form Module	368
XV.C.4.m Water Users and Sources Module	380
XV.D Docker	399
XV.E Node.js	400
XVI. Conclusion	401
XVII. Reference	402



I. Background of the Study

Effective management of water resources is crucial for ensuring sustainable use and compliance with regulations. Within the **Department of Environment and Natural Resources (DENR)**, the **Water Resources Utilization Section (WRUS)** plays a critical role in monitoring water permits, tracking water users, and overseeing source inventories. Traditionally, these functions have relied on **manual record-keeping and fragmented digital files**, which can result in **inefficiencies, delays, and difficulties** in data validation.

The increasing need for real-time data access, geospatial analysis, and digital record management has highlighted the importance of a **centralized system**. Furthermore, field personnel often encounter challenges when working in remote areas with limited internet connectivity, making offline functionality and synchronization mechanisms vital for maintaining accurate records.

To address these challenges, the **WRUS Portal** was developed as a **web-based platform** that integrates multiple operational functions into a single system. It leverages cloud technologies, geospatial mapping tools, and offline data handling to improve efficiency, accuracy, and accessibility of water resource information. By consolidating permits, water inventory data, consumables management, and analytics in one platform, the WRUS Portal strengthens monitoring, reporting, and decision-making within the section.



II. Introduction

The **Water Resources Utilization Section (WRUS) Portal** was developed to support the systematic management of operational functions within the section. It consolidates various modules and tools into a **unified web-based platform** to improve efficiency, data accessibility, and decision-making. The development of this system was accomplished at **no cost** by utilizing free resources and the assistance of **ChatGPT** for coding support and content refinement.

Key Features

- Permit, Water Users, and Sources Management
 - Maintains structured records of permits and water source inventories.
 - Provides organized data to enable effective monitoring, updating, and reporting.
- Interactive Mapping
 - Offers a map-based interface similar to ArcGIS.
 - Displays permit locations, water sources, and related geospatial data.
- Mobile Inventory Form Module
 - Allows field personnel to record water inventory data directly on-site.
 - Supports e-signature capability for field validation and authentication.
- MEMIS Data Integration
 - Enables direct uploading of MEMIS Excel files into the database.
 - Minimizes manual encoding and streamlines bulk data entry.
- Consumables Analytics
 - Tracks usage trends and average monthly consumption.
 - Identifies items that require replenishment based on lifespan estimates.
- Map Routing Module
 - Uses recorded geographic coordinates to generate optimized routes to water permit sites.
 - Aims to improve field navigation and inspection planning.

III. Definition of Terms

Note:

[x] – see links in the reference section.

III.A. System – A group of devices or artificial objects or an organization forming a network, especially for distributing something or serving a common purpose. ^[1]

III.B Module – A module is a function or group of similar functions. They are grouped together within a file and contain the code to execute a specific task when called into a larger application. ^[2]

III.C. Database – a large amount of information stored in a computer system in such a way that it can be easily looked at or changed. ^[3]

III.D. Water Inventory Module – A system component that records, manages, and stores water resource inventory data, including geotagged photos and location information.

III.E. Water Permit Module – A system feature that manages water permits, linking permittees to their water resource data and associated geospatial records.

III.F. Mobile Inventory Form – An offline-enabled form used by field staff to collect water inventory data in remote locations without internet connectivity.

III.G. Javascript - The programming language of the Web. ^[4]

III.H. Geotagged Photo – Geotagging of images refers to the process of adding geographical information, such as latitude, longitude, or other location-related metadata, to an image file. ^[5]

III.I. E-Signature (Electronic Signature) – An electronic signature, or e-signature, is a digital representation of an individual's intent to sign a document or agreement. ^[6]

III.J. Local Storage – localStorage is a property that allows JavaScript sites and apps to save key-value pairs in a web browser with no expiration date. This means the data stored persists even after the user closes the browser or restarts the computer. ^[7]

III.K. IndexedDB – IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs. ^[8]

III.L. API (Application Programming Interface) - An API, or application programming interface, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features, and functionality. ^[9]

III.M. Cloud - The cloud is made up of servers in data centers all over the world. Moving to the cloud can save companies money and add convenience for users. ^[10]

III.N. Firestore (Cloud Firestore) – Firestore is a flexible, scalable database for mobile device, web, and server development from Firebase and Google Cloud. ^[11]

Firestore service is a no-SQL database that stores all of its data in the form of *collections* instead of *tables*. ^[12]

III.O. Firestore Collection – a Firestore collection consists of documents. A document has fields with corresponding values, like a key-value pair. A field can be made up of different types (e.g., string, number, array, boolean, etc.). ^[12]

III.P. Cache – In computing, a cache is a high-speed data storage layer which stores a subset of data, typically transient in nature, so that future requests for that data are served up faster than is possible by accessing the data's primary storage location. ^[13]

III.Q. Synchronization (Sync Manager) – The process of uploading offline data from Local Storage and IndexedDB to Firestore once internet connectivity is available.

III.R. Ledger – a digital record that is used similarly to an accounting ledger (as for maintaining a list of transactions) ^[14]

III.S. Report Generator – A system process that compiles data from Firestore to create dashboards, analytics, and downloadable reports (e.g., Excel files).

III.T. Map Routing – A feature that utilizes geospatial data to provide navigation paths to water sources or field sites using mapping libraries such as Leaflet.js.

III.U. GeoJSON – GeoJSON is a format for encoding a variety of geographic data structures. ^[15]

III.V. Authentication – Authentication is the process of determining whether someone or something is who or what they say they are. ^[16]

III.W. Local Development Server (NPM Live Server) – A tool used during development to preview and test the system locally before deployment.

III.X. Docker – Docker is an open-source platform that enables developers to build, deploy, run, update and manage containers. ^[16]

III.Y. User Interface (UI) – the way in which the information on a computer, phone, etc., and instructions on how to use it are arranged on the screen and shown to the user. ^[17]

III.Z. Analytics – Analytics is the process of discovering, interpreting, and communicating significant patterns in data. ^[18]

III.AA. Source Code – Source code is a group of instructions a programmer writes using computer programming languages. ^[19]

III.AB. Hosting – Web hosting is a cloud service in which a service provider stores all the files that comprise a website on a server and makes the website accessible on the internet. ^[20]

III.AC. Modal – The Modal plugin is a dialog box/popup window that is displayed on top of the current page. ^[21]

III.AD. Data Flow Diagram (DFD) – A data flow diagram (DFD) is a visual representation of the flow of data through an information system or business process. ^[22]

III.AE. IDE (Integrated Development Environment) – An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently. ^[23]

III.AG. Bootstrap – Bootstrap is a free front-end framework for faster and easier web development ^[24]

III.AH. Front-End - The front end of a website is everything the user either sees or interacts with when they visit the website. It is responsible for the total look and feel of an online experience. ^[25]

III.AI. Back-End - The back end of a website is everything that goes on behind the scenes, from servers to databases, and much more. ^[26]

III.AJ. Session – A session is a way to store information (in variables) to be used across multiple pages. ^[27]

III.AK. Project File Structure – refers to the organized arrangement of files and folders within a software project. It defines where source code, configuration files, assets, and modules are placed, ensuring clarity, maintainability, and scalability of the system.

III.AL. GitHub - GitHub is a web-based interface that uses Git, the open source version control software that lets multiple people make separate changes to web pages at the same time. ^[28]

III.AM. AI (Artificial Intelligence) - Artificial intelligence is a field of science concerned with building computers and machines that can reason, learn, and act in such a way that would normally require human intelligence or that involves data whose scale exceeds what humans can analyze. ^[29]

III.AN. ChatGPT - ChatGPT is a generative AI chatbot developed by OpenAI and powered by their proprietary GPT family of generative artificial intelligence (gen AI) models. It uses natural language processing (NLP) to hold lifelike conversations with users and generate content, including articles, text summaries, advice, and more. ^[30]

IV. Objectives

IV.A Centralization – provide a unified, web-based platform for managing permits, water users and sources, consumables, and assets, with integrated mapping, analytics, and inventory functions.

IV.B Efficiency – streamline operations through automation of repetitive processes, the use of mobile inventory forms with e-signature capability, and direct uploading of datasets (e.g., MEMIS Excel files) to the database.

IV.C Accuracy – ensure reliable information through structured data entry, geotagged photo documentation, real-time synchronization, and standardized record-keeping across all modules

IV.D Transparency – improve visibility by offering real-time access to permits, accountabilities, consumable analytics, and interactive maps of water inventory data, including routing features for field operations.

IV. E Security – protect sensitive records with robust user authentication, role-based access control, and detailed audit trails.



V. Scopes and Limitations

V.A. Scopes

V.A.1 User Authentication and Management

- Login System: Users log in with a username and password (no email required).
- Session Handling: Session-like behavior ensures that users cannot access modules (e.g., dashboard, consumables, ICS) without logging in.
- Role-Based Access: Roles such as admin and user are implemented to ensure appropriate access and data security.
- Each user is provided with the capability to update their own password to maintain account security.
- User Management Module:
 - Add, edit, and update user records.
 - Change username and password
 - Update employment status (e.g., permanent, contractual), which also determines access to specific modules such as ICS.
 - “Edit” functionality enables updating user details dynamically without requiring a page reload (AJAX-like behavior).

V.A.2 Water Permit Management

- Search permits by **Permit Number or Permittee Name**.
- Records stored in Firestore with fields such as:
 - permitno
 - permittee
 - latitude
 - longitude
 - diversion point
 - mailing address
- Each permit record can be linked with a **geotagged photo**, providing visual documentation of the site along with its location coordinates.
- The system includes functionality to **generate and download an Excel file** containing permit records, allowing for easy reporting and offline reference.

V.A.3 Consumables Module

- Displays consumable items in a table format.
- Fields include:
 - CID
 - Name
 - Detail
 - Quantity
- Enables real-time updates of stock levels, with potential for reorder tracking.
- Includes a ledger feature that records the assignment of consumables to users, providing traceability of who received which items and when.

V.A.4 Consumables Analytics Module

- Integrates with the Consumables Module to provide data-driven insights.
- Key features include:
 - Determining whether a consumable item requires replenishment based on stock thresholds.
 - Calculating the lifespan of each consumable by analyzing historical usage data.
 - Providing average monthly consumption figures to support procurement planning and inventory forecasting.

V.A.5 ICS Module

- Maintains a record of **Inventory Custodian Slips (ICS)** assigned to each employee.
- Displays all accountable items per employee in a structured list.
- Records may include:
 - ICS number
 - Employee Name
 - item Description
 - quantity and unit cost
 - date Issued
- Access is restricted to employees with **permanent status** and can only view their own accountabilities.

- Each employee is provided with a **summary list of all items under their accountability**, along with the **total cost** of these items.

V.A.6 Water Users and Sources Module

- Provides a structured database of registered water users and their corresponding water sources.
- Records may include:
 - Owner or permittee
 - Permit number (if applicable)
 - Source status (operational, non-operational, on-going)
 - Source type (e.g., groundwater, surface water, deep well, river, reservoir)
 - Location (coordinates and address)
 - Date inspected
 - Representative's name, designation, and contact number
 - Remarks
- Each water user and source entry can be linked with a geotagged photo, providing visual documentation of the site together with its location coordinates.
- The system supports the generation and downloading of Excel files containing permit and water user records, allowing for easy reporting and offline use.
- For permittees, this module is integrated with the Water Permit Management module, linking geotagged photos, permittee details, permit numbers, and corresponding address and coordinates.

V.A.7 Web-based Map Module

- Provides an interactive **map-based interface** displaying the locations of all registered water users and sources.
- Each entry is plotted using its stored geographic coordinates (latitude, longitude).
- Markers on the map distinguish between **permittees** (those with approved permits) and **non-permittees**, ensuring clarity in monitoring and also whether it's a **water source** or not.
- Users can click on a marker to view details such as:
 - Permit number (if applicable)
 - Permittee / Owner
 - Water Source Type (e.g., groundwater, surface water, deep well, river)
 - location description (coordinates and address)
 - status (permittee or non-permittee / water source or non-water source)

- The system incorporates **city and barangay boundaries** using shapefile-derived **GEOJSON data**, enabling users to visualize administrative boundaries on the map for more accurate analysis and monitoring.

V.A.8 Mobile Inventory Form Module

- Provides a mobile-friendly interface for conducting field inventories of water users and sources.
- Enables on-site data entry directly into the system, minimizing the need for manual encoding after inspections.
- Records captured through the form include:
 - Owner or permittee
 - Water source type and status
 - Location (coordinates and address)
 - Inspection date
 - Representative's name, designation, and contact number
 - Remarks and observations
- Supports **geotagged photo capture**, ensuring visual documentation is linked with location data.
- Includes **e-signature functionality**, allowing field representatives or permittees to authenticate and validate inspection records on-site.
- Synchronizes collected data in real time with the main database when internet connectivity is available, while supporting offline data entry with later synchronization.

V.A.9 MEMIS Data Upload Module

- Provides functionality to directly upload **MEMIS Excel files** into the database.
- Automates the transfer of large datasets, reducing the need for manual data entry.
- Ensures uploaded records are validated against predefined formats and field requirements to maintain data integrity.
- Integrates seamlessly with the **Water Users and Sources Module** and the **Water Permit Management Module**, enriching records with official MEMIS data.

V.A.10 Map Routing Module (In Development)

- Designed to provide route generation to water permit locations based on geographic coordinates recorded in the database.
- Intended features include:
 - Automatic plotting of the fastest or most efficient route from the user's current location to the selected water permit site.
 - Integration with the **Water Permit Management Module** and **Water Users and Sources Module** for seamless access to coordinates.
 - Display of step-by-step directions overlaid on the interactive map interface.
- Current Status: The module is partially implemented and remains under development. Completion and future enhancements—including optimized route accuracy, support for multiple waypoints, and improved mobile usability—are contingent on the availability of funding.

V.B. Limitations

V.B.1 Internet Dependency for Synchronization

- While the system supports offline mode through Local Storage and IndexedDB, final synchronization with Firestore requires an active internet connection. Users working in prolonged offline conditions cannot fully update the central database.

V.B.2 Browser-Based Constraints

- The system relies on browser features (Local Storage, IndexedDB, and caching), which may behave differently depending on the browser or version. Performance may degrade on outdated browsers.

V.B.3 Storage Capacity Limitations

- Local Storage has a limited capacity (typically around 5–10 MB per browser). IndexedDB provides larger capacity but still depends on the user's device storage availability. This may restrict the volume of offline data and large file attachments (e.g., geotagged images, e-signatures).

V.B.4 Device and Platform Dependency

- Mobile inventory forms and geolocation features depend on the accuracy and availability of the user's device GPS and sensors. In areas with poor GPS signal, data accuracy may be reduced.

V.B.5 Firestore Quotas and Limits

- Firestore has read, write, and storage limits (e.g., 50,000 document reads per day in the free tier). Heavy usage may incur costs or require quota upgrades.

V.B.6 Security Reliance on Firebase

- Authentication and data security are handled by Firebase Authentication and Firestore rules. Any vulnerabilities or downtime in Firebase services may impact system security and availability.

V.B.7 Data Accuracy Depends on Users

- Since field staff manually encode data and attach geotagged photos, the quality and accuracy of the information depend heavily on the diligence of system users.

V.B.8 Limited Offline Analytics

- While cached data allows offline form entry, full analytics, dashboards, and reports rely on Firestore and cannot be fully accessed without internet connectivity.

VI. SWOT Analysis

VI.A. Strengths

- Centralized system for asset inventory, water permits, and user management.
- Uses **Firestore** as a scalable, cloud-based database with real-time syncing.
- Implements **LocalStorage** and **IndexedDB** for caching and e-signature storage, improving performance.
- Audit trail available through **Ledger entries**, enhancing accountability.
- User-friendly interface built with **Bootstrap** for consistency and accessibility.

VI.B. Weaknesses

- Dependence on internet connection for Firestore; limited offline capabilities despite caching.
- Limited local storage capacity (LocalStorage and IndexedDB may have size restrictions).
- Learning curve for users unfamiliar with digital systems.
- Maintenance requires knowledge of Firebase and Firestore, which may not be common in all teams.

VI.C. Opportunities

- Expand system features with **analytics and reporting tools** for better decision-making.
- Integration with **GIS or mapping tools** for more advanced water source visualization.
- Possibility to extend into a **mobile application** for field inspections and data entry.
- Scalability with Firestore allows accommodating more modules without significant infrastructure changes.
- Potential to improve compliance monitoring with automated notifications and alerts.

VI.D. Threats

- Reliance on third-party services (Firebase/Google Cloud) may pose risks if pricing or policies change.
- Cybersecurity threats such as unauthorized access or data breaches.
- Possible system downtime during updates or connectivity issues.
- Resistance to digital transformation from some staff who prefer manual processes.

VII. Gantt Chart

Start Period: May 19, 2025

Completion Date: August 12, 2025

- █ 1st Phase – Asset Inventory System
- █ 2nd Phase – Water Inventory System
- █ Documentation

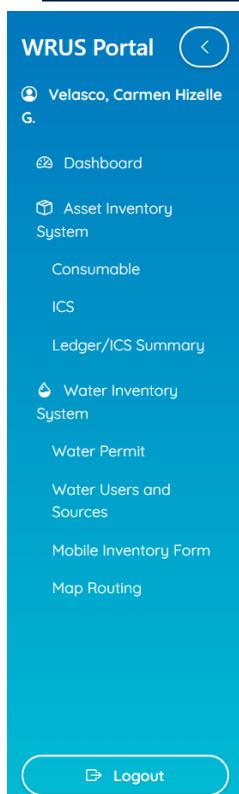
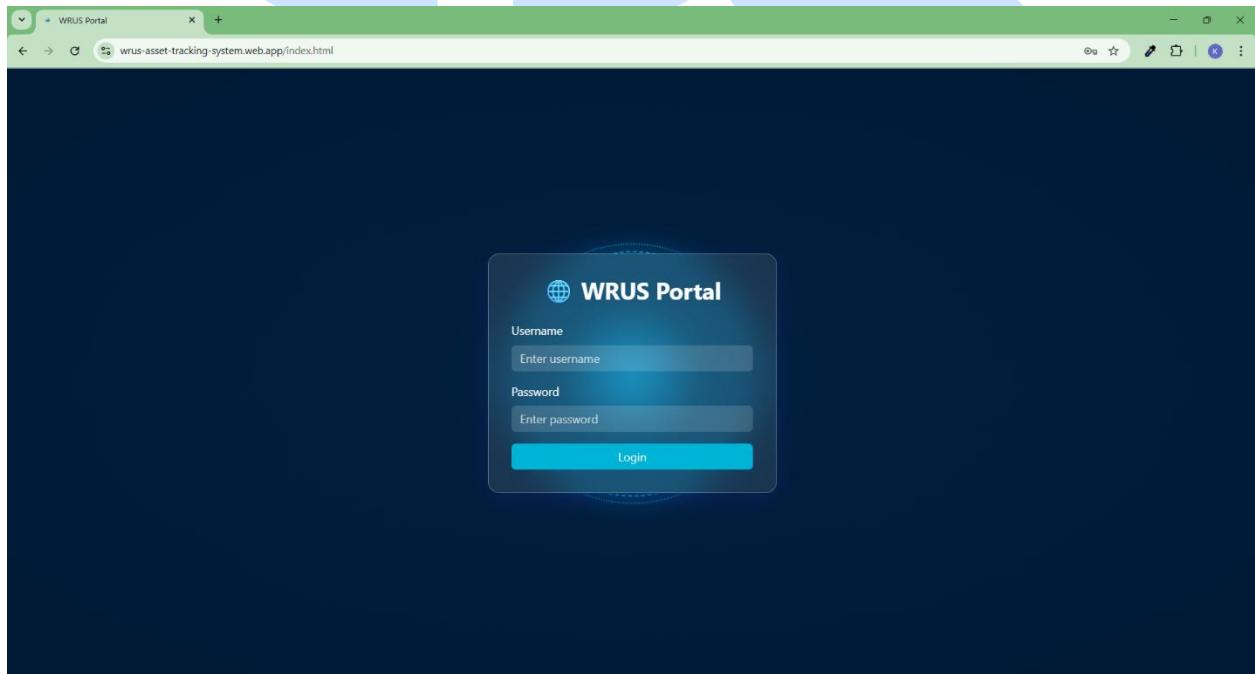
Activity	May	June	July			August		
Consumable Module								
ICS Module								
Analytics Module								
Water Permit Module								
Water Users and Sources Module								
Map Module								
Mobile Inventory Form Module								
MEMIS Excel File to Database Module								
Documentation								

VIII. Interface Overview

To provide a clearer understanding of how users interact with the system, the following section presents an overview of the system's interface, beginning with the Login Page.

VIII.A. Login Module

- The Login Page enables users to sign in with their username and password and access the Dashboard based on their role.



Sidebar Navigation

The sidebar provides streamlined navigation throughout the system. It displays the full name of the logged-in user and a **Logout** button. Key navigation links include:

- Dashboard** – Serves as the landing page and provides a summary overview of the system's data.
- Asset Inventory System** – Contains modules for consumables and employee accountabilities (ICS).
- Water Inventory** – Encompasses water permits, water users and sources, the mobile inventory form, MEMIS Excel file uploads, and the Map Routing module.

VIII.B. Dashboard Page

- The Dashboard serves as the system's **landing page**, providing users with a **centralized view** of key information, tools, and navigation options. It presents an organized **summary of data** and features, allowing users to **quickly access** essential functions and monitor important updates upon logging in.
- The first tab of the dashboard contains the **annual targets** of the Water Resources Utilization Section

The screenshot shows the WRUS Portal dashboard. On the left, a sidebar lists 'Dashboard', 'Asset Inventory System', and 'Water Inventory System'. The main area features a logo and the text 'Water Resources and Utilization Section - National Capital Region'. Below this is a section titled 'Activities - Annual Target' with four items: 1. Conduct of Inventory of Water Users (220), 2. Conduct of Inventory of Water Sources (110), 3. Mapping of Water Users and Sources (1), and 4. Maintenance and Updating of Database for Water Users (1).

- the corresponding **yearly accomplishments**

The screenshot shows the WRUS Portal dashboard. On the left, a sidebar lists 'Dashboard', 'Asset Inventory System', and 'Water Inventory System'. The main area features a table titled 'WRUS Accomplishment — Water Users and Sources — As of September 8, 2025'. The table details the number of permittees, non-permittees, and total users for each year from 2017 to 2025, along with the number of water sources. A separate section titled 'Issuances' is also visible.

Year	Water Users			Water Sources
	Permittees	Non-Permittees	Total	
2017	0	205	205	112
2018	0	188	188	104
2019	0	224	224	186
2021	225	60	285	24
2022	66	199	265	46
2023	53	197	250	62
2024	30	225	255	42
2025	22	38	60	19
Total	396	1336	1732	595

- and the **issuances** that establish and define the Section's functions.

WRUS Portal

Velasco, Carmen Hizelle G.

- Dashboard
- Asset Inventory System
- Water Inventory System

Issuances

An overview of the principal mandates of the Water Resources and Utilization Section (National Capital Region) concerning the conducting of water inventory, identification of water users and sources, preparation of reports, and management of geospatial information.

PD no. 1067 - Water Code of the Philippines

Presidential Decree No. 1067, or the Water Code of the Philippines, declares all waters state-owned and regulates their use, conservation, and protection through the NWRB.

[View PDF](#) [Download PDF](#)

NWRB Resolution No. 001-0904 - Metro Manila Critical Areas

NWRB Resolution No. 001-0904 designates Metro Manila as a critical water area to ensure stricter regulation, conservation, and sustainable management of its water resources.

[View PDF](#) [Download PDF](#)

NWRB 15-1116 - Deputation of DENR Regional Offices on Certain Functions of Water Use Regulation

NWRB Resolution No. 15-1116 authorizes DENR Regional Offices to assist in enforcing water use regulations by deputizing them for specific regulatory functions.

[View PDF](#) [Download PDF](#)

EO 22 - Creating the Water Resources Management Office in the DENR

Executive Order No. 22 establishes the Water Resources Management Office (WRMO) within the DENR to coordinate, integrate, and oversee the sustainable management of the country's water resources.

[View PDF](#) [Download PDF](#)

[Logout](#)

- The second tab of the dashboard presents statistical data on Water Permits and provides an ArcGIS-based mapping feature for Water Users and Sources.
- City/Municipality Permit Statistics** – presents the total number of permits on each city, the count of permits visited, the number of cancelled permits, and the percentage of water permits visited for each city/municipality.”

WRUS Portal

Velasco, Carmen Hizelle G.

- Dashboard
- Asset Inventory System
- Water Inventory System

About **Database** **Account Settings**

Water Permit **Water Users and Sources**

City / Municipality Permit Statistics — As of September 8, 2025				
City / Municipality	Total Permits	Visited	Cancelled	Visited %
Manila	43	0	1	0.00%
Makati	103	0	0	0.00%
Quezon City	203	0	0	0.00%
Pasig	56	0	2	0.00%
Pasay	10	0	0	0.00%
Taguig	71	0	0	0.00%
Marikina	76	1	0	1.32%
Mandaluyong	29	0	0	0.00%
San Juan	12	0	0	0.00%
Caloocan	89	2	0	2.25%
Malabon	15	0	0	0.00%
Valenzuela	64	3	0	4.69%
Parañaque	244	1	0	0.41%

[Logout](#)

- **Water Users and Sources Database**

Displays water users and sources data, along with the ArcGIS mapping feature, organized by city/municipality within the National Capital Region.

The data is further structured into three levels:

- ❖ **City/Municipality**

WRUS Portal

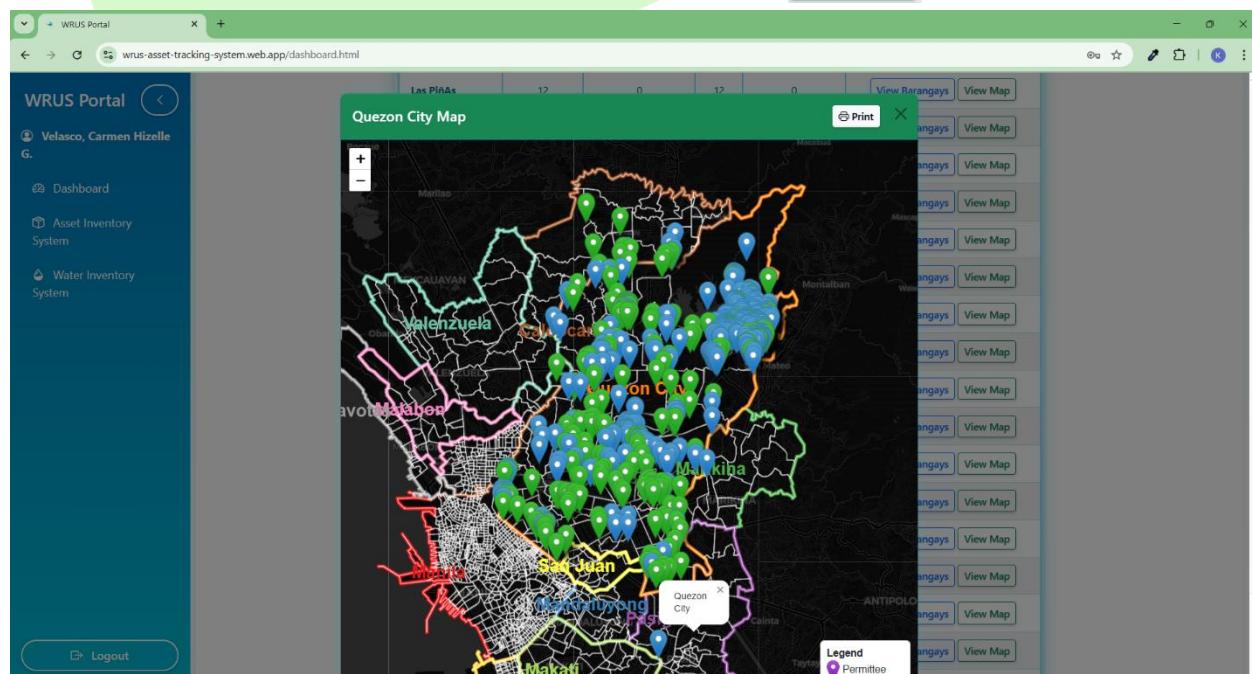
About Database Account Settings

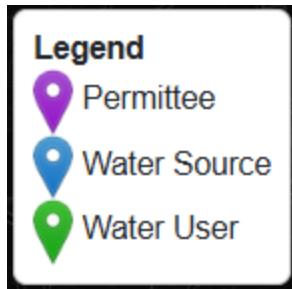
Water Permit Water Users and Sources

Water Users and Sources Database — As of September 8, 2025

City	Water Users			Water Sources	Barangay Details
	Permittees	Non-Permittees	Total		
Caloocan	14	52	66	1	View Barangays View Map
Las Piñas	12	0	12	0	View Barangays View Map
Makati	88	5	93	4	View Barangays View Map
Malabon	0	13	13	2	View Barangays View Map
Mandaluyong	30	4	34	3	View Barangays View Map
Manila	34	10	44	3	View Barangays View Map
Marikina	52	32	84	11	View Barangays View Map
Muntinlupa	0	13	13	0	View Barangays View Map
Navotas	0	5	5	1	View Barangays View Map

Map of all Water Users and Sources in Quezon City (if [View Map](#) is pressed)





- This image shows the legend explaining the symbols and markers used on the map.

- ❖ **Barangays** [View Barangays](#) within each city/municipality (*shown below, list of all barangays with recorded water users and water sources in Quezon City*)

The screenshot shows the WRUS Portal interface. On the left is a sidebar with user information ('Carabeo, Jonsonon Kym V.') and navigation links ('Dashboard', 'Asset Inventory System', 'Water Inventory System'). The main area displays a table titled 'Barangay Breakdown — Quezon City'. The table lists eight barangays with their respective counts of water users and sources. Each row has 'View Users' and 'View Map' buttons.

Barangay	0	1	1	0	View Users	View Map
Bagong Pagasa	0	1	1	0	View Users	View Map
Bagong Silangan	0	191	191	158	View Users	View Map
Bagumbayan	0	1	1	0	View Users	View Map
Bagumbong	0	2	2	0	View Users	View Map
Balara	0	1	1	0	View Users	View Map
Balintawak	0	2	2	2	View Users	View Map
Balon Bato	0	4	4	0	View Users	View Map
Batasan Hills	0	17	17	0	View Users	View Map

Map of all Water Users and Sources in the Brgy. Bagong Silangan if [View Map](#) is pressed

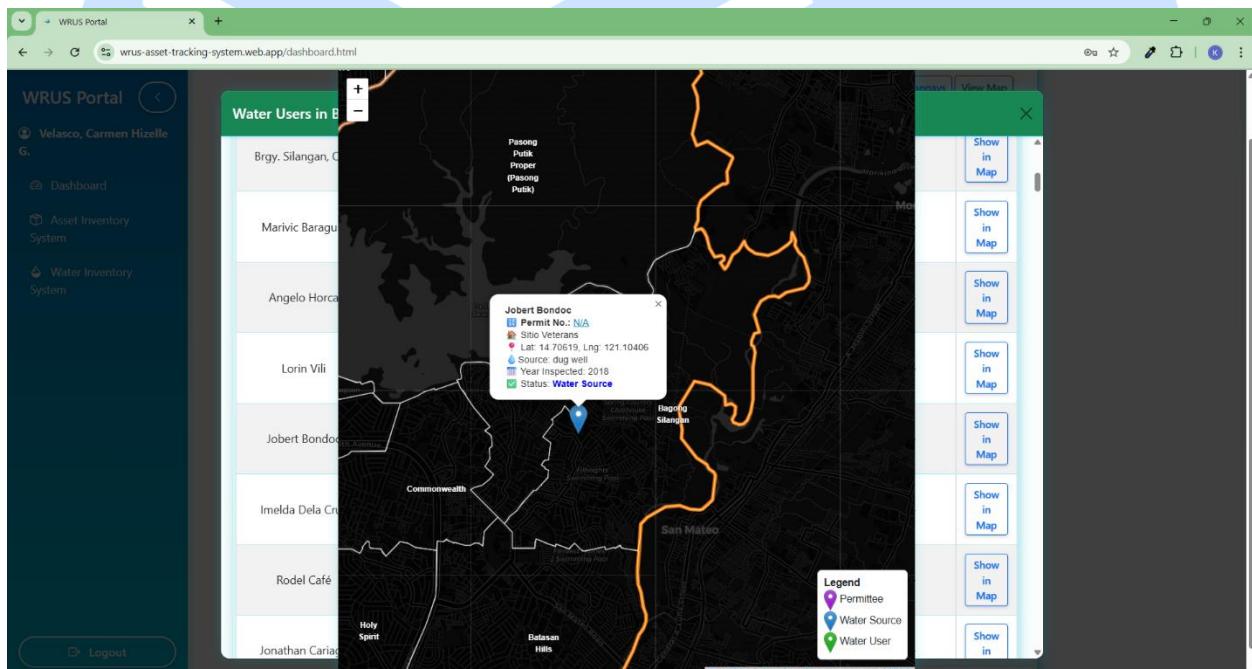
The screenshot shows the WRUS Portal interface with a map of Bagong Silangan, Quezon City. The map highlights the barangay boundaries and shows several blue location pins representing water sources. A callout box on the map identifies the location as 'Bagong Silangan, Quezon City'. The sidebar on the left remains the same as the previous screenshot.

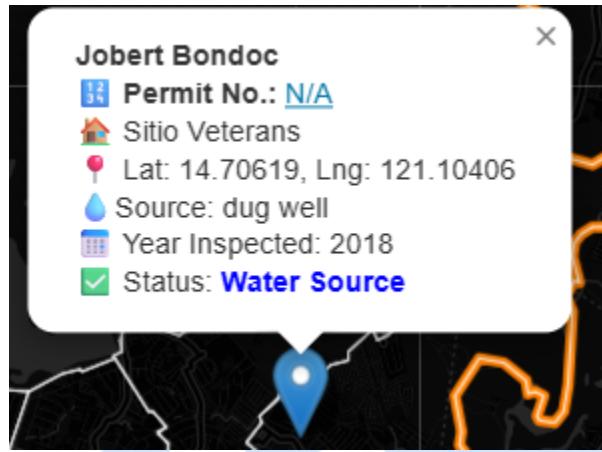
WRUS Portal

Water Users in Bagong Silangan, Quezon City

Owner	Street	Latitude	Longitude	Permit No.	Type	Year Inspected	Is Water Source?	Geotagged Image	View Map
Fausta Antonio	Lingayen Dulo Street	14.01178	121.00185	-	dug well	2024	Yes	-	Show in Map
Luz Vinson	Purok 7, Rolling Hills	14.72899	121.10138	-	hand pump well	2019	Yes	-	Show in Map
Pablo Fabriag	Purok 7, Rolling Hills	14.72832	121.10252	-	hand pump well	2019	Yes	-	Show in Map
Gina Gonzaga	Purok 7, Rolling Hills	14.72872	121.10167	-	hand pump well	2019	Yes	-	Show in Map
Isabelita Roberto	Rolling Hills	14.72707	121.10242	-	hand pump well	2019	Yes	-	Show in Map
Cornelia Dinquian (1)	Sitio Veterans	14.70683	121.10236	-	dug well	2018	Yes	-	Show in Map

Shown below is a single water user in the Brgy. Bagong Silangan if Show in Map is pressed





Details on each Water User on the map.

- Permit No – The permit number of the water user if it is a permittee
- Owner – The owner of the water facilities / deep well.
- Latitude and Longitude of the Water User
- Water Source
- Year Inspected
- Status – Whether it is a water source or not a water source.

- On the third tab of the Dashboard Page, the account settings allow the user to change their password.

VIII.C. Consumable Item Module

CID	Specification	Qty	UoM	Edit	Action
2025-060	Self-inking Stamp	0	pc		
2025-059	Cutter/Utility Knife, for general purpose	0	pcs		
2025-058	Folder, L-type, A4, clear, white, plastic	5	pcs		
2025-057	Battery- CR-2032	0	pcs		
2025-056	Flashdrive, 64GB capacity	3	pcs		
2025-055	Folder L-type, legal, 12/pack, clear, white	8	pcs		
2025-054	Epson Ink, 003, Yellow, 65 ml.	6	bottle		
2025-053	Epson Ink, 003, Magenta, 65ml.	6	bottle		
2025-052	Epson Ink 003, Cyan, 65ml.,	6	bottle		
2025-051	Epson Ink 003, black, 65 ml.	5	bottle		

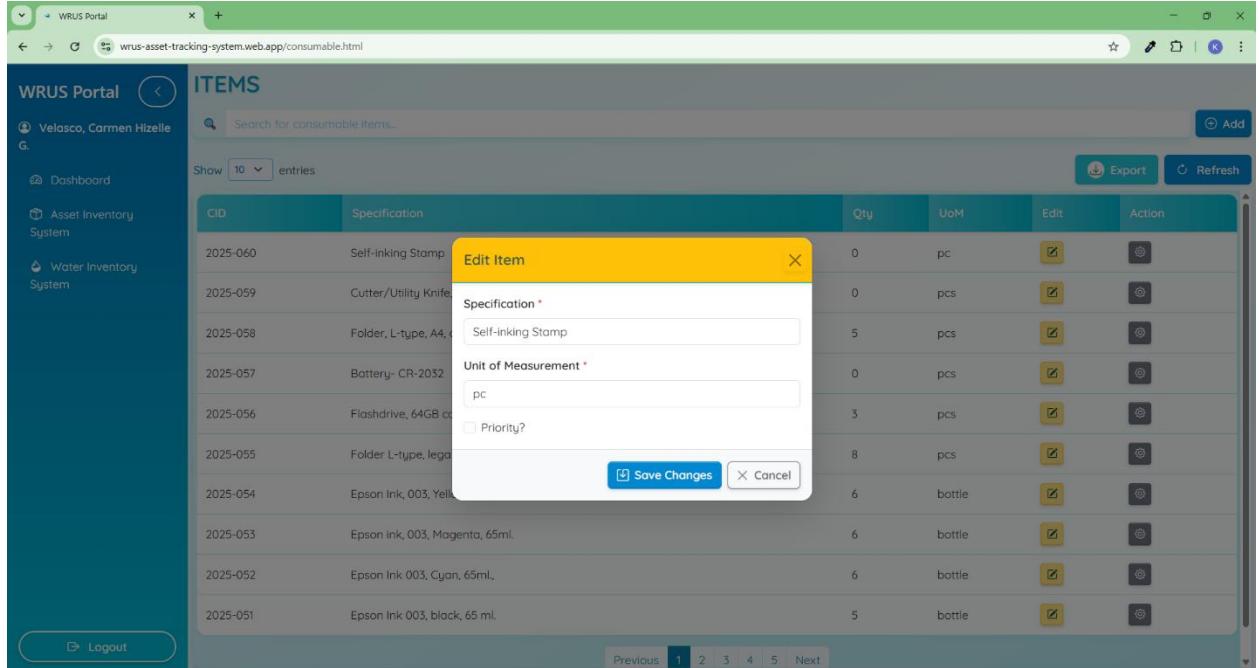
- This page enables users to add new consumable items, search for and update item descriptions, and modify stock levels, with each change reflected in the ledger. It also provides functionality to export all consumable items and consolidate them into an Excel file.
- Add new consumable item modal.** (if is pressed) – enables adding a new item to the database

Add New Item

Specification *	Enter specification
Quantity *	Enter quantity
Unit of Measurement *	e.g., pcs, box, set
Remarks	Optional notes or details

+ Add Item **Cancel**

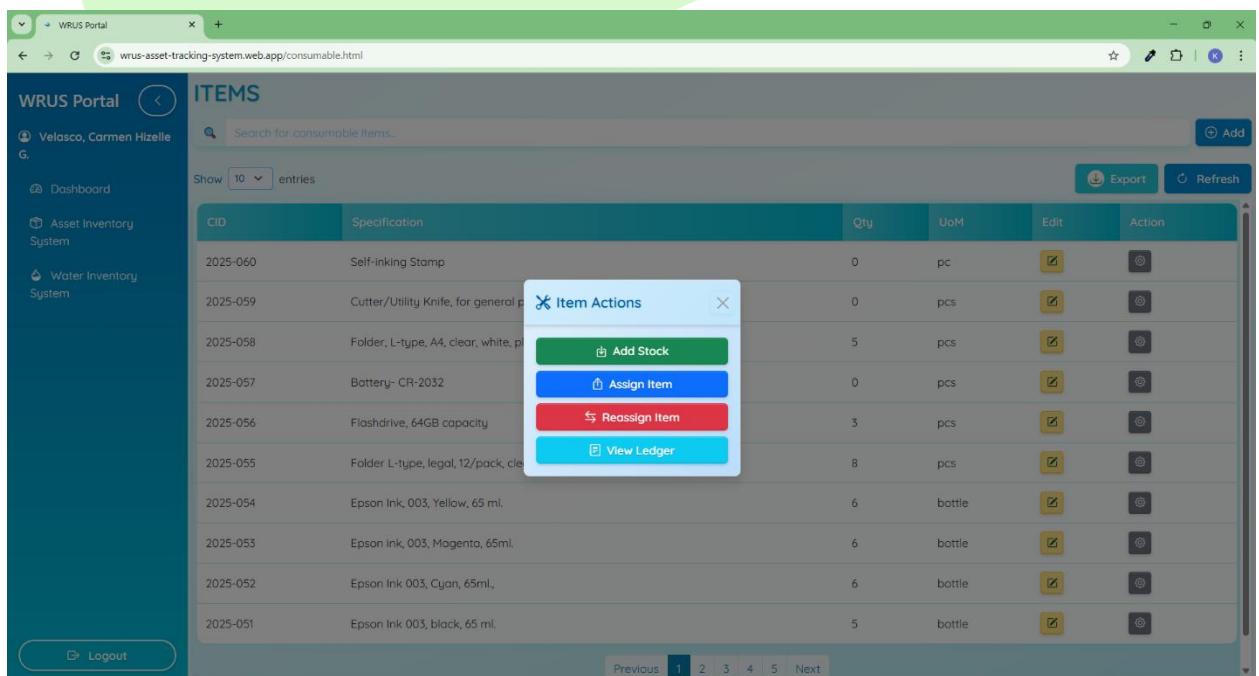
- **Edit Consumable modal** (*If  is pressed) – enables the user to edit item descriptions and set whether it is a priority item)*
- Priority Item – Items marked as priority are automatically included in the Consumables Analytics Module, enabling focused monitoring and analysis of critical inventory items.



The screenshot shows a web-based portal titled "WRUS Portal". On the left sidebar, there are links for "Velasco, Carmen Hizelle G.", "Dashboard", "Asset Inventory System", and "Water Inventory System". The main content area is titled "ITEMS" and contains a table of consumable items. A modal window titled "Edit Item" is open over the table. The modal has fields for "Specification" (containing "Self-inking Stamp") and "Unit of Measurement" (containing "pc"). There is also a checkbox for "Priority?". At the bottom of the modal are "Save Changes" and "Cancel" buttons. The table below the modal lists items with columns for CID, Specification, Qty, UoM, Edit, and Action. The table shows 10 entries, with page navigation buttons at the bottom.

CID	Specification	Qty	UoM	Edit	Action
2025-060	Self-inking Stamp	0	pc		
2025-059	Cutter/Utility Knife, for general purpose	0	pcs		
2025-058	Folder, L-type, A4, clear, white, plastic	5	pcs		
2025-057	Battery- CR-2032	0	pcs		
2025-056	Flashdrive, 64GB capacity	3	pcs		
2025-055	Folder L-type, legal, 12/pack, clear, white, plastic	8	pcs		
2025-054	Epson Ink, 003, Yellow, 65 ml.	6	bottle		
2025-053	Epson Ink, 003, Magenta, 65ml.	6	bottle		
2025-052	Epson Ink 003, Cyan, 65ml.	6	bottle		
2025-051	Epson Ink 003, black, 65 ml.	5	bottle		

- This modal is displayed when the action button  is selected. For each item, it presents three actions that directly affect the stock level (Add Stock, Assign Item, Reassign Item), as well as an option to view the ledger (View Ledger).



The screenshot shows the same "WRUS Portal" interface as the previous one. The "Edit Item" modal has been replaced by a new modal titled "Item Actions". This modal contains four buttons: "Add Stock" (green), "Assign Item" (blue), "Reassign Item" (red), and "View Ledger" (cyan). The background table and sidebar remain the same.

CID	Specification	Qty	UoM	Edit	Action
2025-060	Self-inking Stamp	0	pc		
2025-059	Cutter/Utility Knife, for general purpose	0	pcs		
2025-058	Folder, L-type, A4, clear, white, plastic	5	pcs		
2025-057	Battery- CR-2032	0	pcs		
2025-056	Flashdrive, 64GB capacity	3	pcs		
2025-055	Folder L-type, legal, 12/pack, clear, white, plastic	8	pcs		
2025-054	Epson Ink, 003, Yellow, 65 ml.	6	bottle		
2025-053	Epson Ink, 003, Magenta, 65ml.	6	bottle		
2025-052	Epson Ink 003, Cyan, 65ml.	6	bottle		
2025-051	Epson Ink 003, black, 65 ml.	5	bottle		

- **Add Stock Modal** – Enables the addition of consumable item stock to the database. Note that these changes are permanent and cannot be undone.

WRUS Portal

ITEMS

Search for consumable items...

Show 10 entries

Add

Export Refresh

ID	Specification	Qty	UoM	Edit	Action
2025-060	Self-inking Stamp	0	pc	<input checked="" type="checkbox"/>	<input type="button"/>
2025-059	Cutter/Utility Knife, for general purpose	0	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-058	Folder, L-type, A4, clear, white, plastic	5	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-057	Battery- CR-2032	0	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-056	Flashdrive, 64GB capacity	3	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-055	Folder L-type, legal, 12/pack, clear	8	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-054	Epson Ink, 003, Yellow, 65 ml.	6	bottle	<input checked="" type="checkbox"/>	<input type="button"/>
2025-053	Epson ink, 003, Magenta, 65ml.	6	bottle	<input checked="" type="checkbox"/>	<input type="button"/>
2025-052	Epson Ink 003, Cyan, 65ml.	6	bottle	<input checked="" type="checkbox"/>	<input type="button"/>
2025-051	Epson Ink 003, black, 65 ml.	5	bottle	<input checked="" type="checkbox"/>	<input type="button"/>

Logout

Previous 1 2 3 4 5 Next

WRUS Portal

ITEMS

Search for consumable items...

Show 10 entries

Add

Export Refresh

ID	Specification	Qty	UoM	Edit	Action
2025-060	Self-inking Stamp	0	pc	<input checked="" type="checkbox"/>	<input type="button"/>
2025-059	Cutter/Utility knife, for general purpose	0	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-058	Folder, L-type, A4, clear, white, plastic	5	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-057	Battery- CR-2032	0	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-056	Flashdrive, 64GB capacity	3	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-055	Folder L-type, legal, 12/pack, clear	8	pcs	<input checked="" type="checkbox"/>	<input type="button"/>
2025-054	Epson Ink, 003, Yellow, 65 ml.	6	bottle	<input checked="" type="checkbox"/>	<input type="button"/>
2025-053	Epson ink, 003, Magenta, 65ml.	6	bottle	<input checked="" type="checkbox"/>	<input type="button"/>
2025-052	Epson Ink 003, Cyan, 65ml.	6	bottle	<input checked="" type="checkbox"/>	<input type="button"/>
2025-051	Epson Ink 003, black, 65 ml.	5	bottle	<input checked="" type="checkbox"/>	<input type="button"/>

Are you sure you want to add to this stock? This action cannot be undone.

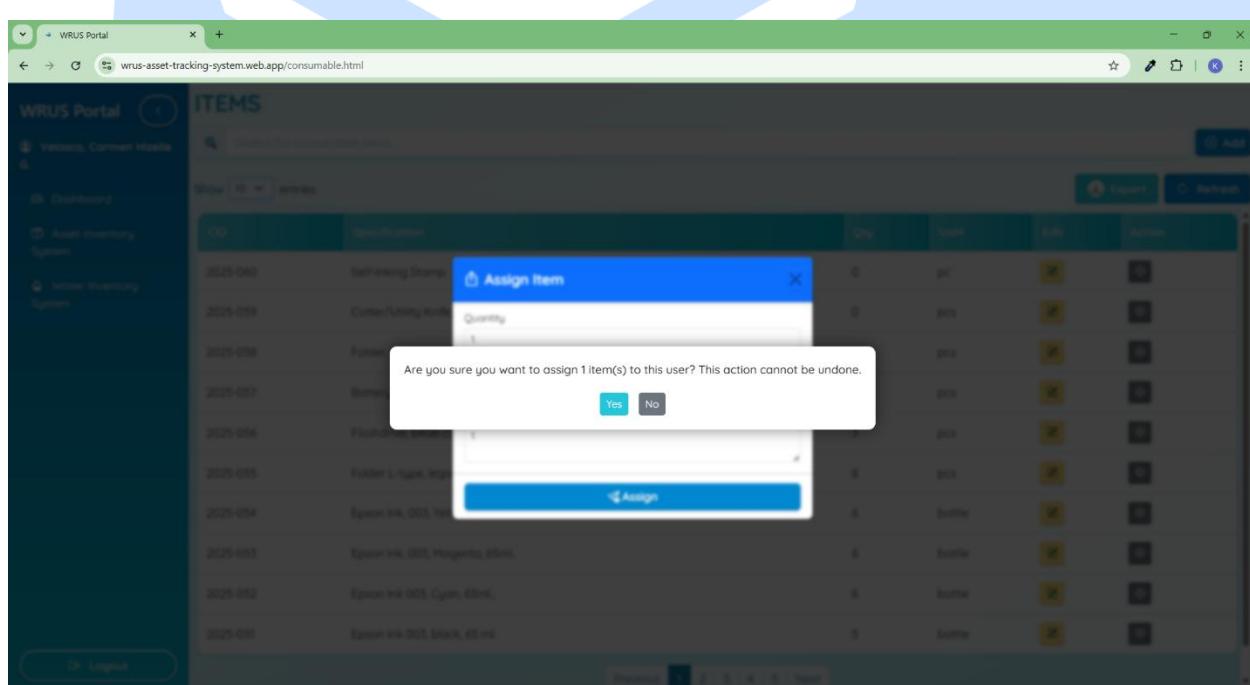
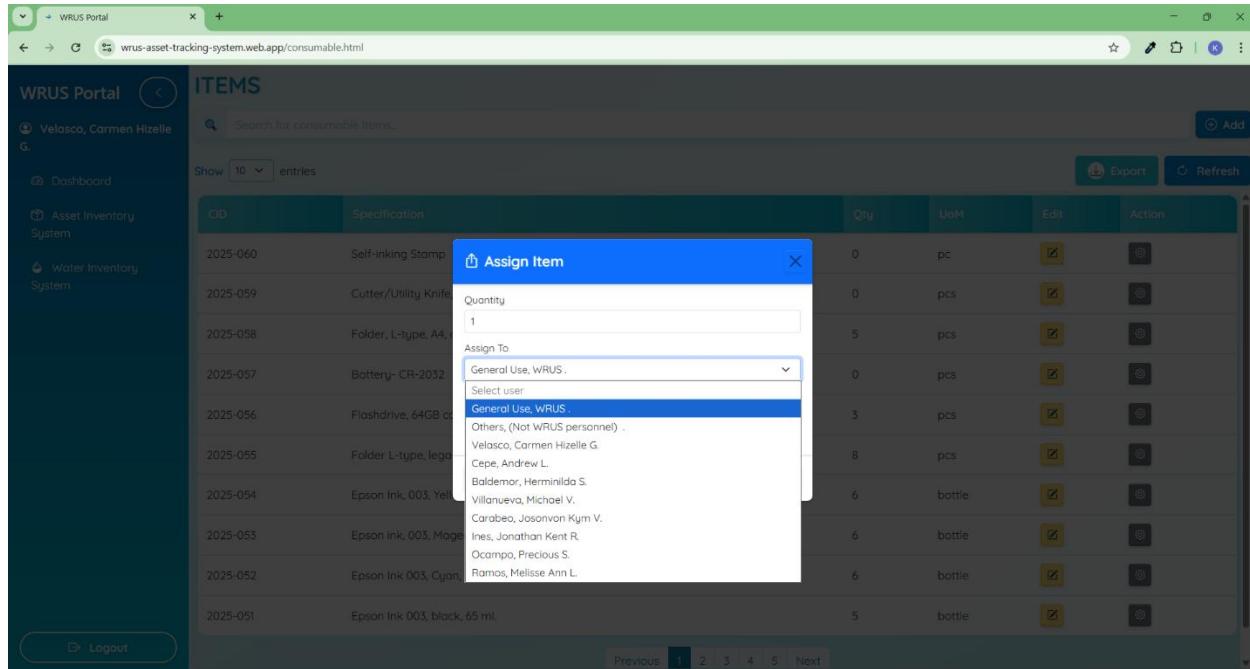
Yes No

Confirm

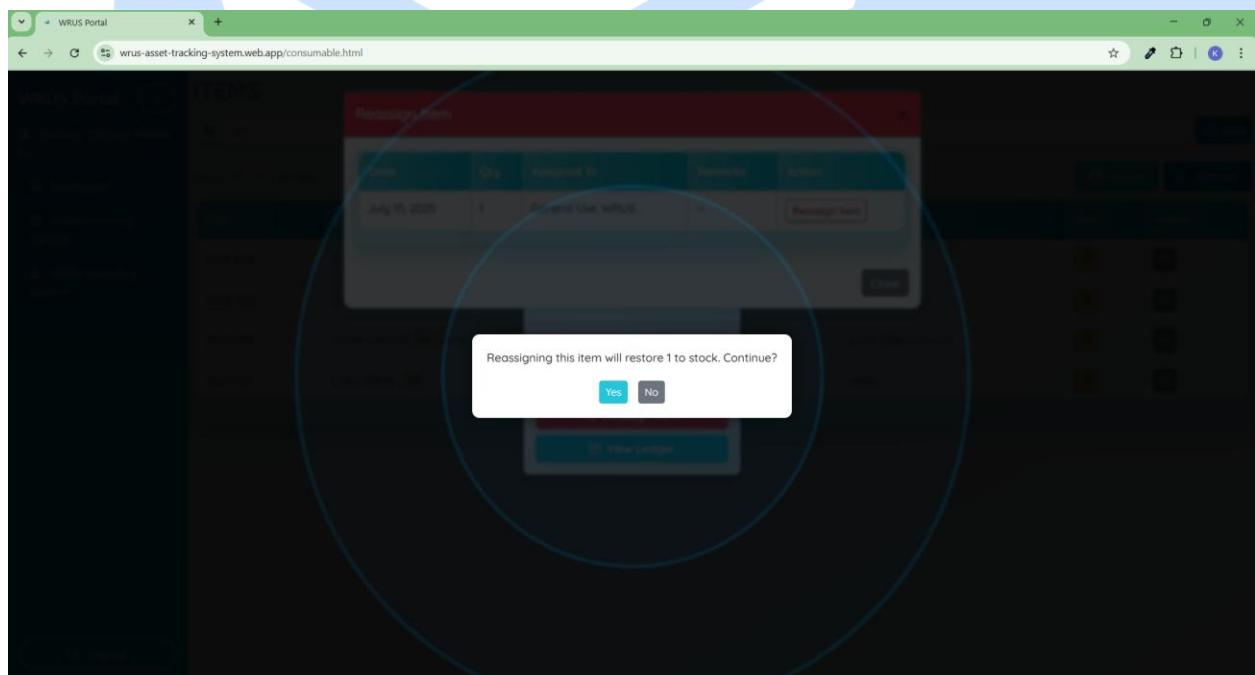
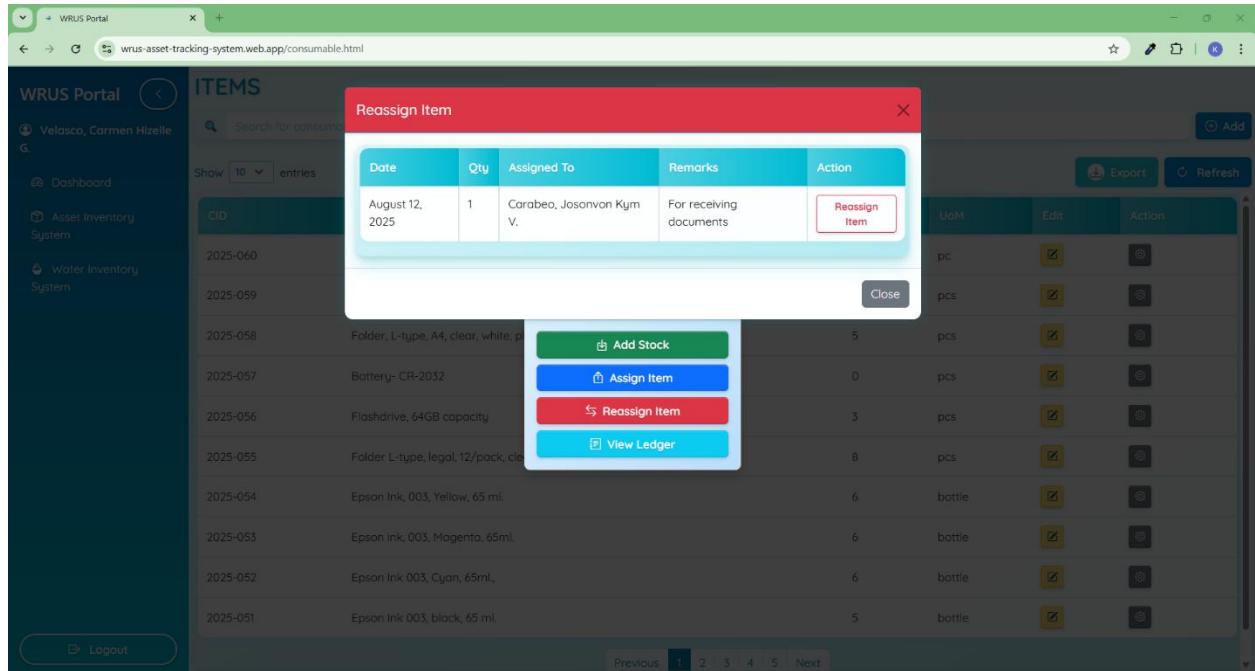
Logout

Previous 1 2 3 4 5 Next

- **Assign Item Modal** – enables the user to assign an item to an employee of the WRUS section. The quantity will be subtracted from the stock level of a consumable item. Also, action can't be undone.



- **Reassign Item Modal** – Enables the reassignment of a consumable item in cases such as a change of ownership or correction of an incorrect assignment. Also, changes can't be undone



- View Ledger Modal** – Enables users to access item transaction ledgers in PDF format, providing a detailed audit trail of each consumable. Ledger entries include quantities issued or returned and the responsible user for each transaction. The first table displays additions to inventory (credits), while the second table displays subtractions from inventory (debits), facilitating proper tracking and reconciliation of stock movements.

Ledger Entries

Date: September 8, 2025
Ledger Report for: Copy Paper, A4
Stocks Left: 3
Unit of Measurement: ream
Add to Stock

Date	Qty	Remarks
June 4, 2025	10	Received Items from PS Stockroom dtd May 2025

Assigned Items

Date	Qty	Assigned To	Remarks
September 2, 2025	1	General Use, WRUS	For WRUS TSS
August 22, 2025	1	General Use, WRUS	—
August 12, 2025	1	General Use, WRUS	for office use
July 21, 2025	1	General Use, WRUS	for WRUS TSS use
July 21, 2025	1	General Use, WRUS	for office use
July 3, 2025	1	General Use, WRUS	For office use

Export Excel Button - enables exporting consumable items from the database into an Excel file based on the search result. If the result is blank, then all the consumable items in the database are exported.

Sample Excel file

ID	Specification	Quantity	Unit
2025-060	Self-inking	0 pc	
2025-059	Cutter/Util	0 pcs	
2025-058	Folder, L-tj	5 pcs	
2025-057	Battery- Cfl	0 pcs	
2025-056	Flashdrive,	3 pcs	
2025-055	Folder L-tj	8 pcs	
2025-054	Epson Ink,	6 bottle	
2025-053	Epson ink,	6 bottle	
2025-052	Epson Ink (6 bottle	
2025-051	Epson Ink (5 bottle	
2025-050	Epson Ink (8 bottle	
2025-049	Epson Ink (8 bottle	
2025-048	Epson ink,	8 bottle	
2025-047	Epson Ink,	7 bottle	
2025-046	Rubber bar	0 box	
2025-045	Sticky note	9 pack	
2025-044	Sign pen, E	10 pcs	
2025-043	Glue, all pu	3 bottle	
2025-042	Fastener, p	22 box	
2025-041	CD-R, 700I	10 pcs	
2025-040	Clip, backf	3 box	
2025-039	Clip, backf	12 pieces	
2025-038	Clip, backf	0 box	
2025-037	Clip, backf	2 box	
2025-036	Steno note	35 pcs	
2025-035	WRUS field	47 pcs	

VIII.D ICS/PAR Module

- The ICS/PAR page enables the user to add, edit, and view the ICS/PAR of their accountabilities.

This screenshot shows the ICS / PAR page of the WRUS Portal. The left sidebar shows the user profile "Carabeo, Jasonvon Kym V." and navigation links for Dashboard, Asset Inventory System, and Water Inventory System. The main content area has a search bar and a table titled "ICS / PAR". The table columns are ICS/PAR No., Assigned To, Description, Date, Total Cost, and Action. The table lists several items assigned to Carabeo, Jasonvon Kym V., including a Toshiba Canvio Basic 3.0 Portable External Hard Drive 2 TB, a USB Extension Hub, a Powerbank, a Headset, an Office Chair, a Toshiba Canvio 1TB-External Hard Drive, an EPSON L14150-Printer, and a TP Link M7450-Pocket WiFi. Each row has edit and delete icons.

ICS/PAR No.	Assigned To	Description	Date	Total Cost	Action
2019-226	Carabeo, Jasonvon Kym V.	Toshiba Canvio Basic 3.0 Portable External Hard Drive 2 TB	2025-06-25	P4,245.00	
LV-2022-25	Carabeo, Jasonvon Kym V.	USB Extension Hub, 4 Ports 150CM Plug and Play	2025-06-25	P499.00	
2023-062	Carabeo, Jasonvon Kym V.	Powerbank 50000 MAH, 21 Fast Charging, Dual Output Three Input Bavin	2025-05-20	P3,750.00	
2022-25	Carabeo, Jasonvon Kym V.	Headset with 7.1 Surround Sound Pro Noise Cancelling Headphone with Mic & RGB LED @900	2025-05-14	P900.00	
2021-506	Carabeo, Jasonvon Kym V.	Office Chair Cardic Flexible Ergonomic	2025-05-06	P7,900.00	
2024-091	Carabeo, Jasonvon Kym V.	Toshiba Canvio 1TB-External Hard Drive, 1TB, USB 3.0 Serial No: 42MET0FWTV8H	2024-06-28	P2,999.00	
2024-168	Carabeo, Jasonvon Kym V.	EPSON L14150-Printer W/ Flatbed Designed to Scan and Copy	2024-06-28	P31,999.00	
2024-168	Carabeo, Jasonvon Kym V.	TP Link M7450-Pocket WiFi, LTE-Advanced Serial No: 22370T5000751	2024-06-28	P7,999.00	

- Add ICS Button** - Enables users to add an ICS entry and also attach a PDF. Access is restricted so that each user can view only their own ICS records, except for administrators who have full visibility.

This screenshot shows the "Add ICS Entry" dialog box overlaid on the ICS / PAR page. The dialog has tabs for "Item Details" and "Assignment Details". The "Item Details" tab is active, showing fields for ICS/PAR No. (empty), Serial Number (empty), Description (empty), Quantity (empty), Unit of Measurement (empty), and Unit Cost (P) (P0.00). The "Assignment Details" tab is also visible. At the bottom right of the dialog are "Save ICS" and "Cancel" buttons. The background ICS / PAR page shows the same list of items as the previous screenshot.

- Edit ICS Button** - Enables users to edit their own ICS entry. The Delete button can be activated upon request of the user.

WRUS Portal

ICS / PAR

Carabeo, Josonvon Kym V.

- Dashboard
- Asset Inventory System
- Water Inventory System

Search Enter item name

ICS/PAR No.	Assigned To
2019-226	Carabeo, J Kym V.
LV-2022-25	Carabeo, J Kym V.
2023-062	Carabeo, J Kym V.
2022-25	Carabeo, J Kym V.
2021-506	Carabeo, J Kym V.
2024-091	Carabeo, Josonvon Kym V.
2024-168	Carabeo, Josonvon Kym V.
2024-168	Carabeo, Josonvon Kym V.

Edit ICS Entry

Item Details Assignment Details

ICS/PAR No.	Serial Number
2019-226	494DT999THGG
Description	
Toshiba Canvio Basic 3.0 Portable External Hard Drive 2 TB	
Quantity	Unit of Measurement
1	unit
Unit Cost (P)	
P4,245.00	

Update ICS **Delete** **Cancel**

Date	Total Cost	Action
2025-06-25	P4,245.00	[Edit] [View]
2025-06-25	P499.00	[Edit] [View]
2025-05-20	P3,750.00	[Edit] [View]
2025-05-14	P900.00	[Edit] [View]
2025-05-06	P7,900.00	[Edit] [View]
2024-06-28	P2,999.00	[Edit] [View]
2024-06-28	P31,999.00	[Edit] [View]
2024-06-28	P7,999.00	[Edit] [View]

Previous 1 Next

- The View PDF Button - enables the user to view their own PDF attachment.

WRUS Portal 1751318837516_ITR-2025-06-0090.pdf 1751318837516_ITR-2025-06-0090.pdf

vkffjcpptpnbidjnubqx.supabase.co/storage/v1/object/public/ics-files/1751318837516_ITR-2025-06-0090.pdf

1 / 1 100% |

INVENTORY TRANSFER REPORT

Entity Name: DENR-NCR
From Accountable Officer/Agency/Fund Cluster: HERMINILDA S. BALDEMOR
To Accountable Officer/Agency/Fund Cluster: JOSONVON KYM V. CARABEO
Transfer Type (Check one only): Donation Rerassigment Resignation Leave of Absence

Date Acquired	Item No.	ICS No./Date	Description	Amount	Condition of Inventory
9/21/2022	1	LV-2022-25	USB EXTENSION HUB, 4 PORTS 150CM PLUG AND PLAY @499	499.00	SERVICEABLE
08/06/2019	1	2019-227	TOSHIBA CANVIO BASIC 3.0 PORTABLE EXTERNAL HARD DRIVE 2 TB SN: 494DT999THGG	4,245.00	SERVICEABLE

*****nothing follows*****

Reason's for Transfer:
TRANSFER OF ACCOUNTABILITY DUE TO RETIREMENT

Approved by: *Charisma D. Ruiz* Released/Issued by: *H. Carabeo* Received by: *Josonvon Kym V. Carabeo*

Signature: *Charisma D. Ruiz* Printed Name: *Charisma D. Ruiz* Designation: *Chief/General Services Section* Date: *6-21-2025*

VIII.E Ledger/ICS Summary Page / Item Analytics Module

This page shows the summary of ledger entries and the ICS entries.

- The **first tab** shows the ledger report and ledger analytics.
- Latest Ledger Report** – shows the list of the latest 10 entries in the ledger.

Date Modified	Consumable	Action	Amount	Remarks	Assigned To
September 02, 2025 at 02:08:34 PM	Copy Paper , A4	Assign Item	1	For WRUS TSS	General Use, WRUS .
August 22, 2025 at 09:36:20 AM	Copy Paper , A4	Assign Item	1		General Use, WRUS .
August 22, 2025 at 09:31:28 AM	Copy Paper, Legal	Assign Item	1		General Use, WRUS .
August 18, 2025 at 01:05:18 PM	Tissue, Facial	Assign Item	1		Velasco, Carmen Hizelle G.
August 12, 2025 at 04:45:09 PM	Alcohol, Ethyl	Assign Item	1		Villanueva, Michael V.
August 12, 2025 at 03:22:45 PM	Alcohol, Ethyl	Assign Item	1		Velasco, Carmen Hizelle G.
August 12, 2025 at 03:10:16 PM	Copy Paper , A4	Assign Item	1	for office use	General Use, WRUS .
August 12, 2025 at 08:41:25 AM	Self-inking Stamp	Assign Item	1	For receiving documents	Carabeo, Josonvon Kym V.
August 12, 2025 at 08:41:00 AM	Self-inking Stamp	Add Stock	1	2024 Inventory	—
August 12, 2025 at 08:40:02 AM	Ruler - metal ruler, 12 inches	Assign Item	1	Ruler - Not yet returned	Ocampo, Precious S.

- Consumable Item Analytics Table**- This table analyzes **priority consumable items** to support inventory management. Key features include:
 - Replenishment Check:** Identifies items that need restocking based on stock thresholds.
 - Lifespan Estimation:** Calculates expected lifespan using historical usage data.
 - Consumption Trends:** Provides average monthly usage to guide procurement planning.

Purpose: Helps users make informed decisions on maintaining and replenishing priority consumables.

CID	Specification	Unit	Stock	Consumed (12 mo)	Lifespan	Avg/Month	Days Left	Resupply?	Recommended
2025-007	Copy Paper , A4	ream	3	7	1 mo	2.3	39 days	▲ Yes	28 ream
2025-003	Battery-Dry Cell, Size AA, 4pcs/pack	pack	14	10	4 mo	3.3	126 days	▲ Yes	29 pack
2025-002	Tissue, Facial	box	17	11	5 mo	3.7	139 days	▲ Yes	30 box
2025-008	Copy Paper, Legal	ream	8	2	1 yr	0.7	360 days	No	0 ream
2025-051	Epson Ink 003, black, 65 ml.	bottle	5	1	1 yr 3 mo	0.3	450 days	No	0 bottle
2025-054	Epson Ink , 003, Yellow, 65 ml.	bottle	6	1	1 yr 6 mo	0.3	540 days	No	0 bottle
2025-053	Epson ink , 003, Magenta, 65ml.	bottle	6	1	1 yr 6 mo	0.3	540 days	No	0 bottle
2025-052	Epson Ink 003, Cyan, 65ml.	bottle	6	1	1 yr 6 mo	0.3	540 days	No	0 bottle
2025-001	Alcohol, Ethyl	bottle	36	4	2 yr 3 mo	1.3	810 days	No	0 bottle
2025-050	Epson Ink 001, Yellow, 70 ml	bottle	8	0		0.0		No	0 bottle
2025-049	Epson Ink 001, Magenta, 70 ml	bottle	8	0		0.0		No	0 bottle
2025-048	Epson ink , 001, Cyan, 70 ml.	bottle	8	0		0.0		No	0 bottle
2025-047	Epson Ink , 001, black, 127 ml.	bottle	7	0		0.0		No	0 bottle

- Explanation of each column
 - ❖ **Consumed (12 mo)** = total quantity used in the past rolling 12 months.
 - ❖ **Lifespan** = estimated from usage rates and current stock.
 - ❖ **Avg/Month** = Consumed (12 mo) ÷ 12 months.
 - ❖ **Days Left** = stock ÷ average daily consumption.
 - ❖ **Resupply?** - flags items projected to run out within a year.
 - ❖ **Recommended** - recommended quantity of items to be procured for next year; Formula $((12 * \text{monthly consumption}) + 10\% \text{ allowance})$.
- **Critically Low Consumable Supplies** - Shows the consumable that has a Quantity of less than or equal to 10, regardless of Unit of Measurement.



- The **second tab** displays consumables and ICS entries of the users, with administrators having full access to view all records.

WRUS Portal

LEDGER/ICS SUMMARY

Ledger Report Inventory Summary

Name	Consumable	ICS
Carabeo, Josonvon Kym V.		
Others, (Not WRUS Personnel)		
General Use, WRUS.		

Previous 1 Next

Show Consumable Summary



- shows the consumable items that are assigned to the user in PDF form.

WRUS Portal

LEDGER/ICS SUMMARY

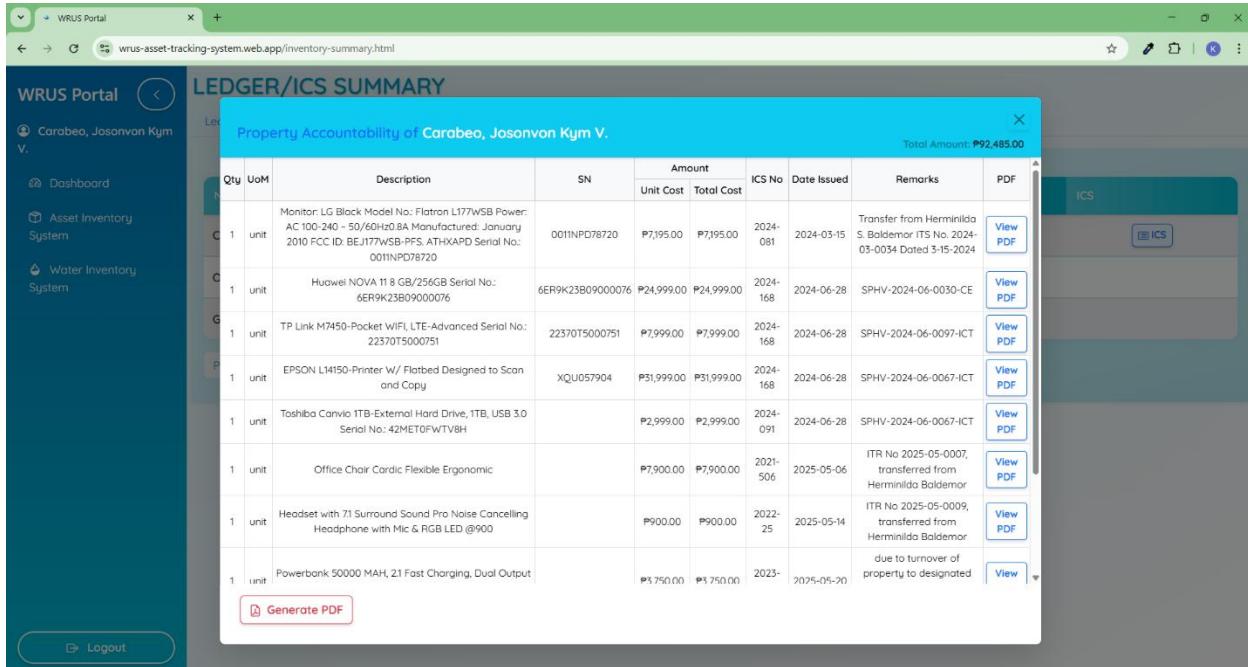
Ledger Report Inventory Summary

Consumable Details

Date	Specification	Unit	Qty	Remarks
August 12, 2025	Self-inking Stamp	pc	1	For notinking documents
July 29, 2025	Flashdrive, 64GB capacity	pos	1	
July 17, 2025	Marker, Fluorescent, Assorted Color	pc	2	-
July 17, 2025	Cuttable/Knife, for general purpose	pos	1	-
July 17, 2025	Folder, L-type, A4, clear, white, plastic	pos	5	-
July 17, 2025	Battery CR-2032	pos	2	-
July 14, 2025	Folder L-type, legal, 12/pack, white, white	pos	5	-
July 14, 2025	Clip, backkit, 10mm, 12/pack/box	box	1	-
July 14, 2025	Sticky notes (Neon color flags)	pack	2	-
July 14, 2025	Sign pen, BSVT, 0.7 mm, black	pos	2	-
July 14, 2025	Clip, backkit, 50 mm	pieces	6	-
July 14, 2025	Clip, backkit, 10mm, 12/pack/box	box	1	-
July 14, 2025	Clip, backkit, 25mm, (12 pieces per box)	box	1	-
July 14, 2025	Ruler - metal ruler, 12 inches	pcs	1	-
July 14, 2025	Ballpoint Pen-retractable ballpoint pen BP-145-F 0.5mm (black)	pos	3	-
July 14, 2025	Correction tape	pos	2	-
July 14, 2025	Tape, transparent, 24mm	pos	1	-
July 14, 2025	Tape, transparent, 48 mm	pos	1	-
July 14, 2025	Tape, masking, 48 mm	pos	1	-

Close

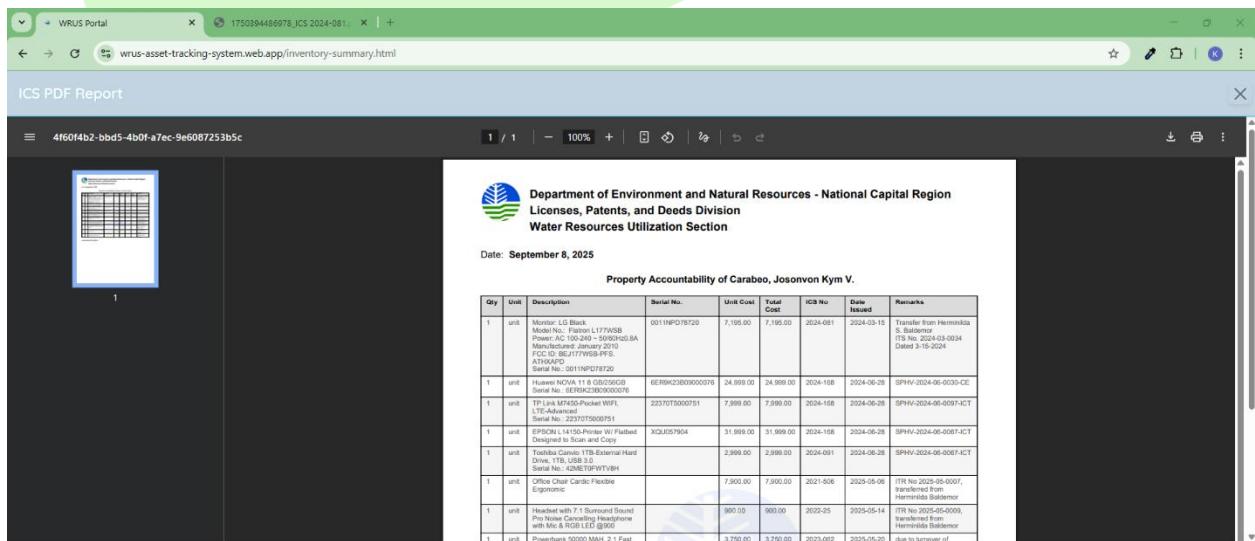
Show ICS Summary  – Displays the ICS accountabilities of users, including the total value of all assigned items.



Qty	UoM	Description	SN	Amount		ICS No	Date Issued	Remarks	PDF
				Unit Cost	Total Cost				
1	unit	Monitor LG Block Model No: Flatron L177wSB Power: AC 100-240 ~ 50/60Hz 0.8A Manufactured: January 2010 FCC ID: BEJ177wSB-PFS, ATHXAPD Serial No: 0011NP078720	0011NP078720	₱7,195.00	₱7,195.00	2024-081	2024-03-15	Transfer from Hemimilda S. Baldemor ITS No. 2024-03-0034 Dated 3-15-2024	View PDF
1	unit	Huawei NOVA 11 8 GB/256GB Serial No.: 6ER9K23B09000076	6ER9K23B09000076	₱24,999.00	₱24,999.00	2024-168	2024-06-28	SPHV-2024-06-0030-CE	View PDF
1	unit	TP Link M7450-Pocket WiFi-LTE-Advanced Serial No: 22370T5000751	22370T5000751	₱7,999.00	₱7,999.00	2024-168	2024-06-28	SPHV-2024-06-0097-ICT	View PDF
1	unit	EPSON L14150-Printer W/ Flatbed Designed to Scan and Copy	XQJU057904	₱31,999.00	₱31,999.00	2024-168	2024-06-28	SPHV-2024-06-0067-ICT	View PDF
1	unit	Toshiba Canvio 1TB External Hard Drive, 1TB, USB 3.0 Serial No: 42MET0FWT8H		₱2,999.00	₱2,999.00	2024-091	2024-06-28	SPHV-2024-06-0067-ICT	View PDF
1	unit	Office Chair Cardic Flexible Ergonomic		₱7,900.00	₱7,900.00	2021-506	2025-05-06	ITR No 2025-05-0007, transferred from Hemimilda Baldemor	View PDF
1	unit	Headset with 7.1 Surround Sound Pro Noise Cancelling Headphone with Mic & RGB LED @900		₱900.00	₱900.00	2022-25	2025-05-14	ITR No 2025-05-0009, transferred from Hemimilda Baldemor	View PDF
1	unit	Powerbank 50000 MAH, 2.1 Fast Charging, Dual Output		₱3,750.00	₱3,750.00	2023-20	2025-05-20	due to turnover of property to designated	View PDF

View PDF  – shows the attached pdf of the ICS entry

Generate PDF  – generates all the ICS entries of the users that can be downloaded and printed.



Qty	Unit	Description	Serial No.	Unit Cost	Total Cost	ICS No	Date Issued	Remarks
1	unit	Monitor LG Block Model No: Flatron L177wSB Power: AC 100-240 ~ 50/60Hz 0.8A Manufactured: January 2010 FCC ID: BEJ177wSB-PFS, ATHXAPD Serial No: 0011NP078720	0011NP078720	7,195.00	7,195.00	2024-081	2024-03-15	Transfer from Hemimilda S. Baldemor ITS No. 2024-03-0034 Dated 3-15-2024
1	unit	Huawei NOVA 11 8 GB/256GB Serial No.: 6ER9K23B09000076	6ER9K23B09000076	₱24,999.00	₱24,999.00	2024-168	2024-06-28	SPHV-2024-06-0030-CE
1	unit	TP Link M7450-Pocket WiFi-LTE-Advanced Serial No: 22370T5000751	22370T5000751	7,999.00	7,999.00	2024-168	2024-06-28	SPHV-2024-06-0097-ICT
1	unit	EPSON L14150-Printer W/ Flatbed Designed to Scan and Copy	XQJU057904	₱31,999.00	₱31,999.00	2024-168	2024-06-28	SPHV-2024-06-0067-ICT
1	unit	Toshiba Canvio 1TB External Hard Drive, 1TB, USB 3.0 Serial No: 42MET0FWT8H		₱2,999.00	₱2,999.00	2024-091	2024-06-28	SPHV-2024-06-0067-ICT
1	unit	Office Chair Cardic Flexible Ergonomic		₱7,900.00	₱7,900.00	2021-506	2025-05-06	ITR No 2025-05-0007, transferred from Hemimilda Baldemor
1	unit	Headset with 7.1 Surround Sound Pro Noise Cancelling Headphone with Mic & RGB LED @900		₱900.00	₱900.00	2022-25	2025-05-14	ITR No 2025-05-0009, transferred from Hemimilda Baldemor
1	unit	Powerbank 50000 MAH, 2.1 Fast		₱3,750.00	₱3,750.00	2023-002	2025-05-20	due to turnover of

VIII.F. Water Permit Module

-This page enables the user to add, edit, and attach a geotagged photo to the Water Permit

The screenshot shows a web-based application titled "WRUS Portal" with a sub-module titled "WATER PERMIT". The left sidebar contains navigation links for "Dashboard", "Asset Inventory System", and "Water Inventory System". The main content area is titled "WATER PERMIT" and displays a table of existing water permits. The columns in the table are "Permit No.", "Permittee", "Diversion Point", "Latitude", "Longitude", and "Action". The table contains four rows of data:

Permit No.	Permittee	Diversion Point	Latitude	Longitude	Action
3044	Mr. Frank Chiongson	Quezon City	14.65278	121.00583	
9540	Tuason Enterprises, Inc.	Marikina	14.65972	121.11222	
9541	Tuason Enterprises, Inc.	Marikina	14.64139	121.11639	
9542	Tuason Enterprises, Inc.	Marikina	14.64778	121.10750	

Add Water Permit - enables the user to add a Water Permit to the database. Also enables the user to attach a PDF file to the Water Permit.

The **first tab** shows the basic info about the Water Permit

The screenshot shows a modal dialog titled "Add Water Permit". The dialog has a blue header bar with the title and a close button. Below the header are two tabs: "Basic Info" (selected) and "Technical Info". The "Basic Info" tab contains the following fields:

- # Permit No. *
- Permittee *
- 📍 Mailing Address (not required)
- 📍 Diversion Point *
- Latitude *
- Longitude *

At the bottom right of the dialog are two buttons: "Close" and "Save Permit".

The **Second Tab** shows the Technical Info about the Water Permit

Add Water Permit

Basic Info Technical Info

Water Source (*not required*) Water Diversion (*not required*)

Flow Rate (L/s) (*not required*) During the Month of (*not required*)

Purpose (e.g. Domestic) Already Visited?

Attach PDF * Is Cancelled?

Choose File No file chosen

Upload a PDF document related to this permit.

Close Save Permit

Edit Water Permit - enables the user to edit the Water Permit.

Edit Water Permit

Basic Info Technical Info

Permit No. Permittee

Mailing Address Diversion Point

Latitude Longitude N E

Cancel Update Permit

Edit Water Permit

Basic Info **Technical Info**

Water Source **Water Diversion**

Flow Rate (L/s) **During the Month of**

Purpose (e.g. Domestic) **Already Visited?**

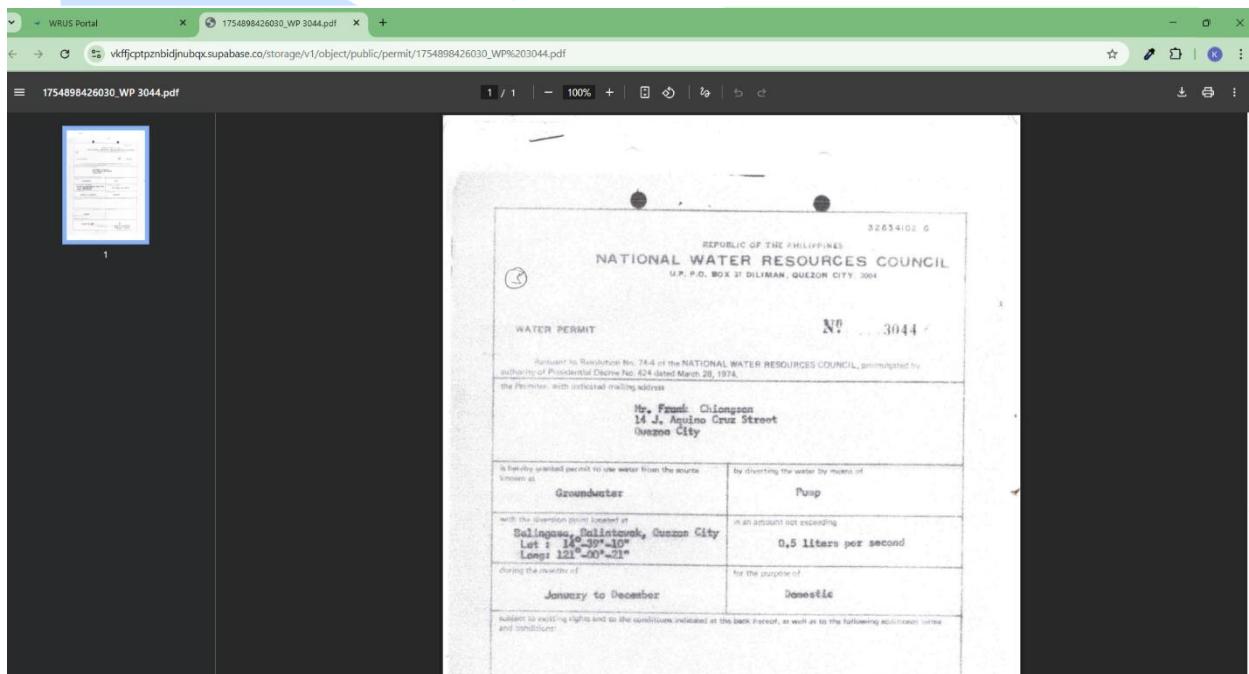
Replace PDF (Optional)

Choose File No file chosen

Upload a new PDF to replace the existing one (optional).

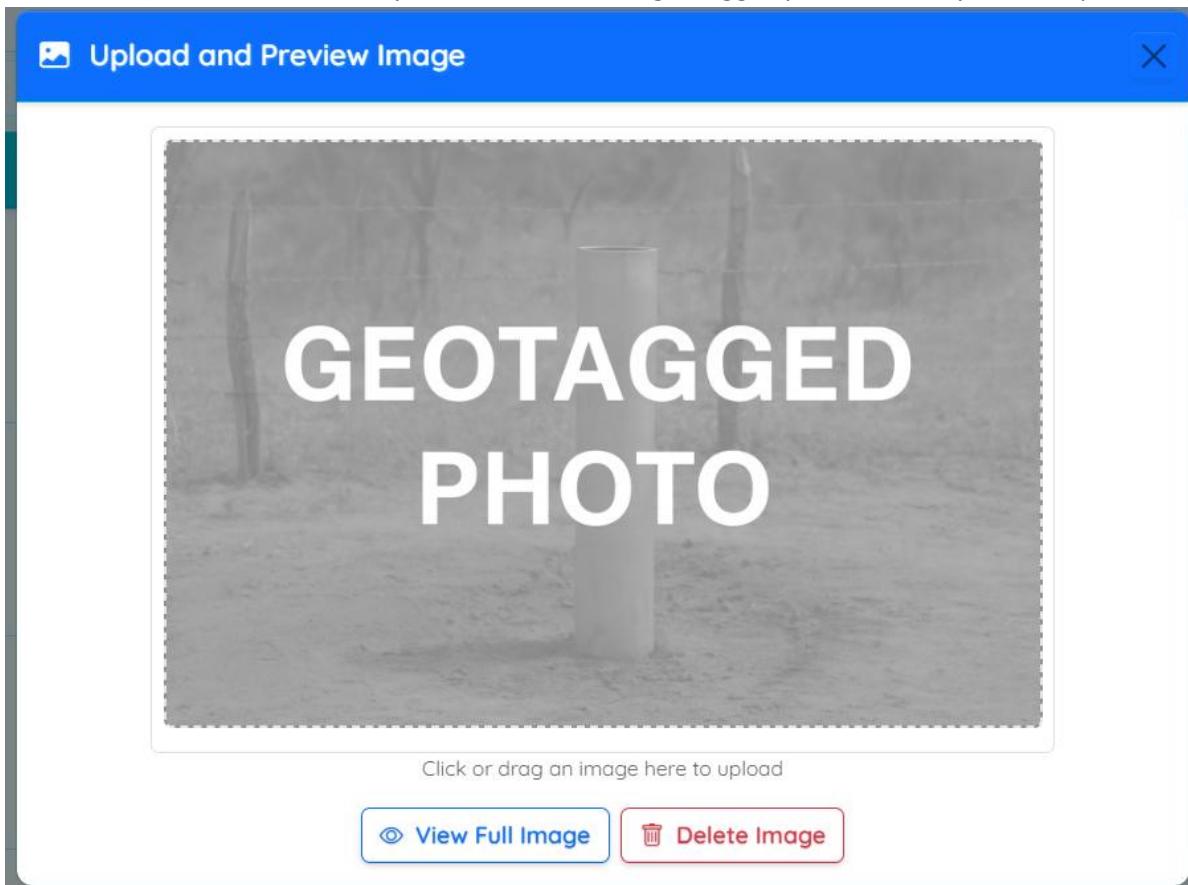
Cancel **Update Permit**

View Water Permit PDF  - enables the user to view the attached Water Permit PDF.

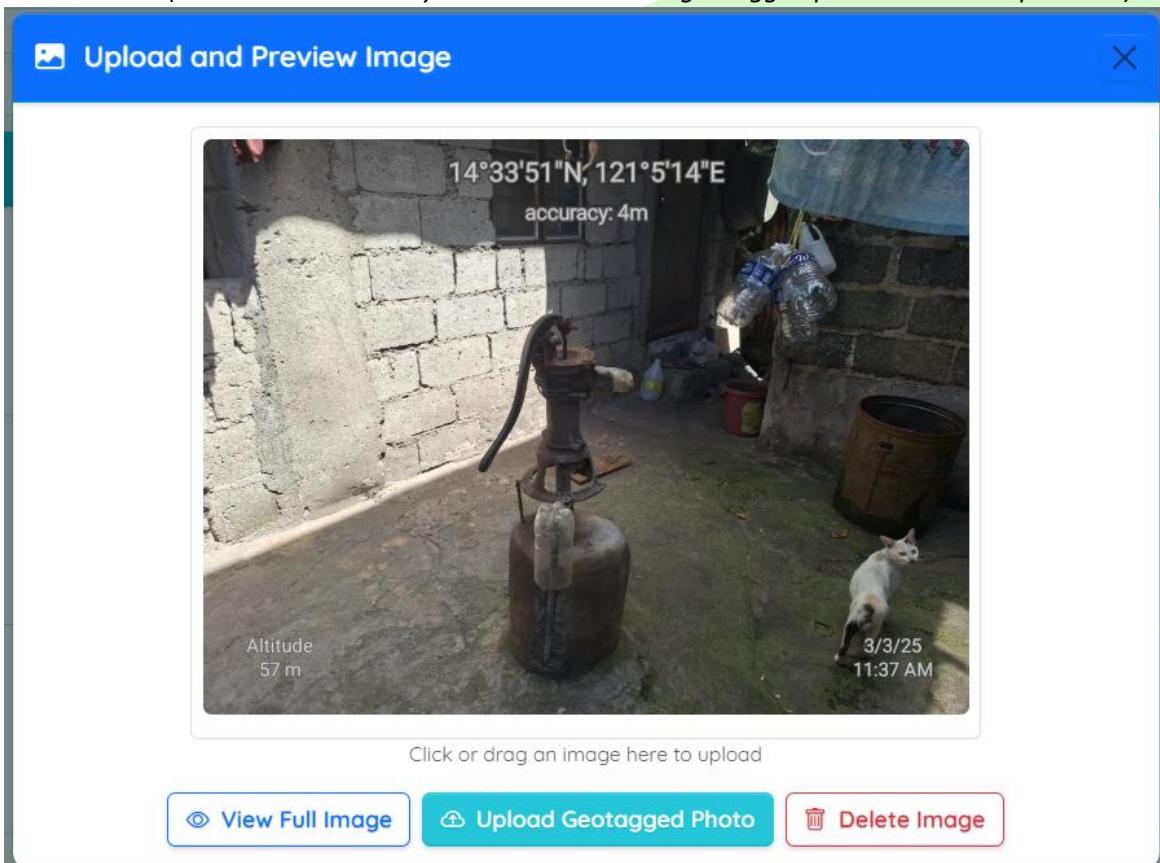


Attached Geotagged Photo  - enables the user to attach a Geotagged Photo to the Water Permit

(Shown below is the system view when the geotagged photo has not yet been uploaded.)



(Shown below is the system view when the geotagged photo has been uploaded.)



Show Filter - The Water Permit Module enables the user to filter the data based on its visitation status.

The screenshot shows the 'WATER PERMIT' section of the WRUS Portal. On the left, there's a sidebar with 'Dashboard', 'Asset Inventory System', and 'Water Inventory System'. The main area has a search bar ('Search Permit No. or City') and a 'Hide Filters' button. Below that, two filter options are shown: 'Show only unvisited permits' (unchecked) and 'Show only visited permits' (checked). A note says '(If checked, shows permittees that are not yet visited. If unchecked, shows all permittees.)'. There's also a 'Rows per page' dropdown set to 10 and an 'Export' button. The table lists three permits:

Permit No	Permittee	Diversion Point	Latitude	Longitude	Action
5674 Visited	Manila Memorial Park	Parañaque	14.45734	121.01742	
020461 Visited	Palmera Homes Inc.	Caloocan	14.74583	121.01028	
16743 Visited	Purefoods Corporation	Marikina	14.62972	121.09375	

A visited Water Permit will receive a badge in the interface.

Export Water Permit to Excel File - This feature enables the user to download the Excel file of the Water Permit. The extracted data will be based on the search query. If no search query is entered, all available data will be included in the extraction.

Shown below is a Sample Excel File

The screenshot shows an Excel spreadsheet titled 'permits-export...'. The table contains 27 rows of data, each representing a water permit. The columns are labeled: Permit No, Permittee, Mailing Adr, Diversion Point, Latitude, Longitude, Water Sou, Water Div, Flow Rate, Purpose, Period of U, Visited, and Cancelled. The data includes various permit numbers, permittees like 'Mr. Frank C', 'JC Realty &', and 'Ronaldo A.', and locations like 'Quezon City', 'Marikina', and 'Metropolit'. The 'Visited' column uses green backgrounds for visited permits, and the 'Cancelled' column has values like 'FALSE' and 'TRUE'.

Permit No	Permittee	Mailing Adr	Diversion Point	Latitude	Longitude	Water Sou	Water Div	Flow Rate	Purpose	Period of U	Visited	Cancelled
30344	Mr. Frank C	Quezon City	14.65278	121.0058							FALSE	FALSE
9540	Tuason Enr	Marikina	14.63972	121.1122							FALSE	FALSE
9541	Tuason Enr	Marikina	14.64139	121.1164							FALSE	FALSE
9542	Tuason Enr	Marikina	14.64778	121.1075							FALSE	FALSE
9543	Tuason Enr	Marikina	14.64694	121.1103							FALSE	FALSE
9544	Tuason Enr	Marikina	14.64944	121.12							FALSE	FALSE
9545	Tuason Enr	Marikina	14.63806	121.1103							FALSE	FALSE
9546	Tuason Enr	Marikina	14.64806	121.1161							FALSE	FALSE
9547	Tuason Enr	Marikina	14.63861	121.1236							FALSE	FALSE
9630	JC Realty &	Marikina	14.65611	121.1136							FALSE	FALSE
10049	Nicolas O.	Marikina	14.65556	121.1192							FALSE	FALSE
9060	Group Dev	Marikina	14.63583	121.0839							FALSE	FALSE
9061	Group Dev	Marikina	14.64361	121.0844							FALSE	FALSE
9538	Tuason Enr	Marikina	14.65139	121.1153							FALSE	FALSE
9539	Tuason Enr	Marikina	14.64472	121.1156							FALSE	FALSE
8081	Metropolit	Marikina	14.66	121.1028							FALSE	FALSE
8082	Metropolit	Marikina	14.62472	121.095							FALSE	FALSE
8083	Metropolit	Marikina	14.63722	121.0994							FALSE	FALSE
8099	San Migue	Marikina	14.63417	121.1119							FALSE	FALSE
8981	Active Rea	Marikina	14.62194	121.1069							FALSE	FALSE
7255	Anaconda	Marikina	14.59361	121.11							FALSE	FALSE
8077	Metropolit	Marikina	14.6275	121.0778							FALSE	FALSE
8079	Metropolit	Marikina	14.65139	121.0936							FALSE	FALSE
6264	Ronaldo A.	Marikina	14.60867	121.1194							FALSE	FALSE
6113	Fortune To	Marikina	14.66167	121.1184							FALSE	FALSE
6114	Fortune To	Marikina	14.66656	121.1136							FALSE	FALSE

VIII.G. Water Users and Sources Module

-Enables the user to add and update Water Inventory data in the database, with the option to attach a geotagged photo in the same manner as in the Water Permit Module.

Owner	Street / Bldg	Barangay	City	Latitude	Longitude	Year	Action
Southbay Homeowners Association, Inc. (Permit #020829) Permittee: 20829	4th District,	Brgy. B.F. Homes,	Parañaque	14.44307	121.04005	2025	
LA FILIPINA DEVELOPMENT CORP. (Permit #4246) Permittee: 4246		Brgy. Oranbo,	Pasig	14.57333	121.06389	2025	
CAPITAL SALES CORP. (Permit #9797) Permittee: 9797	Alfonso Sandoval Avenue,	Brgy. Pinogbuhatan,	Pasig	14.55500	121.08917	2025	
PHILIPPINE STANDARD SANITARY WARES MANUFACTURING CORP. (Permit #4174) Permittee: 4174		Brgy. Santolan,	Pasig	14.61194	121.08639	2025	
LAKEVIEW DEVELOPMENT CORP. (Permit #5345) Permittee: 5345		Brgy. Maybunga,	Pasig	14.55694	121.09083	2025	
Southbay Homeowners Association, Inc. (Permit #010317) Permittee: 10317 Water Source	4th District,	Brgy. B.F. Homes,	Parañaque	14.44319	121.03920	2025	
Southbay Homeowners Association, Inc. (Permit #020831) Permittee: 20831 Water Source	4th District,	Brgy. B.F. Homes,	Parañaque	14.44307	121.04047	2025	

Add Water Inventory Data - enables the user to add a Water Inventory Data to the database.

Permit No.(if permittee)

Owner *

City *

Barangay * (Exclude prefixes like "Brgy.")

Street / Bldg Name *

Type of Water Source *

Source Status

Is Water Source?

Latitude *

Longitude *

Month Inspected

Year Inspected *

Representative

Designation

Phone

Remarks

An additional feature has been integrated to enable linkage between the Permit No. field and the Water Module. As the user types, the system provides a list of suggested results.

Permit No. (if permittee)

<input type="text"/> Q	14
6114 – Fortune Tobacco Corporation	
14814 – Trigold Property Venture, Inc.	
14998 – Dasmarinas Village Association, Inc.	
14455 – Ayala Land, Inc.	
14456 – Ayala Land, Inc.	

Owner *

<input type="text"/>	Southbay Homeowners Association, Inc.
----------------------	---------------------------------------

Barangay * (Exclude prefixes like "Brgy.")

<input type="text"/>	Brgy. B.F. Homes,
----------------------	-------------------

Type of Water Source *

<input type="text"/>	deep well
----------------------	-----------

Edit Water Inventory Data  - enables the user to edit the Water Inventory Data in the Database.

 Edit Water User

Permit No. (if permittee)	Owner *
<input type="text"/> Q 20829	<input type="text"/> Southbay Homeowners Association, Inc.
City *	Barangay * (Exclude prefixes like "Brgy.")
<input type="text"/> Parañaque	<input type="text"/> Brgy. B.F. Homes,
Street / Bldg Name *	Type of Water Source *
<input type="text"/> 4th District,	<input type="text"/> deep well
Source Status	<input type="checkbox"/> Is Water Source?
<input type="text"/> NON OPERATIONAL	
Latitude *	Longitude *
<input type="text"/> 14.443069444444445	<input type="text"/> N <input type="text"/> 121.04005
Month Inspected	Year Inspected *
<input type="text"/> Select Month	<input type="text"/> 2025
Representative *	Designation
<input type="text"/> Nely N. Montanez	<input type="text"/> Village Manager
Phone	Remarks
<input type="text"/> 0905-3231109	<input type="text"/> non operational; for repair
No signature found	
Representative Signature	
<input type="button"/> Cancel	<input style="background-color: yellow; border: 1px solid black; color: green; font-weight: bold; font-size: 10pt; padding: 2px 5px;" type="button" value="Update"/>

Show Filter - The Water Inventory Data Module enables the user to filter the data based on whether it's a Water Source or not

The screenshot shows the WRUS Portal interface with the "Water Permit" module selected. A search bar at the top allows for searching by Permit No. or City. Below the search bar are two filter options: "Show only unvisited permits" (unchecked) and "Show only visited permits" (checked). The table below displays three rows of permit information:

Permit No.	Permittee	Diversion Point	Latitude	Longitude	Action
5674 Visited	Manila Memorial Park	Parañaque	14.45734	121.01742	
020461 Visited	Palmera Homes Inc.	Caloocan	14.74583	121.01028	
16743 Visited	Purefoods Corporation	Marikina	14.62972	121.09375	

If the Water Inventory entry is a **Permittee**, it will receive a green badge **Permittee: 20829** and will be automatically linked to the Water Permit Module. Any changes made to the attached geotagged photo will be reflected in both modules.

If the Water Inventory entry is a **Water Source**, it will receive a blue badge **Water Source**

Export Water Inventory Data to Excel File

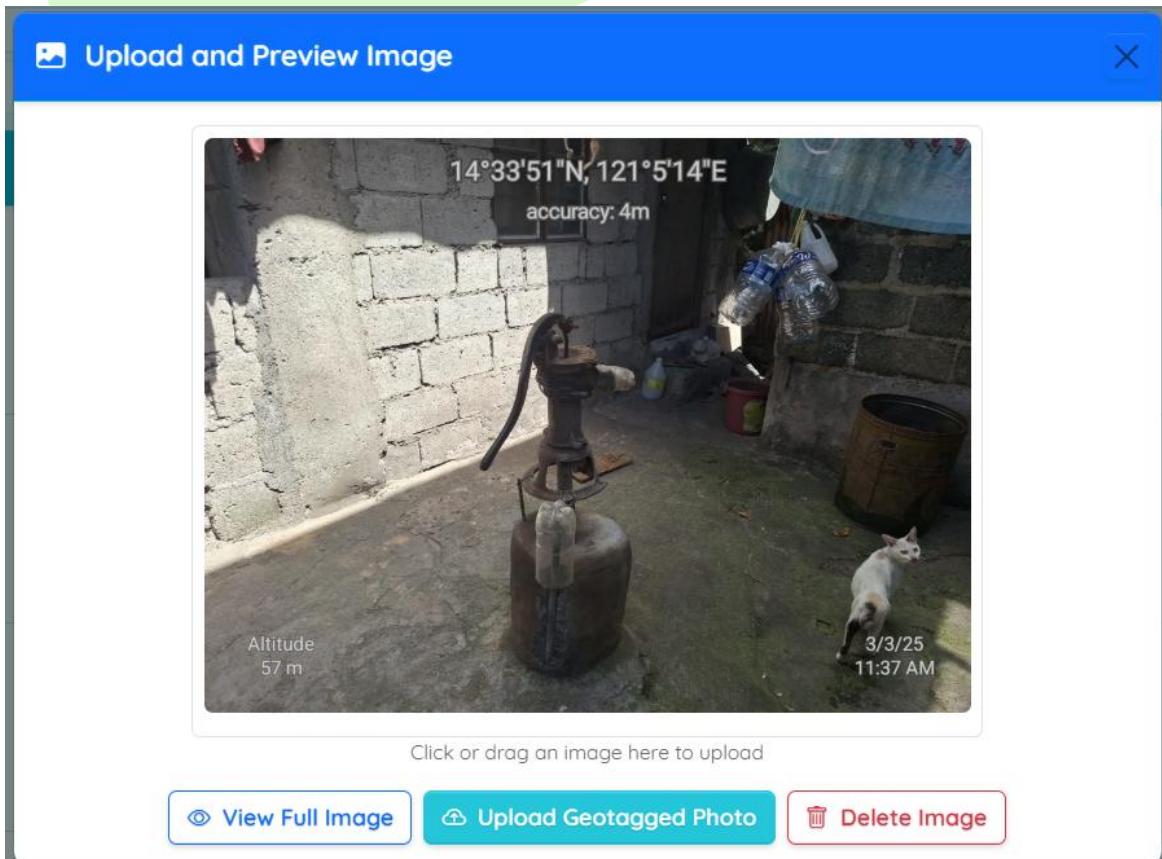


This feature enables the user to download the Excel file of the Water Inventory Data. The extracted data will be based on the search query. If no search query is entered, all available data will be included in the extraction.

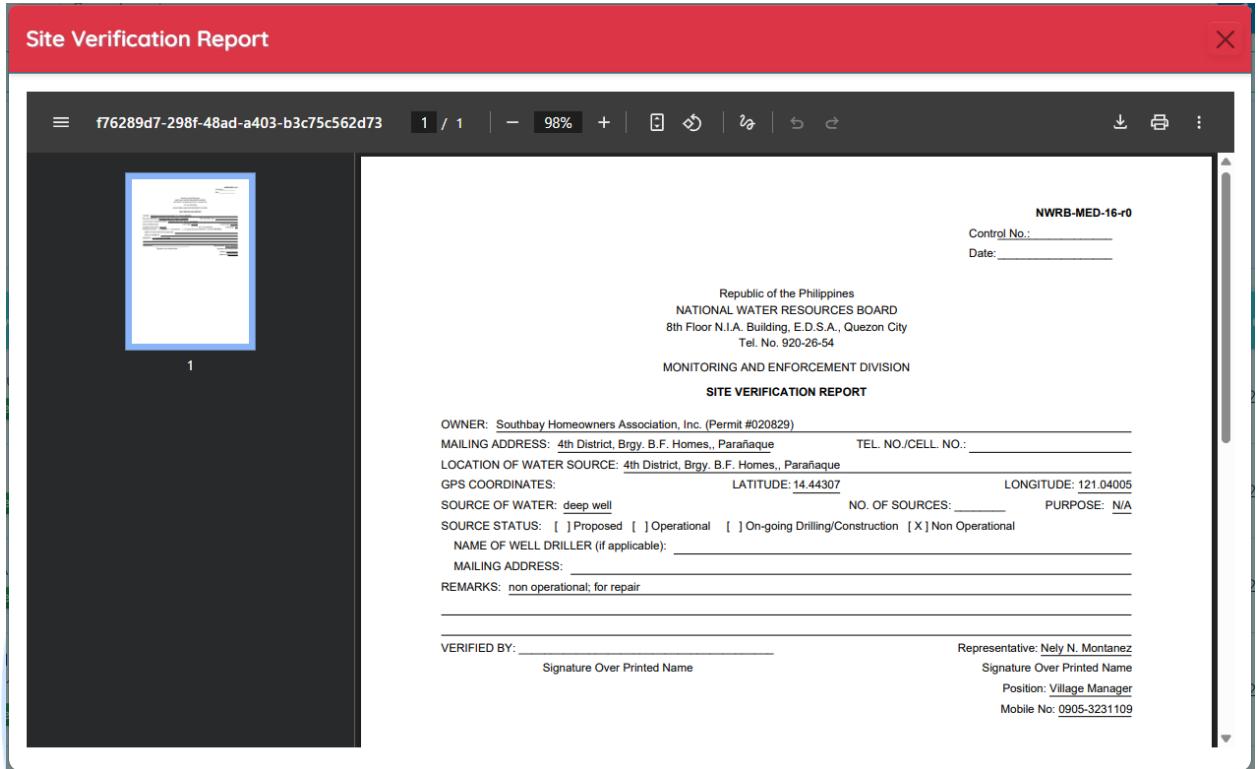
Shown below is a Sample Excel File

The screenshot shows a Microsoft Excel spreadsheet titled "WaterUsersAndSource...". The table contains 27 rows of data, each representing a water source entry. The columns include: Owner, Street, Barangay, City, Latitude, Longitude, Source, Status, Represent, Designatio, Phone, Year_Insp, Month_Ins, and IsWaterSource. The data includes various locations such as Southbay I-4th District, LA FILIPIN/ Brgy. Oran Pasig, CAPITAL S/ Alfonso Sa Brgy. Pinag Pasig, PHILIPPINE Brgy. Santac Pasig, LAKVIEW Brgy. Mayt Pasig, and many others. The "IsWaterSource" column shows values like "No" and "Yes".

Attached Geotagged Photo  - Enables the user to upload and attach a geotagged photo to the Water Inventory Data, using the same procedure as the Water Permit module.



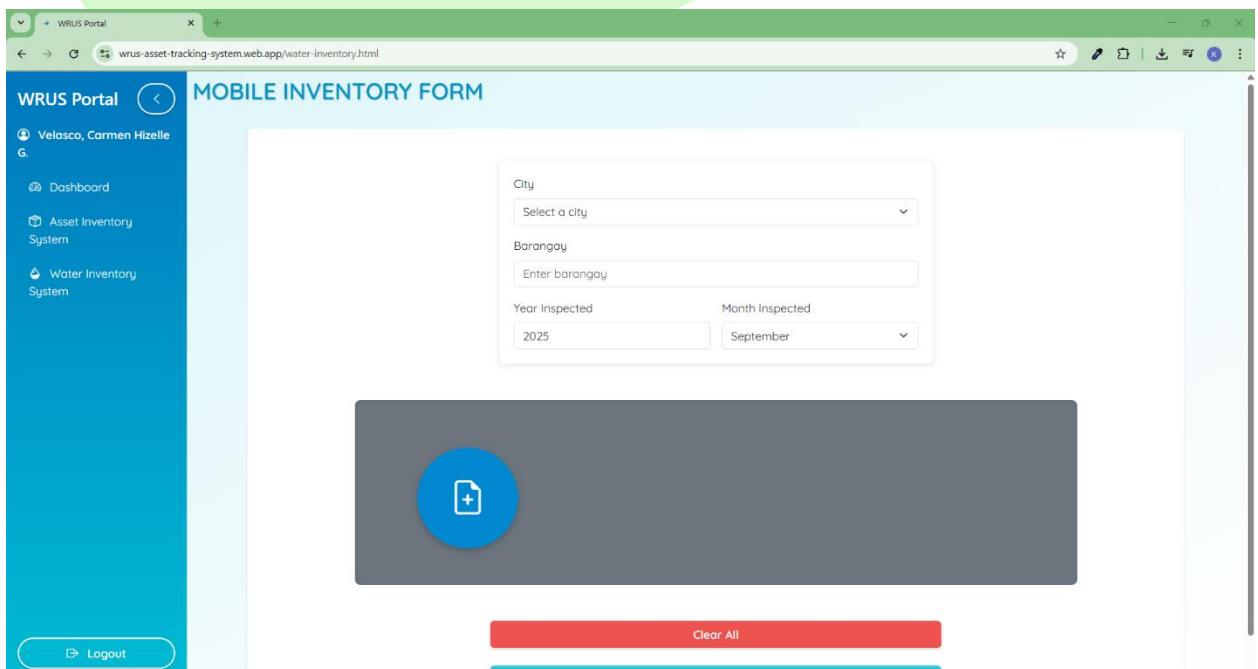
Show Verification Report Form  - allows the user to convert the Water Inventory Data into a PDF form based on the form used by NWRB (NWRB-MED-16-r0). By converting into this form, the user will be able to download the data as a means of verification, complete with an e-signature.



The screenshot shows a PDF document titled "Site Verification Report". At the top right, it says "NWRB-MED-16-r0" and has fields for "Control No." and "Date". The document is from the "Republic of the Philippines, NATIONAL WATER RESOURCES BOARD, 8th Floor N.I.A. Building, E.D.S.A., Quezon City, Tel. No. 920-26-54". It's dated September 2025. The "MONITORING AND ENFORCEMENT DIVISION" section contains a "SITE VERIFICATION REPORT" form. The form includes fields for OWNER (Southbay Homeowners Association, Inc. (Permit #020829)), MAILING ADDRESS (4th District, Brgy. B.F. Homes, Parañaque), GPS COORDINATES (Latitude: 14.44307, Longitude: 121.04005), SOURCE OF WATER (deep well), SOURCE STATUS (Proposed, Operational, On-going Drilling/Construction, Non Operational), NAME OF WELL DRILLER, MAILING ADDRESS, and REMARKS (non operational; for repair). There are also fields for VERIFIED BY and Representative (Nely N. Montanez). At the bottom, there are fields for Signature Over Printed Name, Position (Village Manager), and Mobile No. (0905-3231109).

VIII.H. Mobile Inventory Form Module

- Provides a mobile-friendly interface for conducting field inventories of water users and sources.
- Enables on-site data entry directly into the system, minimizing the need for manual encoding after inspections.



The screenshot shows the "WRUS Portal" mobile inventory form. The left sidebar has links for "Dashboard", "Asset Inventory System", and "Water Inventory System". The main area is titled "MOBILE INVENTORY FORM". It has dropdown menus for "City" (Select a city) and "Barangay" (Enter barangay), and input fields for "Year Inspected" (2025) and "Month Inspected" (September). Below these is a large blue button with a white plus sign. At the bottom, there is a red "Clear All" button.

The first form captures key information so that subsequent forms no longer require repetitive input for fields such as *City*, *Barangay*, *Year Inspected*, and *Month Inspected*. The *Year Inspected* and *Month Inspected* fields are automatically populated based on the current date.

City

Select a city

Barangay

Enter barangay

Year Inspected

2025

Month Inspected

September

Add New Form - Enables the user to add new Water Inventory data, which will be temporarily stored in the browser's local storage. This allows the user to input and accumulate multiple entries while working offline. Fields to be filled up include *Year Inspected*, *Month Inspected*, *Owner*, *Location*, *City*, *Barangay*, *Source of Water*, *Source Status*, *Latitude*, *Longitude*, *Purpose*, *Remarks*, *Representative's name*, *Representative's Designation*, *Representative's Contact Number*, and *Representative's Signature*.

The fields *Year Inspected*, *Month Inspected*, *City*, and *Barangay* are automatically populated once the first form has been completed, thereby streamlining the data-gathering process.

Add Offline Water Inventory Modal

Add Water Inventory

Year Inspected Month Inspected

Owner

Location

City Barangay

Source of Water Source Status

Latitude Longitude

Purpose

Remarks

Representative

Designation Tel / Phone

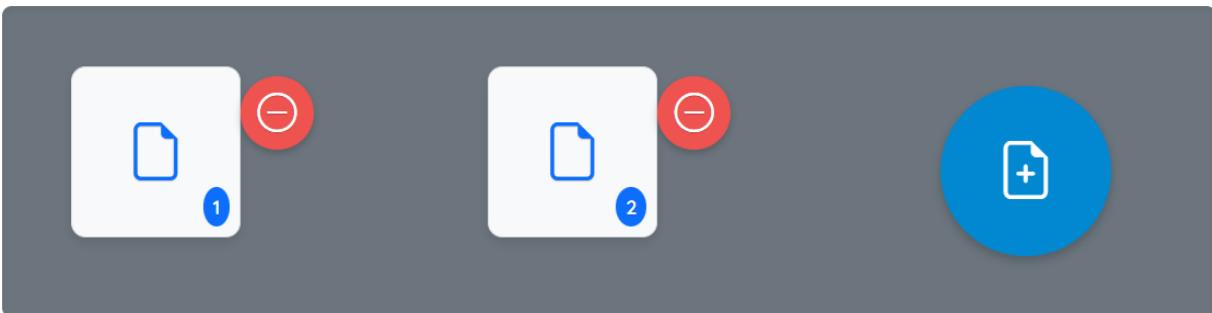
Signature

Save

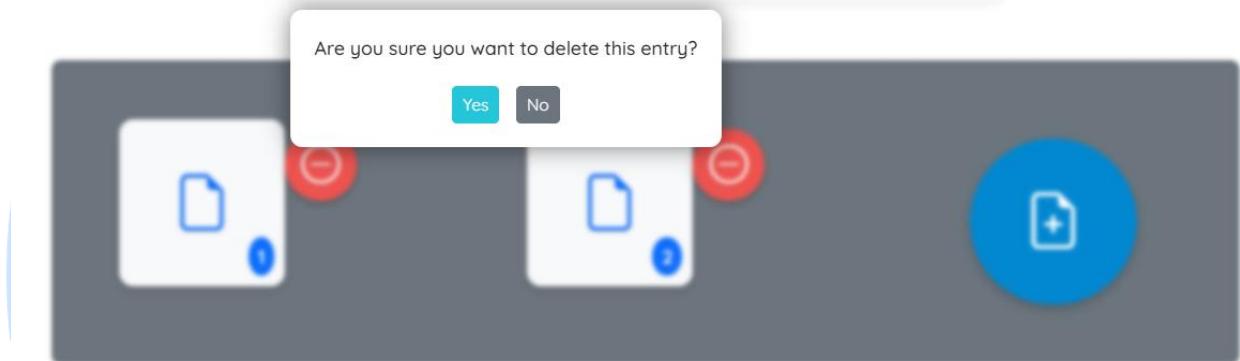
This modal form is used for adding offline water inventory data. It includes fields for basic information like year and month inspected, owner name, location details, source of water, status, coordinates, purpose, remarks, representative, designation, phone number, and a signature area.

The form is presented in a clean, modern design with a blue header and footer. The input fields are white with black borders, and dropdown menus are used for source and status selection. The signature area features a large empty box for drawing and a 'Clear Signature' button.

This illustrates the interface when multiple offline forms have been created.



Pressing the **Delete Entry Button** - Allows the user to delete a single offline entry, for instance, in cases where an error was made.



Or if a user wants a complete reset, the user may click this

Clear All

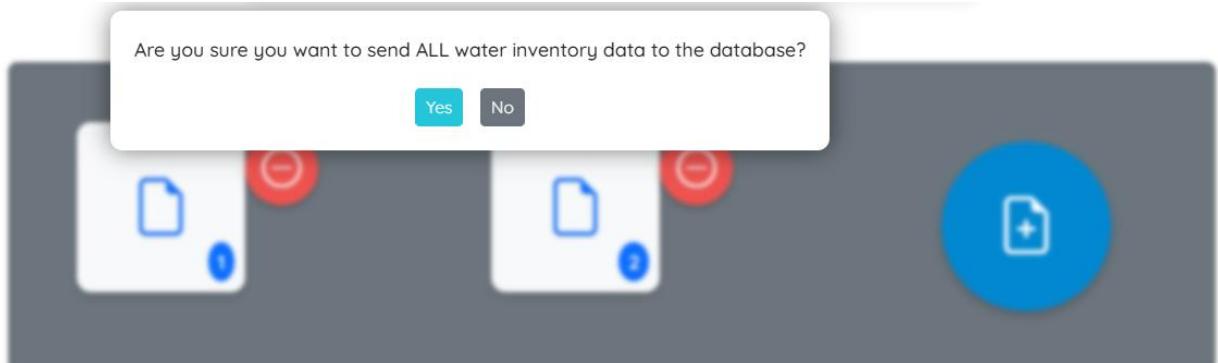
Or if a user wants to update the previous form, he/she may click the icon with a page number.

Subsequently, the user may synchronize and upload the offline data to the database by selecting the *Finalize Entry* button.

Finalize Entry

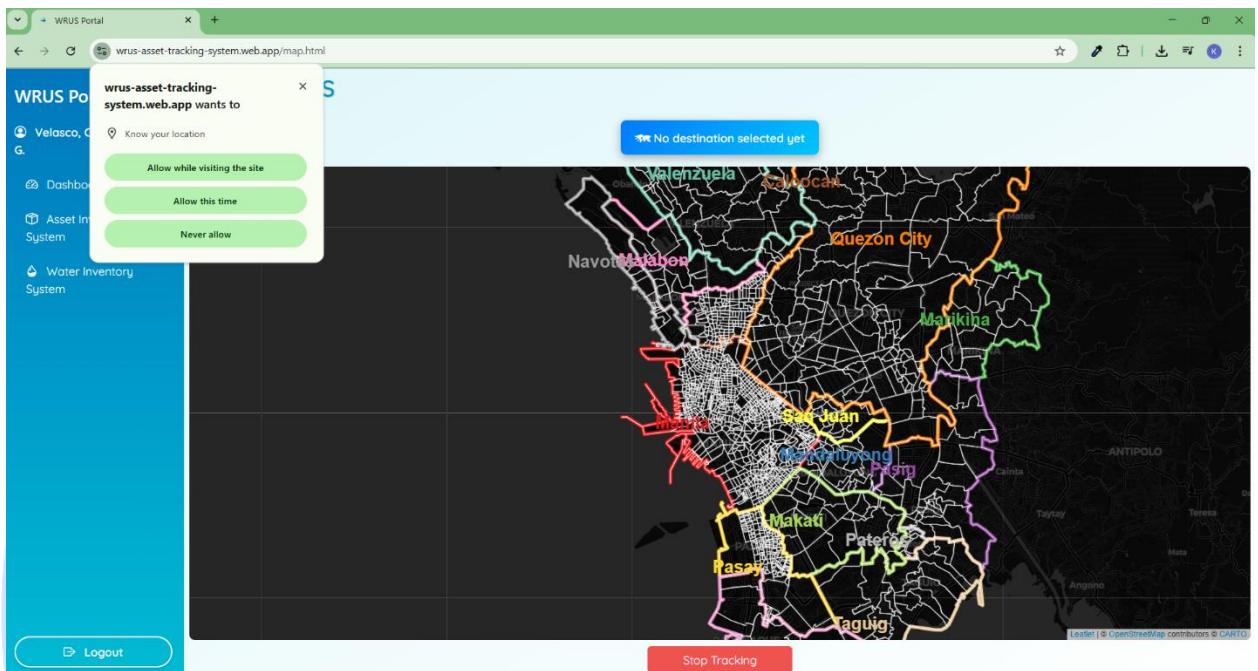
Are you sure you want to send ALL water inventory data to the database?

Yes No

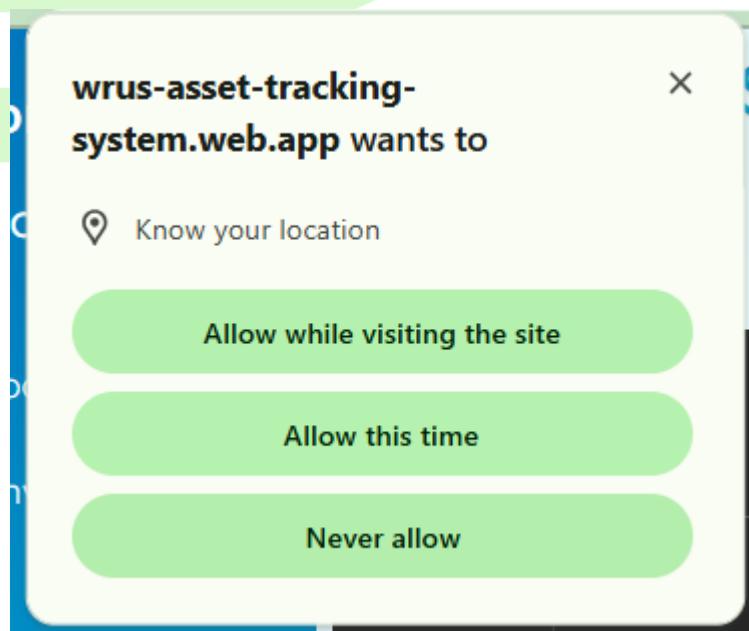


VIII.I. Map Routing Module

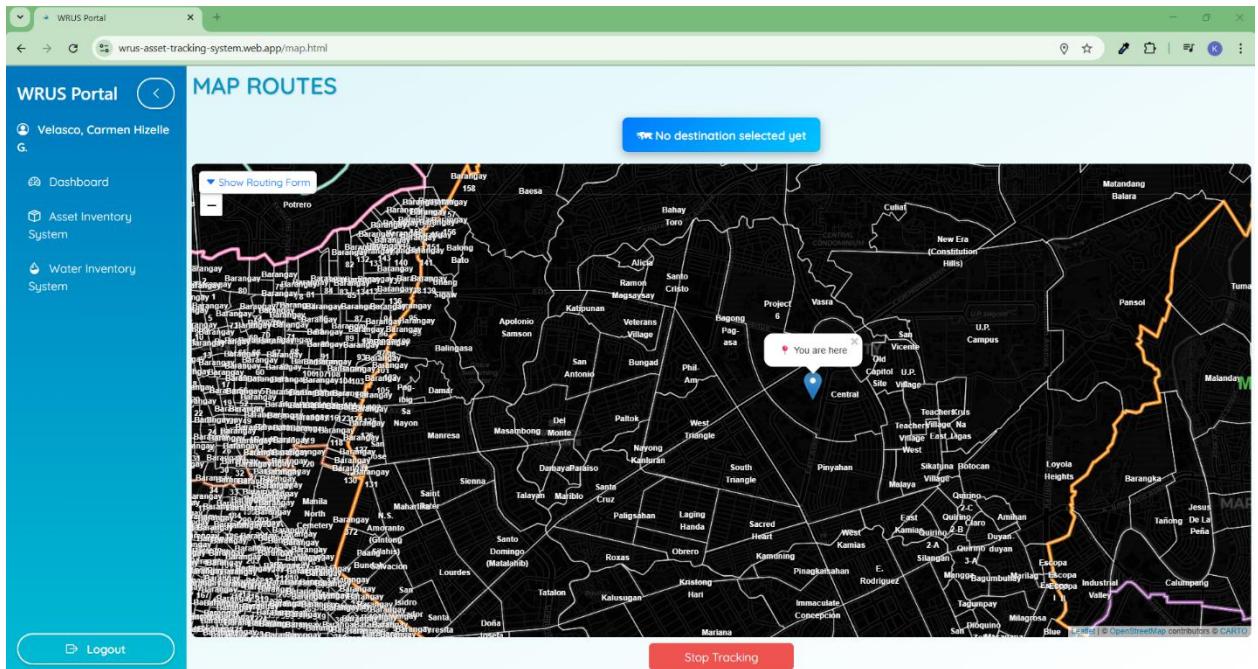
- enables the user to locate the Water Permit location on the system's Web-based Map based on the recorded latitude and longitude in the database.
- This feature is under development, Automatic plotting of the fastest or most efficient route from the user's current location to the selected water permit site.



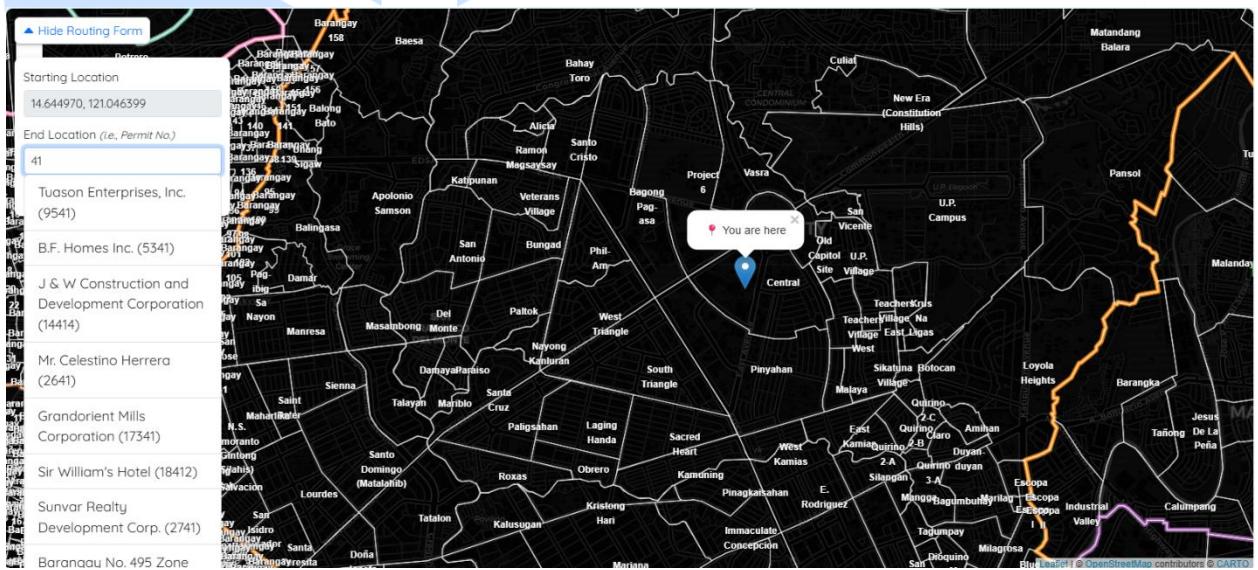
The system will first prompt the user for permission to access their location. Granting this permission allows the system to accurately determine the user's initial location.



Shown below user's initial location



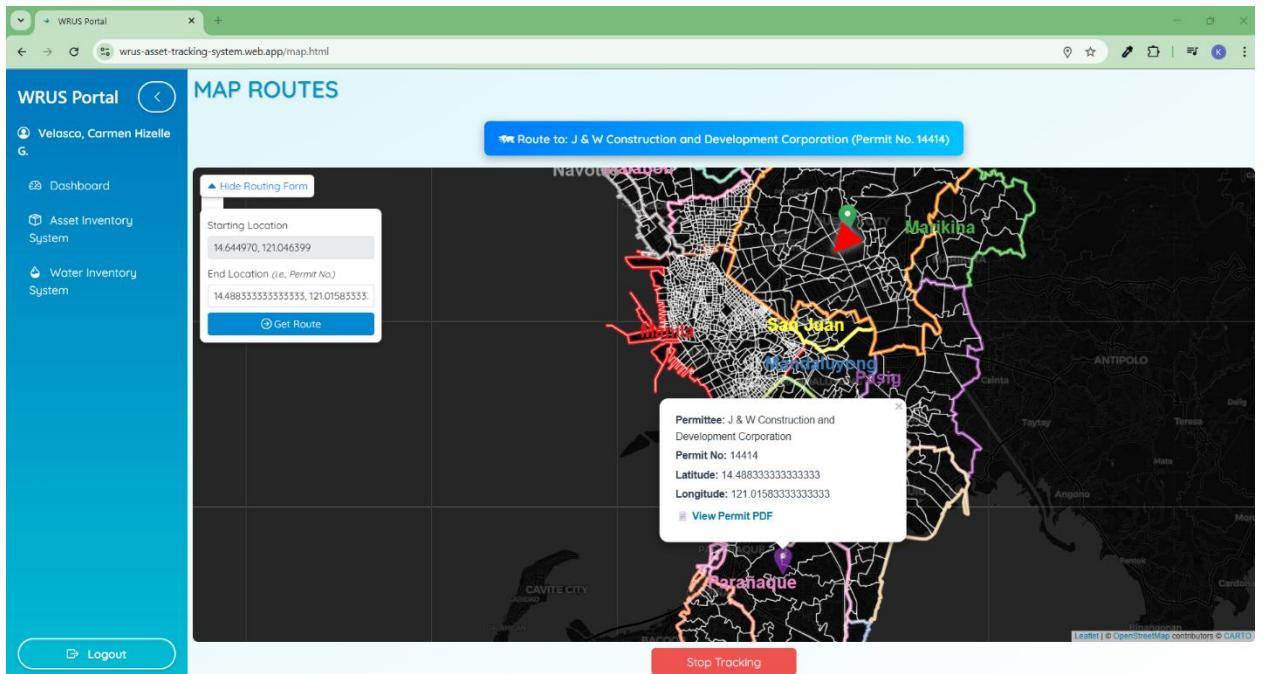
Routing Form ▼ Show Routing Form - enables the user to input the desired end location. The user may enter the Water Permit Number or the Permittee Name. A list of permittees will be shown automatically.



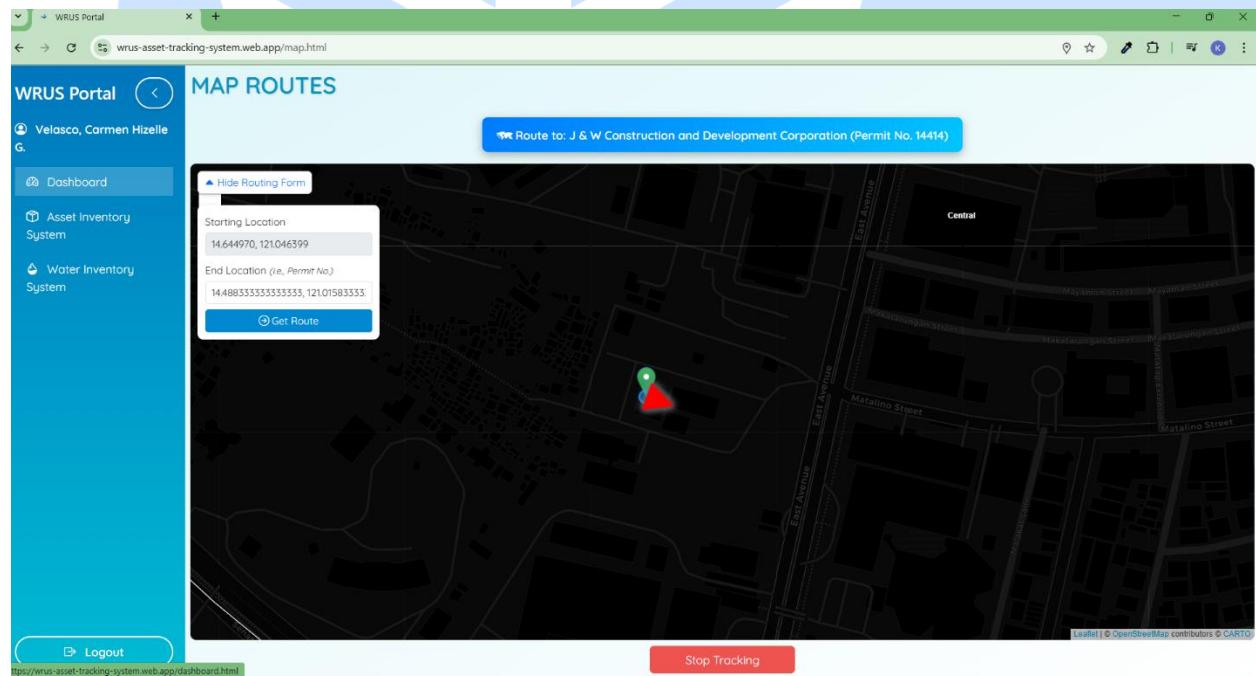
Get Route

Get Route

- Allows the user to plot the end location on the Map.



The initial location will update dynamically based on the user's current location and heading (compass bearing). The arrow will emit a pulse every second to indicate that the user is currently tracking.



The user may stop tracking by pressing **Stop Tracking**

Live tracking stopped.

VIII.J. User Management Module

Enables the administrator to manage the user's information.

The screenshot shows the 'USER MANAGEMENT' section of the WRUS Portal. On the left is a sidebar with icons for Dashboard, User Management (selected), Asset Inventory System, and Water Inventory System. The main area has a search bar and a table with columns: Username, Last Name, First Name, Position, Type, and Action. The table contains the following data:

Username	Last Name	First Name	Position	Type	Action
mjcn	Niez	Mary Joyce	Technical Support Staff	COS	<input checked="" type="checkbox"/> Edit
jadh	Hernando	Jenny Ann	Technical Support Staff	COS	<input checked="" type="checkbox"/> Edit
jsg	Garcia	Jasmin	Technical Support Staff	COS	<input checked="" type="checkbox"/> Edit
molr	Ramos	Melisse Ann	Technical Support Staff	COS	<input checked="" type="checkbox"/> Edit
pso	Oocompo	Precious	Technical Support Staff	COS	<input checked="" type="checkbox"/> Edit
jkrl	Ines	Jonathan Kent	Technical Support Staff	COS	<input checked="" type="checkbox"/> Edit
jkvc	Carabeo	Josonvon Kym	Land Management Inspector	Permanent	<input checked="" type="checkbox"/> Edit
mvv	Villanueva	Michael	Administrator Officer I	Permanent	<input checked="" type="checkbox"/> Edit
hsb	Baldemor	Herminilda	Administrative Officer I	Permanent	<input checked="" type="checkbox"/> Edit
alc	Cepe	Andrew	Engineer II	Permanent	<input checked="" type="checkbox"/> Edit

At the bottom right of the table are navigation buttons: «, 1, 2, ».

Add User Button **+ Add User** - Allows the admin to add a new user

The first tab contains the Account Information related to the system. The administrator may modify the user's username and password, as well as deactivate the account, thereby preventing it from accessing the system.

Add New User

Account Personal Info

Username *

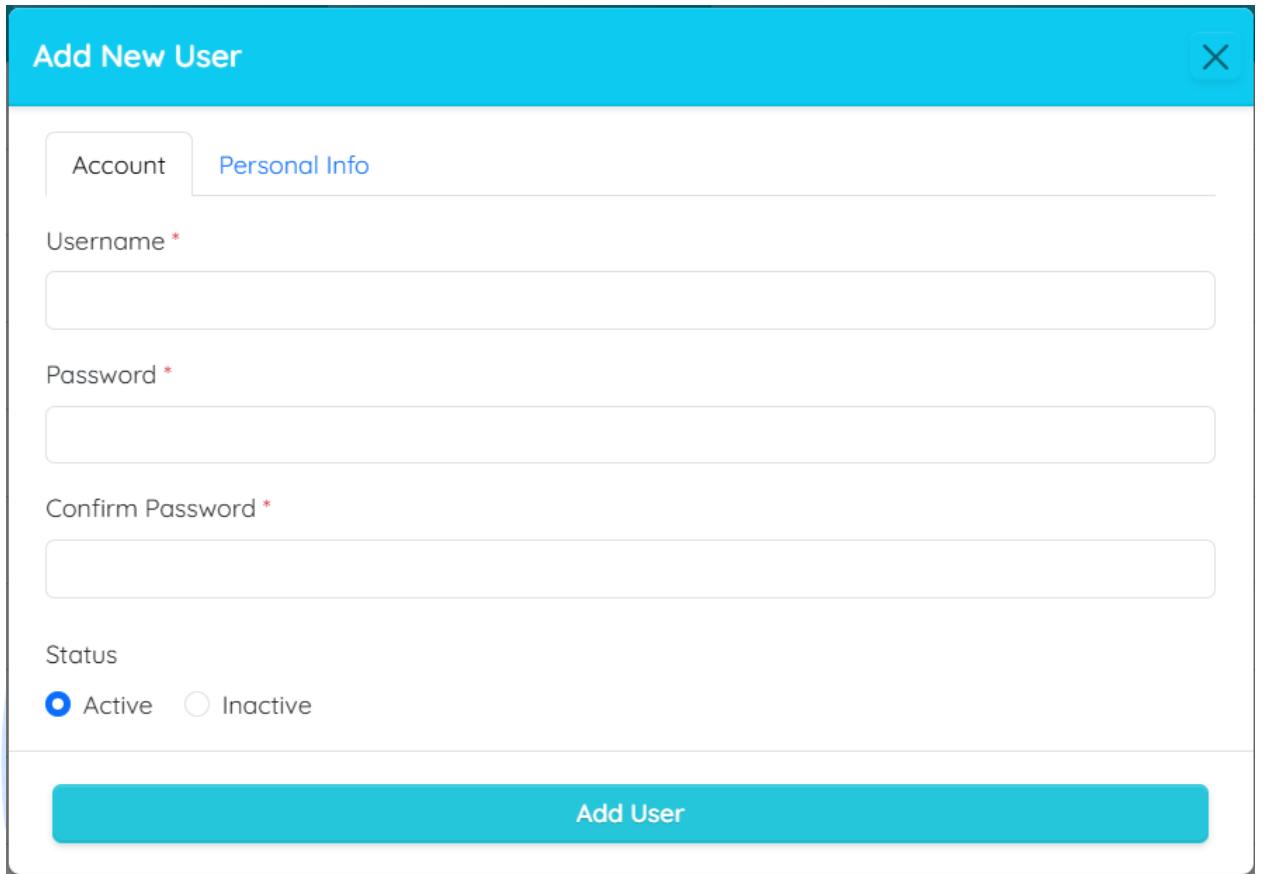
Password *

Confirm Password *

Status

Active Inactive

Add User



The second tab contains the Personal Info of the user based on its employment status and Duty.

Add New User

Account Personal Info

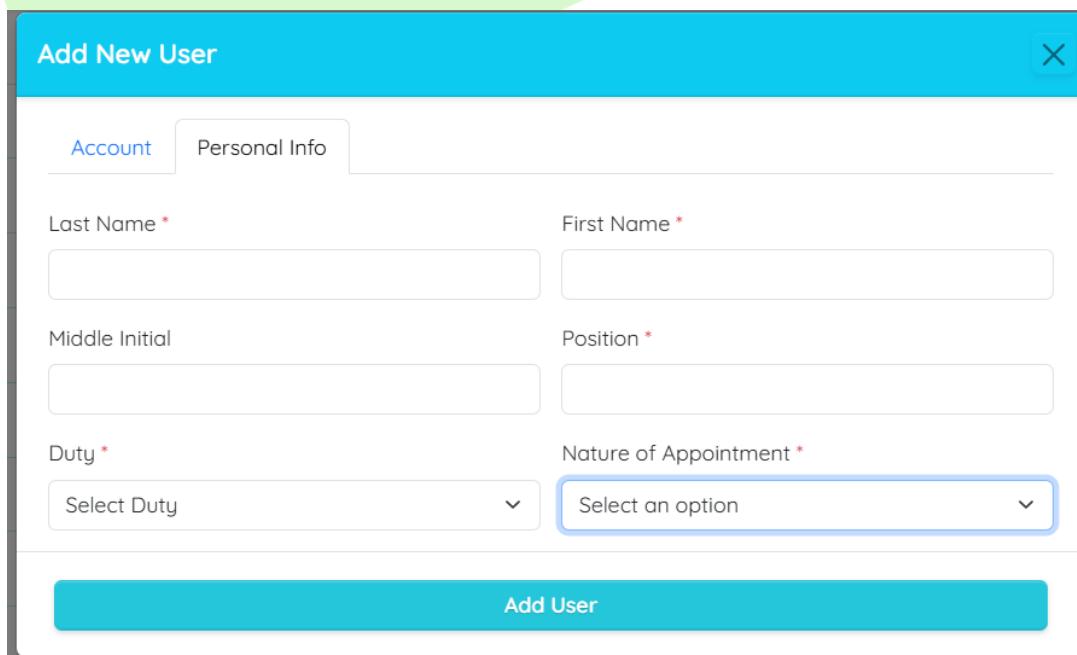
Last Name * First Name *

Middle Initial Position *

Duty * Nature of Appointment *

Select Duty Select an option

Add User



Duty *

Select Duty

- Select Duty
- Supervisor
- Field Work
- Admin Work
- Others

Nature of Appointment *

Select an option

- Select an option
- Permanent
- COS
- Temporary
- Coterminous
- Fixed Term
- Substitute
- Provisional

Edit User [!\[\]\(75f5fa6c53ae03e669fc3d7e4af55ae1_img.jpg\) Edit](#) - allows the administrator to edit the user information.

VIII.J. MEMIS TO DATABASE Module

- Enables the user to upload the MEMIS Excel file directly to the database
- Divided into two types: Permittees and Non-Permittees. The Excel File for Permittees and Non-Permittees have a different format.

Import MEMIS to Database **Import** - Initiates the transfer of Excel data to the database. This action is irreversible; therefore, the administrator must ensure that the Excel file and its fields are accurate before proceeding.

Import data from Excel for year 2017?

Yes No

Import Progress

Processing... Please wait

▲ Do not close or refresh this page

✓ Uploaded row 21: Our Lords Grace Montessori now, owned by SM Prime Holdings
✓ Uploaded row 22: Sacred Heart Novitiate
✓ Uploaded row 23: Manila Water Company, Inc. (1)
✓ Uploaded row 24: Manila Water Company, Inc. (2)
✓ Uploaded row 25: Manila Water Company, Inc. (3)
✓ Uploaded row 26: Manila Water Company, Inc. (4)
✓ Uploaded row 27: Manila Water Company, Inc. (5)
✓ Uploaded row 28: Manila Water Company, Inc. (6)

IX. System Architecture

The WRUS Portal is built using a modular, scalable, and cross-platform web architecture. It combines Firebase services with GitHub version control, Supabase for file storage, and modern deployment options including Render and Docker.

Frontend

- **HTML5, CSS3 (Bootstrap 5)** – For responsive and mobile-friendly layout
- **JavaScript (ES Modules)** – Modern modular JS structure for UI interaction and Firestore communication
- **NPM Live Server** – For local development and real-time testing

Backend / Services

Service	Purpose
Firebase Authentication	Handles secure user login via username/password
Cloud Firestore	NoSQL database for storing users, consumables, ICS/PAR data
Supabase Storage	For storing and managing uploaded PDF files securely
Firebase Hosting (optional)	Used for testing or static deployment of frontend
Render	Used for live public hosting of the system
GitHub	Source code version control and collaboration
Docker	Cross-platform support via Dockerfile for easy deployment

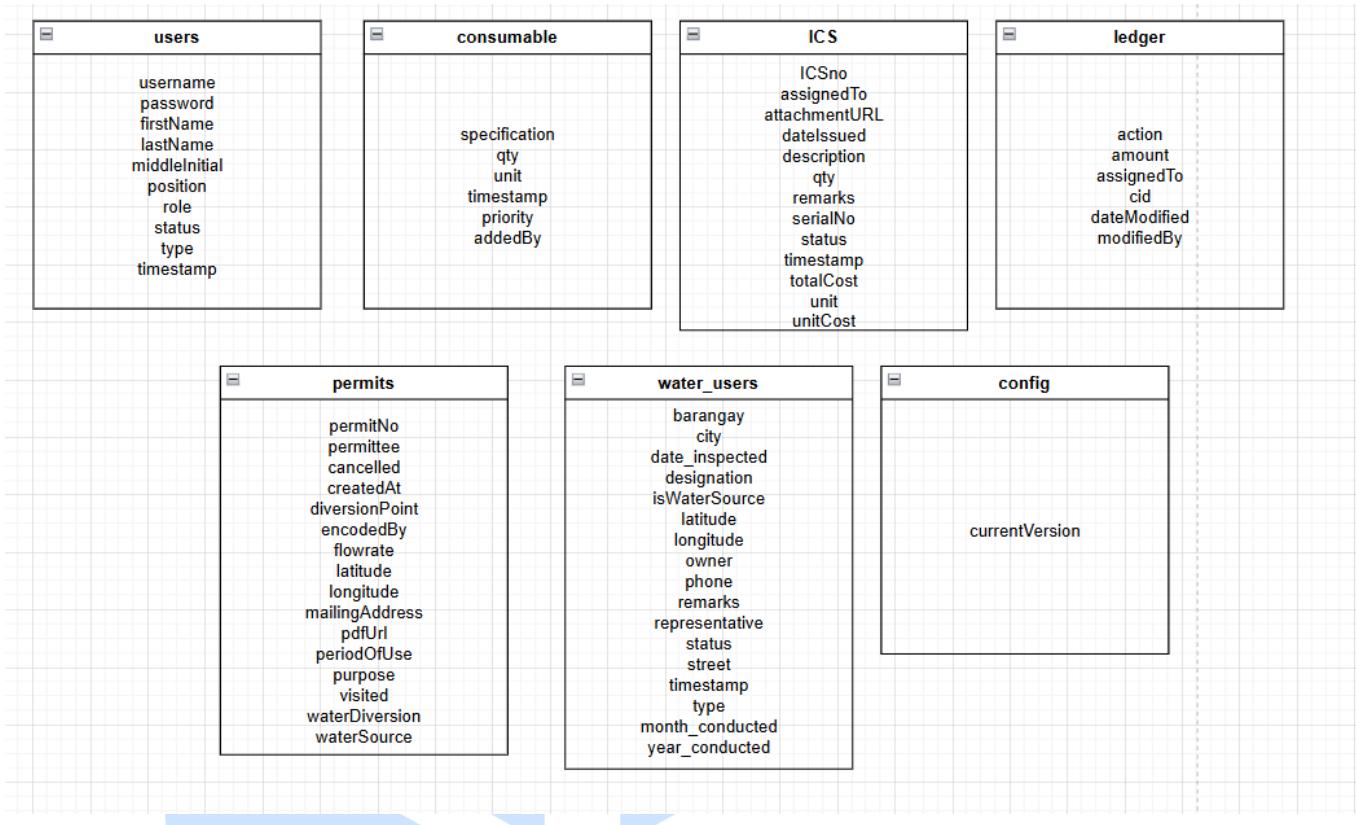
Deployment Flexibility

- The system can be deployed locally using **Live Server** or hosted online via **Render** or **Firebase Hosting**.
- A **Dockerfile** is provided to containerize the application, allowing it to run on any platform (Windows, macOS, Linux) with Docker installed.
- Source code is maintained and versioned using **GitHub**.

X. Technologies Used

- **HTML5** – Markup language for structuring the web interface
- **CSS3 / Bootstrap 5** – For responsive styling and layout
- **JavaScript (ES Modules)** – Modular scripts for frontend logic
- **Cloud Firestore (Firebase)** – Real-time NoSQL database
- **Supabase Storage** – For storing uploaded PDF files
- **Firebase Hosting** – For site deployment
- **Render** – Live hosting for the full system
- **GitHub** – Source code version control and collaboration
- **NPM Live Server** – Local development and testing
- **Docker** – Platform-independent deployment through containerization
- **Leaflet.js** - Interactive mapping library for displaying geospatial data
- **Signature Pad** - Library for capturing and storing digital signatures
- **Visual Studio Code (IDE)** – Integrated development environment for coding, debugging, and project management
- **IndexedDB** – Client-side NoSQL database for storing structured data offline
- **Local Storage** – Browser-based key-value storage for persisting lightweight data
- **Chart.js** – Library for creating interactive and responsive charts
- **ArcGIS** – Used to generate GeoJSON data for city and barangay boundaries
- **ChatGPT** – AI-powered language model used for coding assistance, content refinement, and system documentation

XI. Database Structure



Firestore Database

The primary database for the WRUS Portal is implemented in Google Cloud Firestore, a real-time NoSQL database. It stores structured collections and documents to manage system data efficiently.

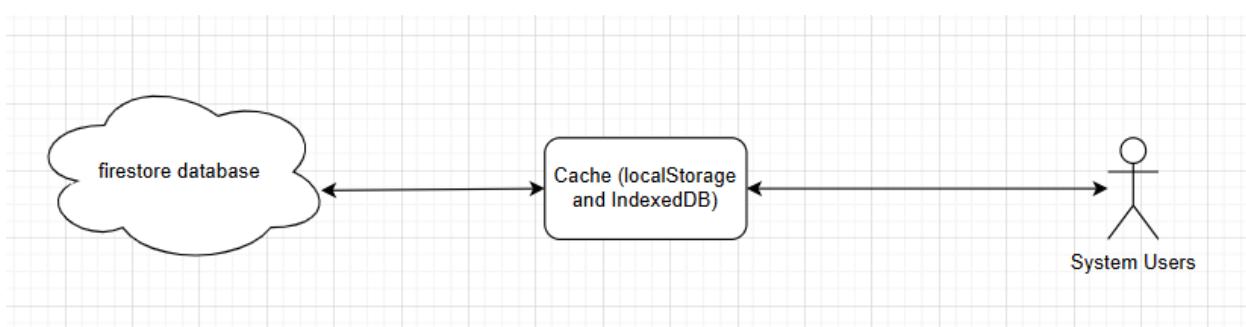
- Collections:
 - **users** → stores account information (username, password, firstName, lastName, middleInitial, position, role, status, type, and timestamp).
 - **consumable** → stores consumable item entries (specification, qty, unit, timestamp, priority, and addedBy).
 - **ICS** → stores ICS/PAR entries (ICSnno, assignedTo, attachmentURL, datelssued, description, qty, remarks, serialNo, status, timestamp, totalCost, unit and, unitCost)
 - **ledger** → Ledger entries for consumable items are stored in the database to provide a reliable audit trail (action, amount, assignedTo, cid, dateModified, and modifiedBy)
 - **permits** → stores the Water Permit entries (permitNo, permittee, cancelled, createdAt, diversionPoint, encodedBy, flowrate, latitude, longitude, mailingAddress, pdfUrl, periodOfUse, purpose, visited, waterDiversion, and waterSource)

- water_users → stores the water inventory data (barangay, city, date_inspected, designation, isWaterSource, latitude, longitude, owner, phone, remarks, representative, status, street, timestamp, type, month_conducted and year_conducted)
- config → for storing the current version of the system (currentVersion)



XII. Data Flow Diagram

XII.A. Level 0 DFD



XII.A.1. Level 0 DFD – WRUS Portal (High-Level System Overview)

External Entities:

- **System Users** – employees or staff interacting with the portal.

Processes:

1. **User Interaction** – Users log in, submit forms, update inventory and permits, and view maps and reports.
2. **Local Storage & IndexedDB (Cache Layer)** – Handles temporary storage of data and large objects (e.g., e-signatures) for faster access and offline use.
3. **Firestore Database (Cloud Storage)** – Main data repository for permits, consumables, user accounts, water inventory, and reports.
4. **Reporting & Visualization** – Aggregates Firestore data for dashboards, exports, and analytics.

Data Stores:

- **Local Storage** – small temporary data, offline forms, cached Firestore data.
- **IndexedDB** – large objects like electronic signatures.
- **Firestore Collections** – structured records (users, permits, consumables, inventory, etc.).

Data Flow:

1. Users interact with the **System UI**.
2. The system fetches data from **Local Storage/IndexedDB** (if available) before querying **Firestore**.
3. Offline forms are stored locally, then synchronized with Firestore once connected.

4. Firestore serves as the **source of truth**, updating both **Cache** and **Reporting modules**.
5. Reports and dashboards are generated from Firestore data and shared with management.

XII.A.2 Local Storage (Cache) and IndexedDB

To optimize performance and enable offline functionality, the system leverages both Local Storage and IndexedDB for client-side data handling before synchronizing with Firestore.

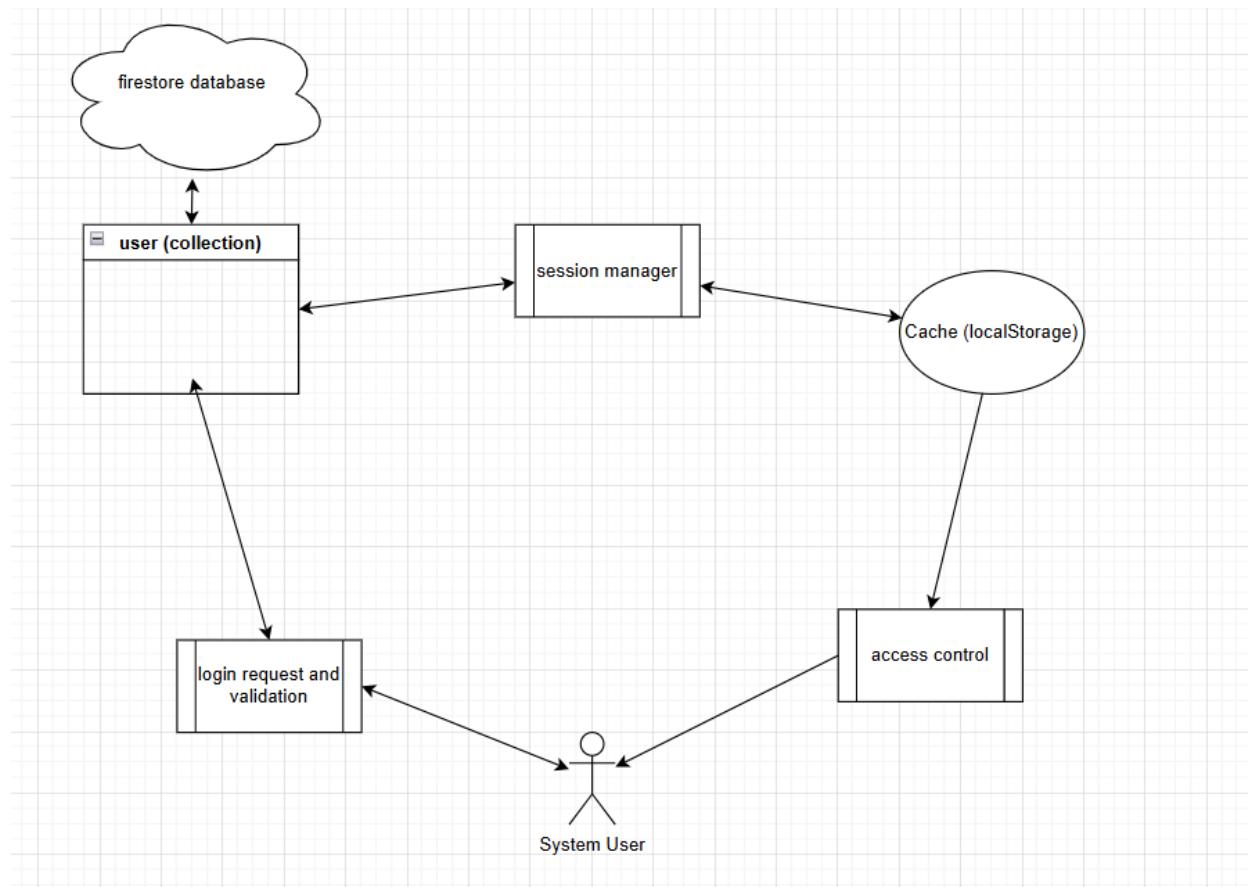
Data Flow:

1. Firestore Database → Cache
 - Upon system initialization, all relevant data from Firestore is loaded into the Local Storage cache.
 - IndexedDB is also used for larger objects, such as electronic signatures, which cannot be efficiently stored in Local Storage.
2. Cache → System Users
 - When a user fetches data, the system retrieves it directly from the cache instead of querying Firestore, ensuring faster response times.
 - Offline forms created by users are stored in Local Storage until the user finalizes the entry.
 - Finalized entries are then uploaded to Firestore once connectivity is available.

Advantages:

- Faster response times by minimizing direct Firestore reads.
- Supports offline capability, allowing users to continue working without internet access.
- Efficient storage of large objects (e.g., e-signatures) through IndexedDB.
- Helps circumvent Firestore's 50,000 document read limit, since most data retrieval operations are served from the cache rather than directly from Firestore.

XII.B.1 Level 1 DFD – Logging (Authentication & Session Management)



External Entities

- **System User** – provides login credentials (username, password).

Processes

1. **Login Request & Validation** – user submits credentials, validated against Firestore User Collection.
2. **Session Manager** – creates and manages session-like data using Local Storage.
3. **Access Control** – grants or denies access to system features based on login status.

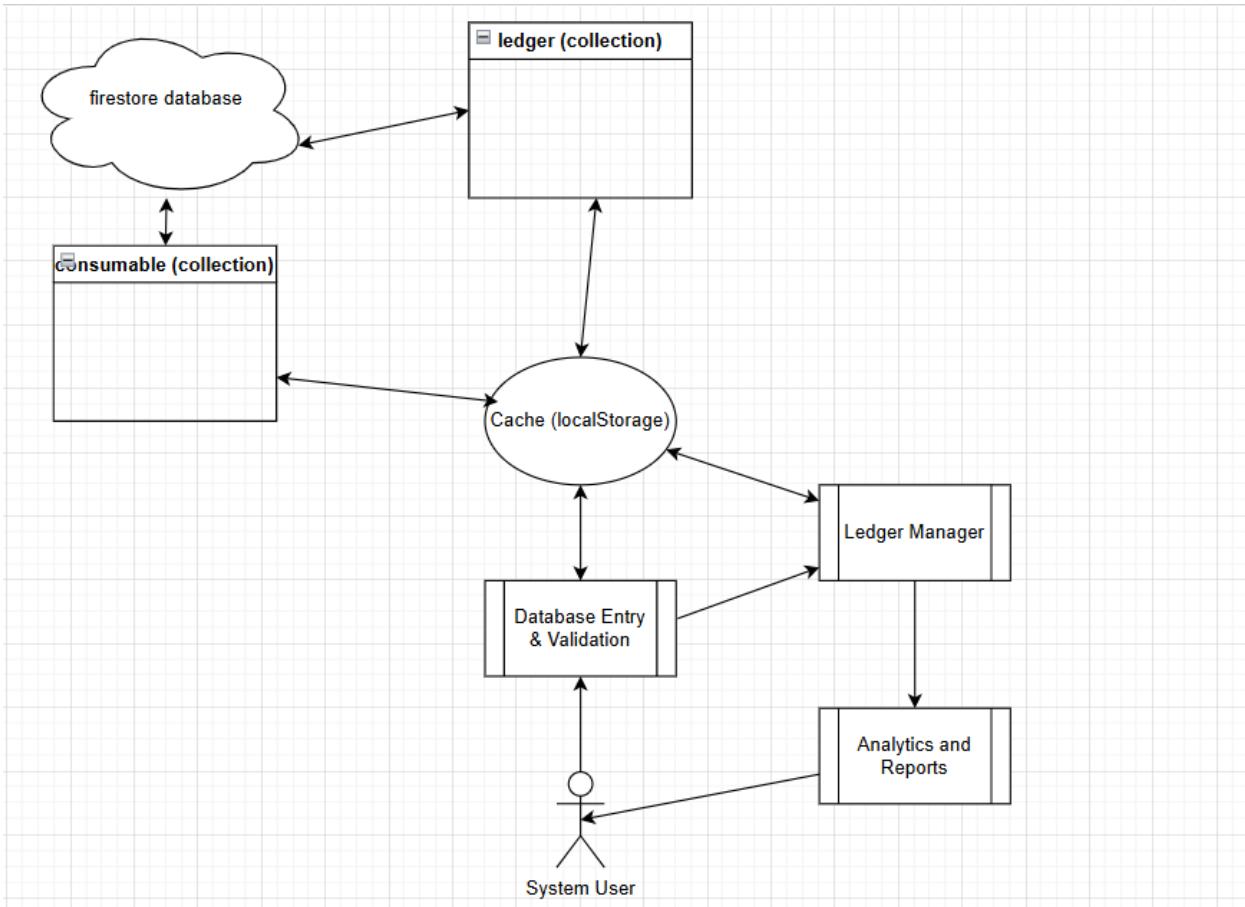
Data Stores

- **Firestore (User Collection)** – stores official user accounts (username, password hash, role, status).
- **Local Storage** – stores temporary session data (e.g., logged-in username, role, login flag).

Data Flow

1. **System User → Login Request & Validation:** enters username and password.
2. **Login Request & Validation → Firestore (User Collection):** queries Firestore to verify credentials.
3. **Firestore (User Collection) → Login Request & Validation:** returns authentication result.
4. **Login Request & Validation → Session Manager → Local Storage:** stores session details (e.g., active user, role).
5. **Local Storage → Access Control → System User:** provides access to system pages/modules if authenticated.





XII.B.2. Level 1 DFD – Consumable and Analytics Module

External Entities

- System User – Staff entering, issuing, or returning consumable items.

Processes

1. Consumable Data Entry – User adds or updates consumable items.
2. Ledger Manager – Records all transactions (issuance, returns, adjustments) for audit trail.
3. Sync Manager – Ensures consumable and ledger data are properly synchronized with Firestore.
4. Firestore Database – Cloud repository storing consumable items and ledger records.
5. Analytics & Reports – Uses ledger data and consumable records to generate summaries, dashboards, and insights.

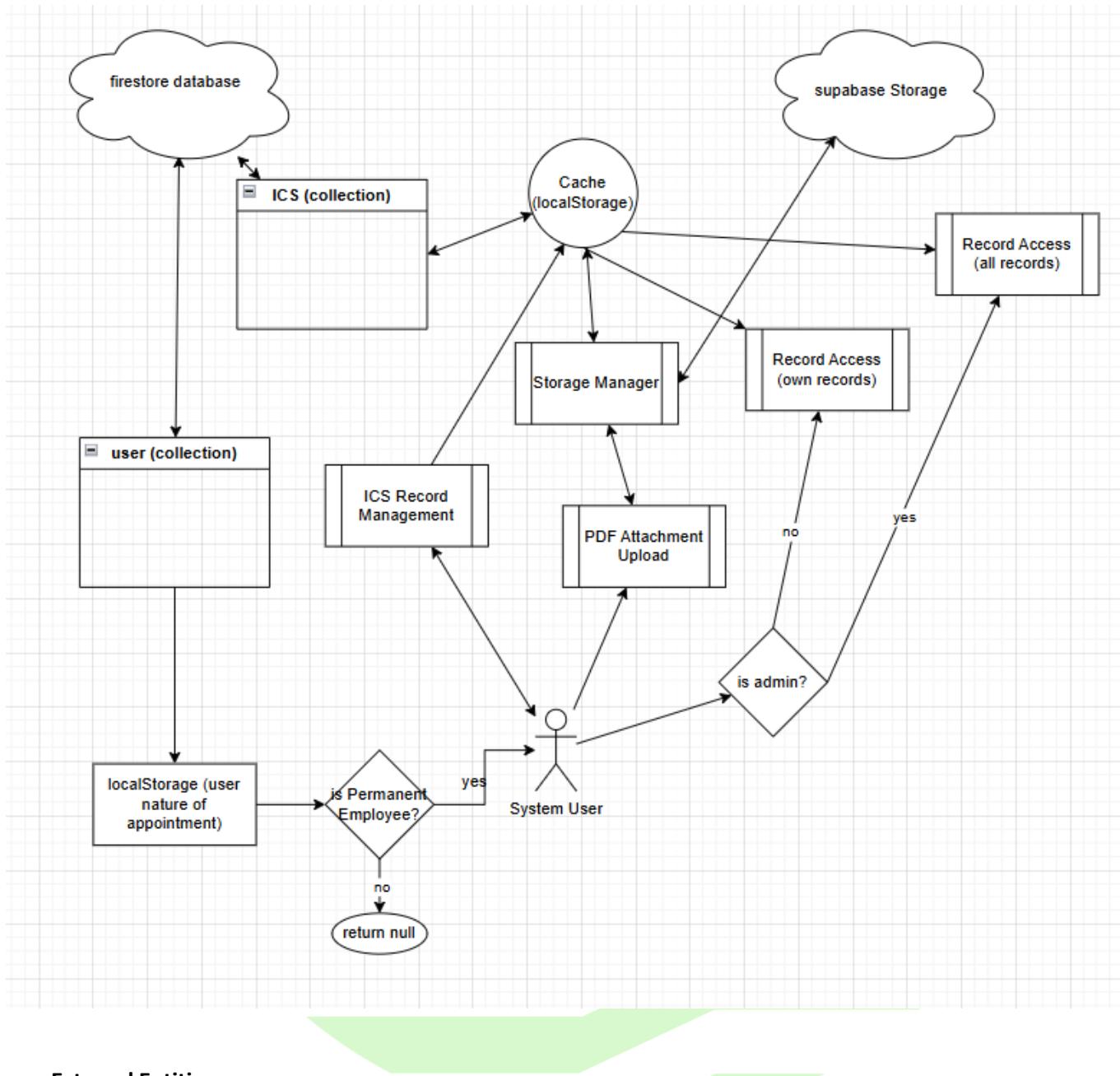
Data Stores

- Firestore Collection: consumables – Stores all consumable item details (e.g., name, quantity, description).
- Firestore Collection: ledger – Stores all consumable transaction records to maintain an audit trail.

Data Flow

- System User → Consumable Data Entry: Adds or edits consumable item information.
- Consumable Data Entry → Cache → consumables Collection: Updates consumable item records in Firestore.
- System User → Cache → Ledger Manager: Issues or returns consumable items.
- Ledger Manager → Cache → ledger Collection: Creates new ledger entries for each transaction.
- consumables Collection → Cache → Analytics & Reports: Provides current inventory status.
- ledger Collection → Cache → Analytics & Reports: Supplies historical transaction data for analysis.
- Analytics & Reports → System User: Delivers dashboards, usage reports, and audit summaries.

XII.B.3. Level 1 DFD – ICS Module



External Entities

- Admin – can add, edit, and upload PDF attachments for all employees' ICS records; can view everything.
- User (Permanent Employee) – can add, edit, and upload PDF attachments, but only for their own ICS accountability.

Processes

1. Authentication & Role Validation – verifies login credentials and determines role (Admin vs User).

2. Cache/localStorage – All data from Firestore is first loaded into the cache. Similarly, all data entered by the user is initially stored in the cache. Any data retrieved from Firestore also passes through the cache before being accessed by the user

3. ICS Record Management

- Add/Edit ICS accountability entries.
- Attach supporting PDFs.

4. Storage Manager – stores records and attachments:

- Saves ICS data to Firestore.
- Saves PDFs to Supabase Storage.

5. Record Access

- Admin: can retrieve and view all employee records.
- User: can only retrieve and view their own records.

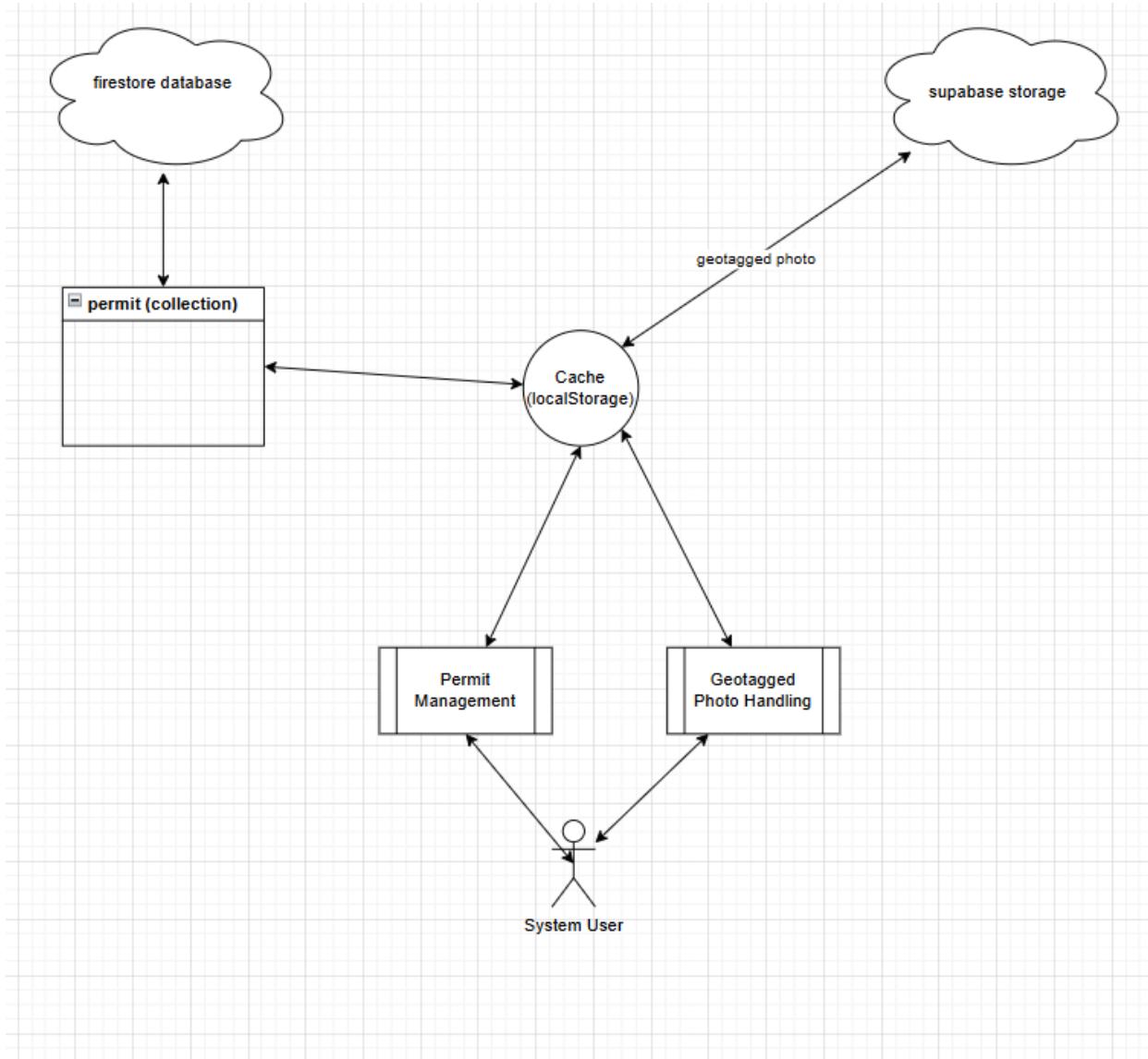
Data Stores

- Supabase Storage – stores PDF attachments of ICS forms.
- Firestore (ICS Collection) – stores accountability entries, tagged with employee IDs and linked PDF references.
- Local Storage (Cache) – temporarily stores ICS entries and metadata for faster access and offline capability before synchronization with Firestore.

Data Flow

1. Admin/User → Authentication & Role Validation.
2. Admin/User → ICS Record Management → Firestore (ICS Records).
3. Admin/User → PDF Attachment Upload → ICS Record Management → Supabase Storage.
4. ICS Record Management → Firestore (ICS Record + PDF link).
5. Admin → Record Access → Firestore (all records).
6. User → Record Access → Firestore (own records only).

XII.B.4. Level 1 DFD – Water Permit Module



External Entity

- System User – enters, edits, and manages permit records; uploads geotagged photos.

Processes

1. Authentication – validates login of the System User.
2. Permit Management – create, update, view, and cache permit details.
3. Geotagged Photo Handling – upload, store, and link geotagged photos.

4. Cache Manager – handles temporary storage and retrieval of permit data from Local Storage for faster access.
5. Excel File Extraction – export file of a water permit based on the search query.

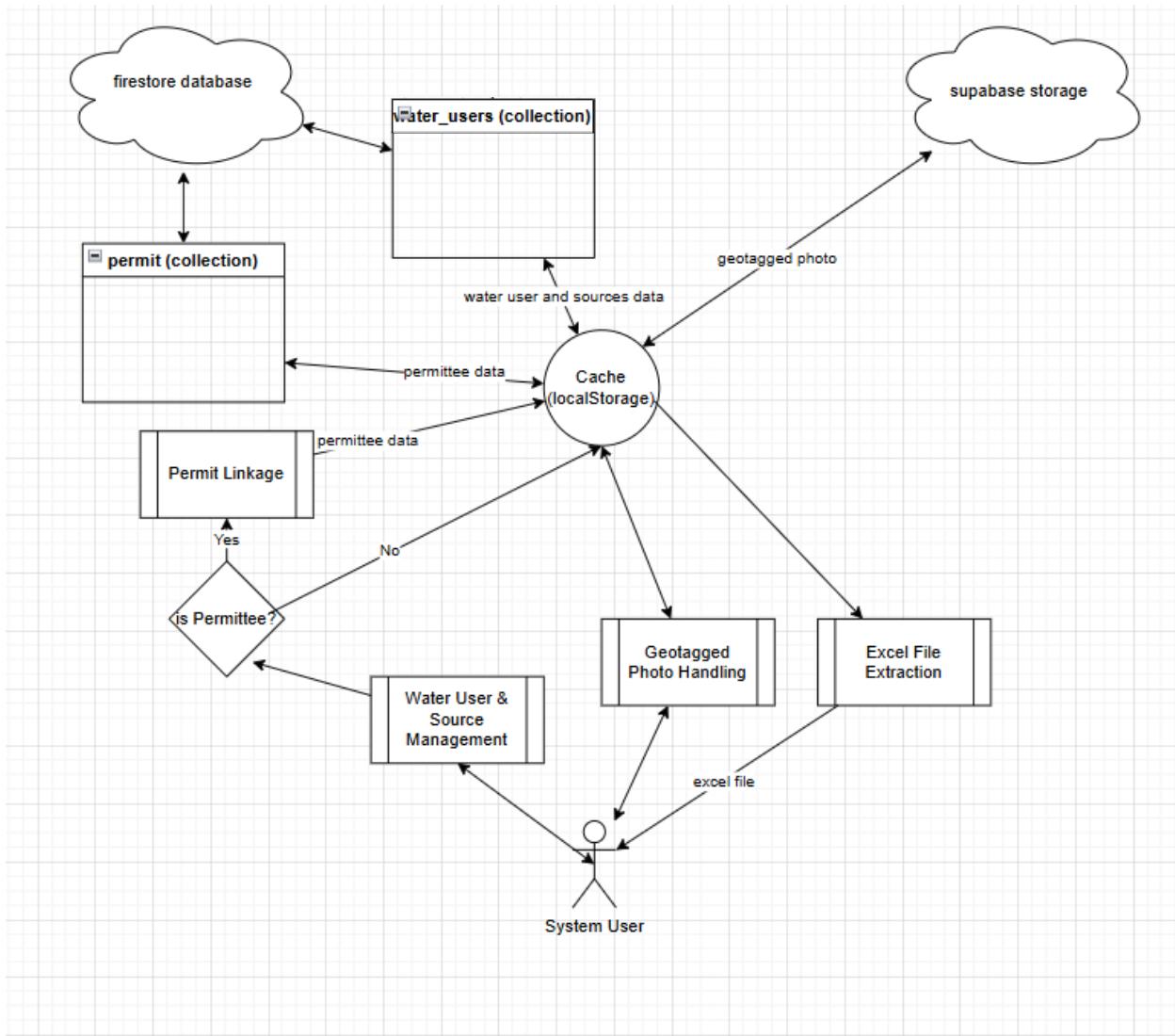
Data Stores

- Firestore (Permit Collection) – stores finalized permit records with links to geotagged photos.
- Supabase Storage – stores geotagged photo files.
- Local Storage (Cache) – temporary browser storage for quick retrieval of permit data.

Data Flow

1. System User → Permit Management → Cache → Firestore (Permit Records).
2. System User → Geotagged Photo Handling → Cache → Supabase Storage.
3. Geotagged Photo Handling → Cache → Firestore (reference link to photo in Supabase).
4. Permit Management ↔ Cache Manager ↔ Local Storage (read/write cache for faster access).
5. Permit Management → Firestore (Retrieve Records) → Cache Manager → Local Storage.
6. Firestore (Retrieve Records) → Cache Manager → Local Storage → Excel File Extraction → System User.

XII.B.5. Level 1 DFD - Water Users and Sources Module



External Entity

- **System User** – enters, edits, and manages water users and source records, uploads geotagged photos, and extracts Excel reports.

Processes

1. **Water User & Source Management** – create, update, and view records of water users and water sources.
2. **Cache Handling** – manages temporary storage of water user and source records for offline use and faster retrieval.

3. **Geotagged Photo Handling** – upload and link geotagged photos to the corresponding water user or source record.
4. **Permit Linkage** – checks if a water user is also a permittee and links the record to the Water Permit Module.
5. **Excel Extraction** – generates downloadable Excel reports of water user and source data.

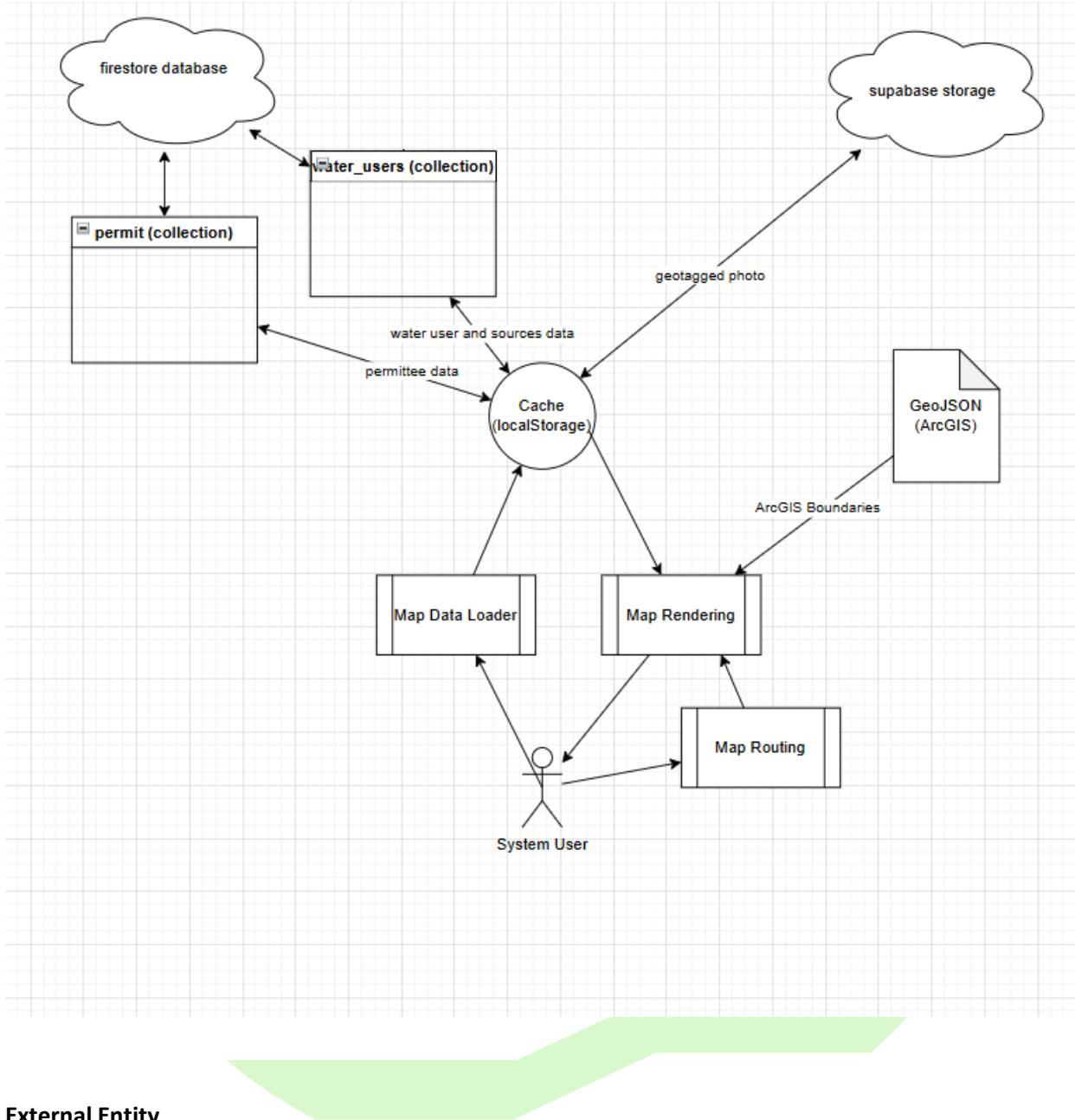
Data Stores

- **Local Storage (Cache)** – temporarily stores water user and source records for offline functionality and quick access.
- **Firestore (water_users & sources Collections)** – stores finalized water user and source records.
- **Firestore (permit Collection)** – stores water permit records, referenced when a water user is also a permittee.
- **Supabase Storage** – stores geotagged photo files, with references saved in Firestore.
- **Excel File (Generated Output)** – exported dataset based on user query or complete record set.

Data Flow

1. **System User → Water User & Source Management → Cache**: new or updated records are first stored in cache.
2. **Cache → Firestore (water_users & sources)**: finalized entries are synchronized from cache to Firestore.
3. **Firestore (water_users & sources) → Cache → System User**: records are loaded from Firestore into cache, then displayed to the System User.
4. **System User → Geotagged Photo Handling → Supabase Storage**: geotagged photos are uploaded to Supabase.
5. **Geotagged Photo Handling → Firestore (water_users & sources)**: reference link to the Supabase photo is stored in Firestore alongside the water user or source record.
6. **Water User & Source Management → Permit Linkage → Firestore (permit Collection)**: links water user records to permit records when the user is a permittee.
7. **System User → Excel Extraction → Excel File**: allows the user to download water users and sources data in Excel format.

XII.B.6. Level 1 DFD – Map Module



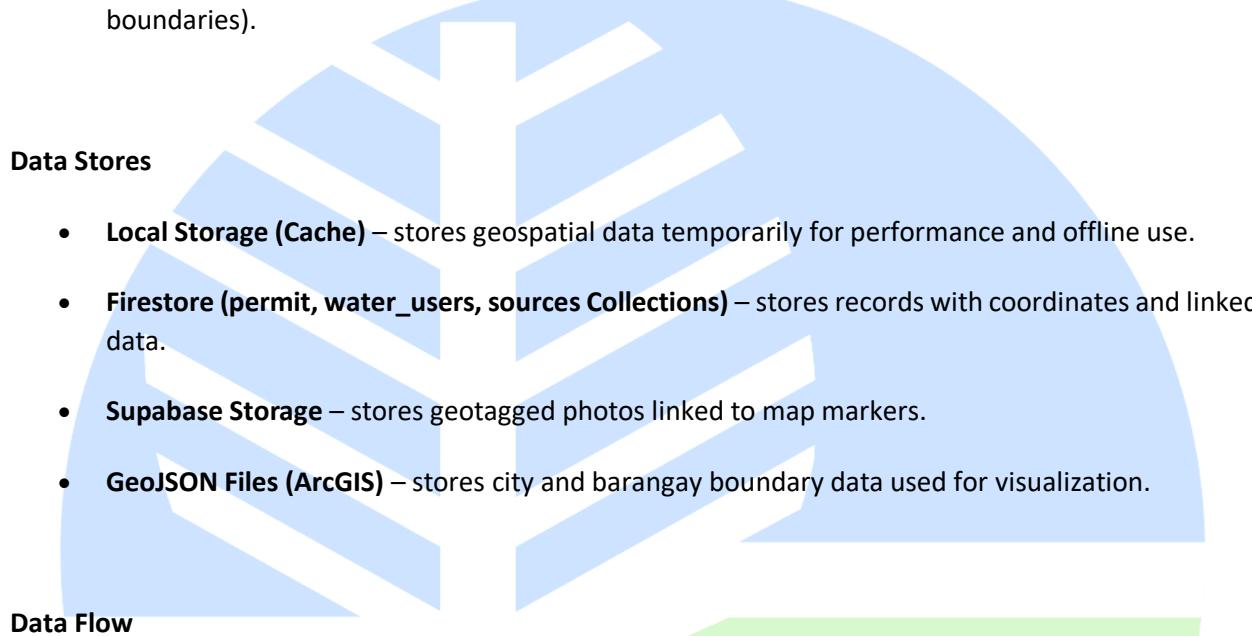
External Entity

- **System User** – views geospatial data, permit locations, water sources, and navigates using map routing.

Processes

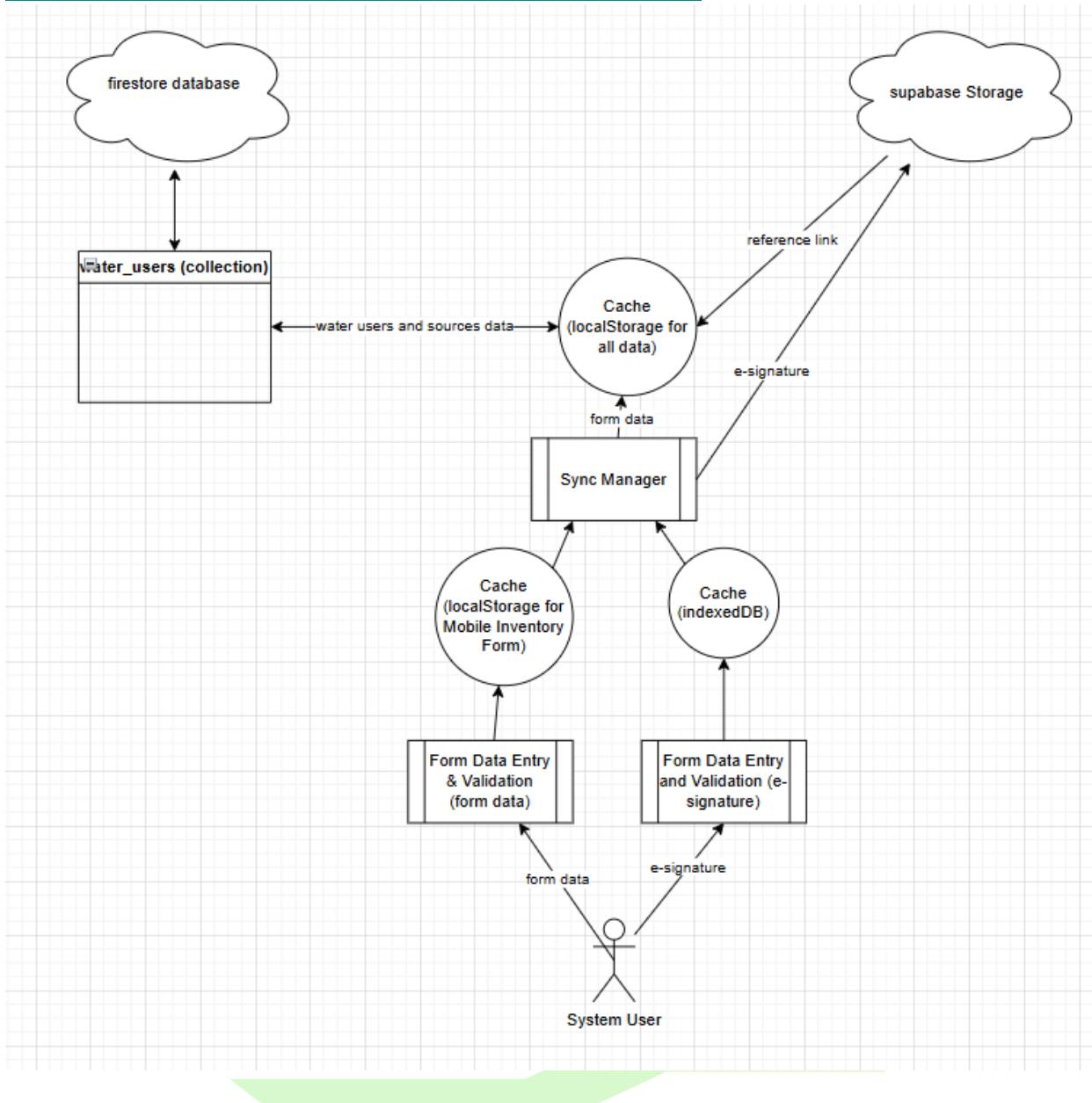
1. **Map Data Loader** – loads permit, water users, and sources data (including geotagged photo references) into the map interface.

2. **Cache Handling** – caches geospatial data for faster map rendering and offline viewing.
3. **Map Rendering (Leaflet.js / ArcGIS GeoJSON)** – displays city, barangay boundaries, permit, and water source markers.
4. **Map Routing** – generates navigation paths based on geographic coordinates.
5. **Layer Management** – allows switching map layers (e.g., permit data, water users, sources, boundaries).



1. **System User → Map Data Loader → Cache:** map data (permits, water users, sources, boundaries) is loaded into cache.
2. **Cache → Map Rendering (Leaflet.js/GeoJSON) → System User:** cached data is displayed as interactive map layers.
3. **Firestore (permit, water_users, sources) → Cache:** records with coordinates are synced from Firestore into cache.
4. **Supabase Storage → Firestore (reference link) → Cache → Map Rendering:** geotagged photo references are pulled from Firestore and displayed as linked map markers.
5. **GeoJSON (ArcGIS boundaries) → Map Rendering → System User:** city and barangay boundary layers are overlaid on the map.
6. **System User → Map Routing → Map Rendering:** user requests route, system generates navigation paths using coordinates.

XII.B.7. Level 1 DFD – Mobile Inventory Form



External Entity

- **System User** – field staff who records and submits water inventory data through the mobile form.

Processes

1. **Form Data Entry & Validation** – user enters water inventory details, attaches geotagged photos, and captures e-signatures.
2. **Cache Handling (Local Storage)** – stores offline draft forms and metadata when no internet is available.
3. **IndexedDB Handling** – stores large objects such as e-signatures and geotagged images.
4. **Sync Manager** – checks connectivity and pushes finalized entries to Firestore and Supabase when online.
5. **Firestore Synchronization** – stores finalized water inventory data in the `water_users` collection.

Data Stores

- **Local Storage (Cache)** – saves offline draft forms and metadata.
- **IndexedDB** – stores e-signatures and large image objects.
- **Firestore (water_users Collection)** – stores finalized water inventory records.
- **Supabase Storage** – stores uploaded geotagged photos, linked by references in Firestore.

Data Flow

1. **System User → Form Data Entry & Validation → Cache (Local Storage)**: offline drafts are saved locally.
2. **System User → Form Data Entry & Validation → IndexedDB**: e-signatures and large geotagged files are stored locally.
3. **Cache & IndexedDB → Sync Manager → Firestore (water_users Collection)**: finalized entries are uploaded once online.
4. **Cache & IndexedDB → Sync Manager → Supabase Storage**: geotagged photos are uploaded, and Firestore stores reference links.
5. **Firestore → Cache**: records are downloaded into cache for faster access and offline availability.

XII.B.8. DFD Level 1 – MEMIS Data to Database Module



Processes:

1. Upload MEMIS Excel File – The user uploads the MEMIS Excel file.
2. Parse Excel File – System reads and extracts water users and sources data.
3. Store to Cache – Extracted data is first stored in local storage cache for faster access.
4. Validate and Transform Data – Ensures entries follow database format (e.g., permit numbers, water source categories).

5. Update Firestore (Water Users and Sources Collection) – Data is pushed to Firestore after validation.

Data Stores:

- Firestore (Water Users & Sources Collection) – Final permanent database storage.

External Entities:

- System User – Uploads MEMIS Excel file, reviews results, downloads extracted file.
- Excel File (MEMIS Data) – Source file containing water users and sources dataset.

Data Flows:

- MEMIS Excel File → Upload → Parse Excel File → Store in Cache → Validate & Transform → Firestore (Water Users & Sources Collection).
- Firestore data → Cached → Served to User on request.



XIII. Project File Structure

XIII. A. Overview

This project is developed using **Object-Oriented Programming (OOP)** principles to ensure modularity, reusability, and ease of maintenance. The overall structure separates core components into dedicated folders for **HTML, CSS, JavaScript, images, and files**, keeping the system organized and scalable.

Within the **JavaScript** directory, responsibilities are further divided into:

- **Data scripts** – handle caching and structured data management.
- **Authentication scripts** – manage user login and access.
- **Component scripts** - UI-related JavaScript like sidebar, spinners, notifications
- **Module-specific scripts** – implement features for each system module.
- **Excel-related scripts** – manage data extraction, formatting, and export.
- **Supabase upload scripts** – handle file and photo uploads to cloud storage.
- **Mapping scripts** – process and render geospatial data.
- **Utility scripts** – provide reusable helper functions.
- **Configuration scripts** – store system settings and Firebase/Supabase integration details.

This separation of concerns improves code readability and ensures that each part of the system can be maintained and extended independently.

XIII.B. Folder Tree Format

```
Dockerfile
firebase.json
package-lock.json
package-lock.json
server.js
public/
  └── index.html          #login page
  └── admin-dashboard.html #landing page for admin users
  └── consumable.html     #consumable module
```



```

└─ js/
  └─ data/
    └─ cache/
      └─ consumable-data.js          #js for Consumable object
      └─ ics-data.js                #js for ICS object
      └─ ledger-data.js             #js for Ledger object
      └─ permit-data.js              #js for Permit object
      └─ user-data.js                #js for User object
      └─ wrus-data.js               #js for WUSData object
    └─ constants/
      └─ metroManilaCities.js        #list of cities in NCR
      └─ months.js                  #list of months
      └─ purpose.js                 #list of usage of water
      └─ sourceStatus.js             #list of status of water user
      └─ waterSources.js            #list of water sources
    └─ geojson/
      └─ Barangays_NCR.geojson      #boundaries of barangays in NCR
      └─ caloocan.geojson           #boundary of Caloocan City
      └─ las_piñas.geojson           #boundary of Las Piñas City
      └─ makati.geojson              #boundary of Makati City
      └─ malabon.geojson             #boundary of Malabon City
      └─ marikina.geojson            #boundary of Marikina City
      └─ muntinlupa.geojson          #boundary of Muntinlupa City
      └─ navotas.geojson             #boundary of Navotas City
      └─ parañaque.geojson           #boundary of Parañaque City
      └─ pasay.geojson                #boundary of Pasay City
      └─ pasig.geojson                #boundary of Pasig City
      └─ pateros.geojson              #boundary of Pateros City

```

└── quezon_city.geojson	#boundary of Quezon City
└── san_juan.geojson	#boundary of San Juan City
└── taguig.geojson	#boundary of Taguig City
└── valenzuela.geojson	#boundary of Valenzuela City
└── mandaluyong.geojson	#boundary of Mandaluyong City
└── manila.geojson	#boundary of Manila City
└── indexedDB/	
└── signature-data.js	#object of indexedDB e-signature
└── auth/	
└── auth.js	#js of authentication object
└── consumable/	
└── consumable-init.js	#initial js for consumable module
└── consumable-ui.js	#functions of consumable module
└── dashboard	
└── admin-dashboard-init.js	#initial js for admin dashboard
└── admin-dashboard-ui.js	#functions of admin dashboard
└── change-password.js	#change password tab in the dashboard
└── city-accomplishment-summary.js	#water inventory data and map
└── dashboard-init.js	#initial js for dashboard
└── permit-city-summary.js	#table statistics for permittees
└── version-control.js	#manages version control
└── yearly-summary.js	#table for yearly statistics
└── excel/	
└── excel-init.js	#initial js for MEMIS Excel module
└── firestore-functions.js	#dummy js file for testing
└── upload-excel.js	#function for Excel module

```

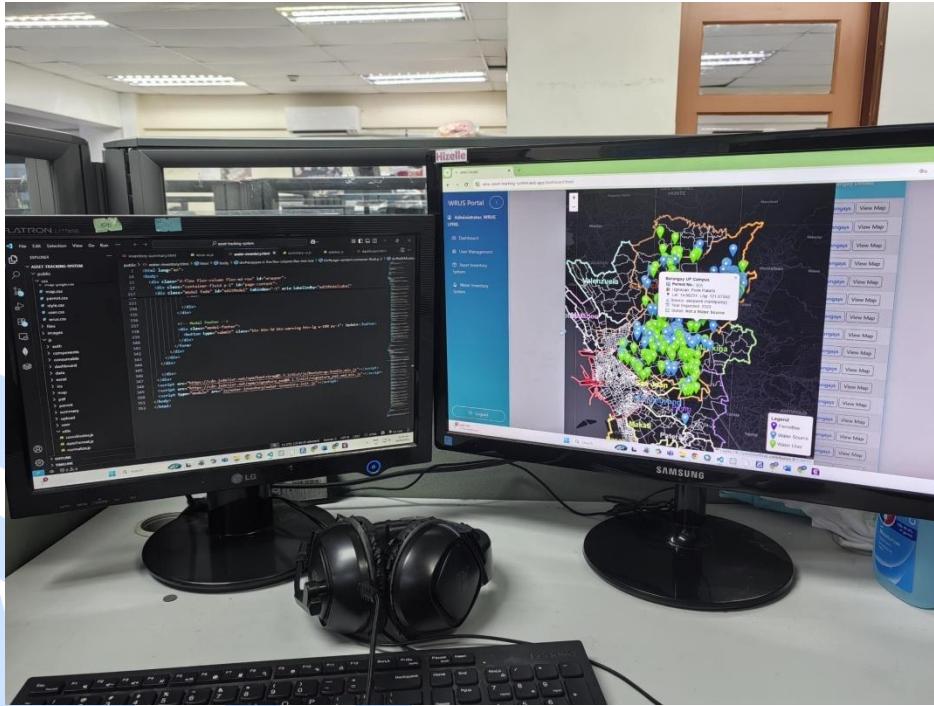
| | └── ics/
| |   | └── ics-init.js          #initial js for ics module
| |   | └── ics-ui.js           #functions of ics module
| | └── map/
| |   | └── cityBoundaries.js    #js for rendering map boundaries
| |   | └── leaflet-main.js      #js for leaflet map (not used)
| |   | └── map-init.js         #js for setup map
| |   | └── mapPrint.js        #module for printing map
| |   | └── routing-page-init.js #initial js for map routing module
| |   | └── routing-page-ui.js   #functions of map routing module
| | └── pdf/
| |   | └── ics-pdf.js          #pdf for ICS user summary
| |   | └── item-consumable-pdf.js #pdf for consumable ledger
| |   | └── user-consumable.pdf.js #pdf for user's consumable ledger
| |   | └── water-user-pdf.js    #pdf of inventory form
| | └── permit/
| |   | └── permit-init.js      #initial js for permit module
| |   | └── permit-ui.js        #functions of permit module
| | └── permit/
| |   | └── summary-init.js     #initial js for analytics module
| |   | └── summary-ui.js       #functions of analytics module
| | └── upload/
| |   | └── supabaseInit_2.js    #supabase config 1st account
| |   | └── supabaseInit.js      #supabase config 2nd account
| |   | └── upload.html          #test html for uploading
| |   | └── upload.js            #js for uploading permit pdf and ics pdf
| |   | └── uploadFirestore.js   #js for saving reference link for firebase
| |   | └── uploadImg.js         #js for uploading geotagged photo

```

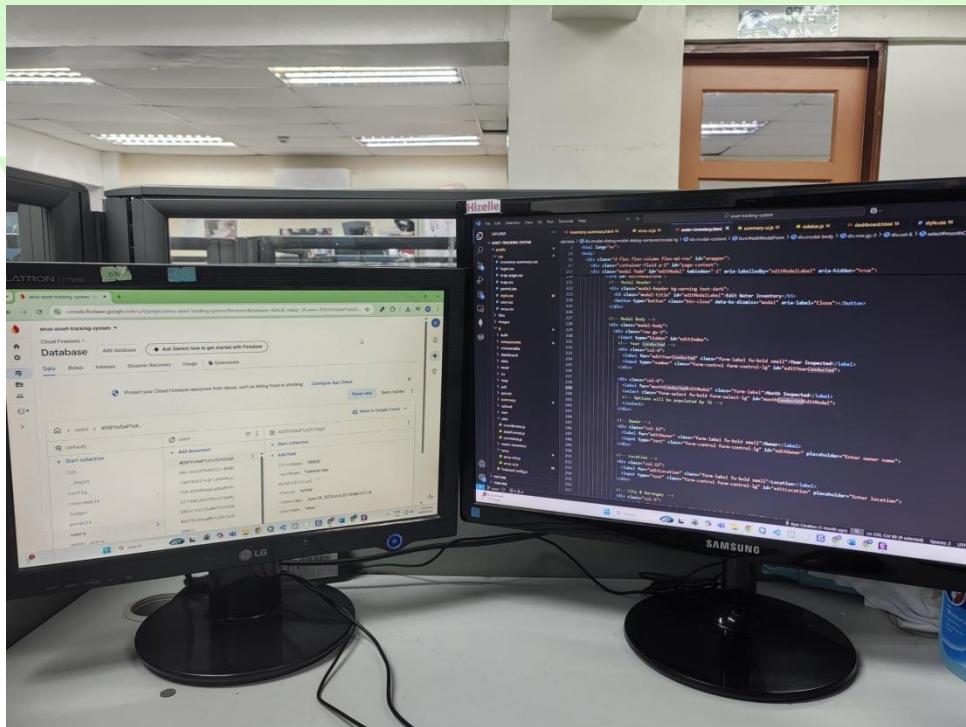
```
| | └── user/
| |   └── user-init.js          #initial js for user management module
| |   └── user-ui.js            #functions of user management module
| └── utils/
|   └── coordinates.js         #helper js for converting coordinates
|   └── dateFormat.js          #helper js for converting date format
|   └── normalize.js           #helper js for normalize brgy or city
| └── water-inventory/
|   └── signature-pad-module.js #js for e-signature pad
|   └── water-inventory-init.js #initial js for mobile inventory module
|   └── water-inventory-ui.js   #functions of mobile inventory module
| └── wrus/
|   └── wrus-init.js           #initial js for water user module
|   └── wrus-ui.js              #functions of water user module
| └── firebaseConfig.js        #config for firebase
| └── login.js                 #js for login page
└── images/
└── files/                    #images like logo and default image
                             #pdf of issuances of WRUS
```

XIV. Photo Documentation

IDE (VS Studio) and System User Interface in Production Phase



Firestore and IDE (VS Studio) in Production Phase



XV. Source Code

XV.A HTML

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>WRUS Portal</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css">
<link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png">
<link rel="stylesheet" href="css/login.css" />
</head>
<body>
<div id="portal-alert" class="portal-alert">
<div class="portal-alert-content">
<h4 id="portal-alert-title">Notice</h4>
<p id="portal-alert-message"></p>
<button id="portal-alert-ok" class="btn btn-primary w-100">OK</button>
</div>
</div>

<div class="portal"></div>
<div id="portal-light"></div>

<div class="card">
<h3 class="card-title text-center mb-4">🌐 WRUS Portal</h3>
<form id="loginForm">
<div class="mb-3">
<label for="username" class="form-label">Username</label>
<input type="text" class="form-control" id="username" placeholder="Enter username" required />
</div>
<div class="mb-3">
<label for="password" class="form-label">Password</label>
<input type="password" class="form-control" id="password" placeholder="Enter password" required />
</div>
<button type="submit" class="btn btn-primary w-100">Login</button>
</form>
<div id="errorMsg" class="mt-3 text-center" style="display: none;">
  Invalid username or password.
</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
<script type="module" src="js/login.js"></script>
</body>
</html>
```

dashboard.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>WRUS Portal</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" />

<link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css"/>
<link rel="stylesheet" href="css/style.css" />
<link rel="stylesheet" href="css/dashboard.css" />
<link rel="stylesheet" href="css/map.css" />
<link rel="stylesheet" href="css/component/rain.css" />
</head>
<body>
<div class="d-flex flex-column flex-md-row" id="wrapper">
<div id="sidebarContainer"></div>

    <!-- Page Content -->
<div class="container-fluid d-flex flex-column vh-100 p-2" id="page-content">
    <!-- Tabs Navigation -->
<ul class="nav nav-tabs flex-shrink-0" id="dashboardTabs" role="tablist">
    <li class="nav-item" role="presentation">
        <button class="nav-link active" id="profile-tab" data-bs-toggle="tab" data-bs-target="#profileTab" type="button" role="tab" aria-controls="profileTab" aria-selected="true">
            About
        </button>
    </li>
    <li class="nav-item" role="presentation">
        <button class="nav-link .btn-3d" id="summary-tab" data-bs-toggle="tab" data-bs-target="#summaryTab" type="button" role="tab" aria-controls="summaryTab" aria-selected="false">
            Database
        </button>
    </li>
    <li class="nav-item" role="presentation">
        <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-target="#settingsTab" type="button" role="tab" aria-controls="settingsTab" aria-selected="false">
            Account Settings
        </button>
    </li>
</ul>

    <!-- Tab Content -->
<div class="tab-content" id="dashboardTabContent">
    <!-- About Tab -->
<div class="tab-pane fade show active" id="profileTab" role="tabpanel" aria-labelledby="profile-tab">
```

```

<div class="profile-banner d-flex flex-column align-items-start p-2 rounded shadow-sm mb-1">

    <!-- Jumbotron -->
    <div class="container-fluid simple-water-jumbotron mt-4">
        <div class="p-0 mb-4 rounded-3 shadow-sm">
            <div class="container-fluid py-5">
                <div class="logo-container">
                    
                </div>
                <h1 class="display-5 fw-bold">
                    Water Resources and Utilization Section - National Capital Region
                </h1>
                <p class="col-md-8 text-light mx-auto">
                    The Water Resources Utilization Section (WRUS) of Licenses, Patents, and Deeds Division (LPDD) is responsible for the conducting of water inventory for water users and sources and verifying water permits for the National Capital Region.
                </p>
            </div>
            <div class="activities-section">
                <h5 class="fw-bold mb-3 text-light"> Activities - Annual Target</h5>
                <ul class="list-group list-group-flush col-md-6 mx-auto">
                    <li class="list-group-item d-flex justify-content-between align-items-center">
                        1. Conduct of Inventory of Water Users
                        <span class="badge badge-3d rounded-pill bg-primary shadow-sm">220</span>
                    </li>
                    <li class="list-group-item d-flex justify-content-between align-items-center">
                        2. Conduct of Inventory of Water Sources
                        <span class="badge badge-3d rounded-pill bg-success shadow-sm">110</span>
                    </li>
                    <li class="list-group-item d-flex justify-content-between align-items-center">
                        3. Mapping of Water Users and Sources
                        <span class="badge badge-3d rounded-pill bg-info shadow-sm">1</span>
                    </li>
                    <li class="list-group-item d-flex justify-content-between align-items-center">
                        4. Maintenance and Updating of Database for Water Users
                        <span class="badge badge-3d rounded-pill bg-dark shadow-sm">1</span>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</div>

<div class="container mt-4 text-center">
    <div class="table-responsive d-inline-block">
        <div class="table-container">
            <div id="yearlyWaterUserSummary" class="my-4"></div>
        </div>
    </div>
</div>

<!-- Issuances Section -->
<div class="container my-5">

```



```
<h2 class="text-center fw-bold text-dark mb-4">Issuances</h2>
<p class="text-center text-dark mb-5">
An overview of the principal mandates of the Water Resources and Utilization Section (National Capital Region) concerning the conducting of water inventory, identification of water users and sources, preparation of reports, and management of geospatial information.
</p>

<div class="row text-center">
<!-- Card 1 -->
<div class="col-md-6 mb-4">
<div class="card h-100 shadow-sm border-0">
<div class="card-header-banner">
    PD no. 1067 - Water Code of the Philippines
</div>
<div class="card-body d-flex flex-column">
<p class="card-text flex-grow-1">
        Presidential Decree No. 1067, or the Water Code of the Philippines, declares all waters state-owned and regulates their use, conservation, and protection through the NWRB.
    </p>
<div class="d-flex gap-2 mt-auto">
<a href=".files/PD 1076 s. 1976.pdf"
    target="_blank"
    class="btn btn-outline-light btn-sm btn-3d">
        View PDF
    </a>
<a href=".files/PD 1076 s. 1976.pdf"
    download
    class="btn btn-outline-light btn-sm btn-3d">
        Download PDF
    </a>
</div>
</div>
</div>
</div>

<!-- Card 2 -->
<div class="col-md-6 mb-4">
<div class="card h-100 shadow-sm border-0">
<div class="card-header-banner">
    NWRB Resolution No. 001-0904 - Metro Manila Critical Areas
</div>
<div class="card-body d-flex flex-column">
<p class="card-text flex-grow-1">
        NWRB Resolution No. 001-0904 designates Metro Manila as a critical water area to ensure stricter regulation, conservation, and sustainable management of its water resources.
    </p>
<div class="d-flex gap-2 mt-auto">
<a href=".files/NWRB Resolution No. 001-0904.pdf"
    target="_blank"
    class="btn btn-outline-light btn-sm btn-3d">
        View PDF
    </a>
<a href=".files/NWRB Resolution No. 001-0904.pdf"
    download
    class="btn btn-outline-light btn-sm btn-3d">
        Download PDF
    </a>
</div>
</div>
</div>
```

```
download
class="btn btn-outline-light btn-sm btn-3d">
Download PDF
</a>
</div>
</div>
</div>
</div>

<!-- Card 3 -->
<div class="col-md-6 mb-4">
<div class="card h-100 shadow-sm border-0">
<div class="card-header-banner">
    NWRB 15-1116 - Deputation of DENR Regional Offices on Certain Functions of Water Use Regulation
</div>
<div class="card-body d-flex flex-column">
<p class="card-text flex-grow-1">
    NWRB Resolution No. 15-1116 authorizes DENR Regional Offices to assist in enforcing water use regulations by deputizing them for specific regulatory functions.
</p>
<div class="d-flex gap-2 mt-auto">
<a href=".//files/NWRB_15-1116 - Deputation of DENR Regional Offices on Certain Functions of Water Use Regulation.pdf"
    target="_blank"
    class="btn btn-outline-light btn-sm btn-3d">
        View PDF
    </a>
<a href=".//files/NWRB_15-1116 - Deputation of DENR Regional Offices on Certain Functions of Water Use Regulation.pdf"
    download
    class="btn btn-outline-light btn-sm btn-3d">
        Download PDF
    </a>
</div>
</div>
</div>
</div>

<!-- Card 4 -->
<div class="col-md-6 mb-4">
<div class="card h-100 shadow-sm border-0">
<div class="card-header-banner">
    EO 22 - Creating the Water Resources Management Office in the DENR
</div>
<div class="card-body d-flex flex-column">
<p class="card-text flex-grow-1">
    Executive Order No. 22 establishes the Water Resources Management Office (WRMO) within the DENR to coordinate, integrate, and oversee the sustainable management of the country's water resources.
</p>
<div class="d-flex gap-2 mt-auto">
<a href=".//files/EO 22 - CREATING THE WATER RESOURCES MANAGEMENT OFFICE IN THE DENR.pdf"
    target="_blank"
    class="btn btn-outline-light btn-sm btn-3d">
```

View PDF

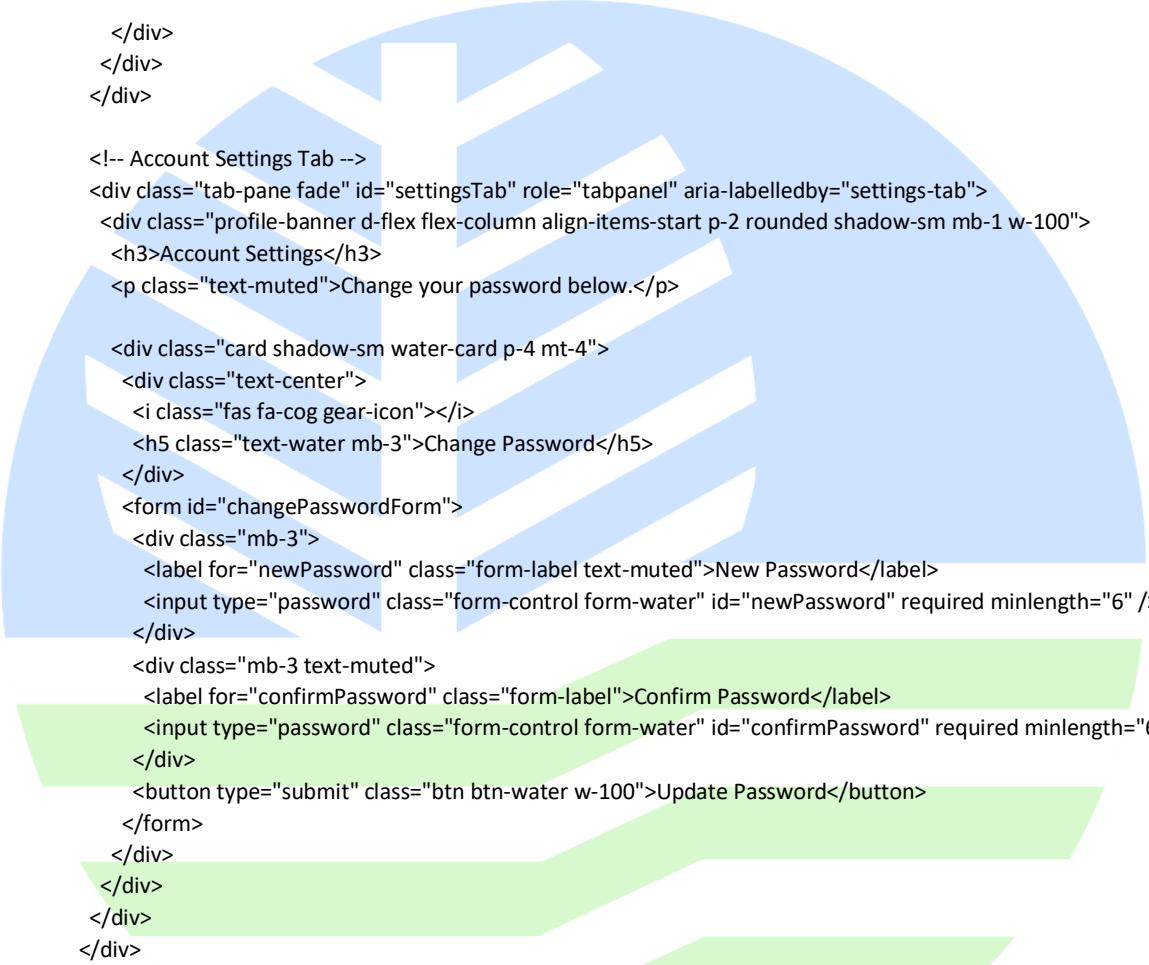
 <a href=".files/EO 22 - CREATING THE WATER RESOURCES MANAGEMENT OFFICE IN THE DENR.pdf" download
 class="btn btn-outline-light btn-sm btn-3d">
 Download PDF

 </div>
 </div>
 </div>
 </div>
 </div>
 </div>
 </div>
 </div>

```

<!-- Database Tab -->
<div class="tab-pane fade" id="summaryTab" role="tabpanel" aria-labelledby="summary-tab">
<div class="profile-banner d-flex flex-column align-items-start p-2 rounded shadow-sm mb-1">
  <!-- Inner Tabs -->
  <ul class="nav nav-tabs mb-3" id="summaryInnerTabs" role="tablist">
    <li class="nav-item" role="presentation">
      <button class="nav-link active" id="permit-summary-tab" data-bs-toggle="tab" data-bs-target="#permitSummary" type="button" role="tab" aria-controls="permitSummary" aria-selected="true">
        Water Permit
      </button>
    </li>
    <li class="nav-item" role="presentation">
      <button class="nav-link" id="city-accom-tab" data-bs-toggle="tab" data-bs-target="#cityAccomplishment" type="button" role="tab" aria-controls="cityAccomplishment" aria-selected="false">
        Water Users and Sources
      </button>
    </li>
  </ul>
  <!-- Inner Tab Content -->
  <div class="tab-content w-100" id="summaryInnerTabsContent">
    <!-- Permit Summary Tab Pane -->
    <div class="tab-pane fade show active" id="permitSummary" role="tabpanel" aria-labelledby="permit-summary-tab">
      <div class="city-table-wrapper text-center my-4">
        <div class="table-responsive d-flex justify-content-center">
          <div class="table-container">
            <div id="cityPermitTable" class="mb-4"></div>
          </div>
        </div>
      </div>
    </div>
    <!-- Accomplishment Per City Tab Pane -->
    <div class="tab-pane fade" id="cityAccomplishment" role="tabpanel" aria-labelledby="city-accom-tab">
    </div>
  </div>

```



```

<div class="city-table-wrapper text-center my-4">
  <div class="table-responsive d-flex justify-content-center">
    <div class="table-container">
      <div id="cityAccomplishmentTable" class="my-4 w-100"></div>
    </div>
  </div>
</div>

</div>
</div>
</div>

<!-- Account Settings Tab -->
<div class="tab-pane fade" id="settingsTab" role="tabpanel" aria-labelledby="settings-tab">
  <div class="profile-banner d-flex flex-column align-items-start p-2 rounded shadow-sm mb-1 w-100">
    <h3>Account Settings</h3>
    <p class="text-muted">Change your password below.</p>

    <div class="card shadow-sm water-card p-4 mt-4">
      <div class="text-center">
        <i class="fas fa-cog gear-icon"></i>
        <h5 class="text-water mb-3">Change Password</h5>
      </div>
      <form id="changePasswordForm">
        <div class="mb-3">
          <label for="newPassword" class="form-label text-muted">New Password</label>
          <input type="password" class="form-control form-water" id="newPassword" required minlength="6" />
        </div>
        <div class="mb-3 text-muted">
          <label for="confirmPassword" class="form-label">Confirm Password</label>
          <input type="password" class="form-control form-water" id="confirmPassword" required minlength="6" />
        </div>
        <button type="submit" class="btn btn-water w-100">Update Password</button>
      </form>
    </div>
  </div>
</div>

<!-- Barangay Modal -->
<div class="modal fade" id="barangayModal" tabindex="-1" aria-labelledby="barangayModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg modal-dialog-centered modal-dialog-scrollable">
    <div class="modal-content">
      <div class="modal-header bg-primary text-white">
        <h5 class="modal-title text-white" id="barangayModalLabel">Barangay Breakdown</h5>
        <button type="button" class="btn-close btn-3d btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <!-- Dynamic content here -->
      </div>
    </div>
  </div>
</div>

```



```

<div class="table-responsive">
  <div id="barangayModalBody"></div>
</div>
</div>
</div>
</div>
</div>

<!-- Water User Modal -->
<div class="modal fade" id="waterUsersModal" tabindex="-1" aria-labelledby="waterUsersModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-xl modal-dialog-centered modal-dialog-scrollable">
    <div class="modal-content">
      <div class="modal-header bg-success text-white">
        <h5 class="modal-title text-white" id="waterUsersModalLabel">Water Users</h5>
        <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <div class="table-responsive">
          <div id="waterUsersModalBody"></div>
        </div>
      </div>
    </div>
  </div>
</div>

<!-- Map Modal -->
<div class="modal fade" id="mapModal" tabindex="-1" aria-labelledby="mapModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header bg-success d-flex justify-content-between align-items-center">
        <h5 class="modal-title text-white" id="mapModalLabel">National Capital Region Map</h5>
        <div>
          <button class="btn btn-3d btn-light btn-sm me-2" id="printMapBtn">
            <i class="bi bi-printer"></i> Print
          </button>
          <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
        </div>
      </div>
      <div class="modal-body p-0">
        <div id="leafletMap"></div>
      </div>
    </div>
  </div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
<script type="module" src="./js/dashboard/dashboard-init.js"></script>
</body>
</html>

```

admin-dashboard.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>WRUS Portal</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css">
<link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png">
<link rel="stylesheet" href="css/style.css" />
<link rel="stylesheet" href="css/dashboard.css" />
<link rel="stylesheet" href="css/admin/admin-dashboard.css" />
<link rel="stylesheet" href="css/component/bubble.css" />
</head>
<body>
<div class="d-flex flex-column flex-md-row" id="wrapper">
<div id="sidebarContainer"></div>

<!-- Page Content -->
<div class="container-fluid p-4 d-flex flex-column" id="page-content" style="min-height: 100vh;">
<h3 class="welcome-text">
    Welcome Admin
</h3>

<div class="scorecard-wrapper">
<div class="scorecard">
<div class="scorecard-header"> User Encoded Permits
<button id="refreshEncodedBtn" class="scorecard-refresh-btn" title="Refresh">
    <i class="bi bi-arrow-clockwise"></i>
</button>
</div>

<!-- Card-style entries injected here -->
<div class="scorecard-grid-container" id="encodedPermitTableBody">
<!-- JS fills this -->
</div>

<p id="noEncodedMessage" class="no-record d-none">No encoded permits found.</p>
</div>
</div>
</div>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script type="module" src="js/dashboard/admin-dashboard-init.js"></script>
</body>
</html>

```

user-management.html

```

<!DOCTYPE html>
<html lang="en">

```

```
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>WRUS Portal</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" />
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet" />
<link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet" />
<link rel="stylesheet" href="css/style.css" />
<link rel="stylesheet" href="css/user.css">
</head>

<body>
<div class="d-flex flex-column flex-md-row" id="wrapper">
<div id="sidebarContainer"></div>
<!-- Page Content -->
<div class="container-fluid p-2" id="page-content">

<div class="mb-4">
<!-- Header at the top -->
<h2 class="file-tab-heading">User Management</h2>

<!-- Search bar and Add button inline -->
<div class="d-flex flex-column flex-sm-row align-items-stretch gap-2">
<!-- Search Bar -->
<div class="input-group flex-grow-1">
<span class="input-group-text" id="search-icon">
<i class="bi bi-search"></i>
</span>
<input
  type="text"
  id="searchInput"
  class="form-control"
  placeholder="Search by username or name..." 
  aria-label="Search"
  aria-describedby="search-icon"
/>
</div>
</div>
</div>

<button id="refreshBtn" class="btn btn-3d btn-primary shadow-sm px-3 py-2 d-flex align-items-center justify-content-center">
<i class="bi bi-arrow-clockwise me-1"></i>
</button>
<!-- Add User Button -->
<button
  id="showAddUserFormBtn"
  class="btn btn-3d btn-outline-success flex-shrink-0"
  style="white-space: nowrap;">
<i class="bi bi-person-plus me-1"></i> Add User
</button>
</div>
```

```

</div>


<div class="container-fluid">
<div class="table-responsive">
<table>
<thead class="table-primary text-center">
<tr>
<th>Username</th>
<th>Last Name</th>
<th>First Name</th>
<th>Position</th>
<th>Type</th>
<th>Action</th>
</tr>
</thead>
<tbody id="usersTableBody"></tbody>
</table>
</div>
</div>


<nav>
<ul class="pagination justify-content-center" id="paginationControlsUsers">
<!-- Pagination buttons will be generated here -->
</ul>
</nav>


<div class="modal fade" id="addUserModal" tabindex="-1" aria-labelledby="addUserModalLabel" aria-hidden="true">
<div class="modal-dialog modal-lg modal-dialog-centered">
<div class="modal-content shadow rounded-3">
<form id="addUserForm" class="needs-validation" novalidate>
<div class="modal-header bg-info">
<h5 class="modal-title text-white fw-semibold" id="addUserModalLabel">Add New User</h5>
<button type="button" class="btn-close btn-3d btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>
</div>
<div class="modal-body px-4 py-3">
<!-- Nav tabs -->
<ul class="nav nav-tabs mb-3" id="userTab" role="tablist">
<li class="nav-item" role="presentation">
<button class="nav-link active" id="account-tab" data-bs-toggle="tab" data-bs-target="#accountTab" type="button" role="tab">Account</button>
</li>
<li class="nav-item" role="presentation">
<button class="nav-link" id="info-tab" data-bs-toggle="tab" data-bs-target="#infoTab" type="button" role="tab">Personal Info</button>
</li>
</ul>

```

```

<!-- Tab panes -->
<div class="tab-content">
    <!-- Account Tab -->
    <div class="tab-pane fade show active" id="accountTab" role="tabpanel">
        <div class="row g-3">
            <div class="col-md-12">
                <label for="username" class="form-label">Username <span class="text-danger">*</span></label>
                <input type="text" class="form-control" id="username" required />
                <div class="invalid-feedback">Username is required.</div>
            </div>
            <div class="col-md-12">
                <label for="password" class="form-label">Password <span class="text-danger">*</span></label>
                <input type="password" class="form-control" id="password" required />
                <div class="invalid-feedback">Password is required.</div>
            </div>
            <div class="col-md-12 pt-2">
                <label class="form-label d-block">Status</label>
                <div class="form-check form-check-inline">
                    <input class="form-check-input" type="radio" name="status" id="statusActive" checked value="active" required />
                    <label selected class="form-check-label" for="statusActive">Active</label>
                </div>
                <div class="form-check form-check-inline">
                    <input class="form-check-input" type="radio" name="status" id="statusInactive" value="inactive" />
                    <label class="form-check-label" for="statusInactive">Inactive</label>
                </div>
                <div class="invalid-feedback">Status is required.</div>
            </div>
        </div>
    </div>

    <!-- Personal Info Tab -->
    <div class="tab-pane fade" id="infoTab" role="tabpanel">
        <div class="row g-3 pt-2">
            <div class="col-md-6">
                <label for="lastName" class="form-label">Last Name <span class="text-danger">*</span></label>
                <input type="text" class="form-control" id="lastName" required />
                <div class="invalid-feedback">Last name is required.</div>
            </div>
            <div class="col-md-6">
                <label for="firstName" class="form-label">First Name <span class="text-danger">*</span></label>
                <input type="text" class="form-control" id="firstName" required />
                <div class="invalid-feedback">First name is required.</div>
            </div>
            <div class="col-md-6">
                <label for="middleInitial" class="form-label">Middle Initial</label>
                <input type="text" class="form-control" id="middleInitial" maxlength="1" />
            </div>
        </div>
    </div>

```

```

</div>
<div class="col-md-6">
  <label for="position" class="form-label">Position <span class="text-danger">*</span></label>
  <input type="text" class="form-control" id="position" required />
  <div class="invalid-feedback">Position is required.</div>
</div>
<div class="col-md-6">
  <label for="duty" class="form-label">Duty <span class="text-danger">*</span></label>
  <select class="form-select" id="duty" required>
    <option value="" disabled selected>Select Duty</option>
    <option value="Supervisor">Supervisor</option>
    <option value="Field Work">Field Work</option>
    <option value="Admin Work">Admin Work</option>
    <option value="Others">Others</option>
  </select>
  <div class="invalid-feedback">Please select duty.</div>
</div>
<div class="col-md-6">
  <label for="natureOfAppointment" class="form-label">Nature of Appointment <span class="text-danger">*</span></label>
  <select class="form-select" id="natureOfAppointment" required>
    <option value="" disabled selected>Select an option</option>
    <option value="Permanent">Permanent</option>
    <option value="COS">COS</option>
    <option value="Temporary">Temporary</option>
    <option value="Coterminous">Coterminous</option>
    <option value="Fixed Term">Fixed Term</option>
    <option value="Substitute">Substitute</option>
    <option value="Provisional">Provisional</option>
  </select>
  <div class="invalid-feedback">Please select nature of appointment.</div>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="modal-footer px-4 py-3">
  <button type="submit" class="btn btn-3d btn-success w-100 fw-semibold">Add User</button>
</div>
</form>
</div>
</div>
</div>

<!-- Edit User Modal with Tabs -->
<div class="modal fade" id="editUserModal" tabindex="-1" aria-labelledby="editUserModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content shadow-lg border-0">
      <form id="editUserForm">
        <div class="modal-header bg-info">
          <h5 class="modal-title text-white" id="editUserModalLabel">

```

```
<i class="bi bi-pencil-square me-2"></i>Edit User
</h5>
<button type="button" class="btn-close btn-3d btn-close-white" data-bs-dismiss="modal" aria-
label="Close"></button>
</div>

<div class="modal-body">
<input type="hidden" id="editUserId" />

<!-- Nav tabs -->
<ul class="nav nav-tabs mb-3" id="editUserTabs" role="tablist">
<li class="nav-item" role="presentation">
<button
  class="nav-link active"
  id="edit-account-tab"
  data-bs-toggle="tab"
  data-bs-target="#editAccountTab"
  type="button"
  role="tab"
  aria-controls="editAccountTab"
  aria-selected="true"
>
  Account Info
</button>
</li>
<li class="nav-item" role="presentation">
<button
  class="nav-link"
  id="edit-personal-tab"
  data-bs-toggle="tab"
  data-bs-target="#editPersonalTab"
  type="button"
  role="tab"
  aria-controls="editPersonalTab"
  aria-selected="false"
>
  Personal Info
</button>
</li>
</ul>

<!-- Tab panes -->
<div class="tab-content">
<!-- Account Info Tab -->
<div
  class="tab-pane fade show active"
  id="editAccountTab"
  role="tabpanel"
  aria-labelledby="edit-account-tab"
>
<div class="mb-3">
<label for="editUsername" class="form-label">Username</label>
<input type="text" class="form-control" id="editUsername" required />

```

```

</div>

<div class="mb-3">
  <label for="editPassword" class="form-label">New Password</label>
  <input type="password" class="form-control" id="editPassword" placeholder="Leave blank to keep current" />
</div>

<div class="mb-3">
  <label for="editConfirmPassword" class="form-label">Confirm Password</label>
  <input type="password" class="form-control" id="editConfirmPassword" placeholder="Leave blank to keep current" />
</div>

<div class="mb-3">
  <label class="form-label">Status</label>
  <div class="form-check">
    <input class="form-check-input" type="radio" name="editStatus" id="editStatusActive" value="active" />
    <label class="form-check-label" for="editStatusActive">Active</label>
  </div>
  <div class="form-check">
    <input class="form-check-input" type="radio" name="editStatus" id="editStatusInactive" value="inactive" />
    <label class="form-check-label" for="editStatusInactive">Inactive</label>
  </div>
</div>

<!-- Personal Info Tab --&gt;
&lt;div class="tab-pane fade" id="editPersonalTab" role="tabpanel" aria-labelledby="edit-personal-tab"&gt;
&lt;div class="row"&gt;
  &lt;div class="col-md-6 mb-3"&gt;
    &lt;label for="editLastName" class="form-label"&gt;Last Name&lt;/label&gt;
    &lt;input type="text" class="form-control" id="editLastName" required /&gt;
  &lt;/div&gt;
  &lt;div class="col-md-6 mb-3"&gt;
    &lt;label for="editFirstName" class="form-label"&gt;First Name&lt;/label&gt;
    &lt;input type="text" class="form-control" id="editFirstName" required /&gt;
  &lt;/div&gt;
&lt;/div&gt;
&lt;div class="mb-3"&gt;
  &lt;label for="editMiddleInitial" class="form-label"&gt;Middle Initial&lt;/label&gt;
  &lt;input type="text" class="form-control" id="editMiddleInitial" maxlength="1" /&gt;
&lt;/div&gt;

&lt;div class="mb-3"&gt;
  &lt;label for="editPosition" class="form-label"&gt;Position&lt;/label&gt;
  &lt;input type="text" class="form-control" id="editPosition" placeholder="Enter position" required /&gt;
&lt;/div&gt;

&lt;div class="mb-3"&gt;
</pre>

```

```

<label for="editDuty" class="form-label">Duty</label>
<select class="form-select" id="editDuty" required>
  <option value="" disabled selected>Select duty</option>
  <option value="Supervisor">Supervisor</option>
  <option value="Field Work">Field Work</option>
  <option value="Admin Work">Admin Work</option>
  <option value="Others">Others</option>
</select>
</div>

<div class="mb-3">
  <label for="editNatureOfAppointment" class="form-label">Nature of Appointment</label>
  <select class="form-select" id="editNatureOfAppointment" required>
    <option value="" disabled selected>Select an option</option>
    <option value="Permanent">Permanent</option>
    <option value="COS">COS</option>
    <option value="Temporary">Temporary</option>
    <option value="Coterminous">Coterminous</option>
    <option value="Fixed Term">Fixed Term</option>
    <option value="Substitute">Substitute</option>
    <option value="Provisional">Provisional</option>
  </select>
</div>
</div>

<div class="modal-footer">
  <button type="submit" class="btn btn-3d btn-success w-100">
    <i class="bi bi-save me-1"></i>Update User
  </button>
</div>
</form>
</div>
</div>
</div>

</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
<script type="module" src="js/user/user-init.js"></script>

```

consumable.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>WRUS Portal</title>

```

```

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet">
<link rel="stylesheet" href="css/style.css" />
<link rel="stylesheet" href="css/consumable.css" />
<link rel="icon" type="image/png" href="images/favicon-32x32.png">
</head>
<body>
<div class="d-flex flex-column flex-md-row" id="wrapper">
  <div id="sidebarContainer"></div>
  <!-- Page Content -->
  <div class="container-fluid p-2" id="page-content">
    <div class="mb-2">
      <h2 class="file-tab-heading">Items</h2>
    </div>
    <!-- Search Bar + Add Button Row -->
    <div class="d-flex flex-column flex-sm-row justify-content-end align-items-stretch gap-2 mb-3">
      <div class="input-group water-theme" style="min-width: 250px;">
        <span class="input-group-text bg-info text-dark border-0"><i class="bi bi-search"></i></span>
        <input type="text" id="searchInput" class="form-control border-0" placeholder="Search for consumable items..." />
      </div>
      <button id="showAddFormBtn" class="btn btn-3d btn-primary" style="white-space: nowrap;">
        <i class="bi bi-plus-circle me-1"></i> Add
      </button>
    </div>
    <div class="d-flex align-items-center justify-content-between gap-2 my-2 flex-wrap">
      <!-- Left: Page size selector -->
      <div class="d-flex align-items-center gap-2">
        <label for="pageSizeSelect" class="form-label mb-0">Show</label>
        <select id="pageSizeSelect" class="form-select form-select-sm" style="width: auto;">
          <option value="10">10</option>
          <option value="25">25</option>
          <option value="50">50</option>
        </select>
        <span>entries</span>
      </div>
      <!-- Right: Buttons side-by-side -->
      <div class="d-flex align-items-center gap-2">
        <button class="btn btn-3d btn-success shadow-sm px-3 py-2 d-flex align-items-center gap-2" id="exportBtn">
          <span class="d-inline-flex justify-content-center align-items-center bg-light text-success rounded-circle" style="width: 24px; height: 24px;">
            <i class="bi bi-download"></i>
          </span>
        </button>
      </div>
    </div>
  </div>
</body>

```

```

    Export
    </button>

    <button id="refreshBtn" class="btn btn-3d btn-primary shadow-sm px-3 py-2 d-flex align-items-center gap-2">
        <i class="bi bi-arrow-clockwise me-1"></i> Refresh
    </button>
</div>
</div>

<div class="table-responsive">
    <table id="consumableTable">
        <thead class="table-success">
            <tr>
                <th>CID</th>
                <th>Specification</th>
                <th>Qty</th>
                <th>UoM</th>
                <th>Edit</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody id="consumableBody">
            <!-- Filled dynamically -->
        </tbody>
    </table>
<div class="d-flex justify-content-center my-3" id="paginationContainer"></div>
</div>

<!-- Add Item Modal -->
<div class="modal fade" id="addItemModal" tabindex="-1" aria-labelledby="addItemModalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg modal-dialog-centered modal-dialog-scrollable">
        <div class="modal-content shadow-lg rounded-4">
            <div class="modal-header bg-info rounded-top-4">
                <h5 class="modal-title fw-semibold text-white" id="addItemModalLabel">Add New Item</h5>
                <button type="button" class="btn-3d btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                <form id="addItemForm">
                    <div class="row g-3">
                        <div class="col-12">
                            <label for="newSpec" class="form-label fw-semibold">Specification <span class="text-danger">*</span></label>
                            <input type="text" class="form-control" id="newSpec" placeholder="Enter specification" required />
                        </div>
                        <div class="col-md-6">
                            <label for="newQty" class="form-label fw-semibold">Quantity <span class="text-danger">*</span></label>
                            <input type="number" class="form-control" id="newQty" placeholder="Enter quantity" min="1" required />
                        </div>
                        <div class="col-md-6">

```

```

        <label for="newUnit" class="form-label fw-semibold">Unit of Measurement <span class="text-danger">*</span></label>
        <input type="text" class="form-control" id="newUnit" placeholder="e.g., pcs, box, set" required />
    </div>

    <div class="col-12">
        <label for="newRemarks" class="form-label fw-semibold">Remarks</label>
        <textarea class="form-control" id="newRemarks" rows="3" placeholder="Optional notes or details"></textarea>
    </div>

</div>
</form>
</div>

<div class="modal-footer bg-light rounded-bottom-4">
    <button type="button" class="btn btn-3d btn-primary" id="addItemBtn">
        <i class="bi bi-plus-lg me-1"></i> Add Item
    </button>
    <button type="button" class="btn btn-3d btn-outline-secondary" data-bs-dismiss="modal">
        <i class="bi bi-x-lg me-1"></i> Cancel
    </button>
</div>
</div>
</div>

<!-- Edit Item Modal -->
<div class="modal fade" id="editModal" tabindex="-1" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered modal-dialog-scrollable">
        <div class="modal-content shadow-lg rounded-4">
            <div class="modal-header bg-warning rounded-top-4">
                <h5 class="modal-title text-muted fw-semibold">Edit Item</h5>
                <button type="button" class="btn-3d btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                <input type="hidden" id="editCID" />
                <div class="mb-3">
                    <label for="editSpec" class="form-label fw-semibold">Specification <span class="text-danger">*</span></label>
                    <input type="text" id="editSpec" class="form-control" placeholder="Update specification" required />
                </div>
                <div class="mb-3">
                    <label for="editUnit" class="form-label fw-semibold">Unit of Measurement <span class="text-danger">*</span></label>
                    <input type="text" id="editUnit" class="form-control" placeholder="e.g., pcs, set, box" required />
                </div>
                <div class="form-check">
                    <input class="form-check-input" type="checkbox" value="" id="priorityCheckbox">
                </div>
            </div>
        </div>
    </div>
</div>

```

```

<label class="form-check-label" for="priorityCheckbox">
  Priority?
</label>
</div>
</div>

<div class="modal-footer bg-light rounded-bottom-4">
  <button type="button" class="btn btn-3d btn-primary" id="saveEditBtn">
    <i class="bi bi-save me-1"></i> Save Changes
  </button>
  <button type="button" class="btn btn-3d btn-outline-secondary" data-bs-dismiss="modal">
    <i class="bi bi-x-lg me-1"></i> Cancel
  </button>
</div>
</div>
</div>


<div class="modal fade" id="actionModal" tabindex="-1" aria-labelledby="actionModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-sm modal-dialog-centered">
    <div class="modal-content bg-light text-dark border rounded-3 shadow">
      <div class="modal-header border-bottom border-secondary-subtle">
        <h5 class="modal-title d-flex align-items-center gap-2" id="actionModalLabel">
          <i class="bi bi-tools"></i> Item Actions
        </h5>
        <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body d-grid gap-2 px-3 py-3">
        <!-- Add Stock -->
        <button id="addStockBtn" class="btn btn-3d d-flex align-items-center justify-content-center gap-2 text-white bg-success border border-success rounded-2">
          <i class="bi bi-box-arrow-in-down"></i> Add Stock
        </button>
        <!-- Assign Item -->
        <button id="assignItemBtn" class="btn btn-3d d-flex align-items-center justify-content-center gap-2 text-white bg-primary border border-primary rounded-2">
          <i class="bi bi-box-arrow-up"></i> Assign Item
        </button>
        <!-- Reassign Item -->
        <button id="reassignItemBtn" class="btn btn-3d d-flex align-items-center justify-content-center gap-2 text-white bg-danger border border-danger rounded-2">
          <i class="bi bi-arrow-left-right"></i> Reassign Item
        </button>
        <!-- View Ledger -->
      </div>
    </div>
  </div>
</div>

```

```

<button id="viewLedgerBtn" class="btn btn-3d d-flex align-items-center justify-content-center gap-2 text-white bg-info border border-info rounded-2">
  <i class="bi bi-journal-text"></i> View Ledger
</button>

</div>
</div>
</div>

<!-- Add Stock Modal -->
<div class="modal fade" id="addStockModal" tabindex="-1" aria-labelledby="addStockModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered modal-sm">
    <div class="modal-content rounded-3 shadow-sm">
      <div class="modal-header bg-info border-secondary-subtle">
        <h5 class="modal-title text-white fw-semibold" id="addStockModalLabel">
          <i class="bi bi-box-arrow-in-down me-2"></i>Add Stock
        </h5>
        <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <div class="mb-3">
          <label for="stockAmount" class="form-label">Quantity to Add</label>
          <input type="number" id="stockAmount" class="form-control" min="1" required placeholder="Enter quantity" />
        </div>
        <div class="mb-2">
          <label for="stockRemarks" class="form-label">Remarks (optional)</label>
          <textarea id="stockRemarks" class="form-control" rows="3" placeholder="Enter remarks..."></textarea>
        </div>
      </div>
      <div class="modal-footer border-top border-secondary-subtle">
        <button id="confirmAddStockBtn" class="btn btn-3d btn-success w-100">
          <i class="bi bi-check-circle me-1"></i>Confirm
        </button>
      </div>
    </div>
  </div>
</div>

<!-- Assign Item Modal -->
<div class="modal fade" id="assignModal" tabindex="-1" aria-labelledby="assignModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content rounded-3 shadow-sm" style="max-width: 500px; margin: auto;">
      <div class="modal-header bg-primary border-bottom border-secondary-subtle">
        <h5 class="modal-title text-white fw-semibold" id="assignModalLabel">
          <i class="bi bi-box-arrow-up me-2"></i>Assign Item

```

```

</h5>
<button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
</div>

<div class="modal-body py-2">
<form id="assignForm">
<div class="mb-2">
<label for="assignQty" class="form-label small mb-1">Quantity</label>
<input type="number" class="form-control form-control-sm" id="assignQty" min="1" required placeholder="Enter quantity" />
</div>

<div class="mb-2">
<label for="userSelect" class="form-label small mb-1">Assign To</label>
<select class="form-select form-select-sm" id="userSelect" required>
<option value="" disabled selected>Select user</option>
</select>
</div>

<div class="mb-2">
<label for="assignRemarks" class="form-label small mb-1">Remarks (optional)</label>
<textarea class="form-control form-control-sm" id="assignRemarks" rows="2" placeholder="Enter remarks..."></textarea>
</div>
</form>
</div>

<div class="modal-footer py-2 border-top border-secondary-subtle">
<button type="button" id="confirmAssignBtn" class="btn btn-3d btn-primary w-100">
<i class="bi bi-send-check me-1"></i>Assign
</button>
</div>
</div>
</div>

<!-- Ledger Modal -->
<div class="modal fade" id="ledgerModal" tabindex="-1" aria-labelledby="ledgerModalLabel" aria-hidden="true">
<div class="modal-dialog modal-xl modal-dialog-scrollable">
<div class="modal-content rounded-3">
<div class="modal-header bg-info">
<h5 class="modal-title text-white" id="ledgerModalLabel">Ledger Entries</h5>
<button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
<div class="px-3 pt-2">
<p class="text-white mb-0">Stocks Left: <strong id="totalQtyDisplay">Loading...</strong></p>
</div>
</div>
<div class="modal-body">
<iframe id="pdfPreviewFrame" width="100%" height="600px" frameborder="0"></iframe>
</div>
</div>
</div>

```

```

</div>
</div>

<!-- Reassign Item Modal -->



<div class="modal-dialog modal-lg" > <!-- Use modal-lg for wider table -->
<div class="modal-content">
<div class="modal-header bg-danger">
<h5 class="modal-title text-white" id="tableModalLabel">Reassign Item</h5>
<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
</div>
<div class="modal-body">

<!-- Table inside modal -->
<div class="table-responsive">
<table class="table table-bordered table-hover">
<thead class="table-light">
<tr>
<th>Date</th>
<th>Qty</th>
<th>Assigned To</th>
<th>Remarks</th>
<th>Action</th>
</tr>
</thead>
<tbody id="reassignTableBody"></tbody>
</table>
</div>
</div>
<div class="modal-footer">
<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
</div>
</div>
</div>

<!-- Firebase + Bootstrap -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<!-- jsPDF and AutoTable -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-autotable/3.5.28/jspdf.plugin.autotable.min.js"></script>
<script src="https://cdn.sheetjs.com/xlsx-latest/package/dist/xlsx.full.min.js"></script>
<script type="module" src="js/consumable/consumable-init.js"></script>
</body>
</html>


```

ics.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>WRUS Portal</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
<link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet">
<link rel="stylesheet" href="css/style.css" />
<link rel="stylesheet" href="css/ics.css" />
</head>
<body>
<div class="d-flex flex-column flex-md-row" id="wrapper">
  <div id="sidebarContainer"></div>
  <!-- Page Content -->
  <div class="container-fluid p-2" id="page-content">
    <div class="d-flex flex-column flex-md-row align-items-start align-items-md-center justify-content-between flex-wrap gap-2 mb-4">
      <h2 class="file-tab-heading">ICS / PAR</h2>
      <div class="d-flex flex-column flex-sm-row align-items-stretch justify-content-end gap-2 w-100 w-md-auto">
        <div class="input-group" style="min-width: 250px;">
          <span class="input-group-text">Search</span>
          <input type="text" id="searchBar" class="form-control" placeholder="Enter item name..." />
        </div>
        <button id="refreshBtn" class="btn btn-3d btn-3d btn-primary shadow-sm px-3 py-2 d-flex align-items-center justify-content-center">
          <i class="bi bi-arrow-clockwise me-1"></i>
        </button>
        <button id="addBtn" class="btn btn-3d btn-primary" style="white-space: nowrap;">
          <i class="bi bi-plus-circle me-1"></i>
        </button>
      </div>
    </div>
    <div class="table-responsive">
      <table>
        <thead class="table-primary">
          <tr>
            <th>ICS/PAR No.</th>
            <th>Assigned To</th>
            <th>Description</th>
            <th>Date</th>
            <th>Total Cost</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody id="icsTableBody">
          <!-- Rows will be inserted via JavaScript -->
        </tbody>
      </table>
    </div>
  </div>
</body>

```

```

<!-- Pagination Control -->
<nav>
  <ul class="pagination justify-content-center" id="paginationControlsICS">
    <!-- Dynamic pagination goes here -->
  </ul>
</nav>

<!-- Add ICS Modal -->
<div class="modal fade" id="addICSModal" tabindex="-1" aria-labelledby="addICSModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg modal-dialog-scrollable">
    <div class="modal-content shadow-lg">
      <form id="addICSForm" class="needs-validation" novalidate>
        <div class="modal-header bg-primary">
          <h5 class="modal-title text-white" id="addICSModalLabel">Add ICS Entry</h5>
          <button type="button" class="btn-close btn-3d btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>
        </div>

        <div class="modal-body px-4 pt-3 pb-1">
          <ul class="nav nav-tabs mb-3" id="icsTab" role="tablist">
            <li class="nav-item" role="presentation">
              <button class="nav-link active" id="tab-item" data-bs-toggle="tab" data-bs-target="#itemTab" type="button" role="tab">Item Details</button>
            </li>
            <li class="nav-item" role="presentation">
              <button class="nav-link" id="tab-assignment" data-bs-toggle="tab" data-bs-target="#assignmentTab" type="button" role="tab">Assignment Details</button>
            </li>
          </ul>
        </div>
      </form>
    </div>
  </div>
</div>

<div class="tab-content" id="icsTabContent">
  <!-- Item Details Tab -->
  <div class="tab-pane fade show active" id="itemTab" role="tabpanel">
    <div class="row g-3">
      <div class="col-md-6">
        <label for="icsNo" class="form-label">ICS/PAR No.</label>
        <input type="text" class="form-control" id="icsNo" required />
        <div class="invalid-feedback">ICS/PAR No. is required.</div>
      </div>

      <div class="col-md-6">
        <label for="serialNumber" class="form-label">Serial Number</label>
        <input type="text" class="form-control" id="serialNumber" />
      </div>
    </div>
    <div class="col-12">
      <label for="description" class="form-label">Description</label>
      <textarea class="form-control" id="description" rows="2" required></textarea>
      <div class="invalid-feedback">Description is required.</div>
    </div>
  </div>
</div>

```



```
<label for="qty" class="form-label">Quantity</label>
<input type="number" class="form-control" id="qty" min="0" required />
<div class="invalid-feedback">Quantity is required.</div>
</div>

<div class="col-md-4">
  <label for="unit" class="form-label">Unit of Measurement</label>
  <input type="text" class="form-control" id="unit" required />
  <div class="invalid-feedback">Unit is required.</div>
</div>

<div class="col-md-4">
  <label for="unitCost" class="form-label">Unit Cost (₱)</label>
  <input type="number" class="form-control" id="unitCost" min="0" step="0.01" required />
  <div class="invalid-feedback">Unit cost is required.</div>
</div>

<div class="col-md-6">
  <label class="form-label">Total Cost (₱)</label>
  <p id="totalCostDisplay" class="form-control-plaintext fw-semibold text-primary">₱0.00</p>
</div>
</div>
</div>

<!-- Assignment Details Tab --&gt;
&lt;div class="tab-pane fade" id="assignmentTab" role="tabpanel"&gt;
  &lt;div class="row g-3 mt-1"&gt;
    &lt;div class="col-md-6"&gt;
      &lt;label for="assignedTo" class="form-label"&gt;Assigned To&lt;/label&gt;
      &lt;select class="form-select" id="assignedTo" required&gt;
        &lt;option value=""&gt;Select User&lt;/option&gt;
      &lt;/select&gt;
      &lt;div class="invalid-feedback"&gt;Assigned user is required.&lt;/div&gt;
    &lt;/div&gt;
    &lt;div class="col-md-6"&gt;
      &lt;label for="dateIssued" class="form-label"&gt;Date Issued&lt;/label&gt;
      &lt;input type="date" class="form-control" id="dateIssued" required /&gt;
      &lt;div class="invalid-feedback"&gt;Date issued is required.&lt;/div&gt;
    &lt;/div&gt;
    &lt;div class="col-md-6"&gt;
      &lt;label for="status" class="form-label"&gt;Status&lt;/label&gt;
      &lt;select class="form-select" id="status" required&gt;
        &lt;option value=""&gt;Select Status&lt;/option&gt;
        &lt;option value="Original"&gt;Original&lt;/option&gt;
        &lt;option value="Transferred"&gt;Transferred&lt;/option&gt;
        &lt;option value="RTS"&gt;RTS&lt;/option&gt;
      &lt;/select&gt;
      &lt;div class="invalid-feedback"&gt;Status is required.&lt;/div&gt;
    &lt;/div&gt;
    &lt;div class="col-md-6"&gt;</pre>
```

```

<label for="attachment" class="form-label">Attach File (PDF, max 1MB)</label>
<input type="file" class="form-control" id="attachment" accept="application/pdf" />
</div>

<div class="col-12">
  <label for="remarks" class="form-label">Remarks</label>
  <textarea class="form-control" id="remarks" rows="2" required></textarea>
  <div class="invalid-feedback">Remarks are required.</div>
</div>
</div>
</div>
</div>
</div>

<div class="modal-footer px-4 py-3">
  <button type="submit" class="btn btn-3d btn-primary">Save ICS</button>
  <button type="button" class="btn btn-3d btn-secondary" data-bs-dismiss="modal">Cancel</button>
</div>
</form>
</div>
</div>
</div>

<!-- Edit ICS Modal -->
<div class="modal fade" id="editICSModal" tabindex="-1" aria-labelledby="editICSMModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      <form id="editICSForm" class="needs-validation" novalidate>
        <div class="modal-header bg-primary">
          <h5 class="modal-title text-white" id="editICSMModalLabel">Edit ICS Entry</h5>
          <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
        </div>

        <div class="modal-body px-4 pt-3 pb-1">
          <input type="hidden" id="editDocId">
          <input type="hidden" id="editCurrentAttachmentUrl">

          <ul class="nav nav-tabs mb-3" id="editIcsTab" role="tablist">
            <li class="nav-item" role="presentation">
              <button class="nav-link active" id="edit-tab-item" data-bs-toggle="tab" data-bs-target="#editItemTab" type="button" role="tab">Item Details</button>
            </li>
            <li class="nav-item" role="presentation">
              <button class="nav-link" id="edit-tab-assignment" data-bs-toggle="tab" data-bs-target="#editAssignmentTab" type="button" role="tab">Assignment Details</button>
            </li>
          </ul>

          <div class="tab-content" id="editIcsTabContent">
            <!-- Item Details Tab -->
            <div class="tab-pane fade show active" id="editItemTab" role="tabpanel">
              <div class="row g-3">
                <div class="col-md-6">

```

```

<label for="editIcsNo" class="form-label">ICS/PAR No.</label>
<input type="text" class="form-control" id="editIcsNo" required />
<div class="invalid-feedback">ICS/PAR No. is required.</div>
</div>

<div class="col-md-6">
  <label for="editSerialNumber" class="form-label">Serial Number</label>
  <input type="text" class="form-control" id="editSerialNumber" />
</div>

<div class="col-12">
  <label for="editDescription" class="form-label">Description</label>
  <textarea class="form-control" id="editDescription" rows="2" required></textarea>
  <div class="invalid-feedback">Description is required.</div>
</div>

<div class="col-md-4">
  <label for="editQty" class="form-label">Quantity</label>
  <input type="number" class="form-control" id="editQty" min="0" required />
  <div class="invalid-feedback">Quantity is required.</div>
</div>

<div class="col-md-4">
  <label for="editUnit" class="form-label">Unit of Measurement</label>
  <input type="text" class="form-control" id="editUnit" required />
  <div class="invalid-feedback">Unit is required.</div>
</div>

<div class="col-md-4">
  <label for="editUnitCost" class="form-label">Unit Cost (₱)</label>
  <input type="number" class="form-control" id="editUnitCost" min="0" step="0.01" required />
  <div class="invalid-feedback">Unit cost is required.</div>
</div>

<div class="col-md-6">
  <label class="form-label">Total Cost (₱)</label>
  <p id="editTotalCostDisplay" class="form-control-plaintext fw-semibold text-primary">₱0.00</p>
</div>
</div>

<!-- Assignment Details Tab --&gt;
&lt;div class="tab-pane fade" id="editAssignmentTab" role="tabpanel"&gt;
  &lt;div class="row g-3 mt-1"&gt;
    &lt;div class="col-md-6"&gt;
      &lt;label for="editAssignedTo" class="form-label"&gt;Assigned To&lt;/label&gt;
      &lt;select class="form-select" id="editAssignedTo" required&gt;
        &lt;option value=""&gt;Select User&lt;/option&gt;
      &lt;/select&gt;
      &lt;div class="invalid-feedback"&gt;Assigned user is required.&lt;/div&gt;
    &lt;/div&gt;
    &lt;div class="col-md-6"&gt;
</pre>

```

```

<label for="editDateIssued" class="form-label">Date Issued</label>
<input type="date" class="form-control" id="editDateIssued" required />
<div class="invalid-feedback">Date issued is required.</div>
</div>

<div class="col-md-6">
    <label for="editStatus" class="form-label">Status</label>
    <select class="form-select" id="editStatus" required>
        <option value="">Select Status</option>
        <option value="Original">Original</option>
        <option value="Transferred">Transferred</option>
        <option value="RTS">RTS</option>
    </select>
    <div class="invalid-feedback">Status is required.</div>
</div>

<div class="col-md-6">
    <label for="editAttachment" class="form-label">Attach File (PDF, max 1MB)</label>
    <input type="file" class="form-control" id="editAttachment" accept="application/pdf" />
</div>

<div class="col-12">
    <label for="editRemarks" class="form-label">Remarks</label>
    <textarea class="form-control" id="editRemarks" rows="2" required></textarea>
    <div class="invalid-feedback">Remarks are required.</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="modal-footer px-4 py-3">
    <button type="submit" class="btn btn-3d btn-primary">
        <i class="bi bi-save"></i> Update ICS
    </button>
    <button type="button" id="deleteICSCButton" class="btn btn-3d btn-danger" disabled>
        <i class="bi bi-trash"></i> Delete
    </button>
    <button type="button" class="btn btn-3d btn-secondary" data-bs-dismiss="modal">
        <i class="bi bi-x-circle"></i> Cancel
    </button>
</div>
</div>
</div>
</div>
</div>

</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/browser-image-compression@2.0.2/dist/browser-image-compression.js"></script>
<script type="module" src="js/ics/ics-init.js"></script>

```

```
</body>  
</html>
```

inventory-summary.html

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8" />  
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
  <title>WRUS Portal</title>  
  <link  
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"  
    rel="stylesheet" />  
  </>  
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet" />  
  
  <link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />  
  <link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet" />  
  <link rel="stylesheet" href="css/style.css" />  
  <link rel="stylesheet" href="css/inventory-summary.css" />  
</head>  
<body>  
  <div class="d-flex flex-column flex-md-row" id="wrapper">  
    <div id="sidebarContainer"></div>  
    <!-- Page Content -->  
    <div class="container-fluid p-2" id="page-content">  
      <h2 class="file-tab-heading">Ledger/ICS Summary</h2>  
      <!-- Nav tabs -->  
      <ul class="nav nav-tabs mb-3" id="summaryTabs" role="tablist">  
        <li class="nav-item" role="presentation">  
          <button  
            class="nav-link active"  
            id="chart-tab"  
            data-bs-toggle="tab"  
            data-bs-target="#chartTab"  
            type="button"  
            role="tab"  
            aria-controls="chartTab"  
            aria-selected="true"  
          >  
            Ledger Report  
          </button>  
        </li>  
        <li class="nav-item" role="presentation">  
          <button  
            class="nav-link"  
            id="table-tab"  
            data-bs-toggle="tab"  
            data-bs-target="#tableTab"  
            type="button"  
            role="tab"  
            aria-controls="tableTab"  
          >
```

```
        aria-selected="false"
    >
    Inventory Summary
</button>
</li>
</ul>


<div class="tab-content" id="summaryTabsContent">

    
    <div
        class="tab-pane fade show active"
        id="chartTab"
        role="tabpanel"
        aria-labelledby="chart-tab"
    >
        
        <div class="card mb-4 shadow-sm">
            <div class="card-header bg-primary text-white">
                <h5 class="mb-0">Latest Ledger Report</h5>
            </div>
            <div class="card-body">
                <div id="ledgerContainer"></div>
            </div>
        </div>

        
        <div class="card mb-4 shadow-sm">
            <div class="card-header bg-success text-dark">
                <h5 class="mb-0">Consumable Item Analytics (Priority Items Only)</h5>
            </div>
            <div class="card-body p-0">
                <div class="table-responsive">
                    <table class="table table-hover table-striped table-bordered mb-0">
                        <thead class="table-dark">
                            <tr>
                                <th>CID</th>
                                <th>Specification</th>
                                <th>Unit</th>
                                <th>Stock</th>
                                <th>Consumed (12 mo)</th>
                                <th>Lifespan</th>
                                <th>Avg/Month</th>
                                <th>Days Left</th>
                                <th>Resupply?</th>
                                <th>Recommended</th>
                            </tr>
                        </thead>
                        <tbody id="priorityBody"></tbody>
                        <tfoot class="table-light">
                            <tr>
                                <td colspan="9" class="text-muted small">
```

```

<strong>Notes:</strong>
<ul class="mb-0 ps-3">
    <li><strong>Consumed (12 mo)</strong> = total quantity used in the past rolling 12 months.</li>
    <li><strong>Lifespan</strong> = estimated from usage rates and current stock.</li>
    <li><strong>Avg/Month</strong> = Consumed (12 mo) ÷ 12 months.</li>
    <li><strong>Days Left</strong> = stock ÷ average daily consumption.</li>
    <li><strong>Resupply?</strong> flags items projected to run out within a year.</li>
    <li><strong>Recommended</strong> recommended quantity of items to be procured for next year; Formula
        ((12*monthly consumption) + 10% allowance).</li>
    </ul>
</td>
</tr>
</tfoot>
</table>
</div>
</div>
</div>

<!-- Critically Low Supplies -->
<div class="card shadow-sm">
    <div class="card-header bg-danger text-white">
        <h5 class="mb-0">Critically Low Consumable Supplies</h5>
    </div>
    <div class="card-body">
        <div id="lowSupplyChartContainer"></div>
    </div>
</div>
</div>

<!-- Inventory Summary Tab -->
<div
    class="tab-pane fade"
    id="tableTab"
    role="tabpanel"
    aria-labelledby="table-tab"
>
<div class="table-responsive shadow-sm rounded bg-white p-3" style="height: auto; overflow-y: visible;">
    <!-- Search Bar -->
    <div class="d-flex justify-content-between align-items-center mb-4">
        <input
            type="text"
            id="searchInput"
            class="form-control w-50"
            placeholder="Search by name...">
    </div>
    <table class="table table-striped">
        <thead class="table-primary text-center">
            <tr>
                <th>Name</th>
                <th>Consumable</th>

```

```

<th>ICS</th>
</tr>
</thead>
<tbody id="usersTableBody">
<!-- JavaScript will inject rows here -->
</tbody>
</table>

<!-- Pagination container - insert your pagination here -->
<nav id="paginationNav" aria-label="Page navigation" class="mt-3"></nav>

</div>
</div>

</div>

<!-- Consumable Modal-->
<div
  class="modal fade"
  id="consumableModal"
  tabindex="-1"
  aria-labelledby="consumableModalLabel"
  aria-hidden="true"
>
<div class="modal-dialog modal-lg modal-dialog-centered">
<div class="modal-content">
<div class="modal-header bg-info">
  <h5 class="modal-title text-white" id="consumableModalLabel">Consumable Details</h5>
  <button
    type="button"
    class="btn-close btn-3d"
    data-bs-dismiss="modal"
    aria-label="Close"
  ></button>
</div>
<div class="modal-body">
  <p class="text-center">Loading...</p>
</div>
<div class="modal-footer">
  <button
    type="button"
    class="btn btn-3d btn-secondary"
    data-bs-dismiss="modal"
  >
    Close
  </button>
</div>
</div>
</div>
</div>

<!-- ICS Items User Modal-->
<div class="modal fade" id="propertyModal" tabindex="-1" aria-labelledby="propertyModalLabel" aria-hidden="true">

```

```

<div class="modal-dialog modal-xl modal-dialog-centered modal-dialog-scrollable">
  <div class="modal-content shadow-lg rounded-3 small">

    <!-- Modal Header -->
    <div class="modal-header bg-info flex-wrap gap-3 justify-content-between align-items-center border-bottom-0 px-4 pt-4 pb-2">
      <div>
        <h5 class="modal-title fw-semibold text-primary" id="propertyModalLabel">
          Property Accountability of <span id="fullName" class="text-white"></span>
        </h5>
      </div>
      <div class="d-flex flex-column align-items-end">
        <button type="button" class="btn-close btn-3d mb-2" data-bs-dismiss="modal" aria-label="Close"></button>
        <strong>Total Amount: <span id="totalAmount" class="text-dark"></span></strong>
      </div>
    </div>

    <!-- Modal Body -->
    <div class="modal-body px-4 pt-2 pb-0 d-flex flex-column" style="gap: 0.5rem;">
      <div id="scrollableTableWrapper" style="flex: 1; overflow-x: auto; overflow-y: auto;">
        <div style="min-width: 1000px;">
          <table class="table table-bordered table-sm align-middle text-center mb-0 property-accountability-table">
            <thead class="table-light text-nowrap">
              <tr>
                <th rowspan="2">Qty</th>
                <th rowspan="2">UoM</th>
                <th rowspan="2">Description</th>
                <th rowspan="2">SN</th>
                <th colspan="2" rowspan="2">Amount</th>
                <th rowspan="2">ICS No</th>
                <th rowspan="2">Date Issued</th>
                <th rowspan="2">Remarks</th>
                <th rowspan="2">PDF</th>
              </tr>
              <tr>
                <th>Unit Cost</th>
                <th>Total Cost</th>
              </tr>
            </thead>
            <tbody id="icsTableBody">
              <!-- ICS data rows go here dynamically -->
            </tbody>
          </table>
        </div>
      </div>
    </div>

    <!-- Modal Footer -->
    <div class="modal-footer justify-content-between align-items-center px-4 border-top-0 pb-4">
      <button id="generatePdfBtn" class="btn btn-3d btn-outline-danger" data-userid="USER_ID_HERE">
        <i class="bi bi-file-earmark-pdf me-1"></i> Generate PDF
      </button>
      <div></div>
    </div>
  </div>
</div>

```

```
</div>

</div>
</div>
</div>

<!-- ICS PDF Viewer Modal -->
<div class="modal fade" id="ICS-pdfModal" tabindex="-1">
  <div class="modal-dialog modal-fullscreen modal-dialog-centered modal-dialog-scrollable">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">ICS PDF Report</h5>
        <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal"></button>
      </div>
      <div class="modal-body">
        <!-- PDF content will be injected here -->
      </div>
    </div>
  </div>
</div>
</div>
</div>
</div>

</div>
</div>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-autotable/3.5.28/jspdf.plugin.autotable.min.js"></script>
<!-- Chart.js -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<!-- Chart.js datalabels plugin -->
<script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels"></script>
<script type="module" src="js/summary/summary-init.js"></script>
</body>
</html>
```

permit.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>WRUS Portal</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
    <link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
    <link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet" />
    <link rel="stylesheet" href="css/style.css" />
    <link rel="stylesheet" href="css/permit.css" />
  </head>
  <body>
    <div class="d-flex flex-column flex-md-row" id="wrapper">
```

```

<div id="sidebarContainer"></div>
<div class="container-fluid p-2" id="page-content">
  <!-- Heading -->
  <div class="mb-2">
    <h2 class="file-tab-heading">Water Permit
    </h2>
  </div>

  <div class="d-flex flex-wrap align-items-center mb-3 gap-2">
    <!-- Search Bar (Always Visible) -->
    <div class="flex-grow-1">
      <div class="input-group water-search">
        <input type="text" id="searchBar" class="form-control" placeholder="Search Permit No. or City">
        <span class="input-group-text">
          <i class="bi bi-search"></i>
        </span>
      </div>
    </div>
    <!-- Add Permit Button (Hide on small screens) -->
    <div class="flex-shrink-0 d-none d-sm-block">
      <button class="btn btn-3d btn-primary shadow-sm px-3 py-2 d-flex align-items-center gap-2" id="addPermitBtn">
        <span class="d-inline-flex justify-content-center align-items-center bg-light text-primary rounded-circle" style="width: 24px; height: 24px;">
          <i class="bi bi-plus-lg"></i>
        </span>
        Add
      </button>
    </div>
  </div>

  <button id="toggleFilter" class="btn btn-sm d-none d-sm-inline-block">
    <i class="bi bi-eye-slash"></i> Hide Filters
  </button>

  <!-- Filter Section -->
  <div class="mb-3" id="filterSection">
    <!-- Legend -->
    <div class="d-flex align-items-center mb-1">
      <span class="badge bg-info me-2" style="width: 20px; height: 20px;">&ampnbsp</span>
      <span>Rows highlighted in <strong>blue</strong> mean the permittee is <strong>already visited</strong>. </span>
    </div>

    <!-- Checkbox Filter -->
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="visitedFilter">
      <label class="form-check-label" for="visitedFilter">
        <strong>Show only unvisited permits</strong><br>
        <small>(If checked, shows permittees that are <strong>not yet visited</strong>. If unchecked, shows <strong>all permittees</strong>.)</small>
      </label>
    </div>
  </div>
</div>

```

```

</div>

<div class="form-check">
  <input class="form-check-input" type="checkbox" id="showVisitedFilter">
  <label class="form-check-label" for="showVisitedFilter">
    <strong>Show only visited permits</strong><br>
    <small>(If checked, shows permittees that are <strong>already visited</strong>. If unchecked, shows <strong>all permittees</strong>.)</small>
  </label>
</div>
</div>

<!-- Pagination and Button Controls -->
<div class="d-flex flex-column flex-sm-row justify-content-between align-items-start align-items-sm-center gap-2 mb-1 mt-1 flex-wrap">

  <!-- Left Side -->
  <div class="d-flex flex-wrap flex-sm-nowrap align-items-center gap-3">

    <!-- Rows per page (hidden on small screens) -->
    <div class="rows-per-page-container d-none d-sm-flex align-items-center">
      <label for="rowsPerPage" class="form-label me-2 mb-0">Rows per page:</label>
      <select id="rowsPerPage" class="form-select d-inline-block w-auto">
        <option value="10" selected>10</option>
        <option value="25">25</option>
        <option value="50">50</option>
      </select>
    </div>
  </div>

  <!-- Pagination and Refresh Button aligned inline on small screens -->
  <div class="d-flex align-items-center gap-2">
    <nav aria-label="Page navigation">
      <ul class="pagination mb-0" id="pagination"></ul>
    </nav>
  </div>

  <!-- Refresh Button (Always visible) -->
  <button id="refreshBtn" class="btn btn-3d btn-primary shadow-sm px-2 py-2 d-flex align-items-center justify-content-center">
    <i class="bi bi-arrow-clockwise"></i>
  </button>
</div>
</div>

<!-- Right Side: Export Button (hidden on small screens) -->
<div class="d-none d-sm-flex flex-row align-items-center gap-2">
  <button class="btn btn-3d btn-success shadow-sm px-3 py-2 d-flex align-items-center gap-2" id="exportPermitsBtn">
    <span class="d-inline-flex justify-content-center align-items-center bg-light text-success rounded-circle" style="width: 24px; height: 24px;">
      <i class="bi bi-download"></i>
    </span>
    Export
  </button>
</div>

```

```

</div>

<!-- Data Table -->
<div class="table-responsive">
  <table id="permitTable">
    <thead class="table-primary">
      <tr>
        <th>Permit No</th>
        <th>Permittee</th>
        <th>Diversion Point</th>
        <th>Latitude</th>
        <th>Longitude</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody id="permitTableBody">
      <!-- Dynamic rows will be inserted here -->
    </tbody>
  </table>
</div>

<!-- Add Permit Modal -->
<div class="modal fade" id="addPermitModal" tabindex="-1" aria-labelledby="addPermitModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">

      <!-- Modal Header -->
      <div class="modal-header bg-primary">
        <h5 class="modal-title text-white" id="addPermitModalLabel">
          <i class="bi bi-plus-circle-fill me-2"></i> Add Water Permit
        </h5>
        <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>

      <!-- Modal Body -->
      <div class="modal-body">
        <form id="permitForm" class="needs-validation" novalidate>

          <!-- Nav Tabs -->
          <ul class="nav nav-tabs mb-3" id="permitTab" role="tablist">
            <li class="nav-item" role="presentation">
              <button class="nav-link active" id="basic-tab" data-bs-toggle="tab" data-bs-target="#basic" type="button" role="tab">Basic Info</button>
            </li>
            <li class="nav-item" role="presentation">
              <button class="nav-link" id="technical-tab" data-bs-toggle="tab" data-bs-target="#technical" type="button" role="tab">Technical Info</button>
            </li>
          </ul>

          <!-- Tab Contents -->
          <div class="tab-content" id="permitTabContent">

```

```

<!-- Tab 1: Basic Info -->
<div class="tab-pane fade show active" id="basic" role="tabpanel">
<div class="row g-3">

    <!-- Permit No -->
    <div class="col-md-6">
        <label class="form-label">
            <i class="bi bi-hash me-1"></i> Permit No. <span class="text-danger">*</span>
        </label>
        <input type="text" class="form-control" id="permitNo" required>
        <div class="invalid-feedback">Permit No. is required.</div>
    </div>

    <!-- Permittee -->
    <div class="col-md-6">
        <label class="form-label">
            <i class="bi bi-person-badge me-1"></i> Permittee <span class="text-danger">*</span>
        </label>
        <textarea class="form-control" id="permittee" rows="2" required></textarea>
        <div class="invalid-feedback">Permittee is required.</div>
    </div>

    <!-- Mailing Address (not required) -->
    <div class="col-md-6">
        <label class="form-label">
            <i class="bi bi-geo-alt me-1"></i> Mailing Address <i>(not required)</i>
        </label>
        <textarea class="form-control" id="mailingAddress" disabled rows="2"></textarea>
    </div>

    <!-- Diversion Point -->
    <div class="col-md-6">
        <label class="form-label">
            <i class="bi bi-geo-fill me-1"></i> Diversion Point <span class="text-danger">*</span>
        </label>
        <select class="form-select" id="diversionPoint" required>
            <option value="">-- Select City --</option>
        </select>
        <div class="invalid-feedback">Diversion Point is required.</div>
    </div>

    <!-- Latitude -->
    <div class="col-md-6">
        <label class="form-label"><i class="bi bi-compass me-1"></i> Latitude <span class="text-danger">*</span></label>
        <div class="row g-2">
            <div class="col-3">
                <input type="number" class="form-control" id="latDegrees" placeholder="°" required>
            </div>
            <div class="col-3">
                <input type="number" class="form-control" id="latMinutes" placeholder="'" required>
            </div>
            <div class="col-3">

```



```

<input type="number" class="form-control" id="latSeconds" placeholder="" required step="any">
</div>
<div class="col-3">
<select class="form-select" id="latDirection" required disabled>
<option value="N">N</option>
<option value="S">S</option>
</select>
</div>
</div>
<div class="invalid-feedback">Complete Latitude is required.</div>
</div>

<!-- Longitude -->
<div class="col-md-6">
<label class="form-label"><i class="bi bi-compass me-1"></i> Longitude <span class="text-danger">*</span></label>
<div class="row g-2">
<div class="col-3">
<input type="number" class="form-control" id="lonDegrees" placeholder="" required>
</div>
<div class="col-3">
<input type="number" class="form-control" id="lonMinutes" placeholder="" required>
</div>
<div class="col-3">
<input type="number" class="form-control" id="lonSeconds" placeholder="" required step="any">
</div>
<div class="col-3">
<select class="form-select" id="lonDirection" required disabled>
<option value="E">E</option>
<option value="W">W</option>
</select>
</div>
</div>
<div class="invalid-feedback">Complete Longitude is required.</div>
</div>

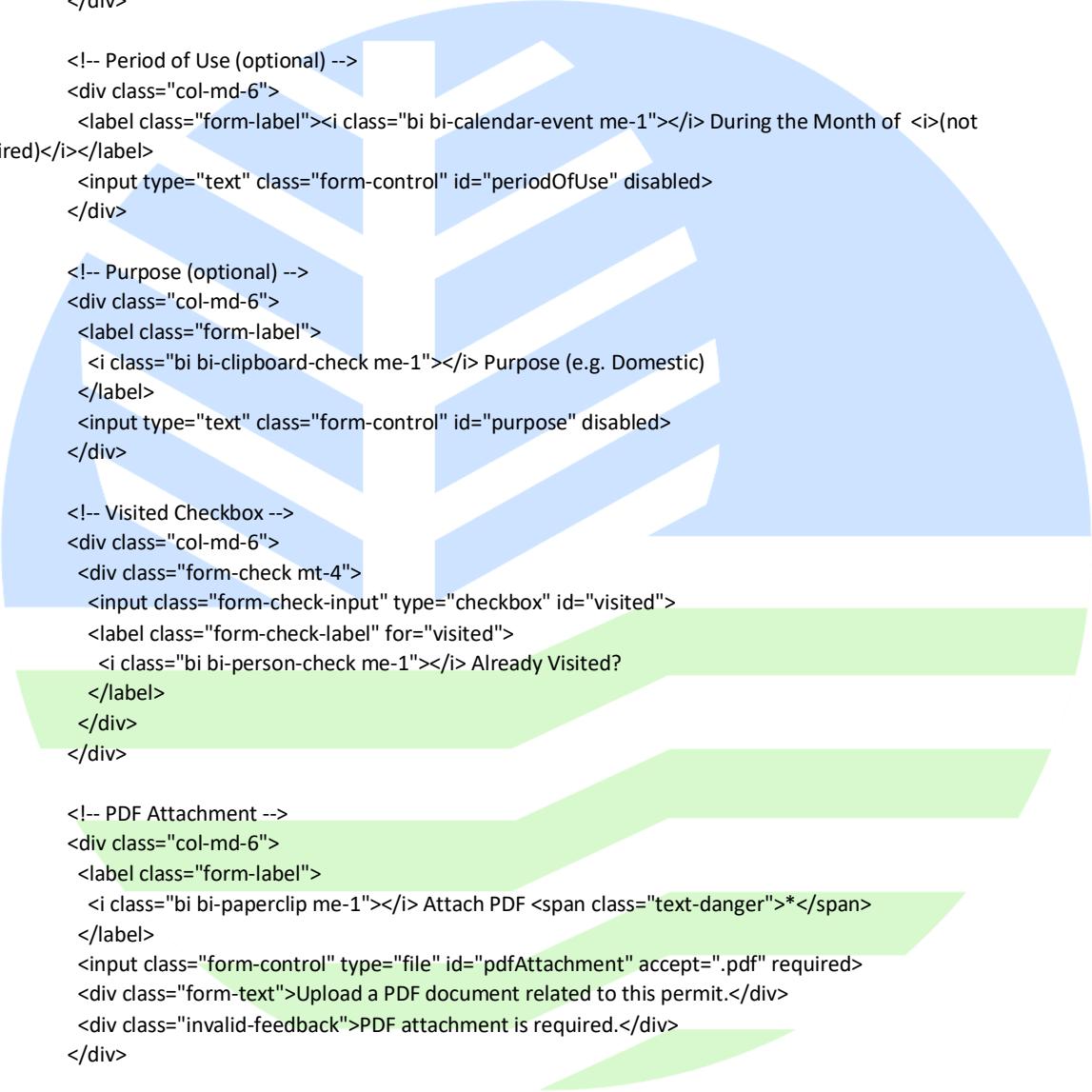
</div>
</div>

<!-- Tab 2: Technical Info -->
<div class="tab-pane fade" id="technical" role="tabpanel">
<div class="row g-3">

<!-- Water Source (optional) -->
<div class="col-md-6">
<label class="form-label"><i class="bi bi-droplet-half me-1"></i> Water Source <i>(not required)</i></label>
<input type="text" class="form-control" id="waterSource" disabled>
</div>

<!-- Water Diversion (optional) -->
<div class="col-md-6">
<label class="form-label"><i class="bi bi-arrow-repeat me-1"></i> Water Diversion <i>(not required)</i></label>
<input type="text" class="form-control" id="waterDiversion" disabled>

```



```

</div>

<!-- Flow Rate (optional) -->
<div class="col-md-6">
  <label class="form-label">
    <i class="bi bi-speedometer me-1"></i> Flow Rate (L/s) <i>(not required)</i>
  </label>
  <input type="number" class="form-control" id="flowRate" min="0" step="any" disabled>
</div>

<!-- Period of Use (optional) -->
<div class="col-md-6">
  <label class="form-label"><i class="bi bi-calendar-event me-1"></i> During the Month of <i>(not required)</i></label>
  <input type="text" class="form-control" id="periodOfUse" disabled>
</div>

<!-- Purpose (optional) -->
<div class="col-md-6">
  <label class="form-label">
    <i class="bi bi-clipboard-check me-1"></i> Purpose (e.g. Domestic)
  </label>
  <input type="text" class="form-control" id="purpose" disabled>
</div>

<!-- Visited Checkbox -->
<div class="col-md-6">
  <div class="form-check mt-4">
    <input class="form-check-input" type="checkbox" id="visited">
    <label class="form-check-label" for="visited">
      <i class="bi bi-person-check me-1"></i> Already Visited?
    </label>
  </div>
</div>

<!-- PDF Attachment -->
<div class="col-md-6">
  <label class="form-label">
    <i class="bi bi-paperclip me-1"></i> Attach PDF <span class="text-danger">*</span>
  </label>
  <input class="form-control" type="file" id="pdfAttachment" accept=".pdf" required>
  <div class="form-text">Upload a PDF document related to this permit.</div>
  <div class="invalid-feedback">PDF attachment is required.</div>
</div>

<!-- Is Cancelled Checkbox -->
<div class="col-md-6">
  <div class="form-check mt-4">
    <input class="form-check-input" type="checkbox" id="isCancelled">
    <label class="form-check-label" for="isCancelled">
      <i class="bi bi-check2-circle me-1"></i> Is Cancelled?
    </label>
  </div>
</div>

```

```

        </div>
    </div>
</div>

</div> <!-- End Tab Content -->

</form>
</div>

<!-- Modal Footer -->
<div class="modal-footer">
    <button type="button" class="btn btn-3d btn-outline-secondary" data-bs-dismiss="modal">
        <i class="bi bi-x-circle me-1"></i> Close
    </button>
    <button type="button" class="btn btn-3d btn-primary" id="savePermitBtn">
        <i class="bi bi-save me-1"></i> Save Permit
    </button>
</div>

</div>
</div>
</div>

<!-- Edit Permit Modal -->
<div class="modal fade" id="editPermitModal" tabindex="-1" aria-labelledby="editPermitModalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">

            <!-- Modal Header -->
            <div class="modal-header bg-warning">
                <h5 class="modal-title text-muted" id="editPermitModalLabel">
                    <i class="bi bi-pencil-square me-2"></i> Edit Water Permit
                </h5>
                <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>

            <!-- Modal Body -->
            <div class="modal-body">
                <form id="editPermitForm" class="needs-validation" novalidate>

                    <!-- Nav Tabs -->
                    <ul class="nav nav-tabs mb-3" id="editPermitTab" role="tablist">
                        <li class="nav-item" role="presentation">
                            <button class="nav-link active" id="edit-basic-tab" data-bs-toggle="tab" data-bs-target="#edit-basic" type="button" role="tab">Basic Info</button>
                        </li>
                        <li class="nav-item" role="presentation">
                            <button class="nav-link" id="edit-technical-tab" data-bs-toggle="tab" data-bs-target="#edit-technical" type="button" role="tab">Technical Info</button>
                        </li>
                    </ul>

                    <!-- Tab Contents -->

```

```

<div class="tab-content" id="editPermitTabContent">

    <!-- Tab 1: Basic Info -->
    <div class="tab-pane fade show active" id="edit-basic" role="tabpanel">
        <div class="row g-3">

            <!-- Permit No (REQUIRED) -->
            <div class="col-md-6">
                <label class="form-label"><i class="bi bi-hash me-1"></i> Permit No.</label>
                <input type="text" class="form-control" id="editPermitNo" required>
                <div class="invalid-feedback">Permit No. is required.</div>
            </div>

            <!-- Permittee (REQUIRED) -->
            <div class="col-md-6">
                <label class="form-label">
                    <i class="bi bi-person-badge me-1"></i> Permittee
                </label>
                <textarea class="form-control" id="editPermittee" rows="2" required></textarea>
                <div class="invalid-feedback">Permittee is required.</div>
            </div>

            <!-- Mailing Address (OPTIONAL) -->
            <div class="col-md-6">
                <label class="form-label">
                    <i class="bi bi-geo-alt me-1"></i> Mailing Address
                </label>
                <textarea class="form-control" id="editMailingAddress" rows="2"></textarea>
            </div>

            <!-- Diversion Point (REQUIRED) -->
            <div class="col-md-6">
                <label class="form-label">
                    <i class="bi bi-geo-fill me-1"></i> Diversion Point
                </label>
                <select class="form-select" id="editDiversionPoint" required>
                    <option value="">-- Select Diversion Point --</option>
                </select>
                <div class="invalid-feedback">Diversion Point is required.</div>
            </div>

            <!-- Latitude (REQUIRED) -->
            <div class="col-md-6">
                <label class="form-label"><i class="bi bi-compass me-1"></i> Latitude</label>
                <div class="row g-2">
                    <div class="col-3">
                        <input type="number" class="form-control" id="editLatDegrees" placeholder="°" required>
                    </div>
                    <div class="col-3">
                        <input type="number" class="form-control" id="editLatMinutes" placeholder="'" required>
                    </div>
                    <div class="col-3">
                        <input type="number" class="form-control" id="editLatSeconds" placeholder=""" required step="any">
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

</div>
<div class="col-3">
<select class="form-select" id="editLatDirection" required disabled>
<option selected value="N">N</option>
<option value="S">S</option>
</select>
</div>
</div>
<div class="invalid-feedback">Complete Latitude is required.</div>
</div>

<!-- Longitude (REQUIRED) -->
<div class="col-md-6">
<label class="form-label"><i class="bi bi-compass me-1"></i> Longitude</label>
<div class="row g-2">
<div class="col-3">
<input type="number" class="form-control" id="editLonDegrees" placeholder="°" required>
</div>
<div class="col-3">
<input type="number" class="form-control" id="editLonMinutes" placeholder="" required>
</div>
<div class="col-3">
<input type="number" class="form-control" id="editLonSeconds" placeholder="" required step="any">
</div>
<div class="col-3">
<select class="form-select" id="editLonDirection" required disabled>
<option selected value="E">E</option>
<option value="W">W</option>
</select>
</div>
</div>
<div class="invalid-feedback">Complete Longitude is required.</div>
</div>

</div>
</div>

<!-- Tab 2: Technical Info -->
<div class="tab-pane fade" id="edit-technical" role="tabpanel">
<div class="row g-3">

<!-- Water Source (OPTIONAL) -->
<div class="col-md-6">
<label class="form-label"><i class="bi bi-droplet-half me-1"></i> Water Source</label>
<input type="text" class="form-control" id="editWaterSource">
</div>

<!-- Water Diversion (OPTIONAL) -->
<div class="col-md-6">
<label class="form-label"><i class="bi bi-arrow-repeat me-1"></i> Water Diversion</label>
<input type="text" class="form-control" id="editWaterDiversion">
</div>

```

```

<!-- Flow Rate (OPTIONAL) -->
<div class="col-md-6">
  <label class="form-label">
    <i class="bi bi-speedometer me-1"></i> Flow Rate (L/s)
  </label>
  <input type="number" class="form-control" id="editFlowRate" min="0" step="any">
</div>

<!-- Period of Use (OPTIONAL) -->
<div class="col-md-6">
  <label class="form-label"><i class="bi bi-calendar-event me-1"></i> During the Month of</label>
  <input type="text" class="form-control" id="editPeriodOfUse">
</div>

<!-- Purpose (OPTIONAL) -->
<div class="col-md-6">
  <label class="form-label">
    <i class="bi bi-clipboard-check me-1"></i> Purpose (e.g. Domestic)
  </label>
  <input type="text" class="form-control" id="editPurpose">
</div>

<!-- Visited Checkbox -->
<div class="col-md-6">
  <div class="form-check mt-4">
    <input class="form-check-input" type="checkbox" id="editVisited">
    <label class="form-check-label" for="editVisited">
      <i class="bi bi-person-check me-1"></i> Already Visited?
    </label>
  </div>
</div>

<!-- PDF Attachment -->
<div class="col-md-6">
  <label class="form-label">
    <i class="bi bi-paperclip me-1"></i> Replace PDF (Optional)
  </label>
  <input class="form-control" type="file" id="editPdfAttachment" accept=".pdf">
  <input type="hidden" id="editPdfExistingUrl">
  <div class="form-text">Upload a new PDF to replace the existing one (optional).</div>
</div>

<!-- Is Cancelled Checkbox -->
<div class="col-md-6">
  <div class="form-check mt-4">
    <input class="form-check-input" type="checkbox" id="editIsCancelled">
    <label class="form-check-label" for="editIsCancelled">
      <i class="bi bi-check2-circle me-1"></i> Is Cancelled?
    </label>
  </div>
</div>

</div>

```

```

</div>

</div> <!-- End Tab Content -->

</form>
</div>

<!-- Modal Footer -->
<div class="modal-footer">
  <button type="button" class="btn btn-3d btn-outline-secondary" data-bs-dismiss="modal">
    <i class="bi bi-x-circle me-1"></i> Cancel
  </button>
  <button type="button" class="btn btn-3d btn-warning" id="updatePermitBtn">
    <i class="bi bi-save me-1"></i> Update Permit
  </button>
</div>

</div>
</div>
</div>

<!-- Geotagged Photo Modal -->
<div class="modal fade" id="imageUploadModal" tabindex="-1" aria-labelledby="imageUploadModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg modal-dialog-centered">
    <div class="modal-content">

      <!-- Modal Header -->
      <div class="modal-header bg-primary">
        <h5 class="modal-title text-white" id="imageUploadModalLabel">
          <i class="bi bi-image-fill me-2"></i> Upload and Preview Image
        </h5>
        <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>

      <!-- Modal Body -->
      <div class="modal-body text-center">

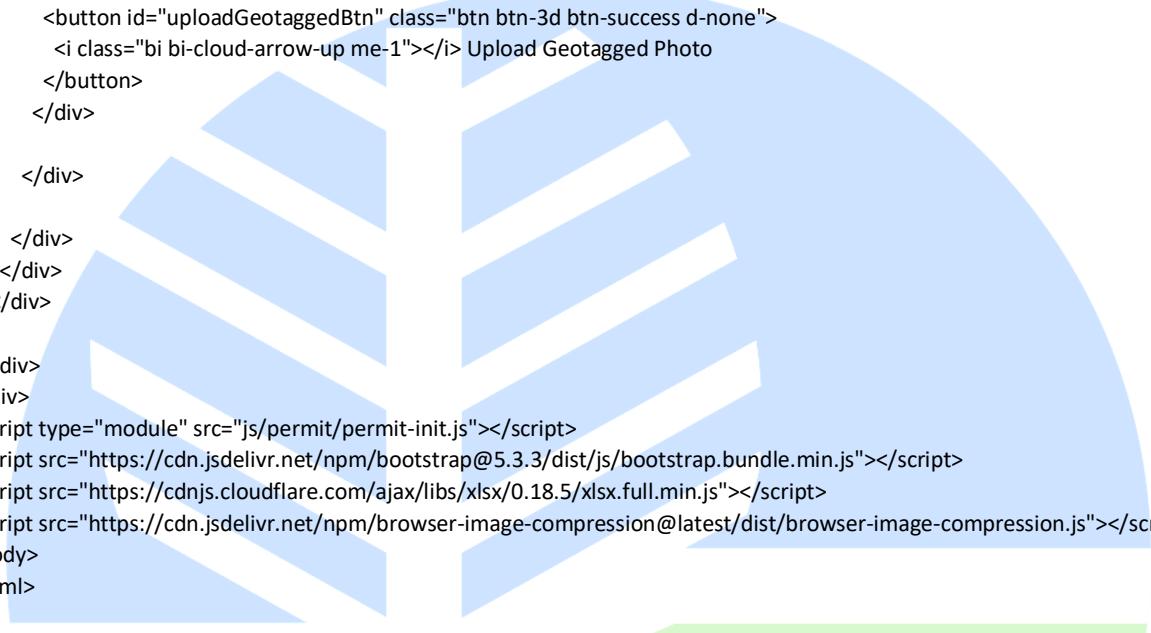
        <!-- Clickable Image Upload + Drop Zone -->
        <label id="dropZone" class="d-inline-block position-relative border rounded p-2" style="cursor: pointer; display: inline-block;">
          
          <input type="file" id="imageInput" accept="image/*" hidden>
        </label>

        <input type="file" id="imageInput" accept="image/*" hidden>
      </div>

      <div class="small text-muted">Click or drag an image here to upload</div>

      <input type="hidden" id="currentGeotaggedUrl">
    </div>
  </div>
</div>

```

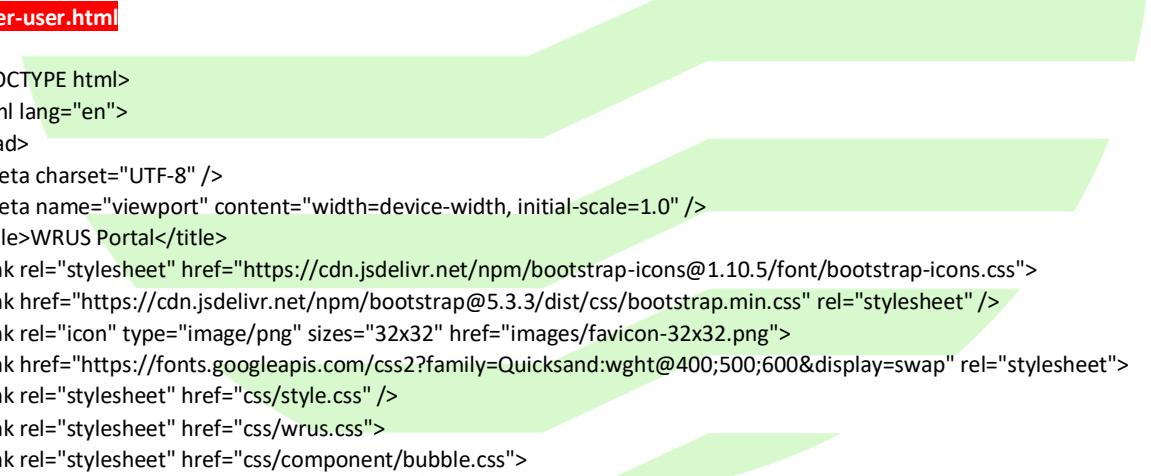


```
<input type="hidden" id="currentPermitNo">


<div class="d-flex justify-content-center gap-2 mt-3 flex-wrap">
  <a id="viewImageLink" href="#" target="_blank" class="btn btn-3d btn-outline-primary d-none">
    <i class="bi bi-eye me-1"></i> View Full Image
  </a>

  <button id="uploadGeotaggedBtn" class="btn btn-3d btn-success d-none">
    <i class="bi bi-cloud-arrow-up me-1"></i> Upload Geotagged Photo
  </button>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<script type="module" src="js/permit/permit-init.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/browser-image-compression@latest/dist/browser-image-compression.js"></script>
</body>
</html>
```

water-user.html



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>WRUS Portal</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
  <link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
  <link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="css/style.css" />
  <link rel="stylesheet" href="css/wrus.css" />
  <link rel="stylesheet" href="css/component/bubble.css" />
</head>
<body>
  <div class="d-flex flex-column flex-md-row" id="wrapper">
    <div id="sidebarContainer"></div>
    <div class="container-fluid p-2" id="page-content">
      <h2 class="file-tab-heading mb-2">
        Water Users and Sources
      </h2>
      <div class="container mb-3">
```

```

<div class="d-flex flex-column flex-lg-row align-items-stretch gap-2">

    <!-- Search Input -->
    <div class="flex-grow-1">
        <input type="text" id="searchInput" class="form-control form-water" placeholder="Search water user or source...">
    </div>

    <!-- Buttons -->
    <div class="d-flex flex-wrap flex-lg-nowrap gap-2 justify-content-center justify-content-lg-start">
        <button id="refreshBtn" class="btn btn-3d btn-primary shadow-sm px-3 py-2 d-flex align-items-center justify-content-center">
            <i class="bi bi-arrow-clockwise me-1"></i>
        </button>

        <button class="btn btn-3d btn-success shadow-sm px-3 py-2 d-flex align-items-center gap-2" id="exportBtn">
            <span class="d-inline-flex justify-content-center align-items-center bg-light text-success rounded-circle" style="width: 24px; height: 24px;">
                <i class="bi bi-download"></i>
            </span>
            Export
        </button>

        <button class="btn btn-3d btn-primary shadow-sm px-3 py-2 d-flex align-items-center justify-content-center" data-bs-toggle="modal" data-bs-target="#addwusModal">
            <i class="bi bi-plus-circle"></i>
        </button>
    </div>
    </div>
</div>

<!-- Water Source Filter Toggle Button -->
<button class="btn btn-outline-dark mb-2" id="toggleWaterSourceFilterBtn">
    Show Water Source Filter
</button>

<!-- Water Source Filter Section (Hidden by Default via CSS) -->
<div class="mb-3 water-source-filter-section d-none" id="waterSourceFilterSection">
    <!-- Checkbox Filter: Show Only Water Sources -->
    <div class="form-check">
        <input class="form-check-input" type="checkbox" id="waterSourceOnlyFilter">
        <label class="form-check-label" for="waterSourceOnlyFilter">
            <strong>Show only water sources</strong><br>
            <small>(If checked, shows only entries marked as <strong>Water Source</strong>. If unchecked, shows <strong>all entries</strong>.)</small>
        </label>
    </div>

    <!-- Checkbox Filter: Show Only Non-Water Sources -->
    <div class="form-check">
        <input class="form-check-input" type="checkbox" id="nonWaterSourceOnlyFilter">
        <label class="form-check-label" for="nonWaterSourceOnlyFilter">
            <strong>Show only non-water sources</strong><br>
        </label>
    </div>
</div>

```

(If checked, shows only entries **not marked** as a Water Source. If unchecked, shows **all** entries.)

```

</label>
</div>
</div>

<div class="table-responsive">
<table>
<thead>
<tr class="bg-info text-white">
<th class="align-middle">Owner</th>
<th class="align-middle">Street / Bldg</th>
<th class="align-middle">Barangay</th>
<th class="align-middle">City</th>
<th class="align-middle">Latitude</th>
<th class="align-middle">Longitude</th>
<th class="align-middle">Year</th>
<th class="align-middle">Action</th>
</thead>
<tbody id="waterUserTableBody">
<!-- Dynamic rows will be inserted here --&gt;
&lt;/tbody&gt;
&lt;/table&gt;
&lt;div class="d-flex justify-content-center mt-3"&gt;
&lt;nav&gt;
&lt;ul id="pagination" class="pagination"&gt;&lt;/ul&gt;
&lt;/nav&gt;
&lt;/div&gt;
&lt;/div&gt;
</pre>


<!-- Add Water User Modal -->



```

<div class="modal fade" id="addwusModal" tabindex="-1" aria-labelledby="addwusModalLabel" aria-hidden="true">
<div class="modal-dialog modal-lg modal-dialog-centered">
<div class="modal-content shadow-sm">
<form id="addWusForm" class="needs-validation" novalidate>
<div class="modal-header bg-primary">
<h5 class="modal-title text-white" id="addwusModalLabel">
<i class="bi bi-droplet-half me-2"></i>Add Water User
</h5>
<button type="button" class="btn-close btn-3d btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>
</div>
<div class="modal-body px-4 py-3">
<div class="row g-3">
<input type="hidden" id="wusGeotagUrl" value="">
<div class="col-md-6">
<label for="permitNoInput" class="form-label">
Permit No.
</label><small class="text-muted fst-italic">(if permittee)</small>
<div class="input-group mt-1">

```


```

```

<span class="input-group-text"><i class="bi bi-search"></i></span>
<input type="text" class="form-control" id="permitNoInput" placeholder="Enter Permit No.">
</div>
<div id="permitSuggestions" class="list-group mt-1" style="z-index: 1055; position: absolute;"></div>
</div>

<div class="col-md-6">
<label for="nameOfWaterUser" class="form-label">Owner <span class="text-danger">*</span></label>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-person-fill"></i></span>
<input type="text" class="form-control" id="nameOfWaterUser" required>
<div class="invalid-feedback">Please enter the water user's name.</div>
</div>
</div>

<div class="col-md-6">
<label for="city" class="form-label">City <span class="text-danger">*</span></label>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-building"></i></span>
<select class="form-select" id="city" required>
<option selected disabled value="">Select City</option>
</select>
</div>
</div>

<div class="col-md-6">
<label for="barangay" class="form-label">Barangay <span class="text-danger">*</span></label>
<small class="text-muted fst-italic">(Exclude prefixes like "Brgy.")</small>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-geo-alt-fill"></i></span>
<input type="text" class="form-control" id="barangay" required placeholder="Enter Barangay">
<div class="invalid-feedback">Please enter the barangay.</div>
</div>
</div>

<div class="col-md-6">
<label for="street" class="form-label">Street / Bldg Name <span class="text-danger">*</span></label>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-signpost-split-fill"></i></span>
<input type="text" class="form-control" id="street" required>
<div class="invalid-feedback">Please enter the street or building name.</div>
</div>
</div>

<div class="col-md-6">
<label for="type" class="form-label">Type of Water Source <span class="text-danger">*</span></label>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-toggles2"></i></span>
<input type="text" class="form-control" id="type" required>
<div class="invalid-feedback">Please specify the water user type.</div>
</div>
</div>

```

```

<div class="col-6">
  <label for="status" class="form-label small">Source Status</label>
  <!-- Select dropdown -->
  <select class="form-select form-select" id="status">
    </select>
  </div>

  <!-- Is Water Source -->
  <div class="col-6">
    <div class="form-check">
      <input class="form-check-input mt-4" type="checkbox" id="isWaterSource">
      <label class="form-check-label mt-4" for="isWaterSource">
        Is Water Source?
      </label>
    </div>
  </div>

  <div class="col-md-6">
    <label for="latitude" class="form-label">Latitude <span class="text-danger">*</span></label>
    <div class="input-group">
      <span class="input-group-text"><i class="bi bi-compass-fill"></i></span>
      <input type="text" class="form-control" id="latitude"
        pattern="^-[1-8]?[0-9](\.\d+)?|90(\.\d+)?$" required>
      <select class="form-select" disabled style="max-width: 70px;">
        <option value="N" selected>N</option>
      </select>
      <div class="form-text"></div>
      <div class="invalid-feedback">Please enter a valid latitude.</div>
    </div>
  </div>
</div>

<div class="col-md-6">
  <label for="longitude" class="form-label">Longitude <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-compass"></i></span>
    <input type="text" class="form-control" id="longitude"
      pattern="^-?((1[0-7][0-9])|([1-9]?[0-9]))(\.\d+)?|180(\.\d+)?$" required>
      <select class="form-select" disabled style="max-width: 70px;">
        <option value="W" selected>W</option>
      </select>
      <div class="form-text"></div>
      <div class="invalid-feedback">Please enter a valid longitude.</div>
    </div>
  </div>
</div>

<div class="row">
  <div class="col-md-6">
    <label for="monthConducted" class="form-label">Month Inspected</label>
    <div class="input-group">
      <span class="input-group-text"><i class="bi bi-calendar-month"></i></span>
      <select class="form-select" id="monthConducted">
        <!-- Options will be populated by JS -->
      </select>
    </div>
  </div>
</div>

```

```

        </div>
    </div>

    <div class="col-md-6">
        <label for="yearConducted" class="form-label">Year Inspected <span class="text-danger">*</span></label>
        <div class="input-group">
            <span class="input-group-text"><i class="bi bi-calendar3"></i></span>
            <input type="number" class="form-control" id="yearConducted" required min="1900" max="2099" placeholder="e.g., 2025">
            <div class="invalid-feedback">Please enter a valid year.</div>
        </div>
    </div>
</div>

    <div class="col-md-6">
        <label for="representative" class="form-label">Representative</label>
        <div class="input-group">
            <span class="input-group-text"><i class="bi bi-people"></i></span>
            <input type="text" class="form-control" id="representative" name="representative">
        </div>
    </div>
</div>

    <div class="col-md-6">
        <label for="designation" class="form-label">Designation</label>
        <div class="input-group">
            <span class="input-group-text"><i class="bi bi-people"></i></span>
            <input type="text" class="form-control" id="designation" name="designation">
        </div>
    </div>
</div>

    <div class="col-md-6">
        <label for="phone" class="form-label">Phone</label>
        <div class="input-group">
            <span class="input-group-text"><i class="bi bi-telephone"></i></span>
            <input type="text" class="form-control" id="phone" name="phone">
        </div>
    </div>
</div>

    <div class="col-md-6">
        <label for="remarks" class="form-label">Remarks</label>
        <div class="input-group">
            <span class="input-group-text"><i class="bi bi-chat-left-text"></i></span>
            <textarea class="form-control" id="remarks" rows="2"></textarea>
        </div>
    </div>
</div>
</div>

<div class="modal-footer">
    <button type="button" class="btn btn-3d btn-outline-secondary" data-bs-dismiss="modal">Cancel</button>
    <button type="submit" class="btn btn-3d btn-primary">
        <i class="bi bi-save me-1"></i> Save
    </button>

```

```

        </div>
    </form>
</div>
</div>
</div>

<!-- Edit Water User Modal -->
<div class="modal fade" id="editwusModal" tabindex="-1" aria-labelledby="editwusModalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg modal-dialog-centered">
        <div class="modal-content shadow-sm">
            <form id="editWusForm" class="needs-validation" novalidate>
                <input type="hidden" id="editWusId">
                <div class="modal-header bg-warning">
                    <h5 class="modal-title text-muted" id="editwusModalLabel">
                        <i class="bi bi-pencil-square me-2"></i>Edit Water User
                    </h5>
                    <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
                </div>

                <div class="modal-body px-4 py-3">
                    <div class="row g-3">
                        <input type="hidden" id="editWusGeotagUrl" value="">

                        <div class="col-md-6">
                            <label for="permitNoInputEdit" class="form-label">
                                Permit No.
                            </label>
                            <small class="text-muted fst-italic">(if permittee)</small>
                            <div class="input-group mt-1">
                                <span class="input-group-text"><i class="bi bi-search"></i></span>
                                <input type="text" class="form-control" id="permitNoInputEdit" placeholder="Enter Permit No.">
                            </div>
                            <div id="permitSuggestionsEdit" class="list-group mt-1" style="z-index: 1055; position: absolute;"></div>
                        </div>
                    <div class="col-md-6">
                        <label for="editOwner" class="form-label">Owner <span class="text-danger">*</span></label>
                        <div class="input-group">
                            <span class="input-group-text"><i class="bi bi-person-fill"></i></span>
                            <input type="text" class="form-control" id="editOwner" required>
                            <div class="invalid-feedback">Please enter the water user's name.</div>
                        </div>
                    </div>
                </div>
            <div class="col-md-6">
                <label for="editCity" class="form-label">City <span class="text-danger">*</span></label>
                <div class="input-group">
                    <span class="input-group-text"><i class="bi bi-building"></i></span>
                    <select class="form-select" id="editCity" required>
                        <option selected disabled value="">Select City</option>
                    </select>
                </div>
            </div>
        </div>
    </form>
</div>
</div>
</div>
</div>

```

```

</div>

<div class="col-md-6">
  <label for="editBarangay" class="form-label">Barangay <span class="text-danger">*</span></label>
  <small class="text-muted fst-italic">(Exclude prefixes like "Brgy.")</small>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-geo-alt-fill"></i></span>
    <input type="text" class="form-control" id="editBarangay" required placeholder="Enter Barangay">
    <div class="invalid-feedback">Please enter the barangay.</div>
  </div>
</div>

<div class="col-md-6">
  <label for="editStreet" class="form-label">Street / Bldg Name <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-signpost-split-fill"></i></span>
    <input type="text" class="form-control" id="editStreet" required>
    <div class="invalid-feedback">Please enter the street or building name.</div>
  </div>
</div>

<div class="col-md-6">
  <label for="editType" class="form-label">Type of Water Source <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-toggles2"></i></span>
    <input type="text" class="form-control" id="editType" required>
    <div class="invalid-feedback">Please specify the water user type.</div>
  </div>
</div>

<div class="col-6">
  <label for="editStatus" class="form-label small">Source Status</label>
  <!-- Select dropdown -->
  <select class="form-select form-select" id="editStatus">
    </select>
</div>

<!-- Is Water Source -->
<div class="col-6">
  <div class="form-check">
    <input class="form-check-input mt-4" type="checkbox" id="editIsWaterSource">
    <label class="form-check-label mt-4" for="editIsWaterSource">
      Is Water Source?
    </label>
  </div>
</div>

<div class="col-md-6">
  <label for="editLatitude" class="form-label">Latitude <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-compass-fill"></i></span>
    <input type="text" class="form-control" id="editLatitude"
      pattern="^-[1-8]?[0-9](\\.[\\d+])?|90(\\.[0+])?$" required>
  </div>
</div>

```

```

<select class="form-select" disabled style="max-width: 70px;">
  <option value="N" selected>N</option>
</select>
<div class="invalid-feedback">Please enter a valid latitude.</div>
</div>
</div>

<div class="col-md-6">
  <label for="editLongitude" class="form-label">Longitude <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-compass"></i></span>
    <input type="text" class="form-control" id="editLongitude" pattern="^-?(((1[0-7][0-9])|([1-9]?[0-9]))|([1-9]?[0-9]))|([1-9]?[0-9])(\.\d+)?|180(\.\d+)?$ required>
    <select class="form-select" disabled style="max-width: 70px;">
      <option value="W" selected>W</option>
    </select>
    <div class="invalid-feedback">Please enter a valid longitude.</div>
  </div>
</div>

<div class="col-md-6">
  <label for="editMonthConducted" class="form-label">Month Inspected</label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-calendar-month"></i></span>
    <select class="form-select" id="editMonthConducted">
      <!-- Options will be populated by JS -->
    </select>
  </div>
</div>

<!-- Year Conducted -->
<div class="col-md-6">
  <label for="editYearConducted" class="form-label">Year Inspected <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-calendar3"></i></span>
    <input type="number" class="form-control" id="editYearConducted" required min="1900" max="2099" placeholder="e.g., 2025">
    <div class="invalid-feedback">Please enter a valid year.</div>
  </div>
</div>

<div class="col-md-6">
  <label for="editRepresentative" class="form-label">Representative <span class="text-danger">*</span></label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-people"></i></span>
    <input type="text" class="form-control" id="editRepresentative" name="editRepresentative" required>
  </div>
</div>

<div class="col-md-6">
  <label for="editDesignation" class="form-label">Designation</label>
  <div class="input-group">
    <span class="input-group-text"><i class="bi bi-people"></i></span>

```

```

<input type="text" class="form-control" id="editDesignation" name="editDesignation">
</div>
</div>

<div class="col-md-6">
<label for="editPhone" class="form-label">Phone</label>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-telephone"></i></span>
<input type="text" class="form-control" id="editPhone" name="editPhone">
</div>
</div>

<div class="col-md-6">
<label for="editRemarks" class="form-label">Remarks</label>
<div class="input-group">
<span class="input-group-text"><i class="bi bi-chat-left-text"></i></span>
<textarea class="form-control" id="editRemarks" rows="2"></textarea>
</div>
</div>
<div class="col-6">
<img src="" alt="" id="image-signature">
<label class="form-check-label" for="image-signature">
    Representative Signature
</label>
</div>
</div>
</div>
</div>
<div class="modal-footer">
<button type="button" class="btn btn-3d btn-outline-secondary" data-bs-dismiss="modal">Cancel</button>
<button type="submit" class="btn btn-3d btn-warning">
    <i class="bi bi-save me-1"></i> Update
</button>
</div>
</form>
</div>
</div>
</div>

<!-- Geotagged Photo Modal -->
<div class="modal fade" id="imageUploadModal" tabindex="-1" aria-labelledby="imageUploadModalLabel" aria-hidden="true">
<div class="modal-dialog modal-lg modal-dialog-centered">
<div class="modal-content">

<!-- Header -->
<div class="modal-header bg-primary">
<h5 class="modal-title text-white" id="imageUploadModalLabel">
    <i class="bi bi-image-fill me-2"></i> Upload and Preview Image
</h5>
<button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
</div>

```

```

<!-- Body -->
<div class="modal-body text-center">
  <!-- Dropzone with Preview -->
  <label id="dropZone" class="d-inline-block position-relative border rounded p-2" style="cursor: pointer;">
    
    <input type="file" id="imageInput" accept="image/*" hidden>
  </label>

  <div class="small text-muted">Click or drag an image here to upload</div>

  <input type="hidden" id="currentGeotaggedUrl">
  <input type="hidden" id="currentPermitNo" name="currentPermitNo" />

  <!-- Action Buttons -->
  <div class="d-flex justify-content-center gap-2 mt-3 flex-wrap">
    <a id="viewImageLink" href="#" target="_blank" class="btn btn-3d btn-3d btn-outline-primary d-none">
      <i class="bi bi-eye me-1"></i> View Full Image
    </a>

    <button id="uploadGeotaggedBtn" class="btn btn-3d btn-3d btn-success d-none">
      <i class="bi bi-cloud-arrow-up me-1"></i> Upload Geotagged Photo
    </button>
  </div>
</div>
</div>

<!-- PDF Viewer Modal -->
<div class="modal fade" id="pdfModal" tabindex="-1" aria-labelledby="pdfModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-xl modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header bg-danger">
        <h5 class="modal-title text-white" id="pdfModalLabel">Site Verification Report</h5>
        <button type="button" class="btn-close btn-3d" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <iframe id="pdfViewer" src="" width="100%" height="600px" style="border: none;"></iframe>
      </div>
    </div>
  </div>
</div>

</div>
</div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>
<script type="module" src="js/wrus/wrus-init.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/browser-image-compression@latest/dist/browser-image-compression.js"></script>

```

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>
</body>
</html>
```

water-inventory.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>WRUS Portal</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
  <link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
  <link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="css/style.css" />
  <link rel="stylesheet" href="css/inventory-form.css" />
</head>
<body>
  <div class="d-flex flex-column flex-md-row" id="wrapper">
    <div id="sidebarContainer"></div>
    <div class="container-fluid p-2" id="page-content">
      <h2 class="file-tab-heading">Mobile Inventory Form</h2>

      <div class="container mt-4">
        <!-- White box with padding -->
        <div class="p-4 bg-white rounded shadow-sm">

          <div class="info-box">
            <div class="row g-3">
              <!-- Select City -->
              <div class="col-12">
                <label for="citySelect" class="form-label">City</label>
                <select class="form-select" id="citySelect">
                  <option value="" disabled selected>Select a city</option>
                </select>
              </div>

              <!-- Barangay -->
              <div class="col-12">
                <label for="barangayInput" class="form-label">Barangay</label>
                <input type="text" class="form-control" id="barangayInput" placeholder="Enter barangay">
              </div>

              <!-- Year Conducted -->
              <div class="col-6">
                <label for="yearConducted" class="form-label">Year Inspected</label>
                <input type="number" class="form-control" id="yearConducted" />
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
```

```

<!-- Month Conducted -->
<div class="col-6">
  <label for="monthConducted" class="form-label">Month Inspected</label>
  <select class="form-select" id="monthConducted">
    <!-- Options will be populated by JS -->
  </select>
</div>
</div>
</div>

<!-- File buttons container -->
<div class="item-container bg-secondary" id="itemContainer">
  <div>
    <button class="btn btn-primary big-plus-btn" data-bs-toggle="modal" data-bs-target="#addModal">
      <i class="bi bi-file-earmark-plus"></i>
    </button>
  </div>
</div>

<div class="text-center mt-3">
  <button id="clearBtn" class="btn btn-3d btn-danger w-50 mb-4">
    Clear All
  </button>
  <button id="finalizeButton" class="btn btn-3d btn-success w-50">
    Finalize Entry
  </button>
</div>
</div>
</div>

<!-- Add Modal -->
<div class="modal fade" id="addModal" tabindex="-1" aria-labelledby="addModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered modal-lg">
    <div class="modal-content">
      <form id="modalForm">
        <!-- Modal Header -->
        <div class="modal-header bg-primary text-white">
          <h5 class="modal-title text-white" id="addModalLabel">Add Water Inventory</h5>
          <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>
        </div>

        <!-- Modal Body -->
        <div class="modal-body">
          <div class="row gy-3">

            <!-- Year Conducted -->
            <div class="col-6">
              <label for="modalYearConducted" class="form-label fw-bold small">Year Inspected</label>
              <input type="number" class="form-control form-control-lg" id="modalYearConducted">
            </div>

            <div class="col-6">

```

```
<label for="monthConductedAddModal" class="form-label">Month Inspected</label>
<select class="form-select fw-bold form-select-lg" id="monthConductedAddModal">
  <!-- Options will be populated by JS -->
</select>
</div>

<!-- Owner -->
<div class="col-12">
  <label for="modalOwner" class="form-label fw-bold small">Owner</label>
  <input type="text" class="form-control form-control-lg" id="modalOwner" placeholder="Enter owner name">
</div>

<!-- Location -->
<div class="col-12">
  <label for="modalLocation" class="form-label fw-bold small">Location</label>
  <input type="text" class="form-control form-control-lg" id="modalLocation" placeholder="Enter location">
</div>

<!-- City & Barangay (side by side on larger screens) -->
<div class="col-6">
  <label for="modalCity" class="form-label fw-bold small">City</label>
  <input type="text" class="form-control form-control-lg" id="modalCity">
</div>

<div class="col-6">
  <label for="modalBarangay" class="form-label fw-bold small">Barangay</label>
  <input type="text" class="form-control form-control-lg" id="modalBarangay">
</div>

<!-- Source of Water -->
<div class="col-6">
  <label for="modalSourceWaterSelect" class="form-label fw-bold small">Source of Water</label>
  <!-- Select dropdown -->
  <select class="form-select form-select-lg" id="modalSourceWaterSelect">
    </select>
</div>

<!-- Source Status -->
<div class="col-6">
  <label for="modalSourceStatus" class="form-label fw-bold small">Source Status</label>
  <!-- Select dropdown -->
  <select class="form-select form-select-lg" id="modalSourceStatus">
    </select>
</div>

<!-- Latitude & Longitude -->
<div class="col-6">
  <label for="modalLatitude" class="form-label fw-bold small">Latitude</label>
  <input type="text" class="form-control form-control-lg" id="modalLatitude" placeholder="Enter latitude">
</div>

<div class="col-6">
  <label for="modalLongitude" class="form-label fw-bold small">Longitude</label>
```

```

<input type="text" class="form-control form-control-lg" id="modalLongitude" placeholder="Enter longitude">
</div>

<div class="col-12">
    <label for="modalPurpose" class="form-label fw-bold small">Purpose</label>

    <div class="input-group">
        <!-- Select dropdown -->
        <select class="form-select form-select-lg" id="modalPurposeSelect">
            </select>
        </div>
    </div>

    <!-- Remarks -->
    <div class="col-12">
        <label for="modalRemarks" class="form-label fw-bold small">Remarks</label>
        <textarea class="form-control form-control-lg" id="modalRemarks" rows="2" placeholder="Enter remarks"></textarea>
    </div>

    <!-- Representative & Signature -->
    <div class="col-12">
        <label for="modalRepresentative" class="form-label fw-bold small">Representative</label>
        <input type="text" class="form-control form-control-lg" id="modalRepresentative" placeholder="Enter representative">
    </div>

    <div class="col-6">
        <label for="modalDesignation" class="form-label fw-bold small">Designation</label>
        <input type="text" class="form-control form-control-lg" id="modalDesignation" placeholder="Enter designation">
    </div>

    <div class="col-6">
        <label for="modalPhone" class="form-label fw-bold small">Tel / Phone</label>
        <input type="text" class="form-control form-control-lg" id="modalPhone" placeholder="Enter tel / phone number">
    </div>

    <div class="col-12 d-flex flex-column align-items-center">
        <label for="modalSignature" class="form-label fw-bold small">Signature</label>

        <div class="border rounded bg-light" style="width: 350px; height: 100px;">
            <canvas id="signatureCanvas" style="width: 100%; height: 100%;"></canvas>
        </div>

        <button type="button" id="clearSignature" class="btn btn-sm btn-outline-secondary mt-2">
            Clear Signature
        </button>

        <!-- Hidden input to store signature image -->
        <input type="hidden" id="modalSignature" name="modalSignature">
    </div>
</div>

```

```

</div>

<!-- Modal Footer -->
<div class="modal-footer">
  <button type="submit" class="btn btn-3d btn-primary btn-lg w-100 py-2"> Save</button>
</div>
</form>
</div>
</div>
</div>

<!-- Edit Modal -->
<div class="modal fade" id="editModal" tabindex="-1" aria-labelledby="editModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered modal-lg">
    <div class="modal-content">
      <form id="editModalForm">
        <!-- Modal Header -->
        <div class="modal-header bg-warning text-dark">
          <h5 class="modal-title" id="editModalLabel">Edit Water Inventory</h5>
          <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
        </div>

        <!-- Modal Body -->
        <div class="modal-body">
          <div class="row gy-3">
            <input type="hidden" id="editIndex">
            <!-- Year Conducted -->
            <div class="col-6">
              <label for="editYearConducted" class="form-label fw-bold small">Year Inspected</label>
              <input type="number" class="form-control form-control-lg" id="editYearConducted">
            </div>

            <div class="col-6">
              <label for="monthConductedEditModal" class="form-label">Month Inspected</label>
              <select class="form-select fw-bold form-select-lg" id="monthConductedEditModal">
                <!-- Options will be populated by JS -->
              </select>
            </div>
          </div>
        </div>

        <!-- Owner -->
        <div class="col-12">
          <label for="editOwner" class="form-label fw-bold small">Owner</label>
          <input type="text" class="form-control form-control-lg" id="editOwner" placeholder="Enter owner name">
        </div>

        <!-- Location -->
        <div class="col-12">
          <label for="editLocation" class="form-label fw-bold small">Location</label>
          <input type="text" class="form-control form-control-lg" id="editLocation" placeholder="Enter location">
        </div>

        <!-- City & Barangay -->
        <div class="col-6">

```

```
<label for="editCity" class="form-label fw-bold small">City</label>
<input type="text" class="form-control form-control-lg" id="editCity">
</div>

<div class="col-6">
  <label for="editBarangay" class="form-label fw-bold small">Barangay</label>
  <input type="text" class="form-control form-control-lg" id="editBarangay">
</div>

<!-- Source of Water -->
<div class="col-6">
  <label for="editSourceWaterSelect" class="form-label fw-bold small">Source of Water</label>
  <select class="form-select form-select-lg" id="editSourceWaterSelect">
    </select>
</div>

<!-- Source Status -->
<div class="col-6">
  <label for="editSourceStatus" class="form-label fw-bold small">Source Status</label>
  <!-- Select dropdown -->
  <select class="form-select form-select-lg" id="editSourceStatus">
    </select>
</div>

<!-- Latitude & Longitude -->
<div class="col-12">
  <label for="editLatitude" class="form-label fw-bold small">Latitude</label>
  <input type="text" class="form-control form-control-lg" id="editLatitude" placeholder="Enter latitude">
</div>

<div class="col-6">
  <label for="editLongitude" class="form-label fw-bold small">Longitude</label>
  <input type="text" class="form-control form-control-lg" id="editLongitude" placeholder="Enter longitude">
</div>

<!-- Purpose -->
<div class="col-12">
  <label for="editPurposeSelect" class="form-label fw-bold small">Purpose</label>
  <div class="input-group">
    <select class="form-select form-select-lg" id="editPurposeSelect">
      </select>
  </div>
</div>

<!-- Remarks -->
<div class="col-12">
  <label for="editRemarks" class="form-label fw-bold small">Remarks</label>
  <textarea class="form-control form-control-lg" id="editRemarks" rows="2" placeholder="Enter remarks"></textarea>
</div>

<!-- Representative & Signature -->
<div class="col-12">
  <label for="editRepresentative" class="form-label fw-bold small">Representative</label>
```

```

<input type="text" class="form-control form-control-lg" id="editRepresentative" placeholder="Enter representative">
</div>

<div class="col-6">
  <label for="editDesignation" class="form-label fw-bold small">Designation</label>
  <input type="text" class="form-control form-control-lg" id="editDesignation" placeholder="Enter designation">
</div>

<div class="col-6">
  <label for="editPhone" class="form-label fw-bold small">Tel / Phone</label>
  <input type="text" class="form-control form-control-lg" id="editPhone" placeholder="Enter tel / phone number">
</div>

<div class="col-12 d-flex flex-column align-items-center">
  <label class="form-label fw-bold small">Signature</label>

  <div class="border rounded bg-light" style="width: 350px; height: 100px;">
    <img src="" alt="no signature" id="image-signature" style="width: 100%; height: 100%; object-fit: contain;" />
  </div>

  <small class="text-muted mt-2">Signature can't be edited</small>
</div>
</div>

<!-- Modal Footer -->
<div class="modal-footer">
  <button type="submit" class="btn btn-3d btn-warning btn-lg w-100 py-2"> Update</button>
</div>
</form>
</div>
</div>
</div>

</div>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/signature_pad@4.1.5/dist/signature_pad.umd.min.js"></script>
<script type="module" src="js/water-inventory/water-inventory-init.js"></script>
</body>
</html>

```

map.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>WRUS Portal</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />

```

```

<link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png">
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"/>
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.2.0/dist/leaflet.css" />
<link rel="stylesheet" href="js/lib/leaflet-routing-machine-3.2.12/dist/leaflet-routing-machine.css" />
<link rel="stylesheet" href="css/style.css" />
<link rel="stylesheet" href="css/map.css">
<link rel="stylesheet" href="css/map-page.css">
<link rel="stylesheet" href="css/component/bubble.css">
</head>
<body>
<div class="d-flex flex-column flex-md-row" id="wrapper">
<div id="sidebarContainer"></div>
<div class="container-fluid p-2" id="page-content">
<h2 class="file-tab-heading mb-2"> Map Routes</h2>
<div id="routeIndicator" class="route-indicator text-center my-3">
<span id="routeHeading">No destination selected yet</span>
</div>
<div class="map-container">
<!-- Map -->
<div id="routingMap"></div>
</div>
<div class="form-button-wrapper">
<!-- Floating Form -->
<button id="toggleRoutingFormBtn" class="btn btn-outline-primary btn-sm mb-2">
<i class="bi bi-caret-down-fill"></i> Show Routing Form
</button>
<!-- Collapsible Form -->
<div id="routingFormContainer" style="display:none;">
<form id="routingForm" class="routing-form">
<input type="hidden" id="endLat">
<input type="hidden" id="endLng">
<div class="mb-2">
<label for="startLocation" class="form-label small mb-1">Starting Location</label>
<input type="text" class="form-control form-control-sm" id="startLocation" placeholder="Enter start point" disabled>
</div>
<div class="mb-2 position-relative">
<label for="endLocation" class="form-label small mb-1">End Location <small><i>(i.e., Permit No.)</i></small></label>
<input type="text" class="form-control form-control-sm" id="endLocation" placeholder="Enter destination">
<ul id="endLocationList" class="list-group position-absolute w-100" style="z-index: 1000; max-height: 200px; overflow-y: auto; display:none;"></ul>
</div>
<button type="submit" class="btn btn-primary btn-sm w-100">
<i class="bi bi-arrow-right-circle"></i> Get Route
</button>
</form>
</div>
</div>

```

```

</div>

<button id="stopTrackingBtn" class="btn btn-danger mt-2">Stop Tracking</button>

</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
<script src="https://unpkg.com/leaflet@1.2.0/dist/leaflet.js"></script>
<script src="js/lib/leaflet-routing-machine-3.2.12/dist/leaflet-routing-machine.min.js"></script>
<script src="https://unpkg.com/leaflet.marker.slider@0.3.0/Leaflet.Marker.SlideTo.js"></script>
<script type="module" src="./js/map/routing-page-init.js"></script>
</body>
</html>

```

excel.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>WRUS Portal</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet" />
  <link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@400;500;600&display=swap" rel="stylesheet" />
  <link rel="stylesheet" href="css/style.css" />
  <link rel="stylesheet" href="css/excel.css" />
  <link rel="icon" type="image/png" href="images/favicon-32x32.png" />
</head>
<body>
  <div class="d-flex flex-column flex-md-row" id="wrapper">
    <div id="sidebarContainer"></div>
    <div class="container-fluid p-2" id="page-content">
      <div class="mb-2">
        <h2 class="file-tab-heading">MEMIS to Database</h2>
      </div>
      <!-- Non-Permittees Form-->
      <div class="container my-4">
        <div class="card shadow-sm p-4 border-danger">
          <h5 class="mb-3 text-danger">Import Excel Non-Permittees</h5>
          <div class="row g-3 align-items-center">
            <!-- File input -->
            <div class="col-md-5">
              <label for="excelFileNonPerm" class="form-label fw-semibold">Select Excel File</label>
              <input type="file" class="form-control" id="excelFileNonPerm" accept=".xlsx, .xls" required>
            </div>
            <!-- Year input -->
          </div>
        </div>
      </div>
    </div>
  </div>
</body>

```

```

<div class="col-md-3">
  <label for="yearConductedNonPerm" class="form-label fw-semibold">Year Inspected</label>
  <input type="number" class="form-control" id="yearConductedNonPerm" placeholder="Enter Year" min="1900"
max="2100" required>
</div>

<!-- Import button -->
<div class="col-md-2 d-flex align-items-end">
  <button id="importBtnNonPerm" class="btn btn-danger w-100">
    <i class="bi bi-file-earmark-arrow-up me-1"></i> Import
  </button>
</div>
</div>
</div>
</div>

<!-- Permittees Form-->
<div class="container my-4">
  <div class="card shadow-sm p-4">
    <h5 class="mb-3">Import Excel Permittees</h5>
    <div class="row g-3 align-items-center">
      <!-- File input -->
      <div class="col-md-5">
        <label for="excelFile" class="form-label fw-semibold">Select Excel File</label>
        <input type="file" class="form-control" id="excelFile" accept=".xlsx, .xls" required>
      </div>
      <!-- Year input -->
      <div class="col-md-3">
        <label for="yearConducted" class="form-label fw-semibold">Year Inspected</label>
        <input type="number" class="form-control" id="yearConducted" placeholder="Enter Year" min="1900" max="2100"
required>
      </div>
      <!-- Import button -->
      <div class="col-md-2 d-flex align-items-end">
        <button id="importBtn" class="btn btn-primary w-100">
          <i class="bi bi-file-earmark-excel me-1"></i> Import
        </button>
      </div>
    </div>
  </div>
</div>

<!-- Import Progress Log -->
<div id="importLogContainer" class="mt-4 d-none">
  <div class="card">
    <div class="card-header bg-primary text-white d-flex align-items-center justify-content-between">
      <span>Import Progress</span>
      <span class="small text-warning fw-bold" id="importWarning">
        ! Do not close or refresh this page
      </span>
    </div>
  </div>
</div>

```

```
</div>
<div class="card-body">

    <!-- Loader -->
    <div id="importLoader" class="d-none text-center mb-3">
        <div class="import-progress-circle"></div>
        <div class="text-secondary small mt-2">Processing... Please wait</div>
    </div>

    <!-- Log output -->
    <div id="importLog" class="bg-light border p-2"
        style="height: 200px; overflow-y: auto; font-family: monospace;">
        <!-- Logs will appear here -->
    </div>
    </div>
    </div>
    </div>
</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>
<script type="module" src="./js/excel/excel-init.js"></script>
<!-- <script type="module" src="firebase-functions.js"></script> -->
</body>
</html>
```

XV.B CSS

style.css

```
html, body {
    height: 100%;
    overflow: hidden;
    overflow-y: auto;
}

body {
    font-family: 'Quicksand', 'Segoe UI', 'Helvetica Neue', sans-serif;
}

#wrapper {
    display: flex;
    height: 100vh;
}

#sidebar {
```

```
flex-direction: column;
overflow: hidden;
height: 100vh;
}

#page-content {
position: relative;
z-index: auto;
flex-grow: 1;
background: linear-gradient(to bottom right, #e0f7fa, #ffffff);
display: flex;
flex-direction: column;
justify-content: flex-start;
overflow-y: auto;
}

#page-content h1,
#page-content h2,
#page-content h3,
#page-content h4,
#page-content h5,
#page-content h6 {
color: #023e8a; /* Darker ocean blue for contrast */
text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.2);
font-weight: 700;
}

#page-content h1 {
color: #0077b6;
font-size: 2.5rem;
border-bottom: 2px solid #0096c7;
padding-bottom: 1px;
}

#page-content h2 {
color: #0096c7;
font-size: 2rem;
padding-bottom: 1px;
}

#page-content h3 {
color: #00b4d8;
font-size: 1.75rem;
}

#page-content h4,
#page-content h5,
#page-content h6 {
color: #023e8a;
text-shadow: 1px 1px 2px rgba(72, 202, 228, 0.4);
font-weight: 600;
}
```

```
.file-tab-heading {  
    font-size:34px; font-weight:500; text-transform:uppercase;  
}  
  
.nav-link.text-white:hover,  
.dropdown-item:hover {  
    background-color: rgba(255, 255, 255, 0.2) !important;  
    color: #fff !important;  
}  
  
.dropdown-menu .dropdown-item:hover {  
    background-color: #0d6efd !important;  
    color: #fff !important;  
}  
  
nav.navbar {  
    min-height: 100vh;  
}  
  
.water-btn {  
    background: white;  
    color: #0288d1;  
    border: none;  
    border-radius: 50px;  
    padding: 8px 16px;  
    transition: background 0.3s, transform 0.2s;  
    box-shadow: 0 4px 10px rgba(0, 123, 255, 0.3);  
}  
  
.water-btn:hover {  
    background: #e1f5fe;  
    transform: scale(1.05);  
}  
  
.water-btn-outline {  
    background: transparent;  
    color: white;  
    border: 2px solid white;  
    border-radius: 50px;  
    padding: 8px 16px;  
    transition: background 0.3s, color 0.3s, transform 0.2s;  
}  
  
.water-btn-outline:hover {  
    background: white;  
    color: #0288d1;  
    transform: scale(1.05);  
}  
  
.water-btn,  
.water-btn-outline {  
    box-shadow: 0 0 8px rgba(255, 255, 255, 0.6);  
}
```

```
/* Layout Setup */
#wrapper {
  display: flex;
  min-height: 100vh;
}

/* Sidebar */
#sidebar {
  width: 250px;
  background: linear-gradient(180deg, #0077be, #00bcd4);
  color: white;
  display: flex;
  flex-direction: column;
  transition: all 0.3s ease;
  z-index: 1045;
}

.logo-container {
  display: flex;
  align-items: center;
  gap: 0.5rem;
}

.wrus-water-title{
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
}

/* WRUS Portal Title – Large & White */
.text-3d {
  font-weight: 800;
  font-size: 1.75rem; /* Adjust size as needed */
  color: #fff;
}

/* Main Content */
#page-content {
  flex: 1;
  transition: margin-left 0.3s;
}

/* Sidebar Collapsed */
#wrapper.sidebar-collapsed #sidebar {
  width: 60px;
}

#wrapper.sidebar-collapsed #page-content {
  margin-left: 0;
}

/* Hide text and collapse submenus when collapsed */
#wrapper.sidebar-collapsed .offcanvas-body h4,
#wrapper.sidebar-collapsed .offcanvas-body .collapse,
```

```
#wrapper.sidebar-collapsed .offcanvas-body .offcanvas-header,  
#wrapper.sidebar-collapsed .offcanvas-body .nav .nav {  
    display: none !important;  
}  
  
#wrapper.sidebar-collapsed #sidebarToggle i {  
    transform: rotate(180deg);  
}  
  
/* Only scroll if needed in mobile view, and limit it */  
.offcanvas-body {  
    overflow-y: hidden; /* Remove forced scrollbars */  
    height: 100%;  
}  
  
.table-responsive {  
    position: relative;  
    background: linear-gradient(to bottom right, #e0f7fa, #b2ebf2);  
    border-radius: 12px;  
    padding: 5px;  
    box-shadow: 0 12px 24px rgba(0, 188, 212, 0.25), 0 4px 6px rgba(0, 0, 0, 0.1);  
    overflow-x: auto;  
    perspective: 1000px; /* Adds 3D context */  
}  
  
.table-responsive table {  
    background: linear-gradient(145deg, #ffffff, #e0f7fa);  
    border-collapse: separate;  
    border-spacing: 0;  
    width: 100%;  
    border-radius: 12px;  
    overflow: hidden;  
    box-shadow: 0 6px 10px rgba(0, 188, 212, 0.3),  
        inset 0 2px 3px rgba(255, 255, 255, 0.6),  
        inset 0 -2px 3px rgba(0, 0, 0, 0.05);  
}  
  
.table-responsive::before,  
.table-responsive::after {  
    content: none; /* Ripple sides removed */  
}  
  
.table-responsive table {  
    background-color: white;  
    border-collapse: separate;  
    border-spacing: 0;  
    width: 100%;  
    border-radius: 12px;  
    overflow: hidden;  
    box-shadow: 0 2px 8px rgba(0, 188, 212, 0.08);  
    position: relative;  
    z-index: 1;  
}
```

```
.table-responsive thead th {  
    position: relative;  
    background: linear-gradient(to right, #4dd0e1, #00bcd4);  
    color: white;  
    font-weight: 600;  
    text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);  
    box-shadow:  
        inset 0 1px 0 rgba(255, 255, 255, 0.1),  
        inset 0 -1px 0 rgba(0, 0, 0, 0.05),  
        0 2px 4px rgba(0, 0, 0, 0.1);  
    z-index: 1;  
    overflow: hidden;  
}  
  
.table-responsive thead th::after {  
    content: " ";  
    position: absolute;  
    top: 0;  
    left: -50%;  
    width: 30%;  
    height: 100%;  
    background: linear-gradient(  
        120deg,  
        rgba(255, 255, 255, 0.05) 0%,  
        rgba(255, 255, 255, 0.15) 50%,  
        rgba(255, 255, 255, 0.05) 100%  
    );  
    transform: skewX(-20deg);  
    animation: shine 10s ease-in-out infinite;  
    pointer-events: none;  
    z-index: 0;  
}  
  
.table-responsive th {  
    padding: 14px;  
    border: none;  
    text-align: left;  
    font-weight: 600;  
}  
  
.table-responsive td {  
    padding: 12px;  
    border-top: 1px solid #b2ebf2;  
    background-color: rgba(255, 255, 255, 0.9);  
}  
  
.table-responsive tbody tr:hover {  
    background-color: rgba(178, 235, 242, 0.6);  
    transition: background-color 0.3s ease, box-shadow 0.3s ease;  
    box-shadow: inset 0 0 8px rgba(0, 188, 212, 0.2);  
}
```

```
/* Optional glow row */
.table-responsive tbody tr.glow-row {
  background-color: #b2ebf2;
  box-shadow: 0 0 12px rgba(0, 188, 212, 0.5);
}

/* Round corners: only top of thead and bottom of last row */
.table-responsive thead th:first-child {
  border-top-left-radius: 12px;
}

.table-responsive thead th:last-child {
  border-top-right-radius: 12px;
}

.table-responsive tbody tr:last-child td:first-child {
  border-bottom-left-radius: 12px;
}

.table-responsive tbody tr:last-child td:last-child {
  border-bottom-right-radius: 12px;
}

/* Loading Overlay */
#loadingOverlay {
  display: none;
  position: fixed;
  z-index: 1055;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0,0,0,0.8);
  justify-content: center;
  align-items: center;
  transition: opacity 0.3s ease;
  perspective: 1000px;
}

.modal-header {
  position: relative;
  background-color: #c2d4e2;
  color: #007c91;
  border-bottom: 1px solid #00bcd4;
  font-weight: 600;
  text-shadow: 0 1px 1px rgba(0, 0, 0, 0.15);
  box-shadow:
    inset 0 1px 0 rgba(255, 255, 255, 0.15),
    inset 0 -1px 0 rgba(0, 0, 0, 0.05),
    0 2px 4px rgba(0, 0, 0, 0.1);
  overflow: hidden;
  z-index: 1;
}
```

```
.modal-header::after {
  content: "";
  position: absolute;
  top: 0;
  left: -50%;
  width: 30%;
  height: 100%;
  background: linear-gradient(
    120deg,
    rgba(255, 255, 255, 0.05) 0%,
    rgba(255, 255, 255, 0.15) 50%,
    rgba(255, 255, 255, 0.05) 100%
  );
  transform: skewX(-20deg);
  animation: shine 10s ease-in-out infinite;
  pointer-events: none;
  z-index: 0;
}

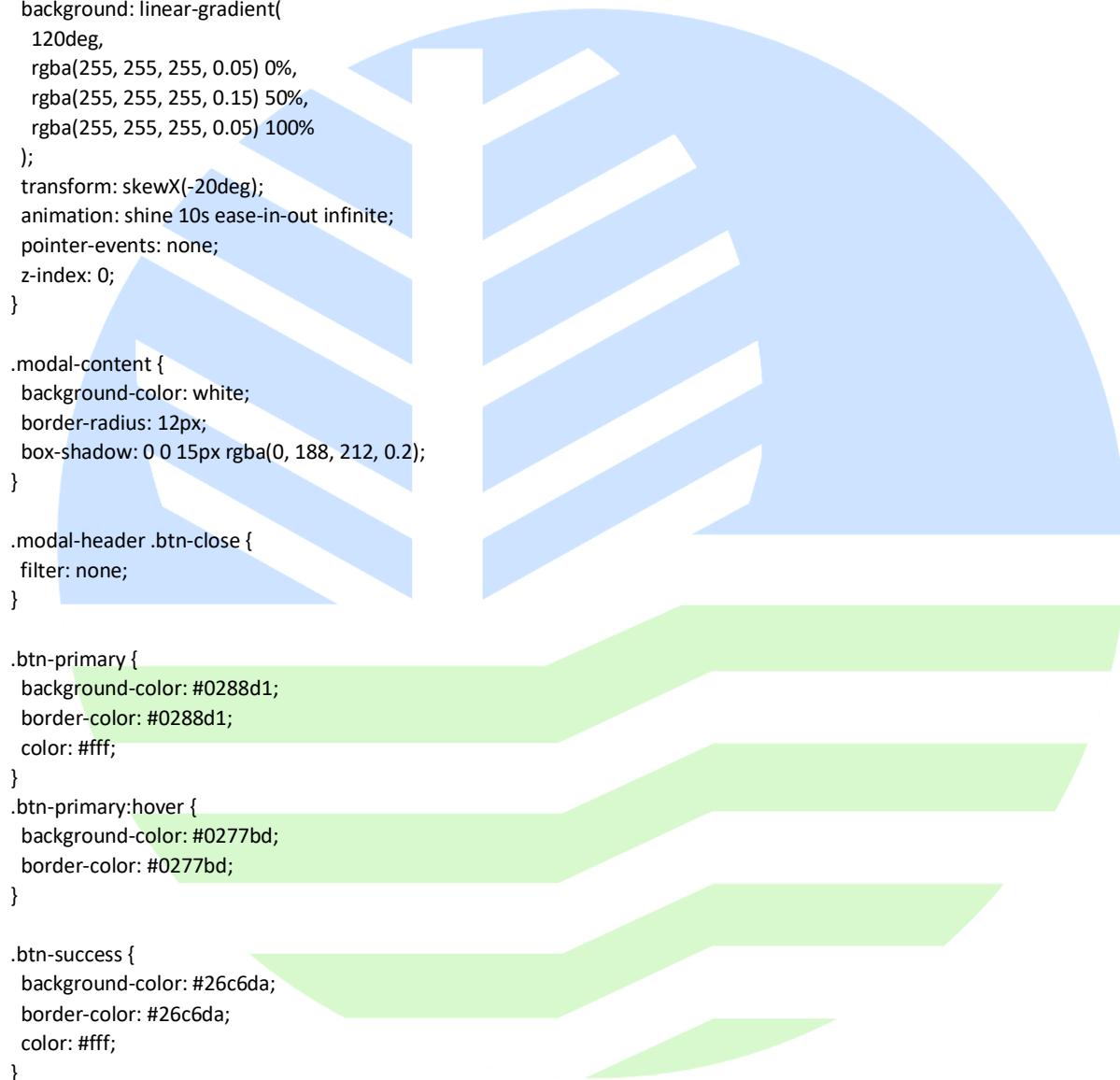
.modal-content {
  background-color: white;
  border-radius: 12px;
  box-shadow: 0 0 15px rgba(0, 188, 212, 0.2);
}

.modal-header .btn-close {
  filter: none;
}

.btn-primary {
  background-color: #0288d1;
  border-color: #0288d1;
  color: #fff;
}
.btn-primary:hover {
  background-color: #0277bd;
  border-color: #0277bd;
}

.btn-success {
  background-color: #26c6da;
  border-color: #26c6da;
  color: #fff;
}
.btn-success:hover {
  background-color: #00acc1;
  border-color: #00acc1;
}

.btn-info {
  background-color: #4fc3f7;
  border-color: #4fc3f7;
```



```
color: #fff;
}

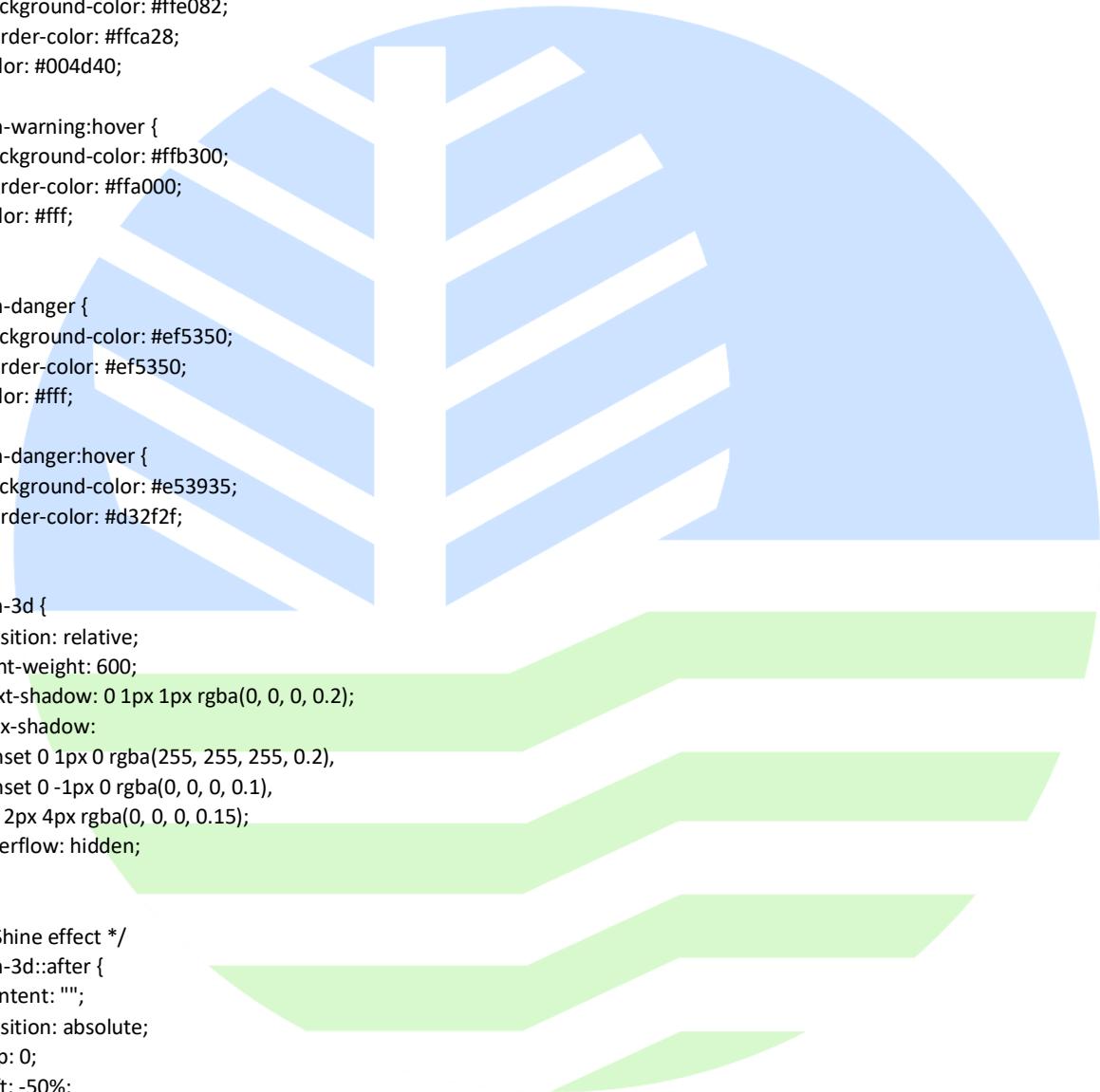
.btn-info:hover {
background-color: #29b6f6;
border-color: #29b6f6;
}

.btn-warning {
background-color: #ffe082;
border-color: #ffca28;
color: #004d40;
}
.btn-warning:hover {
background-color: #ffb300;
border-color: #ffa000;
color: #fff;
}

.btn-danger {
background-color: #ef5350;
border-color: #ef5350;
color: #fff;
}
.btn-danger:hover {
background-color: #e53935;
border-color: #d32f2f;
}

.btn-3d {
position: relative;
font-weight: 600;
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
box-shadow:
inset 0 1px 0 rgba(255, 255, 255, 0.2),
inset 0 -1px 0 rgba(0, 0, 0, 0.1),
0 2px 4px rgba(0, 0, 0, 0.15);
overflow: hidden;
}

/* Shine effect */
.btn-3d::after {
content: "";
position: absolute;
top: 0;
left: -50%;
width: 30%;
height: 100%;
background: linear-gradient(
120deg,
rgba(255, 255, 255, 0.05) 0%,
rgba(255, 255, 255, 0.15) 50%,
rgba(255, 255, 255, 0.05) 100%
);
```



```

        transform: skewX(-20deg);
        animation: shine 10s ease-in-out infinite;
        pointer-events: none;
    }

.water-effect {
    position: relative;
    width: 150px;
    height: 150px;
}

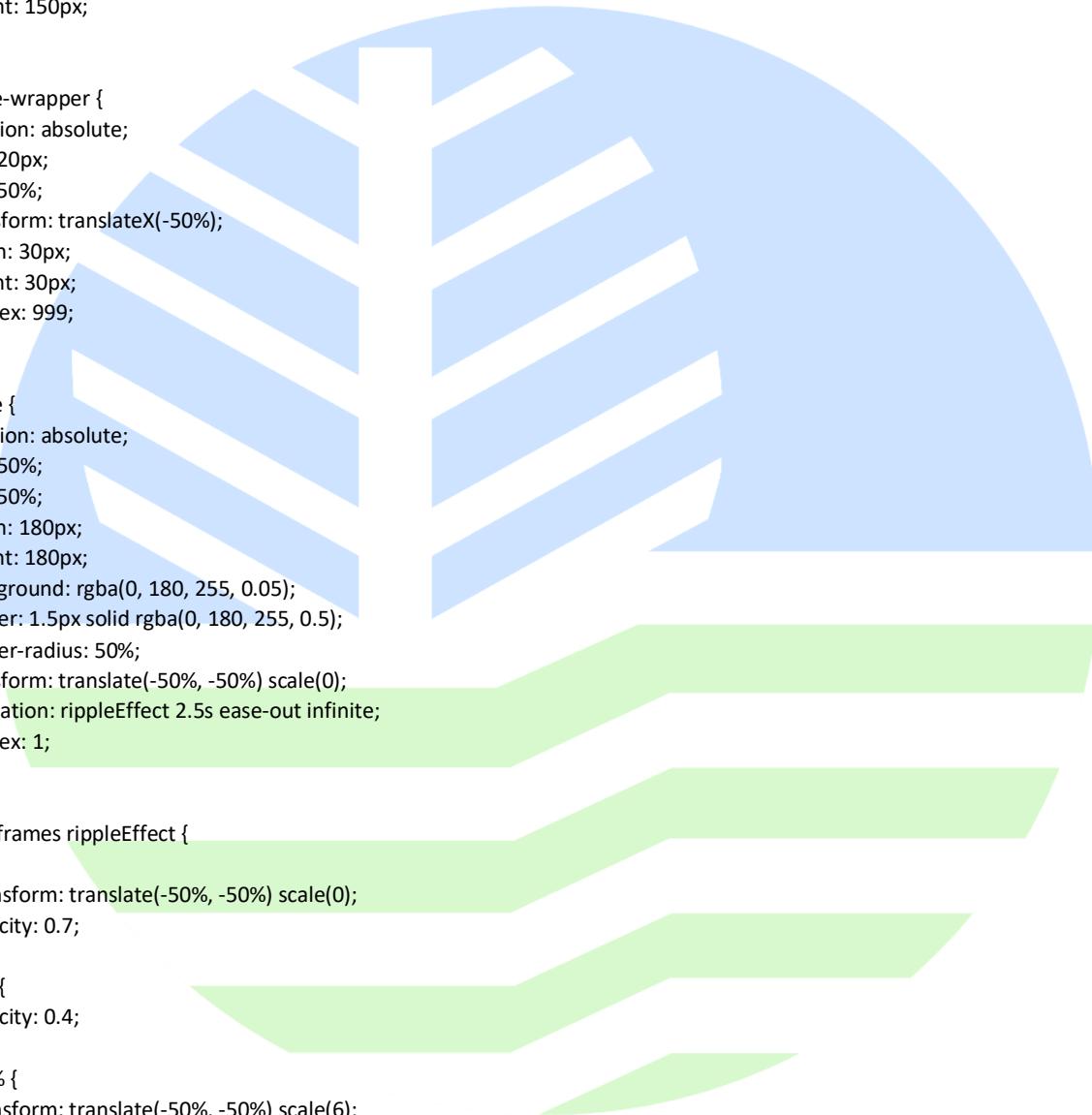
.ripple-wrapper {
    position: absolute;
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    width: 30px;
    height: 30px;
    z-index: 999;
}

.ripple {
    position: absolute;
    top: 50%;
    left: 50%;
    width: 180px;
    height: 180px;
    background: rgba(0, 180, 255, 0.05);
    border: 1.5px solid rgba(0, 180, 255, 0.5);
    border-radius: 50%;
    transform: translate(-50%, -50%) scale(0);
    animation: rippleEffect 2.5s ease-out infinite;
    z-index: 1;
}

@keyframes rippleEffect {
    0% {
        transform: translate(-50%, -50%) scale(0);
        opacity: 0.7;
    }
    60% {
        opacity: 0.4;
    }
    100% {
        transform: translate(-50%, -50%) scale(6);
        opacity: 0;
    }
}

/* Spinning DENR logo */
.denr-spin {
    animation: spinY 1.5s linear infinite;
    transform-origin: center;
}

```



```

transform-style: preserve-3d;
filter: drop-shadow(0 0 6px rgba(0, 150, 200, 0.5)); /* bluish glow */
position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
z-index: 2;
border-radius: 50%;
box-shadow:
  0 0 10px rgba(0, 200, 255, 0.5),
  0 0 20px rgba(0, 200, 255, 0.3) inset;
background: radial-gradient(circle, rgba(0, 200, 255, 0.1), transparent 70%);
backdrop-filter: blur(1px);
}

/* Spin keyframes */
@keyframes spinY {
  0% { transform: translate(-50%, -50%) rotateY(0deg); }
  100% { transform: translate(-50%, -50%) rotateY(360deg); }
}

/* Notification Styling */
.notification {
  position: fixed;
  bottom: 40px;
  left: 50%;
  transform: translateX(-50%) translateY(100%);
  padding: 15px 30px;
  background: #333;
  color: white;
  border-radius: 8px;
  opacity: 0;
  transition: transform 0.5s ease, opacity 0.5s ease;
  z-index: 9999;
}

.notification.show {
  transform: translateX(-50%) translateY(0);
  opacity: 1;
}

.notification.success { background: #00bcd4; }
.notification.error { background: #dc3545; }
.notification.info { background: #007bff; }
.notification.warning { background: #ffc107; color: #222; }

/* 🌐 Confirmation Box */
.custom-confirm-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
}

```

```
background-filter: blur(2px);
display: flex;
align-items: center;
justify-content: center;
z-index: 9999;
}

.custom-confirm-box {
background: white;
padding: 20px;
border-radius: 10px;
min-width: 300px;
box-shadow: 0 0 20px rgba(0,0,0,0.5);
text-align: center;
}

.custom-confirm-message {
margin-bottom: 20px;
font-weight: 500;
}

.custom-confirm-buttons button {
margin: 0 5px;
}

/* Pagination */
.pagination .page-link {
color: #0b7fab; /* Water blue */
border-color: #cde9f6;
background-color: #f4fafd; /* Very light blue */
transition: background-color 0.3s, color 0.3s;
}

.pagination .page-link:hover {
color: #ffffff;
background-color: #0b7fab;
border-color: #0b7fab;
}

.pagination .page-item.active .page-link {
background-color: #0b7fab;
border-color: #0b7fab;
color: #ffffff;
}

@media (max-width: 767.98px) {
nav.navbar {
min-height: auto;
width: 100%;
}
}

#sidebar {
```

```
height: 100vh !important;
display: flex;
flex-direction: column;
}

@media (max-width: 767.98px) {
  #sidebar .offcanvas-body {
    max-height: calc(100vh - 56px); /* adjust based on navbar */
  }
}

@keyframes shine {
  0% {
    left: -50%;
    opacity: 0;
  }
  5% {
    opacity: 0.4;
  }
  15% {
    left: 125%;
    opacity: 0.4;
  }
  20% {
    opacity: 0;
  }
  100% {
    left: 125%;
    opacity: 0;
  }
}
```

admin-dashboard.css

```
.scorecard-grid-container {
  display: flex;
  flex-direction: column;
  gap: 30px;
}

.scorecard-row {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
}

.user-column {
  background: #ff0fdfc;
  padding: 12px;
  border-radius: 10px;
  box-shadow: 0 1px 4px rgba(0, 191, 255, 0.1);
}
```



```
.user-name {
  font-weight: bold;
  color: #00796b;
  margin-bottom: 8px;
  text-align: center;
  font-size: 0.95rem;
  border-bottom: 1px solid #b2dfdb;
  padding-bottom: 4px;
}

.scorecard-column {
  min-width: 220px;
  background: #e0f7fa;
  border-radius: 12px;
  border: 1px solid #81d4fa;
  box-shadow: 0 2px 8px rgba(0, 191, 255, 0.2);
  padding: 10px;
  display: flex;
  flex-direction: column;
}

.scorecard-column .user-header {
  font-weight: bold;
  text-align: center;
  padding: 6px;
  background-color: #00bcd4;
  color: white;
  border-radius: 6px;
  margin-bottom: 10px;
}

.scorecard-entry {
  background: #ffffff;
  border: 1px solid #b2ebf2;
  border-radius: 8px;
  padding: 10px;
  margin-bottom: 14px;
  font-size: 0.85rem;
  color: #004d40;
  box-shadow: 0 1px 4px rgba(0, 191, 255, 0.1);
  position: relative;
}

.scorecard-entry::before {
  content: "";
  position: absolute;
  top: -10px;
  left: 0;
  width: 100%;
  height: 1px;
  background: rgba(0, 0, 0, 0.1);
}
```

```
.scorecard-entry:first-of-type::before {  
    content: none;  
}  
 
```

```
.scorecard-entry i {  
    margin-right: 5px;  
}  
 
```

```
.no-record {  
    padding: 10px;  
    text-align: center;  
    color: #607d8b;  
    font-style: italic;  
    background-color: #e0f2f1;  
    border-radius: 10px;  
    margin-top: 10px;  
}  
 
```

user-management.css

```
.scorecard-grid-container {  
    display: flex;  
    flex-direction: column;  
    gap: 30px;  
}  
 
```

```
.scorecard-row {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    gap: 20px;  
}  
 
```

```
.user-column {  
    background: #f0fdfc;  
    padding: 12px;  
    border-radius: 10px;  
    box-shadow: 0 1px 4px rgba(0, 191, 255, 0.1);  
}  
 
```

```
.user-name {  
    font-weight: bold;  
    color: #00796b;  
    margin-bottom: 8px;  
    text-align: center;  
    font-size: 0.95rem;  
    border-bottom: 1px solid #b2dfdb;  
    padding-bottom: 4px;  
}  
 
```

```
.scorecard-column {  
    min-width: 220px;  
    background: #e0f7fa;  
    border-radius: 12px;  
}
```

```
border: 1px solid #81d4fa;
box-shadow: 0 2px 8px rgba(0, 191, 255, 0.2);
padding: 10px;
display: flex;
flex-direction: column;
}
```

```
.scorecard-column .user-header {
font-weight: bold;
text-align: center;
padding: 6px;
background-color: #00bcd4;
color: white;
border-radius: 6px;
margin-bottom: 10px;
}
```

```
.scorecard-entry {
background: #ffffff;
border: 1px solid #b2ebf2;
border-radius: 8px;
padding: 10px;
margin-bottom: 14px;
font-size: 0.85rem;
color: #004d40;
box-shadow: 0 1px 4px rgba(0, 191, 255, 0.1);
position: relative;
}
```

```
.scorecard-entry::before {
content: "";
position: absolute;
top: -10px;
left: 0;
width: 100%;
height: 1px;
background: rgba(0, 0, 0, 0.1);
}
```

```
.scorecard-entry:first-of-type::before {
content: none;
}
```

```
.scorecard-entry i {
margin-right: 5px;
}
```

```
.no-record {
padding: 10px;
text-align: center;
color: #607d8b;
font-style: italic;
background-color: #e0f2f1;
```

```
border-radius: 10px;  
margin-top: 10px;  
}
```

bubble.css

```
.bubble {  
  position: fixed;  
  background: radial-gradient(circle, rgba(173, 216, 230, 0.6), rgba(0, 119, 190, 0.2));  
  border-radius: 50%;  
  pointer-events: none;  
  opacity: 0;  
  transform: scale(0.5);  
  animation: bubbleDrift 3s ease-out forwards;  
  z-index: 9999;  
}  
  
.bubble.active {  
  opacity: 1;  
}  
  
@keyframes bubbleDrift {  
  0% {  
    transform: translate(0, 0) scale(0.5);  
    opacity: 0.6;  
  }  
  100% {  
    transform: translate(var(--drift-x), var(--drift-y)) scale(1.3);  
    opacity: 0;  
  }  
}
```

rain.css

```
/* 💧 Rain Overlay */  
.rain-overlay {  
  position: fixed;  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background: rgba(0, 0, 0, 0.3);  
  z-index: 9997;  
  pointer-events: none;  
  opacity: 1;  
  transition: opacity 0.4s ease;  
}  
  
.rain-overlay.fade-out,  
.rain-container.fade-out {  
  opacity: 0;  
  transition: opacity 0.4s ease;
```

```

}

/* 💧 Raindrop */
.rain-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  overflow: hidden;
  z-index: 9998;
  pointer-events: none;
}

.raindrop {
  position: absolute;
  top: -20px;
  width: 2px;
  background: rgba(255, 255, 255, 0.7);
  border-radius: 50px;
  animation-name: fall;
  animation-timing-function: linear;
  animation-iteration-count: infinite;
}

/* 💧 Pause Rain */
.rain-container.paused .raindrop {
  animation-play-state: paused;
}

@keyframes fall {
  0% {
    transform: translateY(0) scaleX(1);
    opacity: 1;
  }
  90% {
    opacity: 1;
  }
  100% {
    transform: translateY(100vh) scaleX(0.8);
    opacity: 0;
  }
}

```

consumable.css

```

.water-theme {
  box-shadow: 0 4px 8px rgba(0, 188, 212, 0.2);
  border-radius: 8px;
  overflow: hidden;
}

.water-theme input::placeholder {

```



```
color: #81d4fa;  
}  
  
.water-theme input:focus {  
outline: none;  
box-shadow: 0 0 10px #4dd0e1;  
background: linear-gradient(to right, #ffffff, #b2ebf2);  
}  
  
.input-group-text {  
background-color: #e0f7fa !important;  
color: #0277bd !important;  
}  
  
/* select number of rows */  
  
label[for="pageSizeSelect"] {  
color: #0277bd;  
font-weight: 500;  
}  
  
#pageSizeSelect {  
background-color: #f0f8ff;  
border: 1px solid #64b5f6;  
color: #1565c0;  
font-weight: 500;  
transition: border-color 0.3s ease, box-shadow 0.3s ease;  
}  
  
#pageSizeSelect:focus {  
border-color: #42a5f5;  
box-shadow: 0 0 6px rgba(100, 181, 246, 0.3);  
outline: none;  
}  
  
/* Modal Background – Light Blue Glassy */  
#actionModal .modal-content {  
background: linear-gradient(to bottom right, #e3f2fd, #bbdefb);  
border: 1px solid #90caf9;  
box-shadow: 0 0 18px rgba(100, 181, 246, 0.3);  
border-radius: 12px;  
color: #003c66;  
}  
  
/* Modal Header */  
#actionModal .modal-header {  
border-bottom: 1px solid #90caf9;  
background-color: rgba(179, 229, 252, 0.4);  
}  
  
/* Modal Title Icon and Text */  
#actionModal .modal-title {  
color: #01579b;
```

```
}
```

```
/* Close Button */
```

```
#actionModal .btn-close {
```

```
    filter: invert(40%);
```

```
}
```

```
/* Water-themed Buttons */
```

```
/* Add Stock – Aqua Blue */
```

```
#addStockBtn {
```

```
    background-color: #26c6da;
```

```
    border-color: #26c6da;
```

```
    color: #fff;
```

```
}
```

```
#addStockBtn:hover {
```

```
    background-color: #00acc1;
```

```
    border-color: #00acc1;
```

```
}
```

```
/* Assign Item – Deep Water Blue */
```

```
#assignItemBtn {
```

```
    background-color: #0288d1;
```

```
    border-color: #0288d1;
```

```
    color: #fff;
```

```
}
```

```
#assignItemBtn:hover {
```

```
    background-color: #0277bd;
```

```
    border-color: #0277bd;
```

```
}
```

```
/* View Ledger – Light Sky Blue */
```

```
#viewLedgerBtn {
```

```
    background-color: #4fc3f7;
```

```
    border-color: #4fc3f7;
```

```
    color: #fff;
```

```
}
```

```
#viewLedgerBtn:hover {
```

```
    background-color: #29b6f6;
```

```
    border-color: #29b6f6;
```

```
}
```

dashboard.css

```
.profile-banner {
```

```
    background: #fff;
```

```
    display: flex;
```

```
    flex-direction: column;
```

```
    align-items: center;
```

```
    justify-content: center;
```

```
    border-radius: 12px;
```

```
    box-shadow: 0 4px 12px rgba(0, 123, 255, 0.15);
```

```
box-sizing: border-box;
min-height: 85vh;
}

.logo-container {
  display: inline-block;
  position: relative;
  cursor: pointer;
  padding: 5px;
}

@keyframes spin {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}

.spin-once {
  animation: spin 3s linear;
}

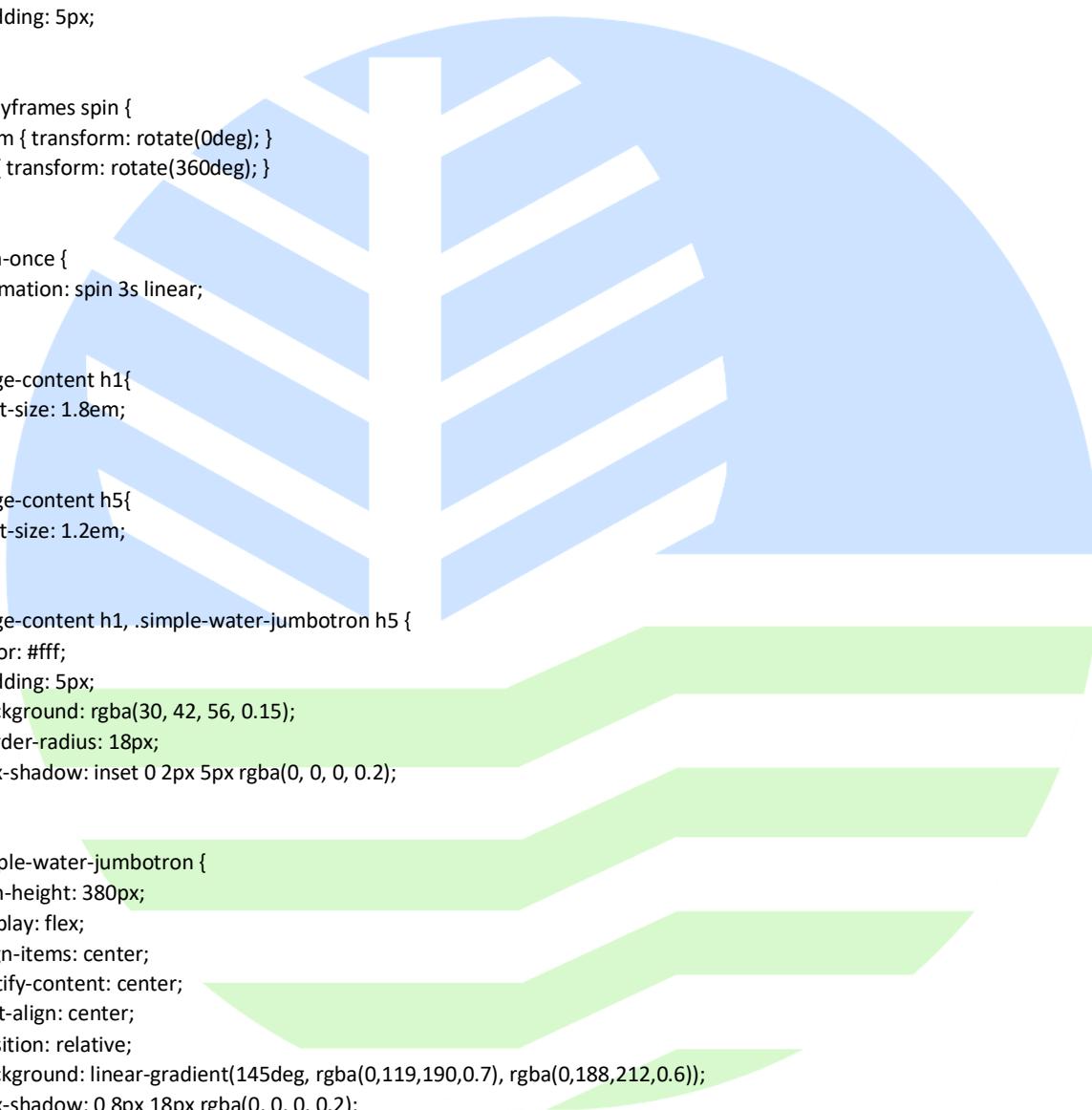
#page-content h1{
  font-size: 1.8em;
}

#page-content h5{
  font-size: 1.2em;
}

#page-content h1, .simple-water-jumbotron h5 {
  color: #fff;
  padding: 5px;
  background: rgba(30, 42, 56, 0.15);
  border-radius: 18px;
  box-shadow: inset 0 2px 5px rgba(0, 0, 0, 0.2);
}

.simple-water-jumbotron {
  min-height: 380px;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  position: relative;
  background: linear-gradient(145deg, rgba(0,119,190,0.7), rgba(0,188,212,0.6));
  box-shadow: 0 8px 18px rgba(0, 0, 0, 0.2);
  border-bottom: 3px solid rgba(255,255,255,0.2);
  overflow: hidden;
  border-radius: 25px;
  padding: 2rem 1rem;
  backdrop-filter: blur(6px);
}

.denr-logo {
```



```
max-width: 120px;
border-radius: 50%;
box-shadow: 0 2px 8px rgba(0, 0, 0, 0.3);
padding: 5px;
background: rgba(255,255,255);
}

.simple-water-jumbotron h1 {
font-weight: 600;
color: #fff;
text-shadow: 2px 2px 8px rgba(0, 0, 0, 0.3);
position: relative;
z-index: 3;
font-size: 2.3rem;
}

.simple-water-jumbotron p {
font-size: 1.2rem;
margin-top: 10px;
text-shadow: 1px 1px 4px rgba(0, 0, 0, 0.25);
position: relative;
z-index: 3;
}

.activities-section {
background: rgba(255, 255, 255, 0.05);
backdrop-filter: blur(12px);
border-radius: 18px;
padding: 1.5rem;
margin-top: 2rem;
box-shadow: 0 6px 15px rgba(0, 0, 0, 0.25);
transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.activities-section .list-group-item {
font-size: 1.1rem;
background: rgba(255, 255, 255, 0.03) !important;
border: 1px solid rgba(255, 255, 255, 0.1);
color: #f8f9fa;
box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.25);
border-radius: 14px;
margin: 5px 0;
}

.badge-3d {
padding: 0.35em 0.7em;
font-size: 0.75rem;
font-weight: 500;
color: white;
border-radius: 0.5rem;
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
box-shadow: 0 1px 3px rgba(0, 0, 0, 0.15);
background: rgba(0,0,0,0.5);
```

```
}

.card {
  position: relative;
  border-radius: 16px;
  color: white;
  padding: 0;
  background: #fff;
  box-shadow:
    0 6px 15px rgba(0, 0, 0, 0.2),
    0 2px 4px rgba(0, 0, 0, 0.1) inset;
  overflow: hidden;
  transform: perspective(1000px) translateZ(0);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.row.text-center > .col-md-6:nth-child(1) .card {
  background: linear-gradient(135deg, #0d6efd, #3d8bfd);
}

.row.text-center > .col-md-6:nth-child(2) .card {
  background: linear-gradient(135deg, #198754, #4caf7d);
}

.row.text-center > .col-md-6:nth-child(3) .card {
  background: linear-gradient(135deg, #0aa0c0, #2eafc7);
}

.row.text-center > .col-md-6:nth-child(4) .card {
  background: linear-gradient(135deg, #dc3545, #e85d68);
}

.card-header-banner {
  position: relative;
  background: rgba(0, 0, 0, 0.25);
  padding: 12px;
  text-align: center;
  font-weight: bold;
  font-size: 1.1rem;
  overflow: hidden;
}

.card-header-banner::before {
  content: "";
  position: absolute;
  top: 0;
  left: -75%;
  width: 50%;
  height: 100%;
  background: linear-gradient(
    120deg,
    rgba(255, 255, 255, 0.2) 0%,
    rgba(255, 255, 255, 0.5) 50%,

```



```
rgba(255, 255, 255, 0.2) 100%
);
transform: skewX(-20deg);
animation: shine 10s infinite;
}

.card-body {
padding: 20px;
}

.card-text {
color: #212529;
padding: 1rem 2rem;
background: #fff;
border-radius: 16px;
box-shadow: inset 0 4px 8px rgba(0, 0, 0, 0.15);
}

.card-body a {
width: 100px;
min-height: 60px;
margin: auto;
display: flex;
align-items: center;
justify-content: center;
text-align: center;
}

/* Base nav-tabs container */
.nav-tabs {
border-bottom: 3px solid #00b4d8;
background: linear-gradient(to right, #e0f7fa, #caf0f8);
border-radius: 0.5rem;
padding: 0.5rem;
overflow-x: auto;
scrollbar-width: none; /* for Firefox */
}
.nav-tabs::-webkit-scrollbar {
display: none; /* for Chrome, Safari */
}

/* Individual tabs */
.nav-tabs .nav-link {
color: #0077b6;
background-color: transparent;
border: none;
border-radius: 0.75rem;
margin: 0 0.25rem;
padding: 0.6rem 1.2rem;
font-weight: 600;
transition: background-color 0.3s ease, color 0.3s ease;
}
```

```
/* Hover effect */
.nav-tabs .nav-link:hover {
  background-color: rgba(0, 180, 216, 0.1);
  color: #023e8a;
}

/* Active tab */
.nav-tabs .nav-link.active {
  background-color: #00b4d8;
  color: #ffffff;
  box-shadow: 0 4px 12px rgba(0, 180, 216, 0.2);
}

/* Water-themed Scorecard (Card Layout, Not Table) */
.scorecard-wrapper {
  margin: 20px auto;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.scorecard {
  background: linear-gradient(135deg, #e0f7fa, #b2ebf2);
  border-radius: 12px;
  overflow: hidden;
  box-shadow: 0 0 10px rgba(0, 191, 255, 0.3);
  border: 1px solid #81d4fa;
  padding: 10px;
}

.scorecard-header {
  background: #00bcd4;
  color: #fff;
  padding: 12px;
  font-weight: 600;
  font-size: 0.95rem;
  letter-spacing: 0.5px;
  border-radius: 10px;
  margin-bottom: 10px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

/* Refresh button inside header */
.scorecard-refresh-btn {
  background-color: transparent;
  border: none;
  color: #ffffff;
  font-size: 1rem;
  padding: 4px 6px;
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.2s ease, transform 0.2s ease;
}
```

```
.scorecard-refresh-btn:hover {  
background-color: rgba(255, 255, 255, 0.15);  
transform: rotate(90deg);  
}  
  
/* New: Card-style items for each month */  
.scorecard-list {  
display: flex;  
flex-direction: column;  
gap: 10px;  
}  
  
.scorecard-item {  
background: #ffffff;  
border: 1px solid #81d4fa;  
border-radius: 10px;  
padding: 10px;  
box-shadow: 0 1px 4px rgba(0, 191, 255, 0.1);  
display: flex;  
justify-content: space-between;  
align-items: center;  
font-size: 0.85rem;  
color: #006064;  
transition: transform 0.2s ease, box-shadow 0.2s ease;  
}  
  
.scorecard-item:hover {  
transform: translateY(-2px);  
box-shadow: 0 3px 10px rgba(0, 191, 255, 0.2);  
}  
  
.scorecard-month {  
font-weight: 600;  
}  
  
.scorecard-count {  
background: #e0f2f1;  
color: #004d40;  
padding: 4px 8px;  
border-radius: 8px;  
font-weight: bold;  
}  
  
.no-record {  
padding: 10px;  
text-align: center;  
color: #607d8b;  
font-style: italic;  
background-color: #e0f2f1;  
border-radius: 10px;  
margin-top: 10px;  
}
```



```
.city-table-wrapper {  
    font-family: 'Segoe UI', sans-serif;  
    max-width: 900px;  
    margin: 0 auto;  
}  
  
.table-container {  
    width: 100%;  
}  
  
/* === General Table Styling for All Summary Tables === */  
#cityPermitTable table,  
#yearlyWaterUserSummary table,  
#cityAccomplishmentTable table {  
    border-radius: 10px;  
    overflow: hidden;  
    border: 1px solid #81d4fa;  
    box-shadow: 0 2px 8px rgba(0, 191, 255, 0.1);  
}  
  
/* Table Headers */  
#cityPermitTable thead th,  
#yearlyWaterUserSummary thead th,  
#cityAccomplishmentTable thead th {  
    background: linear-gradient(to right, #00bcd4, #4dd0e1);  
    color: white;  
    font-size: 0.95rem;  
    font-weight: 600;  
    text-align: center;  
    vertical-align: middle;  
    border: none;  
    position: relative;  
    z-index: 10;  
}  
  
/* Table Cells */  
#cityPermitTable td,  
#cityPermitTable th,  
#yearlyWaterUserSummary td,  
#yearlyWaterUserSummary th,  
#cityAccomplishmentTable td,  
#cityAccomplishmentTable th {  
    vertical-align: middle;  
    padding: 10px;  
    text-align: center;  
    font-size: 0.88rem;  
    color: #004d40;  
}  
  
/* Alternating Row Colors */  
#cityPermitTable tbody tr:nth-child(even),  
#yearlyWaterUserSummary tbody tr:nth-child(even),
```



```
#cityAccomplishmentTable tbody tr:nth-child(even) {
  background-color: #e0f7fa;
}

/* Hover Effects */
#cityPermitTable tbody tr:hover,
#yearlyWaterUserSummary tbody tr:hover,
#cityAccomplishmentTable tbody tr:hover {
  background-color: #b2ebf2;
  transition: background-color 0.2s ease-in-out;
}

/* First Column Style (City Column) */
#cityPermitTable td:first-child,
#yearlyWaterUserSummary td:first-child,
#cityAccomplishmentTable td:first-child {
  font-weight: bold;
  text-align: left;
  padding-left: 16px;
  color: #006064;
}

/* Total Row Highlight */
#cityPermitTable tbody tr:last-child,
#yearlyWaterUserSummary tbody tr:last-child,
#cityAccomplishmentTable tbody tr:last-child {
  background-color: #c8e6c9 !important; /* light green */
  font-weight: bold;
  color: #1b5e20;
  border-top: 2px solid #66bb6a;
}

.city-progress-container {
  display: flex;
  align-items: center;
  gap: 8px;
  min-width: 160px;
  justify-content: center;
}

.city-progress-bar {
  flex-grow: 1;
  height: 18px;
  background-color: #b2ebf2;
  border-radius: 8px;
  overflow: hidden;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.1);
}

.progress-text {
  font-weight: 600;
  color: #006064;
  min-width: 40px;
}
```

```
text-align: right;
font-size: 0.85rem;
}

#yearlyWaterUserSummary table thead th, #cityAccomplishmentTable table thead th,
#barangayModalBody table thead th {
text-align: center;
vertical-align: middle;
border-radius: 0 !important;
}

#settingsTab form .form-label {
font-weight: 500;
}
#passwordStatus {
font-weight: 500;
}

.text-water {
color: #0077b6;
}

.btn-water {
background-color: #00b4d8;
border: none;
color: white;
font-weight: 600;
transition: background-color 0.3s ease;
}

.btn-water:hover {
background-color: #023e8a;
box-shadow: 0 4px 12px rgba(0, 119, 182, 0.3);
filter: brightness(1.05);
color: #fafafa;
}

.form-water:focus {
border-color: #00b4d8;
box-shadow: 0 0 0 0.2rem rgba(0, 180, 216, 0.25);
}

.water-card {
background: linear-gradient(to bottom right, #caf0f8, #ade8f4);
border-left: 6px solid #00b4d8;
border-radius: 1.5rem;
padding: 2rem;
max-width: 600px;
width: 90%;
margin: auto;
box-shadow: 0 8px 20px rgba(0, 180, 216, 0.2);
transition: transform 0.3s ease;
}
```

```
.gear-icon {  
    font-size: 2.5rem;  
    color: #00b4d8;  
    animation: rotateGear 3s linear infinite;  
    display: inline-block;  
    margin-bottom: 0.5rem;  
}
```

```
@keyframes rotateGear {  
    0% { transform: rotate(0deg); }  
    100% { transform: rotate(360deg); }  
}
```

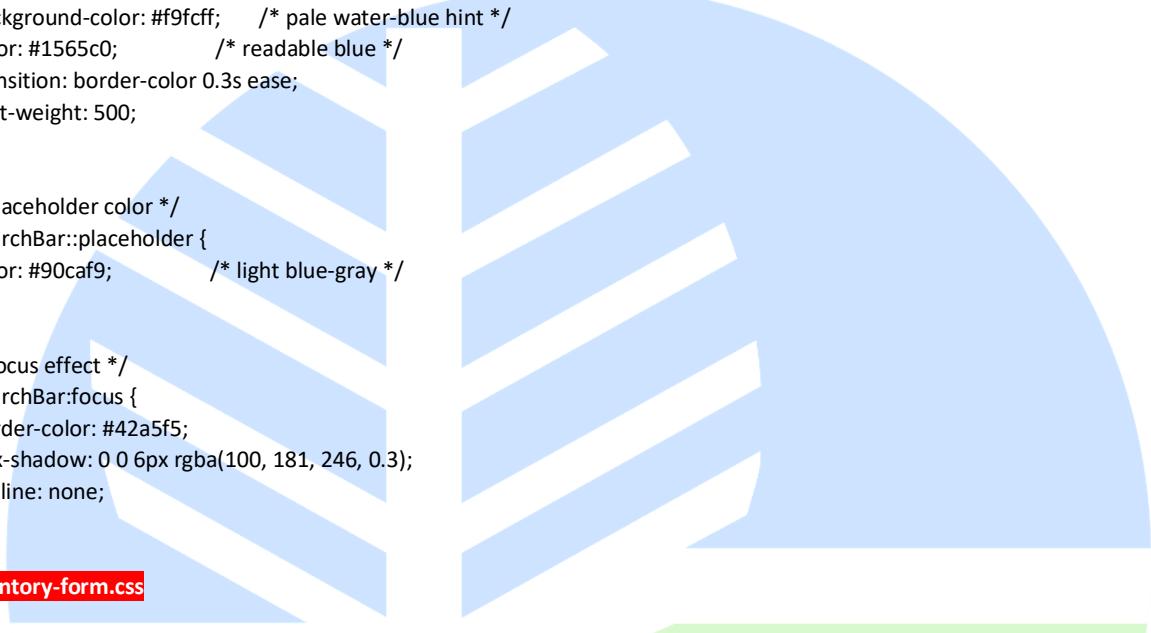
```
.modal {  
    position: fixed;  
    top: 0;  
    left: 0;  
    z-index: 1050;  
}  
.modal-backdrop {  
    z-index: 1040;  
}
```

excel.css

```
.form-label {  
    font-size: 0.9rem;  
    color: #495057;  
}  
.card {  
    border-radius: 10px;  
}  
  
.import-progress-circle {  
    width: 40px;  
    height: 40px;  
    border: 4px solid rgba(0, 123, 255, 0.2);  
    border-top: 4px solid #007bff;  
    border-radius: 50%;  
    animation: import-spin 1s linear infinite;  
    margin: auto;  
}  
  
@keyframes import-spin {  
    to { transform: rotate(360deg); }  
}
```

ics.css

```
/* Input group text (the "Search" label) */  
.input-group-text {
```



```
background-color: #e3f2fd; /* soft sky blue */
border: 1px solid #64b5f6; /* mid-blue border */
color: #1565c0; /* readable water blue */
font-weight: 500;
}

/* Input field */
#searchBar.form-control {
  border: 1px solid #64b5f6; /* water blue border */
  background-color: #f9fcff; /* pale water-blue hint */
  color: #1565c0; /* readable blue */
  transition: border-color 0.3s ease;
  font-weight: 500;
}

/* Placeholder color */
#searchBar::placeholder {
  color: #90caf9; /* light blue-gray */
}

/* Focus effect */
#searchBar:focus {
  border-color: #42a5f5;
  box-shadow: 0 0 6px rgba(100, 181, 246, 0.3);
  outline: none;
}
```

inventory-form.css



```
.info-box {
  background: #fff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  max-width: 600px;
  margin: 20px auto;
}

@media (max-width: 576px) {
  .info-box {
    padding: 15px;
  }
}

.item-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 30px;
  max-width: 1000px;
  margin: 50px auto;
  padding: 50px 0;
  background: #fff;
  border-radius: 8px;
```

```
    box-shadow: inset 0 2px 6px rgba(0, 0, 0, 0.08);  
}
```

```
.item-container > div {  
  display: flex;  
  justify-content: center;  
}
```

```
@media (max-width: 992px) {  
  .item-container {  
    grid-template-columns: repeat(2, 1fr);  
  }  
}  
}
```

```
@media (max-width: 576px) {  
  .item-container {  
    grid-template-columns: 1fr;  
  }  
}
```

```
.big-plus-btn {  
  width: 140px;  
  height: 140px;  
  border-radius: 50%;  
  padding: 0;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);  
  font-size: 3rem;  
}
```

```
.big-plus-btn i {  
  font-size: 3rem;  
}
```

```
.file-link {  
  position: relative;  
  display: inline-flex;  
  align-items: center;  
  justify-content: center;  
  width: 140px;  
  height: 140px;  
  border-radius: 12px;  
  text-decoration: none;  
  background-color: #f8f9fa;  
  border: 1px solid #ddd;  
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);  
  transition: transform 0.2s ease;  
}
```

```
.file-link:hover {  
  transform: scale(1.05);
```

```
}

.file-link i {
  font-size: 3rem;
  color: #0d6efd;
}

.file-number {
  position: absolute;
  bottom: 8px;
  right: 8px;
  background: #0d6efd;
  color: white;
  font-size: 1rem;
  font-weight: bold;
  padding: 4px 8px;
  border-radius: 50%;
}

.delete-entry-btn {
  width: 60px;
  height: 60px;
  border-radius: 50%;
  font-size: 2rem;
  display: flex;
  align-items: center;
  justify-content: center;
  box-shadow: 0 3px 6px rgba(0, 0, 0, 0.2);
  transition: transform 0.2s ease-in-out;
}

.delete-entry-btn:hover {
  transform: scale(1.1);
  background-color: #c82333;
  color: #fff;
}

#signatureCanvas {
  cursor: url('/public/images/pen-cursor.png') 0 32, auto;
}

#image-signature{
  width: 350px;
  height: 100px;
  margin: 0 auto;
  border: 1px solid #999;
}

inventory-summary.css

#loadingOverlay {
  display: flex;
}
```



```
.table td, .table th {  
    vertical-align: middle;  
}  
  
.btn i {  
    vertical-align: middle;  
}  
  
.modal-body {  
    max-height: 70vh;  
}  
  
#scrollableTableWrapper {  
    height: 100%;  
    overflow-x: auto;  
    overflow-y: auto;  
}  
  
.property-accountability-table {  
    min-width: 1000px;  
}  
  
.modal-xl-custom {  
    max-width: 95% !important;  
    width: 95%;  
}  
  
.modal-fullscreen-custom {  
    max-width: 98% !important;  
    width: 98%;  
    margin: auto;  
}  
  
.modal-body {  
    padding: 0 !important;  
}  
  
.modal-body embed {  
    width: 100%;  
    height: 95vh;  
    display: block;  
}  
  
#searchInput.form-control {  
    background-color: #f9fcff;  
    border: 1px solid #64b5f6;  
    color: #1565c0;  
    font-weight: 500;  
    transition: border-color 0.3s ease, box-shadow 0.3s ease;  
}  
  
#searchInput::placeholder {
```

```
color: #90caf9;
}

#searchInput:focus {
  border-color: #42a5f5;
  box-shadow: 0 0 6px rgba(100, 181, 246, 0.3);
  outline: none;
}

.btn-water {
  background-color: #ffffff;
  color: #117a8b;
  border: 1px solid #117a8b;
  transition: all 0.2s ease-in-out;
}

.btn-water:hover {
  background-color: #117a8b;
  color: #ffffff;
}

.btn-water-alt {
  background-color: #e0f0fb;
  color: #0d6efd;
  border: 1px solid #0d6efd;
  transition: all 0.2s ease-in-out;
}

.btn-water-alt:hover {
  background-color: #0d6efd;
  color: #ffffff;
}

.reorder-warning td {
  background: linear-gradient(to right, #fff0f0, #ffe5e5); /* soft red tint */
  color: #660000; /* deep red text for contrast */
  font-weight: bold;
  border-bottom: 1px solid #ffcccc;
}

h5{
  color: #fff !important
}
```

login.css

```
html, body {
  margin: 0;
  padding: 0;
  height: 100%;
}

body {
```

```
margin: 0;
padding: 0;
height: 100vh;
background: radial-gradient(ellipse at center, #001f3f 0%, #001a35 100%);
overflow: hidden;
display: flex;
justify-content: center;
align-items: center;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
position: relative;
}

.portal {
  position: absolute;
  width: 300px;
  height: 300px;
  border-radius: 50%;
  background: radial-gradient(circle, rgba(0, 191, 255, 0.7) 0%, transparent 70%);
  box-shadow: 0 0 40px 20px rgba(0, 191, 255, 0.3), 0 0 60px rgba(0, 191, 255, 0.5) inset;
  animation: portalOpen 2s ease-out forwards;
  z-index: 1;
}

@keyframes portalOpen {
  0% {
    transform: scale(0) rotate(0deg);
    opacity: 0;
  }
  50% {
    transform: scale(1.2) rotate(180deg);
    opacity: 1;
  }
  100% {
    transform: scale(1) rotate(360deg);
    opacity: 1;
  }
}

.portal::before,
.portal::after {
  content: "";
  position: absolute;
  border: 2px dashed rgba(0, 191, 255, 0.7);
  border-radius: 50%;
  top: -20px;
  left: -20px;
  width: 340px;
  height: 340px;
  animation: rotateDash 8s linear infinite;
}

.portal::after {
  border: 2px dotted rgba(0, 191, 255, 0.5);
```

```
top: -30px;  
left: -30px;  
width: 360px;  
height: 360px;  
animation-direction: reverse;  
}  
  
@keyframes rotateDash {  
0% {  
transform: rotate(0deg);  
}  
100% {  
transform: rotate(360deg);  
}  
}  
  
.card {  
backdrop-filter: blur(20px);  
background: rgba(255, 255, 255, 0.1);  
border-radius: 1rem;  
border: 1px solid rgba(255, 255, 255, 0.3);  
box-shadow: 0 8px 32px rgba(31, 38, 135, 0.37);  
padding: 2rem;  
width: 100%;  
max-width: 400px;  
z-index: 10;  
}  
  
.card-title {  
font-size: 2rem;  
font-weight: 700;  
color: #fff;  
}  
  
.form-label {  
color: #fff;  
}  
  
.form-control {  
background: rgba(255, 255, 255, 0.15);  
border: none;  
color: #fff;  
}  
  
.form-control::placeholder {  
color: rgba(255, 255, 255, 0.7);  
}  
  
.form-control:focus {  
background: rgba(255, 255, 255, 0.3);  
color: #fff;  
border: none;  
box-shadow: none;
```



```
}
```

```
.btn-primary {  
background-color: #00b4d8;  
border: none;  
transition: background-color 0.3s ease;  
}  
  
.btn-primary:hover {  
background-color: #0096c7;  
}  
  
#errorMsg {  
color: #ff6b6b;  
}  
  
/* Alert Overlay */  
#portal-alert {  
display: none;  
position: fixed;  
inset: 0;  
background: rgba(0, 10, 20, 0.7);  
backdrop-filter: blur(8px);  
z-index: 9999;  
justify-content: center;  
align-items: center;  
}  
  
/* Show State */  
#portal-alert.show {  
display: flex;  
}  
  
/* Alert Box */  
.portal-alert-content {  
background: #121e2a;  
color: #f0f6ff;  
border-radius: 16px;  
padding: 24px;  
width: 100%;  
max-width: 400px;  
box-shadow: 0 8px 24px rgba(0, 0, 0, 0.6);  
border: 1px solid rgba(0, 191, 255, 0.3);  
display: flex;  
flex-direction: column;  
}  
  
/* Header */  
.portal-alert-header h4 {  
margin: 0;  
font-size: 1.4rem;  
color: #00d4ff;  
text-align: center;
```

```
}

/* Body */
.portal-alert-body {
  margin: 16px 0;
  text-align: center;
  font-size: 1rem;
  line-height: 1.5;
}

/* Footer */
.portal-alert-footer {
  display: flex;
  justify-content: center;
}

/* Button */
.portal-alert-footer button {
  background-color: #00b4d8;
  color: #fff;
  border: none;
  padding: 8px 24px;
  border-radius: 8px;
  cursor: pointer;
  font-weight: 600;
  transition: background-color 0.3s ease;
}

.portal-alert-footer button:hover {
  background-color: #0096c7;
}

.portal-alert-footer button:active {
  background-color: #0077a5;
}

.login-success-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 255, 255, 0.1);
  display: none;
  align-items: center;
  justify-content: center;
  z-index: 9999;
  backdrop-filter: blur(4px);
}

.login-success-content {
  background-color: #000000ee;
  padding: 30px;
```

```
border-radius: 20px;  
text-align: center;  
box-shadow: 0 0 20px #00ffff;  
color: #ffff;  
}
```

```
@keyframes spin {  
0% { transform: rotate(0deg); }  
100% { transform: rotate(360deg); }  
}
```

```
.aqua-container {  
position: relative;  
width: 80px;  
height: 80px;  
margin: 0 auto 15px auto;  
}
```

```
.aqua-wave {  
position: absolute;  
top: 50%;  
left: 50%;  
width: 80px;  
height: 80px;  
background: rgba(0, 255, 255, 0.15); /* lighter cyan */  
border-radius: 50%;  
transform: translate(-50%, -50%);  
animation: aquaPulse 2.5s ease-out infinite;  
z-index: 0;  
}
```

```
@keyframes aquaPulse {  
0% {  
transform: translate(-50%, -50%) scale(1);  
opacity: 0.3;  
}  
50% {  
transform: translate(-50%, -50%) scale(1.2);  
opacity: 0.15;  
}  
100% {  
transform: translate(-50%, -50%) scale(1.4);  
opacity: 0;  
}
```

```
.gear-icon {  
width: 70px;  
height: 70px;  
display: flex;  
align-items: center;  
justify-content: center;  
font-size: 36px;
```

```

border: 4px solid #00ffff;
border-top: 4px solid transparent;
border-radius: 50%;
animation: spin 1.5s linear infinite;
margin: 0 auto;
text-shadow: 0 0 10px #00ffff;
color: #00ffff;
background-color: #000;
position: relative;
z-index: 1;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}

/* Portal Light */

#portal-light {
  position: fixed;
  top: 50%;
  left: 50%;
  width: 150px;
  height: 150px;
  background: radial-gradient(circle, #ffffff 0%, #f0f0f0 70%);
  border-radius: 50%;
  transform: translate(-50%, -50%) scale(0);
  opacity: 0;
  pointer-events: none;
  z-index: 9999;
}

#portal-light.active {
  animation: portalLightOpen 5s ease forwards;
}

@keyframes portalLightOpen {
  0% {
    transform: translate(-50%, -50%) scale(0.2);
    opacity: 0.3;
    background: radial-gradient(circle, #ffffff 0%, #f5f5f5 70%);
  }
  30% {
    transform: translate(-50%, -50%) scale(1);
    opacity: 0.7;
  }
  70% {
    transform: translate(-50%, -50%) scale(10);
    opacity: 0.95;
  }
  100% {
    transform: translate(-50%, -50%) scale(50);
  }
}

```

```
background: white;
opacity: 1;
}

/*
* Portal Loader Styles */
.portal-loader-overlay {
position: fixed;
top: 0;
left: 0;
width: 100%;
height: 100%;
background-color: rgba(0, 0, 0, 0.9);
display: none;
justify-content: center;
align-items: center;
z-index: 9999;
}

.portal-loader-overlay.show {
display: flex;
}

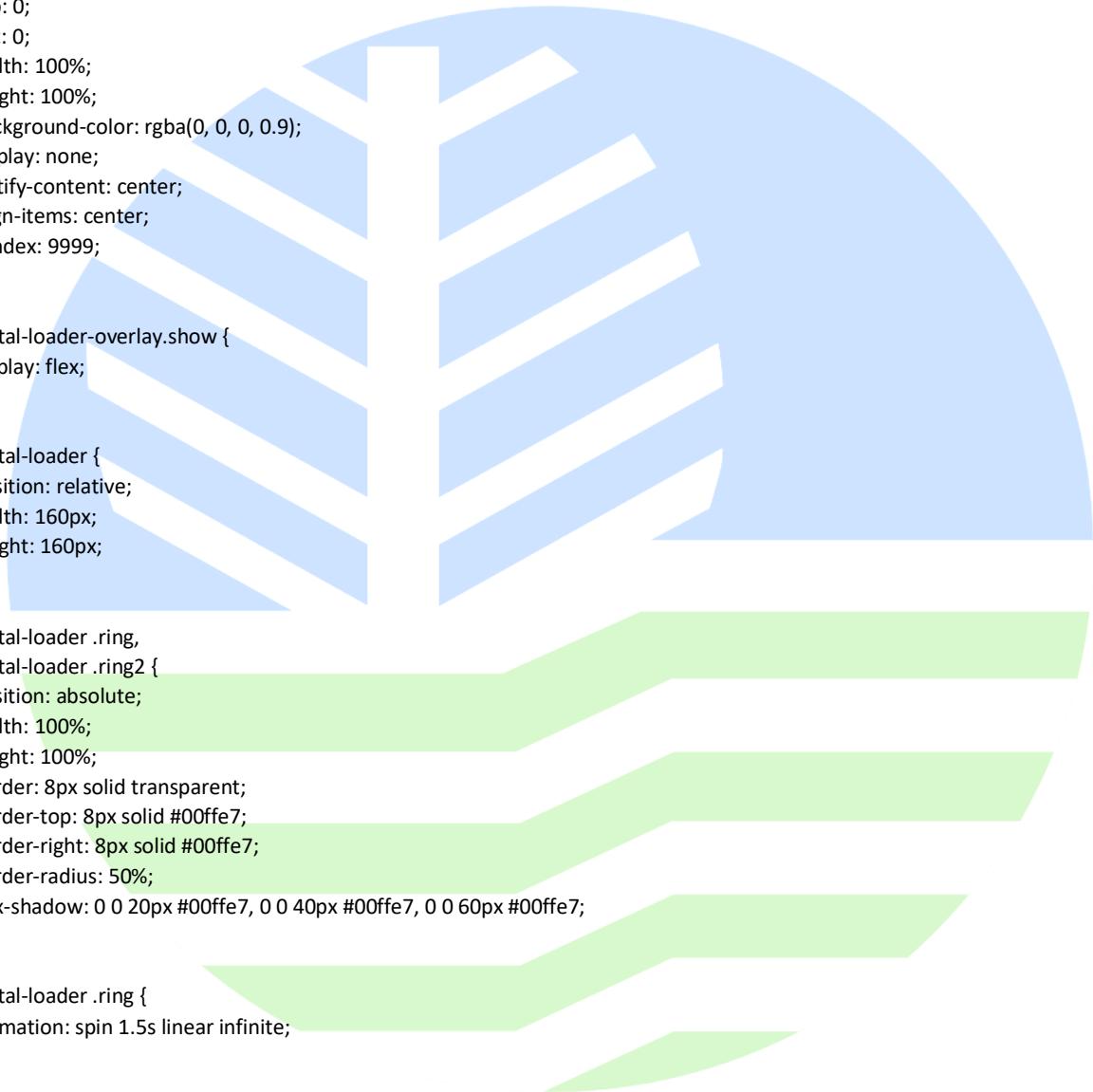
.portal-loader {
position: relative;
width: 160px;
height: 160px;
}

.portal-loader .ring,
.portal-loader .ring2 {
position: absolute;
width: 100%;
height: 100%;
border: 8px solid transparent;
border-top: 8px solid #00ffe7;
border-right: 8px solid #00ffe7;
border-radius: 50%;
box-shadow: 0 0 20px #00ffe7, 0 0 40px #00ffe7, 0 0 60px #00ffe7;
}

.portal-loader .ring {
animation: spin 1.5s linear infinite;
}

.portal-loader .ring2 {
animation: spinReverse 2s linear infinite;
opacity: 0.6;
}

.portal-loader .glow {
position: absolute;
top: 50%;
```



```
left: 50%;  
width: 80%;  
height: 80%;  
transform: translate(-50%, -50%);  
border-radius: 50%;  
background: radial-gradient(circle, rgba(0, 255, 231, 0.3) 0%, transparent 70%);  
box-shadow: 0 0 40px rgba(0, 255, 231, 0.7), 0 0 60px rgba(0, 255, 231, 0.5);  
animation: pulse 2s ease-in-out infinite;  
}
```

```
.portal-loader h2 {  
margin-top: 180px;  
color: white;  
font-family: 'Orbitron', sans-serif;  
font-weight: bold;  
text-transform: uppercase;  
letter-spacing: 2px;  
text-align: center;  
}
```

```
/* Animations */  
@keyframes spin {  
0% { transform: rotate(0deg); }  
100% { transform: rotate(360deg); }  
}
```

```
@keyframes spinReverse {  
0% { transform: rotate(0deg); }  
100% { transform: rotate(-360deg); }  
}
```

```
@keyframes pulse {  
0%, 100% {  
transform: translate(-50%, -50%) scale(1);  
opacity: 0.6;  
}  
50% {  
transform: translate(-50%, -50%) scale(1.1);  
opacity: 1;  
}  
}
```

map-page.css

```
.map-container {  
position: relative;  
height: 100vh;  
width: 100%;  
}
```

```
#routingMap {  
height: 100%;  
width: 100%;
```

```
border-radius: 8px;  
}  
  
/* Floating Form Styling */  
.routing-form {  
position: absolute;  
top: 10px;  
left: 10px;  
width: 250px;  
background: white;  
padding: 10px;  
border-radius: 8px;  
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);  
z-index: 1000;  
}  
  
.form-button-wrapper {  
position: absolute;  
top: 10px;  
left: 10px;  
display: flex;  
flex-direction: column;  
align-items: flex-start;  
gap: 8px;  
z-index: 1100;  
}  
  
#toggleRoutingFormBtn {  
background: #ffffff;  
border: 1px solid #ccc;  
border-radius: 6px;  
box-shadow: 0 2px 5px rgba(0, 0, 0, 0.15);  
}  
  
#toggleRoutingFormBtn:hover {  
background: #000;  
color: #fff;  
}  
  
.routing-form {  
position: static;  
width: 250px;  
background: #ffffff;  
padding: 10px;  
border-radius: 8px;  
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.25);  
}  
  
.route-indicator {  
background: linear-gradient(135deg, #007bff, #00c6ff);  
color: white;  
font-weight: bold;
```

```
font-size: 1rem;  
padding: 12px 20px;  
border-radius: 8px;  
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);  
width: fit-content;  
margin: 10px auto;  
}  
  
.permit-popup {  
font-size: 14px;  
line-height: 1.6;  
color: #2c3e50;  
padding: 6px 2px;  
}  
  
.permit-popup-row {  
margin-bottom: 4px;  
}  
  
.permit-pdf-link {  
color: #007bff;  
text-decoration: none;  
font-weight: bold;  
display: inline-block;  
margin-top: 4px;  
}  
  
.permit-pdf-link:hover {  
color: #0056b3;  
text-decoration: underline;  
}  
  
.pulse-marker {  
background: rgba(0, 150, 255, 0.4);  
border: 3px solid rgba(0, 150, 255, 0.8);  
border-radius: 50%;  
width: 35px; /* ↑ bigger marker */  
height: 35px;  
position: relative;  
}  
  
.pulse-marker::after {  
content: "";  
position: absolute;  
left: -20px; /* adjusted for new size */  
top: -20px;  
width: 75px; /* ↑ bigger pulse ring */  
height: 75px;  
border-radius: 50%;  
background: rgba(0, 150, 255, 0.3);  
animation: pulse 1.5s infinite;  
}
```



```

@keyframes pulse {
0% {
  transform: scale(0.5);
  opacity: 0.8;
}
70% {
  transform: scale(2.2); /* ↑ bigger expansion */
  opacity: 0;
}
100% {
  transform: scale(0.5);
  opacity: 0;
}
}

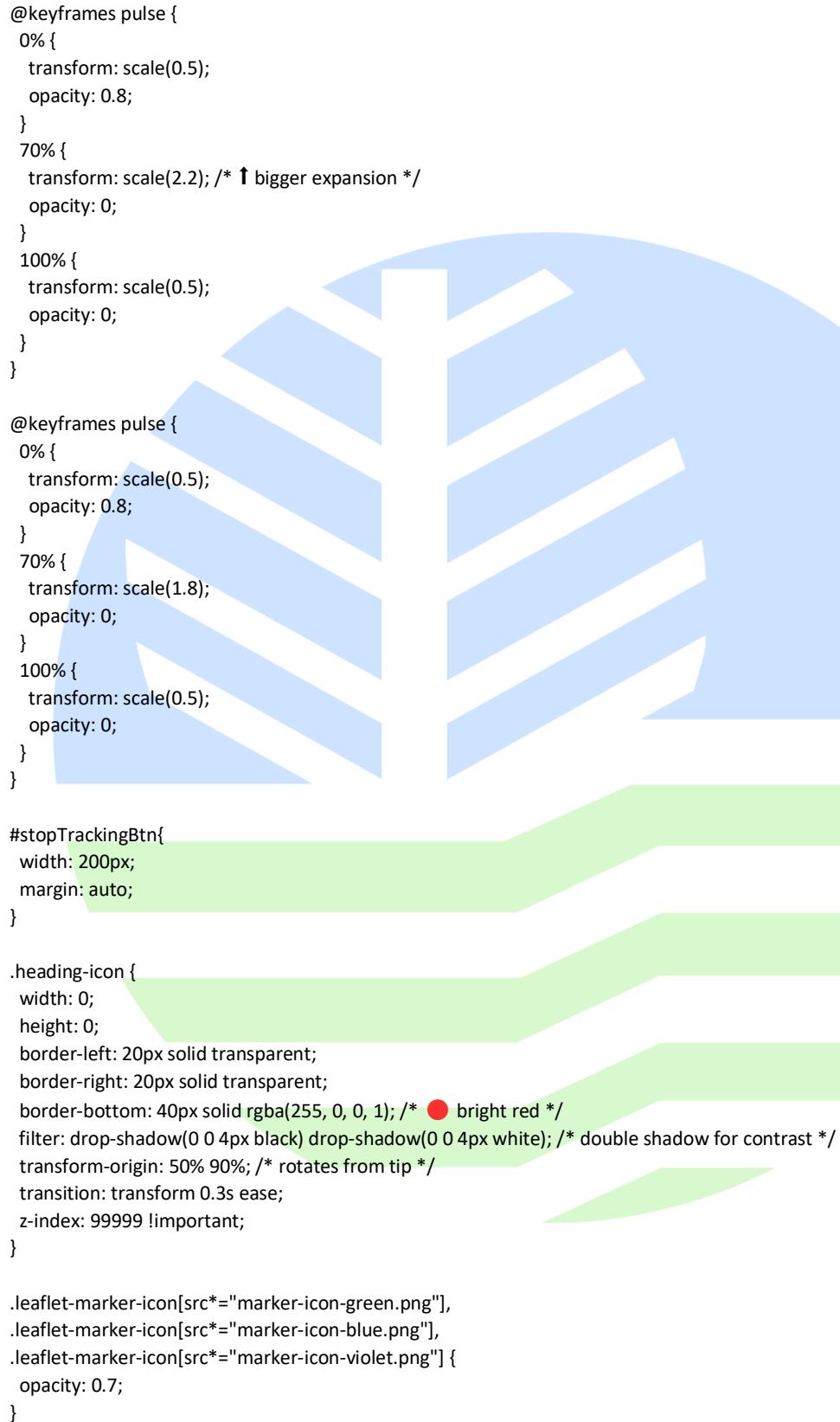
@keyframes pulse {
0% {
  transform: scale(0.5);
  opacity: 0.8;
}
70% {
  transform: scale(1.8);
  opacity: 0;
}
100% {
  transform: scale(0.5);
  opacity: 0;
}
}

#stopTrackingBtn{
width: 200px;
margin: auto;
}

.heading-icon {
width: 0;
height: 0;
border-left: 20px solid transparent;
border-right: 20px solid transparent;
border-bottom: 40px solid rgba(255, 0, 0, 1); /* ● bright red */
filter: drop-shadow(0 0 4px black) drop-shadow(0 0 4px white); /* double shadow for contrast */
transform-origin: 50% 90%; /* rotates from tip */
transition: transform 0.3s ease;
z-index: 99999 !important;
}

.leaflet-marker-icon[src*="marker-icon-green.png"],
.leaflet-marker-icon[src*="marker-icon-blue.png"],
.leaflet-marker-icon[src*="marker-icon-violet.png"] {
  opacity: 0.7;
}

```

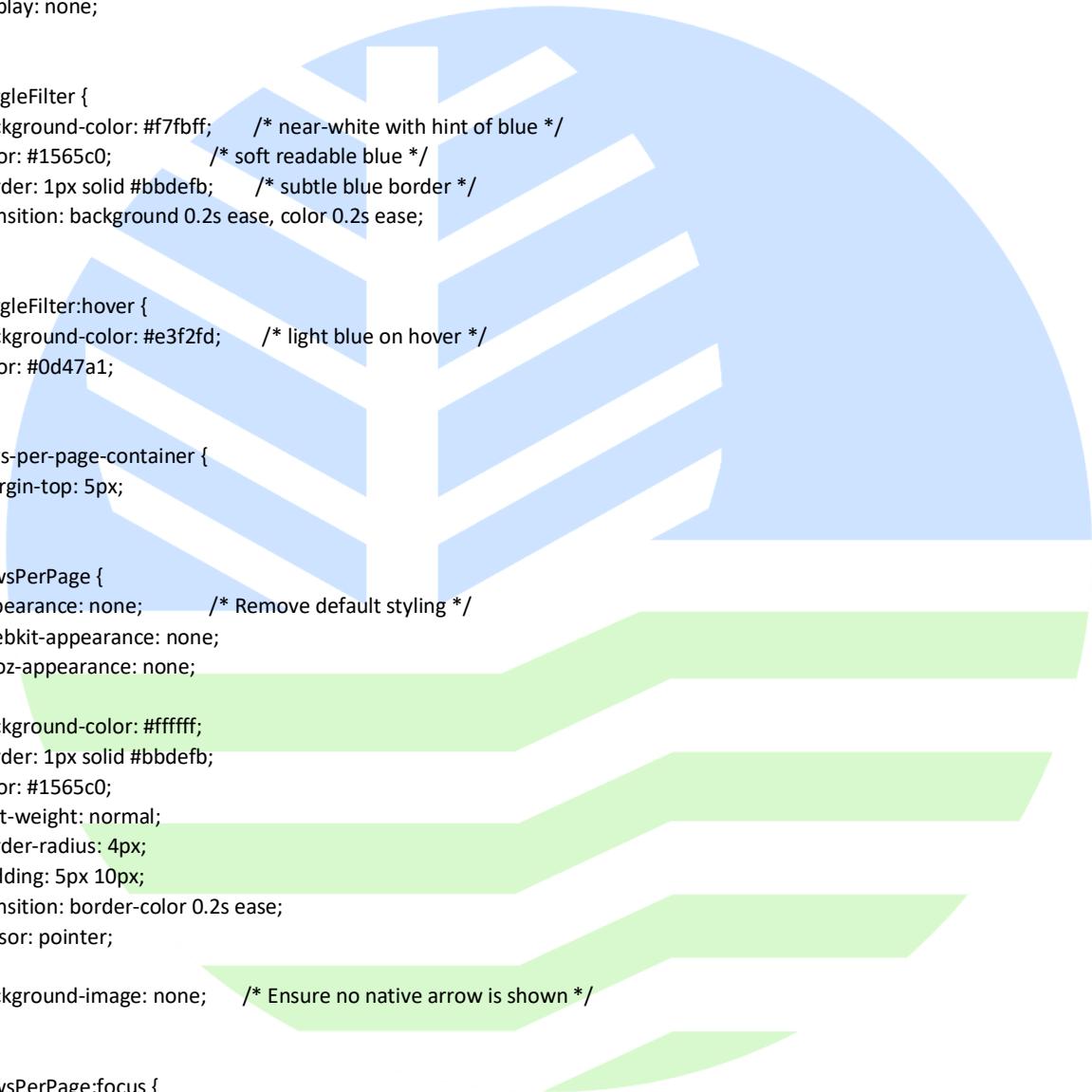


map.css

```
#leafletMap {  
    height: 100vh;  
    width: 100%;  
}  
  
.barangay-label {  
    font-size: 11px;  
    font-weight: bold;  
    color: white;  
    text-shadow: 1px 1px 2px black;  
    text-align: center;  
}
```

permit.css

```
.badge-3d {  
    position: relative;  
    display: inline-block;  
    padding: 0.35em 0.7em;  
    font-size: 0.75rem;  
    font-weight: 600;  
    line-height: 1;  
    color: white;  
    border-radius: 0.5rem;  
    text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);  
    box-shadow:  
        inset 0 1px 0 rgba(255, 255, 255, 0.2),  
        inset 0 -1px 0 rgba(0, 0, 0, 0.1),  
        0 2px 4px rgba(0, 0, 0, 0.15);  
    overflow: hidden;  
}  
  
.badge-danger {  
    background: linear-gradient(to bottom, #dc3545, #c82333);  
}  
  
.badge-success {  
    background: linear-gradient(to bottom, #28a745, #218838);  
}  
  
.badge-3d::after {  
    content: "";  
    position: absolute;  
    top: 0;  
    left: -50%;  
    width: 30%;  
    height: 100%;  
    background: linear-gradient(  
        120deg,  
        rgba(255, 255, 255, 0.05) 0%,  
        rgba(255, 255, 255, 0.15) 50%,
```



```
rgba(255, 255, 255, 0.05) 100%
);
transform: skewX(-20deg);
animation: shine 10s ease-in-out infinite;
pointer-events: none;
}

#filterSection{
display: none;
}

#toggleFilter {
background-color: #f7fbff; /* near-white with hint of blue */
color: #1565c0; /* soft readable blue */
border: 1px solid #bbdefb; /* subtle blue border */
transition: background 0.2s ease, color 0.2s ease;
}

#toggleFilter:hover {
background-color: #e3f2fd; /* light blue on hover */
color: #0d47a1;
}

.rows-per-page-container {
margin-top: 5px;
}

#rowsPerPage {
appearance: none; /* Remove default styling */
-webkit-appearance: none;
-moz-appearance: none;

background-color: #ffffff;
border: 1px solid #bbdefb;
color: #1565c0;
font-weight: normal;
border-radius: 4px;
padding: 5px 10px;
transition: border-color 0.2s ease;
cursor: pointer;

background-image: none; /* Ensure no native arrow is shown */
}

#rowsPerPage:focus {
outline: none;
border-color: #64b5f6;
}

label[for="rowsPerPage"] {
color: #1565c0;
font-weight: 500;
}
```

```
.water-search {  
    border-radius: 4px;  
    overflow: hidden;  
    box-shadow: none;  
}  
  
.water-search input.form-control {  
    background-color: #ffffff;  
    border: 1px solid #bbdefb;  
    color: #1565c0;  
    font-weight: normal;  
    border-right: none;  
    transition: border-color 0.2s ease;  
}  
  
.water-search input.form-control::placeholder {  
    color: #90caf9;  
}  
  
.water-search input.form-control:focus {  
    border-color: #64b5f6;  
    outline: none;  
}  
  
.water-search .input-group-text {  
    background-color: #e3f2fd;  
    border: 1px solid #bbdefb;  
    color: #1565c0;  
    font-weight: 500;  
}  
  
.action-buttons {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
    justify-content: center;  
    gap: 6px;  
}  
  
.action-buttons .btn {  
    width: 36px;  
    height: 36px;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
    padding: 0;  
    border-radius: 6px;  
}  
  
#imageUploadModal .modal-body {  
    min-height: 500px;  
}
```

```
#imagePreview {  
  max-height: 400px;  
  object-fit: contain;  
  transition: all 0.3s ease-in-out;  
}  
  
#imagePreview.placeholder {  
  opacity: 0.5;  
  filter: grayscale(100%);  
  border: 2px dashed #ccc;  
}  
  
@media (max-width: 576px) {  
  #extrasSection {  
    margin-bottom: 1rem;  
  }  
}  
  
user.css  
  
/* Subtle blue-themed input group */  
.input-group .form-control {  
  background-color: #fff;  
  border: 1px solid #bbdefb; /* light blue border */  
  color: #1565c0; /* readable deep blue text */  
  transition: border-color 0.2s ease;  
}  
  
.input-group .form-control::placeholder {  
  color: #90caf9; /* light blue placeholder */  
}  
  
.input-group .form-control:focus {  
  border-color: #64b5f6; /* blue highlight on focus */  
  box-shadow: 0 0 4px rgba(100, 181, 246, 0.3);  
  outline: none;  
}  
  
.input-group-text {  
  background-color: #e3f2fd; /* light blue background */  
  border: 1px solid #bbdefb;  
  color: #1565c0;  
}  
  
/* Add User Button with subtle blue hint */  
#showAddUserFormBtn {  
  border-color: #64b5f6;  
  color: #1565c0;  
  transition: all 0.2s ease;  
}  
  
#showAddUserFormBtn:hover {
```

```
background-color: #64b5f6;  
color: white;  
box-shadow: 0 0 8px rgba(100, 181, 246, 0.3);  
}
```

wrus.css

```
.form-water {  
background-color: #eaf7fc;  
border-color: #b6e0f2;  
transition: border-color 0.2s ease-in-out, box-shadow 0.2s ease-in-out;  
}
```

```
.form-water:focus {  
border-color: #0d6efd;  
box-shadow: 0 0 0.2rem rgba(13, 110, 253, 0.25);  
}
```

```
#image-signature{  
width: 350px;  
height: 100px;  
margin: 0 auto  
}
```

```
tr.table-info {  
position: relative;  
background-color: rgba(33, 150, 243, 0.12);  
border-left: 6px solid #0288d1;  
font-weight: 600;  
color: #003c66;  
transition: background-color 0.2s ease;  
z-index: 1;  
}
```

```
tr.table-info > td,  
tr.table-info > th {  
background-color: inherit;  
color: inherit;  
}
```

```
tr.table-info:hover {  
background-color: rgba(33, 150, 243, 0.18);  
color: #003c66;  
}
```

```
tr.table-info:hover > td,  
tr.table-info:hover > th {  
background-color: inherit;  
color: inherit;  
}
```

```
/* Water Source Filter Section Styles */  
#waterSourceFilterSection {
```

```
border: 1px solid #b6d4fe;
background-color: #e9f5ff;
border-radius: 0.5rem;
padding: 1rem;
transition: all 0.3s ease-in-out;
box-shadow: 0 0 8px rgba(0, 123, 255, 0.1);
}

/* Legend styling */
#waterSourceFilterSection .badge {
  border-radius: 4px;
}

#waterSourceFilterSection .form-check {
  margin-bottom: 0.75rem;
}

#toggleWaterSourceFilterBtn {
  transition: all 0.3s ease-in-out;
  border-color: #00bcd4;
  color: #00acc1;
  background-color: transparent;
}

#toggleWaterSourceFilterBtn:hover {
  background-color: rgba(0, 172, 193, 0.15);
  color: #005f73;
  border-color: #00acc1;
}

.geotag-btn {
  background-color: #0dcaf0;
  color: white;
  transition: background-color 0.2s ease-in-out;
}

.geotag-btn:hover {
  background-color: #31d2f2;
  color: #fafafa;
}

.badge-3d {
  position: relative;
  display: inline-block;
  padding: 0.35em 0.7em;
  font-size: 0.75rem;
  font-weight: 600;
  line-height: 1;
  color: white;
  border-radius: 0.5rem;
  text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
  box-shadow:
    inset 0 1px 0 rgba(255, 255, 255, 0.2),
    0 0 10px 2px rgba(255, 255, 255, 0.2);
}
```



```
        inset 0 -1px 0 rgba(0, 0, 0, 0.1),  
        0 2px 4px rgba(0, 0, 0, 0.15);  
    overflow: hidden;  
}  
  
.badge-permittee {  
    background: linear-gradient(to bottom, #28a745, #218838);  
}  
  
.badge-water-source {  
    background: linear-gradient(to bottom, #007bff, #0062cc);  
}  
  
.badge-water-source-bw {  
    background: linear-gradient(to bottom, #000000, #333333);  
    color: white;  
}  
  
.badge-3d::after {  
    content: "";  
    position: absolute;  
    top: 0;  
    left: -50%;  
    width: 30%;  
    height: 100%;  
    background: linear-gradient(  
        120deg,  
        rgba(255, 255, 255, 0.05) 0%,  
        rgba(255, 255, 255, 0.15) 50%,  
        rgba(255, 255, 255, 0.05) 100%  
    );  
    transform: skewX(-20deg);  
    animation: shine 10s ease-in-out infinite;  
    pointer-events: none;  
}
```

XV.C. Javascript

XV.C.1. Authentication Scripts

login.js

```
import { AuthHandler } from "./auth/auth.js";  
import { db } from "./firebaseConfig.js";  
import {  
    collection,  
    query,  
    where,  
    getDocs
```



```
 } from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio.js";
import bcrypt from "https://esm.sh/bcryptjs@2.4.3";

import { PortalLoader } from './components/portalLoader.js';
import { PortalAlert } from './components/loginAlert.js';
import { LoginSuccess } from './components/loginSuccess.js';
import { PortalLight } from './components/PortalLight.js';

document.addEventListener('DOMContentLoaded', () => {
  AuthHandler.checkLoginStatus();
  PortalLoader.render();
  LoginSuccess.render();
  PortalLight.render();

  const loginForm = document.getElementById('loginForm');

  if (loginForm) {
    loginForm.addEventListener('submit', async (e) => {
      e.preventDefault();

      const username = document.getElementById('username').value.trim();
      const password = document.getElementById('password').value.trim();

      PortalLoader.show();

      try {
        const q = query(collection(db, "users"), where("username", "==", username));
        const querySnapshot = await getDocs(q);

        if (!querySnapshot.empty) {
          const docSnap = querySnapshot.docs[0];
          const userData = docSnap.data();

          const userRole = userData.role?.toLowerCase();
          const userStatus = userData.status?.toLowerCase();
          const userType = userData.type || "";

          if (userRole !== "admin" && userStatus === "inactive") {
            PortalLoader.hide();
            PortalAlert.show("Your account is inactive. Please contact the administrator.");
            return;
          }

          const hashedPassword = userData.password;
          const isMatch = bcrypt.compareSync(password, hashedPassword);

          if (isMatch) {
            const fullName = `${userData.firstName} ${userData.middleInitial}. ${userData.lastName}`;

            localStorage.setItem("userFullName", fullName);
            localStorage.setItem("loggedInUser", docSnap.id);
            localStorage.setItem("userRole", userRole);
            localStorage.setItem("wrusUserId", docSnap.id);
          }
        }
      } catch (error) {
        console.error(error);
      }
    });
  }
})
```

```
localStorage.setItem("userType", userType);

PortalLoader.hide();
LoginSuccess.show();
//PortalLight.trigger();

// ✅ Add delay before redirecting so the overlay is visible
setTimeout(() => {
  window.location.href = userRole === "admin" ? "admin-dashboard.html" : "dashboard.html";
}, 1500); // 4 seconds
} else {
  PortalLoader.hide();
  PortalAlert.show("Incorrect password.");
}
} else {
  PortalLoader.hide();
  PortalAlert.show("User not found.");
}

} catch (error) {
  PortalLoader.hide();
  console.error("Login error:", error);
  PortalAlert.show("Something went wrong. Please try again.");
}
});
}
});
```

firebase.config

```
// firebaseConfig.js
import { initializeApp } from "https://www.gstatic.com/firebasejs/10.11.1/firebase-app.js";
import { getFirestore } from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio-firestore.js";

const firebaseConfig = {
  apiKey: "AlzaSy*****",
  authDomain: "wrus-asset-tracking-system.firebaseio.com",
  projectId: "wrus-asset-tracking-system",
  storageBucket: "wrus-asset-tracking-system.appspot.com",
  messagingSenderId: "94619*****",
  appId: "1:*****"
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

export { db };
```

auth.js

```
export const AuthHandler = {
  checkLoginStatus() {
    const loggedInUser = localStorage.getItem("loggedInUser");
```

```

const userRole = localStorage.getItem("userRole");

if (loggedInUser) {
  this.redirectUser(userRole);
}
};

redirectUser(role) {
  if (role === "admin") {
    window.location.href = "admin-dashboard.html";
  } else {
    window.location.href = "dashboard.html";
  }
}
};

export const SessionGuard = {
  ensureLoggedIn() {
    const loggedInUser = localStorage.getItem("loggedInUser");
    if (!loggedInUser) {
      window.location.href = "index.html";
    }
  }
};

export const AdminGuard = {
  verify() {
    const user = localStorage.getItem("loggedInUser");
    const userRole = localStorage.getItem("userRole");

    // Redirect if not logged in
    if (!user) {
      window.location.href = "index.html";
      return;
    }

    // Redirect if role is not admin
    if (userRole !== "admin") {
      window.location.href = "404.html";
    }
  }
};

```

XV.C.2. Component Scripts

loginAlert.js

```

export const PortalAlert = {
  show(message, title = "Notice") {
    const alert = document.getElementById("portal-alert");
    alert.classList.add("show");
    document.getElementById("portal-alert-message").innerText = message;
    document.getElementById("portal-alert-title").innerText = title;
  }
};

```

```
},
close() {
  const alert = document.getElementById("portal-alert");
  alert.classList.remove("show");
}
};

// Attach the OK button event after DOM is ready
document.addEventListener("DOMContentLoaded", () => {
  const okButton = document.getElementById("portal-alert-ok");
  if (okButton) {
    okButton.addEventListener("click", () => {
      PortalAlert.close();
    });
  }
});
```

loginSuccess.js

```
export const LoginSuccess = {
  render() {
    if (document.getElementById('login-success-overlay')) return;

    const overlay = document.createElement('div');
    overlay.id = 'login-success-overlay';
    overlay.className = 'login-success-overlay';
    overlay.innerHTML = `
      <div class="login-success-content">
        <div class="aqua-container">
          <div class="aqua-wave"></div>
          <div class="gear-icon">
            <i class="fas fa-cog"></i>
          </div>
        </div>
        <p>Login Successful</p>
      </div>
    `;
    document.body.appendChild(overlay);
  },
  show() {
    const overlay = document.getElementById('login-success-overlay');
    if (overlay) overlay.style.display = 'flex';
  },
  hide() {
    const overlay = document.getElementById('login-success-overlay');
    if (overlay) overlay.style.display = 'none';
  }
};
```

notification.js

```
export const NotificationBox = {
  show(message, type = 'success', duration = 3000) {
```

```

const notif = document.createElement('div');
notif.classList.add('notification', type);
notif.textContent = message;
document.body.appendChild(notif);

setTimeout(() => notif.classList.add('show'), 10);

setTimeout(() => {
  notif.classList.remove('show');
  setTimeout(() => notif.remove(), 500);
}, duration);
};

export const Confirmation = {
  show(message, callback) {
    const confirmBox = document.createElement('div');
    confirmBox.classList.add('custom-confirm-overlay');
    confirmBox.innerHTML = `
      <div class="custom-confirm-box">
        <div class="custom-confirm-message">${message}</div>
        <div class="custom-confirm-buttons">
          <button class="btn btn-success btn-sm confirm-yes">Yes</button>
          <button class="btn btn-secondary btn-sm confirm-no">No</button>
        </div>
      </div>
    `;
    document.body.appendChild(confirmBox);

    const cleanup = () => {
      confirmBox.remove();
    };

    confirmBox.querySelector('.confirm-yes').addEventListener('click', () => {
      callback(true);
      cleanup();
    });

    confirmBox.querySelector('.confirm-no').addEventListener('click', () => {
      callback(false);
      cleanup();
    });
  }
};

```

PortalBubble.js

```

export const PortalBubble = {
  trigger() {
    const bubbleCount = 100;

    for (let i = 0; i < bubbleCount; i++) {
      const bubble = document.createElement('div');

```

```

bubble.classList.add('bubble');

// Random start position across the screen
const startX = Math.random() * window.innerWidth;
const startY = Math.random() * window.innerHeight;
bubble.style.left = `${startX}px`;
bubble.style.top = `${startY}px`;

// Random size and opacity
const size = 10 + Math.random() * 40;
bubble.style.width = `${size}px`;
bubble.style.height = `${size}px`;
bubble.style.opacity = Math.random() * 0.6 + 0.3;

// Generate random drift direction
const driftX = (Math.random() - 0.5) * 200; // -100 to 100
const driftY = (Math.random() - 0.5) * 200;

// Set final position using CSS variables
bubble.style.setProperty('--drift-x', `${driftX}px`);
bubble.style.setProperty('--drift-y', `${driftY}px`);

document.body.appendChild(bubble);

// Activate after slight delay
setTimeout(() => bubble.classList.add('active'), Math.random() * 200);

// Remove after animation
setTimeout(() => bubble.remove(), 4000);
}

};

};


```

PortalLight.js

```

export const PortalLight = {
  render() {
    if (!document.getElementById('portal-light')) {
      const div = document.createElement('div');
      div.id = 'portal-light';
      document.body.appendChild(div);
    }
  },
  trigger() {
    const light = document.getElementById('portal-light');
    if (light) {
      light.classList.add('active');

      setTimeout(() => {
        light.classList.remove('active');
      }, 5000); // Matches the CSS animation duration
    }
  }
};


```

```
}

};

export const PortalFade = {
  render() {
    if (!document.getElementById('portal-fade')) {
      const div = document.createElement('div');
      div.id = 'portal-fade';
      document.body.appendChild(div);
    }
  },
  trigger() {
    if (localStorage.getItem('portalFadeShown') === 'true') return;

    localStorage.setItem('portalFadeShown', 'true');

    const fade = document.getElementById('portal-fade');
    if (fade) {
      fade.classList.add('active');

      setTimeout(() => {
        fade.classList.remove('active');
      }, 5000);
    }
  },
  reset() {
    // Optional: Use this if you want to re-enable the effect manually (for testing/debug)
    localStorage.removeItem('portalFadeShown');
  }
};
```

portalLoader.js

```
export const PortalLoader = {
  containerId: 'portalLoaderContainer',

  render(containerId = 'portalLoaderContainer') {
    this.containerId = containerId;
    let container = document.getElementById(containerId);

    if (!container) {
      container = document.createElement('div');
      container.id = containerId;
      container.className = 'portal-loader-overlay';

      container.innerHTML = `
        <div class="portal-loader">
          <div class="ring"></div>
          <div class="ring2"></div>
          <div class="glow"></div>
          <h2>Processing...</h2>
        </div>
      `;
    }
  }
};
```

```

</div>
';

document.body.appendChild(container);
}

},

show() {
const overlay = document.getElementById(this.containerId);
if (overlay) overlay.classList.add('show');
},

hide() {
const overlay = document.getElementById(this.containerId);
if (overlay) overlay.classList.remove('show');
}
};


```

Rain.js

```

export const Rain = {
trigger(dropCount = 100, duration = null) {
// Rain Overlay
this.overlay = document.createElement('div');
this.overlay.classList.add('rain-overlay');
document.body.appendChild(this.overlay);

// Rain Container
this.rainContainer = document.createElement('div');
this.rainContainer.classList.add('rain-container');
document.body.appendChild(this.rainContainer);

// Generate Raindrops
for (let i = 0; i < dropCount; i++) {
const drop = document.createElement('div');
drop.classList.add('raindrop');

drop.style.left = `${Math.random() * 100}vw`;
const delay = Math.random() * 0.5;
const dropDuration = 0.6 + Math.random() * 0.5;
const length = 10 + Math.random() * 20;

drop.style.animationDuration = `${dropDuration}s`;
drop.style.animationDelay = `${delay}s`;
drop.style.height = `${length}px`;

this.rainContainer.appendChild(drop);
}

// ⏳ If duration is provided, auto clear rain after that time
if (duration) {
setTimeout(() => {

```

```

        this.clear();
    }, duration);
}
},
};

clear() {
if (this.rainContainer) this.rainContainer.classList.add('fade-out');
if (this.overlay) this.overlay.classList.add('fade-out');

// Remove elements after fade-out
setTimeout(() => {
    if (this.rainContainer) this.rainContainer.remove();
    if (this.overlay) this.overlay.remove();
    this.rainContainer = null;
    this.overlay = null;
}, 500);
}
};

sidebar.js

import { Users } from '../data/cache/user-data.js'

export const Sidebar = {
    async render(containerId = 'sidebarContainer') {
        const container = document.getElementById(containerId);
        if (!container) {
            console.warn(`Sidebar container with ID "${containerId}" not found.`);
            return;
        }

        const userRole = localStorage.getItem('userRole');
        const isAdmin = userRole === 'admin';

        const currentUserId = localStorage.getItem('wrusUserId');

        const usersMap = await Users.getUsersMap();
        const currentUserName = usersMap[currentUserId] || "Unknown User";

        container.innerHTML =
`<!-- Mobile Navbar -->
<nav class="navbar bg-light d-md-none">
<div class="container-fluid">
    <button class="btn btn-3d water-btn" type="button" data-bs-toggle="offcanvas" data-bs-target="#sidebar" aria-controls="sidebar">
        <i class="bi bi-list"></i> Menu
    </button>
</div>
</nav>

<!-- Sidebar -->
<div id="sidebar"
    class="offcanvas-md offcanvas-start bg-primary text-white p-3 vh-100"

```

```
tabindex="-1"
aria-labelledby="sidebarLabel">

<div class="offcanvas-header d-md-none">
<div class="logo-container">
<h4 class="wrus-water-title m-0">WRUS Portal</h4>
</div>
</div>

<div class="offcanvas-body d-flex flex-column">
<div class="d-flex justify-content-between align-items-center mb-3">
<div class="logo-container d-none d-md-flex align-items-center gap-2">
<h4 class="wrus-water-title m-0">WRUS Portal</h4>
</div>
<button id="sidebarToggle" class="btn btn-3d water-btn-outline d-none d-md-block">
<i class="bi bi-chevron-left"></i>
</button>
</div>
<div class="mb-3 text-white fw-bold">
<i class="bi bi-person-circle me-2"></i> ${currentUserName}
</div>

<ul class="nav flex-column mb-auto">
<li class="nav-item mb-2">
<a href="${isAdmin ? 'admin-dashboard.html' : 'dashboard.html'}" class="nav-link text-white">
<i class="bi bi-speedometer2 me-2"></i> Dashboard
</a>
</li>
` ; `}
<li class="nav-item mb-2">
<a href="user-management.html" class="nav-link text-white">
<i class="bi bi-people me-2"></i> User Management
</a>
</li>
<li class="nav-item mb-2">
<a class="nav-link text-white" data-bs-toggle="collapse" href="#assetMenu" role="button">
<i class="bi bi-box-seam me-2"></i> Asset Inventory System
</a>
<div class="collapse ps-3" id="assetMenu">
<ul class="nav flex-column">
<li><a class="nav-link text-white" href="consumable.html">Consumable</a></li>
<li><a class="nav-link text-white" href="ics.html">ICS</a></li>
<li><a class="nav-link text-white" href="inventory-summary.html">Ledger/ICS Summary</a></li>
</ul>
</div>
</li>
<li class="nav-item mb-2">
<a class="nav-link text-white" data-bs-toggle="collapse" href="#waterMenu" role="button">
```

```

<i class="bi bi-droplet-half me-2"></i> Water Inventory System
</a>
<div class="collapse ps-3" id="waterMenu">
<ul class="nav flex-column">
<li><a class="nav-link text-white" href="permit.html">Water Permit</a></li>
<li><a class="nav-link text-white" href="water-user.html">Water Users and Sources</a></li>
<li><a class="nav-link text-white" href="water-inventory.html">Mobile Inventory Form</a></li>
<li><a class="nav-link text-white" href="map.html">Map Routing</a></li>
${isAdmin ? `

<li class="nav-item mb-2">
<a href="excel.html" class="nav-link text-white">MEMIS to Database
</a>
</li>
` : `}
</ul>
</div>
</li>
</ul>

<button id="logoutBtn" class="btn btn-3d water-btn-outline mt-auto">
<i class="bi bi-box-arrow-right me-2"></i> Logout
</button>
</div>
</div>
`;

// Initialize Offcanvas
this.initializeOffcanvas();

// Logout Functionality
const logoutBtn = container.querySelector('#logoutBtn');
if (logoutBtn) {
  logoutBtn.addEventListener('click', () => {
    const keysToRemove = [
      'userRole',
      'loggedInUser',
      'wrusUserId',
      'userFullName',
      'userType',
      'portalFadeShown'
    ];
    keysToRemove.forEach(key => localStorage.removeItem(key));
    window.location.href = 'index.html';
  });
}

// Toggle Sidebar (Desktop)
const sidebarToggle = container.querySelector('#sidebarToggle');
if (sidebarToggle) {
  sidebarToggle.addEventListener('click', () => {
    const wrapper = document.getElementById('wrapper');
    wrapper.classList.toggle('sidebar-collapsed');
  });
}

```

```
const icon = sidebarToggle.querySelector('i');
if (icon.classList.contains('bi-chevron-left')) {
  icon.classList.remove('bi-chevron-left');
  icon.classList.add('bi-chevron-right');
} else {
  icon.classList.remove('bi-chevron-right');
  icon.classList.add('bi-chevron-left');
}
});

},
};

initializeOffcanvas() {
  if (typeof bootstrap !== 'undefined' && bootstrap.Offcanvas) {
    const sidebarElement = document.getElementById('sidebar');
    if (sidebarElement) {
      bootstrap.Offcanvas.getOrCreateInstance(sidebarElement);
    }
  }
}
};


```

spinner.js

```
export const Spinner = {
  containerId: 'spinnerContainer',

  render(containerId = 'spinnerContainer', rippleCount = 3) {
    this.containerId = containerId;
    let container = document.getElementById(containerId);
    if (!container) {
      container = document.createElement('div');
      container.id = containerId;
      document.body.appendChild(container);
    }

    // Generate ripple spans dynamically
    const ripples = Array.from({ length: rippleCount })
      .map(() => `<span class="ripple"></span>`)
      .join('');

    container.innerHTML =
      `<div id="loadingOverlay">
        <div class="water-effect">
          ${ripples}
          
        </div>
      </div>`;
  }

  this.applyRippleDelays(rippleCount);
},
```

```
applyRippleDelays(rippleCount) {
  const rippleElements = document.querySelectorAll(`#${this.containerId} .ripple`);
  rippleElements.forEach((el, index) => {
    el.style.animationDelay = `${index * 0.8}s`; // Delay each ripple by 0.8s
  });
}

show() {
  const overlay = document.getElementById('loadingOverlay');
  if (overlay) overlay.style.display = 'flex';
}

hide() {
  const overlay = document.getElementById('loadingOverlay');
  if (overlay) overlay.style.display = 'none';
}
};
```

XV.C.3. Data Scripts

consumable-data.js

```
import { db } from "../../firebaseConfig.js";
import { Ledger } from "./ledger-data.js";
import {
  collection,
  query,
  getDocs,
  updateDoc,
  doc,
  orderBy,
  setDoc,
  where,
  getDoc
} from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio-firestore.js";

export const Consumable = {
  collectionRef: collection(db, "consumable"),
  localStorageKey: "cachedConsumables",

  async generateID() {
    const year = new Date().getFullYear();
    const prefix = `${year}-`;

    const q = query(
      this.collectionRef,
      where("__name__", ">=", prefix),
      where("__name__", "<", `${year + 1}-`)
    );
  }

  const snapshot = await getDocs(q);
  const count = snapshot.size + 1;
}
```

```

const padded = String(count).padStart(3, "0");
return `${year}-${padded}`;
},

async add(spec, qty, unit, addedBy, remarks) {
  const id = await this.generateID();

  const newItem = {
    specification: spec,
    qty: Number(qty),
    unit,
    addedBy,
    timestamp: new Date(),
  };

  const docRef = doc(this.collectionRef, id);
  await setDoc(docRef, newItem);

  await Ledger.addEntry({
    cid: id,
    modifiedBy: addedBy,
    amount: Number(qty),
    remarks: remarks || "Opening stock",
    action: "Add Stock",
  });

  // Append to cache instead of invalidating
  const cached = localStorage.getItem(this.localStorageKey);
  const data = cached ? JSON.parse(cached) : [];
  data.unshift({
    id,
    ...newItem,
    timestamp: newItem.timestamp.toLocaleString(),
  });
  localStorage.setItem(this.localStorageKey, JSON.stringify(data));

  return id;
},

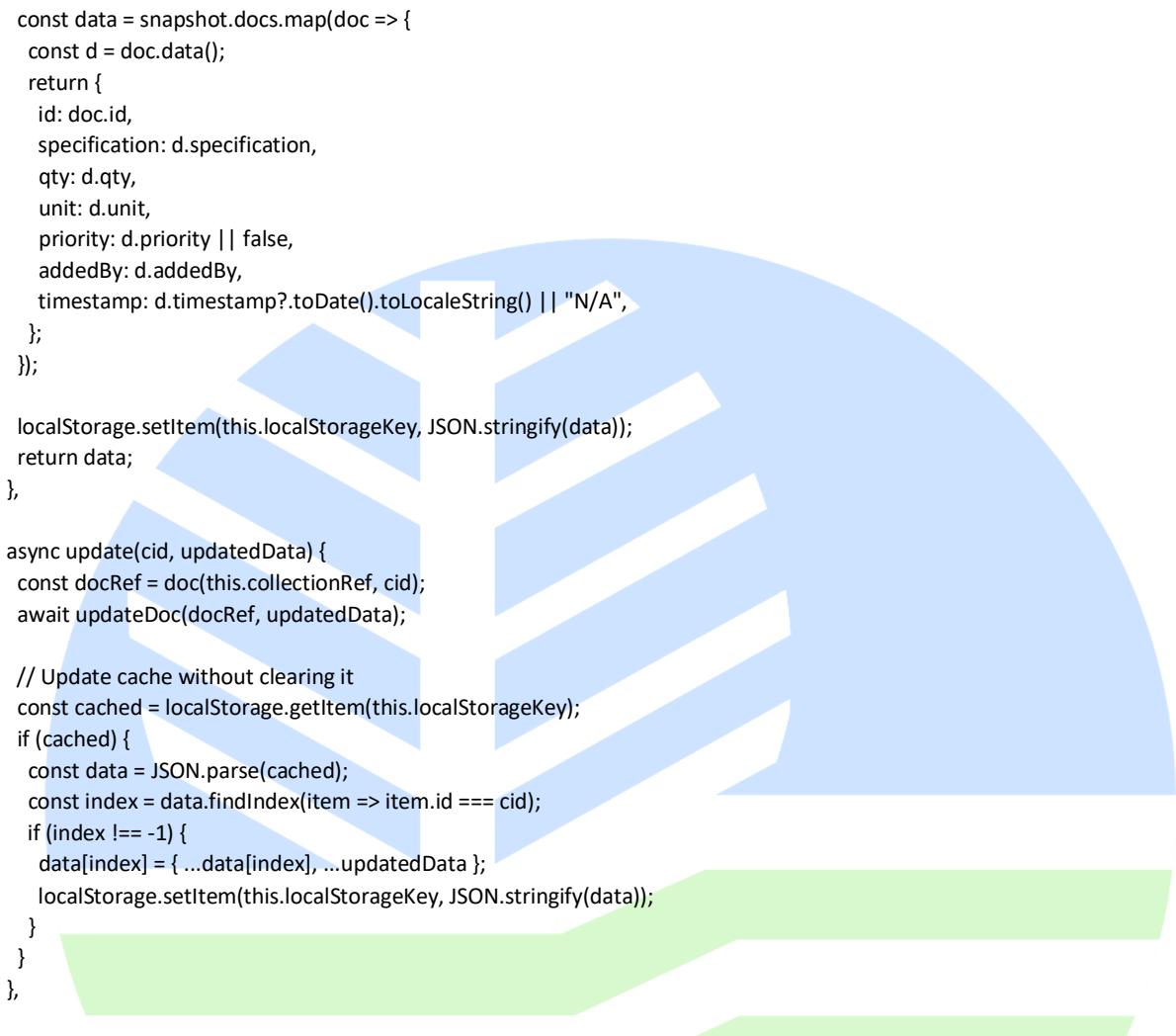
async isSpecDuplicate(spec) {
  const cached = localStorage.getItem(this.localStorageKey);
  const consumables = cached ? JSON.parse(cached) : await this.fetchAll();

  const specLower = spec.toLowerCase();
  return consumables.some(item => item.specification.toLowerCase() === specLower);
},

async fetchAll() {
  const cached = localStorage.getItem(this.localStorageKey);
  if (cached) return JSON.parse(cached);

  const q = query(this.collectionRef, orderBy("timestamp", "desc"));
  const snapshot = await getDocs(q);
}

```



```
const data = snapshot.docs.map(doc => {
  const d = doc.data();
  return {
    id: doc.id,
    specification: d.specification,
    qty: d.qty,
    unit: d.unit,
    priority: d.priority || false,
    addedBy: d.addedBy,
    timestamp: d.timestamp?.toDate().toLocaleString() || "N/A",
  };
});

localStorage.setItem(this.localStorageKey, JSON.stringify(data));
return data;
},

async update(cid, updatedData) {
  const docRef = doc(this.collectionRef, cid);
  await updateDoc(docRef, updatedData);

  // Update cache without clearing it
  const cached = localStorage.getItem(this.localStorageKey);
  if (cached) {
    const data = JSON.parse(cached);
    const index = data.findIndex(item => item.id === cid);
    if (index !== -1) {
      data[index] = { ...data[index], ...updatedData };
      localStorage.setItem(this.localStorageKey, JSON.stringify(data));
    }
  }
}

async addStock(cid, amount, remarks = "") {
  const docRef = doc(this.collectionRef, cid);
  const snapshot = await getDoc(docRef);

  if (!snapshot.exists()) throw new Error("Item not found");

  const newQty = (snapshot.data().qty || 0) + amount;
  await updateDoc(docRef, { qty: newQty });

  await Ledger.addEntry({
    cid,
    modifiedBy: localStorage.getItem("userFullName") || "Unknown",
    amount,
    remarks,
    action: "Add Stock",
  });
}

// Update cache with new qty
const cached = localStorage.getItem(this.localStorageKey);
```

```

if (cached) {
  const data = JSON.parse(cached);
  const index = data.findIndex(item => item.id === cid);
  if (index !== -1) {
    data[index].qty = newQty;
    localStorage.setItem(this.localStorageKey, JSON.stringify(data));
  }
},
};

async assignItem(cid, amount, assignedTo, remarks = "") {
  const docRef = doc(this.collectionRef, cid);
  const snapshot = await getDoc(docRef);

  if (!snapshot.exists()) throw new Error("Item not found");

  const currentQty = snapshot.data().qty || 0;
  if (amount > currentQty) {
    throw new Error(`Cannot assign more than available quantity (${currentQty}).`);
  }

  const newQty = currentQty - amount;
  await updateDoc(docRef, { qty: newQty });

  await Ledger.addEntry({
    cid,
    modifiedBy: localStorage.getItem("userFullName") || "Unknown",
    amount,
    remarks,
    action: "Assign Item",
    assignedTo,
  });
}

// Update cache with new qty
const cached = localStorage.getItem(this.localStorageKey);
if (cached) {
  const data = JSON.parse(cached);
  const index = data.findIndex(item => item.id === cid);
  if (index !== -1) {
    data[index].qty = newQty;
    localStorage.setItem(this.localStorageKey, JSON.stringify(data));
  }
}
};

async getQty(cid) {
  const docRef = doc(this.collectionRef, cid);
  const snapshot = await getDoc(docRef);

  if (!snapshot.exists()) return null;

  const data = snapshot.data();
  return data.qty ?? 0;
}

```

```
},  
  
async fetchConsumablesMap() {  
  const cached = localStorage.getItem(this.localStorageKey);  
  const items = cached ? JSON.parse(cached) : await this.fetchAll();  
  
  const map = {};  
  items.forEach(item => {  
    map[item.id] = {  
      specification: item.specification || "",  
      unit: item.unit || "",  
    };  
  });  
  
  return map;  
}  
  
async refreshCache() {  
  const q = query(this.collectionRef, orderBy("timestamp", "desc"));  
  const snapshot = await getDocs(q);  
  
  const data = snapshot.docs.map(doc => {  
    const d = doc.data();  
    return {  
      id: doc.id,  
      specification: d.specification,  
      qty: d.qty,  
      unit: d.unit,  
      priority: d.priority || false,  
      addedBy: d.addedBy,  
      timestamp: d.timestamp?.toDate().toLocaleString() || "N/A",  
    };  
  });  
  
  localStorage.setItem(this.localStorageKey, JSON.stringify(data));  
  return data;  
}  
  
async autoRefreshDaily() {  
  const key = this.localStorageKey;  
  const dateKey = `${key}_lastRefreshDate`;  
  const today = new Date().toISOString().split("T")[0];  
  
  const lastRefresh = localStorage.getItem(dateKey);  
  if (lastRefresh !== today) {  
    await this.refreshCache();  
    localStorage.setItem(dateKey, today);  
    console.log("[Consumable] Cache auto-refreshed for the day.");  
  }  
}  
  
async autoRefreshEvery8Hours() {  
  const key = "cachedConsumable_lastRefresh";
```

```

const now = Date.now();
const last = localStorage.getItem(key);

if (!last || now - parseInt(last, 10) > 8 * 60 * 60 * 1000) {
  await this.refreshCache();
  localStorage.setItem(key, now.toString());
  console.log("[Consumable] Cache refreshed (8-hour interval).");
}
};

}
);

```

ics-data.js

```

import {
  collection,
  getDoc,
  getDocs,
  doc,
  addDoc,
  updateDoc,
  serverTimestamp,
  query,
  orderBy,
  deleteDoc,
  where
} from "https://www.gstatic.com/firebasejs/10.11.1/firebase-firestore.js";
import { db } from "../..../firebaseConfig.js";
import { NotificationBox } from "../../components/notification.js";

export const ICS = {
  collectionRef: collection(db, "ICS"),
  localStorageKey: "cachedICS",

  // 🔔 Fetch all ICS entries (from cache first)
  async fetchAll() {
    const cached = localStorage.getItem(this.localStorageKey);
    if (cached) {
      const parsed = JSON.parse(cached);
      return parsed.sort((a, b) => b.data.timestamp?.seconds - a.data.timestamp?.seconds);
    }

    return await this.refreshCache();
  },

  // 🔔 Add a new ICS entry (and append to cache)
  async add(icsData) {
    try {
      if (!icsData.ICSno || typeof icsData.ICSno !== 'string' || icsData.ICSno.trim() === '') {
        NotificationBox.show('ICS No. is required.');
        return;
      }

      const dataToSave = {
        ICSno: icsData.ICSno,
        name: icsData.name,
        address: icsData.address,
        city: icsData.city,
        state: icsData.state,
        zip: icsData.zip,
        phone: icsData.phone,
        email: icsData.email,
        website: icsData.website,
        description: icsData.description,
        created_at: serverTimestamp(),
        updated_at: serverTimestamp()
      };
    }
  }
};

```



```
...icsData,
timestamp: serverTimestamp(),
};

const docRef = await addDoc(this.collectionRef, dataToSave);
const newDoc = await getDoc(docRef);
const newEntry = {
id: newDoc.id,
data: newDoc.data()
};

// Append to cache
const cached = localStorage.getItem(this.localStorageKey);
const entries = cached ? JSON.parse(cached) : [];
entries.push(newEntry);
localStorage.setItem(this.localStorageKey, JSON.stringify(entries));
NotificationBox.show('ICS entry saved successfully!');
} catch (err) {
console.error("Error adding ICS entry:", err);
NotificationBox.show('Error saving ICS entry. Check console for details.');
}
},

// 🔺 Update an existing ICS entry (and patch cache)
async update(docId, updatedData) {
const docRef = doc(this.collectionRef, docId);
await updateDoc(docRef, updatedData);

const cached = localStorage.getItem(this.localStorageKey);
if (cached) {
const entries = JSON.parse(cached);
const index = entries.findIndex(item => item.id === docId);

if (index !== -1) {
entries[index].data = {
...entries[index].data,
...updatedData
};
localStorage.setItem(this.localStorageKey, JSON.stringify(entries));
}
}
}

// 🔻 Delete an ICS entry (and remove from cache)
async delete(docId) {
try {
await deleteDoc(doc(this.collectionRef, docId));

const cached = localStorage.getItem(this.localStorageKey);
if (cached) {
const entries = JSON.parse(cached).filter(item => item.id !== docId);
localStorage.setItem(this.localStorageKey, JSON.stringify(entries));
}
}
}
```

```
        } catch (error) {
            console.error("Error deleting ICS document:", error);
            throw error;
        }
    },

// 🔺 Force re-fetch from Firestore and update cache
async refreshCache() {
    const q = query(this.collectionRef, orderBy("timestamp", "desc"));
    const snapshot = await getDocs(q);

    const data = snapshot.docs.map(doc => ({
        id: doc.id,
        data: doc.data(),
    }));

    localStorage.setItem(this.localStorageKey, JSON.stringify(data));
    return data;
}

// 🔺 Fetch ICS assigned to a specific user
async getICSDataByUserId(userId) {
    const q = query(
        this.collectionRef,
        where("assignedTo", "==", userId),
        orderBy("timestamp", "asc")
    );

    const querySnapshot = await getDocs(q);
    const items = [];

    querySnapshot.forEach((doc) => {
        const data = doc.data();
        items.push({
            qty: data.qty || 0,
            unit: data.unit || "",
            description: data.description || "",
            serialNo: data.serialNo || "",
            unitCost: parseFloat(data.unitCost) || 0,
            totalCost: parseFloat(data.totalCost) || 0,
            ICSno: data.ICSno || "",
            dateIssued: data.dateIssued || "",
            remarks: data.remarks || "",
            attachmentURL: data.attachmentURL || "",
            status: data.status || "",
            timestamp: data.timestamp || null
        });
    });

    return items;
}

async autoRefreshDaily() {
```

```
const key = this.localStorageKey;
const dateKey = `${key}_lastRefreshDate`;
const today = new Date().toISOString().split("T")[0];

const lastRefresh = localStorage.getItem(dateKey);

if (lastRefresh !== today) {
  await this.refreshCache();
  localStorage.setItem(dateKey, today);
  console.log("[ICS] Cache auto-refreshed for the day.");
}

};

async autoRefreshEvery8Hours() {
  const key = 'cachedICS_lastRefresh';
  const now = Date.now();
  const last = localStorage.getItem(key);

  if (!last || now - parseInt(last, 10) > 8 * 60 * 60 * 1000) {
    await this.refreshCache();
    localStorage.setItem(key, now.toString());
    console.log("[ICS] Cache refreshed (8-hour interval).");
  }
}
};
```

ledger-data.js

```
import { db } from "../../firebaseConfig.js";
import { Consumable } from "./consumable-data.js";
import {
  collection,
  getDocs,
  getDoc,
  doc,
  addDoc,
  serverTimestamp,
  query,
  where,
  orderBy,
  deleteDoc
} from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio.js";

export const Ledger = {
  collectionRef: collection(db, "ledger"),
  localCacheKey: "ledgerCache",

  async fetchAll() {
    const cached = localStorage.getItem(this.localCacheKey);
    if (cached) {
      try {
        return JSON.parse(cached);
      } catch (error) {
        console.error(`Error parsing cached ledger data: ${error}`);
      }
    }
  }
};
```

```
        } catch (e) {
          console.warn("Failed to parse cached ledger data:", e);
          localStorage.removeItem(this.localCacheKey);
        }
      }

    try {
      const snapshot = await getDocs(query(this.collectionRef, orderBy("dateModified", "desc")));
      const entries = [];

      snapshot.forEach(docSnap => {
        entries.push({ id: docSnap.id, ...docSnap.data() });
      });

      localStorage.setItem(this.localCacheKey, JSON.stringify(entries));
    }

    return entries;
  } catch (error) {
    console.error("Error fetching all ledger entries:", error);
    return [];
  }
}

clearCache() {
  localStorage.removeItem(this.localCacheKey);
}

async addEntry({ cid, modifiedBy, amount, remarks, action, assignedTo = "" }) {
  const entry = {
    cid,
    modifiedBy,
    amount: Number(amount),
    remarks: remarks.trim() || "",
    action,
    assignedTo,
    dateModified: serverTimestamp(),
  };

  await addDoc(this.collectionRef, entry);
}

async getEntriesByCID(cid) {
  const q = query(
    this.collectionRef,
    where("cid", "==", cid),
    orderBy("dateModified", "desc")
  );

  const ledgerSnapshot = await getDocs(q);

  const ledgerEntries = [];
  const userIds = new Set();

```

```

ledgerSnapshot.forEach(docSnap => {
  const data = docSnap.data();
  ledgerEntries.push({ id: docSnap.id, ...data });
  if (data.assignedTo) {
    userIds.add(data.assignedTo);
  }
});

return {
  ledgerEntries,
  userIds: Array.from(userIds),
};
},

async fetchLedgerDataByCID(selectedCID) {
  if (!selectedCID || typeof selectedCID !== "string" || selectedCID.trim() === "") {
    console.error("Invalid selectedCID:", selectedCID);
    totalQtyDisplay.textContent = "Invalid CID";
    return { totalQty: 0, ledgerEntries: [] };
  }

  try {
    const totalQty = await Consumable.getQty(selectedCID);
    totalQtyDisplay.textContent = (totalQty !== null) ? totalQty : "Not found";

    const { ledgerEntries, userIds } = await Ledger.getEntriesByCID(selectedCID);

    const userMap = {};
    await Promise.all(userIds.map(async userId => {
      try {
        const userDoc = await getDoc(db, "users", userId);
        if (userDoc.exists()) {
          const u = userDoc.data();
          userMap[userId] = `${u.lastName}, ${u.firstName} ${u.middleInitial}.`;
        } else {
          userMap[userId] = "Unknown User";
        }
      } catch {
        userMap[userId] = "Error";
      }
    }));
  }
}

// ✅ Enrich ledger with user names
const enrichedEntries = ledgerEntries.map(entry => ({
  ...entry,
  assignedTo: entry.assignedTo ? (userMap[entry.assignedTo] || "-") : "-"
}));

return { totalQty: totalQty ?? 0, ledgerEntries: enrichedEntries };

} catch (error) {
  console.error("Failed to load ledger or total qty:", error);
  totalQtyDisplay.textContent = "Error";
}

```

```
        return { totalQty: 0, ledgerEntries: [] };
    }
},
```

```
async fetchLedgerByUser(userId) {
    const q = query(
        collection(db, "ledger"),
        where("assignedTo", "==", userId)
    );
    const snapshot = await getDocs(q);
    const entries = [];
    snapshot.forEach((doc) => {
        const d = doc.data();
        entries.push({
            cid: d.cid,
            amount: d.amount || 0,
            dateModified: d.dateModified || null,
            remarks: d.remarks || '-'
        });
    });
    return entries;
},
```

```
async deleteEntry(entryId) {
    const docRef = doc(collection(db, "ledger"), entryId);
    await deleteDoc(docRef);
},
```

```
async refreshCache() {
    try {
        const snapshot = await getDocs(query(this.collectionRef, orderBy("dateModified", "desc")));
        const entries = [];

        snapshot.forEach(docSnap => {
            entries.push({ id: docSnap.id, ...docSnap.data() });
        });

        localStorage.setItem(this.localCacheKey, JSON.stringify(entries));
        console.log("Ledger cache refreshed.");
        return entries;
    } catch (error) {
        console.error("Failed to refresh ledger cache:", error);
        return [];
    }
},
```

```
async autoRefreshEvery8Hours() {
    const key = "cachedLedger_lastRefresh";
    const now = Date.now();
    const last = localStorage.getItem(key);

    if (!last || now - parseInt(last, 10) > 8 * 60 * 60 * 1000) {
        await this.refreshCache();
    }
}
```

```
localStorage.setItem(key, now.toString());
console.log("[Ledger] Cache refreshed (8-hour interval).");
}
}

};


```

permit-data.js

```
import { db } from '../../firebaseConfig.js';
import {
  collection,
  query,
  where,
  getDoc,
  getDocs,
  orderBy,
  limit,
  doc,
  setDoc,
  serverTimestamp,
  deleteDoc
} from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio.js";

const permitCollection = collection(db, 'permits');

export const Permit = {
  localStorageKey: "cachedPermits",
  async add(data) {
    try {
      const permitNoToCheck = data.permitNo.trim();

      const duplicateQuery = query(
        permitCollection,
        where('permitNo', '==', permitNoToCheck)
      );
      const duplicateSnapshot = await getDocs(duplicateQuery);

      if (!duplicateSnapshot.empty) {
        throw new Error(`Permit No "${permitNoToCheck}" already exists.`);
      }

      const docRef = doc(permitCollection);
      await setDoc(docRef, {
        ...data,
        createdAt: serverTimestamp(),
      });

      const savedDoc = await getDoc(docRef);
      if (!savedDoc.exists()) {
        throw new Error("Failed to fetch the newly added permit");
      }
    }
  }
};


```

```
}

const savedData = { id: savedDoc.id, ...savedDoc.data() };

const cached = localStorage.getItem(this.localStorageKey);
const parsed = cached ? JSON.parse(cached) : [];

const updatedCache = parsed.filter(item => item.id !== savedData.id);

updatedCache.unshift(savedData);

updatedCache.sort((a, b) => {
  const aDate = a.createdAt?.toDate() ? a.createdAt.toDate() : new Date(a.createdAt);
  const bDate = b.createdAt?.toDate() ? b.createdAt.toDate() : new Date(b.createdAt);
  return bDate - aDate;
});

localStorage.setItem(this.localStorageKey, JSON.stringify(updatedCache));

} catch (error) {
  console.error('Error adding permit:', error.message);
  throw error;
}
},
async getAll(forceRefresh = false) {
  if (!forceRefresh) {
    const cached = localStorage.getItem(this.localStorageKey);
    if (cached) {
      try {
        return JSON.parse(cached);
      } catch (e) {
        console.warn('Cache corrupted, falling back to Firestore');
      }
    }
  }
  try {
    const snapshot = await getDocs(permitCollection);
    let permits = snapshot.docs.map(doc => ({
      id: doc.id,
      ...doc.data(),
    }));
    permits.sort((a, b) => {
      const aDate = a.createdAt?.toDate() ? a.createdAt.toDate() : new Date(a.createdAt);
      const bDate = b.createdAt?.toDate() ? b.createdAt.toDate() : new Date(b.createdAt);
      return bDate - aDate;
    });
    localStorage.setItem(this.localStorageKey, JSON.stringify(permits));
  }
  return permits;
} catch (error) {
```

```
        console.error('Error fetching permits from Firestore:', error.message);
        throw error;
    },
},
async refreshCache() {
    try {
        const snapshot = await getDocs(permitCollection);
        let permits = snapshot.docs.map(doc => ({
            id: doc.id,
            ...doc.data(),
        }));
        permits.sort((a, b) => {
            const aDate = a.createdAt?.toDate() ? a.createdAt.toDate() : new Date(a.createdAt);
            const bDate = b.createdAt?.toDate() ? b.createdAt.toDate() : new Date(b.createdAt);
            return bDate - aDate;
        });
        localStorage.setItem(this.localStorageKey, JSON.stringify(permits));
        console.log(' Local cache refreshed successfully');
    } catch (error) {
        console.error('Error refreshing cache:', error.message);
        throw error;
    }
},
async update(id, data) {
    try {
        const docRef = doc(permitCollection, id);
        await setDoc(docRef, {
            ...data,
            updatedAt: serverTimestamp(),
        }, { merge: true });
        // ✅ Update cache
        const cached = localStorage.getItem(this.localStorageKey);
        if (cached) {
            const parsed = JSON.parse(cached);
            const updated = parsed.map(p => p.id === id ? { ...p, ...data, updatedAt: new Date() } : p);
            localStorage.setItem(this.localStorageKey, JSON.stringify(updated));
        }
    } catch (error) {
        console.error('Error updating permit:', error.message);
        throw error;
    }
},
async delete(id) {
    try {
        await deleteDoc(doc(permitCollection, id));
    }
```

```
const cached = localStorage.getItem(this.localStorageKey);
if (cached) {
  const parsed = JSON.parse(cached);
  const filtered = parsed.filter(p => p.id !== id);
  localStorage.setItem(this.localStorageKey, JSON.stringify(filtered));
}

console.log(`Permit ${id} deleted successfully`);
} catch (error) {
  throw error;
}
},

async autoRefreshDaily() {
  const key = this.localStorageKey;
  const dateKey = `${key}_lastRefreshDate`;
  const today = new Date().toISOString().split("T")[0];

  const lastRefresh = localStorage.getItem(dateKey);

  if (lastRefresh !== today) {
    await this.refreshCache();
    localStorage.setItem(dateKey, today);
    console.log("[Permit] Cache auto-refreshed for the day.");
  }
},

async autoRefreshEvery8Hours() {
  const key = 'cachedPermit_lastRefresh';
  const now = Date.now();
  const last = localStorage.getItem(key);

  if (!last || now - parseInt(last, 10) > 8 * 60 * 60 * 1000) {
    await this.refreshCache();
    localStorage.setItem(key, now.toString());
    console.log("[Permit] Cache refreshed (8-hour interval).");
  }
},

async getById(id) {
  const cached = localStorage.getItem(this.localStorageKey);
  if (cached) {
    const parsed = JSON.parse(cached);
    const found = parsed.find(p => p.id === id);
    if (found) return found;
  }

  try {
    const docRef = doc(permitCollection, id);
    const snapshot = await getDoc(docRef);
    if (snapshot.exists()) {
      const data = { id: snapshot.id, ...snapshot.data() };

      const parsed = cached ? JSON.parse(cached) : [];

```

```
        parsed.push(data);
        localStorage.setItem(this.localStorageKey, JSON.stringify(parsed));

        return data;
    } else {
        throw new Error(`Permit with ID "${id}" not found.`);
    }
} catch (error) {
    console.error('Error in getById:', error.message);
    throw error;
}
};

};


```

user-data.js

```
import { db } from "../../firebaseConfig.js";
import {
    collection,
    addDoc,
    getDocs,
    doc,
    updateDoc,
    query,
    orderBy,
    serverTimestamp
} from "https://www.gstatic.com/firebasejs/10.11.1/firebase-firestore.js";

export const Users = {
    collectionRef: collection(db, "users"),
    localStorageKey: "cachedUsers",

    async add(userData) {
        const docRef = await addDoc(this.collectionRef, {
            ...userData,
            timestamp: serverTimestamp()
        });

        const newUserDoc = await getDoc(docRef);
        const newUser = { id: newUserDoc.id, ...newUserDoc.data() };

        // Append to cached data
        const cached = localStorage.getItem(this.localStorageKey);
        const users = cached ? JSON.parse(cached) : [];
        users.push(newUser);
        localStorage.setItem(this.localStorageKey, JSON.stringify(users));
    },

    async update(id, updatedData) {
        const userRef = doc(db, "users", id);
        try {
            await updateDoc(userRef, updatedData);
        
```

```
// Update cache in-place
const cached = localStorage.getItem(this.localStorageKey);
if (cached) {
  const users = JSON.parse(cached);
  const index = users.findIndex(user => user.id === id);
  if (index !== -1) {
    users[index] = { ...users[index], ...updatedData };
    localStorage.setItem(this.localStorageKey, JSON.stringify(users));
  }
}
} catch (error) {
  console.error("Error updating user:", error);
  throw error;
}
},
async fetchAllDesc() {
  const cached = localStorage.getItem(this.localStorageKey);
  if (cached) {
    const data = JSON.parse(cached);
    return data.sort((a, b) => b.timestamp?.seconds - a.timestamp?.seconds);
  }

  const usersQuery = query(this.collectionRef, orderBy("timestamp", "desc"));
  const userSnapshot = await getDocs(usersQuery);
  const users = userSnapshot.docs.map(doc => ({
    id: doc.id,
    ...doc.data()
  }));
}

localStorage.setItem(this.localStorageKey, JSON.stringify(users));
return users;
},
async fetchAllAsc() {
  const cached = localStorage.getItem(this.localStorageKey);
  if (cached) {
    const data = JSON.parse(cached);
    return data.sort((a, b) => a.timestamp?.seconds - b.timestamp?.seconds);
  }

  const usersQuery = query(this.collectionRef, orderBy("timestamp", "asc"));
  const userSnapshot = await getDocs(usersQuery);
  const users = userSnapshot.docs.map(doc => ({
    id: doc.id,
    ...doc.data()
  }));
}

localStorage.setItem(this.localStorageKey, JSON.stringify(users));
return users;
},
async getUsersMap() {
```

```
const cached = localStorage.getItem(this.localStorageKey);
const users = cached ? JSON.parse(cached) : await this.fetchAllAsc();

const usersMap = {};
users.forEach(user => {
  usersMap[user.id] = `${user.lastName} ${user.firstName} ${user.middleInitial || ""}.trim();
});

return usersMap;
},

async fetchUsersSummary() {
  const cached = localStorage.getItem(this.localStorageKey);
  const users = cached ? JSON.parse(cached) : await this.fetchAllAsc();

  return users
    .filter(user => user.id.toLowerCase() !== "admin")
    .map(user => ({
      id: user.id,
      username: user.username || "",
      lastName: user.lastName || "",
      firstName: user.firstName || "",
      middleInitial: user.middleInitial || "",
      type: user.type || "",
      status: user.status || ""
    }));
}

async refreshCache() {
  const usersQuery = query(this.collectionRef, orderBy("timestamp", "desc"));
  const userSnapshot = await getDocs(usersQuery);
  const users = userSnapshot.docs.map(doc => ({
    id: doc.id,
    ...doc.data()
  }));

  localStorage.setItem(this.localStorageKey, JSON.stringify(users));
  return users;
}

async autoRefreshDaily() {
  const now = Date.now();
  const lastFetch = parseInt(localStorage.getItem("lastUsersFetch") || "0");
  const oneDayMs = 24 * 60 * 60 * 1000;

  if (now - lastFetch > oneDayMs) {
    await this.refreshCache();
    localStorage.setItem("lastUsersFetch", now.toString());
    console.log("User cache refreshed automatically (daily).");
  } else {
    console.log("User cache still fresh. No need to auto-refresh.");
  }
},
```

```
async autoRefreshEvery8Hours() {
  const key = 'cachedUser_lastRefresh';
  const now = Date.now();
  const last = localStorage.getItem(key);

  if (!last || now - parseInt(last, 10) > 8 * 60 * 60 * 1000) {
    await this.refreshCache();
    localStorage.setItem(key, now.toString());
    console.log("[User] Cache refreshed (8-hour interval).");
  }
}
};
```

wrus-data.js

```
import { db } from '../firebaseConfig.js';
import {
  collection,
  addDoc,
  getDocs,
  updateDoc,
  deleteDoc,
  doc,
  serverTimestamp
} from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio.js";

const WUSCollection = collection(db, 'water_users');
const CACHE_KEY = 'cachedWUS';

export const WUSData = {
  async fetchAll() {
    const cached = localStorage.getItem(CACHE_KEY);
    const data = cached ? JSON.parse(cached) : await this.refreshCache();

    return [...data].sort((a, b) => (b.timestamp?.seconds || 0) - (a.timestamp?.seconds || 0));
  },
  async refreshCache() {
    const snapshot = await getDocs(WUSCollection);
    const data = snapshot.docs.map(doc => ({
      id: doc.id,
      ...doc.data()
    }));

    localStorage.setItem(CACHE_KEY, JSON.stringify(data));
    return data;
  },
  async add(data) {
    const newData = {
      ...data,
      timestamp: serverTimestamp()
    };
  }
};
```

```
};

const docRef = await addDoc(WUSCollection, newData);
const newId = docRef.id;

const entry = {
  id: newId,
  ...data,
  timestamp: { seconds: Date.now() / 1000 } // fallback for now
};

const existing = JSON.parse(localStorage.getItem(CACHE_KEY) || '[]');
existing.push(entry);
localStorage.setItem(CACHE_KEY, JSON.stringify(existing));

return { id: newId, ...newData };
},

async update(id, data) {
  const docRef = doc(db, 'water_users', id);
  await updateDoc(docRef, { ...data });

  const existing = JSON.parse(localStorage.getItem(CACHE_KEY) || '[]');
  const updated = existing.map(entry =>
    entry.id === id ? { ...entry, ...data } : entry
  );
  localStorage.setItem(CACHE_KEY, JSON.stringify(updated));
},

async delete(id) {
  const docRef = doc(db, 'water_users', id);
  await deleteDoc(docRef);

  const existing = JSON.parse(localStorage.getItem(CACHE_KEY) || '[]');
  const updated = existing.filter(entry => entry.id !== id);
  localStorage.setItem(CACHE_KEY, JSON.stringify(updated));
},

async autoRefreshDaily() {
  const dateKey = 'cachedWUS_lastRefreshDate';
  const today = new Date().toISOString().split("T")[0];

  const lastRefresh = localStorage.getItem(dateKey);
  if (lastRefresh !== today) {
    await this.refreshCache();
    localStorage.setItem(dateKey, today);
    console.log("[WUS] Cache auto-refreshed for the day.");
  }
},

async autoRefreshEvery8Hours() {
  const key = 'cachedWUS_lastRefresh';
  const now = Date.now();
```

```

const last = localStorage.getItem(key);

if (!last || now - parseInt(last, 10) > 8 * 60 * 60 * 1000) {
  await this.refreshCache();
  localStorage.setItem(key, now.toString());
  console.log("[WUS] Cache refreshed (8-hour interval).");
}
};

};


```

metroManilaCities.js

```

export const METRO_MANILA_CITIES = [
  "Caloocan", "Las Piñas", "Makati", "Malabon", "Mandaluyong", "Manila",
  "Marikina", "Muntinlupa", "Navotas", "Parañaque", "Pasay", "Pasig",
  "Quezon City", "San Juan", "Taguig", "Valenzuela", "Pateros"
];

```

months.js

```

export const months = [
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"
];

```

purpose.js

```

export const purpose = [
  "Domestic Use - Household",
  "Domestic Use - Community",
  "Municipal Use",
  "Industrial Use",
  "Commercial Use",
  "Irrigation Use",
  "Power Generation Use",
  "Fisheries Use",
  "Livestock Use",
  "Recreation Use",
  "Bulk Supply",
  "Others"
];

```

sourceStatus.js

```

export const sourceStatus = [
  "OPERATIONAL",
  "NON OPERATIONAL",
  "ON-GOING CONSTRUCTION"
];

```

waterSources.js

```

export const waterSources = [

```

```
"Deep Well",
"Deep Well - Handpump",
"Deep Well - Motor Pump",
"Dug Well / Shallow Well",
"Spring",
"River",
"Stream / Creek",
"Lake",
"Others"
];
```

signature-data.js

```
export class SignatureDB {
  constructor(dbName = "WaterInventoryDB", storeName = "signatures") {
    this.dbName = dbName;
    this.storeName = storeName;
    this.dbPromise = this.#openDB();
  }

  async #openDB() {
    return new Promise((resolve, reject) => {
      const request = indexedDB.open(this.dbName, 1);

      request.onerror = () => reject("Failed to open IndexedDB");

      request.onsuccess = () => resolve(request.result);

      request.onupgradeneeded = (e) => {
        const db = e.target.result;
        if (!db.objectStoreNames.contains(this.storeName)) {
          db.createObjectStore(this.storeName, { keyPath: "id", autoIncrement: true });
        }
      };
    });
  }

  async #getStore(mode = "readonly") {
    const db = await this.dbPromise;
    const tx = db.transaction(this.storeName, mode);
    return tx.objectStore(this.storeName);
  }

  async saveSignature(signatureDataURL, id) {
    const store = await this.#getStore("readwrite");
    const request = store.put({ id, signature: signatureDataURL });

    return new Promise((resolve, reject) => {
      request.onsuccess = () => resolve(true);
      request.onerror = () => reject("Failed to save signature");
    });
  }
}
```

```

async getSignature(id) {
  const store = await this.#getStore("readonly");
  const request = store.get(id);

  return new Promise((resolve, reject) => {
    request.onsuccess = () => resolve(request.result?.signature || null);
    request.onerror = () => reject("Error getting signature");
  });
}

async clearSignatures() {
  const store = await this.#getStore("readwrite");
  const request = store.clear();

  return new Promise((resolve, reject) => {
    request.onsuccess = () => resolve(true);
    request.onerror = () => reject("Failed to clear signatures");
  });
}

async deleteSignature(id) {
  const store = await this.#getStore("readwrite");
  const request = store.delete(id);

  return new Promise((resolve, reject) => {
    request.onsuccess = () => resolve(true);
    request.onerror = () => reject("Failed to delete signature");
  });
}

```

XV.C.4 Module Scripts

XV.C.4.a Consumable Module

consumable-init.js

```

import { SessionGuard } from "../auth/auth.js";
import { initializePage } from "./consumable-ui.js";

document.addEventListener("DOMContentLoaded", () => {
  SessionGuard.ensureLoggedIn();
  initializePage();
});

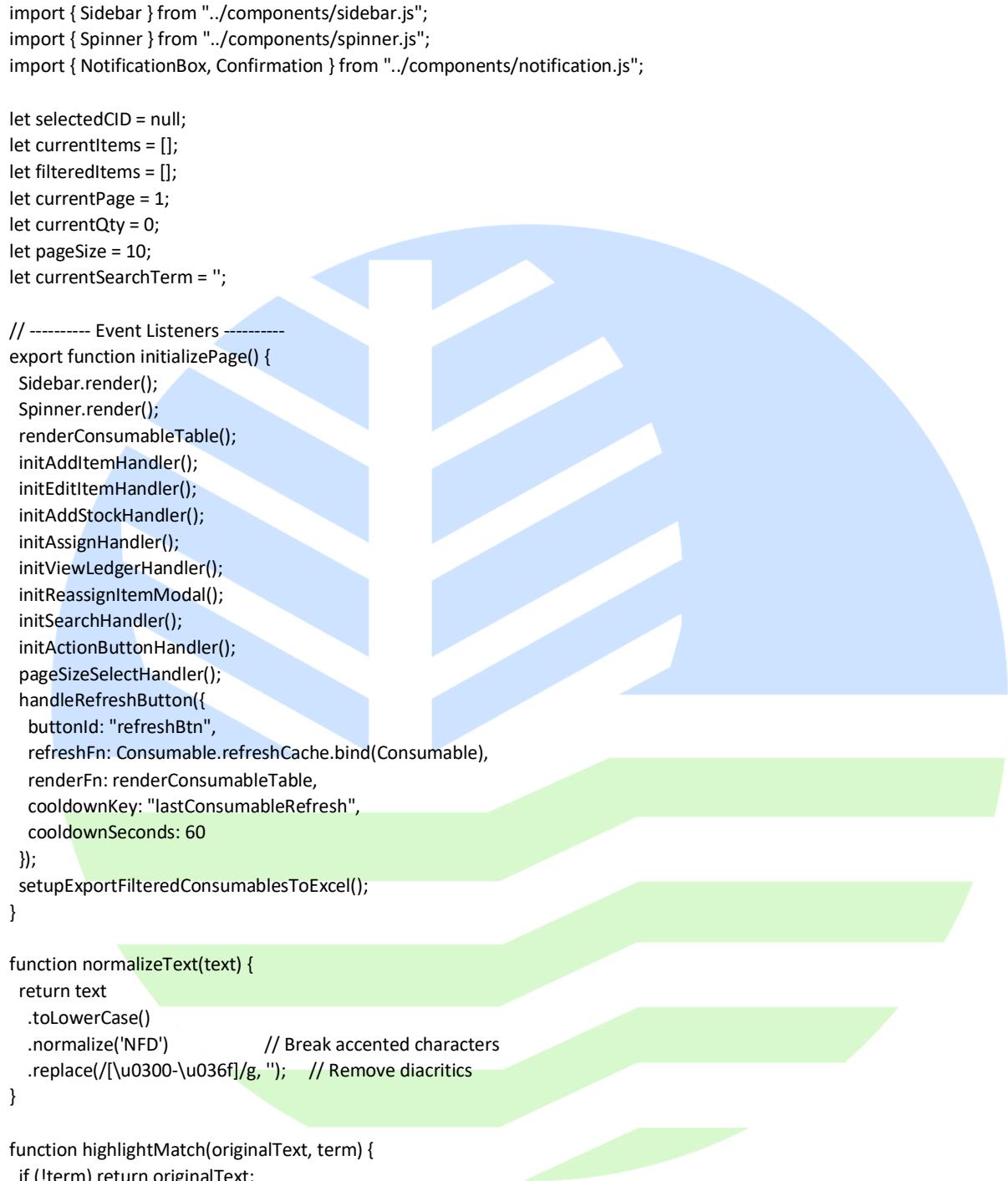
```

consumable-ui.js

```

import { Consumable } from "../data/cache/consumable-data.js";
import { Ledger } from "../data/cache/ledger-data.js"
import { generateLedgerPDFBlob } from './pdf/item-consumable-pdf.js';
import { Users } from "../data/cache/user-data.js";

```



```
import { Sidebar } from "../components/sidebar.js";
import { Spinner } from "../components/spinner.js";
import { NotificationBox, Confirmation } from "../components/notification.js";

let selectedCID = null;
let currentItems = [];
let filteredItems = [];
let currentPage = 1;
let currentQty = 0;
let pageSize = 10;
let currentSearchTerm = "";

// ----- Event Listeners -----
export function initializePage() {
  Sidebar.render();
  Spinner.render();
  renderConsumableTable();
  initAddItemHandler();
  initEditItemHandler();
  initAddStockHandler();
  initAssignHandler();
  initViewLedgerHandler();
  initReassignItemModal();
  initSearchHandler();
  initActionButtonHandler();
  pageSizeSelectHandler();
  handleRefreshButton({
    buttonId: "refreshBtn",
    refreshFn: Consumable.refreshCache.bind(Consumable),
    renderFn: renderConsumableTable,
    cooldownKey: "lastConsumableRefresh",
    cooldownSeconds: 60
  });
  setupExportFilteredConsumablesToExcel();
}

function normalizeText(text) {
  return text
    .toLowerCase()
    .normalize('NFD')           // Break accented characters
    .replace(/[\u0300-\u036f]/g, ""); // Remove diacritics
}

function highlightMatch(originalText, term) {
  if (!term) return originalText;

  const normalizedOriginal = normalizeText(originalText);
  const normalizedTerm = normalizeText(term);

  const index = normalizedOriginal.indexOf(normalizedTerm);
  if (index === -1) return originalText;

  const before = originalText.slice(0, index);
```

```

const match = originalText.slice(index, index + term.length);
const after = originalText.slice(index + term.length);

return `${before}<mark>${match}</mark>${after}`;
}

// ----- Item Handling -----
function initAddItemHandler() {
  const addItemModal = new bootstrap.Modal(document.getElementById("addItemModal"));

  document.getElementById("showAddFormBtn")?.addEventListener("click", () => {
    addItemModal.show();
  });

  document.getElementById("addItemBtn")?.addEventListener("click", async () => {
    Spinner.show();

    try {
      const spec = document.getElementById("newSpec").value.trim();
      const qty = document.getElementById("newQty").value.trim();
      const unit = document.getElementById("newUnit").value.trim();
      const remarks = document.getElementById("newRemarks").value.trim();
      const addedBy = localStorage.getItem("userFullName") || "Unknown";

      if (!spec || !qty || !unit) {
        NotificationBox.show("Specification, Quantity, and Unit are required.");
        return;
      }

      const duplicate = await Consumable.isSpecDuplicate(spec);
      if (duplicate) {
        NotificationBox.show("An item with the same specification already exists.");
        return;
      }

      await Consumable.add(spec, qty, unit, addedBy, remarks);
      NotificationBox.show("Consumable Item successfully added!");

      // Clear form
      document.getElementById("newSpec").value = "";
      document.getElementById("newQty").value = "";
      document.getElementById("newUnit").value = "";
      document.getElementById("newRemarks").value = "";

      addItemModal.hide();
      renderConsumableTable();
    } catch (error) {
      console.error("Error adding item:", error);
      NotificationBox.show("An error occurred while adding the item.");
    } finally {
      Spinner.hide();
    }
  });
}

```

```

}

function initEditItemHandler() {
  const editModal = new bootstrap.Modal(document.getElementById("editModal"));

  document.getElementById("saveEditBtn")?.addEventListener("click", async () => {
    Spinner.show();

    try {
      const cid = document.getElementById("editCID").value;
      const spec = document.getElementById("editSpec").value.trim();
      const unit = document.getElementById("editUnit").value.trim();
      const priority = document.getElementById("priorityCheckbox").checked;

      if (!spec || !unit) {
        NotificationBox.show("Specification and Unit are required.");
        return;
      }

      await Consumable.update(cid, { specification: spec, unit, priority });
      NotificationBox.show("Items description updated successfully.");
      editModal.hide();
      const searchTerm = document.getElementById("searchInput").value.trim();
      renderConsumableTable(searchTerm);
    } catch (error) {
      console.error("Error updating item:", error);
      NotificationBox.show("An error occurred while updating the item.");
    } finally {
      Spinner.hide();
    }
  });
}

// ----- Stock Handling -----
function initAddStockHandler() {
  const actionModal = new bootstrap.Modal(document.getElementById("actionModal"));
  const addStockModal = new bootstrap.Modal(document.getElementById("addStockModal"));

  // Open Add Stock Modal
  document.getElementById("addStockBtn")?.addEventListener("click", () => {
    actionModal.hide();
    document.getElementById("stockAmount").value = "";
    document.getElementById("stockRemarks").value = "";
    addStockModal.show();
  });

  // Confirm Add Stock
  document.getElementById("confirmAddStockBtn")?.addEventListener("click", () => {
    const amount = parseInt(document.getElementById("stockAmount").value.trim(), 10);
    const remarks = document.getElementById("stockRemarks").value.trim();

    if (isNaN(amount) || amount <= 0 || !selectedCID) {
      return NotificationBox.show("Invalid amount.");
    }
  });
}

```

```

}

Confirmation.show(
  "Are you sure you want to add to this stock?\nThis action cannot be undone.",
  async (confirm) => {
    if (!confirm) {
      return; // ❌ User clicked No
    }

    Spinner.show();

    try {
      await Consumable.addStock(selectedCID, amount, remarks);

      // Hide modals after success
      const actionModalInstance = bootstrap.Modal.getInstance(document.getElementById("actionModal"));
      const addStockModalInstance = bootstrap.Modal.getInstance(document.getElementById("addStockModal"));

      addStockModalInstance?.hide();
      actionModalInstance?.hide();

      NotificationBox.show("Items added successfully");
      const searchTerm = document.getElementById("searchInput").value.trim();
      renderConsumableTable(searchTerm);
    } catch (err) {
      console.error("Error adding stock:", err.message);
      NotificationBox.show("Something went wrong. Please try again.");
    } finally {
      Spinner.hide();
    }
  }
);

// ----- Assignment -----
function initAssignHandler() {
  const actionModal = new bootstrap.Modal(document.getElementById("actionModal"));
  const assignModal = new bootstrap.Modal(document.getElementById("assignModal"));

  // Open Assign Modal
  document.getElementById("assignItemBtn")?.addEventListener("click", async () => {
    if (!selectedCID) {
      return console.warn("No CID selected.");
    }

    const userSelect = document.getElementById("userSelect");
    if (!userSelect) return;

    userSelect.innerHTML = "";

    const defaultOption = document.createElement("option");
    defaultOption.disabled = true;

```

```
defaultOption.selected = true;
defaultOption.textContent = 'Select user';
userSelect.appendChild(defaultOption);

try {
  const users = await Users.fetchAllAsc();

  users.forEach(user => {
    if (user.status === 'active') {
      const fullName = `${user.lastName}, ${user.firstName} ${user.middleInitial || ""}`;
      const option = document.createElement("option");
      option.value = user.id;
      option.textContent = fullName.trim();
      userSelect.appendChild(option);
    }
  });
} catch (err) {
  console.error("Error fetching users:", err);
  NotificationBox.show("Failed to load users.");
  return;
}

assignModal.show();
});

// Handle Confirm Assign
document.getElementById("confirmAssignBtn")?.addEventListener("click", () => {
  const qtyInput = document.getElementById("assignQty").value.trim();
  const assignedTo = document.getElementById("userSelect").value;
  const remarks = document.getElementById("assignRemarks").value.trim();
  const amount = parseInt(qtyInput, 10);

  if (isNaN(amount) || amount <= 0) {
    NotificationBox.show("Please enter a valid quantity.");
    return;
  }

  if (!assignedTo) {
    NotificationBox.show("Please select a user to assign to.");
    return;
  }

  // Confirmation dialog
  Confirmation.show(
    `Are you sure you want to assign ${amount} item(s) to this user?\nThis action cannot be undone.`,
    async (confirm) => {
      if (!confirm) return;

      Spinner.show();

      try {
        await Consumable.assignItem(selectedCID, amount, assignedTo, remarks);
      
```

```

NotificationBox.show('Items assigned successfully');

const searchTerm = document.getElementById("searchInput").value.trim();
renderConsumableTable(searchTerm);
assignModal.hide();
actionModal.hide();
document.getElementById("assignForm").reset();

} catch (error) {
  console.error("Error assigning item:", error);
  NotificationBox.show(error.message || "Something went wrong.");
} finally {
  Spinner.hide();
}
};

});

// ----- Ledger -----
function initViewLedgerHandler() {
  document.getElementById("viewLedgerBtn")?.addEventListener("click", async () => {
    Spinner.show();
    const totalQtyDisplay = document.getElementById("totalQtyDisplay");

    try {
      const { totalQty, ledgerEntries } = await Ledger.fetchLedgerDataByCID(selectedCID);

      totalQtyDisplay.textContent = totalQty !== null ? totalQty : "Not found";

      const blob = await generateLedgerPDFBlob(selectedCID, ledgerEntries, totalQty);
      document.getElementById("pdfPreviewFrame").src = URL.createObjectURL(blob);

      new bootstrap.Modal(document.getElementById("ledgerModal")).show();
    } catch (err) {
      console.error("Error generating ledger:", err);
      NotificationBox.show("Failed to generate ledger.");
    } finally {
      Spinner.hide();
    }
  });
}

function initReassignItemModal() {
  const reassignBtn = document.getElementById("reassignItemBtn");
  const reassignModal = document.getElementById("reassignModal");

  if (!reassignBtn || !reassignModal) {
    console.warn("Reassign button or modal not found in DOM.");
    return;
  }

  const modalInstance = new bootstrap.Modal(reassignModal);
}

```

```
reassignBtn.addEventListener("click", async () => {
  const cid = reassignBtn.getAttribute("data-id");
  if (!cid) {
    console.warn("No CID found in data-id attribute.");
    return;
  }

  const tableBody = document.getElementById("reassignTableBody");
  tableBody.innerHTML = "<tr><td colspan='5'>Loading...</td></tr>";

  try {
    Spinner.show();
    const { ledgerEntries } = await Ledger.fetchLedgerDataByCID(cid);
    const assignEntries = ledgerEntries.filter(entry => entry.action === "Assign Item");

    if (assignEntries.length === 0) {
      tableBody.innerHTML = "<tr><td colspan='5'>No assigned entries found.</td></tr>";
      modalInstance.show();
      return;
    }

    const rowsHtml = assignEntries.map(entry => {
      const formattedDate = entry.dateModified?.toDate
        ? entry.dateModified.toDate().toLocaleDateString("en-US", {
          year: "numeric",
          month: "long",
          day: "numeric",
        })
        : "Unknown date";

      return `
        <tr>
          <td>${formattedDate}</td>
          <td>${entry.amount}</td>
          <td>${entry.assignedTo}</td>
          <td>${entry.remarks} || —</td>
          <td>
            <button class="btn btn-3d btn-sm btn-outline-danger reassign-btn" data-entry-id="${entry.id}">
              Reassign Item
            </button>
          </td>
        </tr>
      `;
    }).join("");

    tableBody.innerHTML = rowsHtml;
    modalInstance.show();
  } catch (err) {
    console.error("Error loading assign items:", err);
    NotificationBox.show("Failed to load data.", "danger");
    tableBody.innerHTML = "<tr><td colspan='5'>Failed to load data.</td></tr>";
    modalInstance.show();
  }
}
```

```

} finally {
  Spinner.hide();
}
});

// Handle dynamic "Reassign Item" button clicks
document.getElementById("reassignTableBody").addEventListener("click", async (e) => {
  if (!e.target.classList.contains("reassign-btn")) return;

  const button = e.target;
  const entryId = button.getAttribute("data-entry-id");
  const cid = reassignBtn.getAttribute("data-id");

  try {
    Spinner.show();
    const { ledgerEntries } = await Ledger.fetchLedgerDataByCID(cid);
    const entry = ledgerEntries.find(e => e.id === entryId);

    if (!entry) {
      NotificationBox.show("Ledger entry not found.", "danger");
      return;
    }

    Confirmation.show(
      `Reassigning this item will restore ${entry.amount} to stock. Continue?`,
      async (confirmed) => {
        if (!confirmed) return;

        try {
          Spinner.show();
          await Consumable.addStock(cid, entry.amount, `Reassigned from ${entry.assignedTo}`);
          const searchTerm = document.getElementById("searchInput").value.trim();
          renderConsumableTable(searchTerm);

          button.closest("tr").remove();

          const tableBody = document.getElementById("reassignTableBody");
          if (tableBody.children.length === 0) {
            tableBody.innerHTML = "<tr><td colspan='5'>No assigned entries found.</td></tr>";
          }

          NotificationBox.show("Item reassigned successfully.", "success");
        } catch (err) {
          console.error("Error during reassignment:", err);
          NotificationBox.show("An error occurred while reassigning the item.", "danger");
        } finally {
          Spinner.hide();
        }
      }
    );
  } catch (err) {
    console.error("Failed to process reassignment:", err);
    NotificationBox.show("Failed to process reassignment.", "danger");
  }
});

```

```
        Spinner.hide();
    }
});
}

function initActionButtonHandler() {
    document.addEventListener("click", async (e) => {
        const actionBtn = e.target.closest(".action-btn");
        if (actionBtn) {
            const cid = actionBtn.getAttribute("data-id");
            const qty = actionBtn.getAttribute("data-qty");
            selectedCID = cid;

            // Assign data-id to buttons
            document.getElementById("viewLedgerBtn")?.setAttribute("data-id", cid);
            document.getElementById("reassignItemBtn")?.setAttribute("data-id", cid);
        }
    });
}

// ----- Search -----
function initSearchHandler() {
    document.getElementById("searchInput")?.addEventListener("input", () => {
        const rawTerm = document.getElementById("searchInput").value.trim();
        currentSearchTerm = rawTerm;

        const normalizedTerm = normalizeText(rawTerm);

        filteredItems = currentItems.filter(item =>
            normalizeText(item.specification || "").includes(normalizedTerm) ||
            normalizeText(item.id || "").includes(normalizedTerm)
        );

        localStorage.setItem("filteredConsumables", JSON.stringify(filteredItems));
    });

    currentPage = 1;
    renderTablePage();
    renderPagination();
};

}

function handleRefreshButton({
    buttonId,
    refreshFn,
    renderFn,
    cooldownKey,
    cooldownSeconds = 60
}) {
    if (!buttonId || !refreshFn || !renderFn || !cooldownKey) {
        console.error("handleRefreshButton: Missing required arguments.");
        return;
    }
}
```

```
const refreshBtn = document.getElementById(buttonId);
if (!refreshBtn) {
  console.error(`handleRefreshButton: Button with ID "${buttonId}" not found.`);
  return;
}

let originalText = refreshBtn.textContent;
let cooldownTimer = null;

function startCooldown() {
  const startTime = Date.now();
  const endTime = startTime + cooldownSeconds * 1000;
  localStorage.setItem(cooldownKey, startTime.toString());

  refreshBtn.disabled = true;

  cooldownTimer = setInterval(() => {
    const now = Date.now();
    const secondsLeft = Math.ceil((endTime - now) / 1000);

    if (secondsLeft <= 0) {
      clearInterval(cooldownTimer);
      refreshBtn.disabled = false;
      refreshBtn.textContent = originalText;
    } else {
      refreshBtn.textContent = `Wait (${secondsLeft}s)`;
    }
  }, 1000);
}

// Resume countdown if cooldown still active
const lastRefresh = parseInt(localStorage.getItem(cooldownKey)) || 0;
const now = Date.now();
if (now - lastRefresh < cooldownSeconds * 1000) {
  startCooldown();
}

refreshBtn.addEventListener("click", async () => {
  const now = Date.now();
  const lastRefresh = parseInt(localStorage.getItem(cooldownKey)) || 0;

  if (now - lastRefresh < cooldownSeconds * 1000) {
    return; // Ignore clicks during cooldown
  }

  try {
    await refreshFn();
    renderFn();
    Consumable.refreshCache();
    Ledger.refreshCache();
    const currentSearch = document.getElementById("searchInput").value.trim();
    const searchTerm = document.getElementById("searchInput").value.trim();
    renderConsumableTable(searchTerm);
  }
});
```

```
NotificationBox.show("Refreshed successfully.");
startCooldown();
} catch (err) {
  console.error("Refresh error:", err);
  NotificationBox.show("Failed to refresh data.");
}
});

}

// ----- Table Rendering -----
async function renderConsumableTable(searchTerm = "") {
  Spinner.show();

  try {
    currentItems = await Consumable.fetchAll();

    currentSearchTerm = searchTerm;
    const normalizedTerm = normalizeText(searchTerm);

    filteredItems = searchTerm
      ? currentItems.filter(item =>
        normalizeText(item.specification || "").includes(normalizedTerm) ||
        normalizeText(item.id || "").includes(normalizedTerm)
      )
      : [...currentItems];

    localStorage.setItem("filteredConsumables", JSON.stringify(filteredItems));
    currentPage = 1;
    renderTablePage();
    renderPagination();
  } catch (err) {
    console.error("Error fetching consumables:", err);
    document.getElementById("consumableBody").innerHTML = `<tr><td colspan="8">Error loading items.</td></tr>`;
  } finally {
    Spinner.hide();
  }
}

function renderTablePage() {
  const tbody = document.getElementById("consumableBody");
  tbody.innerHTML = "";

  const start = (currentPage - 1) * pageSize;
  const pagelItems = filteredItems.slice(start, start + pageSize);

  if (pagelItems.length === 0) {
    tbody.innerHTML = `<tr><td colspan="8">No items found.</td></tr>`;
    return;
  }

  pagelItems.forEach(item => {
    const row = document.createElement("tr");
    const highlightedSpec = highlightMatch(item.specification || "", currentSearchTerm);
```

```
const highlightedId = highlightMatch(item.id || "", currentSearchTerm);

row.innerHTML =
<td>${highlightedId}</td>
<td>${highlightedSpec}</td>
<td>${item.qty}</td>
<td>${item.unit}</td>
<td>
<button class="btn btn-3d btn-warning btn-sm edit-btn"
  data-id="${item.id}"
  data-spec="${item.specification}"
  data-unit="${item.unit}"
  data-priority="${item.priority}"
  data-bs-toggle="modal"
  data-bs-target="#editModal">
  <i class="bi bi-pencil-square"></i>
</button>
</td>
<td>
<button class="btn btn-3d btn-secondary btn-sm action-btn"
  data-id="${item.id}"
  data-qty="${item.qty}"
  data-bs-toggle="modal"
  data-bs-target="#actionModal"
  title="Perform Action">
  <i class="bi bi-gear"></i>
</button>
</td>
`;
tbody.appendChild(row);
});

handleEditButtons();
}

function handleEditButtons() {
  const editButtons = document.querySelectorAll(".edit-btn");

  editButtons.forEach(button => {
    button.addEventListener("click", () => {
      const id = button.getAttribute("data-id");
      const spec = button.getAttribute("data-spec");
      const unit = button.getAttribute("data-unit");
      const priority = button.getAttribute("data-priority");

      // Populate the modal fields
      document.getElementById("editCID").value = id;
      document.getElementById("editSpec").value = spec;
      document.getElementById("editUnit").value = unit;
      document.getElementById("priorityCheckbox").checked = priority === "true";
    });
  });
}
```

```
function renderPagination() {
  const paginationContainer = document.getElementById("paginationContainer");
  if (!paginationContainer) return;

  paginationContainer.innerHTML = "";

  const totalPages = Math.ceil(filteredItems.length / pageSize);
  if (totalPages <= 1) return;

  const nav = document.createElement("nav");
  const ul = document.createElement("ul");
  ul.classList.add("pagination");

  // Previous Button
  const prevLi = document.createElement("li");
  prevLi.classList.add("page-item");
  if (currentPage === 1) prevLi.classList.add("disabled");

  const prevBtn = document.createElement("button");
  prevBtn.classList.add("page-link");
  prevBtn.textContent = "Previous";
  prevBtn.addEventListener("click", () => {
    if (currentPage > 1) {
      currentPage--;
      renderTablePage();
      renderPagination();
    }
  });
  prevLi.appendChild(prevBtn);
  ul.appendChild(prevLi);

  // Calculate page range (max 5 pages at a time)
  let startPage = Math.max(1, currentPage - 2);
  let endPage = startPage + 4;

  if (endPage > totalPages) {
    endPage = totalPages;
    startPage = Math.max(1, endPage - 4);
  }

  // Numbered Buttons
  for (let i = startPage; i <= endPage; i++) {
    const li = document.createElement("li");
    li.classList.add("page-item");
    if (i === currentPage) li.classList.add("active");

    const btn = document.createElement("button");
    btn.classList.add("page-link");
    btn.textContent = i;

    btn.addEventListener("click", () => {
```

```
currentPage = i;
renderTablePage();
renderPagination();
});

li.appendChild(btn);
ul.appendChild(li);
}

// Next Button
const nextLi = document.createElement("li");
nextLi.classList.add("page-item");
if (currentPage === totalPages) nextLi.classList.add("disabled");

const nextBtn = document.createElement("button");
nextBtn.classList.add("page-link");
nextBtn.textContent = "Next";
nextBtn.addEventListener("click", () => {
  if (currentPage < totalPages) {
    currentPage++;
    renderTablePage();
    renderPagination();
  }
});
nextLi.appendChild(nextBtn);
ul.appendChild(nextLi);

nav.appendChild(un);
paginationContainer.appendChild(nav);
}

function pageSizeSelectHandler() {
  const pageSizeSelect = document.getElementById('pageSizeSelect');
  if (!pageSizeSelect) return;

  pageSizeSelect.addEventListener('change', () => {
    pageSize = parseInt(pageSizeSelect.value);
    currentPage = 1; // Reset to first page
    renderTablePage();
    renderPagination();
  });
}

function setupExportFilteredConsumablesToExcel() {
  const exportBtn = document.getElementById("exportBtn");
  if (!exportBtn) return;

  exportBtn.addEventListener("click", () => {
    const data = JSON.parse(localStorage.getItem("filteredConsumables") || "[]");

    if (data.length === 0) {
      alert("No data to export.");
    }
  });
}
```

```
return;  
}  
  
const exportData = data.map(item => ({  
  ID: item.id,  
  Specification: item.specification,  
  Quantity: item.qty,  
  Unit: item.unit  
}));  
  
const worksheet = XLSX.utils.json_to_sheet(exportData);  
const workbook = XLSX.utils.book_new();  
XLSX.utils.book_append_sheet(workbook, worksheet, "Consumables");  
  
XLSX.writeFile(workbook, "filtered_consumables.xlsx");  
});  
}
```

XV.C.4.b Dashboard Module

admin-dashboard-init.js

```
import { AdminGuard } from "../auth/auth.js";  
import { Sidebar } from "../components/sidebar.js";  
import {  
  checkAuthentication,  
  refreshAllCachesEvery8Hours,  
  showEncodedPermitsByUserPerMonth,  
  handleEncodedRefreshButton  
} from './admin-dashboard-ui.js';
```

```
function init() {  
  AdminGuard.verify();  
  checkAuthentication();  
  refreshAllCachesEvery8Hours();  
  showEncodedPermitsByUserPerMonth();  
  Sidebar.render();  
  handleEncodedRefreshButton();  
}
```

```
document.addEventListener('DOMContentLoaded', init);
```

admin-dashboard-ui.js

```
import { Permit } from "../data/cache/permit-data.js";  
import { Users } from "../data/cache/user-data.js"  
import { Consumable } from "../data/cache/consumable-data.js";  
import { ICS } from "../data/cache/ics-data.js";  
import { WUSData } from "../data/cache/wrus-data.js";  
import { PortalBubble } from "../components/PortalBubble.js";  
import { NotificationBox } from "../components/notification.js";  
import { Ledger } from "../data/cache/ledger-data.js";
```

```

// 🔒 Authentication Check
export function checkAuthentication() {
  const user = localStorage.getItem("loggedInUser");
  if (!user) window.location.href = "index.html";
}

// 🔄 Refresh Cache Every 8 Hours
export async function refreshAllCachesEvery8Hours() {
  await Promise.all([
    Users.autoRefreshEvery8Hours?(),
    Consumable.autoRefreshEvery8Hours?(),
    ICS.autoRefreshEvery8Hours?(),
    Permit.autoRefreshEvery8Hours?(),
    WUSData.autoRefreshEvery8Hours?(),
    Ledger.autoRefreshEvery8Hours?()
  ]);
}

// 🔄 Handle Refresh Button for Permit Summary
export function handleEncodedRefreshButton() {
  PortalBubble.trigger();
  const refreshBtn = document.getElementById('refreshEncodedBtn');
  const COOLDOWN_SECONDS = 60;
  const LAST_REFRESH_KEY = 'lastEncodedRefresh';

  function getRemainingCooldown() {
    const lastRefresh = localStorage.getItem(LAST_REFRESH_KEY);
    if (!lastRefresh) return 0;
    const elapsed = (Date.now() - parseInt(lastRefresh, 10)) / 1000;
    return Math.max(0, COOLDOWN_SECONDS - Math.floor(elapsed));
  }

  function startCooldown() {
    localStorage.setItem(LAST_REFRESH_KEY, Date.now().toString());
    let remaining = COOLDOWN_SECONDS;
    refreshBtn.disabled = true;
    const originalHTML = refreshBtn.innerHTML;

    const interval = setInterval(() => {
      remaining--;
      refreshBtn.innerHTML = `<i class="bi bi-hourglass-split me-1"></i> ${remaining}s`;
      if (remaining <= 0) {
        clearInterval(interval);
        refreshBtn.disabled = false;
        refreshBtn.innerHTML = originalHTML;
      }
    }, 1000);
  }

  // On load, check if cooldown is active
  const remainingCooldown = getRemainingCooldown();
  if (remainingCooldown > 0) {

```

```
refreshBtn.disabled = true;
let remaining = remainingCooldown;
const originalHTML = refreshBtn.innerHTML;
refreshBtn.innerHTML = `<i class="bi bi-hourglass-split me-1"></i> ${remaining}s`;

const interval = setInterval(() => {
  remaining--;
  refreshBtn.innerHTML = `<i class="bi bi-hourglass-split me-1"></i> ${remaining}s`;
  if (remaining <= 0) {
    clearInterval(interval);
    refreshBtn.disabled = false;
    refreshBtn.innerHTML = originalHTML;
  }
}, 1000);

refreshBtn.addEventListener('click', async () => {
  const remaining = getRemainingCooldown();
  if (remaining > 0) {
    NotificationBox.show(`Please wait ${remaining}s before refreshing again.`);
    return;
  }

  try {
    refreshBtn.disabled = true;
    refreshBtn.innerHTML = `<i class="bi bi-arrow-clockwise me-1"></i> Refreshing...`;
    await showEncodedPermitsByUserPerMonth();
    NotificationBox.show("Encoded permits refreshed.");
    startCooldown();
  } catch (err) {
    refreshBtn.disabled = false;
    refreshBtn.innerHTML = `<i class="bi bi-arrow-clockwise"></i>`;
    console.error(err);
    NotificationBox.show("Error during encoded refresh.");
  }
});

// 📈 Permit Summary by User and Month
export async function showEncodedPermitsByUserPerMonth() {
  try {
    const permits = await Permit.getAll();
    const listContainer = document.getElementById('encodedPermitTableBody');
    const noRecord = document.getElementById('noEncodedMessage');

    listContainer.innerHTML = "";

    if (!permits.length) {
      listContainer.classList.add('d-none');
      noRecord.classList.remove('d-none');
      return;
    }
  }
}
```

```

const usersMap = await Users.getUsersMap();
const summary = {} // { userId: { "Month Year": count } }
const monthSet = new Set();

permits.forEach(p => {
  const userId = p.encodedBy;
  if (!userId) return;

  const date = p.createdAt?.toDate()?.() || new Date(p.timestamp?.seconds * 1000);
  const monthYear = date.toLocaleString('default', { month: 'long', year: 'numeric' });
  monthSet.add(monthYear);

  if (!summary[userId]) summary[userId] = {};
  if (!summary[userId][monthYear]) summary[userId][monthYear] = 0;
  summary[userId][monthYear]++;
});

const monthsSorted = Array.from(monthSet).sort((a, b) => {
  return new Date('1 ' + a) - new Date('1 ' + b);
});

const pastelColors = [
  '#ffe0e0', '#e0f7fa', '#e8f5e9',
  '#f3e5f5', '#fff3e0', '#f0f4c3',
  '#fbe9e7', '#ede7f6'
];

let colorIndex = 0;
const userEntries = Object.entries(summary);
for (let i = 0; i < userEntries.length; i += 3) {
  const row = document.createElement('div');
  row.className = 'scorecard-row';

  const chunk = userEntries.slice(i, i + 3);
  chunk.forEach(([userId, userData]) => {
    const fullName = usersMap[userId] || `Unknown (${userId})`;
    const column = document.createElement('div');
    column.className = 'scorecard-column';
    column.style.backgroundColor = pastelColors[colorIndex % pastelColors.length];
    colorIndex++;

    const header = document.createElement('div');
    header.className = 'user-header';
    header.innerHTML = ` ${fullName}`;
    column.appendChild(header);

    monthsSorted.forEach(month => {
      const count = userData[month] || 0;
      const entry = document.createElement('div');
      entry.className = 'scorecard-entry';
      entry.innerHTML =
        `

<i class="bi bi-calendar3"></i> ${month}</div>
        <div><i class="bi bi-check2-circle"></i> ${count} Permit${count === 1 ? 's' : ''}</div>`;
      column.appendChild(entry);
    });
  });
}


```

```
        `;
        column.appendChild(entry);
    });

    row.appendChild(column);
});

listContainer.appendChild(row);
}

listContainer.classList.remove('d-none');
noRecord.classList.add('d-none');

} catch (error) {
    console.error("Admin permit summary error:", error);
}
}
```

change-password.js

```
import bcrypt from "https://esm.sh/bcryptjs@2.4.3";
import { Users } from "../data/cache/user-data.js";
import { NotificationBox } from "../components/notification.js";
import { Spinner } from "../components/spinner.js";

export function initializePasswordChange() {
    const form = document.getElementById("changePasswordForm");
    const newPassword = document.getElementById("newPassword");
    const confirmPassword = document.getElementById("confirmPassword");

    if (!form || !newPassword || !confirmPassword) {
        console.warn("Password change form or fields not found.");
        return;
    }

    form.onsubmit = async (e) => {
        e.preventDefault();

        Spinner.show();

        try {
            const userId = localStorage.getItem("wrusUserId");
            const pass = newPassword.value.trim();
            const confirm = confirmPassword.value.trim();

            if (!userId) {
                NotificationBox.show("User not found.");
                return;
            }

            if (pass.length < 6) {
                NotificationBox.show("Password must be at least 6 characters.", "error");
                return;
            }

            const user = await Users.get(userId);
            const hashedPass = await bcrypt.hash(pass, 10);

            if (user.password === hashedPass) {
                user.password = pass;
                await user.save();
                NotificationBox.show("Password changed successfully!");
            } else {
                NotificationBox.show("Current password is incorrect.");
            }
        } catch (error) {
            console.error("Error changing password:", error);
            NotificationBox.show("An error occurred while changing the password.", "error");
        }
    };
}
```

```

        }

        if (pass !== confirm) {
            NotificationBox.show("Passwords do not match.", "error");
            return;
        }

        const hashedPassword = await bcrypt.hash(pass, 10);
        await Users.update(userId, { password: hashedPassword });

        NotificationBox.show("Password updated successfully.");
        form.reset();
    } catch (error) {
        console.error("Error changing password:", error);
        NotificationBox.show("An error occurred while updating the password.", "error");
    } finally {
        Spinner.hide();
    }
};

}

```

city-accomplishment-summary.js

```

import { WUSData } from "../data/cache/wrus-data.js";
import { Permit } from "../data/cache/permit-data.js";
import { normalizeBarangay, normalizeCity } from "../utils/normalize.js"
import { updateMapLocation, addMapMarker, clearExtraMarkers } from './map/map-init.js';

export async function initCityAccomplishmentSummary() {
    const data = await WUSData.fetchAll();
    const container = document.getElementById('cityAccomplishmentTable');
    if (!container) return;

    const grouped = {};
    const barangayGrouped = {};

    let grandPermittees = 0;
    let grandNonPermittees = 0;
    let grandWaterSources = 0;

    data.forEach(entry => {
        const city = normalizeCity(entry.city);
        const barangay = normalizeBarangay(entry.barangay);
        const hasPermit = !!entry.permitNo;
        const isOperational = entry.remarks?.toLowerCase() === 'operational';

        if (!grouped[city]) grouped[city] = { permittees: 0, nonPermittees: 0, waterSources: 0 };
        if (!barangayGrouped[city]) barangayGrouped[city] = {};
        if (!barangayGrouped[city][barangay]) barangayGrouped[city][barangay] = { permittees: 0, nonPermittees: 0, waterSources: 0 };

        if (hasPermit) {
            grouped[city].permittees++;
            barangayGrouped[city][barangay].permittees++;
        } else {
            grouped[city].nonPermittees++;
            barangayGrouped[city][barangay].nonPermittees++;
        }

        if (isOperational) {
            grouped[city].waterSources++;
            barangayGrouped[city][barangay].waterSources++;
        }
    });
}


```

```

barangayGrouped[city][barangay].permittees++;
grandPermittees++;
} else {
grouped[city].nonPermittees++;
barangayGrouped[city][barangay].nonPermittees++;
grandNonPermittees++;
}

if (isOperational) {
grouped[city].waterSources++;
barangayGrouped[city][barangay].waterSources++;
grandWaterSources++;
}
});

const today = new Date();
const dateStr = today.toLocaleDateString('en-US', { year: 'numeric', month: 'long', day: 'numeric' });

const cities = Object.keys(grouped).sort();
let html = `
<h3 class="mb-3">Water Users and Sources Database
<span class="fs-6 fw-normal"> — As of ${dateStr}</span>
</h3>
<table class="table table-bordered table-striped align-middle text-center rounded-0">
<thead class="table-success">
<tr>
<th rowspan="2">City</th>
<th colspan="3">Water Users</th>
<th rowspan="2">Water Sources</th>
<th rowspan="2">Barangay Details</th>
</tr>
<tr>
<th>Permittees</th>
<th>Non-Permittees</th>
<th>Total</th>
</tr>
</thead>
<tbody>
`;

cities.forEach(city => {
const { permittees, nonPermittees, waterSources } = grouped[city];
const total = permittees + nonPermittees;
html += `
<tr>
<td>${city}</td>
<td>${permittees}</td>
<td>${nonPermittees}</td>
<td>${total}</td>
<td>${waterSources}</td>
<td>
<button class="btn btn-3d btn-sm btn-outline-primary view-barangay-btn"
data-city="${city}" data-bs-toggle="modal" data-bs-target="#barangayModal">

```

```

        View Barangays
    </button>
    <button class="btn btn-3d btn-sm btn-outline-success view-map-btn"
        data-city="${city}" data-bs-toggle="modal" data-bs-target="#mapModal">
        View Map
    </button>
</td>
</tr>';
});

const grandTotalUsers = grandPermittees + grandNonPermittees;
html += `
<tr class="table-secondary fw-bold">
<td>Total</td>
<td>${grandPermittees}</td>
<td>${grandNonPermittees}</td>
<td>${grandTotalUsers}</td>
<td>${grandWaterSources}</td>
<td></td>
</tr>
</tbody>
</table>`;

container.innerHTML = html;

attachBarangayButtonListener(barangayGrouped);
attachViewUsersListener();
attachViewMapListener();
}

function attachBarangayButtonListener(barangayGrouped) {
document.addEventListener('click', function (e) {
    const btn = e.target.closest('.view-barangay-btn');
    if (!btn) return;
    handleViewBarangayClick(btn, barangayGrouped);
});
}

function handleViewBarangayClick(btn, barangayGrouped) {
const city = btn.getAttribute('data-city');
const modalLabel = document.getElementById('barangayModalLabel');
const modalBody = document.getElementById('barangayModalBody');

modalLabel.textContent = `Barangay Breakdown — ${city}`;

const barangays = barangayGrouped[city];
if (!barangays) {
    modalBody.innerHTML = `

No data available for ${city}.

`;
    return;
}

let totalPermittees = 0;
let totalNonPermittees = 0;

```

```

let totalSources = 0;

let barangayHtml = `


| Barangay   | Water Users    |       |  | Actions |
|------------|----------------|-------|--|---------|
|            | Water Sources  |       |  |         |
| Permittees | Non-Permittees | Total |  |         |


Object.keys(barangays).sort().forEach(brgy => {
  const { permittees, nonPermittees, waterSources } = barangays[brgy];
  const total = permittees + nonPermittees;

  totalPermittees += permittees;
  totalNonPermittees += nonPermittees;
  totalSources += waterSources;

  barangayHtml += `
    <tr>
      <td>${brgy}</td>
      <td>${permittees}</td>
      <td>${nonPermittees}</td>
      <td>${total}</td>
      <td>${waterSources}</td>
      <td>
        <button class="btn btn-3d btn-sm btn-outline-primary view-users-btn"
          data-city="${city}" data-barangay="${brgy}">
          View Users
        </button>
        <button class="btn btn-3d btn-sm btn-outline-success ms-1 view-map-btn"
          data-city="${city}" data-barangay="${brgy}">
          View Map
        </button>
      </td>
    </tr>`;
});

const cityTotal = totalPermittees + totalNonPermittees;

barangayHtml += `
<tr class="table-secondary fw-bold">
  <td>Total</td>
  <td>${totalPermittees}</td>

```

```

<td>${totalNonPermittees}</td>
<td>${cityTotal}</td>
<td>${totalSources}</td>
<td></td>
</tr>
</tbody>
</table>`;

modalBody.innerHTML = barangayHtml;

modalBody.addEventListener('click', handleViewBarangayMapClick);
}

function attachViewUsersListener() {
  document.addEventListener('click', async function (e) {
    const btn = e.target.closest('.view-users-btn');
    if (btn) {
      const city = btn.getAttribute('data-city');
      const barangay = btn.getAttribute('data-barangay');

      const modalLabel = document.getElementById('waterUsersModalLabel');
      const modalBody = document.getElementById('waterUsersModalBody');

      modalLabel.textContent = `Water Users in ${barangay}, ${city}`;
      modalBody.innerHTML = `

Loading...

`;

      try {
        const allUsers = await WUSData.fetchAll();
        const filteredUsers = allUsers.filter(user =>
          normalizeCity(user.city) === city &&
          normalizeBarangay(user.barangay) === barangay
        );

        if (filteredUsers.length === 0) {
          modalBody.innerHTML = `

No water users found for ${barangay}.</p>`;
        } else {
          let usersHtml = `
            <table class="table table-bordered table-striped align-middle text-center">
              <thead class="table-success">
                <tr>
                  <th>Owner</th>
                  <th>Street</th>
                  <th>Latitude</th>
                  <th>Longitude</th>
                  <th>Permit No.</th>
                  <th>Type</th>
                  <th>Year Inspected</th>
                  <th>Is Water Source?</th>
                  <th>Geotagged Image</th>
                  <th>View Map</th>
                </tr>
              </thead>
              <tbody>
`;
        }
      }
    }
  });
}


```

```

`;

filteredUsers.forEach(user => {
  usersHtml += `
    <tr>
      <td>${user.owner || '-'}</td>
      <td>${user.street || '-'}</td>
      <td>${user.latitude !== undefined && user.latitude !== null ? Number(user.latitude).toFixed(5) : '-'}</td>
      <td>${user.longitude !== undefined && user.longitude !== null ? Number(user.longitude).toFixed(5) : '-'}</td>
      <td>${user.permitNo || '-'}</td>
      <td>${user.type || '-'}</td>
      <td>${user.year_conducted || '-'}</td>
      <td>${user.isWaterSource ? 'Yes' : 'No'}</td>
      <td>
        ${user.geotaggedUrl
          ? '<a href="' + user.geotaggedUrl + '" target="_blank" class="btn btn-sm btn-outline-secondary">View</a>'
          : '-'}
      </td>
      <td>
        <button
          class="btn btn-3d btn-sm btn-outline-primary view-map-btn"
          data-id="${user.id}"
        >Show in Map
        </button>
      </td>
    </tr>
  `;
});

usersHtml += `</tbody></table>`;
modalBody.innerHTML = usersHtml;
}

const waterUsersModal = new bootstrap.Modal(document.getElementById('waterUsersModal'), {
  backdrop: false
});
waterUsersModal.show();

} catch (err) {
  console.error(err);
  modalBody.innerHTML = `<p class="text-danger">Error loading water users.</p>`;
}
}

// ✅ This part stays the same since handleViewUserMapClick() now gets the ID
if (e.target.classList.contains('view-map-btn')) {
  await handleViewUserMapClick(e);
}
};

function attachViewMapListener() {
  document.querySelectorAll('.view-map-btn').forEach(button => {

```

```

button.addEventListener('click', async (e) => {
  const clickedCity = e.target.getAttribute('data-city');
  const normalizedClickedCity = normalizeCity(clickedCity);

  document.getElementById('mapModalLabel').textContent = `${clickedCity} Map`;

  const allData = await WUSData.fetchAll();
  const cachedPermits = await Permit.getAll();
  const cityEntries = allData.filter(entry => normalizeCity(entry.city) === normalizedClickedCity);

  clearExtraMarkers();

  if (cityEntries.length > 0) {
    const avgLat = cityEntries.reduce((sum, e) => sum + (parseFloat(e.latitude) || 0), 0) / cityEntries.length;
    const avgLng = cityEntries.reduce((sum, e) => sum + (parseFloat(e.longitude) || 0), 0) / cityEntries.length;

    updateMapLocation(clickedCity, avgLat, avgLng);

    for (const entry of cityEntries) {
      if (entry.latitude && entry.longitude) {
        const sourceStatus = (entry.isWaterSource === true || entry.isWaterSource === 'true')
          ? '<span style="color:blue; font-weight:bold;">Water Source</span>'
          : '<span style="color:green; font-weight:bold;">Not a Water Source</span>';

        const matchedPermitMeta = cachedPermits.find(p => p.permitNo === entry.permitNo);
        let pdfUrl = 'images/permitNotFound.png';

        if (matchedPermitMeta?.id) {
          try {
            const fullPermit = await Permit.getByID(matchedPermitMeta.id);
            if (fullPermit?.pdfUrl) {
              pdfUrl = fullPermit.pdfUrl;
            }
          } catch (err) {
            console.warn(`Failed to fetch permit by ID: ${matchedPermitMeta.id}`, err.message);
          }
        }
      }
    }
  }

  let popupText = `

    <strong>${entry.permittee} || ${entry.owner} || 'Unknown Site'</strong><br>
     Permit No.:</strong>
    <a href="${pdfUrl}" target="_blank">${entry.permitNo} || 'N/A'</a><br>
     ${entry.street} || 'Street not specified'<br>
     Lat: ${parseFloat(entry.latitude).toFixed(5)},<br>
    Lng: ${parseFloat(entry.longitude).toFixed(5)}<br>
     Source: ${entry.type}<br>
     Year Inspected: ${entry.year_conducted} || 'N/A'<br>
     Status: ${sourceStatus}<br>
  `;

  if (entry.geotaggedUrl) {
    popupText += `

```

```

<div style="margin-top: 5px;">
  <a href="${entry.geotaggedUrl}" target="_blank">
    
  </a>
</div>
`;
}

addMapMarker(
  entry.latitude,
  entry.longitude,
  popupText,
  entry.isWaterSource,
  entry.permitNo
);
}
}
} else {
  console.warn(`No entries found for "${clickedCity}"`);
}
});
};

async function handleViewBarangayMapClick(event) {
  if (!event.target.classList.contains('view-map-btn')) return;

  const clickedCity = event.target.getAttribute('data-city');
  const clickedBarangay = event.target.getAttribute('data-barangay');

  const normalizedCity = normalizeCity(clickedCity);
  const normalizedBarangay = normalizeBarangay(clickedBarangay);

  document.getElementById('mapModalLabel').textContent = `${clickedBarangay}, ${clickedCity} Map`;

  const allData = await WUSData.fetchAll();
  const cachedPermits = await Permit.getAll(); // ✅ needed to find id

  const barangayEntries = allData.filter(entry =>
    normalizeCity(entry.city) === normalizedCity &&
    normalizeBarangay(entry.barangay) === normalizedBarangay
  );

  clearExtraMarkers();

  if (barangayEntries.length > 0) {
    const avgLat = barangayEntries.reduce((sum, e) => sum + (parseFloat(e.latitude) || 0), 0) / barangayEntries.length;
    const avgLng = barangayEntries.reduce((sum, e) => sum + (parseFloat(e.longitude) || 0), 0) / barangayEntries.length;

    updateMapLocation(`${normalizedBarangay}, ${normalizedCity}`, avgLat, avgLng);
  }
}

```

```

for (const entry of barangayEntries) {
  if (entry.latitude && entry.longitude) {

    const sourceStatus = (entry.isWaterSource === true || entry.isWaterSource === 'true')
      ? '<span style="color:blue; font-weight:bold;">Water Source</span>'
      : '<span style="color:green; font-weight:bold;">Not a Water Source</span>';

    // ✅ Lookup permit.id first, then fetch full doc
    const matchedPermitMeta = cachedPermits.find(p => p.permitNo === entry.permitNo);
    let pdfUrl = 'images/permitNotFound.png';

    if (matchedPermitMeta?.id) {
      try {
        const fullPermit = await Permit.getById(matchedPermitMeta.id);
        if (fullPermit?.pdfUrl) {
          pdfUrl = fullPermit.pdfUrl;
        }
      } catch (err) {
        console.warn(`Failed to fetch permit by ID: ${matchedPermitMeta.id}`, err.message);
      }
    }

    let popupText =
      `${entry.permittee} || ${entry.owner} || 'Unknown Site'<br>
       12 Permit No.:<br>
       \${entry.permitNo} || 'N/A'<br>
       House ${entry.street} || 'Street not specified'<br>
       • Lat: ${parseFloat(entry.latitude).toFixed(5)},<br>
       Lng: ${parseFloat(entry.longitude).toFixed(5)}<br>
       Water Source: ${entry.type}<br>
       Calendar Year Inspected: ${entry.year_conducted} || 'N/A'<br>
       ✓ Status: ${sourceStatus}<br>
     `;

    if (entry.geotaggedUrl) {
      popupText += `
        <div style="margin-top: 5px;">
          <a href="${entry.geotaggedUrl}" target="_blank">
            
          </a>
        </div>
      `;
    }

    addMapMarker(
      entry.latitude,
      entry.longitude,
      popupText,
      entry.isWaterSource,
      entry.permitNo
    )
  }
}

```

```

    );
}

}

} else {
  console.warn(`⚠️ No entries found for "${normalizedBarangay}, ${normalizedCity}"`);
}

const mapModalInstance = new bootstrap.Modal(document.getElementById('mapModal'));
mapModalInstance.show();
}

async function handleViewUserMapClick(event) {
  if (!event.target.classList.contains('view-map-btn')) return;

  const entryId = event.target.getAttribute('data-id');

  const allData = await WUSData.fetchAll();
  const cachedPermits = await Permit.getAll(); // ✅ needed to find id

  const selectedEntry = allData.find(entry => entry.id === entryId);

  if (!selectedEntry) {
    console.warn(`⚠️ No entry found for ID: ${entryId}`);
    return;
  }

  const lat = parseFloat(selectedEntry.latitude);
  const lng = parseFloat(selectedEntry.longitude);

  document.getElementById('mapModalLabel').textContent =
    `${selectedEntry.owner || 'Unknown Site'} – ${selectedEntry.city || 'Unknown City'}`;

  clearExtraMarkers();

  updateMapLocation(
    `${selectedEntry.owner || 'Unknown Site'} – ${selectedEntry.city || 'Unknown City'}`,
    lat,
    lng
  );

  const sourceStatus = (selectedEntry.isWaterSource === true || selectedEntry.isWaterSource === 'true')
    ? '<span style="color:blue; font-weight:bold;">Water Source</span>'
    : '<span style="color:green; font-weight:bold;">Not a Water Source</span>';

  // ✅ Lookup permit.id first, then fetch full doc
  const matchedPermitMeta = cachedPermits.find(p => p.permitNo === selectedEntry.permitNo);
  let pdfUrl = 'images/permitNotFound.png';

  if (matchedPermitMeta?.id) {
    try {
      const fullPermit = await Permit.getById(matchedPermitMeta.id);
    }
  }
}

```

```

if (fullPermit?.pdfUrl) {
  pdfUrl = fullPermit.pdfUrl;
}
} catch (err) {
  console.warn(`Failed to fetch permit by ID: ${matchedPermitMeta.id}`, err.message);
}
}

let popupText = `
<strong>${selectedEntry.permittee || selectedEntry.owner || 'Unknown Site'}</strong><br>
💡 <strong>Permit No.:</strong>
<a href="${pdfUrl}" target="_blank">${selectedEntry.permitNo || 'N/A'}</a><br>
🏡 ${selectedEntry.street || 'Street not specified'}<br>
📍 Lat: ${lat.toFixed(5)}, Lng: ${lng.toFixed(5)}<br>
💧 Source: ${selectedEntry.type}<br>
📅 Year Inspected: ${selectedEntry.year_conducted || 'N/A'}<br>
✓ Status: ${sourceStatus}<br>
`;

if (selectedEntry.geotaggedUrl) {
  popupText += `
<div style="margin-top: 5px;">
<a href="${selectedEntry.geotaggedUrl}" target="_blank">

</a>
</div>
`;
}

addMapMarker(lat, lng, popupText, selectedEntry.isWaterSource, selectedEntry.permitNo);

const mapModalInstance = new bootstrap.Modal(document.getElementById('mapModal'));
mapModalInstance.show();
}

```

dashboard-init.js

```

import { SessionGuard } from "../auth/auth.js"
import { Sidebar } from "../components/sidebar.js";
import { Spinner } from "../components/spinner.js";
import { Users } from "../data/cache/user-data.js"
import { Consumable } from "../data/cache/consumable-data.js";
import { Ledger } from "../data/cache/ledger-data.js";
import { ICS } from "../data/cache/ics-data.js";
import { Permit } from "../data/cache/permit-data.js";
import { WUSData } from "../data/cache/wrus-data.js";
import { checkForAppUpdate } from "./version-control.js";
import { filterPermitsByCity } from "./permit-city-summary.js";
import { initYearlyWaterUserSummary } from "./yearly-summary.js";
import { initCityAccomplishmentSummary } from "./city-accomplishment-summary.js";

```

```

import { setupMapModal, initMap } from "../map/map-init.js";
import { initMapPrint } from "../map/mapPrint.js";
import { initializePasswordChange } from "./change-password.js";

document.addEventListener('DOMContentLoaded', () => {
  SessionGuard.ensureLoggedIn();
  refreshAllCachesEvery8Hours();
  Sidebar.render();
  Spinner.render();
  filterPermitsByCity();
  initYearlyWaterUserSummary();
  initCityAccomplishmentSummary();
  initMap("leafletMap", true, false);
  setupMapModal("mapModal", "leafletMap");
  initMapPrint("printMapBtn", "leafletMap");
  checkForAppUpdate();
  initializePasswordChange();
});

export async function refreshAllDailyCaches() {
  try {
    Spinner.show();

    await Promise.all([
      Users.autoRefreshDaily?(),
      Consumable.autoRefreshDaily?(),
      ICS.autoRefreshDaily?(),
      Permit.autoRefreshDaily?(),
      WUSData.autoRefreshDaily?()
    ]);
  } catch (error) {
    console.error("Error refreshing daily caches:", error);
  } finally {
    Spinner.hide();
  }
}

export async function refreshAllCachesEvery8Hours() {
  await Promise.all([
    Users.autoRefreshEvery8Hours?(),
    Consumable.autoRefreshEvery8Hours?(),
    ICS.autoRefreshEvery8Hours?(),
    Permit.autoRefreshEvery8Hours?(),
    WUSData.autoRefreshEvery8Hours?(),
    Ledger.autoRefreshEvery8Hours?()
  ]);
}

```

permit-city-summary.js

```

import { Permit } from "../data/cache/permit-data.js";
import { METRO_MANILA_CITIES } from "../data/constants/metroManilaCities.js";

```

```
export async function filterPermitsByCity() {
  const allPermits = await Permit.getAll();

  // Map of city names to possible abbreviation variants
  const CITY_ALIASES = {
    "Manila": ["mla", "manila"],
    "Makati": ["makati"],
    "Quezon City": ["quezon city", "qc", "q.c."],
    "Pasig": ["pasig"],
    "Pasay": ["pasay"],
    "Taguig": ["taguig"],
    "Marikina": ["marikina"],
    "Mandaluyong": ["mandaluyong"],
    "San Juan": ["san juan"],
    "Caloocan": ["caloocan"],
    "Malabon": ["malabon"],
    "Navotas": ["navotas"],
    "Valenzuela": ["valenzuela"],
    "Parañaque": ["paranaque", "parañaque"],
    "Las Piñas": ["las pinas", "las piñas"],
    "Muntinlupa": ["muntinlupa"],
    "Pateros": ["pateros"],
  };

  const cityCounts = Object.entries(CITY_ALIASES).reduce((acc, [city, aliases]) => {
    const matches = allPermits.filter(p => {
      const diversion = (p.diversionPoint || "").toLowerCase().normalize('NFD').replace(/[\u0300-\u036f]/g, "");

      return aliases.some(alias => {
        const normalizedAlias = alias.toLowerCase();
        const regex = new RegExp(`(?:^|\\W)${normalizedAlias}(?:\\W|$)`, 'i');
        return regex.test(diversion);
      });
    });

    if (matches.length > 0) {
      const visitedCount = matches.filter(p => p.visited === true && p.cancelled !== true).length;
      const cancelledCount = matches.filter(p => p.cancelled === true).length;
      const validCount = matches.length - cancelledCount;
      const percent = validCount > 0 ? ((visitedCount / validCount) * 100).toFixed(2) : "0.00";

      acc.push({
        city,
        total: matches.length,
        visited: visitedCount,
        cancelled: cancelledCount,
        percent
      });
    }
  }, []);

  return acc;
}, []);
}
```

```

    renderCityPermitTable(cityCounts);
}

function renderCityPermitTable(cityCounts) {
  const container = document.getElementById('cityPermitTable');
  if (!container) return;

  const today = new Date();
  const dateStr = today.toLocaleDateString(undefined, {
    year: 'numeric',
    month: 'long',
    day: 'numeric'
  });

  // Compute totals
  const totalPermits = cityCounts.reduce((sum, c) => sum + c.total, 0);
  const totalVisited = cityCounts.reduce((sum, c) => sum + c.visited, 0);
  const totalCancelled = cityCounts.reduce((sum, c) => sum + c.cancelled, 0);
  const validCount = totalPermits - totalCancelled;
  const totalPercent = validCount > 0 ? ((totalVisited / validCount) * 100).toFixed(2) : "0.00";

  let html = `
    <h3 class="mb-3">City / Municipality Permit Statistics <span class="fs-6 fw-normal"> — As of ${dateStr}</span></h3>
    <table class="table table-bordered table-striped align-middle text-center">
      <thead class="table-primary">
        <tr>
          <th>City / Municipality</th>
          <th>Total Permits</th>
          <th>Visited</th>
          <th>Cancelled</th>
          <th>Visited %</th>
        </tr>
      </thead>
      <tbody>
    `;

  cityCounts.forEach(({ city, total, visited, cancelled, percent }) => {
    html += `
      <tr>
        <td>${city}</td>
        <td>${total}</td>
        <td>${visited}</td>
        <td>${cancelled}</td>
        <td>
          <div class="city-progress-container">
            <div class="progress city-progress-bar" role="progressbar" title="${percent}% visited"
              aria-valuenow="${percent}" aria-valuemin="0" aria-valuemax="100">
              <div class="progress-bar bg-info" style="width: ${percent}%"></div>
            </div>
            <div class="progress-text">${percent}%</div>
          </div>
        </td>
      </tr>
    `;
  });
}

```

```

';
});

// Add Total row
html += `
<tr class="table-secondary fw-bold">
<td>Total</td>
<td>${totalPermits}</td>
<td>${totalVisited}</td>
<td>${totalCancelled}</td>
<td>
<div class="city-progress-container">
<div class="progress city-progress-bar" role="progressbar" title="${totalPercent}% visited"
aria-valuenow="${totalPercent}" aria-valuemin="0" aria-valuemax="100">
<div class="progress-bar bg-info" style="width: ${totalPercent}%"></div>
</div>
<div class="progress-text">${totalPercent}%</div>
</div>
</td>
</tr>
`;
html += `</tbody></table>;
container.innerHTML = html;
}

```

version-control.js

```

import { db } from "../firebaseConfig.js";
import {
getDoc,
doc
} from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio.js";
import { Confirmation } from "../components/notification.js";

// Local version in code
const localVersion = "1.0.3";

// Check for version update
export async function checkForAppUpdate() {
const versionDoc = doc(db, "config", "version");
const versionSnap = await getDoc(versionDoc);

if (versionSnap.exists()) {
const serverVersion = versionSnap.data().currentVersion;
if (serverVersion !== localVersion) {
showUpdatePrompt(serverVersion);
}
} else {
console.warn("No version info found in Firestore.");
}
}

```

```
// Use your custom Confirmation box
function showUpdatePrompt(serverVersion) {
  Confirmation.show(
    'A new version (${serverVersion}) is available. Reload now?',
    (confirmed) => {
      if (confirmed) {
        location.reload(true); // ⏮ Force refresh
      } else {
        NotificationBox.show("You are using an outdated version.", "warning", 5000);
      }
    }
  );
}
```

yearly-summary.js

```
import { WUSData } from "../data/cache/wrus-data.js";

export async function initYearlyWaterUserSummary() {
  const data = await WUSData.fetchAll();
  const container = document.getElementById('yearlyWaterUserSummary');
  if (!container) return;

  const grouped = {};
  let grandPermittees = 0;
  let grandNonPermittees = 0;
  let grandWaterSources = 0;

  data.forEach(entry => {
    const year = entry.year_conducted || 'Unknown';
    if (!grouped[year]) {
      grouped[year] = {
        permittees: 0,
        nonPermittees: 0,
        waterSources: 0
      };
    }

    const hasPermit = !!entry.permitNo;
    if (hasPermit) {
      grouped[year].permittees++;
      grandPermittees++;
    } else {
      grouped[year].nonPermittees++;
      grandNonPermittees++;
    }

    if (entry.remarks?.toLowerCase() === 'operational') {
      grouped[year].waterSources++;
      grandWaterSources++;
    }
  });
}
```

```

const today = new Date();
const dateStr = today.toLocaleDateString('en-US', {
  year: 'numeric',
  month: 'long',
  day: 'numeric'
});

const years = Object.keys(grouped).sort((a, b) => a - b);

let html = `
<h3 class="mb-3">WRUS Accomplishment — Water Users and Sources
<span class="fs-6 fw-normal"> — As of ${dateStr}</span>
</h3>
<table class="table table-bordered table-striped align-middle text-center rounded-0">
<thead class="table-success">
<tr>
<th rowspan="2">Year</th>
<th colspan="3">Water Users</th>
<th rowspan="2">Water Sources</th>
</tr>
<tr>
<th>Permittees</th>
<th>Non-Permittees</th>
<th>Total</th>
</tr>
</thead>
<tbody>
`;

years.forEach(year => {
  const { permittees, nonPermittees, waterSources } = grouped[year];
  const total = permittees + nonPermittees;

  html += `
<tr>
<td>${year}</td>
<td>${permittees}</td>
<td>${nonPermittees}</td>
<td>${total}</td>
<td>${waterSources}</td>
</tr>
`;
});

const grandTotalUsers = grandPermittees + grandNonPermittees;

html += `
<tr class="table-secondary fw-bold">
<td>Total</td>
<td>${grandPermittees}</td>
<td>${grandNonPermittees}</td>
<td>${grandTotalUsers}</td>
<td>${grandWaterSources}</td>
</tr>
`;

```

```

        </tr>
    </tbody>
</table>';

container.innerHTML = html;
}

```

XV.C.4.c. MEMIS Upload Module

excel-init.js

```

import { SessionGuard } from "../auth/auth.js";
import { Sidebar } from "../components/sidebar.js";
import { importExcelPermittees, importExcelNonPermittees } from "./upload-excel.js"

document.addEventListener("DOMContentLoaded", () => {
  SessionGuard.ensureLoggedIn();
  initializePage();
});

function initializePage(){
  Sidebar.render();
  importExcelNonPermittees();
  importExcelPermittees();
}

```

upload-excel.js

```

import { WUSData } from "../data/cache/wrus-data.js"
import { CoordinateUtils } from "../utils/coordinates.js";
import { NotificationBox, Confirmation } from "../components/notification.js";

//PERMITTEE
export function importExcelPermittees() {
  const logContainer = document.getElementById('importLogContainer');
  const logArea = document.getElementById('importLog');
  const loader = document.getElementById('importLoader');
  const warning = document.getElementById('importWarning');

  const appendLog = (message, type = 'info') => {
    const color = type === 'success' ? 'text-success' : type === 'error' ? 'text-danger' : 'text-secondary';
    logArea.innerHTML += `<div class="${color}">${message}</div>`;
    logArea.scrollTop = logArea.scrollHeight;
  };

  document.getElementById('importBtn').addEventListener('click', async () => {
    const file = document.getElementById('excelFile').files[0];
    const yearConducted = document.getElementById('yearConducted').value.trim();

    if (!file) {
      NotificationBox.show('Please select an Excel file first.', 'error');
      return;
    }
  });
}

```

```
}

if (!yearConducted || isNaN(yearConducted) || yearConducted.length !== 4) {
    NotificationBox.show('Please enter a valid 4-digit year.', 'error');
    return;
}

Confirmation.show(`Import data from Excel for year ${yearConducted}?`, async (confirmed) => {
    if (!confirmed) return;

    // Reset log UI
    logArea.innerHTML = '';
    logContainer.classList.remove('d-none');
    loader.classList.remove('d-none'); // Show loader
    warning.classList.remove('d-none');

    const reader = new FileReader();
    reader.onload = async (e) => {
        const data = new Uint8Array(e.target.result);
        const workbook = XLSX.read(data, { type: 'array' });
        const sheet = workbook.Sheets[workbook.SheetNames[0]];
        const rows = XLSX.utils.sheet_to_json(sheet, { header: 1, defval: '' });

        appendLog(` Total rows read: ${rows.length}`);

        for (let i = 7; i < rows.length; i++) {
            const row = rows[i];
            if (!row || row.length === 0 || row.every(cell => cell === "")) continue;

            try {
                const owner = row[0]?.toString().trim() || "";
                const representative = row[5]?.toString().trim() || "";
                const designation = row[6]?.toString().trim() || "";
                const phone = row[7]?.toString().trim() || "";
                const permitNo = row[10]?.toString().trim() || "";
                const street = row[18]?.toString().trim() || "";
                const barangay = row[19]?.toString().trim() || "";
                let city = row[20]?.toString().trim() || "";
                const type = row[28]?.toString().trim() || "";
                const status = row[29]?.toString().trim() || "";
                const date_inspected = row[34]?.toString().trim() || "";
                const remarks = row[38]?.toString().trim() || "";

                if (city.toLowerCase() === 'quezon') {
                    city = 'Quezon City';
                }

                const isWaterSource = /^operational$/i.test(status);

                const latDeg = parseFloat(row[22]) || 0;
                const latMin = parseFloat(row[23]) || 0;
                const latSec = parseFloat(row[24]) || 0;
                const latitude = CoordinateUtils.dmsToDecimal(latDeg, latMin, latSec, 'N');
            }
        }
    }
})
```

```
const longDeg = parseFloat(row[25]) || 0;
const longMin = parseFloat(row[26]) || 0;
const longSec = parseFloat(row[27]) || 0;
const longitude = CoordinateUtils.dmsToDecimal(longDeg, longMin, longSec, 'E');

const record = {
  owner,
  representative,
  designation,
  phone,
  permitNo,
  street,
  barangay,
  city,
  latitude,
  longitude,
  type,
  status,
  date_inspected,
  remarks,
  isWaterSource,
  year_conducted: yearConducted
};

await WUSData.add(record);
appendLog(`✓ Uploaded row ${i + 1}: ${owner}`, 'success');
} catch (err) {
  appendLog(`⚠ Skipped row ${i + 1} due to error: ${err.message}`, 'error');
}
}

loader.classList.add('d-none'); // Hide loader
warning.classList.add('d-none');
NotificationBox.show('Import complete!');
};

reader.readAsArrayBuffer(file));
});
};

//NON PERMITTEE

export function importExcelNonPermittees() {
  const logContainer = document.getElementById('importLogContainer');
  const logArea = document.getElementById('importLog');
  const loader = document.getElementById('importLoader');
  const warning = document.getElementById('importWarning');

  const appendLog = (message, type = 'info') => {
    const color =
      type === 'success'
```

```

? 'text-success'
: type === 'error'
? 'text-danger'
: 'text-secondary';
logArea.innerHTML += `<div class="${color}">${message}</div>`;
logArea.scrollTop = logArea.scrollHeight;
};

document
.getElementById('importBtnNonPerm')
.addEventListener('click', async () => {
const file = document.getElementById('excelFileNonPerm').files[0];
const year = document.getElementById('yearConductedNonPerm').value;

if (!file || !year) {
NotificationBox.show(
'⚠ Please select an Excel file and enter the year.',
'error'
);
return;
}

Confirmation.show(
`Are you sure you want to import Non-Permittees data for year ${year}?`,
(confirmed) => {
if (!confirmed) return;

// Reset log UI
logArea.innerHTML = "";
logContainer.classList.remove('d-none');
loader.classList.remove('d-none'); // Show loader
warning.classList.remove('d-none');

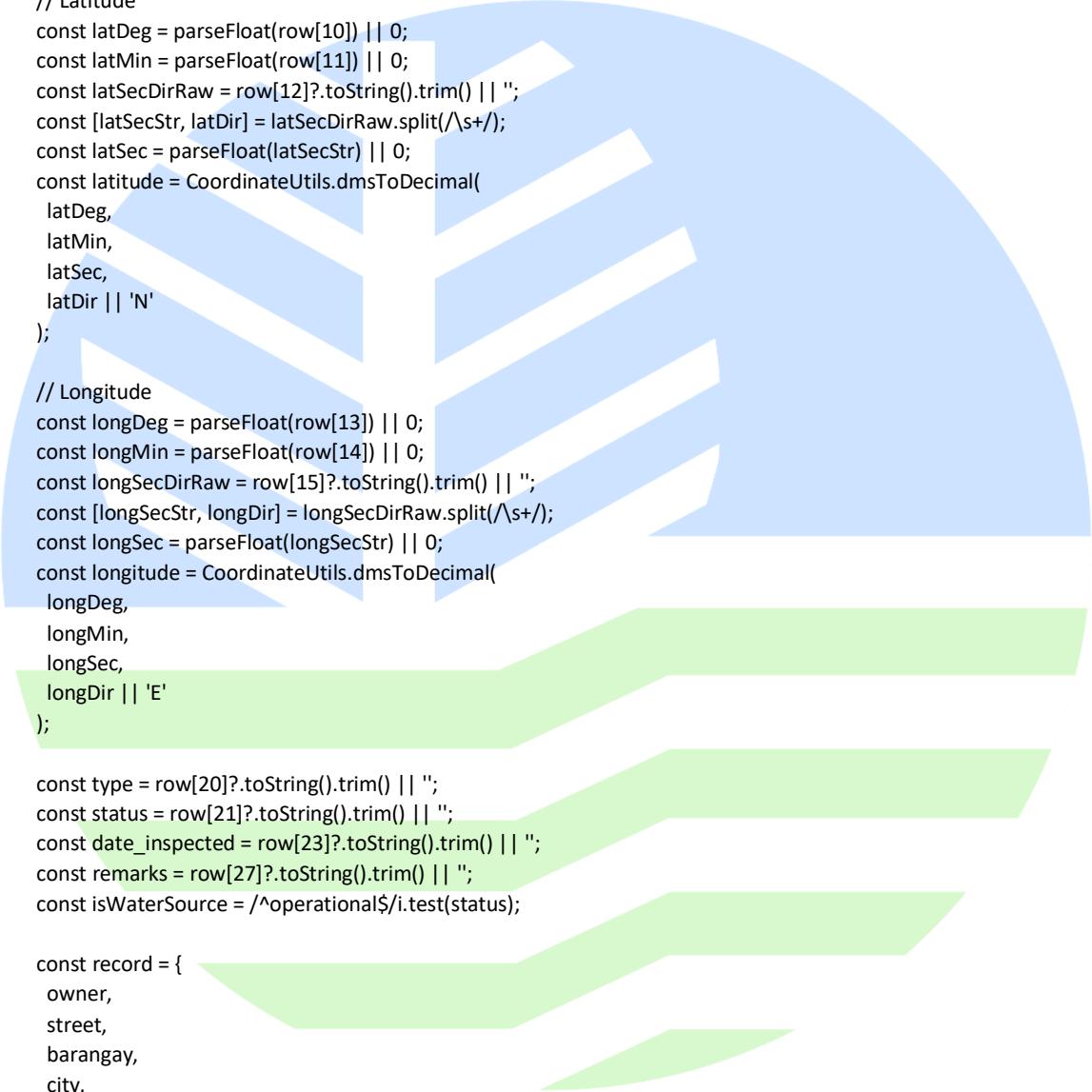
const reader = new FileReader();
reader.onload = async (e) => {
const data = new Uint8Array(e.target.result);
const workbook = XLSX.read(data, { type: 'array' });
const sheetName = workbook.SheetNames[0];
const sheet = workbook.Sheets[sheetName];
const rows = XLSX.utils.sheet_to_json(sheet, {
header: 1,
defval: "",
});
};

appendLog(`📄 Total rows read: ${rows.length}`);

for (let i = 7; i < rows.length; i++) {
const row = rows[i];
if (!row || row.length === 0 || row.every((cell) => cell === ""))
continue;

try {
const owner = row[0]?.toString().trim() || ";

```



```
const street = row[1]?.toString().trim() || "";
const barangay = row[2]?.toString().trim() || "";
let city = row[3]?.toString().trim() || "";
if (city.toLowerCase() === 'quezon') city = 'Quezon City';
const representative = row[5]?.toString().trim() || "";
const designation = row[6]?.toString().trim() || "";
const phone = row[7]?.toString().trim() || "";

// Latitude
const latDeg = parseFloat(row[10]) || 0;
const latMin = parseFloat(row[11]) || 0;
const latSecDirRaw = row[12]?.toString().trim() || "";
const [latSecStr, latDir] = latSecDirRaw.split(/\s+/);
const latSec = parseFloat(latSecStr) || 0;
const latitude = CoordinateUtils.dmsToDecimal(
  latDeg,
  latMin,
  latSec,
  latDir || 'N'
);

// Longitude
const longDeg = parseFloat(row[13]) || 0;
const longMin = parseFloat(row[14]) || 0;
const longSecDirRaw = row[15]?.toString().trim() || "";
const [longSecStr, longDir] = longSecDirRaw.split(/\s+/);
const longSec = parseFloat(longSecStr) || 0;
const longitude = CoordinateUtils.dmsToDecimal(
  longDeg,
  longMin,
  longSec,
  longDir || 'E'
);

const type = row[20]?.toString().trim() || "";
const status = row[21]?.toString().trim() || "";
const date_inspected = row[23]?.toString().trim() || "";
const remarks = row[27]?.toString().trim() || "";
const isWaterSource = /^operational$/i.test(status);

const record = {
  owner,
  street,
  barangay,
  city,
  representative,
  designation,
  phone,
  latitude,
  longitude,
  type,
  status,
  remarks,
```

```
isWaterSource,  
date_inspected,  
year_conducted: year,  
};  
  
await WUSData.add(record);  
appendLog(` ✅ Uploaded row ${i + 1}: ${owner} , 'success');  
} catch (err) {  
appendLog(  
` ⚠️ Skipped row ${i + 1} due to error: ${err.message} ,  
'error'  
);  
}  
}  
  
loader.classList.add('d-none'); // Hide loader  
warning.classList.add('d-none');  
NotificationBox.show('Import complete!');  
};  
  
reader.readAsArrayBuffer(file);  
}  
);  
});  
}  
}
```

XV.C.4.d. ICS Module

ics-init.js

```
import { SessionGuard } from '../auth/auth.js';  
import { initializePage } from './ics-ui.js'  
  
document.addEventListener('DOMContentLoaded', () => {  
SessionGuard.ensureLoggedIn();  
const userRole = localStorage.getItem('userRole');  
const userType = localStorage.getItem("userType");  
checkAccessAndInitialize(userType, userRole);  
});  
  
function checkAccessAndInitialize(userType, userRole) {  
if (userType !== "Permanent" && userRole !== 'admin') {  
const pageContent = document.getElementById("page-content");  
if (pageContent) {  
initializePage();  
pageContent.innerHTML = "<p class='text-danger'>Access denied.</p>";  
}  
} else {  
initializePage();  
}  
}
```

ics-ui.js

```
import { FileService } from './upload/upload.js';
import { ICS } from './data/cache/ics-data.js';
import { Users } from './data/cache/user-data.js'
import { Sidebar } from './components/sidebar.js'
import { Spinner } from './components/spinner.js';
import { NotificationBox, Confirmation } from './components/notification.js';

let currentPage = 1;
let rowsPerPage = 10;
let currentData = []; // All data fetched from Firestore
let filteredData = [] // Data after filtering
let usersMapGlobal = {} // For use in renderFilteredTable

// 🔳 Show modal and load users
export function initializePage(){
  Sidebar.render();
  Spinner.render();
  setupAddBtn();
  setupAddQtyAndCostListeners();
  setupEditQtyAndCostListeners();
  setupFileValidation();
  setupICSFomSubmit();
  renderICSTable();
  initDeleteICSBttn();
  document.getElementById("searchBar").addEventListener("input", applySearchFilter);
  setupEditICSFomSubmit();
  handleRefreshButton({
    buttonId: "refreshBttn",
    refreshFn: ICS.refreshCache.bind(ICS),
    renderFn: renderICSTable,
    cooldownKey: "lastICSRrefresh",
    cooldownSeconds: 60
  });
}

function normalizeText(text) {
  return text
    .toLowerCase()
    .normalize("NFD")
    .replace(/[\u0300-\u036f]/g, "");
}

function highlightMatch(text, term) {
  if (!term) return text;

  const normalizedText = normalizeText(text);
  const normalizedTerm = normalizeText(term);

  const index = normalizedText.indexOf(normalizedTerm);
  if (index === -1) return text;
```

```

// Approximate actual match range
const regex = new RegExp(`(${term})`, "i");
return text.replace(regex, `<mark>$1</mark>`);
}

function setupAddBtn() {
  const addBtn = document.getElementById('addBtn');
  if (addBtn) {
    addBtn.addEventListener('click', () => {
      const modal = new bootstrap.Modal(document.getElementById('addICSModal'));
      modal.show();
      loadUsers();
    });
  }
}

function loadUsers(selectElementId = 'assignedTo', selectedUserId = '') {
  return new Promise((resolve) => {
    const select = document.getElementById(selectElementId);
    select.innerHTML = '<option value="">Select User</option>';

    const currentUserId = localStorage.getItem('wrusUserId');

    Users.fetchAllAsc().then(users => {
      // Admin case: populate full list
      if (currentUserId === 'admin') {
        users.forEach(user => {
          if (user.status === 'active' && user.type === 'Permanent') {
            const option = document.createElement('option');
            option.value = user.id;
            option.textContent = `${user.lastName}, ${user.firstName} ${user.middleInitial}.`;
            if (user.id === selectedUserId) {
              option.selected = true;
            }
            select.appendChild(option);
          }
        });
      }
      select.disabled = false; // Make sure admin dropdown is enabled
    } else {
      // Regular user: only load their own option
      const currentUser = users.find(user => user.id === currentUserId);

      if (currentUser && currentUser.status === 'active' && currentUser.type === 'Permanent') {
        const option = document.createElement('option');
        option.value = currentUser.id;
        option.textContent = `${currentUser.lastName}, ${currentUser.firstName} ${currentUser.middleInitial}.`;
        option.selected = true;
        select.appendChild(option);
      }
      select.disabled = true; // Lock dropdown for regular users
    }
  });
}

```

```

        resolve();
    });
});
}

function setupAddQtyAndCostListeners() {
    const qtyInput = document.getElementById('qty');
    const unitCostInput = document.getElementById('unitCost');

    if (qtyInput && unitCostInput) {
        qtyInput.addEventListener('input', updateAddTotalCost);
        unitCostInput.addEventListener('input', updateAddTotalCost);
    }
}

function updateAddTotalCost() {
    const qty = parseFloat(document.getElementById('qty').value) || 0;
    const unitCost = parseFloat(document.getElementById('unitCost').value) || 0;
    const total = qty * unitCost;

    document.getElementById('totalCostDisplay').textContent =
        `₱${total.toLocaleString('en-PH', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`;
}

function setupEditQtyAndCostListeners() {
    const qtyInput = document.getElementById('editQty');
    const unitCostInput = document.getElementById('editUnitCost');

    if (qtyInput && unitCostInput) {
        qtyInput.addEventListener('input', updateEditTotalCost);
        unitCostInput.addEventListener('input', updateEditTotalCost);
    }
}

function updateEditTotalCost() {
    const qty = parseFloat(document.getElementById('editQty').value) || 0;
    const unitCost = parseFloat(document.getElementById('editUnitCost').value) || 0;
    const total = qty * unitCost;

    document.getElementById('editTotalCostDisplay').textContent =
        `₱${total.toLocaleString('en-PH', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`;
}

function setupFileValidation() {
    const fileInput = document.getElementById('attachment');
    if (fileInput) {
        fileInput.addEventListener('change', function () {
            const file = this.files[0];
            if (file && file.size > 1048576) {
                NotificationBox.show("File exceeds 1MB limit.", "error");
                this.value = "";
            }
        });
    }
}

```

```
});

}

function setupICSFormSubmit() {
  const addICSForm = document.getElementById('addICSForm');
  if (addICSForm) {
    addICSForm.addEventListener('submit', handleICSFormSubmit);
  }
}

async function handleICSFormSubmit(e) {
  e.preventDefault();

  const form = document.getElementById('addICSForm');

  // Trigger Bootstrap validation UI
  form.classList.add('was-validated');

  // STOP if form is invalid
  if (!form.checkValidity()) {
    Spinner.hide();
    return;
  }

  Spinner.show();

  const fileInput = document.getElementById('attachment');
  let fileURL = null;

  try {
    if (fileInput.files.length > 0) {
      const file = fileInput.files[0];
      fileURL = await FileService.uploadICSFfile(file);
      if (!fileURL) throw new Error("File upload failed");
    }
  }

  const icsData = collectICSFormData();
  if (fileURL) {
    icsData.attachmentURL = fileURL;
  }

  await ICS.add(icsData);

  form.reset();
  form.classList.remove('was-validated'); // Reset validation state
  bootstrap.Modal.getOrCreateInstance(document.getElementById('addICSModal')).hide();
  document.getElementById('editAttachment').value = '';
  document.getElementById('attachment').value = '';

  renderICSTable();
} catch (err) {
  console.error(err);
}
```

```
NotificationBox.show("Failed to save. Please try again.");
} finally {
    Spinner.hide();
}
}

function collectICSFFormData() {
    return {
        ICSNo: document.getElementById('icsNo').value.trim(),
        serialNo: document.getElementById('serialNumber').value.trim(),
        assignedTo: document.getElementById('assignedTo').value.trim(),
        description: document.getElementById('description').value.trim(),
        qty: parseInt(document.getElementById('qty').value),
        unit: document.getElementById('unit').value.trim(),
        unitCost: parseFloat(document.getElementById('unitCost').value),
        totalCost: parseFloat(document.getElementById('totalCostDisplay').textContent.replace(/[₽]/g, "")) || 0,
        dateIssued: document.getElementById('dateIssued').value,
        remarks: document.getElementById('remarks').value.trim(),
        status: document.getElementById('status').value,
        attachmentURL: "", // optional default
    };
}

async function renderICSTable(dataSet = null, page = 1, searchTerm = '') {
    Spinner.show();

    const userId = localStorage.getItem("wrusUserId");
    const tableBody = document.querySelector("#icsTableBody");
    tableBody.innerHTML = "";

    try {
        // Fetch new data if not passed in
        if (!dataSet) {
            const [usersMap, icsSnapshot] = await Promise.all([
                Users.getUsersMap(),
                ICS.fetchAll()
            ]);
            usersMapGlobal = usersMap;
            currentData = icsSnapshot;
        }

        let dataToUse = dataSet || currentData;
        const usersMap = usersMapGlobal;

        // Filter by assigned user if not admin
        if (userId !== "admin") {
            dataToUse = dataToUse.filter(entry => entry.data.assignedTo === userId);
        }

        // Reapply search filter if term is provided
        const query = normalizeText(searchTerm.trim());
        if (query) {
            dataToUse = dataToUse.filter(entry => {

```

```

const { ICSno, assignedTo, dateIssued, description } = entry.data;
const assignedName = usersMap[assignedTo] || "Unknown User";
return (
  normalizeText(ICSno || "").includes(query) ||
  normalizeText(assignedName).includes(query) ||
  normalizeText(dateIssued || "").includes(query) ||
  normalizeText(description || "").includes(query)
);
});
};

}

if (dataToUse.length === 0) {
  tableBody.innerHTML = "<tr><td colspan='6'>No ICS entries found.</td></tr>";
  return;
}

// Pagination
currentPage = page;
const startIndex = (page - 1) * rowsPerPage;
const paginatedItems = dataToUse.slice(startIndex, startIndex + rowsPerPage);

// Build table rows
const rowsHtml = paginatedItems.map(entry => {
  const { ICSno, description, dateIssued, assignedTo, attachmentURL, totalCost } = entry.data;
  const assignedName = usersMap[assignedTo] || "Unknown User";

  const formattedCost = typeof totalCost === 'number'
    ? `₱${totalCost.toLocaleString('en-PH', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`
    : '₱0.00';

  return `
<tr>
<td>${highlightMatch(ICSno || '(no ICSno)', searchTerm)}</td>
<td>${highlightMatch(assignedName, searchTerm)}</td>
<td>${highlightMatch(description || "", searchTerm)}</td>
<td>${highlightMatch(dateIssued || "", searchTerm)}</td>
<td>${formattedCost}</td>
<td>
<div class="d-flex gap-1">
<button class="btn btn-3d btn-sm btn-primary flex-fill d-flex align-items-center justify-content-center" data-id="${entry.id}">
  <i class="bi bi-pencil-square"></i>
</button>
<a href="${attachmentURL || '#'}" target="_blank"
  class="btn btn-3d btn-sm btn-secondary flex-fill d-flex align-items-center justify-content-center"
  data-id="${entry.id}">
  <i class="bi bi-eye"></i>
</a>
</div>
</td>
</tr>`;
}).join("");
}

```

```
tableBody.innerHTML = rowsHtml;

// Edit button handlers
document.querySelectorAll(".btn-primary[data-id]").forEach(button => {
  button.addEventListener("click", () => {
    const docId = button.getAttribute("data-id");
    const icsItem = currentData.find(item => item.id === docId);
    if (icsItem) {
      populateEditModal(icsItem);
      new bootstrap.Modal(document.getElementById('editICSModal')).show();
    }
  });
});

renderPaginationControls(dataToUse, page);

} catch (err) {
  console.error("Error rendering ICS table:", err);
  tableBody.innerHTML = "<tr><td colspan='6'>Error loading data.</td></tr>";
} finally {
  Spinner.hide();
}
}

function handleRefreshButton({
  buttonId,
  refreshFn,
  renderFn,
  cooldownKey,
  cooldownSeconds = 60
}) {
  if (!buttonId || !refreshFn || !renderFn || !cooldownKey) {
    console.error("handleRefreshButton: Missing required arguments.");
    return;
  }

  const refreshBtn = document.getElementById(buttonId);
  if (!refreshBtn) {
    console.error(`handleRefreshButton: Button with ID "${buttonId}" not found.`);
    return;
  }

  const originalText = refreshBtn.textContent;
  let cooldownTimer = null;

  function startCooldown() {
    const startTime = Date.now();
    const endTime = startTime + cooldownSeconds * 1000;
    localStorage.setItem(cooldownKey, startTime.toString());

    refreshBtn.disabled = true;

    cooldownTimer = setInterval(() => {
```

```

const now = Date.now();
const secondsLeft = Math.ceil((endTime - now) / 1000);

if (secondsLeft <= 0) {
  clearInterval(cooldownTimer);
  refreshBtn.disabled = false;
  refreshBtn.textContent = originalText;
} else {
  refreshBtn.textContent = `Wait (${secondsLeft}s)`;  

}
}, 1000);
}

// Resume countdown if cooldown is still active
const lastRefresh = parseInt(localStorage.getItem(cooldownKey)) || 0;
const now = Date.now();
if (now - lastRefresh < cooldownSeconds * 1000) {
  startCooldown();
}

refreshBtn.addEventListener("click", async () => {
  const now = Date.now();
  const lastRefresh = parseInt(localStorage.getItem(cooldownKey)) || 0;

  if (now - lastRefresh < cooldownSeconds * 1000) {
    return;
  }

  try {
    await refreshFn();
    renderFn();
    ICS.refreshCache();
    const searchTerm = document.getElementById("searchBar").value.trim();
    renderICSTable(null, 1, searchTerm);
    NotificationBox.show("Refreshed successfully.");
    startCooldown();
  } catch (err) {
    console.error("Refresh error:", err);
    NotificationBox.show("Failed to refresh data.", "error");
  }
});  

}

function renderPaginationControls(dataSet, page) {
  const totalPages = Math.ceil(dataSet.length / rowsPerPage);
  const pagination = document.getElementById("paginationControlsICS");
  pagination.innerHTML = "";

  // Calculate page range (max 5 pages at a time)
  const maxVisiblePages = 5;
  let startPage = Math.max(1, page - Math.floor(maxVisiblePages / 2));
  let endPage = startPage + maxVisiblePages - 1;
}

```

```

if (endPage > totalPages) {
    endPage = totalPages;
    startPage = Math.max(1, endPage - maxVisiblePages + 1);
}

// Previous Button
const prevItem = document.createElement("li");
prevItem.className = `page-item ${page === 1 ? "disabled" : ""}`;
prevItem.innerHTML = '<a class="page-link" href="#">Previous</a>';
prevItem.addEventListener("click", (e) => {
    e.preventDefault();
    if (page > 1) {
        renderICSTable(null, page - 1);
    }
});
pagination.appendChild(prevItem);

// Page Numbers
for (let i = startPage; i <= endPage; i++) {
    const pagelitem = document.createElement("li");
    pagelitem.className = `${i === page ? "active" : ""}`;
    pagelitem.innerHTML = '<a class="page-link" href="#">${i}</a>';
    pagelitem.addEventListener("click", (e) => {
        e.preventDefault();
        renderICSTable(null, i);
    });
    pagination.appendChild(pagelitem);
}

// Next Button
const nextItem = document.createElement("li");
nextItem.className = `page-item ${page === totalPages ? "disabled" : ""}`;
nextItem.innerHTML = '<a class="page-link" href="#">Next</a>';
nextItem.addEventListener("click", (e) => {
    e.preventDefault();
    if (page < totalPages) {
        renderICSTable(null, page + 1);
    }
});
pagination.appendChild(nextItem);

function applySearchFilter() {
    const rawQuery = document.getElementById("searchBar").value.trim();
    renderICSTable(null, 1, rawQuery);
}

async function populateEditModal(icslItem) {
    const data = iclItem.data;

    document.getElementById("editDocId").value = iclItem.id;
    document.getElementById('editIcsNo').value = data.ICSno || "";
    document.getElementById('editSerialNumber').value = data.serialNo || "";
}

```

```

document.getElementById('editDescription').value = data.description || '';
document.getElementById('editQty').value = data.qty || '';
document.getElementById('editUnit').value = data.unit || '';
document.getElementById('editUnitCost').value = data.unitCost || '';
document.getElementById('editRemarks').value = data.remarks || '';
document.getElementById('editDateIssued').value = data.dateIssued || '';
document.getElementById('editStatus').value = data.status || '';
document.getElementById('editCurrentAttachmentUrl').value = data.attachmentURL || '';

// Compute and show total cost
const total = (data.qty || 0) * (data.unitCost || 0);
document.getElementById('editTotalCostDisplay').textContent =
`₱${total.toLocaleString('en-PH', { minimumFractionDigits: 2, maximumFractionDigits: 2 }})`;

// Populate the dropdown and pre-select user
await loadUsers('editAssignedTo', data.assignedTo);
}

function setupEditICSFormSubmit() {
const editICSForm = document.getElementById('editICSForm');
if (editICSForm) {
  editICSForm.addEventListener('submit', handleEditICSSubmit);
}
}

async function handleEditICSSubmit(e) {
e.preventDefault();

const form = document.getElementById('editICSForm');
if (!form.checkValidity()) {
  form.classList.add('was-validated');
  return;
}

Spinner.show();

const docId = document.getElementById('editDocId').value;
if (!docId) {
  NotificationBox.show("Document ID missing. Please reload and try again.", "error");
  Spinner.hide();
  return;
}

const fileInput = document.getElementById('editAttachment');
const newFile = fileInput.files[0];
let attachmentURL = document.getElementById('editCurrentAttachmentUrl').value;

if (newFile) {
  if (attachmentURL) {
    const deleted = await FileService.deleteFileFromStorage(attachmentURL);
    if (!deleted) {
      console.warn("Old file may not have been deleted properly.");
    }
  }
}
}

```

```

}

const uploadedUrl = await FileService.uploadICSCFile(newFile);
if (!uploadedUrl) {
  NotificationBox.show('New file upload failed.');
  Spinner.hide();
  return;
}
attachmentURL = uploadedUrl;
}

const updatedData = {
  ICSNo: document.getElementById('editIcsNo').value.trim(),
  serialNo: document.getElementById('editSerialNumber').value.trim(),
  assignedTo: document.getElementById('editAssignedTo').value.trim(),
  description: document.getElementById('editDescription').value.trim(),
  qty: parseInt(document.getElementById('editQty').value),
  unit: document.getElementById('editUnit').value.trim(),
  unitCost: parseFloat(document.getElementById('editUnitCost').value),
  totalCost: parseFloat(document.getElementById('editTotalCostDisplay').textContent.replace(/[₽,]/g, " ")) || 0,
  dateIssued: document.getElementById('editDateIssued').value,
  remarks: document.getElementById('editRemarks').value.trim(),
  status: document.getElementById('editStatus').value,
  attachmentURL
};

try {
  NotificationBox.show("ICS entry update successful");
  await ICS.update(docId, updatedData);
  bootstrap.Modal.getOrCreateInstance(document.getElementById('editICSModal')).hide();
  document.getElementById('editAttachment').value = "";
  document.getElementById('attachment').value = "";
  const searchTerm = document.getElementById('searchBar').value.trim();
  renderICSTable(null, currentPage, searchTerm);
} catch (err) {
  console.error("❌ Failed to update ICS entry:", err);
  NotificationBox.show("Update failed. Check console for details.", "error");
} finally {
  Spinner.hide();
  form.classList.remove('was-validated');
}

function initDeleteICSCButton() {
  const deleteButton = document.getElementById("deleteICSCButton");

  if (!deleteButton) {
    console.warn("Delete button not found in DOM.");
    return;
  }

  deleteButton.addEventListener("click", () => {
    const docId = document.getElementById("editDocId")?.value;
  });
}

```

```

const attachmentURL = document.getElementById("editCurrentAttachmentUrl")?.value;

if (!docId) {
    console.error("No docId found for deletion.");
    return;
}

Confirmation.show(
    "Are you sure you want to delete this ICS entry?\nThis action is irreversible.",
    async (confirmed) => {
        if (!confirmed) return;

        Spinner.show();
        try {
            // Delete attachment if exists
            if (attachmentURL) {
                const deleted = await FileService.deleteFileFromStorage(attachmentURL);
                if (!deleted) {
                    console.warn("Attachment might not have been deleted.");
                }
            }
        }

        // Delete ICS document
        await ICS.delete(docId);

        NotificationBox.show("ICS entry deleted successfully");
        bootstrap.Modal.getInstance(document.getElementById("editICSModal"))?.hide();
        await renderICSTable();
    } catch (error) {
        console.error("Failed to delete ICS entry:", error);
        NotificationBox.show("Failed to delete ICS entry.", "error");
    } finally {
        Spinner.hide();
    }
});
});
}

```

XV.C.4.e. Map Module

cityBoundaries.js

```

export const cityBoundaries = {
    manila: { geoJson: "js/data/geojson/manila.geojson", color: "#e41a1c", label: "Manila" },
    mandaluyong: { geoJson: "js/data/geojson/mandaluyong.geojson", color: "#377eb8", label: "Mandaluyong" },
    marikina: { geoJson: "js/data/geojson/marikina.geojson", color: "#4daf4a", label: "Marikina" },
    pasig: { geoJson: "js/data/geojson/pasig.geojson", color: "#984ea3", label: "Pasig" },
    quezonCity: { geoJson: "js/data/geojson/quezon_city.geojson", color: "#ff7f00", label: "Quezon City" },
    sanJuan: { geoJson: "js/data/geojson/san_juan.geojson", color: "#ffff33", label: "San Juan" },
    caloocan: { geoJson: "js/data/geojson/caloocan.geojson", color: "#a65628", label: "Caloocan" },
    malabon: { geoJson: "js/data/geojson/malabon.geojson", color: "#f781bf", label: "Malabon" },
    navotas: { geoJson: "js/data/geojson/navotas.geojson", color: "#999999", label: "Navotas" },
}

```

```

valenzuela: { geoJson: "js/data/geojson/valenzuela.geojson", color: "#66c2a5", label: "Valenzuela" },
lasPinas: { geoJson: "js/data/geojson/las_piñas.geojson", color: "#fc8d62", label: "Las Piñas" },
muntinlupa: { geoJson: "js/data/geojson/muntinlupa.geojson", color: "#8da0cb", label: "Muntinlupa" },
parañaque: { geoJson: "js/data/geojson/parañaque.geojson", color: "#e78ac3", label: "Parañaque" },
makati: { geoJson: "js/data/geojson/makati.geojson", color: "#a6d854", label: "Makati" },
pasay: { geoJson: "js/data/geojson/pasay.geojson", color: "#ffd92f", label: "Pasay" },
taguig: { geoJson: "js/data/geojson/taguig.geojson", color: "#e5c494", label: "Taguig" },
pateros: { geoJson: "js/data/geojson/pateros.geojson", color: "#b3b3b3", label: "Pateros" }
};


```

map-init.js

```

import { cityBoundaries } from "./cityBoundaries.js";

let mapInitialized = false;
export let map;
let marker;
let extraMarkers = [];
let geoWatchId = null;
let liveStartMarker = null;
let routeZoomed = false;
let livePulseMarker = null;

export function initMap(mapDivId = "leafletMap", showLegend = true, enableGeolocation = true) {
  if (mapInitialized) return;

  map = L.map(mapDivId).setView([14.6091, 121.0223], 12);

  L.tileLayer("https://s.basemaps.cartocdn.com/dark_all/{z}/{x}/{y}{r}.png", {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors &copy; <a href="https://www.carto.com/">CARTO</a>',
    subdomains: "abcd",
    maxZoom: 19
  }).addTo(map);

  //  Only run geolocation if enabled
  if (enableGeolocation && navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      (position) => {
        const userLat = position.coords.latitude;
        const userLng = position.coords.longitude;

        map.setView([userLat, userLng], 14);

        marker = L.marker([userLat, userLng]).addTo(map)
          .bindPopup("📍 You are here")
          .openPopup();
      },
      () => {
        console.warn("Geolocation failed. Using default location.");
        marker = L.marker([14.6091, 121.0223]).addTo(map)
          .bindPopup("National Capital Region")
          .openPopup();
      }
    );
  }
}


```

```

        }
    );
} else if (!enableGeolocation) {
    console.warn("📍 Geolocation is disabled. Using default location.");
    marker = L.marker([14.6091, 121.0223]).addTo(map)
        .bindPopup("National Capital Region")
        .openPopup();
} else {
    console.warn("Geolocation is not supported by this browser.");
    marker = L.marker([14.6091, 121.0223]).addTo(map)
        .bindPopup("National Capital Region")
        .openPopup();
}

// Only add legend if requested
if (showLegend) {
    legend.addTo(map);
}

// Load boundaries
loadBarangayBoundaries("js/data/geojson/Barangays_NCR.geojson");

Object.keys(cityBoundaries).forEach(loadCityBoundary);

mapInitialized = true;
}

export function setupMapModal(modalId = "mapModal", mapDivId = "leafletMap") {
    const mapModal = document.getElementById(modalId);

    if (!mapModal) {
        console.error(`Modal with ID "${modalId}" not found.`);
        return;
    }

    mapModal.addEventListener("shown.bs.modal", function () {
        if (!mapInitialized) {
            initMap(mapDivId);
        } else {
            map.invalidateSize();
        }
    });
}

export function updateMapLocation(city, lat, lng) {
    if (!map) return;

    map.setView([lat, lng], 13);

    if (marker) {
        marker.setLatLng([lat, lng]).bindPopup(city).openPopup();
    } else {
        marker = L.marker([lat, lng]).addTo(map).bindPopup(city).openPopup();
    }
}

```

```

}

// Define blue & green markers (using Leaflet default icon colors)
const blueIcon = new L.Icon({
  iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-blue.png',
  shadowUrl: 'https://unpkg.com/leaflet@1.9.4/dist/images/marker-shadow.png',
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  shadowSize: [41, 41]
});

const greenIcon = new L.Icon({
  iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-green.png',
  shadowUrl: 'https://unpkg.com/leaflet@1.9.4/dist/images/marker-shadow.png',
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  shadowSize: [41, 41]
});

const violetIcon = new L.Icon({
  iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-violet.png',
  shadowUrl: 'https://unpkg.com/leaflet@1.9.4/dist/images/marker-shadow.png',
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  shadowSize: [41, 41]
});

const legend = L.control({ position: 'bottomright' });

legend.onAdd = function () {
  const div = L.DomUtil.create('div', 'info legend');
  div.style.background = 'white';
  div.style.padding = '8px';
  div.style.border = '1px solid #ccc';
  div.style.borderRadius = '6px';
  div.style.fontSize = '14px';
  div.style.lineHeight = '18px';

  div.innerHTML =
    `Legend  

      Permittee  

      Water Source  

      Water User
  `;

  return div;
};

```

```
export function addMapMarker(lat, lng, label, isWaterSource, permitNo) {
  if (!map) return;

  let markerIcon;

  if (permitNo && permitNo.trim() !== "") {
    markerIcon = violetIcon;
  } else {
    markerIcon = isWaterSource ? blueIcon : greenIcon;
  }

  const m = L.marker([lat, lng], { icon: markerIcon })
    .addTo(map)
    .bindPopup(label || "Unknown Site");

  extraMarkers.push(m);
}

export function clearExtraMarkers() {
  if (!map) return;

  extraMarkers.forEach(m => map.removeLayer(m));
  extraMarkers = [];
}

function loadBarangayBoundaries(geoJsonUrl = "Barangays_NCR.geojson") {
  if (!map) return;

  fetch(geoJsonUrl)
    .then(response => response.json())
    .then(data => {
      L.geoJSON(data, {
        style: {
          color: "#ffffff",
          weight: 0.9,
          opacity: 1,
          fillOpacity: 0
        },
        onEachFeature: (feature, layer) => {
          const brgy = feature.properties.BRGY_NAME || "Unknown Barangay";
          const city = feature.properties.ADM2_EN || "Unknown City";
          layer.bindPopup(`<strong>${brgy}</strong><br><em>${city}</em>`);

          // Add barangay label
          const labelText = feature.properties.ADM4_EN;
          const coords = feature.geometry.coordinates;
          let latlng;

          if (feature.geometry.type === "Polygon") {
            latlng = getPolygonCentroid(coords[0]);
          } else if (feature.geometry.type === "MultiPolygon") {
            latlng = getPolygonCentroid(coords[0][0]);
          }
        }
      });
    })
}
```

```

    }

    if (latlng) {
      const label = L.divIcon({
        className: "barangay-label",
        html: labelText,
        iconSize: null
      });

      const marker = L.marker(latlng, { icon: label, interactive: false });

      if (map.getZoom() >= 14) marker.addTo(map);

      if (!map._barangayLabels) map._barangayLabels = [];
      map._barangayLabels.push(marker);
    }
  });
}).addTo(map);

// Show/hide labels based on zoom level
map.on("zoomend", () => {
  if (!map._barangayLabels) return;
  const showLabels = map.getZoom() >= 14;

  map._barangayLabels.forEach(marker => {
    if (showLabels && !map.hasLayer(marker)) {
      marker.addTo(map);
    } else if (!showLabels && map.hasLayer(marker)) {
      map.removeLayer(marker);
    }
  });
});
}).catch(err => console.error("Error loading GeoJSON:", err));
}

function loadCityBoundary(cityKey) {
  if (!map || !cityBoundaries[cityKey]) return;

  const { geoJson, color, label } = cityBoundaries[cityKey];

  fetch(geoJson)
    .then(response => response.json())
    .then(data => {
      const boundaryLayer = L.geoJSON(data, {
        style: {
          color,
          weight: 5,
          opacity: 1,
          fillOpacity: 0
        },
        onEachFeature: (feature, layer) => {
          const brgy = feature.properties.BRGY_NAME || "Unknown Barangay";
        }
      }).addTo(map);
    });
}

```

```

        const city = feature.properties.ADM2_EN || "Unknown City";
        layer.bindPopup('<strong>${brgy}</strong><br><em>${city}</em>');
    }
}).addTo(map);

const center = boundaryLayer.getBounds().getCenter();

const cityLabel = L.divIcon({
    className: `${cityKey}-label`,
    html: `<div style="font-size: 24px; font-weight: bold; color: ${color};">${label}</div>`,
    iconSize: [180, 40]
});

L.marker(center, { icon: cityLabel }).addTo(map);
})
.catch(err => console.error(`Error loading ${geoJson}:`, err));
}

function getPolygonCentroid(coords) {
    let area = 0, x = 0, y = 0;
    for (let i = 0, j = coords.length - 1; i < coords.length; j = i++) {
        const [x0, y0] = coords[j];
        const [x1, y1] = coords[i];
        const f = x0 * y1 - x1 * y0;
        area += f;
        x += (x0 + x1) * f;
        y += (y0 + y1) * f;
    }
    area *= 0.5;
    return [y / (6 * area), x / (6 * area)];
}

let liveHeadingMarker = null; // ✨ New global variable to track heading marker

export function plotRouteOnMap(startLat, startLng, endLat, endLng, permitInfo = null, resetRoute = true, headingDeg = null) {
    if (!map) return;

    if (resetRoute) {
        clearExtraMarkers();
        routeZoomed = false;
    }

    // ✅ START MARKER (Main Icon)
    if (resetRoute) {
        liveStartMarker = L.marker([startLat, startLng], { icon: greenIcon })
            .addTo(map)
            .bindPopup("📍 Starting Point")
            .openPopup();
        extraMarkers.push(liveStartMarker);
    }

    // ✅ Pulsating effect overlay
    livePulseMarker = L.marker([startLat, startLng], {

```

```

icon: L.divIcon({
  className: "pulse-marker",
  iconSize: [20, 20]
}),
interactive: false
}).addTo(map);
extraMarkers.push(livePulseMarker);

// ✅ Heading arrow marker (NEW)
liveHeadingMarker = L.marker([startLat, startLng], {
  icon: L.divIcon({
    className: "heading-arrow",
    html: `<div class="heading-icon" style="transform: rotate(${headingDeg || 0}deg);></div>`,
    iconSize: [30, 30],
    iconAnchor: [15, 35], // ← push the arrow up
  }),
  interactive: false
}).addTo(map);
extraMarkers.push(liveHeadingMarker);

} else {
  // ✅ Move start marker
  if (liveStartMarker?.slideTo) {
    liveStartMarker.slideTo([startLat, startLng], { duration: 800, keepAtCenter: false });
  } else if (liveStartMarker) {
    liveStartMarker.setLatLng([startLat, startLng]);
  }
}

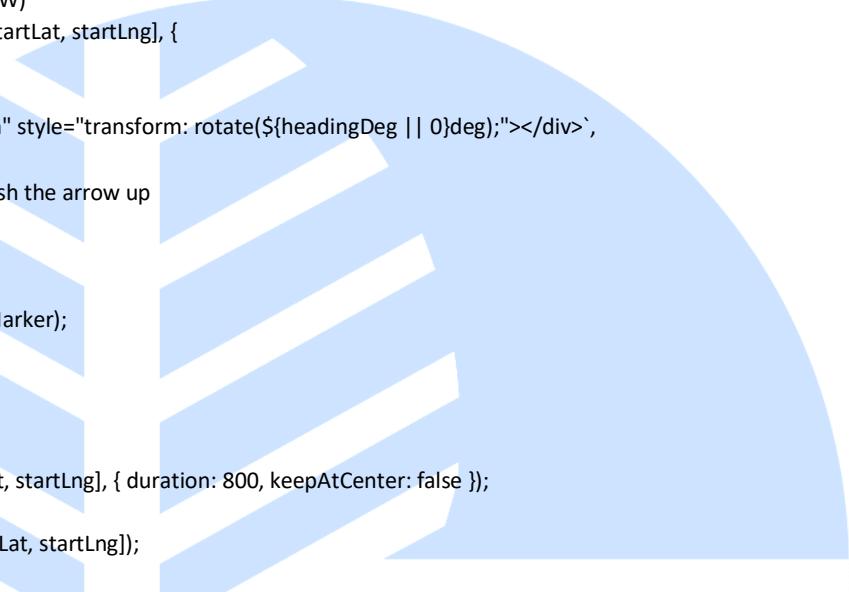
// ✅ Move pulse marker
if (livePulseMarker) {
  livePulseMarker.setLatLng([startLat, startLng]);
}

// ✅ Move heading marker & rotate it
if (liveHeadingMarker) {
  liveHeadingMarker.setLatLng([startLat, startLng]);
  if (headingDeg !== null) {
    liveHeadingMarker._icon.querySelector(".heading-icon").style.transform = `rotate(${headingDeg}deg)`;
  }
}
}

// ✅ END MARKER (only added once)
if (resetRoute) {
  let popupContent = "🏁 Destination";

  if (permitInfo) {
    popupContent = `
      <div class="permit-popup">
        <div class="permit-popup-row"><strong>Permittee:</strong> ${permitInfo.permittee}</div>
        <div class="permit-popup-row"><strong>Permit No:</strong> ${permitInfo.permitNo}</div>
        <div class="permit-popup-row"><strong>Latitude:</strong> ${permitInfo.lat}</div>
    `;
  }
}
}

```




```

<div class="permit-popup-row"><strong>Longitude:</strong> ${permitInfo.lng}</div>
${permitInfo.pdfUrl
? `<div class="permit-popup-row">
  <a href="${permitInfo.pdfUrl}" target="_blank" class="permit-pdf-link">  View Permit PDF</a>
</div>
: `<div class="permit-popup-row"><em>No PDF available</em></div>`}
</div>
`;
}

const endMarker = L.marker([endLat, endLng], { icon: violetIcon })
.addTo(map)
.bindPopup(popupContent)
.openPopup();

extraMarkers.push(endMarker);

// ✅ Fit map only once
if (!routeZoomed) {
  map.fitBounds(L.latLngBounds([[startLat, startLng], [endLat, endLng]]), { padding: [50, 50] });
  routeZoomed = true;
}
}

export function removePulseMarker() {
  if (livePulseMarker) {
    map.removeLayer(livePulseMarker);
    livePulseMarker = null;
    console.log(" ● Pulse marker removed (via cleanup function).");
  }
}

mapPrint.js

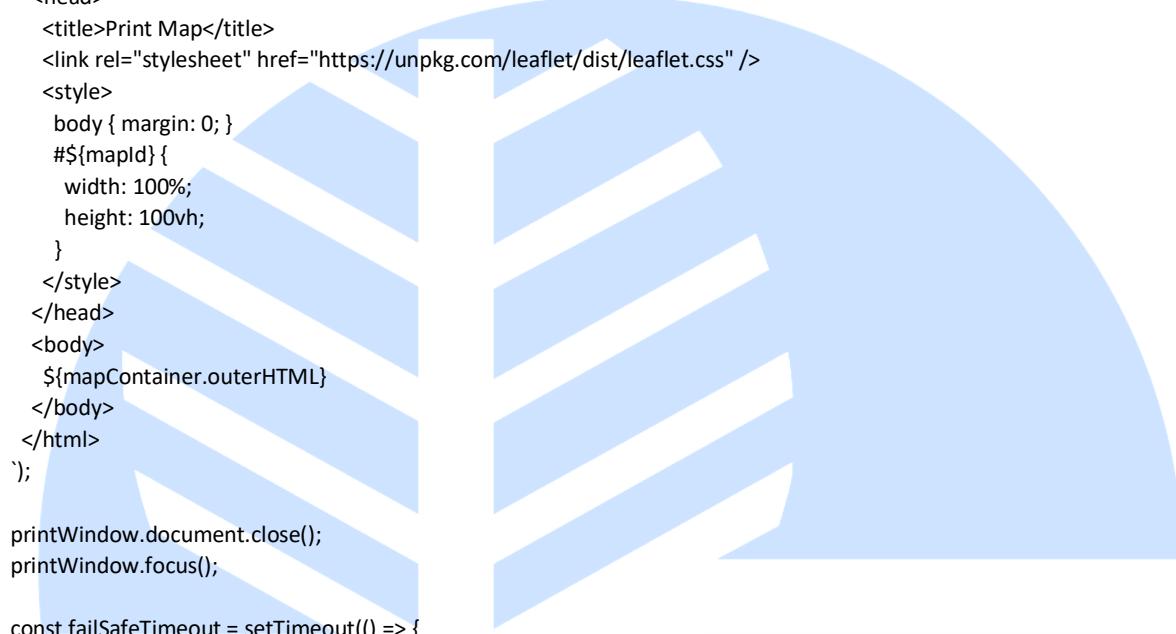
export function initMapPrint(buttonId, mapId) {
  const printBtn = document.getElementById(buttonId);

  if (!printBtn) {
    console.error(`Print button with ID '${buttonId}' not found.`);
    return;
  }

  printBtn.addEventListener("click", function () {
    const mapContainer = document.getElementById(mapId);

    if (!mapContainer) {
      console.error(`Map container with ID '${mapId}' not found.`);
      return;
    }
  })
}

```



```

if (typeof Spinner !== "undefined" && Spinner.show) {
  Spinner.show();
}

const printWindow = window.open("", "", "width=900,height=700");
printWindow.document.write(`
<html>
<head>
<title>Print Map</title>
<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
<style>
body { margin: 0; }
#${mapId} {
  width: 100%;
  height: 100vh;
}
</style>
</head>
<body>
${mapContainer.outerHTML}
</body>
</html>
`);

printWindow.document.close();
printWindow.focus();

const failSafeTimeout = setTimeout(() => {
  if (typeof Spinner !== "undefined" && Spinner.hide) {
    Spinner.hide();
  }
  console.warn("Spinner auto-hidden by failsafe timeout.");
}, 5000);

setTimeout(() => {
  printWindow.print();
  printWindow.close();
}

if (typeof Spinner !== "undefined" && Spinner.hide) {
  Spinner.hide();
}

clearTimeout(failSafeTimeout);
}, 1500);
});
}

```

routing-page-init.js

```

import { SessionGuard } from "../auth/auth.js";
import { initializePage } from "./routing-page-ui.js";

```

```
document.addEventListener('DOMContentLoaded', ()=>{
  SessionGuard.ensureLoggedIn();
  initializePage();
})
```

routing-page-ui.js

```
import { Permit } from "../data/cache/permit-data.js";
import { Sidebar } from "../components/sidebar.js";
import { NotificationBox } from "../components/notification.js";
import { Spinner } from "../components/spinner.js";
import { PortalBubble } from "../components/PortalBubble.js";
import { initMap, plotRouteOnMap, removePulseMarker } from "./map-init.js";

export function initializePage(){
  Sidebar.render();
  Spinner.render();
  toggleFormShowHide();
  setCurrentLocationInStartInput();
  enableEndLocationAutocomplete();
  initMap("routingMap", false);
  getRouteSubmitForm();
  stopTrackingHandler();
}

let selectedPermitInfo = null;
let geoWatchId = null;
let trackingLogInterval = null;
let lastLat = null;
let lastLng = null;

function toggleFormShowHide(){
  const toggleBtn = document.getElementById('toggleRoutingFormBtn');
  const formContainer = document.getElementById('routingFormContainer');

  toggleBtn.addEventListener('click', () => {
    const isVisible = formContainer.style.display === 'block';
    formContainer.style.display = isVisible ? 'none' : 'block';
    toggleBtn.innerHTML = isVisible
      ? '<i class="bi bi-caret-down-fill"></i> Show Routing Form'
      : '<i class="bi bi-caret-up-fill"></i> Hide Routing Form';
  });
}

function setCurrentLocationInStartInput() {
  const startInput = document.getElementById("startLocation");

  if (!startInput) {
    console.warn(" ! startLocation input field not found.");
    return;
  }

  if (!navigator.geolocation) {
```

```
console.warn("⚠️ Geolocation not supported by this browser.");
startInput.value = "14.609100, 121.022300";
return;
}

navigator.geolocation.getCurrentPosition(
(position) => {
  startInput.value = `${position.coords.latitude.toFixed(6)}, ${position.coords.longitude.toFixed(6)}`;
},
(error) => {
  console.warn("⚠️ Geolocation failed:", error.message);
  startInput.value = "14.609100, 121.022300";
}
);
}

function enableEndLocationAutocomplete() {
  const input = document.querySelector("#endLocation");

  input.addEventListener("keyup", async (e) => {
    const term = e.target.value.trim().toLowerCase();
    if (term.length < 2) return;

    try {
      Spinner.show();

      const permits = await Permit.getAll();

      const permitMatches = permits
        .filter(p =>
          p.permittee?.toLowerCase().includes(term) ||
          p.permitNo?.toLowerCase().includes(term)
        )
        .map(p => ({
          id: p.id,
          label: `${p.permittee} (${p.permitNo})`,
          lat: p.latitude || "",
          lng: p.longitude || "",
          permitNo: p.permitNo,
          permittee: p.permittee,
          pdfUrl: p.pdfUrl || null
        }));
    }
    catch (err) {
      console.error(err);
    }
  });
}

let dropdown = document.querySelector("#endLocationDropdown");
if (!dropdown) {
  dropdown = document.createElement("div");
  dropdown.id = "endLocationDropdown";
  dropdown.className = "list-group position-absolute w-100";
  input.parentNode.appendChild(dropdown);
}
dropdown.innerHTML = "";

permitMatches.forEach(item => {
```

```

const option = document.createElement("button");
option.type = "button";
option.className = "list-group-item list-group-item-action";
option.textContent = item.label;

option.onclick = () => {
  input.value = `${item.lat}, ${item.lng}`;
  dropdown.innerHTML = "";

  document.querySelector("#endLat").value = item.lat;
  document.querySelector("#endLng").value = item.lng;

  selectedPermitInfo = item;

  document.getElementById("routeHeading").textContent =
    `Route to: ${item.permittee} (Permit No. ${item.permitNo})`;

  PortalBubble.trigger();
};

dropdown.appendChild(option);
});

} catch (err) {
  console.error("✖ Autocomplete error:", err);
} finally {
  Spinner.hide();
}
});

}

export function getRouteSubmitForm() {
  document.getElementById("routingForm").addEventListener("submit", function (e) {
    e.preventDefault();

    Spinner.show();

    const endLat = parseFloat(document.getElementById("endLat").value);
    const endLng = parseFloat(document.getElementById("endLng").value);

    // ✅ If already tracking, stop the old watch & interval first
    if (geoWatchId !== null) {
      navigator.geolocation.clearWatch(geoWatchId);
      geoWatchId = null;
    }

    if (trackingLogInterval) {
      clearInterval(trackingLogInterval);
      trackingLogInterval = null;
    }

    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(async (position) => {
        const startLat = position.coords.latitude;

```

```

const startLng = position.coords.longitude;

// ✅ Reset last position for heading calculation
lastLat = startLat;
lastLng = startLng;

// ✅ Initial plot with current location
plotRouteOnMap(startLat, startLng, endLat, endLng, selectedPermitInfo, true, 0);
Spinner.hide();

// ✅ Notify user that tracking started
NotificationBox.show("Live tracking has started.", "success");

// ✅ Start live tracking
geoWatchId = navigator.geolocation.watchPosition(
  (pos) => {
    const newLat = pos.coords.latitude;
    const newLng = pos.coords.longitude;

    // ✅ Compute heading based on movement
    let headingDeg = 0;
    if (lastLat !== null && lastLng !== null) {
      headingDeg = calculateBearing(lastLat, lastLng, newLat, newLng);
    }

    // ✅ Update map with heading
    plotRouteOnMap(newLat, newLng, endLat, endLng, selectedPermitInfo, false, headingDeg);

    // ✅ Update last position for next calculation
    lastLat = newLat;
    lastLng = newLng;
  },
  (err) => console.warn("⚠️ Tracking error:", err),
  { enableHighAccuracy: true, maximumAge: 0, timeout: 5000 }
);

// ✅ Heartbeat interval (for UI indicators)
trackingLogInterval = setInterval(() => {
  // Could be used for UI indicators or logs
}, 1000);
};

} else {
  console.warn("❌ Geolocation not supported.");
  plotRouteOnMap(14.6091, 121.0223, endLat, endLng, selectedPermitInfo, true, 0);
  Spinner.hide();
}

function calculateBearing(lat1, lng1, lat2, lng2) {
  const φ1 = (lat1 * Math.PI) / 180;
  const φ2 = (lat2 * Math.PI) / 180;
}

```

```

const Δλ = ((lng2 - lng1) * Math.PI) / 180;

const y = Math.sin(Δλ) * Math.cos(φ2);
const x = Math.cos(φ1) * Math.sin(φ2) -
    Math.sin(φ1) * Math.cos(φ2) * Math.cos(Δλ);

let θ = Math.atan2(y, x);
θ = (θ * 180) / Math.PI;
return (θ + 360) % 360;
}

function stopTrackingHandler() {
    document.getElementById("stopTrackingBtn").addEventListener("click", () => {
        if (geoWatchId !== null) {
            navigator.geolocation.clearWatch(geoWatchId);
            geoWatchId = null;
            NotificationBox.show("🔴 Live tracking stopped.", "error");
        }

        if (trackingLogInterval) {
            clearInterval(trackingLogInterval);
            trackingLogInterval = null;
        }

        removePulseMarker();
    });
}

```

XV.C.4.f. PDF Scripts

ics-pdf.js

```

import { ICS } from "../data/cache/ics-data.js";
import { Users } from "../data/cache/user-data.js"
import { NotificationBox, Confirmation } from "../components/notification.js";

export async function generatePdfICS(userId) {
    try {
        const { jsPDF } = window.jspdf;
        const doc = new jsPDF({ orientation: "portrait" }); // Portrait mode

        if (typeof doc.autoTable !== "function") {
            throw new Error("autoTable plugin not loaded.");
        }

        const logoUrl = './images/denr logo.png';
        const loadImageAsBase64 = (url) =>
            fetch(url)
                .then(res => res.blob())
                .then(blob => new Promise((resolve) => {
                    const reader = new FileReader();

```



```

reader.onloadend = () => resolve(reader.result);
reader.readAsDataURL(blob);
});

const logoData = await loadImageAsBase64(logoUrl);

// Fetch user info
const users = await Users.fetchAllAsc();
const user = users.find(u => u.id === userId);
const fullName = user
? `${user.lastName}, ${user.firstName} ${user.middleInitial}.`
: "";
const label = "Property Accountability of ";
const fullText = `${label}${fullName}`;

// Fetch ICS data
const allItems = await ICS.getICSDataByUserId(userId);
const icsItems = allItems.filter(item => item.status !== 'RTS');

const now = new Date();
const today = now.toLocaleDateString("en-US", {
  year: "numeric",
  month: "long",
  day: "numeric"
});

// Header
doc.addImage(logoData, "PNG", 14, 8, 15, 15);
doc.setFont("helvetica", "bold");
doc.setFontSize(12);
doc.setTextColor(0, 0, 0);
doc.text("Department of Environment and Natural Resources - National Capital Region", 32, 14);
doc.text("Licenses, Patents, and Deeds Division", 32, 20);
doc.text("Water Resources Utilization Section", 32, 26);

doc.setFont("helvetica", "normal");
doc.setFontSize(10);
doc.text("Date:", 14, 36);
const dateLabelWidth = doc.getTextWidth("Date:");
doc.setFont("helvetica", "bold");
doc.text(today, 14 + dateLabelWidth + 2, 36);

// Custom Title
doc.setFont("helvetica", "bold");
doc.setFontSize(10); // Smaller font
const fullWidthWidth = doc.getTextWidth(fullText);
const centerX = doc.internal.pageSize.getWidth() / 2;
const startX = centerX - (fullWidthWidth / 2);
doc.text(fullText, startX, 45);

// Table
const headers = [
  "Qty", "Unit", "Description", "Serial No.",

```

```

    "Unit Cost", "Total Cost", "ICS No", "Date Issued", "Remarks"
  ];


  const rows = icsItems.map(item => [
    item.qty,
    item.unit,
    item.description,
    item.serialNo,
    item.unitCost.toLocaleString(undefined, { minimumFractionDigits: 2 }),
    item.totalCost.toLocaleString(undefined, { minimumFractionDigits: 2 }),
    item.ICSno,
    typeof item.dateIssued?.toDate === "function"
      ? item.dateIssued.toDate().toLocaleDateString("en-US", {
          year: "numeric", month: "short", day: "2-digit"
        })
      : item.dateIssued || "-",
    item.remarks || ""
  ]);

  doc.autoTable({
    head: [headers],
    body: rows,
    startY: 50,
    styles: {
      font: "helvetica",
      fontSize: 7, // smaller font size
      lineColor: [0, 0, 0],
      lineWidth: 0.1,
      cellPadding: { top: 1, bottom: 1, left: 2, right: 2 } // reduced padding
    },
    headStyles: {
      fillColor: [220, 220, 220],
      textColor: 0
    },
    tableLineColor: [0, 0, 0],
    tableLineWidth: 0.1
  });

  // Total Amount
  const totalAmount = icsItems.reduce((sum, item) => sum + item.totalCost, 0);
  doc.setFont("helvetica", "bold");
  doc.setFontSize(9); // Smaller font
  const totalAmountText = `Total Amount PHP: ${totalAmount.toLocaleString(undefined, { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`;
  doc.text(totalAmountText, 14, doc.lastAutoTable.finalY + 10); // Left-aligned

  // Watermark
  const pageCount = doc.getNumberOfPages();
  const pageWidth = doc.internal.pageSize.getWidth();
  const pageHeight = doc.internal.pageSize.getHeight();
  const watermarkWidth = 60;
  const watermarkHeight = 60;
  const x = (pageWidth - watermarkWidth) / 2;

```

```

const y = (pageHeight - watermarkHeight) / 2;

for (let i = 1; i <= pageCount; i++) {
  doc.setPage(i);
  doc.saveGraphicsState();
  doc.setGState(new doc.GState({ opacity: 0.07 }));
  doc.addImage(logoData, 'PNG', x, y, watermarkWidth, watermarkHeight);
  doc.restoreGraphicsState();
}

const filename = `ics-report-${now.toISOString().slice(0, 19).replace(/[:T]/g, "-")}.pdf`;

const isMobile = /Mobi|Android/i.test(navigator.userAgent);
if (isMobile) {
  await new Promise((resolve) => {
    Confirmation.show(
      "Do you want to save this ICS report as a PDF?",
      (confirmed) => {
        if (confirmed) {
          doc.save(filename);
        }
        resolve();
      }
    );
  });
} else {
  const blob = doc.output("blob");
  const blobUrl = URL.createObjectURL(blob);
  const modalBody = document.querySelector("#ICS-pdfModal .modal-body");
  modalBody.innerHTML = `
    <div class="w-100" style="height:100vh;">
      <embed src="${blobUrl}" type="application/pdf" class="w-100 h-100" />
    </div>
  `;
}

const modal = new bootstrap.Modal(document.getElementById("ICS-pdfModal"));
modal.show();
}

} catch (err) {
  console.error("Failed to generate ICS PDF:", err);
  NotificationBox.show("Failed to generate PDF. Please try again.", "error");
}
}

```

item-consumable-pdf.js

```

import { db } from "../firebaseConfig";
import { doc, getDoc } from "https://www.gstatic.com/firebasejs/10.11.1.firebaseio-firestore.js";

export async function generateLedgerPDFBlob(selectedCID, ledgerEntries, totalQty) {
  const { jsPDF } = window.jspdf;
  const pdfDoc = new jsPDF();

```

```
const logoUrl = './images/denr logo.png';

const loadImageAsBase64 = (url) =>
  fetch(url)
    .then(res => res.blob())
    .then(blob =>
      new Promise(resolve => {
        const reader = new FileReader();
        reader.onloadend = () => resolve(reader.result);
        reader.readAsDataURL(blob);
      })
    );
  );

const logoData = await loadImageAsBase64(logoUrl);

// Header logo
pdfDoc.addImage(logoData, 'PNG', 14, 10, 15, 15);

let specification = "Not Found";
let unit = "—";

try {
  const docRef = doc(db, "consumable", selectedCID);
  const docSnap = await getDoc(docRef);
  if (docSnap.exists()) {
    const data = docSnap.data();
    specification = data.specification || "N/A";
    unit = data.unit || "—";
  }
} catch {
  specification = "Error fetching specification";
}

const formatDate = (dateObj) =>
  dateObj.toLocaleDateString("en-US", {
    year: "numeric",
    month: "long",
    day: "numeric"
  });

const today = formatDate(new Date());

// Header text
pdfDoc.setFont("helvetica", "bold");
pdfDoc.setFontSize(12);
pdfDoc.text("Department of Environment and Natural Resources - NCR", 32, 16);
pdfDoc.text("Licensing, Patents, and Deeds Division", 32, 22);
pdfDoc.text("Water Resources Utilization Section", 32, 28);

// Details
pdfDoc.setFont("helvetica", "normal");
pdfDoc.setFontSize(10);
```

```

const addLabel = (label, value, y) => {
  pdfDoc.setFont("helvetica", "normal");
  pdfDoc.text(label, 14, y);
  const labelWidth = pdfDoc.getTextWidth(label);
  pdfDoc.setFont("helvetica", "bold");
  pdfDoc.text(value, 14 + labelWidth + 2, y);
};

addLabel("Date:", today, 36);
addLabel("Ledger Report for:", specification, 40);
addLabel("Stocks Left:", String(totalQty), 46);
addLabel("Unit of Measurement:", unit, 50);

let currentY = 56;

// Filter for Add Stock entries
const addStockEntries = ledgerEntries.filter(e => e.action === "Add Stock");
if (addStockEntries.length > 0) {
  pdfDoc.setFont("helvetica", "bold");
  pdfDoc.text("Add to Stock", 14, currentY);
  currentY += 6;

  const inventoryTable = addStockEntries.map(entry => [
    entry.dateModified ? formatDate(entry.dateModified.toDate()) : "--",
    entry.amount,
    entry.remarks || "--"
  ]);

  pdfDoc.autoTable({
    head: [["Date", "Qty", "Remarks"]],
    body: inventoryTable,
    startY: currentY,
    theme: 'grid',
    didParseCell: function (data) {
      if (data.section === 'body' && data.column.index === 1) {
        data.cell.styles.textColor = [0, 128, 0]; // green for qty
      }
    }
  });
}

currentY = pdfDoc.lastAutoTable.finalY + 10;
}

// Filter for Assign Item entries
const assignEntries = ledgerEntries.filter(e => e.action === "Assign Item");
if (assignEntries.length > 0) {
  pdfDoc.setFont("helvetica", "bold");
  pdfDoc.text("Assigned Items", 14, currentY);
  currentY += 6;

  const assignTable = assignEntries.map(entry => [
    entry.dateModified ? formatDate(entry.dateModified.toDate()) : "--",
    entry.amount,
  ]);
}

```

```
entry.assignedTo || "-",
entry.remarks || "-"
]));

pdfDoc.autoTable({
  head: [["Date", "Qty", "Assigned To", "Remarks"]],
  body: assignTable,
  startY: currentY,
  theme: 'grid',
  didParseCell: function (data) {
    if (data.section === 'body' && data.column.index === 1) {
      data.cell.styles.textColor = [220, 20, 60]; // crimson for qty
    }
  }
});
}

return pdfDoc.output("blob");
}
```

user-consumable-pdf.js

```
export async function generateConsumablePDF(user, entriesArray, consumablesMap) {
  const { jsPDF } = window.jspdf;

  const doc = new jsPDF();

  // Load logo
  const logoUrl = './images/denr logo.png';
  const loadImageAsBase64 = (url) =>
    fetch(url)
      .then((res) => res.blob())
      .then(
        (blob) =>
          new Promise((resolve) => {
            const reader = new FileReader();
            reader.onloadend = () => resolve(reader.result);
            reader.readAsDataURL(blob);
          })
      );
}

const logoData = await loadImageAsBase64(logoUrl);

const fullName = `${user.lastName}, ${user.firstName} ${user.middleInitial}.`;

// Date formatter
const formatDateForTable = (dateObj) =>
  dateObj.toLocaleDateString("en-US", {
    year: "numeric",
    month: "long",
    day: "2-digit"
  });
}
```

```
const now = new Date();
const today = now.toLocaleDateString("en-US", {
  year: "numeric",
  month: "long",
  day: "numeric"
});

// Header and logo
doc.addImage(logoData, "PNG", 14, 10, 15, 15);
doc.setFont("helvetica", "bold");
doc.setFontSize(12);
doc.text("Department of Environment and Natural Resources - National Capital Region", 32, 16);
doc.text("Licenses, Patents, and Deeds Division", 32, 22);
doc.text("Water Resources Utilization Section", 32, 28);

doc.setFont("helvetica", "normal");
doc.setFontSize(10);
doc.text("Date:", 14, 36);
let dateLabelWidth = doc.getTextWidth("Date:");
doc.setFont("helvetica", "bold");
doc.text(today, 14 + dateLabelWidth + 2, 36);

doc.setFont("helvetica", "normal");
const labelText = "Consumable Item Ledger Report for:";
doc.text(labelText, 14, 42);

const titleLabelWidth = doc.getTextWidth(labelText);
doc.setFont("helvetica", "bold");
doc.text(fullName, 14 + titleLabelWidth + 2, 42);

const tableBody = entriesArray.map(({ cid, amount, dateModified, remarks }) => {
  const spec = consumablesMap[cid]?.specification || "-";
  const unit = consumablesMap[cid]?.unit || "-";

  let formattedDate = "-";
  if (dateModified && typeof dateModified.toDate === "function") {
    formattedDate = formatDateForTable(dateModified.toDate());
  }

  return [
    formattedDate,
    spec,
    unit,
    amount,
    remarks
  ];
});

// Table with Remarks
doc.autoTable({
  startY: 50,
  head: [["Date", "Specification", "UoM", "Qty", "Remarks"]],
```

```

body: tableBody,
styles: { font: "helvetica", fontSize: 10 },
});

// Watermark
const pageCount = doc.getNumberOfPages();
const pageWidth = doc.internal.pageSize.getWidth();
const pageHeight = doc.internal.pageSize.getHeight();
const watermarkWidth = 100;
const watermarkHeight = 100;
const x = (pageWidth - watermarkWidth) / 2;
const y = (pageHeight - watermarkHeight) / 2;

for (let i = 1; i <= pageCount; i++) {
  doc.setPage(i);
  doc.saveGraphicsState();
  doc.setGState(new doc.GState({ opacity: 0.07 }));
  doc.addImage(logoData, 'PNG', x, y, watermarkWidth, watermarkHeight);
  doc.restoreGraphicsState();
}

// Ask to save on mobile, or show modal on desktop
const isMobile = /Mobi|Android/i.test(navigator.userAgent);
const filename = `consumable-report-${now.toISOString().slice(0, 19).replace(/[:T]/g, "-")}.pdf`;

if (isMobile) {
  const confirmSave = window.confirm("Do you want to save this report as a PDF?");
  if (confirmSave) {
    doc.save(filename);
  }
} else {
  // Show PDF in modal on desktop
  const blob = doc.output("blob");
  const blobUrl = URL.createObjectURL(blob);
  const modalBody = document.querySelector("#consumableModal .modal-body");
  modalBody.innerHTML = ``;
}
}

```

water-user-pdf.js

```

export async function generateWaterUserPDF(user, options = {}) {
  const { jsPDF } = window.jspdf;
  const doc = new jsPDF({ unit: "mm", format: "A4" });

  doc.setFont("helvetica", "normal");
  doc.setFontSize(8);

  // Top Right: NWRB-MED-16-r0, Control No., Date
  const pageWidth = doc.internal.pageSize.getWidth();
  const margin = 20;
  const rightX = pageWidth - margin;

```

```

doc.setFont(undefined, "bold");
doc.text("NWRB-MED-16-r0", rightX, 15, { align: "right" });

doc.setFont(undefined, "normal");
doc.text("Control No.:", rightX - 40, 20);
doc.text("_____ ", rightX - 5, 20, { align: "right" });

doc.text("Date:", rightX - 40, 25);
doc.text("_____ ", rightX - 5, 25, { align: "right" });

// Center-aligned header
const center = pageWidth / 2;
let y = 35;

doc.text("Republic of the Philippines", center, y, { align: "center" });
y += 4;
doc.text("NATIONAL WATER RESOURCES BOARD", center, y, { align: "center" });
y += 4;
doc.text("8th Floor N.I.A. Building, E.D.S.A., Quezon City", center, y, { align: "center" });
y += 4;
doc.text("Tel. No. 920-26-54", center, y, { align: "center" });

y += 6;
doc.text("MONITORING AND ENFORCEMENT DIVISION", center, y, { align: "center" });

y += 6;
doc.setFont(undefined, "bold");
doc.text("SITE VERIFICATION REPORT", center, y, { align: "center" });

doc.setFont(undefined, "normal");

const lineSpacing = 5;
const pad = 20;
y += 8;

const lines = [
  ['OWNER:', user.owner],
  ['MAILING ADDRESS:', `${user.street} ${user.barangay}, ${user.city}`],
  ['LOCATION OF WATER SOURCE:', `${user.street} ${user.barangay}, ${user.city}`],
  [
    'GPS COORDINATES:',
    `LATITUDE: ${Number(user.latitude).toFixed(5)} LONGITUDE: ${Number(user.longitude).toFixed(5)}`,
  ],
  ['SOURCE OF WATER:', user.type || "N/A"],
  ['REMARKS:', user.remarks || "N/A"],
];

lines.forEach(([label, value], index) => {
  // Special handling for GPS COORDINATES line
  if (label === "GPS COORDINATES:") {
    doc.text(label, pad, y);
  }
  // LATITUDE centered

```

```

const latLabel = "LATITUDE: ";
const latValue = user.latitude ? Number(user.latitude).toFixed(5) : "N/A";
const latLabelWidth = doc.getTextWidth(latLabel);
const latValueWidth = doc.getTextWidth(latValue);
const latX = (pageWidth - (latLabelWidth + latValueWidth)) / 2;

doc.text(latLabel, latX, y);
doc.text(latValue, latX + latLabelWidth, y);
doc.line(latX + latLabelWidth, y + 1, latX + latLabelWidth + latValueWidth, y + 1);

// LONGITUDE right-aligned
const longLabel = "LONGITUDE: ";
const longValue = user.longitude ? Number(user.longitude).toFixed(5) : "N/A";
const longLabelWidth = doc.getTextWidth(longLabel);
const longValueWidth = doc.getTextWidth(longValue);
const longX = pageWidth - pad - (longLabelWidth + longValueWidth);

doc.text(longLabel, longX, y);
doc.text(longValue, longX + longLabelWidth, y);
doc.line(longX + longLabelWidth, y + 1, longX + longLabelWidth + longValueWidth, y + 1);

y += lineSpacing;
return;
}

// Default rendering
doc.text(label, pad, y);

const labelWidth = doc.getTextWidth(label) + 2;
const valueX = pad + labelWidth;
const valueWidth = doc.getTextWidth(value);
const underlineY = y + 1;

// Dynamic full-width underline logic
let lineEndX;
if (
  label === "OWNER:" || 
  label === "LOCATION OF WATER SOURCE:" || 
  label === "REMARKS:"
) {
  lineEndX = pageWidth - pad;
} else {
  lineEndX = valueX + valueWidth;
}

doc.text(value, valueX, y);
doc.line(valueX, underlineY, lineEndX, underlineY);

// MAILING ADDRESS: => TEL. NO./CELL. NO.
if (label === "MAILING ADDRESS:") {
  const telLabel = "TEL. NO./CELL. NO.: ";
  const telUnderlineWidth = 40;
  const telLabelWidth = doc.getTextWidth(telLabel);
}

```

```

const telTotalWidth = telLabelWidth + telUnderlineWidth;
const telX = pageWidth - pad - telTotalWidth;

doc.text(telLabel, telX, y);
doc.line(telX + telLabelWidth, underlineY, telX + telLabelWidth + telUnderlineWidth, underlineY);
}

// SOURCE OF WATER: => NO. OF SOURCES and PURPOSE
if (label === "SOURCE OF WATER:") {
  const noOfSourcesLabel = "NO. OF SOURCES:";
  const noOfSourcesValue = "_____"; // fixed-width underline
  const noOfSourcesText = `${noOfSourcesLabel} ${noOfSourcesValue}`;
  const noOfSourcesWidth = doc.getTextWidth(noOfSourcesText);

  const purposeLabel = "PURPOSE:";
  const purposeValue = user.purpose || "N/A";
  const purposeLabelWidth = doc.getTextWidth(purposeLabel);
  const purposeValueWidth = doc.getTextWidth(purposeValue);
  const spacingBetween = 10;
  const spacingAfterLabel = 2;

  // Total width of both parts
  const totalWidth =
    noOfSourcesWidth + spacingBetween + purposeLabelWidth + spacingAfterLabel + purposeValueWidth;

  const startX = pageWidth - pad - totalWidth;

  // Draw NO. OF SOURCES
  doc.text(noOfSourcesLabel, startX, y);
  const underlineStartX = startX + doc.getTextWidth(noOfSourcesLabel + " ");
  doc.text(noOfSourcesValue, underlineStartX, y);

  // Draw PURPOSE
  const purposeX = startX + noOfSourcesWidth + spacingBetween;
  doc.text(purposeLabel, purposeX, y);
  const purposeValueX = purposeX + purposeLabelWidth + spacingAfterLabel;
  doc.text(purposeValue, purposeValueX, y);

  // Underline the value of PURPOSE
  doc.line(
    purposeValueX,
    underlineY,
    purposeValueX + purposeValueWidth,
    underlineY
  );

  y += lineSpacing;

  // Determine checkbox states
  const status = (user.status || "").toUpperCase();
  const checked = "X";
  const empty = " ";
}

```

```

// Default: Non-Operational is checked
let proposed = '[ ${empty} ] Proposed';
let operational = '[ ${empty} ] Operational';
let ongoing = '[ ${empty} ] On-going Drilling/Construction';
let nonOperational = '[ ${checked} ] Non Operational';

// Override defaults if status matches
if (status === "PROPOSED") {
  proposed = '[ ${checked} ] Proposed';
  nonOperational = '[ ${empty} ] Non Operational';
} else if (status === "OPERATIONAL") {
  operational = '[ ${checked} ] Operational';
  nonOperational = '[ ${empty} ] Non Operational';
} else if (status === "ON-GOING CONSTRUCTION") {
  ongoing = '[ ${checked} ] On-going Drilling/Construction';
  nonOperational = '[ ${empty} ] Non Operational';
}

// Combine the line
const sourceStatusLine = `SOURCE STATUS:  ${proposed} ${operational} ${ongoing} ${nonOperational}`;

// Render on PDF
doc.text(sourceStatusLine, pad, y);
y += lineSpacing;

// NAME OF WELL DRILLER line (indented)
const drillerLabel = "  NAME OF WELL DRILLER (if applicable):";
const drillerLabelWidth = doc.getTextWidth(drillerLabel);
doc.text(drillerLabel, pad, y);
doc.line(pad + drillerLabelWidth + 2, y + 1, pageWidth - pad, y + 1);
y += lineSpacing;

// MAILING ADDRESS line (indented)
const addrLabel = "  MAILING ADDRESS:";
const addrLabelWidth = doc.getTextWidth(addrLabel);
doc.text(addrLabel, pad, y);
doc.line(pad + addrLabelWidth + 2, y + 1, pageWidth - pad, y + 1);
}

// Extra lines after REMARKS
if (label === "REMARKS:") {
  y += lineSpacing;
  doc.line(pad, y + 1, pageWidth - pad, y + 1);
  y += lineSpacing;
  doc.line(pad, y + 1, pageWidth - pad, y + 1);
}

y += lineSpacing;
});

// Insert signature image if available
if (user.signUrl) {
  const img = await loadImageAsDataURL(user.signUrl);
}

```

```

if (img) {
  const imgWidth = 40;
  const imgHeight = 15;
  const imgX = pageWidth - pad - imgWidth;
  doc.addImage(img, "PNG", imgX, y, imgWidth, imgHeight);
  y += 18;
}
}

// Line 1: VERIFIED BY (left) and REPRESENTATIVE (right)
const verifiedByLabel = "VERIFIED BY: ";
const verifiedLine = "_".repeat(40); // Adjust as needed
const verifiedByText = verifiedByLabel + verifiedLine;
doc.text(verifiedByText, pad, y);

const repLabel = "Representative: ";
const repValue = user.representative || "N/A";
const repLabelWidth = doc.getTextWidth(repLabel);
const repValueWidth = doc.getTextWidth(repValue);
const repTotalWidth = repLabelWidth + repValueWidth;
const repX = pageWidth - pad - repTotalWidth;

doc.text(repLabel, repX, y);
doc.text(repValue, repX + repLabelWidth, y);

// Underline only the representative value
const underlineY = y + 1;
doc.line(
  repX + repLabelWidth,
  underlineY,
  repX + repLabelWidth + repValueWidth,
  underlineY
);

y += lineSpacing;

// Line 2: Signature labels (aligned under their respective fields)
const verifiedSignatureText = "Signature Over Printed Name";
doc.text(verifiedSignatureText, pad + 25, y); // indent slightly under left side

const repSignatureText = "Signature Over Printed Name";
const repSignatureWidth = doc.getTextWidth(repSignatureText);
const repSignatureX = pageWidth - pad - repSignatureWidth;
doc.text(repSignatureText, repSignatureX, y);

y += lineSpacing;

// Line 3: Position (right-aligned with underline)
const positionLabel = "Position: ";
const positionValue = user.designation || "";
const positionText = positionLabel + positionValue;
const positionTextWidth = doc.getTextWidth(positionText);
const positionX = pageWidth - pad - positionTextWidth;

```

```
doc.text(positionText, positionX, y);

// Draw underline for value only
const labelWidth = doc.getTextWidth(positionLabel);
const valueX = positionX + labelWidth;
const valueWidth = doc.getTextWidth(positionValue);
doc.line(valueX, y + 1, valueX + valueWidth, y + 1); // y+1 to place the underline below text

y += lineSpacing;

// Line 4: Mobile No. (right-aligned with underline)
const mobileLabel = "Mobile No: ";
const mobileValue = user.phone || "";
const mobileText = mobileLabel + mobileValue;
const mobileTextWidth = doc.getTextWidth(mobileText);
const mobileX = pageWidth - pad - mobileTextWidth;
doc.text(mobileText, mobileX, y);

// Draw underline for value only
const mobileLabelWidth = doc.getTextWidth(mobileLabel);
const mobileValueX = mobileX + mobileLabelWidth;
const mobileValueWidth = doc.getTextWidth(mobileValue);
doc.line(mobileValueX, y + 1, mobileValueX + mobileValueWidth, y + 1);

y += lineSpacing;

// Final output
if (options.returnBlob) {
  return doc.output("blob");
} else {
  doc.save(`SiteVerification-${user.id || "unknown"}.pdf`);
}

// Helper to load image from URL
async function loadImageAsDataURL(url) {
  try {
    const response = await fetch(url);
    const blob = await response.blob();
    return await new Promise((resolve) => {
      const reader = new FileReader();
      reader.onloadend = () => resolve(reader.result);
      reader.readAsDataURL(blob);
    });
  } catch (e) {
    console.warn("Failed to load signature image:", e);
    return null;
  }
}
```

XV.C.3.g. Permit Module

permit-init.js

```
import { SessionGuard } from "../auth/auth.js";
import { initializePage } from "./permit-ui.js";

document.addEventListener('DOMContentLoaded', ()=>{
  SessionGuard.ensureLoggedIn();
  initializePage();
})
```

permit-ui.js

```
import { Sidebar } from "../components/sidebar.js";
import { Spinner } from "../components/spinner.js";
import { Permit } from "../data/cache/permit-data.js"
import { WUSData } from "../data/cache/wrus-data.js";
import { METRO_MANILA_CITIES } from '../data/constants/metroManilaCities.js';
import { FileService } from "../upload/upload.js";
import { GeotaggedFileService } from "../upload/uploadImg.js";
import { CoordinateUtils } from "../utils/coordinates.js";
import { NotificationBox } from "../components/notification.js";

export function initializePage(){
  Sidebar.render();
  Spinner.render();
  populateSelect("#editDiversionPoint", METRO_MANILA_CITIES, "-- Select City --");
  populateSelect("#diversionPoint", METRO_MANILA_CITIES, "-- Select City --");

  handleAddButton();
  handleAddFormSubmit();
  loadPermit();
  setupToggle('toggleFilter', 'filterSection', 'Show Filters', 'Hide Filters');
  initializePermitUpdate();
  handleRefreshButton();
  setupExportButtonListener();
  setupImageUploadModal();

}

//Global Variable
let currentPage = 1;

function populateSelect(selectId, itemsArray, placeholder = "-- Select an option --") {
  const selectElement = document.querySelector(selectId);
  if (!selectElement) return;

  selectElement.innerHTML = "";

  if (placeholder) {
    const placeholderOption = document.createElement("option");
    placeholderOption.value = "";
    selectElement.appendChild(placeholderOption);
  }

  itemsArray.forEach(item => {
    const option = document.createElement("option");
    option.value = item;
    selectElement.appendChild(option);
  })
}
```



```
placeholderOption.textContent = placeholder;
placeholderOption.disabled = true;
placeholderOption.selected = true;
selectElement.appendChild(placeholderOption);
}

itemsArray.forEach(item => {
  const option = document.createElement("option");
  option.value = item;
  option.textContent = item;
  selectElement.appendChild(option);
});

function handleAddButton(){
  document.getElementById('addPermitBtn').addEventListener('click', () => {
    const permitModal = new bootstrap.Modal(document.getElementById('addPermitModal'));
    permitModal.show();
  });
}

function handleAddFormSubmit() {
  document.getElementById('savePermitBtn').addEventListener('click', async () => {
    Spinner.show();

    const form = document.getElementById('permitForm');

    // Form validation
    if (!form.checkValidity()) {
      form.classList.add('was-invalid');
      Spinner.hide();
      return;
    }

    const visited = document.getElementById('visited').checked;
    const cancelled = document.getElementById('isCancelled').checked;
    const pdfFile = document.getElementById('pdfAttachment').files[0];

    let pdfUrl = null;

    // Upload PDF if present
    if (pdfFile) {
      pdfUrl = await FileService.uploadPermitFile(pdfFile);
      if (!pdfUrl) {
        Spinner.hide();
        return;
      }
    }

    // Convert Latitude DMS to Decimal
    const latDegrees = parseFloat(document.getElementById('latDegrees').value) || 0;
    const latMinutes = parseFloat(document.getElementById('latMinutes').value) || 0;
    const latSeconds = parseFloat(document.getElementById('latSeconds').value) || 0;
```

```

const latDirection = document.getElementById('latDirection').value;

const latitude = CoordinateUtils.dmsToDecimal(latDegrees, latMinutes, latSeconds, latDirection);

// 🌏 Convert Longitude DMS to Decimal
const lonDegrees = parseFloat(document.getElementById('lonDegrees').value) || 0;
const lonMinutes = parseFloat(document.getElementById('lonMinutes').value) || 0;
const lonSeconds = parseFloat(document.getElementById('lonSeconds').value) || 0;
const lonDirection = document.getElementById('lonDirection').value;

const longitude = CoordinateUtils.dmsToDecimal(lonDegrees, lonMinutes, lonSeconds, lonDirection);

const data = {
  permitNo: document.getElementById('permitNo').value.trim(),
  permittee: document.getElementById('permittee').value.trim(),
  mailingAddress: document.getElementById('mailingAddress').value.trim(),
  diversionPoint: document.getElementById('diversionPoint').value.trim(),
  latitude: latitude, // Decimal latitude
  longitude: longitude, // Decimal longitude
  waterSource: document.getElementById('waterSource').value.trim(),
  waterDiversion: document.getElementById('waterDiversion').value.trim(),
  flowRate: parseFloat(document.getElementById('flowRate').value) || 0,
  purpose: document.getElementById('purpose').value.trim(),
  periodOfUse: document.getElementById('periodOfUse').value.trim(),
  visited: visited,
  cancelled: cancelled,
  pdfUrl: pdfUrl || '',
  encodedBy: localStorage.getItem("wrusUserId") || null
};

try {
  await Permit.add(data);
  renderPermitTable(currentPage);
  NotificationBox.show('Permit successfully added');

  // Reset form
  form.reset();
  form.classList.remove('was-validated');
  const modal = bootstrap.Modal.getInstance(document.getElementById('addPermitModal'));
  modal.hide();
} catch (error) {
  NotificationBox.show(`Error: ${error.message}`, 'error');
  console.error('Error:', error);
} finally {
  Spinner.hide();
}
};

function setupToggle(buttonId, sectionId, labelShow, labelHide) {
  const button = document.getElementById(buttonId);
  const section = document.getElementById(sectionId);

```

```
if (!button || !section) {
  console.error(`setupToggle error: Check IDs "${buttonId}" or "${sectionId}" — one or both not found.`);
  return;
}

// Initialize button label based on current visibility
const isInitiallyHidden = section.style.display === 'none' || getComputedStyle(section).display === 'none';

button.innerHTML = isInitiallyHidden
  ? `<i class="bi bi-eye"></i> ${labelShow}`
  : `<i class="bi bi-eye-slash"></i> ${labelHide}`;

// Add toggle event
button.addEventListener('click', () => {
  const isHidden = section.style.display === 'none' || getComputedStyle(section).display === 'none';

  section.style.display = isHidden ? 'block' : 'none';
  button.innerHTML = isHidden
    ? `<i class="bi bi-eye-slash"></i> ${labelHide}`
    : `<i class="bi bi-eye"></i> ${labelShow}`;
});

}

async function renderPermitTable() {
  const tableBody = document.getElementById('permitTableBody');
  const searchBar = document.getElementById('searchBar');
  const visitedFilter = document.getElementById('visitedFilter');
  const showVisitedFilter = document.getElementById('showVisitedFilter');
  const pagination = document.getElementById('pagination');
  const rowsPerPageSelect = document.getElementById('rowsPerPage');

  function normalizeText(str) {
    return str
      .normalize('NFD')
      .replace(/([\u0300-\u036f])/g, "")
      .toLowerCase();
  }

  function highlightMatch(text, searchTerm) {
    if (!searchTerm) return text;

    const normalizedText = normalizeText(text);
    const normalizedSearch = normalizeText(searchTerm);

    const startIndex = normalizedText.indexOf(normalizedSearch);
    if (startIndex === -1) return text;

    const endIndex = startIndex + searchTerm.length;

    return (
      text.slice(0, startIndex) +
      '<mark>' + text.slice(startIndex, endIndex) + '</mark>' +
      text.slice(endIndex)
    );
  }
}
```

```
};

}

let rowsPerPage = parseInt(rowsPerPageSelect.value);
let currentPage = 1;
let filteredPermits = [];
let searchTerm = "";

tableBody.innerHTML = "";
pagination.innerHTML = "";

try {
  const permits = await Permit.getAll(); // <-- Your Firestore data fetching here

  function renderRows(data) {
    tableBody.innerHTML = "";

    if (data.length === 0) {
      tableBody.innerHTML = `|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No data found | | | | | | | | |
`;
      return;
    }

    const start = (currentPage - 1) * rowsPerPage;
    const end = start + rowsPerPage;
    const paginatedItems = data.slice(start, end);

    paginatedItems.forEach((permit) => {
      const latitude = permit.latitude ? Number(permit.latitude).toFixed(5) : "";
      const longitude = permit.longitude ? Number(permit.longitude).toFixed(5) : "";
      const isVisited = permit.visited === true;
      const isCancelled = permit.cancelled === true;

      let badges = "";
      if (isCancelled) {
        badges += `<span class="badge-3d badge-danger me-1">Cancelled</span>`;
      }
      if (isVisited) {
        badges += `<span class="badge-3d badge-success">Visited</span>`;
      }

      const row = document.createElement('tr');

      row.innerHTML = `
        <td>
          ${highlightMatch(permit.permitNo || "", searchTerm)}
          ${badges ? `<div class="mt-1">${badges}</div>` : ""}
        </td>
        <td>${highlightMatch(permit.permittee || "", searchTerm)}</td>
        <td>${highlightMatch(permit.diversionPoint || "", searchTerm)}</td>
        <td>${highlightMatch(latitude, searchTerm)}</td>
        <td>${highlightMatch(longitude, searchTerm)}</td>
        <td>
      
```

```

<div class="action-buttons">
  <button class="btn btn-3d btn-sm btn-warning edit-btn" data-id="${permit.id}">
    <i class="bi bi-pencil-square"></i>
  </button>
  <a href="${permit.pdfUrl}" target="_blank" class="btn btn-3d btn-sm btn-info">
    <i class="bi bi-eye-fill"></i>
  </a>
  <button class="btn btn-3d btn-sm ${isVisited ? 'btn-secondary' : 'btn-success'} visit-btn" data-id="${permit.id}">
    <i class="bi ${isVisited ? 'bi-check-circle-fill' : 'bi-geo-alt-fill'}"></i>
  </button>
</div>
</td>
<td>
</td>
</tr>

tbody.appendChild(row);

const editButton = row.querySelector('.edit-btn');
const visitButton = row.querySelector('.visit-btn');

editButton.addEventListener('click', () => {
  const permitId = editButton.getAttribute('data-id');
  const permitData = permits.find((p) => p.id === permitId);
  if (permitData) {
    populateEditModal(permitData);
    const editModal = new bootstrap.Modal(document.getElementById('editPermitModal'));
    editModal.show();
  }
});

visitButton.addEventListener('click', async () => {
  const imageModalEl = document.getElementById('imageUploadModal');
  const imageModal = new bootstrap.Modal(imageModalEl);
  const imagePreview = document.getElementById('imagePreview');
  const permitId = visitButton.getAttribute('data-id');
  const placeholderURL = './images/placeholder.png';

  let imageUrl = placeholderURL;

  try {
    const doc = await Permit.getByID(permitId);
    if (doc?.geotaggedUrl) {
      imageUrl = doc.geotaggedUrl;
    }
  } catch (error) {
    console.error('Error loading permit or image:', error);
  }

  setupImageUploadModal(permitId, imageUrl, permit.permitNo);
  imageModal.show();
});
}

```

```
function renderPagination(totalItems) {
  pagination.innerHTML = '';
  const totalPages = Math.ceil(totalItems / rowsPerPage);
  if (totalPages <= 1) return;

  const createButton = (label, page, disabled = false, active = false) => {
    const li = document.createElement('li');
    li.className = `page-item ${disabled ? 'disabled' : ''} ${active ? 'active' : ''}`;
    const btn = document.createElement('button');
    btn.className = 'page-link';
    btn.innerText = label;
    btn.addEventListener('click', () => {
      if (!disabled) {
        currentPage = page;
        renderRows(filteredPermits);
        renderPagination(filteredPermits.length);
      }
    });
    li.appendChild(btn);
    return li;
  };

  const maxPageButtons = 5;
  let startPage = Math.max(1, currentPage - Math.floor(maxPageButtons / 2));
  let endPage = startPage + maxPageButtons - 1;

  if (endPage > totalPages) {
    endPage = totalPages;
    startPage = Math.max(1, endPage - maxPageButtons + 1);
  }

  pagination.appendChild(createButton('«', currentPage - 1, currentPage === 1));
  for (let i = startPage; i <= endPage; i++) {
    pagination.appendChild(createButton(i, i, false, currentPage === i));
  }
  pagination.appendChild(createButton('»', currentPage + 1, currentPage === totalPages));
}

function applyFilters() {
  searchTerm = normalizeText(searchBar.value.trim());

  const isVisitedChecked = visitedFilter.checked;
  const isShowVisitedChecked = showVisitedFilter.checked;

  filteredPermits = permits.filter((permit) => {
    const matchesSearch =
      (permit.permitNo && normalizeText(permit.permitNo).includes(searchTerm)) ||
      (permit.permittee && normalizeText(permit.permittee).includes(searchTerm)) ||
      (permit.diversionPoint && normalizeText(permit.diversionPoint).includes(searchTerm)) ||
      (permit.latitude && normalizeText(String(permit.latitude)).includes(searchTerm)) ||
      (permit.longitude && normalizeText(String(permit.longitude)).includes(searchTerm));

    let matchesVisited = true;
```

```

if (isVisitedChecked && !isShowVisitedChecked) {
  matchesVisited = permit.visited === false;
} else if (!isVisitedChecked && isShowVisitedChecked) {
  matchesVisited = permit.visited === true;
} else if (isVisitedChecked && isShowVisitedChecked) {
  matchesVisited = false;
}

return matchesSearch && matchesVisited;
});

renderRows(filteredPermits);
renderPagination(filteredPermits.length);

// Store filtered results for export
localStorage.setItem('cachedPermitsExcel', JSON.stringify(filteredPermits));
}

rowsPerPageSelect.addEventListener('change', () => {
rowsPerPage = parseInt(rowsPerPageSelect.value);
currentPage = 1;
renderRows(filteredPermits);
renderPagination(filteredPermits.length);
});

searchBar.addEventListener('input', applyFilters);
visitedFilter.addEventListener('change', () => {
  showVisitedFilter.disabled = visitedFilter.checked;
  applyFilters();
});
showVisitedFilter.addEventListener('change', () => {
  visitedFilter.disabled = showVisitedFilter.checked;
  applyFilters();
});

applyFilters(); // Initial filter and render

} catch (error) {
  console.error('Error rendering permit table:', error.message);
  tableBody.innerHTML = `<tr><td colspan="9" class="text-center text-danger">Error loading data</td></tr>`;
}
}

function handleRefreshButton() {
const refreshBtn = document.getElementById('refreshBtn');
const COOLDOWN_SECONDS = 60;
const LAST_REFRESH_KEY = 'lastPermitRefresh';

function getRemainingCooldown() {
  const lastRefresh = localStorage.getItem(LAST_REFRESH_KEY);
  if (!lastRefresh) return 0;
  const elapsed = (Date.now() - parseInt(lastRefresh, 10)) / 1000;
  return Math.max(0, COOLDOWN_SECONDS - Math.floor(elapsed));
}

```

```
}

function startCooldown() {
  localStorage.setItem(LAST_REFRESH_KEY, Date.now().toString());
  let remaining = COOLDOWN_SECONDS;
  refreshBtn.disabled = true;
  const originalText = "↻ Refresh";

  const interval = setInterval(() => {
    remaining--;
    refreshBtn.innerText = `Wait ${remaining}s`;
    if (remaining <= 0) {
      clearInterval(interval);
      refreshBtn.disabled = false;
      refreshBtn.innerText = originalText;
    }
  }, 1000);
}

// On load, check if cooldown is active
const remainingCooldown = getRemainingCooldown();
if (remainingCooldown > 0) {
  refreshBtn.disabled = true;
  refreshBtn.innerText = `Wait ${remainingCooldown}s`;

  let remaining = remainingCooldown;
  const interval = setInterval(() => {
    remaining--;
    refreshBtn.innerText = `Wait ${remaining}s`;
    if (remaining <= 0) {
      clearInterval(interval);
      refreshBtn.disabled = false;
      refreshBtn.innerText = "↻ Refresh";
    }
  }, 1000);
}

refreshBtn.addEventListener('click', async () => {
  const remaining = getRemainingCooldown();
  if (remaining > 0) {
    NotificationBox.show(`Please wait ${remaining}s before refreshing again.`, 'error');
    return;
  }

  try {
    refreshBtn.disabled = true;
    refreshBtn.innerText = "Refreshing...";
    await Permit.refreshCache();
    renderPermitTable(currentPage);
    NotificationBox.show("Refreshed successfully.");
    startCooldown();
  } catch (err) {
    refreshBtn.disabled = false;
  }
})
```

```

refreshBtn.innerText = "⟳ Refresh";
console.error(err);
NotificationBox.show("Error during refresh.", "error");
}

};

}

async function loadPermit(){
Spinner.show();
try{
  await renderPermitTable(currentPage);
} finally {
  Spinner.hide();
}
}

function populateEditModal(permit) {
  document.getElementById('editPermitForm').setAttribute('data-id', permit.id);
  // Basic Info
  document.getElementById('editPermitNo').value = permit.permitNo || "";
  document.getElementById('editPermittee').value = permit.permittee || "";
  document.getElementById('editMailingAddress').value = permit.mailingAddress || "";
  document.getElementById('editDiversionPoint').value = permit.diversionPoint || "";

  // Parse Latitude (Decimal to DMS)
  if (permit.latitude) {
    const { degrees, minutes, seconds, direction } = CoordinateUtils.decimalToDMS(permit.latitude, 'lat');
    document.getElementById('editLatDegrees').value = degrees;
    document.getElementById('editLatMinutes').value = minutes;
    document.getElementById('editLatSeconds').value = seconds;
  }

  // Parse Longitude (Decimal to DMS)
  if (permit.longitude) {
    const { degrees, minutes, seconds, direction } = CoordinateUtils.decimalToDMS(permit.longitude, 'lon');
    document.getElementById('editLonDegrees').value = degrees;
    document.getElementById('editLonMinutes').value = minutes;
    document.getElementById('editLonSeconds').value = seconds;
  }

  // Technical Info
  document.getElementById('editWaterSource').value = permit.waterSource || "";
  document.getElementById('editWaterDiversion').value = permit.waterDiversion || "";
  document.getElementById('editFlowRate').value = permit.flowRate || "";
  document.getElementById('editPeriodOfUse').value = permit.periodOfUse || "";
  document.getElementById('editPurpose').value = permit.purpose || "";
  document.getElementById('editVisited').checked = !!permit.visited;
  document.getElementById('editIsCancelled').checked = !!permit.cancelled;
  document.getElementById('editPdfExistingUrl').value = permit.pdfUrl || "";
}

function initializePermitUpdate() {
  const updateBtn = document.getElementById('updatePermitBtn');

```

```
if (!updateBtn) {
  console.error('Update button not found.');
  return;
}

updateBtn.addEventListener('click', async (e) => {
  e.preventDefault();
  Spinner.show();

  try {
    const form = document.getElementById('editPermitForm');
    if (!form) {
      console.error('Edit form not found.');
      return;
    }

    if (!form.checkValidity()) {
      form.classList.add('was-validated');
      return;
    }

    const id = form.getAttribute('data-id');
    if (!id) {
      console.error('No document ID found for update.');
      return;
    }

    // Handle Coordinates
    const latDegrees = parseFloat(document.getElementById('editLatDegrees').value) || 0;
    const latMinutes = parseFloat(document.getElementById('editLatMinutes').value) || 0;
    const latSeconds = parseFloat(document.getElementById('editLatSeconds').value) || 0;
    const latDirection = document.getElementById('editLatDirection').value.trim();

    const lonDegrees = parseFloat(document.getElementById('editLonDegrees').value) || 0;
    const lonMinutes = parseFloat(document.getElementById('editLonMinutes').value) || 0;
    const lonSeconds = parseFloat(document.getElementById('editLonSeconds').value) || 0;
    const lonDirection = document.getElementById('editLonDirection').value.trim();

    const latitudeDecimal = CoordinateUtils.dmsToDecimal(latDegrees, latMinutes, latSeconds, latDirection);
    const longitudeDecimal = CoordinateUtils.dmsToDecimal(lonDegrees, lonMinutes, lonSeconds, lonDirection);

    // Handle PDF Upload
    const fileInput = document.getElementById('editPdfAttachment');
    let permitFileUrl = document.getElementById('editPdfExistingUrl').value || "";

    if (fileInput && fileInput.files.length > 0) {
      const file = fileInput.files[0];

      // Delete existing file before uploading a new one
      if (permitFileUrl) {
        const deleted = await FileService.deleteFileFromPermitBucket(permitFileUrl);
        if (!deleted) {
          throw new Error('Failed to delete existing PDF before uploading new one.');
        }
      }
    }
  } catch (error) {
    console.error(error);
  }
})
```

```

        }

    // Upload the new file
    const uploadedUrl = await FileService.uploadPermitFile(file);

    if (uploadedUrl) {
        permitFileUrl = uploadedUrl;
    } else {
        throw new Error('Failed to upload permit file.');
    }
}

// Prepare data for update
const data = {
    permitNo: document.getElementById('editPermitNo').value.trim(),
    permittee: document.getElementById('editPermittee').value.trim(),
    mailingAddress: document.getElementById('editMailingAddress').value.trim(),
    diversionPoint: document.getElementById('editDiversionPoint').value.trim(),

    latitude: latitudeDecimal,
    longitude: longitudeDecimal,

    waterSource: document.getElementById('editWaterSource').value.trim(),
    waterDiversion: document.getElementById('editWaterDiversion').value.trim(),
    flowRate: parseFloat(document.getElementById('editFlowRate').value) || 0,
    periodOfUse: document.getElementById('editPeriodOfUse').value.trim(),
    purpose: document.getElementById('editPurpose').value.trim(),
    visited: document.getElementById('editVisited').checked,
    cancelled: document.getElementById('editIsCancelled').checked,
};

pdfUrl: permitFileUrl
};

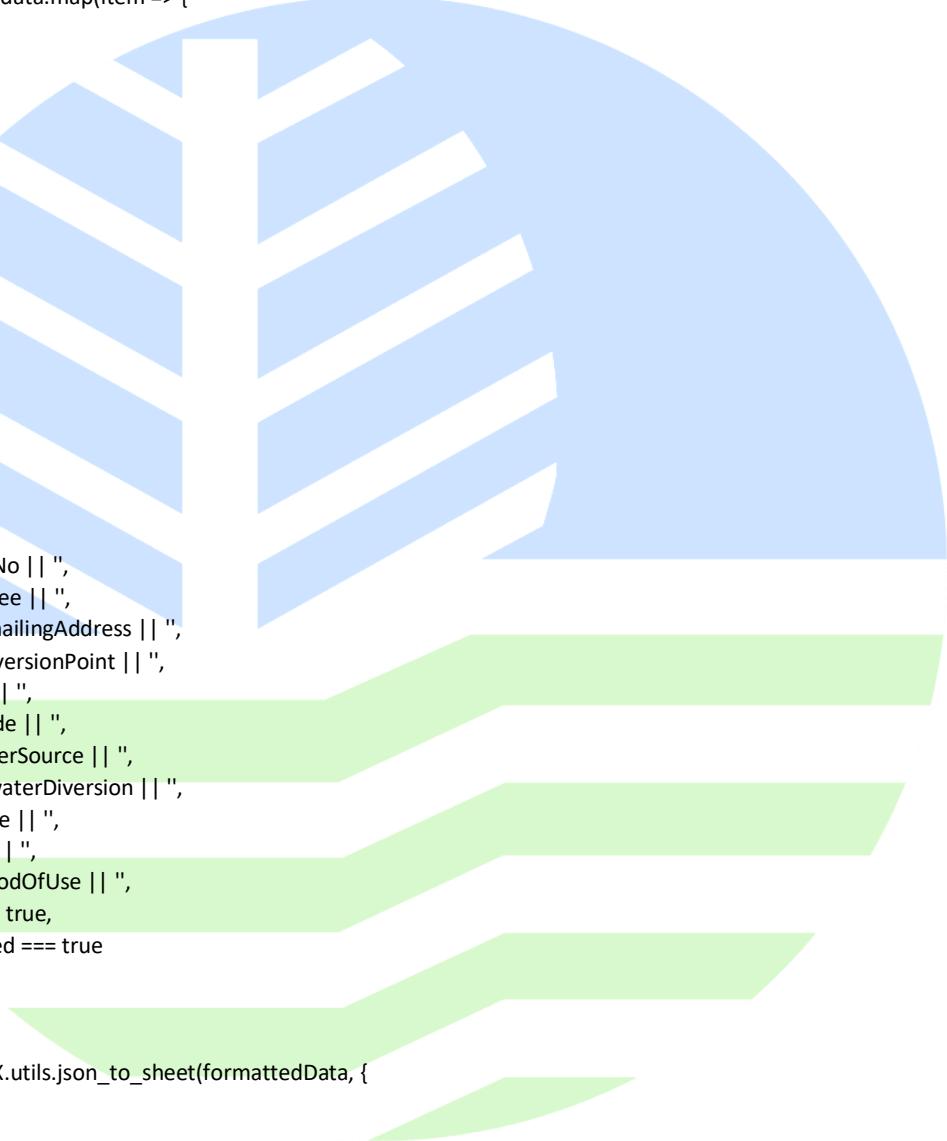
// Update Firestore
await Permit.update(id, data);
renderPermitTable(currentPage);
NotificationBox.show('Permit updated successfully');

const editModal = bootstrap.Modal.getInstance(document.getElementById('editPermitModal'));
if (editModal) editModal.hide();

form.classList.remove('was-validated');

} catch (error) {
    console.error('Error updating permit:', error);
    NotificationBox.show('Failed to update permit. Please try again.', 'error');
} finally {
    Spinner.hide();
}
});
}
}

```



```
function setupExportButtonListener() {
  const exportBtn = document.getElementById('exportPermitsBtn');
  if (!exportBtn) return;

  exportBtn.addEventListener('click', () => {
    const data = JSON.parse(localStorage.getItem('cachedPermitsExcel')) || [];
    if (!data || data.length === 0) return;

    const formattedData = data.map(item => {
      const {
        permitNo,
        permittee,
        mailingAddress,
        diversionPoint,
        latitude,
        longitude,
        waterSource,
        waterDiversion,
        flowRate,
        purpose,
        periodOfUse,
        visited,
        cancelled
      } = item;

      return {
        "Permit No": permitNo || '',
        "Permittee": permittee || '',
        "Mailing Address": mailingAddress || '',
        "Diversion Point": diversionPoint || '',
        "Latitude": latitude || '',
        "Longitude": longitude || '',
        "Water Source": waterSource || '',
        "Water Diversion": waterDiversion || '',
        "Flow Rate": flowRate || '',
        "Purpose": purpose || '',
        "Period of Use": periodOfUse || '',
        __visited: visited === true,
        __cancelled: cancelled === true
      };
    });
  });

  const worksheet = XLSX.utils.json_to_sheet(formattedData, {
    skipHeader: false
  });

  const range = XLSX.utils.decode_range(worksheet['!ref']);
  for (let R = 1; R <= range.e.r; ++R) {
    const visited = formattedData[R - 1].__visited;
    if (visited) {
      for (let C = 0; C <= range.e.c; ++C) {
        const cellAddress = XLSX.utils.encode_cell({ r: R, c: C });
        if (!worksheet[cellAddress]) continue;
      }
    }
  }
}
```

```

        worksheet[cellAddress].s = {
          fill: { fgColor: { rgb: "DFF0D8" } }
        };
      }
    }
}

formattedData.forEach(row => delete row.__visited);

const workbook = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(workbook, worksheet, "Permits");
XLSX.writeFile(workbook, "permits-export.xlsx");
});

}

function setupImageUploadModal(permitId, geotaggedUrl = "", permitNo = "") {
  const imageInput = document.getElementById('imageInput');
  const imagePreview = document.getElementById('imagePreview');
  const modal = document.getElementById('imageUploadModal');
  const dropZone = document.getElementById('dropZone');
  const uploadBtn = document.getElementById('uploadGeotaggedBtn');
  const placeholderURL = './images/placeholder2.png';
  const viewImageLink = document.getElementById('viewImageLink');
  const hiddenInput = document.getElementById('currentGeotaggedUrl');

  const setImage = (src, isPlaceholder = true) => {
    imagePreview.src = src;
    imagePreview.classList.toggle('placeholder', isPlaceholder);
    uploadBtn.classList.toggle('d-none', isPlaceholder);
  };

  const handleFile = (file) => {
    if (file && file.type.startsWith('image/')) {
      const reader = new FileReader();
      reader.onload = (e) => setImage(e.target.result, false);
      reader.readAsDataURL(file);
    }
  };
}

imageInput.onchange = () => {
  const file = imageInput.files[0];
  handleFile(file);
};

// Drag & drop
['dragenter', 'dragover'].forEach(eventName => {
  dropZone.addEventListener(eventName, e => {
    e.preventDefault();
    dropZone.classList.add('border-primary');
  });
});

['dragleave', 'drop'].forEach(eventName => {

```

```
dropZone.addEventListener(eventName, e => {
  e.preventDefault();
  dropZone.classList.remove('border-primary');
});

dropZone.ondrop = (e) => {
  const file = e.dataTransfer.files[0];
  if (file) {
    handleFile(file);
  }
};

modal.addEventListener('show.bs.modal', () => {
  setImage(placeholderURL, true);
  hiddenInput.value = geotaggedUrl || '';
  document.getElementById('currentPermitNo').value = permitNo || '';

  if (geotaggedUrl) {
    viewImageLink.href = geotaggedUrl;
    viewImageLink.classList.remove('d-none');
    createDeleteButtonIfNeeded(geotaggedUrl);
  } else {
    viewImageLink.classList.add('d-none');
    removeDeleteButton();
  }
}, { once: true });

// Replace upload button
uploadBtn.replaceWith(uploadBtn.cloneNode(true));
const freshUploadBtn = document.getElementById('uploadGeotaggedBtn');
freshUploadBtn.classList.add('d-none');
freshUploadBtn.addEventListener('click', () => {
  const permitNo = document.getElementById('currentPermitNo').value;
  handleGeotaggedUpload(permitId, permitNo);
});

imageInput.onchange = () => {
  const file = imageInput.files[0];
  if (file) {
    const reader = new FileReader();
    reader.onload = (e) => {
      imagePreview.src = e.target.result;
      imagePreview.classList.remove('placeholder');
      freshUploadBtn.classList.remove('d-none');
    };
    reader.readAsDataURL(file);
  }
};

// Add a delete button only if it doesn't exist
function createDeleteButtonIfNeeded() {
  if (document.getElementById('deleteGeotaggedBtn')) return;
```

```
const deleteBtn = document.createElement('button');
deleteBtn.id = 'deleteGeotaggedBtn';
deleteBtn.className = 'btn btn-3d btn-outline-danger';
deleteBtn.innerHTML = '<i class="bi bi-trash me-1"></i> Delete Image';

deleteBtn.addEventListener('click', async () => {
  Spinner.show();
  try {
    try {
      deleteBtn.disabled = true;
      deleteBtn.innerHTML = '<span class="spinner-border spinner-border-sm me-1"></span> Deleting...';

      // Delete from storage
      await GeotaggedFileService.delete(hiddenInput.value);

      await Permit.update(permitId, {
        geotaggedUrl: '',
        visited: false
      });

      const wusEntries = await WUSData.fetchAllDesc();
      const matchingEntry = wusEntries.find(entry => entry.permitNo === permitNo);

      if (matchingEntry) {
        await WUSData.update(matchingEntry.id, {
          geotaggedUrl: ''
        });
      }

      setImage(placeholderURL, true);
      hiddenInput.value = '';
      viewImageLink.classList.add('d-none');
      removeDeleteButton();
      renderPermitTable(currentPage);

      NotificationBox.show('Image deleted and permit updated.');

    } catch (err) {
      NotificationBox.show(`Failed to delete image: ${err.message}`, 'error');
    } finally {
      deleteBtn.disabled = false;
      deleteBtn.innerHTML = '<i class="bi bi-trash me-1"></i> Delete Image';
    }
  } finally{
    Spinner.hide();
  };
});

viewImageLink.parentElement.appendChild(deleteBtn);
}
```

```
// Remove delete button if exists
function removeDeleteButton() {
  const deleteBtn = document.getElementById('deleteGeotaggedBtn');
  if (deleteBtn) deleteBtn.remove();
}

async function handleGeotaggedUpload(permitId, permitNo) {
  Spinner.show();

  try {
    const fileInput = document.getElementById("imageInput");
    const file = fileInput.files[0];

    if (!file) {
      NotificationBox.show("No file selected.", "error");
      return;
    }

    if (!permitId) {
      NotificationBox.show("Permit ID is required.", "error");
      return;
    }

    // Fallback to hidden input if permitNo wasn't passed
    if (!permitNo) {
      permitNo = document.getElementById('currentPermitNo').value;
    }

    const existingUrl = document.getElementById('currentGeotaggedUrl').value;

    if (existingUrl) {
      await GeotaggedFileService.delete(existingUrl);
    }

    const newImageUrl = await GeotaggedFileService.upload(file);
    if (!newImageUrl) {
      NotificationBox.show("Upload failed.", "error");
      return;
    }

    // Update Permit document
    await Permit.update(permitId, {
      geotaggedUrl: newImageUrl,
      visited: true
    });

    // Update corresponding WUSData entry using permitNo
    const wusEntries = await WUSData.fetchAllDesc();
    const matchingEntry = wusEntries.find(entry => entry.permitNo === permitNo);

    if (matchingEntry) {
      await WUSData.update(matchingEntry.id, {
        geotaggedUrl: newImageUrl
      });
    }
  } catch (error) {
    console.error(error);
  }
}
```

```
        geotaggedUrl: newImageUrl
    });
}

renderPermitTable(currentPage);
NotificationBox.show("Geotagged image updated successfully.");

// Close the modal
const modalEl = document.getElementById('imageUploadModal');
const modal = bootstrap.Modal.getInstance(modalEl);
if (modal) modal.hide();
} catch (error) {
    console.error("Error during upload:", error.message);
    NotificationBox.show("An error occurred: " + error.message, 'error');
} finally {
    Spinner.hide();
}
}
```

XV.C.3.h. Analytics Module

summary-init.js

```
import { SessionGuard } from './auth/auth.js';
import {
    initDOMElements,
    initUI,
    loadData,
    renderTable
} from './summary-ui.js';

document.addEventListener("DOMContentLoaded", async () => {
    SessionGuard.ensureLoggedIn();
    initDOMElements();
    initUI();
    await loadData();
    renderTable();
});
```

summary-ui.js

```
import { generatePdfICS } from './pdf/ics-pdf.js';
import { generateConsumablePDF } from './pdf/user-consumable-pdf.js'
import { Consumable } from './data/cache/consumable-data.js';
import { Ledger } from './data/cache/ledger-data.js';
import { ICS } from './data/cache/ics-data.js';
import { Users } from './data/cache/user-data.js';
import { Sidebar } from './components/sidebar.js';
import { Spinner } from './components/spinner.js';
import { updateReportTimestamp } from './utils/dateFormat.js';

// Global variables
```



```
let users = [];
let filteredUsers = [];
let currentPage = 1;
const usersPerPage = 10;

// INIT FUNCTIONS
let tableBody, searchInput, pageContent, paginationNav, modalElement, bsModal;

export function initDOMElements() {
  tableBody = document.getElementById("usersTableBody");
  searchInput = document.getElementById("searchInput");
  pageContent = document.getElementById("page-content");
  modalElement = document.getElementById("consumableModal");

  if (!modalElement) {
    console.error("Modal element #consumableModal not found!");
    return;
  }

  bsModal = new bootstrap.Modal(modalElement);

  paginationNav = document.createElement("nav");
  paginationNav.className = "d-flex justify-content-center mt-3";
  pageContent.appendChild(paginationNav);
}

export function initUI() {
  Sidebar.render();
  Spinner.render();
  hideSearchIfNotAdmin();
  setupSearchHandler();
  setupGenerateICSHandler();
  renderLatestLedger();
  renderCriticalLowSupplyChart("lowSupplyChartContainer");
  renderPriorityTable();
}

export async function loadData() {
  Spinner.show();
  try {
    users = await Users.fetchUsersSummary();
    filteredUsers = [...users];
  } finally {
    Spinner.hide();
  }
}

function setupSearchHandler() {
  searchInput?.addEventListener("input", () => {
    const query = searchInput.value.trim().toLowerCase();
    filteredUsers = users.filter(u => {
      const fullName = `${u.lastName}, ${u.firstName} ${u.middleInitial}`.toLowerCase();
      return fullName.includes(query);
    });
  });
}
```

```
});

currentPage = 1;
renderTable();
};

}

function hideSearchIfNotAdmin() {
  const userRole = localStorage.getItem("userRole");
  if (userRole !== "admin") {
    searchInput?.style.setProperty("display", "none");
  }
}

export function renderTable() {
  Spinner.show();
  try {
    tableBody.innerHTML = "";

    const currentUserId = localStorage.getItem("wrusUserId");
    const isAdmin = currentUserId === "admin";

    const visibleUsers = isAdmin
      ? filteredUsers
      : filteredUsers.filter(user =>
        user.id === currentUserId || (!user.type || user.type.trim() === ""))
    );

    const start = (currentPage - 1) * usersPerPage;
    const pageUsers = visibleUsers.slice(start, start + usersPerPage);

    pageUsers.forEach(user => {
      const row = document.createElement("tr");
      row.appendChild(createNameCell(user));
      row.appendChild(createConsumableCell(user));
      row.appendChild(createICSCell(user));
      tableBody.appendChild(row);
    });

    renderPagination(Math.ceil(visibleUsers.length / usersPerPage));
  } finally {
    Spinner.hide();
  }
}

function createNameCell(user) {
  const cell = document.createElement("td");
  cell.textContent = `${user.lastName}, ${user.firstName} ${user.middleInitial}.`;
  cell.classList.add("fw-medium", "text-capitalize");
  return cell;
}

function createConsumableCell(user) {
  const cell = document.createElement("td");
```

```
cell.classList.add("text-center");

const btn = document.createElement("button");
btn.className = "btn btn-3d btn-water btn-sm px-2 py-1 fw-semibold rounded";
btn.innerHTML = `Consumable`;
btn.setAttribute("data-id", user.id);

btn.addEventListener("click", async () => {
  Spinner.show();
  try {
    const [consumablesMap, ledgerEntries] = await Promise.all([
      Consumable.fetchConsumablesMap(),
      Ledger.fetchLedgerByUser(user.id)
    ]);

    ledgerEntries.sort((a, b) => (b.dateModified?.toMillis() || 0) - (a.dateModified?.toMillis() || 0));

    await generateConsumablePDF(user, ledgerEntries, consumablesMap);
    bsModal.show();
  } finally {
    Spinner.hide();
  }
});

cell.appendChild(btn);
return cell;
}

function createICSCell(user) {
  const cell = document.createElement("td");
  cell.classList.add("text-center");

  if (user.type !== "Permanent") {
    return cell;
  }

  const btn = document.createElement("button");
  btn.className = "btn btn-3d btn-water-alt btn-sm px-2 py-1 fw-semibold rounded";
  btn.innerHTML = `ICS`;
  btn.setAttribute("data-id", user.id);

  btn.addEventListener("click", () => {
    showPropertyModal({
      lastName: user.lastName,
      firstName: user.firstName,
      middleInitial: user.middleInitial
    }, user.id);
  });

  cell.appendChild(btn);
  return cell;
}
```

```
function renderPagination(totalPages) {
  const paginationNav = document.getElementById("paginationNav");
  if (!paginationNav) {
    console.warn("Pagination container #paginationNav not found");
    return;
  }

  paginationNav.innerHTML = "";
  const ul = document.createElement("ul");
  ul.className = "pagination";

  const addItem = (label, disabled, onClick, active = false) => {
    const li = document.createElement("li");
    li.className = `page-item ${disabled ? "disabled" : ""} ${active ? "active" : ""}`;

    const btn = document.createElement("button");
    btn.className = "page-link";
    btn.textContent = label;
    btn.onclick = onClick;

    li.appendChild(btn);
    ul.appendChild(li);
  };

  addItem("Previous", currentPage === 1, () => {
    if (currentPage > 1) {
      currentPage--;
      renderTable();
    }
  });
}

let startPage = Math.max(1, currentPage - 2);
let endPage = startPage + 4;

if (endPage > totalPages) {
  endPage = totalPages;
  startPage = Math.max(1, endPage - 4);
}

for (let i = startPage; i <= endPage; i++) {
  addItem(i, false, () => {
    currentPage = i;
    renderTable();
  }, i === currentPage);
}

addItem("Next", currentPage === totalPages, () => {
  if (currentPage < totalPages) {
    currentPage++;
    renderTable();
  }
});
```

```

    paginationNav.appendChild(ul);
}

async function showPropertyModal(userFullName, userId) {
  document.getElementById("fullName").textContent =
    `${userFullName.lastName}, ${userFullName.firstName} ${userFullName.middleInitial}.`;

  const generateBtn = document.getElementById("generatePdfBtn");
  generateBtn.dataset.userid = userId;

  const icsTableBody = document.getElementById("icsTableBody");
  const totalAmountSpan = document.getElementById("totalAmount");

  icsTableBody.innerHTML = "";
  totalAmountSpan.textContent = "₱0.00";

  const allICSIItems = await ICS.getICSDataByUserId(userId);
  const icsItems = allICSIItems.filter(item => item.status !== 'RTS');

  let totalAmount = 0;
  icsItems.forEach(item => {
    totalAmount += item.totalCost;

    const row = document.createElement("tr");
    row.innerHTML = `
      <td>${item.qty}</td>
      <td>${item.unit}</td>
      <td>${item.description}</td>
      <td>${item.serialNo}</td>
      <td>₱${item.unitCost.toLocaleString(undefined, { minimumFractionDigits: 2 })}</td>
      <td>₱${item.totalCost.toLocaleString(undefined, { minimumFractionDigits: 2 })}</td>
      <td>${item.ICSno}</td>
      <td>${item.dateIssued}</td>
      <td>${item.remarks}</td>
      <td>
        ${item.attachmentURL}
        ? '<a href="${item.attachmentURL}" target="_blank" class="btn btn-3d btn-sm btn-outline-primary">View PDF</a>'
        : '<span class="text-muted">N/A</span>'
      </td>
    `;
    icsTableBody.appendChild(row);
  });

  totalAmountSpan.textContent = `₱${totalAmount.toLocaleString(undefined, { minimumFractionDigits: 2 })}`;

  const modal = new bootstrap.Modal(document.getElementById("propertyModal"));
  modal.show();
}

function setupGenerateICSHandler() {
  document.getElementById("generatePdfBtn")?.addEventListener("click", () => {
    const userId = document.getElementById("generatePdfBtn").dataset.userid;
    if (userId) {

```

```

        generatePdfICS(userId);
    }
});
}

async function renderLatestLedger(containerId = "ledgerContainer") {
    const container = document.getElementById(containerId);
    if (!container) {
        console.error(`Container with ID "${containerId}" not found.`);
        return;
    }

    Spinner.show();

    try {
        container.innerHTML = "<p>Loading latest ledger entries...</p>";

        let ledgerEntries = await Ledger.fetchAll();
        ledgerEntries = ledgerEntries.slice(0, 10);

        if (ledgerEntries.length === 0) {
            container.innerHTML = "<p>No ledger entries found.</p>";
            return;
        }

        const consumableMap = await Consumable.fetchConsumablesMap();
        const usersMap = await Users.getUsersMap();

        container.innerHTML = "";

        // Responsive wrapper
        const responsiveWrapper = document.createElement("div");
        responsiveWrapper.className = "table-responsive";
        container.appendChild(responsiveWrapper);

        // Table
        const table = document.createElement("table");
        table.className = "table table-striped table-bordered table-hover align-middle";

        // Table header
        const thead = document.createElement("thead");
        thead.innerHTML = `
            <tr class="text-center align-middle">
                <th scope="col">Date Modified</th>
                <th scope="col">Consumable</th>
                <th scope="col">Action</th>
                <th scope="col" class="text-end">Amount</th>
                <th scope="col">Remarks</th>
                <th scope="col">Assigned To</th>
            </tr>
        `;
        table.appendChild(thead);
    }
}

```

```
// Table body
const tbody = document.createElement("tbody");

for (const entry of ledgerEntries) {
  const consumableSpec = consumableMap[entry.cid]?.specification || entry.cid;

  // --- Refactored date formatting (MMMM dd, YYYY + hh:mm:ss AM/PM) ---
  let dateStr = "N/A";
  if (entry.dateModified) {
    let dateObj;
    if (entry.dateModified.seconds) {
      dateObj = new Date(entry.dateModified.seconds * 1000);
    } else if (entry.dateModified.toDate) {
      dateObj = entry.dateModified.toDate();
    } else if (entry.dateModified instanceof Date) {
      dateObj = entry.dateModified;
    } else {
      dateObj = new Date(entry.dateModified);
    }

    if (!isNaN(dateObj)) {
      dateStr = dateObj.toLocaleString("en-US", {
        month: "long", // MMMM
        day: "2-digit", // dd
        year: "numeric", // YYYY
        hour: "2-digit",
        minute: "2-digit",
        second: "2-digit",
        hour12: true // AM/PM
      });
    }
  }
}

// -----
const assignedToUsername = entry.assignedTo && usersMap[entry.assignedTo]
  ? usersMap[entry.assignedTo]
  : entry.assignedTo || "-";

const tr = document.createElement("tr");
tr.innerHTML = `
<td class="text-center">${dateStr}</td>
<td>${consumableSpec}</td>
<td class="text-center">${entry.action || ""}</td>
<td class="text-end">${entry.amount ?? ""}</td>
<td>${entry.remarks || ""}</td>
<td>${assignedToUsername}</td>
`;;
tbody.appendChild(tr);
}

table.appendChild(tbody);
responsiveWrapper.appendChild(table);
```

```

} catch (error) {
  console.error("Error rendering latest ledger:", error);
  if (container) container.innerHTML = "<p>Error loading ledger data.</p>";
} finally {
  Spinner.hide();
}
}

async function renderCriticalLowSupplyChart(containerId = "chartsContainer") {
  const container = document.getElementById(containerId);
  if (!container) {
    console.error(`Container with ID "${containerId}" not found.`);
    return;
  }
  Spinner.show();

  try {
    const consumables = await Consumable.fetchAll();

    // Filter all consumables with qty <= 10, sorted ascending
    const criticalItems = consumables
      .filter(item => item.qty <= 10)
      .sort((a, b) => a.qty - b.qty);

    if (criticalItems.length === 0) {
      container.innerHTML = "<p>No critically low supply items found.</p>";
      return;
    }

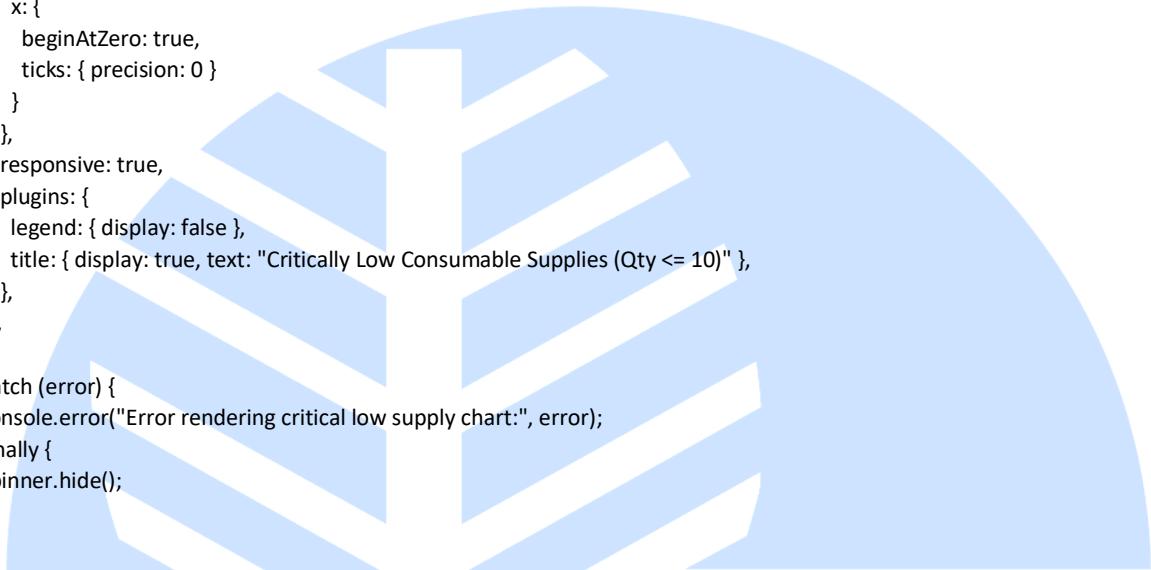
    const labels = criticalItems.map(item => item.specification || item.id);
    const data = criticalItems.map(item => item.qty);

    // Clear previous canvas and list if exists
    let oldCanvas = document.getElementById("chartCriticalLowSupply");
    if (oldCanvas) container.removeChild(oldCanvas);
    let oldList = document.getElementById("criticalLowSupplyList");
    if (oldList) container.removeChild(oldList);

    // Create canvas for chart
    const canvas = document.createElement("canvas");
    canvas.id = "chartCriticalLowSupply";
    canvas.style.marginBottom = "1rem";
    container.appendChild(canvas);

    // Create horizontal bar chart
    new Chart(canvas, {
      type: "bar",
      data: {
        labels,
        datasets: [
          {
            label: "Qty",
            data,
          }
        ]
      }
    });
  }
}

```



```

backgroundColor: "rgba(255, 99, 132, 0.6)",
borderColor: "rgba(255, 99, 132, 1)",
borderWidth: 1,
},
},
options: {
indexAxis: 'y', // horizontal bar chart
scales: {
x: {
beginAtZero: true,
ticks: { precision: 0 }
}
},
responsive: true,
plugins: {
legend: { display: false },
title: { display: true, text: "Critically Low Consumable Supplies (Qty <= 10)" },
},
},
});
} catch (error) {
console.error("Error rendering critical low supply chart:", error);
} finally {
Spinner.hide();
}
}

async function renderPriorityTable() {
const tbody = document.getElementById("priorityBody");
tbody.innerHTML = `<tr><td colspan="10">Loading...</td></tr>`; // now 10 columns

const consumables = await Consumable.fetchAll();
const ledgerEntries = await Ledger.fetchAll();

// Filter only priority items
const priorityItems = consumables.filter(c => c.priority === true);

// Determine rolling window based on earliest ledger entry date
let earliestDate = new Date();
ledgerEntries.forEach(l => {
const date = l.dateModified instanceof Date
? l.dateModified
: (l.dateModified?.seconds ? new Date(l.dateModified.seconds * 1000) : new Date(l.dateModified));

if (date < earliestDate) {
earliestDate = date;
}
});

const today = new Date();
const oneYearAgo = new Date(today);
oneYearAgo.setFullYear(today.getFullYear() - 1);

```

```

// Window starts from earliest data OR 12 months ago, whichever is later
const windowStart = earliestDate > oneYearAgo ? earliestDate : oneYearAgo;

// Calculate number of months in the window (avoid zero)
const monthDiff = (today.getFullYear() - windowStart.getFullYear()) * 12 +
    (today.getMonth() - windowStart.getMonth());
const monthsInWindow = Math.max(1, monthDiff || 1);

// Lead time threshold in months
const leadTimeMonths = 12;

// Build rows with extra 'daysLeft' for sorting
const data = priorityItems.map(item => {
    // Filter ledger entries for this CID within the rolling window
    const totalConsumed = ledgerEntries
        .filter(l => {
            const date = l.dateModified instanceof Date
                ? l.dateModified
                : (l.dateModified?.seconds ? new Date(l.dateModified.seconds * 1000) : new Date(l.dateModified));
            return (
                l.cid === item.id &&
                l.action === "Assign Item" &&
                date >= windowStart &&
                date <= today
            );
        })
        .reduce((sum, l) => sum + (l.amount || 0), 0);

    // Calculate lifespan in years/months
    let lifespanText = "";
    if (totalConsumed > 0) {
        const years = item.qty / (totalConsumed / monthsInWindow * 12); // normalized to yearly rate
        const wholeYears = Math.floor(years);
        const months = Math.round((years - wholeYears) * 12);

        if (wholeYears > 0 && months > 0) {
            lifespanText = `${wholeYears} yr ${months} mo`;
        } else if (wholeYears > 0) {
            lifespanText = `${wholeYears} yr`;
        } else {
            lifespanText = `${months} mo`;
        }
    }

    // Analytics
    const avgMonthly = totalConsumed / monthsInWindow;
    const daysLeft = avgMonthly > 0 ? (item.qty / avgMonthly) * 30 : Infinity;
    const reorderPoint = avgMonthly * leadTimeMonths;
    const needsReorder = item.qty < reorderPoint;

    // Recommended reorder qty (rounded up, min 0)
    let recommended = Math.max(0, Math.ceil(12 * avgMonthly - item.qty));
}

```

```
const recommendedAddUp = Math.ceil(recommended*(0.1));
recommended = recommended + recommendedAddUp;
recommended = `${recommended} ${item.unit}`;

return {
  item,
  totalConsumed,
  lifespanText,
  avgMonthly,
  daysLeft,
  needsReorder,
  recommended
};
});

// Sort by daysLeft ascending (lowest first, Infinity last)
data.sort((a, b) => {
  if (a.daysLeft === Infinity && b.daysLeft === Infinity) return 0;
  if (a.daysLeft === Infinity) return 1;
  if (b.daysLeft === Infinity) return -1;
  return a.daysLeft - b.daysLeft;
});

// Render sorted rows
const rows = data.map(d =>
  <tr class="${d.needsReorder ? 'reorder-warning' : ''}">
    <td>${d.item.id}</td>
    <td>${d.item.specification}</td>
    <td>${d.item.unit}</td>
    <td>${d.item.qty}</td>
    <td>${d.totalConsumed}</td>
    <td>${d.lifespanText || ""}</td>
    <td>${d.avgMonthly.toFixed(1)}</td>
    <td>${d.daysLeft === Infinity ? "" : d.daysLeft.toFixed(0) + " days"}</td>
    <td>${d.needsReorder ? "⚠ Yes" : "No"}</td>
    <td>${d.recommended}</td>
  </tr>
).join("");

tbody.innerHTML = rows || `<tr><td colspan="10">No priority items found.</td></tr>`;
}
```

XV.C.3.i Image Upload Scripts

supabaseInit.js

```
import { createClient } from "https://cdn.jsdelivr.net/npm/@supabase/supabase-js/+esm";

const supabaseUrl = 'https://vkffjcptpznbidjnubqx.supabase.co';
const supabaseKey =
'eyJhbGciOiJIUzI1*****';
*****;
```

```
export const supabase = createClient(supabaseUrl, supabaseKey);
```

supabaseInit2.js

```
import { createClient } from "https://cdn.jsdelivr.net/npm/@supabase/supabase-js/+esm";
```

```
const supabaseUrl = 'https://pbjtlrgmaclqgjzpimaz.supabase.co';
const supabaseKey =
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX*****';
```

```
export const supabase = createClient(supabaseUrl, supabaseKey);
```

upload.js

```
import { supabase } from './supabaseInit.js';
import { NotificationBox } from './components/notification.js';
```

```
export const FileService = {
  async uploadICSFile(file) {
    const fileName = Date.now() + '_' + file.name;

    const { data, error } = await supabase
      .storage
      .from('ics-files')
      .upload(fileName, file);

    if (error) {
      console.error('ICS file upload failed:', error.message);
      NotificationBox.show('ICS file upload failed.');
      return null;
    }
  }
};
```

```
const publicURL = supabase
  .storage
  .from('ics-files')
  .getPublicUrl(data.path).data.publicUrl;
```

```
return publicURL;
},
```

```
async uploadPermitFile(file) {
  const fileName = Date.now() + '_' + file.name;

  const { data, error } = await supabase
    .storage
    .from('permit')
    .upload(fileName, file);

  if (error) {
    console.error('Permit file upload failed:', error.message);
    NotificationBox.show('Permit file upload failed.');
  }
}
```

```
return null;
}

const publicURL = supabase
  .storage
  .from('permit')
  .getPublicUrl(data.path).data.publicUrl;

return publicURL;
},

async deleteFileFromStorage(url, bucket = 'ics-files') {
  if (!url) return true;

  try {
    const decodedUrl = decodeURIComponent(url);
    const publicPrefix = `/storage/v1/object/public/${bucket}/`;
    const pathIndex = decodedUrl.indexOf(publicPrefix);

    if (pathIndex === -1) {
      console.warn(`⚠️ Couldn't find base path in URL for bucket ${bucket}`);
      return false;
    }

    let relativePath = decodedUrl.substring(pathIndex + publicPrefix.length);
    relativePath = relativePath.replace(/^\//, "").trim();

    const { error } = await supabase.storage
      .from(bucket)
      .remove([relativePath]);

    if (error) {
      console.error(`❌ Delete error from ${bucket}:`, error.message);
      return false;
    }
  }

  return true;
} catch (err) {
  console.error(`❌ Error during deleteFileFromStorage:`, err);
  return false;
},
};

async deleteFileFromPermitBucket(url, bucket = 'permit') {
  console.log(url);
  if (!url) return true;

  try {
    const decodedUrl = decodeURIComponent(url);
    const publicPrefix = `/storage/v1/object/public/${bucket}/`;
    const pathIndex = decodedUrl.indexOf(publicPrefix);
```



```
if (pathIndex === -1) {
  console.warn(`⚠️ Couldn't find base path in URL for bucket '${bucket}'`);
  return false;
}

let relativePath = decodedUrl.substring(pathIndex + publicPrefix.length);
relativePath = relativePath.replace(/^\//).trim();

console.log(`🗑️ Deleting file from ${bucket} bucket: ${relativePath}`);

const { error } = await supabase.storage
  .from(bucket)
  .remove([relativePath]);

if (error) {
  console.error(`❌ Delete error from '${bucket}': ${error.message}`);
  return false;
}

console.log(`✅ File deleted successfully.`);
return true;

} catch (err) {
  console.error(`❌ Error in deleteFileFromPermitBucket: ${err}`);
  return false;
}
};

};
```

uploadImage.js

```
// uploadImg.js
import { NotificationBox } from './components/notification.js';
import { supabase } from './supabaseInit_2.js';

export const GeotaggedFileService = {
  async upload(file) {
    if (!file || !(file instanceof File)) {
      NotificationBox.show("No valid file selected.");
      return null;
    }

    const options = {
      maxSizeMB: 0.1,
      maxWidthOrHeight: 1920,
      useWebWorker: true,
    };

    try {
      const compressedFile = await imageCompression(file, options);
    }
  }
};
```

```
// Clean up filename: remove or replace unsafe characters
const timestamp = Date.now();
const sanitizedFileName = `${timestamp}_${file.name.replace(/[^w\-\.\(\)]+/g, '')}.replace(/\s+/g, '_')}`;

const { data, error } = await supabase
  .storage
  .from('geotagged')
  .upload(sanitizedFileName, compressedFile);

if (error) {
  console.error("Upload failed:", error.message);
  NotificationBox.show("Upload failed: " + error.message);
  return null;
}

// No need to encode path here — Supabase handles it for the URL
const publicURL = supabase
  .storage
  .from('geotagged')
  .getPublicUrl(data.path).data.publicUrl;
return publicURL;
} catch (err) {
  console.error("Compression error:", err.message);
  NotificationBox.show("Compression failed: " + err.message);
  return null;
}
},
```

```
async delete(publicUrl) {
  if (!publicUrl) {
    console.warn("No URL provided for deletion.");
    return;
  }

  try {
    const decodedUrl = decodeURIComponent(publicUrl);
    const match = decodedUrl.match(/\^geotagged\^(.+)$/);

    if (!match || !match[1]) {
      console.error("Invalid geotagged URL format.");
      return;
    }

    const filePath = match[1].replace(/^\^/, ""); // Remove any leading slashes

    const { error } = await supabase.storage.from('geotagged').remove([filePath]);

    if (error) {
      console.error("Delete failed:", error.message);
      NotificationBox.show("Failed to delete old image.");
    } else {
```

```

        }
    } catch (err) {
        console.error("Error parsing URL for deletion:", err.message);
        NotificationBox.show("Failed to process deletion.");
    }
}
};

export async function uploadSignatureToSupabase(base64Data, fileName) {
    const blob = await (await fetch(base64Data)).blob();

    const { data, error } = await supabase.storage
        .from('signature')
        .upload(fileName, blob, {
            contentType: 'image/png',
            upsert: true
        });

    if (error) throw error;

    // Get public URL
    const { data: publicURLData } = supabase.storage.from('signature').getPublicUrl(fileName);
    return publicURLData.publicUrl;
}

```

XV.C.3.j. User Management Module

user-init.js

```

import { initializePage } from './user-ui.js';

document.addEventListener('DOMContentLoaded', () => {
    initializePage();
});

```

user-ui.js

```

import bcrypt from "https://esm.sh/bcryptjs@2.4.3";
import { Users } from '../data/cache/user-data.js'
import { Spinner } from '../components/spinner.js';
import { Sidebar } from '../components/sidebar.js';
import { AdminGuard } from '../auth/auth.js";
import { NotificationBox } from '../components/notification.js';

let currentPage = 1;
const usersPerPage = 10;
let allUsers = [];

export function initializePage(){
    Sidebar.render();
    AdminGuard.verify();
}

```

```

Spinner.render();
loadUsers();
initializeAddUserModal();
handleRefreshButton({
  buttonId: "refreshBtn",
  refreshFn: Users.refreshCache.bind(Users),
  renderFn: () => renderUsersTable(1),
  cooldownKey: "lastUsersRefresh"
});
handleSearchBar();
}

async function loadUsers() {
  Spinner.show();
  try {
    allUsers = await Users.fetchAllDesc();
    renderUsersTable();
  } finally {
    Spinner.hide();
  }
}

function initializeAddUserModal() {
  const showAddUserFormBtn = document.getElementById('showAddUserFormBtn');
  const addUserForm = document.getElementById('addUserForm');
  const addUserModalEl = document.getElementById('addUserModal');
  const modal = new bootstrap.Modal(addUserModalEl);

  // 👉 Handle opening the modal
  showAddUserFormBtn.addEventListener('click', () => {
    addUserForm.reset();
    addUserForm.classList.remove("was-validated");
    modal.show();
  });

  // 👉 Handle form submission
  addUserForm.addEventListener('submit', async function (event) {
    event.preventDefault();
    event.stopPropagation();

    const username = document.getElementById("username").value.trim();
    const password = document.getElementById("password").value;
    const confirmPassword = document.getElementById("confirmPassword").value;
    const lastName = document.getElementById("lastName").value.trim();
    const firstName = document.getElementById("firstName").value.trim();
    const middleInitial = document.getElementById("middleInitial").value.trim();
    const position = document.getElementById("position").value.trim();
    const duty = document.getElementById("duty").value;
    const natureOfAppointment = document.getElementById("natureOfAppointment").value;
    const status = document.querySelector('input[name="status"]':checked')?.value;

    // 🔒 Password match validation
  });
}

```

```
const confirmPasswordInput = document.getElementById("confirmPassword");
if (password !== confirmPassword) {
  confirmPasswordInput.setCustomValidity("Passwords do not match");
} else {
  confirmPasswordInput.setCustomValidity("");
}

this.classList.add("was-validated");

if (!this.checkValidity()) return;

Spinner.show();
try {
  const existingUsers = await Users.fetchAllDesc();

  // 🔍 Username duplication check
  const usernameExists = existingUsers.some(
    (user) => user.username?.toLowerCase() === username.toLowerCase()
  );
  if (usernameExists) {
    NotificationBox.show("Username already exists.", "error");
    return;
  }

  // 🔍 Supervisor uniqueness check
  if (duty === "Supervisor") {
    const supervisorExists = existingUsers.some(
      (user) => user.duty === "Supervisor"
    );
    if (supervisorExists) {
      NotificationBox.show("A user with 'Supervisor' duty already exists. Only one supervisor is allowed.", "error");
      return;
    }
  }

  // 🔒 Password hashing
  const hashedPassword = await bcrypt.hash(password, 10);

  // 📁 Add to database
  await Users.add({
    username,
    password: hashedPassword,
    lastName,
    firstName,
    middleInitial,
    position,
    duty,
    type: natureOfAppointment,
    status,
    role: "user",
  });
}
```

```
NotificationBox.show("User successfully added.");
await loadUsers();

// ✅ Reset form and close modal
addUserForm.reset();
addUserForm.classList.remove("was-validated");
modal.hide();
} catch (err) {
  console.error("Error creating user:", err.message);
  NotificationBox.show("Error creating user: " + err.message, "error");
} finally {
  Spinner.hide();
}
});

function renderUsersTable(page = 1, searchQuery = "") {
  const usersTableBody = document.getElementById("usersTableBody");

  // Helper to normalize text (remove diacritics, lower case)
  function normalizeText(str) {
    return str.normalize('NFD').replace(/[\u0300-\u036f]/g, "").toLowerCase();
  }

  // Highlight match in original text
  function highlightMatch(text, query) {
    if (!query) return text;
    const normalizedText = normalizeText(text);
    const normalizedQuery = normalizeText(query);
    const index = normalizedText.indexOf(normalizedQuery);
    if (index === -1) return text;

    const originalIndex = [...text].findIndex(_ =>
      normalizeText(text.slice(_)).startsWith(normalizedQuery)
    );

    return (
      text.slice(0, originalIndex) +
      "<mark>" + text.slice(originalIndex, originalIndex + query.length) + "</mark>" +
      text.slice(originalIndex + query.length)
    );
  }
}

let filteredUsers = allUsers.filter(user => user.type && user.type.trim() !== "");

if (searchQuery) {
  const normQuery = normalizeText(searchQuery);
  filteredUsers = filteredUsers.filter(user =>
    normalizeText(user.username || "").includes(normQuery) ||
    normalizeText(user.firstName || "").includes(normQuery) ||
    normalizeText(user.lastName || "").includes(normQuery)
  );
}
```

```
const start = (page - 1) * usersPerPage;
const end = start + usersPerPage;
const paginatedUsers = filteredUsers.slice(start, end);

usersTableBody.innerHTML = "";

paginatedUsers.forEach(user => {
  const username = highlightMatch(user.username || "", searchQuery);
  const lastName = highlightMatch(user.lastName || "", searchQuery);
  const firstName = highlightMatch(user.firstName || "", searchQuery);
  const position = user.position || "";
  const type = user.type || "";

  const row = document.createElement("tr");
  row.innerHTML = `
    <td>${username}</td>
    <td>${lastName}</td>
    <td>${firstName}</td>
    <td>${position}</td>
    <td>${type}</td>
    <td>
      <button class="btn btn-3d btn-sm btn-outline-primary edit-btn" data-id="${user.id}">
        <i class="bi bi-pencil-square me-1"></i> Edit
      </button>
    </td>
  `;
  usersTableBody.appendChild(row);
});

initializeEditFunctionality();
renderPaginationControls(Math.ceil(filteredUsers.length / usersPerPage), page, searchQuery);
}

function handleRefreshButton({
  buttonId,
  refreshFn,
  renderFn,
  cooldownKey,
  cooldownSeconds = 60
}) {
  if (!buttonId || !refreshFn || !renderFn || !cooldownKey) {
    console.error(`handleRefreshButton: Missing required arguments.`);
    return;
  }

  const refreshBtn = document.getElementById(buttonId);
  if (!refreshBtn) {
    console.error(`handleRefreshButton: Button with ID "${buttonId}" not found.`);
    return;
  }

  let originalText = refreshBtn.textContent;
```

```
let cooldownTimer = null;

function startCooldown() {
  const startTime = Date.now();
  const endTime = startTime + cooldownSeconds * 1000;
  localStorage.setItem(cooldownKey, startTime.toString());

  refreshBtn.disabled = true;

  cooldownTimer = setInterval(() => {
    const now = Date.now();
    const secondsLeft = Math.ceil((endTime - now) / 1000);

    if (secondsLeft <= 0) {
      clearInterval(cooldownTimer);
      refreshBtn.disabled = false;
      refreshBtn.textContent = originalText;
    } else {
      refreshBtn.textContent = `Wait (${secondsLeft}s)`;
    }
  }, 1000);
}

// On load, resume cooldown if within cooldown time
const lastRefresh = parseInt(localStorage.getItem(cooldownKey)) || 0;
const now = Date.now();
const elapsed = now - lastRefresh;
const cooldownMs = cooldownSeconds * 1000;

if (elapsed < cooldownMs) {
  const remaining = Math.ceil((cooldownMs - elapsed) / 1000);
  startCooldown(); // Resume countdown
}

refreshBtn.addEventListener("click", async () => {
  const now = Date.now();
  const lastRefresh = parseInt(localStorage.getItem(cooldownKey)) || 0;
  const cooldownMs = cooldownSeconds * 1000;

  if (now - lastRefresh < cooldownMs) {
    // Already handled by interval timer
    return;
  }

  try {
    await refreshFn();
    await Users.refreshCache();
    renderUsersTable();
    renderFn();
    NotificationBox.show("Refreshed successfully.");
    startCooldown();
  } catch (err) {
    console.error("Refresh error:", err);
  }
})
```

```

        NotificationBox.show("Failed to refresh data.", "error");
    }
});
}

function initializeEditFunctionality() {
    // Handle Edit Button Click
    document.querySelectorAll(".edit-btn").forEach(button => {
        button.addEventListener("click", async () => {
            const userId = button.getAttribute("data-id");
            const users = await Users.fetchAllDesc();
            const user = users.find(u => u.id === userId);
            if (!user) return;

            // Populate form
            document.getElementById("editUserId").value = user.id;
            document.getElementById("editUsername").value = user.username || "";
            document.getElementById("editLastName").value = user.lastName || "";
            document.getElementById("editFirstName").value = user.firstName || "";
            document.getElementById("editMiddleInitial").value = user.middleInitial || "";
            document.getElementById("editPosition").value = user.position || "";
            document.getElementById("editDuty").value = user.duty || "";
            document.getElementById("editNatureOfAppointment").value = user.type || "";

            // Status Radio
            if (user.status === "active") {
                document.getElementById("editStatusActive").checked = true;
            } else {
                document.getElementById("editStatusInactive").checked = true;
            }

            // Clear password fields
            document.getElementById("editPassword").value = "";
            document.getElementById("editConfirmPassword").value = "";

            // Show modal
            const modal = new bootstrap.Modal(document.getElementById("editUserModal"));
            modal.show();
        });
    });

    // Handle Form Submit
    const form = document.getElementById("editUserForm");
    form.onsubmit = async (e) => {
        e.preventDefault();
        Spinner.show();

        try {
            const id = document.getElementById("editUserId").value.trim();
            const username = document.getElementById("editUsername").value.trim();
            const lastName = document.getElementById("editLastName").value.trim();
            const firstName = document.getElementById("editFirstName").value.trim();
            const middleInitial = document.getElementById("editMiddleInitial").value.trim();
        
```

```
const position = document.getElementById("editPosition").value.trim();
const duty = document.getElementById("editDuty").value;
const type = document.getElementById("editNatureOfAppointment").value;
const status = document.querySelector('input[name="editStatus"]:checked')?.value;

const password = document.getElementById("editPassword").value.trim();
const confirmPassword = document.getElementById("editConfirmPassword").value.trim();

// ✅ Basic Validation
if (!username || !lastName || !firstName || !position || !duty || !type || !status) {
  NotificationBox.show("Please fill in all required fields.", "warning");
  return;
}

const users = await Users.fetchAllDesc();

// ✅ Username duplicate check
const usernameExists = users.some(
  (user) =>
    (user.username || "").toLowerCase() === username.toLowerCase() &&
    user.id !== id
);

if (usernameExists) {
  NotificationBox.show("Username already exists. Please choose another one.", "error");
  return;
}

// ✅ Password confirmation
if ((password || confirmPassword) && password !== confirmPassword) {
  NotificationBox.show("Passwords do not match.", "error");
  return;
}

if (duty === "Supervisor") {
  const existingSupervisor = users.find(
    (user) => user.duty === "Supervisor" && user.id !== id
  );
  if (existingSupervisor) {
    NotificationBox.show("A supervisor already exists. Only one supervisor is allowed.", "error");
    return;
  }
}

// ✅ Prepare update data
let updateData = {
  username,
  lastName,
  firstName,
  middleInitial,
  position,
  duty,
}
```

```
        type,
        status
    };

    if (password) {
        const hashedPassword = await bcrypt.hash(password, 10);
        updateData.password = hashedPassword;
    }

    await Users.update(id, updateData);

    NotificationBox.show("User updated successfully.");

    const modalEl = document.getElementById("editUserModal");
    const modalInstance = bootstrap.Modal.getOrCreateInstance(modalEl);
    modalInstance.hide();

    setTimeout(() => {
        document.body.classList.remove("modal-open");
        document.querySelectorAll(".modal-backdrop").forEach((el) => el.remove());
    }, 300);

    await loadUsers();
} catch (error) {
    console.error("Error submitting edit form:", error);
    NotificationBox.show("An error occurred while updating the user.", "error");
} finally {
    Spinner.hide();
}
};

}


```

```
function renderPaginationControls(totalPages, currentPage, searchQuery = "") {
    const pagination = document.getElementById("paginationControlsUsers");
    pagination.innerHTML = "";

    if (totalPages <= 1) return; // No need to show pagination for 1 page

    const maxVisiblePages = 5;
    let startPage = Math.max(1, currentPage - Math.floor(maxVisiblePages / 2));
    let endPage = startPage + maxVisiblePages - 1;

    if (endPage > totalPages) {
        endPage = totalPages;
        startPage = Math.max(1, endPage - maxVisiblePages + 1);
    }

    const ul = document.createElement("ul");
    ul.className = "pagination justify-content-center";

    // Previous Button
    const prevItem = document.createElement("li");
    prevItem.className = `page-item ${currentPage === 1 ? "disabled" : ""}`;
    prevItem
```

```
prevItem.innerHTML = `<a class="page-link" href="#" aria-label="Previous"><span aria-hidden="true">&laquo;</span></a>`;  
prevItem.addEventListener("click", (e) => {  
  e.preventDefault();  
  if (currentPage > 1) {  
    renderUsersTable(currentPage - 1, searchQuery);  
  }  
});  
ul.appendChild(prevItem);  
  
// Page Number Buttons  
for (let i = startPage; i <= endPage; i++) {  
  const pagelItem = document.createElement("li");  
  pagelItem.className = `page-item ${i === currentPage ? "active" : ""}`;  
  pagelItem.innerHTML = `<a class="page-link" href="#">${i}</a>`;  
  pagelItem.addEventListener("click", (e) => {  
    e.preventDefault();  
    renderUsersTable(i, searchQuery);  
  });  
  ul.appendChild(pagelItem);  
}  
  
// Next Button  
const nextItem = document.createElement("li");  
nextItem.className = `${currentPage === totalPages ? "disabled" : ""}`;  
nextItem.innerHTML = `<a class="page-link" href="#" aria-label="Next"><span aria-hidden="true">&raquo;</span></a>`;  
nextItem.addEventListener("click", (e) => {  
  e.preventDefault();  
  if (currentPage < totalPages) {  
    renderUsersTable(currentPage + 1, searchQuery);  
  }  
});  
ul.appendChild(nextItem);  
  
pagination.appendChild(ul);  
}  
  
function handleSearchBar() {  
  document.getElementById("searchInput").addEventListener("input", (e) => {  
    const query = e.target.value.trim();  
    currentPage = 1;  
    renderUsersTable(currentPage, query);  
  });  
}
```

XV.C.3.k. Utility Scripts

coordinates.js

```
export const CoordinateUtils = {  
  // Convert DMS to Decimal  
  dmsToDecimal(degrees, minutes, seconds, direction) {  
    let decimal = Math.abs(degrees) + (minutes / 60) + (seconds / 3600);  
    if (['S', 'W'].includes(direction.toUpperCase())) {  
      decimal *= -1;  
    }  
    return decimal;  
  },  
  
  // Convert Decimal to DMS  
  decimalToDMS(decimal) {  
    const absolute = Math.abs(decimal);  
    const degrees = Math.floor(absolute);  
    const minutesFloat = (absolute - degrees) * 60;  
    const minutes = Math.floor(minutesFloat);  
    const seconds = (minutesFloat - minutes) * 60;  
  
    const direction = decimal >= 0 ? 'N/E' : 'S/W';  
  
    return {  
      degrees,  
      minutes,  
      seconds: parseFloat(seconds.toFixed(2)),  
      direction  
    };  
  },  
  
  // Format DMS to string  
  formatDMS({degrees, minutes, seconds, direction}) {  
    return `${degrees}° ${minutes}' ${seconds}" ${direction}`;  
  }  
};
```

dateFormat.js

```
export function updateReportTimestamp() {  
  const now = new Date();  
  const formatted = now.toLocaleString('en-US', {  
    month: 'long', // MMMMM  
    day: '2-digit', // dd  
    year: 'numeric', // YYYY  
    hour: '2-digit',  
    minute: '2-digit',  
    second: '2-digit',  
    hour12: true // AM/PM format  
  });
```

```
const timestampElement = document.getElementById('reportTimestamp');
if (timestampElement) {
  timestampElement.textContent = `Report generated on ${formatted}`;
}
}
```

normalize.js

```
export function normalizeBarangay(brgy) {
  if (!brgy || brgy.trim() === "" || ['0', 'n/a', 'bgy', 'bgy.'].includes(brgy.toLowerCase())) {
    return 'Unknown';
  }
  return brgy
    .toLowerCase()
    .replace(/^(brgy\.)?|barangay|bgy\.(?|bgy)\s*/i, "")
    .replace(/\s+/g, ' ')
    .trim()
    .replace(/\b\w/g, c => c.toUpperCase());
}

export function normalizeCity(city) {
  if (!city || city.trim() === "" || city === '0' || city.toLowerCase() === 'n/a') {
    return 'Unknown';
  }
  return city.trim().replace(/\b\w/g, c => c.toUpperCase());
}
```

XV.C.3.I. Mobile Inventory Form Module

signature-pad-module.js

```
export function initSignaturePad({
  canvasId,
  clearBtnId,
  hiddenInputId,
  formId,
  modalIds = []
} = {}) {
  if (typeof SignaturePad === 'undefined') {
    console.error("SignaturePad library not loaded. Please include the CDN first.");
    return;
  }

  const canvas = document.getElementById(canvasId);
  const clearBtn = document.getElementById(clearBtnId);
  const hiddenInput = document.getElementById(hiddenInputId);
  const form = document.getElementById(formId);

  if (!canvas || !clearBtn || !hiddenInput || !form) {
    console.error(`Signature Pad: Missing elements for canvasId: ${canvasId}`);
    return;
  }
```

```

const signaturePad = new SignaturePad(canvas, {
  backgroundColor: '#ffffff',
  penColor: 'black'
});

function resizeCanvas() {
  const ratio = Math.max(window.devicePixelRatio || 1, 1);
  canvas.width = canvas.offsetWidth * ratio;
  canvas.height = canvas.offsetHeight * ratio;
  canvas.getContext("2d").scale(ratio, ratio);
  signaturePad.clear();
}

modalIds.forEach(modalId => {
  const modal = document.getElementById(modalId);
  if (!modal) {
    console.warn(`Modal with ID "${modalId}" not found.`);
    return;
  }
  modal.addEventListener('shown.bs.modal', () => {
    resizeCanvas();
  });
});

window.addEventListener("resize", () => {
  const anyModalOpen = modalIds.some(id => {
    const modal = document.getElementById(id);
    return modal && modal.classList.contains('show');
  });

  if (anyModalOpen) {
    const data = signaturePad.toData();
    resizeCanvas();
    signaturePad.fromData(data);
  }
});

clearBtn.addEventListener('click', () => signaturePad.clear());

form.addEventListener('submit', () => {
  hiddenInput.value = signaturePad.isEmpty() ? "" : signaturePad.toDataURL();
});

return signaturePad;
}

```

water-inventory-init.js

```

import { SessionGuard } from "../auth/auth.js";
import { initializePage } from "./water-inventory-ui.js"

document.addEventListener('DOMContentLoaded', ()=>{

```

```
SessionGuard.ensureLoggedIn();
initializePage();
})
```

water-inventory-ui.js

```
import { Sidebar } from "../components/sidebar.js";
import { Spinner } from "../components/spinner.js";
import { WUSData } from "../data/cache/wrus-data.js";
import { METRO_MANILA_CITIES } from "../data/constants/metroManilaCities.js";
import { months } from "../data/constants/months.js";
import { waterSources } from "../data/constants/waterSources.js";
import { sourceStatus } from "../data/constants/sourceStatus.js";
import { purpose } from "../data/constants/purpose.js";
import { initSignaturePad } from "./signature-pad-module.js";
import { SignatureDB } from "../data/indexedDB/signature-data.js";
import { NotificationBox, Confirmation } from "../components/notification.js";
import { uploadSignatureToSupabase } from "../upload/uploadImg.js";

export function initializePage() {
  Sidebar.render();
  Spinner.render();
  initDynamicPlusButtons();
  populateMonthSelect('monthConducted');
  populateMonthSelect('monthConductedAddModal');
  populateMonthSelect('monthConductedEditModal');
  initForm();
  waterSourceSelectModal('modalSourceWaterSelect');
  waterSourceSelectModal('editSourceWaterSelect');
  populateWaterSourceStatus('modalSourceStatus');
  populateWaterSourceStatus('editSourceStatus');
  populatePurposeSelect('modalPurposeSelect');
  populatePurposeSelect('editPurposeSelect');
  initSignaturePad({
    canvasId: "signatureCanvas",
    clearBtnId: "clearSignature",
    hiddenInputId: "modalSignature",
    formId: "modalForm",
    modalIds: ["addModal"]
  });
  finalizeButtonHandler();
  clearAll();
  initEditWaterInventoryFormListener();
  console.log(localStorage.getItem('waterInventory'));
}

let editingIndex = null;
let itemCount = 0;
const signatureDB = new SignatureDB();

function initForm() {
  const citySelect = document.getElementById("citySelect");
```

```
const yearInput = document.getElementById("yearConducted");
const monthSelect = document.getElementById("monthConducted");

// Populate cities
if (citySelect) {
  METRO_MANILA_CITIES.forEach(city => {
    const option = document.createElement("option");
    option.value = city;
    option.textContent = city;
    citySelect.appendChild(option);
  });
}

// Set current year
if (yearInput) {
  const currentYear = new Date().getFullYear();
  yearInput.value = currentYear;
}

// Populate months
if (monthSelect) {
  const currentMonthIndex = new Date().getMonth();
  months.forEach((month, index) => {
    const option = document.createElement("option");
    option.value = month;
    option.textContent = month;
    if (index === currentMonthIndex) {
      option.selected = true;
    }
    monthSelect.appendChild(option);
  });
}

function initDynamicPlusButtons() {
  const itemContainer = document.getElementById('itemContainer');
  const modalForm = document.getElementById('modalForm');
  const modalElement = document.getElementById('addModal');
  const modal = bootstrap.Modal.getOrCreateInstance(modalElement);

  if (!localStorage.getItem('waterInventory')) {
    localStorage.setItem('waterInventory', JSON.stringify([]));
  }

  let savedData = JSON.parse(localStorage.getItem('waterInventory'));
  itemCount = savedData.length;

  itemContainer.innerHTML = "";

  savedData.forEach((item, index) => {
    const fileDiv = document.createElement('div');
    fileDiv.classList.add('entry-wrapper', 'text-center');
```

```
fileDiv.innerHTML = `

<a href="#" class="btn-3d file-link mb-2" data-index="${index}"
    data-bs-toggle="modal" data-bs-target="#editModal">
    <i class="bi bi-file-earmark"></i>
    <span class="file-number">${index + 1}</span>
</a>
<button class="btn btn-3d btn-danger btn-lg mt-2 delete-entry-btn" data-id="${item.id}">
    <i class="bi bi-dash-circle"></i>
</button>
`;

itemContainer.appendChild(fileDiv);
});

const plusDiv = document.createElement('div');
plusDiv.innerHTML = `

<button id="addWaterBtn" class="btn btn-primary big-plus-btn mt-3"
    data-bs-toggle="modal" data-bs-target="#addModal">
    <i class="bi bi-file-earmark-plus"></i>
</button>
`;
itemContainer.appendChild(plusDiv);

document.getElementById('addWaterBtn').addEventListener('click', () => {
    editingIndex = null;
    autoPopulateAddModal();
});

const fileLinks = itemContainer.querySelectorAll('.file-link');
fileLinks.forEach(link => {
    link.addEventListener('click', function () {
        const index = parseInt(this.dataset.index);
        editingIndex = index;
        populateEditModal(index);
    });
});

attachDeleteButtonListeners();

handleModalFormSubmit(modalForm, modal);
}

function handleModalFormSubmit(modalForm, modal) {
    if (modalForm.dataset.listenerAttached) return;

    modalForm.addEventListener('submit', async function (e) {
        e.preventDefault();

        const formData = {};
        let signatureDataURL = null;

        for (let element of modalForm.elements) {
            if (element.id && element.type !== 'submit' && element.type !== 'button') {
```

```
        formData[element.id] = element.type === 'checkbox' ? element.checked : element.value;
    }
}

const canvas = document.getElementById('signatureCanvas');
if (canvas) {
    signatureDataURL = canvas.toDataURL();
}

let savedData = JSON.parse(localStorage.getItem('waterInventory')) || [];
let recordId = null;

if (
    typeof editingIndex === 'number' &&
    editingIndex >= 0 &&
    editingIndex < savedData.length
) {
    formData.id = savedData[editingIndex].id || generateUniqueId();
    savedData[editingIndex] = formData;
    recordId = formData.id;
    editingIndex = null;
} else {
    recordId = generateUniqueId();
    formData.id = recordId;
    savedData.push(formData);
}

localStorage.setItem('waterInventory', JSON.stringify(savedData));

if (signatureDataURL) {
    try {
        await signatureDB.saveSignature(signatureDataURL, recordId);
    } catch (err) {
        console.error("Error saving signature:", err);
    }
}

initDynamicPlusButtons();
modalForm.reset();
modal.hide();
});

modalForm.dataset.listenerAttached = "true";
}

function generateUniqueId() {
    const now = new Date();
    return 'id-' + now.toISOString().replace(/[-:.TZ]/g, '') + '-' + Math.random().toString(36).substr(2, 5);
}

function attachDeleteButtonListeners() {
    const deleteBtns = document.querySelectorAll('.delete-entry-btn');
```

```
deleteBtns.forEach(btn => {
  btn.addEventListener('click', function () {
    const id = this.dataset.id; // use unique id instead of index

    Confirmation.show(
      'Are you sure you want to delete this entry?',
      async (confirmed) => {
        if (!confirmed) return;

        Spinner.show();

        try {
          let savedData = JSON.parse(localStorage.getItem('waterInventory')) || [];
          const newData = savedData.filter(entry => entry.id !== id);
          localStorage.setItem('waterInventory', JSON.stringify(newData));

          await signatureDB.deleteSignature(id);

          initDynamicPlusButtons();
        } catch (err) {
          console.error("Error deleting entry:", err);
          NotificationBox.show("Failed to delete entry. Please try again.", "error");
        } finally {
          Spinner.hide();
        }
      }
    );
  });
});

function populateMonthSelect(selectID) {
  const selectEl = document.getElementById(selectID);

  selectEl.innerHTML = "";

  // Optional default
  const defaultOption = document.createElement('option');
  defaultOption.value = "";
  defaultOption.textContent = 'Select month...';
  defaultOption.disabled = true;
  defaultOption.selected = true;
  selectEl.appendChild(defaultOption);

  months.forEach(purpose => {
    const option = document.createElement('option');
    option.value = purpose;
    option.textContent = purpose.replace(' Use', ""); // Optional: remove ' Use' from label
    selectEl.appendChild(option);
  });
}

function populateWaterSourceStatus(selectID) {
```

```

const selectEl = document.getElementById(selectID);

selectEl.innerHTML = "";

// Optional default
const defaultOption = document.createElement('option');
defaultOption.value = "";
defaultOption.textContent = 'Select status...';
defaultOption.disabled = true;
defaultOption.selected = true;
selectEl.appendChild(defaultOption);

sourceStatus.forEach(status => {
  const option = document.createElement('option');
  option.value = status;
  option.textContent = status;
  selectEl.appendChild(option);
});

function autoPopulateAddModal() {
  const selectedCity = document.getElementById('citySelect')?.value || "";
  const barangay = document.getElementById('barangayInput')?.value || "";
  const yearConducted = document.getElementById('yearConducted')?.value || "";
  const monthConducted = document.getElementById('monthConducted')?.value || "";

  const modalYear = document.getElementById('modalYearConducted');
  const modalCity = document.getElementById('modalCity');
  const modalBarangay = document.getElementById('modalBarangay');
  const modalMonth = document.getElementById('monthConductedAddModal');

  if (modalYear) modalYear.value = yearConducted;
  if (modalCity) modalCity.value = selectedCity;
  if (modalBarangay) modalBarangay.value = barangay;
  if (modalMonth) modalMonth.value = monthConducted;
}

async function populateEditModal(index) {
  const savedData = JSON.parse(localStorage.getItem('waterInventory'));
  const record = savedData[index];
  const id = record.id;

  document.getElementById('editIndex').value = index;

  document.getElementById('editYearConducted').value = record.modalYearConducted || "";
  document.getElementById('monthConductedEditModal').value = record.monthConductedAddModal || "";
  document.getElementById('editOwner').value = record.modalOwner || "";
  document.getElementById('editLocation').value = record.modalLocation || "";
  document.getElementById('editCity').value = record.modalCity || "";
  document.getElementById('editBarangay').value = record.modalBarangay || "";
  document.getElementById('editSourceWaterSelect').value = record.modalSourceWaterSelect || "";
  document.getElementById('editSourceStatus').value = record.modalSourceStatus || "";
  document.getElementById('editLatitude').value = record.modalLatitude || "";
}

```

```
document.getElementById('editLongitude').value = record.modalLongitude || '';
document.getElementById('editPurposeSelect').value = record.modalPurposeSelect || '';
document.getElementById('editRemarks').value = record.modalRemarks || '';
document.getElementById('editRepresentative').value = record.modalRepresentative || '';
document.getElementById('editDesignation').value = record.modalDesignation || '';
document.getElementById('editPhone').value = record.modalPhone || '';

// Load and display the signature image
const imageSignature = document.getElementById('image-signature');
if (imageSignature && id) {
  try {
    const signatureDataURL = await signatureDB.getSignature(id)
    if (signatureDataURL) {
      imageSignature.src = signatureDataURL;
      imageSignature.style.display = "block"; // Ensure it's visible
    } else {
      imageSignature.src = "";
      imageSignature.style.display = "none"; // Hide if not found
    }
  } catch (err) {
    console.error("Error loading signature image:", err);
    imageSignature.src = "";
    imageSignature.style.display = "none";
  }
}
}

function waterSourceSelectModal(selectID){
  const selectEl = document.getElementById(selectID);

  selectEl.innerHTML = "";

  // Optional default
  const defaultOption = document.createElement('option');
  defaultOption.value = "";
  defaultOption.textContent = 'Choose source...';
  defaultOption.disabled = true;
  defaultOption.selected = true;
  selectEl.appendChild(defaultOption);

  // Dynamically add the options
  waterSources.forEach(source => {
    const option = document.createElement('option');
    option.value = source;
    option.textContent = source;
    selectEl.appendChild(option);
  });
}

function populatePurposeSelect(selectID) {
  const selectEl = document.getElementById(selectID);

  selectEl.innerHTML = "
```

```
// Optional default
const defaultOption = document.createElement('option');
defaultOption.value = '';
defaultOption.textContent = 'Select purpose...';
defaultOption.disabled = true;
defaultOption.selected = true;
selectEl.appendChild(defaultOption);

purpose.forEach(purpose => {
  const option = document.createElement('option');
  option.value = purpose;
  option.textContent = purpose.replace(' Use', ""); // Optional: remove ' Use' from label
  selectEl.appendChild(option);
});

function initEditWaterInventoryFormListener() {
  const editForm = document.getElementById('editModalForm');
  if (!editForm) return;

  if (editForm.dataset.listenerAttached) return;

  editForm.addEventListener('submit', function (e) {
    e.preventDefault();

    let savedData = JSON.parse(localStorage.getItem('waterInventory')) || [];

    const index = parseInt(document.getElementById('editIndex').value, 10);

    const existingId = savedData[index].id;

    const updatedRecord = {
      id: existingId,
      modalYearConducted: document.getElementById('editYearConducted').value,
      monthConductedAddModal: document.getElementById('monthConductedEditModal').value,
      modalOwner: document.getElementById('editOwner').value,
      modalLocation: document.getElementById('editLocation').value,
      modalCity: document.getElementById('editCity').value,
      modalBarangay: document.getElementById('editBarangay').value,
      modalSourceWaterSelect: document.getElementById('editSourceWaterSelect').value,
      modalLatitude: document.getElementById('editLatitude').value,
      modalLongitude: document.getElementById('editLongitude').value,
      modalPurposeSelect: document.getElementById('editPurposeSelect').value,
      modalRemarks: document.getElementById('editRemarks').value,
      modalRepresentative: document.getElementById('editRepresentative').value,
    };

    savedData[index] = updatedRecord;

    localStorage.setItem('waterInventory', JSON.stringify(savedData));

    initDynamicPlusButtons();
  });
}
```

```
const editModal = bootstrap.Modal.getInstance(document.getElementById('editModal'));
editModal.hide();
});

editForm.dataset.listenerAttached = "true";
}

function clearAll() {
const btn = document.getElementById('clearBtn');

btn.addEventListener('click', () => {
Confirmation.show(
"Are you sure you want to CLEAR all water inventory data from the cache? This action cannot be undone.",
async (confirmed) => {
if (!confirmed) return;

Spinner.show();

try {
// Clear local storage
localStorage.removeItem('waterInventory');

// Clear IndexedDB
await signatureDB.clearSignatures();

initDynamicPlusButtons();

NotificationBox.show("All water inventory and signature data have been cleared.");
} catch (err) {
console.error("Error clearing data:", err);
NotificationBox.show("Failed to clear data. Please try again.", "error");
} finally {
Spinner.hide();
}
}
);
});
}

function finalizeButtonHandler() {
const btn = document.getElementById('finalizeButton');

btn.addEventListener('click', () => {
Confirmation.show(
"Are you sure you want to send ALL water inventory data to the database?",
async (confirmed) => {
if (!confirmed) return;

const localData = JSON.parse(localStorage.getItem('waterInventory') || '[]');
if (localData.length === 0) {
NotificationBox.show("🚩 No water inventory data found in localStorage.", "error");
return;
}
```

```
}

Spinner.show();
btn.disabled = true;

try {
  for (const item of localData) {
    let signUrl = "";

    try {
      const signatureDataURL = await signatureDB.getSignature(item.id);
      if (signatureDataURL) {
        const fileName = `signature-${item.modalOwner || "unknown"}-${Date.now()}.png`;
        signUrl = await uploadSignatureToSupabase(signatureDataURL, fileName);
      }
    } catch (uploadErr) {
      console.error("Failed to upload signature:", uploadErr);
    }

    const statusValue = item.modalSourceStatus || "";
    const isWaterSource =
      statusValue === "OPERATIONAL" || statusValue === "ON-GOING CONSTRUCTION";

    const payload = {
      year_conducted: item.modalYearConducted || "",
      month_conducted: item.monthConductedAddModal || "",
      owner: item.modalOwner || "",
      street: item.modalLocation || "",
      city: item.modalCity || "",
      barangay: item.modalBarangay || "",
      type: item.modalSourceWaterSelect || "",
      status: statusValue,
      latitude: item.modalLatitude || "",
      longitude: item.modalLongitude || "",
      purpose: item.modalPurposeSelect || "",
      remarks: item.modalRemarks,
      representative: item.modalRepresentative || "",
      designation: item.modalDesignation || "",
      phone: item.modalPhone || "",
      signUrl,
      isWaterSource // ✅ Added field
    };

    await WUSData.add(payload);
  }

  localStorage.removeItem('waterInventory');

  // ✅ Clear all signatures from IndexedDB
  await signatureDB.clearSignatures();

  NotificationBox.show("All water inventory data has been successfully sent!");
  initDynamicPlusButtons();
}
```

```

        } catch (err) {
          console.error("✖ Error while sending data:", err);
          NotificationBox.show("An error occurred while sending the data. Please try again.", "error");
        } finally {
          Spinner.hide();
          btn.disabled = false;
        }
      }
    );
  });
}

```

XV.C.3.m. Water Users and Sources Module

wrus-init.js

```

import { SessionGuard } from "../auth/auth.js";
import { initializePage } from "./wrus-ui.js";

document.addEventListener('DOMContentLoaded', ()=>{
  SessionGuard.ensureLoggedIn();
  initializePage();
})

```

wrus-ui.js

```

import { WUSData } from '../data/cache/wrus-data.js';
import { Permit } from '../data/cache/permit-data.js';
import { Sidebar } from '../components/sidebar.js';
import { Spinner } from '../components/spinner.js';
import { NotificationBox } from '../components/notification.js';
import { METRO_MANILA_CITIES } from '../data/constants/metroManilaCities.js';
import { months } from '../data/constants/months.js';
import { sourceStatus } from '../data/constants/sourceStatus.js';
import { GeotaggedFileService } from '../upload/uploadImg.js';
import { PortalBubble } from '../components/PortalBubble.js';
import { generateWaterUserPDF } from '../pdf/water-user-pdf.js';

export function initializePage(){
  Sidebar.render();
  Spinner.render();
  initializeSearchBar();
  loadWaterUser();
  setupWaterSourceFilterToggle();
  setupAddModalCities();
  setupEditModalCities();
  setupModalMonth('monthConducted');
  setupModalMonth('editMonthConducted');
  populateWaterSourceStatus('status')
  populateWaterSourceStatus('editStatus')
  setupPermitAutoComplete('permitNoInput', 'permitSuggestions', 'nameOfWaterUser', 'city', 'wusGeotagUrl');
}

```

```

setupPermitAutoComplete('permitNoInputEdit', 'permitSuggestionsEdit', 'editOwner', 'editCity', 'editWusGeotagUrl');

handleAddForm();
handleEditForm();
handleRefreshButton({
  buttonId: "refreshBtn",
  refreshFn: WUSData.refreshCache,
  renderFn: renderWaterUsers,
  cooldownKey: "lastWUSRefresh"
});
initializeExportButton();
showWaterUserPDF();
}

let currentPage = 1;
const rowsPerPage = 10;
let allUsers = [];
let searchTerm = "";
let currentSearchTerm = "";

function setupWaterSourceFilterToggle() {
  const section = document.getElementById('waterSourceFilterSection');
  const button = document.getElementById('toggleWaterSourceFilterBtn');
  const waterSourceOnly = document.getElementById('waterSourceOnlyFilter');
  const nonWaterSourceOnly = document.getElementById('nonWaterSourceOnlyFilter');

  // Toggle visibility of filter section
  button.addEventListener('click', () => {
    const isHidden = section.classList.contains('d-none');
    section.classList.toggle('d-none');
    button.textContent = isHidden ? 'Hide Water Source Filter' : 'Show Water Source Filter';
  });

  // Disable the other checkbox when one is checked and re-render table
  waterSourceOnly.addEventListener('change', () => {
    nonWaterSourceOnly.disabled = waterSourceOnly.checked;
    renderWaterUsers(); // refresh table with filter
  });

  nonWaterSourceOnly.addEventListener('change', () => {
    waterSourceOnly.disabled = nonWaterSourceOnly.checked;
    renderWaterUsers(); // refresh table with filter
  });
}

function setupAddModalCities() {
  const citySelect = document.getElementById("city");
  if (!citySelect) return;

  citySelect.innerHTML = '<option selected disabled value="">Select City</option>';

  METRO_MANILA_CITIES.forEach(city => {
    const option = document.createElement("option");
    option.value = city;
  });
}

```

```

option.textContent = city;
citySelect.appendChild(option);
});
}

function setupEditModalCities() {
const citySelect = document.getElementById("editCity");
if (!citySelect) return;

citySelect.innerHTML = '<option selected disabled value="">Select City</option>';

METRO_MANILA_CITIES.forEach(city => {
const option = document.createElement("option");
option.value = city;
option.textContent = city;
citySelect.appendChild(option);
});
}

function setupModalMonth(selectID){
const monthSelect = document.getElementById(selectID);

monthSelect.innerHTML = "";
if (monthSelect) {
monthSelect.innerHTML = `<option value="">Select Month</option>` +
months.map(month => `<option value="${month}">${month}</option>`).join("");
}
}

function populateWaterSourceStatus(selectID) {
const selectEl = document.getElementById(selectID);

selectEl.innerHTML = "";

// Optional default
const defaultOption = document.createElement('option');
defaultOption.value = "";
defaultOption.textContent = 'Select status...';
defaultOption.disabled = true;
defaultOption.selected = true;
selectEl.appendChild(defaultOption);

sourceStatus.forEach(status => {
const option = document.createElement('option');
option.value = status;
option.textContent = status;
selectEl.appendChild(option);
});
}

async function setupPermitAutoComplete(inputId, suggestionsId, ownerInputId = null, citySelectId = null, geotagUrlId = null) {
const permitInput = document.getElementById(inputId);
const suggestionsBox = document.getElementById(suggestionsId);
}

```

```
const ownerInput = ownerInputId ? document.getElementById(ownerInputId) : null;
const citySelect = citySelectId ? document.getElementById(citySelectId) : null;
const geotagUrlInput = geotagUrlId ? document.getElementById(geotagUrlId) : null;

if (!permitInput || !suggestionsBox) {
  console.warn(`Elements with IDs '${inputId}' or '${suggestionsId}' not found.`);
  return;
}

let allPermits = [];

try {
  allPermits = await Permit.getAll();
} catch (err) {
  console.error('Error loading permits:', err.message);
  return;
}

permitInput.addEventListener('input', () => {
  const query = permitInput.value.trim().toLowerCase();
  suggestionsBox.innerHTML = '';

  if (!query) {
    suggestionsBox.style.display = 'none';
    return;
  }

  const matches = allPermits
    .filter(p => p.permitNo?.toLowerCase().includes(query))
    .slice(0, 5);

  if (matches.length === 0) {
    suggestionsBox.style.display = 'none';
    return;
  }

  matches.forEach(p => {
    const item = document.createElement('button');
    item.type = 'button';
    item.className = 'list-group-item list-group-item-action';
    item.textContent = `${p.permitNo} — ${p.permittee || 'Unnamed'}`;
    item.addEventListener('click', async () => {
      PortalBubble.trigger();
      permitInput.value = p.permitNo;

      if (ownerInput) {
        ownerInput.value = p.permittee || '';
      }

      if (citySelect && p.diversionPoint) {
        const diversionLower = p.diversionPoint.toLowerCase();
        let matched = false;
```

```
// Loop through all options and try to match the city name in diversionPoint
for (const option of citySelect.options) {
  const optionText = option.textContent.toLowerCase();
  if (diversionLower.includes(optionText)) {
    option.selected = true;
    matched = true;
    break;
  }
}

if (!matched) {
  citySelect.selectedIndex = 0; // fallback to default (first) option
}

// Set geotagged URL if the field exists
if (geotagUrlInput) {
  try {
    const geotaggedUrl = p.geotaggedUrl || '';
    geotagUrlInput.value = geotaggedUrl || '';
  } catch (err) {
    console.error('Error getting geotagged URL:', err.message);
    geotagUrlInput.value = '';
  }
}

suggestionsBox.innerHTML = '';
suggestionsBox.style.display = 'none';
});
suggestionsBox.appendChild(item);

suggestionsBox.style.display = 'block';
});

// Hide suggestions when clicking outside
document.addEventListener('click', (e) => {
  if (!permitInput.contains(e.target) && !suggestionsBox.contains(e.target)) {
    suggestionsBox.style.display = 'none';
  }
});
}

function handleAddForm() {
  const form = document.getElementById('addWusForm');

  form.addEventListener('submit', async function (e) {
    e.preventDefault();
    e.stopPropagation();

    form.classList.add('was-validated');
    if (!form.checkValidity()) {
      return;
    }
  });
}
```

```
}

const data = {
  permitNo: form.permitNoInput.value.trim(),
  owner: form.nameOfWaterUser.value.trim(),
  city: form.city.value.trim(),
  barangay: form.barangay.value.trim(),
  street: form.street.value.trim(),
  latitude: form.latitude.value.trim(),
  longitude: form.longitude.value.trim(),
  type: form.type.value.trim(),
  status: form.status.value.trim(),
  remarks: form.remarks.value.trim(),
  month_conducted: form.monthConducted.value.trim(),
  year_conducted: form.yearConducted.value.trim(),
  representative: form.representative.value.trim(),
  designation: form.designation.value.trim(),
  phone: form.phone.value.trim(),
  isWaterSource: form.isWaterSource.checked,
  geotaggedUrl: form.wusGeotagUrl.value.trim()
};

const permitData = {
  permitNo: form.permitNoInput.value.trim(),
  permittee: form.nameOfWaterUser.value.trim(),
  diversionPoint: form.city.value.trim(),
  latitude: form.latitude.value.trim(),
  longitude: form.longitude.value.trim(),
  waterSource: form.type.value.trim(),
  geotaggedUrl: form.wusGeotagUrl.value.trim(),
  visited: true
};

Spinner.show();
const submitButton = this.querySelector('button[type="submit"]');
submitButton.disabled = true;

try {
  await WUSDData.add(data);
  allUsers = [];
  const cachedPermits = await Permit.getAll();
  const matchingPermit = cachedPermits.find(p => p.permitNo === data.permitNo);

  if (matchingPermit) {
    await Permit.update(matchingPermit.id, { visited: true });
  } else {
    await Permit.add(permitData);
    console.log(permitData);
  }
}

form.reset();
form.classList.remove('was-validated');
```

```
const modal = bootstrap.Modal.getInstance(document.getElementById('addwusModal'));
modal.hide();

await loadWaterUser();
NotificationBox.show('Water User added successfully!');
} catch (err) {
  console.error('Error adding Water User:', err);
  NotificationBox.show('Could not add Water User. Please try again.', 'error');
} finally {
  Spinner.hide();
  submitButton.disabled = false;
}
});

}

async function loadWaterUser() {
  Spinner.show();
  try {
    allUsers = await WUSData.fetchAll();
    await renderWaterUsers();
  } finally {
    Spinner.hide();
  }
}

async function renderWaterUsers(term = '') {
  const tableBody = document.getElementById('waterUserTableBody');
  tableBody.innerHTML = '';

  if (allUsers.length === 0) {
    allUsers = await WUSData.fetchAll();
  }

  function normalizeText(str) {
    return str.normalize('NFD').replace(/[\u0300-\u036f]/g, '').toLowerCase();
  }

  function highlightMatch(text, searchTerm) {
    if (!searchTerm) return text;
    const normalizedText = normalizeText(text);
    const normalizedSearch = normalizeText(searchTerm);

    const index = normalizedText.indexOf(normalizedSearch);
    if (index === -1) return text;

    const originalIndex = [...text].findIndex((_, i) =>
      normalizeText(text.slice(i)).startsWith(normalizedSearch)
    );

    return (
      text.slice(0, originalIndex) +
      '<mark>' + text.slice(originalIndex, originalIndex + searchTerm.length) + '</mark>' +
      text.slice(originalIndex + searchTerm.length)
    );
  }
}
```

```

    );
}

const waterSourceOnly = document.getElementById('waterSourceOnlyFilter');
const nonWaterSourceOnly = document.getElementById('nonWaterSourceOnlyFilter');

let filteredUsers = allUsers;

if (waterSourceOnly.checked) {
  filteredUsers = filteredUsers.filter(u => u.isWaterSource === true);
} else if (nonWaterSourceOnly.checked) {
  filteredUsers = filteredUsers.filter(u => u.isWaterSource === false);
}

const searchTerm = term.trim();
const normalizedSearch = normalizeText(searchTerm);

if (normalizedSearch) {
  filteredUsers = filteredUsers.filter(u =>
    normalizeText(u.id || "").includes(normalizedSearch) ||
    normalizeText(u.owner || "").includes(normalizedSearch) ||
    normalizeText(u.street || "").includes(normalizedSearch) ||
    normalizeText(u.barangay || "").includes(normalizedSearch) ||
    normalizeText(u.city || "").includes(normalizedSearch) ||
    normalizeText(u.permitNo || "").includes(normalizedSearch) ||
    (u.latitude || "").toString().includes(normalizedSearch) ||
    (u.longitude || "").toString().includes(normalizedSearch) ||
    (u.year_conducted || "").includes(normalizedSearch)
  );
}

localStorage.setItem('filteredWaterUsers', JSON.stringify(filteredUsers));

const totalPages = Math.ceil(filteredUsers.length / rowsPerPage) || 1;
if (currentPage > totalPages) currentPage = totalPages;

const start = (currentPage - 1) * rowsPerPage;
const end = start + rowsPerPage;
const usersToDisplay = filteredUsers.slice(start, end);

usersToDisplay.forEach(user => {
  const tr = document.createElement('tr');

  const ownerName = highlightMatch(user.owner || "", searchTerm);

  const badges = [];
  if (user.permitNo) {
    badges.push(`<span class="badge-3d badge-permittee me-1">Permittee: ${user.permitNo}</span>`);
  }
  if (user.isWaterSource) {
    badges.push(`<span class="badge-3d badge-water-source me-1">Water Source</span>`);
  }
  if (user.geotaggedUrl && user.geotaggedUrl.trim() !== "") {
    badges.push(`<span class="badge-3d badge-geotagged me-1">Geotagged</span>`);
  }

  tr.innerHTML = `
    <td>${user.id}</td>
    <td>${ownerName}</td>
    <td>${user.street}</td>
    <td>${user.barangay}</td>
    <td>${user.city}</td>
    <td>${user.permitNo}</td>
    <td>${user.latitude}</td>
    <td>${user.longitude}</td>
    <td>${user.year_conducted}</td>
    <td>${user.isWaterSource ? "Yes" : "No"}</td>
    <td>${user.geotaggedUrl}</td>
    <td>${badges.join("<br>")}</td>
  `;

  table.appendChild(tr);
}

```

```

badges.push(`<span class="badge-3d badge-water-source-bw me-1">Geotagged</span>`);
}

const badgesHTML = badges.length > 0 ? `<div class="mt-1">${badges.join(' ')}</div>` : "";

tr.innerHTML =
<td>${ownerName}${badgesHTML}</td>
<td>${highlightMatch(user.street || "", searchTerm)}</td>
<td>${highlightMatch(user.barangay || "", searchTerm)}</td>
<td>${highlightMatch(user.city || "", searchTerm)}</td>
<td>${highlightMatch(user.latitude ? Number(user.latitude).toFixed(5) : "", searchTerm)}</td>
<td>${highlightMatch(user.longitude ? Number(user.longitude).toFixed(5) : "", searchTerm)}</td>
<td>${highlightMatch(user.year_conducted || "", searchTerm)}</td>
<td>
<button class="btn btn-3d btn-sm btn-warning edit-btn" data-id="${user.id}">
  <i class="bi bi-pencil-square"></i>
</button>
<button class="btn btn-3d btn-sm btn-info geotag-btn" data-id="${user.id}">
  <i class="bi bi-geo-alt-fill"></i>
</button>
<button class="btn btn-3d btn-sm btn-danger" data-id="${user.id}">
  <i class="bi bi-file-earmark-pdf-fill"></i>
</button>
</td>
`;

const geotagBtn = tr.querySelector('.geotag-btn');
geotagBtn.addEventListener('click', () => {
  const geotaggedUrl = user?.geotaggedUrl || "";
  setupImageUploadModal(user.id, geotaggedUrl, user.permitNo);
  const modal = new bootstrap.Modal(document.getElementById('imageUploadModal'));
  modal.show();
});

tableBody.appendChild(tr);
});

attachEditListeners(filteredUsers);
renderPagination(totalPages);
}

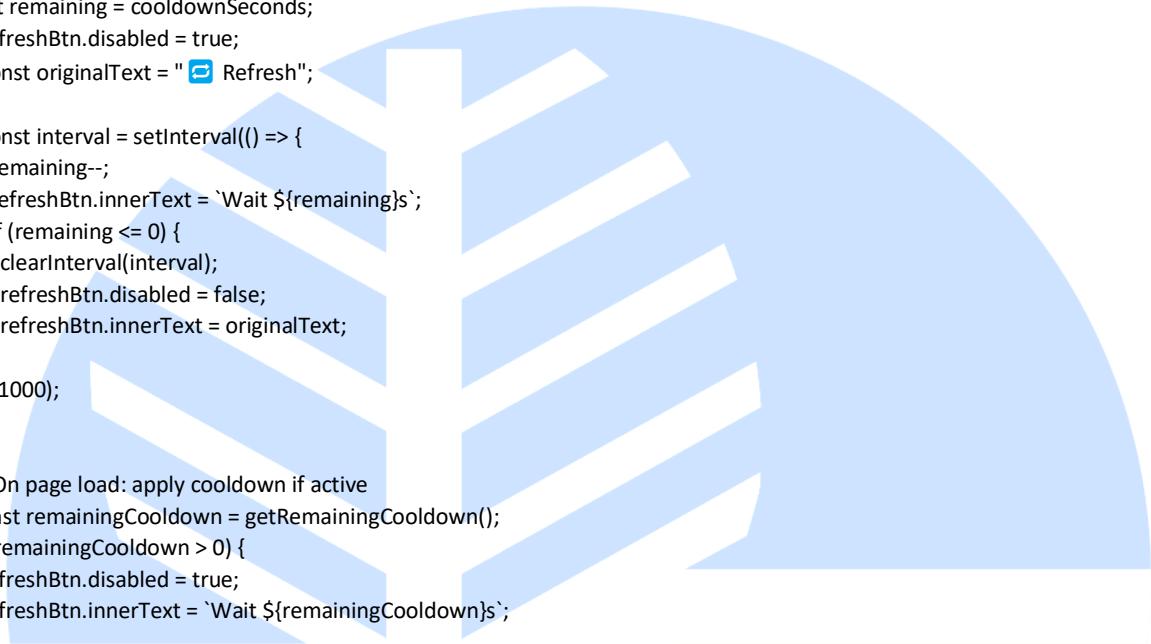
function handleRefreshButton({
  buttonId,
  refreshFn,
  renderFn,
  cooldownKey,
  cooldownSeconds = 60
}) {
  const refreshBtn = document.getElementById(buttonId);
  const LAST_REFRESH_KEY = cooldownKey;

  function getRemainingCooldown() {
    ...
  }

  refreshBtn.addEventListener('click', () => {
    const now = Date.now();
    const lastRefreshTime = localStorage.getItem(LAST_REFRESH_KEY);
    const cooldownMs = cooldownSeconds * 1000;
    const remainingMs = cooldownMs - (now - (lastRefreshTime || 0));
    const remainingSecs = Math.floor(remainingMs / 1000);
    const remainingMins = Math.floor(remainingSecs / 60);
    const remainingHrs = Math.floor(remainingMins / 60);

    if (remainingHrs > 0) {
      renderFn(`Last Refresh: ${remainingHrs} hours ago`);
    } else if (remainingMins > 0) {
      renderFn(`Last Refresh: ${remainingMins} minutes ago`);
    } else if (remainingSecs > 0) {
      renderFn(`Last Refresh: ${remainingSecs} seconds ago`);
    } else {
      renderFn(`Last Refresh: Just now`);
    }
    localStorage.setItem(LAST_REFRESH_KEY, now);
  });
}

```



```
const lastRefresh = localStorage.getItem(LAST_REFRESH_KEY);
if (!lastRefresh) return 0;
const elapsed = (Date.now() - parseInt(lastRefresh, 10)) / 1000;
return Math.max(0, cooldownSeconds - Math.floor(elapsed));
}

function startCooldown() {
  localStorage.setItem(LAST_REFRESH_KEY, Date.now().toString());
  let remaining = cooldownSeconds;
  refreshBtn.disabled = true;
  const originalText = "↻ Refresh";

  const interval = setInterval(() => {
    remaining--;
    refreshBtn.innerText = `Wait ${remaining}s`;
    if (remaining <= 0) {
      clearInterval(interval);
      refreshBtn.disabled = false;
      refreshBtn.innerText = originalText;
    }
  }, 1000);
}

// On page load: apply cooldown if active
const remainingCooldown = getRemainingCooldown();
if (remainingCooldown > 0) {
  refreshBtn.disabled = true;
  refreshBtn.innerText = `Wait ${remainingCooldown}s`;

  let remaining = remainingCooldown;
  const interval = setInterval(() => {
    remaining--;
    refreshBtn.innerText = `Wait ${remaining}s`;
    if (remaining <= 0) {
      clearInterval(interval);
      refreshBtn.disabled = false;
      refreshBtn.innerText = "↻ Refresh";
    }
  }, 1000);
}

refreshBtn.addEventListener('click', async () => {
  const remaining = getRemainingCooldown();
  if (remaining > 0) {
    NotificationBox.show(`Please wait ${remaining}s before refreshing again.`, 'warning');
    return;
  }

  try {
    refreshBtn.disabled = true;
    refreshBtn.innerText = "Refreshing...";
    await refreshFn();
    await WUSData.refreshCache();
  } catch (error) {
    console.error(error);
    refreshBtn.disabled = false;
    refreshBtn.innerText = "↻ Refresh";
  }
});
```

```

        await renderWaterUsers(currentSearchTerm);
        renderFn();
        NotificationBox.show("Refreshed successfully.");
        startCooldown();
    } catch (err) {
        refreshBtn.disabled = false;
        refreshBtn.innerText = "⟳ Refresh";
        console.error(err);
        NotificationBox.show("Error during refresh.", "error");
    }
});

}

function attachEditListeners(filteredUsers) {
    const tableBody = document.getElementById('waterUserTableBody');
    tableBody.querySelectorAll('.edit-btn').forEach(button => {
        button.addEventListener('click', () => {
            const id = button.getAttribute('data-id');
            const user = filteredUsers.find(u => u.id === id);
            if (!user) return;

            // Populate form fields
            document.getElementById('editWusId').value = user.id || '';
            document.getElementById('editOwner').value = user.owner || '';
            document.getElementById('editStreet').value = user.street || '';
            document.getElementById('editBarangay').value = user.barangay || '';
            document.getElementById('editCity').value = user.city || '';
            document.getElementById('editLatitude').value = user.latitude || '';
            document.getElementById('editLongitude').value = user.longitude || '';
            document.getElementById('editType').value = user.type || '';
            document.getElementById('editStatus').value = user.status || '';
            document.getElementById('editRemarks').value = user.remarks || '';
            document.getElementById('editMonthConducted').value = user.month_conducted || '';
            document.getElementById('editYearConducted').value = user.year_conducted || '';
            document.getElementById('editIsWaterSource').checked = !!user.isWaterSource;
            document.getElementById('editRepresentative').value = user.representative || '';
            document.getElementById('editDesignation').value = user.designation || '';
            document.getElementById('editPhone').value = user.phone || '';
            document.getElementById('permitNoInputEdit').value = user.permitNo || '';
            document.getElementById('editWusGeotagUrl').value = user.geotaggedUrl || '';
            document.getElementById('image-signature').src = user.signUrl || './images/noSignatureFound.png';

            const modal = new bootstrap.Modal(document.getElementById('editwusModal'));
            modal.show();
        });
    });
}

function renderPagination(totalPages) {
    const pagination = document.getElementById('pagination');
    pagination.innerHTML = "";

    // Previous Button

```

```
const prevItem = document.createElement('li');
prevItem.className = `page-item ${currentPage === 1 ? 'disabled' : ''}`;
prevItem.innerHTML = `
<a class="page-link" href="#">Previous</a>
`;
prevItem.addEventListener('click', (e) => {
  e.preventDefault();
  if (currentPage > 1) {
    currentPage--;
    renderWaterUsers(searchTerm);
  }
});
pagination.appendChild(prevItem);

// Numbered Buttons (Limit to 5 pages around current page)
const maxVisible = 5;
let startPage = Math.max(1, currentPage - Math.floor(maxVisible / 2));
let endPage = startPage + maxVisible - 1;
if (endPage > totalPages) {
  endPage = totalPages;
  startPage = Math.max(1, endPage - maxVisible + 1);
}

for (let i = startPage; i <= endPage; i++) {
  const pagelitem = document.createElement('li');
  pagelitem.className = `page-item ${i === currentPage ? 'active' : ''}`;
  pagelitem.innerHTML = `<a class="page-link" href="#">${i}</a>`;
  pagelitem.addEventListener('click', (e) => {
    e.preventDefault();
    currentPage = i;
    renderWaterUsers(searchTerm);
  });
  pagination.appendChild(pagelitem);
}

// Next Button
const nextItem = document.createElement('li');
nextItem.className = `page-item ${currentPage === totalPages ? 'disabled' : ''}`;
nextItem.innerHTML = `
<a class="page-link" href="#">Next</a>
`;
nextItem.addEventListener('click', (e) => {
  e.preventDefault();
  if (currentPage < totalPages) {
    currentPage++;
    renderWaterUsers(searchTerm);
  }
});
pagination.appendChild(nextItem);

function handleEditForm() {
  const form = document.getElementById('editWusForm');
```

```

form.addEventListener('submit', async (e) => {
  e.preventDefault();
  e.stopPropagation();

  form.classList.add('was-validated');
  if (!form.checkValidity()) return;

  Spinner.show();
  const submitBtn = form.querySelector('button[type="submit"]');
  submitBtn.disabled = true;

  const id = document.getElementById('editWusId').value;
  const permitNo = document.getElementById('permitNoInputEdit').value.trim();

  const payload = {
    owner: document.getElementById('editOwner').value.trim(),
    street: document.getElementById('editStreet').value.trim(),
    barangay: document.getElementById('editBarangay').value.trim(),
    city: document.getElementById('editCity').value.trim(),
    latitude: document.getElementById('editLatitude').value.trim(),
    longitude: document.getElementById('editLongitude').value.trim(),
    type: document.getElementById('editType').value.trim(),
    status: document.getElementById('editStatus').value.trim(),
    remarks: document.getElementById('editRemarks').value.trim(),
    month_conducted: document.getElementById('editMonthConducted').value.trim(),
    year_conducted: document.getElementById('editYearConducted').value.trim(),
    representative: document.getElementById('editRepresentative').value.trim(),
    designation: document.getElementById('editDesignation').value.trim(),
    phone: document.getElementById('editPhone').value.trim(),
    isWaterSource: document.getElementById('editIsWaterSource').checked,
    permitNo,
    geotaggedUrl: document.getElementById('editWusGeotagUrl').value.trim()
  };

  const permitPayload = {
    permittee: document.getElementById('editOwner').value.trim(),
    diversionPoint: document.getElementById('editCity').value.trim(),
    latitude: document.getElementById('editLatitude').value.trim(),
    longitude: document.getElementById('editLongitude').value.trim(),
    waterSource: document.getElementById('editType').value.trim(),
    permitNo,
    geotaggedUrl: document.getElementById('editWusGeotagUrl').value.trim(),
    visited: true
  };

  try {
    //  Update the WUS entry first
    await WUSData.update(id, payload);

    //  Find the matching permit by permitNo and mark visited = true
    const permits = await Permit.getAll();
    const matchingPermit = permits.find(p => p.permitNo === permitNo);
  }
}

```

```
if (matchingPermit) {
  //await Permit.update(matchingPermit.id, { visited: true });
} else {
  //await Permit.add(permitPayload)
}

// ✅ UI handling after success
const modal = bootstrap.Modal.getInstance(document.getElementById('editwusModal'));
modal.hide();

form.reset();
form.classList.remove('was-validated');

allUsers = await WUSData.fetchAll();
await renderWaterUsers(currentSearchTerm);

NotificationBox.show('Water User updated successfully!');
} catch (err) {
  console.error('Error updating Water User:', err);
  NotificationBox.show('Failed to update. Please try again.', 'error');
} finally {
  Spinner.hide();
  submitBtn.disabled = false;
}
});

function initializeearchBar() {
  const searchBarInput = document.getElementById('searchInput');
  searchBarInput.addEventListener('input', () => {
    currentSearchTerm = searchBarInput.value;
    renderWaterUsers(currentSearchTerm);
  });
}

function initializeExportButton() {
  const exportBtn = document.getElementById('exportBtn');
  if (!exportBtn) return;

  exportBtn.addEventListener('click', () => {
    const XLSX = window.XLSX; // Get XLSX from global scope

    if (!XLSX || !XLSX.utils) {
      alert('XLSX library not loaded.');
      return;
    }

    const filteredUsers = JSON.parse(localStorage.getItem('filteredWaterUsers') || '[]');

    if (filteredUsers.length === 0) {
      alert('No filtered data to export.');
      return;
    }
  });
}
```

```

}

const dataForExcel = filteredUsers.map(user => ({
  Owner: user.owner || '',
  Street: user.street || '',
  Barangay: user.barangay || '',
  City: user.city || '',
  Latitude: user.latitude || '',
  Longitude: user.longitude || '',
  Source: user.waterSource || '',
  Status: user.status || '',
  Representative: user.representative || '',
  Designation: user.designation || '',
  Phone: user.phone || '',
  Year_Inspected: user.year_conducted || '',
  Month_Inspected: user.month_conducted || '',
  IsWaterSource: user.isWaterSource ? 'Yes' : 'No'
}));

const worksheet = XLSX.utils.json_to_sheet(dataForExcel);
const workbook = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(workbook, worksheet, 'FilteredUsers');
XLSX.writeFile(workbook, 'WaterUsersAndSources.xlsx');
});

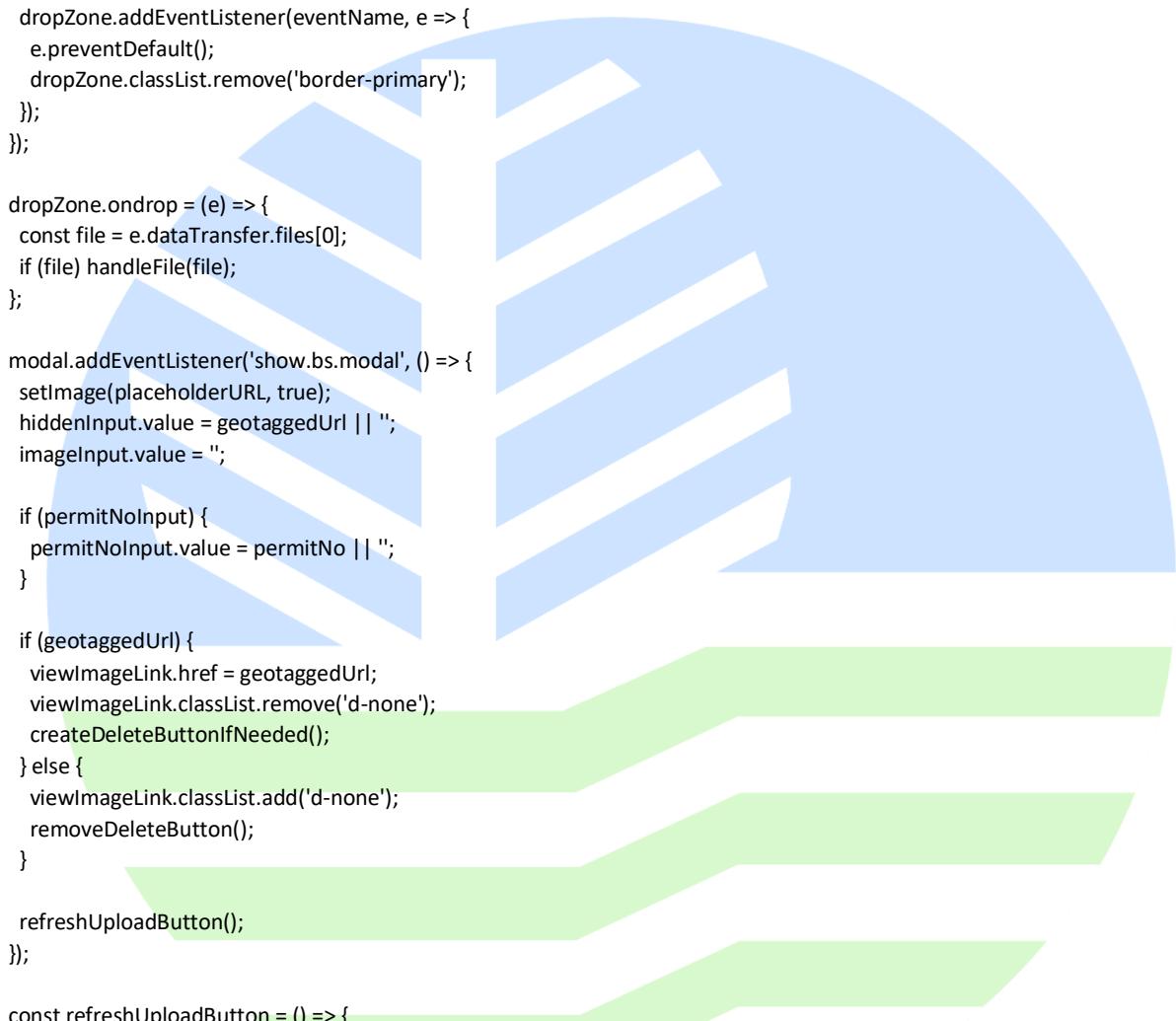
}

function setupImageUploadModal(wusId, geotaggedUrl = "", permitNo = "") {
  const imageInput = document.getElementById('imageInput');
  const imagePreview = document.getElementById('imagePreview');
  const modal = document.getElementById('imageUploadModal');
  const dropZone = document.getElementById('dropZone');
  const placeholderURL = './images/placeholder2.png';
  const viewImageLink = document.getElementById('viewImageLink');
  const hiddenInput = document.getElementById('currentGeotaggedUrl');
  const permitNoInput = document.getElementById('currentPermitNo');

  const setImage = (src, isPlaceholder = true) => {
    imagePreview.src = src;
    imagePreview.classList.toggle('placeholder', isPlaceholder);
    const uploadBtn = document.getElementById('uploadGeotaggedBtn');
    if (uploadBtn) {
      uploadBtn.classList.toggle('d-none', isPlaceholder);
    }
  };

  const handleFile = (file) => {
    if (file && file.type.startsWith('image/')) {
      const reader = new FileReader();
      reader.onload = (e) => setImage(e.target.result, false);
      reader.readAsDataURL(file);
    }
  };
}

```



```
[ 'dragenter', 'dragover' ].forEach(eventName => {
  dropZone.addEventListener(eventName, e => {
    e.preventDefault();
    dropZone.classList.add('border-primary');
  });
});

[ 'dragleave', 'drop' ].forEach(eventName => {
  dropZone.addEventListener(eventName, e => {
    e.preventDefault();
    dropZone.classList.remove('border-primary');
  });
});

dropZone.ondrop = (e) => {
  const file = e.dataTransfer.files[0];
  if (file) handleFile(file);
};

modal.addEventListener('show.bs.modal', () => {
  setImage(placeholderURL, true);
  hiddenInput.value = geotaggedUrl || '';
  imageInput.value = '';

  if (permitNoInput) {
    permitNoInput.value = permitNo || '';
  }

  if (geotaggedUrl) {
    viewImageLink.href = geotaggedUrl;
    viewImageLink.classList.remove('d-none');
    createDeleteButtonIfNeeded();
  } else {
    viewImageLink.classList.add('d-none');
    removeDeleteButton();
  }
}

refreshUploadButton();
});

const refreshUploadButton = () => {
  let uploadBtn = document.getElementById('uploadGeotaggedBtn');
  if (!uploadBtn || !uploadBtn.parentNode) {
    console.warn('uploadBtn or its parentNode is null. Skipping button refresh.');
    return;
  }

  const newBtn = uploadBtn.cloneNode(true);
  uploadBtn.parentNode.replaceChild(newBtn, uploadBtn);
  uploadBtn = newBtn;
  uploadBtn.classList.add('d-none');
  uploadBtn.addEventListener('click', () => handleGeotaggedUpload(wusId, permitNo));
};


```



```
imageInput.onchange = () => {
  const file = imageInput.files[0];
  const uploadBtn = document.getElementById('uploadGeotaggedBtn');
  if (file && file.type.startsWith('image/')) {
    const reader = new FileReader();
    reader.onload = (e) => {
      setImage(e.target.result, false);
    }
    if (uploadBtn) {
      uploadBtn.classList.remove('d-none');
    }
  };
  reader.readAsDataURL(file);
}

function createDeleteButtonIfNeeded() {
  removeDeleteButton();

  const deleteBtn = document.createElement('button');
  deleteBtn.id = 'deleteGeotaggedBtn';
  deleteBtn.className = 'btn btn-3d btn-outline-danger';
  deleteBtn.innerHTML = '<i class="bi bi-trash me-1"></i> Delete Image';

  deleteBtn.addEventListener('click', async () => {
    Spinner.show();
    try {
      deleteBtn.disabled = true;
      deleteBtn.innerHTML = '<span class="spinner-border spinner-border-sm me-1"></span> Deleting...';

      await GeotaggedFileService.delete(hiddenInput.value);

      if (permitNo) {
        try {
          const permits = await Permit.getAll();
          const targetPermit = permits.find(p => p.permitNo === permitNo);
          if (targetPermit) {
            await Permit.update(targetPermit.id, { geotaggedUrl: "" });
          } else {
            console.warn(`Permit with Permit No "${permitNo}" not found. Skipping Permit update.`);
          }
        } catch (permitErr) {
          console.warn("Error updating permit record:", permitErr);
        }
      }
      try {
        if (wusId) {
          await WUSData.update(wusId, { geotaggedUrl: "" });
        } else {
          console.warn("No userId provided. Skipping WUSData update.");
        }
      } catch (wusErr) {
        console.error("Error updating WUSData record:", wusErr);
      }
    } finally {
      deleteBtn.disabled = false;
      deleteBtn.innerHTML = '<i class="bi bi-trash me-1"></i> Delete Image';
    }
  });
}
```

```
        console.warn("Error updating WUS record:", wusErr);
    }

    setImage(placeholderURL, true);
    hiddenInput.value = "";
    viewImageLink.classList.add('d-none');
    removeDeleteButton();

    allUsers = await WUSData.fetchAll();
    await renderWaterUsers(currentSearchTerm);

    NotificationBox.show('Image deleted and water user updated.');
} catch (err) {
    NotificationBox.show(`Failed to delete image: ${err.message}`, 'error');
} finally {
    deleteBtn.disabled = false;
    deleteBtn.innerHTML = '<i class="bi bi-trash me-1"></i> Delete Image';
    Spinner.hide();
}
});

viewImageLink.parentElement.appendChild(deleteBtn);
}

function removeDeleteButton() {
    const deleteBtn = document.getElementById('deleteGeotaggedBtn');
    if (deleteBtn) deleteBtn.remove();
}

async function handleGeotaggedUpload(wusId, permitNo) {
    Spinner.show();

    try {
        const fileInput = document.getElementById("imageInput");
        const file = fileInput.files[0];

        if (!file) {
            NotificationBox.show("No file selected.", "warning");
            return;
        }

        const existingUrl = document.getElementById('currentGeotaggedUrl').value;

        // Delete old image from storage
        if (existingUrl) {
            await GeotaggedFileService.delete(existingUrl);
        }

        // Upload new image
        const newImageUrl = await GeotaggedFileService.upload(file);
        if (!newImageUrl) {
            NotificationBox.show("Upload failed.", "error");
        }
    } catch (err) {
        NotificationBox.show(`Failed to upload image: ${err.message}`, 'error');
    }
}
```

```

        return;
    }

let permitUpdated = false;

// ✅ Try updating the Permit if permitNo exists
if (permitNo) {
    try {
        const permits = await Permit.getAll(); // cached
        const targetPermit = permits.find(p => p.permitNo === permitNo);

        if (targetPermit) {
            await Permit.update(targetPermit.id, {
                geotaggedUrl: newImageUrl
            });
            permitUpdated = true;
        } else {
            console.warn(`Permit No "${permitNo}" not found in cache. Skipping permit update.`);
        }
    } catch (permitErr) {
        console.error("Error updating permit record:", permitErr);
    }
}

// ✅ Always update WUS record
await WUSData.update(wusId, {
    geotaggedUrl: newImageUrl
});

allUsers = await WUSData.fetchAll();
await renderWaterUsers(currentSearchTerm);

NotificationBox.show(
    permitUpdated
    ? "Geotagged image updated for both Water User and Permit."
    : "Geotagged image updated for Water User only."
);

// Close modal
const modalEl = document.getElementById('imageUploadModal');
const modal = bootstrap.Modal.getInstance(modalEl);
if (modal) modal.hide();
} catch (error) {
    console.error("Error during upload:", error.message);
    NotificationBox.show("An error occurred: " + error.message, "error");
} finally {
    Spinner.hide();
}
}

export function showWaterUserPDF() {
    document.getElementById('waterUserTableBody').addEventListener('click', async function (e) {
        const pdfBtn = e.target.closest('.btn-danger');

```

```
if (pdfBtn && pdfBtn.querySelector('.bi-file-earmark-pdf-fill')) {  
    const userId = pdfBtn.getAttribute('data-id');  
  
    const cachedUsers = JSON.parse(localStorage.getItem('filteredWaterUsers') || '[]');  
    const user = cachedUsers.find(u => u.id === userId);  
  
    if (!user) {  
        NotificationBox.show('User data not found.', 'error');  
        return;  
    }  
  
    // ✅ Generate PDF blob  
    const blob = await generateWaterUserPDF(user, { returnBlob: true });  
    const blobUrl = URL.createObjectURL(blob);  
  
    // ✅ Show in modal  
    const iframe = document.getElementById('pdfViewer');  
    iframe.src = blobUrl;  
  
    const modal = new bootstrap.Modal(document.getElementById('pdfModal'));  
    modal.show();  
}  
});  
}
```

XV.D. Docker



```
# Use Node.js Alpine image  
FROM node:20-alpine  
  
# Create app directory  
WORKDIR /app  
  
# Copy package files  
COPY package*.json ./  
  
# Install dependencies  
RUN npm install  
  
# Copy everything else  
COPY ..  
  
# Expose port and start server  
EXPOSE 3000  
CMD ["node", "server.js"]
```



XV.E. Node.js

server.js

```
import express from "express";
import path from "path";
import { fileURLToPath } from "url";

const app = express();
const port = process.env.PORT || 3000;

const __dirname = path.dirname(fileURLToPath(import.meta.url));
app.use(express.static(path.join(__dirname, "public")));

app.get("*", (req, res) => {
  res.sendFile(path.join(__dirname, "public", "index.html"));
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

XVI. Conclusion

The WRUS Portal was developed to address the challenges of manual record-keeping, fragmented files, and limited access to real-time data faced by the Water Resources Utilization Section. By consolidating water permits, user and source inventories, consumables management, and analytics into a single web-based platform, the system reduces inefficiencies and provides more reliable data management.

With the integration of cloud technologies, geospatial mapping, and offline data handling, the portal equips field personnel and office staff with the tools needed to maintain accurate and accessible records, even in areas with limited internet connectivity. This supports timely monitoring, improves data validation, and streamlines reporting across all operational functions.

Ultimately, the WRUS Portal enhances organizational efficiency and strengthens sustainable water resource management. It improves daily workflows and supports informed decision-making within the Department of Environment and Natural Resources, aligning with its mandate of regulatory compliance and responsible utilization of water resources.

System URL Link

<https://wrus-asset-tracking-system.web.app/>

Github Page

<https://github.com/redsmite/wrus-asset-tracking-system>

XVII. Reference

- [1] <https://www.merriam-webster.com/dictionary/system>
- [2] <https://www.freecodecamp.org/news/javascript-modules-explained-with-examples/>
- [3] https://dictionary.cambridge.org/dictionary/english/database#google_vignette
- [4] <https://www.w3schools.com/js/>
- [5] <https://tool.geoimgr.com/>
- [6] <https://www.globalsign.com/en/blog/electronic-signatures-vs-digital-signatures#what-is-electronic-signature>
- [7] <https://blog.logrocket.com/localstorage-javascript-complete-guide/>
- [8] https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API
- [9] <https://www.ibm.com/think/topics/api>
- [10] <https://www.cloudflare.com/learning/cloud/what-is-the-cloud/>
- [11] <https://cloud.google.com/products/firestore?hl=en>
- [12] <https://how.dev/answers/what-is-a-firebase-collection>
- [13] <https://aws.amazon.com/caching/>
- [14] <https://www.merriam-webster.com/dictionary/ledger>
- [15] <https://geojson.org/>
- [16] <https://www.ibm.com/think/topics/docker>
- [17] <https://dictionary.cambridge.org/dictionary/english/user-interface>
- [18] <https://www.oracle.com/asean/business-analytics/what-is-analytics/>
- [19] <https://www.indeed.com/career-advice/career-development/what-is-source-code>
- [20] <https://www.ibm.com/think/topics/web-hosting>
- [21] https://www.w3schools.com/bootstrap/bootstrap_ref_js_modal.asp
- [22] <https://www.ibm.com/think/topics/data-flow-diagram>
- [23] <https://aws.amazon.com/what-is/ide/>
- [24] https://www.w3schools.com/bootstrap/bootstrap_get_started.asp
- [25] <https://airfocus.com/glossary/what-is-a-front-end/>
- [26] <https://airfocus.com/glossary/what-is-a-back-end/>
- [27] https://www.w3schools.com/php/php_sessions.asp
- [28] <https://digital.gov/resources/an-introduction-github>
- [29] <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=en>
- [30] <https://www.ibm.com/think/topics/chatgpt>