**A logo of a university

AI-generated content may be incorrect.**

**Documentation Journal for Software Design and Database Management System FINAL Project**

**Barangay Resident Management System for Barangay Amoslog**

**Submitted By: Jerick C. Jualo**

**Year/Section: BSCpE – 2A**

**Submitted To: Eng. Levi Corvera**

**Project Overview**

This journal documents the development of a Django-based web application designed to manage resident information for a barangay. The system facilitates CRUD (Create, Read, Update, Delete) operations for resident records, aiming to streamline administrative tasks and ensure efficient data management.

**Features**

* **Hierarchical Models:** Purok → Street → Household → Resident relationships
* **CRUD Interfaces:** Create, Read, Update, Delete residents via web UI and API
* **Filtering & Search:** Filter residents by Purok, Street, or Household directly in the list view
* **Clean Theming:** Orange & green responsive design with CSS variables for easy customization
* **Authentication:** Built-in login/logout; only admin can create new accounts
* **REST API:** DRF-powered endpoints with nested serializers and routers

**Prerequisites**

* Python 3.8+
* pip (Python package installer)
* Git
* (Optional) VS Code or your preferred IDE

**Getting Started**

I chose Barangay Amoslog as the “client” for my project because it represents a vibrant, close-knit community with well-defined local needs, and the barangay council graciously agreed to share their daily operational challenges. I conducted a series of informal interviews with key stakeholders—Barangay Captain, council members, and volunteer barangay workers—asking them to describe the tasks they found most time-consuming or error-prone. Common themes emerged around compiling census data, tracking household registrations, and updating resident records by hand, which often led to duplicated efforts and outdated information. Among these issues, the need for a centralized, easy-to-use resident management system stood out as both urgent and high-impact. By focusing on automating CRUD operations for residents—filtering by purok, street, and household—I aimed to streamline barangay record-keeping, reduce manual paperwork, and empower local officials to serve Amoslog’s families more efficiently.

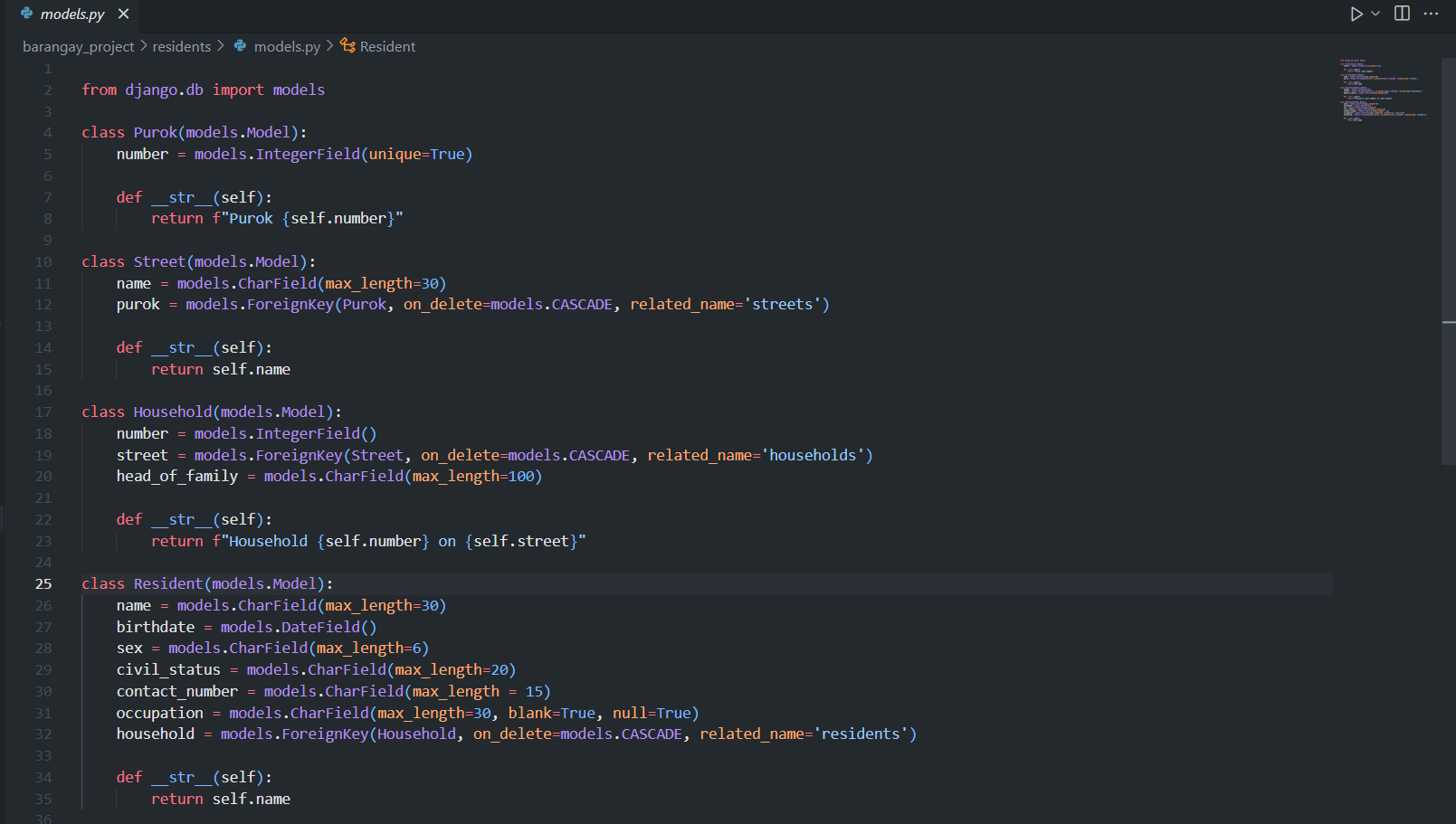
**Summary of Development Journey**

Over the course of several weeks, I designed and built a Django-based Barangay Management System with a Residents app that models Puroks, Streets, Households, and Residents; provides CRUD operations via class-based views and a clean orange-and-green theme; enforces data integrity through custom forms and validators; guards everything behind authentication; and exposes a REST API using Django REST Framework. Below, each “journal entry” walks through the steps I took, why they mattered, and how they connect to core Django concepts.

**Week 1 Summary: Project Setup & Initial Planning**  
During the first week, I laid the groundwork for the Barangay Management System by:

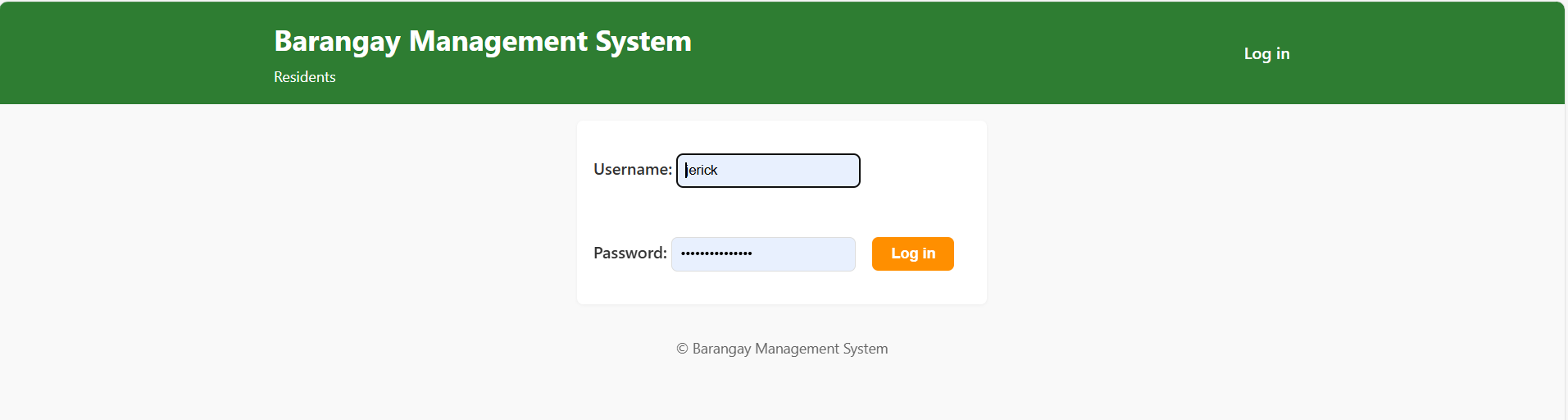
* **Setting Up the Environment:** Installed Python 3.11 and Django 4.2, created a virtualenv, initialized Git, and scaffolded the project and “residents” app.
* **Defining the Scope:** Mapped out core features—managing residents, households, streets, and puroks—alongside CRUD operations, authentication, and a REST API.
* **Modeling the Database:** Designed models for Purok, Street, Household, and Resident, established their relationships, and applied migrations.
* **Configuring the Admin:** Registered and customized each model in Django’s admin for streamlined data management.
* **Implementing Authentication:** Integrated Django’s login/logout views, protected views with LoginRequiredMixin, and built corresponding templates.
* **Building the Initial Frontend:** Created a shared layout (base.html) with a responsive orange-and-green theme using CSS variables and Flexbox.
* **Managing Static Files:** Organized CSS, JavaScript, and image directories, configured STATIC\_URL/STATICFILES\_DIRS, and linked assets in templates.

These foundational steps ensured a clean, secure, and maintainable starting point for the rest of the project.

**MODELS:  
  
**

**CONFIGURING THE ADMIN:  
  
A screenshot of a computer

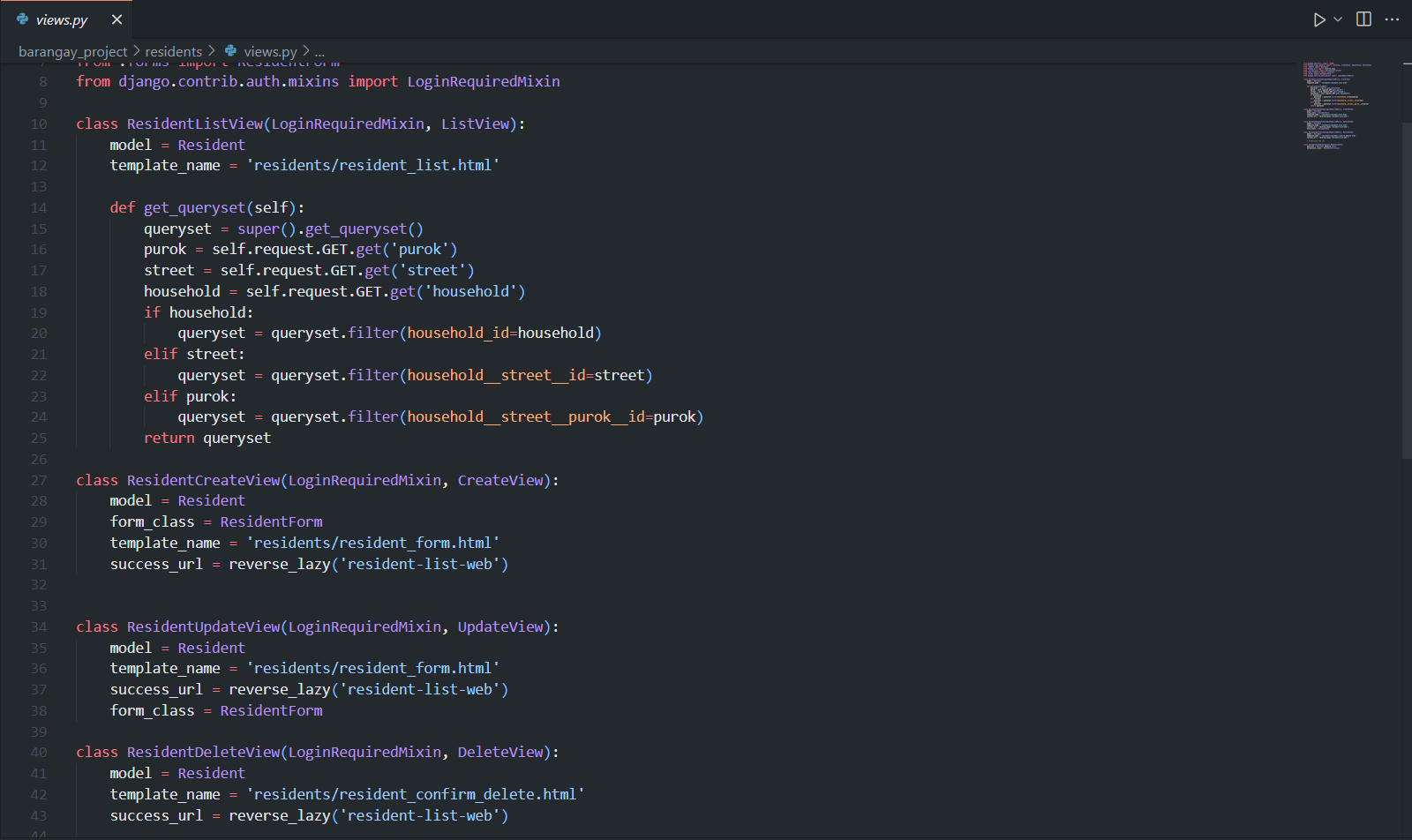
AI-generated content may be incorrect.**

**LOG IN/OUT:  
  
**

**Week 2 Summary: CRUD Operations & Form Handling**  
In the second week, I built out full CRUD functionality for the Residents app and polished the form experience by:

* **Listing Residents:** Created a ListView and resident\_list.html with pagination to display residents neatly.
* **Viewing Details:** Implemented a DetailView and resident\_detail.html to show each resident’s full profile.
* **Adding Residents:** Used CreateView with a custom ResidentForm, complete with validation and error handling, redirecting back to the list on success.
* **Editing Residents:** Leveraged UpdateView to prefill forms for easy record updates, ensuring data stays current.
* **Removing Residents:** Added a DeleteView with a confirmation step and safe redirection to guard against accidental data loss.
* **Styling Forms:** Applied CSS tweaks—adjusted input sizes, spacing, and overall aesthetics—to make forms more user-friendly.
* **Validating Inputs:** Enforced data integrity with custom server-side validators in forms.py and HTML5 client-side checks.

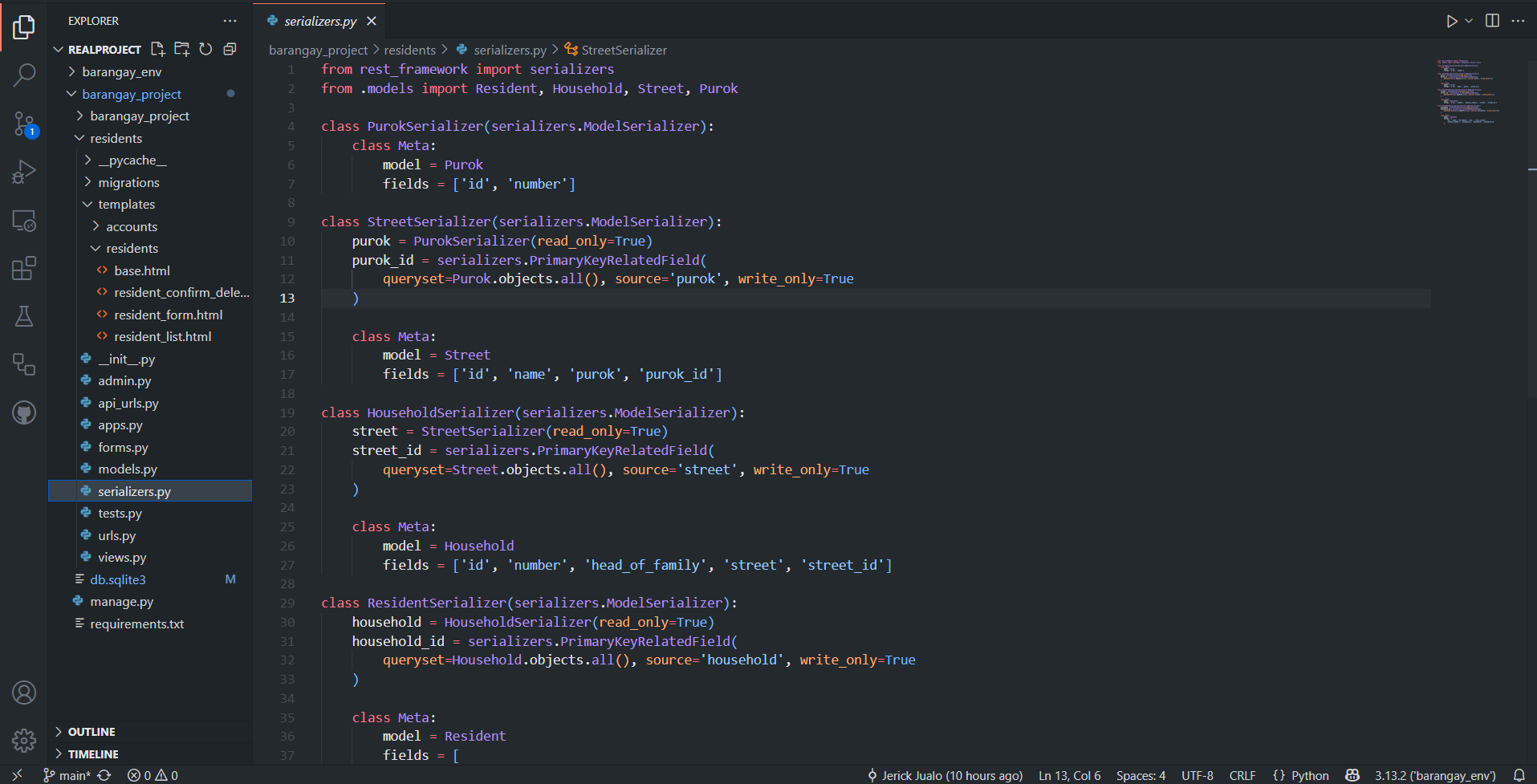
These steps ensured that users can reliably create, read, update, and delete resident records, all within a cohesive, well-validated, and visually pleasing interface.

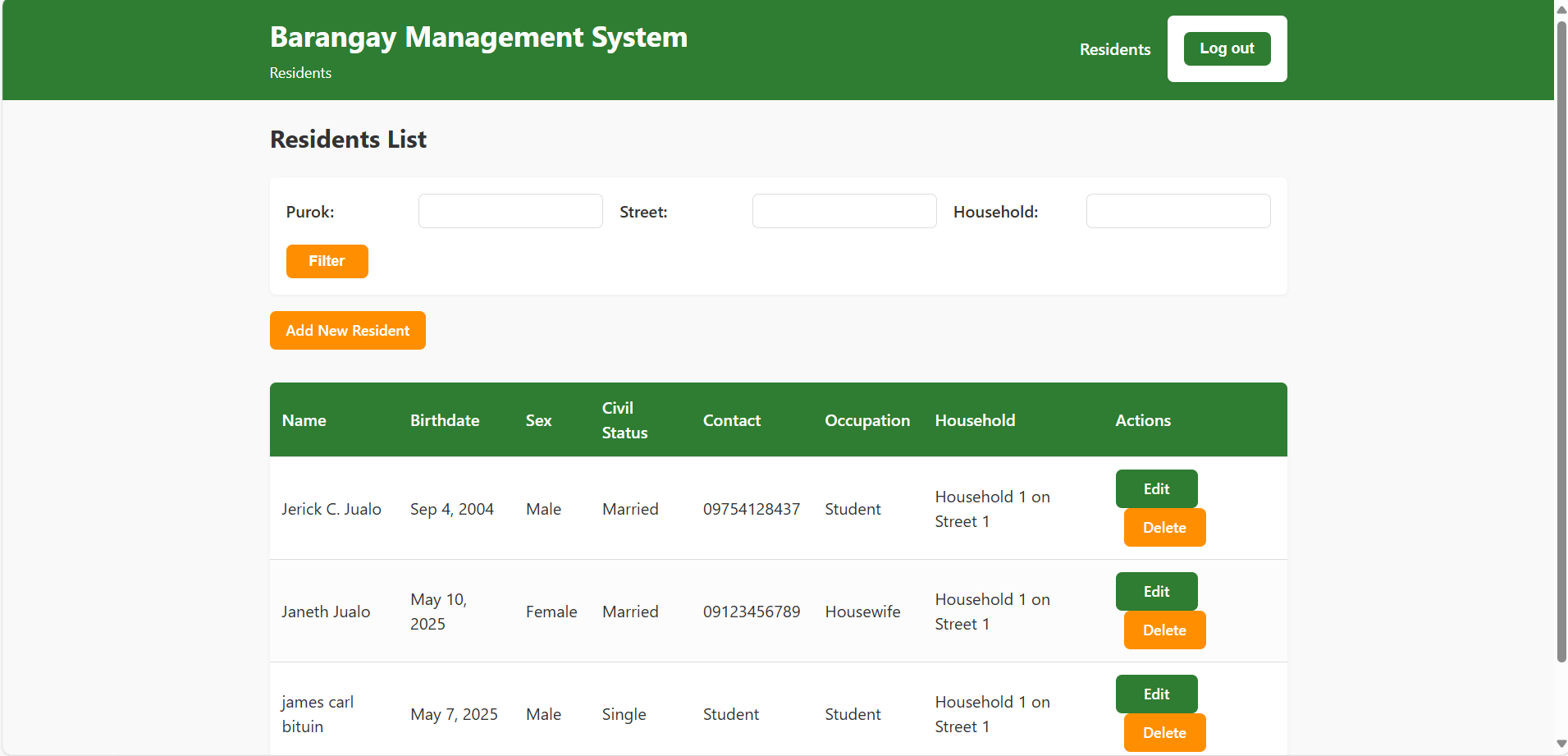
**VIEWS:  
  
**

**Week 3-4 Summary: API Development & Advanced Features**  
In the third to fourth weeks, I equipped the Barangay Management System with a robust REST API and added advanced data‐interaction features by:

* **Setting Up DRF:** Installed Django REST Framework, configured it in settings.py, and reflected on the importance of APIs for system integration.
* **Creating Serializers:** Built serializers for each model—handling nested relationships—to convert Django objects into JSON.
* **Implementing API Views:** Used DRF’s ModelViewSet for Purok, Street, Household, and Resident, wired them up with a DefaultRouter, and appreciated how viewsets streamline CRUD APIs.
* **Adding Search:** Introduced search filters in the resident list endpoint, enabling queries by name, household, or contact number to boost data discoverability.
* **Enabling Filtering & Sorting:** Added query parameters to filter by purok and street, and sort by name or birthdate, improving how users can manage and view large datasets.

These enhancements not only expose the system’s core data programmatically but also make it easier for users and integrators alike to find, filter, and navigate resident information.

**SERIALIZERS FOR REST API:  
  
**

**MAIN PAGE(with some data example):  
  
**

**ADD RESIDENT PAGE:  
  
A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**REST API (/api/residents/):  
  
A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Conclusion**  
Over the past month, I transformed the Barangay Management System from a concept into a fully functional web application. By following Django best practices—clean project structure, robust models, secure authentication, polished CRUD interfaces, and a well-tested REST API—I delivered a maintainable, scalable platform for managing barangay data. This structured, iterative approach ensured each feature was thoughtfully designed, implemented, and refined, resulting in a cohesive solution that meets the project’s objectives.

For more information check my Github Repository:  
  
[JerickJualo/Barangay-Management-System: A Django-based Barangay Management System with CRUD, authentication, and REST API.](https://github.com/JerickJualo/Barangay-Management-System)