Overview Report

Garion, Adrian, Ethan

For our project, we decided to use the “Utility Service System” with database integration as a bonus aspect of the rubric.

Our project has the following class structure:

- **utilityService** (represents a utility service)
- **bill** (represents a service bill)
- **customer** (represents a service customer)
- **customer_menu** (the command line interface for customer options)
- **provider** (represents a service provider)
- **provider_menu** (the command line interface for provider options)
- **dbManager** (handles database communication and operations)
- **main** (offers a main menu for login/registration)

How does the system work?

- This system acts as a portal to manage utility services from the perspective of a customer (**customer class**) or a utility provider (**provider class**).
- The program begins with a login/registration for both customers and providers (**main class**).
- Logging in as a provider (**provider_menu class**) using a unique provider ID allows you to manage the providers offered utility services (**provider class**), including adding/editing/deleting services, viewing existing services, and viewing total sales from services.
- Logging in as a customer (**customer_menu class**) using a unique customer ID allows you to manage that customer's services and operations surrounding them (**customer class**), including searching for services and subscribing to them, viewing current subscribed services, and paying bills.
- When a customer subscribes to a service, a bill is generated (**bill class**) and that bill is associated with a specific customer and a bill is also for a subscribed service which is under a provider.
- During any modifications to the customers, providers, services, and bills alongside local adjustments to container the database is also modified (**dbManger class**).

Phases of Development

Phase 1 (Set Up):
The first step we took as a group was to lay the groundwork for the project. This included:

- Writing up a written overview of the system to determine what the system would exactly do and how we wanted to implement those ideas
- Creating a Discord server as a communication platform for messaging/meetings, and creating a schedule of when to meet
- Setting up a GitHub for file code sharing/management/branching, etc
- Creating a class diagram to visualize how these written rough ideas would look as classes, functions, and attributes
- Creating a database to be populated and integrated later into our code

Phase 2 (Development):
Now that we had all the tools we needed to begin development, we assigned roles and began:

- Garion developed: bill, customer, and customer_menu classes
- Adrian developed: utilityService, provider, and provider_menu classes
- Ethan developed: dbManager class, database integration in all 8 classes

Phase 3 (Finalization):
With the development now complete, we had to put our work together by:

- Merging all classes together and creating a main to run the code as a whole, not just in our own parts
- Teaching each other how to use their features
- Identifying errors and missing functions that needed to be fixed/implemented, and dividing the jobs
- Commenting and formatting all code

Running the Code

To run our code and see how this system works in action simply refer to the README file, which will walk you through the compile/running commands and even a guide to viewing the SQLite database.