**NYC Water Charges**

Rachel Friedman | rfriedman113@gmail.com

CISC 3810 | Project 2 | Documentation

[Click here to access the web application](https://nyc-water-consumption.herokuapp.com/)

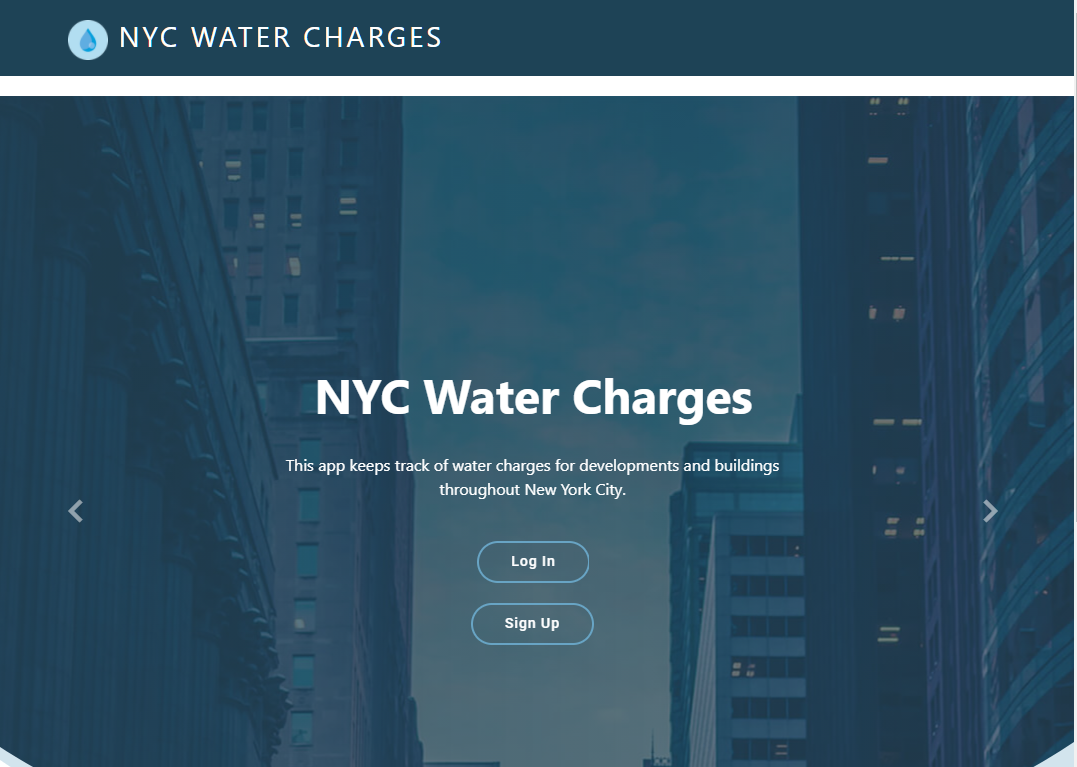
Important! Please note:

Since I am using the free tiers of Heroku:

1. The initial load time of the application may take up to a full minute. Please be patient.
2. Since the maximum records allowed was 10K, I had to combine some tables and set it up differently than my original schema in order to save rows. I also had to eliminate some data, so I kept bills going back to 2018 only (instead of 2013).

Table of Contents

* [About this Project](#about) 2
* [What is the purpose of this software?](#purpose) 2
* NYCHA Users (Admin users)
  + [Log In](#admin) 3
  + [View Buildings](#admin_bldg) 4
  + [View Charges](#admin_charges) 6
  + [Delete Bill](#delete) 7
  + [Edit Charge](#edit) 9
* Development / Building Managers (Other users)
  + [Create an account](#signup) 12
  + [Log In](#login_user) 11
  + [View Buildings](#user_bldg) 15
  + [View Charges](#user_bills) 16
* [List of Developments and Development IDs](#dev_ids) 12
* [Schemas and ER Diagrams](#schemas_er) 18
* [Implementing CRUD features](#crud) 17
* [Technical Documentation](#technical) 17
* [Hosting on Heroku](#heroku) 19



About this Project

This web application was created for CISC 3810. The goal of this project was to create a full stack application, complete with a database layer, business layer and a user-friendly front end interface. I stored the data in a PostgreSQL database, which I then hosted on Heroku. I used Python for the business layer, with Flask for the web framework, along with SQLAlchemy as an Object Relational Mapper and flask-login to handle user authentication and sessions. I used HTML 5, CSS, Javascript (some jQuery) to create the front end. I used some Bootstrap features for the design, as well as the data tables.

What purpose does this software serve?

The primary purpose of this software is for the New York City Housing Authority to keep track of all water charges related bills for all developments and buildings in New York City.

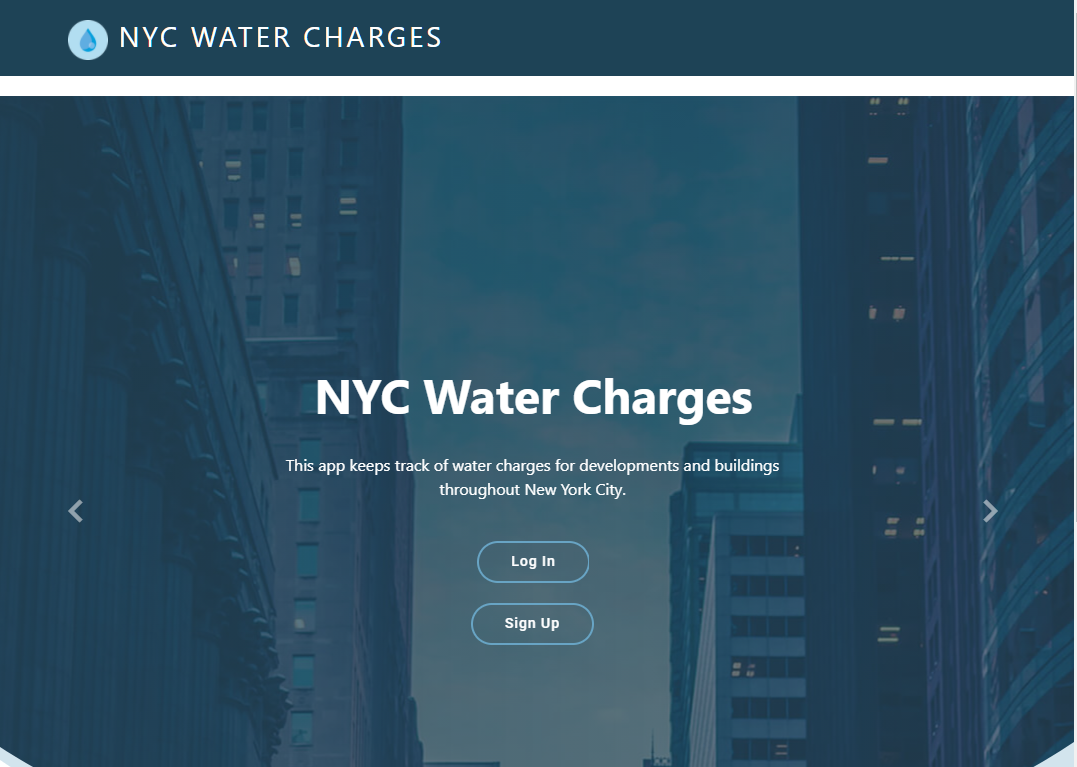
The secondary purpose of this software is for managers of those developments and buildings to be able to view the water service charges associated with their developments and buildings.

The software provides different levels of access for NYCHA users and for development/building users. NYCHA users have the ability to edit and delete charges. Development/Building users have the ability to view (but not edit or delete) charges.

Login (NYCHA Users)

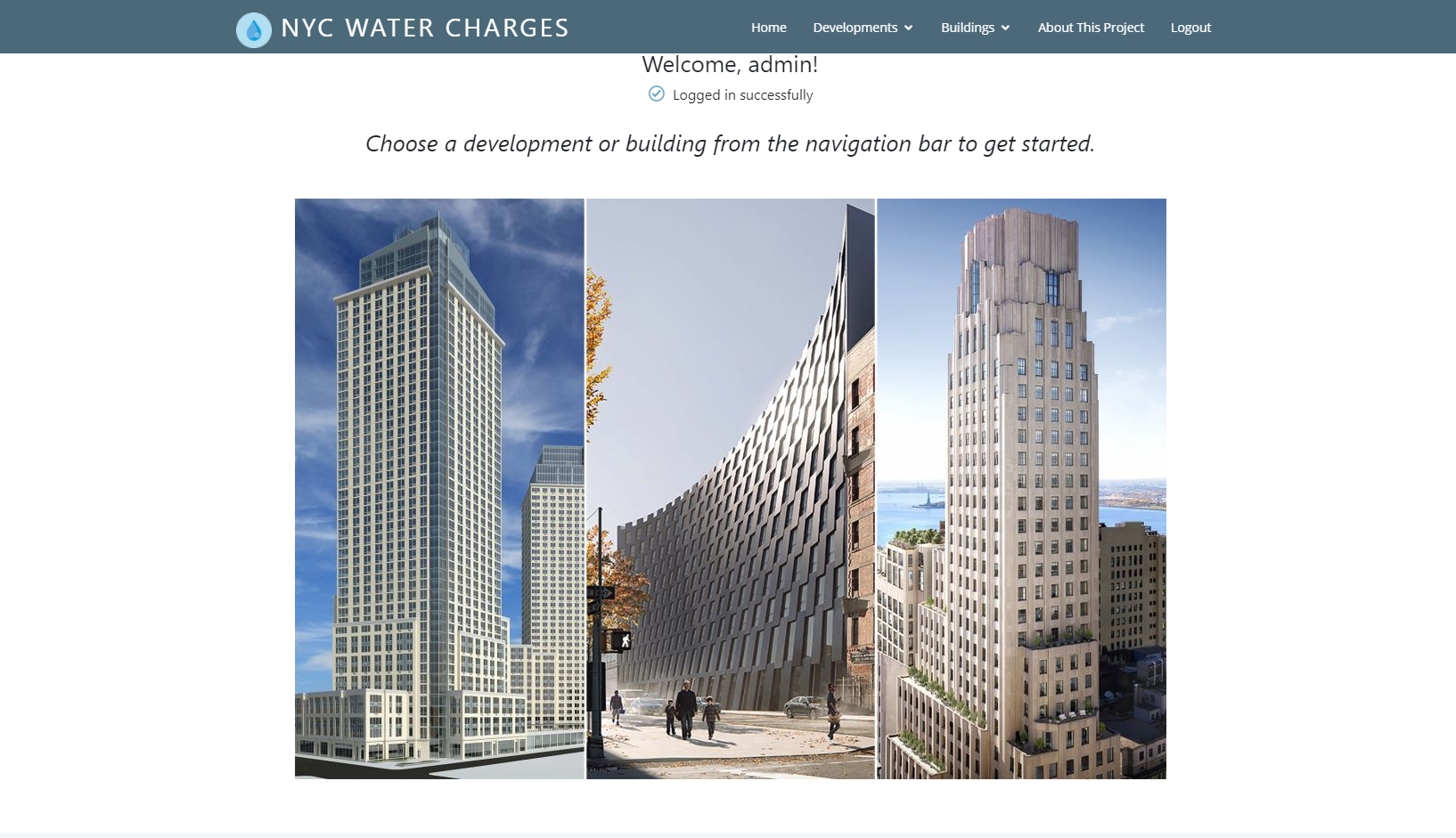
Log In with the following credentials:

* Username: admin
* Password: nycha

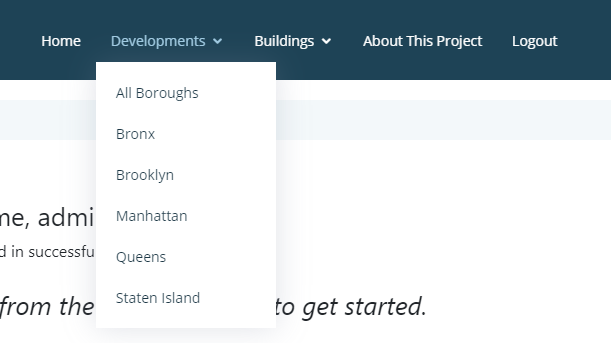


View Developments and Buildings (NYCHA Users)

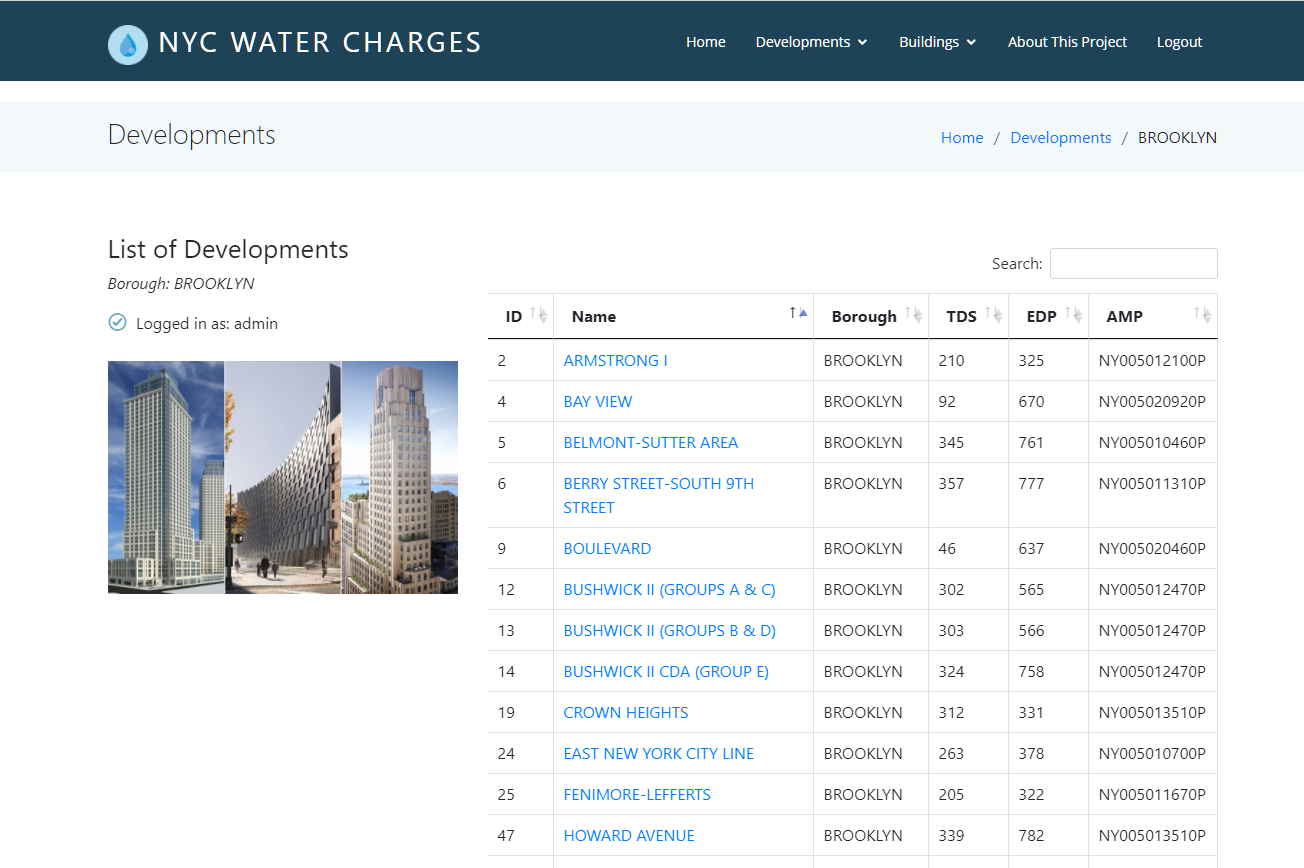
1. Click on development (or Building) from the navigation bar.



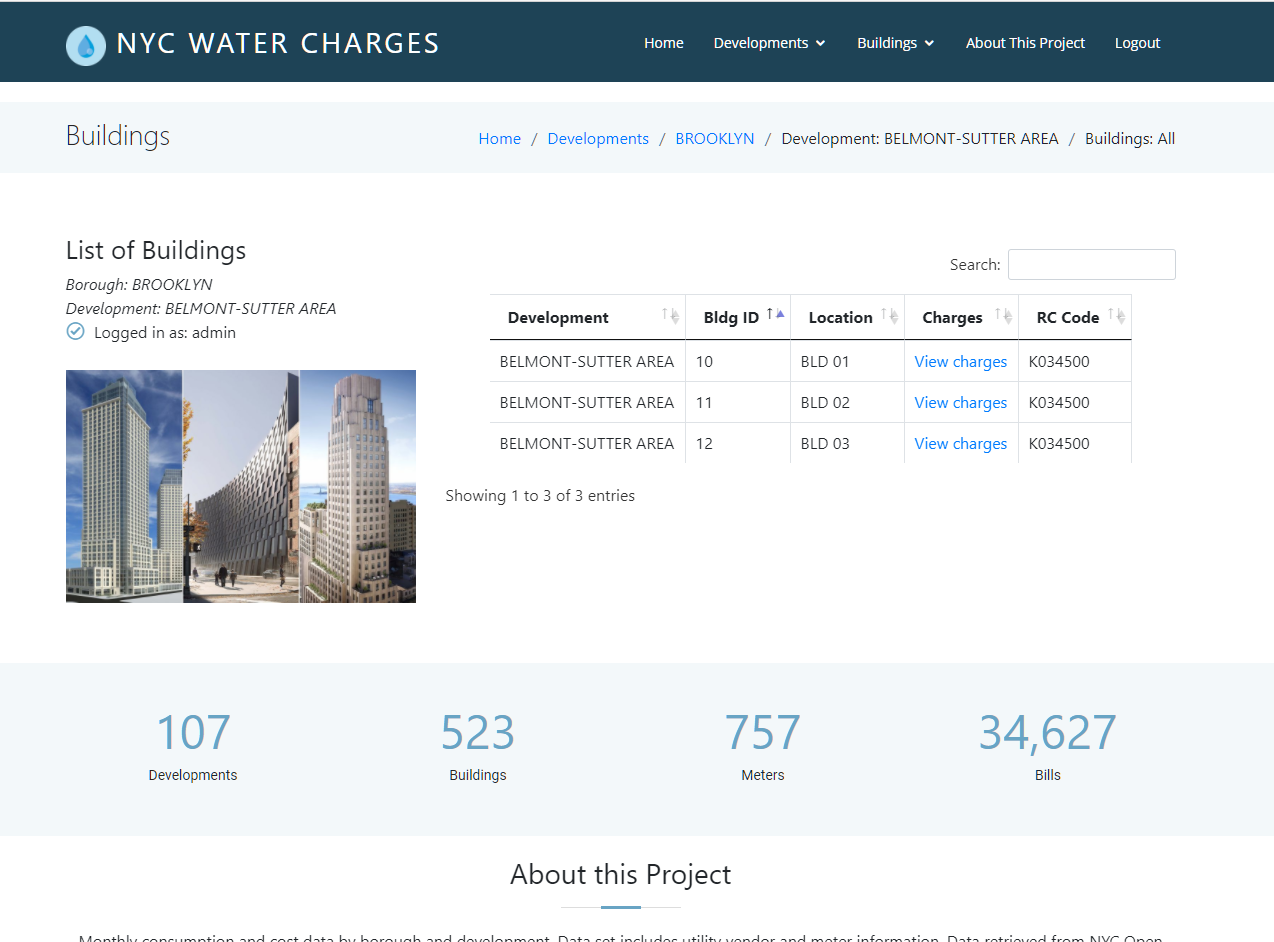
1. Choose a borough.



1. You will be presented with a list of all developments in that borough.

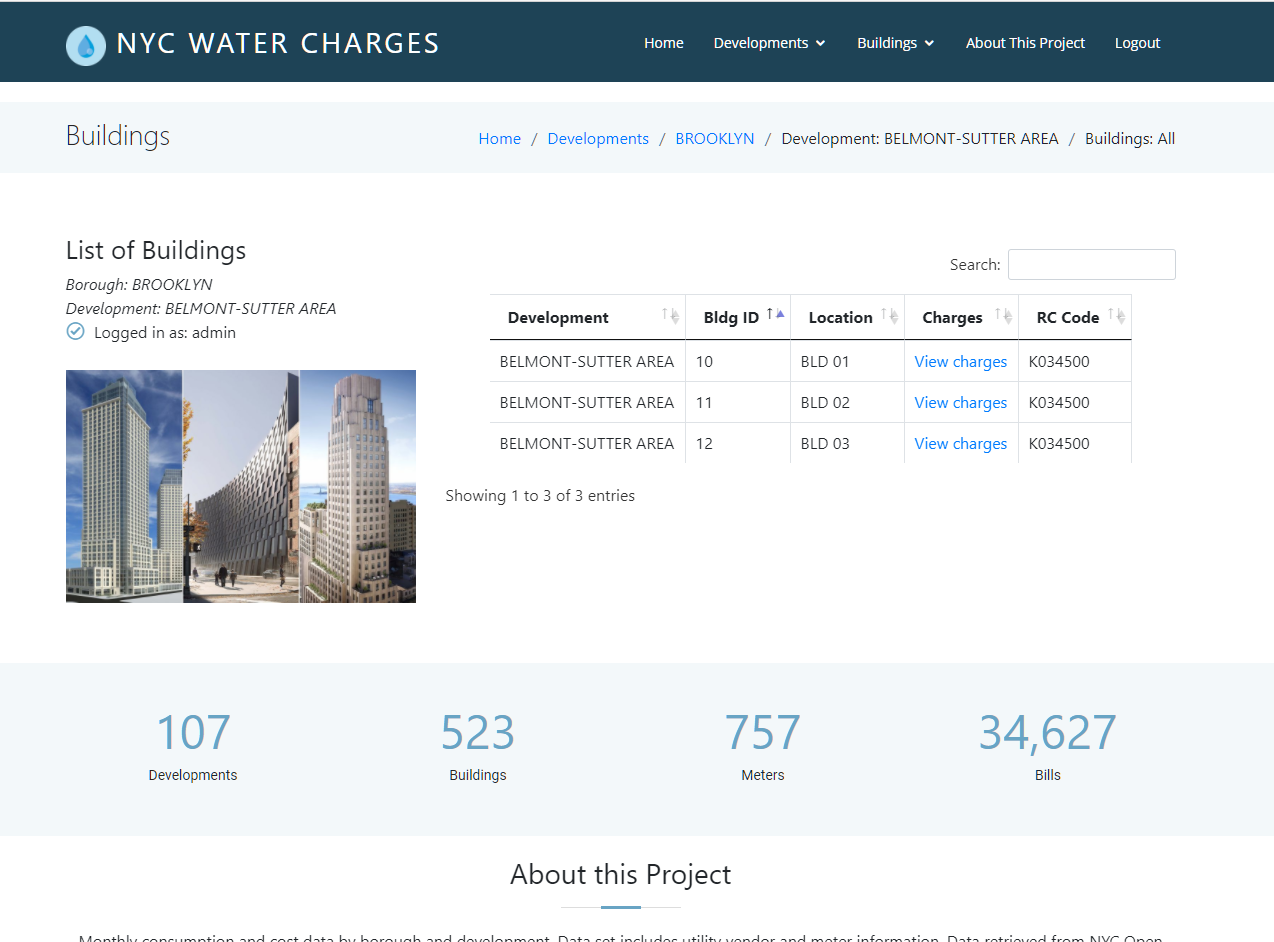
****

1. Click on a development to see a list of buildings owned by that development.

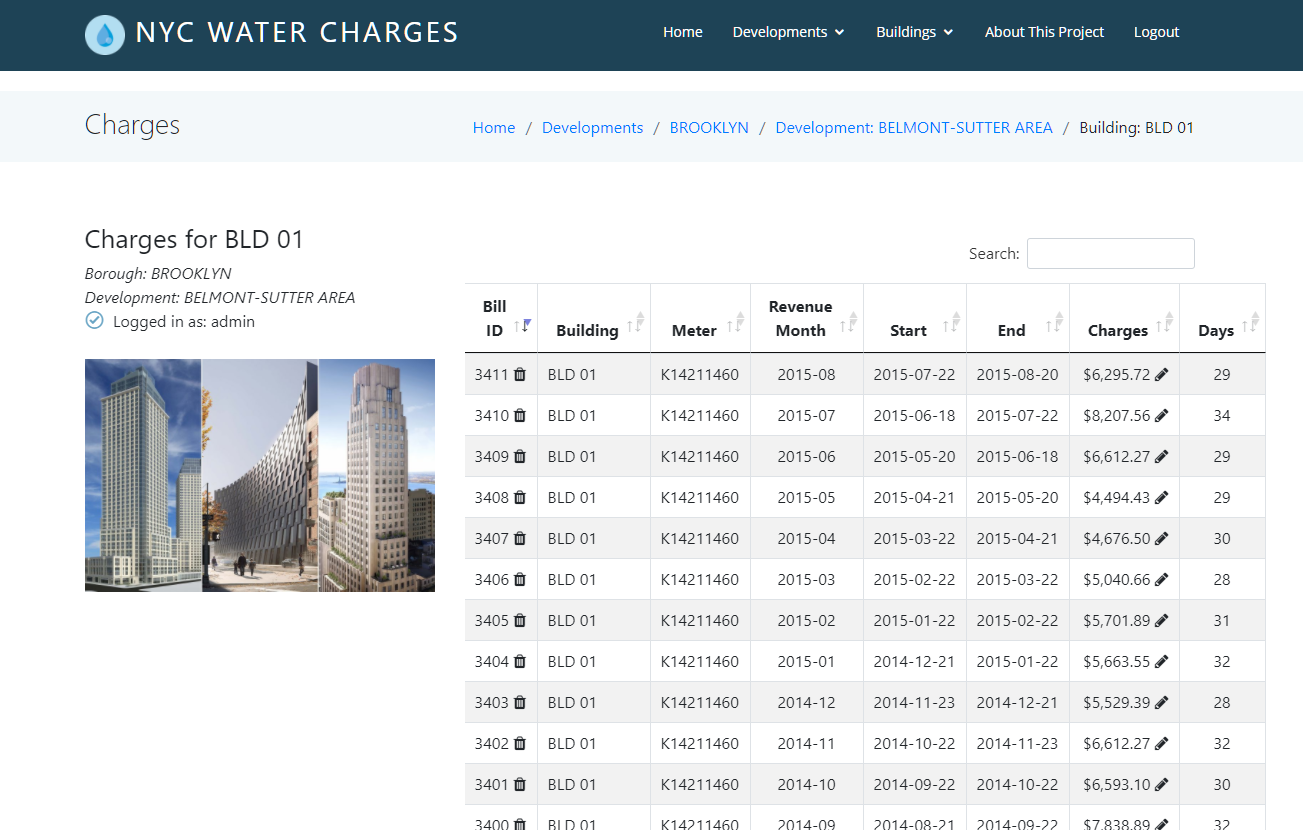
****

View Charges (NYCHA User)

1. When vewing a list of buildings, click on **View charges** to view charges associated with that building.

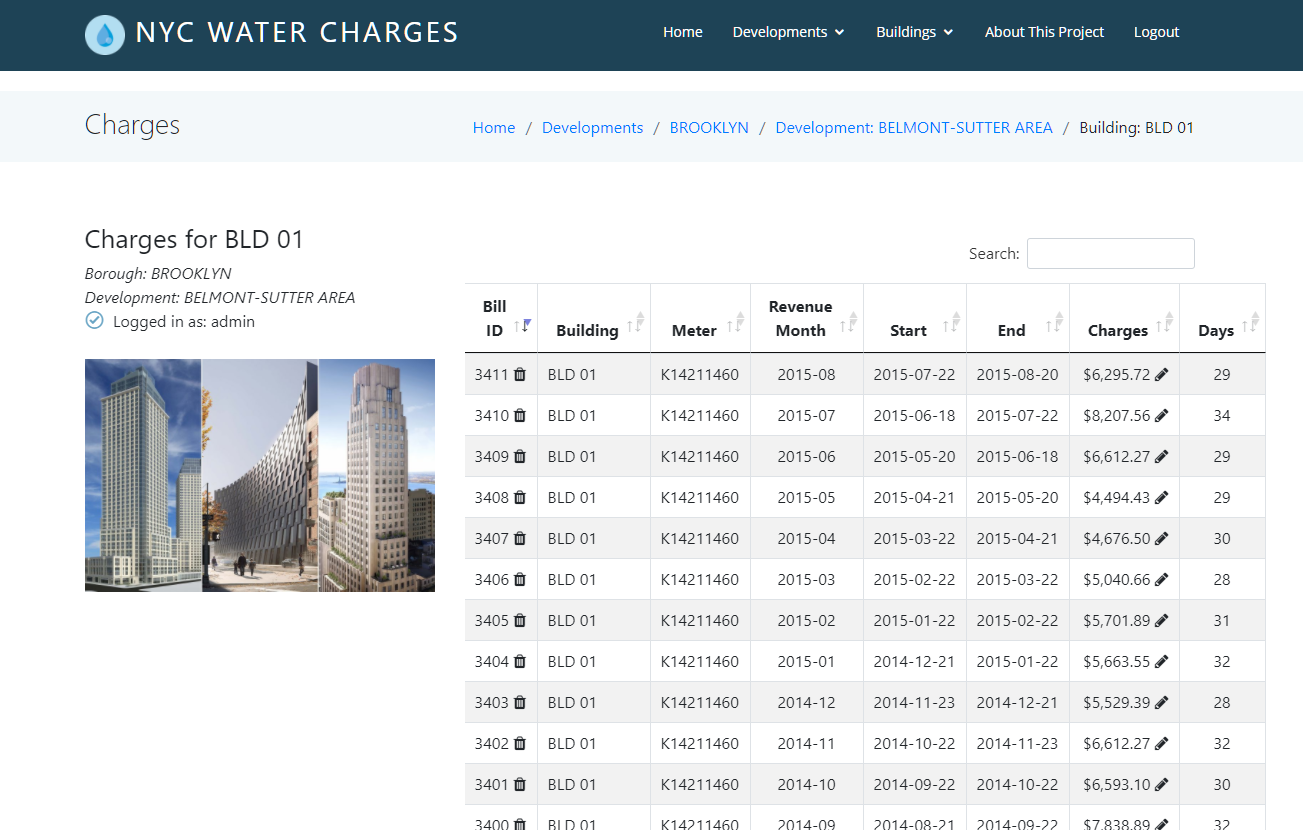
****

1. As a NYCHA/admin user, you are able to **delete** a bill, or **edit** a charge.

****

Delete a Bill (NYCHA User)

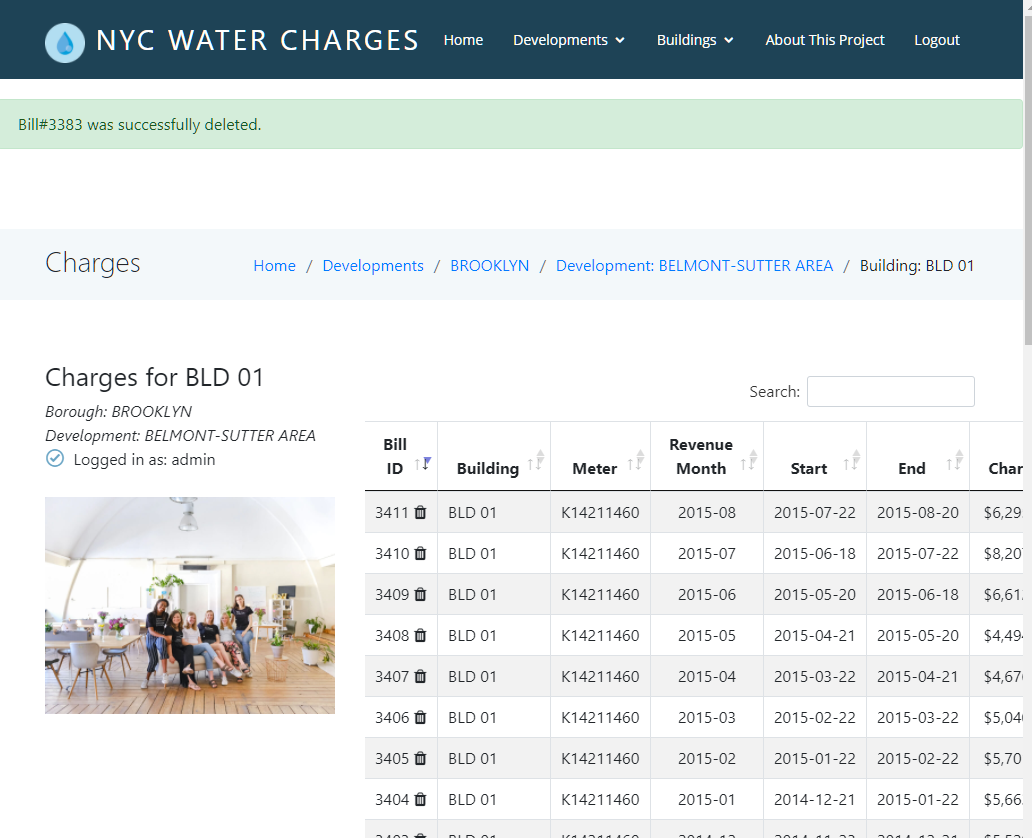
1. After viewing the list of charges, click on the Delete symbol next to the Bill ID.

****

1. Press OK to confirm that you wish to delete, or cancel if you wish to cancel.

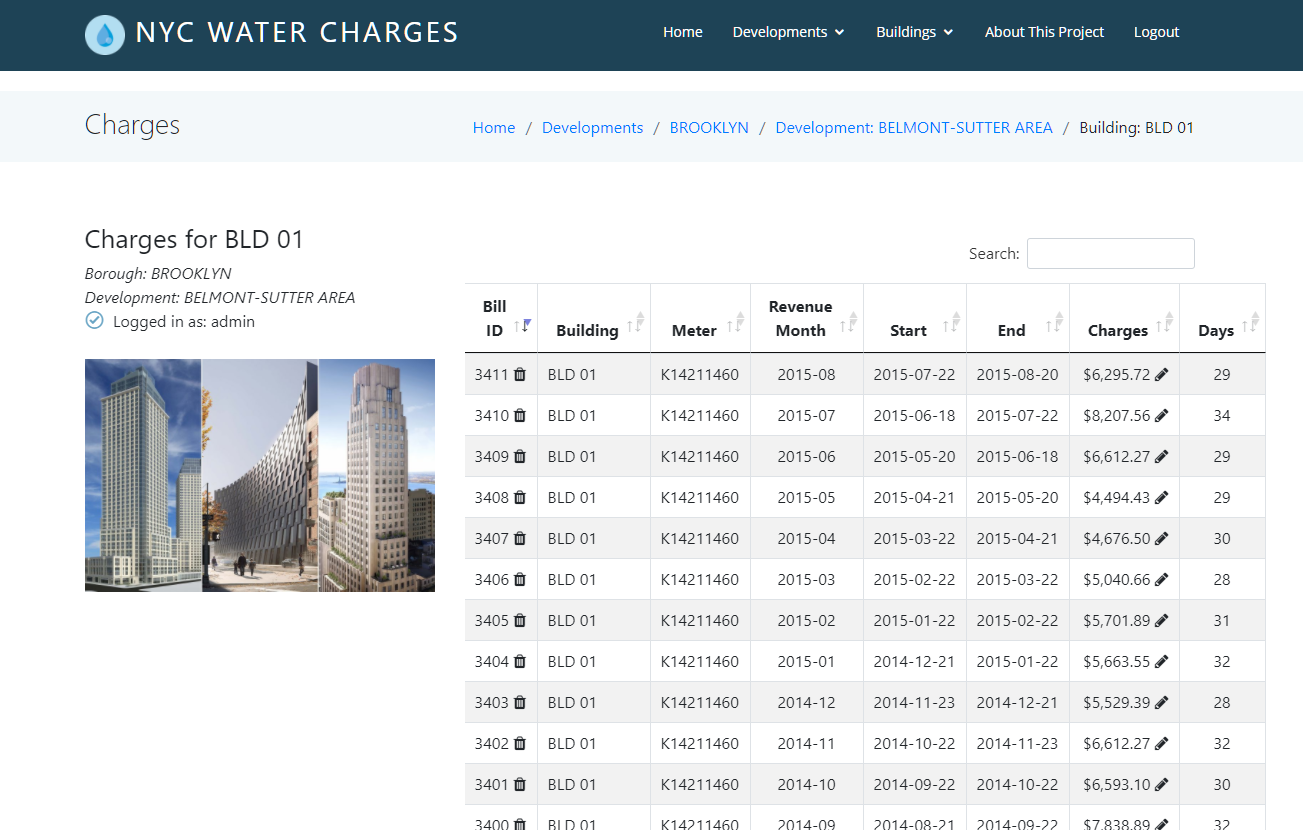
****

1. A message will pop up on your screen, notifying you that the bill was deleted. That bill will no longer appear in the list of charges.

****

Edit a Charge (NYCHA User)

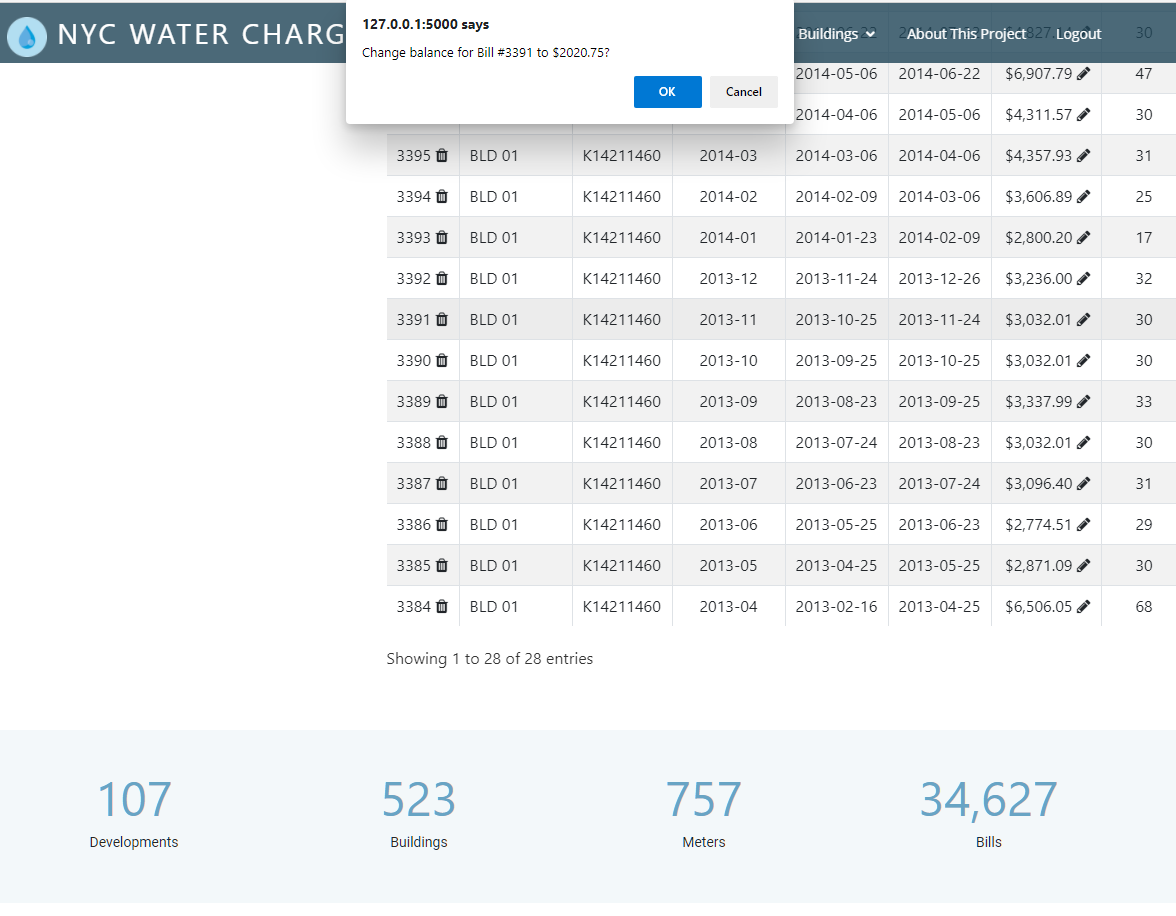
1. Click on the Edit symbol next to the charge to edit the charge.



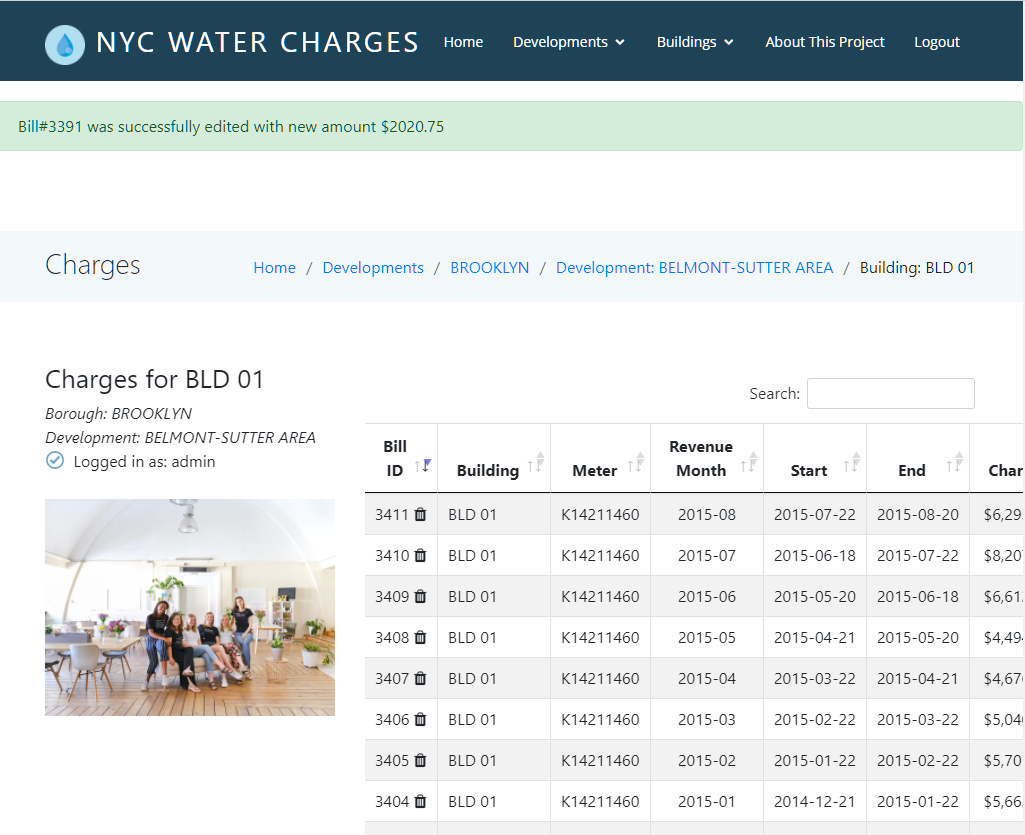
1. Enter the amount of the new charge.

****

1. Press OK to confirm, or cancel if you change your mind.



1. After confirming, a message will pop up on the screen, confirming that the amount was updated.



Log In (Dev/Building Users)

For demonstration purposes, I created some development accounts. Feel free to skip the Sign In section and log in with one of the following accounts:

Username: bay4

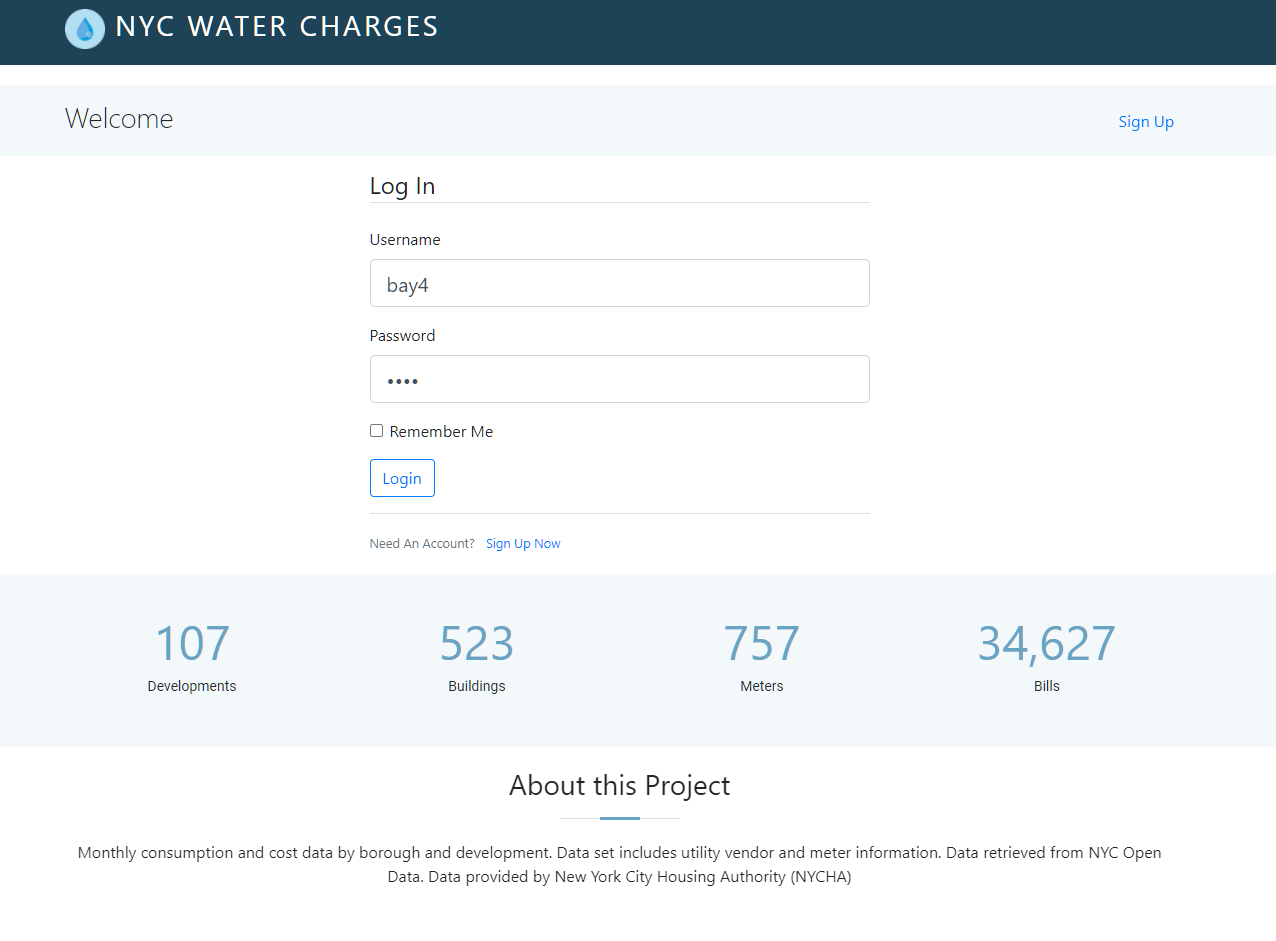
Password: bay4

Username: belmont5

Password: belmont5

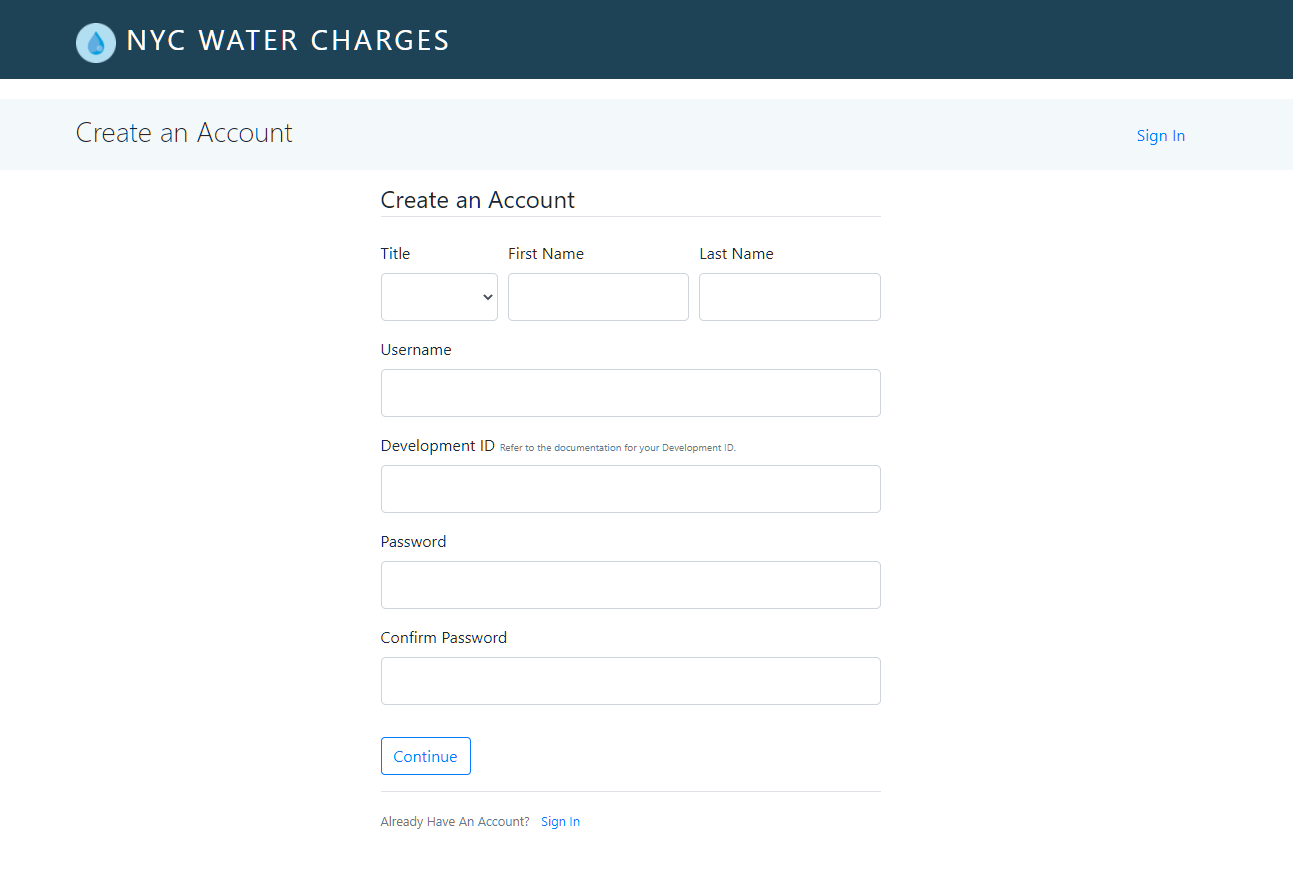
Username: berry6

Password: berry6

****

Sign Up (Dev/Building Users)

Note that there is only 1 admin account, which has already been created. Any user signing up will be signing up as a development/building manager and will have to use the associated development id.

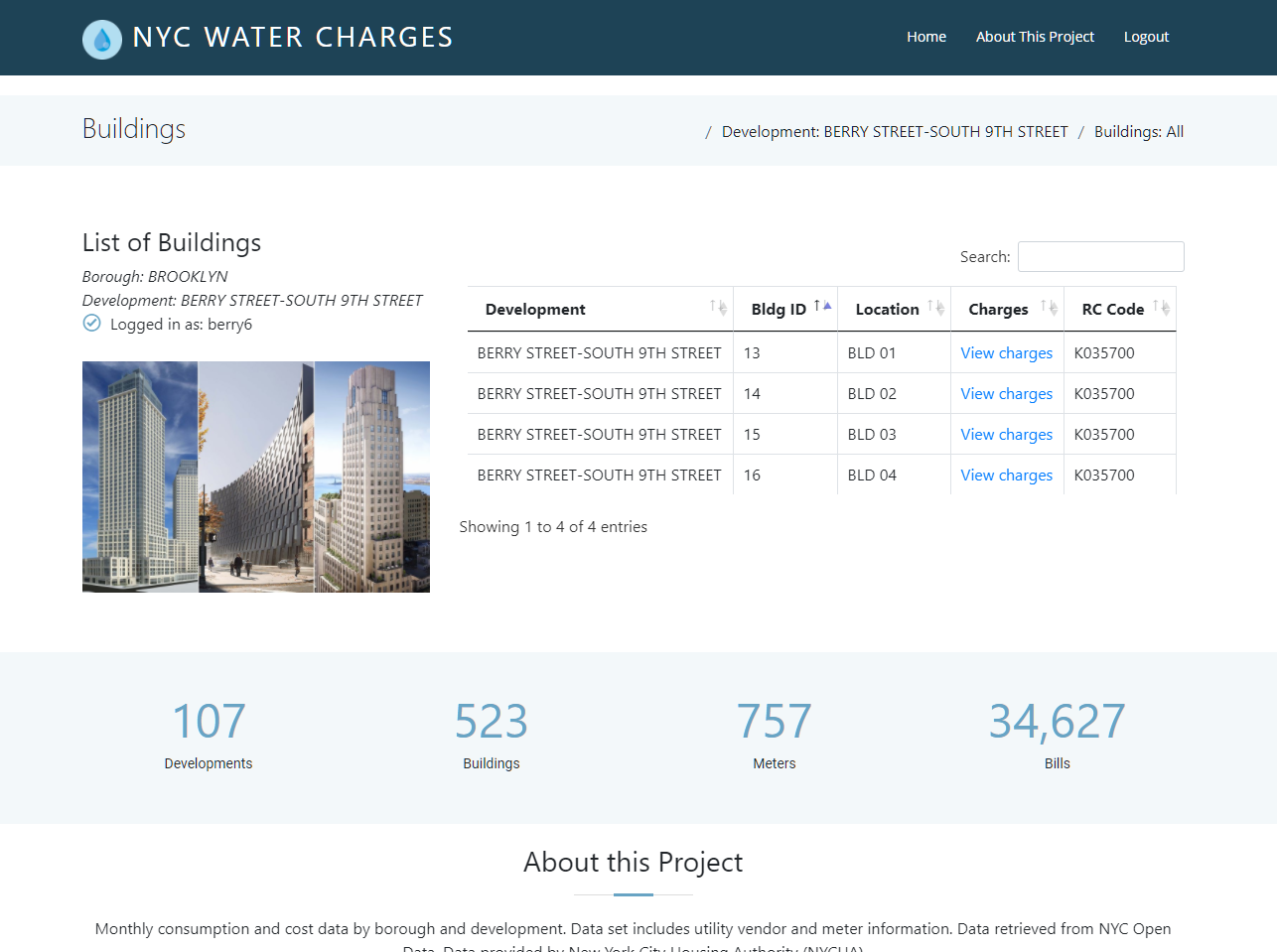
  
  
To create an account for a specific development, create a username and password and choose the associated development id. Only numbers 1-127 can be used. Refer to list of developments and development id’s below:

List of Developments and Development IDs

|  |  |  |
| --- | --- | --- |
| Dev\_id | Name | Borough |
| 1 | 154 WEST 84TH STREET | MANHATTAN |
| 2 | ARMSTRONG I | BROOKLYN |
| 3 | BAISLEY PARK | QUEENS |
| 4 | BAY VIEW | BROOKLYN |
| 5 | BELMONT-SUTTER AREA | BROOKLYN |
| 6 | BERRY STREET-SOUTH 9TH STREET | BROOKLYN |
| 7 | BETANCES III, 9A | BRONX |
| 8 | BETANCES VI | BRONX |
| 9 | BOULEVARD | BROOKLYN |
| 10 | BOYNTON AVENUE REHAB | BRONX |
| 11 | BRYANT AVENUE-EAST 174TH STREET | BRONX |
| 12 | BUSHWICK II (GROUPS A & C) | BROOKLYN |
| 13 | BUSHWICK II (GROUPS B & D) | BROOKLYN |
| 14 | BUSHWICK II CDA (GROUP E) | BROOKLYN |
| 15 | CLAREMONT REHAB (GROUP 2) | BRONX |
| 16 | CLAREMONT REHAB (GROUP 4) | BRONX |
| 17 | CLINTON | MANHATTAN |
| 18 | CONLON LIHFE TOWER | QUEENS |
| 19 | CROWN HEIGHTS | BROOKLYN |
| 20 | EAST 004TH STREET REHAB | MANHATTAN |
| 21 | EAST 152ND STREET-COURTLANDT AVENUE | BRONX |
| 22 | EAST 165TH STREET-BRYANT AVENUE | BRONX |
| 23 | EAST 173RD STREET-VYSE AVENUE | BRONX |
| 24 | EAST NEW YORK CITY LINE | BROOKLYN |
| 25 | FENIMORE-LEFFERTS | BROOKLYN |
| 26 | FHA REPOSSESSED HOUSES (GROUP I) | FHA |
| 27 | FHA REPOSSESSED HOUSES (GROUP II) | FHA |
| 28 | FHA REPOSSESSED HOUSES (GROUP III) | FHA |
| 29 | FHA REPOSSESSED HOUSES (GROUP IV) | FHA |
| 30 | FHA REPOSSESSED HOUSES (GROUP IX) | FHA |
| 31 | FHA REPOSSESSED HOUSES (GROUP V) | FHA |
| 32 | FHA REPOSSESSED HOUSES (GROUP V) | FHA |
| 33 | FHA REPOSSESSED HOUSES (GROUP VI) | FHA |
| 34 | FHA REPOSSESSED HOUSES (GROUP VII) | FHA |
| 35 | FHA REPOSSESSED HOUSES (GROUP VII) | FHA |
| 36 | FHA REPOSSESSED HOUSES (GROUP VIII) | FHA |
| 37 | FHA REPOSSESSED HOUSES (GROUP X) | FHA |
| 38 | FHA REPOSSESSED HOUSES (GROUP X) | FHA |
| 39 | FIRST HOUSES | MANHATTAN |
| 40 | FORT WASHINGTON AVENUE REHAB | MANHATTAN |
| 41 | FRANKLIN AVENUE I CONVENTIONAL | BRONX |
| 42 | FRANKLIN AVENUE II CONVENTIONAL | BRONX |
| 43 | FRANKLIN AVENUE III CONVENTIONAL | BRONX |
| 44 | GOMPERS | MANHATTAN |
| 45 | HARRISON AVENUE REHAB (GROUP B) | BRONX |
| 46 | HIGHBRIDGE REHABS (ANDERSON AVENUE) | BRONX |
| 47 | HOWARD AVENUE | BROOKLYN |
| 48 | HOWARD AVENUE-PARK PLACE | BROOKLYN |
| 49 | HUNTS POINT AVENUE REHAB | BRONX |
| 50 | INGERSOLL | BROOKLYN |
| 51 | INGERSOLL | BROOKLYN |
| 52 | INTERNATIONAL TOWER | QUEENS |
| 53 | LONGFELLOW AVENUE REHAB | BRONX |
| 54 | LOWER EAST SIDE I INFILL | MANHATTAN |
| 55 | LOWER EAST SIDE II | MANHATTAN |
| 56 | LOWER EAST SIDE III | MANHATTAN |
| 57 | LOWER EAST SIDE REHAB (GROUP 5) | MANHATTAN |
| 58 | MANHATTANVILLE REHAB (GROUP 2) | MANHATTAN |
| 59 | MARCY AVENUE-GREENE AVENUE SITE B | BROOKLYN |
| 60 | MARLBORO | BROOKLYN |
| 61 | MELROSE | BRONX |
| 62 | MORRISANIA AIR RIGHTS | BRONX |
| 63 | NDF - CENTRAL MAINTENANCE SHOP, 23 ASH ST | NON DEVELOPMENT FACILITY |
| 64 | NOSTRAND | BROOKLYN |
| 65 | OCEAN BAY APARTMENTS (OCEANSIDE) | QUEENS |
| 66 | OCEAN HILL-BROWNSVILLE | BROOKLYN |
| 67 | PARK ROCK REHAB | BROOKLYN |
| 68 | POLO GROUNDS TOWERS | MANHATTAN |
| 69 | QUEENSBRIDGE NORTH | QUEENS |
| 70 | QUEENSBRIDGE SOUTH | QUEENS |
| 71 | RALPH AVENUE REHAB | BROOKLYN |
| 72 | RANGEL | MANHATTAN |
| 73 | RED HOOK EAST | BROOKLYN |
| 74 | REHAB PROGRAM (COLLEGE POINT) | QUEENS |
| 75 | REHAB PROGRAM (TAFT REHABS) | MANHATTAN |
| 76 | REID APARTMENTS | BROOKLYN |
| 77 | SAMUEL (CITY) | MANHATTAN |
| 78 | SAMUEL (MHOP) I | MANHATTAN |
| 79 | SAMUEL (MHOP) II | MANHATTAN |
| 80 | SHELTON HOUSE | QUEENS |
| 81 | SMITH | MANHATTAN |
| 82 | SOTOMAYOR HOUSES | BRONX |
| 83 | SOUTH BRONX AREA (SITE 402) | BRONX |
| 84 | SOUTH JAMAICA I | QUEENS |
| 85 | SOUTH JAMAICA II | QUEENS |
| 86 | STANTON STREET | MANHATTAN |
| 87 | STAPLETON | STATEN ISLAND |
| 88 | STEBBINS AVENUE-HEWITT PLACE | BRONX |
| 89 | STERLING PLACE REHABS (SAINT JOHNS-STERLING) | BROOKLYN |
| 90 | STERLING PLACE REHABS (STERLING-BUFFALO) | BROOKLYN |
| 91 | STUYVESANT GARDENS II | BROOKLYN |
| 92 | TAFT | MANHATTAN |
| 93 | TAPSCOTT STREET REHAB | BROOKLYN |
| 94 | TAPSCOTT STREET REHAB | FHA |
| 95 | THOMAS APARTMENTS | MANHATTAN |
| 96 | UNION AVENUE-EAST 166TH STREET | BRONX |
| 97 | UPACA (SITE 5) | MANHATTAN |
| 98 | UPACA (SITE 6) | MANHATTAN |
| 99 | VAN DYKE I | BROOKLYN |
| 100 | WASHINGTON HEIGHTS REHAB (GROUPS 1&2) | MANHATTAN |
| 101 | WASHINGTON HEIGHTS REHAB PHASE III | MANHATTAN |
| 102 | WASHINGTON HEIGHTS REHAB PHASE IV (C) | MANHATTAN |
| 103 | WASHINGTON HEIGHTS REHAB PHASE IV (D) | MANHATTAN |
| 104 | WEST FARMS ROAD REHAB | BRONX |
| 105 | WEST FARMS SQUARE CONVENTIONAL | BRONX |
| 106 | WHITMAN | BROOKLYN |
| 107 | WILLIAMSBURG | BROOKLYN |

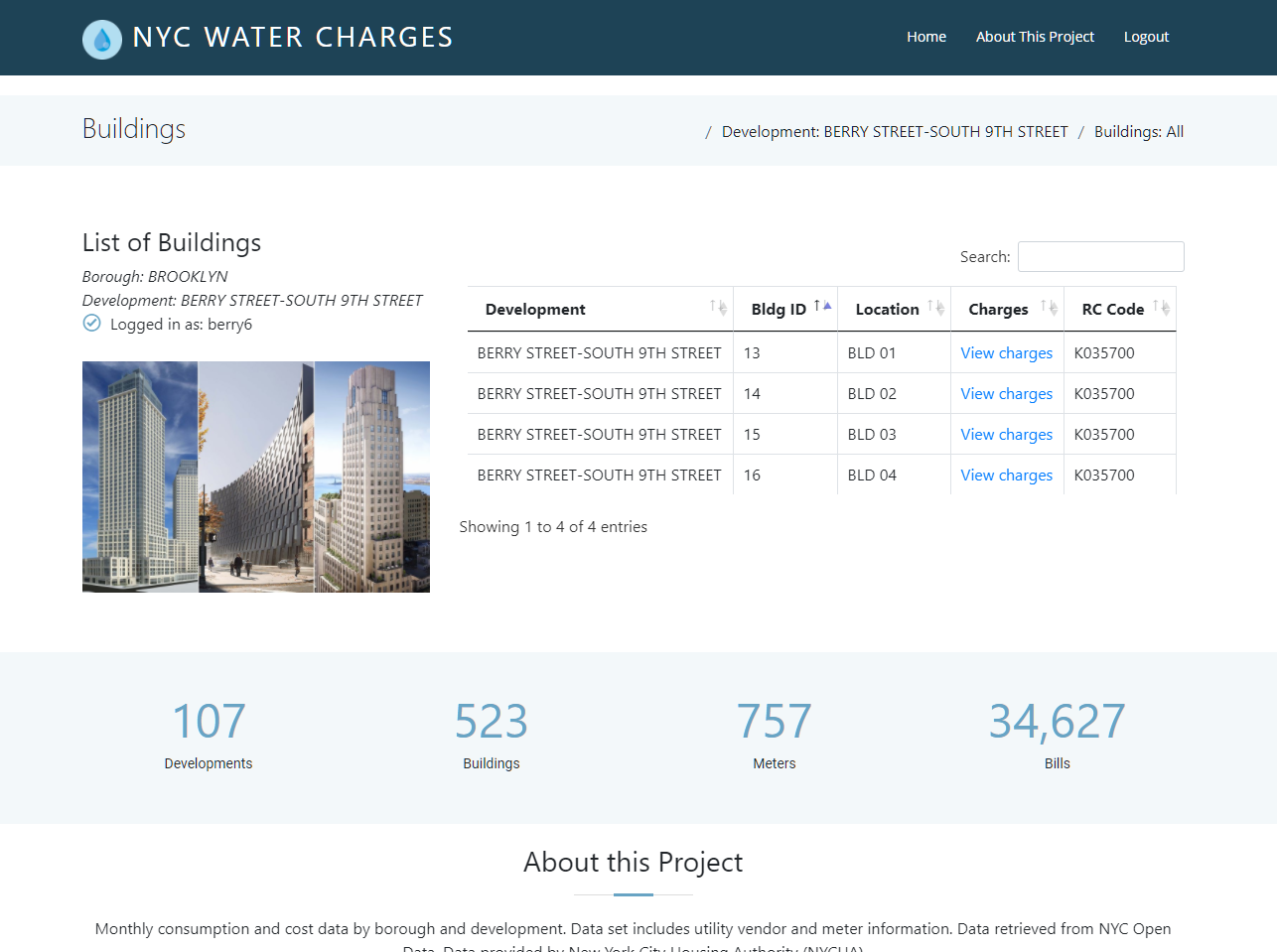
View Your Buildings (Dev/Building Users)

After you log in, you will automatically be redirected to your list of buildings. Note that as a dev/building user, you do not have the option to view other developments or other buildings.

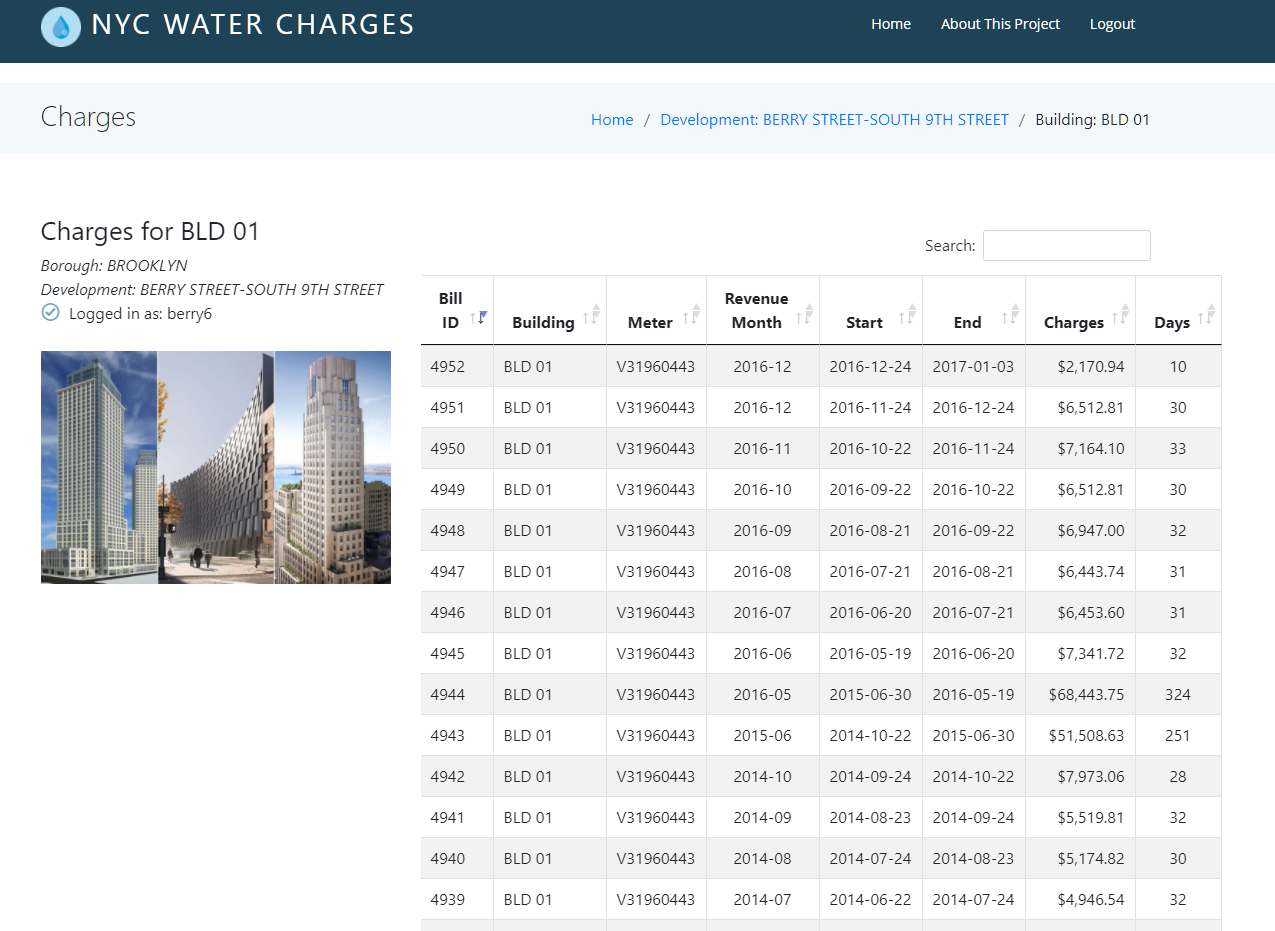
****

View Your Bills (Dev/Building Users)

1. To view the charges associated with that building, click on View charges. Note that as a user, you do NOT have the option to delete or edit a bill.

****

1. You will be presented with a list of bills associated with that building.

****

CRUD Implementation

This software incorporates all the CRUD functionality associated with a typical database application. Below is how these functionalities were implemented.

**Create**: This is implemented by allowing the creation of new accounts. A new user is added to the Users database table. The process is initiated when a user signs up to create a new account.

**Read**: This is implemented by querying the database to display all relevant information. The data is displayed and formatted using Boostrap datatables and jQuery.

**Update**: This is implemented by allowing NYCHA users to edit the charges associated with a specific bill. This process is triggered when a user with an admin account clicks on the “Edit” icon next to a bill. This feature is only available to admin users and will not appear when logged in as a building user.

**Delete**: This is implemented by allowing NYCHA users to delete a bill. This process is triggered when a user with an admin account clicks on the “Delete” icon next to the bill id. This feature is only available to admin users and will not appear when logged in as a building owner.

Technical Documentation

The technology stack is as follows:

The business layer was created in Python. I used Flask as the web framework and SQLAlchemy as the ORM. In addition, I used flask-login to implement user authentication and session management.

The data was provided by [NYC OpenData](https://data.cityofnewyork.us/Housing-Development/Water-Consumption-And-Cost-2013-2020-/66be-66yr) in the form of a csv file with 42.6K rows and 25 columns. I normalized the data and then stored it in a Postgres database, hosted on Heroku.

The Front End was created using HTML, CSS, Javascript, Bootstrap and some jQuery.

Both the database and the application are hosted on Heroku.

Schemas and ER Diagram

The following tables and attributes (primary key, *foreign key*) are included in this database:

* **Development** (dev\_id, name*, borough\_id*, tds, edp, amp, funding)
* **Borough** (borough\_id, name)
* **Building** (building\_id, *dev\_id*, location, rc\_code)
* **Meter** (meter\_id, type, scope, meter\_number)
* **Service** (service\_id, *vendor\_id*, revenue\_month, start\_date, end\_date, days, consumption)
* **Cost** (cost\_id, *rate\_id*, current\_charges, water\_sewer\_charge, other\_charges)
* **Rate** (rate\_id, class)
* **Bill** (bill\_id, *building\_id, meter\_id, service\_id,* *cost\_id*, umis\_bill\_id, estimated*,* bill\_analzyed)
* **User**(username, *dev\_id*, password)
* **Transaction**(trans\_id, *user\_id, bill\_id,* trans\_date, amount\_paid, balance)

**Diagram

Description automatically generated**

Extra Credit: Hosting on Heroku

Both the database and the web application are hosted on Heroku.

How to host a **web application** with a Python backend on Heroku:

1. In your project directory, run pip freeze > requirements.txt to gather all the Python dependencies used in the project.
2. Create a file named Procfile and add the following content to ensure that Heroku knows what commands it needs to run the application: web: gunicorn run:app
3. Create a new repository in Github to upload all your necessary files for the application.
4. Initialize your current directory and the remote branch by running the following git commands:
   * git init
   * git remote add origin addressgoeshere
5. Use the following commands to add, then commit, then push your files to your repository:
   * git add .
   * git commit -m “Message goes here”
   * git push origin master
6. Create a new application on Heroku.
7. Click on Deploy, and for the deployment method, choose Github.
8. Search for the repository created in the previous steps.
9. Run heroku git:remote -a yourapplication to add the Heroku remote to your local repository
10. Now you can deploy to Heroku by simply running git add, git commit, then git push heroku master.

How to provision the **database** on Heroku:

1. In the Overview section of the relevant Heroku app, click on Configure Add-ons then search for Postgres.
2. Select the Heroku Postgres add-on.
3. Click on it to edit.
4. Click on Settings 🡪 View Credentials to view all the settings needed to connect to the database.
5. Log in to the database using those settings and create all the necessary tables.
6. In Python, retrieve the database connection string by calling the appropriate environment variable.   
   For example: DB\_URL = os.environ['NYC\_WATER\_DATABASE\_URL']
7. Use that URL to configure the app in Python/Flask:  
   app.config['SQLALCHEMY\_DATABASE\_URI'] = DB\_URL
8. Your app can now communicate with your database.
9. Optional: use SQLAlchemy as your ORM to communicate with your database.

app = Flask(\_\_name\_\_)

db = SQLAlchemy(app)

engine = create\_engine(DB\_URL)