PDB Manager Application Design

DSCI-532::Final Project 3

Mark Green

7/17/2022

Contents

Introduction

Passive Diffusion Bag (PDB) sampling is becoming a more common practice for long-term groundwater monitoring sites in the environmental industry due to it's potential cost-savings for sites with volatile organic compounds (VOC) as the contaminants-of-concern (COC). However, current methods (manually updated excel spreadsheets…) for managing PDB deployments and sampling can be cumbersome for operations teams. The *PDB Manager* web application aims to solve this issue by providing an user interface for recording PDB deployment/sampling information and efficiently tracking and querying these data from a database.

Previously, the database design was discussed in detail from the transformation of the original dataset to the development of SQL statements for performing CREATE, READ, and UPDATE operations on the database in accordance with the PDB deployment/sampling model. This report was comprehensive for the most critical functionality of the web application, although a couple other features are planned including:

- Analyzing existing database to determine which wells currently have PDBs deployed to them, and applying the `deployPDB()` function as necessary to accurately reflect this in the database.
- Adding functionality to existing queries to allow filtering on wells with/without PDBs currently deployed.
- Adding feature to allow assignment of lot numbers in batch by date range.
  - additional table for lot number assignment by date range
  - modify read queries to join lot numbers

The optional features may or may not be added by the end of this project. In either case, they will not be documented further in this report.

This report discusses the web application architecture and layout for the *PDB Manager* application in the sections below.

Section 1: Web Application Architecture

MVC Design

> A typical web application follows a MVC schema.
>
> - **Model**: The backend that contains all the data logic
> - **View**: The frontend or graphical user interface (GUI)
> - **Controller**: The brains of the application that controls how data is displayed
>
> -Final Project part 3 description

The *PDB Manager* architecture follows the MVC design as specified above. In this case, the **Model** is a SQLite database. This is where the database is stored, and SQL is used to Create, Read, and Update data as necessary.

Next, the **Controller** is the RShiny application *server* script. The server script handles the interactions between R input variables and the SQLite database using the `RSQLite` and `DBI` packages.

Finally, the **Viewer** is the RShiny application *ui* script. The ui script routes maps the user interface to input variables for the server, and renders messages and objects (tables) as specified by the server. The ui script also governs the overall front-end layout and appearance of the user interface, and incorporates other web elements (css, html, images, etc.) as necessary.

A diagram of the application architecture is shown below:

Access and Interactivity

These three MVC components, and any supporting objects, are hosted for free on RShiny Server using https://shinyapps.io/. Access to the application will be provided by the `shinyauthr` package. When the user provides an acceptable username/password then access to the application functions will be provided. The general interactivity for the user will include the ability to input and select data, as well as view tables.

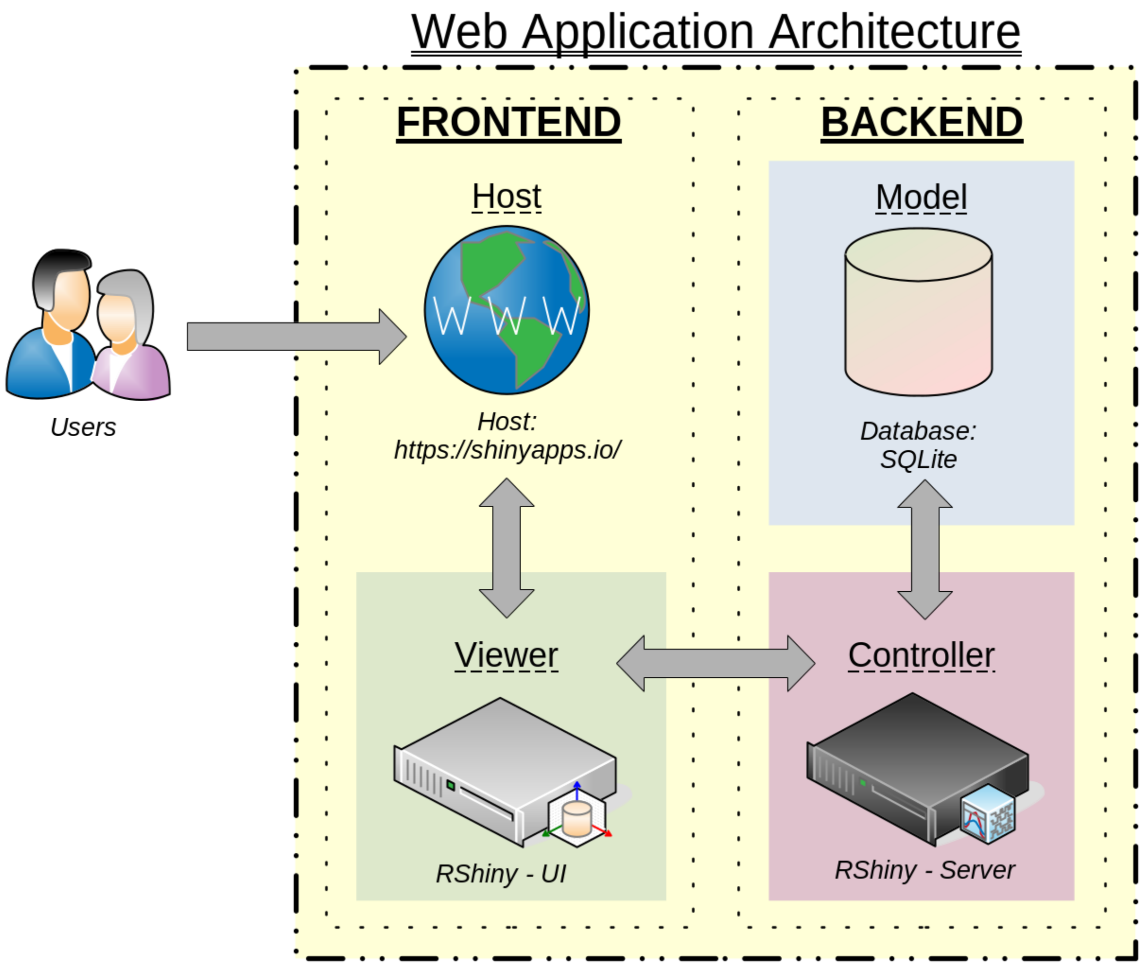
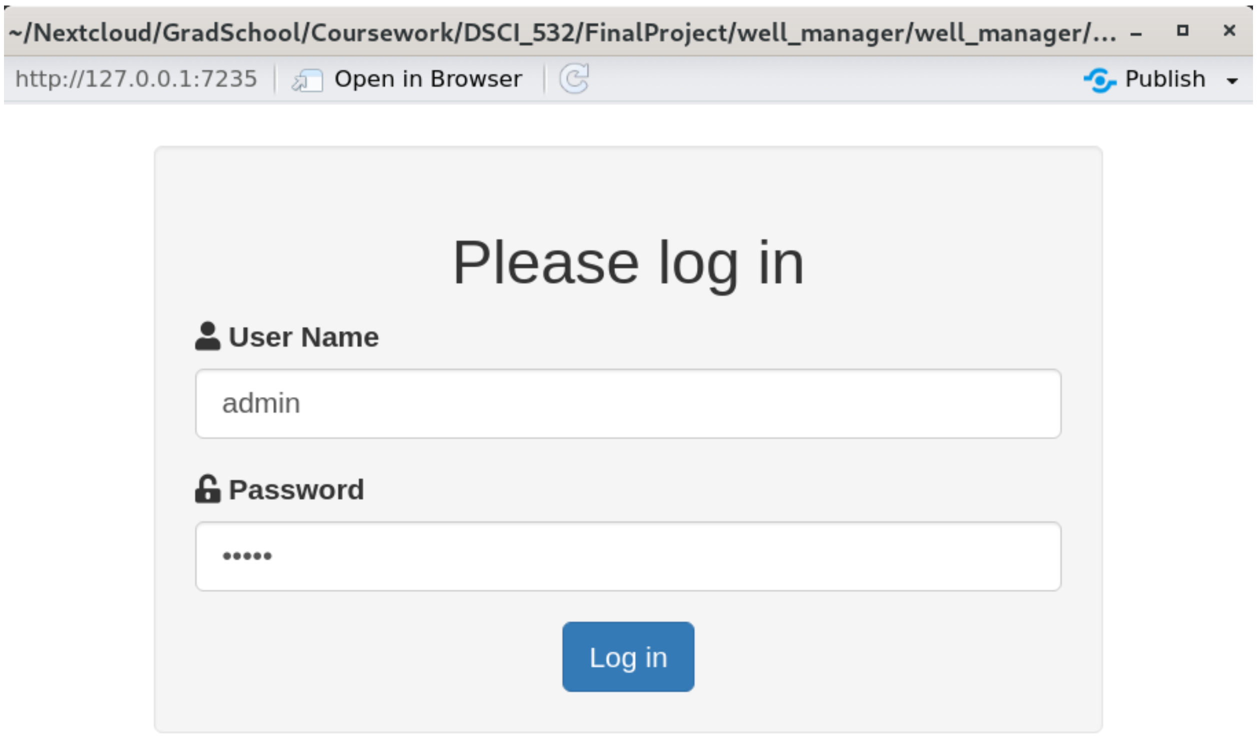


Figure 1: PDB-Manager web application architecture with MVC designation

Section 2: Web Application Layout

The *PDB-Manager* application is designed to be an internally facing enterprise application. Thus, the initial layout given will be a user login screen. No other elements will be displayed until correct credentials are provided as part of the enterprise security protocol.


~/Nextcloud/GradSchool/Coursework/DSCI_532/FinalProject/well_manager/well_manager/...
http://127.0.0.1:7235
Open in Browser
Publish

Please log in

User Name

admin

Password

•••••

Log in

Figure 2: Login page

Once authenticated, the user is taken to an initial homepage. The content of this homepage will be some basic information about the using the application, and external links. A logout button exists for when a user completes their session.

The various menu options will be available as tabs on a navigation panel at the top of the screen. In addition to the homepage, there are the following tabs:

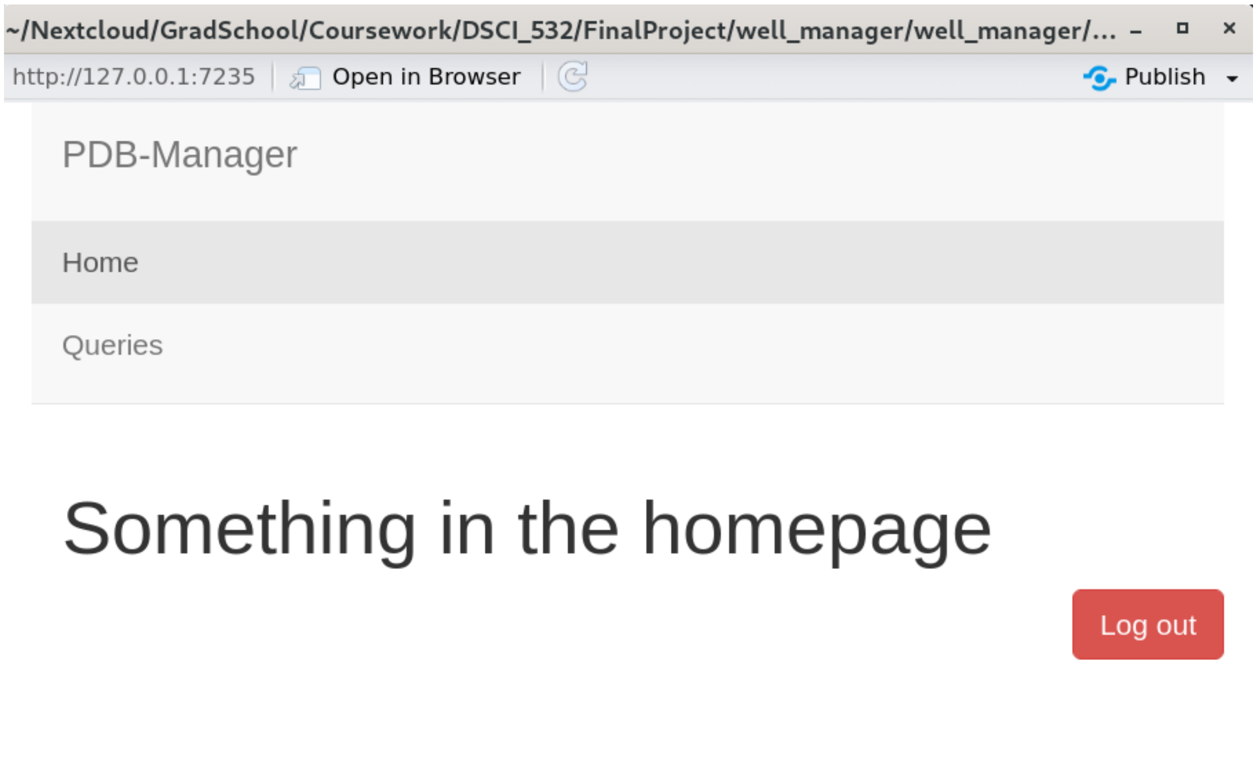
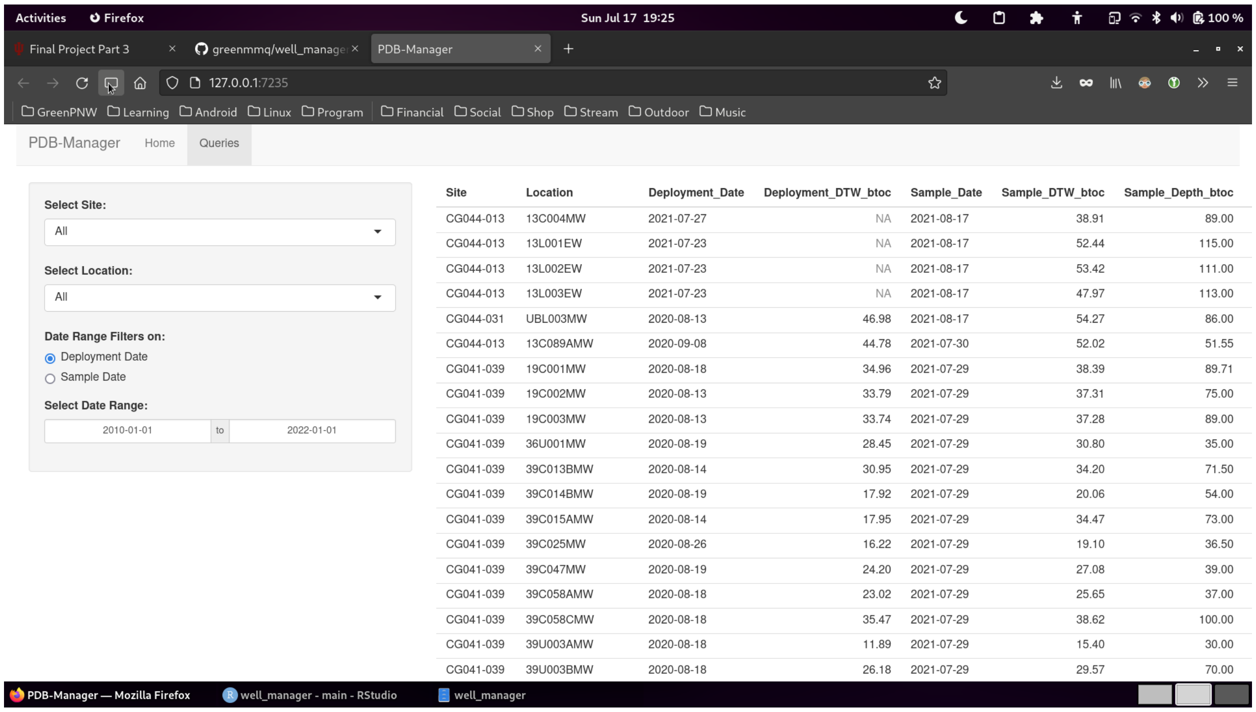

~/Nextcloud/GradSchool/Coursework/DSCI_532/FinalProject/well_manager/well_manager/...
http://127.0.0.1:7235
Open in Browser
Publish
PDB-Manager
Home
Queries
Something in the homepage
Log out

Figure 3: Homepage



Site	Location	Deployment_Date	Deployment_DTW_btoc	Sample_Date	Sample_DTW_btoc	Sample_Depth_btoc
CG044-013	13C004MW	2021-07-27	NA	2021-08-17	38.91	89.00
CG044-013	13L001EW	2021-07-23	NA	2021-08-17	52.44	115.00
CG044-013	13L002EW	2021-07-23	NA	2021-08-17	53.42	111.00
CG044-013	13L003EW	2021-07-23	NA	2021-08-17	47.97	113.00
CG044-031	UBL003MW	2020-08-13	46.98	2021-08-17	54.27	86.00
CG044-013	13C089AMW	2020-09-08	44.78	2021-07-30	52.02	51.55
CG041-039	19C001MW	2020-08-18	34.96	2021-07-29	38.39	89.71
CG041-039	19C002MW	2020-08-13	33.79	2021-07-29	37.31	75.00
CG041-039	19C003MW	2020-08-13	33.74	2021-07-29	37.28	89.00
CG041-039	36U001MW	2020-08-19	28.45	2021-07-29	30.80	35.00
CG041-039	39C013BMW	2020-08-14	30.95	2021-07-29	34.20	71.50
CG041-039	39C014BMW	2020-08-19	17.92	2021-07-29	20.06	54.00
CG041-039	39C015AMW	2020-08-14	17.95	2021-07-29	34.47	73.00
CG041-039	39C025MW	2020-08-26	16.22	2021-07-29	19.10	36.50
CG041-039	39C047MW	2020-08-19	24.20	2021-07-29	27.08	39.00
CG041-039	39C058AMW	2020-08-18	23.02	2021-07-29	25.65	37.00
CG041-039	39C058CMW	2020-08-18	35.47	2021-07-29	38.62	100.00
CG041-039	39U003AMW	2020-08-18	11.89	2021-07-29	15.40	30.00
CG041-039	39U003BMW	2020-08-18	26.18	2021-07-29	29.57	70.00

Figure 4: Explore (currently labeled ‘Queries’)

1. `Explore`
   - The Explore tab allows users to run read queries pdb deployment and sampling data.
   - A sidebar panel provides inputs to filter the data, and a table displaying the results is rendered on the main panel.
     - `TODO` make the table interactive
     - `TODO` Add ‘filter on PDB in well’ feature
2. `Update`
   - The Update tab allows users to input data. Two options exist for standard users:
     a. deploy_pdb() function
     b. collect_sample() function
   - This tab will consist of a conditional reactive user interface dialog which will output a confirmation message or appropriate error message to the user, as applicable
   - No screenshot as this is not yet developed
3. `Admin`
   - The Admin tab will allow for some admin functionalities, like provisioning user accounts, and assigning lot numbers.
   - `TODO` - build lot number assignment funcionality.

The overall aesthetics of the application layout are very simple. Default colors, buttons, and functions from the RShiny application are good enough for the simplicity of the application. Drop-down buttons, radio buttons, daterange selectors are all interactive features of this application.

Section 3: Work Assessment

Thus far I am moderately satisfied with my task completion - 7/10. Right now the time commitment for this project is relatively large, over 12 hours per project phase. There has been an immense learning curve with RShiny, but I think I am starting to get the hang of it.

There is a lot of functionality I would like to add to this application, but at this point I am setting out to finish the minimum viable product (MVP) from the original proposal. It would be nice to make this application more full featured. With more time, it would also be nice to add some theming and some more interactivity. Currently the application is pretty static, and not very flashy. Though it does get the job done.

Check out my progress so far on github!