IFB299 Artifacts List

Sebastian Sherry n9161350

# Release 1:

## Admin.html and admin.js:

Admin.html and admin.js come together to form the admin portal for the project. The html file is very basic, as most of the page is generated through javascript. The page is the basic skeleton table and a button leading to a page to add violations and heath issues.

The javascript uses Jquery’s getJson method to access the database using specially made urls that access the database. This data is then used to populate the tables. Each table is generated when selected from a radio button in the html.

## db.py and dbTest.py:

Db.py implements the database connector for the project. Each table has their own insert and update function, while the retrieve and delete functions are universal for all tables. The connector is implemented as class for easy use in lookup.py. The file also contains a main function used to input the initial test values into the database. This function is usually run to reset the database for examples or when new tables/values are added.

DbTest.py is a script written to test the initial database implementation to prove all the connector functions work. Theses tests do not prove that the database will work in a production environment, as they lack the required structure to but unit tests. This script fills more of a proof of concept role, with better tests to be written later in the development cycle.

## Lookup.py:

Lookup.py is the bridge between views and the database connector. It contains various methods to perform back end data manipulation and retrieval. The included functions include (but are not limited to): Login credential checking, retrieving parking permits and all violations, and adding permits and violations to the database. All functions requiring database access have their own database object created and closed at the end of the function to prevent issues with multiple users accessing the database at once.

## Profile.html:

Profile.html is a simple page that displays all relevant user data for any user on the database (besides admin). Using functions from lookup.py and flask’s jinja2 templating, the page is quite dynamic. For example, if the user doesn’t have a parking permit, the table displaying the permit information is replaced with text stating they don’t have a permit. Similarly, if a user doesn’t have a parking violation, the table isn’t shown.

When the user does have a violation, they can find all relevant data on this page. Each violation has a button which will take the user to payment form when implemented.

## Views.py:

Views.py is the main file used to run the project. When executed by the server, it displays and handles the website. This file defines all the pages on the site, which ones require the user to be logged in or an admin, and also passes data to and from lookup.py functions to pages where required. The file also contains two helper functions which help format data and initialize cookies.

Sessions are extensively used throughout the site. These sessions help keep track of user information such as email and if they have admin privileges. This data is useful for retrieving relevant data and access control to the admin portal.

# Release 2:

## Admin.js

Admin.js received two new features in this sprint. The first being dynamically displaying a list of users for the admin page. The second is the fine payment notification functions. These functions make calls to specific urls that send emails to users with fines. The method has to pass the citation number of the violation for the email to be correctly sent.

## Db.py

Db.py contains new tables and associated functions to handle fine payments. These tables are fine payments, which holds the citation number, type and payment status for each violation, and payment details, which holds the billing and card information for a payed fine. The main function was also updated to insert all the generated violations into fine payments, as well as mark 2 violations as payed in the payment details table.

## Edithealth.html

Edithealth.html is a form used to edit health issues from the admin page. The page pre fills all the relevant fields from information stored in the database using flask’s jinja2 templating. The page also handles empty values for the resolution date, time and description fields, which aren’t always filled depending on the status of the issue.

## Lookup.py

Lookup.py has been extended to support a whole bunch of new features of the website. Some methods such as getProfileInfo have been update to only display violations which haven’t been paid, and addViolation not adds an entry into the fine payment table also.

New helper methods were also added to produce cleaner code. These include getFine, which returns all the data required to display a fine for payment. GetFineAmount is also a new method that calculates how long overdue a fine is and outputs how much extra to charge on top of the base fine rate.

## Views.py

Views.py includes a whole bunch of new features to support the second release. The payment form is handled using a new url ‘/paymentform/<id>’, where the id is the fine ID, which is passed as variable into the python function via the url. New helper methods were also added, such as protectCardNum, which replaces the middle 8 digits of a credit card number with X’s, and getHeaderInfo, which returns various useful data for the header bar such as number of notifications and if the user is an admin or not.