PROJECT 2
Lockit

Pd 9 | Somewhat Useful
August Ray Jones, Marcus Ng, Levi Olevsky, James Smith

WHAT DOES IT DO?

Lockit offers you an easy way to buy, sell, or trade lockers. Users are able to view lockers up for sale or trade and contact the seller using the integrated Gmail API to automatically set up an email conversation. Locker offers can be sorted by price or floor. A map and description of the location will give the user a general idea of where each locker is located.

***DISCLAIMER: WE ARE NOT RESPONSIBLE FOR ANY SCAMS. SPEAK TO OUR CUSTOMER SERVICE REPRESENTATIVE**

OUR APIs

Gmail API: Acts as login account to the website and is used to send emails on behalf of the user to the seller for further communication.

THE INTERACTIVE MAP

The interactive map is a way that users can better describe where their locker is in the school since numbers aren't descriptive. You can select your floor which will then set the map to be the correct floor plan. From here you simply click the approximate location of your locker and it will stick a big red dot at that location. Those who view your post or trade request will then see that floor plan and the big red dot.

LOCKIT LIBRARY - COMPONENTS

HTML files

See 'Files' section for details

Stylesheets

Bootstrap's as well as our own

auth.py

Handles the database, including login information and locker possession

locker.py

Manages and retrieves locker information

offer.py

Manages database for selling/trading

main.py

Flask app, handles front-end navigation and operations

api.py

Handles Gmail API calls

user.db

Contains user information including user gmail OAuth token and locker ID

locker.db

Contains locker ID and the location.

offers.db

Contains offer information

FILES

template.html

login/logout button, profile button, and homepage button on a toolbar, footer with team info and “customer service”

home.html

Displays the 5 newest lockers for sale posts

Displays the 5 newest locker trade posts

signup.html

Asks for profile information for first time setup

- Name
- Email (unique)
- Password
- Locker Number (optional; can be added later)

login.html

Form for username and password

Has a link to the create account page

profile.html

Displays ‘Your Lockers’

- Clicking on a locker allows you to edit info about your locker

Add another locker

Shows number of locker trade requests

Change password

edit.html

Edit information about your locker

Delete the locker

offers.html

Accept or deny on the page
Shows locker offers in a grid layout
Search bar to search for specific locker numbers
Options to sort lockers by price and floor
Hovering over a locker offer expands the cell with more information

display.html

Show information about locker
Display location with map

post.html

Form

Select which one of your lockers to post(displays your owned lockers)
Trading or Selling(dropdown)
Price(appears if you select Selling)
Floor(dropdown)
Interactive map
Description

style.css

To make the website pretty

map.js

Runs the interactive map

bootstrap.min.css and bootstrap.min.js

Incorporating Bootstrap

main.py

- **add_session(username, password)** - creates a session cookie for the current user with valid credentials or returns error message
- **logout()** - Deletes a session key of a user, if one is logged in
- **root()** - redirects to /home
- **home()** - provides functionality for the /home route
- **login()** - provides functionality for the /login route
- **signup()** - provides functionality for the /create route
- **profile()** - provides functionality for the /profile route
- **edit()**-provides functionality for the /edit route
- **offers()**-provides functionality for the /edit route
- **edit()**-provides functionality for the /edit route

api.py (Gmail)

- **send_email_sell(buyer_email, seller_email)** - send email to specified email from current user

- **send_email_trade(trader1_email, trader2_email)**

auth.py

- **login(email, password)** - Logs user in if email and password are correct
- **encrypt_password(password)** - Returns SHA-256 version of password
- **create_account(name, email, password, lockers)** - Creates a new entry in .profiles table
- **does_email_exist(email)** - Returns true if email exists, false otherwise
- **get_lockers(email)** - Return string array of lockerIDs
- **is_valid_email(email)** - Parse email on '@' and return true if index 1 is "stuy.edu", false otherwise

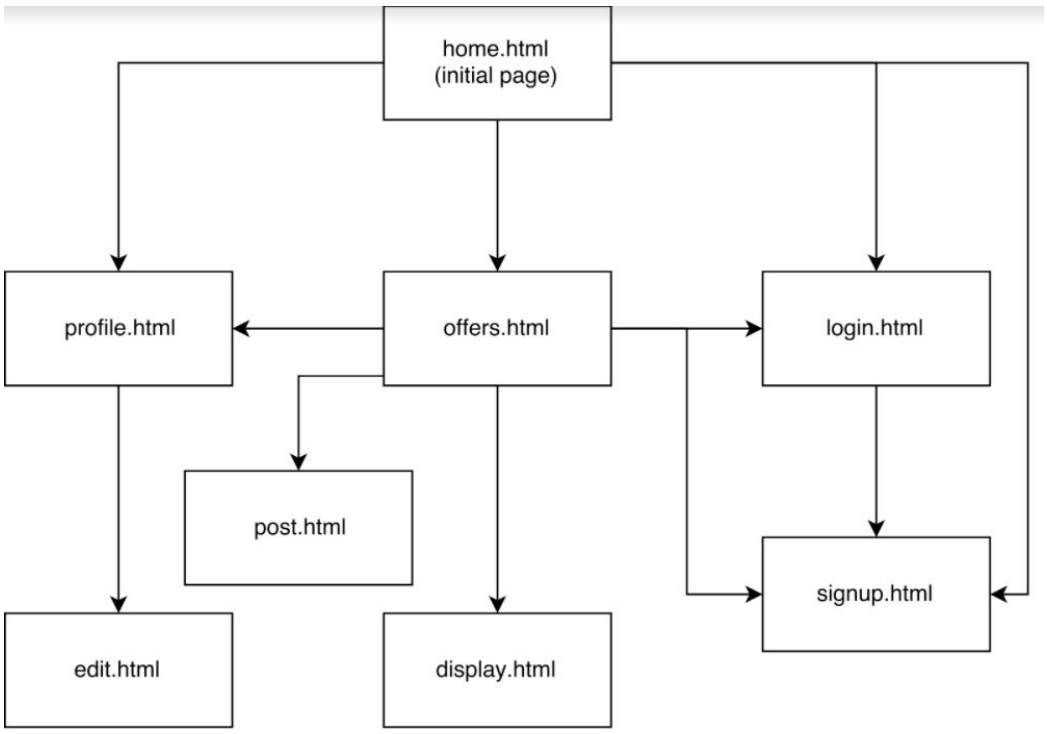
locker.py

- **create_locker(lockerID, floor, coords)** - Add locker information to database
- **get_email(lockerID)** - Return locker's owner's email
- **get_floor(lockerID)** - Return locker's floor
- **get_coords(lockerID)** - Return locker's coords
- **remove_locker(lockerID)** - Remove locker from locker database and user database
- **does_locker_exist(lockerID)** - Return true if locker exists in database, false otherwise
- **transfer_locker(lockerID, email (from), email (to))** - Transfer locker

offer.py

- **create_offer(lockerID, type, price, description)** - Create offer for locker based on locker information (type: sell = 0, trade = 1)
- **does_offer_exist(lockerID)** - Return true if offer exists in database, false otherwise
- **remove_offer(lockerID)** - Removes offer from offer.db
- **accept_offer(lockerID)** - Remove offer from offer.db and transfer locker to new user

SITEMAP



DATABASE SCHEMA

Profile Table

The profile table contains 5 fields: name, email, password, lockerIDs, and ban status. Name and password are used to log the user in. Email is a primary key so that you can't have a duplicate account. LockerIDs, is a string representation of a list of locker IDs that the user has selected as their locker. To edit the lockers list, eval(lockerList) can be used to get a working list, then the new ID can be appended, then repr(lockerList) can be used to turn it back into a list to update that field.

Name	**Email (primary key)**	**Password *encrypted**	**LockerIDs (String of a list)**	**Banned**
watson	watson@stuy.edu	ibm135	'[2-17, 10-57, 5-274]'	false
sherlock	sholmes@stuy.edu	shrek	'[3-160, 6-09]'	false
moriarty	moriarty@stuy.edu	p455w0rd3	'[]'	true

Locker Table

The locker table contains 4 fields: lockerID, owner, floor, and description. There is only one owner per locker.

LockerID (primary key)	**Owner(email)**	**Floor**	**Map Coordinates(tuple)**
2-17	watson@stuy.edu	2	(512, 14)
7-160	smitty_oo@stuy.edu	7	(69, 420)

Offer Table

The offer table contains 6 fields: offerID, lockerID, email of the seller, type (sell/trade), price, and description. The description allows the seller to add more information to their offer if they are looking for specific requirements.

OfferID (primary key)	**LockerID**	**Email of Seller**	**Type (sell = 0 trade = 1)**	**Price**	**Description**
0	2-17	watson@stuy.edu	0	$70	Great condition
1	7-160	sherlock@stuy.edu	1	$0	A second floor locker

TASK ASSIGNMENTS

Levi: Project Manager, HTML, javascript, customer service representative
Marcus: Database
James: Frontend
Ray: Bootstrap, HTML, CSS, javascript