Software Design Specification

Fight-Covid-19 System

CSCI 4050/6050 Fall 2020

Team 9A

Dan Doss	xxxxxxxxx
Pranay Kumar	xxxxxxxxx
Cassandra Blosser	xxxxxxxxx
Nathan Cooley	xxxxxxxxx
Ben Ware	xxxxxxxxx

Table of Contents

Revision History

Version	Name	Reason For Changes	Date
1.0		Initial Revision	12/01/2020

1. Introduction

1.1 Purpose

The purpose of this Software Design Specification is to describe the design details of the Fight COVID-19 System. This document will provide information regarding the system's design and architecture to aid in implementation and future development. User-Interface design and database design are also outlined to provide a detailed illustration of the system.

1.2 System Overview

The Fight COVID-19 System is a web-application for reporting positive COVID cases within the University of Georgia community. Users will be able to view current anonymized data of positive cases reported, sign-up for COVID-19 Surveillance Testing provided by the University Health Center, and check the latest state-wide COVID data. Users will also be able to create an account using their UGA email. Once authenticated, they can sign-up for notifications for specific buildings, days, and times. Authenticated users can also report positive COVID-19 test cases along with 3 buildings they frequented recently.

2. Design Considerations

- When a user reports a test other users signed up for that building and time automatically receives an email.
    - Our system automatically handles this using a custom View.
- Users should be able to see a graph of Athens to get an idea of how many cases are occurring in the county.
    - Our system pulls data directly from https://covid19.biglocalnews.org/ to display this information.
- Users should also be able to access a list of all reported tests on UGA's campus, regardless of whether or not they signed up for that building.
    - Our system stores every reported test in the SQLite database and displays the list as a scrollable menu below the county Covid data.
- Users can access a personal account page to view and change their building signup preferences with ease.
    - Our system utilizes AJAX to efficiently edit data in place, providing visual accessibility for the user.
- The system should respond efficiently to user input, and should clearly display when a loading action is in progress if necessary.
    - A loading screen has been developed in instances in order to provide the user with adequate visual feedback.
- All users should be members of the UGA community.
    - Our signup page enforces this with a helpful message should the user attempt to use a non uga.edu email address.

- Account creation should require verification before allowing a user to report tests or sign up for buildings.
    - Users receive an account confirmation email with the help of a Django email verification package.

2.1 Constraints.

- We intended to link our system with the University index of buildings and classrooms, but EITS would not allow us access to that data, so we had to scale back our thinking. As a result, tests are reported by building only, not by classroom and building.
- We initially wished to graph out Covid data for the University to display how case numbers change over time. Unfortunately, that was well beyond the scope of our project so instead we have pulled data from https://covid19.biglocalnews.org/.
- The aforementioned graph takes a few seconds to load, so a loading screen was added to provide clarity to the user.
- We chose SQLite for ease of use and for fast computation. For a more robust system, a more powerful database engine may be needed.

2.2 Design Methodology

Our team adopted the Agile methodology throughout the duration of the project. Work was done in a series of two week sprints, with an emphasis on implementing features and working software over documentation. After an initial period of planning our intended components and features, our team worked independently on different aspects of the system. Before each sprint, our team would have a brief sprint planning meeting, in which we would assemble a list of tasks and then assign them to members. Each completed task would be marked off, allowing for an easy visual reference of what was left for that sprint. If any items were not finished in time, they would be moved to the next sprint list.

In order to keep the team on track and ensure tasks were getting finished in a timely manner, we used Discord to track tasks and host weekly team meetings. Meetings occurred every Wednesday evening, with extra meetings being scheduled before project deadlines as necessary. All design decisions were made within these meetings in order to ensure everyone remained on the same page. Additional communication such bug reports, meeting coordination, design screenshots, and deadline reminders were handled in text channels within Discord.

3. The Fight Covid-19 System Architecture

The architecture provides the top-level design view of a system and provides a basis for more detailed design work. In this section we will present the high-level level system decomposition and the details of the services provided by system components.

3.1 Overview

We used Django, a python web framework, for our application. Django utilizes an MVC architecture style. The architecture is diagrammed in Figure 1. Thus our programs architecture is essentially identical to the MVC architecture from our class slides, save for the controller's use of an email server to notify people about cases, the softwares primary function, as well as the minor function of email verification.

Django has the benefit of being bundled with a lot of default functionality. This is particularly noticeable for security because Django has administrative accounts and stored password hashing implemented by default. By using a standardized security implementation, the chance of a security oversight in implementation is severely reduced.

In order to keep our system organized, our file structure follows standard Django convention. Thanks to this straightforward hierarchy, implementing features and updates have been simple.

Architecture Diagram

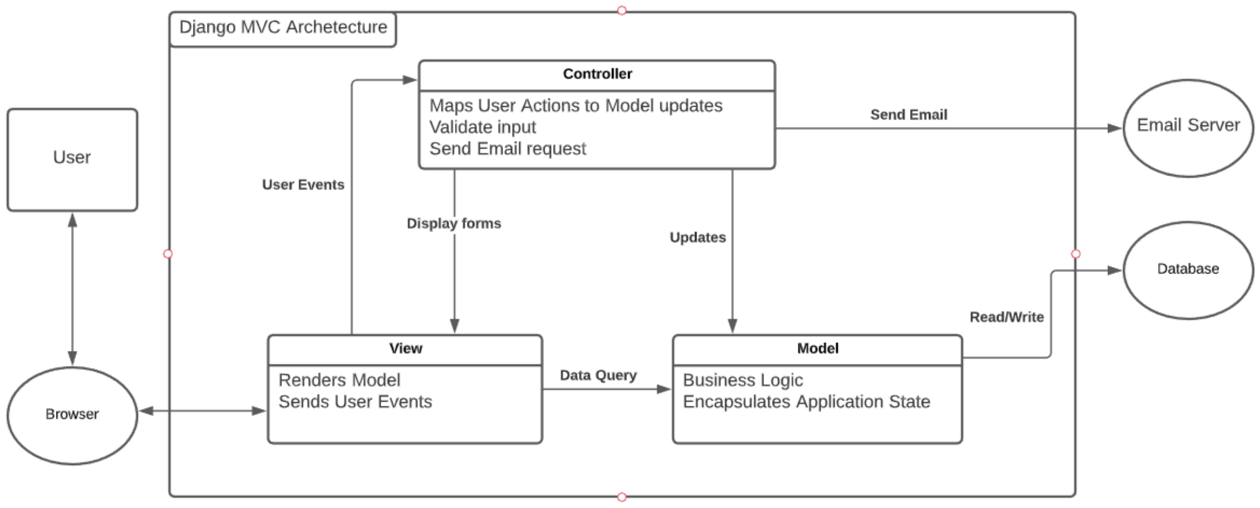


Figure 1. Architecture Diagram

3.2 The View Component

The view component consists of two folders: a templates folder for HTML templates and a static folder for Javascript, CSS, and other related files such as images (/templates and /static, respectively.)

Django uses a HTML like code to insert data, format the HTML components, and retrieve data from forms. Both the templates and the data are received from the controller component.

3.3 The Controller Component

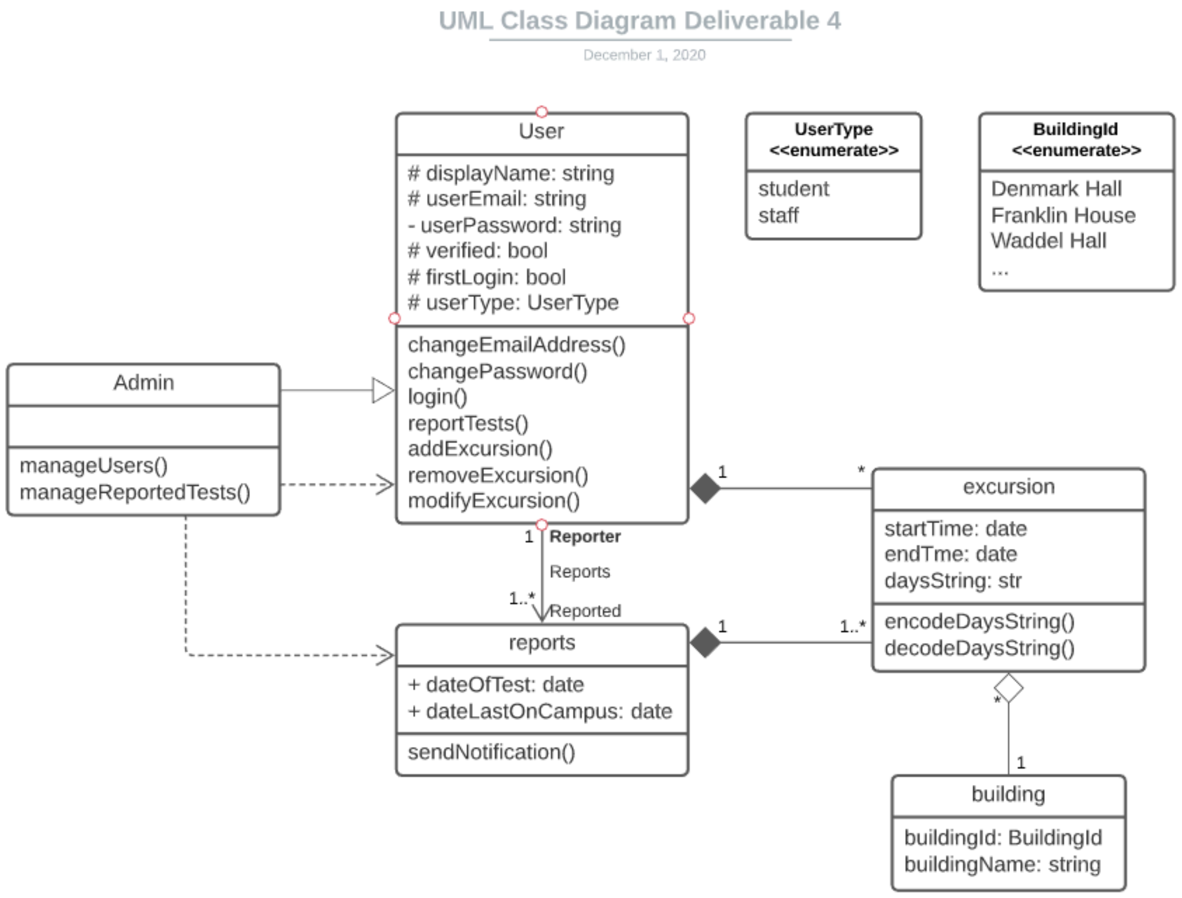
The major components that make up the controller are urls and views. When a user accesses some url of the site such as sitename.com/home, the django looks through the defined urls and finds one that matches. Once such a url is found, the view related to that url is consulted and the view handles everything related to that url such as which html template to render, what dynamic information needs to be passed to that template, how to process get/post requests, and any other business logic that needs to be performed.

3.4 The Model Component

Django provides built in functionality for model creation and interaction. A class is made for each table in the database and the class inherits from a Model class provided by Django. All the attributes and fields needed are defined as members of the class. Django then takes these classes and uses a built in object-relational mapping layer to convert the defined classes into tables in the database defined in the settings file, in our case sqlite3.

3.4.1 Design Class Diagram for the Model Component

As we discussed on zoom, please regrade this diagram for deliverable 4.



4. Design Patterns Used

Since we used the Django web development framework for our project, we utilized the Model View Controller design pattern. Django projects are inherently set up in a manner following the MVC pattern.

The Fight Covid-19 System follows the MVC approach to software design, specifically using Django's own methodology. The standard MVC pattern divides the system into three layers; the Model handles data representation, the View handles the UI as rendered in the browser, and the Control handles the business logic for the system. Django considers itself an MTV, or "Model Template View" system. Each piece is analogous with its respective part in the MVC, but with a modified naming convention. It is worth noting that in Django terminology. "View" refers to handling business logic, not the UI.

One design pattern we used was lazy initialization for creating our report objects. Initially, we created a report object and iterated over the excursion forms creating excursion objects linked to the report. If no excursion forms were valid, the report object had to be destroyed. To prevent this unnecessary processing, the creation of the report object was delayed to when the first valid excursion form was validated. This was further enhanced by checking for a valid form before iterating through them.

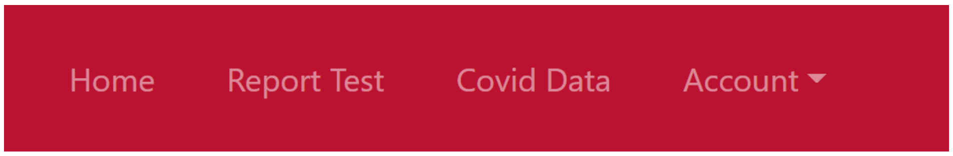
Aside from these, we did not implement any particular design patterns of our own because of the flexibility provided by Django and its handling of lower level functionality.

Django itself implements many patterns. For example, it has adapters for the database, which prevents the need to directly change the database and allows for easy switching between different available databases such as from sqlite3 to PostgreSQL.

5. User Interface Design

5.1 Application Controls

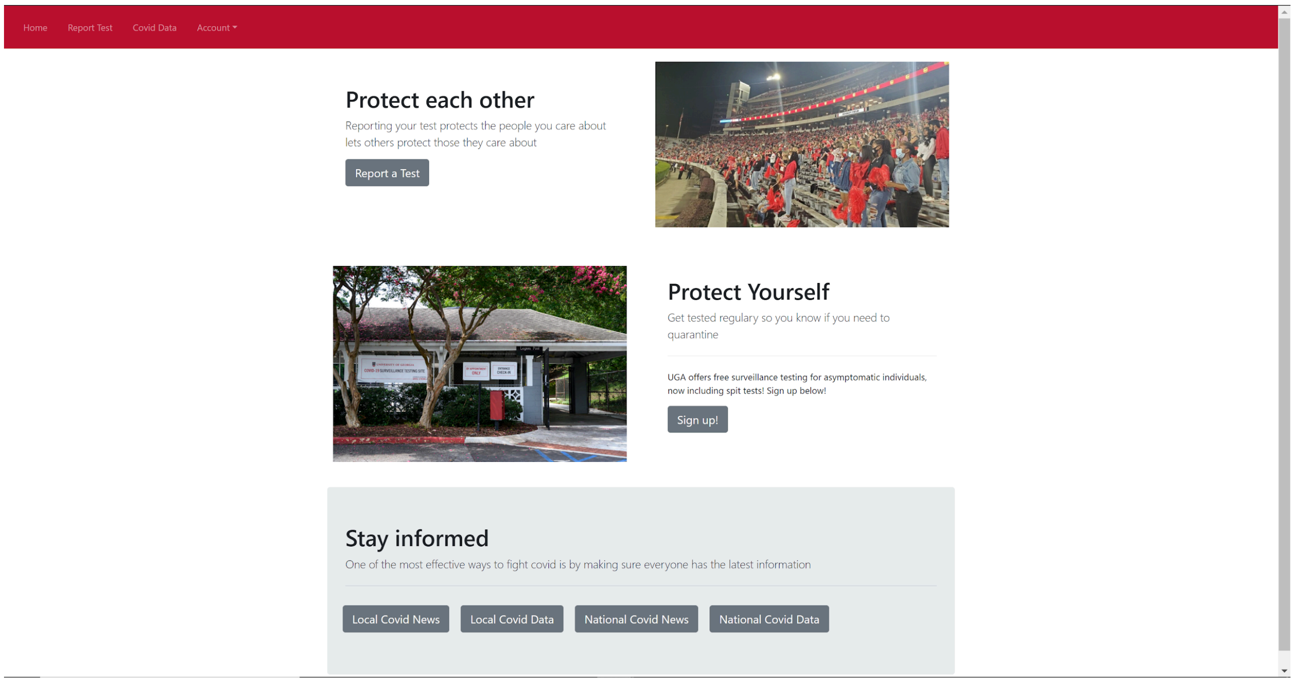
All screens not associated with login will have the following toolbar in the upper left hand corner.



They will also have a white background with either a white or grey body and grey or red buttons in bootstrap 4 style.

Pages that are associated with logging in (login, logout, forgot password, and create account) will have a white vertical popup that slides down from the top center of the screen and a dark grey background. The popup will have the UGA logo at the top and red buttons to create the same feel as logging into an official uga page.

5.2 Screen 1-8



Protect each other

Reporting your test protects the people you care about lets others protect those they care about

Protect Yourself

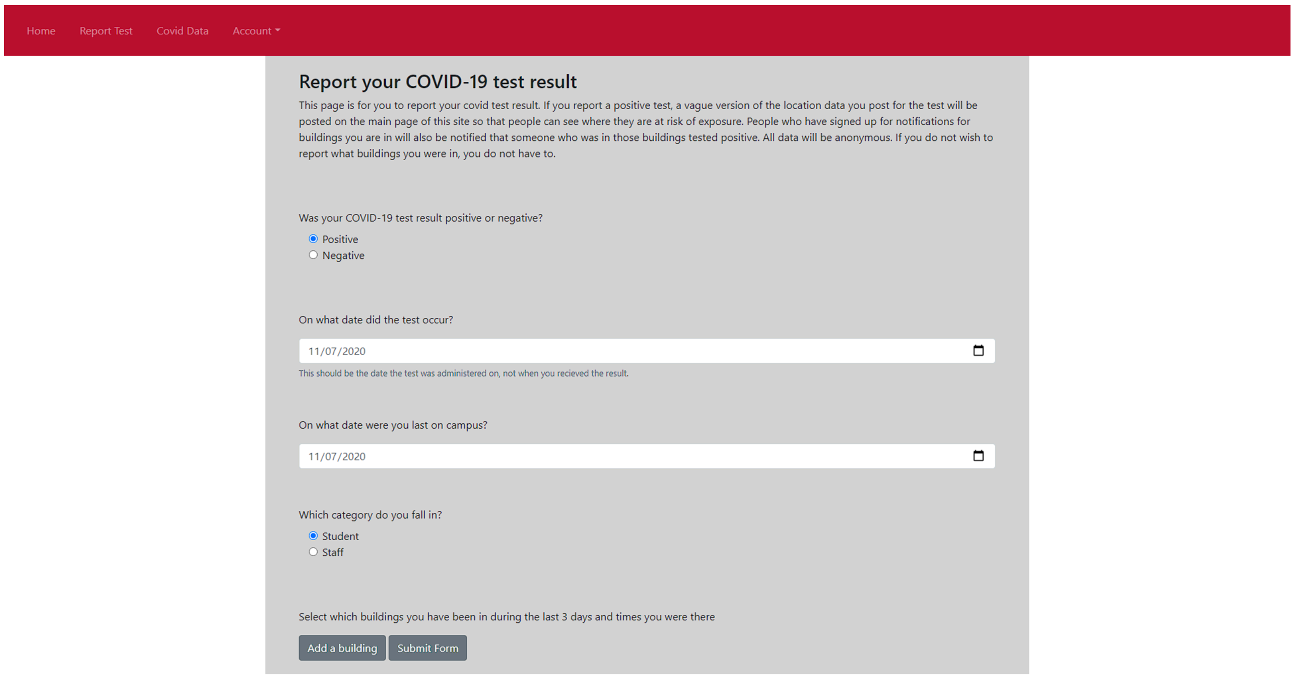
Get tested regulary so you know if you need to quarantine

UGA offers free surveillance testing for asymptomatic individuals, now including spit tests! Sign up below!

Stay informed

One of the most effective ways to fight covid is by making sure everyone has the latest information

This is the home page with the standard navbar at the top. The page will always look like this for all users.



Report your COVID-19 test result

This page is for you to report your covid test result. If you report a positive test, a vague version of the location data you post for the test will be posted on the main page of this site so that people can see where they are at risk of exposure. People who have signed up for notifications for buildings you are in will also be notified that someone who was in those buildings tested positive. All data will be anonymous. If you do not wish to report what buildings you were in, you do not have to.

Was your COVID-19 test result positive or negative?

[✓ Positive] Negative

On what date did the test occur?

11/07/2020

This should be the date the test was administered on, not when you recieved the result.

On what date were you last on campus?

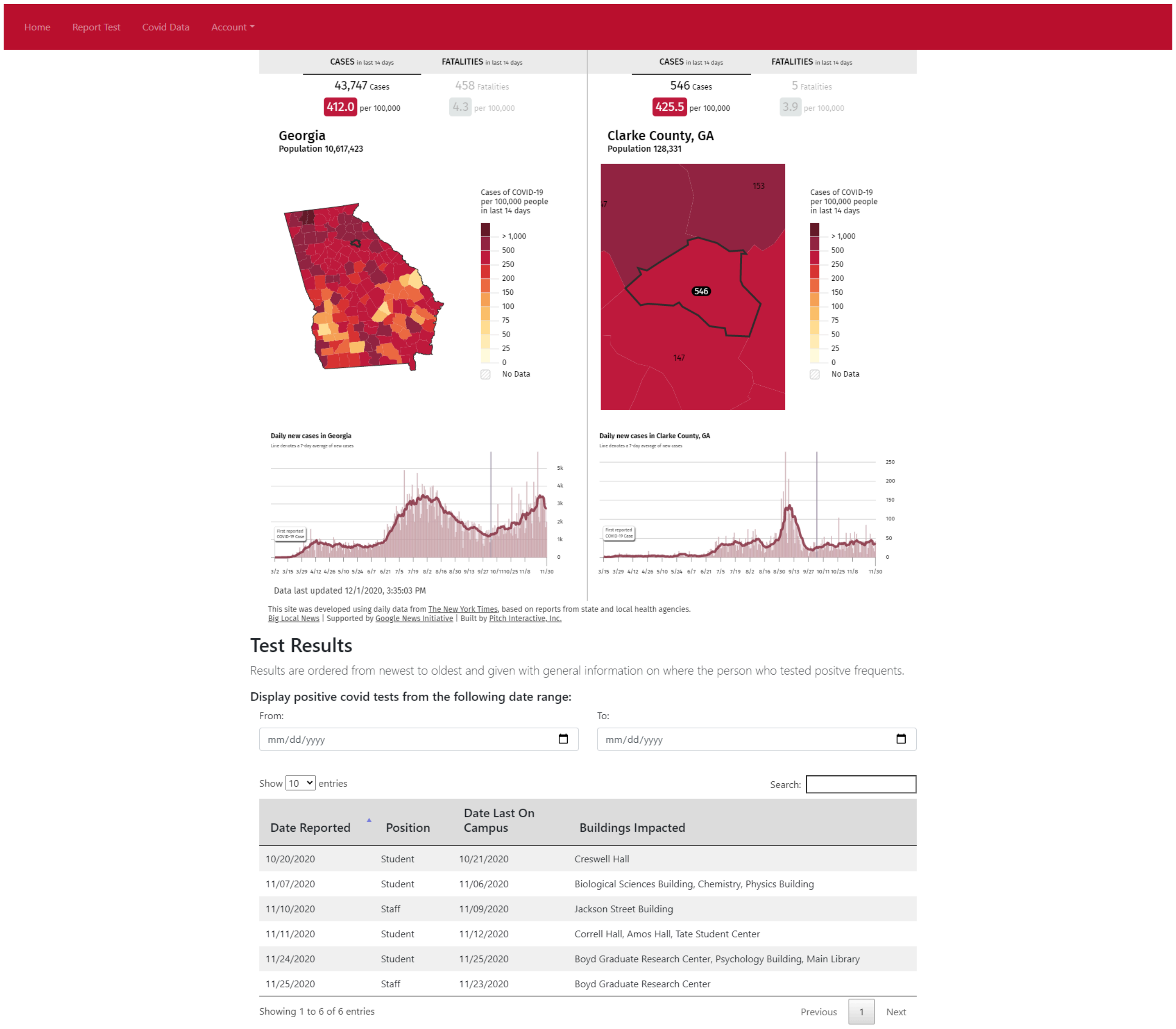
11/07/2020

Which category do you fall in?

[✓ Student] Staff

Select which buildings you have been in during the last 3 days and times you were there

This is the page for users to report a test. If the user is not logged in, trying to access this page will redirect them to the login page and back to this page once they are logged in.



CASES in last 14 days | FATALITIES in last 14 days

43,747 Cases | 458 Fatalities

412.0 per 100,000 | 4.3 per 100,000

Georgia
Population 10,617,423

CASES in last 14 days | FATALITIES in last 14 days

546 Cases | 5 Fatalities

425.5 per 100,000 | 3.9 per 100,000

Clarke County, GA
Population 128,331

Data last updated 12/1/2020, 3:35:03 PM

This site was developed using daily data from The New York Times, based on reports from state and local health agencies.
Big Local News | Supported by Google News Initiative | Built by Pitch Interactive, Inc.

Test Results

Results are ordered from newest to oldest and given with general information on where the person who tested positve frequents.

Display positive covid tests from the following date range:

From: mm/dd/yyyy

To: mm/dd/yyyy

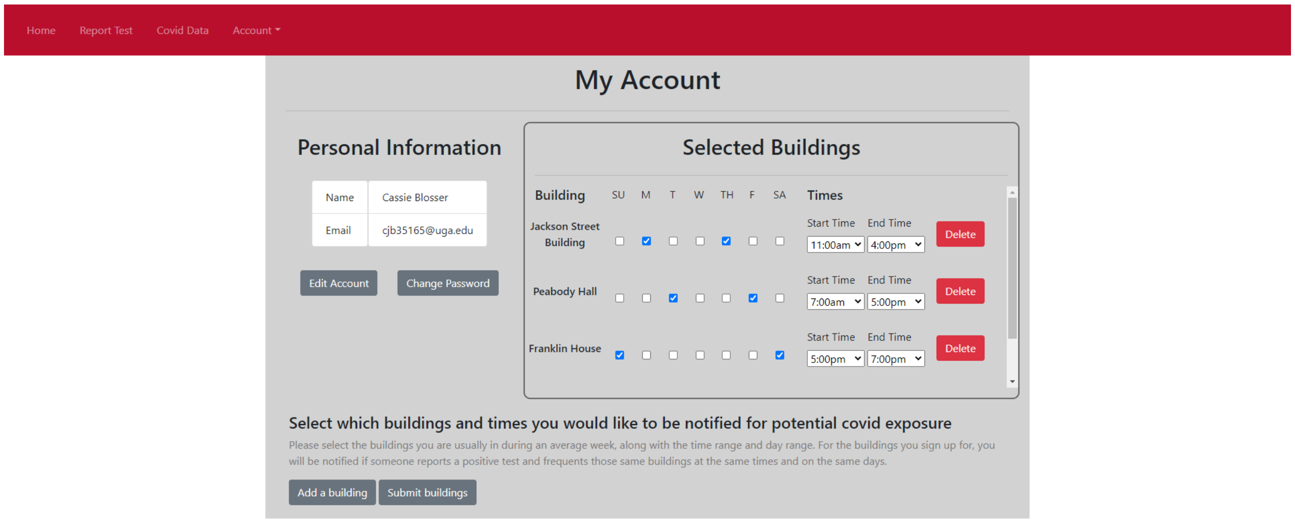
Show 10 entries

Search:

Date Reported	Position	Date Last On Campus	Buildings Impacted
10/20/2020	Student	10/21/2020	Creswell Hall
11/07/2020	Student	11/06/2020	Biological Sciences Building, Chemistry, Physics Building
11/10/2020	Staff	11/09/2020	Jackson Street Building
11/11/2020	Student	11/12/2020	Correll Hall, Amos Hall, Tate Student Center
11/24/2020	Student	11/25/2020	Boyd Graduate Research Center, Psychology Building, Main Library
11/25/2020	Staff	11/23/2020	Boyd Graduate Research Center

Showing 1 to 6 of 6 entries

This is the page for displaying covid data. The top section is loaded from an embedded link and sometimes takes a few seconds to load. A loading icon will be displayed in this case. The bottom section is data from our database and allows the user to scroll through it separate from the main page and select which dates to see data from. It also includes the standard navbar.



My Account

Personal Information

Name	Cassie Blosser
Email	cjb35165@uga.edu

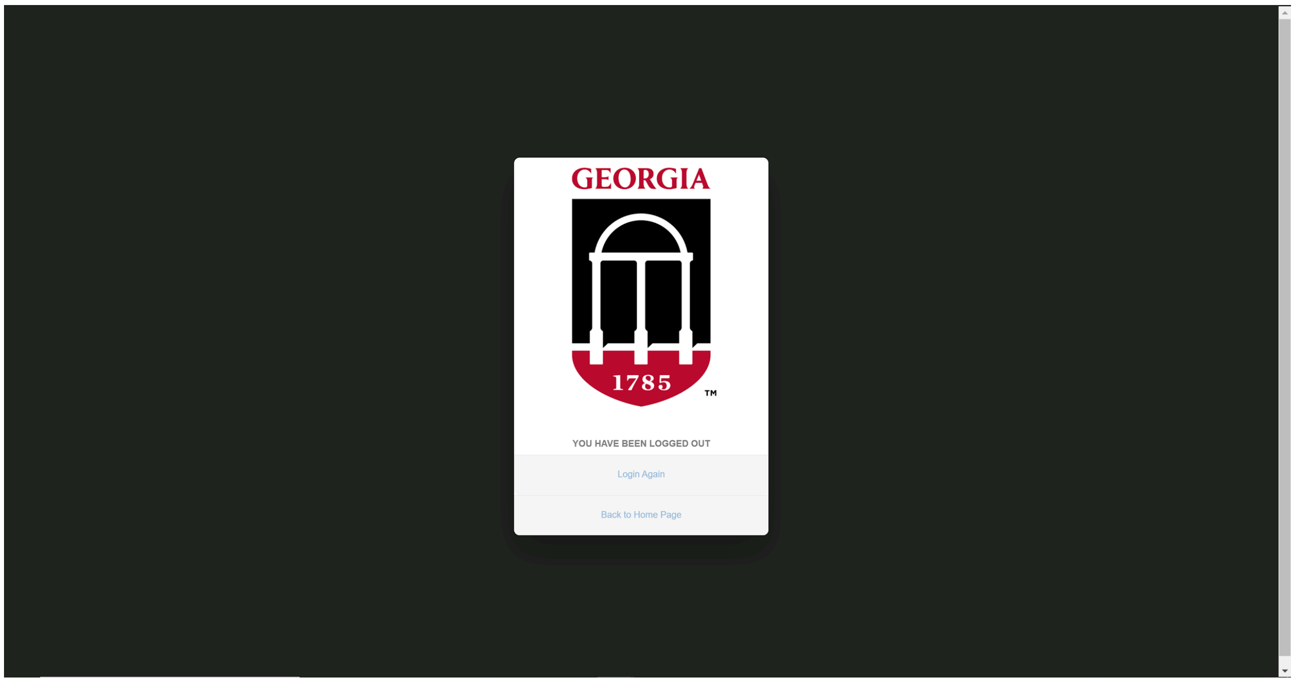
Selected Buildings

Building	SU	M	T	W	TH	F	SA	Times		
								Start Time	End Time	
Jackson Street Building	☐	✓	☐	☐	✓	☐	☐	11:00am	4:00pm	Delete
Peabody Hall	☐	☐	✓	☐	☐	✓	☐	7:00am	5:00pm	Delete
Franklin House	✓	☐	☐	☐	☐	☐	✓	5:00pm	7:00pm	Delete

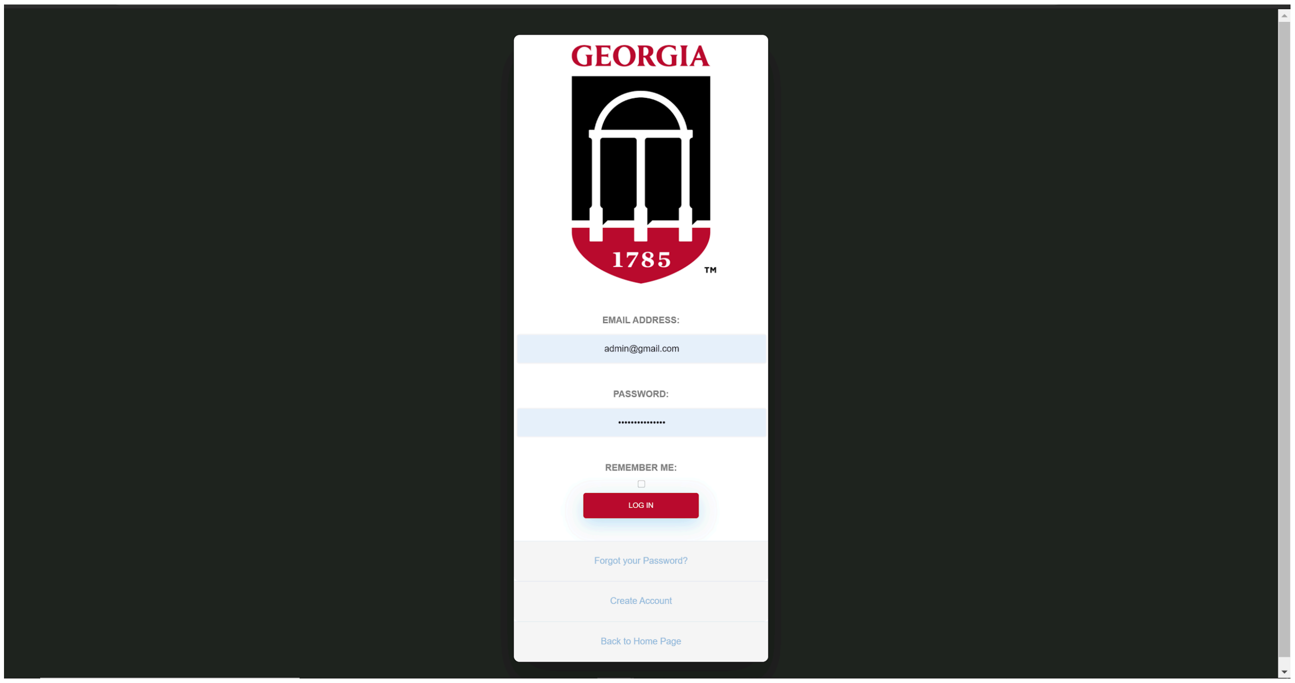
Select which buildings and times you would like to be notified for potential covid exposure

Please select the buildings you are usually in during an average week, along with the time range and day range. For the buildings you sign up for, you will be notified if someone reports a positive test and frequents those same buildings at the same times and on the same days.

This is the account information page. It always looks like this. Edit account opened a modal to edit your name. Change password takes the user to a separate page. On the right you can see what buildings you are signed up for notifications for and remove them, This scrolls independent of the page. The bottom section allows you to add buildings to this list.

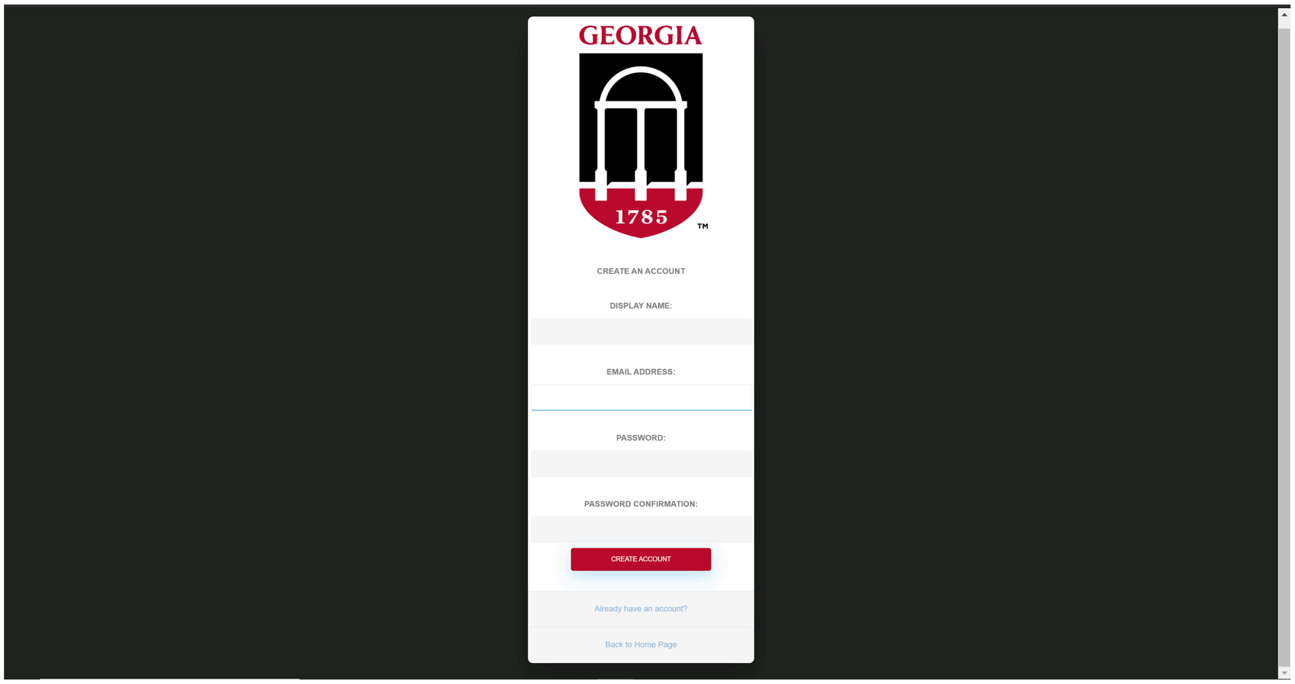


This is the logout page.

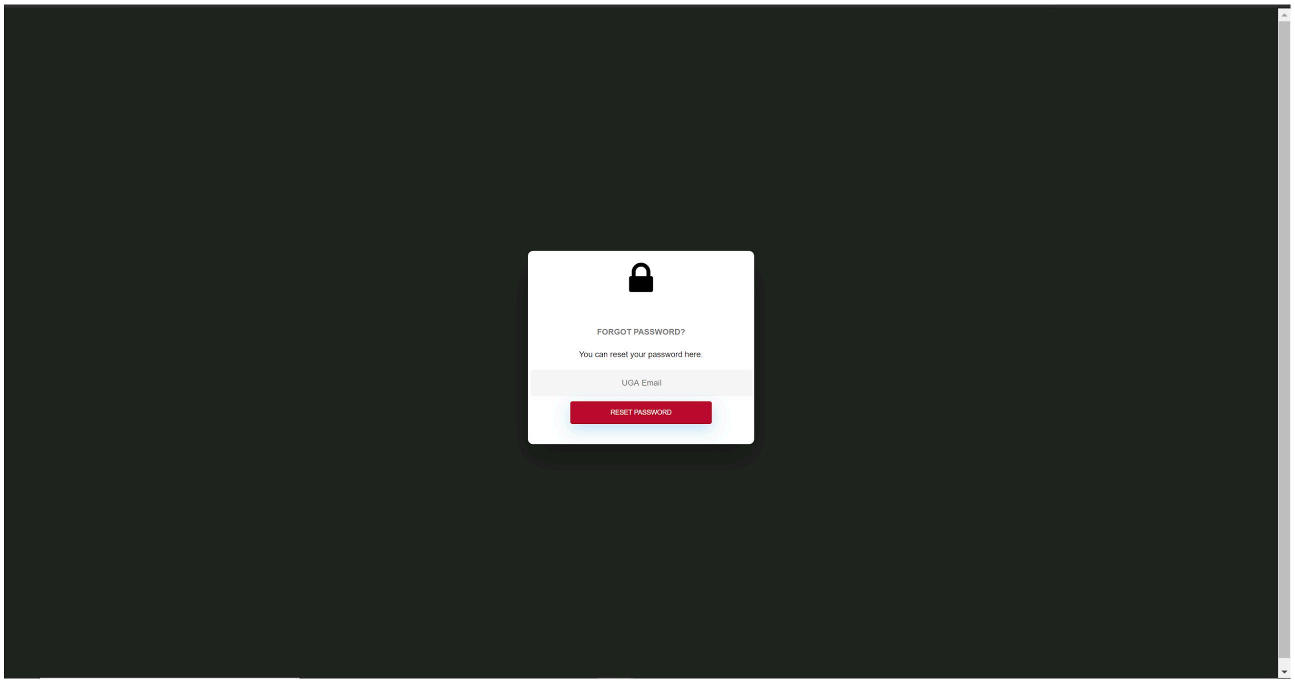


admin@gmail.com

This is the login page.



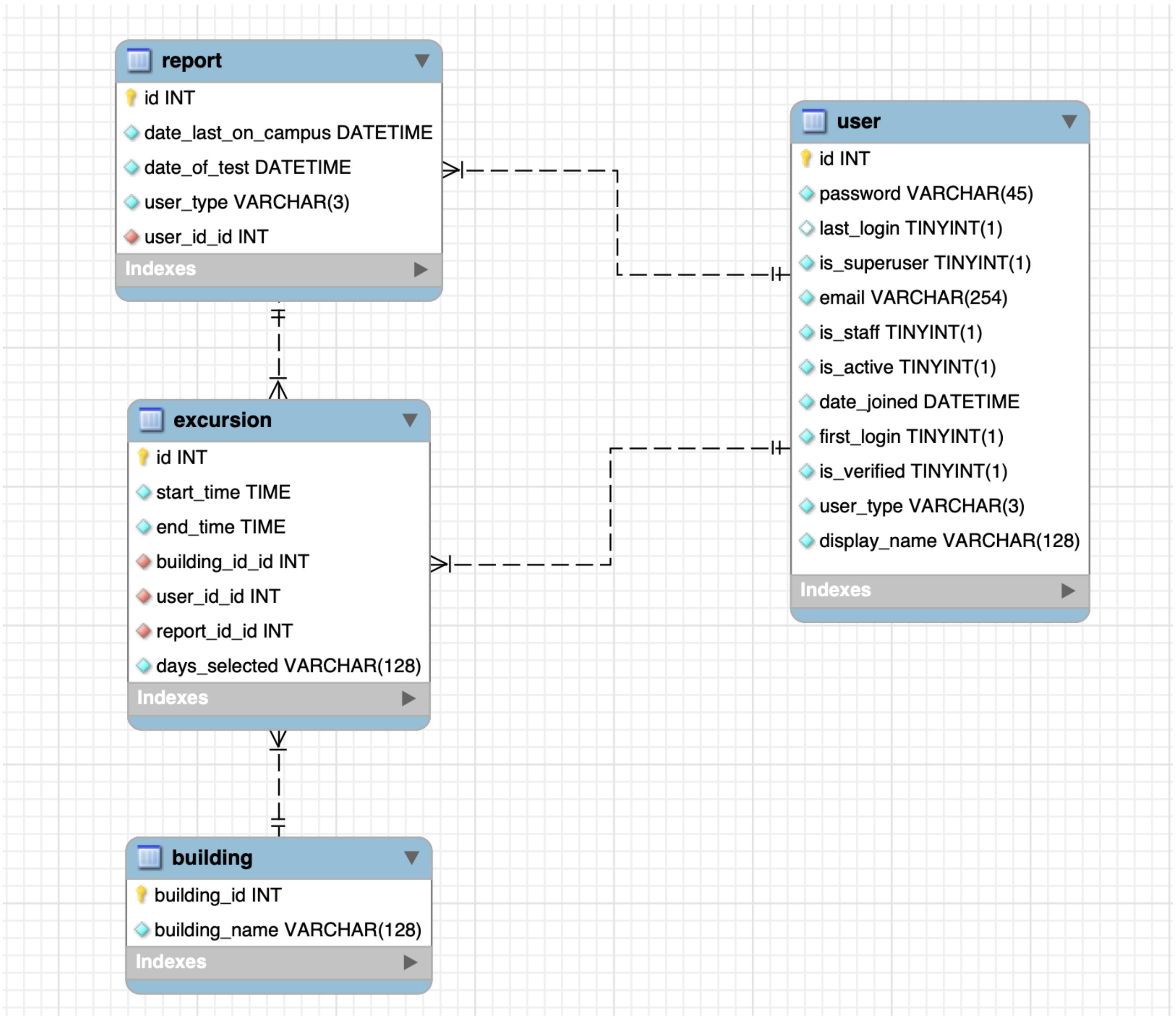
This is the create account page.



This is the forgot password page.

6. Database Design

6.1 Entity Relationship Diagram



report
id INT
date_last_on_campus DATETIME
date_of_test DATETIME
user_type VARCHAR(3)
user_id_id INT
Indexes
user
id INT
password VARCHAR(45)
last_login TINYINT(1)
is_superuser TINYINT(1)
email VARCHAR(254)
is_staff TINYINT(1)
is_active TINYINT(1)
date_joined DATETIME
first_login TINYINT(1)
is_verified TINYINT(1)
user_type VARCHAR(3)
display_name VARCHAR(128)
Indexes
excursion
id INT
start_time TIME
end_time TIME
building_id_id INT
user_id_id INT
report_id_id INT
days_selected VARCHAR(128)
Indexes
building
building_id INT
building_name VARCHAR(128)
Indexes