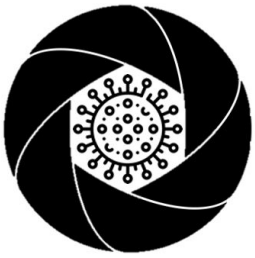


COVID Lens

Isaac Taylor, Mark He, Reagan Berhe, Seth Goodwin, Tammy Ogunkale

Table of Contents

Project Definition

COVID Lens is an iOS application that will be utilized as a viable data-driven tool to help combat the current safety and social threats posed by the COVID-19 virus. The global COVID-19 pandemic was declared on March 11, 2020 by the World Health Organization and, as the new year is slowly approaching, the virus shows no apparent signs of disappearing soon. This suggests a necessity of new data-driven tools and methods to help society work around and cope with COVID-19 while promoting safety. The main goal of COVID Lens is to produce a novel COVID-19 tracking and public awareness system that will inform and protect members of the UNCG community from the COVID-19 virus. COVID Lens will regularly update and inform users of the possibility of COVID-19 exposure in their area. Users will be provided with information and resources - from reliable sources such as the Centers of Disease Control and World Health Organization - to stay up-to-date with current COVID-19-related news as well as current safety measures. Users will also be able to view a real-time map that displays areas where positive COVID-19 diagnoses have been reported.

Functional Requirements

- The application should be able to track user location data and update it in real time.
- The application should be able to display data on a real-time map.
- The application should be able to gather required data from users.
- Users should be able to successfully sign up and sign in using their login credentials.
- The application should provide users with guidance and reminders regarding COVID-19 safety measures.
- The user should be able to view current COVID-19 statistics in their area.
- Users should be able to self-report a positive diagnosis for COVID-19 while preserving anonymity.
- The application should provide users with nearby COVID-19 testing locations.

Usability Requirements

- Users will be able to adjust settings to personalize their experience with the app.
- Authorized users will have fast access to data upon request.
- The functionality of the user interface should be clear to the users.
- Elements of the user interface (e.g., menus) should be easy to understand.
- Users should be able to use and navigate the application with no prior knowledge.
- The user interface should be visually appealing to the user.
- The user interface actions and elements should maintain consistency.

System Requirements

- Supporting any major version of iOS 13 or higher.
- Basic internet connectivity or cellular data service.
- Additional recommended requirements:
    - Bluetooth-supported devices
    - Mobile location data enabled

User Requirements

- Users must have an email address to sign-up and sign-in.
- Users must own an Apple mobile device running iOS 13 or higher.
- Allow user to access basic Application & Privacy Permissions.
    - Notifications
    - Camera
    - Location Services
    - Photos
    - Bluetooth Sharing

Security Requirements

- Application must maintain the confidentiality of all sensitive data that is classified confidential.
    - Emails
    - Passwords
    - Location data
    - User diagnosis report
- Application libraries/database/APIs must follow and ensure the confidentiality of login information.
    - Sign-up/sign-in APIs.
    - User's personal medical data.

Project Specifications

Focus Domain

COVID Lens will serve to inform and protect members of the UNCG community from the COVID-19 virus by containing it as it encourages users to self-quarantine when needed and social distance. It would also focus on informing the UNCG community on preventive measures from the virus.

Libraries / Frameworks

- MySQL for database services
- Swift for front end development
- Python for data analysis
- PHP for REST API/ back end development

Platform

An iOS application available for Apple mobile devices.

Genre

An iOS application fitting into the Health and Education genres. COVID Lens strives to promote health and safety of the community and to keep users informed and educated about the current threats imposed by the COVID-19 virus.

Feasibility

Cost(s)

- The list of softwares, applications, and other virtual equipment are all free and open-sourced. Thus, the estimated goal for the financial cost of developing this application is of no cost.

Potential Risks

- An API or third party application could fail to load information needed to run COVID Lens.
- Users are liable to upload valid proof of positive diagnosis and false information can be provided in this approach if not evaluated correctly.

Risk Minimization

- In regards to unexpected API behavior, this risk will be minimized by adaptable architecture for simple switching to new API's
- By having trusted human verifiers, the risk of false or misleading information about positive cases can be reduced.

Potential Benefits

Successful implementation of COVID Lens will likely:

- Contribute to help minimize the risk of further infections.
- Promote more awareness of the severity of the situation.

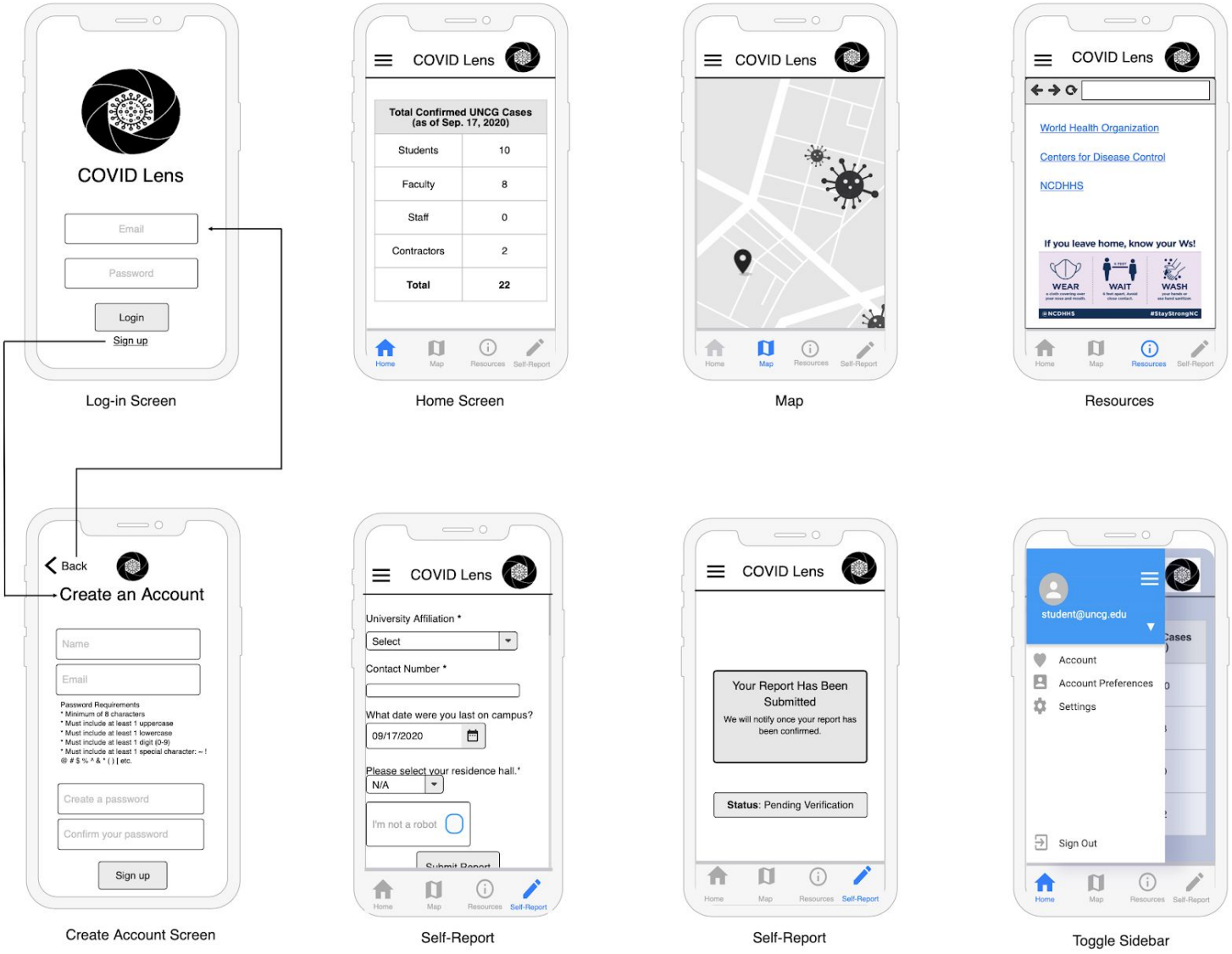
- Encourage self-quarantine, social distancing, and wearing face coverings.
- Inform or help users self-educate themselves.

Overall Viability

Taking the overall costs, potential risks, and benefits into consideration, we believe that there is a high chance of success in regards to meeting design expectations and desired outcomes.

System-Design Perspective

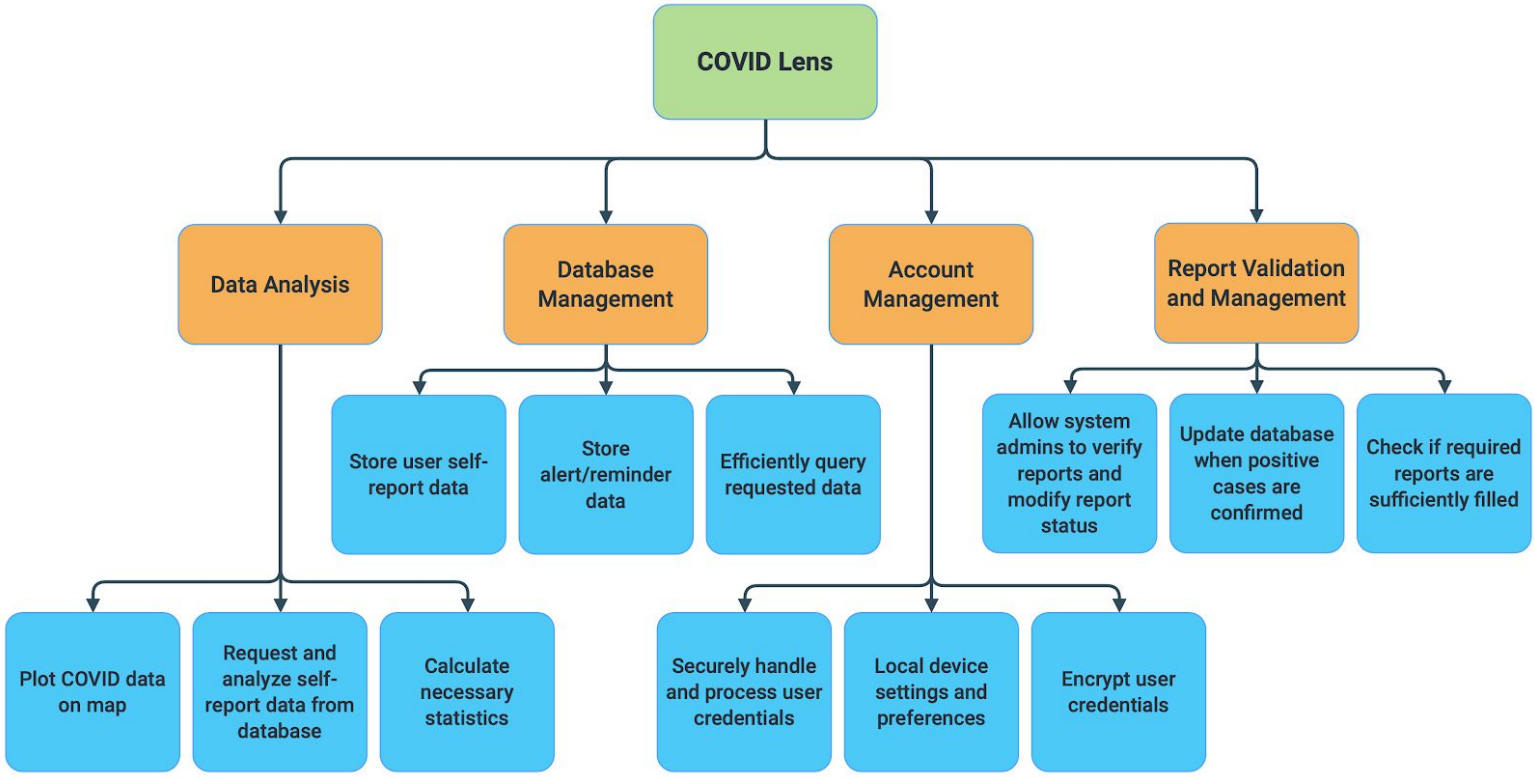
Application Wireframe Models



- **Log-in Screen**: Allows users to log-in to the COVID Lens application.
- **Create Account Screen**: Allows users to create a COVID Lens account.
- **Home Screen**: Will show an updated count for all positive COVID cases reported by UNCG-associated individuals.

- **Map**: Will display an icon (resembling a virus) that will correspond to each UNCG residence hall location. Each icon will display the number of confirmed COVID cases at each residence hall.
- **Resources**: Will provide relevant information, reminders, and links to inform users about COVID-related news and safety measures.
- **Self-Report**: Allows users to self-report a positive COVID diagnosis. Will be updated with the current status of their report.
- **Toggle Sidebar**: Allows users to manage their account, change settings within the app, and sign out.

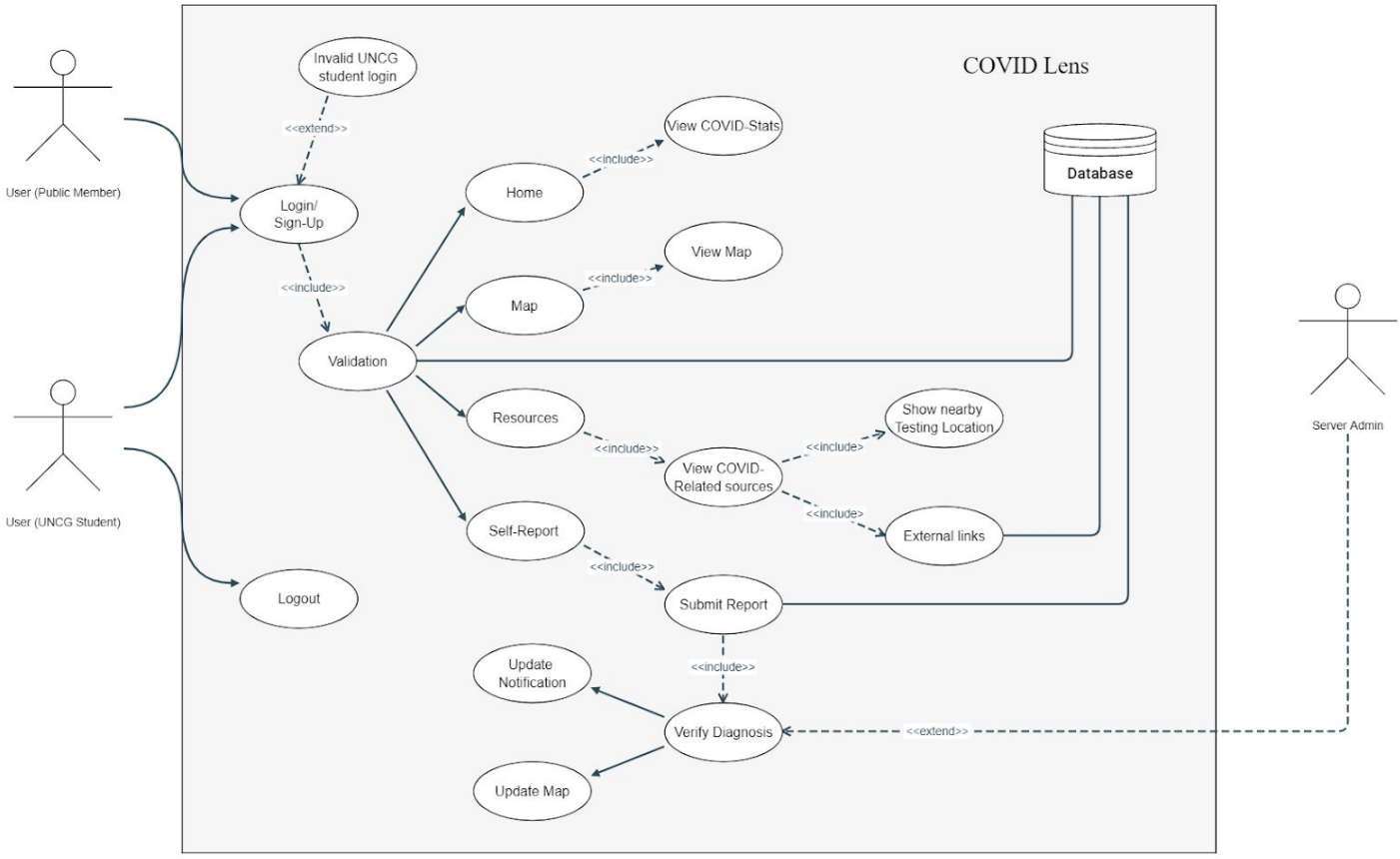
Top-Down Diagram



The top-down diagram above models the sub-processes of each individual subsystem as follows:

1. Data Analysis
    - Plot COVID data on map
    - Request and analyze self-report data
    - Calculate necessary statistics
2. Database Management
    - Store user self-report data
    - Store alert/reminder data
    - Efficiently query requested data
3. Account Management
    - Securely handle and process user credentials
    - Local devices settings and preferences
    - Encrypt user credentials
4. Report Validation and Management
    - Allow system admins to verify and modify report status
    - Update database when positive are confirmed
    - Check if required reports are sufficiently filled

Use-Case Diagram

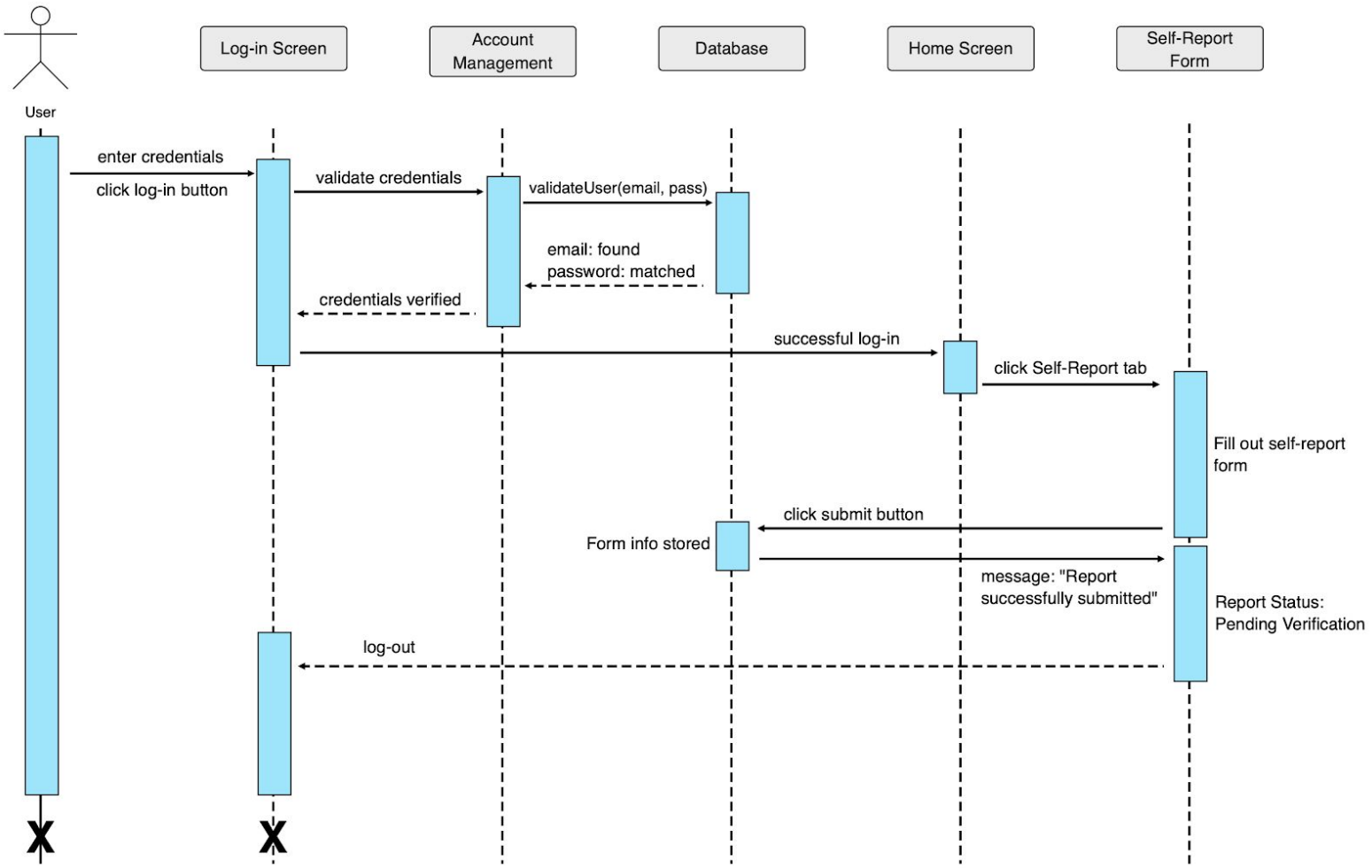


A general view on the application's expected system. Users who downloaded the application may or may not be UNCG students, however we make a clear distinction when they're asked to sign in with a valid UNCG username and password. Upon successfully logging in, the user data provided by their Google account is stored to the database for better user experience. After that, the users will land on one of the four main navigation tabs with each having their own purpose.

- Home - users view most updated statistics of total confirmed cases on campus.
- Map - Interactable map UI with displayable cases from each resident's halls on campus.
- Resources - users are shown relevant information from official, trusted sources
    - Testing Location - searches for & displays the nearest COVID-19 Testing Centers.
    - External Links - List of related HTTPS URLs for results that matched certain keywords. External links are stored in the database.
- Self-Report - Users are asked to fill in required information on form.
    - Upon successful submission of the report, data will be filtered to the database. Server admins are able to review these diagnosis reports and determine whether

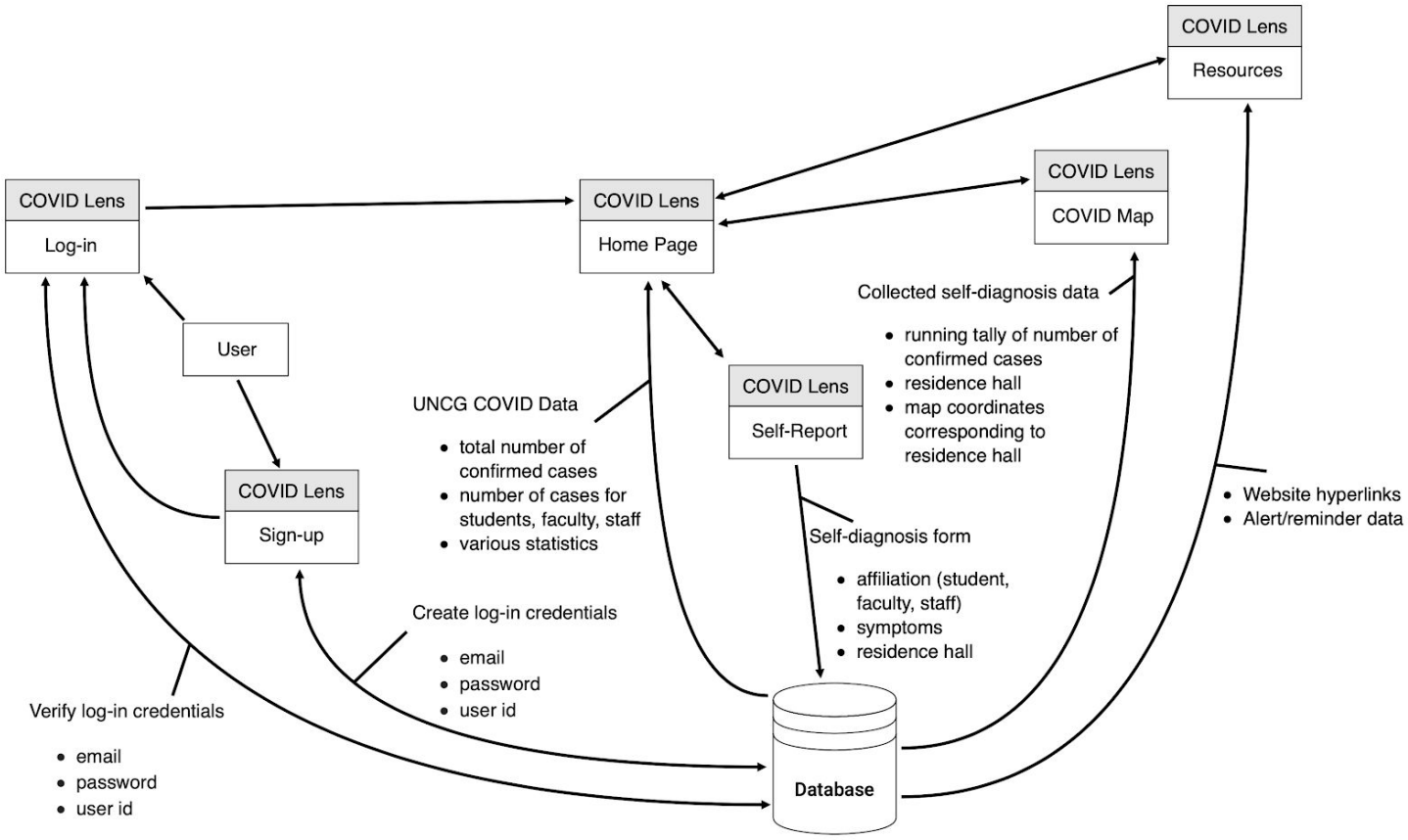
the report is valid. If validated, the data will be updated to the map and update notification will be sent.

Sequence Diagram



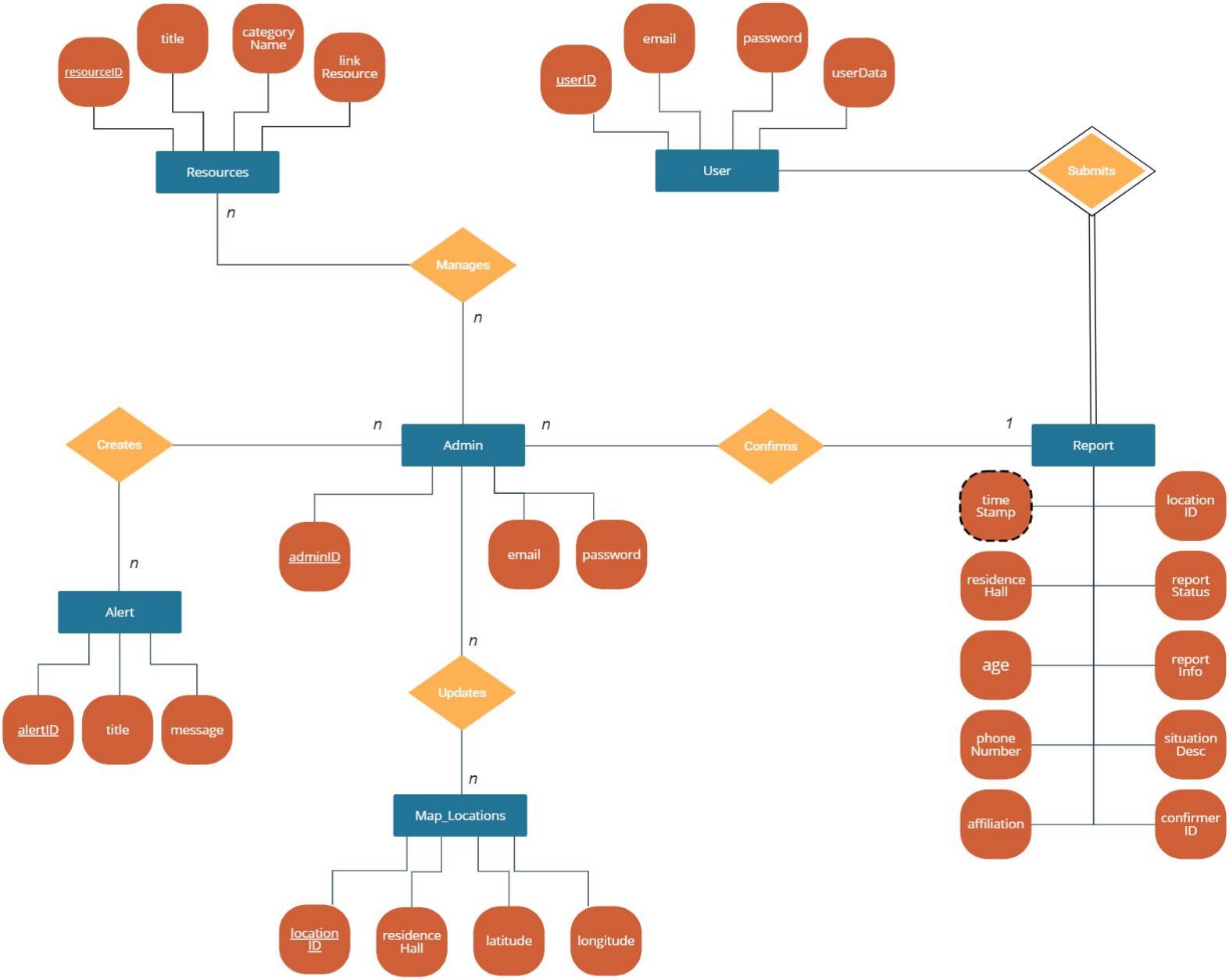
The sequence diagram above models the scenario of a user logging into the application and then submitting a self-report form. After the user enters their log-in credentials and clicks “log-in”, the Account Management system verifies their login credentials with the system Database. Upon successful login, the user is taken to the Home Screen which shows various statistics regarding COVID-19 cases on the UNCG campus. The user then clicks on the Self-Report navigation tab, fills out the self-report form, and then submits the form. The information contained within the form is stored in the Database. Then the user is notified that their report was successfully submitted and the report status is updated to “Pending Verification.” Lasty, the user logs out of their account and is taken back to the Log-in Screen.

Data Flow Diagram



The data flow diagram above provides a high-level model of how data is being used within the COVID Lens application. Upon successful registration and login, the user is presented with the Home Page. All login credentials are securely stored in the Database. The Home Page pulls UNCG COVID data from the Database and will present it in tabular format. The Self-Report page will collect and store users' self-report information in the Database. The COVID Map uses the users' self-diagnosis data stored in the Database to plot data on the map. Lastly, the Resources page will provide links and other useful resources (e.g., reminders) from the Database.

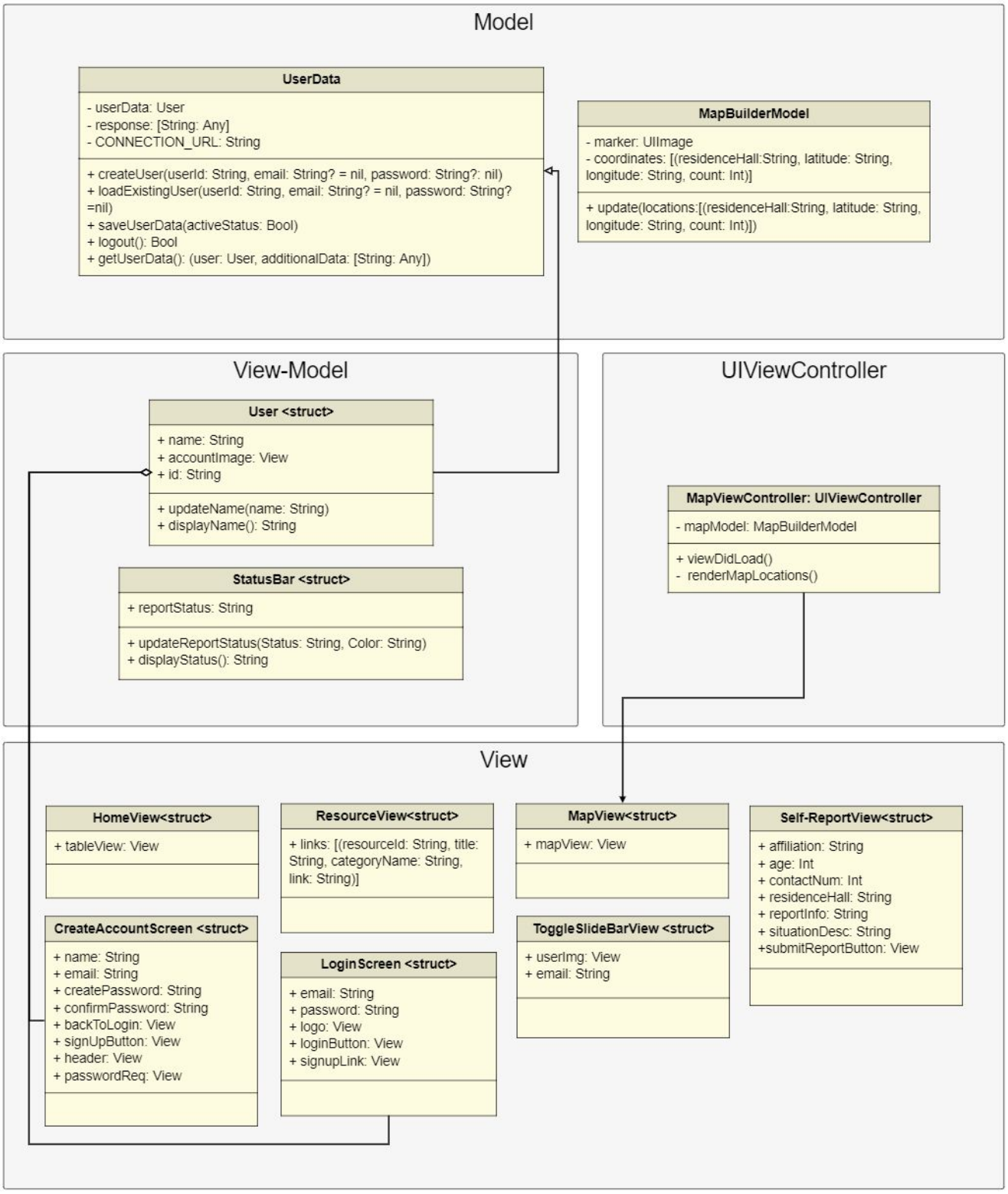
Entity Relationship Model (E-R Model)



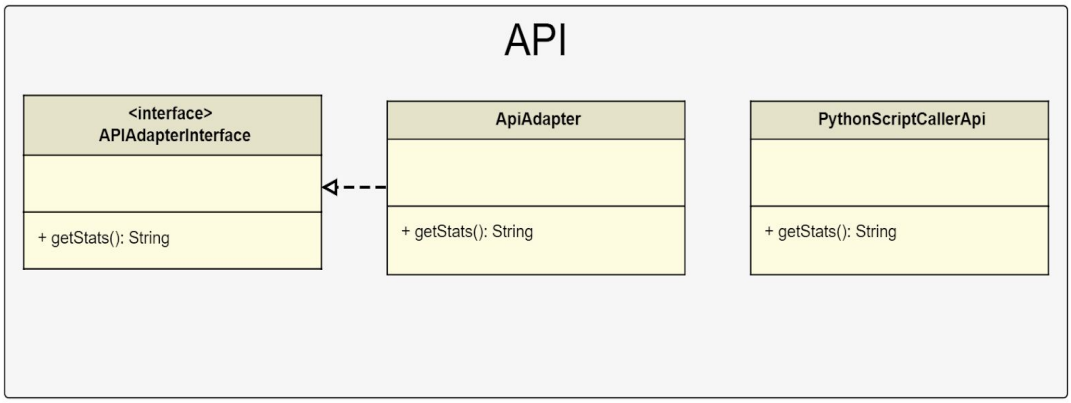
A visual representation of the structure of data defined in the database. Out of the six entities, Admin has the most relationships and may directly interact with the Resources, Alert, Map_Locations, and Report entities. Admins are identified by their unique adminID and verified with their associated email & password. The User entity is related to the Reports entity (which contains the most attributes) by the Submits relation. Users are identified by their unique userID and likewise verified with their emails and passwords. These two main entities make up the relations that connect to other entities and their attributes. Total participation of an entity is denoted by the double lines.

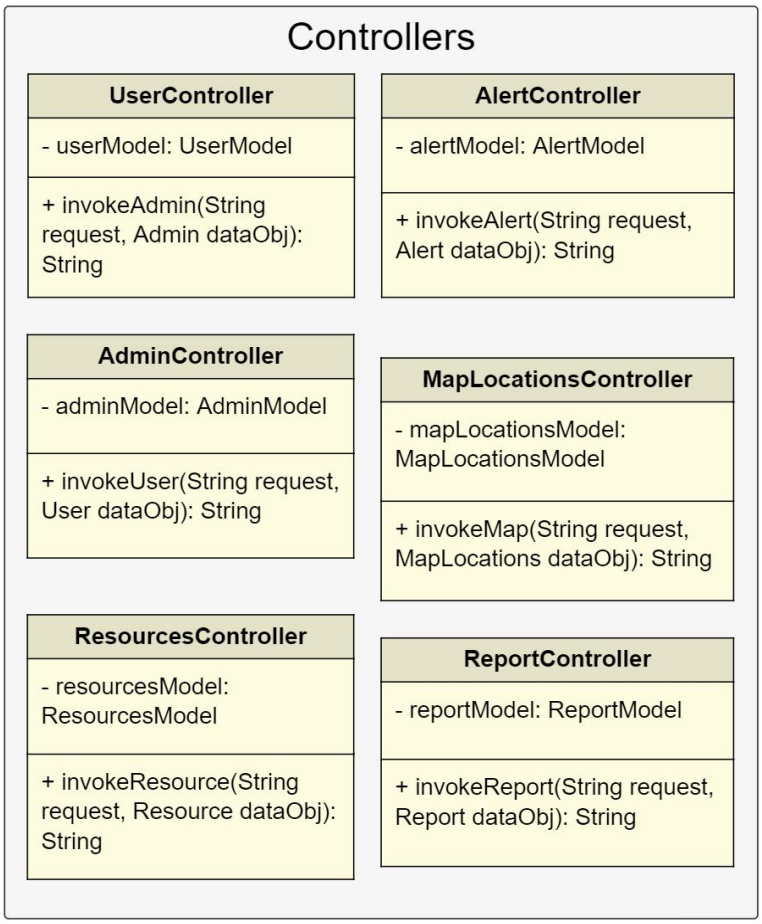
UML Class Diagram

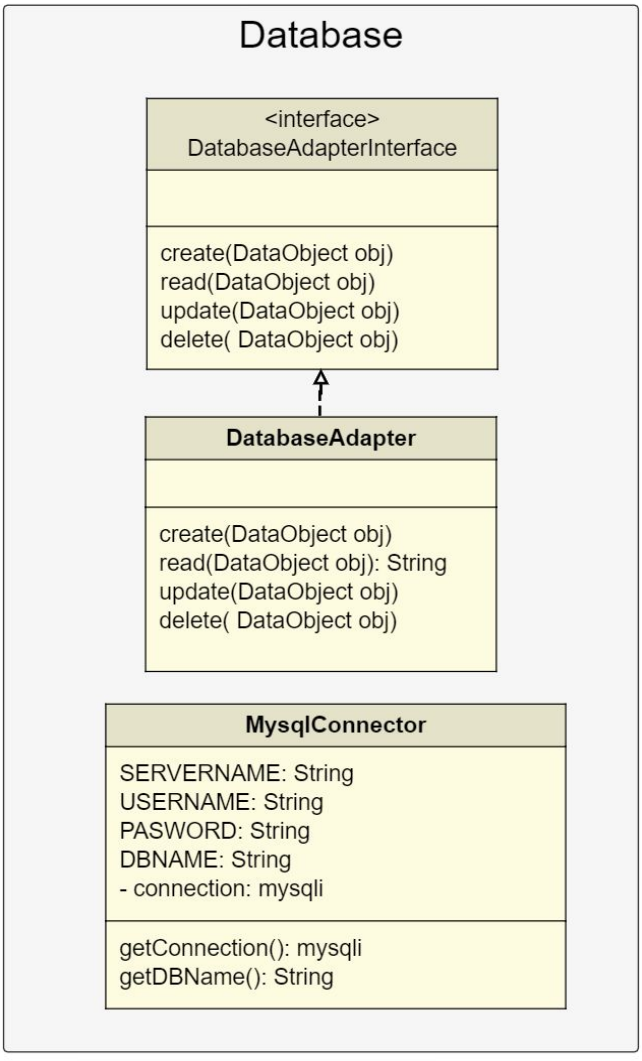
Swift (MVVM)

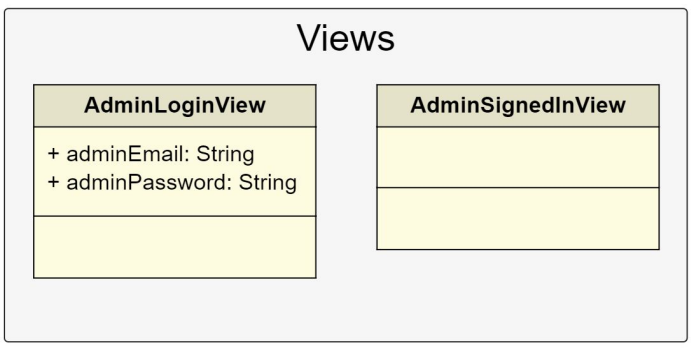


PHP (MVC)


API
<interface>
APIAdapterInterface
+ getStats(): String
ApiAdapter
+ getStats(): String
PythonScriptCallerApi
+ getStats(): String


Controllers
UserController
- userModel: UserModel
+ invokeAdmin(String request, Admin dataObj): String
AlertController
- alertModel: AlertModel
+ invokeAlert(String request, Alert dataObj): String
AdminController
- adminModel: AdminModel
+ invokeUser(String request, User dataObj): String
MapLocationsController
- mapLocationsModel: MapLocationsModel
+ invokeMap(String request, MapLocations dataObj): String
ResourcesController
- resourcesModel: ResourcesModel
+ invokeResource(String request, Resource dataObj): String
ReportController
- reportModel: ReportModel
+ invokeReport(String request, Report dataObj): String


Database
<interface>
DatabaseAdapterInterface
create(DataObject obj)
read(DataObject obj)
update(DataObject obj)
delete(DataObject obj)
DatabaseAdapter
create(DataObject obj)
read(DataObject obj): String
update(DataObject obj)
delete(DataObject obj)
MysqlConnector
SERVERNAME: String
USERNAME: String
PASWORD: String
DBNAME: String
- connection: mysqli
getConnection(): mysqli
getDBName(): String

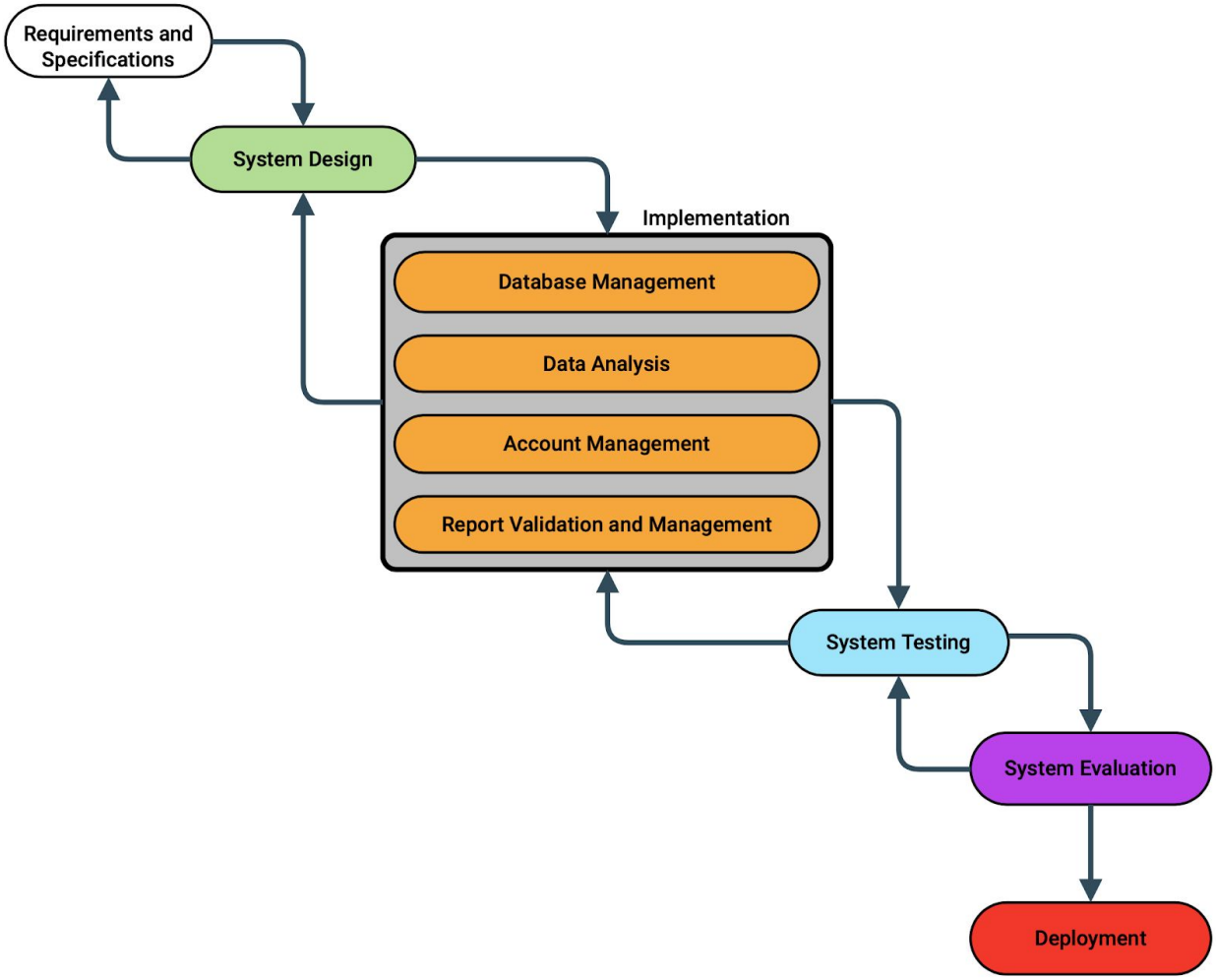

Views
AdminLoginView
+ adminEmail: String
+ adminPassword: String
AdminSignedInView

Data Dictionary

Table Name	Attribute Name	Data Type	Description	References	Key
admin_account	adminID	VARCHAR	The unique ID of the admin		PK
admin_account	email	VARCHAR	The email address of the admin		
admin_account	password	CHAR	The password of the admin		
admin_account	signedIn	ENUM	Set ‘true’ if admin is signed in		
alert	alertID	INT	Unique ID assigned to each specific alert		PK
alert	title	VARCHAR	The header of a specific alert		
alert	message	TEXT	The content that an alert contains		
map_locations	locationID	VARCHAR	Unique ID assigned to each residence hall location		
map_locations	residenceHall	VARCHAR	The name of the UNCG residence hall		
map_locations	latitude	DECIMAL	The latitude coordinate of the residence hall		
map_locations	longitude	DECIMAL	The longitude coordinate of the residence hall		
report	submitterID	VARCHAR	The userID assigned to each report	user_account (userID)	FK
report	timeStamp	TIMESTAMP	Records `Y-m-d H:m:s` using `current_timestamp()`		
report	age	INT	The age of the user		
report	phoneNumber	VARCHAR	The phone number of user		
report	affiliation	VARCHAR	Whether the user is a faculty, student, or staff		
report	locationID	VARCHAR	Unique ID assigned to each residence hall location		PK
report	reportStatus	ENUM	User chooses either ‘confirmed’ or ‘unconfirmed’		

report	reportInfo	VARCHAR	Information regarding the choice/type of report submitted		
report	situationDesc	VARCHAR	Additional information describing the nature of a report		
report	confirmerID	VARCHAR	The ID of the admin who validated the report		
resources	resourceId	VARCHAR	The unique identifier for a specific resource		PK
resources	title	VARCHAR	Title of external link displayed to user		
resources	categoryName	VARCHAR	The keyword that each link in the catalog shares		
resources	linkResource	VARCHAR	The catalog of related hypertext URL		
user_account	googleID	VARCHAR	The unique ID of the user's registered Google account		
user_account	userID	INT	The unique ID of the user		PK
user_account	email	VARCHAR	The email address of the user		
user_account	password	CHAR	The password to the users account		
user_account	userData	TEXT	The first and last name, preferences & settings of the users		
user_account	signedIn	ENUM	Set 'true' if user is signed in		

Development Methodology



Algorithm Analysis

- Time complexity breakdown by subsystems:
    - **Database Management** - The expected time complexity for simple querying, is generally *O(n)*. The time complexity to access an index would be *O(1)*.
    - **Data Analysis** - Calculation of simple statistics is expected to take linear time, but sorting brings the expected complexity to *O(n log n)*.
    - **Account Management** - Creating, deleting, and editing/accessing an account is expected to be *O(1)* time complexity. However, searching for an account would require *O(n)* time complexity.
    - **Report Validation and Management** - Operations such as creating, deleting, and editing a user's report would take *O(1)* time. Displaying all user reports is expected to be *O(n)* time complexity.

      The overall system is expected to have a time complexity of *O(n log n)* mainly because of the operations that will likely be performed on the data during data analysis.

Implementation and Testing

API Fail Safe Defaults

- Making sure that the app receives default failure messages when performing unauthorized or invalid requests

API / Server Performance Testing

- Optimal Response Times for Data Requests

API Functionality Unit Tests

- Test information from application to database using test data to simulate the connectivity to the database
- Calls for data from the database that could be returned and analized
- Data response is analyzed in Python

	Create	Read	Update	Delete	Sign In Request	Sign Out Request
Admin	✔	✔	✔	✔	✔	✔
Alert	✔	✔	✔	✔		
Map Locations	✔	✔	✔	✔		
Resources	✔	✔	✔	✔		
Report	✔	✔	✔	✔		
User	✔	✔	✔	✔	✔	✔

UI Testing

- Testing the different elements of the User Interface
- Ensure that visual components are displayed correctly and perform as expected

SwiftUI Element	**Pass / Fail**
Display sign in page	Pass
Display sign up page	Pass
Display home tab	Pass
Display Google maps (maps tab)	Pass

Display custom map icons	Pass
Display resources tab	Pass
Functionality of resource links	Pass
Display settings tab	Pass
View report status	Pass