Vaccine Clinic Information and Scheduling System (VCISS)

Project Plan

Ian Parish (ip), Jiacheng Wei (jw),
Thomas Renn (tr), Xing Qian (xq) - 4-26-2021 - v1.05

Table of Contents

1. PP Revision History 1
2. Management Plan 2
2.1. Team Organization 2
2.2. Work Distribution 2
2.3. Decisions and Communication 3
2.4. Actual Distribution of Work 3
3. Schedule 3
3.1. Schedule Milestones 3
3.2. Monitoring and Reporting 4
4. Rationale 4
4.1. Build Plan Reasoning 4
4.2. Risk 5
4.3. Risk Reduction 5

1. PP Revision History

Date	Author	Description
4-12-2021	ip, jw, tr, xq	Created the initial document.
4-16-2021	ip	Updated risks following group meeting with Prof Hornof
4-20-2021	ip	Removed section 5 (Build Plan) for redundancy
4-24-2021	ip	Updated Milestones section to reflect actual work
4-26-2021	ip	Added section 2.4 to outline actual work completed by team

2. Management Plan

2.1 Team Organization

- **Project Manager: Ian Parish**
- **Assistant Project Manager: Jiacheng Wei**
    - The PM will be responsible for setting deadlines and ensuring requirements are met in a timely manner, as well as maintaining relevant documentation and coordinating work.

- **Lead Engineer: Xing Qian**
    - The lead engineer will be responsible for coordinating and delegating the development of all required components of the VCISS. A co-lead should not be necessary, as all members of the team will be involved in development of the system.

- **Lead Test Analyst: Thomas Renn**
- **Assistant Test Analyst: Ian Parish**
    - The lead test analyst will be responsible for testing module functions and developing test cases.

- **Integration Specialist: Jiacheng Wei**
    - The integration specialist will be responsible for integrating each module into the greater VCISS and coordinating with the Lead Test Analyst and Lead Engineer to ensure proper functionality.

2.2 Work Distribution

- **Ian Parish**:
    - Module: Appointment Scheduling Module
    - Document maintenance

- **Jiacheng Wei:**
    - Module: Account/Login Module

- **Thomas Renn:**
    - Module: Search/Results Module

- **Xing Qian:**
    - Module: Clinic Finder and Information Module

2.3 Decisions and Communication

Group NoMad communicated largely over a private Discord server due to many schedule conflicts and to overcome any communication barriers. No meetings were held outside of Discord; decisions were suggested and made via electronic communication.

2.4 Actual Distribution of Work

This section will serve as a description of actual contributions by each group member.

Ian Parish

- Implemented the Appointment Scheduling module and hooked up the Appointment Cancellation function created by Jiacheng Wei. Developed the respective Scheduling methods, views, and pages.
- Performed PM duties such as organizing the group work, schedule, fostering continuous communication, and maintaining documentation.

Thomas Renn

- Defined and created the Clinic and ScheduleTime models used in the database, and created a test script to generate data for use in development and showcasing.
- Worked closely with Xing to test and implement some of the filters in the Clinic Search/Results module.

Jiacheng Wei

- Implemented the User Sign up/Login/Information module and its respective methods, views, and pages.
- Implemented the ground work for the Appointment Cancellation function

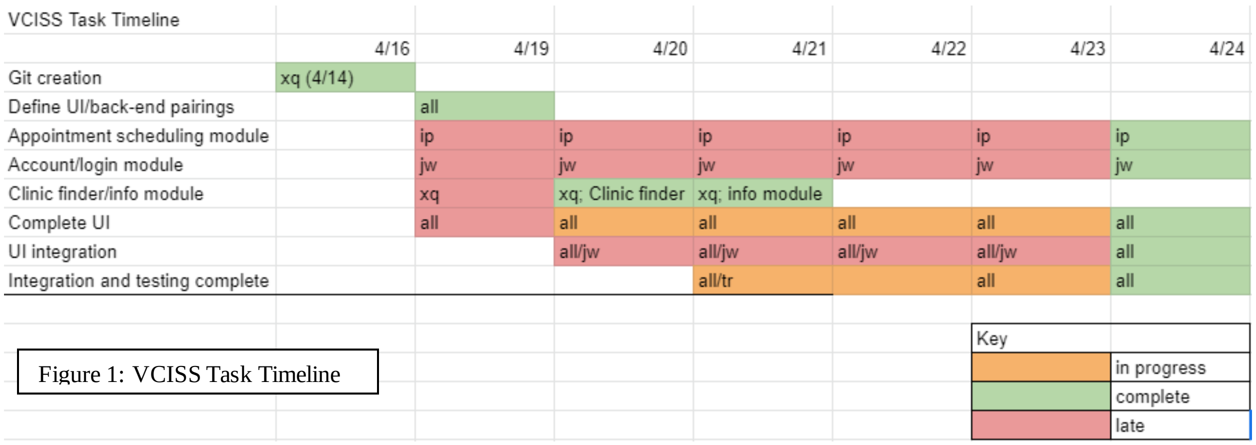
Xing Qian

- Led the team in Django framework knowledge; implemented the majority of the user interface, including the initial set up of the frontend using bootstrap library and CSS styles for the VCISS.
- Implemented the Clinic Search/Results module and its respective methods, views, and pages.
- Created initial demo of user authentication and the simple login logic.

3. Schedule

3.1 Schedule Milestones

The following chart reflects the actual productivity of team members throughout the project. It is worth discussing that much of the work was completed later than anticipated. The primary cause of this is a steeper learning curve than initial planning allowed for. Most of the group was unfamiliar with the Django framework and thus some modules were delayed; fortunately, much of this delay fell within available slack time and we were able to integrate and polish on time, as planned.



VCISS Task Timeline

	4/16	4/19	4/20	4/21	4/22	4/23	4/24
Git creation	🟩 xq (4/14)						
Define UI/back-end pairings		🟩 all					
Appointment scheduling module		🟥 ip	🟥 ip	🟥 ip	🟥 ip	🟥 ip	🟩 ip
Account/login module		🟥 jw	🟥 jw	🟥 jw	🟥 jw	🟥 jw	🟩 jw
Clinic finder/info module		🟥 xq	🟩 xq; Clinic finder	🟩 xq; info module			
Complete UI		🟥 all	🟧 all	🟧 all	🟧 all	🟧 all	🟩 all
UI integration			🟥 all/jw	🟥 all/jw	🟥 all/jw	🟥 all/jw	🟩 all
Integration and testing complete				🟧 all/tr	🟧	🟧 all	🟩 all

Key	
🟧	in progress
🟩	complete
🟥	late

Figure 1: VCISS Task Timeline

3.2 Monitoring and Reporting

Project progress was monitored via the schedule (see Figure 1), and continuous communication via a private Discord server, as well as the commit history provided by Github.

4. Rationale

4.1 Build Plan Reasoning:

We use Git for code management and code synchronization. Everyone develops in their own branch, creating pull requests when synchronization is need. Convention should be built as early as possible so that cooperation efficiency can be increased a lot.

Other things like data modeling and UI design should also be done early. Once UI design finished, team members can discuss and promote what each module exactly could and should do. And then what kind of service our backend RESTful API should provide. In this way, backend logic can be construed by the need of the frontend and divided into several modules.

Designing data schema is also important, we should pay more attention to what one kind of data should contain. Like whether the user table should include appointment information or put it into another individual appointment table. After the schema is designed, how the backend does CURD is in the obvious way.

Frontend and backend are relatively Interrelated and independent. They can be developed independently since we have many ways to mock data or http requests to debug. But we should connect them as early as possible since it can release many problems. Each module involves frontend as well as backend, connecting both and mock user behavior can test the functionality and stability of the system.

4.2 Risks

- Since all team members are newbies to the system development cycle, the main risk is the lack of knowledge to build the system. This risk leads to inconsistent team steps, so the team needs to plan and allocate work.
- A risk that was identified after our initial group meeting with Professor Hornof is that the majority of the group is unfamiliar with the Django framework we will use to implement the User Interface and API, and may cause some delay in the planned completion dates.
- Miscommunication regarding separate modules and how they should interact may hold some substantial risk and cause problems with integration.
- It is possible that some functional requirements may require change or omission due to time constraints.

4.3 Risk Reduction

The strategy adopted by the team is to build all the databases and UI/modules in one week so that the team gets another week to integrate all the parts, test the functionalities of each section, and improve the code and documentation. Therefore, the team can spend a reasonable amount of time on the team project to optimize the system. To combat the risk of potential delays caused by a lack of understanding of the selected framework, the team took 24 hours to do some research and complete tutorials, with the option (as suggested by Professor Hornof) to reconvene and select a new project approach; the team decided that it would be most pertinent to continue with the original plan. Additionally, in order to reduce the risk of integration issues, the team remained in constant communication via Discord to discuss module specifications and interrelations.

Vaccine Clinic Information and Scheduling System (VCISS)
Software Requirements Specification

Ian Parish (ip), Jiacheng Wei (jw),
Thomas Renn (tr), Xing Qian (xq) - 4-26-2021 - v1.03

Table of Contents

1. SRS Revision History 6
2. The Concept of Operations (ConOps) 6
- **2.1. Current System or Situation** 7
- **2.2. Justification for a New System** 7
- **2.3. Operational Features of the Proposed System** 7
- **2.4. User Classes** 7
- **2.5. Modes of Operation** 7
- **2.6. Operational Scenarios (Also Known as "Use Cases")** 8

3. Specific Requirements 9
- **3.1. External Interfaces (Inputs and Outputs)** 9
- **3.2. Functions** 10
- **3.3. Usability Requirements** 13
- **3.4. Performance Requirements** 13
- **3.5. Software System Attributes** 13

1. SRS Revision History

Date	Author	Description
4-11-2021	ip, jw, tr, xq	Created the initial document.
4-14-2021	ip	Updated units of measurement that did not make sense
4-25-2021	ip	Removed "account creation" use case and fixed formatting

2. The Concept of Operations (ConOps)

2.1. Current System or Situation

Our world is currently suffering under the effects of a global pandemic. Fortunately, various vaccinations have been developed and are ready for mass distribution. However, this presents a new problem - systems must be created to coordinate the distribution of the vaccines in the most expedient, efficient, and appropriate manner possible. We propose a Vaccine Clinic Information and Scheduling System (VCISS) in order to tackle this problem that will offer clinic administrators a platform to manage traffic and appointments and will offer the general public a simple system to interface with for making appointments with local clinics or vaccination centers.

2.2. Justification for a New System

The website vaccinefinder.org is partnered with the CDC to provide locations of clinics or pharmacies that are offering vaccines and list the available stock at this location. This website is not always consistent with the actual availability of vaccine stock and available appointments, which can be seen from following the secondary links to get a scheduled appointment. The VCISS should provide vaccine appointment information based on actual user data and clinic data to provide a more useful user-experience to limit the time wasted finding an available provider. The VCISS website will provide user registration and login systems to facilitate the provision of records for users. It will keep actual user records to ensure that user information is up to date along with clinic information.

2.3. Operational Features of the Proposed System

The main core feature of the VCISS system is the appointment scheduling functionality. The VCISS will have users create an account or someway to interact with the system, so that they are able to interface with the scheduling system. This system will contain records of available time slots at clinics where the user can request to see available appointments in a certain area. The users will then be able to schedule an appointment at these open time slots. This scheduled appointment will be viewable by the account that made the request and will update the entire system to remove that appointment time slot from the clinics availability.

2.4. User Classes

In the current implementation, there exists only two user classes:

- General user (members of the public looking to schedule their COVID-19 vaccination)
- Database admin

2.5. Modes of Operation

There are 2 major user modes of operating:

1) User: The user already has an account and can access the scheduling system interface.
2) Admin: The administrator can edit/make changes to the database data as necessary.

2.6. Operational Scenarios (Also Known as “Use Cases”)

There are 2 major use cases for the system:

1) **<u>User schedules an appointment:</u>**
   a. **Brief description:** This use case describes how a patient would use the VCISS to schedule an appointment to be vaccinated
   b. **Actors:** Users with access to a valid, verified account
   c. **Preconditions:**
      i. User has navigated to the VCISS main page and logged in with valid a valid account
   d. **Steps to Complete the Task**:
      i. User selects an onscreen button to navigate to the scheduler
      ii. User selects relevant filters (such as vaccine type and distance from their saved address to possible clinics)
      iii. User selects an onscreen button to initiate search and is served a results page
      iv. Of the available options, the user clicks on a link to their preferred clinic’s info page
      v. User is presented with available times and can select one of them to schedule an appointment
      vi. User receives confirmation via a confirmation page, as well as an email confirmation (or to their phone, depending on their preference)
   e. **Postconditions:**
      i. Users are successfully scheduled for a vaccination appointment at a clinic of their choosing.
      ii. Users can log in to view or cancel their appointment information

2) **<u>Patient wants to view or cancel their appointment:</u>**
   a. **Brief description:** This use case describes how a patient would use the VCISS to view information for an already scheduled appointment, or cancel the appointment
   b. **Actors:** A patient with access to a valid, verified account that has a confirmed appointment scheduled.
   c. **Preconditions:**
      i. User has navigated to the VCISS main page and logged in with valid a valid account
   d. **Steps to Complete the Task:**
      i. User selects an onscreen button to navigate to their personal information
      ii. Among the various account personalization options, the user can select on onscreen button to view their upcoming appointment(s)
      iii. The user is served a page that shows their scheduled appointments, along with basic clinic information, and appointment time/date and a cancellation button

   iv. If the user selects the cancellation button, they are served with a warning pop up message indicating that their appointment will indeed be cancelled, with accept or reject buttons
   v. If the user selects the reject button, their cancellation request will not be carried out and the pop up is dismissed
   vi. If the use selects the accept button, they are served a cancellation confirmation page, and sent an email confirming their cancellation

e. **Postconditions:**
   i. User's scheduled appointment has been cancelled
   ii. The appointment time is freed for the clinic

3. Specific Requirements

Unless otherwise specified, these requirements are absolutely necessary.

3.1. External Interfaces (Inputs and Outputs)

1) **Patient account (input/output):**
   a. The patient account is both an output of the system and used as an input to access the scheduling system.
   b. The source of the patient account is created the first time a user interacts with the system. The patient account can then be used with an appointment request.
   c. Valid ranges for this interface are verified accounts.
   d. The patient account can be verified and measured as a valid account by checking the system records.
   e. The patient account will be associated with an email address in the form of a string.

2) **Schedule request (input)**
   a. A schedule request is a request to see a list of available time slots and locations for creating an appointment request.
   b. The source of the schedule request will be sent from a user account to the system records.
   c. This input will contain a description of a geographical area, such as a state or city, along with other filterable options that the system can processes.
   d. The units of measure for a schedule request will be a response generated by the system; if the appointment is available, a response of "1" is generated and a confirmation page is served. If not available, a response of "2" is generated and an error page will be served.
   e. The data formats will be in the form of an API request.

3) **Schedule response (output)**
   a. A schedule response is the product of a request to see a list of available time slots for a specific clinic location or multiple clinic locations in a geographical area.

b. The destination of the output will the user account that sent the request. The appointment time slots will be available from a database to maintain system records.
c. This output will contain the addresses and time slots associated with the specific filtered parameters that the user entered in the schedule request to generate the schedule response.
d. The data format of a schedule response will be in the form of an API response.

4) **<u>Appointment request (input)</u>**
   a. An appointment request will be used to schedule a single appointment for a user account that contains the date, time and location of a possible appointment.
   b. The source of the appointment request will be generated by a user account.
   c. Valid ranges of inputs will be from a verified time slot, at a verified clinic location within the system by a verified user account.
   d. The appointment request can be measured by if the request was approved by the system.
   e. The date portion of this input will be in the format mm/dd. The time portion will be formatted as hh:mm in military time. The location portion will be formatted as an entity of 5 components:
      i. The numbered address
      ii. The street name
      iii. Town/city
      iv. State
      v. Zip code

5) **<u>Appointment Response (output)</u>**
   a. The appointment response can have either an approved status or denied status. If approved, the response contains the date, time and location of the appointment. If denied, gives a short description of why the request was denied.
   b. Destination of output: The user account will receive a message, where they can view it later.
   c. The valid range of outputs are approved or denied, which are determined by the processing of an appointment request.
   d. The appointment success will be measured by appointment availability.
   e. The appointment response can have either an approved status or denied status. If approved, the response contains the date, time and location of the appointment. If denied, gives a short description of why the request was denied.

3.2. Functions

1) **Account creation**
   a. Emails can be validated by using specified string formatting.
   b. Sequence for processing inputs:
      i. The user enters in all the required information they need to create an account.

   c. Error handling/response:
      i. In the case where the account creation does not work, they user will be prompted to try entering the information again.
   d. Relationship of outputs to inputs:
      i. The non-user enters in the required user information to identify distinct accounts. Once that is complete
      ii. The user then has received the output of an account that they are able to use to access the scheduling system.

2) **Generate a schedule response**
   a. This user request will be verified that it has all the required fields to pass the schedule request.
   b. Sequence of operations:
      i. User sends the request to the database via an API request.
      ii. The database queries for information given the parameters of the schedule request.
   c. Response:
      i. If there is an error in the request, there will be a response from the database describing the error. If it is with the request, the user must fix their schedule request. If it is the database has an error, then the user must wait for the problem to be resolved.
   d. Relationship of outputs to inputs:
      i. The schedule request is processed by the database.
      ii. The database responds to the user account with a schedule response.

3) **Generate appointment response:**
   a. The inputs for this function are a valid appointment request, which can be validated via the system records to make sure there is an availability at the requested location and time.
   b. Sequence of operations:
      i. The user must generate a schedule request to see available time slots.
      ii. From the available time slots, the patient must generate an appointment request that they are able to attend.
      iii. The request is then validated that there is an open availability for the appointment request.
   c. Error handling:
      i. In the case where multiple requests are made simultaneously by different users, the user making the request with the highest priority will be given the appointment. If there is still a conflict, the user with the oldest age will be given an appointment.
   d. Relationship of outputs to inputs:
      i. The appointment request is validated at above (b).
      ii. If the request is approved, this time slot is removed from the system records and given to the patient in the form of an appointment response.

4) **View appointments:**

a. The input for this is a valid account that can be verified against system records, and the user must be able to log in for security purposes.
b. The sequence for processing inputs is as follows:
   i. The user gains access to their account.
   ii. The user then sends a request to the record system to view their current appointment.
c. Error handling:
   i. In the case where a user does not revive a response, they may have to wait a certain amount of time before retrying their response as a possibility that there are to many users attempting to access this service.
d. Relationship of outputs to inputs:
   i. The inputs are validated as above in section b.
   ii. The record system then responds with the user accounts appointment time and location, if the user has set up and appointment.

5) Cancel previously scheduled appointment:

a. The input for this can be validated by checking the record system to see if a user has an appointment to cancel.
b. Sequence of processing inputs:
   i. The user's account appointment if verified by the system, to make sure it exists.
c. Error handling:
   i. The user may not have an appointment to cancel, so if the user attempts this, they will receive an error that there is no appointment to cancel.
d. Relationship of output to inputs:
   i. The input is a request to cancel an appointment of a specific user account.
   ii. The output is a response from the record system that notifies the user of a canceled appointment and adds the time slot back into the record system.

6) Filter schedule response (not absolutely required):

a. This is an intermediate step to sending a schedule response, where the filter items are already validated prior to the request entering the filter.
b. Sequence of processing inputs:
   i. The user sends a schedule request and adds the appropriate filter categories to the schedule request.
   ii. The record system then applies the filter categories from the schedule request and generates the response.
c. Error handling:
   i. The user may apply to many filters that removes all the available appointments from the schedule response. The response will let the user know that there are no available schedules for the given request.
d. Relationship of outputs to inputs:
   i. The input is a schedule request with filters.
   ii. The response is then filtered and outputs a scheduled response with only the specified filter categories.

3.3. Usability Requirements

1) 90% of the users should be able to create an account and schedule an appointment in under than 15 minutes if there are appointments available to be scheduled.
2) 95% of users should be able to cancel a scheduled appointment in under 5 minutes.
3) There should be at least 2 filterable categories for a schedule request.
4) Users should be able to view their appointment details within 5 minutes of placing an appointment request.
5) The scheduling response system should always be available during the week to see if there are available appointments (not absolutely necessary, but should be as close to this as possible).

3.4. Performance Requirements

1) Schedule requests for a small geographical location (under 100 clinic locations) should take no longer than 5 seconds to pull up available clinics within the area.
2) Schedule requests for a medium geographical location (100 to 1000 clinic locations) should take no longer than 1 minutes to pull up available clinics within the area.
3) When a user requests a filtered schedule request, the response should only include scheduling information relevant to the user's request.
4) An appointment response should take less than 1 minutes to notify the user if they received the time slot for the location or not.

3.5. Software System Attributes

1) **Account security**
    a. It is necessary that a user be confident that their personal information is not accessible or obtainable by unauthorized persons or entities, and that authorized personnel are able to obtain and modify private personal information. We must strive for 0 security vulnerabilities.

2) **Reliability**
    a. The VCISS must be able to properly and flawlessly schedule appointments for any user. If the system fails, it is due to external inputs (clinic information) and is handled elegantly

3) **Usability**
    a. All demographics of people will be interacting with our system, and as such, must be simple and approachable for a user with as little as no technical background. A simple and straightforward approach is required

Vaccine Clinic Information and Scheduling System Software Design Specification

Ian Parish (ip), Thomas Renn (tr), Jiacheng Wei (jw), Xing Qian (xq)

4-26-2021 – v1.03

Table of Contents

1. SDS Revision History 14
2. System Overview 14
3. Software Architecture 15
4. Software Modules 15
4.1. User Module 15
4.2. Clinic Search/Results Module 16
4.3. Appointment Scheduling Module 17
5. Dynamic Models of Operational Scenarios (Use Cases) 18
5.1. Account Creation 18
5.2. Scheduling an Appointment 19

1. SDS Revision History

Date	Author	Description
4-11-2021	ip, tr, jw, xq	Created the initial document.
4-16-2021	ip	Updated software modules & diagrams
4-25-2021	ip	Updated table of contents; replaced Use Case models with UML sequence diagrams

2. System Overview

The Vaccine Clinic Information and Scheduling System (VCISS) offers people a way to easily schedule an appointment to receive a vaccine. The user can use the system to determine which clinic is ideal based on which group they fall into, their preferred vaccine type, and distance. The system will be organized into a back-end and a front-end. The back-end will handle communication with partnered clinics to retrieve available appointment and location information, as well as house the logic for determining results based on user search filters and the security implementation for user accounts. The front-end will comprise of the webpages the users are served and the interface by which they will create or log in to an account, perform their clinic search, and schedule appointments.

3. Software Architecture

Much of the system architecture will be conveniently handled by the Django framework, which automates an API and much of the backend. Utilizing this framework, much of the work can be focused on the primary functional modules. These key modules of the system include:

- User module, defining authentication, records, and login/logout
    - User registration
    - User login/logout
    - User information

- Clinic Search/Results module, defining the search filter functionality and how results are to be displayed
    - Clinic search and filters
    - Clinic search results

- Appointment Scheduling Module, defining how appointments are to be managed
    - Appointment scheduling
    - Appointment cancellation
    - Handle appointment availability

Each of these modules will require its own respective User Interface and integration; each page will extend the style of the home page to ensure continuity and clarity of use.

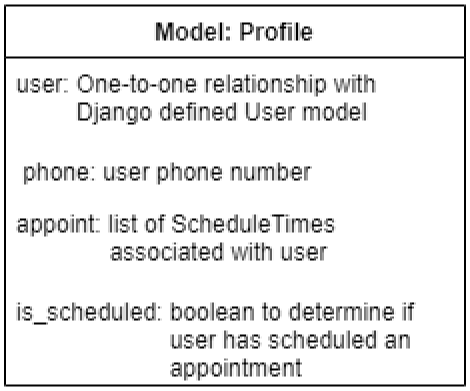
4. Software Modules

4.1. User Module

The User Module contains user registration, login/logout procedures, and handles how to display user information. Individual front-end pages exist of Login, Logout, Sign up, and User Information.

Database models:

- User; as defined by the existing Django framework

- 

Model: Profile

user: One-to-one relationship with Django defined User model

phone: user phone number

appoint: list of ScheduleTimes associated with user

is_scheduled: boolean to determine if user has scheduled an appointment

Figure 1: Profile database model, with relevant attributes

Methods:

- **signup(request):** Renders the User registration page.

- **account_signup(request):** Creates a User object and associates a Profile object.

- **account_login(request):** If user is not already authenticated, retrieves user information provided in input fields and logs the user in to the system.

- **account_logout(request):** If user is logged in, logs them out of the system.

- **account_info(request):** If user is logged in, renders User Information page with all relevant information presented.

- **edit(request, id):** If user is authenticated, allows them to edit and save their personal information.

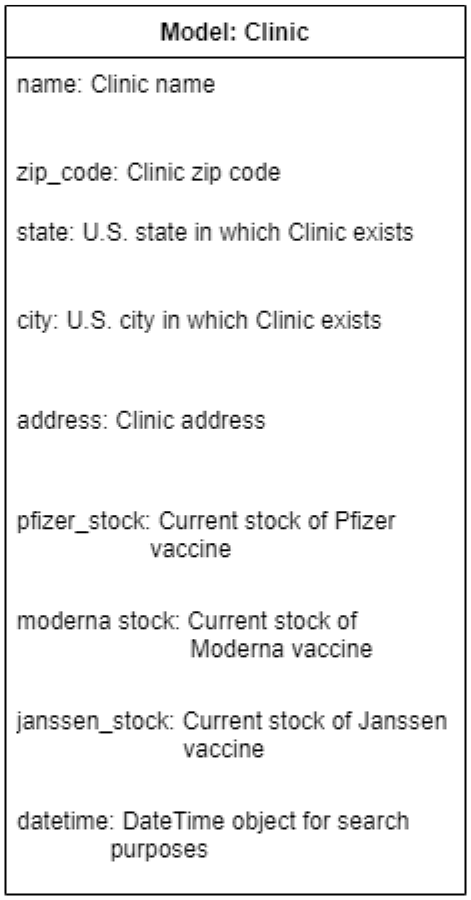
Rationale

The VCISS needs this module to recognize the unique user and provide services for logged in users. It will need to perform authentication to protect the system and private user information. All data generated by users will be bound to a unique ID in the database, where their information (such as address and appointments) is stored.

4.2. Clinic Search/Results Module

The Clinic module will provide the initial search results page that includes the list of clinics returned by the user's filtered search, and will also provide specific clinic information cards, which will also allow users to schedule an appointment for the clinic being displayed. It will interface with the database to determine relevant clinics and their availability, as well as to determine which clinics to offer as a search result, based not only on the user's filters but the location data saved to their account.

Database model:

- 

Model: Clinic

name: Clinic name

zip_code: Clinic zip code

state: U.S. state in which Clinic exists

city: U.S. city in which Clinic exists

address: Clinic address

pfizer_stock: Current stock of Pfizer vaccine

moderna stock: Current stock of Moderna vaccine

janssen_stock: Current stock of Janssen vaccine

datetime: DateTime object for search purposes

Figure 2: Clinic database model, with relevant attributes

Methods:

- **clinics_all(request):** Renders first 10 available clinics (default).

- **clinics_search(request):** Retrieves user filter options from the Search page and renders the appropriate results.

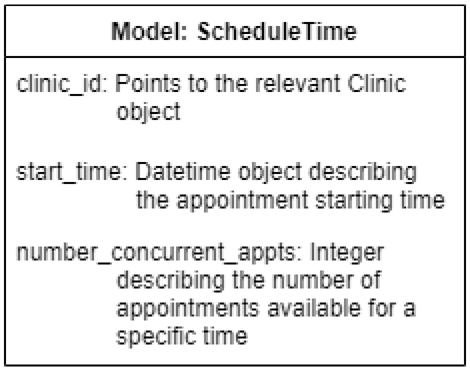
Rationale

This module is required in order to handle all requests and serve all results pertaining to clinics. It makes sense that clinics will require their own module, as user data will require different parameters (as defined in section 4.2), and likely will not be able to be handled or presented in the same manner.

4.3. Appointment Scheduling Module

The Scheduling module will house the logic for determining what appointment times are available for individual clinics. It will interface with the user interface (web pages) via the Django generated API, as well as the database to retrieve both clinic scheduling information and user appointment data.

Database model:

- 

Model: ScheduleTime

clinic_id: Points to the relevant Clinic object

start_time: Datetime object describing the appointment starting time

number_concurrent_appts: Integer describing the number of appointments available for a specific time

Figure 3: ScheduleTime database model, with relevant attributes

Methods:

- **clinic_schedule(request, clinic_id):** Renders the available appointment times for the Clinic object specified by clinic_id. Available appointments will be displayed as a clickable link; appointment times that are full will not be clickable.

- **schedule_appt(request, sched_id):** If the user is authenticated and appointments are still available for the ScheduleTime object specified by sched_id, associates the ScheduleTime object with the user's Profile and decrements number_concurrent_appts. On success, renders a confirmation page. On failure, renders a descriptive error page.

- **cancel(request, id):** Removes the ScheduleTime object specified by id from the user's Profile object. Increments the ScheduleTime number_concurrent_appts to make available for another user.

Rationale

In order to build and maintain a robust scheduling system, it makes sense to create a unique module for it. It is the heart of the VCISS and should be maintainable as a standalone service to manage scalability and reusability.

5. Dynamic Models of Operational Scenarios (Use Cases)

5.1 Scheduling an Appointment

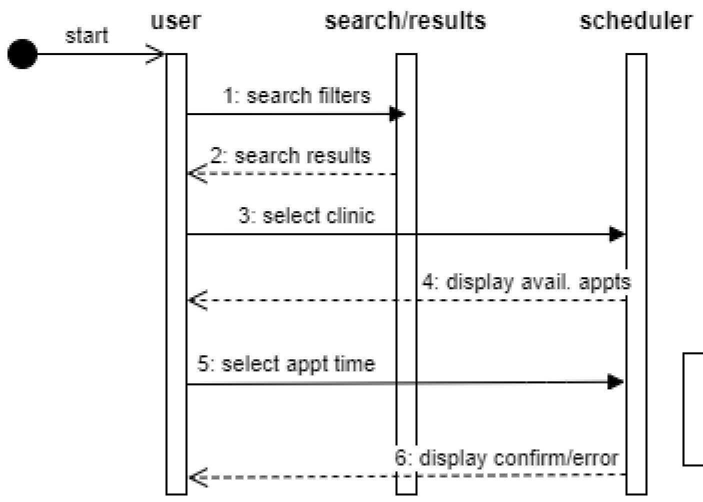


Figure 4: UML standard sequence diagram to show how a user would schedule an appointment

5.2 Canceling an Appointment

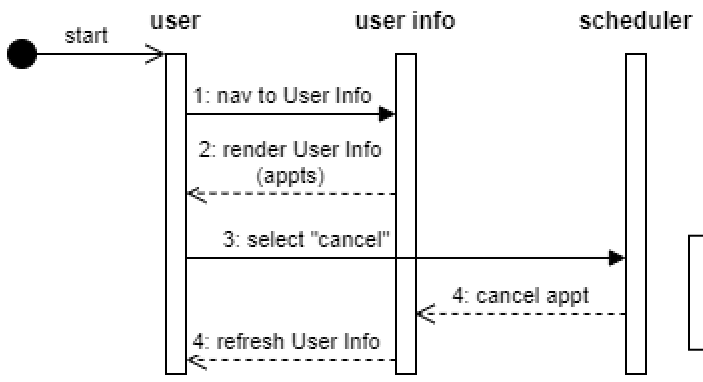


Figure 5: UML standard sequence diagram to show how a user would cancel an appointment