Criteria A Planning (416 words)

Identifying the client

The client is Yu, student from UWC ISAK Japan. He is leader of the gym club. Gym club goes to the Kazakoshi gym with the school bus every day of the week except Sunday. Due to COVID-19 restrictions, only 5 students per session can go. Yu needs to contact each student every week and decide who goes on which day. He needs to manually check attendance every session and then organize it and keep track of it. If someone misses the session, he needs to send them an email (see appendix). Because all that takes him a lot of time, he wants a system that will automate his work as much as possible. He also wants a secure solution where he will be the only one with an overview of attendance and assign a day to each member.

Proposing solution

Based on emails and the interview (see appendix), I summarised client's needs in success criteria (see success criteria), which was discussed and confirmed by the client. To meet all of the success criteria I decided to create web application using Python and Flask framework.

Justifying the solution

Web application

What leader of the club sets for the members must synchronize with members see and vice-versa. Thus, web application is the most convenient solution because the web server will provide communication between leader's device and member devices. Also, the leader and the member will be able to access the application from any device as long as they have an internet connection.

Python and Flask

Python allows me to use Flask for my development. Using Python and Flask over PHP for backend development makes my application more secure (source), which is essential for my client. Python also makes OOP considerably simplified compared to PHP which makes development less time-consuming (source). Python is one of the most popular programming languages which means a substantial amount of resources, libraries and prewritten code(source). Flask-Migrate (for my database), Flask-Login (for secure login), Flask-Mail (Mail functionality) are just a few of the functionalities that Flask offers, which I will use to meet client's requirements. I chose Flask over Django because it is simpler, lightweight, and minimalist framework. (source) . All the logic required on the back end of the application will be handled by python, which I am much more familiar with, compared to PHP or Java, which will make my development faster. This is important for the client because they will get the final product delivered in less time.

Success criteria

A. Member functionalities

1. Member can register on the website
2. Member can log in on the website
3. Members can see, which day still has free spots
4. They can choose the day to go to the gym
5. If the day is already full, they can't choose it anymore
6. Member can see which day they chose
7. When member missed a day, they get an email notification

B. Leader Functionalities

1. Can log in and see leader specific pages
2. Can see user selections and overwrite them
3. On the specific day, leader sees members who are supposed to come that day, and mark them present, excused, or absent
4. Club leader can see total number of present excused and absent for each member
5. Can sort members by their absences – ascending and descending
6. Club leader can remove students from the club

Record of tasks

Task number	Planned action	Planned outcome	Time estimated	Target Completion Date	Criteria
1	**Planning** Find client	Finding client	5 days	Sep 8	A
2	**Planning** Respond to client email	Respond to client email	15 min	Sep 8	A
3	**Planning, Design** Brainstorming and drawing sketches of the application	Have sketches of the application drawn	2 hours	Sep 10	A, B
4	**Planning** Interview with the client	Having idea of what client wants	30 min	Sep 15	A
5	**Design** Modifying sketches based on the meeting I had with the client	Have more accurate sketches	4 hours	Oct 1	B
6	**Development** Setting up the data base	Have database set up	2 hours	Partially Oct 15	C

7	**Development Testing** the data base	I can add and delete users and add or update user day selections	30min	Oct 15	C
8	**Development** Do forms and routes part of registration	Have forms and routes of registration done	1 hour	Oct 17	C
9	**Development** Finish with registration and login	Have registration and login working	2 hours	Oct 22	C
10	**Development testing** Test registration and log in	User can register, and log in	15min	Oct 22	C
11	**Development** Show different views based on user role	Show different view if normal user logs in or if admin logs in	3 days	Nov 1	C
12	**Development testing** Test if admin can see only users' section, and admin sees the 3 leader sections	Based on who logins in, different HTML templates are showed	15min	Nov 7	C
13	**Development** Create form, view function in routes and html template for users' section	Users can select the day, which updates the data base	5 hours	Nov 10	C
14	**Development** Adding logic to users' section: if day is full, users are not able to select the day	Form is not submitted if day is already full	2 hours	Nov 11	B, C
15	**Development testing**	Have user functionalities working	15 min	Nov 11	C

	Test if user functionalities are working				
16	**Development** Create schedule view function, HTML and form for leaders	Have html schedule page, with table of users and ability to overwrite user's selection in database	1hour	Nov 15	C
17	**Development** Do research, figure out how to show form in each line of the table, and send data about position of the form in the table back to the view function	Have knowledge on how to send data about form position to view function	2 hours	Nov 15	C
18	**Development testing** Make sure scheduling functionalities are working	Scheduling view is working	15min	Nov 15	C
19	**Development** Make attendance view function, html template and forms, update the models with attendance columns	Leader sees the students who were supposed to show up today, their attendance can be marked	1hour	Nov 22	C
20	**Development testing** Test attendance functionalities	Attendance functionalities are working	15min	Nov 22	C
21	**Development** Members page for leaders create view function.	List of members functionalities created	1 hour	Nov 25	C

	Remove forms, and html template				
22	**Development Testing** table of members working	Table of members is working	15 min	Nov 25	C
23	**Development** Add emailing functionality	When member is absent, applications send automatic email to them	3 hours		C
24	**Development** Revisit user day selection logic, make it more efficient	Instead of multiple if sentences application uses for loop and one if sentence	30min	Nov 31	B, C
25	**Planning,** Updating client on current development and possible improvements	Client is consulted for suggested improvements via email	20 min	Nov 31	A, B
26	**Development** Adding sort functionality	Leader can sort members by absences i	2 hours	Nov 31	A, B

Criteria B Design

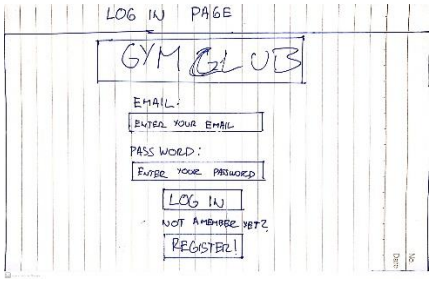


Figure 1First sketch of login page

Figure 2First sketch of registration page

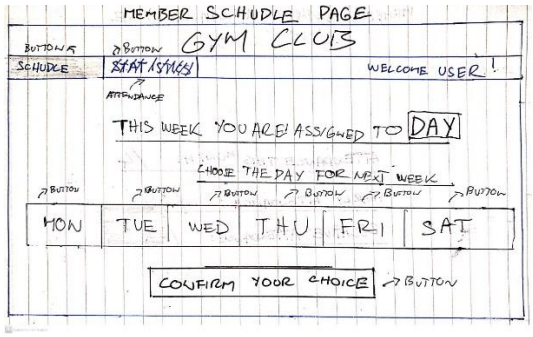


Figure 3First sketch of schedule page for members, where they see which day is assigned to them and they can choose which day they want to go next week

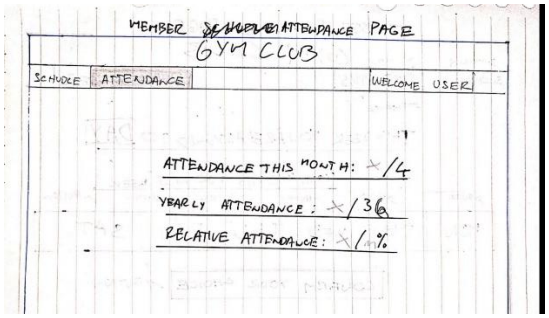


Figure 4 First sketch of attendance statistics for members (ended up not being used)

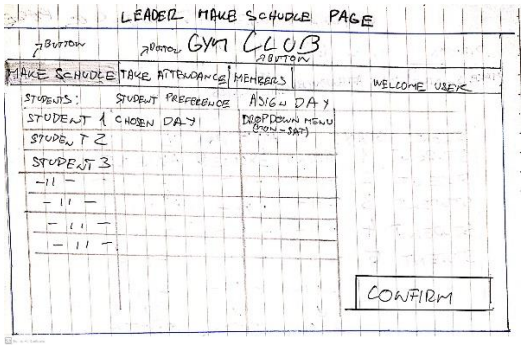


Figure 5 First sketch of make a schedule page for leader, where leader can see student preferred day and assigns day for next week to them

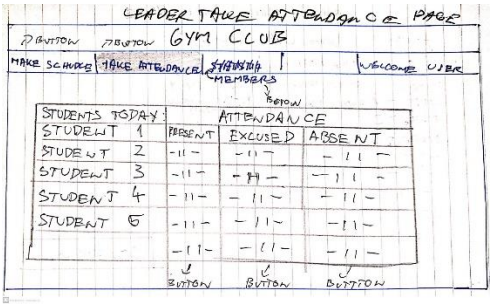


Figure 6 First sketch of take attendance page for leaders, where they can mark member present, excused or absent

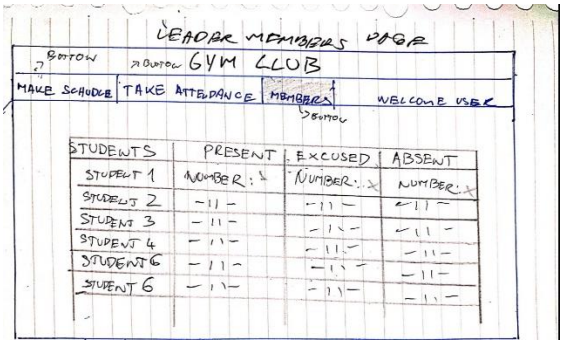


Figure 7 First sketch of attendance statistics page for leaders

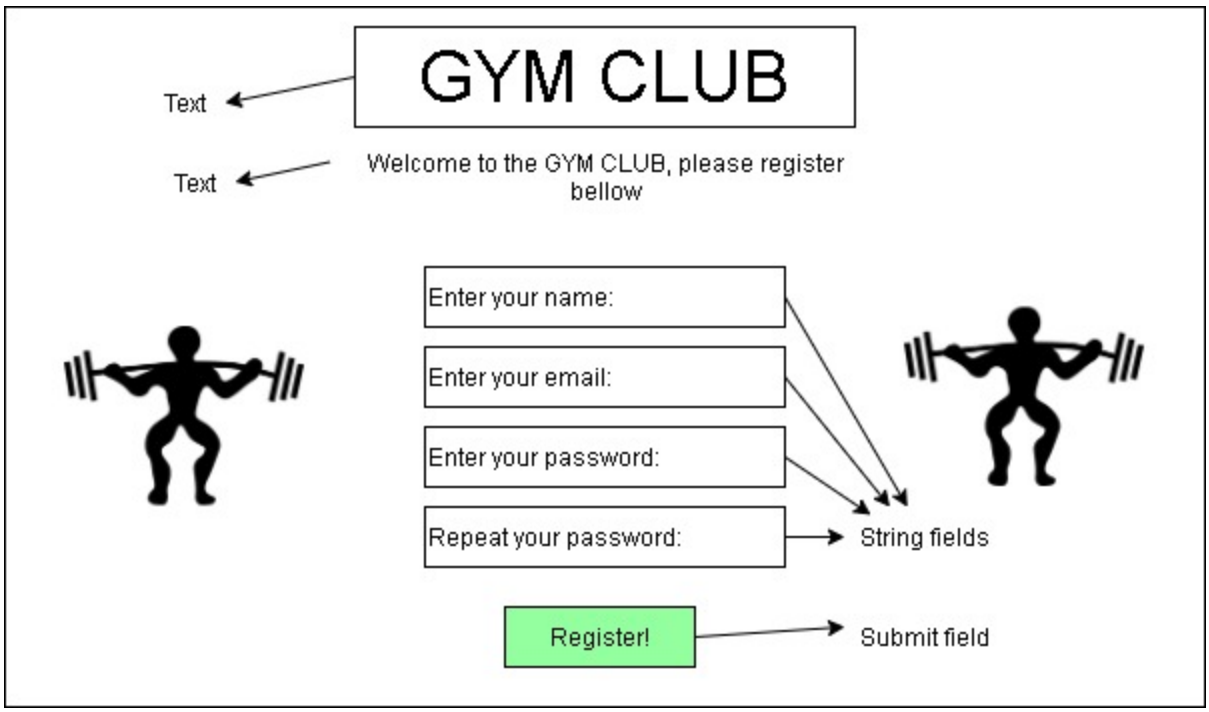


Figure 8 Design of registration page

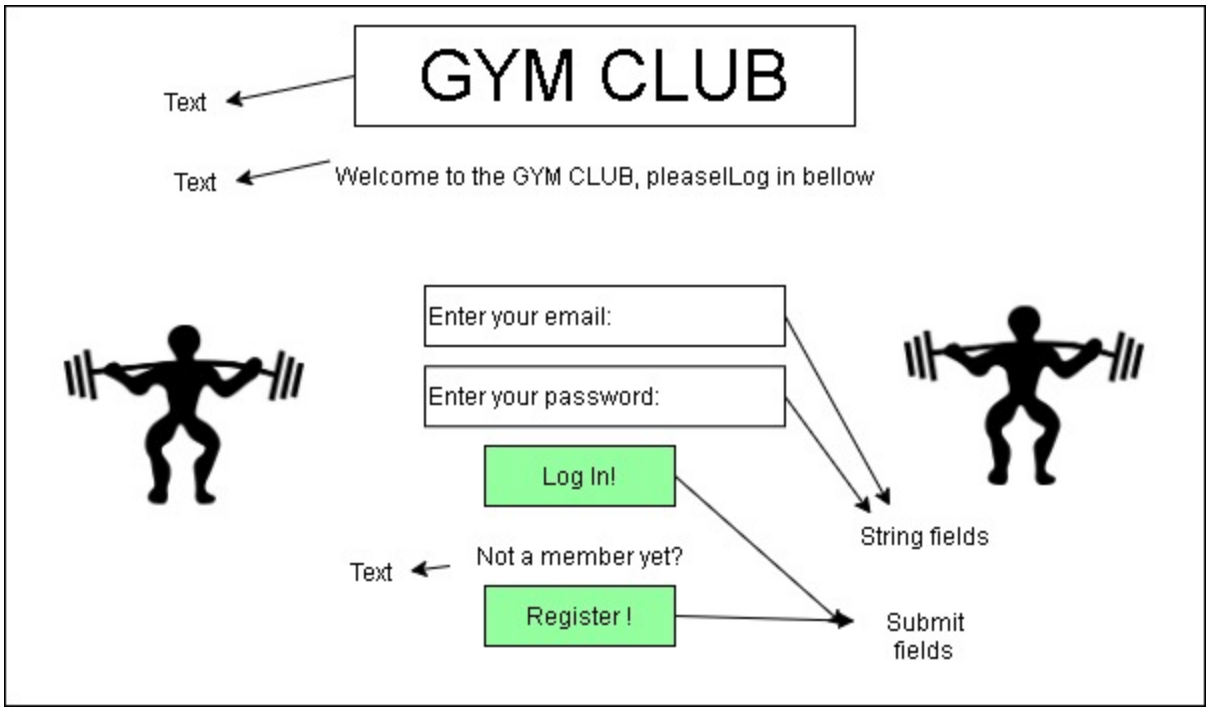


Figure 9 Design of login page

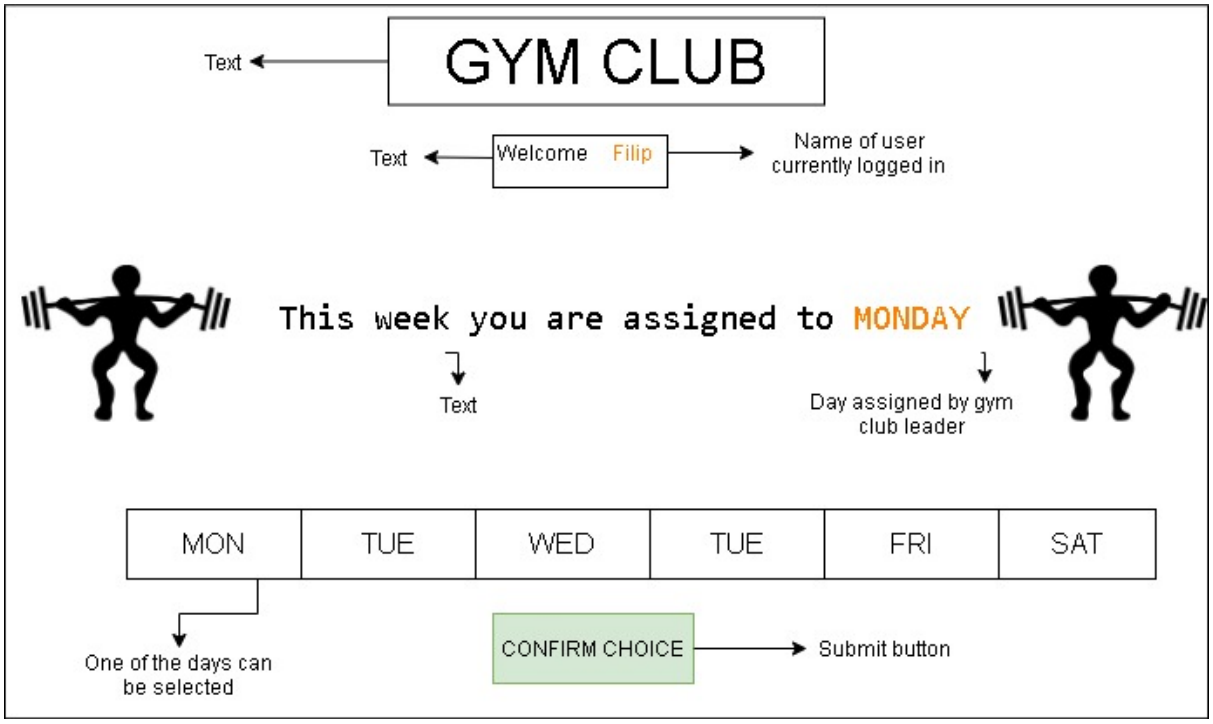


Figure 10 Design of member schedule page, where they can see which day they are going to the gym, they can also choose which day would they like to go next week

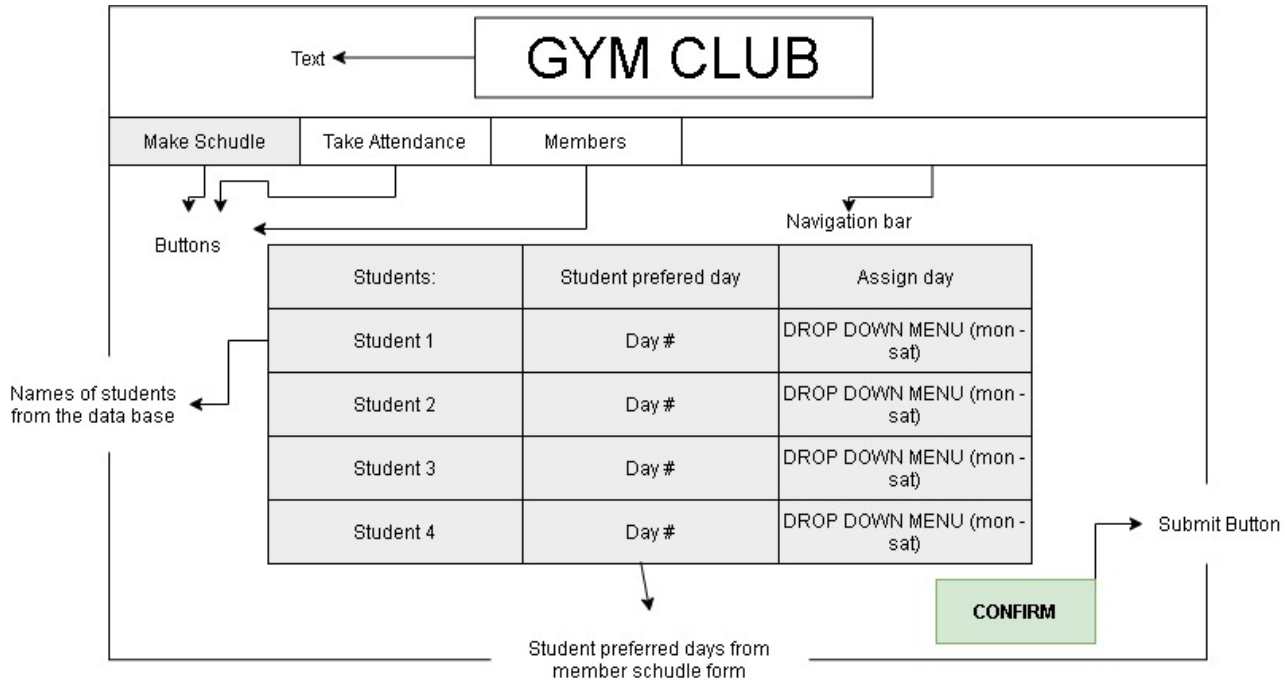


Figure 11 Design of page where leader makes schedule for next week

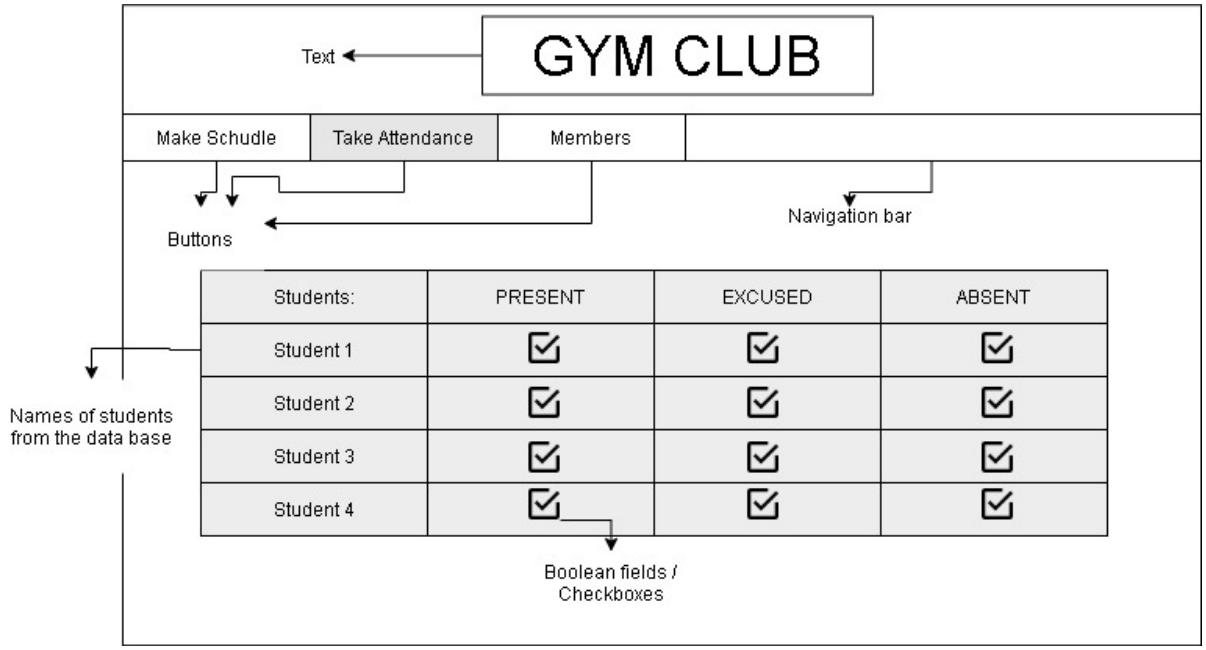


Figure 12 Design of a page where leader takes attendance on the day when club is happening

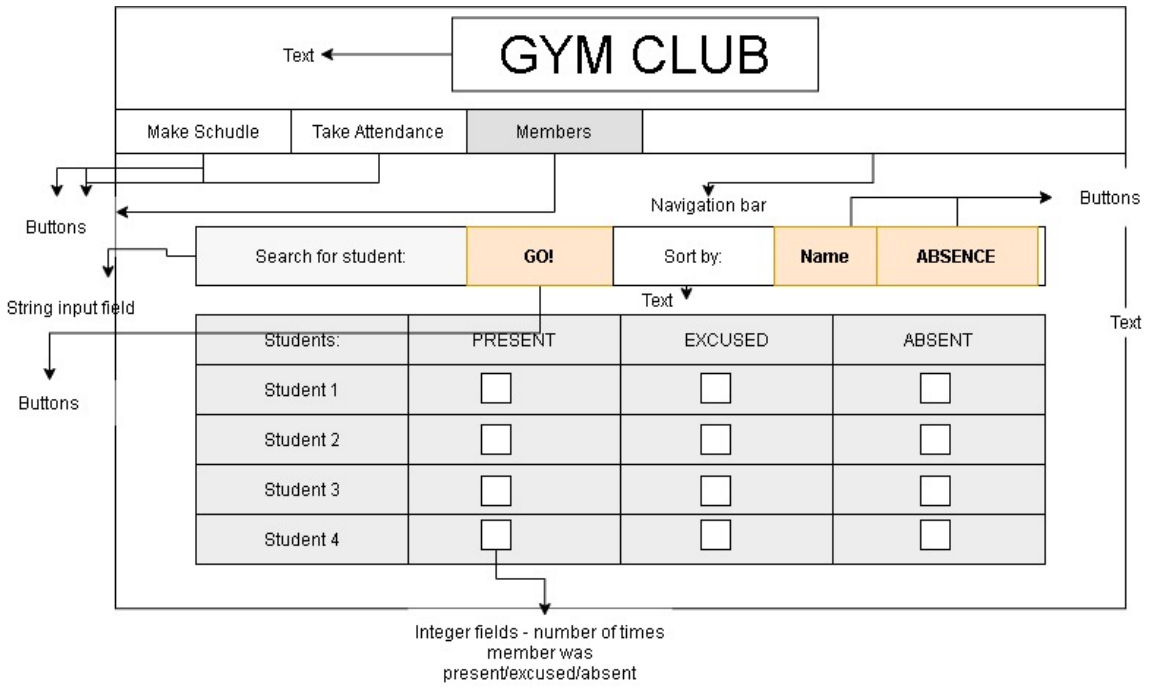


Figure 13 Design of member statistics page where leader can search for members, look at their attendance or sort them by attendance

System diagram

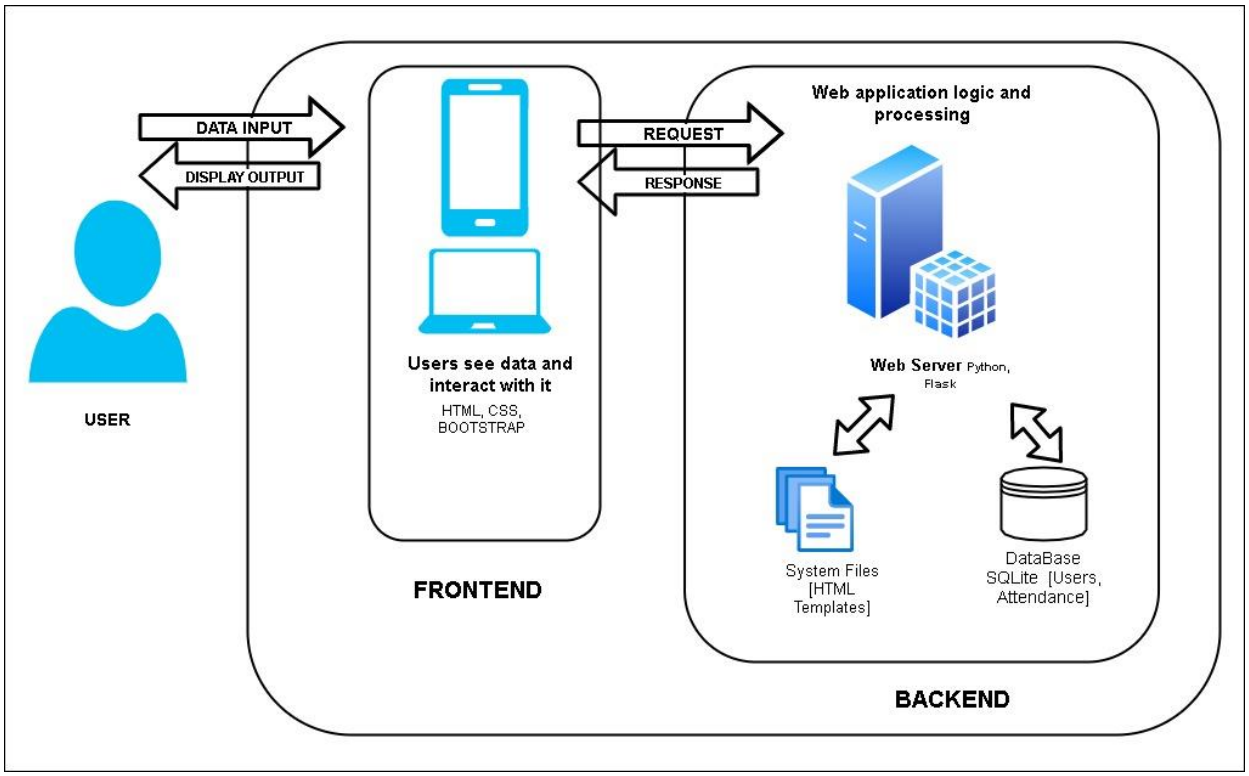


Figure 14 System diagram of my application show flow of data/information. Front end is responsible for showing information to the user, backend does all the logical operations and processing. They work together to display final output back to user

Database structure

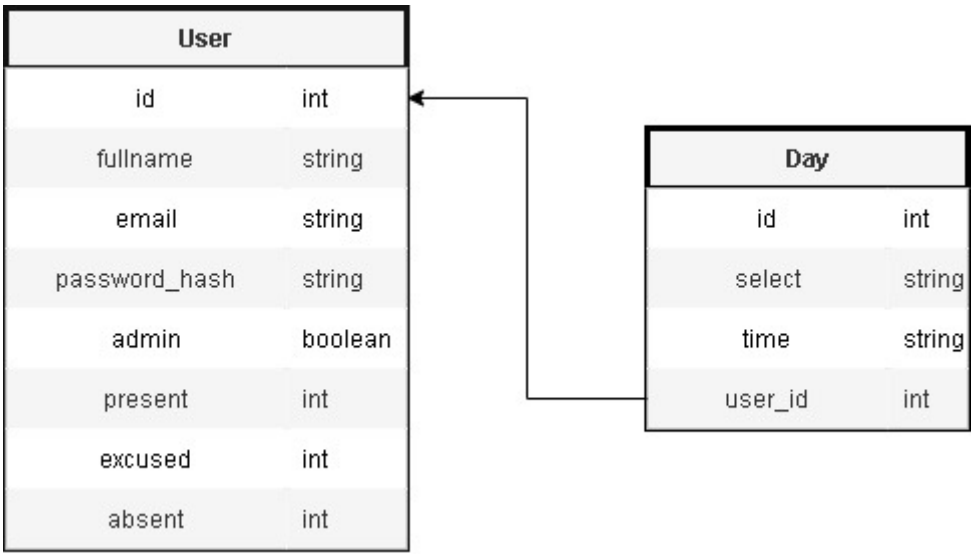


*Figure 15 **Data base structure**, consists of two models, User model where data about users is stored including user's attendance, day model is in relationship with user model, each user can choose the day that they want to go to the gym, this is connected through user id*

Flow diagram of registration

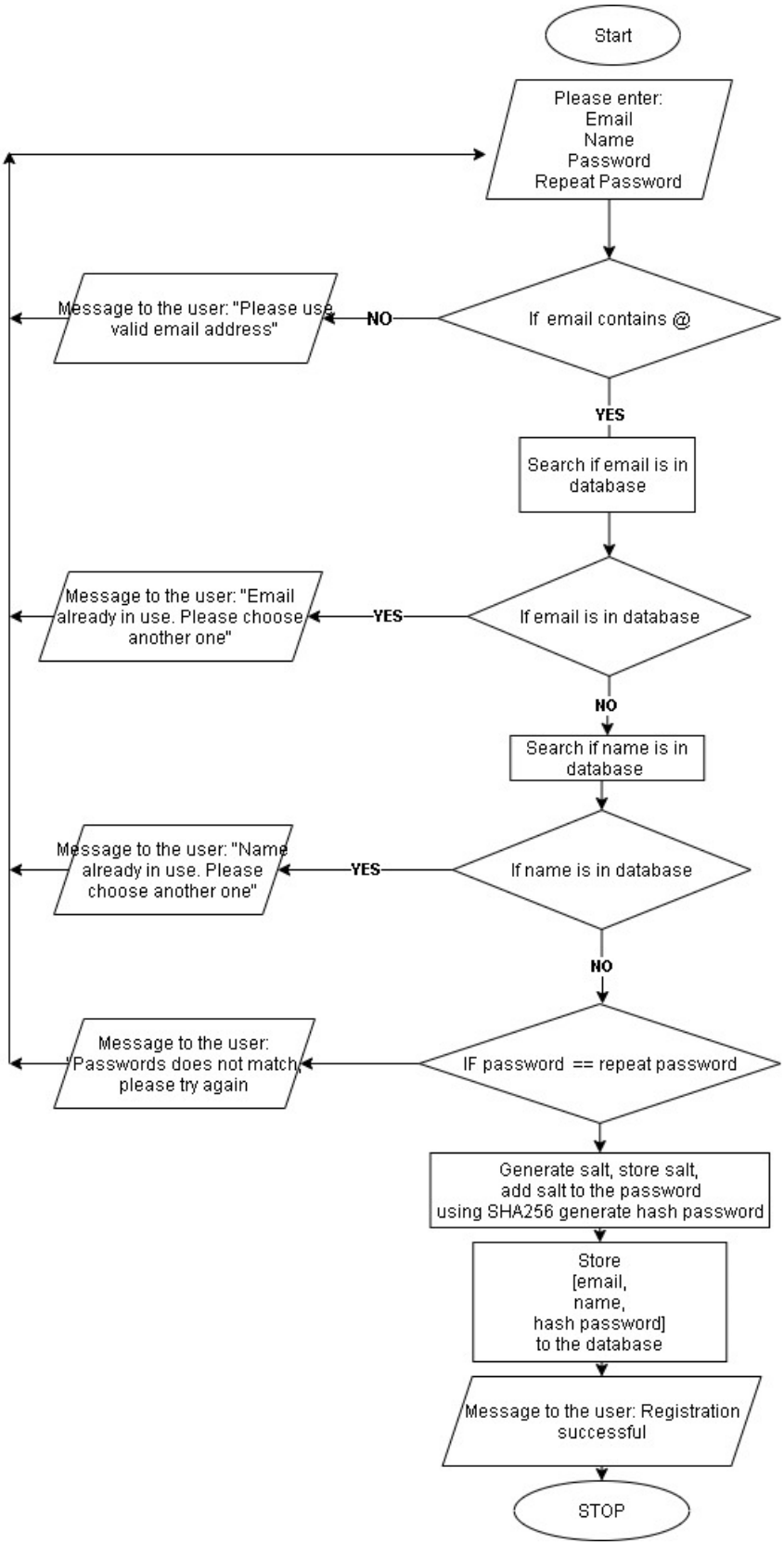


Figure 16 Flow diagram of my registration process, it shows the process of verifying registration details. New user is registered and stored into data base only if all the requirements are ok. Email needs to have @, email can't be already used by another user, name can't be already used by another user and password and repeat password strings must match, satisfies A1 success criteria

Count number of day selections diagram

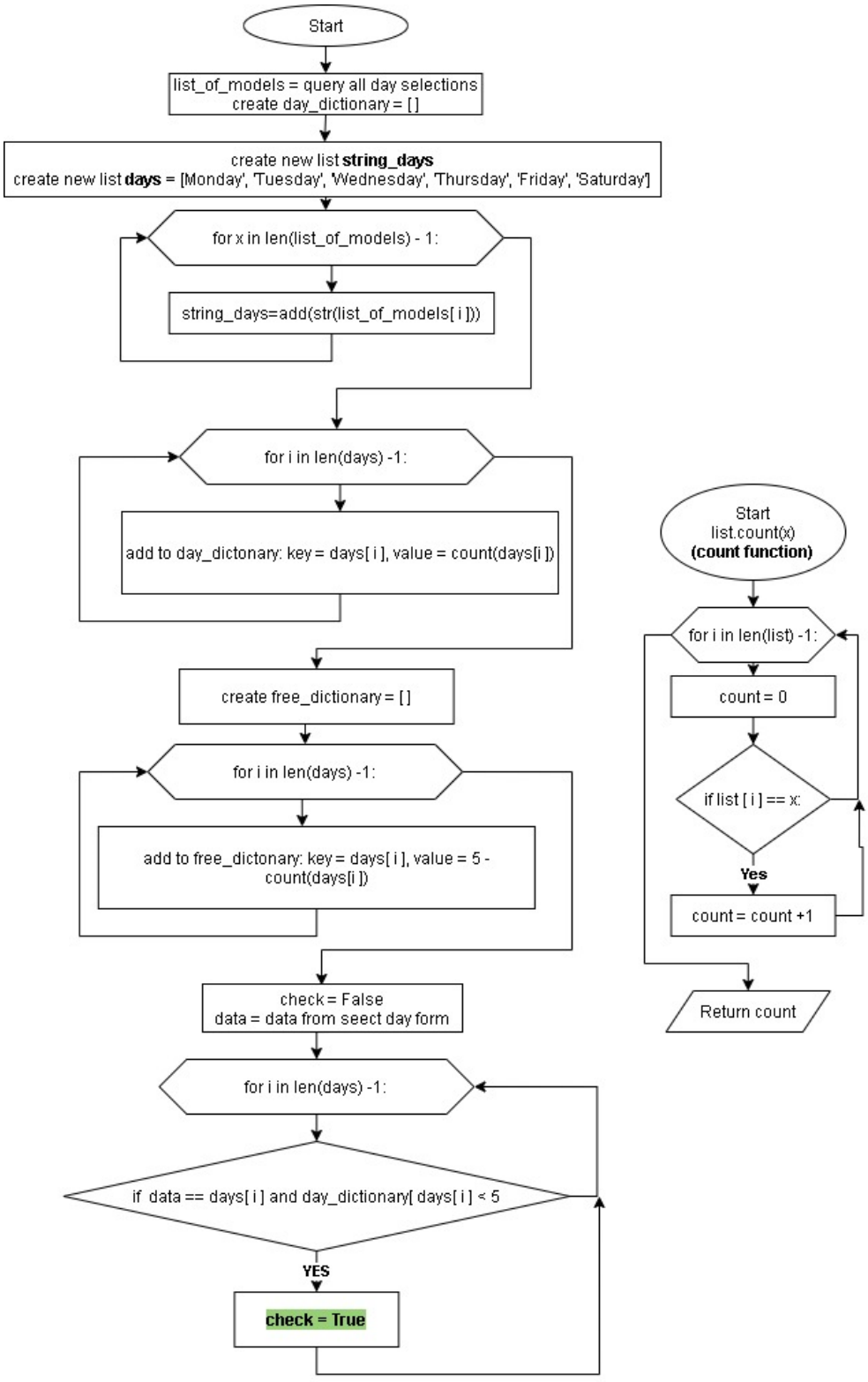


Figure 17 This diagram is doing one essential thing, to satisfy A5 success criteria. If day that user wants to choose, has already been selected more than 5 times, check variable does not change in True. Only If check = True the day selection will be stored or updated in the database. To make this check work, another smaller function is implemented, that counts number appearances of given string in a given array.

Sorting algorithm flow diagram

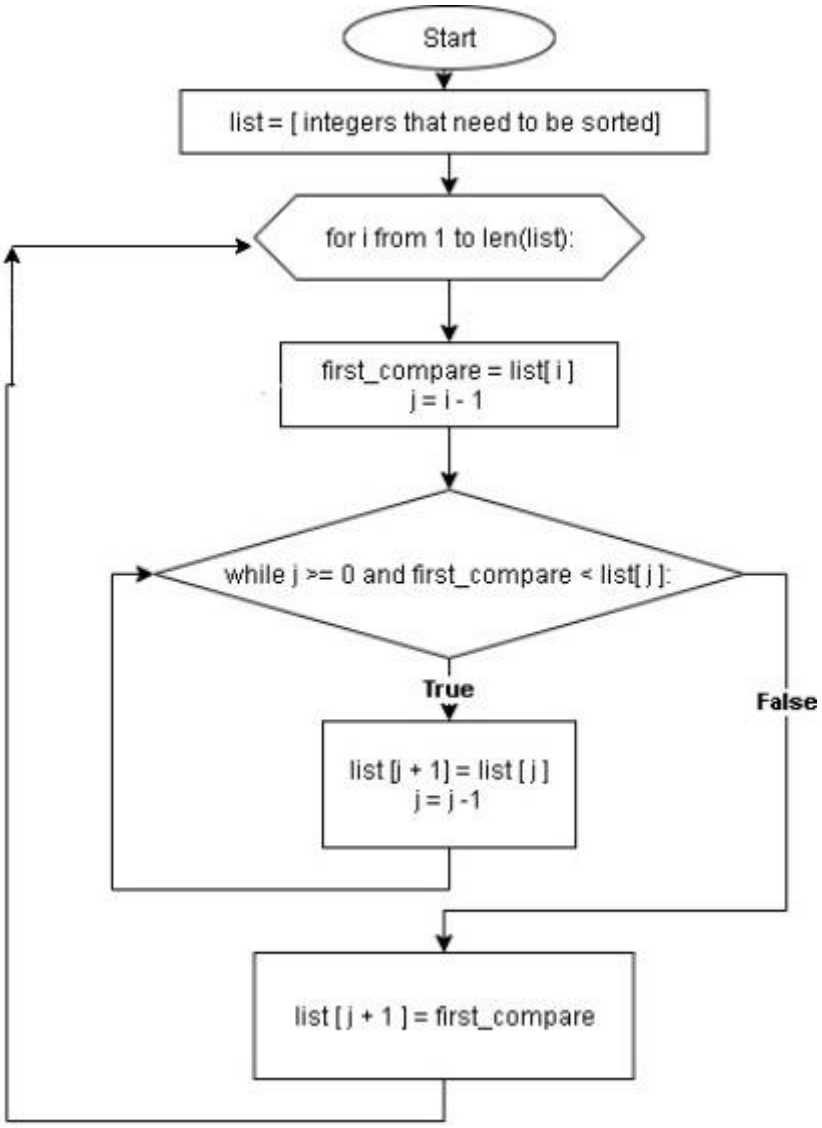


Figure 18 Insertion sort algorithm to sort the list from lowest to highest value, for loop in length of the array is initialized. Current value of the loop is compared to previous number, if its smaller it needs to be compared to every element before. All the bigger values are moved so the value being compared can go to the front. This will be used to sort the members according to absences

Test plan

Success criteria	Testing steps	Met?	Evidence
A1. User can register on the website	1. Enter email to email field 2. Enter name to name field 3. Enter password to password field		**Expected outcome:** new user gets stored into the data base **Outcome:**

	4. Enter password again to repeat password field 5. Press submit button		
A2. Member can log in	1. Enter email 2. Enter password		**Expected outcome:** Login page redirects user to the webpage **Outcome:**
A3. Members can see, which day has still free spots	1. Go to the user page 2. See the table of free spots		**Expected outcome:** Users can see how many free spots there are for each day
A4. User can choose the day of the week they would like to go	1. on the first page that opens they select one of the offered days 2. they press submit		**Expected outcome:** in leader view, leader sees preferred day that member selected
A5. If the day they chose is already full, they are not able to select it	1. Log in with 5 different demo user accounts 2. With each of them select Monday as a preferred day 3. Then log in with the 6th user and try to select Monday		**Expected outcome:** 6th user will not be able to select Monday
A6. User can see which day they chose	1. User logs in 2. Looks at the page in front of him		**Expected outcome:** it tells him the day when he is going to the gym next week
A7. When member missed the day, they get email notification	1. Go to attendance page 2. Mark one of the members absent 3. Go check that email		**Expected outcome** There will be notification on the email
B1. Leader can log in and see leader specific pages	1. Log in with leader admin account 2. Check the nav bar		**Expected outcome:** We can see leader specific pages
B2. Leader can see user selections and overwrite them	1. Go to schedule page 4. Selects a Tuesday from a dropdown menu for specific user 5. Submits the selection 6. That user logs in the application		**Expected outcome:** User should see the day that leader assigned to him
B3. On the specific day, leader sees members who are supposed to come that day, and mark them	1. Goes to member statistics view and checks present number 2. Goes to attendance view		**Expected outcome:** In member statistics view, present number should increase by 1, in the

present, excused, or absent	3. Selects that user 1 is present 4. Submits the form		attendance view leader should only see users assigned to that day
B4. Club leader can see total number of present excused and absent for each member	1. Logs in the application 2. Goes to member page		**Expected outcome:** All the members are shown. For each member number of present, excused and absent can be seen
B5. Club leader can sort member by absences	1. Leader logs in the applications 2. Goes to members' page 3. On the right top side of the table pressed sort by absences		**Expected outcome:** Users are shown in order by number of absences
B6. Club leader can remove students from the club	1. Logs in the application 2. Goes to member page 3. It presses on a user and selects remove		**Expected outcome:** All records of user are deleted from data bases

Criterion C – Development (1005 words)

Techniques used

- Algorithmic thinking
- Pattern recognition
- Abstraction
- Object oriented programming
- Application structure organization
- Creation of original templates
- Integration of components using advanced features from other applications

Firstly, to satisfy A1, A2 and B1 success criteria we need to create registration and login forms. Flask form builds form, so we don't need to do it in html.

```
16  class RegistrationForm(FlaskForm):
17      fullname = StringField('Enter your full name', validators=[DataRequired()])
18      email = StringField('Enter your Email', validators=[DataRequired(), Email()])
19      password = PasswordField('Password', validators=[DataRequired()])
20      password2 = PasswordField(
21          'Repeat Password', validators=[DataRequired(), EqualTo('password')])
22      submit = SubmitField('Register')
```

Class Registration form is inheriting from FlaskForm, which is module that builds the form. *Validator=[DataRequired(), Email()]* checks that field is not empty and that string entered is an email.

Validator=[EqualTo(‘ ‘) checks if entered string is equal to string passed in parentheses. To make sure that email and username are not already used by another user, I added two methods to the class.

```
def validate_username(self, fullname):
    user = User.query.filter_by(fullname=fullname.data).first()
    if user is not None:
        raise ValidationError('Please use a different name.')

def validate_email(self, email):
    user = User.query.filter_by(email=email.data).first()
    if user is not None:
        raise ValidationError('Please use a different email address.')
```

User.query.filter_by(fullname=fullname.data).first() queries user database, if there if already a user with the name entered in the form. If the output of query is not, so the validation error is raised. To make registration form work, we also need logic behind it.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    form = RegistrationForm()
    if form.validate_on_submit():
        user = User(fullname=form.fullname.data,
                    email=form.email.data,
                    password=form.password.data)
        db.session.add(user)
        db.session.commit()
        flash('Congratulations, you are now a registered user!')
        return redirect(url_for('login'))
    return render_template('register.html', title='Register', form=form)
```

First, I check if user is already logged in

If current_user.is_authenticated:

Return redirect(url_for(‘index’)

Then, user is added to the database session and then committed to the database. Before password is stored to the database, model is set up to hash it so all the passwords in my database are stored as hashes. `self.password_hash = generate_password_hash(password)` . Generate_password_hash is one of the functions from werkzeug security library that adds salt to the password and then hashes it.

To satisfy A3, B2, B3, B4 criteria I have to generate the table in html template. Because the number of rows is not fixed, but dynamic – based on the number of users or day selections, I could not do the table in html template manually. I had to use for-loop to generate it.

```
<table class="table table-striped table-bordered">
    <thread>
        <tr>
            <th scope="Col">Day</th>
            <th scope="Col">Free spots</th>
        </tr>
    </thread>
    <tbody>
        {% for day in list_of_days %}
            <tr>
                <th scope="row">{{ day }}</th>
                <td>{{ free_spots_dictionary[day] }}</td>
            </tr>
        {% endfor %}
    </tbody>
</table>
```

I am using template engine Jinja 2, to create html templates with Python. To implement python command into the template this syntax with curly brackets is used *{% command %}.*

To satisfy B2, B3 and B4, there is one additional requirement. Every row in the table, needs to have a form in last column. When form is submitted, change according to the position of the button in the table, needs to be made in the data base. To do that, I had to learn how to build forms natively with HTML, so I could send the data about the position back-end.

```
<td>
  <form action="{{ url_for('leader_members') }}" method="POST">
    <input type="hidden" name="fullname" value="{{ user.fullname }}">
    {{ form.hidden_tag() }}
    {{ form.remove() }}
</form>
</td>
```

Action= " " specifies where the data from the form will be sent. *Name = "fullname"* is name of the variable that will be sent and *value=" {{ user.fullname}} "* Value is this case is, one user row from User model, specifically fullname row.

To position and style elements on the webpage, I could use CSS, which would be time consuming. To make it more time efficient, I am using Bootstrap a free and open-source CSS framework.

```
<table class="table table-striped table-bordered">
```

For SC A4 and A5, before I add member day selection to the database, I need to count how many times this day was already selected by users. Representation of my Day model, is name of selected day.

```
def __repr__(self):
    return f"{self.select}"
```

When I *Day.query.all( )* I will get the list of models, to do anything with that, I need to change it to the list of strings. Because of the way representation is set for the day model, once list of models changed to list of strings, list will be filled with names of the selected days. To convert from models to strings I used for loop and *str( )* function.

```
############ COUNT THE NUMBER OF DAY SELECTIONS #############
day_list_model = Day.query.all()# get all day selections
day_list_string = []# create new list
for day in day_list_model:# transform list of models into list of strings
    day_list_string.append(str(day))
```

Now I need to find number of each day in the list. I used *count ( )* function and assigned outputted number to a different variable for each day.

```
mon_count = day_list_string.count('Monday')# count number of mondays in the list
tue_count = day_list_string.count('Tuesday')# count number of tuesdays in the list
wed_count = day_list_string.count('Wednesday')# count number of wednesdays in the list
thu_count = day_list_string.count('Thursday')# count number of thursdays in the list
fri_count = day_list_string.count('Friday')# count number of fridays in the list
sat_count = day_list_string.count('Saturday')# count number of saturdays in the list
```

I set *count_check = False.* I check what user selected in the form. If user selection is Monday, I need to check that *mon_count < 5.* If Monday was not selected more than 4 times yet, I can set *count_check = True.* I repeat this process for every day.

```
count_check = False
if form.user_selected_day.data == 'Monday' and mon_count < 5:
    count_check = True
if form.user_selected_day.data == 'Tuesday' and tue_count < 5:
    count_check = True
if form.user_selected_day.data == 'Wednesday' and wed_count < 5:
    count_check = True
if form.user_selected_day.data == 'Thursday' and thu_count < 5:
    count_check = True
if form.user_selected_day.data == 'Friday' and fri_count < 5:
    count_check = True
if form.user_selected_day.data == 'Saturday' and sat_count < 5:
    count_check = True
```

This was first version of count check. Using pattern recognition, it can be noticed that there is a lot of repeating in that code. To avoid that, dictionary and for loop can be used. To generate dictionary I use dictionary comprehension (source).

```
days_dictionary = {day:day_list_string.count(day) for day in list_of_days } #dictionary with count of day selection
```

day, is key that is going to be used in dictionary. With for loop for each day *count ( )* function counts number of appearances of that day in the *day_list_string* list of selected days. *List_of_days* is just list of strings that are going to be counted with *count ( )* function.

```
list_of_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
```

Instead of six if sentences used above which are repeating and take up 12 lines, I can use one for loop and one if sentence that take up only 3 lines.

```
for day in list_of_days:
    if form.user_selected_day.data == day and days_dictionary[day] < 5:
        count_check = True
```

For loop iterates through list of days, which is used to get each day from the dictionary and compare it against 5, for each form selection. If value is less than 5, count check is true.

When *count_check* allows applying the change into database, there are two possible cases. User haven't chosen any day yet, which means if we look for that user id in the Day model, the query will not be *None*. In that case we just add new row to the DB

```
check = Day.query.filter_by(user_id=user.id).first()
```

```
if check is None:
    check = Day(select=form.user_selected_day.data,
        time=datetime.now(),
        user_id=user.id)
```

Second case is when user already made a choice, but they are changing it – in that case, we don't add a new row to the DB, but just update the existing one.

```
check = Day.query.filter_by(user_id=user.id).first()
if check is not None:
    # if user already made a selection, just update
    check.select = form.user_selected_day.data
```

To meet SC B5 I used insertion sort algorithm.

```
############### Sorting algorithm ####################
def min_to_max(sort):

    for i in range(1, len(sort)):

        first_compare = sort[i]

        # move all the elements that are bigger
        j = i-1
        while j >= 0 and first_compare.absent < sort[j].absent :
                sort[j + 1] = sort[j]
                j -= 1
        sort[j + 1] = first_compare
```

It compares current cycle of the for loop *i* with previous one. If its bigger it stays there, if its smaller all the values that are bigger move one position forward, so the element that's currently being sorted can move to the right position.

The algorithm sorts the models from min to max. To sort them from max to min I use python *reversed()* function:

```
min_to_max(users)
users = list(reversed(users))
```

To meet SC B3 I need to show only the users who selected the current day. To find out which day is today, Python date time function is used

```
current_day = datetime.now().strftime("%A") # returns today's day as a string
```

Then all of the selections that has been filled out are compared against the current day to get the list of people who should be attendant today.

```
##### GET ALL THE USERS THAT SELECTED TODAY, PUT THEM IN A LIST########
users = User.query.all()
today_members = []
for user in users:
    if user.day_select.first() is not None:
        users_selection = user.day_select.first().select # users selection as a string
        if users_selection == current_day:
            today_members.append(user.fullname)
```

The list gets passed to html to build a table.

Appendix

Email 1

Dear Filip,

Thank you for agreeing to work with me.

As you already know, I am the gym club leader. My job is mostly organizing weekly transportation of our members to the Kazakoshi gym with school buses. We are going there 2-3 times a week.

Firstly, there are not enough spots on the bus, to take all the members there every time. For that reason, I need to contact each member of the club separately at the beginning of the week to ask them which day they would like to go. When I collect all the answers, I need to assign each student to one of the days and then contact them again to let them know which day they were assigned to.

Secondly, I am manually taking attendance in the gym. I need to keep track of it, and if someone misses the day, they were assigned to, 3 times, I need to notify them about that and kick them out of the club. Members have no way of knowing how many times they missed the club, if they are not keeping track of it for themselves.

All this is time consuming. I need a system which would automate as much work as possible.

I would like to hear your ideas and suggestions on solving that problem.

Best regards
Yu

Email 2

Dear Yu,

I am happy to help you out.

I suggest building a dynamic website. Each user will be able to log in, which would give them a personalized view of the website. If the gym club leader logs in, they can assign the gym day to each member. They can also mark the attendance for each member, and see the overall summary for each member. If the club member logs in, they can pick which day they would like to go. When the day is assigned to them, they get a message of that. They can also see which days they missed. I would also propose 2 additional improvements:

1. I can add monthly and yearly summaries to members pages, so they can keep track of their workouts.
2. I can add a warning that pops up for the members, if they miss the day they were supposed to come.

I would like to hear your feedback on my proposal. If there is anything you would like to change or add, please do not hesitate to contact me.

Best regards
Filip

Email 3

Dear Filip,

Everything you proposed sounds great.

However, do you think you could send warnings to the member via email? (not just a pop-up warning message as you proposed). To make sure we are on the same page, I would propose a meeting, where we can talk about my requirements. I think it will be easier to clarify everything in person.

Best regards
Yu

Email 4

Dear Yu,

Sorry for the late reply. Of course, we can set up the meeting, to go through my sketches and finalize them. I propose Monday evening, what do you think?

Best,
Filip

Meeting transcript

Developer
So, based on the emails that we exchanged I created a system sketches, so the 2 sketches you see in-front of you see in front of you are log in and registration
Client
That's perfect, I think I was not clear enough in the emails, but I don't want anyone else to have access to attendance data and day selection, so system must be secure
Developer
Of course, registration page - they can enter the name that will show up for you in application in their email passwords as well logging is just, they can log in if there will register if there not theirs button for them to register that looks OK right?
Client
That looks great
Developer
then when member logs into application they see that screen in front of them so by default the schedule page will show in front them they will also see navigation bar with him option to enter statistics page on the schedule they will see which day of the week is assigned to them and they will have Option to select which day they want to go to the gym next week
Client
looks great, but because of Covid restrictions we will be going there 6 times a week: every day except Sunday, and we can only take 5 people per session so for the signups can you make an option where if there's already 5 people this day the option is disabled and they have to choose another day
Developer
yeah, I can I can do that so OK everything else looks fine?
Client
Yeah
Developer
And then the attendance page they can see how many times they went to the gym this month how many times they went to the gym this year and relative attendance
Client

Yeah I think because we won't need the member attendance page since we only allow one absence due to the cancellation fees they were asking for full commitment I don't think we will needed it

Developer

Aha so I guess what your saying it doesn't make sense to include it because if they miss once they will be out

Client

Yeah, most likely

Developer

OKAY, and then there is two designs if leader logs in so the 1st 1 will be… The 1st page is make a schedule where you'll see student the student preference and the day and a drop down menu to assign the day to them

Client

OK

Developer

you will see all students for the following week. What do you think about that?

Client

Looks good

Developer

And then on the day of the gym club you will be able to mark students present excuse or absent

Client

OK

Developer

and you would only see the 5 students that are scheduled for this day will show up on your screen

Client

OK I have a question what happens when student misses the session they were required to go to and that happens quite often like do you think you can add an option where they get an automatic email if they miss once?

Developer

so when they miss a session an email will be automatically sent to them notifying them that they missed. On the last design page you will see all the members with statistics how many times they were present excused and how many times they were absent

Client

OK that's good

Developer

I think that's all, just to summarise, the bigger change is an option to automatically disable the day when it gets full

Client

Yeah that would be great!

Developer

I think that’s all thank you

Client

Thank you

Email 5

Dear Filip,

After the meeting we had yesterday, I realized that it would be handy because of the large number of members this year if you could add a search by student and sort by absences functionalities to the member's page. That would make keeping track of attendance much easier for me. Additionally, because I will might need to kick students out of the club, I need to be able to remove students that are not part of the club anymore.

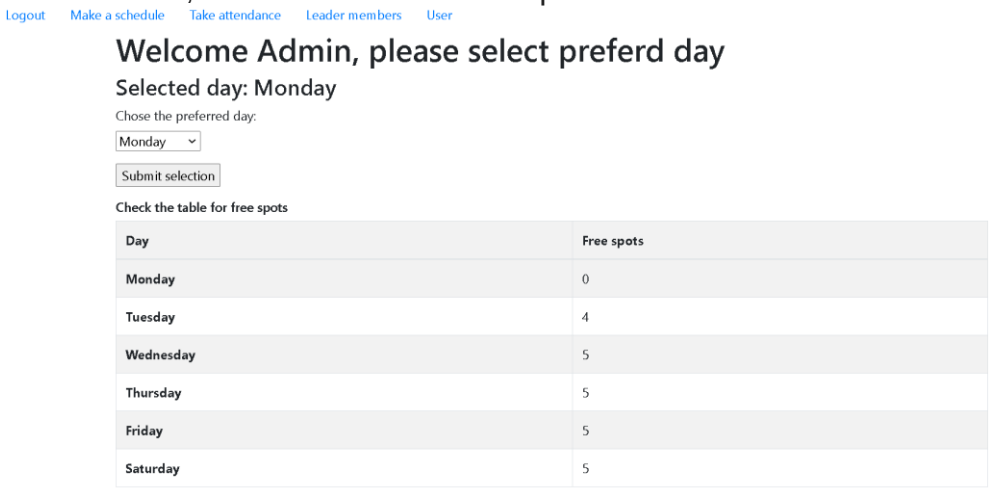
Best regards Yu

Email 6

Dear Yu,

I don't think search functionality is necessary since you have a search option in a browser where the application will be run. What are your thoughts on that?

Additionally, since members are not able to select the day that is already full, assigning them a day is not a crucial part of the application anymore, but for convenience reasons I can keep the functionality, to give you the ability to deal with the possible scheduling issues or changes. Also, since users won' be able to select days that are already full, that would leave them to guess and try all the days, so for convenience, I added a table of free spots. Picture attached below:



Please let me know if you have any questions, concerns or objections to my proposals

Best
Filip

Email 7

Dear Filip,

I am fine with not having the search functionality in members section, as long as I am able to sort them by absences.

About member day select section: yes, having an option to overwrite member selections as an admin is really needed.
Your idea about including table of free spots is great and would add to functionality of the app.
I would like to know what is your estimation on app completion, so we can start using it?

Best
Yu

Email 8

Dear Yu,

the application will be ready for beta testing in the following 2 weeks. Just to confirm that the final product will be what you had in mind, here are the success criteria, which I created based on our previous communication.

A. Member functionalities
   1. Member can register on the website
   2. Member can log in on the website
   3. Members can see, which day still has free spots
   4. They can choose the day to go to the gym
   5. If the day is already full, they can't choose it anymore
   6. Member can see which day they chose
   7. When the member missed a day, they get an email notification

B. Leader Functionalities
   1. Can log in
   2. Can see user selections and overwrite them
   3. On a specific day, the leader sees members who are supposed to come that day, and mark them present, excused, or absent
   4. Club leader can see the total number of present excused and absent for each member
   5. Can sort members by their absences – ascending and descending
   6. Club leader can remove students from the club

If there is anything missing, please let me know ASAP.

Best,

Filip

Email 9

Dear Filip,

Key functionalities stated are as discussed, and are meeting all of my expectations.

I am looking forward for user testing where I will be able to give you more feedback.

There is one little thing that I would add, do you think you could implement sort on number of present sessions as well?

Thank you for all of your efforts with the application.

Best Yu