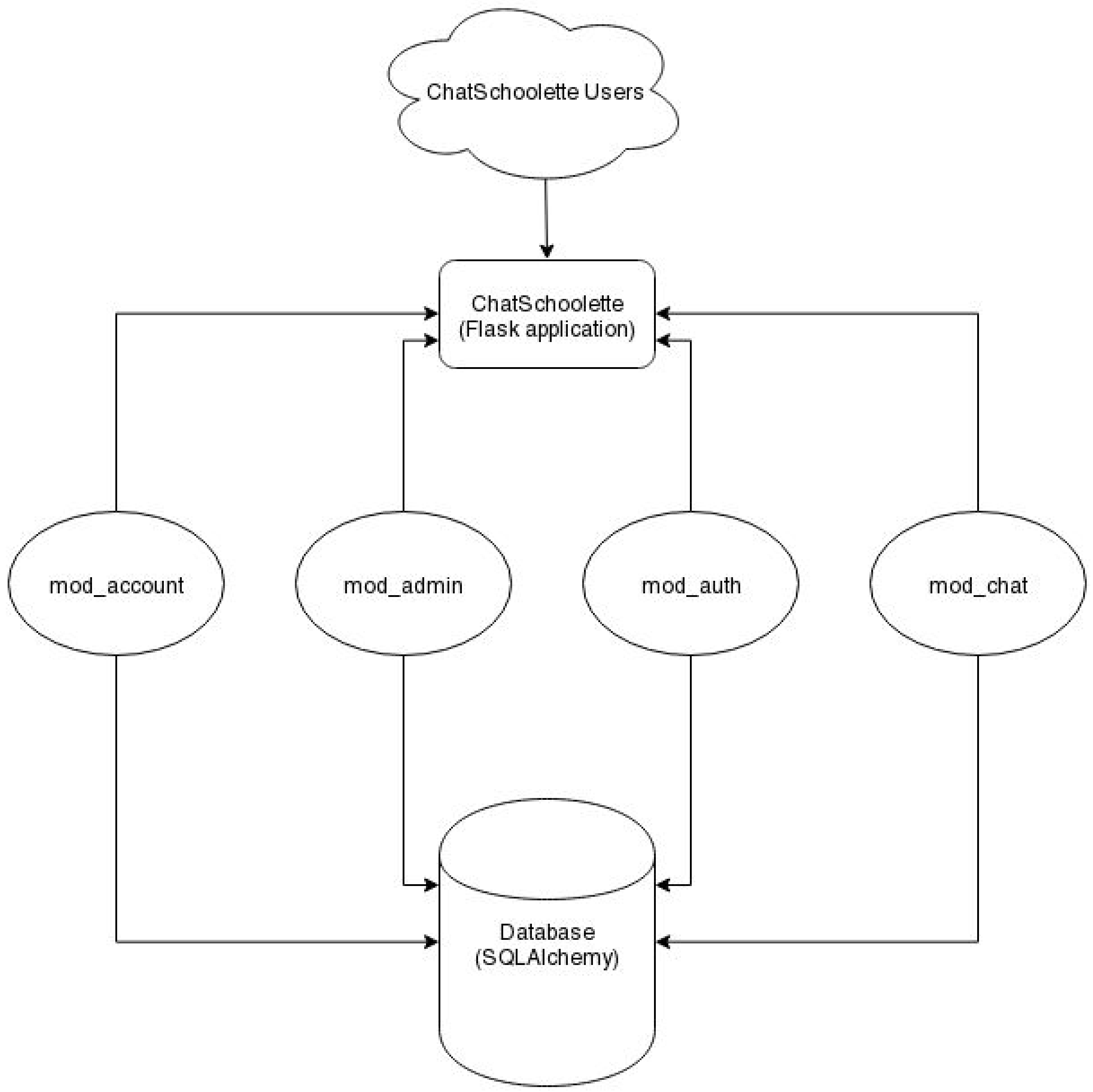
ChatSchoolette

Defect Log

Team 1: Logan Gore, Kyle Rodd, Sang Rhee, GeonHee Lee, Stephen Hong, Tyler Springer

How the Project Works

Module Diagram



Classification of Components

ChatSchoolette is comprised of four major modules: ACCOUNT, ADMINISTRATION, AUTHORIZATION, and CHAT.

The ACCOUNT module stores all of the user's data. It has methods for retrieving and updating a user's profile; this includes items like the user's profile picture, profile description, list of interests, gender, and birthdate.

The next module we have is ADMINISTRATION. This module defines what administrator accounts are authorized to do. This module defines functions for things like listing ALL site users, banning users, reading a user's chat logs to determine if they've been behaving inappropriately, or viewing a user's profile. This module is necessary to make sure ChatSchoolette remains a clean and fun environment for students to make new friends.

The AUTHORIZATION module defines all of the functions for determining if a user really is who he or she says they are. This module has functions for logging users in and out, resetting lost passwords, and loading users into sessions so cookies can be tracked for functions like "Remember Me."

The last module is CHAT. Unsurprisingly, ChatSchoolette relies heavily upon the CHAT module to function well as a website. The CHAT module defines the functions for users to request a chat partner, list their chatting preferences, and (obviously) connecting to a chat session.

All of our modules are tied together by the Python-Flask framework. If you've never seen Flask before, it basically intuitively ties everything in a project together as long as the individual Python files are defined within a module. As I just outlined before this, this is what we've done. As a result, any change to an individual submodule is automatically propagated throughout the site with no other changes required within the code. Because we are using Python-Flask, we never have to have the modules talk to each other. Moreover, our components should NOT interact with each other directly, EVER. Any resource needed by a module should be retrieved directly from the database or through the Flask API. This is the point of the modularization of the project.

We use SQLAlchemy for our database operations. This is an incredibly powerful package that automatically defines all INSERT, UPDATE, DELETE, etc. queries for you as long as you define classes that extend "db.Model". This leads us to not ever having to write or rewrite SQL code as all changes are automatically picked up by SQLAlchemy. Furthermore, if we ever want to move from SQLITE to another database, we just redefine the Database URI and SQLAlchemy can migrate everything for us.

Running the Project

1. `git clone` https://www.github.com/gorel/chatschoolette
2. `Navigate to` https://dashboard.tokbox.com/users/sign_up `to get your OpenTok API_KEY and API_SECRET (needed .`
3. `Navigate to` https://www.google.com/recaptcha/admin `and sign up for a recaptcha public/private key`
4. `cd chatschoolette`
5. `[sudo] pip install -r requirements.txt`

At this point, you'll need to create a config.py file. This should be at the SAME LEVEL as run.py. This is not in the base repository because it contains secret passwords and such that shouldn't be uploaded to Github. Below I'll paste the relevant code to put into config.py. Make sure to read it carefully to change keys and passwords as needed.

```
DEBUG = True
TESTING = True
import os
```

```
BASE_DIR = os.path.abspath(os.path.dirname(__file__))
SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, 'cs.db')
DATABASE_CONNECT_OPTIONS = {}
THREADS_PER_PAGE = 2
CSRF_ENABLED = True
CSRF_SESSION_KEY = "PLACE A RANDOM PASSWORD KEY HERE"
SECRET_KEY = "PLACE A DIFFERENT RANDOM PASSWORD KEY HERE"
RECAPTCHA_PUBLIC_KEY = "PLACE YOUR RECAPTCHA PUBLIC KEY HERE"
RECAPTCHA_PRIVATE_KEY = "PLACE YOUR RECAPTCHA PRIVATE KEY HERE"
UPLOADED_IMAGES_DEST = "chatschoolette/static/profile_pictures"
```

6. `mkdir chatschoolette/static/profile_pictures`
7. `export OPENTOK_API_KEY=your api key from tokbox.com`
8. `export OPENTOK_API_SECRET=your api secret from tokbox.com`
9. `export SITE_URL=localhost:8000 (or whatever site you host on)`

Note: The export lines will have to be run EVERY TIME you start a terminal unless you put the export lines into your .bashrc (or equivalent).

10. `python create_db.py`
11. `python run.py`
12. If you'll notice, nothing shows up… this is intended by our defects (See, we're giving you a freebie!).
    a. Right click the page
    b. inspect element
    c. expand "<body>"
    d. Uncheck "margin: 100%;"
13. Navigate to localhost:8000 (you can change the port number in run.py if you wish)

Tips and Tricks

1. We noticed that some group members had issues with activation emails being sent either very slowly (4+ hour wait time before email send) or sometimes never at all. Since the CHAT module requires your members to be active before allowing you to chat, you can run the following after you register a member to work around this:

```
python
from chatschoolette import db
from chatschoolette.mod_auth.models import User
for user in User.query.all():
        user._is_active = True
db.session.commit()
```

2. For some testing, you will probably need to have an administrator account. The process for promoting an account to have admin privileges is similar to manually setting a user as active:

```
python
from chatschoolette import db
from chatschoolette.mod_auth.models import User
for user in User.query.all():
        user.is_admin = True
db.session.commit()
```

3. A generic trick, but look into how SQLAlchemy works. We never had to write a single DB query because of SQLAlchemy's witchcraft and sorcery. Any sort of edit to the database can be accomplished by treating the DB model like an object in python (hence, ORM = Object Relational Mapping). Just remember to call `db.session.commit()`!

4. There is a bug we had that we *think* we fixed, but it may still crop up on you occasionally. Sometimes, a user gets "stuck" in a ChatRoom, and even after logging out, they can't join any new chat rooms. To fix this, do the following:

```
python
from chatschoolette import db
from chatschoolette.mod_chat.models import ChatRoom
[db.session.delete(c) for c in ChatRoom.query.all()]
db.session.commit()
```

5. You may notice that users are not getting placed into the same ChatRoom, even when no one else is using the site. Remember, this is Chat**Schoo**lette, not ChatRoulette. If the users had different email domains when registering for the site, they can't be matched. If you really want a hack around this, after each user creation, run the following:

```
python
from chatschoolette import db
from chatschoolette.mod_auth.models import User
for user in User.query.all():
       user.profile.domain = 'purdue.edu'
db.session.commit()
```

6. Keep getting locked out by Captcha? Just plain hate the system? Worried the world will discover you're actually a robot?

```
vim chatschoolette/chatschoolette/mod_auth/forms.py
:%s/RecaptchaField()/BooleanField('I hate Captcha')/g
```

List of Defects

B/W = Black/White box testing

Sev. = {1: critical, 2: important, 3: low}

#	Defect	Output Before Seeding	Output After Seeding	Suggested Correction	B/W	Sev.
1	AUTH: When registering, passwords don't have to match	Password and confirm must match	Passwords don't have to match	Add a form check that password === confirm	B	2
2	AUTH: On login, password doesn't have to be correct, as long as username exists	Username and password must match db entry	Username must be in db, password doesn't matter	Change db query to check user password if username is found	B	1
3	AUTH: All notifications sent to user have text completely irrelevant to the notification's purpose	Auth notifications have useful texts giving the user info about the notification	Auth notifications make references to semi-random internet memes	Set notification text to something relevant to the notification	B	3
4	AUTH: When registering, the account, by default, is set as admin	New user account should not be set as admin	When a new user create their account, they automatically become an admin	When initializing a new User, set is_admin=False	B	1
5	AUTH: User have to be 35 and older to be able to register	User have to be at least 18 to be able to register	New user have to be at least 35 to register	Change the restriction from < 35 to < 18	W	2
6	ADMIN: User have access to admin control	They should see "Sorry! You don't have permission to view that page" alert message.	User have access to admin control	if not current_user.is_admin, display error message and redirect them to main page.	B	1
7	ADMIN: Admin views banned users and him/herself	Admin only sees active users other than him/herself	Admin sees ALL users	Filter the user query WHERE user.id != current_user_id and banned = FALSE	B	2
8	ADMIN: Reset password feature doesn't check if user exist	If user does not exist, system should return an error message	System stops admin	if user is none, then it should display error message and stop preceding..	B	3
9	ADMIN: When reset password function is activated alert message print out wrong user name	System prints out wrong attribute of user.	System should print out correct user name.	instead of printing out user.queue_position, print out user.username	W	3
10	ADMIN: Debug is set to true.	debug is on which means when program returns error, anyone	debug should set it false	debug=TRUE should be set to debug=FALSE	W	1

		have python shell access.				
11	ACCOUNT: Even if user provides a profile picture, default picture is shown	User can upload a profile picture and see it on account pages	User only sees John Cena for profile pictures	If user has profile picture, show it instead of John Cena default.png	B	2
12	ACCOUNT: Clicking “My Account” takes you to the register screen	Clicking “My Account” takes you to your account	Clicking “My Account” takes you to register screen	Change account controller to render the account template, not redirect to registration page	B	2
13	ACCOUNT: When a new user is created, they are friends with EVERY OTHER USER on the site	New users have zero friends	New users have ALL THE FRIENDS	New users friends list should be initialized to empty	B	1
14	ACCOUNT: My account page does not print “about user” section properly, but always prints out the default statement	It should properly print out an introduction of user’s account	It prints out the default statement “__ is too cool to fill out this section” when it has to be there when user did not enter anything on that section	Check if user really did not enter anything on “about user section”	B	2
15	ACCOUNT: When edit for the account, ‘other’ is the only option for gender	there should be all options for gender including male, female, others and prefer not to say	Other is the only option, so user has be other in order to change any information	add all the option from gender section when users edit their account	B	2
16	CHAT: User account activation is not required to access chat module.	User should see “You must activate your account before chatting”	User is allowed to proceed to the chat module	Add check that the current user is active before letting allowing access to chat module. Direct user to home page if not.	B	2
17	CHAT: Any user can enter a ChatRoom if they have the room id	User can only enter ChatRoom if they are assigned to it	User can enter any ChatRoom	Add check that user is authorized user for the ChatRoom	W	1
18	CHAT: User is sent to video chat regardless of if they specified text chat	User would be redirected to a video chat if they specified video and a text chat if they specified text	User is always redirected to video chat	Change the page that the user is directed to when text chat has been specified.	W	1
19	CHAT: All Chat timestamps are for Christmas morning of 1995	Chat timestamps are set to the time the chat message was sent	Chat timestamps are set to Christmas ‘95	Chat timestamps are set at the time of their sending to datetime.datetime.now()	B	2
20	CHAT: Text chat does not	Text displayed to both	Regardless of	Change html page to	B	2

	properly display text typed by user	users is whatever the user typed in the input box.	what is typed, “John Cena” is displayed as chat message	output the input message value instead of the string “John Cena”		
21	UX: Login button in top right corner says “LOGAN” instead of “LOGIN”	button says “LOGIN”	button says “LOGAN”	Correct HTML page for navbar to change to “LOGIN”	B	3
22	UX: Logout button has become disabled.	Can click the “Logout” button to logout.	Button is disabled not allowing for clicks.	Remove disabled attribute from “Logout” input tag.	B	2
23	UX: User sees Admin Controls button even if they aren’t an admin	User only sees Admin Controls button if user is an admin	All users see Admin Controls button	Only show admin controls button if user is admin	B	2
24	UX: Everything is shifted by 100% margin.	User sees page as is.	User sees items shifted.	Remove margin line from body object in css.	B	2
25	UX: Scrollbar for vertical axis is missing.	Scrolling as page sees fit to see all information.	Scrolling can no long occur, scrollbar missing.	Remove overflow y hidden from body object in css.	B	2