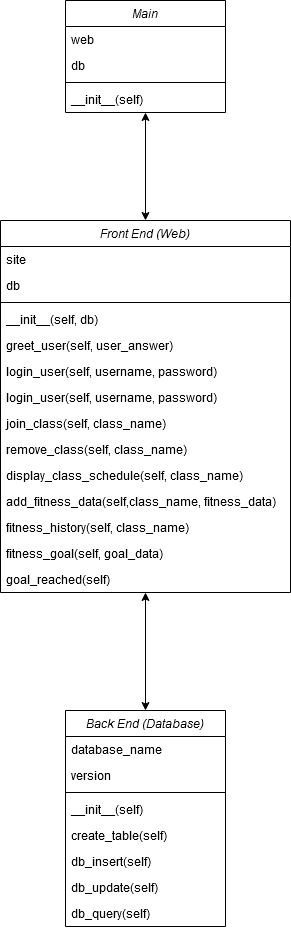
**Class Diagram**

****

**Pseudocode**

**Main subsystem:**

class Main(object):

def \_\_init\_\_()self:

instantiate database instance

instantiate web instance, passing in the database

**DB Subsystem:**

class DB(object):

def \_\_init\_\_(self):

set name of file

set version of file

def create\_table(self):

try to open file

create file if not already existing

create structure of the database in the file if not already there

close the file

def db\_insert(self):

open the file

inserts new data into the database

close the file

def db\_update(self):

open the file

update existing entry in database

close the file

def db\_query(self):

open the file

finds data requested

close the file

returns queried data

**Web Subsystem:**

class Web(object):

def \_\_init\_\_(self, db):

set up instance of the Web class

def greet\_user(self, user\_answer):

displays greeting/ask user if they would like to join the program

returns true if yes, false otherwise

def login\_user(self, username, password): **(Requirements 10001, 10002)**

asks if new or existing user

populates the login database

creates new user if requested

require password to have 1 capital letter, 1 special character, and at least 8 characters long

attempts login using credentials found in database

returns true if good login, false otherwise

def join\_class(self, class\_name): **(Requirement 10003)**

check if class exists, handle error if it doesn’t exist

add user to class in database

def remove\_class(self, class\_name): **(Requirement 10003)**

check if user is registered for the class, handle error if user isn’t registered

remove user from class in database

def display\_class\_schedule(self, class\_name): **(Requirement 10004)**

returns class schedule user has registered for

def add\_fitness\_data(self, class\_name, fitness\_data): **(Requirement 10005)**

check if user is registered for the class, handle error if user isn’t registered

update user’s fitness data

def fitness\_history(self, class\_name): **(Requirement 10006)**

check if user is registered for the class, handle error if user isn’t registered

returns user’s fitness history

def fitness\_goal(self, goal\_data): **(Requirement 10007)**

check if user is registered for the class, handle error if user isn’t registered

store fitness goal data in the database

def goal\_reached(self): **(Requirement 10008)**

displays congrats message to user upon reaching fitness goal

**Unresolved Risks:**

Unresolved risks and risk mitigation: User data and passwords will be stored in plain-text. Possible mitigation: Storing the data in an encrypted file will secure user’s information. Hashing the user’s password will help protect the user’s information.