Final Report - Team 6

Jason MacManiman
Renee McLain
Eric Miller

1. **A short introduction to the project.**
    - ***Executive summary of what you did.***

        Our team created a program for managing the contents of a postgresql database which was hosted in a Virtual Machine. The program is written in Python and makes use of the ncurses library to create a user interface. The database holds information for a bakery, the schema for which consists of tables for cakes, ingredients and recipes. The recipes table contains only foreign ID's for the cakes and the ingredients, resulting in relational storage The user can manage the contents of the database by adding, editing or deleting ingredients, cakes or recipes.

    - ***What is its importance?***

    It makes it easier for users who don't know programming to interact with a database. You don't have to worry about knowing how to write queries, you can just use the ncurses interface.

    - ***Who was/were your client(s)?***

    Kevin McGrath

    - ***What were their roles?***

    Describing the product as well as different use cases. Also answering questions the team had along the way.

2. **How did the project change since the original design document?**

Our original plan was to include customers into our database scheme but upon discussion we realized that might be too much going in, so we chose a more basic scheme to just include the business product at this time. We decided if we had time at the end we could always grow what we had, but ultimately we wanted to get our project up and running

before we added more features to it. We first created our schema and ERD which included customers, but then we cancelled that and decided to go with the basics. After re-configuring our ERD and schema, we were able to begin crafting the table creation commands for later use.

Also, at first we were thinking that we were going to host our data on Google App Engine. While researching how to get this up and running Jason emailed Prof McGrath as he felt that was not the direction we needed to be taking but instead hosting the database locally on a VM. This was indeed the case so we needed to switch directions on database management.

3. **Project documentation.**
    - ***How does your program work?***

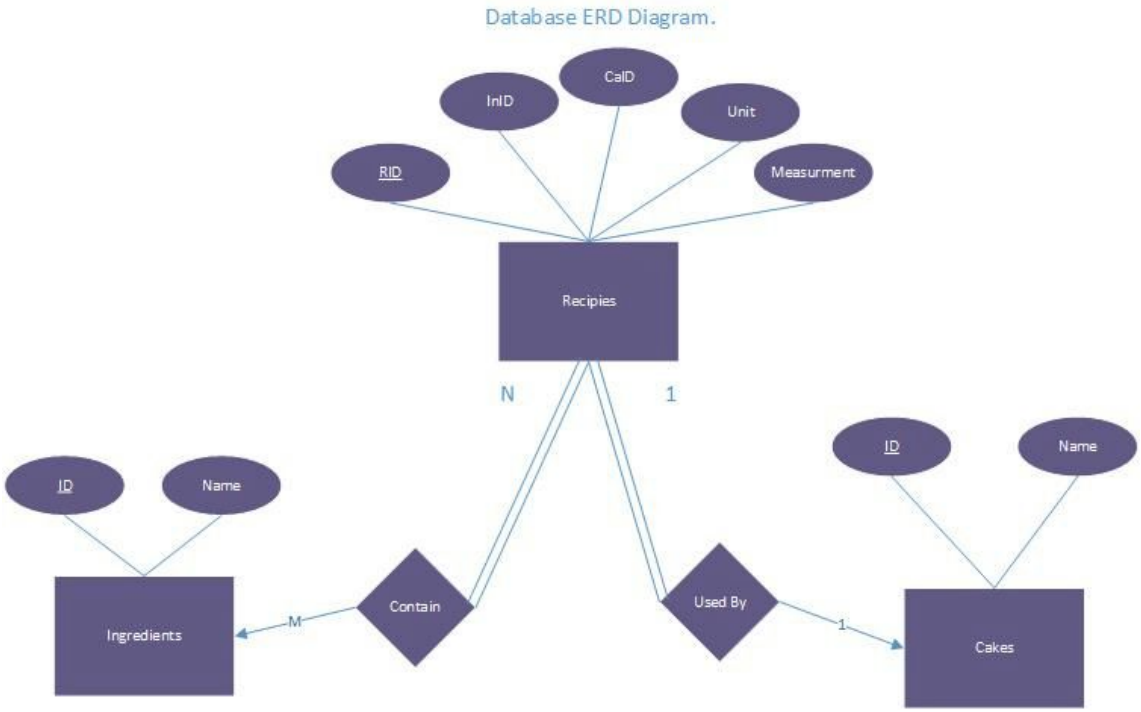
        After you set up the necessary software, have your VM running and your database created, you navigate via your terminal to the location of the program file and run the program (i.e. # python group6project.py). You are taken to the front page. Here, you can cursor through the options of choosing a cake or an ingredient.

        If you choose cake you can add a cake to the database, delete a cake from the database, or choose to look at the cake's recipe. If you choose to add a cake you are then taken to a screen where you can add the name of the cake as well as the ingredients in the cake. If you add the ingredients in the cake the sql query automatically updates the recipe for that cake. You can also choose (by cursor) to go back to the main menu.

        At the main menu, if you choose to look at the ingredients you are taking to a page where you can scroll through all of the ingredients in the database. You are given options to add or delete ingredients as well. Again, you are given the option (by cursor) to scroll back to the main page.

- ***What is its structure?***

This program is structured around creating and referencing recipes for cakes, and is centered around three element tables. These tables are for ingredients, cakes, and recipes respectively.



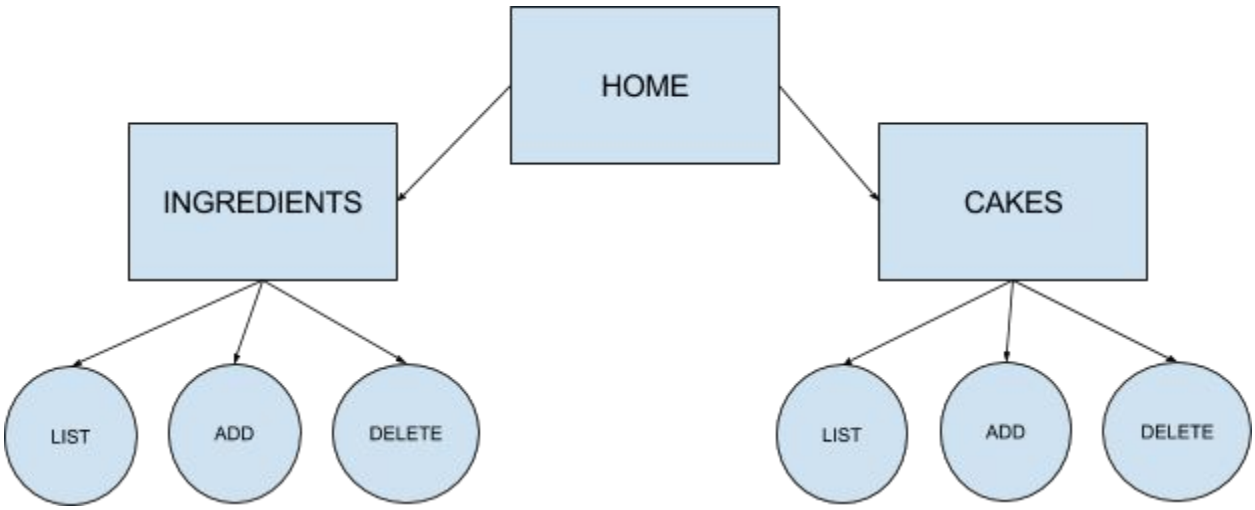
The ingredients and cakes simply have an ID number and name, however they are united into a complete recipe within the recipe table. This recipe table pairs instances of a cake with instances of an ingredient, along with a unit of measure and quantity. This allows for different recipes to use the same ingredients, but have unique quantities employed as the user requires.

In the final program, opening up a cake profile calls on the recipie table and finds instances where the target cake's ID number appears. From there the accompanying ingredient IDs are referenced to the ingredients table to pull the ingredient's name, and printed along with the unit/measurement to form a complete recipe.

- ***What is its Theory of Operation? (Block and flow diagrams are good here.)***

  https://en.wikipedia.org/wiki/Theory_of_operation

When you run the software you go to the homescreen, and from there you have two paths in front of you. you can choose cakes or recipes. From either of these pages, you're given 3 options of listing what's in the db, adding to it or deleting an entry.



- ***How does one install your software? This is where any VM information should live.***

1. Download and Install VirtualBox → https://www.virtualbox.org/wiki/Downloads

2. Download and Install Vagrant → https://www.vagrantup.com/

3. Before continuing, reboot your computer into your BIOS and make sure you have the ability of virtualization activated.

4. In your directory of choice, clone the following repository: https://github.com/jackdb/pg-app-dev-vm myapp

5. Navigate to the myapp/ folder and remove the README and LICENSE files.

6. Inside your myapp/ folder, type 'vagrant up' into the console. This should go through several rounds of checking and installing software, it's actually installing the 'box' as vagrant calls it for the operating system your VM will be using (in this case, 64-bit Ubuntu, a linux distro). It will also install postgresql and output the relevant ports and login information for how you can access it.

7. Download & install & setup postgreSQL (*in your VM!*) → http://www.postgresql.org/download/

8. In your VM, change to the 'postgres' user account and launch the psql command prompt with the following 2 commands. Then, create the database and tables with the CreateTables.txt file provided below.

    # sudo su - postgres
    # psql

9. Back in your main environment, with VM server running, ensure you can access the port that your virtual server is listening in on.

10. This is where we will launch the python/ncurses program to access the database and have ncurses provide the visual interface (see below).

- ***How does one run it?***

VirtualBox shares files between your computer and your VM. In order to get to your files you need to do this on the command line in your myapp folder:

vagrant up

vagrant ssh

cd /

cd mnt

cd bootstrap

You are now in the correct place to run the program. To launch type this:

python < need final file name >.py

- ***Are there any special hardware, OS, or runtime requirements to run your software?***

  Special hardware: no

  OS: The Virtual Machine has to be in a linux environment for running the program.

  Runtime requirements: You will need python 2.7 and the ncurses library installed

- ***Please provide a written getting started guide.*** This should include getting set up, as well as several examples, with screenshots, of how to use your project. If you prefer to do this in video form, you are welcome to do so in addition to the written version.

1. Download and Install VirtualBox → https://www.virtualbox.org/wiki/Downloads

2. Download and Install Vagrant → https://www.vagrantup.com/

3. Before continuing, reboot your computer into your BIOS and make sure you have the ability of virtualization activated.

4. In your directory of choice, clone the following repository: https://github.com/jackdb/pg-app-dev-vm myapp

5. Navigate to the myapp/ folder and remove the README and LICENSE files.

6. Inside your myapp/ folder, type 'vagrant up' into the console. This should go through several rounds of checking and installing software, it's actually installing the 'box' as vagrant calls it for the operating system your VM will be using (in this case, 64-bit Ubuntu, a linux distro). It will also install postgresql and output the relevant ports and login information for how you can access it.

7. Download & install & setup postgreSQL (*in your VM!*) → http://www.postgresql.org/download/

8. In your VM, change to the 'postgres' user account and launch the psql command prompt with the following 2 commands. Then, create the database and tables with the CreateTables.txt file provided below.

    # sudo su - postgres

    # psql

9. Back in your main environment, with VM server running, ensure you can access the port that your virtual server is listening in on.

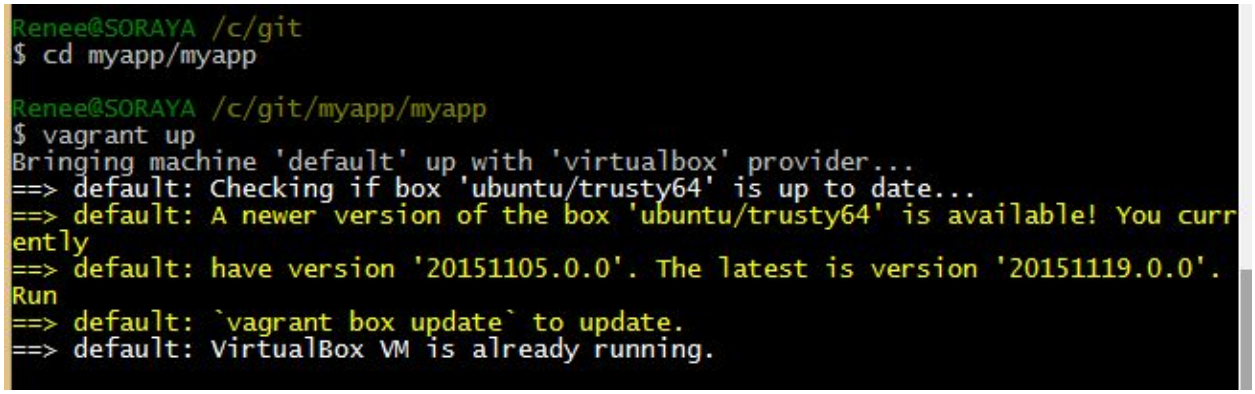
10. This is where we will launch the python/ncurses program to access the database and have ncurses provide the visual interface (see below).

- ***How does one run it?***

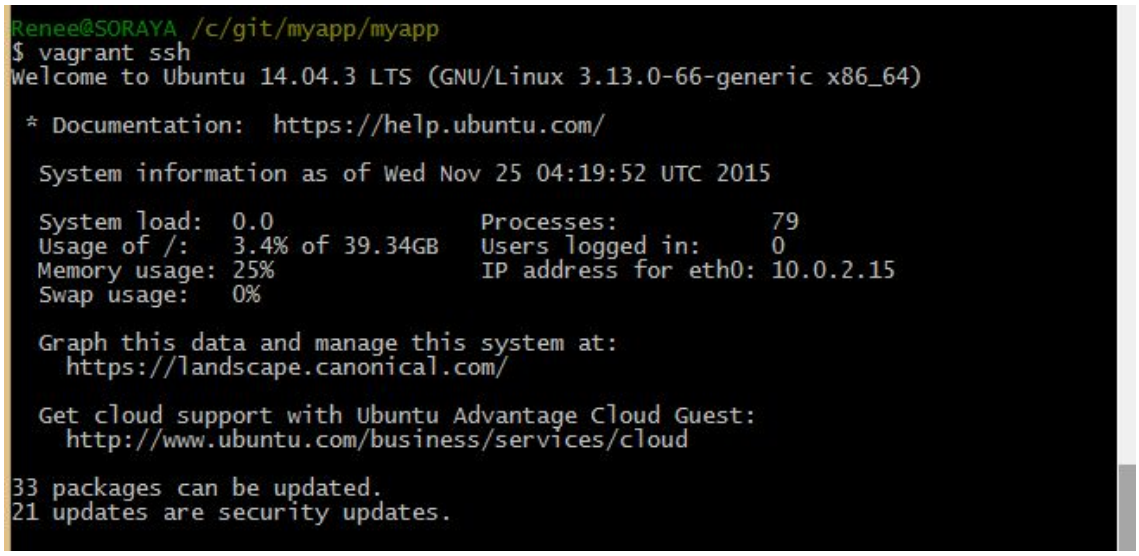
    VirtualBox shares files between your computer and your VM. In order to get to your files you need to do this on the command line in your myapp folder:

    find the folder in you directory that contains myapp.

vagrant up



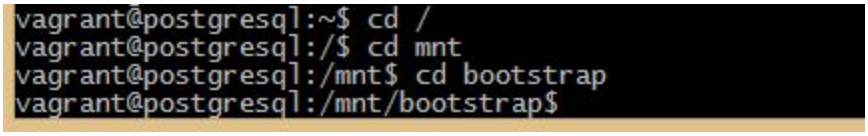
vagrant ssh



cd /

cd mnt

cd bootstrap



You are now in the correct place to run the program. To launch type this:

python < need final file name >.py

4. **How did you learn new technology?**

We utilized many different resources, including websites and books and youtube videos. We also relied on each other for clarity and direction as well as coming together as a group to solve problems.

- ***What websites were helpful? (Listed in order of helpfulness.)***

MOST IMPORTANT, MOST USED WEBSITE OF ALL:

www.google.com

NCURSES

https://docs.python.org/3.4/library/curses.html

https://docs.python.org/dev/howto/curses.html

https://code.google.com/p/master-mind/downloads/detail?name=Programmer%27s.Guide.To.NCurses.pdf

http://stackoverflow.com/questions/11067800/ncurses-key-enter-is-fail

http://stackoverflow.com/questions/22646951/input-limit-on-python-curses-getstr

The following link is to the first video in a series of 16 on ncurses in python. All 16 were watched:

https://www.youtube.com/watch?v=6u2D-P-zuno&list=PL1H1sBF1VAKXLJ3cHisqjy4nGYDMqYIzo

VMWARE/VAGRANT

https://wiki.postgresql.org/wiki/PostgreSQL_For_Development_With_Vagrant

https://www.vagrantup.com/

PYTHON

http://www.tutorialspoint.com/python/

https://wiki.postgresql.org/wiki/Using_psycopg2_with_PostgreSQL

http://stackoverflow.com/questions/19426448/creating-a-postgresql-db-using-psycopg2

http://pythoncentral.io/cutting-and-slicing-strings-in-python/

http://stackoverflow.com/questions/1874113/checking-if-a-postgresql-table-exists-under-python-and-probably-psycopg2

http://stackoverflow.com/questions/15478127/remove-final-character-from-string-python

http://twigstechtips.blogspot.com/2010/09/python-psycopg-doesn-insert-or-delete.html

http://pythoncentral.io/cutting-and-slicing-strings-in-python/

http://stackoverflow.com/questions/21662532/python-list-indices-must-be-integers-not-tuple

POSTGRESQL

http://www.postgresql.org/ used to download

https://help.ubuntu.com/community/PostgreSQL used to troubleshoot

The following link is to the first video in a series of 30. Videos were watched as needed:

https://www.youtube.com/watch?v=CkjQSkWl0F0&list=PLFRIKEguV54bgwAcgFiOs5GMo3q2DhVDj

http://www.postgresonline.com/downloads/special_feature/postgresql83_psql_cheatsheet.pdf

- ***What, if any, reference books really helped?***

- *Programmer's Guide to NCurses by Dan Gookin* (also used .pdf version listed above)
- *Hello World Python Programming by Warren and Carter Sande*

- ***Were there any people on campus that were really helpful?***

No, no one on this team lives close to campus. That was not a resource we were able to use.

5. **What did you learn from all this?**

Going into this project, none of us had previously worked with the ncurses library, postgresql, or even with a virtual machine before. As such every aspect of the project related to these technical aspects was a new experience that we had to learn as we went along. This taught us not only about these systems, but further deepened our understanding of basic python as well. As the CS program at Oregon state was primarilally C/C++ based until now, this was a great opportunity to become more familiar with this programming language.

On a less technical level, this was a good learning experience for understanding group coding projects on a larger scale. Since we all were working remotely, this forced us to quickly learn how to efficiently communicate, share code updates, and to use extensive commenting in our code.

6. One per team member. Renee McLain

- What technical information did you learn?

I learned a lot about ncurses, python, virtual machines and postgresql.

I also learned that I needed hyper-V turned on in Cloud/Mobile Devt for my emulator to work and I needed it turned off for my VM to work. The solution was to keep turning on and off my machine to enable/disable hyper-V while I switched gears on which class I was working on.

- What non-technical information did you learn?

  This entire term seemed to be a lot about solving problems, with all of my classes not just this one. It seemed like I was constantly solving problems, both technical and non-technical, so I would say that non-technical information I learned was how to focus on one thing at a time and solve that before moving to the next issue. The Hyper-V issue was one but there were others, like how to share files between my vm and my machine, how to get all of this working and playing nicely together..

- What have you learned about project work?

  That everyone gets to take a piece of the puzzle and own it together. By that I mean the individual gets to work on it to create it but the team gets to help find solutions to problems. There were a lot of different pieces to this project. Being able to compartmentalize what needed to be done was imperative. Creating our timeline was huge and we referred to that document a lot.

- What have you learned about project management?

  Time for flexibility!! Things changed. We had to adapt. People's schedules got in the way sometimes of meetings. We had to reschedule. However, every member of this time was flexible which made the team work very well together. Also, I just wanted to say that each team member met EVERY weekly deadline that we set for that week. There were a few weeks where the deadlines set up in the original TimeLine were not met or had to be pushed back but that was discussed beforehand.

- What have you learned about working in teams?

  I really enjoyed my team members a lot and I think we worked together fantastically - this is really the best team I have had at OSU. We were able to come together with problems and find a solution. Both of my team members were infinitely valuable in helping put this project together. I liked that we are all able to build on the solution… where one member was stuck we could bring it to the table and the next member was able to look at the problem with different perspective.

- If you could do it all over, what would you do differently?

  I think if I had time I would have liked to create a more thorough database, for example I would like to have an actual recipe that orders the steps you need to take to complete the cake. Also, I think it would be good to add some other pieces into the database as far as customers are concerned: ie ordering, contact info, delivery info ect. Our product is a fantastic start, but there is really so much more that we can do with it.

7. One per team member. Jason MacManiman

- What technical information did you learn?

  The technical information I have learned is threefold. Firstly, I had never really used or understood Virtual Machines and what their benefits are. This project gave me a great opportunity to learn how to install and use Virtual Box and what the pros/cons are of running software in a VM. Secondly, I was able to learn some of the differences between MySQL (which I had used previously in Web Development and Databases) with postgreSQL. The syntax of creating tables and writing queries was different in some places, but having the background in MySQL made picking up another relational database language much

easier. Thirdly, I was able to learn more Python which I've only used briefly in past courses, as well as the ncurses library.

- What non-technical information did you learn?

  Non-technically I learned about working with others to try and work through issues with little direction from our assignment. With so many design decisions left to us it was a very different experience from previous group projects in this program where the rubric is strict and your decisions are basically left to task delegation and meeting times. For this project we were able to have discussions about the pros and cons of design decisions, like when we knew we bit off more than we could chew with our first database model and backed off to something simpler to focus on the programmatic implementation.

- What have you learned about project work?

  The biggest thing I've learned about project work is that organization, clear goals and communication are absolutely essential. Our group stayed organized by having a shared folder in Google Drive where we kept all our work that we needed access to. This kept us from having too many disparate files floating around through e-mail. We also held regular meetings twice a week to make sure our lines of communication were not only open but used consistently. Related to our communication was the need for clear goals. When we set weekly goals and there were concrete deliverables that made it much easier to gauge our progress on the project. For example, instead of a task for me being listed as "Jason will work on VM stuff this week" it would be "Jason will document the steps to install VirtualBox and postgreSQL using Vagrant so that it is repeatable.".

- What have you learned about project management?

  I feel like the aspects of project management I've learned are a repeat of the above answer with one important add-on: flexibility. Being able to manage a project requires you to be flexible with the team members you're working with and our group really excelled at this. We all had a lot going on this term and when one of us would need to push a meeting back or reschedule for another day, we all worked together to find another time when all three of us could meet. This related to the first question as well about our communication as a group. Having open and frequent communication allows for more flexibility for everyone involved.

- What have you learned about working in teams?

  As I stated above, working in teams requires organization, clear goals, communication and flexibility.

- If you could do it all over, what would you do differently?

  I would absolutely use Git to manage our code if I were to do this project again. The number of confusions we had about which version was the most recent could have been avoided using a version control system like Git, and would have made it easier to see all the recent changes from the last commit.

8. One per team member. Eric Miller

   - What technical information did you learn?

For myself, I have not had many opportunities to work with python outside of the Cloud/Mobile assignments in CS 496, so this project was a great opportunity to both refresh what I had learned in that course as well as deepen my understanding of the language.

However while I had a small amount of Python experience coming into this class, I had never worked before worked with ncurses, and exploring this interface method was one of the most interesting aspects of the project for me. Until now I had never worked on a program that did not interact with the user through a CLI, so learning how to create windows and figuring out how to create an intuitive menu system from scratch was a challenging but very enjoyable process for me.

For postgreSQL, I was initially worried about learning a new database language as I have only ever used mySQL. In the end however after taking some time to learn about postgreSQL, I realized that our previous instruction in mySQL helped to ease us into this language without too much trouble. Having heard that postgreSQL is a common standard in the programming world, I am glad that this project introduced me to the common syntax of this language.

Finally, I got to learn quite a bit about virtual machines and how to run them. I had only a vague understanding of what a virtual machine even was before this semester, but by the end was transferring files to and running programs from within a VM on my desktop on a daily basis.

- What non-technical information did you learn?

It is a skillset that I have been working on slowly throughout my time in OSU's computer science program, but time management. Particularly with this class as it was entirely group focused, I had to really be careful in how I spent my time split between my various projects and responsibilities.

What was especially important to consider was that for CS 419, my time management directly affected others outside of myself. This pushed me to become more and more organized as the project went on.

- What have you learned about project work?

Communication is key to having a cohesive, functioning program. In this case I mean beyond just communicating through emails and phone calls, but also communicating through a program's coding and comments. When working on a programming project I had to quickly adapt my coding style to be less 'selfish' and allow for my code to communicate on my behalf when another group member was working with it.

This is principally done through extensive commenting- trying to go step beyond what I normally would because in a group you can't make assumptions about how others might interpret what you are trying to do. Likewise having easily

understandable variable names and sending out a set of update notes for each new code version were essential aspects of working collectively on a coding assignment.

- What have you learned about project management?

To be always ready for a contingency plan and to be flexible. Unlike when working at the same company for example, all three of us have separate classes, jobs, and schedules that do not always align. As such it was important to keep a line of communication open so that meeting times and deadlines could be adjusted as needed.

As such at times it was difficult keeping a regular schedule, but fortunately our group was very good at being flexible, quick to communicate, and to be willing to compromise. These three attributes were what helped us manage project development in a consistent manner.

- What have you learned about working in teams?

Good, coordinated teamwork is better than tackling a big problem on your own. I don't think that I could have handled nearly all the aspects of this assignment if I was doing it by myself.

I was so fortunate to have really great team members for this class, and it made a huge difference. Every week we were able to build upon each other's knowledge and split tasks in a manner that maximized our overall efficiency. In my time at OSU I have had both good and bad groups to work with, but this was the first one to show me how powerful group work is when you have the right mix of people

- If you could do it all over, what would you do differently?

Now that I am more comfortable with postgreSQL and ncurses, I think that I would like to have tackled a project that was bigger in scope. Perhaps a further expansion of the existing system to include things like stock management and customer orders.

As it was however, a good portion of our initial planning and preparation phase was spent in learning and testing the new tools we were presented with. As such we had to expend a lot of time and effort in figuring out ncurses, VM, and postgreSQL. Now that we know how to use these, I think that we would be able to build a larger program should we have to start again.

9. Be honest here -- no B.S.

Appendix 1: Essential Code Listings.

You don't have to include absolutely everything, but if someone wants to understand your project, there should be enough here to learn from.

Creating a window:

Below is the general syntax used to create a new window object. The first two lines are the color schemes used for the window text in general, and highlighted text when the user selects it. Line 3 is the printing out of the window itself (including dimensions), and line 5 is an example of printing out text to a specified coordinate on the window. The 'keypad' option on line 6 allows for the program to receive the user's function key input (like the arrow keys) which are essential for navigating the menu.

```
##initialize ingredient menu window
curses.init_pair(3, curses.COLOR_YELLOW, curses.COLOR_BLUE)
curses.init_pair(1, curses.COLOR_RED, curses.COLOR_WHITE)
win = curses.newwin(20, 75, 2, 2)
win.bkgd(' ', curses.color_pair(3))
win.addstr(2,8,"INGREDIENTS",curses.A_UNDERLINE)
win.keypad(True)
win.refresh()
```

Menu Navigation:

Menu navigation is done by keeping track of what row in the menu is currently highlighted/selected by the user (held in an integer called cursor) and adjusting the menu screen accordingly. For example each option in the menu has a highlighted/non-highlighted version of its text depending on if that row has been selected or not:

```
cursor = 1; ##cursor location
end = 1;##menu loop condition

##get user menu input
while end != 0:

    ##highlisht cursor location
    if cursor == 1:
        win.addstr(4,8,"Ingredients",curses.color_pair(1))
    else:
        win.addstr(4,8,"Ingredients",curses.color_pair(3))
    if cursor == 2:
        win.addstr(6,8,"Cakes",curses.color_pair(1))
    else:
        win.addstr(6,8,"Cakes",curses.color_pair(3))
    if cursor == 3:
        win.addstr(8,8,"Exit",curses.color_pair(1))
    else:
        win.addstr(8,8,"Exit",curses.color_pair(3))
    win.refresh()
    c = stdscr.getch()
```

To change the cursor (and thus highlighted selection on the menu), arrow key input increments or decrements the cursor integer. Getting the arrow key input was the reason for the prior *win.keypad(true)* function call.

Likewise to actually select a menu option, various functions are called when ‘enter’ is pressed based off of the cursor position. for example if the cursor position is ‘2’, the text ‘Open Main Menu’ might be highlighted and pressing enter will open the cake function:

```
##down arrow conditions
if c == curses.KEY_DOWN:
    if cursor < 3:
        cursor = cursor + 1
    else:
        cursor = 1
##up arrow conditions
elif c == curses.KEY_UP:
    if cursor > 1:
        cursor = cursor - 1
    else:
        cursor = 3

##ENTER key conditions
elif c == 10: ##ASCII value for enter key.  Taken from http://stackoverflow.cc
    if cursor == 3:
        end = 0
    if cursor == 2:
         cake_menu()
         win.addstr(2,8,"SELECT MENU OPTION WITH ARROWS:",curses.A_UNDERLINE)
    if cursor == 1:
        ingredient_menu()
        win.addstr(2,8,"SELECT MENU OPTION WITH ARROWS:",curses.A_UNDERLINE)
```

Database Calls:

Whether retrieving information from the database or adding/removing from it, database calls were all made through the ***psycopg2*** library. This starts with programming in the appropriate port number and database ID/password for connecting with the database inside the virtual machine at the very top of our code:

```
conn = psycopg2.connect(database="myapp", user="postgres", password="dbpass",
                        host="localhost", port="5432")
```

From here on out in the code we can reference this database information to make a calls. Here is an example of a database call that retrieves a list of all the ingredient names and stores them in an array for printing to the screen. It should also be noted that the data structure returned by a call is a tuple:

```
##Get ingredient list from the database
db_call = conn.cursor()
db_call.execute("SELECT ingrname FROM bakery.ingr;") #
object_list = []
object_list = db_call.fetchall() # and store the resul
new_object_list = [str(i)[2:-3] for i in object_list]
```

Notice that we reference to the ‘conn’ object setup at the start of the code to make our initial connection, and ***execute()*** to sent a string as a postgreSQL command.For actually making changes to the database (as

opposed to just retrieving data), we have to add an extra ***commit()*** function to make the changes permanent:

```
##add cake name to cake DB
db_call = conn.cursor()
db_call.execute("INSERT INTO bakery.cake VALUES (DEFAULT, '"+new_cake_entry.name+"');")
db_call.connection.commit() ##commit function from http://twigstechtips.blogspot.com/201
```

Without this ***commit()*** function, any changes made to the database revert upon closing the program.

Creating a cake profile object:

From a database call that creates an array of cake names:

```
##Get database cake profile names
db_call = conn.cursor()
db_call.execute("SELECT cakename FROM bakery.cake;") #
object_list = []
object_list = db_call.fetchall() # and store the resul
new_object_list = [str(i)[2:-3] for i in object_list]
i = 0
```

Each index in the cake menu takes the name of the corresponding cake in the cake array:

```
##if cursor is on the first list object
if cursor == 1:
    win.addstr(4,10,"-                    ",curses.color_pair(3))
    win.addstr(4,10,str(new_object_list[index]),curses.color_pair(1))
else:
    win.addstr(4,10,"-                    ",curses.color_pair(3))
    win.addstr(4,10,str(new_object_list[index]),curses.color_pair(3))
```

When a user selects a cake from the menu, a call is made to the function *"create_cake_ob(cake name)"* so that a fully developed cake object (that includes an accompanying recipe) can be created and then easily referenced for display to the screen:

```
cake_profile = create_cake_obj(str(new_object_list[index]))
open_profile(cake_profile)
```

The object being used to store the cake information has the following format:

```
##cake profile object
class cake_entry:
    def __init__(self,input_name, input_ingredients, input_id):
        self.name = input_name
        self.ingredients = input_ingredients
        self.id = input_id

 ##ingredient object for cake profile
class ingredient_entry:
    def __init__(self,input_name, count, unit):
        self.name = input_name
        self.count = count
        self.unit = unit
```

Note that all ingredients that a cake uses are stored in an array of ingredient objects (ingredient_entry) that contain the unit of measurement / quantity along with the ingredient name.

For the function to create the cake object itself, it starts with the following:

```
def create_cake_obj(name):

    ##get the cake name
    input_name = name

    ##get the cake ID
    db_call = conn.cursor()
    db_call.execute("SELECT CakeID FROM bakery.cake WHERE CakeName ='"+name+"';")
    id_call = []
    id_call = db_call.fetchall() # and store the results in a list
    ID_string = str(id_call[0])[1:-2]
```

The first step is to determine the name of the cake to add to the object, which is easily determined from the function's parameter. To get the ID, a database call is made to find the ID of the cake that shares the name as the one used in the function (as seen above).

The next step is a bit more tricky. An array of tuples that contain a quantity, unit, and ingredient name is created via database call using the cake ID to find recipes related to that cake in the recipe database:

```
 db_call = conn.cursor()
db_call.execute("
        SELECT Quantity, Unit, IngrName
        FROM bakery.recipe, bakery.ingr
        WHERE Cid='"+ID_string+"' AND IngrID = Iid;
")
object_list = []
object_list = db_call.fetchall()
```

Once there is an array of tuples containing this information, each tuple's information can then be used to create an ingredient object, which is appended into an array:

```
##list of ingredient_entry objects
ingredient_list = []
for i in object_list:
    ingr_data = i
    ingredient_list.append(ingredient_entry(str(ingr_data[2]),str(ingr_data[0]),str(ingr_data[1])))
```

Now with an array of ingredient objects, a cake name, and a cake ID, the last step is to initialize a new cake profile object using these as parameters:

```
##create cake entry object using new ingredient entry list, inp
cake_object = cake_entry(input_name,ingredient_list,ID_string)

return cake_object
```

The result is a cake object that contains its name, ID, and a full recipe including ingredients/measurements.

User text input:

```
##get user keyboard input
##code for limiting keyboard input from http://stackoverflow.com/questions/22646951/input-limit-on-python-curses-getstr
while finish == 0:
    input_win.border('|','|','-','-','*','*','*','*')
    input_win.refresh()
    char_input = input_win.getch(5,x)

    if char_input == 10:
        if len(user_input) > 0:
            finish = 1
        else:
            finish = 0

    elif char_input == 263 :
        if len(user_input) > 0:
            input_win.addstr(5,x-1," ")
            x = x -1
            user_input = user_input[:-1] ##syntax from http://stackoverflow.com/questions/15478127/remove-final-character-from-string-python
        else:
            user_input = user_input ##do nothing

    elif len(user_input) > 15:
        curses.beep()

    elif (char_input >= 48 and char_input <= 57) or (char_input >= 65 and char_input <= 90) or (char_input >= 97 and char_input <= 122):
        input_win.addstr(5,x, chr(char_input))
        user_input = user_input + chr(char_input)
        x = x + 1

    else:
        user_input = user_input
```

It becomes important in many parts of this program to get user keyboard input for things such as new ingredient names that the user wishes to add. In order to get alphanumeric input of a certain min/max length, the following code is used:
While it may look confusing at first, each branching 'if' case is simply for a different type of keyboard input, represented by the ASCII numbers that correspond to each key.

For example '10' is the enter key- the keyboard input loop is ended when this is pressed and there is at least on character entered. Farther down, '262' is the delete key, and this removes the last char in the string so long as it is > 0. Finally, 48 - 57, 65 - 90, 97-122, represent all alphanumeric characters and only these are accepted as valid ingredient name input. This code also rejects input beyond 15 character in:

```
elif len(user_input) > 15:
    curses.beep()
```

Thus between the '>0' clause when pressing the enter key and the code above, user input is limited to between 1-15 characters.

Appendix 2: Anything else you want to include

Photos, easter eggs, hidden features, etc.

Testing
11/24/2015

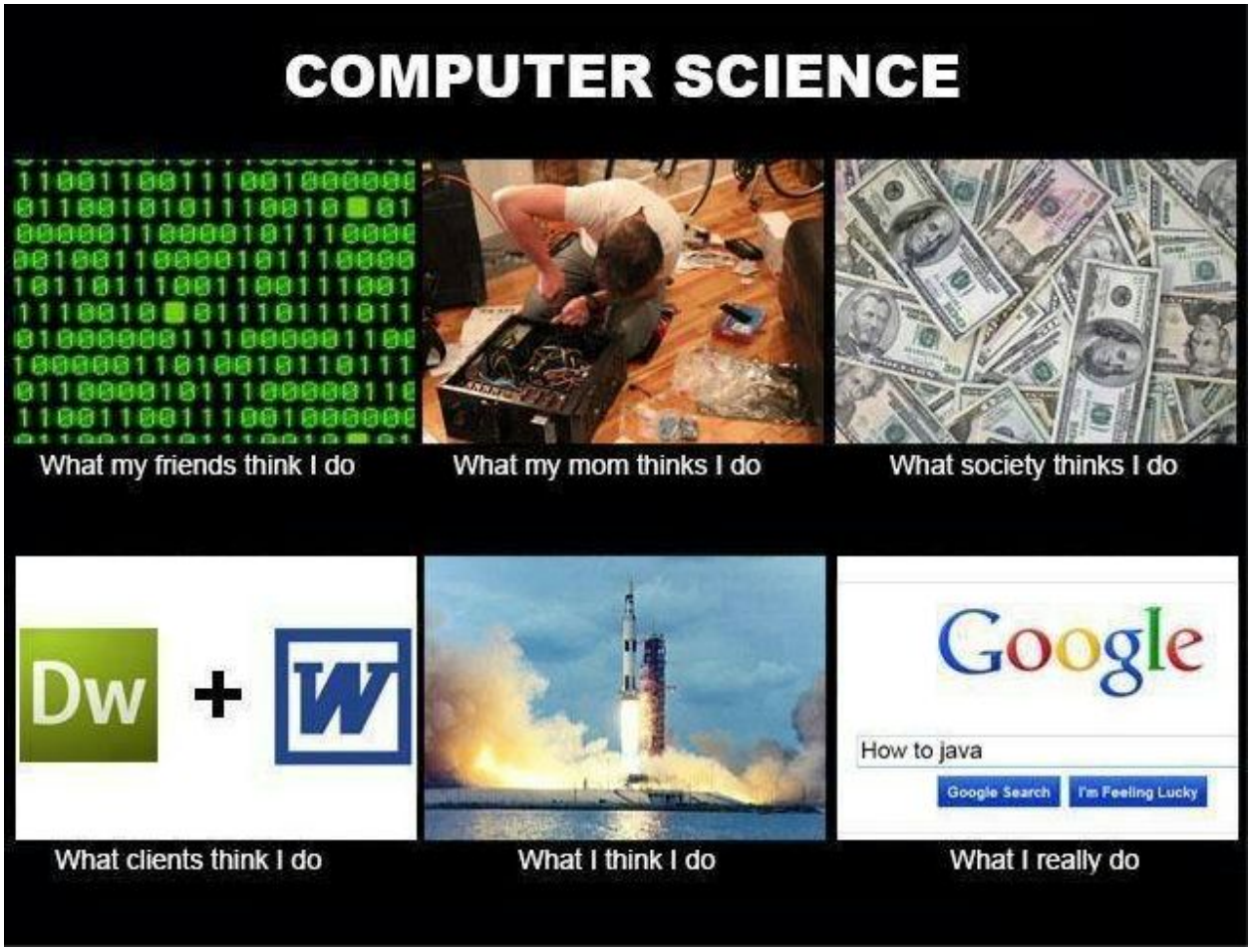
1. Opened Database

Action	Expected Outcome	Actual Outcome
Selected Ingredients	Goes to Ingredient page	Goes to Ingredient page
Select List Ingredients	Populated Ingr list pops up	Populated Ingr list pops up
Selected Next	Goes to next page of Ingr	**Nothing, not enough ingr**
Selected Back	Goes back one page of Ingr	**Nothing**
Selected Ingr Menu	Takes me back to Ingr Menu	Takes me back to Ingr Menu
Selected Add Ingredient	Pop up to Enter name of Ingr	Pop up to Enter name of Ingr
Added Ingredient in PopUP	"Ingredient Entered" message	"Ingredient Entered" message
Entered key to return to Ingr Menu	Return to Ingr Menu	Returned to Ingr Menu
Selected "Main Menu"	Return to Main Menu	Returned to Main Menu
Selected "List Cakes"	Populated Cake List pops up	Populated Cake List pops up
Selected a Cake	Recipe populates	**CRASH**
Selected "Search by Ingredient"	Pop up to list ingredient	pop up to list ingredient
Entered Ingredient to search	pops up	**CRASH**
Selected Add Cake	Ingredient count PopUp	Ingredient count PopUp
Used arrow to find ingr count	Numbers ^ as arrow is pressed	Numbers ^ as arrow is pressed
Input selected	PopUp to input each ingredient	PopUp, input each ingredient
Ingredient Entered	Accepts ingredient	flows to measurement window
Quantity entered as a char	Fail + error message	Fail, **NO error message**
Finish entries + Enter	Back to Cake Menu	**CRASH**
Main Menu selected	Taken back to main menu	Taken back to main menu
Selected Exit	Exits program	Exits program

Final Testing after errors were addressed
12/07/2015

1. Opened Database

Action	Expected Outcome	Actual Outcome
Selected Ingredients	Goes to Ingredient page	Goes to Ingredient page
Select List Ingredients	Populated Ingr list pops up	Populated Ingr list pops up
Selected Next	Goes to next page of Ingr	Goes to next page of Ingr
Selected Back	Goes back one page of Ingr	Goes back one page of Ingr
Selected Ingr Menu	Takes me back to Ingr Menu	Takes me back to Ingr Menu
Selected Add Ingredient	Pop up to Enter name of Ingr	Pop up to Enter name of Ingr
Added Ingredient in PopUP	"Ingredient Entered" message	"Ingredient Entered" message
Entered key to return to Ingr Menu	Return to Ingr Menu	Returned to Ingr Menu
Selected "Main Menu"	Return to Main Menu	Returned to Main Menu
Selected "List Cakes"	Populated Cake List pops up	Populated Cake List pops up
Selected a Cake	Recipe populates	Recipe populates
Selected "Search by Ingredient"	Pop up to list ingredient	pop up to list ingredient
Entered Ingredient to search	pops up	pops up
Selected Add Cake	Ingredient count PopUp	Ingredient count PopUp
Used arrow to find ingr count	Numbers ^ as arrow is pressed	Numbers ^ as arrow is pressed
Input selected	PopUp to input each ingredient	PopUp, input each ingredient
Ingredient Entered	Accepts ingredient	flows to measurement window
Quantity entered as a char	Fail + error message	Fail + NO error message
Finish entries + Enter	Back to Cake Menu	Back to Cake Menu
Main Menu selected	Taken back to main menu	Taken back to main menu
Selected Exit	Exits program	Exits program



Yes, we each just paid OSU 30K to teach us the very important life skill of how to effectively Google.

Happy Holidays Professor!!