Final Deliverable Document
Group 14: Pitt Jungle

Members: Sushruti Bansod, Sherryl Augustine, Luiza Urazaeva, David Ladeji, Trent Lessig

Table of Contents/Checklist

Software configuration is complete.

Document includes

The Software Plan

1.0 Scope

When students are starting college, it is the first time they are on their own. Students tend to focus their time on academics, which although is important, can be detrimental to their overall college experience. We hope our product can bridge the gap between academics and leisure by introducing students to dining options other than university dining halls, as well as other local events and attractions.

Our team will put together a product called PittJungle. It will mainly be used as a navigation tool for Pitt students to find local events, eateries, academic recommendations and activities on and around campus. It will allow students to create groups and communities with people who have similar interests. This will help new students create a life outside of class which is vital for a college lifestyle.

1.1 Functions

- User Authentication
    - Users will be able to create an account and log in
- Event and Food Recommendations
    - Users will be able to leave ratings at restaurants or events.
- Filter Functionality
    - Users will be able to choose and filter types of food based on cuisine and dining style.
    - This filter will also apply to locations and events by dividing them into subgroups.
- Friend Feature
    - Users will be able to see what other users liked and thus make friends with similar tastes.
    - Users will be able to friend similar people and add them to their network

Wishlist Features:

- Academic resources
    - Users will be able to recommend courses to their friends on the network
    - Users will be able to host study sessions and get academic help
- Chatbot
    - Users will be able to interact with a chatbot to get recommendations on food/academic material
- App Development
- Users will be able to write reviews for the restaurants and events.

1.2 Performance

- Website/Web application should be user friendly
- Multiple users should be able to login and use features
- Event/Restaurant ratings should been seen by other users
- Chatbot should have an acceptable rate of response/accuracy
- Users should be able to post an event

1.3 Limitations

- The user will not be able to alter information about the event or restaurant.
- Users will not be able to change login information.

2.0 Tasks

- User Interface/Graphic Design
- Database/Profile Management
- Deep learning/Recommendation System
- Testing
- Researching restaurants and places in Pittsburgh

3.0 Resources

3.1 Hardware

- Linux Server
- Windows computer

3.2 Software

- User Interface - React or Flask
- Graphic Design - Illustrator or Photoshop
- Databases - SQLite
- Chatbot/ML - Python, Bootstrap, Watson Assistant
- Software Testing Suite
- Chrome
- Visual Studio Code

3.3 People

	Project Manager	GUI Designer	Databases/ Backend	Testing	Research	Chatbot/ML/ Recommender
Sherryl		x		x	x	
Sush	x	x		x		
Luiza			x	x		
David			x	x		x
Trent				x	x	x

4.0 Cost

Task	Requirements	Plan/Design	Code	Test	Validate/ Maintain	Total hours	Rate	Total
Sush	40 hours	80 hours	160 hours	80 hours	80 hours	440 hours	$100	$44,000
Sherryl	40 hours	80 hours	160 hours	80 hours	80 hours	440 hours	$100	$44,000
Trent	40 hours	80 hours	160 hours	80 hours	80 hours	440 hours	$100	$44,000
Luiza	40 hours	80 hours	160 hours	80 hours	80 hours	440 hours	$100	$44,000
David	40 hours	80 hours	160 hours	80 hours	80 hours	440 hours	$100	$44,000
Other Resources			Lines of Code		10,000		$1	$10,000
Total Cost								$230,000

Function	Optimisitc	Most	Pessimistic	Expected	Deviation	$/Line	Cost
Interface	2500	4000	5400	3983.33	250	$25	$2,375
User/Event Databases	1500	2000	2250	1958.33	83.33333	$30	$2,850
Recommendation System	2000	3000	4000	3000	166.6667	$30	$2,850
Event Posting	750	1200	1400	1158.33	75	$12	$1,140
Review/Comment	850	1250	1500	1225	66.66667	$12	$1,140
Total Expected LOC and COST				**11325**			**$10,355**

5.0 Schedule

Week	1 (9/1-9/5)	2 (9/6-9/12)	3 (9/13 – 9/19)	4 (9/20 – 9/26)	5 (9/27 – 10/3)	6 (10/4 – 10/10)
Task	Requirements	Plan	Design	Code	Code/Test	Code/Test
Assignment		**9/8 Milestone 1**		**9/22 Milestone 2**		**10/4 Milestone 3**

Week	7 (10/11 – 10/17)	8 (10/18 – 10/24)	9 (10/25 – 10/31)	10 (11/1- 11/7)	11 (11/8 – 11/14)	12 (11/15 – 11/19)
Task	Code/Test	Code/Test	Code/Test	Code/Test	Validate	Maintain
Assignment		**10/20 Milestone 4**	**10/27 Milestone 4**		**11/10 Milestone 4**	**11/17 Submission**

Software Requirement Specifications

1. **Product Overview and Summary**

College is a time when teens start to make decisions for themselves for the first time. From their schedule of classes to when to eat. For many, this can be a stressful time without the right guidance or help. With this newfound independence, students can lead to students spending too much time in academics which can greatly affect their college experience and mental health.

Our team hopes to put together an app that acts as a guidance/navigation tool. Directed towards Pitt students, our app will allow students to find something to invest time in outside of academics. We plan to implement a netflix like interface to introduce students to dining options other than Market Central and Perch. We want to provide students options based on their preferences, therefore we will be implementing a deep learning aspect that will show suggestions based on previous likes. Similarly, we will implement a page specified for certain attractions around pittsburgh. Pittsburgh is a very culturally diverse city, therefore a guide on places to visit such as certain museums, and parks will be beneficial for students to get to know this city. Lastly, we want to include an events tab. Which users can edit and add to. This will allow people to see concerts, meet-ups and different types of local events within the city.

Our concept will be implemented using React, Python and other softwares. By dividing the team into groups, we hope to speed up the coding and testing process while also securing quality in our website. Until further discussion, this product will be a website, however, if time allows it, our team hopes to turn this into an app for easy access. We decided to call it PittJungle so students can better navigate the concrete jungle that is the University of Pittsburgh.

2. **Information Description**

2.1. User interface

- PittJungle helps the users find restaurant events by preference in the Pittsburgh area. It allows the Pitt students to create groups with people who have common preferences and interests. The main function of the website is to find optimal food and leisure options for students.

- The users can start the website by opening PittJungle website in their browsers. At the landing page, the news related to the events and food in Pittsburgh will be shown.

- If the user is visiting PittJungle website the first time, the user must register in order to access the user-specific area. In order to register, the user must first provide a full name, email address, Pitt username, cell phone number, a unique

username as well as password. Then the website stores the user information. After that user can login by typing username and a password in a login area.

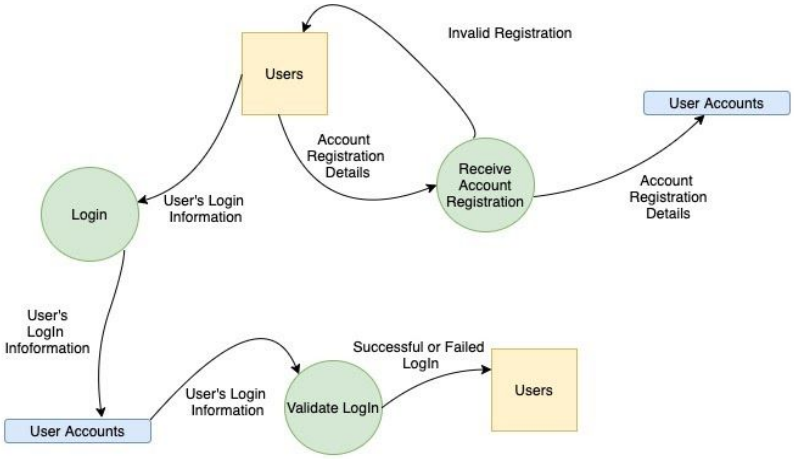
- The PittJungle website will have the FAQ link at the bottom right corner. In the FAQ, all user related questions will be addressed. The FAQ section will be regularly updated based on the questions and feedback from the users. The email and contact information will be posted in the FAQ. The users will have the ability to send questions and feedback via email. The popular, important questions and their answers will be posted in the FAQ. The administrator will need to make sure that no private information is revealed in the FAQ posts.

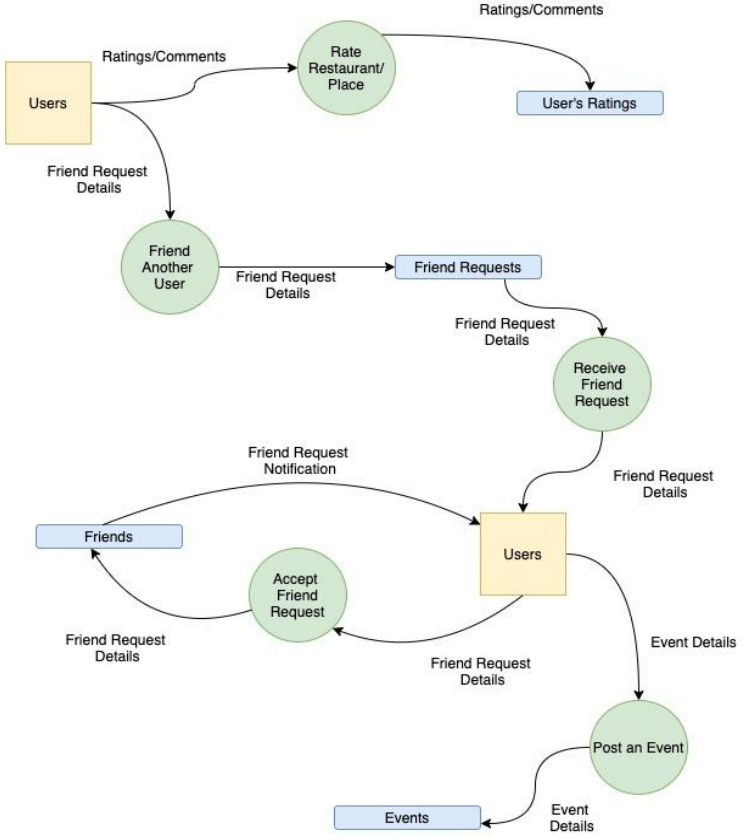
- A sample run can be described as simulation of the user searching for a restaurant. The user logs into the PittJungle. Then the website displays a list of options of restaurants and events based on the user's previous queries. If the user is logging the first time, a short survey will pop up to find out the user's preferences. The user can choose the options for food and types of events from drop-down menus. The AI algorithms process the user's query and display the result. The user can select the restaurant from the results and find out information about the restaurant.

- Since the project is going to be a website, the users will be able to choose filters from the drop-down menus and hyperlinks for filtering restaurants and events. There will be 3 modes: 1) students, 2) restaurant or event planner partner and 3) admin or moderator.

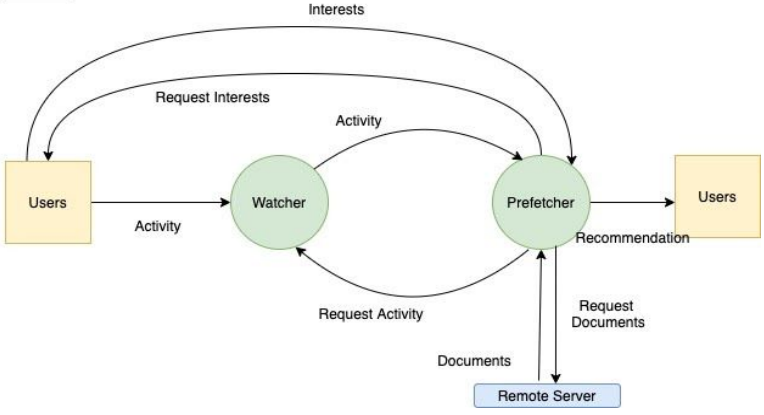
- Each mode of operation will have various commands and different levels of permissions. The user can view, recommend or "like" the restaurants. The restaurant partners can add or edit the information about their restaurants but they cannot create their own account. The admins will have all the commands and permission at their disposal so that they can create users or restaurant partners and override their actions if necessary.

2.2. **High level data flow diagram**





Recommendation System:



2.3. Data structure (or object) representation

- We will have an object that represents a user.
    - When the user is registering for an account we will ask for their name, major and their year in school
    - The first and last names, the user's major and their year in school will all be stored in their individual user object.
    - Eg. user(first_name, last_name, major, year_in_school)
        - first_name: user
        - last_name: user
        - major: user
        - year_in_school: user
- We will have an object that represents a particular place that users can visit.
    - The name and address of the place will provide the user with basic information about the place.
    - The type of the place will be used in the filter functionality. For example, The Carnegie Museum of Art will have a type of "Museum" and "Art". This way users can filter out places they want to visit based on these types/tags.
    - Eg. place(name, address, type)
        - name: place
        - address: place
        - type: place
- We will have an object that represents an event that users can post
    - When a user posts an event they will need to enter the name, date, time, address of the event. They will also be able to classify the event into a type. This classification will be used in the filter functionality.
    - The type of event will allow users to filter and find events they want to attend. For example, a user can create an event with the type "Study". This indicates that the event that the user is creating is an academic event such as a group study session.
    - Eg. event(name, date, time, address, type)
        - name: event
        - date: event
        - time: event
        - address: event
        - type: event
- We will have an object that will represent and show the details of a friend request.

- When a user sends a friend request, our application will need to keep track of who is sending the friend request, who is receiving the friend request and the status of the request.
- So, we will keep track of the source and destination of the friend requests.
- We will also keep track of the status of the friend request. The request could still be pending or the “destination” user could have accepted it or declined it. So, the 3 possible values for the status are: Pending, Accepted, Declined
  - eg . friendRequest(sourceUser, destinationUser, status)
    - sourceUser: friendRequest
    - destinationUser: friendRequest
    - status: friendRequest

2.4. Data elements (or objects) dictionary

- Create a list of data objects. We research what kind of data we need. For example, we will need the user information and user preferences.
- Set the properties of data elements. We will need to decide what size and what type each data element will be. We will also need to set which data serves as indexes to build relationships between tables.
- Classify the data. In this step, we categorize data according to their domains and relationships
- Build schema for database. After classifying the data, we build a dictionary for data elements. For example, restaurant hours will be of type Date time and restaurant names will be strings of maximum 80 characters length. Both data types will be columns in the restaurant tables. Consequently, the users will have restaurant IDs of the restaurants they like. We then can match restaurant ID to the IDs in the restaurant and retrieve all the restaurants the user likes.
- Sample Table:
  Table User
  User id: Integer
  Username: String (20)
  First Name: Sring (80)
  Last Name: String (80)
  Email: String (80)
  Password_Hash: Integer
  City: String (80)

3. Functional Description

3.1. Functions

The IC cards for our project are as follows :

IC Card IC Name: Register_account

Description: Process details to add new account
Interaction Pattern:

Mixed

My Task: Add new user account, transits to next state if error occurs
Time Critical Condition: Under 5 minutes
Name of Other IC: None
Message to Other IC: None
Other IC's Task: None
Card 1of 2 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Create_event

Description: Function that uses event details to create an event
Interaction Pattern:

Mixed

My Task: Create an event
Time Critical Condition: Under 5 minutes
Name of Other IC: None
Message to Other IC: None
Other IC's Task: None
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Add_friend function

Description: Adds a new friend to the users involved
Interaction Pattern:

Mixed

My Task: Add friend if friend request is accepted
Time Critical Condition: Under 5 minutes
Name of Other IC: None
Message to Other IC: None
Other IC's Task: None
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Log_in function

Description: Authenticates user account details
Interaction Pattern:

Mixed

My Task: Process user account details, transits to next state if error occurs
Time Critical Condition: Under 2 minutes
Name of Other IC: None
Message to Other IC: None
Other IC's Task: None
Card 1of 2 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Watcher

Description: Monitors documents and interacts with the prefetcher
Interaction Pattern:

Quiet State
My Task: When activity is recieved, transits to next state
Time Critical Condition: Under 5 minutes
Name of Other IC: None
Message to Other IC: None
Other IC's Task: None
Card 1of 2 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Register_account

Description: Process details to add new account
Interaction Pattern:

By Myself no Interacton
My Task: Send error message
Time Critical Condition: Under 5 minutes
Name of Other IC: New_user
Message to Other IC: Invalid registration
Other IC's Task: None
Card 2of 2 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Prefetcher

Description: Prefetches documents
Interaction Pattern:

By Myself no Interacton
My Task: Send recommendations
Time Critical Condition: Under 5 minutes
Name of Other IC: User
Message to Other IC: None
Other IC's Task: None
Card 4of 4 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: User

Description: Creator of event
Interaction Pattern:

By Myself no Interacton
My Task: Send event details
Time Critical Condition: Under 2 minutes
Name of Other IC: Create_event
Message to Other IC: None
Other IC's Task: Create an event
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: User

Description: Send friend request
Interaction Pattern:

By Myself no Interacton
My Task: Send friend request details
Time Critical Condition: Under 2 minutes
Name of Other IC: Add_friend function
Message to Other IC: None
Other IC's Task: Add friend if friend request is accepted
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Watcher

Description: Monitors documents and interacts with the prefetcher
Interaction Pattern:

By Myself no Interacton
My Task: Send activity
Time Critical Condition: Under 5 minutes
Name of Other IC: Prefetcher
Message to Other IC: None
Other IC's Task: Request document
Card 2of 2 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: New_user

Description: Send details to register new account
Interaction Pattern:

By Myself no Interacton
My Task: Send user details
Time Critical Condition: Under 2 minutes
Name of Other IC: Register_account
Message to Other IC: None
Other IC's Task: Add new user account
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Log_in function

Description: Authenticates user account details
Interaction Pattern:

By Myself no Interacton
My Task: Send error message
Time Critical Condition: Under 2 minutes
Name of Other IC: Returning_user
Message to Other IC: Invalid log in
Other IC's Task: None
Card 2of 2 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Returning_user

Description: Send details to log in
Interaction Pattern:

By Myself no Interacton
My Task: Send user details
Time Critical Condition: Under 2 minutes
Name of Other IC: Log_in function
Message to Other IC: None
Other IC's Task: Process user account details
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Prefetcher

Description: Prefetches documents
Interaction Pattern:

By Myself with Interaction
My Task: Request user activity
Time Critical Condition: Under 5 minutes
Name of Other IC: Watcher
Message to Other IC: request activity
Other IC's Task: Send user activity
Card 2of 4 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Prefetcher

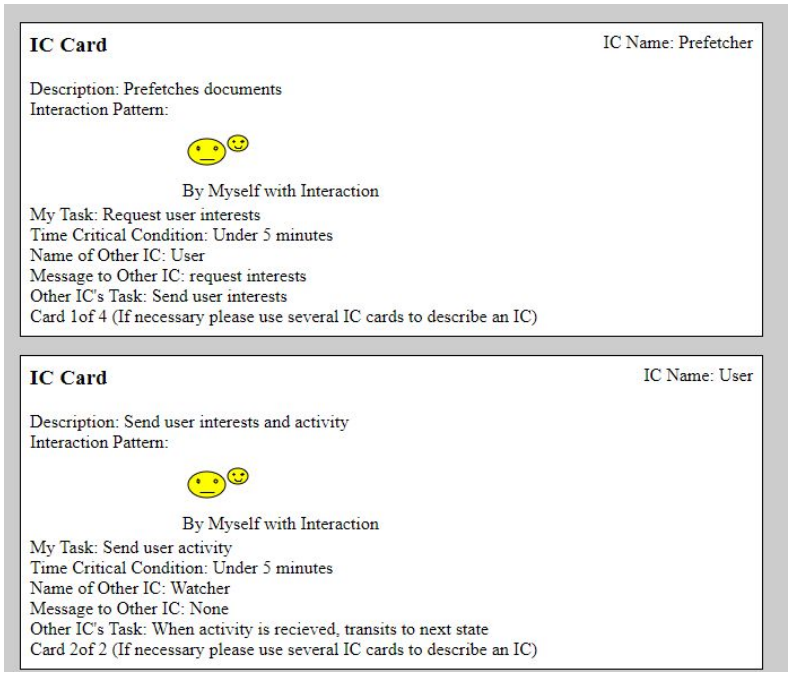
Description: Prefetches documents
Interaction Pattern:

By Myself with Interaction
My Task: Request document
Time Critical Condition:
Name of Other IC: Remote server
Message to Other IC: Request document
Other IC's Task: Send document
Card 3of 4 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Remote server

Description: Send document
Interaction Pattern:

By Myself with Interaction
My Task: When request is received, sends document
Time Critical Condition: Under 5 minutes
Name of Other IC: Prefetcher
Message to Other IC: None
Other IC's Task: Send recommendation
Card 1of 1 (If necessary please use several IC cards to describe an IC)



IC Card IC Name: Prefetcher

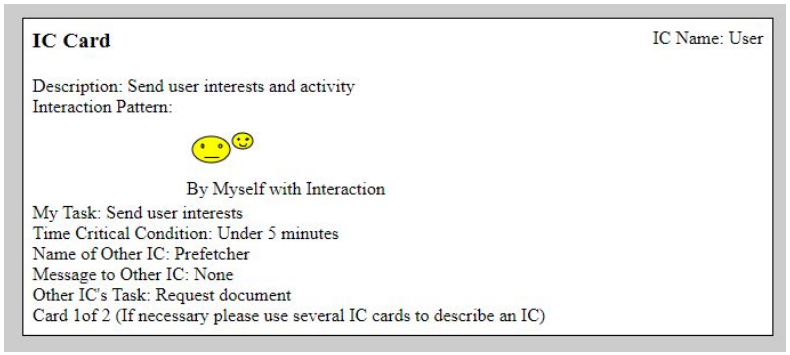
Description: Prefetches documents
Interaction Pattern:

By Myself with Interaction
My Task: Request user interests
Time Critical Condition: Under 5 minutes
Name of Other IC: User
Message to Other IC: request interests
Other IC's Task: Send user interests
Card 1of 4 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: User

Description: Send user interests and activity
Interaction Pattern:

By Myself with Interaction
My Task: Send user activity
Time Critical Condition: Under 5 minutes
Name of Other IC: Watcher
Message to Other IC: None
Other IC's Task: When activity is recieved, transits to next state
Card 2of 2 (If necessary please use several IC cards to describe an IC)



IC Card IC Name: User

Description: Send user interests and activity
Interaction Pattern:

By Myself with Interaction
My Task: Send user interests
Time Critical Condition: Under 5 minutes
Name of Other IC: Prefetcher
Message to Other IC: None
Other IC's Task: Request document
Card 1of 2 (If necessary please use several IC cards to describe an IC)

3.2. Processing narrative

The functions that power the website include user authentication, even and food recommendation, filter functionality, friend feature and administrative functions.

- User Authentication. This functionality allows to authenticate the users so that the users have secure accounts. It also keeps the user's private information confidential. In order to create an account, the user has to click on the "register" button, then the user has to provide private information like full name, cell phone numbers, physical and email addresses. We use tested public authentication libraries that deploy RSA encryption to implement secure authentication. To login, the user has to enter username and password in the login page. After username and password matches with username and password stored in the database, the website shows the user's landing page.

- Event and Food Recommendations. The PittJungle website will have capability to recommend based on the users' ratings. The user will be able to leave ratings at the restaurant and events. When the user finds a restaurant or event that matches to their preference, they leave a rating. Overall higher positive ratings will allow the restaurant to show up higher on the rankings in the search results.

- Filter Functionality. The users will also be able to choose, and filter types of food based on cuisine and dining style. The users will be able to select the options from drop-down menus. After the user selects the options to filter, the PittJungle can show recommendations based on the filter results. This filter will also apply to locations and events by dividing them into subgroups.

- Friend Feature. The PittJungle will incorporate social network features to improve the user experience. The user will be able to follow other users and be a fan of the restaurant or events. They will also be able to see what other users liked. Consequently, they can make friends with similar tastes. The user can search specific users by their full name or email address so they can follow a specific person whom they may have met at the restaurant or event. To give users full control of their preferences, they will be able unfollow friends. If the restaurant or event falls out of their favor, the user can also unfollow the restaurants or events they liked and followed previously.

- Administrative Functions. The administrators of the PittJungle will have permissions to add, delete users and override their actions if necessary. For example, if the user needs to be kicked out of the PittJungle, the administrator will be able to do so. Administrators will also be able to add and delete restaurant profiles on PittJungle. The restaurant won't have the ability to self-register their establishment. Admins need to properly vet the restaurants so that fake restaurants will not show up on the search result. If the restaurant registered on the PittJungle network ceases to exist, the admins will have the ability to remove the restaurant from the listings.

3.3. Design constraints

3.3.1 Functional Constraints

3.3.1.1 User Likes/Dislikes

Users will only be able to like or dislike a specific place or event once

3.3.1.2 User Authentication

Users will be limited, meaning no username or password changes will be accepted

3.3.1.3 Friend Requesting

Users will be able to send requests to other users for acceptance. If denied, another request can be sent

3.3.1.4 Filter Functionality

Users will only be able to sort by the provided sorting tags

3.3.1.5 Event Posting

A user who posts an event will not be able to change the event. A comment within the event could clear up misunderstanding.

3.3.1.6 Recommendation System Data Collection

The recommender system will be limited to the amount of data stored for each user. Data will be added to user 'log' based on likes/dislikes.

3.3.1.7 Help Mode

This aspect will not be interactive. It will be a user manual and/or video.

3.3.2 Other design constraints

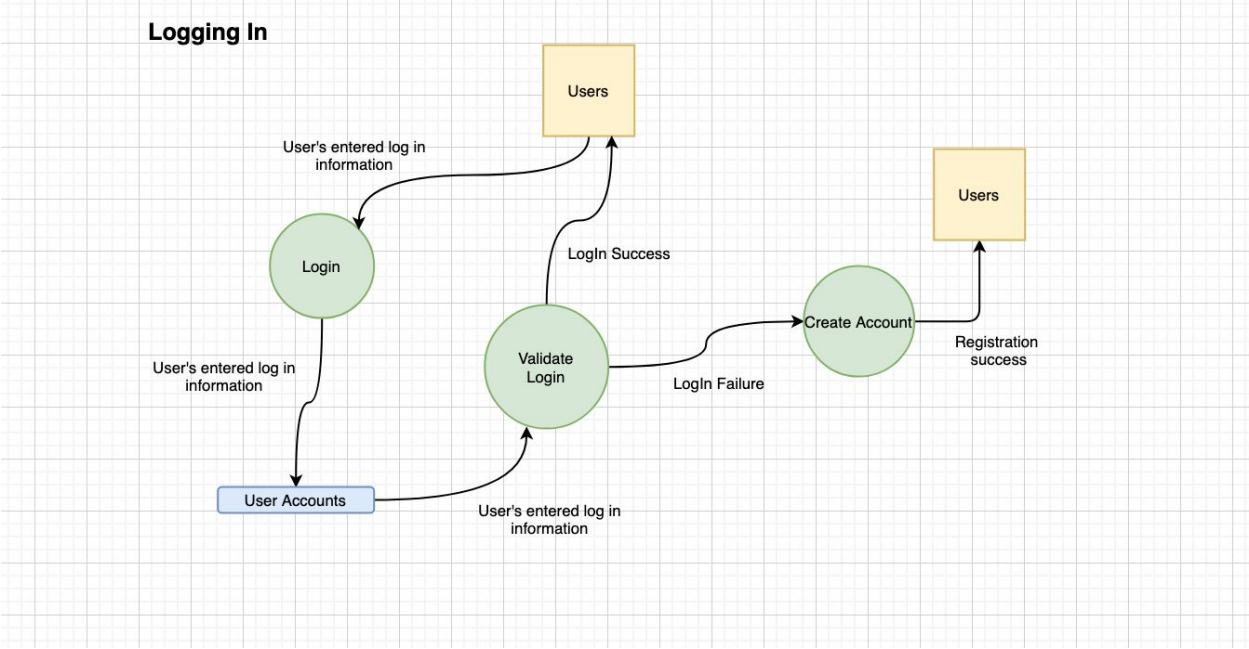
3.3.2.1 Data

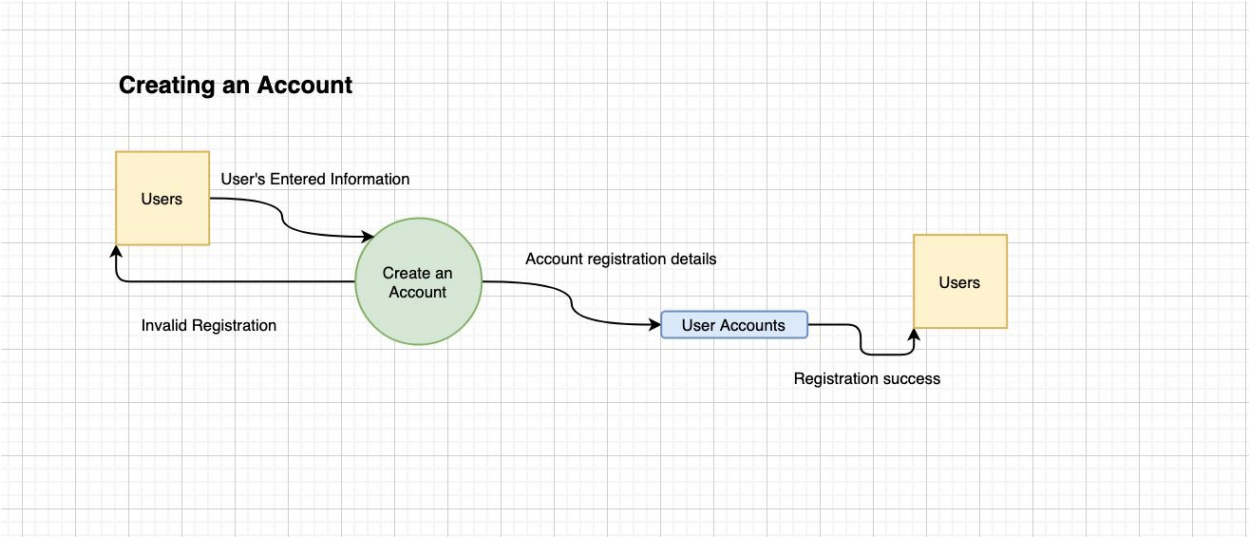
As the team is not surveying/polling the places and activities of what we are including, the data is constrained to what is actively available to us over the internet. Regarding student posted events, this is responsible to them to post legitimate material

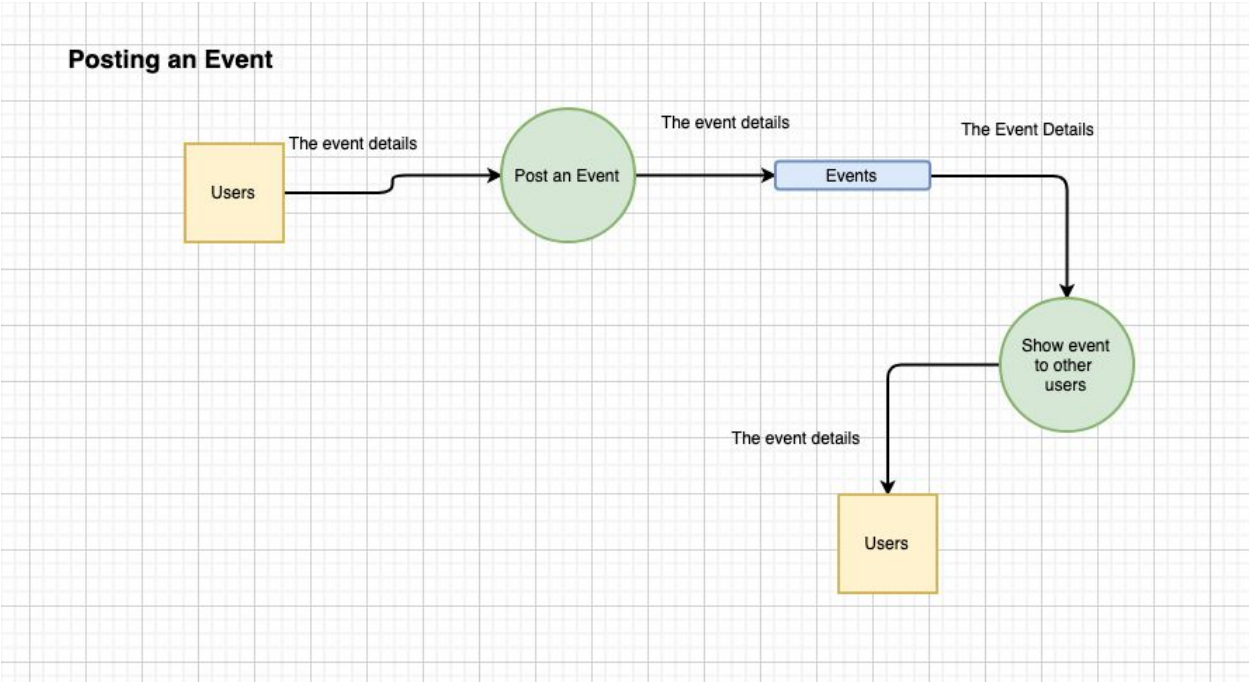
3.3.2.2 Geolocation

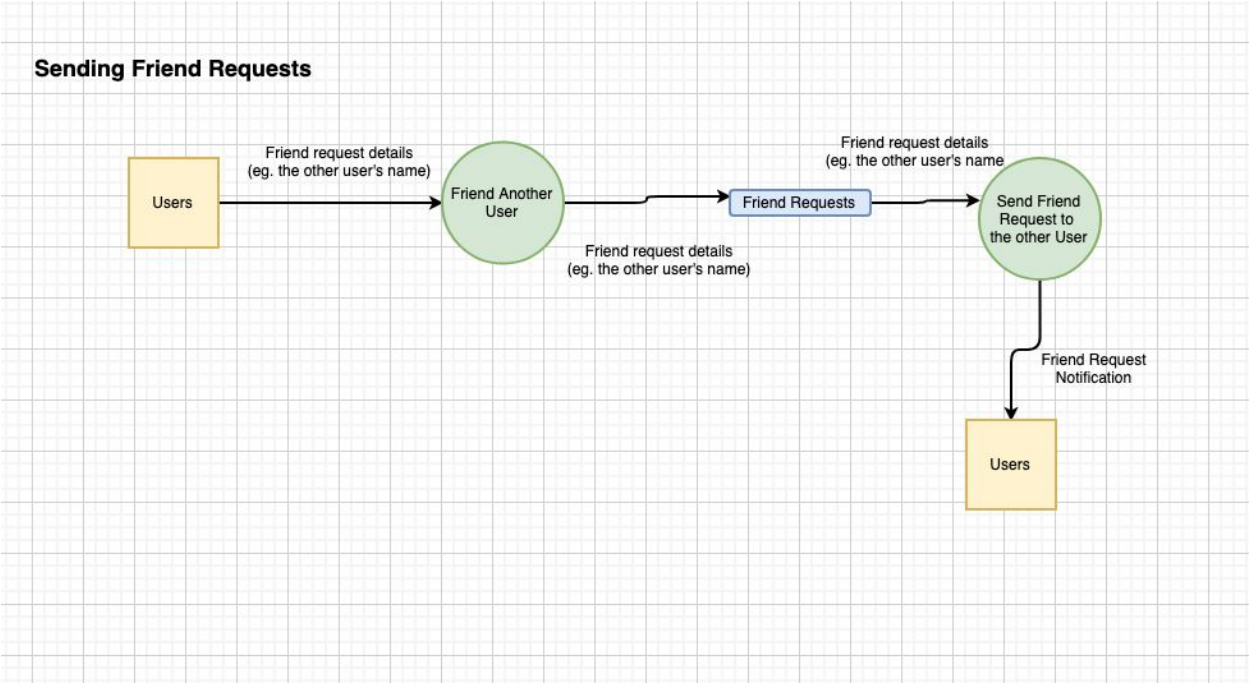
Since obtaining locations of various access points is no easy task, the team has decided to manually enter distances from a fixed point (i.e. Pitt's address)

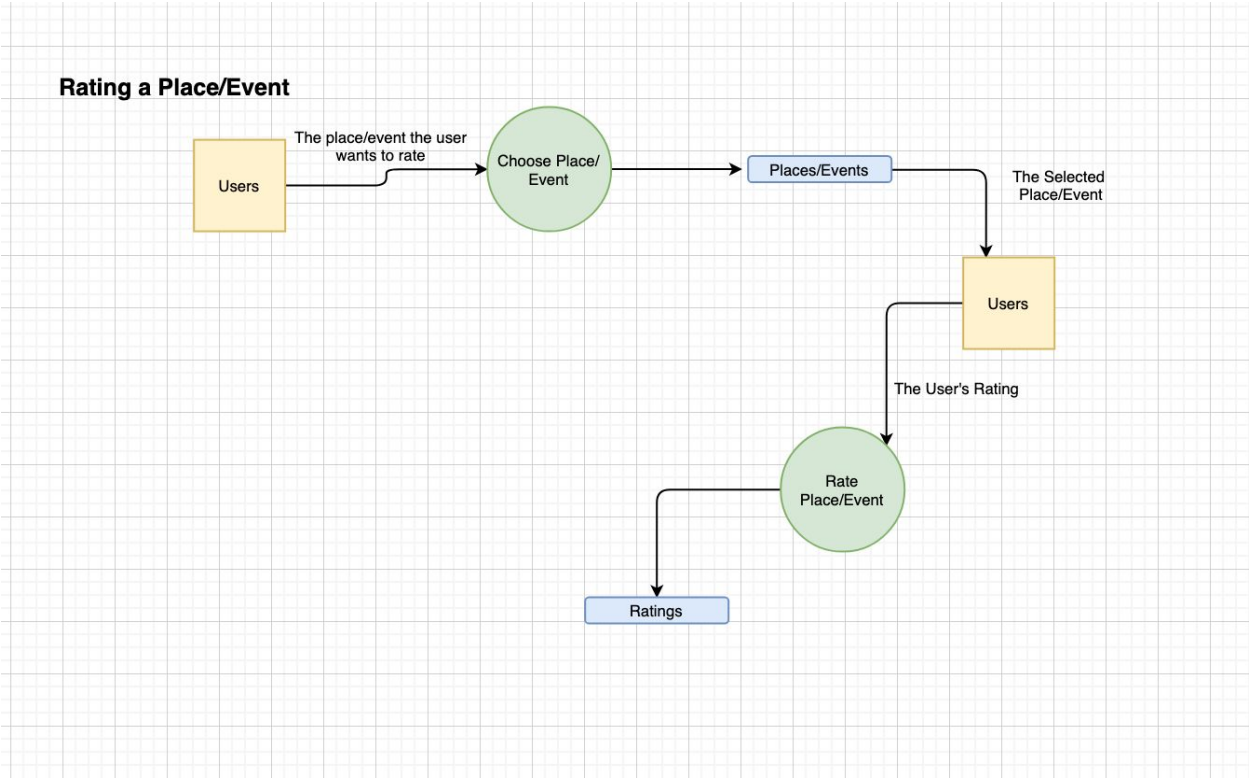
3.4 Diagrams

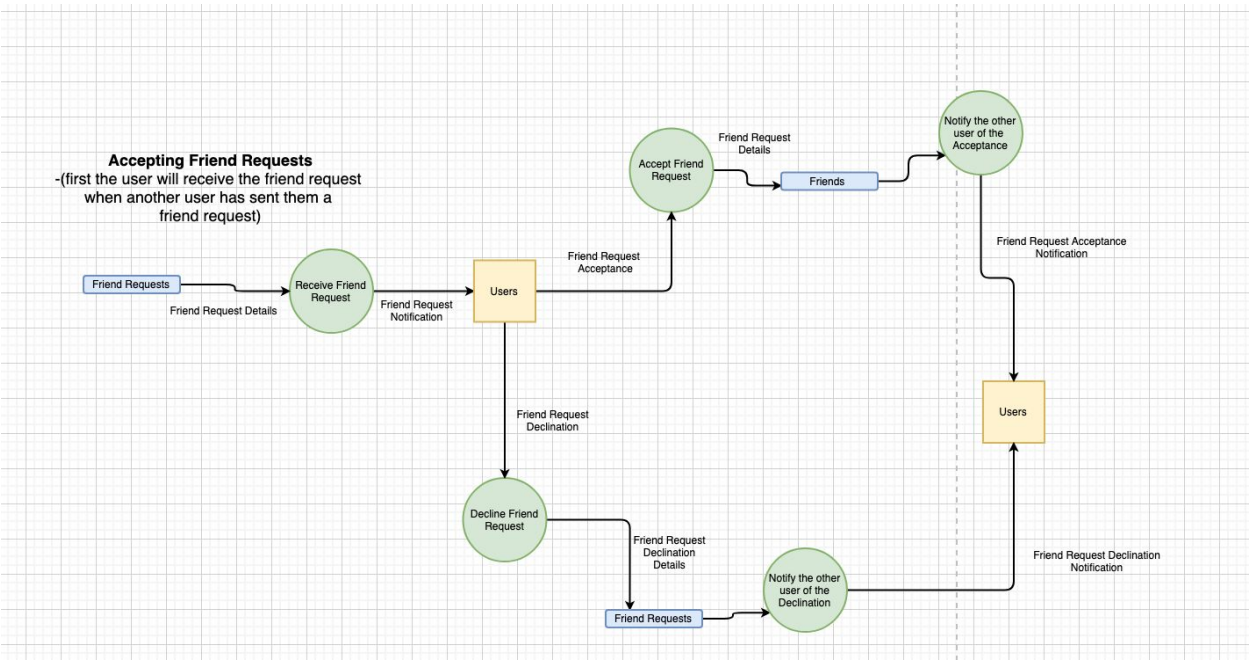

Logging In
Users
User's entered log in information
Login
LogIn Success
Users
Create Account
Registration success
User's entered log in information
Validate Login
LogIn Failure
User Accounts
User's entered log in information

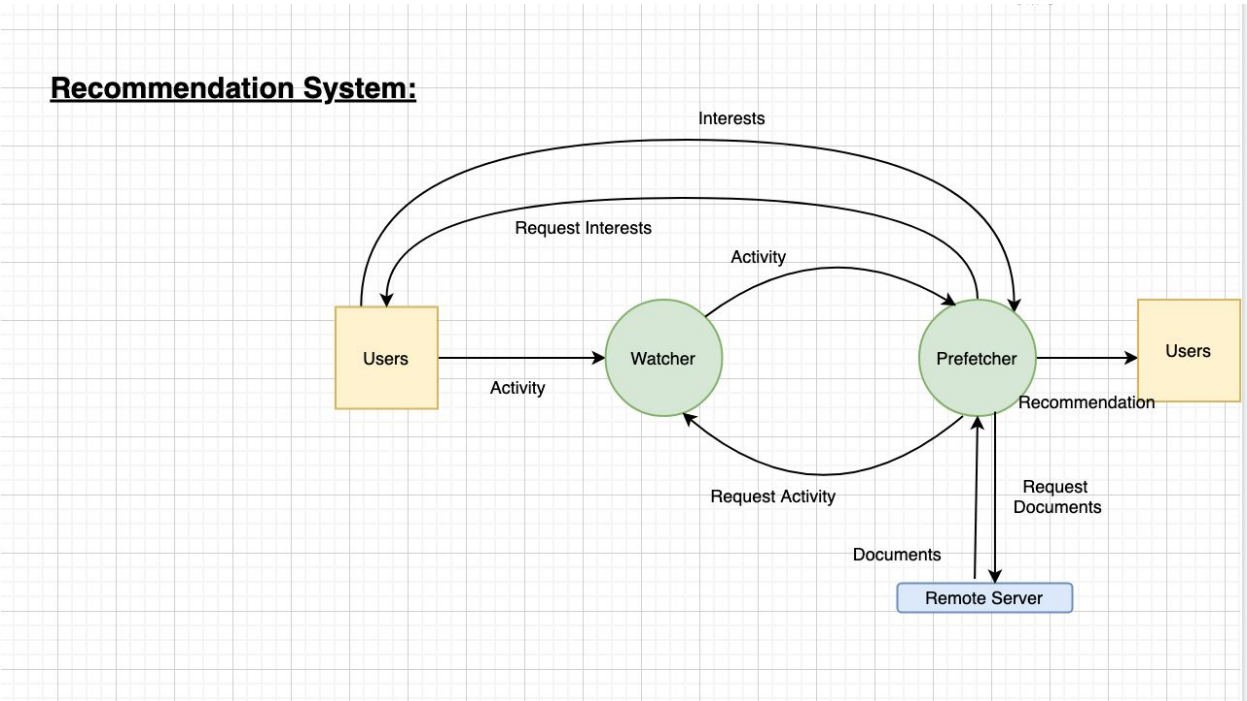

Creating an Account
Users
User's Entered Information
Create an Account
Account registration details
Users
Invalid Registration
User Accounts
Registration success


Posting an Event
Users
The event details
Post an Event
The event details
Events
The Event Details
Show event to other users
The event details
Users


Sending Friend Requests
Users
Friend request details (eg. the other user's name)
Friend Another User
Friend request details (eg. the other user's name)
Friend Requests
Friend request details (eg. the other user's name
Send Friend Request to the other User
Friend Request Notification
Users


Rating a Place/Event
Users
The place/event the user wants to rate
Choose Place/ Event
Places/Events
The Selected Place/Event
Users
The User's Rating
Rate Place/Event
Ratings


Accepting Friend Requests
-(first the user will receive the friend request when another user has sent them a friend request)
Friend Requests
Friend Request Details
Receive Friend Request
Friend Request Notification
Users
Friend Request Acceptance
Accept Friend Request
Friend Request Details
Friends
Notify the other user of the Acceptance
Friend Request Acceptance Notification
Users
Friend Request Declination
Decline Friend Request
Friend Request Declination Details
Friend Requests
Notify the other user of the Declination
Friend Request Declination Notification



4. Performance requirements
   - The System should be able to handle
     - 50 concurrent users
     - 200 user-profiles
     - Review and event posting under 10 seconds

5. Exception conditions/exception handling
   - If System crashes, the team works until it can go live again
   - If overload happens, reviews older than 3 months will automatically be deleted
   - New and non-chain restaurants and attractions will be given priority
   - Database constantly updated

6. Implementation Priorities
   - Due time constraints, we will prioritize the important functions of the website's functionality.
     - User Registration
     - User Login
     - Event Registration
     - Restaurant Registration
     - Like Functionality
     - Filtering Functionality
     - Recommender systems
   - Wish List
     - AI chat
     - Mobile app

7. Foreseeable modifications and enhancements
- Upgrade of the recommendation system algorithm
- Make all pages user friendly
- Enable users to modify and delete only the events and activities they created.

8. Acceptance criteria
Documents that will be delivered:
- Software plan
- User manual
- Test plan
- Requirements
- Sample source code
- Software design
- Test results

Tests to accept product are as follows:
- Ensure a new user can be registered
- Ensure users can create events
- Ensure admin can manage users and events on the site
- Ensure users can log into their accounts
- Ensure events and locations can be filtered
- Ensure users can befriend other users
- Ensure users can comment on and like locations/events.
- Ensure multiple concurrent users are supported
- Ensure account details and other requested information are supplied quickly

9. Sources of information

9.1 Software Vendor Resources

9.1.1 Flask

Welcome to Flask — Flask 0.10.1 documentation

9.1.2 React

Getting Started – React

9.1.3 Python and Database Resources

Our Documentation
Databases
Do You Know Python Has A Built-In Database?
Python - MySQL Database Access

9.2 Student Activity / Food Vendor / Pitt ID Resources

9.2.1 Activities On and Around Campus and Pittsburgh

9.2.1.1 Recreation and Sporting Facility Information

Facility Information

9.2.1.2 Sporting Event Discounted Activities

GNC Student Rush | Pittsburgh Penguins
Student Ticket Information - Pitt Panthers #H2P
Student Discount | Pittsburgh Pirates

9.2.1.3 Hiking and Outdoors

Hiking Trails in Laurel Highlands | Tours, Map & Information
Explore Ohiopyle State Park
Best Trails in Schenley Park - Pennsylvania
Coopers Rock State Forest - West Virginia State Parks
Mount Davis : Climbing, Hiking & Mountaineering
Hiking at McConnells Mill State Park
Exploring The Abandoned Pennsylvania Turnpike

9.2.1.4 Entertainment

The Stage AE | Pittsburgh, PA | Latest Events and Information
Petersen Events Center
Heinz Field in Pittsburgh, PA - Home of the Steelers and Panthers
Escape Room Pittsburgh - A real life immersive experience
AMC Waterfront 22 - West Homestead, Pennsylvania 15120
Pittsburgh Paintball Park - PA's Ranked #1 Newest Theme Park
Kennywood: Best Amusement Park for Kids & Families
Sports Bar - Homestead - Restaurants
Skydive Rick's
Kayak Pittsburgh
Pittsburgh International Race Complex
Randyland
All The Pittsburgh Haunted Houses, Hayrides + Fright Farms

9.2.2 Free with Pitt ID

Free Museum Visits
Home | Soldiers & Sailors Memorial Hall & Museum Trust, Inc.
The Andy Warhol Museum
Official site of the Duquesne Incline
Carnegie Museum of Natural History
Carnegie Museum of Art Connects People to Art, Ideas, and One Another
Phipps Conservatory
Welcome to Mattress Factory | Mattress Factory
Senator John Heinz History Center
Software Download Service at My Pitt
FAQ: Buses and Shuttles
Carnegie Science Center: Home

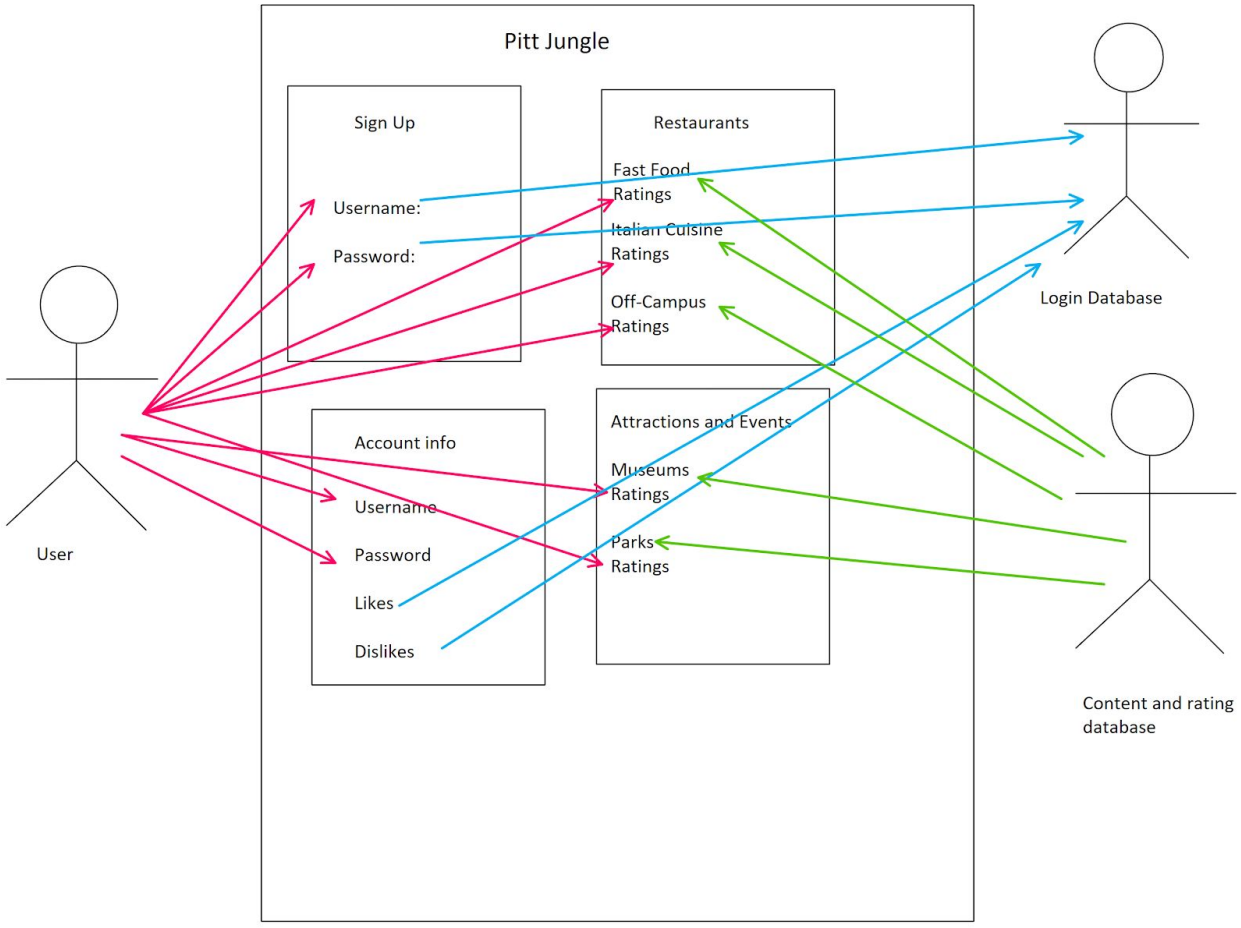
9.2.3 Food Vendor Resources

A comprehensive list containing hundreds of dining options throughout the city of Pittsburgh as well as grocery and farmers market options. Go a step further and check out the 30+ breweries/distilleries/wineries Pittsburgh has to offer.

Food (and Drink) for Thought

Object Oriented Analysis for Online Testing System

1. System Overview

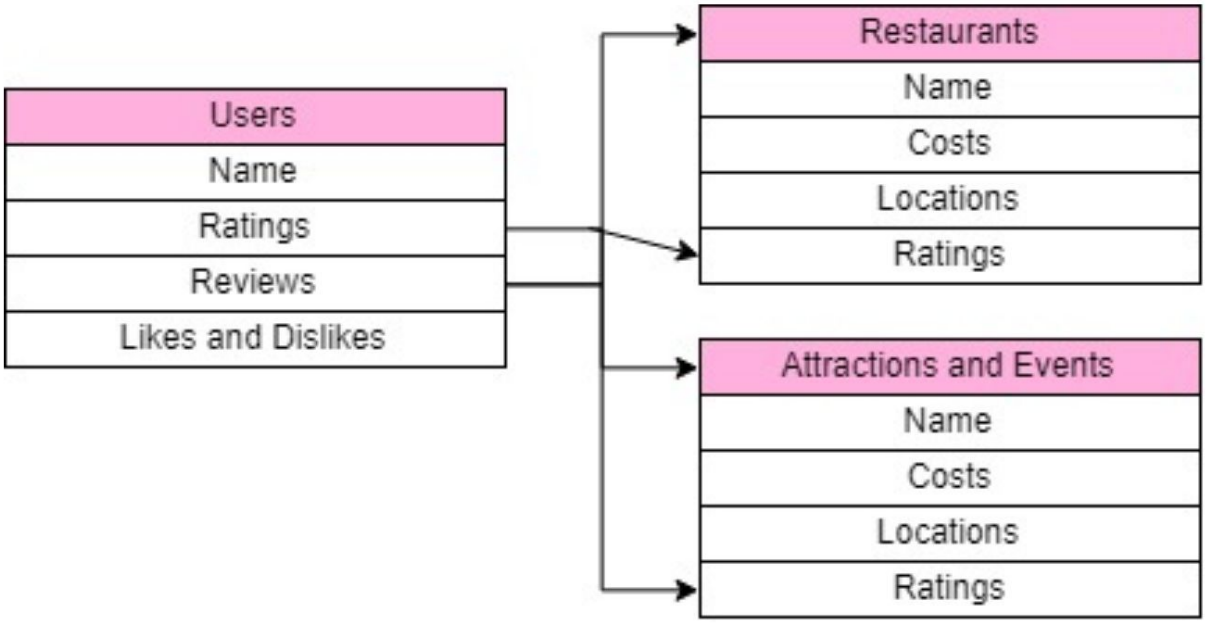


2. The Class Model

2.1. The Classes

- Users(name, ratings, reviews, likes and dislikes)
- Restaurants(name, costs, location, ratings)
- Attractions and Events(name, costs, location, ratings)

2.2. The Class Diagram



3. The Dynamic Model

3.1. The Scenarios

3.1.1 User-side scenarios

- User logs in
- The user password is wrong
- The user's username is wrong
- The user registers an account
- The user confirms their email address by clicking the link from email.
- The user cannot confirm their email address by clicking the link from email.
- The user likes a restaurant or an event.
- The user changes personal information.
- The user filters events or restaurants by their preferences.
- The user becomes friends with another user.

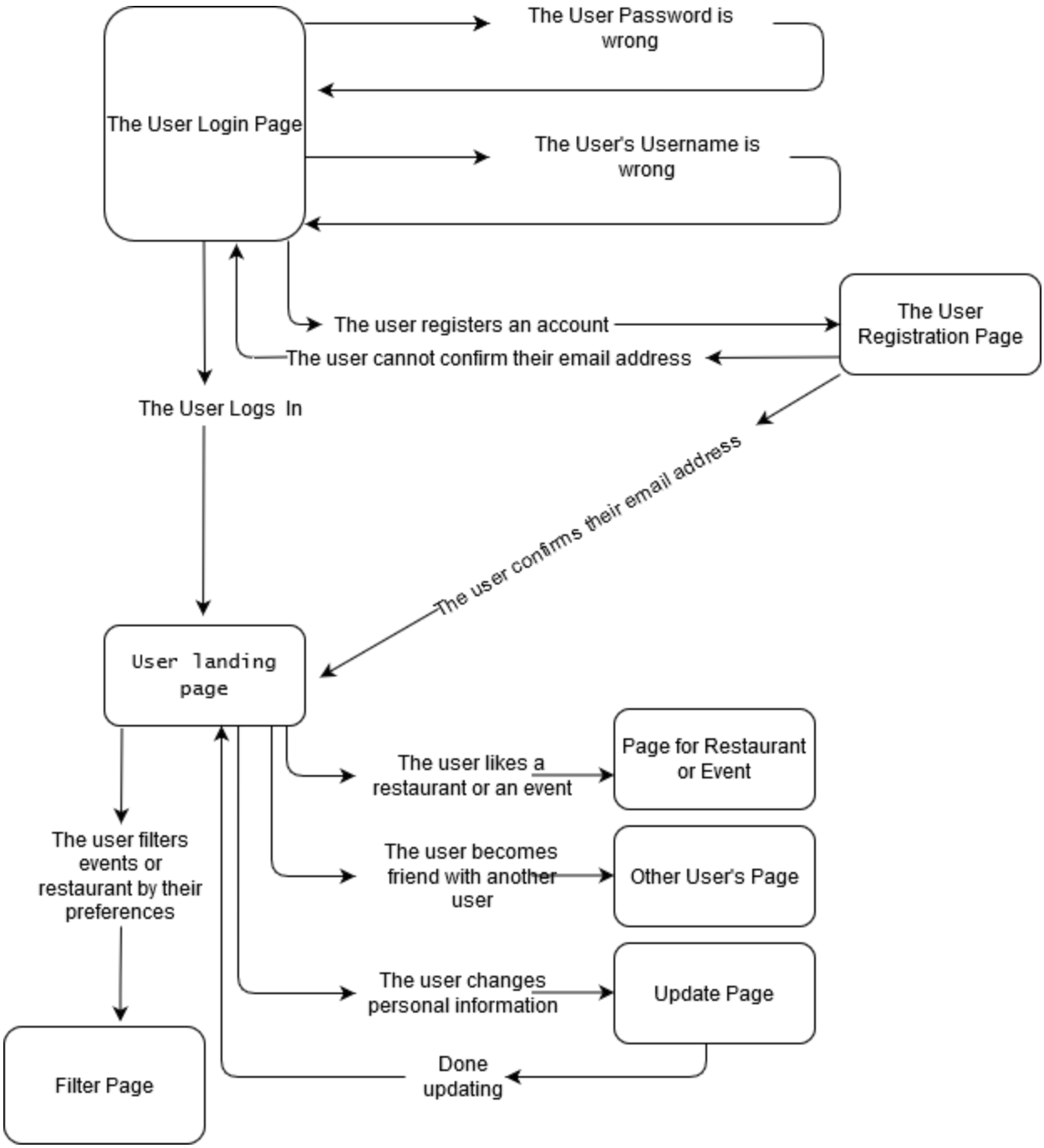
3.1.2 Partner-side scenarios. Partners are restaurant managers or event organizers.

- The partner logs in
- The partner's password is wrong
- The partner's username is wrong
- The partner registers an account
- The partner confirms their email address by clicking the link from email

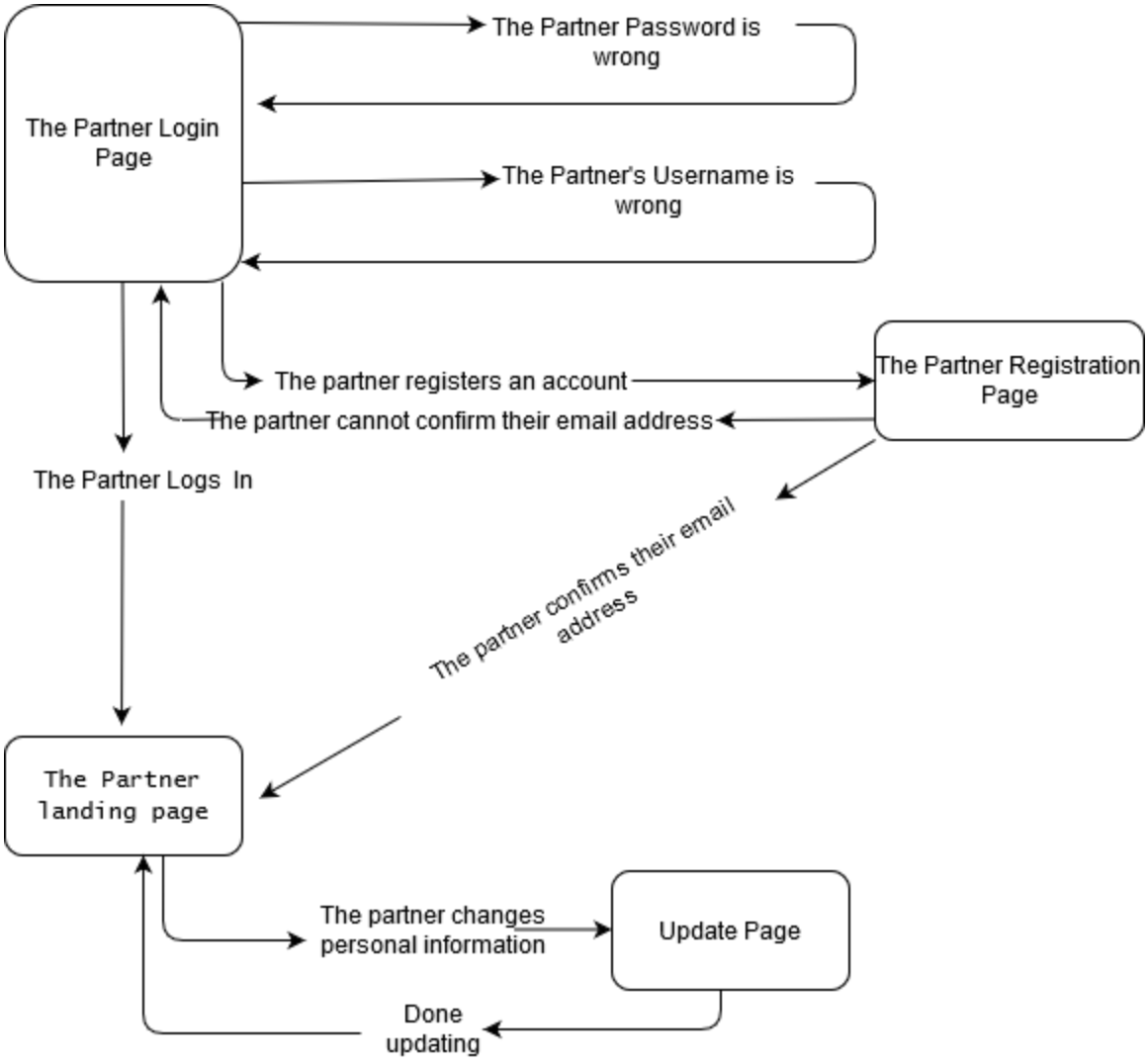
- The partner cannot confirm their email address by clicking the link from email
- The partner updates information of the restaurant or the event

3.2. The State Diagrams

3.2.1 User-side scenarios state diagram

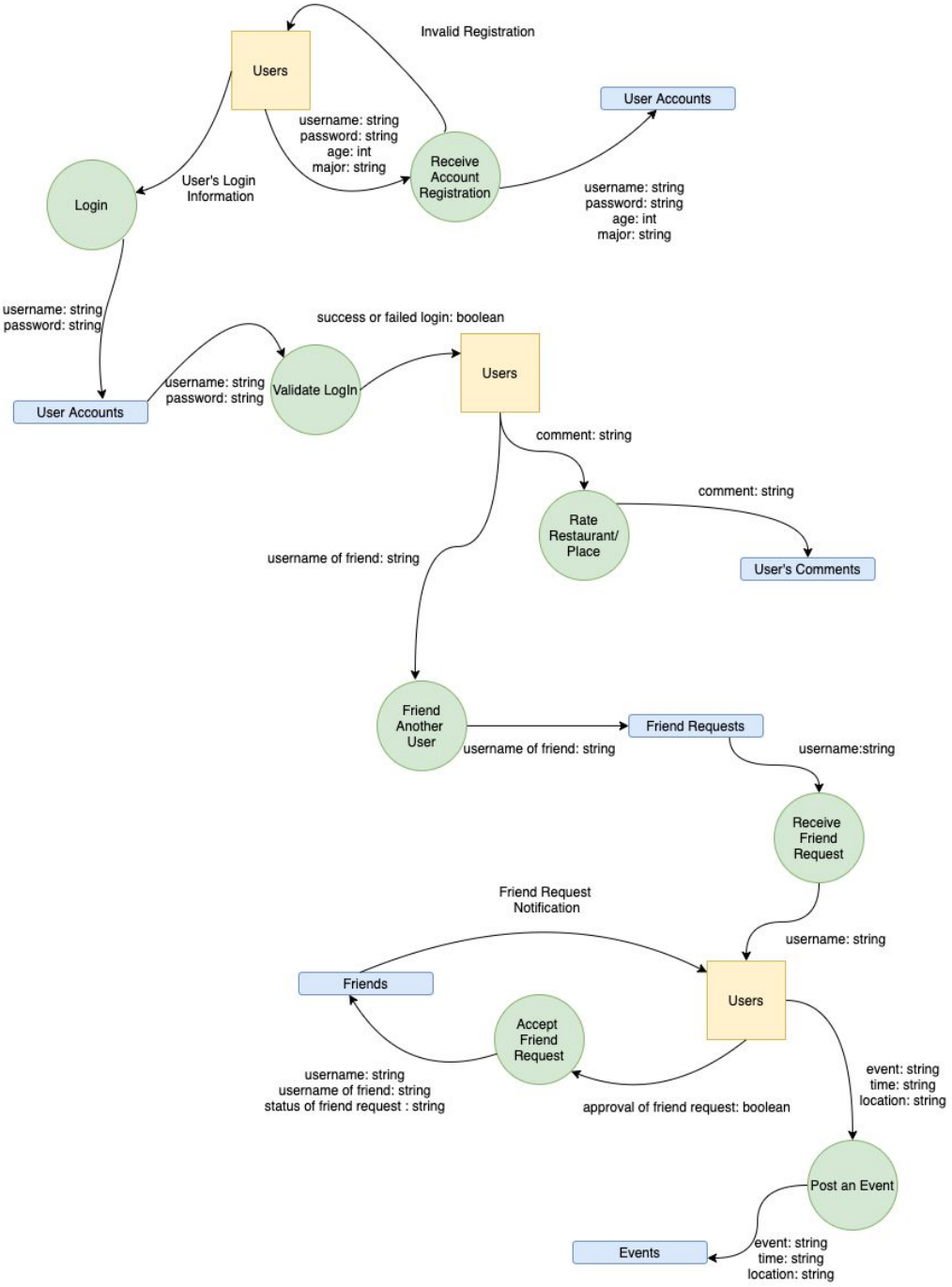


3.2.2 Partner-side scenarios state diagram

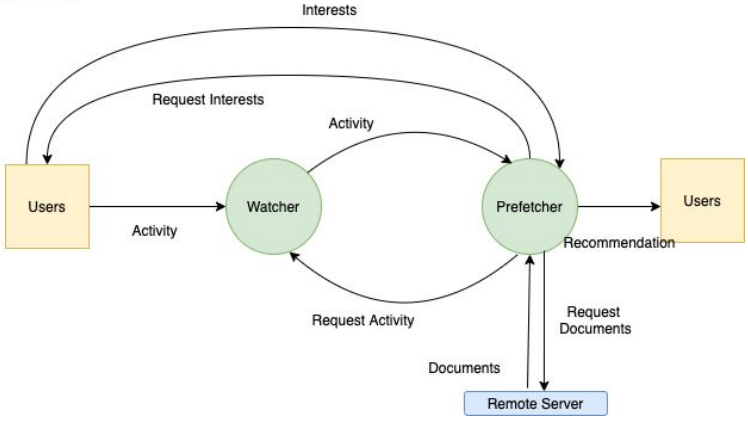


4. The Functional Model

Our functional model shows the different functions within our application and how the different functions connect to each other.



Recommendation System:



Object-Oriented Design for Online Testing System

1.

Module Name: create_event
Module Type: Method
Return Type: Boolean
Input arguments: Event Name, Event Date, Occupancy Limitation
Output arguments: Boolean if Event Created Successfully
Error messages: Message if Event could not be posted
Files accessed: None
Files changed: None
Modules called: User Database, Event Database
Narrative: Method used by users to create/post their own event. The Module will add event link to respective user in user database as well as adding it to the event database

2.

Module Name: add_friend
Module Type: Method
Return Type: Boolean
Input arguments: User id
Output arguments: Boolean stating whether two users are friends
Error messages: Message if Friend could not be added or if already friends
Files accessed: None
Files changed: None
Modules called: User Database
Narrative: A user will be able to click an “add friend” button which will pass/fetch information from the User database. This will confirm if a link has been set up between two users or not, thus verifying friend status.

3.

Module Name: Event Database
Module Type: Object
Return Type: void
Input arguments: None
Output arguments: None
Error messages: None
Files accessed: None
Files changed: None
Modules called: None
Narrative: This is the database that contains all events

4.
Module Name: register_account
Module Type: Method
Return Type: Boolean
Input arguments: New User Information
Output arguments: Boolean indicating registration success/failure
Error messages: Message if invalid registration occurs
Files accessed: None
Files changed: None
Modules called: User Database
Narrative: This method will check the "New_User" information within the user database to make sure an account does not already exist with the provided information and create the necessary user if information does not already exist

5.
Module Name: login
Module Type: Method
Return Type: Boolean
Input arguments: User Credentials [Username, Password]
Output arguments: Boolean Indicating Valid/Failed Login
Error messages: Message if username/password do not exist
Files accessed: None
Files changed: None
Modules called: User Database
Narrative: Valid Username and Password must be entered to login to the system

6.
Module Name: Watcher
Module Type: Object
Return Type: void
Input arguments: None
Output arguments: User activity
Error messages: None
Files accessed: None
Files changed: None
Modules called: Prefetcher, User Database
Narrative: Monitors the activity of users and interacts with the prefetcher in the recommendation system

7.
Module Name: filter
Module Type: Method
Return Type: Ordered List of items
Input arguments: Specified tags to filter by (Food, Hiking, Events, etc)
Output arguments: None
Error messages: None
Files accessed: Food/Event/Activity Databases
Files changed: None
Modules called: None
Narrative: A user can select tags to filter by which will sort a page based on the requested filtering tags and display the most relevant to the user

8.
Module Name: leave_ratings
Module Type: Method
Return Type: void
Input arguments: Numbered rating for the given object of interest
Output arguments: None
Error messages: None
Files accessed: Event/Activity/Food/User Database
Files changed: Event/Activity/Food/User Database
Modules called: None
Narrative: A user will be able to choose whether or not to leave a numbered or starred rating for the given object of interest. This will update the databases to reflect ratings and user interests/disinterests

9.
Module Name: Admin_login
Module Type: Method
Return Type: Boolean
Input arguments: Name, Password, Pin
Output arguments: Boolean indicating successful/failed admin login
Error messages: Message if invalid admin
Files accessed: User Database
Files changed: None
Modules called: Returning_User
Narrative: Admin logins will be slightly different to better verify if admin is logging in. The fields that will be compared to in the user database will be name, password and pin

10.
Module Name: User_Display_Recommendations
Module Type: Method
Return Type: Object/List
Input arguments: User interests and/or page views/visits
Output arguments: Object/List of probability calculated likelihood of user interests
Error messages: None
Files accessed: User Database
Files changed: None
Modules called: None
Narrative: This method will calculate a recommendation page based on what the user has stated for interests and/or what the user visits more likely than other pages. This will be part of the recommender system and will use probabilistic analysis to gauge what to show

11.
Module Name: User_Update_Recommendations
Module Type: Method
Return Type: None
Input arguments: Tags from items view
Output arguments: None
Error messages: None
Files accessed: None
Files changed: None
Modules called: User Database
Narrative: This method will add additional interests to a specific user's entry in the user database. This will allow a more continuous feedback loop to have a more real-time recommendation system and therefore a less biased recommendation (with more more “interests” being added for the user)

12.
Module Name: User Database
Module Type: Object
Return Type: void
Input arguments: User id [Identification]
Output arguments: Users data
Error messages: None
Files accessed: None
Files changed: None
Modules called: None

Narrative: This is the database for all users. Gets accessed by different modules to supply specific user data.

13.

Module Name: User
Module Type: Object
Return Type: void
Input arguments: None
Output arguments: None
Error messages: None
Files accessed: None
Files changed: None
Modules called: create_event, add_friend, Prefetcher, Watcher
Narrative: Sends interests to prefetcher, post events and sends friend requests.

14.

Module Name: Prefetcher
Module Type: Object
Return Type: void
Input arguments: None
Output arguments: Recommendation documents
Error messages: None
Files accessed: None
Files changed: None
Modules called: Remote_server, User, Watcher
Narrative: Prefetches documents for users and interacts with the server and watcher to do so. These documents are the recommendations.

Test Plan

1. Purpose
    1.1. This test plan applies to the application PittJungle. It will test the functions and usability.

2. Unit Testing
    2.1. User Module: This module maintains the user's account information
        2.1.1. Equivalence classes for the user module:
            2.1.1.1. Equivalence classes for item "name":
                2.1.1.1.1. String with the user's name - acceptable
                2.1.1.1.2. anything else (int, boolean, float, etc.) - error
            2.1.1.2. The equivalence classes for item "age":
                2.1.1.2.1. Integer with 2 digits maximum - acceptable
                2.1.1.2.2. Anything else (String, boolean, integer with more than 2 digits) - error
            2.1.1.3. The equivalence classes for item "college":
                2.1.1.3.1. String with user's college - acceptable
                2.1.1.3.2. anything else (int, boolean, float, etc.) - error
        2.1.2. Checklist: The following black-box tests will be performed.
            2.1.2.1. (✔) Create a new user account
            2.1.2.2. (✔) Users can log in with their account information

    2.2. Place Module: This module maintains the different places that users' can view
        2.2.1. Equivalence classes for the place module:
            2.2.1.1. Equivalence classes for item "name"
                2.2.1.1.1. String with the name of the place - acceptable
                2.2.1.1.2. Anything else (int, boolean, float, etc.) - error
            2.2.1.2. The equivalence classes for item "location"
                2.2.1.2.1. String with the location of the place - acceptable
                2.2.1.2.2. anything else (int, boolean, float, etc.) - error
            2.2.1.3. The equivalence classes for item "tags"
                2.2.1.3.1. Must be a String within the predefined tags list (eg. "Nature", "Science", "Art") - acceptable
                2.2.1.3.2. Any other String, int, boolean, etc. - error
        2.2.2. Checklist: The following black-box tests will be performed
            2.2.2.1. (✔) Users can view and filter out places based on tags

    2.3. Rating Module: This module maintains the rating user can give for a place
        2.3.1. Equivalence classes for the rating module

2.3.1.1. Equivalence classes for item “score”

2.3.1.1.1. An integer from 0-10 - acceptable

2.3.1.1.2. Anything else (String, boolean, etc.) - error

2.3.1.2. Equivalence classes for item “comment”

2.3.1.2.1. String with a length that is less than or equal to a length of 250 - acceptable

2.3.1.2.2. Anything else (String with a length greater than 250, int, boolean etc.) - error

2.3.2. Checklist: The following black-box tests will be performed

2.3.2.1. (X) Users can rate a place or an event
This has been removed from our features
Deemed unnecessary and no added benefit

2.4. Event Module: This module maintains and contains the information/details of a specific event

2.4.1. Equivalence classes for the event module

2.4.1.1. Equivalence classes for item “Title”

2.4.1.1.1. String - acceptable

2.4.1.1.2. Anything else (int, double, boolean etc.) - error

2.4.1.2. Equivalence classes for item “Location”

2.4.1.2.1. String - acceptable

2.4.1.2.2. Anything else (int, boolean etc.) - error

2.4.1.3. Equivalence classes for item “Date”

2.4.1.3.1. MM/DD/YYYY String in this format - acceptable

2.4.1.3.2. Anything else (int, boolean, string with a different format) - error

2.4.2. Checklist: The following black-box tests will be performed

2.4.2.1. (✔) Users can post events

2.4.2.2. (✔) Users can view events posted by other users

2.5. Testing will be done by Sherryl Augustine, Sushrati Bansod, Luiza Urazaeva, David Ladeji, Trent Lessig

3. Integration Testing

3.1. Purpose

When each module is added or when any module is changed, the test plan ensures all the modules work.

3.2. Checklist

- add Permissions module and run all test cases for Permissions.

- add Role module and run all test cases for Permissions and Role module.

- add User module and run all test cases for the Login and User modules.

- add Follow module and repeat all test cases for the Login, User, Role and Follow modules

- add Comment module and repeat all test cases for the Login, User, Role and Comment modules.

- add the Filter module and repeat all test cases for the Login, User, Role, and Filter modules.

- add Partner module and repeat all test cases for the Login, Partner, Role, modules.

- add Restaurant Profile Module and repeat all test cases for the Login, Partner, Role, and Restaurant Profile Module.

- add Event Profile Module and repeat all test cases for the Login, the Role, Partner, and Event Profile Modules.

- add the Anonymous User module and repeat all test cases for the Permissions, Role, Filter, and Anonymous User modules.

- add Admin module and repeat all test cases for the Login, User, Role, Permissions, Follow, Comment, Filter, Partner, Restaurant Profile, Event Profile, Anonymous User, and Admin modules.

3.3. Who will perform the tests: Sherryl Augustine, Sushrati Bansod, Luiza Urazaeva, David Ladeji, Trent Lessig

4. System Testing

4.1. Purpose - The following tests are designed to ensure the system functions as a whole in the manner dictated by the specifications. They are to be performed in a top-down fashion with the intent of replicating a typical user's experience with the entire system upon release.

4.2. System Testing Checklist

✔	Enter the product's URL in a web browser and test that the login page displays properly

✔	Click the “Sign Up Now” link to load the register new account screen
✔	On the create new account screen, enter a valid email address and password
✔	Provide all specified information in the proper fields and click “Sign Up” which directs you to the login page to log into the new account
✔	Enter valid email and password for a registered user and click “Log In” which directs you to the homepage of the application
✔	Click on the “Attractions” link in the navigation bar which directs you to the Attractions page
✔	Click on the “Events” link in the navigation bar which directs you to the Events page
✔	Select any event object to be directed to its page which displays more information on the event
✔	Select the event’s author’s username to be directed to their account page
✔	Click on the “Follow” button to start following this user
✔	Click on the “Restaurants” link in the navigation bar which directs you to the Restaurants page
✔	Click on “My Account” which directs you to your account’s profile page
✔	Click on “Update Account Info” to change account information
✔	Provide all specified information in the proper fields and click “Update” which directs you to your account page
✔	On the navigation bar, click on “PittJungle” to be directed to the home page
✔	Click “Log out” to return to the home page

4.3. Who will perform the tests: Sherryl Augustine, Sushrati Bansod, Luiza Urazaeva, David Ladeji, Trent Lessig

5. Acceptance Testing

5.1. Purpose

The purpose of the acceptance testing portion of our test plan is meant to verify the functionality of the system in the sense that it meets the requirements laid out

throughout the semester. All acceptance tests that are run should be achieved within an adequate level

5.2. Acceptance Testing Checklist

	Unit, Integration and System Checklists are complete
	Ensure upwards of 300 users can be created/stored in the database
	Ensure upwards of 300 places/events/activities/food options can be created and stored in a database
✔	Test speed of the pages and make sure no page takes more than 7 seconds to load
	Ensure that at least 40 users can be logged in at once
✔	Ensure recommender system can pull tags from individual user clicks to update probability of recommending something
✔	Ensure recommender system stores “click” info with individual user in the database
	Ensure a variety of initial places/activities/food venues to allow variety (at least 15 per classifying tag)
✔	Ensure recommendations are placed on the home screen (Once data is collected, show top recommendation and when enough data is collected, increase the amount of shown recommendations to no more than 5 recommendations)

5.3. Who will perform the tests: Sherryl Augustine, Sushrati Bansod, Luiza Urazaeva, David Ladeji, Trent Lessig

Test Cases Performed/Conducted

Login

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 1 Black box	Unit Test - Test email	Empty $email	Show error message	11/08/2020 ✔	None	Prints out “Please fill out this field”

Test 2 White box	Unit Test - Test email	Non format $email	Show error message	11/08/2020 ✔	None	States Invalid email
Test 3 Black box	Unit Test - Test email	Unregistered $email	Show error message	11/06/2020 ✔	None	Screen waiting for confirmation
Test 4 Black box	Unit Test - Test email	Registered $email	success	11/06/2020 ✔	None	Full access to site
Test 5 Black box	Unit Test - Test password	Empty $pw	Show error message	11/08/2020 ✔	None	Invalid password or email
Test 6 Black box	Unit Test - Test password	Invalid $email & $pw	Show error message	11/08/2020 ✔	None	Invalid entries
Test 7 Black box	Unit Test - Test password	Valid $email & $pw	success	11/06/2020 ✔	None	Logs in
Test 8 Black box	System Test - Sign up	Click link	Direct to register page	11/08/2020 ✔	None	Redirects

Register new user

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 9 Black box	Unit Test - Test username	Empty $username	Show error message	11/08/2020 ✔	None	Asks to fill out the field
Test 10 White box	Unit Test - Test username	$username less than 2 chars	Show error message	11/08/2020 X	Sent confirmation email	TBD
Test 11 White box	Unit Test - Test username	$username of 2 chars	success	11/08/2020 ✔	None	Registers and sends confirm email
Test 12 White box	Unit Test - Test username	$username of 20 chars	success	11/08/2020 ✔	None	Registers and sends confirm email
Test 13 White box	Unit Test - Test username	$username more than 20	Show error message	11/08/2020 ✔	None	“Usernames must be letters,

		chars				numbers, characters of length x"
Test 14 Black box	Unit Test - Test email	Empty $email	Show error message	11/08/2020 ✔	None	"Please fill out field"
Test 15 Black box	Unit Test - Test email	Non format $email	Show error message	11/08/2020 ✔	None	Invalid Email message
Test 16 Black box	Unit Test - Test email	Unregistered $email	Show error message	11/08/2020 ✔	None	Says to confirm email to continue
Test 17 Black box	Unit Test - Test email	Registered $email	success	11/08/2020 ✔	None	Email already exists
Test 18 Black box	Unit Test - Test password	Empty $pw	Show error message	11/08/2020 ✔	None	Please fill out field
Test 19 Black box	Unit Test - Test confirm password	Empty $pw2	Show error message	11/08/2020 ✔	None	Please fill out field
Test 20 Black box	Unit Test - Test passwords	Same $pw and $pw2	success	11/08/2020 ✔	None	Registers user
Test 21 Black box	Unit Test - Test passwords	$pw different from $pw2	Show error message	11/08/2020 ✔	None	Message stating Passwords must match
Test 22 Black box	Unit Test - Test username	$username not unique	Show error message	11/08/2020 ✔	None	Username exists already in the database
Test 23 Black box	Unit Test - Test username	Valid $username	success	11/08/2020 ✔	None	Registers user if unique
Test 24	Unit Test -	$email not	Show error	11/08/2020	None	Email

Black box	Test email	unique	message	✔		exists message
Test 25 White box	Unit Test - Test age	$age more than 99	Show error message	--	--	--
Test 26 Black box	Unit Test - Test age	String value for $age	Show error message	--	--	--
Test 27 White box	Unit Test - Test age	$age of 0	Show error message	--	--	--
Test 28 Black box	Unit Test - Test age	Valid $age	success	--	--	--
Test 29 Black box	System Test - Sign In	Click Link	Direct to login page	11/08/2020 ✔	None	Login form appears

Update Account Info

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 30 Black box	Unit Test - Test username	No change on $username	success	11/08/2020 ✔	None	No error
Test 31 Black box	Unit Test - Test username	Non unique $username	Show error message	11/09/2020 ✔	None	Username Already Exists
Test 32 Black box	Unit Test - Test username	Empty $username	Show error message	11/08/2020 ✔	None	Field needs to be filled out message
Test 33 White box	Unit Test - Test username	$username less than 2 chars	Show error message	11/08/2020 X	Username under profile updated	TBD
Test 34 White box	Unit Test - Test username	$username of 2 chars	success	11/08/2020 ✔	None	Account updated
Test 35 White box	Unit Test - Test username	$username of 20 chars	success	11/08/2020 ✔	None	Account updated

Test 36 White box	Unit Test - Test username	$username more than 20 chars	Show error message	11/08/2020 ✔	None	Account not updated
Test 37 Black box	Unit Test - Test username	New unique $username	success	11/08/2020 ✔	None	Username updated
Test 38 Black box	Unit Test - Test email	No change on $email	success	11/08/2020 ✔	None	No change
Test 39 Black box	Unit Test - Test email	Non unique $email	Show error message	11/08/2020 ✔	None	Exists in database
Test 40 Black box	Unit Test - Test email	Non format $email	Show error message	11/08/2020 ✔	None	Invalid format for email message
Test 41 Black box	Unit Test - Test email	Unregistered $email	Show error message	11/09/2020 ✔	None	Invalid Email
Test 42 Black box	Unit Test - Upload picture	No change to $picture	Show error message	TBD	TBD	TBD
Test 43 White box	Unit Test - Upload picture	$picture not a jpg or png	Show error message	TBD	TBD	TBD
Test 44 Black box	Unit Test - Upload picture	Valid $picture	Success	TBD	TBD	TBD
Test 45 Black box	System Test - Delete account	Click the Delete button	Modal opens up	TBD	TBD	TBD
Test 46 Black box	System Test - Delete account	Click the Cancel button	Modal closes	TBD	TBD	TBD
Test 47 Black box	System Test - Delete account	Click the Delete button	Account gets deleted. Redirects to login page	TBD	TBD	TBD

Comment

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 48 Black box	Unit Test - Add Comment	Empty textfield	Show error message	11/09/2020 ✔	None	Textfield needs to be filled out to submit comment
Test 49 White box	Unit Test - Add Comment	More than 500 chars	Show error message	11/09/2020 X	Comment Posted	TBD
Test 50 White box	Unit Test - Add Comment	Exactly 500 chars	success	11/09/2020 ✔	None	Comment Posted
Test 51 White box	Unit Test - Add Comment	Less than 2 chars	Show error message	11/09/2020 X	Comment Posted	TBD
Test 52 White box	Unit Test - Add Comment	Exactly 2 chars	success	11/09/2020 ✔	None	Comment Posted
Test 53 Black box	Unit Test - Edit Comment	Empty textfield	Show error message	11/09/2020 ✔	None	Asks to fill out comment field if nothing
Test 54 White box	Unit Test - Edit Comment	More than 500 chars	Show error message	11/09/2020 X	Allows Edit	TBD
Test 55 White box	Unit Test - Edit Comment	Exactly 500 chars	success	11/09/2020 ✔	None	Comment Posted
Test 56 White box	Unit Test - Edit Comment	Less than 2 chars	Show error message	11/09/2020 X	Allows Edit	TBD
Test 57 White box	Unit Test - Edit Comment	Exactly 2 chars	success	11/09/2020 ✔	None	Comment Posted
Test 58 Black box	System Test - Delete comment	Click the Delete button	Modal opens up	11/09/2020 ✔	None	“Disabled by admin”
Test 59 Black box	System Test - Delete comment	Click the Delete button	Comment deleted	11/09/2020 ✔	None	“Disabled by admin”

Test 60 Black box	System Test - Delete comment	Click the Cancel button	Modal closes	11/09/2020 ✔	None	“Disabled by admin”

Recommendation System

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 61	Unit Test - Food Probability	User’s food tags	Success: Probability associated with each food classification	11/09/2020 ✔	None	Can be viewed on screen
Test 62	Unit Test - Activity Probability	User’s Activity Tags	Success: Probability associated with each activity classification	11/09/2020 ✔	None	Can be viewed on screen
Test 63	Unit Test - New Clicks	Tags obtained from click	Success: An updated probability distribution	11/09/2020 50/50	Not all pages done to fully test individual tags	Pages need to be completed
Test 64	Unit Test - Add User ID	Integer	Success: An integer associated with each user	X - is not necessary	None	Has been added to User database model
Test 65	Unit Test - Add User ID	String	Error	X - no longer needed	None	Has been added to User Database Model

Test 66	Unit Test - Display Recommendation	Highest Individual user probability	Success	11/09/2020 ✔	None	Distribution shown on screen
Test 67	Unit Test - Display Recommendations	Top 3 Probabilities	Success	TBD	TBD	TBD
Test 68	Unit Test - Display Recommendation	None	Success	11/09/2020 ✔	None	Distribution shown on screen
Test 69	Unit Test - Probability Distribution	Probabilities	Success: Written to file to verify	11/09/2020 ✔	None	File shows up in project folder
Test 70	Unit Test - New Clicks Added to User	Tags associated with clicks	Success: Updated individual user's database	11/09/2020 ✔	None	Tags update by specified value

User Event

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 71	Unit test - Enter Title	String	Success	11/09/2020 ✔	None	Submission allowed
Test 72	Unit Test - Enter Title	Non-string	Error Message	11/09/2020 X Resolved 11/09/2020 ✔	Numbers worked	Made input location convert to string so this issue is resolved
Test 73	Unit Test - Location	String	Success	11/09/2020 ✔	None	Submits with no error

Test 74	Unit Test - Location	Non-String	Error Message	11/09/2020 X Resolved 11/09/2020 ✔	Same issue with test 72	Resolved by making input fields convert everything to string (Test case now obsolete)
Test 75	Unit Test - Date	YYYY-MM-DD String Format	Success	11/08/2020 ✔	none	Submits date
Test 76	Unit Test - Date	Anything else	Error Message	11/09/2020 ✔	None	Error Message given
Test 77	Unit Test - Click Post	Button Click	No Error	11/09/2020 ✔	None	Event Posted, Can be seen upon return to event page

Rating

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 78	Unit Test - Give Rating	Integer between 0-10	Success	TBD	TBD	TBD
Test 79	Unit Test - Give Rating	Integer <0 or > 10	Error: invalid rating	TBD	TBD	TBD
Test 80	Unit Test - Give Rating	Non-Integer	Error: invalid format	TBD	TBD	TBD

Place

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 81	Unit Test - Place Name	String Length >0 & <150	Success	11/09/2020 ✔	None	Gets posted to restaurant page

Test 82	Unit Test - Place Name	String <0 | > 150	Error Message: Invalid length	11/09/2020 ✔	None	Gives max length user can submit as place name
Test 83	Unit Test - Place Name	Integer, boolean, double, etc	Error: Invalid format	11/09/2020 X Resolved 11/09/2020 ✔	Numbers allowed	Input fields now convert everything to string during submission
Test 84	Unit Test - Place Location	String Length >0 & <200	Success	11/09/2020 ✔	None	Submits with form
Test 85	Unit Test - Place Location	String Length <0 | >200	Error: Invalid Length	11/09/2020 ✔	None	Gives max length location can be for the place
Test 86	Unit Test - Place	Integer, Float, Double, boolean, etc	Error: invalid format	11/09/2020 X Resolved 11/09/2020 ✔	Numbers allowed	Input fields now convert everything to string during submission
Test 87	Unit Test - Item Tag	String from Predefined list	Success	11/09/2020 ✔	None	Gets added to tag list on place
Test 88*	Unit Test - Item Tag	Anything outside of predefined list	Error: Tag is invalid	11/09/2020 X	Any tag name is added	TBD

User

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
Test 89	Unit Test - Name	String	Success	11/09/2020 ✔	None	Name shows to screen
Test 90	Unit Test - Name	Integer, Boolean, Float,	Error: Must be of format	11/09/2020 ✔	None	Name must be

		Double, etc	string			letters
Test 91	Unit Test - College	String	Success	TBD	TBD	TBD
Test 92	Unit Test - College	Integer, Boolean, Float, Double	Error: Must be of format string	TBD	TBD	TBD

Explanation for test cases not tested and some limitations:

Test cases 25 through 28 were originally thought to be necessary, but for functional and further developmental operations, this was chosen to be eliminated. If, by the final demo, this is decided to be added as an additional layer, it will be done, but the team has decided that these test cases would not hinder a final working product by removing them. Test cases 42 through 47 are wishlist tests, so they have not yet been tested. It is an item the team would like to add by the end of the project submission because it would add further user functionality (adding a picture to the users profile and/or deleting their account). Test cases 58 to 60 are limited to an admin user (where the tests have been performed logged in under an admin user). The team understands that this would also be suited for a regular user to be able to delete their own comments, so it has been added to the wishlist to expand this feature to regular users as well. But for now, it is a constraint placed on the user that they cannot delete their comments unless an admin disables/enables them.

Test case 67 regarding the recommender system was unable to be tested at this time. Clicks for general pages/tags were updating within the users model, but advanced testing of seeing if one specific value based on a specific click has yet to be tested. The recommender does, however, print out the recommendation in words as well as the weight that choice has compared to the others. The probability distribution of all related items are printed along with it, as well as written to a file every time a user accesses the recommendation page, to show updates are occuring. Test cases 78 to 80 were also unable to be tested at this time for the rating feature. This is still being worked on and integrated into the restaurants, attractions and events pages. Test case 88 may be changed so that any tag can be added as long as a predefined tag is included. This way, a better description can be provided to the user without limiting functionality. Lastly, test cases 91 and 92 are wishlist tests as well. It will be included if the team can get to adding in the academic information feature to the website. If unable to reach by the deadline, these items are

a good future implementation to have to broaden the scope of the website and be even more pertinent to newer college students.

Source Code

```
from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, BooleanField,
SelectField,\
    SubmitField, DateField
from flask_wtf.file import FileField
from wtforms.validators import DataRequired, Length, Email, Regexp
from wtforms import ValidationError
from flask_pagedown.fields import PageDownField
from ..models import Role, User
from datetime import date


class NameForm(FlaskForm):
    name = StringField('What is your name?', validators=[DataRequired()])
    submit = SubmitField('Submit')


class EditProfileForm(FlaskForm):
    name = StringField('Real name', validators=[Length(0, 64)])
    location = StringField('Location', validators=[Length(0, 64)])
    about_me = TextAreaField('About me')
    submit = SubmitField('Submit')


class EditProfileAdminForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(1,
64),
                                             Email()])
    username = StringField('Username', validators=[
        DataRequired(), Length(1, 64),
        Regexp('^[A-Za-z][A-Za-z0-9_.]*$', 0,
               'Usernames must have only letters, numbers, dots or '
               'underscores')])
    confirmed = BooleanField('Confirmed')
    role = SelectField('Role', coerce=int)
    name = StringField('Real name', validators=[Length(0, 64)])
    location = StringField('Location', validators=[Length(0, 64)])
    about_me = TextAreaField('About me')
    submit = SubmitField('Submit')

    def __init__(self, user, *args, **kwargs):
        super(EditProfileAdminForm, self).__init__(*args, **kwargs)
```

```
        self.role.choices = [(role.id, role.name)
                             for role in
Role.query.order_by(Role.name).all()]
        self.user = user

    def validate_email(self, field):
        if field.data != self.user.email and \
                User.query.filter_by(email=field.data).first():
            raise ValidationError('Email already registered.')

    def validate_username(self, field):
        if field.data != self.user.username and \
                User.query.filter_by(username=field.data).first():
            raise ValidationError('Username already in use.')


class PostForm(FlaskForm):
    body = PageDownField("What's on your mind?",
validators=[DataRequired()])
    submit = SubmitField('Submit')


class CommentForm(FlaskForm):
    body = StringField('Enter your comment', validators=[DataRequired()])
    submit = SubmitField('Submit')

class RestaurantForm(FlaskForm):
    name = StringField('Restaurant Name', validators=[
        DataRequired(), Length(1, 64)])
    phone = StringField('Telephone', validators=[DataRequired(), Length(1,
12)])
    address = StringField('Address', validators=[
        DataRequired(), Length(1, 100)])
    tags = StringField('Tags', validators=[Length(0,64)])
    about_me = TextAreaField('About this restaurant')
    file=FileField('Image')
    submit = SubmitField('Submit')


class EventForm(FlaskForm):
    title = StringField('Title', validators=[
        DataRequired(), Length(1, 64)])
    start_date = DateField('Start date ',
validators=[DataRequired()],default=date.today)
    end_date = DateField('End date: ',
validators=[DataRequired()],default=date.today)
    tags = StringField('Tags', validators=[Length(0,64)])
```

```
    about_me = TextAreaField('Description')
    file=FileField('Image')
    submit = SubmitField('Submit')

class AttractionForm(FlaskForm):
    attraction_name = StringField('Attraction Name', validators=[
        DataRequired(), Length(1, 64)])
    phone = StringField('Telephone', validators=[DataRequired(), Length(1,
12)])
    address = StringField('Address', validators=[
        DataRequired(), Length(1, 100)])
    tags = StringField('Tags', validators=[Length(0,64)])
    about_me = TextAreaField('About this attraction')
    file=FileField('Image')
    submit = SubmitField('Submit')

class HikeForm(FlaskForm):
    hike_name = StringField('Resource Name', validators=[
        DataRequired(), Length(1, 64)])
    address = StringField('Location', validators=[
        DataRequired(), Length(1, 100)])
    tags = StringField('Tags', validators=[Length(0,64)])
    about_me = TextAreaField('About this academic resource')
    file=FileField('Image')
    submit = SubmitField('Submit')

class FilterRestaurant(FlaskForm):
    filterRestaurant = SelectField(u'Filter By: ', choices=[('Show All',
'Show All'),
                            ('Dining', 'Dining'), ('Dessert', 'Dessert'),
('Pizza', 'Pizza'), ('Chinese', 'Chinese'),
                            ('Healthy', 'Healthy'), ('Bars', 'Bars'),
('OutsideCampus', 'OutsideCampus'), ('FastFood', 'FastFood')])


    submit = SubmitField('Filter Food')

class FilterAttraction(FlaskForm):
        filterActivity = SelectField(u'Filter By: ', choices=[('Show All',
'Show All'),
                            ('Nature', 'Nature'), ('Hiking', 'Hiking'),
('Entertainment', 'Entertainment'), ('Extreme', 'Extreme'),
                            ('Sports', 'Sports'), ('ArtHistory',
'ArtHistory'), ('ScienceTechnology', 'ScienceTechnology'), ('Sports',
'Sports'),
                     ('Movies', 'Movies'), ('Concerts', 'Concerts'),
('Kayaking', 'Kayaking')])
```

```
    submit = SubmitField('Filter Attractions')
```

This is an example of our source code. The rest of it can be found at our github repository link - https://github.com/tlessig8771/CS1530-G14.git

User's Manual

1. Product overview

Our product is a social networking site called PittJungle that is specifically for Pitt students. It will allow Pitt students to explore the city of Pittsburgh, while building a community outside of academics. It's main functions are:

- Users will be able to explore different restaurants/places to visit in Pittsburgh
- Users will be able to comment/rate places they have visited
- Users will be able to friend other users on the site
- Users will be able to view recommendations for new places to explore based on what they have already visited
- Students will be able to filter out places to explore based on tags such as "nature", "art", "science"
    - This will allow the user to find places that fit very specific criteria
- Users will be able to post an event that others on the site will be able to see

2. Getting started

2.1. Log in

The user will be able to log into the system via the login link located on the right hand side of the screen. This is shown in section 2.3 of the user manual. The user will be taken to a login page where they can enter the necessary information. Extra online help will be available on the 'user help' tab located on the upper right of the navigation bar. This page will include the user manual and videos to show the user how the site is to be used if deemed necessary. Provisional sample runs and examples are provided in the sections below.

2.2. Help mode

A help mode is a feature that shows the user how to use the product. This can include the functionality of certain sections and how to maximize the use. We will implement the help mode in our website by adding a tab on the navigation bar. This tab will take the user to a page which has illustrations and videos along with short descriptions on how to use it. We will include specifics on how to scroll through each

2.3. Provisional Sample Run and Description on How to Use

To use the full functionality of the PittJungle website, a user must complete the following steps. First, a user would want to log in using the login page. This will be located by the "Login" section on the home screen upon arrival to the site. An example of where this would be located is shown in light green in figure 2.3.1. Users that are not logged in will have a screen such as the one depicted in figure 2.3.2 and are free to use the site regularly but with limited features given to them (this will be explained further later on). See the numbered steps below to gain a better understanding of the use of this site.

1. Arrival to the webpage

   As previously mentioned, the main home screen to PittJungle will show up upon access. This page will have a short description of what the site is trying to accomplish. The page will also have all the tabs of what the site is to contain shown and able to be accessed by 'guests.' In figure 2.3.1, users or 'guests' at this point should see a green backgrounded login link. Click this to login or create an account to obtain full access to everything PittJungle will provide.

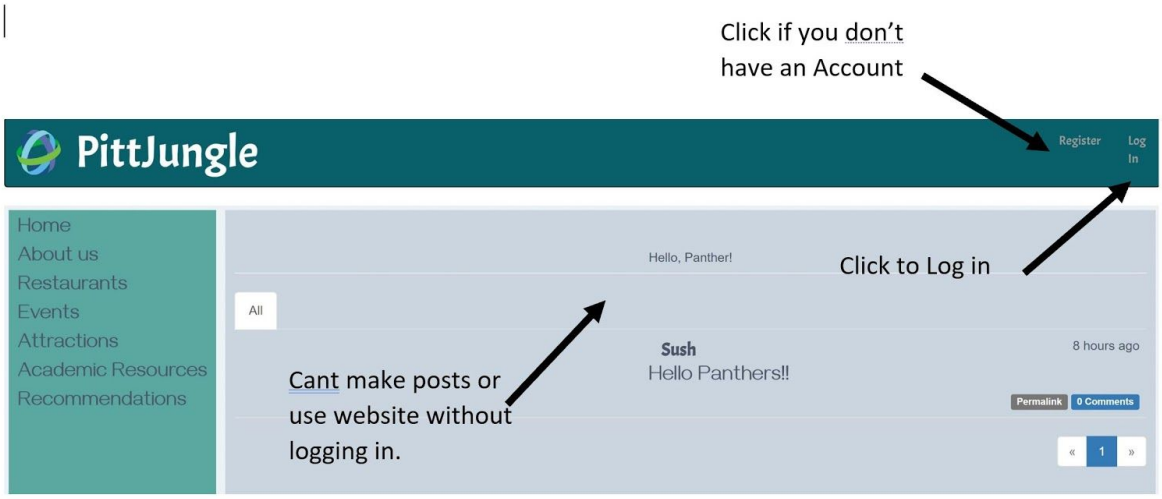


Figure 2.3.1: Provisional home screen with login link

2. Login Page

   After clicking the login link, users will be provided with the following page shown in figure 2.3.2. From here, you should enter a username and password. If you have already signed up or created an account before, just click 'login'. Otherwise, click 'create account.' This will add you to the user database. No two usernames will be allowed. If a username is taken, you will be prompted with an alert stating this, and will have to enter a new one.

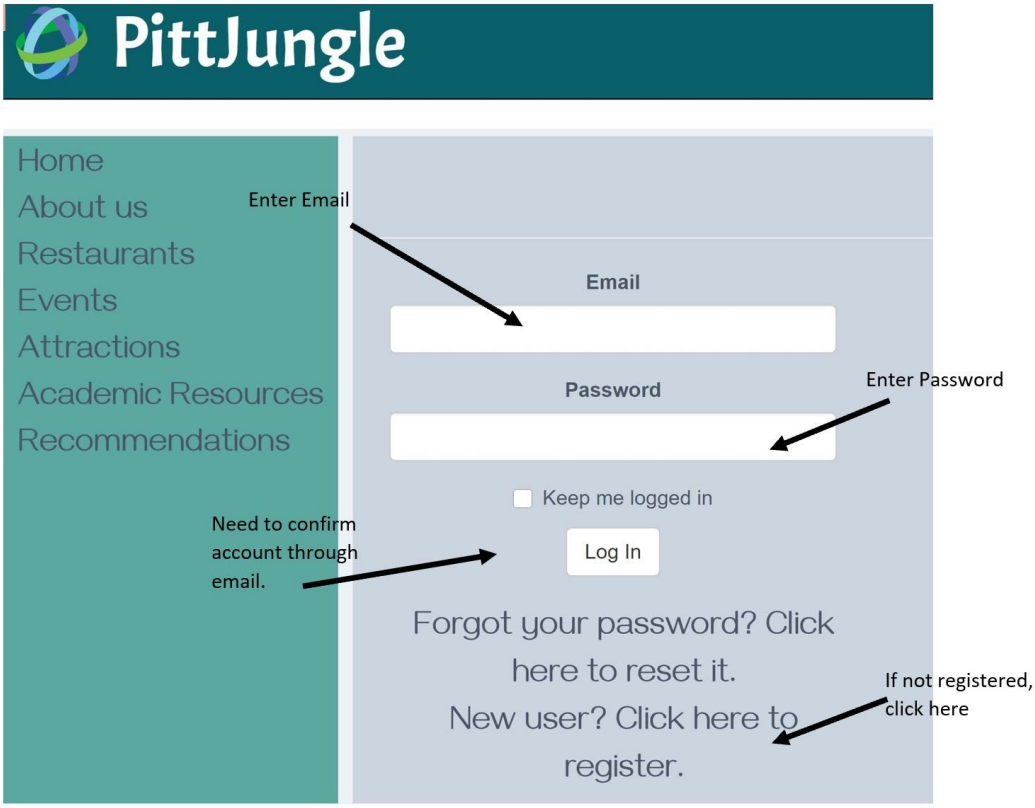


Figure 2.3.2: Provisional Login Screen

3. Non-User Screen vs. Logged in User Screen

   Figure 2.3.3 shows a non-user provisional screen. It is almost identical in layout as a logged in user but a few features will have been removed. Figure 2.3.4 shows a logged in user screen. The key differences between the two figures are as follows:

   1. The filter by option is unavailable for non-users
   2. The Campus Events and *Academic Resources tab will be restricted to logged in users

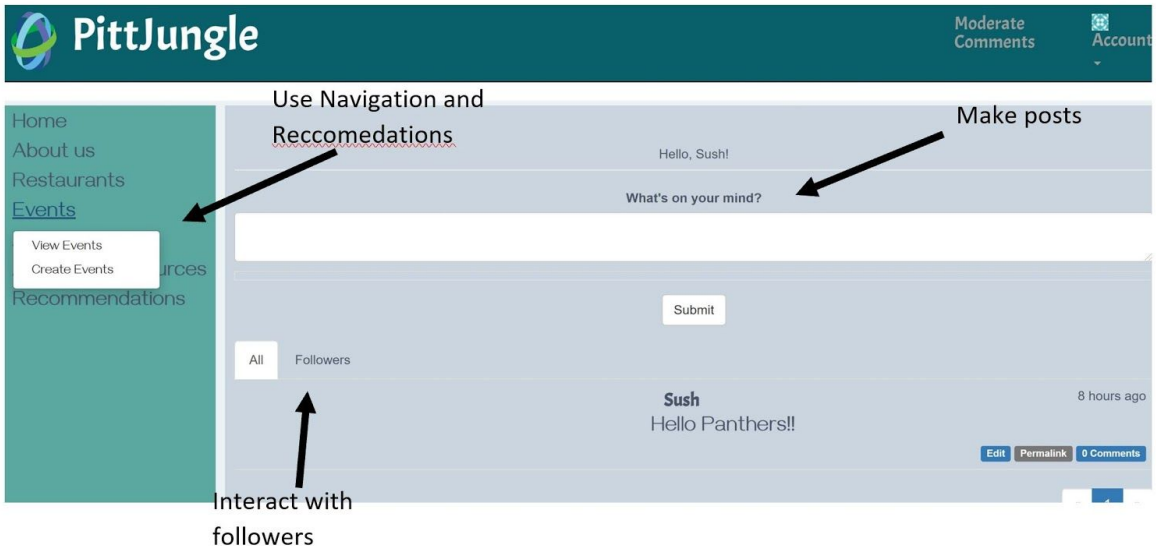


Figure 2.3.4: Provisional Logged in User Screen Example Scenario

The logged in user screen shown above steps through what has been done or what can be done to alter the page and even database. What is meant by altering the page and database? First, a logged in user can filter by anything that is provided in the drop down menu 'Filter By.' This will move whatever your selection is to the top so you can view those first. Secondly, a logged in user will be kept track of in terms of likes of certain pages/tiles or events. This will help in updating the users "interests" to better gauge what to recommend to you upon logging in. For example, if you view various dessert and pizza places pages, you should not log in the next time being shown vegetables and waffles. You will also be able to like or dislike places and see the number of likes/dislikes given by other users.

4. Clicking on a Page of Interest

You can view the tiles shown to you by clicking on them. This will prompt you with another page such as the one shown in figure 2.3.5. From here you will be provided with an image or images of the chosen place, a description of what it is, what they have to offer, how far away it is from Pitt's address, and the address. This page is nothing more than to provide you with more information to give better insight if this is what you are looking for.

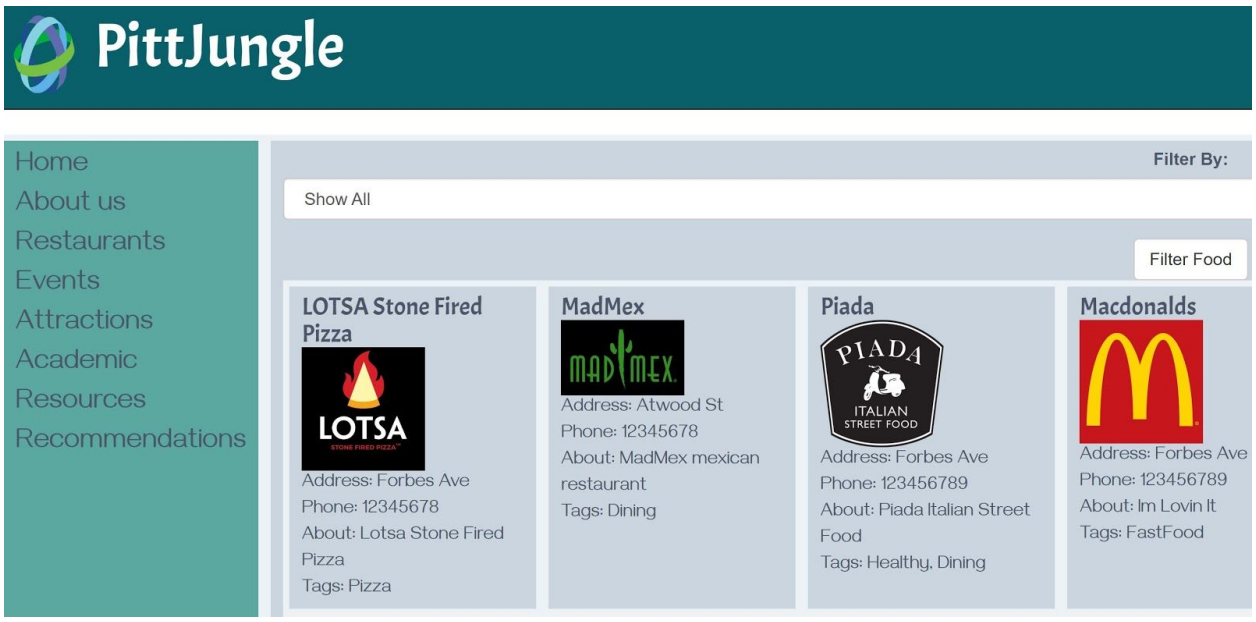


Figure 2.3.5: Provisional Place of Interest Page Description

5. Campus Events

Figure 2.3.6 will show the provisional design of the campus events page. Here you will be able to see events posted around campus from other students/users, be able to post an event of your own and comment on these posts to get more details if interested. For now, once an event is posted it cannot be changed. Therefore, the comment section may be a good way to update your post if you choose to create one. Here is where you would be able to add friends as well. This would allow you to create your own network of people from campus to go and explore with. This enables you to see their interests and therefore gauge similarities between the two of you.

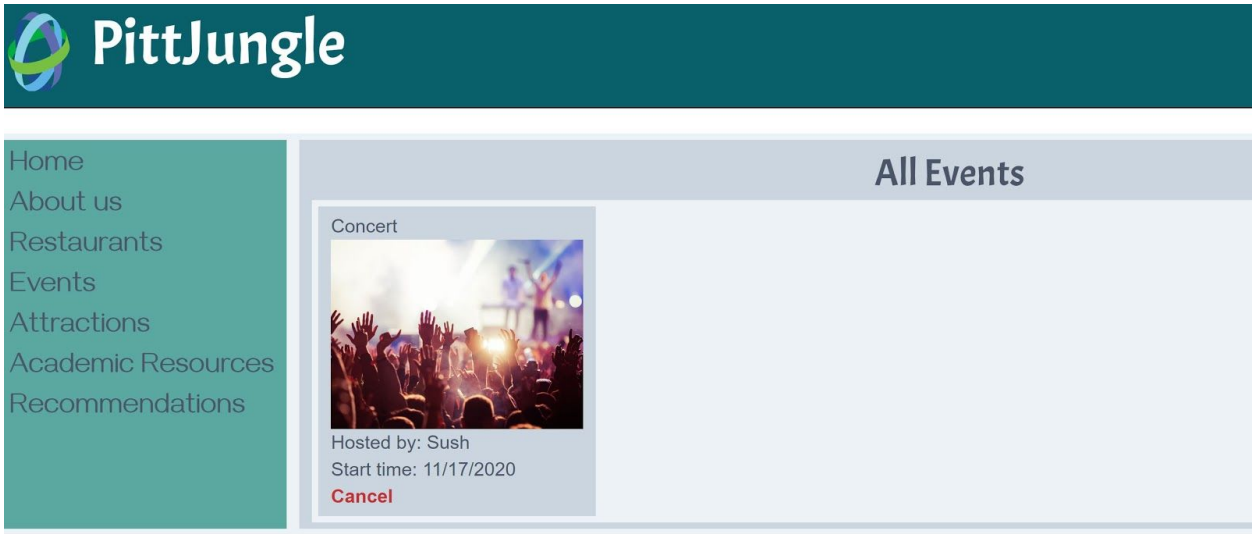


Figure 2.3.6: Provisional Campus Events Page

6. User Help

Here you will be able to find this manual on how to use the site, what the features are and how to use them. It will be updated with new figures and examples as new functionality becomes integrated. This is the go to page if you are ever lost on what something is or how to use it.

Software Maintenance

The maintenance of source codes and other relevant files are managed using the team's Github repository. The link can be found in the source code segment above. Throughout the development process, this repository was utilised for the communication of ideas, templates, and starter code. All potential updates would eventually get pushed to this repository to be included in the final product. Future modifications can only be made by authorized team members. However, anyone can fork this project and maintain their own version of Pittjungle on their personal system.

Revision history

MS1:

8/30/20 - Sherryl - Software Plan Functions, Scope and Tasks are Started
9/01/20 -
Sushruti - Scope includes more detail; Functions given short descriptions
Trent - Performance Criteria Set
9/03/20 -
Sushruti - Group Members and Roles
Luiza - Hardware and Software Components Listed
David - Grammatical Corrections and extra details applied to roles
9/06/20 - ALL - Proofreading and submission for Tuesday September 8th 2020

MS2:

9/15/20 - Sherryl - Data Flow Diagrams Created
9/18/20 -
Sushruti - Cost/Schedule Tables Created and summary of project overview given
Trent - Prelim User Manual Created and Design Constraints Posted
9/20/20 -
Sushruti - Help Mode Described
Sherryl - DFD revised and updated and product overview given
Luiza - User Interface Description and Processing Narrative Explained
Trent - Design Constraints Updated
David - Functions described withIC Cards as well as Modes of Operation
9/21/20 -
ALL - Grammatical Corrections and ready for submission on **9/22/20**
Trent - Revised LOC Table

MS3:

10/03/20 -

Sushruti - Performance Reqs set and Exception Handling specified

10/04/20 -

David - List of future modifications and Acceptance criteria
Luiza - Data Elements/Objects specified in detail
Sherryl - Detailed DFD's for each function provided
Trent - Sources of information provided and revision history updated

10/05/20 -

ALL - Grammatical corrections made and ready for submission on **10/06/20**

MS4:

10/17/20 - David/Trent - OOD section specifying classes/modules in more depth

10/18/20 -

Sushruti - OOA System overview, DFD diagrams and class models
Luiza - OOA Dynamic Model and Scenarios
Sherryl - Functional Models

10/19/20 -

ALL - Review Document / Finalize Changes / Ready For Submission on **10/20/20**

MS5:

MS6:

Web Pages:

9/20/20 - Sherryl and Sushruti - Base Home page layout created

Backend and Databases:

9/21/20 - Luiza - User model database created

Recommender System:

10/04/20 - Trent - Research on Python databases and recommender systems
10/17/20 - Trent - Python Version of rec system using dictionaries and writing to and reading from files

Functions:

9/22/20 - David and Luiza - User login created (non-integrated to website)
10/01/20 - David - Filter functionality being researched and worked on

Installation Procedure

Clone the repository from the Github:
https://github.com/tlessig8771/CS1530-G14/tree/master/thepittjungle

TA is included as a collaborator.

To run in Windows

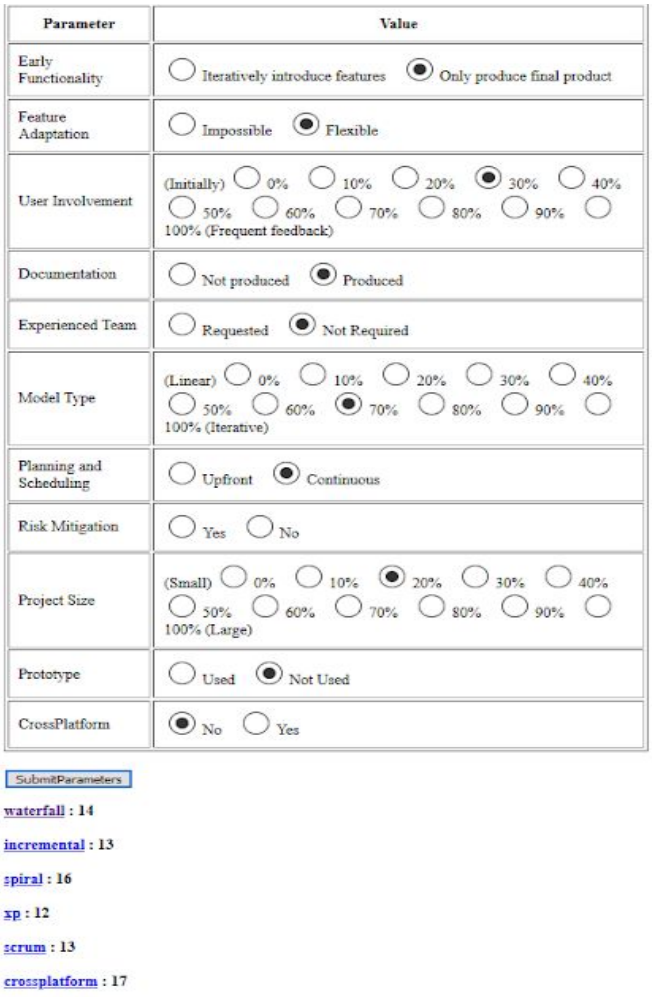
1. cd thepittjungle
2. py -3 -m venv venv
3. venv\Scripts\activate
4. pip install -r requirements.txt // Do this only once. postgres must be installed
5. set FLASK_APP=pittjungle.py
6. set MAIL_USERNAME=thepittjungle
7. set MAIL_PASSWORD=CS1630Project In order to set admin account: set FLASKY_ADMIN=email@example.com ==> email of the user to be set as admin it needs to set before the admin registers as user.
8. flask deploy
9. flask run

To run in MacOS:

1. cd thepittjungle
2. python3 -m venv venv
3. . venv/bin/activate
4. pip install -r requirements.txt // Do this only once. postgres must be installed
5. export FLASK_APP=pittjungle.py
6. export MAIL_USERNAME=thepittjungle
7. export MAIL_PASSWORD=CS1630Project export FLASKY_ADMIN=email@example.com ==> email of the user to be set as admin it needs to set before the admin registers as user.
8. flask deploy
9. flask run

3. Modes of operation

The user selects their category of choice by clicking on its header. Then users would use the arrow heads provided to browse through the various posts. Users then click on the post they are interested in and are directed to the post's page where its information is provided. This system does not support any special commands or dialogue.



Parameter	Value
Early Functionality	○ Iteratively introduce features ◉ Only produce final product
Feature Adaptation	○ Impossible ◉ Flexible
User Involvement	(Initially) ○ 0% ○ 10% ○ 20% ◉ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Frequent feedback)
Documentation	○ Not produced ◉ Produced
Experienced Team	○ Requested ◉ Not Required
Model Type	(Linear) ○ 0% ○ 10% ○ 20% ○ 30% ○ 40% ○ 50% ○ 60% ◉ 70% ○ 80% ○ 90% ○ 100% (Iterative)
Planning and Scheduling	○ Upfront ◉ Continuous
Risk Mitigation	○ Yes ○ No
Project Size	(Small) ○ 0% ○ 10% ◉ 20% ○ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Large)
Prototype	○ Used ◉ Not Used
CrossPlatform	◉ No ○ Yes

SubmitParameters

waterfall : 14

incremental : 13

spiral : 16

xp : 12

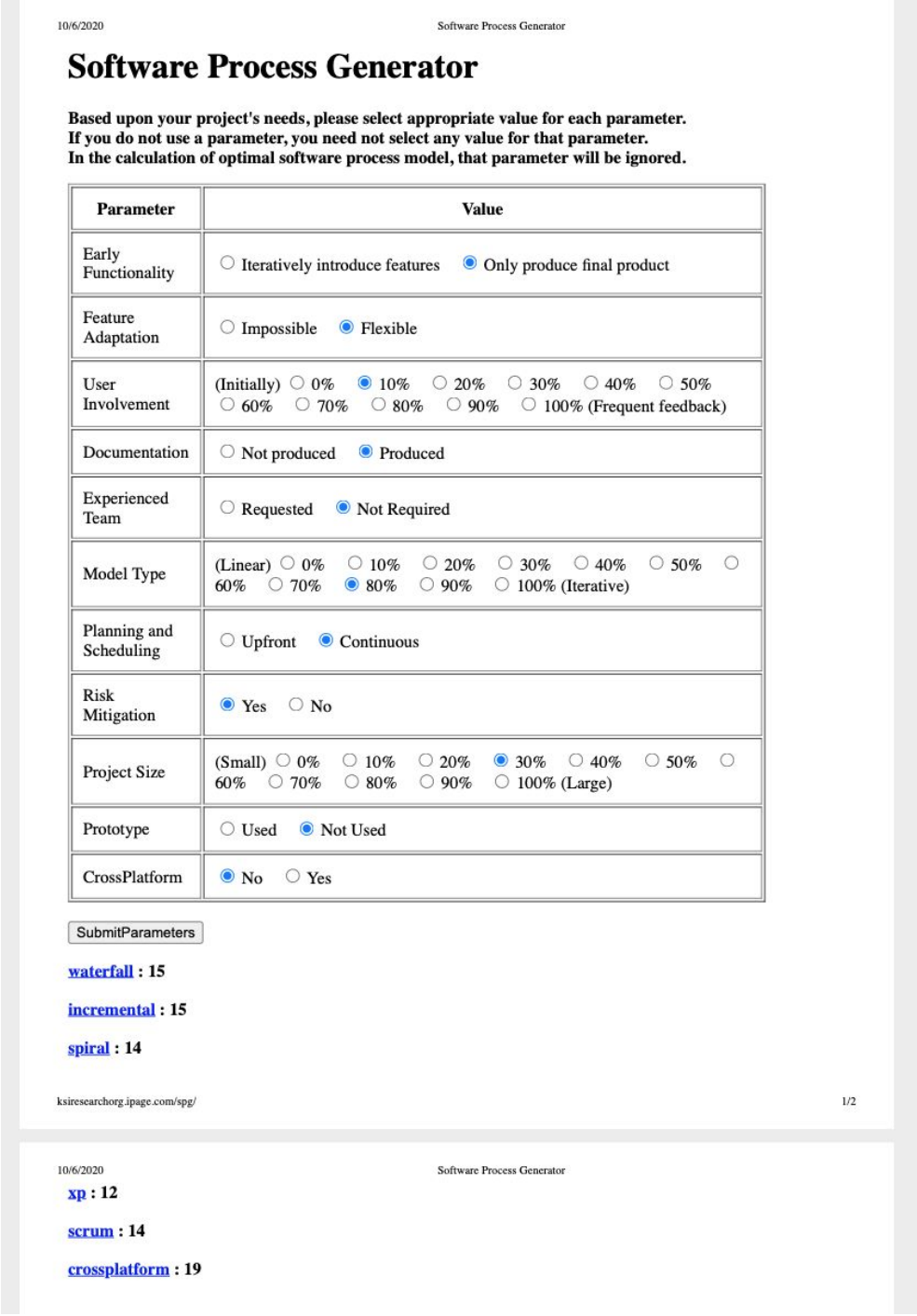
scrum : 13

crossplatform : 17

SPG Output

The software model suggested the XP and spiral models from our answers to the parameters. I was expecting the Waterfall method since that is the one that was emphasized in class. We also assumed the point of the milestones was to create a waterfall model for our project.

In this SPG analysis, the xp model was the one that it suggested. No, I would not have expected an extreme coding software model to follow. Of the ones listed, I would have expected the spiral, waterfall or incremental models because I feel those were discussed more and I would have a better understanding of how to create one compared to an xp model.



10/6/2020 Software Process Generator

Software Process Generator

Based upon your project's needs, please select appropriate value for each parameter.
If you do not use a parameter, you need not select any value for that parameter.
In the calculation of optimal software process model, that parameter will be ignored.

Parameter	Value
Early Functionality	○ Iteratively introduce features ◉ Only produce final product
Feature Adaptation	○ Impossible ◉ Flexible
User Involvement	(Initially) ○ 0% ◉ 10% ○ 20% ○ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Frequent feedback)
Documentation	○ Not produced ◉ Produced
Experienced Team	○ Requested ◉ Not Required
Model Type	(Linear) ○ 0% ○ 10% ○ 20% ○ 30% ○ 40% ○ 50% ○ 60% ○ 70% ◉ 80% ○ 90% ○ 100% (Iterative)
Planning and Scheduling	○ Upfront ◉ Continuous
Risk Mitigation	◉ Yes ○ No
Project Size	(Small) ○ 0% ○ 10% ○ 20% ◉ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Large)
Prototype	○ Used ◉ Not Used
CrossPlatform	◉ No ○ Yes

SubmitParameters

waterfall : 15

incremental : 15

spiral : 14

ksiresearchorg.ipage.com/spg/ 1/2

10/6/2020 Software Process Generator

xp : 12

scrum : 14

crossplatform : 19

The Software Process Generator suggested the xp model. I was not expecting this model, as it is an extreme programming model. I was expecting the waterfall model where there is a linear, incremental approach to software development and where there is also testing in every phase.

Parameter	Value
Early Functionality	○ Iteratively introduce features ◉ Only produce final product
Feature Adaptation	○ Impossible ◉ Flexible
User Involvement	(Initially) ○ 0% ◉ 10% ○ 20% ○ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Frequent feedback)
Documentation	◉ Not produced ○ Produced
Experienced Team	◉ Requested ○ Not Required
Model Type	(Linear) ○ 0% ○ 10% ○ 20% ○ 30% ○ 40% ◉ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Iterative)
Planning and Scheduling	◉ Upfront ○ Continuous
Risk Mitigation	◉ Yes ○ No
Project Size	(Small) ○ 0% ○ 10% ○ 20% ○ 30% ○ 40% ○ 50% ◉ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Large)
Prototype	◉ Used ○ Not Used
CrossPlatform	○ No ◉ Yes

SubmitParameters

waterfall : 18

incremental : 14

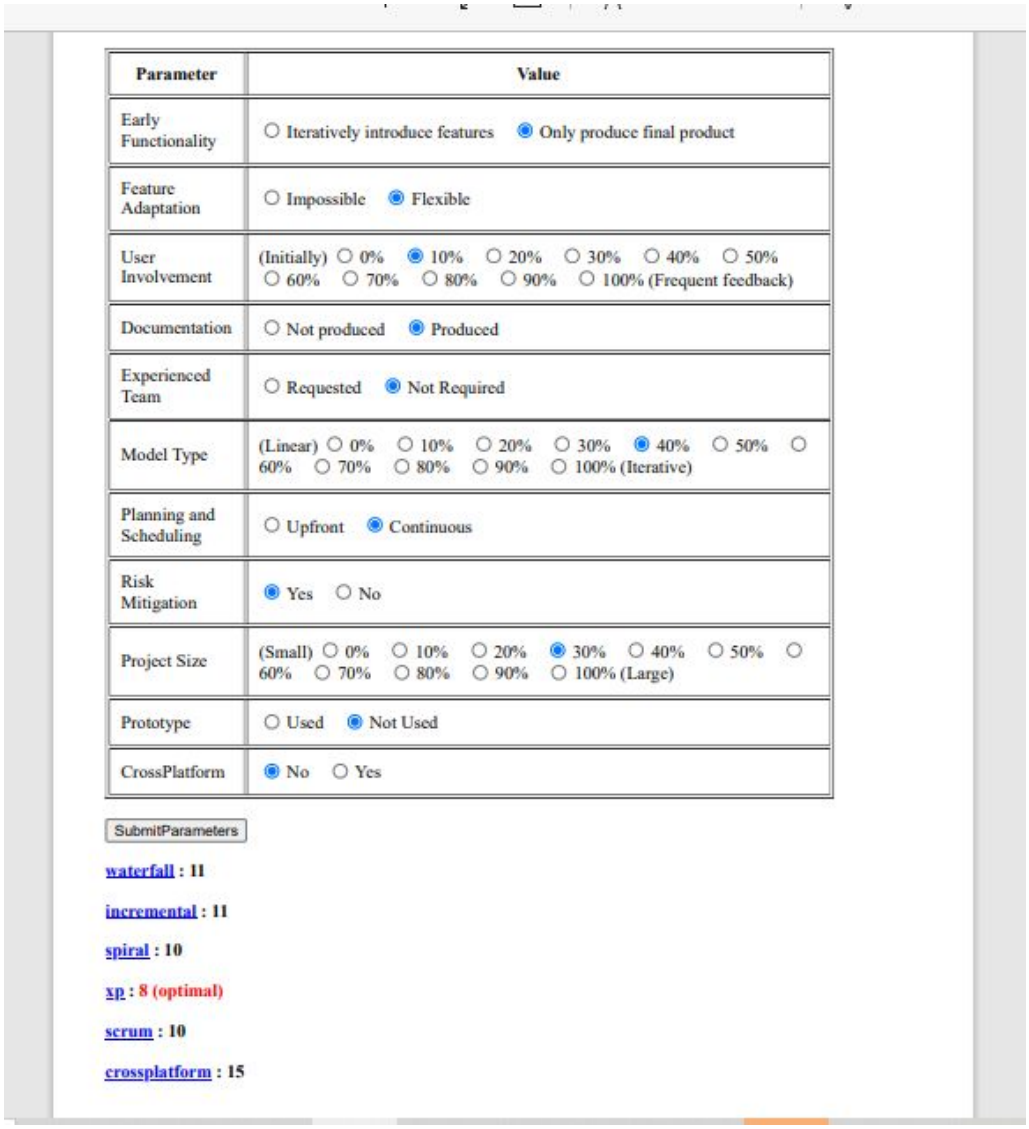
spiral : 9 (optimal)

xp : 15

scrum : 15

crossplatform : 12

SPG tool recommended spiral model. I did expect this model because it suggests to do Risk Analysis, then follow Requirements. Because we have to mitigate the risks, we may have to do repeated use of prototypes. Even though it may cause schedule delays, in the long run, spiral models will assure quality software.



Parameter	Value
Early Functionality	○ Iteratively introduce features ◉ Only produce final product
Feature Adaptation	○ Impossible ◉ Flexible
User Involvement	(Initially) ○ 0% ◉ 10% ○ 20% ○ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Frequent feedback)
Documentation	○ Not produced ◉ Produced
Experienced Team	○ Requested ◉ Not Required
Model Type	(Linear) ○ 0% ○ 10% ○ 20% ○ 30% ◉ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Iterative)
Planning and Scheduling	○ Upfront ◉ Continuous
Risk Mitigation	◉ Yes ○ No
Project Size	(Small) ○ 0% ○ 10% ○ 20% ◉ 30% ○ 40% ○ 50% ○ 60% ○ 70% ○ 80% ○ 90% ○ 100% (Large)
Prototype	○ Used ◉ Not Used
CrossPlatform	◉ No ○ Yes

SubmitParameters

waterfall : 11

incremental : 11

spiral : 10

xp : 8 (optimal)

scrum : 10

crossplatform : 15

The SPG tool recommended the xp model. However, I was expecting the waterfall model because of its linear nature. But since the waterfall model's planning is done before moving to the next stage of the development process begins and I'm using continuous planning, I understand how the waterfall isn't the best design for my project.

First Iteration/ First Sprint

1. Introduction

The features we had included in our initial wishlist were as follows:

- ➢ **Academic Resources**
  - Users will be able to recommend courses to their friends on the network
  - Users will be able to host study sessions and get academic help
- ➢ **Chatbot**
  - Users will be able to interact with a chatbot to get recommendations on food/academic material
- ➢ **App Development**
- ➢ **Writing Reviews for Restaurants and Events**

The team had decided to go with the first item in our wishlist:

- ➢ **Academic Resources**
  - Users will be able to create events under the academic resources tab for study sessions or group study lessons.
  - This could also be a great place for TAs to show their office hours (and provide a zoom link) as well as study sessions.
  - This wouldn't be limited to students in the class. Anyone could join for questions and potentially broaden their knowledge range if they are from another major.

2. How was the function selected

As the team had already implemented a basis for creating events, such as the Event page where students can create activities/events on and around campus that they are holding, it would be relatively straightforward to implement the Academic Resource wishlist feature for creating study sessions. As you will read shortly, the team made a list of pros and cons for each wishlist feature in order to compare and contrast them in picking the appropriate function to implement. To briefly summarize the chosen implementation and its functionality, Teaching Assistants could post their office hour sessions with their zoom links, students could host study sessions (quite similar to the event creation, by providing a time, place, date and description), or order to obtain general help regarding academics at the University of Pittsburgh.

2.1. Participating Stakeholders (or their surrogates)

	Project Manager	GUI Designer	Databases/ Backend	Testing	Research	Recommender
Sherryl		x		x	x	
Sush	x	x		x		
Luiza			x	x		
David			x	x		x
Trent				x	x	x

2.2. Summary of the discussion among stakeholders

➢ **<u>Pros</u> of implementing Academic Section**
- Similar layout to the events sections
- Quickly integrated and straightforward

➢ **<u>Cons</u> of implementing Academic section**
- Not very ambitious
- Could be confused with the 'events' section

➢ **<u>Pros</u> of the Chatbot**
- Attention grabber
- Resourceful
- Increased User Interaction

➢ **<u>Cons</u> of the Chatbot**
- Time Consuming and Intensive
- Beyond the teams experience level

➢ **<u>Pros</u> of App Development**
- Becomes Mobile Friendly
- Easier access
- Broadens reach of our software

➢ **<u>Cons</u> of App Development**
- Also time intensive
- Limited to Android deployment
- Must learn Android Studio

➢ **<u>Pros</u> of Writing Reviews**
- Users give feedback to places on/around campus
- Just like leaving comments

➢ **<u>Cons</u> of Writing Reviews**
- Would it really impact users' feelings towards the site?
- Not ambitious nor challenging

2.3. Decision and Rationale

We decided to implement the academic resources tab in our app. We chose this because we had implemented something similar for the Events tab and would just need to modify the Events implementation to the appropriate design for the academic resources. This feature implementation was the most feasible for the team. Given our time constraints, we immediately ruled out Chatbot and App Development. These two features would require an extensive amount of work and the team felt it would be better to have this be a future feature addition to ensure it is done correctly. The team also ruled out leaving reviews because we felt it would not impact a user's feelings toward the site. Therefore the academic resources feature would be implemented because the team felt users would appreciate having an all-in-one access website that could include office hours to study sessions regarding their topic of interest.

3. Regression Tests Results

Test #	Description	Input	Expected Output	Date Tested	Problem	Solution
1	New User Registration	Filled out Form	Success: User receives email to verify. User can login	11/11/2020 ✔	None	Email Received, Link is clickable and account becomes verified
2	User Event Creation	Filled out Form	Success: Form submitted and event shows on event page	11/11/2020 ✔	None	Can be viewed on screen
3	Recommendation System	Tags obtained from clicking	Success: An updated probability distribution and suggestions shown on page	11/11/2020 ✔	None	Recommendations shown on Rec page to user

4	Commenting	Input field	Success: Comment shows on main screen	11/11/2020 ✔	None	Comment seen on main screen by any user
5	Admin Comment Management	Button Input	Success: Admin can enable/disable comments if deemed necessary	11/10/2020 ✔	None	Comment shows up as disabled if disabled
6	Add Friend	Follow/Unfollow button option	Success	11/09/2020 ✔	None	Friend list shows on main profile and shows how many followers and how many you follow
7	Logging In	Login Form	Success	11/09/2020 ✔	None	User gets logged in
8	Filtering	Drop-Down menu and submit button	Success	11/14/2020 ✔	None	The selection submitted only shows corresponding items associated with the filter
9	Update Profile	Update profile form and submission	Success: Updated account shows on users main profile page	11/12/2020 ✔	None	Users can see profile of others user and what they included with it
10	Recommendation Probability Distribution Written to file	Rec. Objects	Success	11/15/2020 ✔	None	Written to file “UserProbs.txt” *is updated each time the rec page is visited
11	View Other Events	Events Page	Success	11/14/2020 ✔	None	Users can see events posted by other users

4. Conclusion

If we had more time, we would implement Writing Reviews, Mobile App and Chatbot. These features would enrich the user experience. Implementing the above-mentioned features would hinder us from finishing the core features necessary to have a proper functioning website. After discussing with the team, we decided to implement Academic Resources because we already had experience of how to implement similar features. We did not have to alter our database drastically. Of course, we would gain a great deal of experience if we had a chance to develop a Mobile App for our project. In today's world, an app companion for a website is a must. A mobile app for our website would definitely enhance the user experience. Another future development would be to update and enhance the user interface. Every website and application go through various design layouts before a team decides on the best one. It would be best to have a focus group test out a few layouts/interfaces and obtain feedback from them to decide on what to go with.

Appendix: Project Presentation

PittJungle Key Highlights

Created By: Sushruti Bansod, Sherryl Augustine, Luiza Urazaeva, David Ladeji, Trent Lessig

Key Audience

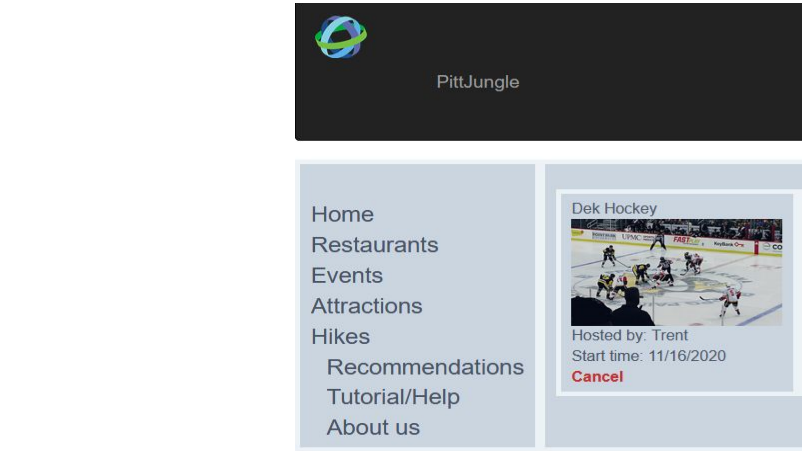
University of Pittsburgh Students

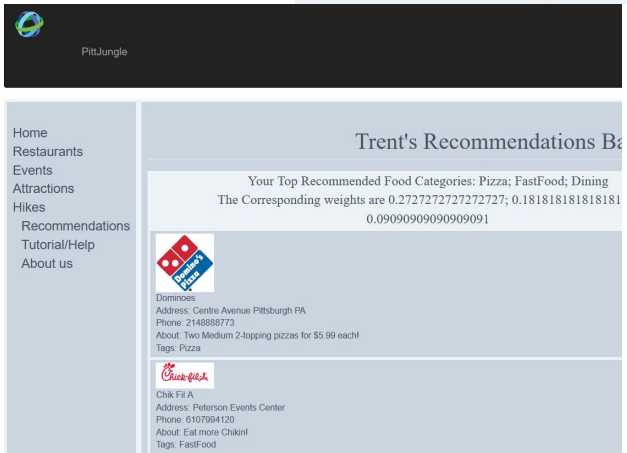
- Helping to expand students horizon for things to do, places to eat, people to study with, on and around the campus

Prominent Features

Event Creation -> Students can meet new people easily by posting about fun events they are hosting

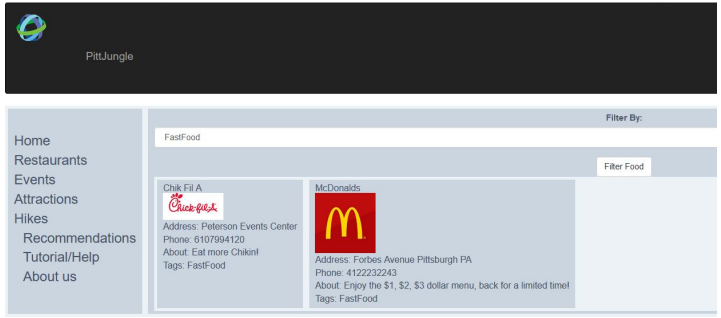
Recommendation Page -> Each user will have a unique recommendation page that will show their top 3 categories from the Restaurants page and Attractions page. From here the page will show results associated with the categories it has recommended (in order of occurrence or the probability they would like this over the other)





Prominent Features

Filtering -> Users can filter the Restaurants and Attractions pages to see specific items by category



Account Verification -> Users must verify their account by clicking on a link provided via email upon successful account creation (added security benefit)

Dear Trent,

Welcome to **PittJungle**!

To confirm your account please click here.

Alternatively, you can paste the following link in your browser's address bar:

http://127.0.0.1:5000/auth/confirm/eyJhbGciOiJIUzI1NiIsImlhdCI6MTYwNTU2NjMzMCwiZXhwIjoxNjA1NTY5OTMwfQ.eyJjb25maXJtIjoxfQ.ZnhDzmWf11WiRt3ZEN4bECc91N-d5L3fQLGvd2jps6Y

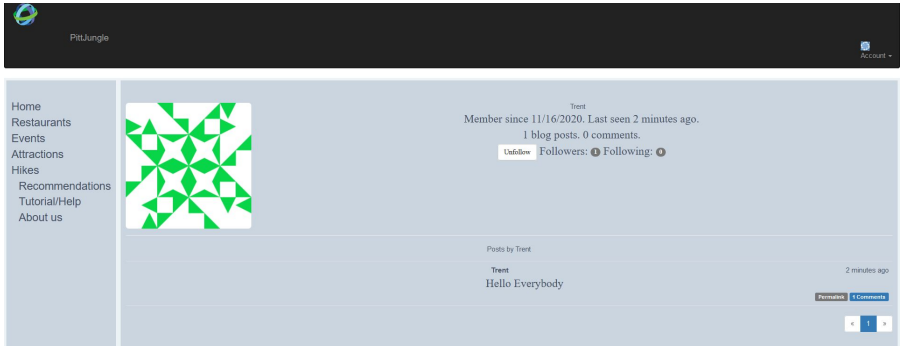
Sincerely,

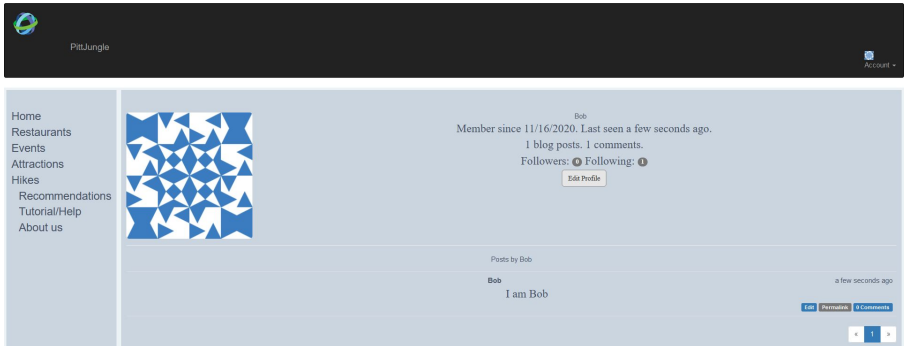
The PittJungle Team

Note: replies to this email address are not monitored.

Prominent Features

Friend/Commenting -> Users can follow one another via the comment section of the home page. Users will be able to see all posts made by users here and are able to comment back and forth with one another. Admin users can disable inappropriate comments





The Team Members Contributions

Sushruti - Project Manager, UI, Front End Design/Layout, Research, Testing

Sherryl - Research, Front End Design/Layout, UI, Testing

Luiza - Backend (python, html, flask), Database Models, Integration, Chat/Friend, Adding Items, Email/Login Verification, Forms, Page Templates

David - Filter Functionality, Backend Help, Recommendation help, Testing and Integration

Trent - Recommendation System, Filter Functionality help, Research, Testing and Integration