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

- Software Plan .................................................................2
- Requirements ...............................................................5
- Object Oriented Analysis ..............................................24
- Object Oriented Design .................................................30
- Test plan .................................................................35
- Test results ..............................................................40
- Sample source codes ..................................................49
- User manual ..............................................................52
- Software Maintenance procedure ................................60
- Installation procedure ................................................59
- Report on first iteration/first sprint ............................65

Appendix: Project Presentation
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 Team Assignments

<table>
<thead>
<tr>
<th>Project Manager</th>
<th>GUI Designer</th>
<th>Databases/Backend</th>
<th>Testing</th>
<th>Research</th>
<th>Chatbot/ML/Recommender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sherryl</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Sush</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luiza</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Trent</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

### 4.0 Cost

<table>
<thead>
<tr>
<th>Task</th>
<th>Requirements</th>
<th>Plan/Design</th>
<th>Code</th>
<th>Test</th>
<th>Validate/Maintain</th>
<th>Total hours</th>
<th>Rate</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sush</td>
<td>40 hours</td>
<td>80 hours</td>
<td>160 hours</td>
<td>80 hours</td>
<td>80 hours</td>
<td>440 hours</td>
<td>$100</td>
<td>$44,000</td>
</tr>
<tr>
<td>Sherryl</td>
<td>40 hours</td>
<td>80 hours</td>
<td>160 hours</td>
<td>80 hours</td>
<td>80 hours</td>
<td>440 hours</td>
<td>$100</td>
<td>$44,000</td>
</tr>
<tr>
<td>Trent</td>
<td>40 hours</td>
<td>80 hours</td>
<td>160 hours</td>
<td>80 hours</td>
<td>80 hours</td>
<td>440 hours</td>
<td>$100</td>
<td>$44,000</td>
</tr>
<tr>
<td>Luiza</td>
<td>40 hours</td>
<td>80 hours</td>
<td>160 hours</td>
<td>80 hours</td>
<td>80 hours</td>
<td>440 hours</td>
<td>$100</td>
<td>$44,000</td>
</tr>
<tr>
<td>David</td>
<td>40 hours</td>
<td>80 hours</td>
<td>160 hours</td>
<td>80 hours</td>
<td>80 hours</td>
<td>440 hours</td>
<td>$100</td>
<td>$44,000</td>
</tr>
<tr>
<td><strong>Other Resources</strong></td>
<td>Lines of Code</td>
<td>10,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$1</td>
<td>$10,000</td>
</tr>
</tbody>
</table>

**Total Cost**: $230,000

### Function Optimization

<table>
<thead>
<tr>
<th>Function</th>
<th>Optimistic</th>
<th>Most</th>
<th>Pessimistic</th>
<th>Expected</th>
<th>Deviation</th>
<th>$/Line</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>2500</td>
<td>4000</td>
<td>5400</td>
<td>3983.33</td>
<td>250</td>
<td>$25</td>
<td>$2,375</td>
</tr>
<tr>
<td>User/Event Databases</td>
<td>1500</td>
<td>2000</td>
<td>2250</td>
<td>1958.33</td>
<td>83.3333</td>
<td>$30</td>
<td>$2,850</td>
</tr>
<tr>
<td>Recommendation System</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
<td>3000</td>
<td>166.6667</td>
<td>$30</td>
<td>$2,850</td>
</tr>
<tr>
<td>Event Posting</td>
<td>750</td>
<td>1200</td>
<td>1400</td>
<td>1158.33</td>
<td>75</td>
<td>$12</td>
<td>$1,140</td>
</tr>
<tr>
<td>Review/Comment</td>
<td>850</td>
<td>1250</td>
<td>1500</td>
<td>1225</td>
<td>66.6667</td>
<td>$12</td>
<td>$1,140</td>
</tr>
<tr>
<td><strong>Total Expected LOC and Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td>11325</td>
<td></td>
<td></td>
<td>$10,355</td>
</tr>
</tbody>
</table>

### 5.0 Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Task</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (9/1-9/5)</td>
<td>Requirements</td>
<td>9/8 Milestone 1</td>
</tr>
<tr>
<td>2 (9/6-9/12)</td>
<td>Plan</td>
<td>9/22 Milestone 2</td>
</tr>
<tr>
<td>3 (9/13 – 9/19)</td>
<td>Design</td>
<td>10/4 Milestone 3</td>
</tr>
<tr>
<td>4 (9/20 – 9/26)</td>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>5 (9/27 – 10/3)</td>
<td>Code/Test</td>
<td></td>
</tr>
<tr>
<td>6 (10/4 – 10/10)</td>
<td>Code/Test</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Week</th>
<th>Task</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 (10/11 – 10/17)</td>
<td>Code/Test</td>
<td>10/20 Milestone 4</td>
</tr>
<tr>
<td>8 (10/18 – 10/24)</td>
<td>Code/Test</td>
<td>10/27 Milestone 4</td>
</tr>
<tr>
<td>9 (10/25 – 10/31)</td>
<td>Code/Test</td>
<td>11/10 Milestone 4</td>
</tr>
<tr>
<td>10 (11/1 – 11/7)</td>
<td>Validate</td>
<td>11/17 Submission</td>
</tr>
<tr>
<td>11 (11/8 – 11/14)</td>
<td>Maintain</td>
<td></td>
</tr>
</tbody>
</table>

### 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: String (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: Register account**

- **Description:** Process details to add new account
- **Interaction Pattern:** Mixed
- **My Task:** Add new user account, transmits 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 1 of 2** (if necessary please use several IC cards to describe an IC)

**IC Card: 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 1 of 1** (if necessary please use several IC cards to describe an IC)

**IC Card: 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 1 of 1** (if necessary please use several IC cards to describe an IC)

**IC Card: Log in function**

- **Description:** Authenticates user account details
- **Interaction Pattern:** Mixed
- **My Task:** Process user account details, transmits 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 1 of 2** (if necessary please use several IC cards to describe an IC)
IC Card

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

Quiet State
My Task: When activity is received, 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 1 of 2 (If necessary please use several IC cards to describe an IC)

IC Card

Description: Process details to add new account
Interaction Pattern:

By Myself no Interaction
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 2 of 2 (If necessary please use several IC cards to describe an IC)

IC Card

Description: Prefetches documents
Interaction Pattern:

By Myself no Interaction
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 3 of 4 (If necessary please use several IC cards to describe an IC)

IC Card

Description: Creator of event
Interaction Pattern:

By Myself no Interaction
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 1 of 1 (If necessary please use several IC cards to describe an IC)
IC Card

IC Name: Returning user

Description: Sends details to log in

Interaction Pattern:

- By Myself no Interaction

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 1 of 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 document
Time Critical Condition: Under 5 minutes
Name of Other IC: Remote server
Message to Other IC: Request document
Other IC's Task: Send document
Card 3 of 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: Under 5 minutes
Name of Other IC: Remote server
Message to Other IC: Request document
Other IC's Task: Send document
Card 3 of 4 (if necessary please use several IC cards to describe an IC)

IC Card

IC Name: Remote server

Description: Sends 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 1 of 1 (if necessary please use several IC cards to describe an IC)
3.2. Processing narrative

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

- **User Authentication.** This functionality allows users to authenticate so that 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

Creating an Account
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
### Acceptance Testing

#### 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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>Click the “Sign Up Now” link to load the register new account screen</td>
</tr>
<tr>
<td>✔️</td>
<td>On the create new account screen, enter a valid email address and password</td>
</tr>
<tr>
<td>✔️</td>
<td>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</td>
</tr>
<tr>
<td>✔️</td>
<td>Enter valid email and password for a registered user and click “Log In” which directs you to the homepage of the application</td>
</tr>
<tr>
<td>✔️</td>
<td>Click on the “Attractions” link in the navigation bar which directs you to the Attractions page</td>
</tr>
<tr>
<td>✔️</td>
<td>Click on the “Events” link in the navigation bar which directs you to the Events page</td>
</tr>
<tr>
<td>✔️</td>
<td>Select any event object to be directed to its page which displays more information on the event</td>
</tr>
<tr>
<td>✔️</td>
<td>Select the event’s author’s username to be directed to their account page</td>
</tr>
<tr>
<td>✔️</td>
<td>Click on the “Follow” button to start following this user</td>
</tr>
<tr>
<td>✔️</td>
<td>Click on the “Restaurants” link in the navigation bar which directs you to the Restaurants page</td>
</tr>
<tr>
<td>✔️</td>
<td>Click on “My Account” which directs you to your account’s profile page</td>
</tr>
<tr>
<td>✔️</td>
<td>Click on “Update Account Info” to change account information</td>
</tr>
<tr>
<td>✔️</td>
<td>Provide all specified information in the proper fields and click “Update” which directs you to your account page</td>
</tr>
<tr>
<td>✔️</td>
<td>On the navigation bar, click on “PittJungle” to be directed to the home page</td>
</tr>
<tr>
<td>✔️</td>
<td>Click “Log out” to return to the home page</td>
</tr>
</tbody>
</table>
throughout the semester. All acceptance tests that are run should be achieved within an adequate level

5.2. Acceptance Testing Checklist

<table>
<thead>
<tr>
<th>Test Cases Performed/Conducted</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>Unit Test - Test email</td>
<td>Empty $email</td>
<td>Show error message</td>
<td>11/08/2020 ✔</td>
<td>None</td>
<td>Prints out “Please fill out this field”</td>
</tr>
<tr>
<td>Test #</td>
<td>Description</td>
<td>Input</td>
<td>Expected Output</td>
<td>Date Tested</td>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>----------------------------</td>
<td>----------------------------------</td>
<td>---------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Test 2</td>
<td>Unit Test - Test email</td>
<td>Non format $email</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
<td>State Invalid email</td>
</tr>
<tr>
<td>Test 3</td>
<td>Unit Test - Test email</td>
<td>Unregistered $email</td>
<td>Show error message</td>
<td>11/06/2020</td>
<td>✔</td>
<td>Screen waiting for confirmation</td>
</tr>
<tr>
<td>Test 4</td>
<td>Unit Test - Test email</td>
<td>Registered $email</td>
<td>success</td>
<td>11/06/2020</td>
<td>✔</td>
<td>Full access to site</td>
</tr>
<tr>
<td>Test 5</td>
<td>Unit Test - Test password</td>
<td>Empty $pw</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
<td>Invalid password or email</td>
</tr>
<tr>
<td>Test 6</td>
<td>Unit Test - Test password</td>
<td>Invalid $email &amp; $pw</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
<td>Invalid entries</td>
</tr>
<tr>
<td>Test 7</td>
<td>Unit Test - Test password</td>
<td>Valid $email &amp; $pw</td>
<td>success</td>
<td>11/06/2020</td>
<td>✔</td>
<td>Logs in</td>
</tr>
<tr>
<td>Test 8</td>
<td>System Test - Sign up</td>
<td>Click link</td>
<td>Direct to register page</td>
<td>11/08/2020</td>
<td>✔</td>
<td>Redirects</td>
</tr>
</tbody>
</table>

Register new user

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 9</td>
<td>Unit Test - Test username</td>
<td>Empty $username</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
<td>Asks to fill out the field</td>
</tr>
<tr>
<td>Test 10</td>
<td>Unit Test - Test username</td>
<td>$username less than 2 chars</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>X</td>
<td>Sent confirmation email</td>
</tr>
<tr>
<td>Test 11</td>
<td>Unit Test - Test username</td>
<td>$username of 2 chars</td>
<td>success</td>
<td>11/08/2020</td>
<td>✔</td>
<td>Registers and sends confirm email</td>
</tr>
<tr>
<td>Test 12</td>
<td>Unit Test - Test username</td>
<td>$username of 20 chars</td>
<td>success</td>
<td>11/08/2020</td>
<td>✔</td>
<td>Registers and sends confirm email</td>
</tr>
<tr>
<td>Test 13</td>
<td>Unit Test - Test username</td>
<td>$username more than 20</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
<td>“Usernames must be letters,”</td>
</tr>
<tr>
<td>Test 14</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Empty $email</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>------------------------</td>
<td>--------------</td>
<td>--------------------</td>
<td>------------</td>
<td>---</td>
</tr>
<tr>
<td>Test 15</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Non format $email</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 16</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Unregistered $email</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 17</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Registered $email</td>
<td>success</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 18</td>
<td>Black box</td>
<td>Unit Test - Test password</td>
<td>Empty $pw</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 19</td>
<td>Black box</td>
<td>Unit Test - Test confirm password</td>
<td>Empty $pw2</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 20</td>
<td>Black box</td>
<td>Unit Test - Test passwords</td>
<td>Same $pw and $pw2</td>
<td>success</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 21</td>
<td>Black box</td>
<td>Unit Test - Test passwords</td>
<td>$pw different from $pw2</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 22</td>
<td>Black box</td>
<td>Unit Test - Test username</td>
<td>$username not unique</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 23</td>
<td>Black box</td>
<td>Unit Test - Test username</td>
<td>Valid $username</td>
<td>success</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Test 24</td>
<td></td>
<td></td>
<td>$email not</td>
<td>Show error</td>
<td>11/08/2020</td>
<td>✔</td>
</tr>
<tr>
<td>Black box</td>
<td>Test email</td>
<td>unique</td>
<td>message</td>
<td>exists message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>------------</td>
<td>------------------------------</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 25</td>
<td>Unit Test - Test age</td>
<td>$age more than 99</td>
<td>Show error message</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 26</td>
<td>Unit Test - Test age</td>
<td>String value for $age</td>
<td>Show error message</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 27</td>
<td>Unit Test - Test age</td>
<td>$age of 0</td>
<td>Show error message</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 28</td>
<td>Unit Test - Test age</td>
<td>Valid $age</td>
<td>success</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 29</td>
<td>System Test - Sign In</td>
<td>Click Link</td>
<td>Direct to login page</td>
<td>11/08/2020</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Update Account Info**

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 30</td>
<td>Unit Test - Test username</td>
<td>No change on $username</td>
<td>success</td>
<td>11/08/2020</td>
<td>None</td>
<td>No error</td>
</tr>
<tr>
<td>Test 31</td>
<td>Unit Test - Test username</td>
<td>Non unique $username</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>None</td>
<td>Username Already Exists</td>
</tr>
<tr>
<td>Test 32</td>
<td>Unit Test - Test username</td>
<td>Empty $username</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>None</td>
<td>Field needs to be filled out message</td>
</tr>
<tr>
<td>Test 33</td>
<td>Unit Test - Test username</td>
<td>$username less than 2 chars</td>
<td>Show error message</td>
<td>11/08/2020</td>
<td>Username under profile updated</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 34</td>
<td>Unit Test - Test username</td>
<td>$username of 2 chars</td>
<td>success</td>
<td>11/08/2020</td>
<td>None</td>
<td>Account updated</td>
</tr>
<tr>
<td>Test 35</td>
<td>Unit Test - Test username</td>
<td>$username of 20 chars</td>
<td>success</td>
<td>11/08/2020</td>
<td>None</td>
<td>Account updated</td>
</tr>
<tr>
<td>Test 36</td>
<td>White box</td>
<td>Unit Test - Test username</td>
<td>$username more than 20 chars</td>
<td>Show error message</td>
<td>11/08/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>--------------------------</td>
<td>-----------------------------</td>
<td>--------------------</td>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>Test 37</td>
<td>Black box</td>
<td>Unit Test - Test username</td>
<td>New unique $username</td>
<td>success</td>
<td>11/08/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>Test 38</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>No change on $email</td>
<td>success</td>
<td>11/08/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>Test 39</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Non unique $email</td>
<td>Show error message</td>
<td>11/08/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>Test 40</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Non format $email</td>
<td>Show error message</td>
<td>11/08/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>Test 41</td>
<td>Black box</td>
<td>Unit Test - Test email</td>
<td>Unregistered $email</td>
<td>Show error message</td>
<td>11/09/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>Test 42</td>
<td>Black box</td>
<td>Unit Test - Upload picture</td>
<td>No change to $picture</td>
<td>Show error message</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 43</td>
<td>White box</td>
<td>Unit Test - Upload picture</td>
<td>$picture not a jpg or png</td>
<td>Show error message</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 44</td>
<td>Black box</td>
<td>Unit Test - Upload picture</td>
<td>Valid $picture</td>
<td>Success</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 45</td>
<td>Black box</td>
<td>System Test - Delete account</td>
<td>Click the Delete button</td>
<td>Modal opens up</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 46</td>
<td>Black box</td>
<td>System Test - Delete account</td>
<td>Click the Cancel button</td>
<td>Modal closes</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 47</td>
<td>Black box</td>
<td>System Test - Delete account</td>
<td>Click the Delete button</td>
<td>Account gets deleted. Redirects to login page</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test #</td>
<td>Description</td>
<td>Input</td>
<td>Expected Output</td>
<td>Date Tested</td>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>------------------------</td>
<td>-----------------------</td>
<td>---------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Test 48</td>
<td>Unit Test - Add Comment</td>
<td>Empty textfield</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None Textfield needs to be filled out to submit comment</td>
</tr>
<tr>
<td>Test 49</td>
<td>Unit Test - Add Comment</td>
<td>More than 500 chars</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>X</td>
<td>Comment Posted</td>
</tr>
<tr>
<td>Test 50</td>
<td>Unit Test - Add Comment</td>
<td>Exactly 500 chars</td>
<td>success</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None Comment Posted</td>
</tr>
<tr>
<td>Test 51</td>
<td>Unit Test - Add Comment</td>
<td>Less than 2 chars</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>X</td>
<td>Comment Posted</td>
</tr>
<tr>
<td>Test 52</td>
<td>Unit Test - Add Comment</td>
<td>Exactly 2 chars</td>
<td>success</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None Comment Posted</td>
</tr>
<tr>
<td>Test 53</td>
<td>Unit Test - Edit Comment</td>
<td>Empty textfield</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None Asks to fill out comment field if nothing</td>
</tr>
<tr>
<td>Test 54</td>
<td>Unit Test - Edit Comment</td>
<td>More than 500 chars</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>X</td>
<td>Allows Edit TBD</td>
</tr>
<tr>
<td>Test 55</td>
<td>Unit Test - Edit Comment</td>
<td>Exactly 500 chars</td>
<td>success</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None Comment Posted</td>
</tr>
<tr>
<td>Test 56</td>
<td>Unit Test - Edit Comment</td>
<td>Less than 2 chars</td>
<td>Show error message</td>
<td>11/09/2020</td>
<td>X</td>
<td>Allows Edit TBD</td>
</tr>
<tr>
<td>Test 57</td>
<td>Unit Test - Edit Comment</td>
<td>Exactly 2 chars</td>
<td>success</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None Comment Posted</td>
</tr>
<tr>
<td>Test 58</td>
<td>System Test - Delete</td>
<td>Click the Delete button</td>
<td>Modal opens up</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None “Disabled by admin”</td>
</tr>
<tr>
<td>Test 59</td>
<td>System Test - Delete</td>
<td>Click the Delete button</td>
<td>Comment deleted</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None “Disabled by admin”</td>
</tr>
<tr>
<td>Test #</td>
<td>Description</td>
<td>Input</td>
<td>Expected Output</td>
<td>Date Tested</td>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------</td>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Test 60</td>
<td>System Test - Delete comment</td>
<td>Click the Cancel button</td>
<td>Modal closes</td>
<td>11/09/2020</td>
<td>✔</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“Disabled by admin”</td>
</tr>
</tbody>
</table>

### Recommendation System

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 61</td>
<td>Unit Test - Food Probability</td>
<td>User’s food tags</td>
<td>Success: Probability associated with each food classification</td>
<td>11/09/2020</td>
<td>✔</td>
<td>Can be viewed on screen</td>
</tr>
<tr>
<td>Test 62</td>
<td>Unit Test - Activity Probability</td>
<td>User’s Activity Tags</td>
<td>Success: Probability associated with each activity classification</td>
<td>11/09/2020</td>
<td>✔</td>
<td>Can be viewed on screen</td>
</tr>
<tr>
<td>Test 63</td>
<td>Unit Test - New Clicks</td>
<td>Tags obtained from click</td>
<td>Success: An updated probability distribution</td>
<td>11/09/2020</td>
<td>✔</td>
<td>Not all pages done to fully test individual tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50/50</td>
<td></td>
<td>Pages need to be completed</td>
</tr>
<tr>
<td>Test 64</td>
<td>Unit Test - Add User ID</td>
<td>Integer</td>
<td>Success: An integer associated with each user</td>
<td></td>
<td></td>
<td>Has been added to User database model</td>
</tr>
<tr>
<td>Test 65</td>
<td>Unit Test - Add User ID</td>
<td>String</td>
<td>Error</td>
<td></td>
<td></td>
<td>Has been added to User Database Model</td>
</tr>
<tr>
<td>Test 66</td>
<td>Unit Test - Display Recommendation</td>
<td>Highest Individual user probability</td>
<td>Success</td>
<td>Date</td>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------</td>
<td>------------------------------------</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Distribution shown on screen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test 67</th>
<th>Unit Test - Display Recommendations</th>
<th>Top 3 Probabilities</th>
<th>Success</th>
<th>Date</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test 68</th>
<th>Unit Test - Display Recommendation</th>
<th>None</th>
<th>Success</th>
<th>Date</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Distribution shown on screen</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test 69</th>
<th>Unit Test - Probability Distribution</th>
<th>Probabilities</th>
<th>Success: Written to file to verify</th>
<th>Date</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>File shows up in project folder</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test 70</th>
<th>Unit Test - New Clicks Added to User</th>
<th>Tags associated with clicks</th>
<th>Success: Updated individual user’s database</th>
<th>Date</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Tags update by specified value</td>
<td></td>
</tr>
</tbody>
</table>

---

## User Event

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 71</td>
<td>Unit test - Enter Title</td>
<td>String</td>
<td>Success</td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Submission allowed</td>
</tr>
<tr>
<td>Test 72</td>
<td>Unit Test - Enter Title</td>
<td>Non-string</td>
<td>Error Message</td>
<td>11/09/2020 X</td>
<td>Numbers worked</td>
<td>Made input location convert to string so this issue is resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/09/2020 ✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test 73</td>
<td>Unit Test - Location</td>
<td>String</td>
<td>Success</td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Submits with no error</td>
</tr>
<tr>
<td>Test #</td>
<td>Description</td>
<td>Input</td>
<td>Expected Output</td>
<td>Date Tested</td>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>----------------------------</td>
<td>-----------------</td>
<td>-------------</td>
<td>--------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Test 74</td>
<td>Unit Test - Location</td>
<td>Non-String</td>
<td>Error Message</td>
<td>11/09/2020</td>
<td>Same issue with test 72</td>
<td>Resolved by making input fields convert everything to string (Test case now obsolete)</td>
</tr>
<tr>
<td>Test 75</td>
<td>Unit Test - Date</td>
<td>YYYY-MM-DD String Format</td>
<td>Success</td>
<td>11/08/2020</td>
<td>none</td>
<td>Submits date</td>
</tr>
<tr>
<td>Test 76</td>
<td>Unit Test - Date</td>
<td>Anything else</td>
<td>Error Message</td>
<td>11/09/2020</td>
<td>None</td>
<td>Error Message given</td>
</tr>
<tr>
<td>Test 77</td>
<td>Unit Test - Click Post</td>
<td>Button Click</td>
<td>No Error</td>
<td>11/09/2020</td>
<td>None</td>
<td>Event Posted, Can be seen upon return to event page</td>
</tr>
</tbody>
</table>

### Rating

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 78</td>
<td>Unit Test - Give Rating</td>
<td>Integer between 0-10</td>
<td>Success</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 79</td>
<td>Unit Test - Give Rating</td>
<td>Integer &lt;0 or &gt; 10</td>
<td>Error: invalid rating</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Test 80</td>
<td>Unit Test - Give Rating</td>
<td>Non-Integer</td>
<td>Error: invalid format</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>

### Place

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 81</td>
<td>Unit Test - Place Name</td>
<td>String Length &gt;0 &amp; &lt;150</td>
<td>Success</td>
<td>11/09/2020</td>
<td>None</td>
<td>Gets posted to restaurant page</td>
</tr>
<tr>
<td>Test 82</td>
<td>Unit Test - Place Name</td>
<td>String &lt;0</td>
<td>&gt; 150</td>
<td>Error Message: Invalid length</td>
<td>11/09/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td>----------------</td>
<td>---------------------------------</td>
<td>----------------</td>
<td>------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Test 83</td>
<td>Unit Test - Place Name</td>
<td>Integer, boolean, double, etc</td>
<td>Error: Invalid format</td>
<td>11/09/2020 X Resolved 11/09/2020 ✔</td>
<td>Numbers allowed</td>
<td>Input fields now convert everything to string during submission</td>
</tr>
<tr>
<td>Test 84</td>
<td>Unit Test - Place Location</td>
<td>String Length &gt;0 &amp; &lt;200</td>
<td>Success</td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Submits with form</td>
</tr>
<tr>
<td>Test 85</td>
<td>Unit Test - Place Location</td>
<td>String Length &lt;0</td>
<td>&gt;200</td>
<td>Error: Invalid Length</td>
<td>11/09/2020 ✔</td>
<td>None</td>
</tr>
<tr>
<td>Test 86</td>
<td>Unit Test - Place</td>
<td>Integer, Float, Double, boolean, etc</td>
<td>Error: invalid format</td>
<td>11/09/2020 X Resolved 11/09/2020 ✔</td>
<td>Numbers allowed</td>
<td>Input fields now convert everything to string during submission</td>
</tr>
<tr>
<td>Test 87</td>
<td>Unit Test - Item Tag</td>
<td>String from Predefined list</td>
<td>Success</td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Gets added to tag list on place</td>
</tr>
<tr>
<td>Test 88*</td>
<td>Unit Test - Item Tag</td>
<td>Anything outside of predefined list</td>
<td>Error: Tag is invalid</td>
<td>11/09/2020 X</td>
<td>Any tag name is added</td>
<td>TBD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 89</td>
<td>Unit Test - Name</td>
<td>String</td>
<td>Success</td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Name shows to screen</td>
</tr>
<tr>
<td>Test 90</td>
<td>Unit Test - Name</td>
<td>Integer, Boolean, Float,</td>
<td>Error: Must be of format</td>
<td>11/09/2020 ✔</td>
<td>None</td>
<td>Name must be</td>
</tr>
</tbody>
</table>
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 occurring. 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**

```python
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)
```

49
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'),
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/tlessiq8771/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”
     o 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](image)

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.
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
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.
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.
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 with IC 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Functionality</td>
<td>Iteratively introduce features</td>
</tr>
<tr>
<td>Feature Adaptation</td>
<td>Flexible</td>
</tr>
<tr>
<td>User Involvement</td>
<td>90%</td>
</tr>
<tr>
<td>Documentation</td>
<td>Not produced</td>
</tr>
<tr>
<td>Experienced Team</td>
<td>Requested</td>
</tr>
<tr>
<td>Model Type</td>
<td>Linear</td>
</tr>
<tr>
<td>Planning and Scheduling</td>
<td>Continuous</td>
</tr>
<tr>
<td>Risk Mitigation</td>
<td>Yes</td>
</tr>
<tr>
<td>Project Size</td>
<td>Small</td>
</tr>
<tr>
<td>Prototype</td>
<td>Used</td>
</tr>
<tr>
<td>CrossPlatform</td>
<td>No</td>
</tr>
</tbody>
</table>

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.
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.

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

Submit Parameters

**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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Functionality</td>
<td>Iteratively introduce features</td>
</tr>
<tr>
<td>Feature Adaptation</td>
<td>Impossible</td>
</tr>
<tr>
<td>User Involvement</td>
<td>(Initially) 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% (Frequent feedback)</td>
</tr>
<tr>
<td>Documentation</td>
<td>Not produced</td>
</tr>
<tr>
<td>Experienced Team</td>
<td>Requested</td>
</tr>
<tr>
<td>Model Type</td>
<td>(Linear) 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% (Iterative)</td>
</tr>
<tr>
<td>Planning and Scheduling</td>
<td>Uplift</td>
</tr>
<tr>
<td>Risk Mitigation</td>
<td>Yes</td>
</tr>
<tr>
<td>Project Size</td>
<td>(Small) 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% (Large)</td>
</tr>
<tr>
<td>Prototype</td>
<td>Used</td>
</tr>
<tr>
<td>CrossPlatform</td>
<td>No</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th></th>
<th>Project Manager</th>
<th>GUI Designer</th>
<th>Databases/Backend</th>
<th>Testing</th>
<th>Research</th>
<th>Recommender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sherryl</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Sush</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luiza</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Trent</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

2.2. Summary of the discussion among stakeholders

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

➢ **Pros of the Chatbot**
  - Attention grabber
  - Resourceful
  - Increased User Interaction

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

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

➢ **Cons of implementing Academic section**
  - Not very ambitious
  - Could be confused with the ‘events’ section

➢ **Cons of the Chatbot**
  - Time Consuming and Intensive
  - Beyond the teams experience level

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

➢ **Cons 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

<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Input</th>
<th>Expected Output</th>
<th>Date Tested</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New User Registration</td>
<td>Filled out Form</td>
<td>Success: User receives email to verify. User can login</td>
<td>11/11/2020</td>
<td>✔</td>
<td>Email Received, Link is clickable and account becomes verified</td>
</tr>
<tr>
<td>2</td>
<td>User Event Creation</td>
<td>Filled out Form</td>
<td>Success: Form submitted and event shows on event page</td>
<td>11/11/2020</td>
<td>✔</td>
<td>Can be viewed on screen</td>
</tr>
<tr>
<td>3</td>
<td>Recommendation System</td>
<td>Tags obtained from clicking</td>
<td>Success: An updated probability distribution and suggestions shown on page</td>
<td>11/11/2020</td>
<td>✔</td>
<td>Recommendation shown on Rec page to user</td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Success Criteria</td>
<td>Date</td>
<td>Grade</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Commenting</td>
<td>Input field&lt;br&gt;Success: Comment shows on main screen</td>
<td>11/11/2020</td>
<td>✔</td>
<td>Comment seen on main screen by any user</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Admin Comment Management</td>
<td>Button Input&lt;br&gt;Success: Admin can enable/disable comments if deemed necessary</td>
<td>11/10/2020</td>
<td>✔</td>
<td>Comment shows up as disabled if disabled</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Add Friend</td>
<td>Follow/Unfollow button option&lt;br&gt;Success</td>
<td>11/09/2020</td>
<td>✔</td>
<td>Friend list shows on main profile and shows how many followers and how many you follow</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Logging In</td>
<td>Login Form&lt;br&gt;Success</td>
<td>11/09/2020</td>
<td>✔</td>
<td>User gets logged in</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Filtering</td>
<td>Drop-Down menu and submit button&lt;br&gt;Success</td>
<td>11/14/2020</td>
<td>✔</td>
<td>The selection submitted only shows corresponding items associated with the filter</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Update Profile</td>
<td>Update profile form and submission&lt;br&gt;Success: Updated account shows on users main profile page</td>
<td>11/12/2020</td>
<td>✔</td>
<td>Users can see profile of others user and what they included with it</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Recommendation Probability Distribution</td>
<td>Rec. Objects&lt;br&gt;Success</td>
<td>11/15/2020</td>
<td>✔</td>
<td>Written to file “UserProbs.txt”&lt;br&gt;*is updated each time the rec page is visited</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>View Other Events</td>
<td>Events Page&lt;br&gt;Success</td>
<td>11/14/2020</td>
<td>✔</td>
<td>Users can see events posted by other users</td>
<td></td>
</tr>
</tbody>
</table>
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)
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