

TITLE: LESS THAN THREE

SUBTITLE: e-Vent | SEMESTER: Spring 2018

Installation requirements: You need to run command: "pip3 install django rest framework" + "pip install django-money" + "pip3 install django-password-reset" to install necessary dependencies.

Link to Slides presentation: <https://prezi.com/view/EzL9OjXA1gcGQiiB814u/>

Link to Single slide: <https://tinyurl.com/y9hzm9ss>

Overview: e-Vent is an application that consolidates all the events taking place around the 5 college area in one easy to access website. Users of this application will be able to view nearby and filter them by location, category and price. They can also sign up to create and save events and view their profile. This application provides a unique platform for students in the 5 college to access and advertise events with ease.

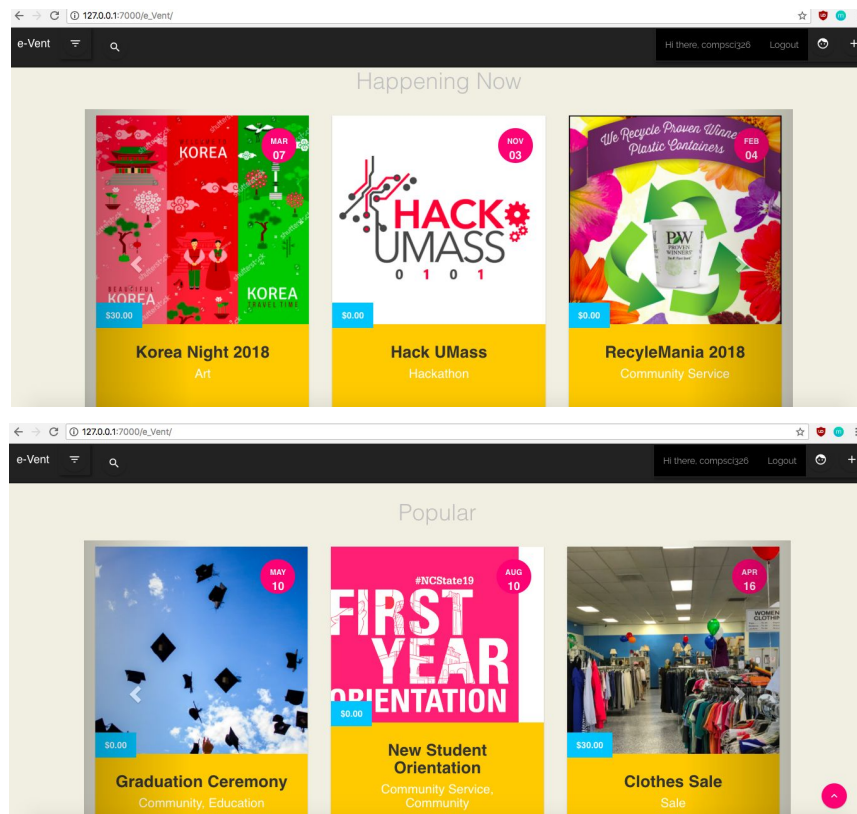
Team Members: Vi Nguyen, Shanieka Pereira, Anh Chau Pham, Aastha Niraula, Depeng Kong, Harry Hu

Git Repo: <https://github.com/aasthan/web-programming>

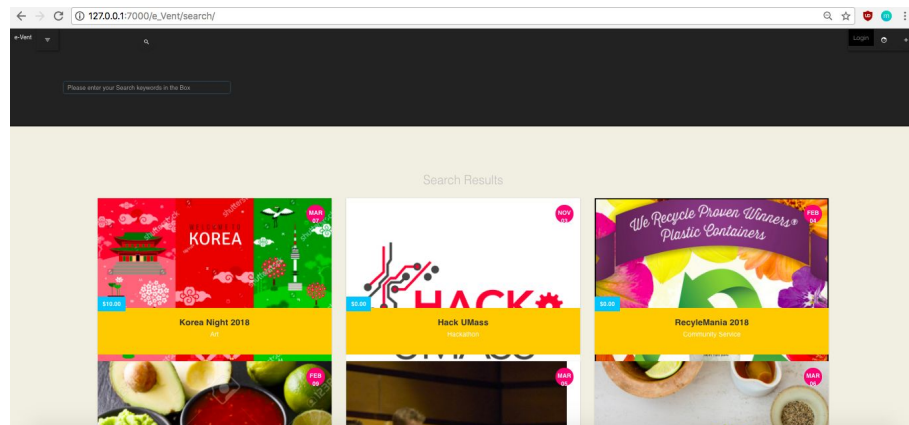
1. User Interface:

The basic interface of our app e-Vent that stays constant on most of the view pages is a navigation bar with the app logo, toggle button for filter, search button that expands to a search bar, login/logout button and a create event button as well as a footer at the bottom of the page.

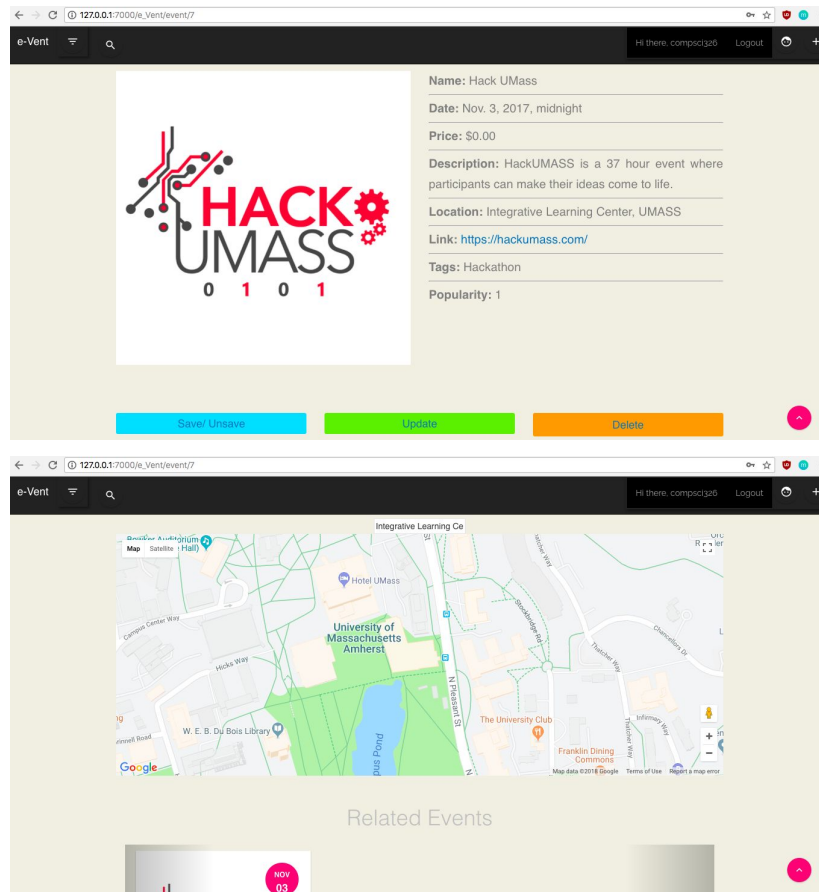
- **IndexView** is the homepage of the app. It includes two carousels that display events that are happening now and are popular in the app respectively.



- **EventSearchView** is the page that can be used to search in the app and renders the search results. The search icon links to this page and users can type on the text field to perform search.

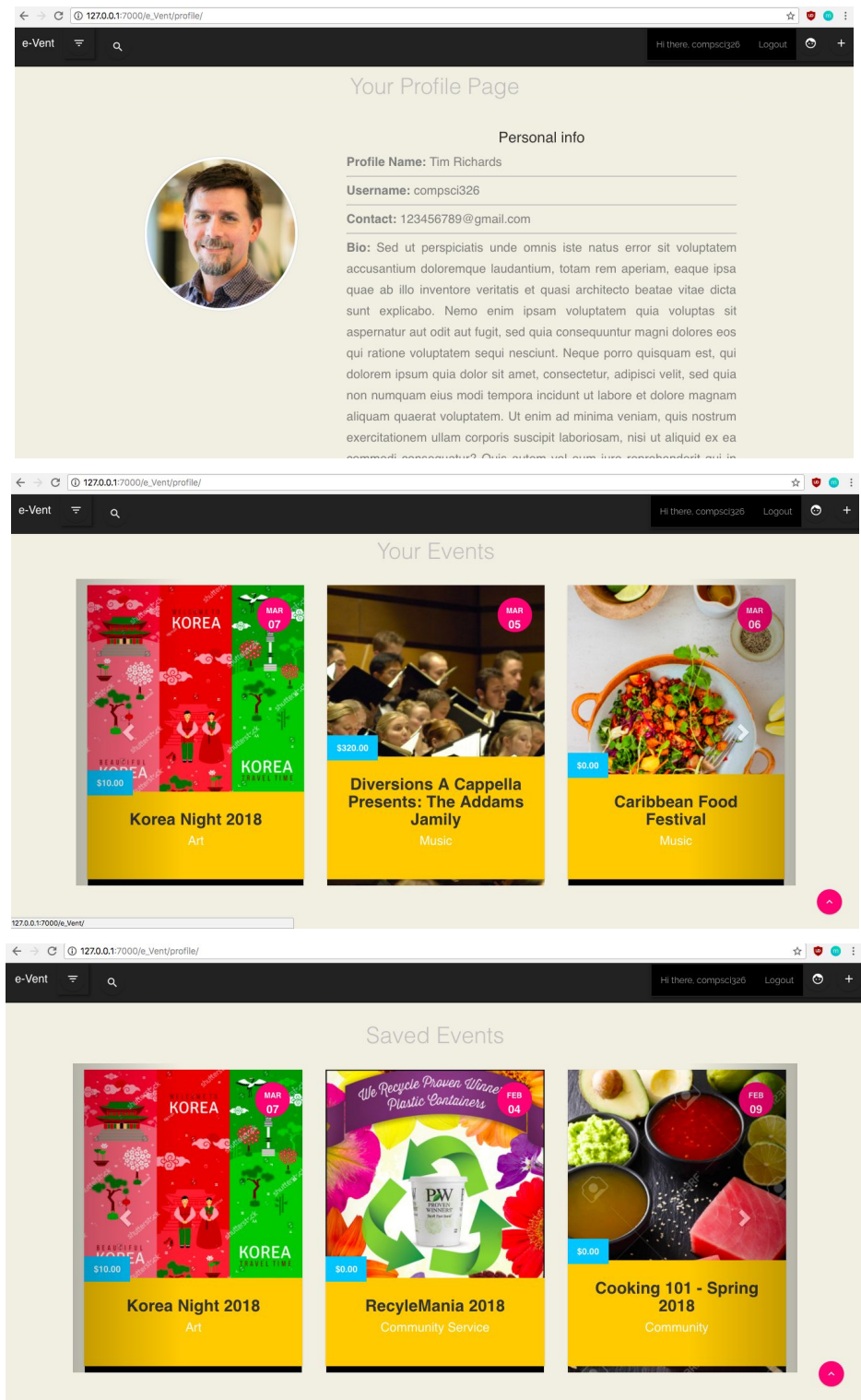


- **EventDetailView** displays a specific event and its detailed information. Once the user clicks on an event, it leads to the specific EventDetailView page that renders its image, description, links to lead to save/unsave, update and delete views. It also has a map with the location of the event and a carousel with related events displayed.



- **YourEventsView** is the Profile page of the view. It displays basic information such as the picture, name, contact details and bio. The two carousels at the bottom of the page render “Your Events”

which are the events created by the user and “Saved Events” which are the events saved by the user.



- **EventCreateView** displays a form that is linked to the add button on the navigation bar. This form can be used by registered users to create an event and display it on the application.

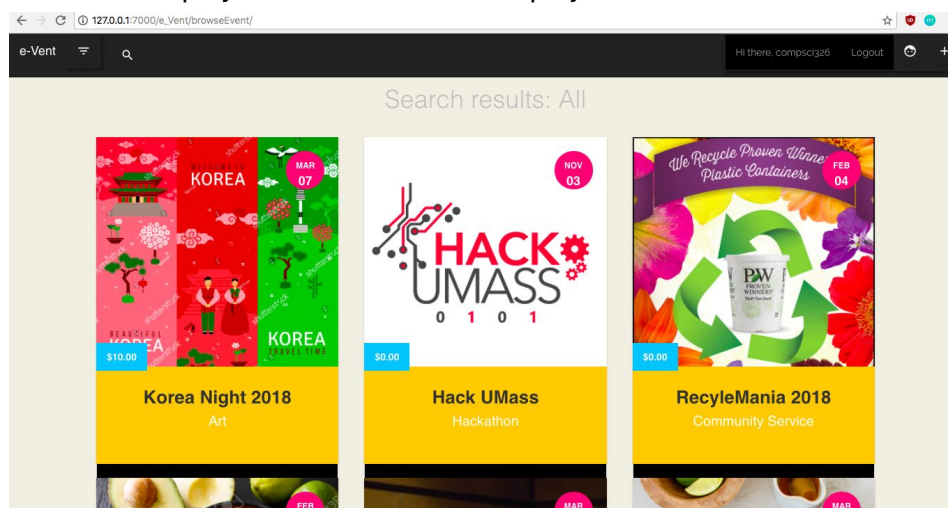
The top screenshot shows the 'Let's Add an Event!' form with the following fields:

- Name of Event:
- Organizer:
- Start Time:
- End Time:
- Location:
- Entry Fee:
- Category:

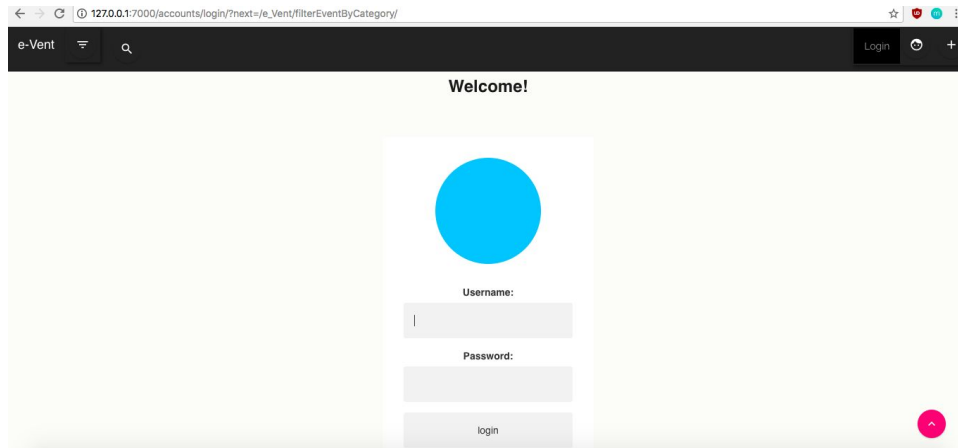
The bottom screenshot shows the continuation of the form with the following fields:

- Description:
- Link to the Event:
- Upload Image:
-

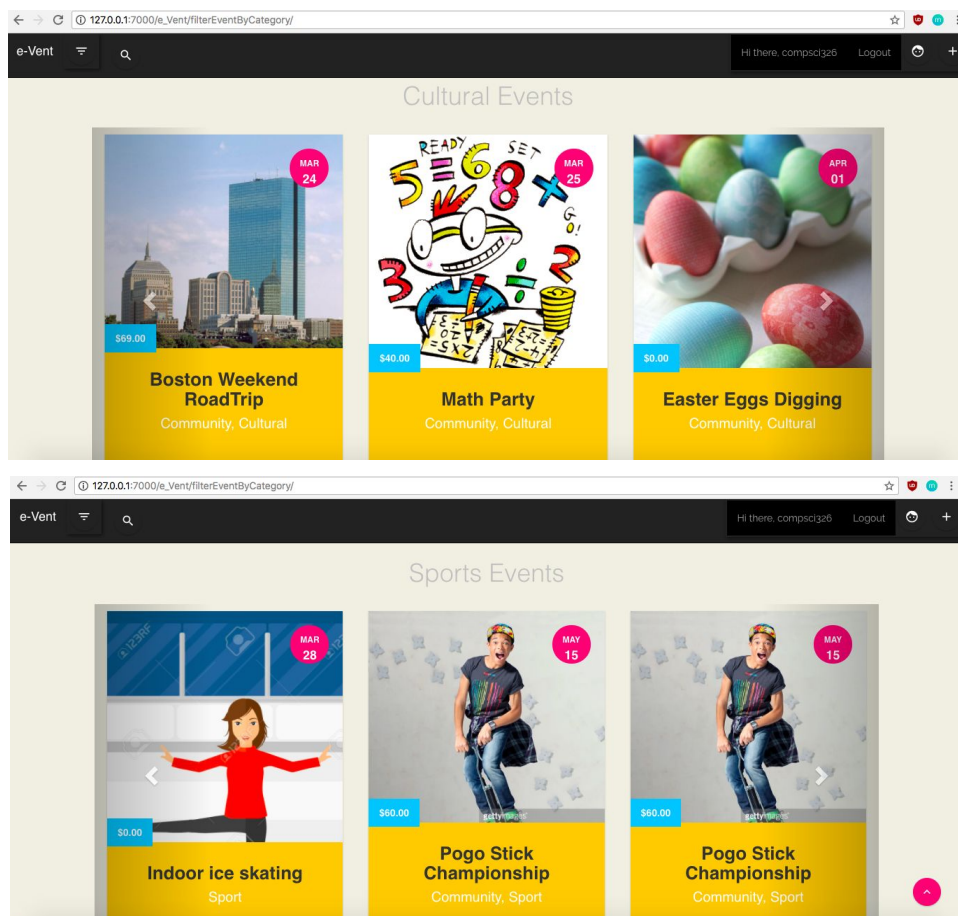
- **BrowseEventsView** can be accessed by clicking on “All” from the toggle down button in the navigation bar. This displays all events in a tile display form instead of a carousel.



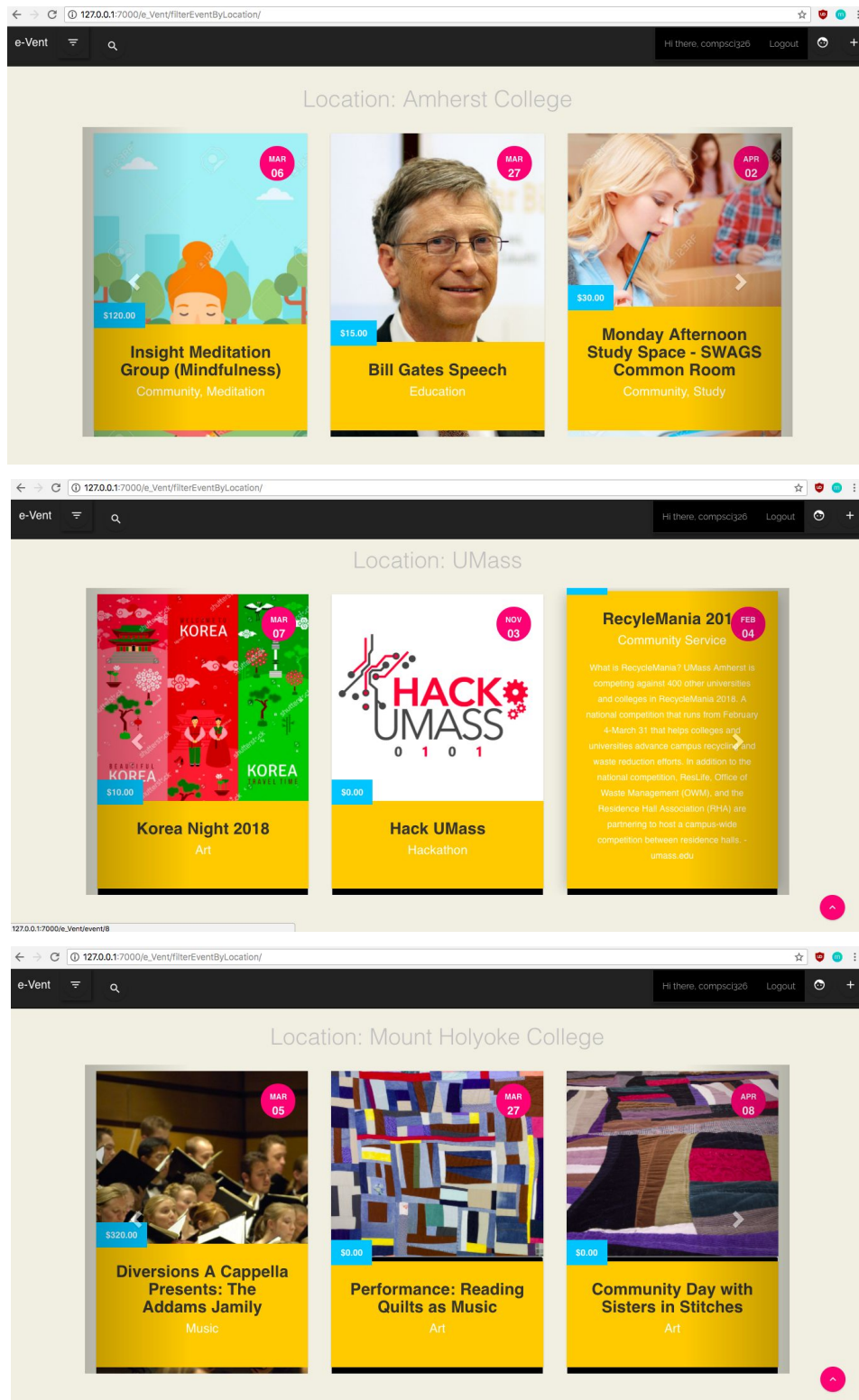
- **LoginView** displays a form that is used to authenticate users in the application. It is linked to the “Login” button on the navigation bar.



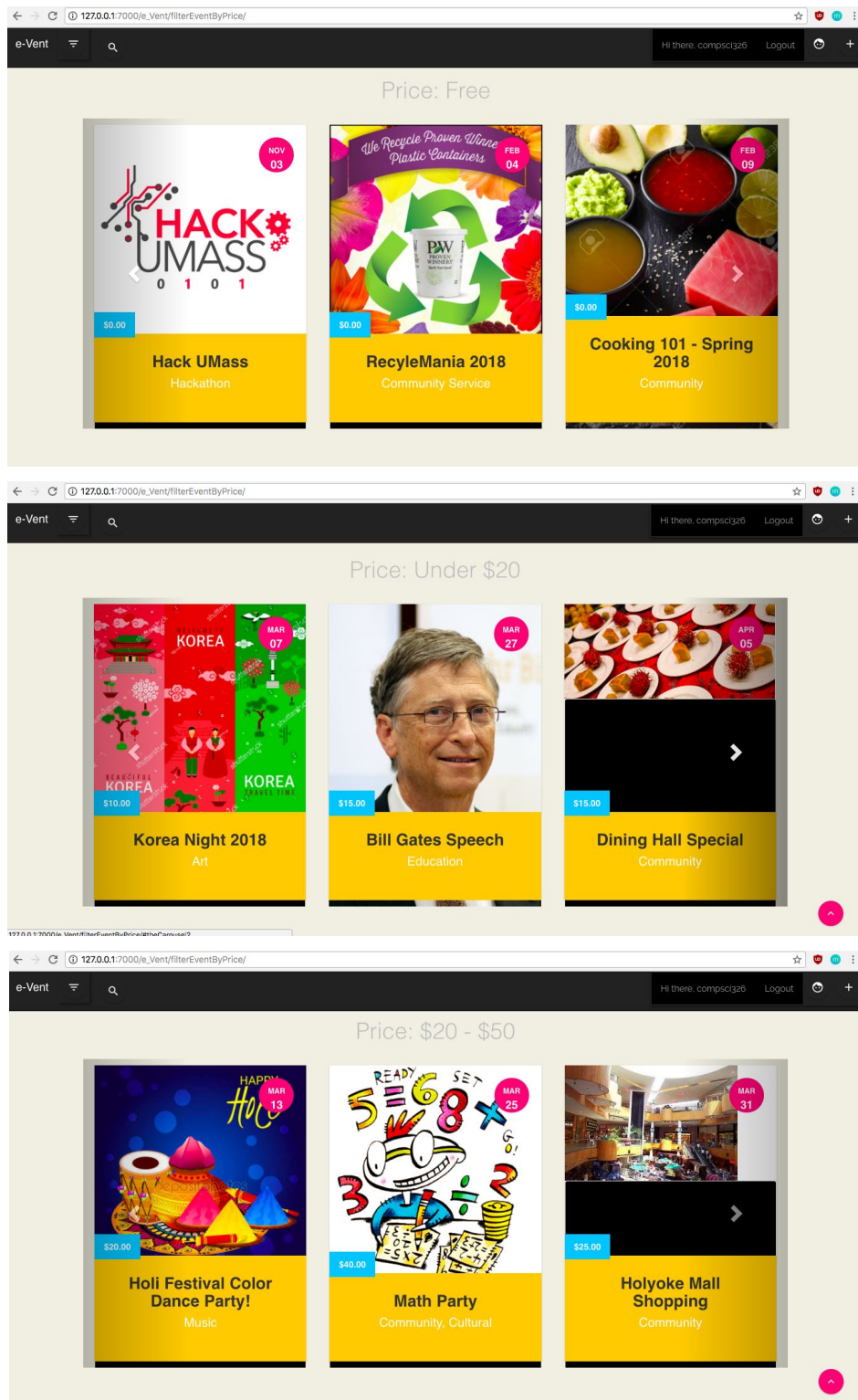
- **FilterEventByCategoryView** displays events that have been filtered according to different categories and are displayed in various carousels. This view can be accessed from the “Category” option from the toggle down icon in the navigation bar.



- **FilterEventByLocationView** displays events that have been filtered according to the five college locations and are displayed in various carousels. This view can be accessed from the “Location” option from the toggle down icon in the navigation bar.



- **FilterEventByPriceView** displays events that have been filtered according to different price ranges and are displayed in various carousels. This view can be accessed from the “Price” option from the toggle down icon in the navigation bar.



- **EventUpdate** is linked to the “Update” button in the EventDetailView page. It renders a form that has the details from the specific event. Authenticated users can update the information in the form.

The top screenshot shows the 'Let's Add an Event!' form. The fields are as follows:

- Name of Event: Korea Night 2018
- Organizer: compsci326
- Start Time: 2017-03-07 06:00:00
- End Time: 2018-04-03 06:00:00
- Location: UMass Bowker Auditorium
- Entry Fee: \$30.00
- Category: Music (selected)

The bottom screenshot shows the form with a modal dialog for specifying the category and description. The modal has the following fields:

- Please specify the category
- Description: Please specify the category

The form also includes a 'Link to the Event' field with the URL <https://www.umass.edu/asian/korean-language-prog> and an 'Upload Image' field with the current image [imgs/Korea_eAbiFin.jpg](#). At the bottom, there are 'CANCEL' and 'SUBMIT' buttons.

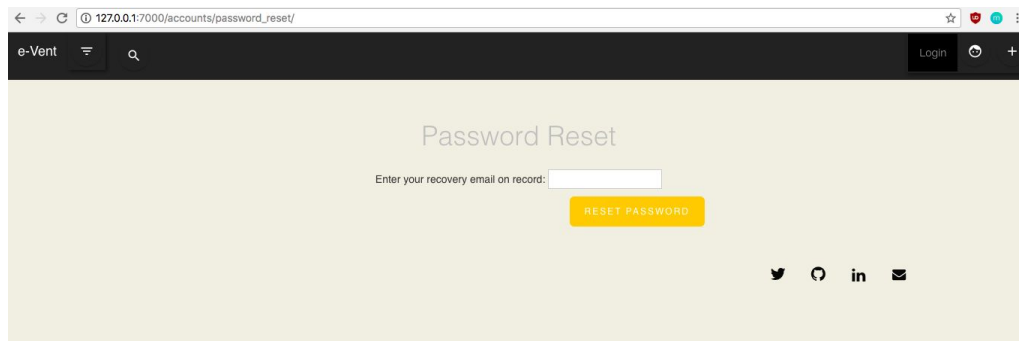
- **EventDelete** is linked to the “Delete” button in the EventDetailView page. It renders a page that confirms if the authenticated user wants to delete the specific event.

The screenshot shows the 'Delete Event' page. The confirmation message is:

Are you sure you want to delete the event: Cooking 101 - Spring 2018?

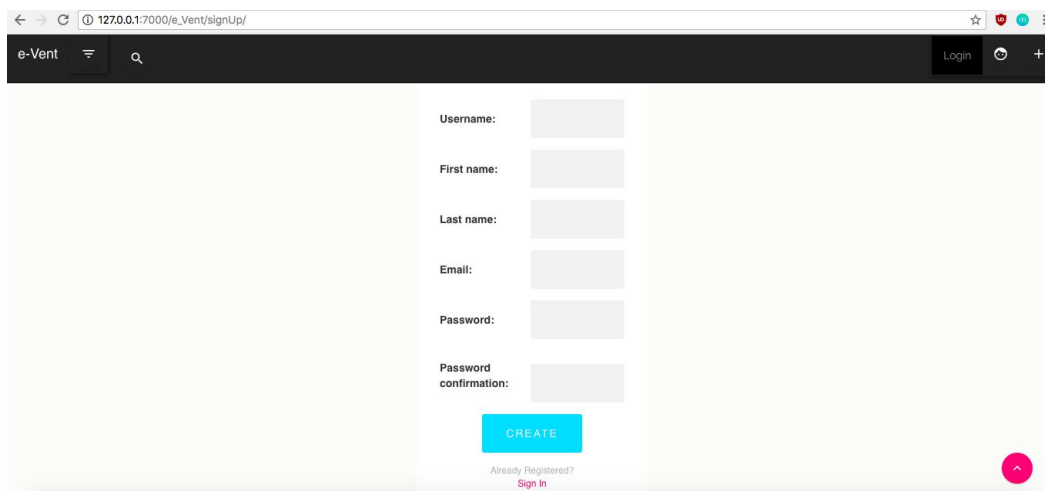
Below the message is a button labeled 'Yes, delete.' At the bottom right, there are social media icons for Twitter, Facebook, LinkedIn, and Email.

- **PasswordResetView** is the view that is displayed when the user clicks on the “Lost Password” button in the Login view. The user is able to enter their email address and will receive a link to reset their password in their email.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:7000/accounts/password_reset/`. The page has a dark header with the "e-Vent" logo, a search icon, and a "Login" button. The main content area is light beige and features the heading "Password Reset". Below the heading is a form with the label "Enter your recovery email on record:" followed by a text input field. A yellow button labeled "RESET PASSWORD" is positioned below the input field. At the bottom right of the form, there are social media icons for Twitter, GitHub, LinkedIn, and Email.

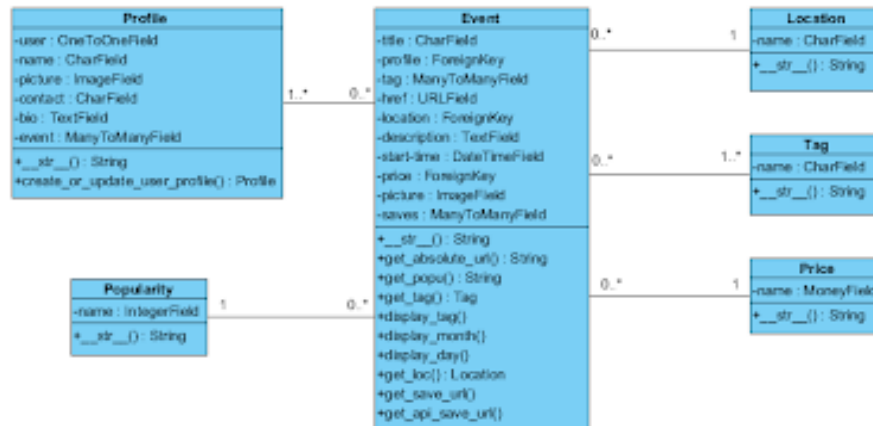
- **SignUp** is the view that is displayed when the user clicks on “Create an account” from the Login view. The page renders a form that can be used to enter information to create an account.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:7000/e_Vent/signup/`. The page has a dark header with the "e-Vent" logo, a search icon, and a "Login" button. The main content area is light beige and features a registration form. The form includes the following fields: "Username:", "First name:", "Last name:", "Email:", "Password:", and "Password confirmation:". Below these fields is a blue button labeled "CREATE". At the bottom of the form, there is a link that says "Already Registered? Sign In". A pink circular button with a white arrow is located in the bottom right corner of the page.

2.Data Model: (A final up-to-date diagram of your data model including a brief description of each of the entities in your model and their relationships.)

- We recognize that Popularity and Price should not be models; however, making the switch messed up with our database and would cause us to wipe out our database; therefore, we decided that we will leave them as models still and work on it in the near future.



- The application we used to draw diagram has expired but the most up-to-date model only has minor changes:
- In Event model, there are 11 fields and 10 functions.
 - Title field is the name of this event object. We use the CharField, because one event can only have one name, and the title should be unique and required in the database.
 - Profile field is for the creator of this event object. We use the ForeignKey because one event can only have one user, but a user can have multiple events. It associate with the profile model.
 - Tag field is used to describe the category of this event object. We use the ManyToManyField because one event can have multiple categories, and a category can have multiple events. It associate with the tag model.
 - Href field is for external link of this event object. The content of this entry is url, so we use URLField.
 - Location field is used to describe the location of this event object. We use the ForeignKey because one event can only have one location, but a location can hold multiple events. It associate with the location model.
 - Description field is used for description of this event. We use the TextField, because it is not required.
 - Start_time field is the start time of this event object. We use the DateTimeField, because start time is time, and we want a proper field to regular the format of time.
 - End_time field is the end time of this event object. We use the DateTimeField, because end time is time, and we want a proper field to regular the format of time.
 - Price field is for the price of this event object. We use the ForeignKey because one event can only have one price, but multiple events can have the same price. It associate with the price model.

- j. Picture field is used for the picture to represent this event object. We use the ImageField because we want verify the uploaded files are valid pictures.
- k. Saves field is used for the saved events function that users can save this event and belong to this field. We use the ManyToManyField because one event can have multiple users who save it, and a user can save multiple events. It also needs authorized users to save the events.
- l. __str__ function returns the name of the event, which is title field.
- m. Get_popu function returns the popularity of the event, which is counted through the number of instances in saves field.
- n. Get_absolute_url function returns the url of this event, which is self.id.
- o. Get_tag function returns the all tags that this event belong to. All tags are in the tag field.
- p. Display_tag function returns the string of all tags.
- q. Display_month function returns the month of start time.
- r. Display_day function returns the day of start time.
- s. Get_loc function returns the location of the event.
- t. Get_save_url and get_api_save url are two functions to achieve the saved events function.
- In Profile model, there are 6 fields and 2 functions.
 - a. User field is for the authorized user who owns this profile. We use the OnetoOneField because an authorized user can only have one profile, and a profile can be owned by only one user.
 - b. Name field is the nickname of this profile object. We use the CharField, because one profile can only have one name, and the name should be unique and required in the database.
 - c. Picture field is used for the picture to represent this profile object. We use the ImageField because we want verify the uploaded files are valid pictures.
 - d. Contact field is for the method to contact the user. We use the CharField, because one profile can only have one contact method(email or cell phone number), and the contact should be unique and required in the database.
 - e. Bio field is used for description of this user/profile. We use the TextField, because it is not required.
 - f. Event field is used for the saved events function that users can save this event. Events saved belong to this field. We use the ManyToManyField because one event can have multiple users who save it, and a user can save multiple events.
 - g. __str__ function returns username of this profile, which represents this profile object.
 - h. Create_or_update_user_profile function used an if statement to distinguish between the creating request and updating request.
- In Popularity model, there are 1 field and 1 function.
 - a. Name field is for the event popularity which is counted by number of people who saved the event. We use IntergerField because the popularity of an event is an interger, and the lowest popularity for an event should be 0.
 - b. __str__ function returns the number of the popularity.
- In Location model, there are 1 field and 1 function.
 - a. Name field is for the name of the location. We use the CharField, because one location can only have one name, and the name should be unique and required in the database.
 - b. __str__ function returns the name of the location.
- In Price model, there are 1 field and 1 function.
 - a. Name field is for the number of the price of the event. We use MoneyField because the price should be represented by money, and the price can be accurate in 2 decimals.
 - b. __str__ function returns the number of the price.

- In Tag model, there are 1 field and 1 function.
 - a. Name field is for the name of the tag. We use the CharField, because one tag can only have one name, and the name should be unique and required in the database.
 - b. `__str__` function returns the number of the tag.

3.URL Routes/Mappings: *(A final up-to-date table of all the URL routes that your application supports and a short description of what those routes are used for. You should also indicate any authentication and permissions on those routes.)*

Our application contains the following URL routes:

- 1.path('search/', views.EventSearchView.as_view(), name='search'),
 - This URL route is for the search page.
 - No special authentication or permission needed for visiting this page.
- 2.path('', views.indexView, name='index'),
 - This URL route is for the index page.
 - This is the default homepage for our application.
 - No special authentication or permission needed for visiting this page.
- 3.path('event/<int:pk>', views.EventDetailView.as_view(), name='event-details'),
 - This URL route is for browsing a single event.
 - It is dynamically generated based on the event number in the database.
 - No special authentication or permission needed for visiting this page.
- 4.path('event/<int:pk>/save/', views.PostSaveToggle.as_view(), name='save'),
 - This URL route is for the save function of our application.
 - This page will be automatically transferred back to the Event Detail page after the database is updated.
 - Authentication and log in is required.
- 5.path('event/api/<int:pk>/save/', views.PostSaveAPIToggle.as_view(), name='api-save'),
 - This URL route is for the api of save function of our application.
 - Authentication and log in is required.
 - This page is used by Django Rest Framework and is not shown to all users.
- 6.path('profile/', views.YourEventsView.as_view(), name='profile'),
 - This URL route is for browsing the current user profile.
 - Authentication and login is required.
- 7.path('form/', views.EventFormView.as_view(), name='form'),
 - This URL route is for create a new event.
 - This is used for template only, and not used by the final application.
 - This page is not shown to all users.
- 8.path('browseEvent/', views.browseEventsView, name='browse-Event'),
 - This URL route is for displaying all the events in the database.
 - No special authentication or permission needed for visiting this page.
- 9.path('login/', views.LoginView.as_view(), name='login'),
 - This URL route is for displaying the login page form.
 - No special authentication or permission needed for visiting this page.
- 10.path('filterEventByPrice/', views.filterEventByPriceView, name='filter-event-by-price'),
 - This URL route is for the page which filters all events in database by price and displays them sorted based on "Free", "\$0 - \$20", "\$20 - \$50" and "Over \$50".
 - No special authentication or permission needed for visiting this page.
- 11.path('filterEventByLocation/', views.filterEventByLocationView, name='filter-event-by-location'),

- This URL route is for the page which filters all events in database by location and displays them sorted based on each of the five colleges.
 - No special authentication or permission needed for visiting this page.
- 12.path('filterEventByCategory/', views.filterEventByCategoryView, name='filter-event-by-category'),
- This URL route is for the page which filters all events in database by category and displays them sorted based on "Cultural Events" and "Sport Events".
 - No special authentication or permission needed for visiting this page.
- 13.path('event/<int:pk>/update', views.EventUpdate.as_view(), name='event_update'),
- This URL route is for updating a single event.
 - Authentication and login is required.
- 14.path('event/<int:pk>/delete', views.EventDelete.as_view(), name='event_delete'),
- This URL route is for deleting a single event.
 - Authentication and login is required.
- 15.path('event/create/', views.EventCreate.as_view(), name='event_create'),
- This URL route is for creating a single event.
 - Authentication and login is required.
- 16.path('signUp/', views.signup, name='signup'),
- This URL route is for displaying the sign up page form.
 - No special authentication or permission needed for visiting this page.

4.Authentication/Authorization: *(A final up-to-date description of how users are authenticated and any permissions for specific users (if any) that you used in your application. You should mention how they relate to which UI views are accessible.)*

- **Anonymous (unauthenticated) users** can view all on-going and popular events as displayed in homepage. They can also see each event's details but is not allowed to "save" anything yet. In short, anonymous users can only access index page + eventDetail page. To view profile page, they will be redirected to sign up/log in page first.
- **Authenticated users** have more permissions than anonymous users. They have their own profiles and can "save" any events they are interested in. Moreover, they are allowed to create more events. Only the owner "user" can update/delete them. In short, authenticated users have access to all pages. In profile page, they will see their "saved" events and also events created by them.
- **Superuser** (compsci326) has all the permissions and can access the database.

5.Team Choice: For team choice, we have incorporated a wide range of additional functional and decorative add-ons:

- For each events, we implement the function to Filter events based on Category, Location and Price. We did this by writing some filtering code in views.py.
- We are able to sort Event based on its Popularity count to display the most popular events. For example, an event with the largest popularity, and then the second largest, the third events...will show up on the "Popular" carousel. We temporarily set the popular carousel to have 10 most saved events for now.
- We also incorporate Ajax and Django REST framework to implement the "Save" button of any particular Event: If an user clicks "Save" on an event, the event will "know" the current logged in user and changes the database accordingly. In addition, the event's popularity count will also increase by 1.
- We also implement django-money to add support for Money fields in my models and form.
- We programmed the function for the user to be able to reset password. In particular, when the user enter their re-set email, the app will send them a link to that email, the user can follow the link, enter new password and log in as usual.

- In addition, the user can sign up using our app instead of the Django admin app. We connected our Profile model and Django's User model, created a new Sign-up template and views.
- On the search bar, if you enter any search keyword, we coded the function and created a new search template so that your search results (that match the keyword) will show up on that search result page.
- Please see URL above for all our team choice URLs accordingly.

6.Conclusion:

Completing this project helped us learn a lot about web development:

- We learnt about creating a stub UI shell with CSS, HTML and Javascript and implementing back-end area including setting up models, user authentication, Django admin site. We also learnt how to connect the front-end and back-end using Django.
- For app-related functions, we had to learn how to create a search function and learned to implement functions that allowed a registered user to create events, save events, and have those events displayed on their profile.
- For some additional team choice, we had to do a lot of research outside work and create a search function, filter and display events based on tags and display events based on their popularity (how many users saved the event) from events we populated for our database. At the beginning of our project, one challenge was implementing the popularity function as we had made it a foreign key, during the project implementing the search function was a difficulty for us as well.
- Another difficulty is that many of us had not used Github prior to taking this course and it caused many issues throughout the semester as we ran into conflicts when merging the data as well as issues where one member's work would get deleted when another member pushed their code, however we were able to resolve this problem.
- However, this project has helped us learn to work in team more effectively. We are still inexperienced which leads to a lot of team problems such as communication, dividing tasks, lack of technical knowledge..etc. However, we definitely learned a lot and hope to gain more experience in the future.