Final Project Report: CSE 541 Database Systems - I

PennConnect
A social network for connecting PSU students across the globe

By:

Parth Vidyadhar Natu (pzn5087)

Namitha Nambiar (nmn5265)

Soundarya Nurani Sundareswara (sxn5310)

Rajal Nivargi (rfn5089)

TABLE OF CONTENTS

INTRODUCTION

Social Networks such as Facebook, Reddit, Instagram are some of the most popular web applications used today. Other than the secondary functionalities that most social networks demonstrate, the most important one amongst them is the ability to create, remove and maintain relationships between the users on the platform.

We feel the most elegant way to represent these relations would be to use a graph database – while using a relational database to store data for the secondary functionalities of the web application. This means a combination of using Neo4j and MySQL for the deployment.

PROJECT OBJECTIVES & MOTIVATION

Social networks are a unique problem from the database side of things. They require us to use hybrid approach to store data — involving usage of a graph database for storing the relationships between the users and a general-purpose relational database for storing data about posts, comments and the user profile. It is imperative that the database is designed in a way that allows for efficient retrieval of data and relationships — especially in the generation of a dynamic user-specific news feed.

A graph database is an online database management system with Create, Read, Update and Delete (CRUD) operations working on a graph data model. Graph databases are generally built for use with online transaction processing systems. Unlike other databases, relationships take priority in graph databases. This means the application doesn't have to infer data connections using foreign keys. By assembling the simple abstractions of nodes and relationships into connected structures, graph databases enable us to build sophisticated models that map closely to our problem domain.

Thus, our project objective is as follows:

"To explore usage of graph databases and relational databases as the backend in the implementation of social networks and create a functional social network web application."

FUNCTIONALITIES

To explore usage of graph databases and relational databases as the backend in the implementation of social networks and create a functional social network web application.

The social network application should have the following functionalities:

- Sign in and Sign out sessions
- A user-specific news feed
- UI for profile, groups and events
- The ability to create, view, edit and delete comments
- Authentication for users
- Schedule events
- Create, view, edit and delete users, posts, events
- Friend and unfriend users
- Search for users, groups and events
- Upvote and downvote posts
- Analytics for administration

DATA SOURCE

Relationship Data: http://socialcomputing.asu.edu/datasets/Digg

- Number of nodes: 116893
    - The nodes represent the users that form a part of the social network
- Number of edges: 7261524
    - The edges represent the relationships between the users

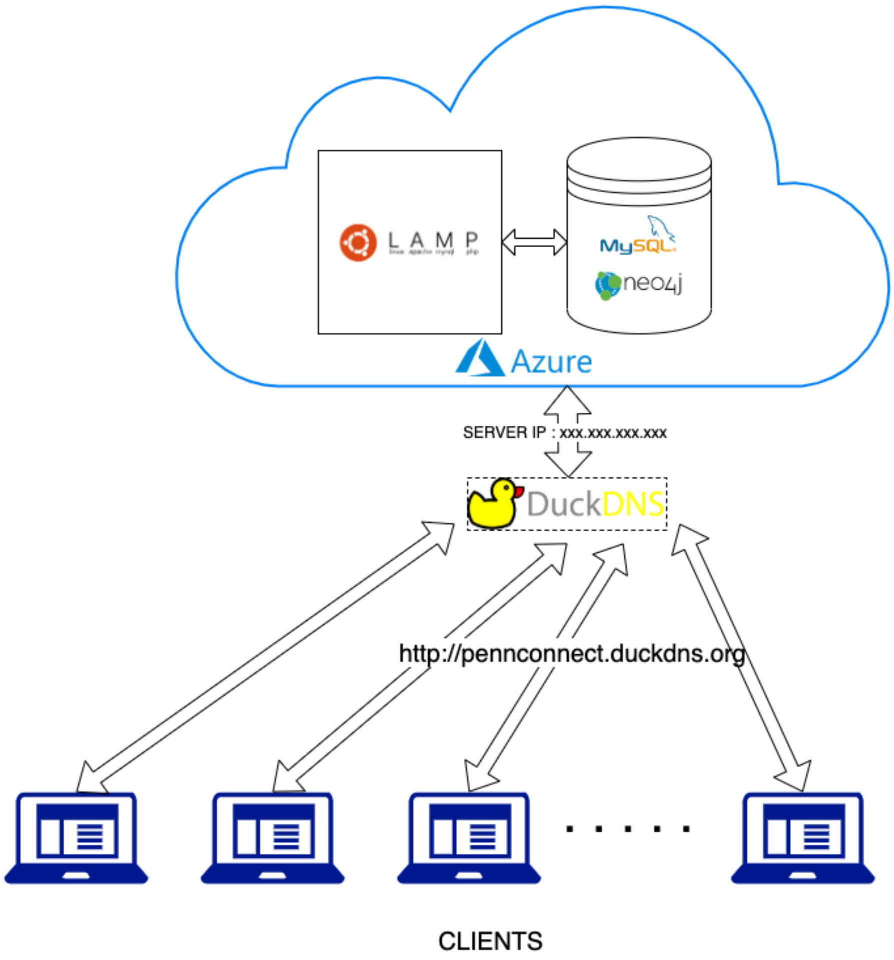
User Data: https://randomuser.me/

- This website allows us to generate random user data with various attributes that include but are not limited to: Name, E-mail ID, Gender, Profile Picture

Posts Data: https://data.world/martinchek/2012-2016-facebook-posts

- This contains post data from Facebook from 2012 to 2016 from various news sources.

ARCHITECTURE DIAGRAM



The architecture of the system can be divided into the "Server-Side" and "Client-Side" Components. The server side consists of the LAMP Software stack which includes the Linux OS, Apache web server, MySQL RDBMS and PHP as the scripting language. In addition to these, we have also integrated Neo4j into the server side of the application's architecture. Neo4j is the graph database that is used to store the relationship between users to indicate whether they are friends or not. DuckDNS is a Dynamic DNS service that provides a free domain for the machine IP. The client side of the application consists of web browsers that interact with the Apache Web Server hosted on the backend to complete requests made by application users.

TECHNOLOGIES USED

Neo4j:

Neo4j is one of the most popular graph database management systems. It uses nodes, edges and attributes to store data. The nodes and edges can have any number of attributes.

Motivation to use Neo4j:

- Easy to depict users and relationships with "friends" using graphs. Each user is a node (vertex) and the edges correspond to relationships with other users.
- Highly scalable – a social networks can have millions of users and each user can have many friends. Therefore, we require a database management system that can easily scale up to the required numbers without there being any compromise in the performance. Neo4j provides for fast reads/writes, and storage of high volumes of data without effecting the query processing speed and data integrity.
- Easy data retrieval − With Neo4j, it is possible to not only represent but to also easily retrieve (traverse/navigate) connected data faster when compared to other databases.

Neo4j uses Cypher as a query language. It is a powerful and declarative query language. Cypher is easy to learn. The user can easily create and retrieve relationships between nodes without having to use highly complex queries.

MySQL:

MySQL is a widely used open-source relational database management system. It uses tables with rows and columns to store data. Each row can have multiple attributes (columns).

Motivation to use MySQL:

It is easy to represent data in the form of tables. E.g. user profile, posts made by users, events that users have opted to attend.

- Typically, an application such as a social network has millions of users. Profile information of each user can be easily represented using tables, with "name", "age", "email ID", etc. being just some of the relevant attributes. Apart from these, posts made by each user must be linked back to that users' profile ID, in addition to storing the various comments on each of those posts. All in all, the application requires a lot of data to be stored in tabular format. We require an efficient database management system that can parse through the various tables and glean the necessary information with as much robustness as possible. MySQL is a relational database management system that can work fast and efficiently with huge datasets, therefore, perfectly fitting the needs of PennConnect.
- Referential Integrity: This is a very important feature that can will come in handy when maintaining a track of posts made by each user, where the profile ID of each user would be used as a foreign key. Since we cannot have a post without a corresponding profile ID of

the user that created it, this application requirement can be implemented using the referential integrity feature in MySQL.

MySQL uses Structured Query Language (SQL) as its query language. SQL is a fast and efficient query language which is more importantly, easy to learn and understand. SQL provides a simple way of manipulating data which is beneficial when working with large datasets.

Front-end web application framework:

Bootstrap:

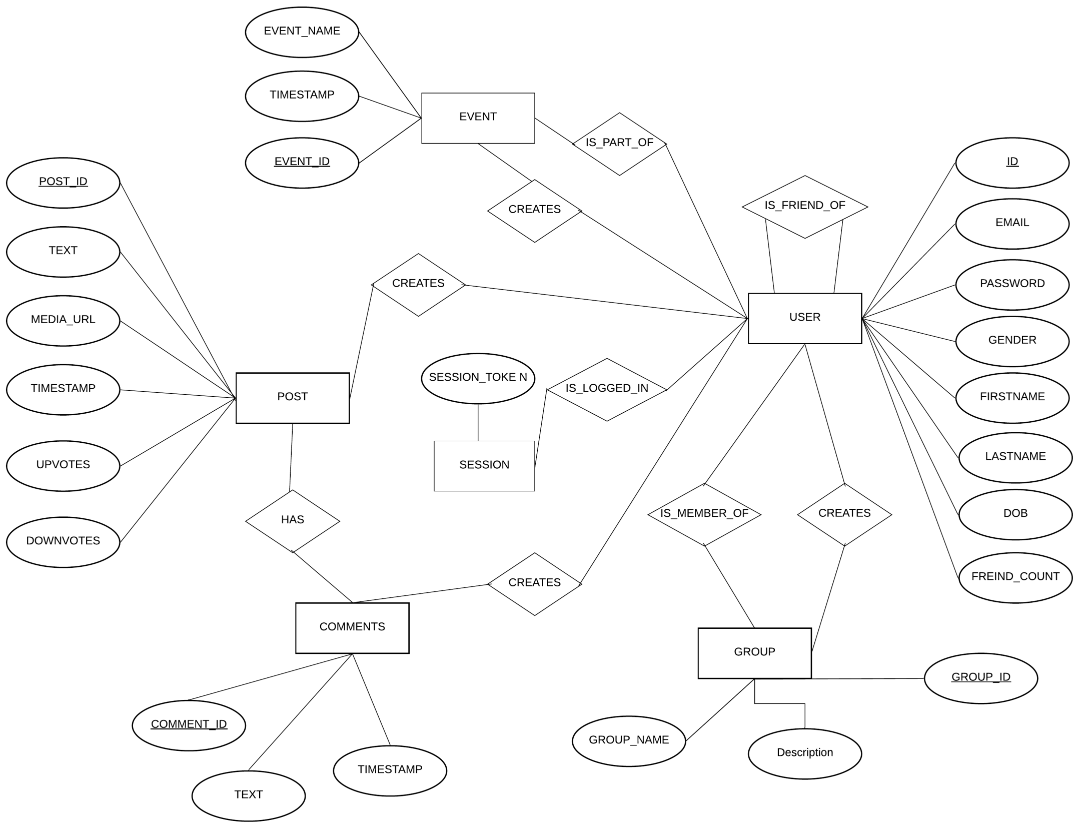
The UI for PennConnect is built using the Bootstrap which is an open source front end framework based on HTML, CSS and JavaScript.

Motivation for using Bootstrap:

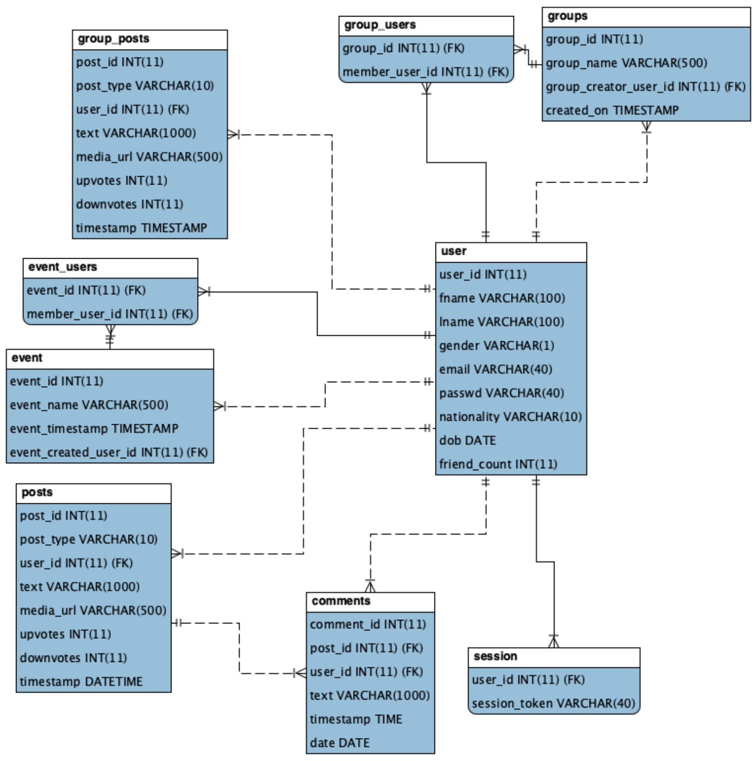
- Lightweight and customizable
- Responsive Web design
- Can include several JavaScript plugins using jQuery

DATABASE DESIGN

E-R Diagram:



Schema:



▫ **<u>Description:</u>**

Table Name	Description
user	This is the table containing data about all users. • user_id (PRIMARY KEY): Unique ID for each user. • fname, lname: Name of the user • gender: User's gender • email: User's email ID. Used during login. • passwd: Hashed password for the user. • nationality: Two letter identifier for the nationality of the user. • dob: User's date of birth • friend_count: Count of all the user's friends. This field is updated from Neo4j.
posts	Contains all post data. • post_id (PRIMARY KEY): Unique ID for each post. • post_type: Can be 'link', 'text' or 'photo' • user_id: User ID for the creator of the post. Referenced as a foreign key in 'user' table. • text: Text content of the post • media_url: URL of the media content in the post. • upvotes, downvotes: number of positive and negative votes for the post. • timestamp: time at which the post was created.
group_posts	Contains data for posts in a group. • post_id (PRIMARY KEY): Unique ID for each post. • post_type: Can be 'link', 'text' or 'photo' • user_id: User ID for the creator of the post. Referenced as a foreign key in 'user' table. • text: Text content of the post • media_url: URL of the media content in the post. • upvotes, downvotes: number of positive and negative votes for the post. • timestamp: time at which the post was created. • group_id: References to which group the post was posted.

event_users	Lists the user's signed up for an event. • event_id: unique ID for the event. Referenced as a foreign key in 'event' table. • member_user_id: ID for the member user. Referenced as a foreign key in 'user' table.
session	Contains the mapping for user to the session. • session_token (PRIMARY KEY): Unique session token generated using MySQLs UUID() method. • user_id: User ID of the logged in user. Referenced as a foreign key in 'user' table.
event	Contains data about an event. • event_id (PRIMARY KEY): Unique ID for an event. • event_name: Name for the event. • event_timestamp: Date and time for the event. • event_created_user_id: User ID for the user who created the event. Referenced as a foreign key in 'user' table.
comments	Contains data about the comments for a post. • comment_id (PRIMARY KEY): Unique ID for the comment. • post_id: Unique ID for each post. Referenced as a foreign key in 'posts' table. • user_id: User ID for the user who created the comment. Referenced as a foreign key in 'user' table. • text: Text for the comment. • timestamp: Date and time for the comment.
group_users	Lists the user's signed up for a group. • group_id: unique ID for the event. Referenced as a foreign key in 'groups' table. • member_user_id: ID for the member user. Referenced as a foreign key in 'user' table.
groups	Lists the user's signed up for a group. • group_id (PRIMARY KEY): Unique ID for an event. • group_name: Name for the event. • group_creator_user_id: Date and time for the event. • created_on: Date and time for creation of the group.

▫

BACKEND DESIGN

LAMP Stack:

We have used a Linux-based server comprising of four software components to form our software stack to support PennConnect. They are:

- Operating system – Linux
- Web Server - Apache
- Object-oriented scripting language - PHP
- Relational Database Management System – MySQL

Linux doesn't mandatorily require any specific distribution in order to install the different components of the software stack on a server. The installation of packages is very easy in a Linux environment which makes it very easy to work on. The Apache webserver is responsible for handling the web page requests that come from a web browser. Apache is open source and has a modular design. It provides many extensions that makes it easy to bind the server with the rest of the software stack components. MySQL is one of the most popularly used relational database management systems. MySQL perfectly suites the storage requirements needed by PennConnect in order to efficiently store information regarding its users and their activities. PHP sits right on top of our protocol stack and is the server-side scripting language that we have chosen. PHP is compatible with most web-development frame works and makes it easy to write the server-side APIs required to service the requests that come in from our application from the client side.

API Framework:

We have used REST APIs to interface the frontend with the backend databases. Each API is either a GET or a POST call to the backend to write/retrieve data from the database. The API Framework has been written in such a way that requests of any HTTP Method types can be supported by the same framework. The HTTP request received by the backend server is parsed by the PHP file "api-gateway.php". The request is broken down into a token level in order to analyse the request and its method. If the received request is in the accepted format, the query is sent to be processed. The "process-requests.php" file gets the request to process the API call. Depending on the HTTP method type, we further categorize the actions that are to be taken. Once the action to be taken has been determined by parsing the API, the necessary functions are invoked from the library file "common-functions.php". The requested is completed, and the return message is sent back to the client side.

A concise list of APIs and its method type can be looked up in the Appendix.

FRONTEND DESIGN

The application is modularized into pages (Example, Login, Newsfeed, Profile). Each page contains an HTML, a CSS and a JS file. The webpage is encoded using different bootstrap components in the HTML file while the presentation specifications are enabled using the CSS file. The business logic which makes the application interactive, is present in the JavaScript file. JavaScript supports event driven, functional and imperative programming.

AJAX – Asynchronous JavaScript and XML:

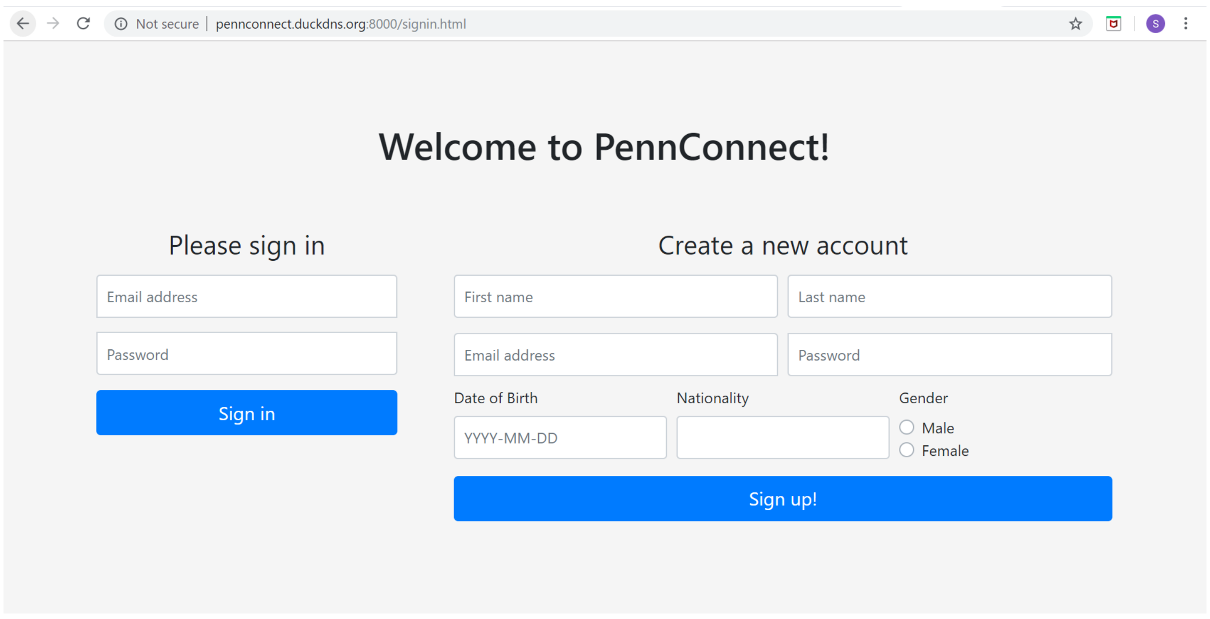
Bootstrap enables the use of jQuery AJAX methods to interact with the backend. With AJAX calls, data of different types like text, XML or JSON can be fetched from a server which could then be loaded on the webpage asynchronously.

PennConnect uses the AJAX GET and POST methods to request data from the server.

Pages & functionalities:

Login page – The user can login to PennConnect by creating an account or using the user's email address and password.

Screenshot:



Newsfeed page – The logged in user could view his friends' posts. The posts paginate on scrolling the page. On initial load, the first ten posts are displayed. On scrolling further, the next ten are displayed. This is done to improve the performance during the initial load of the page.

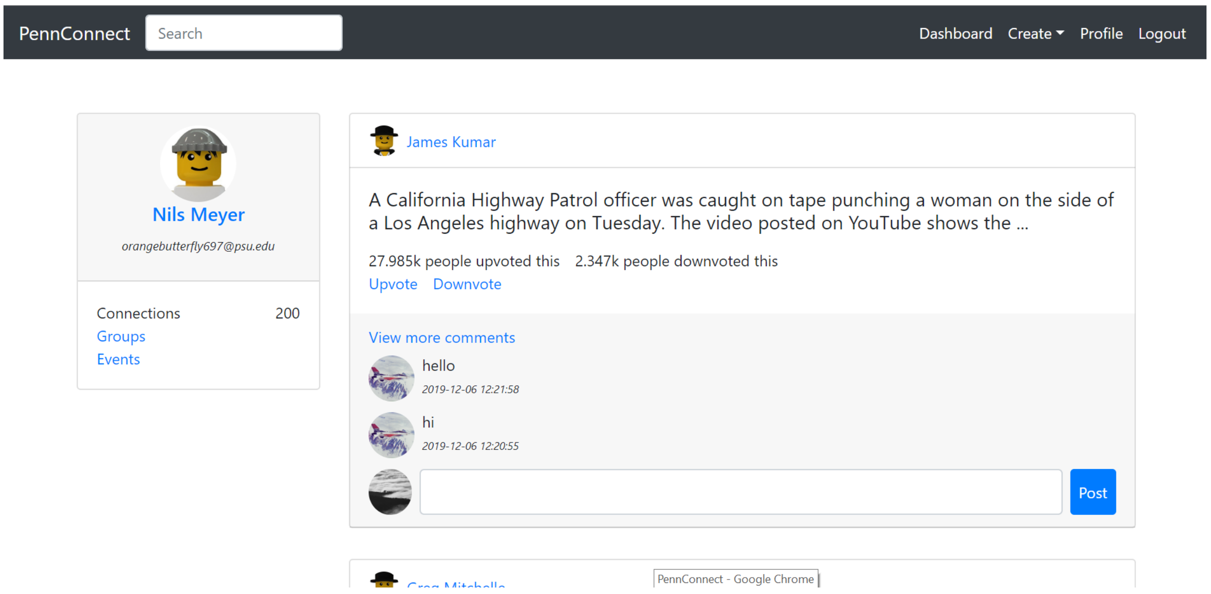
Each post on the page displays information such as the number of upvotes, number of downvotes and the number of comments. The logged-in user could upvote, downvote and comment on each post.

Additionally, the newsfeed page contains a navigation bar which supports the following functionalities:

1. Creation of a post
2. Creation of a group
3. Creation of an event
4. Search for user profiles, groups and events
5. Navigation to user profile page
6. Sign out from the application

Screenshots:

1. **Newsfeed page**



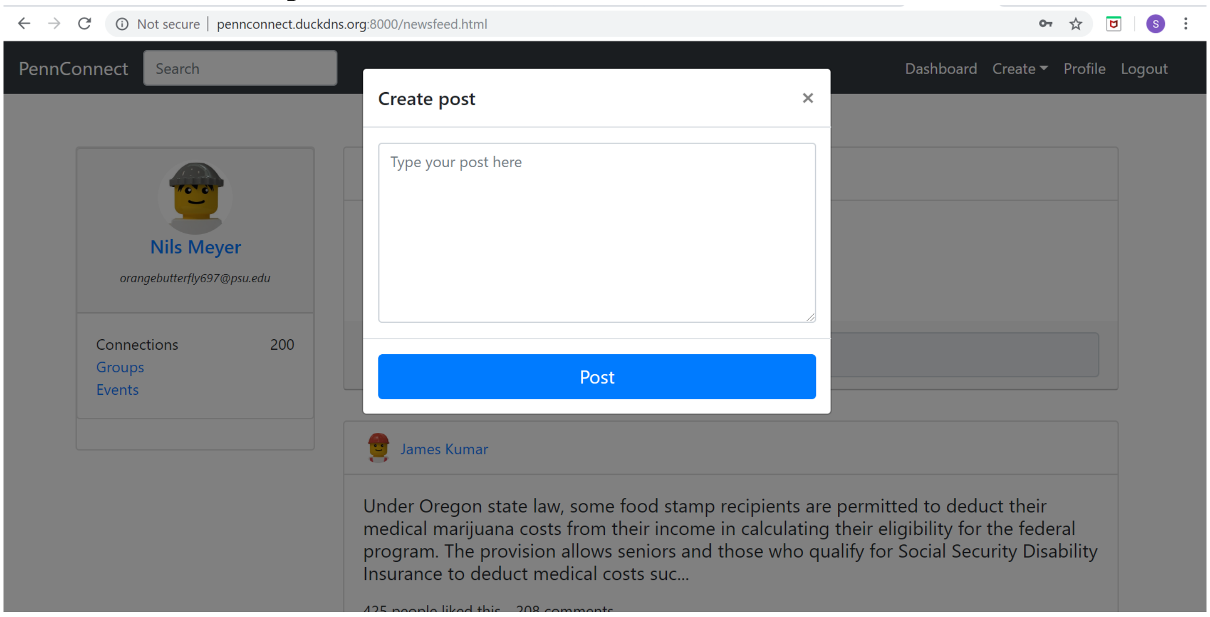
A California Highway Patrol officer was caught on tape punching a woman on the side of a Los Angeles highway on Tuesday. The video posted on YouTube shows the ...

27.985k people upvoted this 2.347k people downvoted this

hello
2019-12-06 12:21:58

hi
2019-12-06 12:20:55

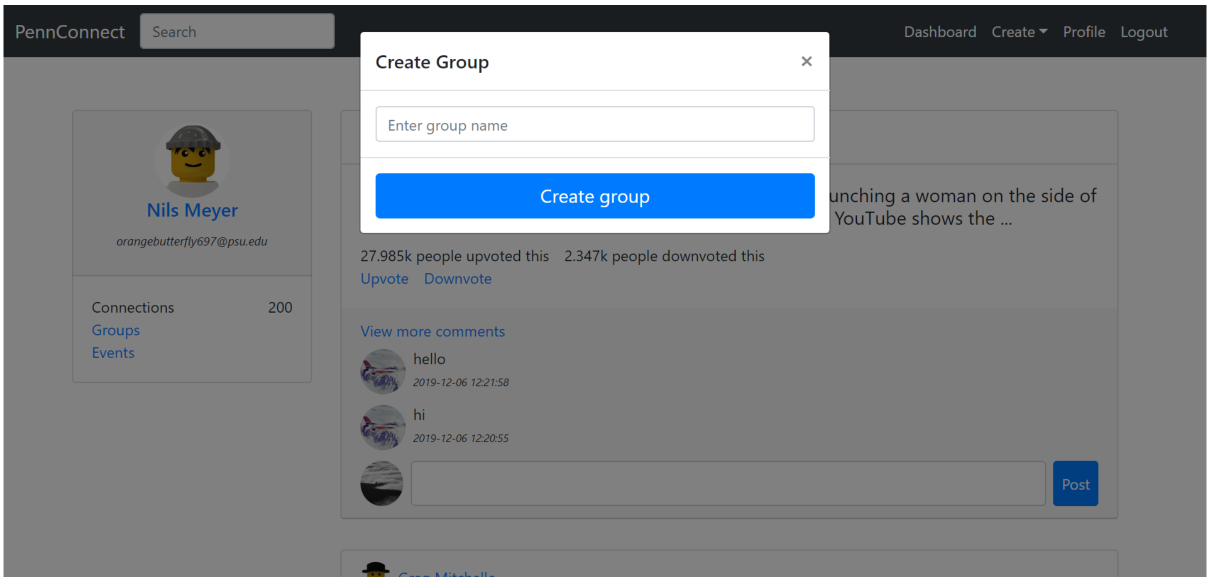
2. Creation of a post



Not secure | pennconnect.duckdns.org:8000/newsfeed.html

Under Oregon state law, some food stamp recipients are permitted to deduct their medical marijuana costs from their income in calculating their eligibility for the federal program. The provision allows seniors and those who qualify for Social Security Disability Insurance to deduct medical costs suc...

3. Creation of a group



unching a woman on the side of YouTube shows the ...

27.985k people upvoted this 2.347k people downvoted this

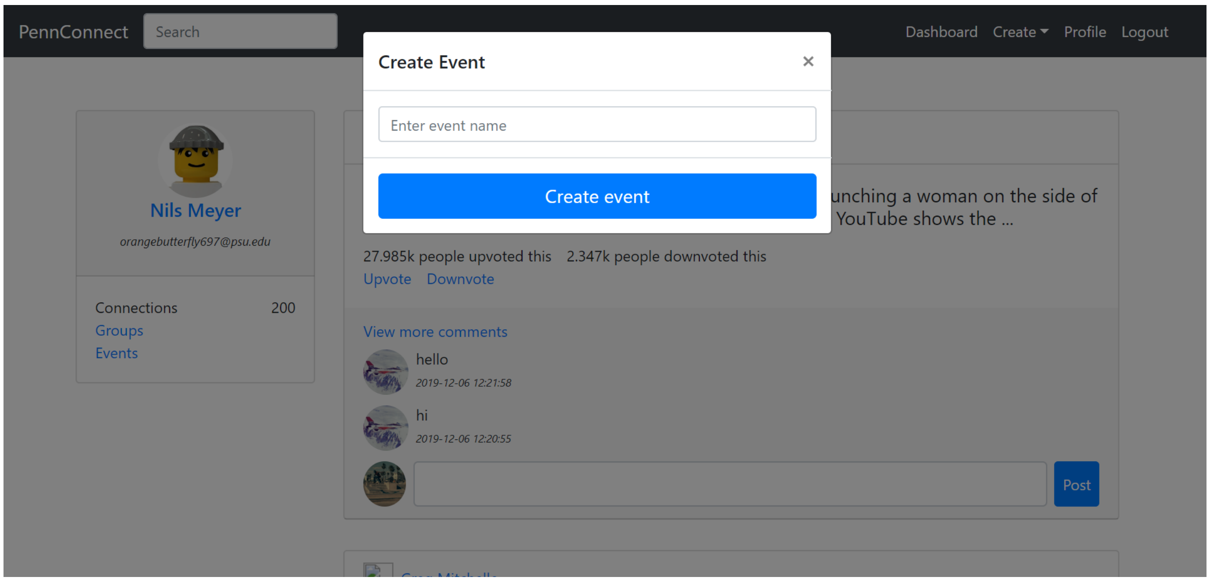
Upvote Downvote

View more comments

hello
2019-12-06 12:21:58

hi
2019-12-06 12:20:55

4. Creation of an event


PennConnect Search Dashboard Create Profile Logout

Create Event ×

Enter event name

Create event

Nils Meyer

orangebutterfly697@psu.edu

Connections 200

Groups

Events

unching a woman on the side of YouTube shows the ...

27.985k people upvoted this 2.347k people downvoted this

Upvote Downvote

View more comments

hello

2019-12-06 12:21:58

hi

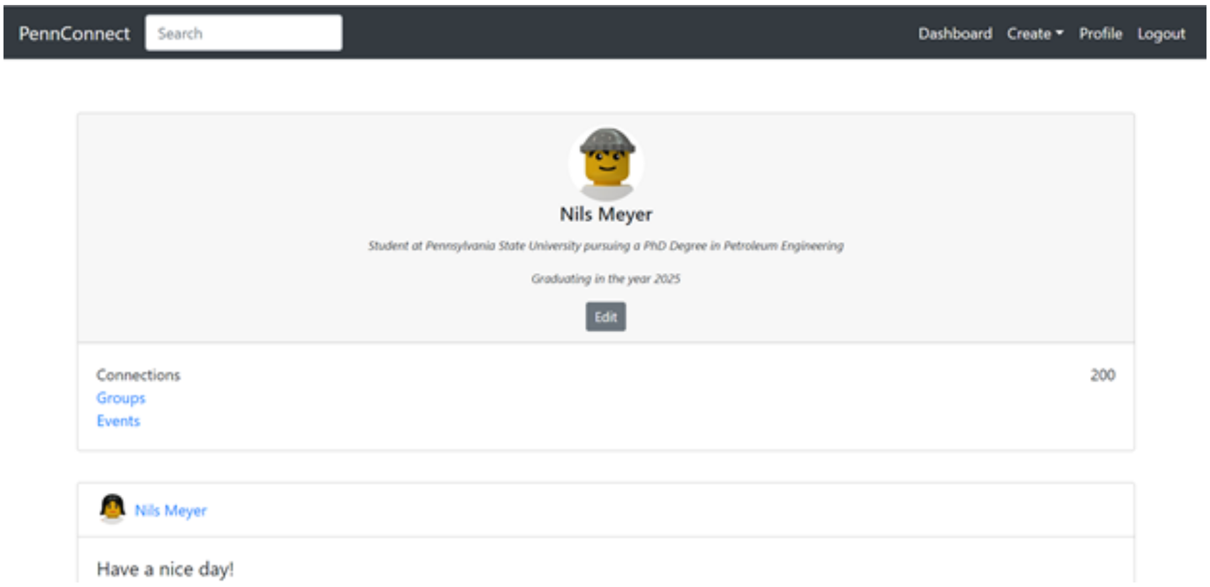
2019-12-06 12:20:55

Post

Profile page – The user could view his profile details and his own posts. The user could also edit his profile details

Screenshots:

1. Profile page


PennConnect Search Dashboard Create Profile Logout

Nils Meyer

Student at Pennsylvania State University pursuing a PhD Degree in Petroleum Engineering

Graduating in the year 2025

Edit

Connections 200

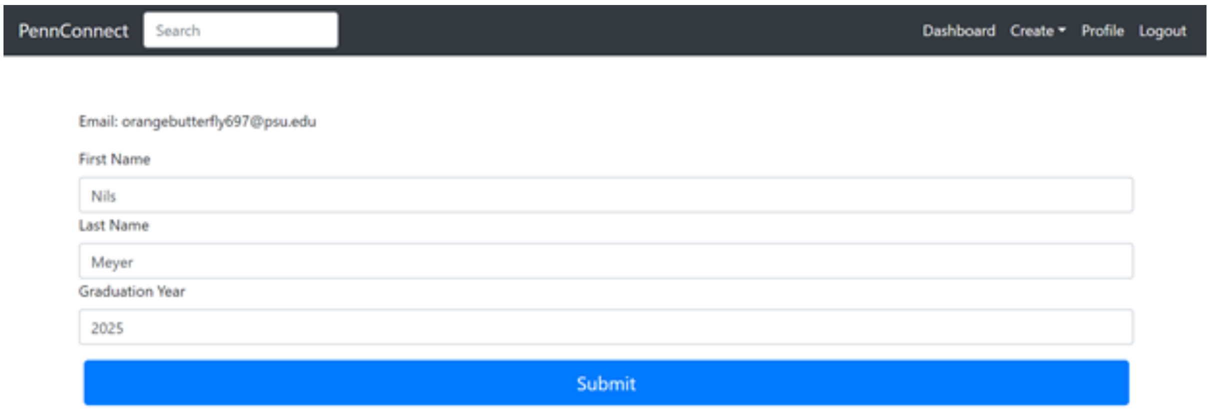
Groups

Events

Nils Meyer

Have a nice day!

2. Edit Profile



Email: orangebutterfly697@psu.edu

First Name

Nils

Last Name

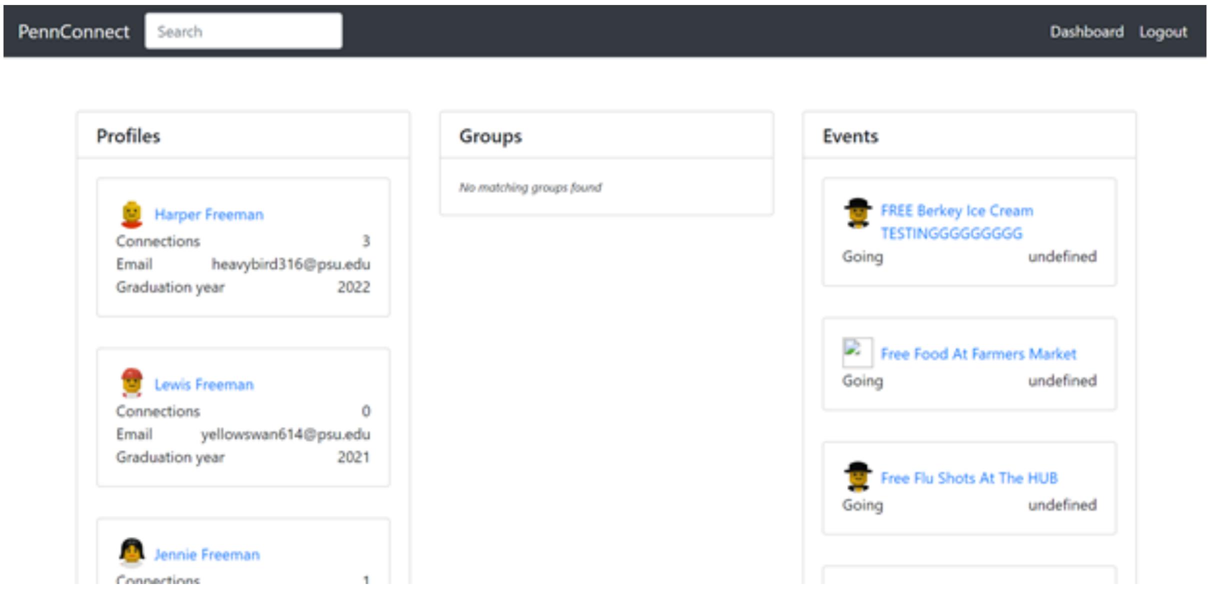
Meyer

Graduation Year

2025

Search page – In the search bar, the user could enter a string and search for a user profile, group or event

Screenshot:



Profiles

Harper Freeman
Connections 3
Email heavybird316@psu.edu
Graduation year 2022

Lewis Freeman
Connections 0
Email yellowswan614@psu.edu
Graduation year 2021

Jennie Freeman
Connections 1

Groups

No matching groups found

Events

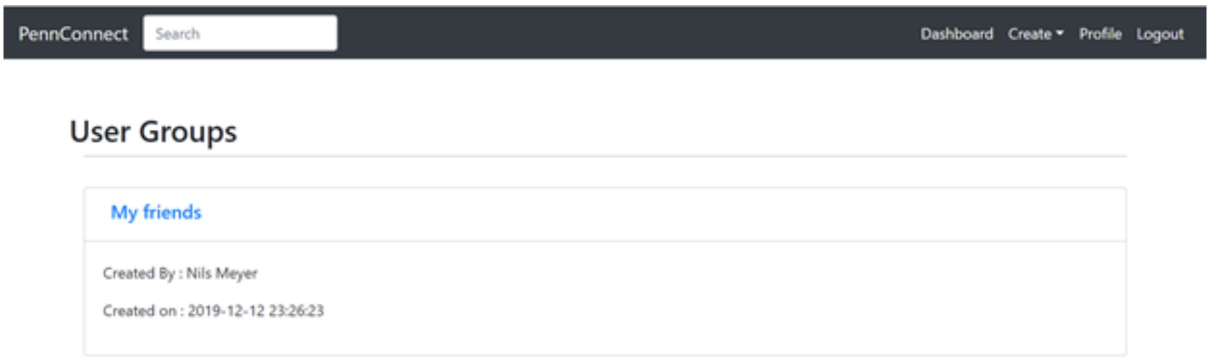
FREE Berkey Ice Cream TESTINGGGGGGGGG
Going undefined

Free Food At Farmers Market
Going undefined

Free Flu Shots At The HUB
Going undefined

User group page – From the profile page, the user could navigate to see a list of groups he is part of.

Screenshot:



User Groups

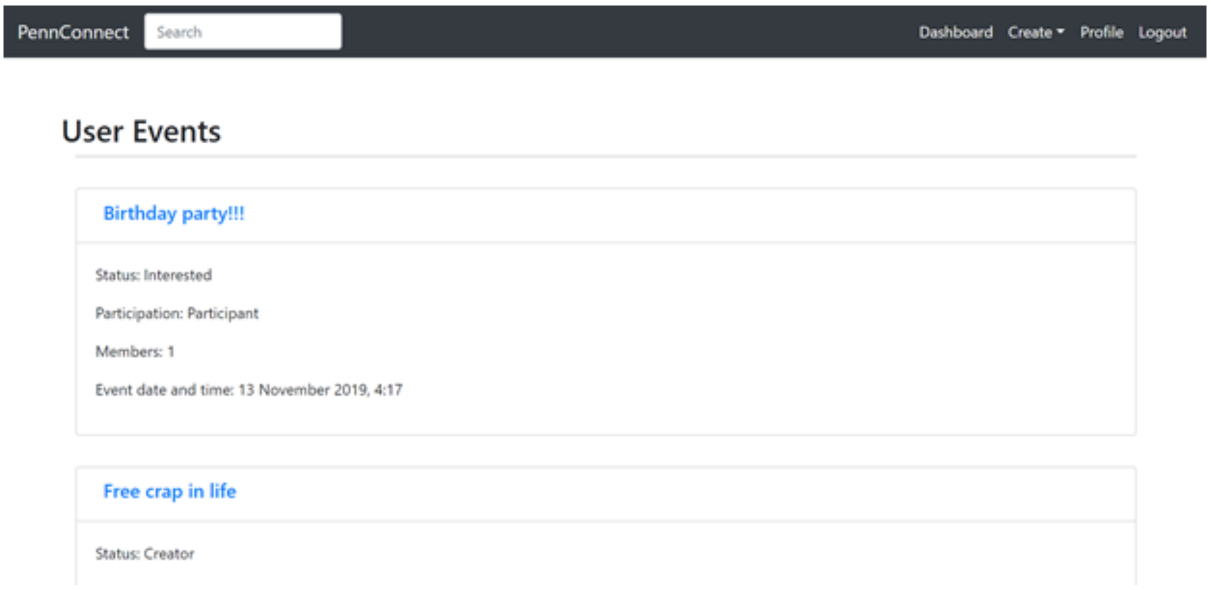
My friends

Created By : Nils Meyer

Created on : 2019-12-12 23:26:23

User event page - From the profile page, the user could navigate to see a list of events he is part of.

Screenshot:



User Events

Birthday party!!!

Status: Interested

Participation: Participant

Members: 1

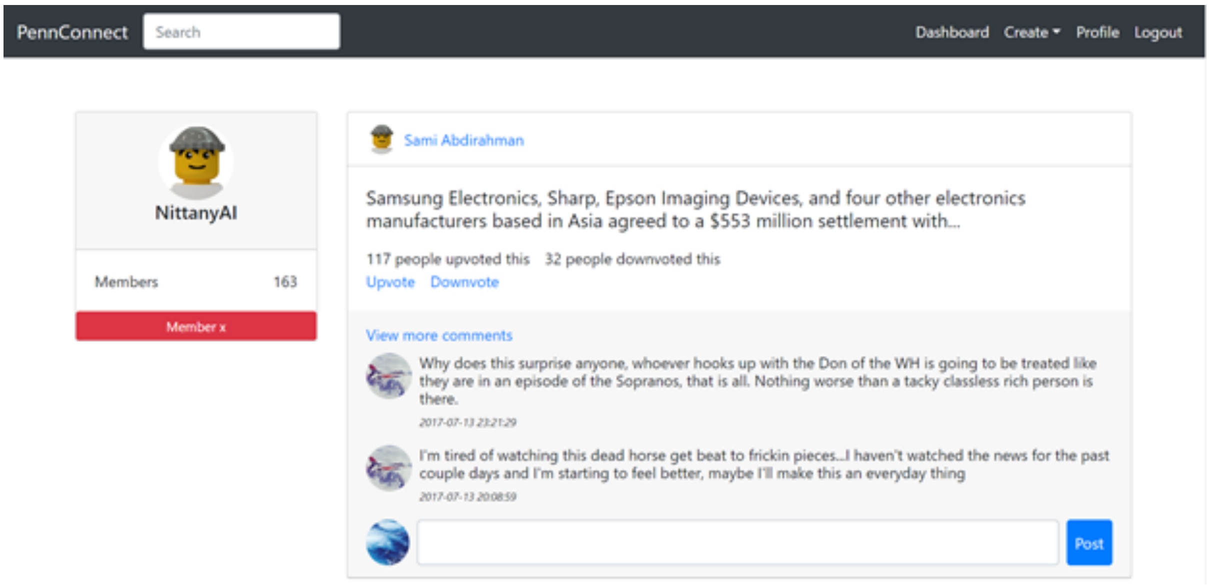
Event date and time: 13 November 2019, 4:17

Free crap in life

Status: Creator

Group page - The logged-in user could see the detailed information of different groups and their posts. He could also check if he is a member.

Screenshot:



NittanyAI

Members 163

Member x

Sami Abdirahman

Samsung Electronics, Sharp, Epson Imaging Devices, and four other electronics manufacturers based in Asia agreed to a $553 million settlement with...

117 people upvoted this 32 people downvoted this

Upvote Downvote

View more comments

Why does this surprise anyone, whoever hooks up with the Don of the WH is going to be treated like they are in an episode of the Sopranos, that is all. Nothing worse than a tacky classless rich person is there.

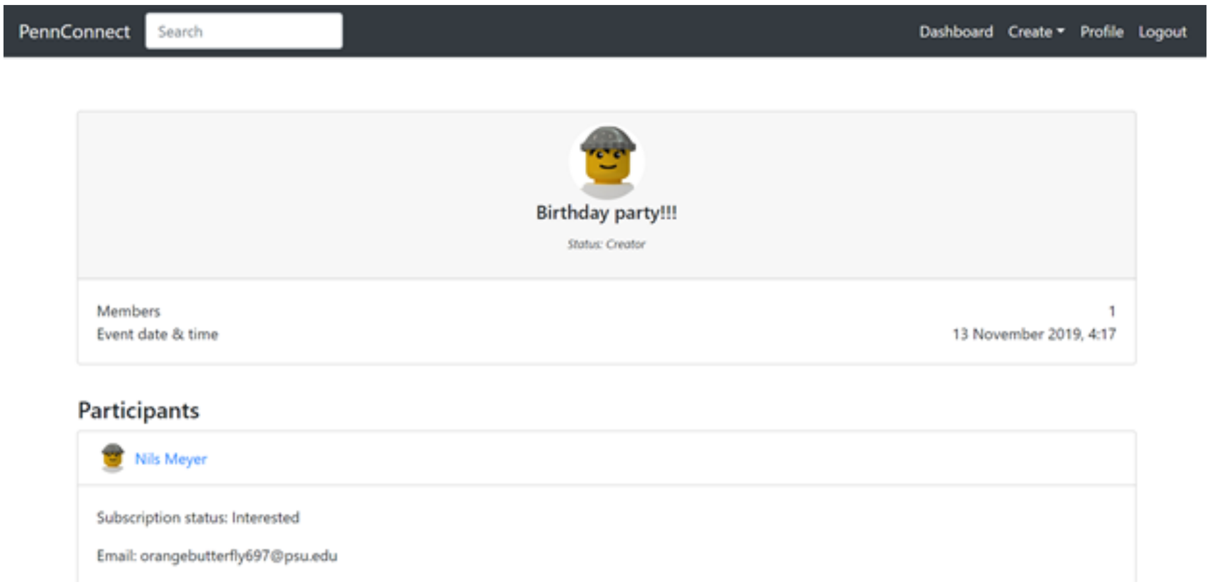
2017-07-13 23:21:29

I'm tired of watching this dead horse get beat to frickin pieces...I haven't watched the news for the past couple days and I'm starting to feel better, maybe I'll make this an everyday thing

2017-07-13 20:08:59

Event page - The logged-in user could see the detailed information of different events and their participants. He could also check his subscription status to the event.

Screenshot:



Birthday party!!!

Status: Creator

Members 1

Event date & time 13 November 2019, 4:17

Participants

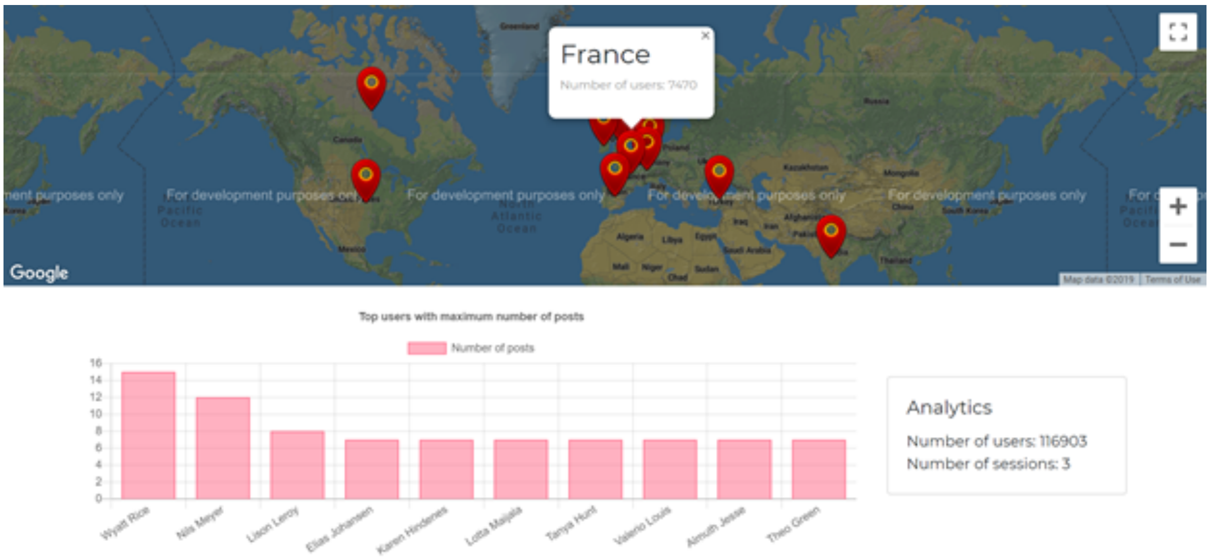
Nils Meyer

Subscription status: Interested

Email: orangebutterfly697@psu.edu

Analytics page – The analytics page is used by the administrator of the application to learn about the users and the data in the application. It provides some analytics based on the user's data.

Screenshot:



France

Number of users: 7470

Analytics

Number of users: 116903

Number of sessions: 3

DETAILED IMPLEMENTATION OF FUNCTIONALITIES

- **Sign up:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/

The sign-up API is an HTTP POST method. When a user signs up for an account with their user details, an entry is created in the user table and a unique user-id is assigned to this new user. The password that the user would like to use for the account is hashed before it is stored in the database. In addition, an entry is also created in the Neo4j database for this user.

- **Sign in:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/login

The sign-in API is an HTTP POST method. When a user wants to sign-in, the user will enter their email ID and password. A session token or cookie is generated when the user logs in and will be returned to the client side. This session token is also stored in the backend session table along with the user's unique user ID. From this point on, any requests from the currently logged in user, should contain the session token or the cookie in order to complete user specific requests.

- **Sign out:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/logout

When a user logs out, the above indicated API, which is an HTTP POST method, is invoked. The user's session is then deleted from the sessions table in the backend. A success is then indicated back to the client-side when the user's current session has been cleared from the database.

- **Create new post:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/post

As the name suggests, this is an HTTP POST method which is called whenever a user creates a new post. The text content, time stamp, date, any uploaded media, etc., relating to the post must be entered into the database. Each time a new post is created by a user, a new entry is created in the posts table and a unique post ID is generated for it. Each of these posts will be associated with the user ID of the user that created it, which becomes the foreign key referenced from the user table.

- **Get post IDs:**

API URL:http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/newsfeed

This HTTP POST method is used to retrieve all the associated post IDs of the posts that should be displayed on a user's newsfeed page. When this API is called, we match the session token with the user's user ID and trigger a Neo4j call to find all the user's friends. Then we retrieve the posts IDs of posts made by the friends and return a list of IDs to the client side.

- **Fetch a user’s posts:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/getpost

This API is used to return the contents of posts that are associated with a list of POST IDs requested by the client side. A query is executed to select the contents of posts associated with specific post IDs and is then returned to the client side.

- **Get profile details:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/current-user/

This is an HTTP Get method which is used to retrieve a user’s profile information from the database. The cookie that is set in the HTTP request is matched against the values in the session table in the backend database to find the user’s user ID. Once the user ID is found, the specific user’s details are retrieved from the user table and returned to the client side.

- **Search for friend’s profile details:**

API URL: http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/this-user/{id}

This is an HTTP GET call that can be used to retrieve the details of any user that is being searched for. The user ID of the user that is being searched for, will be contained in this API, and a database lookup will be done in order to retrieve all the details that is being requested for.

- **Create Comment:**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/create_comment/

This is an HTTP post call that can be used to create a comment for a specific post ID. The post ID and comment details are passed to the API in JSON format and the user ID of the user creating the comment is retrieved based on the cookie.

- **Retrieve Comments**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/getcomment/

This is an HTTP POST method that is used to retrieve the comments based on a comment ID. Here the input that is passed to the API is the post ID of the post to which the comment belongs to. The comment is then matched against the corresponding post ID in the comments table and returned to the front-end.

- **Edit Post**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/editpost/

This is an HTTP POST call that is used to edit a post made by a user. Here, the Post ID, the columns that are to be edited, and their new values are to be passed in JSON format to the back end. The columns and values are matched, and a check is done to ensure that there is not mismatch between the columns provided and their values. The details of the post are then updated in the posts table of the pennconnect database

- **Edit User Profile**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/edit-profile/

This is an HTTP POST call that is used to edit a user's profile. Obviously, the user can edit only his/her profile. Here, the profile ID is obtained using the cookie that will be passed in the API call. The columns that are to be edited, and their new values are to be passed in JSON format to the back end. The columns and values are matched, and a check is done to ensure that there is not mismatch between the columns provided and their values. The details of the profile are then updated in the user table of the pennconnect database.

- **Create Event**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/create_event/

This HTTP POST method is used to create an event. The user needs to enter the event details that are then formatted in JSON and sent to the backend. The event is then populated in the events table and an event ID is assigned to this event. On successful creation, the newly created event details are returned to the front-end.

- **Delete Event**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/create_event/

This is an HTTP POST method. This API is used to delete an event. The event ID is passed to the backend, all data formatted in JSON. If the event ID matches an event in the events table of the pennconnect database, the entry is removed from the database.

- **Manage Event Subscription**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/manage_event_subscription/

This HTTP POST method is used to manage a user's subscription status to a particular event. This API can be used to mark the user's participation as either "Interested", "Going" or "Cancel". The response is JSON formatted. The event ID and the user's participation status is passed to the backend. The appropriate event is looked up in the events table, and the user's status is updated. The number of subscribers to the event is also accordingly controlled whenever this API call is made.

- **Retrieve User's Events**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_event_list_for_user/

This HTTP POST call is used to retrieve all the events that a user is associated with. The user can either be a creator/admin for an event, or a participant with his/her status as "Interested" or "Going". The list of events is retrieved by matching the user ID based on the cookie value that is contained in the API call.

- **Retrieve Event:**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_this_event/

This HTTP POST call is called in order to navigate to an event's home page. This API expects the event ID of the event that we are trying to load. The event is then looked up in the events table of the pennconnect database. The event details are then returned to the front-end.

- **Search**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/search/

This HTTP POST method is used to search for users, groups and events that match a search string that is entered by the user. This search string is passed to the backend and the user, groups and events table is scanned for potential matches. The details of the user, groups and events are the returned to the frontend.

- **Create Group**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/creategroup

Creates a group while taking the Group Name as a parameter

- **Edit Group Post**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/editgrouppost/

Takes the column names as an input parameter and a corresponding list of attributes to change in the database

- **Group Posts**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_group_post_data
http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_group_posts

Retrieves the list of group posts and uses pagination on client side to display the post dynamically

- **Analytics**

http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/analytics

Display geographical distribution of users and the corresponding counts, the number of active sessions and users and the distribution of the top 10 users with the highest post counts

TECHNICAL ISSUES

- **Maintaining polyglot persistence across the graph and relational database**

Neo4j is often used in combination with other data sources (polyglot persistence), such as relational or NoSQL databases. Data can be fetched from the different systems in order to be manipulated by an application and subsequently written back to Neo4j. Other solutions can involve loading of data from a few databases into Neo4j in order to efficiently do connected data analysis on the combined data set.

- **Efficient data retrieval**

While using graph databases to store data, we expect to increase the speed of data retrieval in the database. However, they do not create better relationships. Improved search is an advantage but only if the relationship was captured effectively in the first place. The integration of graph database and relational database may cause this discrepancy.

- **Security and Privacy in the graph database**

Graph databases allow visibility across the data – enabling organizations to share data and show how data is connected. Though it provides a better mechanism for solving complex problems, the data is exposed to privacy and security threats. For example, one's identity can be further laid bare if the graph supercomputer becomes the device of choice to further mine our social and financial transactions for purposes of surveillance, targeted advertising, and other overt exploits that tend to rob individuals of their privacy. In the real-life implementation of the social network model suggested, a secure way of safeguarding data becomes imperative.

DETAILED WEEKLY SCHEDULE

Week	Progress	Work done/Functionalities
16th and 17th September	Submission of Proposal	Requirement Analysis and technologies identified
17th – 27th September	E-R diagram	
27th – 4th October	User Sign-Up	A sign-up page for the user to set up his/her account
4th – 11th October	Login & Logout	A login in page that takes the user to his/her "wall", with a log-out option
11th – 18th October	Display User Profile	Required to load a user's profile page
18th to 25th October	Create new post & fetch a user's posts	This feature enables the user to create posts and load all the posts made by that user are displayed.
25th to 31st October	Newsfeed	Display newsfeed with pagi-nation.
1st to 7th November	UI for profiles, groups and events	A home page to display the details of a user's profile. An event or a group.
8th to 15th November	Create, view comments	Option to create and view comments for posts made by users and in groups.
16th to 22ndNovember	Friend and unfriend users. Upvote and downvote posts	Provide an option to mark another user as a friend or unfriend an existing friend. Option to upvote and down-vote for each post.
22nd to 29thNovember	Search for users, groups and events	Search bar in which the user can enter a search string and all the users, events and groups that match this search string are displayed.
29th to 6thDecember	Analytics for administration	In the admin view, we can have an overview for analyt-ics

APPENDIX

Functionality	API	**HTTP Method**
Sign in	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/login	POST
Sign up	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/user	POST
Get the list of friends' post IDs	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/newsfeed	POST
Get the post content based on the post IDs	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/getpost	POST
Create a post for the logged in user	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/post	POST
Get the current user	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/current-user/	GET
Get the user details defined by "id"	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/this-user/{id}	GET
Get User profile details based on "id"	http://pennconnect.duckdns.org:8000/user_profile.php?user_id={id}	GET
Logout	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/logout	POST
Create Comments	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/create_comment/	POST
Retrieve Comments	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/getcomment/	POST
Edit Post	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/editpost/	POST
Edit Profile	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/edit-profile/	POST
Create Event	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/create_event/	POST
Delete Event	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/create_event/	POST
Manage Event Subscription	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/manage_event_-subscription/	POST

Retrieve all of a User's Events	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_event_list_-for_user/	POST
Retrieve an Event	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_this_event/	POST
Search	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/search/	POST
Create Group	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/creategroup	POST
Edit Group Post	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/editgrouppost/	POST
Get Group Details	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/getgroupdetails	POST
Get Group Post Data	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_group_-post_data	POST
Get list of group posts	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/get_group_posts	POST
Analytics	http://pennconnect.duckdns.org:8000/api-gateway.php/penn-connect/analytics	POST

REFERENCES

- Bootstrap Documentation: https://getbootstrap.com/docs/4.3/getting-started/introduction/
- CSS Documentation: https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- JS Wiki : https://en.wikipedia.org/wiki/JavaScript
- Implementing AJAX in jQuery :https://www.w3schools.com/jquery/jquery_ajax_get_-post.asp
- Cookies in PHP : https://www.w3schools.com/php/php_cookies.asp
- PHP Session variables : https://www.w3schools.com/php/php_sessions.asp
- Cypher Documentation : https://neo4j.com/developer/cypher-query-language/