MUNSN

Architecture

Document Version: 1.0

Date: March 19th 2017

Prepared by: Andrew Way

Confirmation Email Server

Description

Send email to provided email address containing a unique link that connects the user to server for authentication.

Inputs

1. Account Validation: Authorization object {userid, authkey, expiry}

Processing

1. Form email text body containing userid and authentication link based on authkey

Outputs

1. External Email Server: Email containing link with authkey, sent to email address corresponding to userid

Account Validation

Description

Manage authentication keys in the authentication table by responding to users accessing an authentication link from their email. Add new authentication keys to the table when a user registers. Authenticate user by changing user authentication status in user table.

Inputs

1. User Management: User object (From db, Users.add)
2. Request: Validation request by clicking on email link

Processing

1. Add new user object into to auth collection
2. Check if key exists, and if so, if expired.

Outputs

1. Confirmation Email Server: {userid, authkey, expiry}
2. User Management: set auth true for a given user id (Users.update) if key exists and not expired

User Management

Description

Store users in a table, and manage that list of users by deleting, updating, finding, or creating entries.

Inputs

1. Request: Find user by ID
2. Request: Delete user
3. Request: Register new user
4. Request: Update existing user

Processing

1. Search for user corresponding to uid in user table
2. Delete the user corresponding to uid in user table
3. Add a new unconfirmed user to user table
4. Update user with entered parameters (password, email, visibility, address)

Outputs

1. Response: User object
2. Response: Success status
3. Response: Success Status
    a. Account Validation: call Auth.add to add the same user object to the temporary auth table

4. Response: Success Status

Group Management

Description

Store groups in a table, and manage that list of groups by deleting, updating, finding, or creating entries.

Inputs

1. Request: Find Group by ID
2. Request: Delete group
3. Request: Create new group
4. Request: Update Group properties

Processing

1. Search for group corresponding to gid in group table
2. Delete the group corresponding to gid in group table
3. Add a new group to the group table with the given properties
4. Update group with entered parameters (password, email, visibility, address)

Outputs

1. Response: Group object
2. Response: Success Status
3. Response: Success Status
4. Response: Success Status

Group Member Management

Description

Store and manage a list of users within a group by finding and returning group members,

Inputs

1. Request: Find Group Users (gid)
2. Request: Add group user (User object)
3. Request: Remove group user (gid,uid)

Processing

1. Find group user corresponding to gid in group member table
2. Add new entry into group member table created from user object
3. Delete group user corresponding to gid and uid from group

Outputs

1. Response: JSON object of group users
2. Response: Success status
3. Response: Success status

Group Requests

Description

Store requests to join a group, invitations to join group, send requests or invitations to front end when requested, and delete requests or invitations when cancelled, accepted, or rejected.

Inputs

1. Request: Add group request
2. Request: Find group invitations
3. Request: Find group requests
4. Request: Delete a request

Processing

1. Add join request to group join requests
2. Find the pending group invitations associated with a particular group ID
3. Find the join requests associated with a particular group ID
4. Find the join request associated with a request id
    a. Delete the request

Outputs

1. Response: Success status
2. Response: Group invitation JSON
3. Response: Group join request JSON
4. Response: Success status
    a. Group Management: Group.addMember if request was accepted

Friend Management

Description

Store and manage a list of users within a friends list belonging to a particular user

Inputs

1. Request: Get suggested friends list
2. Request: Add new friend
3. Request: Delete friend
4. Request: Get friends (fid)

Processing

1. Return list of friends
2. Add new entry into friend table created from user object
3. Delete user corresponding to fid and uid from friend table
4. Find the friend in the user's friend's list corresponding to fid

Outputs

1. JSON object of user's suggested friends
2. Success status
3. Success status
4. Friend object corresponding to fid

Friend Requests

Description

Store friend requests between users, send friend requests to front end when requested, and delete friend requests between users when cancelled, accepted, or rejected.

Inputs

1. Request: Add friend request
2. Request: Find friend requests the user sent
3. Request: Find friend requests received
4. Request: Delete a friend request

Processing

1. Add new friend sent friend request in the sender's request entry
    a. Add new received friend request in the recipients friend request row
2. Find the sent friend requests corresponding to the user
3. Find the received friend requests belonging to a particular user
4. Find the friend request corresponding to a friend id, uid
    a. Delete the friend request

Outputs

1. Success status
2. Sent friend requests
3. Received friend requests
4. Response:
    a. Friend Management: Call to add friend function if request accepted by recipient user

Comments

Description

Store and manage chains of paragraphs attached to a post. Comment histories are stored and associated with each comment.

Inputs

1. Request: Create new comment
2. Request: Update comment
3. Request: Delete comment

4. Request: Get comment

Processing

1. Append new content id to end of content ids for a particular post in the post table
    a. Create new comment in the Comment table (parent post ID as foreign key)
    b. Store text data into the history field
2. Add new comment body to the beginning of array in history field of the comment associated with the given comment id
3. Remove the comment from the comment table associated with the given comment id
4. Find comment in comment table associated with the given comment id

Outputs

1. Success Status
    a. Refresh post with the latest comments
2. Success Status
3. Success Status
    a. Send request to Post to delete the comment id from the parent post comment array
4. Comment

Posts

Description

Store, transfer, and modify post data.

Inputs

1. Request: Find Post by post ID
2. Request: Find Post by owning user id
3. Request: Delete post
4. Request: Update post
5. Request: Add group post
6. Request: Add timeline post

Processing

1. Use the given pid to find the post in the post table
2. Find collection of posts belonging to a particular user id
3. Find post corresponding to given post id and delete the entry from the post table
4. Find the post corresponding to given post id and update the entry from the post table
5. Add a new post into the table with a reference to the containing group
6. Add a new post into the table with a reference to the containing timeline

Outputs

1. Post
2. Post
3. Success status

4. Success status
5. Success status
6. Success Status

Polls

Description

Store, update, and provide information on polls such as poll questions, poll results, poll respondents.

Inputs

1. Request: Create poll
2. Request: Answer poll
3. Request: Get Poll

Processing

1. Create a new entry in the poll table constructed from the poll data
2. Search for the poll corresponding to the poll id and update the answers with the answer object
3. Search for the poll corresponding to the poll id

Outputs

1. Success status
2. Success status
    a. Poll is refreshed with the answer
3. Poll

Schedule

Description

Store and manage database of courses. Users can add new courses to the database via the API. Upon request, return the courses a user is enrolled in, in a JSON object that can be easily parsed and used for the calendar.

Inputs

1. Request: Update course
2. Request: Add Course
3. Request: Find course by course id
4. Request: Find a course
5. Request: Delete a course from the database
6. Request: Get Courses for a particular user

Processing

1. Access course entry in courses row corresponding to course id and modify properties

2. Add new entry in course table based on information from course object
3. Find the course corresponding to the course id in the course table
4. Find the course
5. Remove the course in the course table corresponding to the course id
6. Return all courses associated with a particular user id

Outputs

1. Success status
2. Success status
3. Course given by course id
4. Course
5. Success status
6. JSON object of courses

Instant Messenger

Description

Facilitate instant messaging between two users. Chat histories are stored and readable in the chat window.

Inputs

1. Request: Send message
2. Request: Open Chat

Processing

1. Read in message
    a. Store message in database
    b. Update recipients chat history
2. Return chat history (to some extent)

Outputs

1. If chat window is open, send the latest chat history (to some extend) to user
    a. If chat window is closed, send a notification to the recipient user
2. Segment of chat history

Lost & Found

Description

Store and manage posts containing information on lost and found objects in lost and found table. Provide item location information to users upon request.

Inputs

1. Request: Create new lost or found object
2. Request: Update lost or found object

3. Request: Return lost or found object

Processing

1. Transfer image to File Transfer module for storage
    a. Take in text information and store in a new entry in the Lost & Found table
    b. Save coordinates for object
2. Search of object given by id in the table and update the corresponding information
3. Search for object given by id in the table

Outputs

1. Success status
2. Success status
3. Lost or found object

File Transfer

Description

Handle client-server and server-client file transfer.

Inputs

1. Upload Resume
2. Upload group picture
3. Upload profile picture
4. Upload post picture
5. Request: Obtain resume
6. Request: Get post picture
7. Request: Get timeline picture
8. Request: Get group picture

Processing

1. Upload file to the resume database
2. Upload file to the picture database
3. Upload file to the picture database
4. Upload file to the picture database
5. Find picture associated with pid
6. Find picture associated with uid
7. Find picture associated with gid

Outputs

1. Success status
2. Success status
3. Success status
4. Success status
5. Resume

6. Post Picture
7. Timeline Picture
8. Group picture

User Interface

Description:
HTML and CSS files that comprise the collection of user-interactable input sources (buttons, text fields, hyperlinks)

Inputs

- Text
- Mouse clicks

Processing

- Modify interface according to client-side scripts following user interaction
- Perform API function calls

Outputs

- Modified user interface