Group Project – Data structure and Pseudo Code

Advanced Web ProgrammingCOMP1230

Maxim Bondarenko

Malik Iavari

2016

## Data Structure

**User**

* Id – all ID’s are auto incremented
* email
* password
* confirmedEmail – this will be set to TRUE when user confirms an email via a link that was sent to him/her

**Project**

* id
* title
* colour

**Tags**

* id
* title
* colour – CSS RGB code of the colour of the tag

**Categories**

* id
* title
* projectId

**User-to-Tag –** stores what Tags user can access

* id
* userId
* tagId

**Task-to-Tag**

* id
* taskId
* tagId

**Tasks**

* id
* description
* start
* end
* userId
* categoryId
* projectId
* taskToTagId

**User-to-Project**

* id
* userId
* projectId

## Page Pseudo Code

### Each page will do the following on load

1. Check session values to see if user is logged in or not
   1. If user is logged in
      1. check if there is a task that was not finished before (i.e. “end” of task is NULL) if there is such a task than continue it
      2. get all tasks that user tracked during the current week
      3. get all available projects, categories and tags user can use
   2. If user is not logged in
      1. get all tasks that where stored in a session
      2. if there was a task that was started during a session but not stopped – then continue it
2. Display data

### Each time user starts a task

1. Send the following information via AJAX POST request:
   1. action(set to “start”)
   2. userId
   3. task description
   4. projectId
   5. categoryId
   6. tagIds
   7. start time
2. On success backend will return taskId and frontend provides visual feedback
3. On failure – notify user

### Each time user modifies a task

1. Send the following information via AJAX POST request:
   1. action(set to “update”)
   2. userId
   3. taskId
   4. task description
   5. projectId
   6. categoryId
   7. tagIds
   8. start
   9. end
2. On success - provide visual feedback
3. On failure – notify user

### Each time user stops a task

1. Send the following information via AJAX POST request:
   1. action(set to “end”)
   2. taskId
   3. end
2. On success provide visual feedback
3. On failure – notify user

### Each time user loads report page

1. Send the following information via AJAX POST request:
   1. userId
2. On success – display report
3. On failure – notify user

### Each time user changes timeframe on report page

1. Send the following information via AJAX POST request:
   1. userId
   2. start
   3. end
2. On success – display report
3. On failure – notify user

### Each time user adds a project

1. Send the following information via AJAX POST request:
   1. title
   2. colour
2. On success – return projectId and provide visual feedback
3. On failure – notify user

### Each time user removes a project

1. Send the following information via AJAX POST request:
   1. projectId
2. On success –provide visual feedback
3. On failure – notify user

### Each time user modifies a project

1. Send the following information via AJAX POST request:
   1. projectId
   2. title
   3. colour
2. On success – provide visual feedback
3. On failure – notify user

### Each time user adds a tag

1. Send the following information via AJAX POST request:
   1. title
   2. colour
2. On success – return tagId and provide visual feedback
3. On failure – notify user

### Each time user removes a tag

1. Send the following information via AJAX POST request:
   1. tagId
2. On success –provide visual feedback
3. On failure – notify user

### Each time user modifies a tag

1. Send the following information via AJAX POST request:
   1. tagId
   2. title
   3. colour
2. On success – provide visual feedback
3. On failure – notify user