ChuckNotice - Software Requirements Specification

Author: Samson Daniel

Date last edit: 28-07-2022

# Description

ChuckNotice is a task management system that uses the power of Chuck Norris to improve team productivity. Features include:

Essential

* Accessible on any device with a web browser
* User Right Groups: Users can be assigned different rights by the admin
* Task Planning, allows creation, editing and deletion of tasks
* Task Scheduling, allows setting and monitoring of deadlines and dependencies
* Progress Tracking: Waiting, Open, In Progress, Finished status
* Chuck Norris quotes from the chucknorris API as completion awards
* Roundhouse kicks for missed deadlines (meaning you get kicked Chuck Norris style)

Additional

* Priority
* Recurring Tasks
* Collaboration: Comment on, Share, Receive updates on progress of relevant tasks
* Calendar
* Reporting and Visualization

# Data Details

## Mongo Data Architecture:

The Mongo data has been structured in a way compatible with SQL despite it’s NoSQL nature. Although this is strictly speaking not the most efficient way to implement a NoSQL database this was done so that the data could be easily moved to a SQL data platform, should the desire arise. As well as to demonstrate the understanding of SQL data architecture of the author.

Some features are required to be unique or adhere to certain value restrictions for the sake of functionality.

**Users:**

* \_id, String, unique identifier
* name, String, needs to be unique, used to log in and describe/greet the user
* password, String, hash, used to authenticate the user
* userGroup, String, used to determine user rights: [admin, manager, employee, roundhouse-kicked]
* penalties, Number, used to determine whether to roundhouse-kick the user, initially 0

**Tasks:**

* \_id, String, unique identifier
* title, String
* description, String
* deadline, Datestring
* status, String [Waiting, Open, In Progress, Finished]
* created by, user id string
* created\_on, Datestring

**Assignments**: (links users to tasks, many to many possible, but not all users have them. A task needs to have at least one user assigned to it to be valid. General tasks need to be assigned to all users.)

* \_id, String, unique identifier
* user\_id, String
* task\_id, String

**Dependencies**: (links tasks to other tasks, many to many possible, but not all tasks need have them)

* \_id, String, unique identifier
* task\_id, String, the task that depends
* dependency\_id, String, the id of the task that it depends on. Meaning tasks that need to have the Finished status before this one can be started (Waiting 🡪 Open). Not all tasks have dependencies.

## Client Data Classes:

List features can be generate through the corresponding relation data tables: Assignments and Dependencies.

User:

* Mongo User features
* Assignments: Array of unique task\_ids that are assigned to this user

Task:

* Mongo Task Features
* Dependencies: Array of unique task\_ids that this task depends on

# Implementation Details

**Admin User Management Interface**

This is used by the admin(s) only to register and manage users. Features:

* Add new user with temporary password
* Edit existing users: change the user rights only. If changed from roundhouse kicked status, penalties reset to 0.

Requirements:

* Passwords need to be hashed before being saved to the database.
* Penalties need to be initialized as 0.

**Login page**

This is used by all visitors to log in. Note that there is no registration option (at this point), because only admins are allowed to add users. Passwords need to hashed before being compared with the password in the database. Once the user is logged in he receives cookies to manage his session. The pages he will be able to access and actions he will be able to perform depend on user rights.

Once logged in the user is able to log out using the nav bar.

**Task overview page**

This is the default page. Allow user to see an overview of all tasks. List is sortable and searchable, does not include task description, dependencies. Clicking on a task will navigate to the task detail page, where it is also editable. Only managers can access the detail/edit page of tasks that they are not assigned to.

~~Also links to the add task page.~~ Also includes add task option.

Displays the Chuck Norris quote of the day from the chucknorris API.

**Add Task ~~Page~~ Option**

Fields include: Title, Description, Deadline (optional), Multiplicative Dependency selection (a new one appears every time you add one so you can add more dependencies) (optional), (Assign to if manager, also multiplicative and optional)

Then added to the task database through submit button, as well as the corresponding dependency and assignment data.

**Edit Task Page/Task Detail Page**

Allows you to view and edit a single task in detail. Same options as add Task page. Also allows you to change the progress status on tasks you are assigned to. Tasks with waiting status cannot be Completed until changed to Open.

Also allows for deletion of tasks. When a task is deleted, the assignments and dependencies corresponding to it also need to be removed from the database.

Finally, whenever a task is completed/deleted all tasks that had it as a dependency perform a dependency check. If ALL dependencies of those tasks are completed they change from Waiting to Open status.

**Task Deadline Monitoring**

The app automatically checks periodically for overdue tasks and flags them in the task overview (deadline < current datetime). Also adds penalties to the user for missed tasks. For now, the max amount of penalties is hard coded as 3\*, and missing a deadline simply gives you 1 penalty. And of course if the max amount is reached the user is round house kicked. The userGroup feature of the user is updated to “roundhouse kicked” and he is redirected to the roundhouse kick page.

**Roundhouse kick page**

This page is the only page that users that have been roundhouse kicked can visit, awaiting recovery of user rights by the admin. It obviously features a very intimidating picture/video of Chuck Norris’ roundhouse kick, as well as a statement that tells the user he has been roundhouse kicked.

**Chuck notice**

Give users a Chuck Norris quote whenever they complete a task. Uses the chucknorris API to get a random quote.

# Expansion Ideas

\*: In the future possibly add a settings data collection in mongo with one row and several attributes, including max\_penalties. This can then be expanded further for several use cases by adding additional rows, although this would also require individual user and task tables (or filtering by adding a use case feature).

# Sources

<https://www.ntaskmanager.com/blog/task-management-tool-features/>

<https://www.zoomshift.com/how-it-works>

<https://www.youtube.com/watch?v=vG824vBdYY8>: Coding Project Walkthrough - DEMO of the 1 coding project guaranteed to get you a development job

<https://stackoverflow.com/questions/4126811/how-do-you-track-record-relations-in-nosql>

<https://stackoverflow.com/questions/14289386/many-to-many-relationship-with-nosql-database>