Application for Task Management amongst multiple Users- Documentation

Classes and Methods used in the application code

MainPage

get():
The get method renders the mainpage.html view in the internet browser. The page displays the user's own task boards and those in which the user is a member of. The user is also shown with an input text box and a button to add a new taskboard. If a current user is logged in, the page renders a logout url, and login url if otherwise welcoming a guest. The user modal used in this application has a task boards property which contains all the taskboards that the user owns, or is a member of. This list of task boards is iterated to find those which the user owns, and these taskboards are marked as ('Created by you') in the main task boards list in the mainpage. Key of the user entity in the User model is also passed to the view via template values.

post():
The post method of the mainpage handles creation of a new task board. If a guest user does this, the user is taken to the error page with a message: 'Please login to create a new taskboard'. Input data validation is very important to keep the data safe and correct, thus check for empty string is done here. When a user successfully creates a task board, a new instance of Taskboard model is created and saved with the owner property filled with the entity key of the current user object from User model.

AddTask
AddTask handles the creation of tasks in a task board. This class doesn't have a get method.The view of this class thats, addtask.html is rendered by using template values from ViewTaskBoard class's post method.

post():
The post method of the addtask class takes in title, due date and assignee of a task. The assignee can be any member of the task board. As the input validation, the length of the input title is checked after stripping the text input, and continues if there is at least one valid character in the title. Other validations are done to check if another task exists with the same name. According to the requirement of the project, two tasks don't exist in the same task board with the same name. For adding a new task, an instance of Task modal is created with title, due date, isCompleteted (a boolean whether the task is completed or not. This is False by default), assignee (to whom the task is assigned), CompletionDateTime (time and date at which the task is completed). A task can be assigned to a member user or the owner itself, or the user can choose to assign the task to someone later. If the task is created

unassigned, it is highlighted red. The due date for the task is validated to be a date greater than or equal to today. This check is done in the post method of addtask class.

EditTask

The Edittask class has one method, post(self). It doesn't have a get method as the edittask.html view is rendered as template values from ViewTaskboard post method.

post():

The post method is used to edit any task. While posting to the form, the task entity key, task board entity key, the title of the task, due date, checkbox to determine whether the task is completed or not, the assignee of the task are passed. These information are used to modify the existing task entity and save to the datastore with the new details. For the due date, it is validated to be greater than or equal to today's date only if it is modified. The title is checked to be unique in the current task board by iterating through all the other tasks in the board and checking if the title name matches with the new one. The current task board entity is obtained by using the entity key of the task board. The tasks property of the current task board is used to get all the tasks in this task board. The user is redirected back to the task board if the editing works successfully. If there is an input validation error (like empty title), or if the new title exists for another task in the board, the user is taken to the error page and informed of the issue.

ViewTaskboard

ViewTaskbaord handles the user operations linked with a task boards, and displays information about the task board to the user.

get():

This get method organizes and prepares information about the task board and its tasks to display to the user.

- The task board entity key is also passed to this get method when the user navigates to this page from the mainpage. This key is used to form the task board key.
- The task board entity key is used to iterate through all of its tasks property. Each element in the repeated property (list) is the key of a task entity. In this way, all tasks that belong to this task board are taken to the view html through template values. ‘Tasks_keys’ has all the task keys and ‘tasks’ has all the task entities.
- Current user object is also passed to the view. ‘Users_keys’ list stores the user objects of all users in the system. This list is used to fill the dropdown of users to choose to invite to the taskboard.
- Other information such as the count of total number of tasks in the board, total completed today, total completed, and total active tasks are also passed to the view as template values and displayed as information to the user.
- All the tasks belonging to the task board are displayed on the page. The ones which are unassigned are highlighted red, to distinguish them from other tasks.They will be highlighted red until they are assigned to some member user.

post():

The post method of ViewTaskboard handles many operations that are connected with the taskboard.

- Inviting a user to the taskboard: Only the creator of the task board can invite a user to the taskboard. If a member user is again chosen to be invited to the taskboard, the error message is displayed to inform the user about this and asks to choose another user. The owner user is already a member of the board and cannot be invited. The chosen user's key is received from the form post. Each user has a property called taskboards which is the list of all task boards which the user owns, or is a member of.The task board also has a property called guests which stores the keys of all users that are its members. The user key is added to this repeated property of the current task board and both changes are saved to the datastore. The user is redirected back to the view task board page. Upon successful invitation to a taskboard, the invited user can see the task in the task board list in the main page. An invited user can also be assigned a task when adding/editing a task.
- Adding a new task: The addtask.html view is rendered when a user does an add task form post from viewtaskboard.html. The list of member users, the user object, and the task board key are passed as template values. This process involves creating a new instance of the Task modal and saving to the datastore. The task entity has a title, duedate, isCompleted which is false by default, an assignee which the user selected while adding a task, and completion time, which is None by default. All of this are handled by the post() method in AddTask class described before.
- Edit board: Any member user (member of the current task board) can edit the task board by renaming it. A use can change the text of the task board by clicking on the name and clicking the edit board button. It changes the name property of the task board entity and saves changes to the datastore.
- Edit task: Any member user can edit a task. The edit task button is displayed against each task. The post method renders edittask.html view and passes the task board entity, its key, user object, task key, and the member users of the task board as template values to this view. The editing task process is handled by a form post in the view, which is handled in EditTask class which is described earlier.
- Delete task: Any user member of the board can delete a task. When a task is deleted, the task key is removed from the tasks property of the taskboard, and then the key is deleted. This ensures all the details including the key of the task is wiped from the datastore and no corrupt data remains in the system.
- View and modify users: Only the creator of the board can access the functionality to invite a user, and also view and delete the users in the board. The user is presented with the viewmodifyusers view along with task board owner details, guests details, and the task board entity key passed to the view via template values. The get method of this ViewAndModifyUsers class handles the process of removing the user.
- Removing the board: A task board can only be removed after removing all users from it and deleting all its tasks. Only the owner of a task board can access this functionality and warns the owner user via an error page if there are tasks or guest users in the taskboard, if there is any.
- Accessibility- back button: The will be taken to the homepage directly by clicking this button.

- Checking a task and marking its completion time: This is the default task the post method does if none other is specified. This is used to check a task by clicking on the checkbox in the viewtaskboard view on each task. If the user is checking a task, then the current date and time are displayed and persisted for that task for the completionDateTime property, along with marking the task complete. If the user unchecks, the task is marked again as not complete, and the completionDateTime property is set to None.

ViewModifyUsers

get():
The get method of ViewModifyUsers class handles the deletion of guest users (users which are guests/ invited to the taskboard) of a taskboard. Viewmodifyusers.html view is rendered from the post method of viewtaskboard class. The user is displayed with the guests user emails as a hyperlink. Clicking on the link will redirect the user to this get method. The task board key, the email id of the user chosen to be deleted from the board are retrieved as url parameters in the get method. There are three processes which are carried out here:

- The task board key is deleted from the task boards list of keys in the user model
- The user entity key is removed from the guests property of the taskboard
- Each task has an assignee property which stores the key of the member user to whom the task is assigned to. This needs to be checked. All the tasks of the task board are retrieved from the task board entity as tasks are a property of the task board entity. This list of task keys are iterated and checked if the chosen user is the owner of any of these tasks. If yes, the task's assignee is set to None and saved to the datastore. This will make the task highlight in red color. All unassigned tasks are highlighted red until they are assigned to some member user.

Data Structures and models used:

Datastructures used:

- Member_board_keys: this is a list of all task boards that the current user is a guest of
- Own_board_keys : list of all task boards that the current user is the owner of.
- Tasks_keys : list of all keys of all tasks in a particular taskboard
- Users_keys : list of all users in the system. This contains each user object.
- Guests : the guest users (member users, not the owner) of a taskboard.

Models used:

1. **Taskboard**

The task board model is used to identify a single task board in the system. An instance of this model is created when the user creates a new task board and save. A task board model consists of the following properties namely name, owner, tasks, guests, and guests_assigned.

- Name- string property
  Stores the name of the task board

- Owner- key property
  Stores the key of the user who is the owner of this taskboard

- Tasks- key property, repeated =true
  A list of all keys of all tasks in this taskboard. When a new task is added to the board, the corresponding task entity's key is added to this list.

- Guests - key property of users (repeated-true)
  A list of keys of all guest users in this task board. Guest users are those users who are invited to be a part of this task board by the creator of the board.

2. Task

Task model is used to identify a single task in the system. An instance of this model is created when a user creates a new task inside a task board and save. A task model consists of the following properties namely title, due date, isCompleted, assignee, completionDateTime.

- Title- string property
  The title of the task

- Duedate- Date property
  The due date of the task. When adding a new task/ editing a task, this is validated to make sure that a user cannot enter a new due date that is before today's date. If a duedate is earlier than today when editing, and if it is left unchanged, the user is allowed to edit and save the other properties of the task.

- isCompelted- boolean pro[perty
  True if the task is marked completed and false otherwise

- Assignee- key property
  This property stores the single key of the member user to whom the task is assigned to.

- completionDateTiem- datetime property
  The date time when the task is marked completed either from the view task board page or the edit task page.

3. User

The user model is used to identify a single user in the system. An instance of this model is created when a new user log-in to the application. A user model consists of the following properties:

- Email_a: string property
  This is the email address of the user and the primary way to identify a user other than its key.

- TaskBoards- key property of Taskboard model with repeated=True
  Stores at the task board keys of which the user owns (user is creator) and the user is a guest (user got invited to it).

Application design and design decisions:

- **Renaming of task board**
  The user needs to just click on the name of the task board in the viewtask board page and can rename the task board with a single button click. This saves the user time and complexity of having another form specifically for renaming/editing a board.

- **Custom error page**
  A custom error page is used to redirect the user to in case of any errors, or to inform the user of any special situation for example, validation error.

- **Tasks displayed as tables**
  Into the application UI, each task is displayed as a single table with information about the task as each row. This design is considered innovative and could make it easy for a user to use the application and distinguish between each task.