ASSIGNMENT 02

Name: Gokul Sreeletha Kannan
Student ID:2985939
Course: MSc Computing
Assignment: Building an Application for Task Management amongst multiple Users
Module: Cloud Platforms and Applications

INTRODUCTION

This is a documentation for the web application designed using webapp2 framework on the Google app engine. Interfaces are rendered using the Jinja2 template engine and ndb is the database used to store data. The application is a database for Taskboards. The user can login and create their own account in the database. Each user can create taskboards and invite other users to join in. Any invited user in a taskboard can create tasks and assign them to others. They can also edit the existing task or delete them from the board. The author of the taskboard can add and delete users from the board and also change the name of the taskboard and also can delete the whole taskboard.

DATA MODELS

In this assignment, There are three data models used. Every model is inherited from ndb.model.

Task(): The Task model is used to hold tasks created. It has the following attributes.

- *title* : StringProperty(), This is a single string used to hold the title of a task.
- *dueDate*: StringProperty(), This attribute is used to hold the due date of the task. This is a single string.
- *user* : StringProperty(), This is a string used to hold the email of the user that the task has been assigned.
- *isCompleted*: BooleanProperty(), This is a boolean that is used to know if the task is completed or active.
- *completedOn*: StringProperty(), This is a string that has None as its initial value. If a task is marked completed, then the date and time is stored here.

Taskboard(): The Taskboard is used to hold individual taskboards. Each taskboard should have the following attributes.

- *name* : StringProperty(), This is a string that holds the name of a taskboard.
- *author*: StringProperty(), This is a string that holds the email address of the user who created the taskboard.
- *users* : StringProperty(), This is a list of strings, which holds email addresses of all the users that are invited into the taskboard

- *tasks* : StructuredProperty(), This is a list of tasks that are created for the taskboard. It is linked to the Task model.

User(): The User model is used to hold the details about individual users. Each user should have the following attributes.

- *email* : StringProperty(), This is a string that holds the email of the user.
- *name*: StringProperty(), This is a string that holds the username of the user.
- *taskBoards*: KeyProperty(), This is a collection of keys of taskboards that the user has created.
- *invitedTBs*: KeyProperty(), This is a collection of keys of taskboards that the user has been invited to.

MainPage

The MainPage class is the home page of the application it resides in main.py, which is the main handler of the application. The MainPage is rendered using main.html. In the main page user can login. There is only a get() method here. In the method, they can find two links, one to create a taskboard (that routes to AddTask class) and the other to view all the taskboards (that routes to UserTaskboards class). The class also automatically signs up a new user that login for the first time and saves there data into the User model.

AddTask

The AddTask class is used to create a new taskboard. This is only accessible to a logged in user. The class consists of a get() and post() methods. In the get() method, the current user is fetched and pushed to the template as template_values. The template used to render this class is addTask.html. In the template there is a form that has a textfield for name and two buttons for creating a taskboard and cancelling the whole operation and return to home page. The data in the form is passed into the post() method. In the post() method, we get the action to find out which button the user clicked. If the user clicked the Create button, then the value in the field called boardName is fetched and stored into a variable called taskboardName. Then we create a new object of the class Taskboard that has the taskboardName as the name and the email address of the current user as author. Then it is entered into the database. Then the key of the Taskboard object is appended into the taskBoards list of currentUser. If the user clicked on the Cancel button, then he is redirected to MainPage.

UserTaskboards

The UserTaskboards class is used to show all the taskboards in an individual users database. This class only has a get() method. In this method, the current user is fetched and there are two separate lists for the taskboards created by the user (currentTBs) and the taskboards into which the user has invited to (invitedTBs). These two lists are filled with respective taskboards obtained from the keys in the User database and is passed into the template. This page is rendered using userTaskboards.html. In the template, there are two separate for loops for each taskboard collection. Each taskboard is displayed as a link button that has id of the taskboard as the value is directing to TBoard class where the user can view respective taskboard.

TBoard

The TBoard class is the one that has the most functionalities in the application. It shows the details about a single taskboard. There are two methods, get() and post().

In the get() method, the current user object is fetched from the database and is stored in the variable currentUser. Then an empty list called permittedUsers is created to hold the users that are permitted in the taskboard. The taskboard object is obtained from the database using the key variable in the url that holds the id and is stored in the variable as tb. Then all users except the one created the taskboard is queried and is stored in the variable allUsers. Then all the users in the users field of tb is queried and is appended into permittedUsers. Also, the counter are initialized. totalTasks holds the length of the tasks list of tb. pendingTasks, completedTasks, completedToday are all initialized to 0. Then, today variable is created to get the current date to know if the task is completed today. Then there is a small loop that deals with the values of the counters above mentioned. The loop iterates through each of the task in tasks of tb and checks if they are completed. If they are not completed, then pendingTasks is incremented. If they are completed then completedTasks is incremented and then further checks if they are completed on the same date as today, if yes completedToday is also incremented. In the template values, all these values are passed into tboard.html to render. The same naming is used for the variables to avoid confusions.

In the post() method, the current logged in user object is stored in currentUser and the taskboard object is stored in tb. There is a variable called action that is used to know which button, the user clicked.

If the user clicked on AddUser, this deals with inviting a user to the board. The user object is taken from the database and is stored in invitedUser. Then the if condition checks if the user has already been invited. If yes, an error is printed. Otherwise, the key of the taskboard is appended into the invitedTBs of the invitedUser and the user email is appended into users in tb.

If the user clicked on AddTask, this button deals with adding a task into the taskboard. Initially, the taskTitle, dueDate and assignedUser is fetched. Then all the titles of the tasks in tb is selected and appended in taskTitles. If the taskTitles does not contain taskTitle, then a new Task object is created with respective attributes and is appended into tb.tasks. Otherwise, an error is shown indicating that the task title already exists in tb.

If the user clicked MarkCompleted, this button is coupled with checkbox isCompleted in the table. It is used to mark all the checked tasks are finished. Initially, all the indices of the marked tasks are stored in marked. Then, looping through each index in marked, the respective isCompleted on tb.tasks is changed to True and the completedOn is filled with current date and time. Due to the if condition in the interface, the checkbox of the checked tasks disappears, replacing them with the value in completedOn.

Finally, if the user clicked on DeleteTaskboard, the user has to be the author of the taskboard. This is checked using an if loop in the interface. Then, the length of tb.users and tb.tasks are checked to see if they are 0. If they are not, then an error is shown. Otherwise, the taskboard object tb is deleted from the Taskboard model and its key is deleted from User.taskBoards.

EditTask

The EditTask class allows the user to edit an individual task. This class is rendered using editTask.html template. This class has two methods, get() and post(). In the get() method, the taskboard and index of the task is taken from the url and permitted users are queried. All these values along with the current user is passed into the template. In the post() method, if the user selects Edit, the taskboard object and index is fetched and is values in the respective task fields are replaced with the new ones. If the user selects Delete, the respective task is deleted using the del command. If the user selects Cancel, the user is redirected to the TBoard.

EditBoard

The EditBoard class allows the author of the board to rename the board and delete users from the board. It is rendered using the editBoard.html template. In the get() method, the taskboard object is fetched in tb and the permittedUsers are also fetched. Both values are passed into template along with the length of permittedUsers as noOfUsers. This is done so that if there are no permitted users, the option to delete users can be hidden using an if statement. In the post() method, the taskboard object is stored as tb. The value of the input button is stored in action. If the user clicked EditName, if the boardName field is not empty, then tb.name is replaced with the new value. If the user clicked RemoveUsers, then, all the checked values are fetched in selectedUsers. Then, firstly, All the tasks assigned to each of the user is changed as Unassigned. The tasks with Unassigned users are highlighted red in TBoard. Then, each of the users are queried and key of the taskboard in invitedTBs is removed. Then, each of the user email is removed from tb.users.