Project Features list:

1. Enter tasks: ability to put in tasks and a description of the task to add it into the calendar.
2. Calendar view of tasks: we will have a monthly and a daily of view of everything the user has entered (part 1).
    a. creation and dump date in database: we will have the user input the start date (and if needed, end time) and put that in our database.
3. Day view of tasks (rank by importance): the user will be able to see their tasks by the day, and these tasks will be ordered based on how important the user ranked the tasks to be.
4. Alerts for upcoming tasks (rank by importance): we will have important and upcoming tasks on the side to remind the user not to forget about them.
5. Email notifications: the user can set how often they want to be reminded (hourly, daily, etc.) and whether it's for every task or many tasks at once.
6. Categorize using color coding: The user will be able to input what a certain task is categorized as (academic, social, sport, etc.), and we will color code each one for readability.
7. Customize task importance: the user can rank how important each task is (urgent, important, slightly important, etc.).

Requirements for features (at least 6)

Enter Tasks:

When a customer selects the button "add task" a minimum of two form items will display. The first form item will be titled "name of task", and this will be where the user types in a short description of the task. This form will be required input. The second form item will be an option description of the task, which is not required input from the user. These form items will take strings as input, with a maximum of 180 characters for the first input, and 360 characters for the second input. There will also be a drop down menu which prompts the user to select a "due date" for the task, with the user selecting a valid date. This input is required. The software will validate that a valid user is logged in. Once the user has input something into the first form and selected a valid due date, a button will appear that will allow the user to actually add the task. A button to cancel the task will always be available to the user. If the customer adds the task, the software will add that task to the database table which holds the tasks for the logged in user, and send out reminders based on criteria either set in user settings, or by the level of importance, which is described in another feature. If a valid user is not logged in, the software will display a login screen which prompts the user to log in or register first in order to add the task. If the user cancels the task, the add task view will go away and

they will return to their user homepage. Aside from the user inputting information, this process should take at most two seconds.

Day view of tasks:
The day view of tasks page will display only the tasks that the user has scheduled on the day of access. Each task will have an edit button (that looks like a pencil) which when clicked, will open a popup to allow the user to change the name description or importance of a task. If the user does not have any tasks scheduled then a message will be displayed that reads “No tasks to complete!”. Tasks will be displayed in a row format as cards stacked on top of one another. The tasks will be ordered by level of importance(not at all important, slightly important, important, very important, and extremely important), and will have colors corresponding to their importance(blue, green, yellow, orange, and red respectively). Each task will have a large check mark button labeled “mark complete” that, when pressed, will grey out the task signifying to the user that the task no longer needs to be done. At the bottom of the page the user will be able to navigate to the next days tasks as well as the previous days tasks. There will also be a button enabling the user to add a task. When the page is loaded the front end will first verify that the user is logged into a verified account, it will then retrieve the user’s task data from the database where it is stored. It will then display the tasks. If the user alters any task or adds a new one, the application will verify their account and update the database accordingly.

Set task importance:
When a user creates a task part of the form will include a drop down menu with five different levels of importance(not at all important, slightly important, important, very important, and extremely important). This drop down menu will also be included in a pop up that appears when a user edits a task. Each task will have a color dictated by its level of importance (blue, green, yellow, orange, and red respectively). Selecting or editing the importance of a task will change its color. Since the user’s task are shown based on level of importance, changing a task’s importance will also adjust the order in which the tasks are shown. After a task’s importance is set or changed the application will first verify that the user is logged in and using a valid account, then the application will save any task alterations to the database that contains the user’s tasks and task information.

Calendar View of tasks:
The calendar view of tasks will display tasks the user entered for a week. The user can navigate to coming weeks with arrow buttons. The user’s tasks and events will be represented by buttons arranged in columns for each day, represented in different

symbols and colors depending on its category and importance. The button will show the name of the task. The user can click on each button to open a task view window that lists the complete information for the task, including buttons to mark “completion” and to delete the task. An edit button brings up the form of the task to make changes. The user exits the task view by pressing the x button. At the bottom of each day, a block of information lists the estimated amount of work time the user should expect and the amount of free time according to inputted information. The calendar is implemented using bootstrap tables. Tasks are fetched from the database with queries for each day of the calendar ordered by time and other factors.

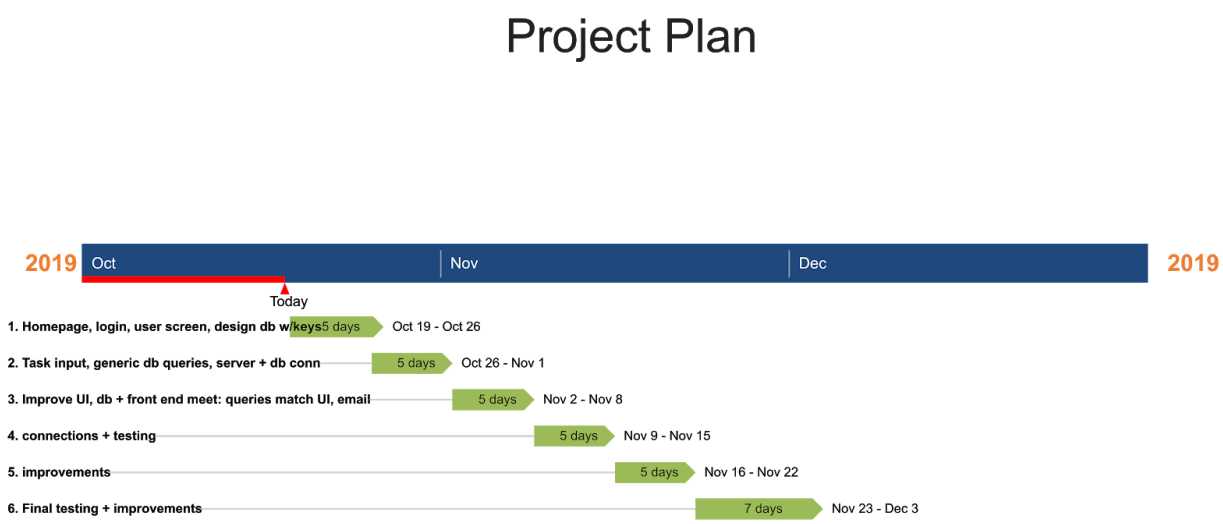
Categorize using color coding:
For clarity, organization, and classification, task titles will be color coded. When the user enters his or her task, there will be a selection of categories that the user can label the task as, such as academic, social, sports, etc. For each of these categories, we will assign a color to it; for example, academic as blue and sports as red. This way, the user can immediately recognize what type of event this is. We will write JavaScript functions that relate the category to the color, so depending on which category is selected, that color will appear as the font for the task title.

Server & Web page communication:
The server will be written in Python, which handle the http request from web page, such as updating the database with the information provided by the incoming http data package, or sending back the information from the database queried by the client side.

Upcoming tasks alerts:
User can rank tasks by importance, and back-end can sort these tasks by time and importance. When a task coming up. The back-end will alert the user by different importance. The more important task, alerts are more frequent. Alert can not only send to Front-end to make a reminder but also can send a reminder to the email the user left.

Email Notification:
The back-end server will scan the tasks list, and send Emails to corresponding users whom signed the automatic Email notification preference; Different tags (importance), such as “not important” or “very important”, may be applied to each task, which will change the frequency of Email notifying. More specifically, there will be a thread scan database constantly, and a dispatching thread that waiting for email sending. For the former one, it put all email tasks that need to be sent out into a list, then active the spin lock of dispatching thread.

GANTT chart:



1. Front end: Initialize the default homepage, login, user homepage. Design the database and implement all primary keys. Create server and database, begin thinking of connecting them.
    a. Front end: homepage % defaults - Gibson and Sophie
    b. Database: initialize tables w/ keys - Huilin & Yvonne
    c. Server: server creation and DB conn - Yi & Leyen
2. Create functionality for front end task input, create generic database queries, finish connection between server and database. Database & server will work together to finalize & improve the API and the connection between the database & server.
    a. Front end: task input - Gibson & Sophie
    b. Database: generic queries - Huilin & Yvonne
    c. Server: server & database connection - Yi & Leyen
3. Improve user interface, check that queries match front-end functionality, implement email notifications
    a. Front end: user interface, meet with db team: Gibson & Sophie

    b. Database: finalize queries, meet with front-end team: Huilin & Yvonne
    c. Server: email notifications - Yi & Leyen
4. Make sure that the front end, database, and server are all working cohesively. Begin writing and implementing tests for all functionality. Front end & database will work together to make sure the flow of information is correct.
    a. Front end: check to make sure that the correct information is being received from the backend, and the correct information is being sent to the backend - Gibson & Sophie
    b. Database: check to make sure that the correct information is being sent to and received from the front end. - Huilin & Yvonne
    c. Server: check to make sure that the API is working correctly. Check for bugs in server and all methods - Yi & Leyen
5. Based on information gathered in testing, resolve all bugs and problems. Implement improvements on functionality, UI, and time/space complexity. Work on improving experience, run time, and appearance.
    a. Front end: improve UI. Make sure that all methods & functions are readable and organized efficiently in the file structure. Improve appearance & user experience - Gibson & Sophie
    b. Database: change set up based on problems found in testing. Work with server team to improve efficiency of API calls - Huilin & Yvonne
    c. Server: make changes based on problems found in testing. Improve server efficiency. Work with database team to improve efficiency of API calls - Yi & Leyen
6. Final testing: Make any final tweaks/fixes that are needed. Try to streamline the site and improve user experience as much as possible. This phase will be done as a group to minimize miscommunication during finalization. We will also need to create and practice a presentation to give to the class.