BRIGHTSPACEBOT

Team 14: Design Document

Chai Hyun Park
Katherine Xiong
Kristy Huang
Raveena Nair
Shaun Thomas

1. Purpose

Many college students utilize the online platform, BrightSpace, for accessing class content and managing assignments: such as downloading lectures or documents and checking due dates. The BrightspaceBot aims to automate these redundant and time-consuming tasks so that students are well prepared for their classes. Aside from automating tasks, BrightspaceBot also provides a chat system for users to interact and get answers to queries from the bot and a management system for BrightSpace notifications to prevent unnecessary distractions.

1.1 Functional Requirements

1.1.a - Basic Functionalities:

As a student:

1. I would like to set up my Brightspace account with BrightSpace Bot
2. I would like to be able to connect (authenticate) my Brightspace account to BrightspaceBot.
3. I would like to configure my bot through an intuitive website
4. I would like to register for an account on the website
5. I would like to log into the website
6. I would like to update my personal information such as my major on the website
7. I would like to configure whether I want to interact with the bot through email, discord, or just the website itself
8. I would like to update the destination I want my files to be downloaded
9. I would like to customize the number of notifications I get from the bot
10. I would like to specify the type of notifications I get from the bot (section being updated, grade updated, course announcements, etc.)
11. I want my notifications to be separated in different text channels on Discord, based on the type of notification
12. I want to be able to specify if I want all BrightSpace Bot emails to go to a specific folder
13. I want to be able to choose my bots name if desired

1.1.b - Automated Tasks:

As a student:

1. I would like the bot to sync the due dates of my upcoming assignments with the calendar of my choice
2. I would like to have the upcoming quizzes for a class integrated into my calendar of choice
3. I would like to have office hours for the classes I select to be integrated into my calendar of choice
4. I would like to have any discussion posts that are due to be marked as an event on my calendar
5. I would like to easily search for a specific student in my class
6. I would like the bot to prioritize which class to focus on based on grades and assignments that are due soon
7. I would like the bot to compare class schedules with other Brightspace users to suggest potential study groups
8. I would like to have my files from Brightspace to be downloaded to a location of my choice automatically by a schedule or manually.
9. I would like the bot to organize the course files when downloading on my local device or shared drive.
10. I would like the bot to download different types of files by different schedules
11. I would like the bot to mark the default sections on a BrightSpace course as read (checkmark appears)
12. I would like the bot to be able to rename the files the bot downloads for my courses to names that I specify
13. I would like the bot to archive past assignments

1.1.c - Chat System:

As a student:

1. I would like the bot to understand the sentences I type without having to explicitly write certain commands
2. I would like to ask the bot what my upcoming assignments are in order to prepare myself for classes
3. I would like the bot to tell me if I have any upcoming exams or quizzes that I haven't completed yet
4. I would like to change any configuration settings I need through the bots chat system
5. I would like the bot to suggest classes I should be focused on based on factors like grades, unsubmitted assignments, deadlines

6. I would like to ask the bot which weeks have the most assignments due
7. I would like to ask what my overall points received are in all my classes
8. I would like to know what my current letter grade in a class is
9. I would like to know if a grader has provided feedback on an assignment
10. I would like to know what feedback a grader has provided
11. I would like the bot to provide me with quick links to the module page that I want to visit

1.1.d - Proactive Notifications:

As a student:

1. I would like to configure how frequently I want to have notifications from BrightSpace Bot
2. I would like to be able to get notifications from classes before the class starts so that I am prepared to attend the class.
3. I would like to be able to get notifications daily so that I can be up-to-date about my classes.
4. I would like to know if my professor has uploaded a lecture to Kaltura Media Gallery for the course so I can complete other tasks while that is being set up.
5. I would like the bot to tell me when an instructor has updated or added a new section to the Brightspace course so I can stay up to date with the changes made.
6. I would like the bot to remind me of any upcoming discussion threads I need to post on.
7. I would like the bot to remind me that after I post to the discussion thread, to reply to other students' posts so I am engaging with peers in my class
8. I would like the bot to notify me if my grade has been updated so I don't miss important updates.
9. I would like to configure the bot to notify if there are any recorded lectures

1.2 Non-Functional

1.2.a - Accuracy

1. I would like the application to understand human-generated phrases, rather than specific commands

2. I would like the bot to send notifications at accurate times regarding course schedules
3. I would like the bot to accurately save and download the course files to the designated drives or devices

1.2.b - Efficiency

1. I want the bot to have a response time of at most 5 seconds
2. I want the bot to better optimize the time students spend on BrightSpace
3. I want the bot to download a file whenever an instructor updates the course page
4. I want the bot to synchronize the events to my local calendar every time the instructor announces new events
5. I want the bot to download files to a specific path before going to classes

1.2.c - Security

1. I want the bot to remember my credentials in order to automate the login process for efficiency purposes.
2. I want the bot to have my user information in safe security

1.2.d - Flexibility

1. I want the bot to be able to run on various platforms, including Email and Discord.
2. I want a web page that allows me to set my configuration settings when I initiate BrightSpace Bot for the first time.

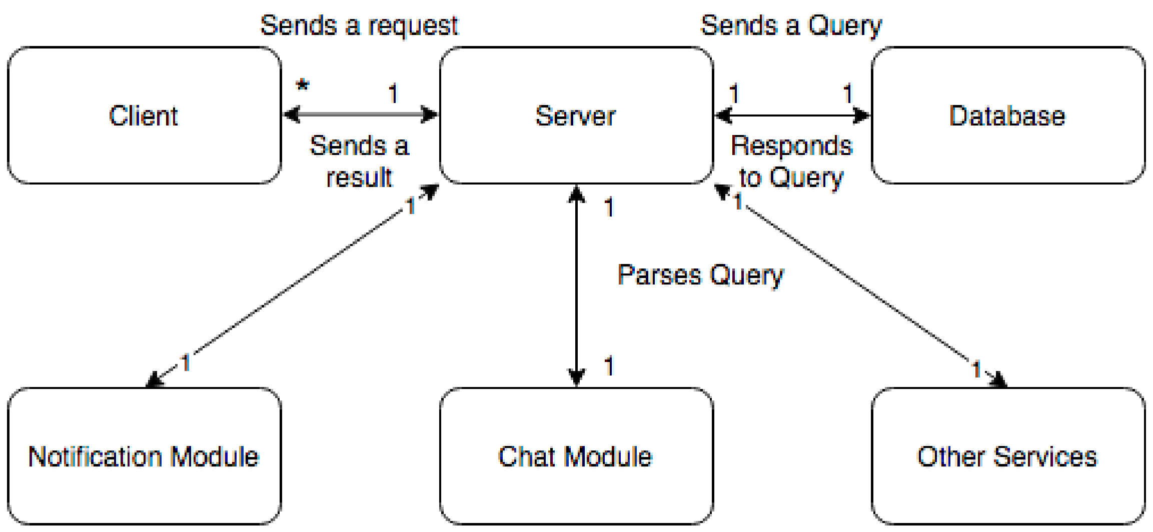
1.2.f - Usability

1. I would like the bot to have a simple installation process

2. Design Outline

High Level Overview:

This project is an application that enables students who use Brightspace to connect to a bot that will automate cumbersome tasks for them. The bot connects to Brightspace and will have access to the student's class content and assignments. To this extent, the bot has the resources to download lectures, documents and report due dates to the student. For the high-level architecture, this project will follow a client-server model as shown below.

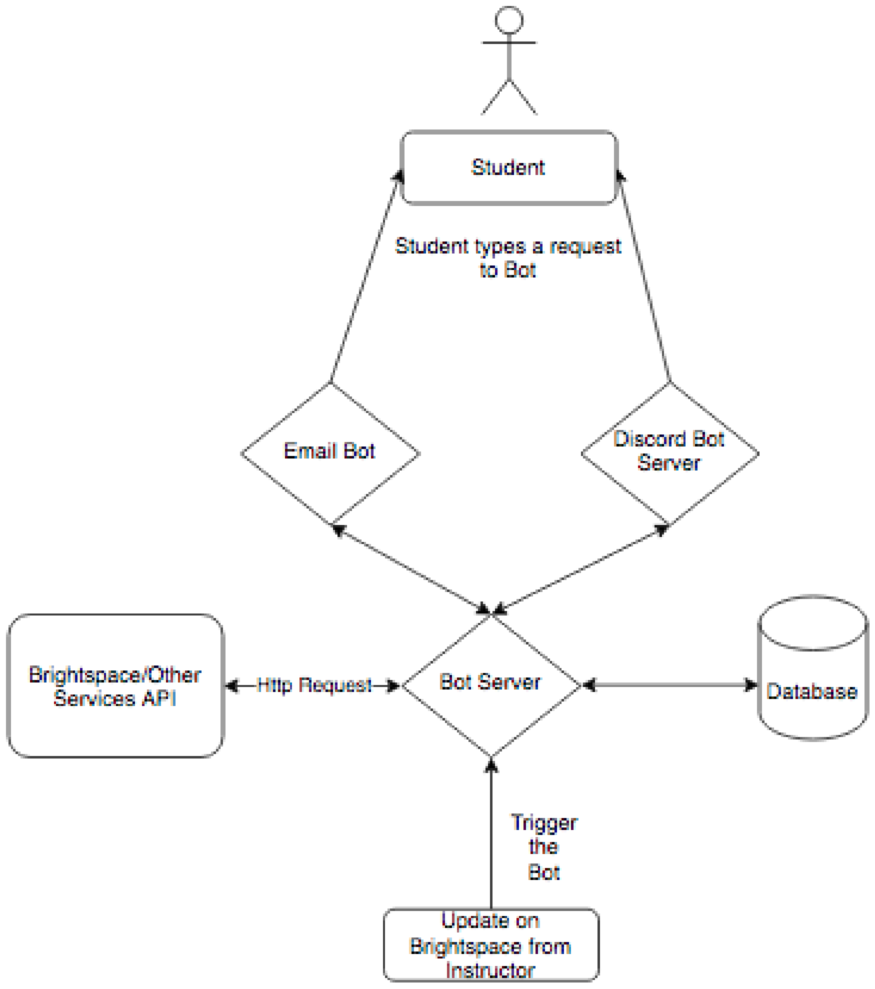


The client will start off by sending a request to the server. The server will then be in charge of processing all the logic to answer the request, whether that means accessing the Chat Module that parses phrases to recognize a task, the UserInfo Database, BrightSpace API Services, or Notification Module that sends proactive notifications. After the server obtains the appropriate response, it relies on the result back to the client-side which will be processed or displayed appropriately.

Detailed Overview of Architecture:

The user has two options of where they want to interact with BrightSpace Bot, either through Discord or Email. Both platforms can be used to ask queries to the bot and get notifications. The Bot System can either be triggered by a user query or any update from any BrightSpace course they are registered for. If a user types a request to the bot, then the bot server will take that request and parse it through the Chat Module.

Afterward, the request will then be executed either by accessing the BrightSpace API, the UserInfo Database, or other services. When accessing the BrightSpace API, HTTP Requests will be crafted and executed in order to obtain the necessary information. If it is information that has been saved in the database, then instead of an HTTP Request, a call to the database will be made to request the appropriate information. Afterward, this request is replied back to the student. In another scenario, BrightSpace Bot will always be listening to updates from BrightSpace about any updates, announcements, or deadlines that have been created in any of the courses the student is enrolled in. If something of those sorts is done, then the bot will listen to that trigger and make an HTTP Request to obtain the new information updated to the course page. Once that request is obtained from the services by the Bot server, then that reply gets formatted as a notification to the user in the platform they are interacting with BrightSpace Bot.


Student
Student types a request to Bot
Email Bot
Discord Bot Server
Brightspace/Other Services API
Http Request
Bot Server
Database
Trigger the Bot
Update on Brightspace from Instructor

3. Design Issues

Functional Issues

1. What information do we need for the users to log in?

   Option 1: user name, user password
   Option 2: user name, Boilerkey
   Option 3: user name, user password, Boilerkey

   Choice: option 2

   Since it is Brightspace where our Bot will be mainly used, we will need to retrieve the same information as the user uses to log in to Brightspace which are the user name and the Boilerkey.

2. Will the users have to log in every time?

   Option 1: yes, the users have to log in every time
   Option 2: no, the user information will be stored in the bot so no need for a login
   Option 3: if first time using bot, yes log in required
   Option 4: If not first time using the bot, no it will already have login information about the user into a database

   Choice: option 3 & 4

   It is not efficient to store all the user data in Brightspace to the Bot database since there could be students who don't utilize the Bot nor always ask the user to log in repeatedly which will not meet our goal to reduce time-consuming works on Brightspace. Thus, an implementation of both options 3 and 4 is needed so the BrightspaceBot can store user data efficiently in the database and be convenient for users without repeated log-ins.

3. When will the automation begin?

   Option 1: when user requests it
   Option 2: it is set to user preference through configuration
   Option 3: both

   Choice: option 3

   First when the user login to the Bot it will have the user set a preferred time to automate tasks which are downloading class files into the local devices or cloud storages. In order to trigger this, the user has to request to the chat model then the server will respond and change specific settings to deliver the course materials to the appropriate time and storage.

4. When will the users get notifications?

Option 1: daily
Option 2: as soon as new updates happen in Brightspace
Option 3: set a time to get notifications
Option 4: All of above

Choice: option 4

We decided to implement all three of the options due to 3 cases of notification: announcements with no event, announcements with events, close deadline due. For announcements with events posted by course instructors, we will design the bot to notify the students right away. With no-event announcements the bot will only store the notification to a specific data and may inform the student at the configured time that the student as set.

5. Where will the downloaded documents be stored?
   Option 1: Local device
   Option 2: Cloud storage
   Option 3: Both
   Option 4: set by user preference

   Choice: option 4

   When the user login to the Bot and triggers a request to download files, the bot will ask to specify the path to download the files. The options will be to the specified local device or a cloud storage system like GoogleDrive or OneNote etc. There will be two different methods of storing the course files for these two cases.

Non-functional Issues

1. Which languages are appropriate for our backend logic?
   Option 1: Python
   Option 2: Java
   Option 3: C#
   Option 4: Node.js

   Choice: Option 1

   Discussion: We decided to implement our backend in Python as that is a common language people use to develop Bots. There are a lot of online resources that we can use and reference for guidance. Although all of us have experience in Java through CS 180, we thought that it would be nice to use a language that has better resources and also allows us to learn new technologies. A few of us are familiar with Python, so there will be some of us to help those who are not proficient in the programming language. Additionally, a lot of discord bots and email bots are made using Python, and Python has a lot of different libraries that allow for easy data manipulations.

2. How do users set up the bot?

   Option 1: Download and follow steps from GitHub README
   Option 2: With a click of a button, all installation processes are automatic
   Option 3: Have a website that does some of the installation processes automatically, but also have some commands be done manually.

   Choice: Option 3

   Discussion: we weren't sure how feasible it was to have the user have to follow the steps to set up the bot using the GitHub README. Although it would be easier for the developers, we must remember that not everyone using our services will be familiar with GitHub and so that might cause our target audience to not be met. Additionally, we weren't sure how well things would run automatically, nor how to implement that. After doing some reading on Discord Bots and Email Bots, we saw that many developers create a website that saves all the user configuration information, but when specifically creating a discord bot, there are some things the user must do through the Discord Platform that is out of our control, which we will provide documentation on in our website

3. How will we store user information and configuration settings?

   Option 1: SQL Database
   Option 2: MongoDB

   Choice: SQL Database

   Discussion: Currently, we are going to use a SQL Database to store all the user and configuration settings information that we need. We thought this would be easier as some of us have experience with this technology, and SQL is a secure and reliable database management service. There are a lot of online resources and tools to help us out.

4. How will our bot always listen to the announcements that are being made on BrightSpace?

   Option 1: The bot is always running in the background
   Option 2: The bot only activates when triggered

   Choice: Option 1

Discussion: We have decided to have the bot always running as that is what causes the bot to be the most effective for the user. If it is not always listening, and the user has to trigger the bot each time, then the purpose of the bot won't be executed and things will not be done proactively or automatically.

5. What front end language should we use for our BrightSpace Bot website?

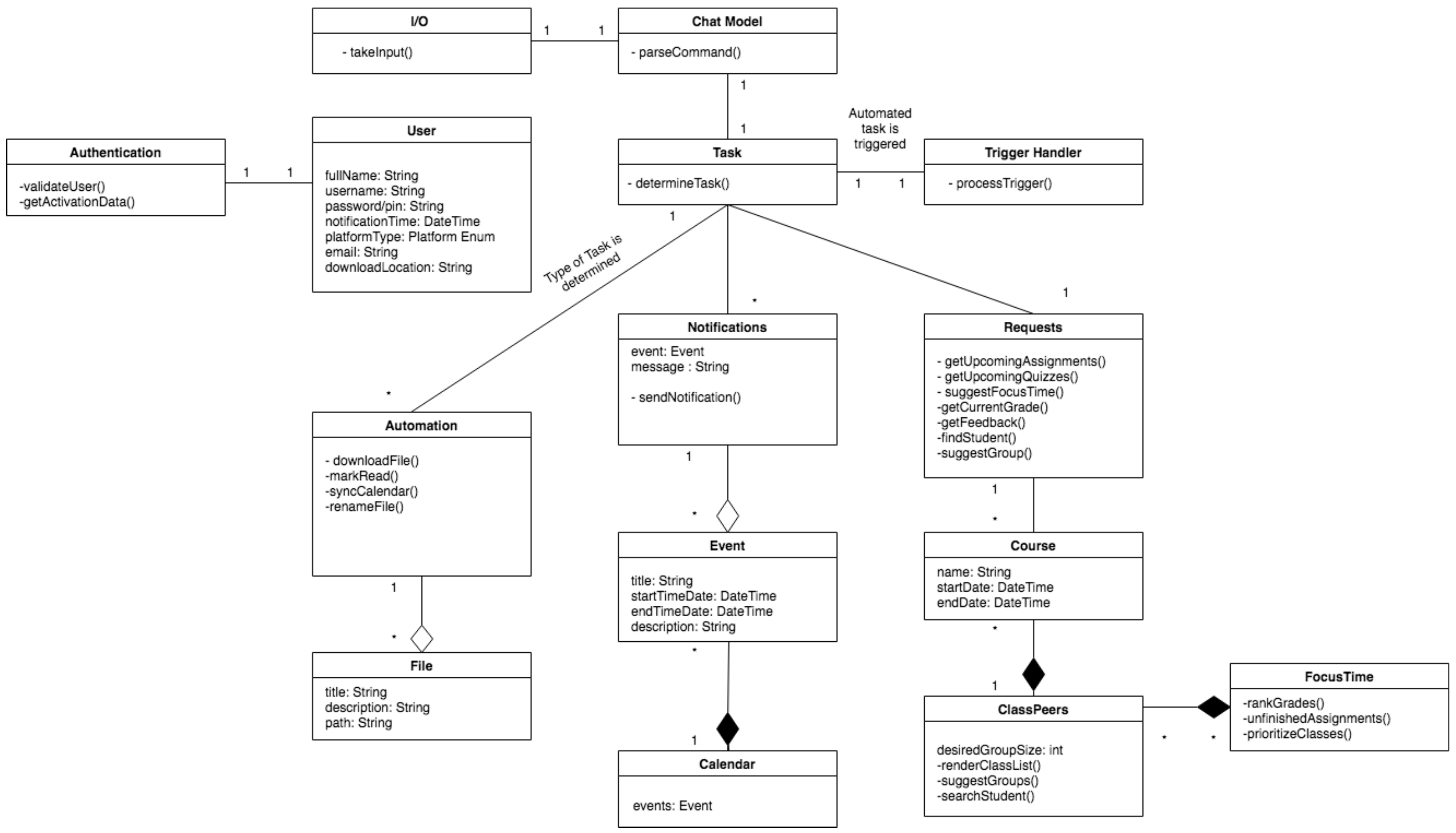
   Option 1: React
   Option 2: HTML & Javascript
   Option 3: Angular

   Choice: Option 1

   Discussion: Currently, we have decided with React to make our BrightSpace Bot website. This is because there are a lot of online resources to make interactive websites with React and so it will be easy to pick up. Not only does it provide stable code, but it also allows for the best performance and ensures faster reading. There are also reusable components in React so our code will be more optimized.

4. Design Detail

Class Design



Description of Classes and Models

Our classes were based on the tasks that we wanted BrightSpace Bot to provide for the user: automation, notifications, and requests. These generic tasks are then broken up into smaller sub-classes that are specific to the requests or actions they deal with. Further detail of the classes are as follows:

I/O:

- This class will be processing the input given by the client that is provided either through Discord or Email

Trigger Handler:

- This class will be the one handling any updates, announcements, or deadlines created on a BrightSpace course

User:

- Stores users information that can be accessed from the UserInfo Database
- Quick access to commonly used values stored from the database

Authentication:

- Handles methods to authenticate the user using BoilerKey
- Gets Authentication Data from services and provides it to BrightSpace in order to have the bot be able to access data from user's courses
- Validates a user based on the credentials provided

Chat Model:

- Parses all requests a user types through Discord Chat or Email
- Basic trained model to be able to understand phrases the user types based on keywords and context
- Classifies the task and data needed that the user is requesting

Automation:

- A general class that handles all the automated tasks that BrightSpace Bot offers
- Interacts with File and Calendar class to:
    - Download a file based on the location the user has specified in their configuration settings
    - Mark certain content sections as "Read" on BrightSpace
    - Automatically add events to the calendar that has been specified in configuration settings

Request:

- Handles general requests made by the user that does not fall under automated tasks
- Interacts with ClassPeers, Course, FocusTime, Event, and Calendar class to:
    - List upcoming assignments, quizzes, and deadlines that the student should be aware of
    - Provide the user with their current grade for a specified course
    - Provide the user with any feedback for a specified assignment
    - Suggest study groups based on the desired study group size and mutual classes shared among peers

Notification:

- Provides proactive notifications to the user based on their notification hours preferences
- Based on the current context and upcoming deadlines, provides 24 hours advance notice and reminders
- Updates users of announcements and instructor have made on a course

Course:

- Represents a course on BrightSpace a user is enrolled in
- Used to specify which course they want information for

ClassPeers:

- Class used to read through class roster on the specified course
- Based on courses the user is enrolled in, study groups are suggested

- Can filter through class list to search for a specific student

FocusTime:

- Class dedicated to analyzing current grades and unfinished assignments to prioritize study time

File:

- Model to represent content posted to BrightSpace

Calendar:

- Model to hold a list of events a calendar will showcase

Event:

- Data class to hold information about deadlines a course has provided

Database Models

In our current architecture, we are utilizing two database structures. One database will be used to store User Information and the other will be used for Configuration Settings. Each entry in the User Information database will correlate to an entry in the Configuration Settings linked by User ID numbers. An overview of the models are shown below:

User Information:

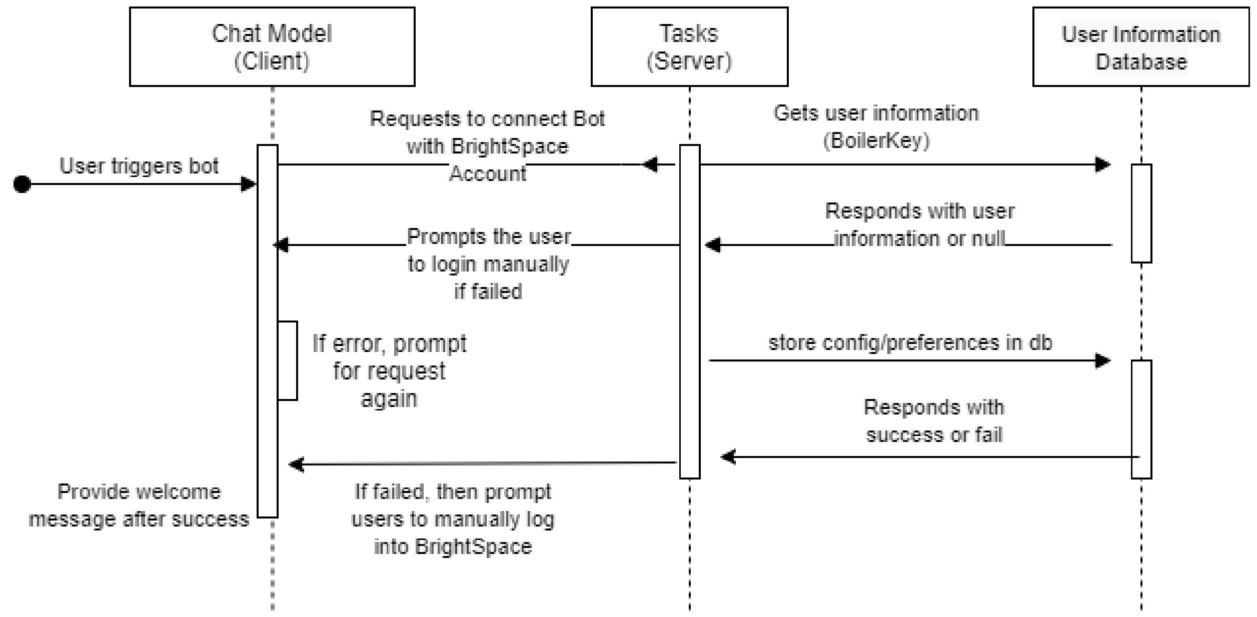
- User ID
- Fullname
- Email
- Password/Pin
- Platform Type

Configuration Settings:

- User ID
- Preferred Notification Time
- Storage Path (for downloading files to)
- Calendar Location
- Bot Preferred Name

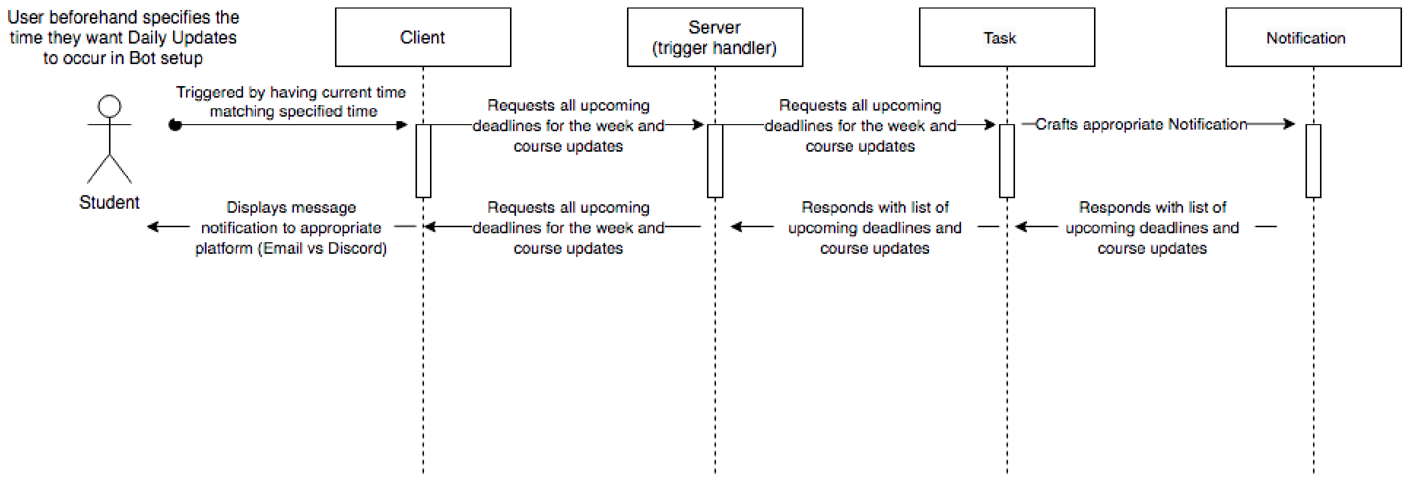
Sequences of the process

User Authentication


Chat Model (Client)
Tasks (Server)
User Information Database
User triggers bot
Requests to connect Bot with BrightSpace Account
Gets user information (BoilerKey)
Responds with user information or null
Prompts the user to login manually if failed
If error, prompt for request again
store config/preferences in db
Responds with success or fail
If failed, then prompt users to manually log into BrightSpace
Provide welcome message after success

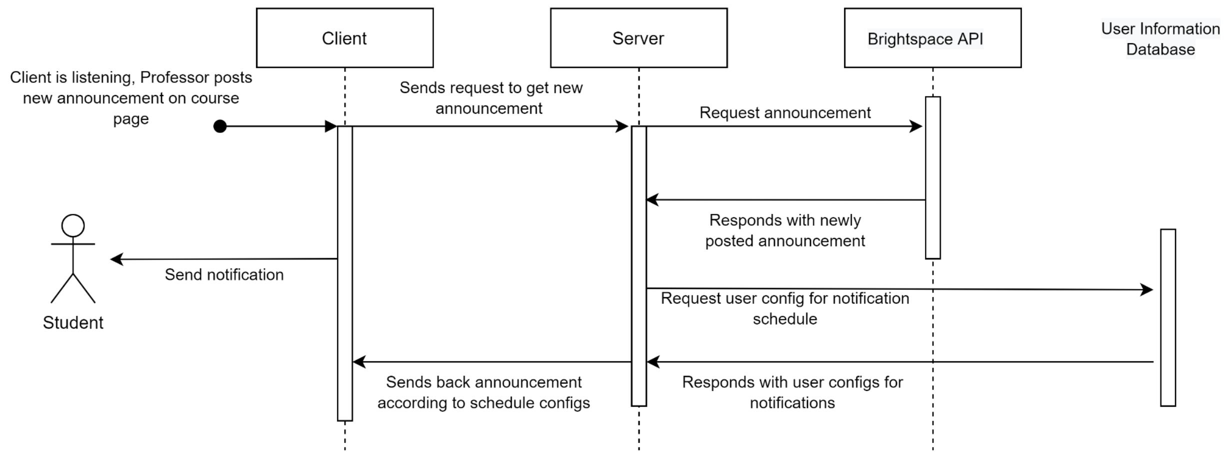
When the user first triggers the Bot, the chat model (client) will request the Brightspace Bot for connection. With that request, it will trigger the task to dictate what kind of service the bot should execute. Once the task is decided, then the User Information Database will be accessed to get the appropriate username and password pin for the bot owner. The database will either respond with the user information or with a null value if any errors are produced. If an error was produced, the user will be prompted for another request. If it fails again, then the user will be asked to manually log into BrightSpace. Once authentication succeeds, then a welcome/success message is provided to the user.

Notifications: Daily Updates


User beforehand specifies the time they want Daily Updates to occur in Bot setup
Client
Server (trigger handler)
Task
Notification
Student
Triggered by having current time matching specified time
Requests all upcoming deadlines for the week and course updates
Requests all upcoming deadlines for the week and course updates
Crafts appropriate Notification
Displays message notification to appropriate platform (Email vs Discord)
Requests all upcoming deadlines for the week and course updates
Responds with list of upcoming deadlines and course updates
Responds with list of upcoming deadlines and course updates

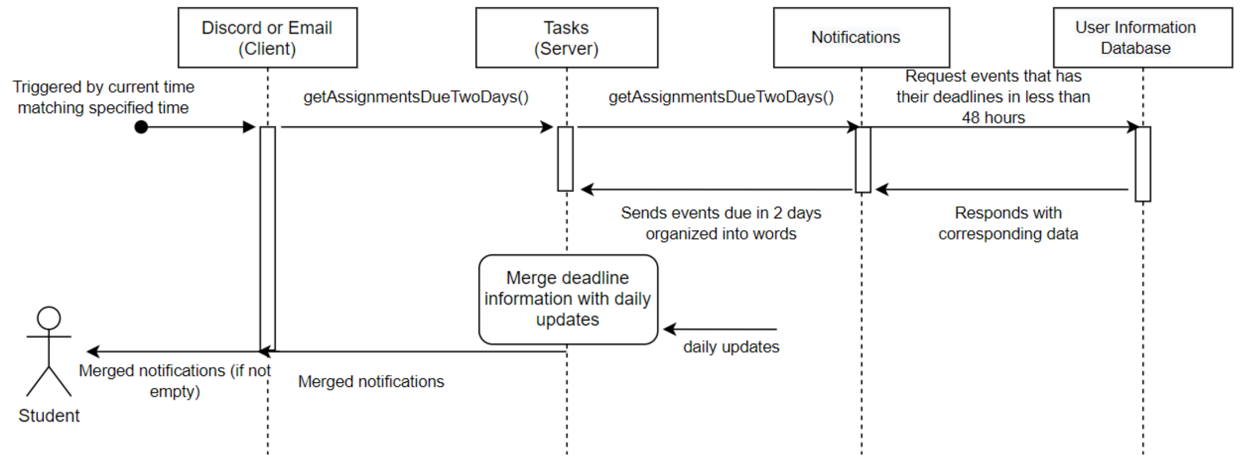
When setting up the bot, ahead of time, the user will be prompted to display the specific time they want to get notified for Daily Updates. Then, when the current time matches the specified time, the bot will be triggered and the client will request all upcoming deadlines for the week and any course updates that have happened today. That request will be passed from the server side, and will be given to the Task class that parses the request and determines what type of action the request is: Automation, Request, or Notification. Based on the trigger, this scenario depicts a Notification Task, so the request gets passed to the Notification Class that crafts the appropriate notification for the user. Then that message is passed all the way over to the client side, and the appropriate Daily Notification is displayed on the appropriate platform (Email or Discord) to the user.

Notifications: Announcements


Client
Server
Brightspace API
User Information Database
Client is listening, Professor posts new announcement on course page
Sends request to get new announcement
Request announcement
Responds with newly posted announcement
Student
Send notification
Request user config for notification schedule
Sends back announcement according to schedule configs
Responds with user configs for notifications

One common notification that Brightspace users have to deal with is instructor announcements. Throughout the semester, the professor will create announcements in Brightspace that often times can be missed by the student in the midst of other coursework. In our system, the client (bot) will be listening for new announcements being posted on the student's course pages. When an instructor posts an announcement, the sequence will be triggered wherein the client first sends a request to receive the announcement to the server. The server will then make Brightspace API calls to grab the announcement. The server makes a request to the user information database to grab the student's configurations for notifications. These configurations have been set beforehand. The server, having received these configurations, will send the announcement as a notification either immediately or on a schedule set by the student. As a notification, the server will work with the Notification class. Thus, the announcement will eventually be sent to the student on a timely basis.

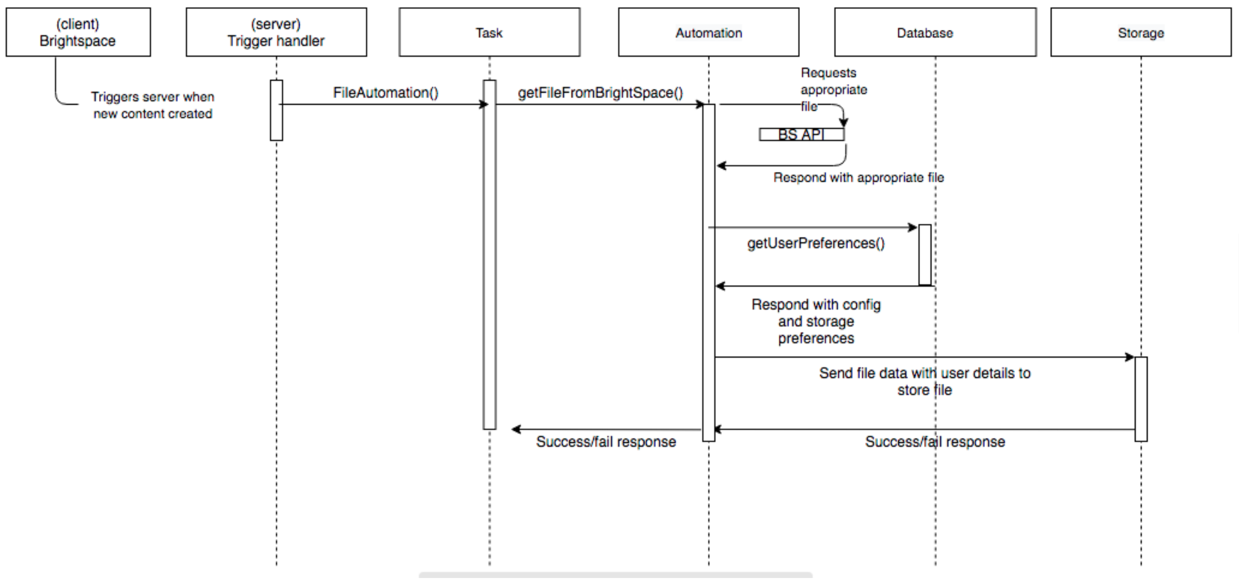
Notifications: 48 hour upcoming deadline


Discord or Email (Client)
Tasks (Server)
Notifications
User Information Database
Triggered by current time matching specified time
getAssignmentsDueTwoDays()
getAssignmentsDueTwoDays()
Request events that has their deadlines in less than 48 hours
Sends events due in 2 days organized into words
Responds with corresponding data
Merge deadline information with daily updates
daily updates
Merged notifications (if not empty)
Merged notifications
Student

One of the core functionalities of our bot is to cut down the amount of notifications and show useful information from them to students in an efficient way. Specifically, the bot is triggered to show one organized notification containing everything happening on BrightSpace each day. However, when there are assignments being due in the near future, we want our users to know about this information at least 2 days ahead, giving them sufficient time to work through their assignments.

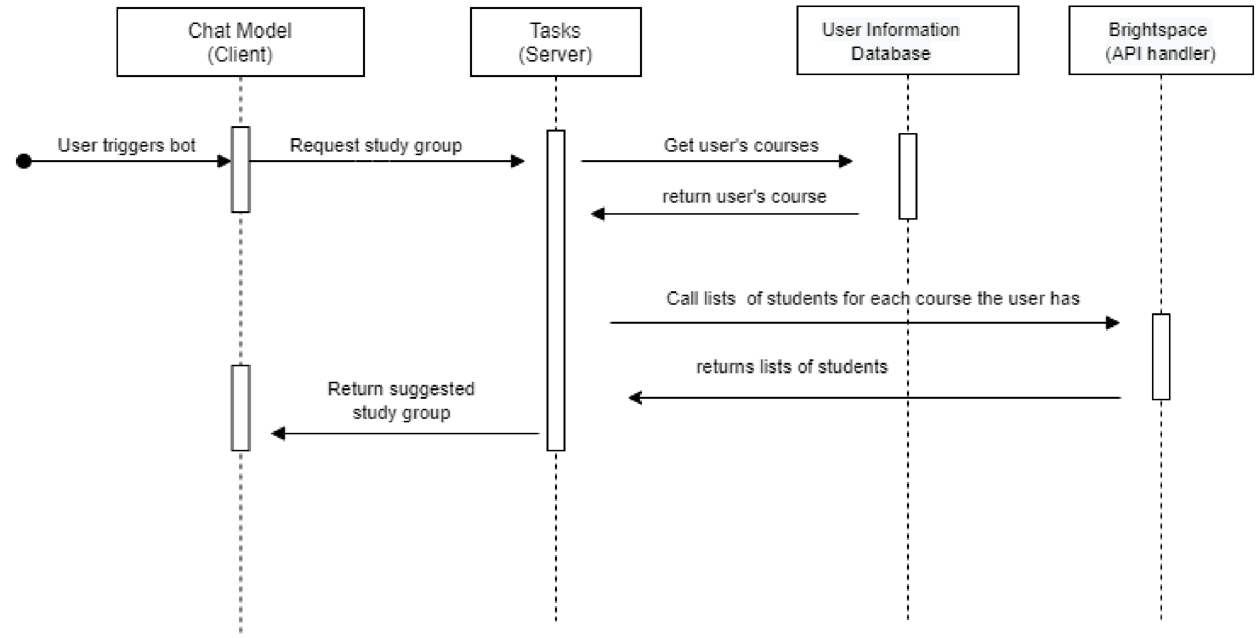
When the daily update is triggered, besides notifications and upcoming deadlines, we would include an extra section in our one notification sent, to notify the users about deadlines that are less than 48 hours ahead. The process to obtain data from the dataset is similar to that of regular daily updates. However, instead of sending another notification to the user, the bot would merge deadline information with other regular ones.

Automation: Downloading files


(client) Brightspace
(server) Trigger handler
Task
Automation
Database
Storage
Triggers server when new content created
FileAutomation()
getFileFromBrightSpace()
Requests appropriate file
BS API
Respond with appropriate file
getUserPreferences()
Respond with config and storage preferences
Send file data with user details to store file
Success/fail response
Success/fail response

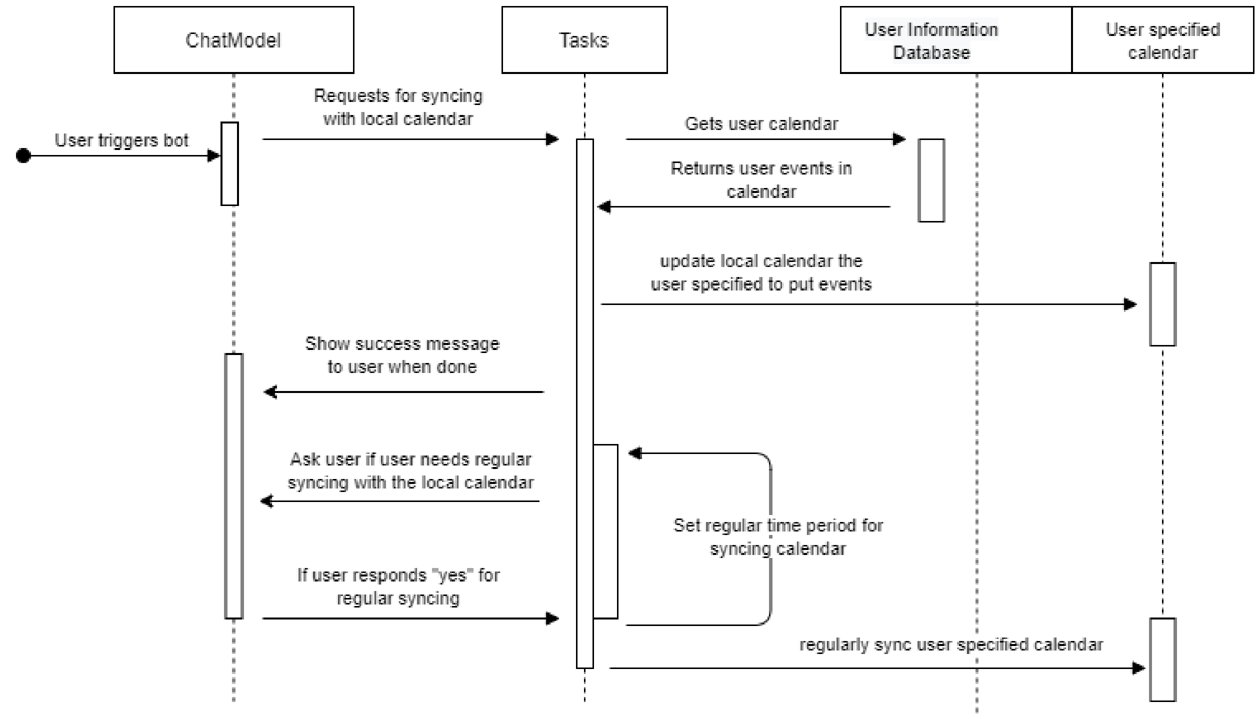
When a professor updates their BrightSpace page with new files, that will trigger the bot. The bot then sends an API call to BrightSpace to request the file. If successful, the BrightSpace API will send the appropriate file data back to our backend code. The backend will then make a call to our database to retrieve the user's configuration and storage preferences such as where to store the file and which folder. With both the user config information and the file, the backend can now make an API call to the specified storage method (such as Google Drive) to upload that file to the desired folder.

Automation: Suggesting study groups


Chat Model (Client)
Tasks (Server)
User Information Database
Brightspace (API handler)
User triggers bot
Request study group
Get user's courses
return user's course
Call lists of students for each course the user has
returns lists of students
Return suggested study group

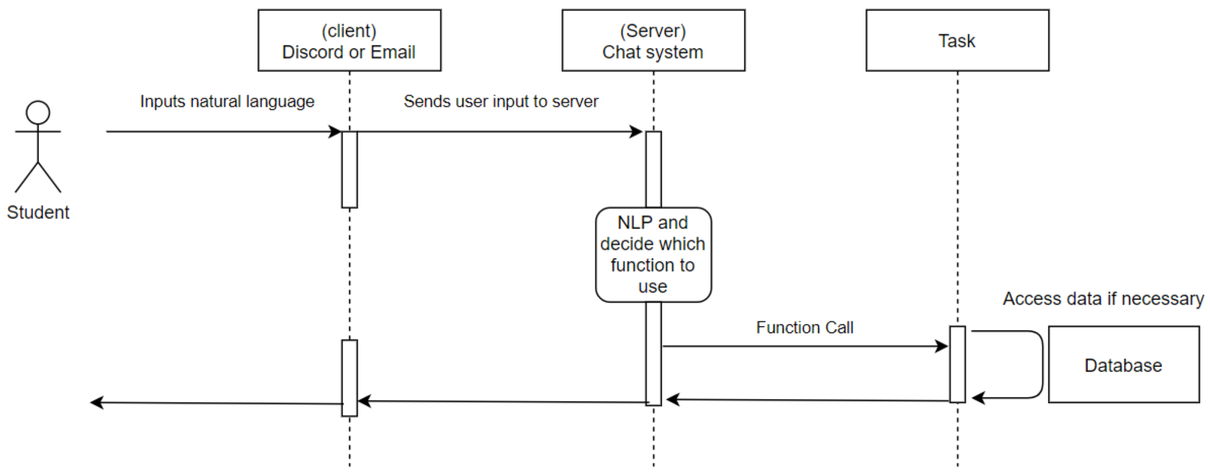
When the user requests to search for common study groups, the chat model will process the request to the server. Then from the serve,r it will call user's course data from the database. It will then call the Brightspace API and return all the student list with common courses. Then the server will arrange other users with the most common courses and suggest to the user.

Automation: Synchronize calendar & update new events


ChatModel
Tasks
User Information Database
User specified calendar
User triggers bot
Requests for syncing with local calendar
Gets user calendar
Returns user events in calendar
update local calendar the user specified to put events
Show success message to user when done
Ask user if user needs regular syncing with the local calendar
Set regular time period for syncing calendar
If user responds "yes" for regular syncing
regularly sync user specified calendar

When requesting the bot for synchronization to a local calendar, the request will go through the chat model. In the chat model, it will trigger the appropriate task to the server. Then from the server it will get the user's calendar from the user info database. The calendar stored in the database is a arraylist of events sorted by time. For instance, the first element of the arrayList will the most first event of student's schedule. From the database, the events are retrieved. In the server, the task will update it to the user specified calendar to update the events to the local calendar. After the task is done, it will show a success message to the user through the chatmodel. Then it will ask whether the user wants the calender to be regularly sync rather than requesting to synchronize the calendar every time. If the user demands to synchronize the calendar regularly, inside the server a time period to automatically update the calendar will be set. After then at the specified period the local calendar will by synchronized with the upcoming events in Brightspace. For the case where the user does not requires regularly synchronization, the chat model will finish the calendar task.

Requests:



Our bot takes requests in the form of natural languages, and decides which functionality the user is intended to use by parsing the users' input.
The class Chat System is responsible for natural language processing for our bot. All incoming requests from the students will be sent to this class before moving forward to the actual tasks. We will train and apply a machine learning model using one hot encoding of functionalities. The training process is done separately from the main program and the model will only be inferencing in our program.
After the chat system decides which functionality to use, it will call the corresponding function in class Task. Afterwards, Task will access the database if necessary, and return responds to the user.