WolfPlanner 2.0

Abhishek Bhardwaj
North Carolina State University
abhardw3@ncsu.edu

Madhu Vamsi Kalyan Machavarapu
North Carolina State University
mmachav@ncsu.edu

Sanya Kathuria
North Carolina State University
skathur2@ncsu.edu

Shivam Chamoli
North Carolina State University
schamol@ncsu.edu

ABSTRACT

With the number of tasks increasing day-by-day, it is becoming difficult for students to manage so many tasks considering the varying amount of time available and required to complete them. Students have to manage many tasks such as attend classes, complete assignments before deadlines, work on projects, attend meetings and prepare for interviews. With such a large number of tasks lined up for the student to do in a fixed time frame, it becomes very hard for the student to come up with an ideal time table which would help the student complete all the activities and also maintain its quality to an optimum level. It requires continuous and active effort by the student to create weekly schedules leading to wastage of valuable time. Manual effort in scheduling might lead to poor planning which in turn will result in excessive workloads.

In today's age where time is a critical asset for a student's success, we have come with an idea for replacing and reducing human effort by a bot which generates a personalized schedule for the student. The bot makes a schedule for the students taking into account their deadlines, breaks and dependencies between tasks. WolfPlanner, an interactive chatbot specifically made for NC State students that generates a schedule for the student based on upcoming assignments, projects, interviews, events, etc (information about which will be given to the system by the student). Furthermore, to keep up with the current technology trend of interactive assistants, we have integrated this service with Slack thereby avoiding downloading of stand-alone application. For now, we restrict our domain of users to students for ease of data collection and implementation. In the existing system, we have faced numerous challenges, including usability, multi-agent coordination, scalable constraint reasoning, robust execution, and unobtrusive learning. Our research advances basic solutions to the fundamental problems; however, integrating Personal task scheduler into a deployed system has raised other important issues for the successful adoption of new technology. As a personal assistant, it must integrate easily into a user's real environment, support their normal workflow, respect their authority and privacy, provide natural user interfaces, and handle the issues that arise with deploying such a system in an open environment. To mediate that, we have come up with various improvements in the previous team's scheduling algorithm.

Keywords

Scheduler, Chatbot, Calendar, Time-table, Planner, Slack, Meeting Scheduler,

1. INTRODUCTION

In this paper, we discuss WolfPlanner - a personalized task scheduling bot for students, which generates a weekly schedule of tasks and guarantees completion of tasks within the respective deadlines without compromising on the quality of work. In order to get the best out of their education, it is important that students focus on important tasks such as their coursework and studies and spend less time on wasteful tasks like creating a time-table. The best way forward is to automate such tasks and WolfPlanner does exactly the same. Today, it is a world of personal interactive assistants and according to Gartner's report[1], 85% of customer interactions will be managed without a human by 2020. So we have developed a chatbot that is able to assist the user in scheduling a wide of range of day-to-day tasks (for a student) which includes classes, assignments, project meetings, discussions, readings to name a few. The application is a chatbot interface which will take in all of these conditions as inputs from the students to help them manage their time efficiently. While generating the task schedule, we have also taken care of time spent in activities like sleeping, eating, buffer time between two tasks such as commuting time from classroom to the destination where a project meet is scheduled and ignoring meagre chunks of time for major tasks. The bot will also give the student a free time slot in which the student can do any other activity which he wishes to do. In universities, students are involved in many group activities, and scheduling these meetups are often challenging due to conflicting schedules. They take up a lot of time and lead to hassles. So, having a collaborative approach is needed.

X-XXXXX-XX-X/XX/XX.
DOI:

The bot is of a great help as it has the data of the weekly schedule of all its users. The schedule generated will help the students come up with a suitable time for arranging meetings. All user will know their free slots and can agree on a common time accordingly.
The motive of our work was to develop a chatbot with which a user can communicate, provide data, ask for change in schedule, etc. thus saving time for schedule generation. A chatbot feature also guarantees ease of use of application.The bot generates schedules on a weekly or biweekly basis as per user preference. The user requests a set of tasks every week and the priority is set according to deadline. As per the user preference, the bot will generate an appropriate schedule taking into account all of the details. Further, this generated schedule may be exported to Google Calendar for easy access across all the devices. We have hosted our database on Mlab.com where the user data is stored on MongoDB. Mlab provides scalability and reliability to our system, and security to user's data. The user details are stored as objects on the NoSQL MongoDB environment making it convenient to access the details of the user who is logged in without any requirement of costly join operations which would have been needed in case of relational databases. Also storing user data in tables might lead to redundancy and lot of NULL data. Hence, we decided to use MongoDB as our database system. The advanced version of WolfPlanner works with its user and also keeps into account other users, to schedule meetings and commitments in its user's calendar. It can negotiate and provide the users with useful information about upcoming events while protecting her privacy, and learn about their prioritization preferences. To achieve these goals, we plan to integrate commercially available calendaring tools and project meetings , goal directed process management, and preference learning.

2. MOTIVATION

A major problem in student's life is managing homeworks, projects, interviews, meetings etc. and still get some free time. Our study revealed that majority of survey respondents (87%) want an application/bot which generates a schedule for them. However, a lot of people (67%) don't use any calendar for organizing their day. This shows that people wanted an application for a functionality which other calendars do not provide. Hence we came up with the idea to fulfil the user's requirements by creating an interactive bot which generates schedules for the user and also organizes meeting times for them. This will help the users manage their time efficiently and work productively. Moreover, we will be extending it to include priorities of tasks, breaks and neglection of small chunks of free time for tedious tasks. Furthermore, we will be extending the scheduling algorithm to accommodate these changes.
We can point to two main reasons why we need to change the existing system. The first is a failure to take into account the intensely personal nature of the human time management problem. Most individuals, even the busiest knowledge workers, are reluctant to relinquish control over the management of their own time. Further, people have different preferences and practices regarding how they schedule their time, how they negotiate appointments with others, and how much information they are willing to share when doing so. To be successful, a time management solution must thus address these basic issues of authority, privacy, and preferences. These type of features are not handled as of now. The second reason is that adoption of such a system is contingent on its practical usability including important user interface issues within a real-world deployment.
Despite the proliferation of calendar tools to organize, display, and track commitments, most individuals still expend a considerable amount of effort managing their schedules. The research community has primarily focused on automated meeting scheduling: finding feasible time slots for meetings given a set of requirements on participants, times, and locations.

3. LITERATURE REVIEW

Despite the success of calendar tools in organizing and tracking activities, a lot of time is spent by people on managing their time properly. Berry et al.[2] came up with PTime, a personalized time management agent which reduces this problem of time management considerably, when it comes to arranging meetings. PTime schedules meetings based on factors like participants, time and locations. Our bot will go a step further and automatically schedule each and every task of a user along with the meetings. The biggest challenges in any task planner are the constraints in schedules and which scheduling method is used.
1. Constraints in scheduling
2. Preemptive tasks
3. Penalty for errors in scheduling

Traditional algorithms face a lot of problems when it comes to preemptive scheduling. For example, schedule quality of FCFS scheduling algorithm is highly sensitive to the preemption strategy used. Schwiegelshohn et al.[3] demonstrated that by doing certain improvements in FCFS it can be better than any of the other preemptive algorithms but the parameter settings and efficiency depends heavily on the workload. As our task scheduler deals with multiple tasks which can arrive at any time, using FCFS might lead to compromising efficient task scheduling as our workload itself is dynamic in nature. After reviewing some papers we found that the algorithm implemented by L. J. Wilkerson et al.[4] will be an effective algorithm for our task scheduler. We then compared this algorithm with First Fit algorithm[5] and found that First Fit Algorithm will be quite suitable too. For the Improvement, we are planning to execute the Multi-Level Feedback Queue scheduling algorithm along with the necessary accommodations for priorities of tasks, breaks and extension.
For the Improvement, we are planning to execute the Modified Multi-Level Feedback Queue scheduling algorithm along with the necessary accommodations for priorities of tasks, breaks and extension to a weekly schedule.

3.1 IMPROVEMENTS

3.1.1 Existing System

The current system has a chatbot with which a user can communicate, provide data, ask for change in schedule, etc. thus saving time for schedule generation. A chatbot feature also guarantees ease of use of application.The bot generates schedules on a daily as per user preference. The user requests a set of tasks every week and the priority is set according to deadline. As per the user preference, the bot will generate an appropriate schedule taking into account all of the details. Further, this generated schedule may be

exported to Google Calendar for easy access across all the devices.
The database on Mlab.com where the user data is stored on MongoDB. Mlab provides scalability and reliability to our system, and security to user's data. The user details are stored as objects on the NoSQL MongoDB environment making it convenient to access the details of the user who is logged in without any requirement of costly join operations which would have been needed in case of relational databases. Also storing user data in tables might lead to redundancy and lot of NULL data. Hence, we decided to use MongoDB as our database system.
The advanced version of WolfPlanner works with its user and also keeps into account other users, to schedule meetings and commitments in its user's calendar. It can negotiate and provide the users with useful information about upcoming events while protecting his/her privacy, and learn about their prioritization/ preferences. To achieve these goals, we plan to integrate commercially available calendaring tools and project meetings , goal directed process management, and preference learning.

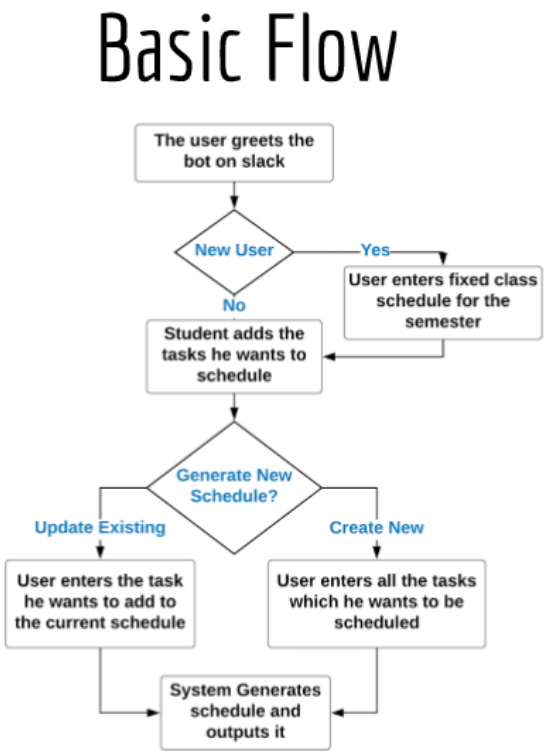

Figure 1: Flow of the system

3.1.2 *Correct the implementation of daily scheduling*

Previously,the only scheduling implemented is for daily scheduling, which does not schedule properly either. After testing the application thoroughly, we found out that a 10 hours schedule + 8 hours schedule in the same day is not scheduled correctly. Moreover, the user is allowed to create schedules for a date that has passed. We have impemented a scheduling algorithm which accomodates a weekly scheduling keeping in mind priority and deadline of the task.

Figure 2: Schedule given by a student, who has 1 submission the next day and 4 submissions a week later

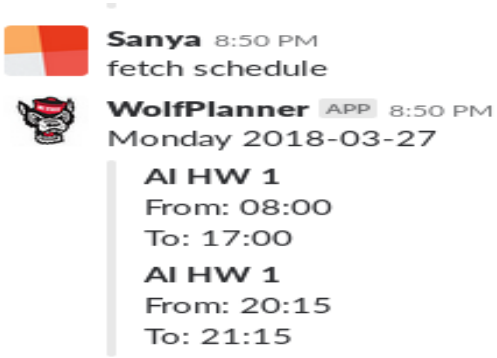

Figure 3: Output of WolfPlanner according to the present system

3.1.3 *Adding the Project meetings*

The bot has the data of the weekly schedule of all its users. So we wanted to leverage and suggest a common free slot so that a meeting can be organized.It also includes the meeting venue and time needed to reach that venue. If a group has to meet to work on a particular project, this feature will give the group members an idea to plan the stages of the project as they know beforehand when the meetings can be scheduled.

Days/Time	12-2 AM	2-4 AM	4-6 AM	6-8 AM	8-10 AM	10-12 PM	2-4 PM	4-6 PM	6-8 PM	8-10 PM	10-12 PM
Mon 05/03					class	Spatial	Lunch	Spatial + break	Spatial + break		
Tues 06/03						SE	Lunch	DevOps + break	SE+ break	Devops+break	
Wed 07/03					class	class	Lunch	DevOps + break	SE+break		
Thurs 08/03						DevOps	Lunch	Maths +break	AI+break	SE+break	
Fri 09/03					class	class	Lunch	SE+ break	DevOps+ break	Maths +break	AI+ break
Sat 10/03											
Sun 11/03											

Last two days are for ad-hoc events. Break signifies a break of 30 minutes i.e 15 minutes per hour

Figure 4: sample schedule which we plan to implement

3.2 Improve the present scheduling algorithm

By taking into consideration the functionalities that were being leveraged in WolfPlanner 1.0, we have attempted to improve the existing scheduling algorithm keeping in mind the folliwng parameters:

1. Priorities of task .
2. Breaks in between tedious tasks.
3. Neglecting small chunks of free time for tedious tasks.

3.3 User Survey

In this section, we have described analysis of results for different questions that we asked while taking the survey to understand the utility of implementing the better version of existing time scheduling WolfPlanner and whether they need improvements such a product or not. The questions which we asked are as follows:

1. Do you face difficulties in completing tasks because of not being able to manage time properly? Figure 5: Response for Question 1

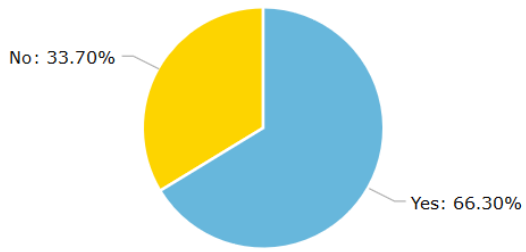

Figure 5: Difficulty due to time management

2. Do you face an issue to set up project (or other) meetings due to conflicting schedules?
   The purpose of this question so that we could evaluate the need of collaborative scheduling methods. The results suggest that people do face the issue of setting up meetings as their schedules conflict. Hence, indicating a need of a collaborative personal scheduler.
   Figure 6: Response for Question 2

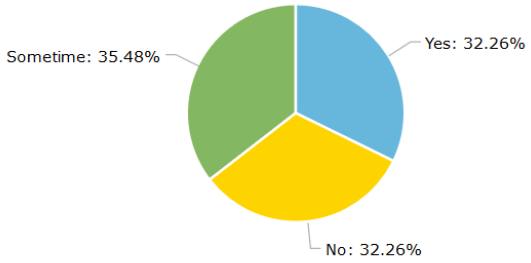

Figure 6: Issue in setting up project meeting due to conflicting schedules

3. How much do you rely on Google Calendar (or similar) for planning your day-to-day tasks?
   This question was to understand the percentage of people depending on Calendar applications for daily use. This result tell us that people do not rely on calendar applications for planning their activities.But the following question (Q4) tells us that they do want a software which makes schedule for them.

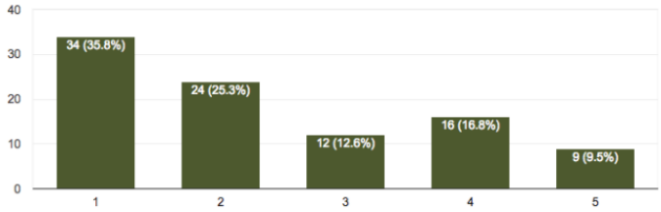

Figure 7: How much do you rely on google calender

4. Do you feel the need for a software which makes a weekly schedule for you taking into account your pending tasks and preferences?
   The results of this question showed us that more than three-fourth of the population need an application that helps the user. The purpose of this question was to understand if the people face difficulties in time management. From the survey, the results tells us that there are not many people who do not experience the problem of task completion due to ineffective time management, so our bot might help to solve this problem.
   Figure 8: Response for Question 4

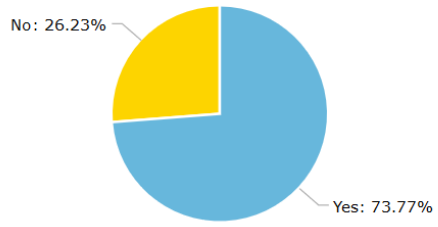

Figure 8: Need of software for weekly schedule

5. Do you think WolfPlanner will be more user friendly if it accommodates features like including regular breaks, project meetings and prioritization of tasks? This question was asked to understand the satisfaction level of people with the existing time scheduling and their interest in a more flexible scheduling algorithm. From the survey. The results showed that people will be more comfortable if we accommodate features like that since the bot becomes more practical to use.We also got few feedbacks saying it will be more real to accommodate breaks.
   Figure 9: Response for Question 5.

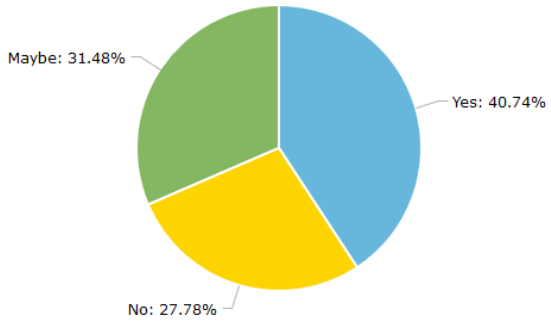

Figure 9: Do you think WolfPlanner will be more user friendly if it accommodates features like including regular breaks, project meetings and prioritization of tasks

Survey Conclusion Based on the 92 responses we got from our survey we can conclude that a personalized time management bot will be very helpful for the people. Most of the respondents do face a problem of completing tasks efficiently due to lack of proper time management. Using a bot which will schedule tasks for the individual will help with proper management of tasks and also save the user's time as there is no need to schedule tasks manually. The bot will also take care of meeting scheduling which is hassle for some of the respondents. The survey tells us that not many people rely on calendar applications like Google Calendar for task scheduling but would like to use a software like our bot. From this result we can conclude that people will start using softwares similar to calendar applications for task scheduling if there are more important features like dynamic schedule generation.

4. IMPLEMENTATION

4.1 Comparative Study

The main improvements in WolfPlanner 2.0 over the previous WolfPlanner 1.0 include the feature of generating a weekly schedule on the basis of priority. For Example: When the similar kind of tasks were added for both the applications. WolfPlanner is not able to generate a weekly planner but only a daily plan, (As shown in figure 3), Whereas WolfPlanner 2.0 is abe to generate schedule for the week. It also assigns AI HW2 before AI HW3 as HW2 has more priority over HW3 and both ave same deadlines. (As shown in figure 20). As you can observe it also accomodates tasks like Badminton and breaks which makes it a much more flexible and user-friendly application.

4.2 Software engineering methodologies

4.2.1 Agile Methodology

We plan to use Agile methodology for development of our application.So the features were built initially as small incremental features, and then the final one was built on top of those.

4.2.2 Code Review

Code Review practices were followed by creating different branches in github other than master and working on those before pushing on master.In this way, we managed to reduce bugs and error in our code and code was reviewed by the members before commiting to the master. Going forward, we will do the same.

4.3 Working Of the Bot

1. Authenticate the slack user using the OAuth handler of Botkit and Slack
   (a) When a user starts communication with the bot, the user is first authenticated. Since, we have used the BotKit toolkit, it provides a handler which helps us authenticate the user. 1.2 Slack sends a OAuth request oauth which is caught at our server and the user is verified.
2. For every new user create a new document in the MongoDB MLab collection.
   (a) When the user enters on the Slack interface, Slack sends POST to the server at slack/receive. These are events (message.im and message.channel) which the bot has subscribed to. So whenever there is a message posted our server is notified. 2.2 The user is checked if an entry is already present in the MLab student collection. If the user if present then the user details for the further steps. 2.3 If the user document is not present, then create one. Initial entry into the user document is the user's slackID and the user's unityID.
3. Get the user input(either Tasks or Courses) and store them into the user's Mlab entry

(a) When the user requests to add tasks/courses, Slack sends a HTTP POST to our server at message. These are classified as Interactive Messages. 3.2 The bot sends a request to open up a dialog using the dialog.open Slack API. 3.3 The user fills up the necessary details and this is received by the bot controller. The bot connects to the MLab using mongoose, and the document is updated.

4. When 'fetch schedule' command is invoked, the bot sends the user identification details to the scheduling algorithm.

   (a) When the user requests for fetch schedule, the bot fetches the user's unityID and generates a day-date dictionary which is the current week's dates(according to which the schedule is to be generated)

   (b) Now, the bot uses the python-shell package to call the core-scheduling algorithm, passing the above variables as arguments to it. The algorithm working is described below. The output of the algorithm is a generated schedule which is stored in the respective user document. The schedule is generated both on the basis of priority an deadline. The user can fetch a weekly planner by adding his/her tasks using the bot. The free time is generated for each day and then taska are planned such that free time is filled with these tasks along with keeping in mind other activities that a human may do like taking breaks and extra-cirricular activities, etc.

   (c) The bot displays the generated schedule on the Slack Interface.

5. Once the schedule is generated, the bot fetches the schedule from the MLab user document. This fetched scheduled is formatted in a way which is easily understandable to the user and the bot dis- plays this formatted schedule.

4.4 Scheduling Algorithm

The fundamental problems that MLFQ tries to resolve are two-fold. Firstly, It works to improve the 'turnaround' time (amount of time taken to finish the task). In our case that is really important because once the user starts working on a task he would like to finish that task within some sensible timeframe and not let it extend for a lifetime. Secondly, It wants the user to feel that the system is responsive (time taken by the system to schedule the job), in our context that means that the user wants to start working on tasks as soon as possible, when he is updating the list of tasks at hand. Moreover, everyone wants a break from the monotony of doing the same task and thus, this is where scheduling of multiple tasks as 'Round Robin' would be helpful. All this integrated with the priority system and the accommodations for various things, we believe that it could improve the lives of our peers. The algorithm is as follows :

Rule 1: If Priority(A) > Priority(B), A is higher on the schedule.

Rule 2: If Priority(A) = Priority(B), A & B are scheduled in 'Round Robin' of 2 hours with 15 minutes break in between.

Rule 3: When a job enters the system, it is placed in the appropriate queue (the topmost queue if it is of highest priority).

Rule 4: Once a job uses up its time allotment at a given level, its priority is reduced (i.e., it moves down one queue).

Rule 5: After some time period S, move all the jobs in the system to the topmost queue to avoid 'aging'.

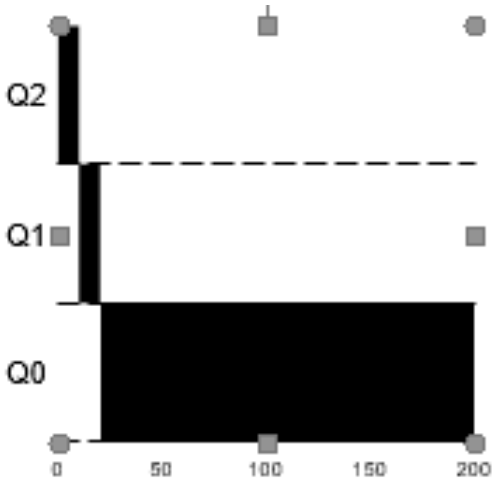

Figure 10: Long running job over time

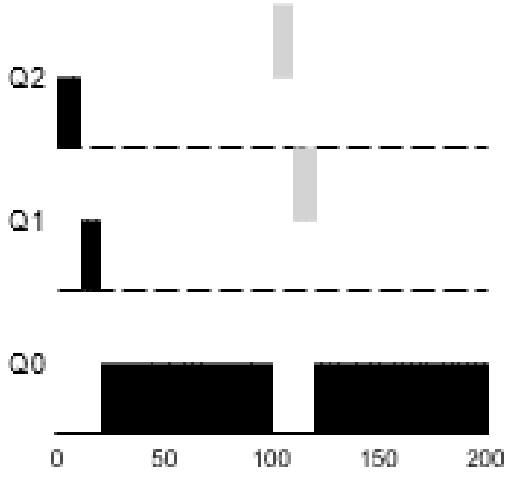

Figure 11: Along came an interactive job

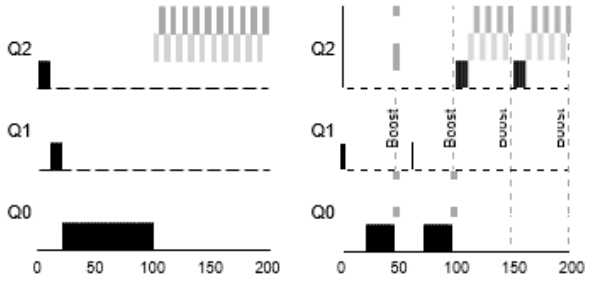

Figure 12: Without priority boost *left* **and with priority boost** *right*

4.4.1 Algorithm Explanation

1. If the user has a long running job, we will be decrementing its priority on the basis of the maximum allowed time limit in each queue level. See Figure 10.

2. Now, if the user wants to schedule another job of the highest priority, the older process is rescheduled and the new process is scheduled. See Figure 11.

3. Suppose a process is running for a long time, now two processes of higher priority come and hog the schedule (aging), see Figure 4 (left). To take care of that we do priority boosting, see Figure 12 (right).

5. WHY IS WOLPLANNER 2.0 BETTER

1. It still performs all the functions that Wolfplanner used to perform i.e manage projects, assignments, homeworks and deadlines.

2. In-comprehensive daily schedule vs priority based weekly schedule: Although the architecture for the application was rock-solid, the previous version of wolfplanner performed a daily schedule (definitely due to a time constraint) which did not adhere to the restrictions provided by the user. The new Wolfplanner takes into account the priority of the tasks and brings into the picture the concept of breaks, which we humans love.

3. Plan a meeting: Is planning a meeting based on the schedules of teammates still a hassle. No more! Wolfplanner finds out the common free times that you have with the user that you want to plan a meeting with. You decide!

4. Parse a course: Want to know what are the various deadlines for a subject? Use this functionality which tells you all the deadlines in the coming week for a particular subject so that you can plan the schedule easily. It works brilliantly for well structured courses (well-defined deadlines) such as our very own Software Engineering, where we know all the deadlines before hand.

6. EXECUTION FLOW

6.1 Coding and Testing Schedule

Initially we took the help of Team M to understand the setup and to setup the environment. While setting up the environment we started writing our python scripts in parallel for parsing a course functionality as this was an independent module and were done with the functionality within a week. When our system was set up we looked through the code, made the design and decided to just create a function being called that would create the weekly schedule based on priorities and call that function instead of the function being called for the daily schedule. We also made changes to the front-end and the database to accommodate priority. We tested our independent functionalities and then integrated the system and performed integration testing on the system.

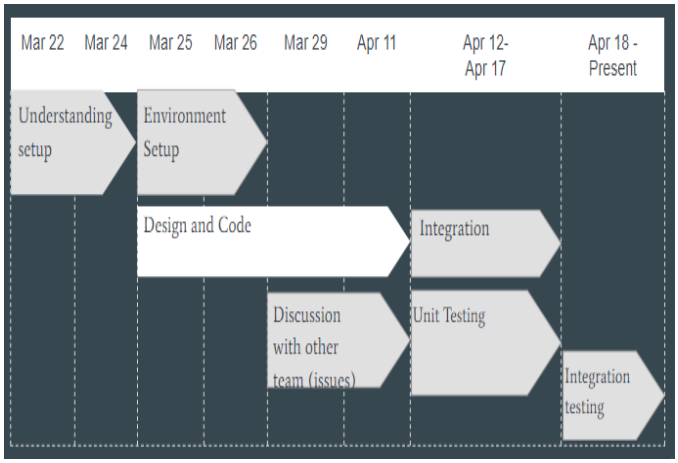

Figure 13: Coding and testing schedule

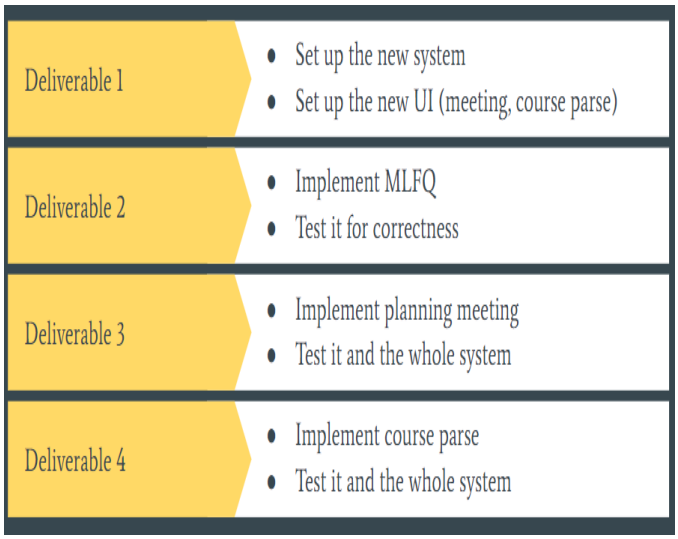

Figure 14: deliverables list

6.2 Features and Application flow

6.2.1 Adding a new user

The bot fetches the details based on the Unity ID mentioned by the user when he/she first uses the application,

so that we do not need to ask him/her to enter the details already available on the university portal. It is easy to retrieve these details by making a request call to the University Campus Directory. If the fetch is not successful then an error is returned indicating that the unity ID is wrong. If the fetch is successful, then the student details are parsed from the HTML Page. The Figure 15 shows when a new user is added and how the user entry is retrieved. It also shows different functionalities that the bot can help you with.

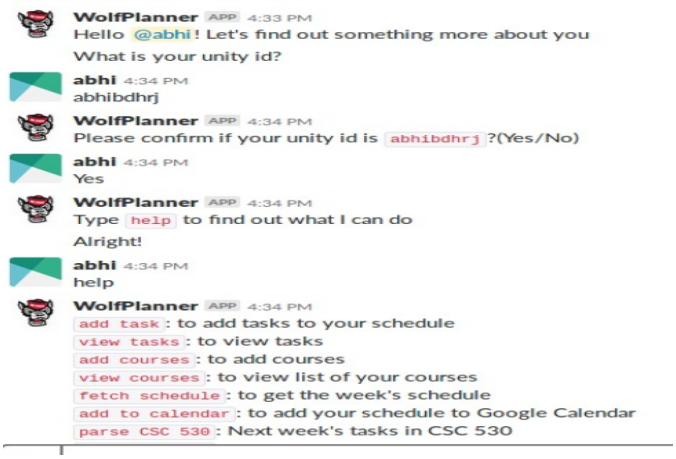

Figure 15: Adding a new user

6.2.2 *Adding courses*

When the user types add courses a new dialog will open up with a form. The student can put the details of the courses from that form and submit it.Details such as course number, course name, days of week and timings can be entered. The added courses are added in the MongoDB database

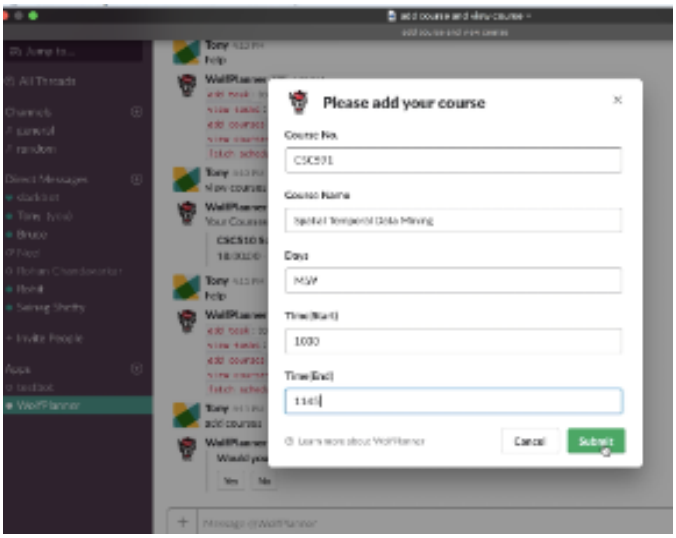

Figure 16: Adding a course

6.2.3 *View courses*

When the user types view courses, all the courses of the user are retrieved from the database. A list of all courses added is displayed along with the days and timings.

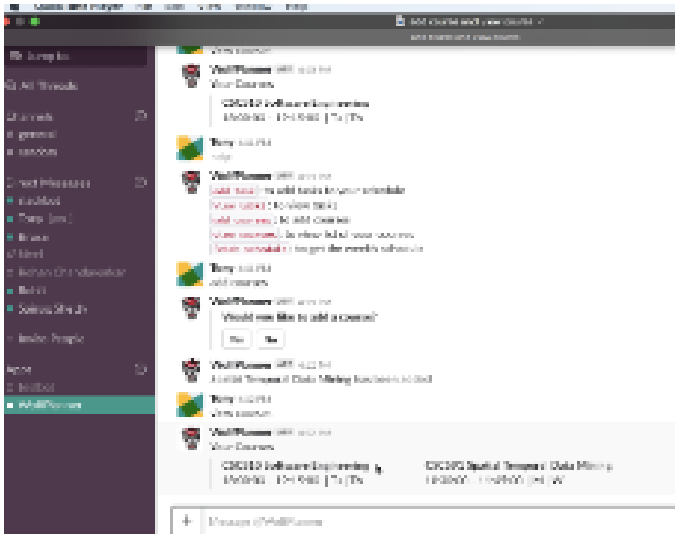

Figure 17: Viewing courses

6.2.4 *Add tasks according to priority*

Similar to WolfPlanner 1.0, when the user types add task a new dialog will open up with a form with an additional priority feature in WolfPlanner 2.0. The student can put the details of the task from that form and submit it.Details such as task name, task type, task duration and task deadline can be entered from that from.The added tasks are added in the MongoDB database.

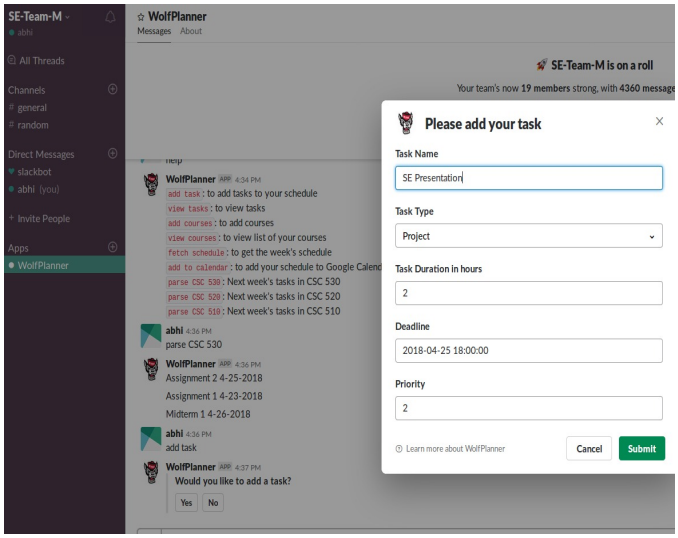

Figure 18: Adding task with priority

6.2.5 *View tasks*

When the user types âĂŹview tasksâĂŹ, all the tasks of the user are retrieved from the database. A list of all tasks added is displayed along with the hours scheduled for the task.

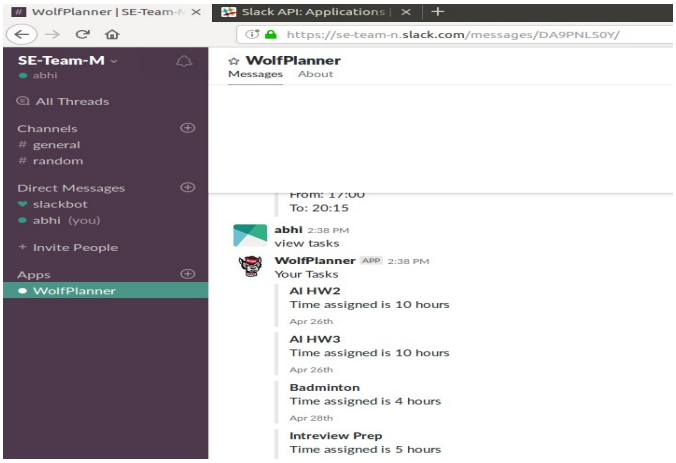

Figure 19: Viewing Tasks

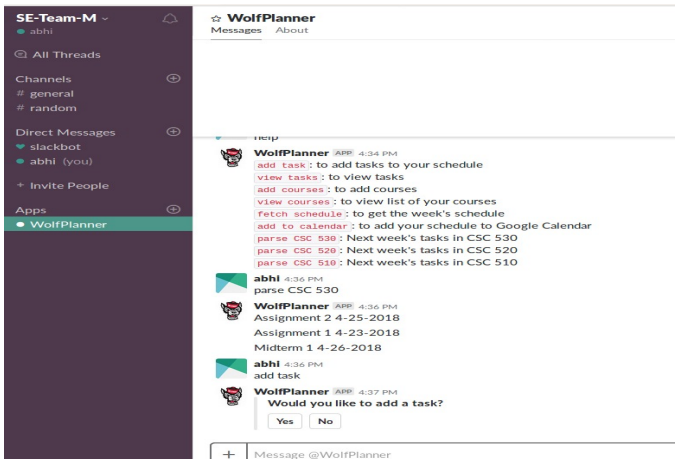

Figure 21: Parsing Text file

6.2.6 *Fetch Schedule*

WolfPlanner 2.0 successfully is able to fetch a weekly schedule for the user depending upon the tasks he/she had added. A weekly planner comprising of list of all tasks added is displayed along with the hours scheduled for the task on the basis of priority. For example, The figure 21 shows the schedule generated based on priority such that AI HW-1 is assigned before AI HW-2 as the priority was higher and deadline date was same. It also accommodates for breaks such as badminton with a lower priority which makes it usage more user-friendly and practical.

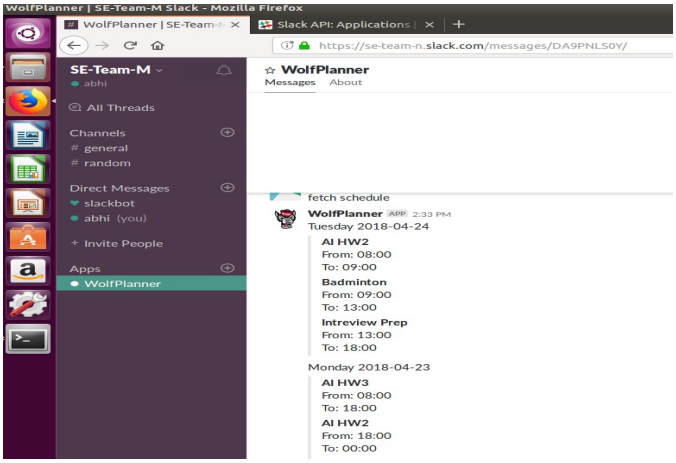

Figure 20: Output Schedule

6.2.7 *Parse Text File to generate schedule*

WolfPlanner 2.0 successfully is able to fetch a schedule for the user by parses the course and returning the tasks for the next 7 days, to help the user to plan the schedule in a better way.

7. EVALUATIONS AND SURVEYS

We will be evaluating the improvised Real-time version of WolfPlanner with traditional, scheduled and on-demand methodologies of task scheduling on the basis of various parameters:

1. Predictability: The traditional methods are less predictability because the whole process is manual and cannot predict all the scenarios of other userâĂŹs availability. The scheduled approach works best for predictability because there is a prior meeting booked so both the users will be able to plan better. For the on-demand methodology, it will highly depend how scheduling algorithm works.

2. Ease of use - Our system is easy to use because of the interactive and easy to understand real time scheduling and interactive bot interface.

3. Scalability - Our system is scalable because of the resources AWS provides.

4. Privacy - Our system ensures privacy of the user because it does not allow a user to access any other userâĂŹs schedule/tasks without the other userâĂŹs permission.

5. Reliability - Our system will not be reliable since no backup is maintained of the data. If the server goes down or due to some system error data is lost, there will be no way to recover the data.

6. The real-time methodology bot will be the most costly as compared to the traditional methodology which does not require any resources for scheduling and involves manual effort mostly.

Table 1: Evaluation of Different Methodology

Methodology	1	2	3	4	5	6
Traditional	-	+	+	++	-	- -
Scheduled	++	++	-	+	+	+
On-Demand	-	- -	++	- -	- -	++
Real-Time	-	++	++	+	++	- -

Figure 22: Evaluation parameters

7.1 User Surveys

1. Time taken to get schedule from wolfplanner 1.0
   We asked this question so as the users have a comparison basis regarding time for wolfplanner 2.0 for comparison.

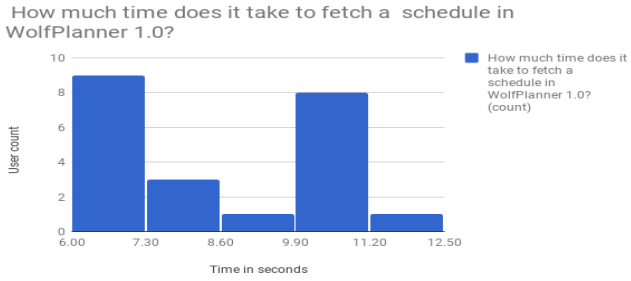

Figure 23: Time taken to get schedule in wolfplanner 1.0

2. How much time does it take to fetch schedule on wolfplanner 2.0
   We asked this question because we wanted the users to actually test the system by giving tasks and verifying that the system generates output. The schedule was coming in under 15 sec for most people.

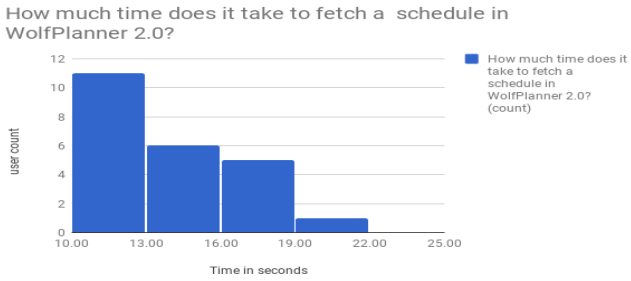

Figure 24: Time taken to get schedule from wolfplanner

3. Were you able to parse a file correctly
   We asked this question so that people actually test the functionality.

4. were you able to fetch schedule on the basis of priority

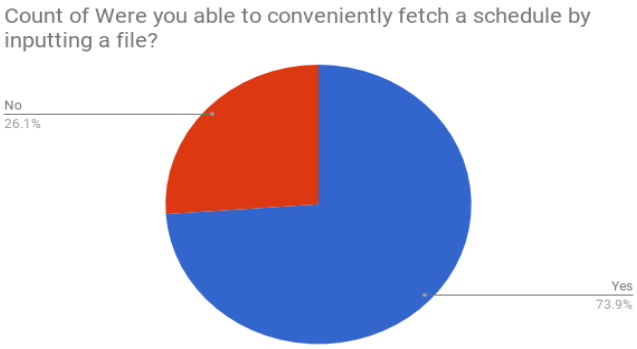

Figure 25: Were you able to parse a file correctly

5. Were you able to fetch a schedule based on priority of tasks
   We asked this question so that people check the output schedule and check if it is based on priority.

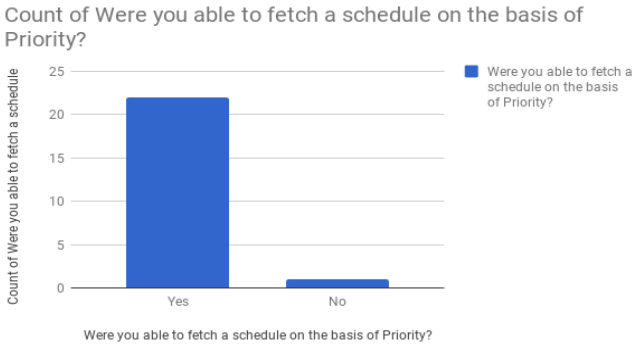

Figure 26: Schedule based on priority

6. How much time does it take them to make a manual schedule.
   We asked this question so that the users actually make a schedule without wolfplanner for comparison.

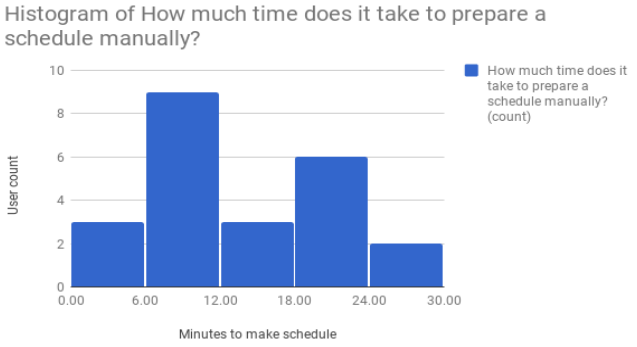

Figure 27: Time taken to make schedule manually

8. CHALLENGES FACED AND SOLUTIONS

WolfPlanner 2.0 is an improvement for WolfPlanner 1.0 as it provides a more flexible and user-friendly approach to generate a weekly schedule based on parameters like priority of the task along with its deadline. It is also able to accomodate breaks and plan meetings. But, as the baseline for WolfPlanner 2.0 remains the same as WolfPlanner 1.0, therefore there were few issues that we still continue to face while working on the improvements in the existing bot.

1. The application allows the user to use fetch schedule feature without adding tasks or courses.
   Solution: The order of the functionalities has not been defined in the flow of the application, but has been mentioned in the documentation of the application README file. If someone misses this detail in the documentation, there is a chance of confusion and possible error in output. We have made the fix that does not allow the user to fetch schedule in a similar way as we have implemented for view tasks vs add tasks (If you try to view tasks without adding, the bot will tell you that give you an option to add tasks and will not give an error/no output) and view courses vs add courses, and have released it along with few other fixes.

2. The users had to use the help command continuously to know the exact keywords to enter.
   Solution: This issue arose because of slight inconsistency in the names of the keywords to be added for functionality. Some keywords had a singular noun such as add task but similar keywords had a plural noun such as add courses. This confused the user slightly and hence had to refer to the help command continuously. This was a mistake on our part and we have rectified it and made the commands consistent to avoid such confusion.

3. Duration of tasks might not be known always.
   Solution: The add task module requires the user to input the number of hours expected to be needed to complete that particular task. This is necessary because the number of hours of each task are a deciding factor for generating a schedule for that user. But, it is difficult for the user to know this before hand and it may change eventually. To solve this problem, we plan to add a modify task module which will allow the user to change the duration of the task according to new knowledge so that he/she can make a better estimate. We can also implement some techniques from the future scope to solve this problem.

9. FUTURE WORK

1. There is always a scope for improving the scheduling. We could add learning capabilities for our bot, in such a way that it can preempt the workload of the user and try to distribute it before hand over the semester.

2. Improvement in parse course functionality, so right now we just have a working POC for the parsing of the schedule from a file and we have done it only for 3 subjects. We can make the system work in a way so as to accept the subjects that the users have for the semester and parse the schedule so that a user has a predefined semester based schedule based on the deadlines and he can add new tasks in the middle of the schedules.

3. Another improvement is that, the bot should be able to learn and guess the priorities of tasks on its own and not leave the burden to the user. The priority should be a choice for the user not a burden.

4. Using Machine Learning Approaches, the bot can predict time required to complete a certain task by a specific user using Machine Learning approaches and set task duration accordingly. The bot will first learn the behavior of a certain user using initial tasks and then give personalized schedule for subsequent tasks. For example, if a user X takes more than the average hours to complete AI Homework 1, then the bot will assign more time than normal for user X in case of AI Homework 2.

5. Integrating the bot with Google maps to find time required for commute. We can find time required from position A to position B using Google Maps API and the bot can take that time into consideration and schedule tasks if they are at different geographical locations. So if a student has a lecture at location A and then has to attend a project meeting at location B, the task scheduler will consider time to go from A to B and reserve time for it in the schedule.

10. CONCLUSION

We have implemented a bot which makes a priority based schedule for the students. The bot takes various tasks as input to be scheduled keeping in mind their priorities and deadline. It additionally takes in the course schedule file for adding tasks automatically to the schedule.The bot communicates with the user regarding the action to be performed. Users can perform various tasks like add course, add tasks with priority, view tasks, view courses and fetch schedule. Before fetching schedule, the user has to provide courses and tasks. The user describes what he wandts to do , the bot extracts the intent from the user input and takes necessary actions. If required, it makes changes to database and and/or calls core modules. If the user asks for a schedule, it calls the MLFQ based scheduling module and returns its output in a user friendly display.

11. ACKNOWLEDGEMENTS

We would like to extend special thanks to our Professor, Timothy Menzies and our mentor, Ken Tu for helping us to come up with this idea and giving us the opportunity to work on this project and guiding us throughout the process.

12. REFERENCES

1. Gartner.com, Gartner Customer 360 Summit 2011

2. Berry, Pauline, et al.Deploying a personalized time management agent. Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems.

3. Schwiegelshohn, Uwe, and Ramin Yahyapour. Analysis of first-come-first-serve parallel job scheduling. SODA. Vol. 98. 1998.

4. Wilkerson, L. J., and J. D. Irwin. An improved method for scheduling independent tasks. AIIE transactions 3.3 (1971): 239-245.

5. William Stallings Operating Systems: Internals and Design Principles 6th Prentice Hall Press Upper Saddle River, NJ, USA AËĘ lâĂŹ2008 ISBN:0136006329 9780136006329

6. MongoDB Documentation,https://docs.mongodb.com/

7. Slack Documentation Slack Reference Documentation : Retrieved from: https://api.slack.com/getting-started-Department of Product Why Slack is popular? Retrieved from: https//api.slack.com/getting-started

8. Node.js Documentation About Node.js Retrieved from:https://nodejs.org/en/about/

9. Botkit Documentation About Botkit Retrieved from:https://botkit.ai/docs/

10. Python Documentation Python Retrieved from: https://www.python.org/about/ [Accessed:

13. EVALUATION CHITS

1. RFO
2. ZJT
3. IEW
4. ZKY
5. PLG
6. TTL
7. QXV
8. YRI
9. VWW
10. NHT
11. BGX
12. ZFF
13. ECN
14. QWF
15. SGR
16. UDX
17. JLS
18. VMY
19. CVV
20. AZL
21. KJR
22. HKK