WolfTutor

A system to enable peer tutoring built on Slack

Monica Metro
NC State University
mgmetro@ncsu.edu

Zachery DeLong
NC State University
zpdelong@ncsu.edu

Zhangqi Zha
NC State University
zzha@ncsu.edu

ABSTRACT

Peer tutoring is generally accepted as a great way to help students engage with material. WolfTutor is one system, integrated with the popular chat program Slack, that seeks to facilitate peer-to-peer tutoring interactions by providing a mechanism for registering tutors, scheduling tutoring sessions, and incentivizing students to become tutors. This paper proposes an enhancement to WolfTutor's tutoring matching process to help enable students to better choose what tutors they want to match with.

1. INTRODUCTION

1.1 What is WolfTutor

WolfTutor is a system that seeks to enable students to tutor other students in a course-setting. Wolf Tutor is a slack-based chat app that attempts to connect potential tutors in given subjects with students who need help with a course. The idea here is peer tutoring, not expert tutoring. At first blush, it seems like the app is an app to actually facilitate peer tutoring, which is not entirely accurate. WolfTutor has functions to register tutors and students and to schedule tutoring sessions. It does not have functionality to actually perform the tutoring itself, but is a logistic tool to enable the coordination required to schedule tutoring sessions. WolfTutor is also gamified. It rewards tutors who are highly rated with a points system which can be implemented in a number of different contexts to help incentivize students to tutor other students.

Make the case for why a tool like this needs to exist.

1.2 WolfTutor Use Cases

1. Register to be a tutor
2. Schedule an appointment with a tutor
3. Review a tutor
4. View rewards
5. Check availability
6. Check subjects
7. Redeem/buy points
8. Check/receive rewards.

1.3 Similar or Comparable Systems

Well, this system is not actually specific to tutoring entirely.

It could reasonably be compared to any scheduling system Compare and contrast with career help services tool.

Sesh App.

1.4 Initial Study

Detailed next is the proposed change and the process of user surveying we followed to identify that information. Following that, we will briefly discuss the design of the WolfTutor app In the final sections of the paper, we will discuss our evaluation plan for the proposed enhancement and talk about the plan to accomplish that goal.

2. ENHANCEMENTS

2.1 Pain Points

The original creators of WolfTutor proposed three additional features for future development. The first was to integrate an online platform to conduct tutoring sessions online. The second was to allow both the tutor and the student to sync their session reservation with their calendar, e.g. Google Calendar. Lastly, to update the scheduling algorithm such that a user could edit or delete reservations in addition to being able to create and view them. Our team provided two other features: increasing matching options between tutors and students and allowing students to browse their reservation history.

After careful consideration, three main pain points were decided upon that consisted of the proposed enhancements. Integration of an online platform was discarded because it was not thought to facilitate the quality of the match between tutor and student.

Scheduling and Calendar Sync.

WolfTutor currently only allows users seeking a tutoring session to create and view reservations. In real life, people change their mind or events come up such that a schedule change is in order. The ability to cancel or reschedule reservations would make WolfTutor more applicable to real

PLDI '13 June 16–19, 2013, Seattle, WA, USA

ACM ISBN 123-4567-24-567/08/06…$15.00

DOI: 10.475/123_4

world scheduling scenarios. In addition, facilitating intergration with commonly used calendars such as Google and IOS Calendar extends this principle of making scheduling easier.

Increasing Matching Options.

In regards to matching with tutors, WolfTutor currently displays a list of tutor's and their ratings based on the student's desired subject. Then, the student may attempt to book a reservation with a tutor they choose. By expanding matching options, a student should be able easily find a higher quality match with a tutor. For example, a student could filter tutors by location by selecting only locations they want to meet up at in a checkbox. Similarily, if a student has a few tutors they prefer, they could select those names from a checkbox such that WolfTutor only displays those tutors. This type of filtering system is often used by medical facilities with multiple practioners and multiple practing sites. It is also used by NC State University's scheduling tool on epack.

History and Recommendations.

WolfTutor does currently allow for student's to review and rate the tutors that they met with previously. However, there is no way to view a student's reservation history past the most recent tutor. WolfTutor also does not provide a way for a student to easily re-book a new reserveration with a tutor they chose previously if they liked the tutor and would like to schedule a reservation with them again. Adding these enhancements would increase ease of use by helping a student distinguish between a competent and non-competent tutor they've had in the past.

2.2 Initial Study

A survey was conducted to determine which enhancement the intended user base (students) prefered the most. The survey was separated into three parts.

Background.

A background section measured how prevalent scheduling was within the daily life of the participants. Most participants admitted having to schedule a meeting within the last year.

Priorities.

The next section revealed which enhancement the participants thought was a priority by asking them to rate the level of important the enhancement was on a scale from 1 (least important) to 5 (most important). To avoid bias, instead of explaining the enhancements out, the survey questions were created around the base point of the enhancement:

- "When scheduling a tutoring meeting, picking the location is very important to me."
- "When scheduling a tutoring meeting, the competency of the tutor is very important to me."
- "When scheduling a tutoring meeting, being able to agree on a time quickly and easily is very important to me."

The question regarding competency scored the highest level of important among the participants collectively. Scheduling and increased matching (by location) followed respectively.

Trade Offs.

The objective of the third section was to validiate the results in the second section by asking the participants which enhancements they would be willing to compromise on in order to get the enhancement they thought was more important. The questions included:

- "When scheduling a tutoring meeting, I am willing to make trade-offs on the competency of the tutor and time if I can specify the location of the meeting."
- "When scheduling a tutoring meeting, I am willing to make trade-offs on location and time if I can specify the competency of my tutor."
- "When scheduling a tutoring meeting, I am willing to make trade-offs in location and competency if I can specify the time of the meeting."

Again, competency scored the highest as the most important enhancement collectively. Scheduling followed in second place and location matching in third.

Other.

An option was given to participants to suggest a new enhancement. The only received response was to integrate the application with Skype.

2.3 Chosen Enhancement

3. ARCHITECTURE

3.1 Original Architecture

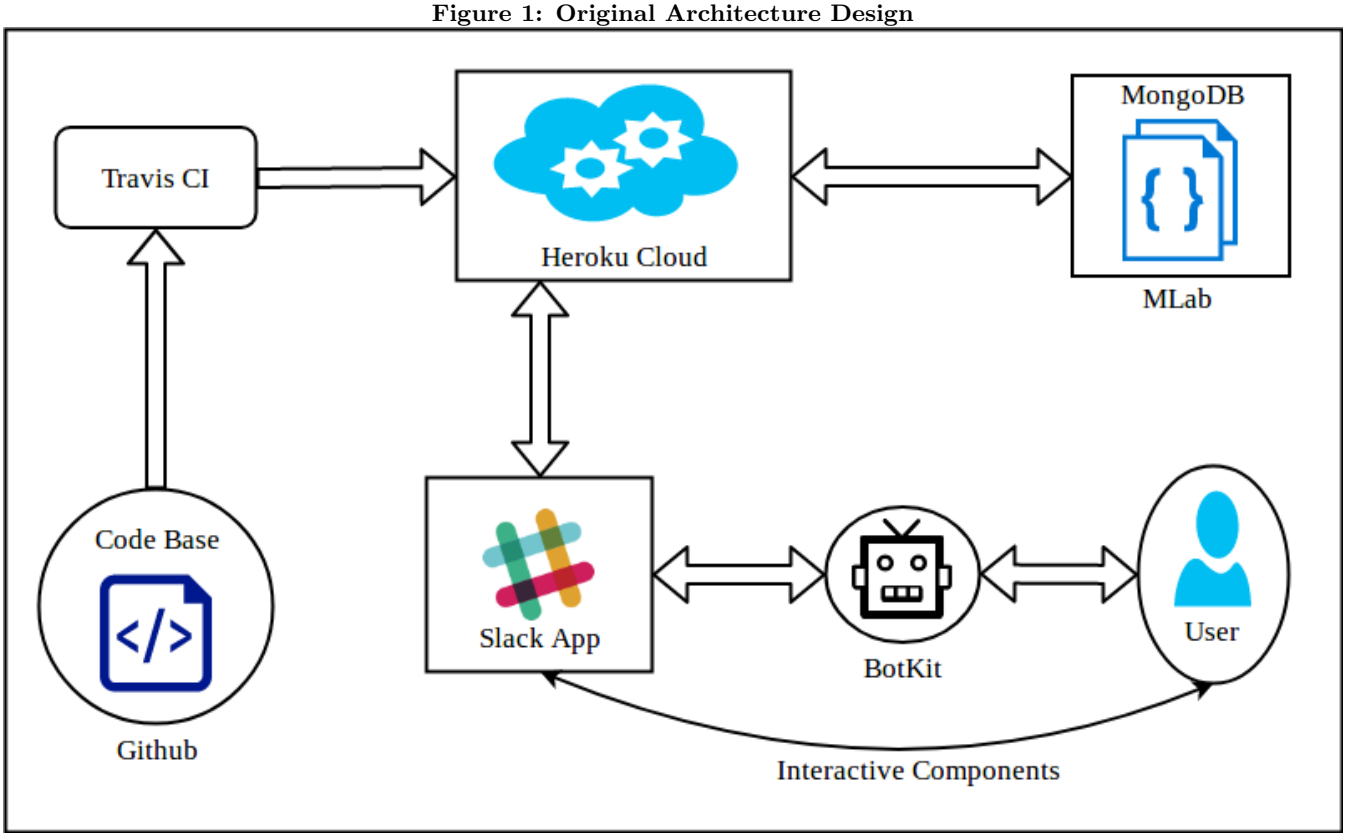
This section outlines the various components of the system and how they interact with one and another. The detailed architecture is described in figure 3.1.

Broadly speaking, the architecture is divided in three main components: the slack app, the heroku cloud and the mongoDB database. This separation of concerns is a major advantage since database is independently hosted on mlab server and can be accessed from anywhere with valid authorization credentials. The heroku cloud hosts the logic of the slack app, which communicates with the user. The system also implemented the continuous deployment, continuous integration pipeline with Travis CI, which directly pushes the code to heroku cloud once the build passes (also indicated in the readme section of the repository). We have written test cases which acts as a sanity check for any commit made to the master branch, before deployment so as to not push broken code on the production server at heroku. Following is the detailed description of each component of our architecture.

Slack App.

Slack App is generated in api.slack.com dashboard. It is currently developed only for one workspace and it is configured to communicate to the heroku server which is hosted at wolftutor.herokuapp.com url. Also creating a slack app gives us various authentication tokens like access token and verification token which are required for Slack to authorize that the requests and responses are coming from a valid production server. Slack app also gives us configuration of the bot like its name its icon, etc. This will be visible to the user when the user interacts with the bot. Slack bot is a part of

Figure 1: Original Architecture Design



the slack app, where bot is considered like a user (bot-user) in slack terminology. Slack app also allows access to interactive components like dialogs and message menus for the overall user experience to be more rich and interesting.

BotKit.

Botkit is an external library that integrates with Realtime API which can detect patterns in the user queries. The logic to be executed when a particular pattern is detected is written in the slack app (NodeJS code hosted on heroku). For instance, on saying hi user should be given an option to enroll in the system. This is one example of working of the Botkit module.

Code Base.

Code base is our Github repository where we maintain our code. Following the general convention, we made new branch for every new logical feature and also linked issues with particular merge commits. All this activity can be viewed in our repository. We have also used conventional practises of separating database queries and put them all together in model directory. Also all of our interactive components are in distributed in different folders.

Continuous Integration Module.

The master branch of our repository is linked with Travis CI to be pushed to heroku server if the build on travis CI passes. The tests written in mocha acts as a final sanity check before deploying the code on the production server. Any code that is pushed on the master is directly built on Travis CI and deployed on heroku server if build passes.

Heroku Server.

Heroku server is the home of our NodeJS application. We have used single dyno (free version) to host the application. The code pushed on master is deployed here by Travis and we always have the latest code in the production server. Also we have included the Procfile where it is indicated what command to execute to run the application (npm start). This is necessary for heroku to understand the starting point of the application.

Database.

Mlab is a server that hosts our Mongo Database. The advantage of having a remote server for mlab is that everyone can access the same data at simultaneously. In mongo, only a mongo URI is required for accessing the database along with valid user credentials. This makes it very simple to test the application locally as well as run in on the server.

User.

User is anyone who is signed in into the Slack workspace where the slack app is added. He/She can communicate with the app in variety of ways as described in the use-cases section. Also he/she can interact with the app using dialogs and message drop-downs and buttons to give a particular command. See details in use-cases section.

4. PROGRESS

5. VALIDATION

6. CONCLUSION

6.1 Schedule