Pupil

Peer-to-peer tutoring platform

Eugen Istoc
College of Computing
Georgia Institute of Technology
Atlanta, US
eugenistoc@gatech.edu

***Abstract*—The purpose of this paper is to provide rationale and describe the motivation for building the "Pupil" peer to peer tutoring platform. The paper presents the case that peer tutoring brings a unique experience to university campuses. Research has demonstrated that peer learning is an effective means of communicating complex concepts. Furthermore, university adaptation on campuses can further add value to the student experience as it provides a source of extra revenue while investing in the overall intellect of the student body.**

To different forms of peer interactions are presented: tutor and student. Each has its role in this platform. A tutor is connected to a seeking student, while a student is broadcasted to available tutors. The interaction between the two parties are managed and established by the platform, thus a hedge of protection is implicitly recognized.

Keywords—peer to peer tutoring; educational technology; mobile first tutoring

I. INTRODUCTION

The goal of this project is to develop a mobile tool for connecting students to each other for one on one homework help. The target audience for the platform is explicitly limited to university students and and to specific college campus where this platform is adapted. There are two fold benefits in this kind of peer-to-peer tutoring platform. First, it provides a resource for students that need homework help for a fee and, secondly, it enables students who have the advantage of having progressed further in their studies profit through this platform.

A key aspect to this project compared to other existing tools, is that an emphasis is placed on local university campus. The peers are encouraged to help, generally, students that they know, and/or, students that are part of the same student body as themselves. By doing this, a more comfortable tutoring experience is achieved

II. BACKGROUND

Peer-to-peer tutoring is not a new concept. There are many products that exist which attempt to tackle this form of tutoring. A very popular solution is called Code Mentor._ The underlying principle behind the platform is that users can connect with experts in almost any technical field available. Driven primarily by a rating system, users and experts are naturally connected to each other.

A. Validation tutors

This project takes a unique approach to peer-to-peer tutoring. By restricting the tutor and student space strictly to university campuses the added benefit of domain specificity is added. That is, tutors are much more contextually aware of the materials which the students are facing. Furthermore, the added restriction of forcing the tutors to be able to offer tutoring services only for classes which they, themselves, have already taken offers the student a higher level of confidence in the tutoring which they expect to receive from a tutor.

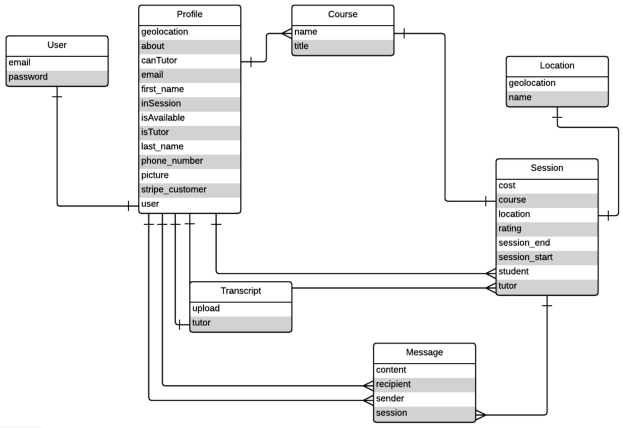
There are several ways in which a tutor's transcript can be validated. For the sake of expediting this proof of concept, a tutor simply needs to submit a copy of their transcript as an attachment, and a simple review of the classes that they are wanting to tutor for is reviewed. More formally, however, an official transcript would have been requested from the school to ensure proper credentials and minimum class requirements are met during the review process

B. Connecting with tutors

One key selling point of this project is that students are not as likely to meet or interact with people that they do not personally know. Thus one of the goals for this project was to reduce that skepticism as much as possible by introducting the mechonisms that were described earlier, as well as implementing a few other safety features. Namely, by design, all payments are handled at a flat rate. That is, there is not to be any discussion relating to payments between tutors and students as all tutors are payed equally. This model, of course, is variable and may be tweaked in the future, but the idea is to remove any dsicussion of payments between students as tutors as to not create an uneasy commnication environment. Secondly, a rating system between the tutors and the students help maintain a healthy community as the number of "bad" peers are reduced over time. A rating mechanism will never match two peers again if either one of them rated the other three stars or below.

III. Technical Implementation

For fast prototyping several platforms were utilized in order to expedite the development process. A platform called *Stamplay* was utilized as a PaaS allowing me to bring many other technologies together in one place. Modeling the data and designing the data model (and constrains) was all accomplished via this platform.



A. Technology Stack

It was established that this application is to be built primarly using web technologies. Commonly referred to as a hybrid mobile application in that it is build primiarly for the web yet can interact with native mobile hardware via bindings.

B. Tools

- AngularJS: The javascript framework used to build the application. This is the main component as it is used for building the user interfaces, organizing the data model, and commnicating with the backend services.
- Ionic: The mobile addon for AngularJS enabling the easy deployment to all mobile devices. Furthermore, it provides a native-looking fluid user interfaces with the same codebase.
- Cordova: A native library that bootstraps the web application biult with AngularJS and Ionic to the device deployed to. It exposes a set of APIs to the AngularJS application which the application can use.

Although there are more components used in this project, the above listed componets would be considered the biggest ones that are worth mentioning.

C. Third party dependencies

As mentioned, *Stamplay* is primarly used for modeling the data and enforcing data integraty. The are a few other services that are used, however, on this project.

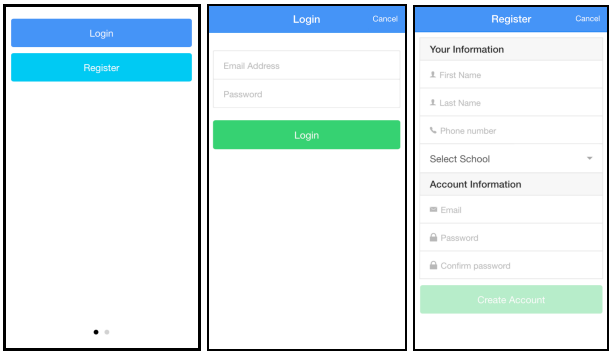
- Twilio: Offers APIs to interact with text messaging. This allows the application to validate users by sending them a validation code via text message.
- Pusher: Adds a realtime component to the application. This is the component that is instrumental in connecting tutors and students.
- TravelingMailBox: Provides the app “Pupil” a physical mailing address to which prospective tutors may have their official transcript sent. This service scans all incoming mail in favor of digital copies.
- Stripe: All operations relating to payment are handled via this component. Transactions costs are absorbed by the platform, which in turn are passed to the tutors.

D. Some Considerations

- Due to the fact that this is a hybrid application, all data acesssing routines are done over http. This can be an issue, at times, due to the fact that http connections tend to be more costly.
- UI and data caching are in place in order to reduce any latency which exists between data refreshes.
- Web UI components are generally slower than their native couterparts, thus, a minimalist approach to component selection was adapted.

IV. Using the Application

Launching the application for the first time gives the user the option to either *Register*, or *Login*.



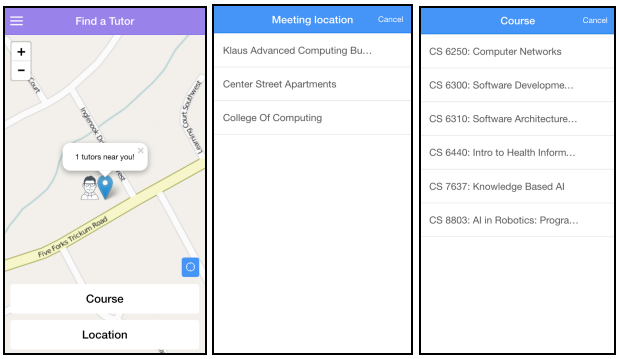
The authentication is handled by the *Stamplay* services, therfore, therfore a Javascript Web Token is utilized in maintaining a valid authorization throughout the lifetime of the application. When registering for the first time, however, the registering user’s school is requested as there are only a set of supported campus where this app would be initially deployed.

There are manh tasks that take place when a user registered. Among those are: creating a profile, setting up a *Stripe* customer, and sending a text message. All of these tasks are configured to run appropriately in *Stamplay*. Upon the successful completion of all of those tasks in sequence, the user would be considered registers and authenticated.

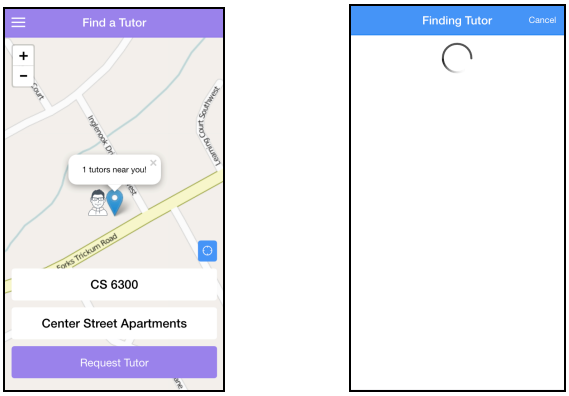
A. The Student View

Both the student view and the tutor view share the same codebase of the application. Toggling betwee nthe two views is

implemented via a simpel toggle switch on the main menu. The goal is to increase usability and to encourage students to also become tutors.



The main student dasahboard page initially shows the number of tutors that exist around them. This will given them an idea of the pool of tutors that there are to choose from. The have the possibility, howver, to filter the pool of tutors based on a *course* criteria. Clicking the *Course* button presents the student with a list of classes that all the available tutors can tutor for as a whole. Selecting a course, will filter the map to only tutors that meet that criteria. The last options is for selecting a location. The locations available on this screen are predefined to enforce a safe meeting places for students and tutors to meet. These locations are determined on a campus by campus basis are are carefully considered to be generic enough to offer safety, and specific enough to enable conveniency.



Once a Course and location has been selected, the application will then put out a request for a tutoring session given the said parameters. On the backend, however, a new session object is created using the provided parameters, and is put in a *PENDING* state. A *PENDING* is a session which was not yet served by any tutor and is awaying a connection from a tutor.

B. *Session States*

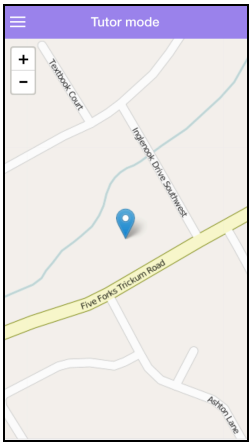
Sessions in the application can be in 1 of 5 states. They are implemented as a declarative *ENUM* in JavaScript and are described below:

- PENDING: Session is brand new an is waiting to be matched to a tutor
- AWAITING_HAND_SHAKE: Connected to a tutor, but waiting for the tutor to accept the request.
- PENDING_START: Tutor has accepted the session, but has not started the session yet.
- IN_PROGRESS: Session has been started by the tutor.
- FINISHED: Session has been marked as finished by both parties, and the duration has been locked in.

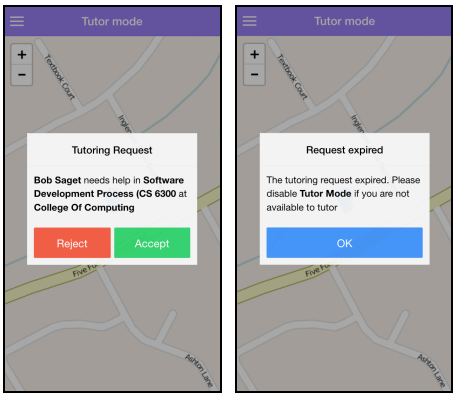
Sessions moves between these states can be initiated by the student or by the tutor. Context typically determines the flow from one state to another.

C. *The Tutor View*

The tutor view dash board is, in fact, much simpler. Since the tutor is simply in the position to accept tutoring requests, all they would see is their location on a map and their availability.

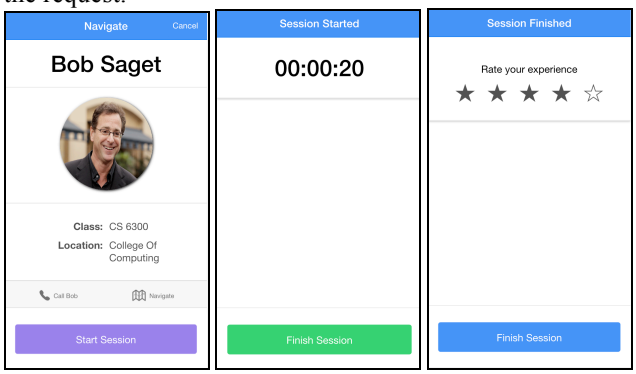


Although the tutor has no possibility to of knowing how many students are around them, the are not required to keep that application open in order to be considered *available*. Push notifications are used anytime a student requests a tutor and an available tutor does not have the application in the foreground.



Benefitting from the *Pusher* realtime component, the tutor is notified instataneously regarding a tutoring request. In the request, basic information is included like the student's name, the course in question, and the location where the student wishes to meet with the tutor. Furthermore, the tutor has 10 seconds to respond to the studente before the request expires.

When a request appears on the tutor's screen, the request is put in a AWAITING_HAND_SHAKE state to prevent other tutors from being match to this student. If, however, this particular tutor fails to respond in the given time, or for some reason rejects this request, it status is restored back to PENDING to enable other tutors to pick up the request.



When a tutor accepts a session, the session doesn't yet start until they physically are able to meet up with the student at the location specified by the request. They have the option to call and/or navigate to the student's location. If and only if the tutor has successfully arrived at the student's location are they able to click *Start Session* in order to start the session. It is at ths point that the session moves from a *PENDING_START* state to a *IN_PROGRESS* state.While the session is in progress a timer runs on both the student and tutor applications indicating the legnth of the current sesssion. Furthermore, the backend is updated very second with the session length in order to accurate calculate the payment amount when the session has finished.

Lastly, at the end of the session, both the student and the tutor are presented with a rating option which is used to improve the overal health of the Pupil ecosytem. When either a tutor or a student rates the other party a 3 star or below, they will never be matched again with the same party.

REFERENCES

[1] N. Falchikov, Learning together: peer tutoring in higher education. London: RoutledgeFalmer, 2001.

[2] K. J. Topping and S. W. Ehly, Peer-assisted learning. Mahwah, NJ: L. Erlbaum Associates, 1998.

[3] K. J. Topping, The peer tutoring handbook: promoting co-operative learning. London: Croom Helm, 1988.

[4] S. Goodlad and B. Hirst, Peer tutoring: a guide to learning by teaching. London: Kogan Page, 1989.

[5] J. MacDonald, Blended learning and online tutoring: planning learner support and activity design. Aldershot: Gower, 2008.

[6] P. S. Koskinen and R. M. Wilson, Tutoring: a guide for success. New York: Teachers college, Columbia University, 1982.