Running Head: Lab 1 - Tutor Dash Description

Lab 1 - Tutor Dash Description

Alexander Wojtowicz

CS411W - Professional Workforce Development II

Old Dominion University - Team Gold

Professor T. Kennedy

07 October 2019

Version 2

Table of Contents

1. Introduction ………………………………………………………………………………. 4
2. Tutor Dash Product Description ……………………………………………………….. 8
    2.1. Objectives and Goals of Tutor Dash ………………………………………….... 12
    2.2. Features and Capabilities ……………………………………………………… 12
    2.3. Major Functional Components ………………………………………………… 16
        2.3.1. Software ……………………………………………………………… 17
        2.3.2. Hardware …………………………………………………………….. 18
        2.3.3. Entities ……………………………………………………………….. 18
3. Identification of Case Study ………………………………………………………….... 19
4. Tutor Dash Prototype Description ……………………………………………………... 20
    4.1. Prototype Architecture ………………………………………………………… 23
        4.1.1. Prototype Hardware ………………………………………………….. 23
        4.1.2. Prototype Software ………………………………………………….... 23
    4.2. Critical Design Components ………………………………………………….... 24
        4.2.1. Database …………………………………………………………….... 25
        4.2.2. UI/UX ……………………………………………………………….... 26
        4.2.3. Algorithms …………………………………………………………… 36
    4.3. Prototype Features and Capabilities …………………………………………… 43
    4.4. Risks and Mitigations ………………………………………………………….. 44
    4.5. Prototype Objectives and Goals ……………………………………………….. 46
    4.6. Prototype Development Challenges …………………………………………… 46
5. Glossary ……………………………………………………………………………….. 48
6. References …………………………………………………………………………….. 49

List of Figures

1. Students Seeking Tutoring Advertisement 6
2. Students Seeking Academic Assistance 7
3. How Tutor Dash Affects The Current Process Flows 10
4. Tutor Dash Process Flow 11
5. Major Functional Component Diagram 16
6. UI/UX Full Implementation 27
7. UI/UX Phase 1 28
8. UI/UX Phase 2 29
9. UI/UX Phase 3 31
10. UI/UX Phase 4A 32
11. UI/UX Phase 4B 33
12. UI/UX View 1 - Settings 34
13. UI/UX View 2 - User Profile 35
14. PDF Transcript Parser Algorithm Flow 36
15. Relative Distance Calculator Algorithm Flow 38
16. Competitive Pay-Rate Algorithm Flow 39
17. Web-Conference Appointment Creator Algorithm Flow 40
18. Payment Logic Algorithm Flow 41
19. Risk Matrix Overview 44
20. Risk Matrix Descriptions 45

List of Tables

1. RWP vs. Prototype 22
2. Database Schema 25

1. Introduction

Tutor Dash is proposed to be a mobile android application that would greatly benefit university students' perception of tutoring. This application will serve as a hosting platform where university students can search for tutors, advertise their own tutoring services, or do both. In a way, Tutor Dash will be similar to Uber in concept, but the functionality and interface will be much different. Tutor Dash will be a platform exclusively for university students with the aim of improving the current processes of searching for tutors and providing tutoring services. Tutor Dash will accomplish this by consolidating various features already present on other digital tutoring platforms with new features that have yet to be implemented on any other platform. Ultimately, Tutor Dash's goals are to make tutoring services more accessible to students looking for tutors and to make finding students to tutor easier for tutors looking for new clients.

There are several reasons why an application like Tutor Dash is necessary. The biggest reason is the lack of centralization within the market. The current market for private tutors is saturated with options that do not fulfill the demands of university students. Both students who are searching for tutors and student tutors who are searching for new clients have a difficult time finding who they are seeking. Tutor Dash aims to solve this problem by creating a centralized platform where both students searching for tutors and student tutors searching for new clients can interact and find who they are looking for in real-time.

There are two groups of users relevant to Tutor Dash's domain; the first of which is university students looking for tutors. Students looking for tutors face several problems relating to the lack of sufficient university-offered tutoring resources. University-offered tutoring resources are generally offered exclusively during day-time business hours (i.e., 9am to 5pm

Monday through Friday), however studies have shown that students are more likely to study outside of these hours (Evans, 2017). These tutoring resources are also very limited in scope. For example, in the spring of 2019 at Old Dominion University, only 5% of all offered courses had an associated tutoring program provided by the university (“Course-Specific Tutoring”, 2019). For the courses that universities do offer tutoring, schedules for sessions are inflexible and/or inaccurate, leaving many students unable to attend (“Campus Tutoring”, 2019). For all of these reasons, students will end up either looking elsewhere for tutors or perform poorly in their courses.

When students look for tutors outside of their university, there are several decision obstacles that they must face as illustrated in Figure 1. A student searching for a private tutor will first have to choose a platform in which he/she wants to search. The student will then invest his/her time in searching for the right tutor. In order for this tutor to be compatible with the student, there are several factors that both the tutor and the student must agree on such as availability, domain knowledge, and delivery method. If a student does find a compatible tutor, he/she will then schedule a meeting. This meeting will most likely be scheduled at some time within the next few days or weeks. After the meeting occurs, the student will pay the tutor, and hopefully he/she now understands the course material.

[This space intentionally left blank]

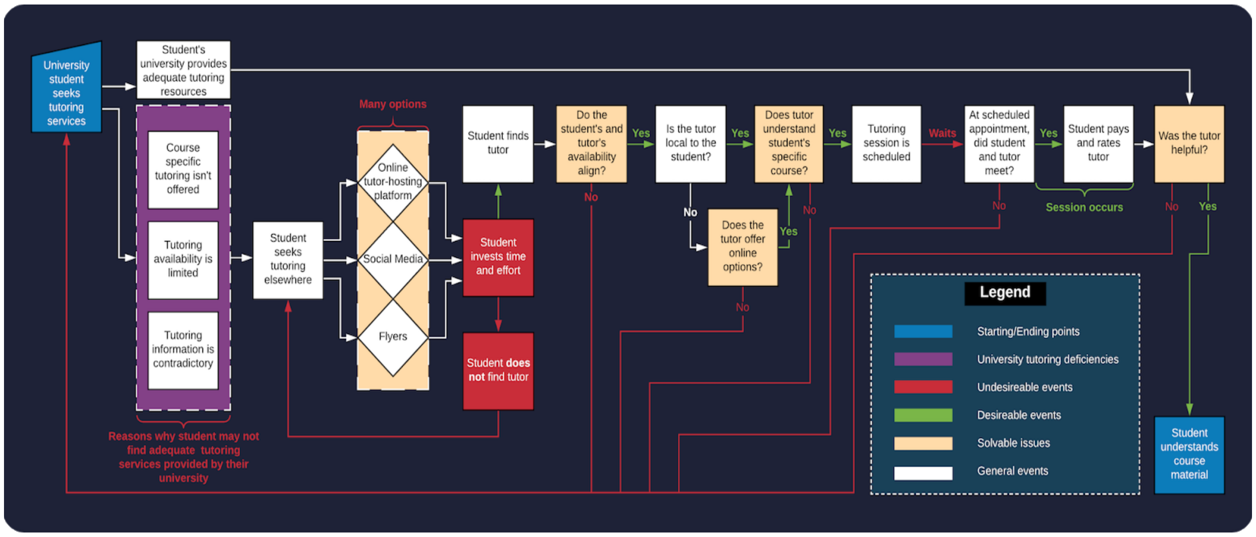

University student seeks tutoring services
Student's university provides adequate tutoring resources
Course specific tutoring isn't offered
Tutoring availability is limited
Tutoring information is contradictory
Reasons why student may not find adequate tutoring services provided by their university
Student seeks tutoring elsewhere
Many options
Online tutor-hosting platform
Social Media
Flyers
Student invests time and effort
Student does not find tutor
Student finds tutor
Do the student's and tutor's availability align?
Yes
No
Is the tutor local to the student?
Yes
No
Does the tutor offer online options?
Yes
No
Does tutor understand student's specific course?
Yes
No
Tutoring session is scheduled
Waits
At scheduled appointment, did student and tutor meet?
Yes
No
Student pays and rates tutor
Session occurs
Was the tutor helpful?
No
Yes
Student understands course material
Legend
Starting/Ending points
University tutoring deficiencies
Undesireable events
Desireable events
Solvable issues
General events

Figure 1. Students Seeking Academic Assistance

This is the current process flow for tutees looking for tutors. The diagram illustrates many of the decision obstacles that make it difficult for students to find compatible tutors.

There are several problems with this current process. If a student is having trouble understanding course material, then he/she is not going to want to go through the hassle of finding a tutor. Finding the right tutor involves a major time investment on the student's part, and this deters many from pursuing or even initiating their search. There are many obstacles that get in the way of the student's objective, and even if the student does go through the trouble of finding a compatible tutor, then there is still the chance that the tutor was unhelpful.

The second group of users relevant to Tutor Dash's domain is the group of private student tutors searching for new clients. This group mainly faces problems relating to the lack of centralization in the current private tutoring market. There are several tutor-hosting platforms in existence, so in order for tutors to stay competitive, it is important for tutors to advertise themselves on the right platform(s). Knowing which platform(s) is/are right for the tutor is subjective though, as there are no obvious options. Like their counterparts, student tutors looking for new clients also face several decision obstacles. These obstacles closely mirror those that students looking for tutors face.

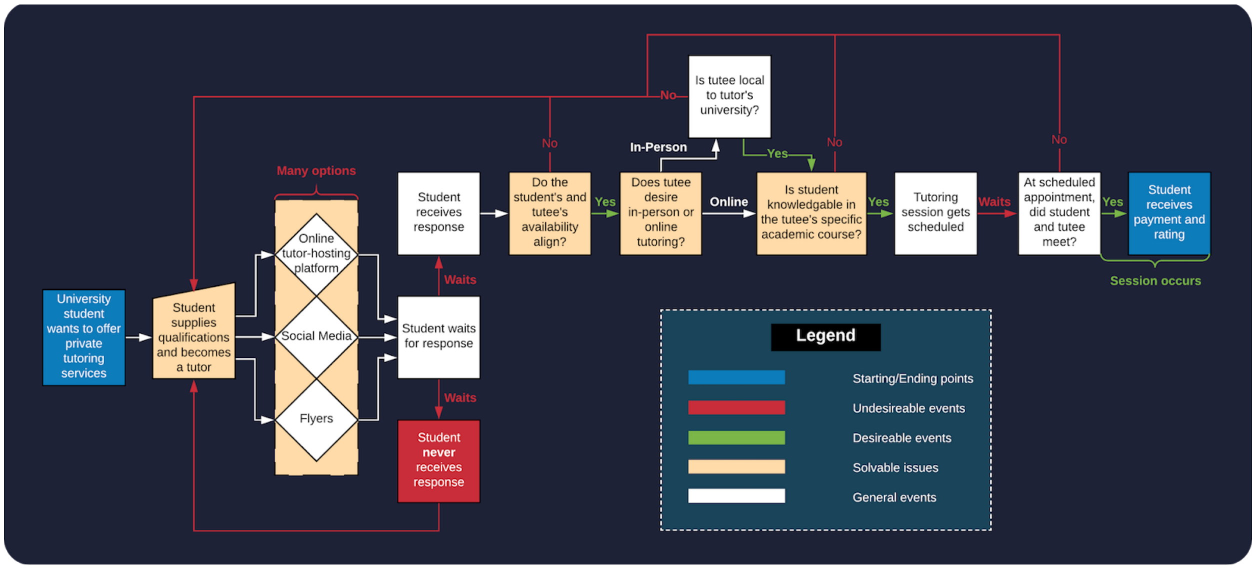

University student wants to offer private tutoring services
Student supplies qualifications and becomes a tutor
Many options
Online tutor-hosting platform
Social Media
Flyers
Student waits for response
Waits
Student receives response
Waits
Student never receives response
Do the student's and tutee's availability align?
No
Yes
Does tutee desire in-person or online tutoring?
In-Person
Online
Is tutee local to tutor's university?
No
Yes
Is student knowledgable in the tutee's specific academic course?
No
Yes
Tutoring session gets scheduled
Waits
At scheduled appointment, did student and tutee meet?
No
Yes
Student receives payment and rating
Session occurs
Legend
Starting/Ending points
Undesireable events
Desireable events
Solvable issues
General events

Figure 2. Students Seeking Tutoring Advertisement

This is the current process flow for tutors looking for tutees. Tutors looking for tutees face similar obstacles to tutees looking for tutors as shown in Figure 1.

A student looking to become a private tutor must first present their qualifications to one or more of the various platforms currently present on the market. After this, he/she must then wait for a potential client to contact him/her. If a potential client contacts him/her, then several factors must be agreed upon between this student and the potential client (e.g., availability, domain knowledge, and delivery method). If both the student tutor and potential client agree to meet, then this meeting will be scheduled at some point in the future. After the meeting occurs, the tutor will then be paid based on the implementation of their platform’s payment method. It is also customary that after the meeting concludes, a rating will be given to the tutor based on his/her performance.

Much like the current process for students seeking tutors described in Figure 1, the current process for student tutors seeking new clients has several problems. With so many options available, it is difficult for student tutors to find new clients that will benefit from their expertise. Student tutors also face the same decision obstacles their counterparts face. In addition to these obstacles, tutors have the added requirement of becoming qualified on their platform. Many platforms on the market such as Skooli, Wyzant, and Tutor.com all require degrees as their minimum qualifying requirement. For undergraduate degree-seeking university students, these platforms are not accessible since these students have not yet earned their degree(s). Platforms such as Facebook and Care.com will often be utilized by university student tutors since qualifications are optional. However, these platforms do not serve primarily as tutor-hosting platforms. Due to this, the likelihood of a potential client finding the student tutor is not high.

Tutor Dash aims to bring these two groups of users together in one centralized platform. By providing a platform specifically targeted for university students, Tutor Dash will serve as a niche tool to fulfill the demands of university students involved on both sides of the private tutoring market. Effectively, this will mitigate many of the frustrations that both groups of users face when trying to find their counterparts.

2. Tutor Dash Product Description

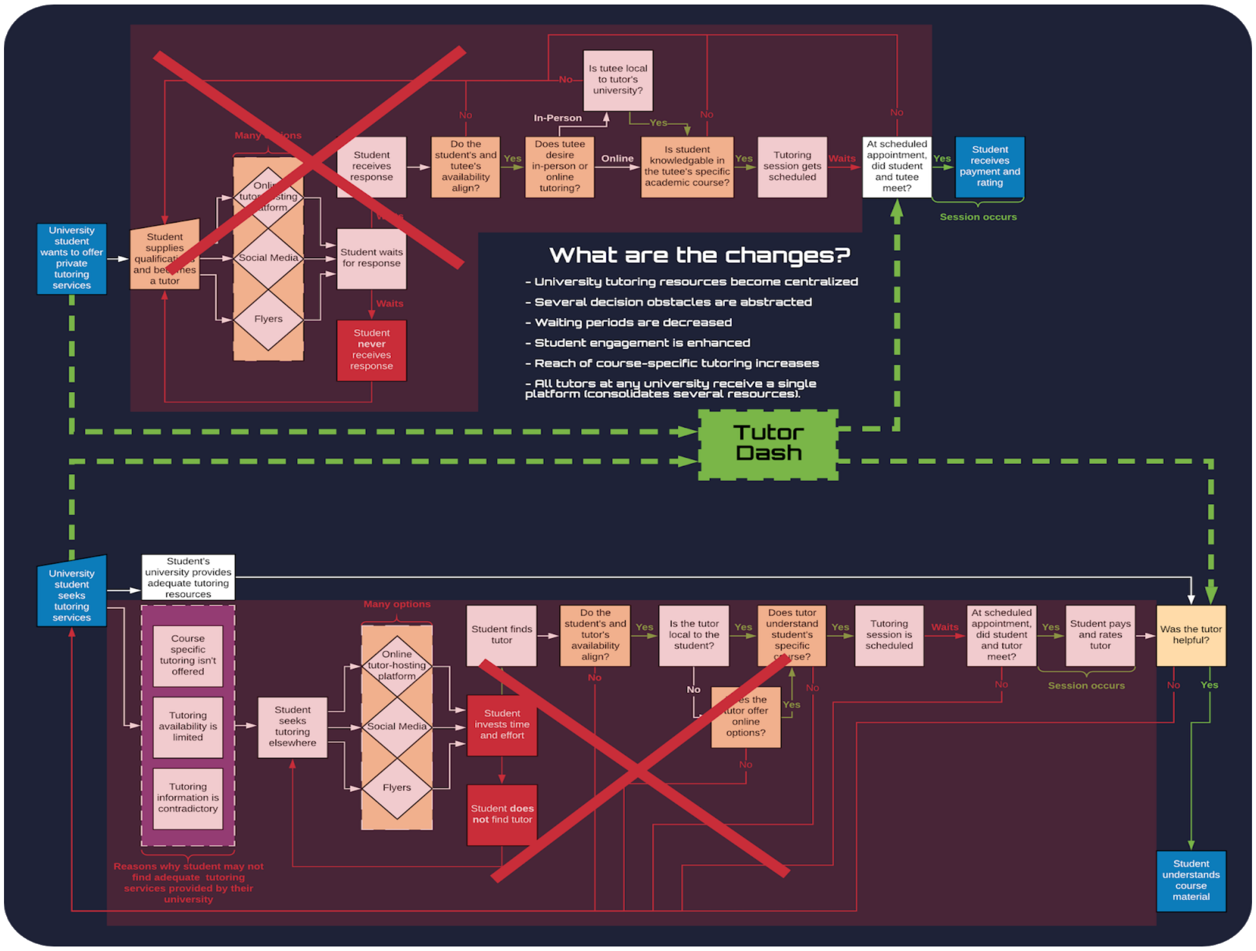
Tutor Dash will be an android application supported for mobile devices running Nougat OS v.7.0 and higher. The aim of this application is to make tutoring more convenient for both student tutors and student tutees at universities by providing an intuitive interface with several features that will benefit the overall user experience. This application will be developed and designed solely for university students to alleviate some of the uncertainty when communicating with strangers over an app service. By providing a niche tool for university students that the private tutoring market demands, Tutor Dash will be a unique product that not only has the potential to be profitable, but it also has the potential to increase student engagement and learning across multiple universities.

Tutor Dash will accomplish these claims by implementing a variety of features which include the following:

1. Constraining the user-base strictly to university students
2. Qualifying tutors in real-time based on previously-taken courses
3. Implementing multiple rating systems for both tutors and tutees
4. Providing a toggling availability option for both tutors and tutees
5. Allowing any course at any given university to be offered in-app
6. Implementing open availability and 24/7 scheduling
7. Providing a meaningful notification system
8. Automatically calculating competitive pay-rates based on market parameters
9. Allowing for both in-person and online meetings
10. Providing a mechanism for automatic payments within the app.

The biggest distinction between other tutor-hosting platforms and Tutor Dash is that Tutor Dash allows university students to become qualified based on their previously-taken courses in real-time. This concept of “real-time” qualification refers to the ability for students to supply their credentials and become qualified almost instantly. By implementing real-time functionality, tutors will be able to download the app and start tutoring in a matter of minutes as opposed to weeks. Tutor Dash will also implement an exclusive feature that calculates tutor pay-rates automatically to mitigate experienced tutors from monopolizing the market. This feature will be discussed in Section 4.2.3 and illustrated in Figure 16.

A more in-depth look on how Tutor Dash will affect the current process flows is illustrated in Figure 3. The decision obstacles that currently make finding tutors difficult for tutees and vice versa will be completely eliminated by Tutor Dash's implementation because it will fill the gap that currently exists in the private tutoring market. The premise is simple; both tutors and tutees log into the app, and they will be able to find who they need immediately in real-time.


University student wants to offer private tutoring services
Student supplies qualifications and becomes a tutor
Many options
Online tutor-hosting platform
Social Media
Flyers
Student receives response
Student waits for response
Waits
Student never receives response
Do the student's and tutee's availability align?
No
Yes
Does tutee desire in-person or online tutoring?
In-Person
Online
Is tutee local to tutor's university?
Is student knowledgable in the tutee's specific academic course?
Tutoring session gets scheduled
At scheduled appointment, did student and tutee meet?
Student receives payment and rating
Session occurs
Tutor Dash
University student seeks tutoring services
Student's university provides adequate tutoring resources
Course specific tutoring isn't offered
Tutoring availability is limited
Tutoring information is contradictory
Reasons why student may not find adequate tutoring services provided by their university
Student seeks tutoring elsewhere
Student finds tutor
Student invests time and effort
Student does not find tutor
Do the student's and tutor's availability align?
Is the tutor local to the student?
Does the tutor offer online options?
Does tutor understand student's specific course?
Tutoring session is scheduled
At scheduled appointment, did student and tutor meet?
Student pays and rates tutor
Was the tutor helpful?
Student understands course material

What are the changes?

- University tutoring resources become centralized
- Several decision obstacles are abstracted
- Waiting periods are decreased
- Student engagement is enhanced
- Reach of course-specific tutoring increases
- All tutors at any university receive a single platform (consolidates several resources).

Figure 3. How Tutor Dash Affects the Current Processes

Tutor Dash merges the two current process flows illustrated in Figures 1 and 2. By providing a niche tool that connects students at the source, many of the current decision obstacles are bypassed.

Figure 4 illustrates a more comprehensive flow of how Tutor Dash will work. Users will first have to download the application onto their android phone, and then they will need to create a user account. At this point, the user can specify whether he/she would like to be qualified as a tutor by uploading his/her official academic transcript into the application or remain only as a tutee. Every user will be a tutee by default even if he/she never exercises this part of his/her membership. Membership eligibility is determined by the user's university email address, so if the user tries to sign up with an email that does not have a domain supported by the application, then his/her membership request will be denied. Once the user's membership has been instantiated, he/she will be ready to use the application.

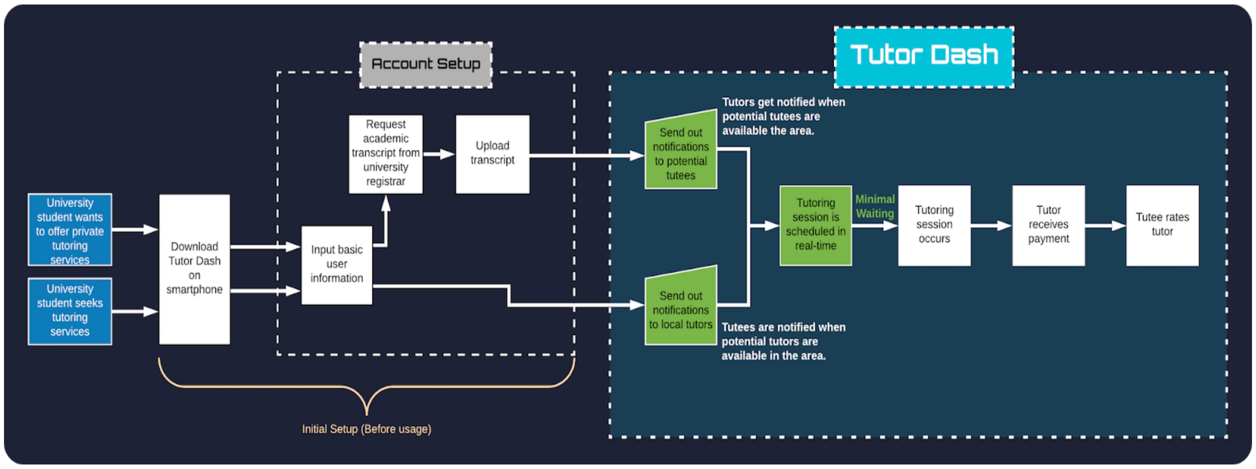

Account Setup
Tutor Dash
University student wants to offer private tutoring services
University student seeks tutoring services
Download Tutor Dash on smartphone
Input basic user information
Request academic transcript from university registrar
Upload transcript
Send out notifications to potential tutees
Tutors get notified when potential tutees are available the area.
Send out notifications to local tutors
Tutees are notified when potential tutors are available in the area.
Tutoring session is scheduled in real-time
Minimal Waiting
Tutoring session occurs
Tutor receives payment
Tutee rates tutor
Initial Setup (Before usage)

Figure 4. Tutor Dash Process Flow

Tutors and tutees must download the app, and then sign up for an account. Once membership has been established, users will then be notified about other potential compatible users based on their login status (i.e., logged in as tutor or logged in as tutee).

Users will be able to search for other users depending on which discovery mode they are currently viewing. This feature will be discussed later in Section 4.2.2 and illustrated in Figure 8. Searching will be one way of finding other potential tutors or tutees, but Tutor Dash will also implement a notification system that will alert users when potential compatible matches are found. This behavior will be discussed later in section 4.2.2. By default all notifications will be set to ‘on’, but users will have the ability to configure their notifications to disallow certain alerts (see Figure 12).

2.1. Objectives and Goals of Tutor Dash

The ultimate goal of Tutor Dash is to fill the gap in the private tutoring market by providing university students across several campuses with an intuitive tool that they can utilize to find private tutors and/or find new clients to tutor. Tutor Dash aims to accomplish this goal by providing several features that will automate many of the decisions and processes that students currently face when searching for tutors/tutees. These features include initiating searches, determining compatibility, and setting up payments. Long-term goals for Tutor Dash include contributing to the lowering of DFWI rates across several universities, as well as establishing long-term repeat customers.

2.2. Features and Capabilities

Tutor Dash will consolidate features from various platforms that already exist in the private tutoring market. Examples of such features include a rating system for tutors, searching based on predefined parameters, and in-app payments. However, Tutor Dash’s implementation of these features will be handled differently in an attempt to optimize and automate these processes. In addition, Tutor Dash will also implement features not present in any other product or platform currently available on the market such as real-time qualification and pay-rate calculation.

The user-base of Tutor Dash will be constrained to only identifiable university students. This will allow users to feel more comfortable when using the app since they will know that all other users are in a similar demographic and situation. This check will be handled when a new user attempts to sign up with Tutor Dash. The user will be required to provide a valid university-affiliated email address in order to sign up. The user must also confirm that the provided email is in fact theirs before they are allowed to use the app.

Tutors who sign up with Tutor Dash will be qualified within seconds. When signing up for Tutor Dash, users will have the option to become a tutor right away. This feature will also be available later if the user does not want to go through this process upon signing up. To become a tutor, users will be required to provide an official university academic transcript, and after this transcript has been analyzed, it will generate a list of eligible courses based on grade that the user will be able to offer tutoring services in. The user will be able to choose to any number of courses in this list, and offered courses will have the ability to be modified later if desired.

Both tutors and tutees will be granted a rating system. Tutors will have two rating systems: one to encapsulate their average rating amongst all tutored courses, and one for each of their tutored courses. This distinction is important to make because tutors may be more effective tutoring certain courses than others. Tutees will also be granted one rating system to encapsulate their average rating amongst all course tutoring sessions that they attended. This tutee rating scheme will be implemented in an attempt to deter absences from tutoring sessions.

Similar to how availability works in Uber, in Tutor Dash, both tutors and tutees will have the ability to toggle their availability. This will allow users to see when other users are available to chat or schedule a meeting. More importantly though, toggling availability from ‘off’ to ‘on’ will signal an alert to other users who are potentially compatible. This will allow users to find who they are looking for faster than just simply searching.

Any course that a university offers will be a tutorable course option in Tutor Dash. Tutees will search for tutors to help them with particular courses, so by allowing every course at any given university to be tutorable, this will help supplement the limited scope of course tutoring programs that their university provides.

Tutors and tutees will be allowed to schedule meetings at any time. By implementing this feature, students will not have to rely on the inflexible hours of university tutoring programs in order to receive course-specific tutoring help. Meetings will also be treated as calendar events so that users can visually see their upcoming schedule before determining if a new meeting event will/will not conflict.

Tutor Dash will implement a unique feature where users will receive notifications based on potential compatible matches. When a user A sets his/her availability to ‘on’, if another user B is looking for users that fit the criteria for user A, then this will be considered a potential compatible match, and user B will be notified that A is available. This feature will apply to both users who are tutors and users who are tutees.

Users will be able to communicate within Tutor Dash via chat. A chat will be implemented so that users can discuss topics with each other in real-time before deciding to schedule a meeting. Chat history will also be stored so users can revisit conversations.

Tutors will have the ability to deny meeting requests. When a tutee sends a request to a tutor for a meeting, the tutor will have the option to either deny or accept this request. Not all meetings will fit a tutor’s schedule, so it is critical that tutors have the option to deny requests for meetings.

In an effort to maintain competition in Tutor Dash and prevent monopolization of the market, pay-rates for each tutor will be calculated automatically. This means that tutors will not be able to set their own pay-rates, rather an algorithm will take into account several parameters relating to the current market environment such as course demand and tutor rating, and use these to generate a competitive pay-rate for that tutor. Pay-rates will be supplied for each course that a tutor offers, and rates will be updated synchronously at least twice a day.

To accommodate the needs of all students, tutors and tutees will be able to attend meetings in-person or via web conferencing. When tutors decide to offer new courses, they will be required to select whether they want to offer tutoring sessions for this course in-person, via web-conferencing, or both. These delivery methods will be have the ability to be modified later if desired.

Users will be able to pay other users within the scope of Tutor Dash. Before users are allowed to schedule meetings, they will be required to set up at least one payment method. For outgoing payments, a user will be able to select their credit/debit card, digital wallet, or PayPal as a payment method, but for incoming payments, users will only be allowed to select their PayPal as the method. Though the functionality for payments will be included in-app, transactions and payment credential storage will be handled by a reliable third-party service named Braintree.

[This space intentionally left blank]

2.3. Major Functional Components

There are several major functional components of Tutor Dash. These components have been broken up into three categories: software, hardware, and entities. Software components refer to software that Tutor Dash will either invent or implement using an existing API, hardware components refer to hardware specifications of the target devices that will allow Tutor Dash to run its software, and entities refer to components that may or may not be software but are vital to the functionality of Tutor Dash. Figure 5 illustrates how all of Tutor Dash's major functional components will be interconnected.

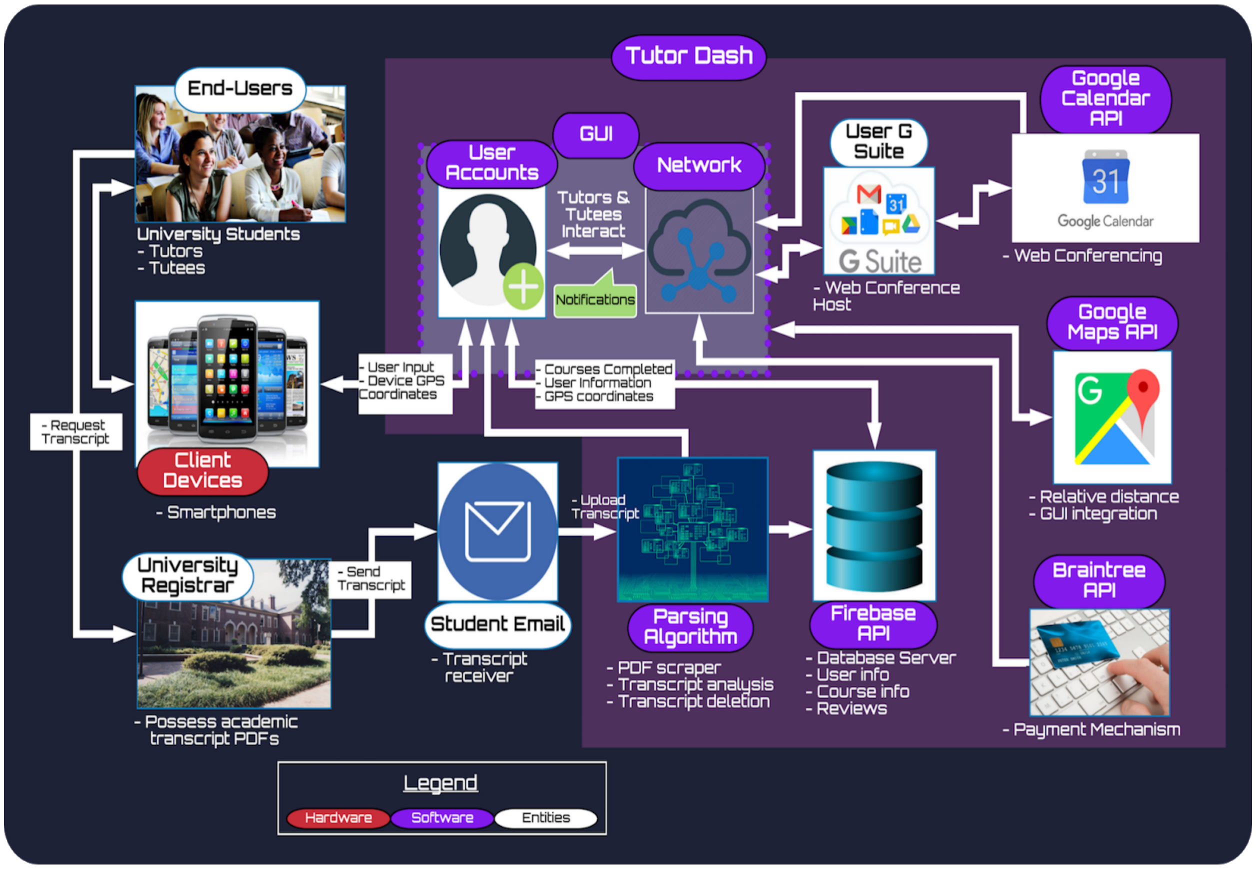


Figure 5. Major Functional Component Diagram

Tutor Dash has an extensive back-end. To do this, it will leverage several well-documented APIs suited for Android such as Firebase, Braintree, Google Maps, and Google Calendar.

2.3.1. Software

There are several software APIs that Tutor Dash will be leveraging. In particular, Firebase will be used extensively throughout Tutor Dash. Firebase is a collection of scalable tool kits and services that many modern mobile and web apps use as their primary method of storing data. Firebase's Cloud Firestore API will serve as Tutor Dash's database server and Firebase Authentication will be used to implement boiler-plate membership features such as logging in and out and password resets. For development, Firebase Test Lab will also be leveraged to test for app bugs and simulate user interactions.

Braintree is another API that Tutor Dash will rely on heavily. Braintree is a reliable third-party service that several big-name apps such as Uber and Poshmark use to handle their in-app transactions. Braintree will handle all payment transactions between users in Tutor Dash and all payment credentials will be stored in Braintree's "Vault". When users add payment methods to their Tutor Dash account, Braintree will be the service storing these methods, and when users authorize payments, Braintree will handle the transactions.

Tutor Dash will also make use of multiple Google APIs. Google Calendar will be used so that users can add and/or modify Tutor Dash events on their personal Google Calendar. This API will also be used for meetings via web conferencing since Google Hangouts will be the streaming service of choice. Google Maps will also be used in Tutor Dash. Some users may want to know how far other users are away from them, so by leveraging the Google Maps API, this will allow not only for these distance calculations to be easier, but it will also allow for a visual google map view to be embedded within UI of the Tutor Dash application.

Some new pieces of software will also be invented for Tutor Dash. For the back-end, a parsing algorithm will serve as the primary component for qualifying tutors in-real time. Users will upload their academic transcript, and this algorithm will determine which courses that user is eligible to tutor. It will also add new courses to the database to give users searching for other users in the app a better experience. Another algorithm unique to Tutor Dash will be the pay-rate calculator. This algorithm will automatically calculate the most competitive rate that a tutor can charge for any course they offer and subsequently set that as their pay-rate. For the front-end, an entire user interface will be built using XML layout files. This UI will be custom-made, and it will wrap the user account and network components of Tutor Dash by providing an intuitive experience that will not require much learning to use.

2.3.2. Hardware

Tutor Dash will primarily be a cloud-based application, but it will require a physical device to host the application. Devices hosting Tutor Dash will be required to run at least Android Nougat OS v.7.0 and include GPS capabilities in order for the app to function properly.

2.3.3. Entities

There are a few entity components that Tutor Dash will make use of in order to achieve its success. Tutor Dash will be developed for a particular set of end users (i.e., university students), and these users will be considered entities, as they are vital to Tutor Dash's process flow illustrated in Figure 4. These users will be university students seeking tutoring services and/or students seeking academic assistance. The users will interact with other entities such as their student email (for signing up), and their university registrar (for requesting an academic transcript). The last identified entity is the user's G Suite. G Suite will be used in Tutor Dash in order to connect to the user's Google Calendar so that Tutor Dash meetings can be placed onto the user's schedule.

3. Identification of Case Study

Tutor Dash will be developed solely for two groups of users that serve as counterparts to each other. The first group of users are students seeking academic assistance in the private tutoring market. These users may be seeking help routinely or occasionally. The second group of users are university students looking to provide tutoring services. Students included in this group may already be private tutors simply looking to expand their client-base, or they may be students who are looking to earn extra income in their spare time.

By providing supplemental tutoring for specific university courses, even though Tutor Dash will not be university-affiliated, it will effectively expand the scope of tutoring services provided by any given university. Potentially, if enough tutors were to sign up for memberships, Tutor Dash could expand this scope to cover 100% of all university-offered courses. Universities can also benefit from Tutor Dash even though they will not be directly involved. DFWI rates would most likely decrease as usage of the application increases since students will be more engaged in their course material. This result would not only reflect positively on universities, but it would also reflect positively on students as they would save on tuition costs due to not having to repeat courses.

Tutor Dash also has the potential to establish long-term repeat customers at several different universities. The rating systems and core features that Tutor Dash will implement will promote users to return to the app. Similar to Uber, with Tutor Dash, as users improve their ratings, so do their benefits. Tutees with high ratings will have a higher likelihood of being granted tutoring sessions, and tutors with high ratings will receive higher calculated pay-rates.

4. Tutor Dash Prototype Description

The prototype for Tutor Dash will be very similar to the real-world product (RWP), however, there will be one major difference; the prototype will be limited to the scope of Old Dominion University students, while the real-world product assumes several universities. Old Dominion University will serve as the test environment for the application, but the development will assume that the product will be expanded upon in the future. Because Tutor Dash will be implementing scalable APIs, expanding the app's user-base in the future will not be a difficult task to complete. Table 1 illustrates a direct comparison of the Tutor Dash prototype to the Tutor Dash RWP.

The prototype will include most of the features that will be present in the RWP such as the notification system, real-time scheduling, and automatic payment calculations, but there are some features that will be eliminated. Since Tutor Dash is intended to be a user-run application, the real-world product assumes there will be a team of maintainers to moderate activity. The prototype will not include a team of human resources, so as a result, features such as banning malicious users, reporting/appealing false negative reviews, and awarding bonuses/incentives to long-term customers will not be present in the prototype.

Some features of the prototype will be implemented, but they will only be partially functional. For example, pay-rates will be calculated synchronously at least twice a day based on several parameters present in the current market environment, but the actual values of these parameters will not be known during the prototyping phase since the values are dependent on user activity. Another example of a partially functional feature that will be present in the prototype is refunding. In the RWP, refunds would be granted manually on a user-by-user basis, so it is likely that the prototype will include this feature, but it will not be made of any use since it will require a team of moderators to fully implement.

[This space intentionally left blank]

Feature	RWP	Prototype
On-the-fly tutor qualification based on transcript	Fully-Functional	Fully-Functional
University student verification based on email	Fully-Functional	Fully-Functional
Search results tailored based on tutor/tutee mode	Fully-Functional	Fully-Functional
Real-time scheduling	Fully-Functional	Fully-Functional
Weighted ratings for every course	Fully-Functional	Fully-Functional
Reviews and comments on user profiles	Fully-Functional	Fully-Functional
In-app payments/deposits (any transactions)	Fully-Functional	Fully-Functional
In-app messaging/history of conversations	Fully-Functional	Fully-Functional
Web conference and in-person meeting support	Fully-Functional	Fully-Functional
Relative distance user A is from user B appears in query	Fully-Functional	Fully-Functional
Night mode	Fully-Functional	Eliminated
Automated pay rate calculation for every course	Fully-Functional	Partially Functional - Mean & std. dev. of pay-rates will need to be mocked up
Reporting features	Fully-Functional	Partially Functional - Users can report, but no action will occur
Re-authentication when navigating back into app	Fully-Functional	Fully-Functional - However, this feature may disrupt the user experience
Refunds due to poor experiences	Fully-Functional	Partially Functional - Most likely, this will not be automated, but it still will exist
Free sessions/monetary bonuses	Fully-Functional	Eliminated
Blacklisting of users	Fully-Functional	Partially Functional - Capabilities will be implemented but not used
Support of multiple universities	Fully-Functional	Eliminated
Cross-platform support	Eliminated	Eliminated
Firebase console linked to test suite(s) with mockups	Eliminated	Fully-Functional

Table 1. RWP vs. Prototype

The real world product is not too much different than the development prototype as many of the proposed features will be fully functional in both. However, the prototype will need to have some data mocked up to simulate various user accounts.

4.1. Prototype Architecture

Since its inception, the Tutor Dash product was intended to be scalable. By selecting reliable and scalable third-party APIs that will serve as the application's core, the prototype will be designed to scale well in the future if necessary. Because of this, the architecture of the prototype will remain mostly the same as the RWP. Figure 5 describes the major functional components of the Tutor Dash RWP, but this same diagram can be applied to the prototype with only a couple of exceptions. The entities for both university registrar and end users will be constrained from general universities to solely ODU.

4.1.1. Prototype Hardware

Like the Tutor Dash RWP, the Tutor Dash prototype will require hardware to run. The only hardware constraints that will be required for the prototype is that devices will be required to run Android Nougat v.7.0 or higher and have GPS capabilities. These devices may be physical or virtual machines, and the prototype will utilize both in its development process.

4.1.2. Prototype Software

The APIs that the prototype will leverage will be exactly the same as the RWP. By doing this, it will allow the prototype to not only be scalable, but it will also allow for easy expansion when adding new features since the APIs will remain constant.

The prototype will also implement five major algorithms which all utilize some third-party API. The PDF transcript parsing algorithm will implement a PDF parsing library named PDFBox, and it will allow users to upload their official academic transcript into the app so they can become qualified to tutor courses in which they received above-average marks in the past. A relative distance calculating algorithm that leverages the Google Maps API will be used to show approximate locations in a visual display that other users are from any given user. The pay-rate algorithm that utilizes real-time Firebase APIs will compute pay-rates synchronously based on various market parameters for every course that a tutor offers in an attempt to control monopolization and stimulate demand. A web-conference creator algorithm will be utilized by implementing the Google Calendar API to generate a calendar event on a user's Google Calendar with Hangouts enabled to serve as the meeting medium. There will also be an algorithm that will utilize both Braintree and Firebase to handle automatic payments based on the numerous outcomes that may occur before, during, and after meetings.

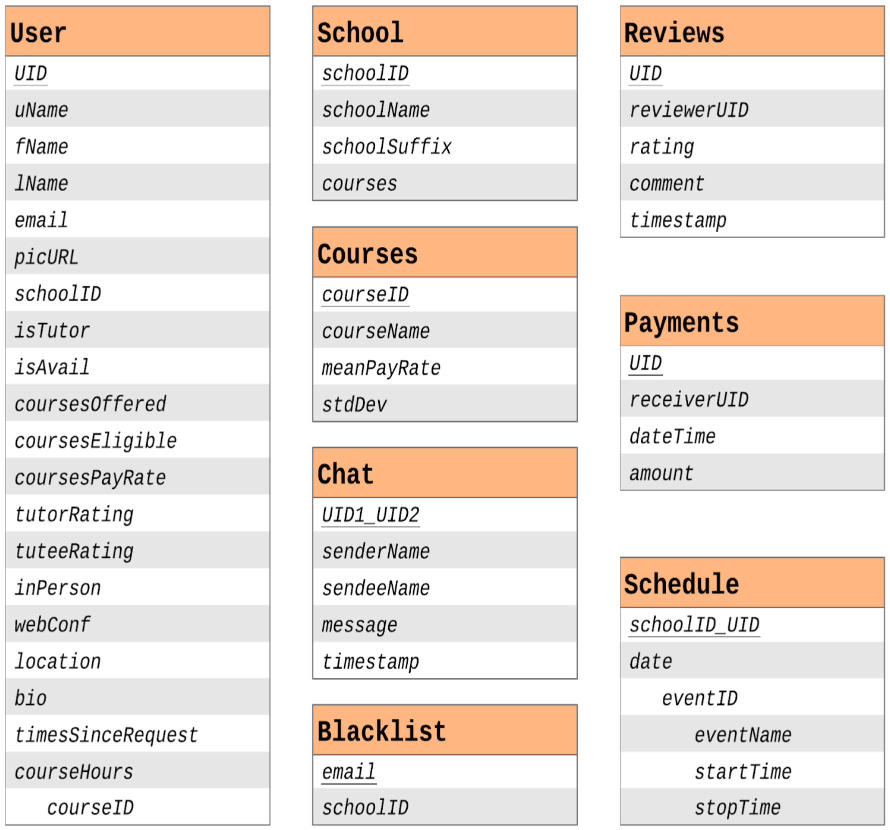
4.2. Critical Design Components

The entirety of Tutor Dash can be broken down into three critical design components. These components are the database, UI/UX, and algorithms. Each component will serve its own purpose within Tutor Dash. The database will act as the network and server that users will communicate and connect to, the UI/UX will serve as the font-end of the app to deliver content in an intuitive fashion, and the algorithms will serve as the back-end to implement the unique features that Tutor Dash will present.

[This space intentionally left blank]

4.2.1. Database

The primary database that Tutor Dash will be using is the Cloud Firestore provided by Google's Firebase. Unlike a relational SQL database, Cloud Firestore is set up to resemble a cloud file system by mimicking a document-like or JSON-like tree directory structure. The Cloud Firestore that Firebase provides is used by several well-known applications such as Lyft, Venmo, and Trivago due to its intuitive file structure and well-documented APIs. Another advantage of this database is that content can be updated and modified in real-time, so delays after updates will not be a concern. Given that Tutor Dash will be a mobile application with plans to scale up in the future and have content modified in real-time, this type of database will be optimal. A more comprehensive look at how Tutor Dash's database will be structured is illustrated in Table 2.



User
UID
uName
fName
lName
email
picURL
schoolID
isTutor
isAvail
coursesOffered
coursesEligible
coursesPayRate
tutorRating
tuteeRating
inPerson
webConf
location
bio
timesSinceRequest
courseHours
courseID

School
schoolID
schoolName
schoolSuffix
courses

Courses
courseID
courseName
meanPayRate
stdDev

Chat
UID1_UID2
senderName
sendeeName
message
timestamp

Blacklist
email
schoolID

Reviews
UID
reviewerUID
rating
comment
timestamp

Payments
UID
receiverUID
dateTime
amount

Schedule
schoolID_UID
date
eventID
eventName
startTime
stopTime

Table 2.
Database Schema

Unlike a traditional relational SQL database, Firebase does not utilize Entity-Relationship Diagrams. This table shows each JSON document as it exists in the Tutor Dash prototype.

The database schema for Tutor Dash will be structured into several collections of fields known as documents. These documents that Tutor Dash will be using are User, School, Course, Chat, Blacklist, Review, Payment, and Schedule. Each document will have a designated field that will be used to cross-reference another document. These fields are “UID” for User, “schoolID” for School, “courseID” for Course, “UID1_UID2” for Chat, “email” for Blacklist, “UID” for Review, “UID” for Payment, and “schoolID_UID” for Schedule. It is worth noting that traditional primary keys are not enforced in the Cloud Firestore Database, so this allows the primary reference field to change later on if necessary.

4.2.2. UI/UX

The UI/UX or user interface & user experience component of Tutor Dash will be what conveys all of the back-end functionality to the user. This component will be broken down into four distinct phases and two distinct views as illustrated in Figure 6. A phase refers to a collection of the screens that the user will experience in a particular order, while a view refers to a collection of screens that the user can experience in any particular order. Each phase will be visited sequentially, while each view will be visited non-sequentially.

[This space intentionally left blank]

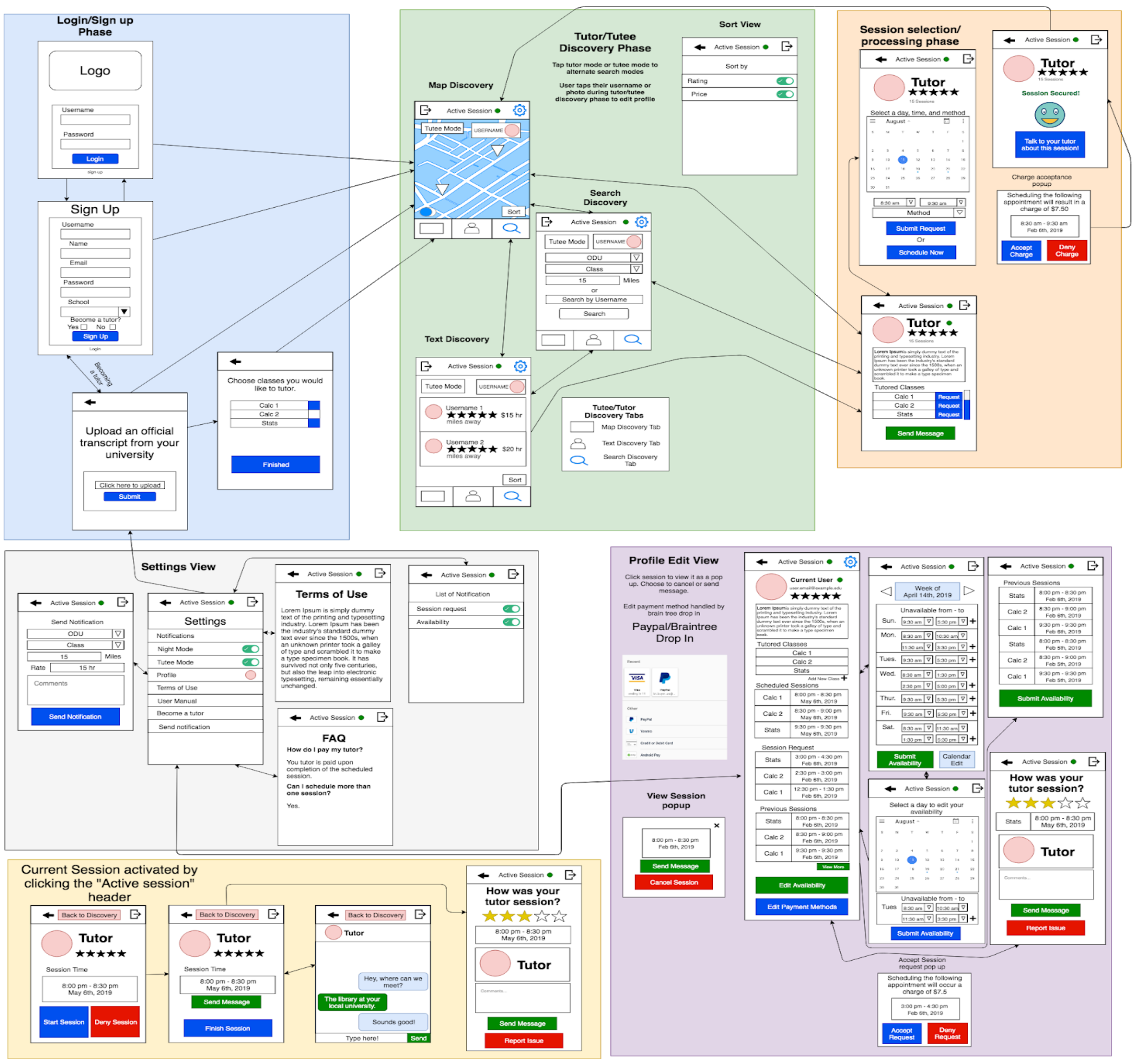

Login/Sign up Phase
Tutor/Tutee Discovery Phase
Map Discovery
Search Discovery
Text Discovery
Sort View
Tutee/Tutor Discovery Tabs
Session selection/ processing phase
Charge acceptance popup
Settings View
Profile Edit View
View Session popup
Current Session activated by clicking the "Active session" header
Accept Session request pop up

Figure 6. UI/UX Full Implementation

The UI/UX will be composed of six phases and two views. Phases are areas of the UI/UX that are visited in a specific order, while views are areas of the UI/UX that are not visited in any specific order.

Phase 1 of the UI/UX will be designated as the login and signup phase. During this phase, users will have the ability to either sign up for a new Tutor Dash account, or login to an existing account. A more in-depth look at how this phase will look on the front end and flow is shown in Figure 7.

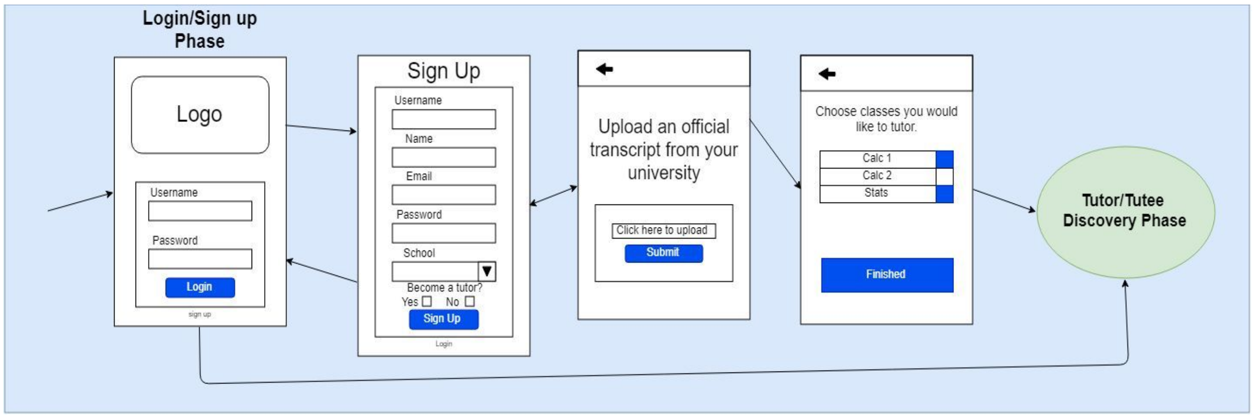


Figure 7. UI/UX Phase 1

This figure illustrates the login/sign up phase. During this phase, the user will either sign up for a new account or log in with an existing account. If they are signing up and wish to become a tutor, then they can provide their transcript during this phase if desired.

When a user starts up the app, he/she will be prompted to login. If he/she has an account, then he/she will login using his/her account credentials and be transferred immediately to the next phase of the UI/UX. However, if the user does not have an account, then he/she will need to sign up. The user will need to provide his/her first name, last name, email, password, and school in order to sign up. The user will also have the ability to sign up as a tutor by selecting a checkbox when he/she creates an account. As long as the user's email domain is valid for the school that he/she selected, an account will be created. If a user selected to become a tutor, then the additional step of becoming qualified for courses is offered as an option. During this step, the user will provide a PDF version of his/her academic transcript and based on the grades present in the transcript, this will generate a list of courses that this user is eligible to tutor. After the user selects the desired courses, sign up is complete and the user will then be transferred to the next phase.

Phase 2 of the UI/UX will be the tutor/tutee discovery phase, and this is the phase where users look for one another. During this phase, users will have the ability to search and filter results based on the parameters and inputs they choose. Users will also have the ability to visually observe approximately where other users who have their location permissions turned on are currently located by toggling between the tutor and tutee view modes. Figure 8 illustrates how all of these screens for this phase will appear and flow.

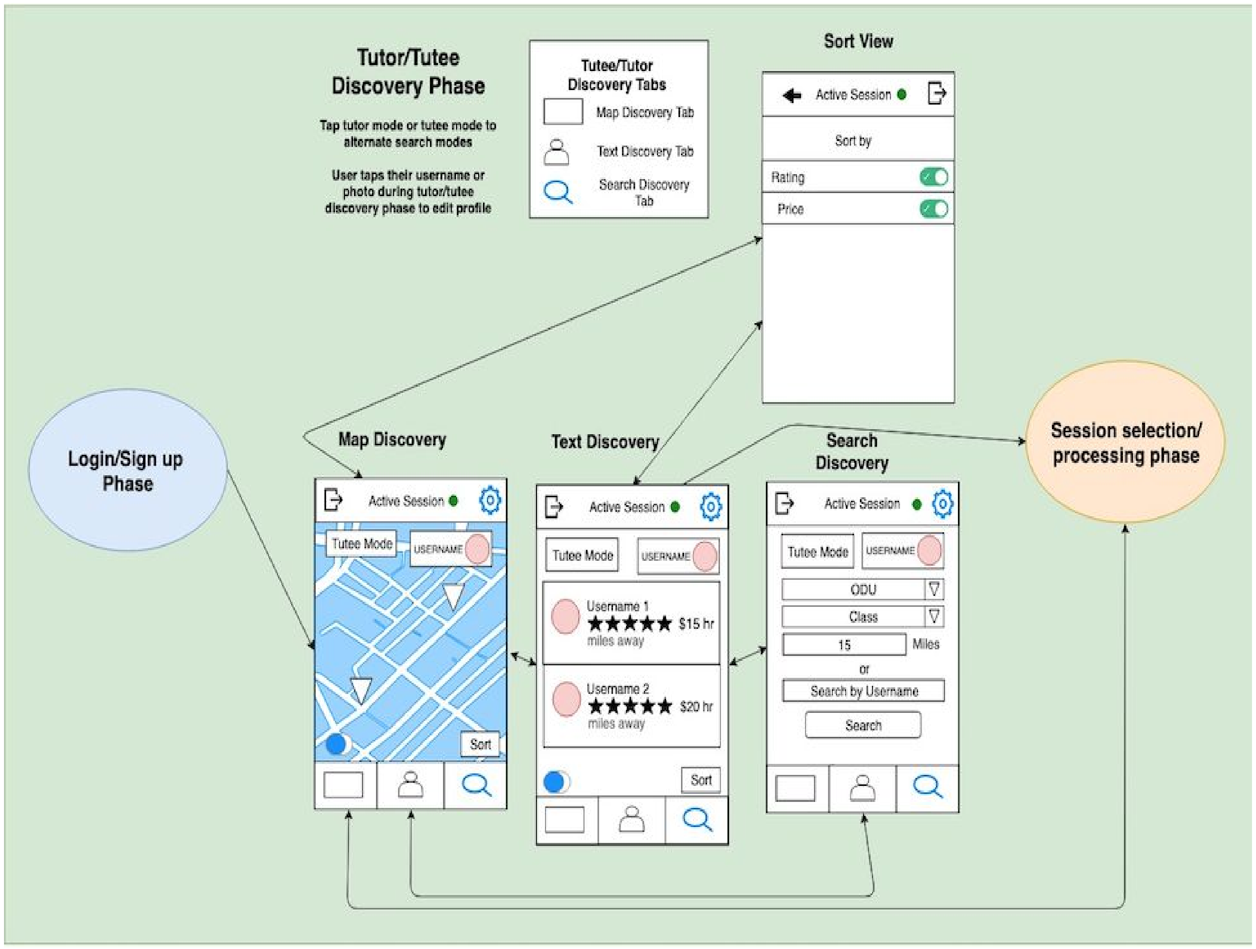

Tutor/Tutee Discovery Phase
Tap tutor mode or tutee mode to alternate search modes
User taps their username or photo during tutor/tutee discovery phase to edit profile
Tutee/Tutor Discovery Tabs
Map Discovery Tab
Text Discovery Tab
Search Discovery Tab
Sort View
Active Session
Sort by
Rating
Price
Login/Sign up Phase
Map Discovery
Text Discovery
Search Discovery
Session selection/ processing phase
Tutee Mode
USERNAME
Sort
Username 1 $15 hr miles away
Username 2 $20 hr miles away
ODU
Class
15 Miles
or
Search by Username
Search

Figure 8. UI/UX Phase 2

This figure illustrates the tutor/tutee discovery phase. During this phase, users will be able to interact with the database by searching for other users based on their own search parameters.

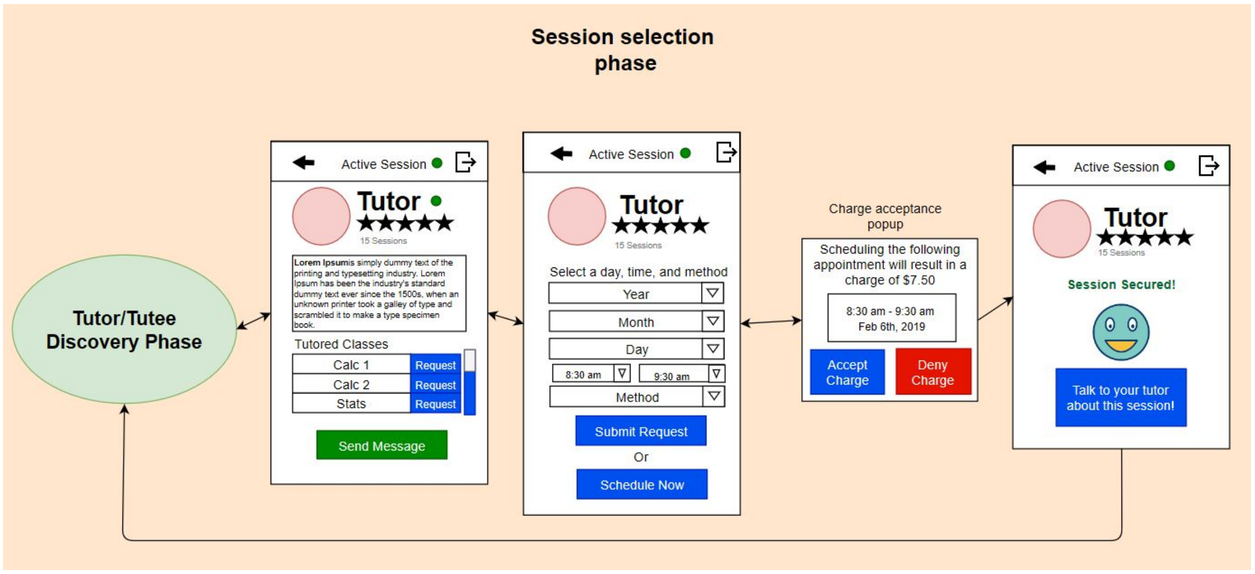
When a user enters this phase, he/she will see a map view. This map view will have both a tutor and tutee viewing mode. Depending on who the user is looking for will determine which users are displayed on the map. If the user is viewing in tutee mode, then this user will be treated as a tutee looking for tutors. If the user is viewing in tutor mode, then this user will be treated as a tutor looking for tutees. If this user sees other users on this map view, he/she can click on that user's icon, and this will expand on the information, such as the rating for that user.

By default, users will be navigated to the map view, but they can navigate to other views if desired. The text discovery view will present the same information that the map view presents, but it will not display the map. Users will be able to browse for nearby tutors/tutees depending on which viewing mode they have selected, and the information displayed for each user in this viewing mode will be the same information that is displayed when a user clicks on another user's icon in the map viewing mode.

The search view will be another way that users can find one another. In this viewing mode, users will be able to query the database for specific criteria such as school, course, and distance. Users can filter these results based on rating and price if applicable. Also present in this search viewing mode is a button that allows users to search for other users by username. This feature will be especially useful in the case that a user wants to find another user they attended a session with in the past.

Phase 3 of the UI/UX will be the session selection phase, and it is where tutees will be able to decide if potentially compatible tutors will be available for sessions that fit their schedules. As long as users are viewing in tutee mode, they will be able to enter this phase.

Figure 9 illustrates the front-end appearance and flow that will occur between the various screens included in this phase.



Lorem Ipsumis simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Scheduling the following appointment will result in a charge of $7.50

Figure 9. UI/UX Phase 3

This figure illustrates the session selection phase. During this phase, the user will create a request for a session that fits theirs and the tutor's schedules. Once this request is sent, the tutor will be notified.

The precondition for a tutee entering this phase is that the tutee must have selected a tutor for which he/she will try to schedule a session with. The tutee with be initially navigated to a screen which will show all the courses that this tutor offers. On this screen, there will be two options presented. The tutee will either be able to select a course for which he/she wishes to schedule a request for, or he/she will be able to send a message to the tutor. If the tutee sends a message, this message will be sent to the tutor and he/she will be notified immediately.

If the tutee selects a course to request, then he/she will be navigated to a screen which prompts him/her to either select an available time or try to schedule the meeting immediately. By selecting the option to schedule a meeting immediately, this will find the next available time slot that exists in the tutor's availability. If the tutee submits a request that conflicts with either theirs or the tutor's schedule, then he/she will be alerted, but this will not prohibit the request from being sent. The corresponding tutor will receive the tutee's request immediately, and it is at this point that he/she can either accept or deny the request. Once the tutor responds, the tutee will receive this response, and if it was an "accept" response, then this event will be added to both users' Google Calendars and in-app schedules.

Phase 4 of the UI/UX will be the active session phase. Users will enter this phase at the time that a session is set to begin. Users will be reminded prior to the session's start time, and once the session begins, a variety of events will occur. Figure 10 shows the active session phase and how it will connect to the rest of the phases in the UI/UX.

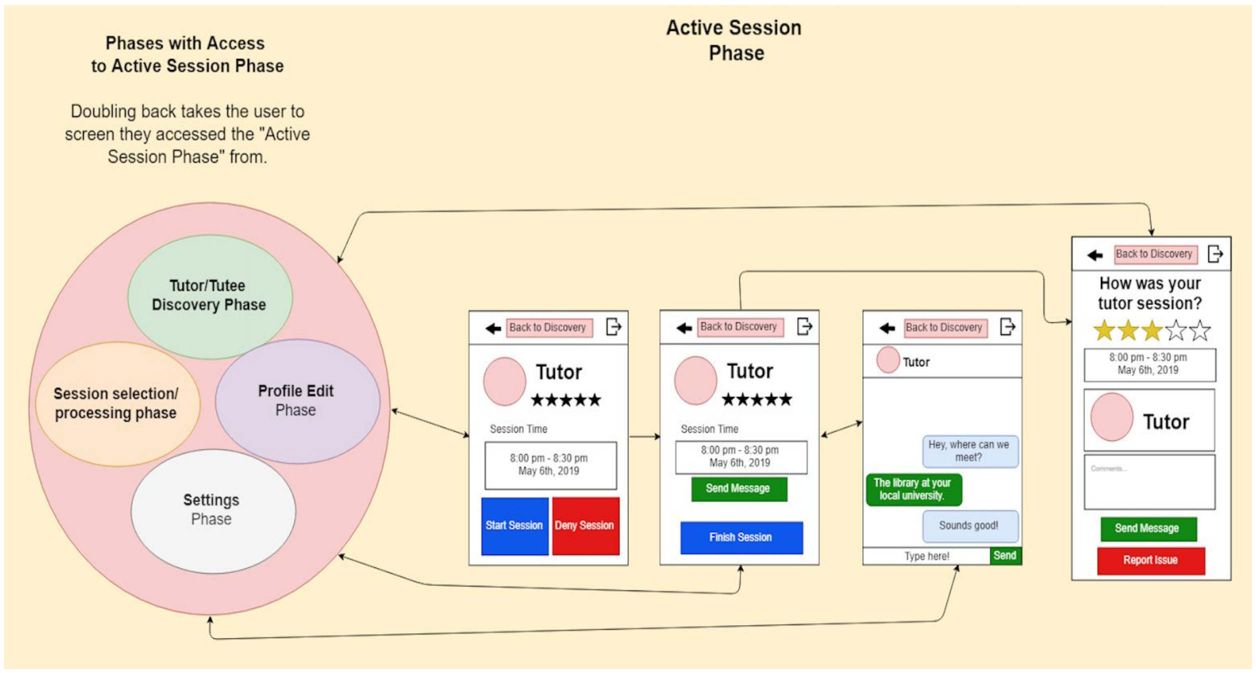

Phases with Access to Active Session Phase
Doubling back takes the user to screen they accessed the "Active Session Phase" from.
Active Session Phase
Tutor/Tutee Discovery Phase
Session selection/ processing phase
Profile Edit Phase
Settings Phase
Back to Discovery
Tutor
Session Time
8:00 pm - 8:30 pm May 6th, 2019
Start Session
Deny Session
Send Message
Finish Session
Hey, where can we meet?
The library at your local university.
Sounds good!
Type here!
Send
How was your tutor session?
Comments...
Report Issue

Figure 10. UI/UX Phase 4A

This figure illustrates the active session phase. When a session has started, both tutor and tutee will be prompted to acknowledge that both users are in the meeting. The session will then proceed with a timer, and after the session ends, the tutor will be paid. Both users will then be prompted to rate the corresponding user.

Both a tutee and tutor who have a meeting scheduled together will enter this phase. When the meeting is set to begin, both users will receive an alert which will navigate them both to a screen prompting them to acknowledge the start of the session. This acknowledgement is known as the "handshake agreement", and it is discussed later in section 4.2.3 and illustrated in Figure 18. After both users acknowledge the session, both of their availabilities will be turned off automatically. This means when other users are viewing the app in the map viewing mode, these two users who are currently in a meeting will not show up on the map. Figure 11 illustrates how this behavior affects the front-end of the map viewing mode from Figure 8.

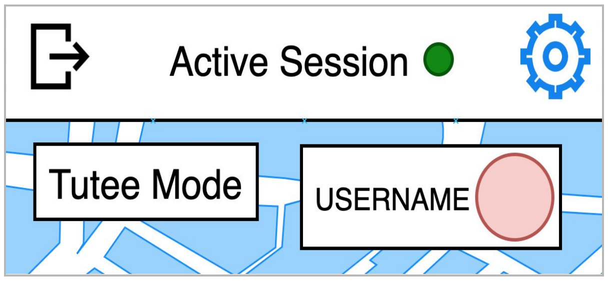


Figure 11. UI/UX Phase 4B

This figure illustrates an important feature of the active session phase which is the ability for users to toggle their availability manually, but they will become unavailable during a session. When compatible tutors become available, tutees are notified, and vice versa.

A session will end once both users agree to end it. Both users will do this by clicking on a "Finish Session" button, and once both users have clicked it, they will then be navigated to a screen where they must rate each other based on performance. After the ratings for both users have been provided, the tutor will be automatically paid based on his/her pay-rate for that course and the length of the session. A more in-depth explanation as to how payments will be handled on the back-end are discussed later in section 4.2.3.

View 1 of the UI/UX will serve as a collection of screens to store several different options. In this view, users will be able to customize and examine several pieces of miscellaneous information relating to Tutor Dash's functionality and user experience. Users will be able to navigate to their settings from anywhere in the app once they are logged in. Figure 12 illustrates this settings view and how it connects to the rest of the user interface. The settings available to all users include the following:

- Modify notification alerts to receive more or less alerts than the default
- View Google Calendar and/or in-app calendar for upcoming events
- View member profile
- View Terms of Use agreement
- View documentation on how to use Tutor Dash
- Manually resend local alerts out to potentially compatible users
- Update or add tutor eligibility status

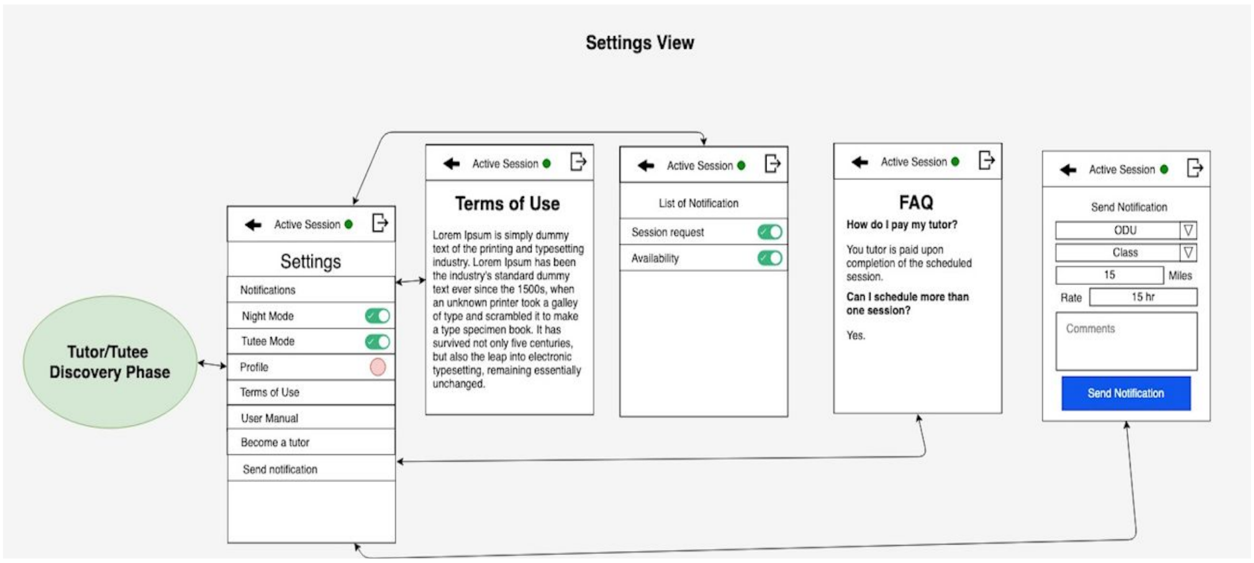

Settings View
Tutor/Tutee Discovery Phase
Active Session
Settings
Notifications
Night Mode
Tutee Mode
Profile
Terms of Use
User Manual
Become a tutor
Send notification
Terms of Use
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.
List of Notification
Session request
Availability
FAQ
How do I pay my tutor?
You tutor is paid upon completion of the scheduled session.
Can I schedule more than one session?
Yes.
Send Notification
ODU
Class
15
Miles
Rate
15 hr
Comments
Send Notification

Figure 12. UI/UX View 1 - Settings

The settings view will give users the ability to modify notifications, toggle night mode (not present in prototype), view their calendar, view their profile, review the terms of use agreement, view the user manual, sign up to become a tutor, and send out local availability alerts.

View 2 of the UI/UX will serve as the user profile view, and it will be where users can access their public and private information associated with their membership. When a user logs in, he/she will be able to observe this view by navigating to it from the settings. When viewing other users' profiles, private information will be hidden, while public information will remain readable to anyone. This public information will include username, email, ratings, tutored classes, scheduled sessions, session requests, previous sessions, and availability. Figure 13 illustrates how the front-end of this view will be laid out.

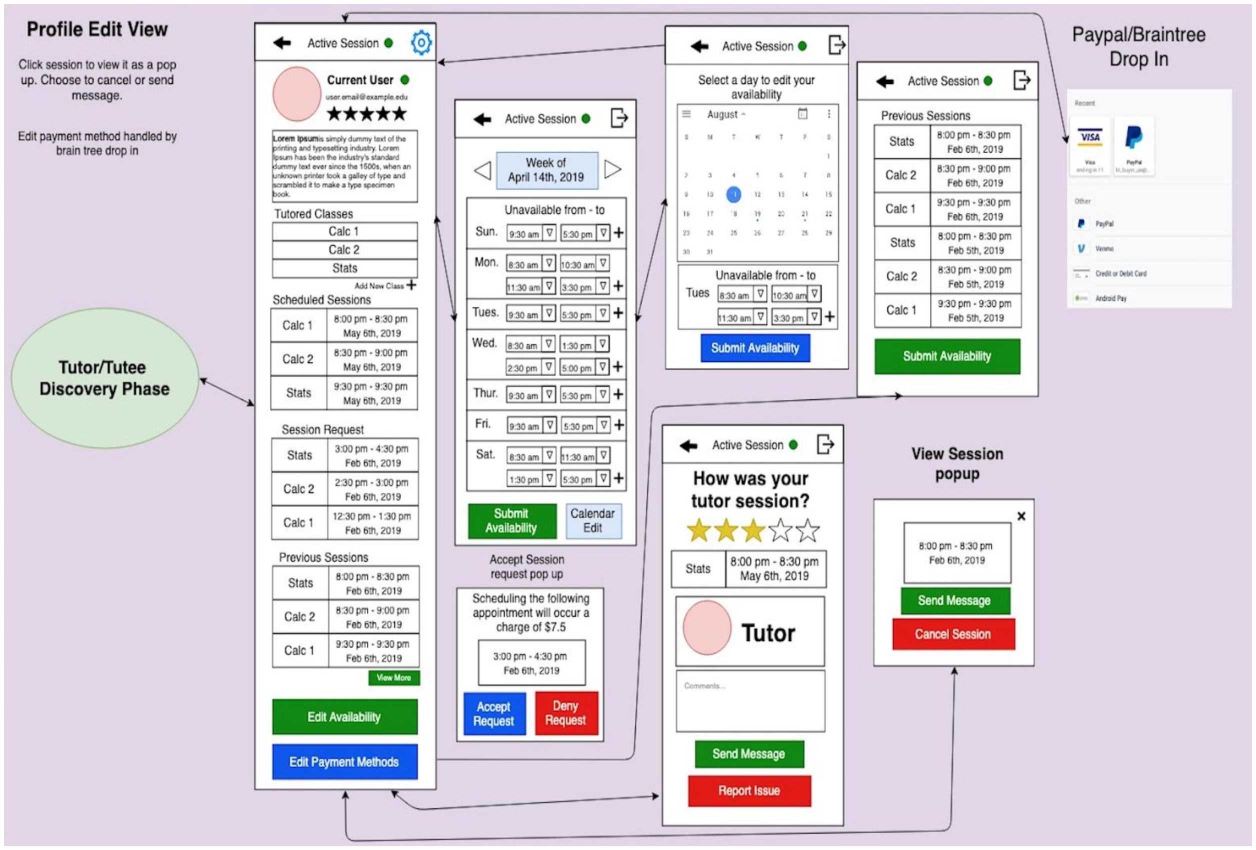


Figure 13. UI/UX View 2 - User Profile

The profile edit view will allow the user to view and/or modify public and private information stored in their user profile. All information is viewable, but not all information on the profile is able to be modified (e.g., rating, previous sessions).

4.2.3. Algorithms

The features that Tutor Dash plans on offering will be implemented using five major algorithms, with each of them serving a distinct purpose. These algorithms include a PDF transcript parsing algorithm, a relative distance calculator, a competitive pay-rate calculator, a web conference appointment creator, and an algorithm to handle various situations surrounding payments.

The PDF transcript parsing algorithm shown in Figure 14 will serve two purposes. The first purpose will be to handle the parsing of a PDF. After a user uploads his/her transcript into the app to be parsed, he/she will then see a list of courses in which he/she is eligible to tutor, and it is the responsibility of this algorithm to read in the text, extract the meaningful course-related information, and finally send it to the front-end to be displayed to the user.

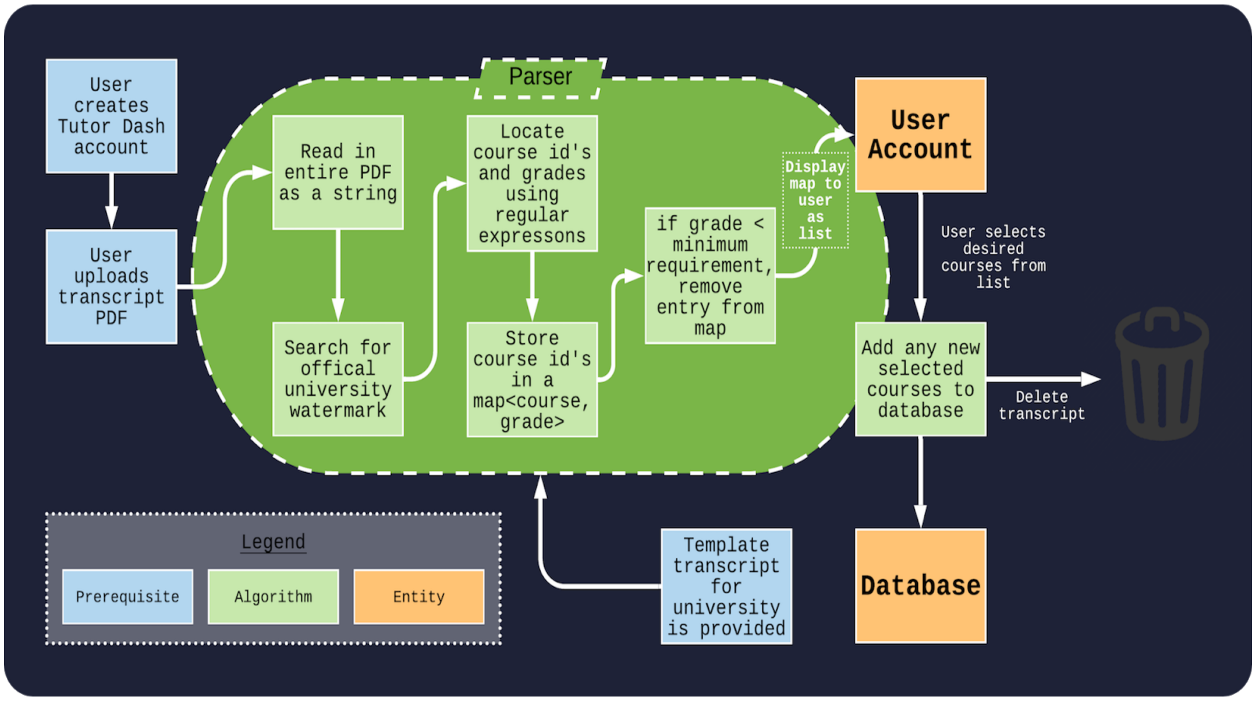

User creates Tutor Dash account
User uploads transcript PDF
Parser
Read in entire PDF as a string
Search for offical university watermark
Locate course id's and grades using regular expressions
Store course id's in a map<course, grade>
if grade < minimum requirement, remove entry from map
Display map to user as list
User Account
User selects desired courses from list
Add any new selected courses to database
Delete transcript
Database
Template transcript for university is provided
Legend
Prerequisite
Algorithm
Entity

Figure 14. PDF Transcript Parser Algorithm Flow

This figure illustrates the PDF transcript parsing algorithm. Transcripts will be parsed as PDFs, and if they are valid transcripts, the user will be qualified to tutor selected courses based on their previous grades present in the text.

The second purpose of the transcript parsing algorithm will be to add new offered courses to the database. In the search view mode of the tutor/tutee discovery phase of the UI/UX shown in Figure 8, users will have the ability to search for other users based on course. Only courses that have at least one available tutor will show up as a selectable option, so it will be important that this algorithm keeps the database up-to-date so that users will not waste their time querying the database for information that does not exist.

This algorithm will be achieved by implementing a Java PDF parsing library called PDFBox to parse the transcript and Firebase APIs to connect to and update the existing database. The parameters for this algorithm will be the university name, the PDF transcript itself, and the minimum qualifying grade that would determine whether a user is eligible to tutor any course found in the transcript.

The job of the relative distance calculator shown in Figure 15 will be to determine the approximate distance that a set of users B who appear in a search query are from a single user A. When A searches and subsequently gets results returned to him/her in the tutor/tutee discover view (Figure 8), as long as A and a user in the set B have both of their location permissions enabled, A will be able to see an approximate distance that this user in B is from A. These locations will also be displayed on the map view in the tutor/tutee discovery view.

[This space intentionally left blank]

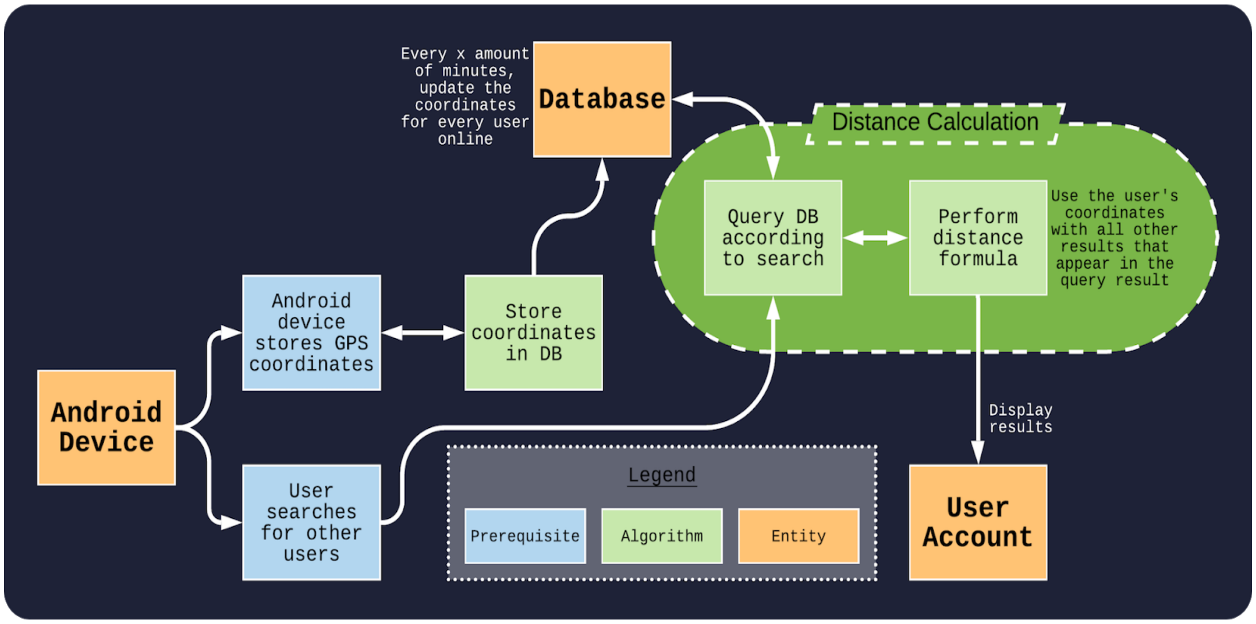

Every x amount of minutes, update the coordinates for every user online
Database
Distance Calculation
Query DB according to search
Perform distance formula
Use the user's coordinates with all other results that appear in the query result
Android device stores GPS coordinates
Store coordinates in DB
Android Device
Display results
User searches for other users
Legend
Prerequisite
Algorithm
Entity
User Account

Figure 15. Relative Distance Calculator Algorithm Flow

This figure illustrates the relative distance calculation algorithm. Coordinates from Android devices will be stored in the database and updated frequently. By using the distance formula, an estimate for distance away other users are from the current user is calculated.

This algorithm will leverage Android location services as well as the Google Maps API to hook into the front-end. The Cloud Firestore API will be used to update locations as often as necessary, and most likely this will be run synchronously at fixed points during the day. The parameters for this algorithm will include time, the time interval between location updates, and the android device GPS coordinates of the user who initiated the search query and every user who appears in the search query result.

The pay-rate calculator described in Figure 16 will be particularly significant to Tutor Dash, as this algorithm will handle the calculating of pay-rates for every course that a tutor offers. By comparing several environment parameters that will be stored in the database, this algorithm will be able to accurately predict the best possible price that a tutor can charge and still receive business in a reasonable amount of time.

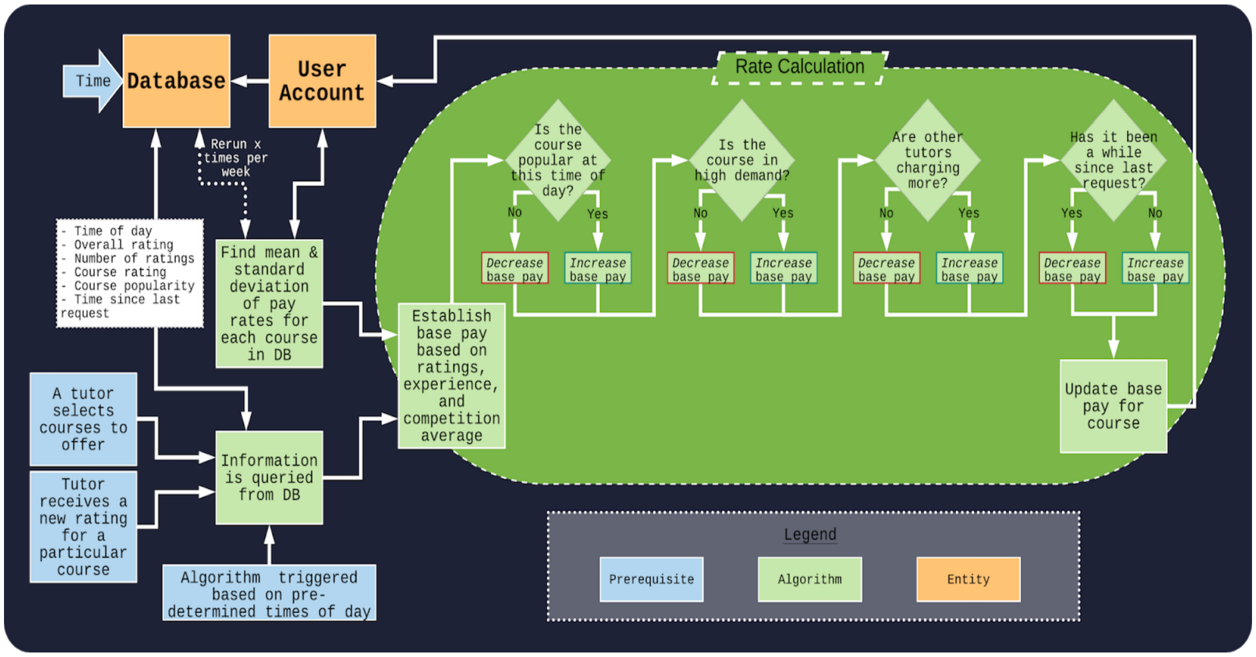

Time
Database
User Account
Rate Calculation
Rerun x times per week
- Time of day
- Overall rating
- Number of ratings
- Course rating
- Course popularity
- Time since last request
Find mean & standard deviation of pay rates for each course in DB
Establish base pay based on ratings, experience, and competition average
Is the course popular at this time of day?
Is the course in high demand?
Are other tutors charging more?
Has it been a while since last request?
No
Yes
Decrease base pay
Increase base pay
Update base pay for course
A tutor selects courses to offer
Tutor receives a new rating for a particular course
Information is queried from DB
Algorithm triggered based on pre-determined times of day
Legend
Prerequisite
Algorithm
Entity

Figure 16. Competitive Pay-Rate Algorithm Flow

This figure illustrates the pay-rate calculation algorithm. Users will not set their pay, rather an algorithm will determine their pay based on course popularity, course demand, other tutors' pay-rates, and time since last request for course tutoring session.

This algorithm will be achieved by implementing several APIs. To determine the demand of any particular course, the mean and standard deviation of pay-rates that other tutors with similar ratings and experience are charging must be calculated. In order to do this, the database must be updated with these values as frequently as possible, but not so often that it would affect the performance of the server. Cloud Firestore will be utilized heavily in this algorithm as it will allow for frequent real-time writes to occur to the database and functionality exists in the API to calculate mean and standard deviations of certain fields present in the database.

There are several parameters that will be required for this algorithm to function. To accurately calculate a tutor's pay-rate for any given course, this algorithm will need the tutor's overall rating, the tutor's course rating, the demand of the course, the tutor's experience level, the time of day, the mean and standard deviation of pay-rates for the course, and time since the last request for the course.

The purpose of the web-conference appointment creator algorithm will be to automate the process of creating calendar events with web conferencing. Illustrated in Figure 9, if a tutor accepts a session request from a tutee, a calendar event will be created and added to both users' Google Calendars. If the session delivery method happens to be via web-conference though, then the extra step of attaching web conferencing to the event will be required. Figure 17 shows the logical process flow for this algorithm.

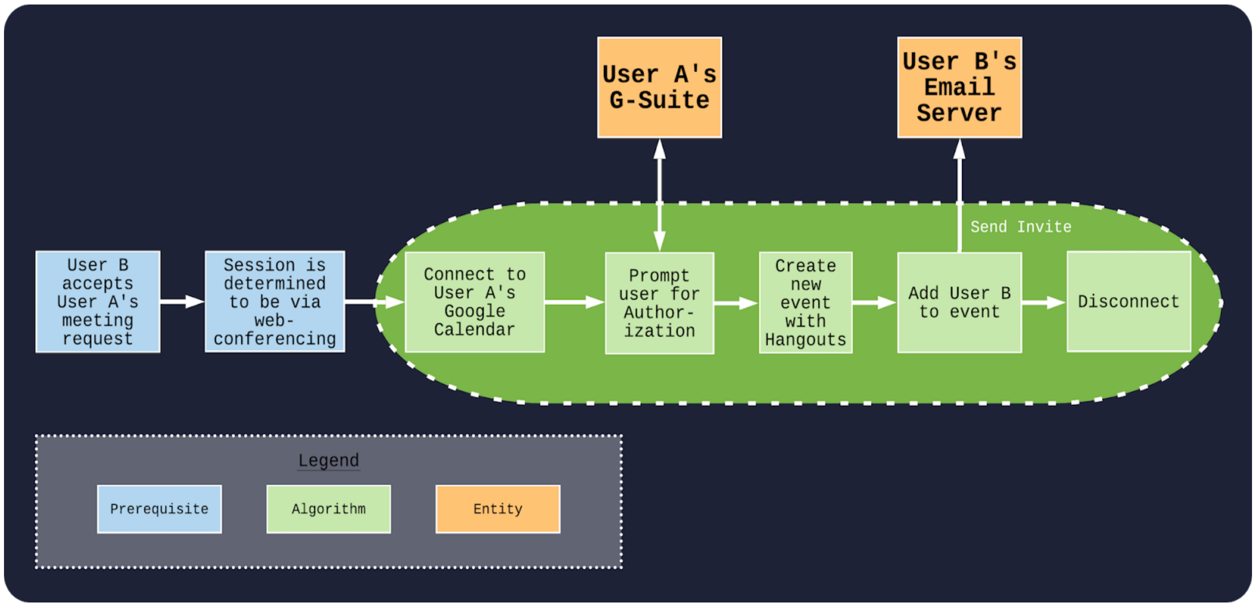

User A's G-Suite
User B's Email Server
Send Invite
User B accepts User A's meeting request
Session is determined to be via web-conferencing
Connect to User A's Google Calendar
Prompt user for Author-ization
Create new event with Hangouts
Add User B to event
Disconnect
Legend
Prerequisite
Algorithm
Entity

Figure 17. Web-Conference Appointment Creator Algorithm Flow

This figure illustrates the web-conference event creator algorithm. If a session is determined to be via online, then an event with Google Hangouts conferencing will be created on the tutor's Google Calendar and sent to the tutee automatically.

Web-conferencing will not be hosted within the scope of Tutor Dash's software, rather it will be handled by Google Hangouts. In order to create events and access a user's Google Calendar, the Google Calendar API will be leveraged so that events can be created and customized programmatically. The parameters for this algorithm are the start/end times of the session and both the tutor and tutee's email addresses.

The final algorithm that will be implemented within Tutor Dash is the algorithm that will automate the process of paying tutors. The prerequisites for this algorithm are that tutors must have an incoming payment method setup in their Tutor Dash account and tutees must have an outgoing payment method set up in their Tutor Dash account. Figure 18 demonstrates the logical flow of how payments will be handled.

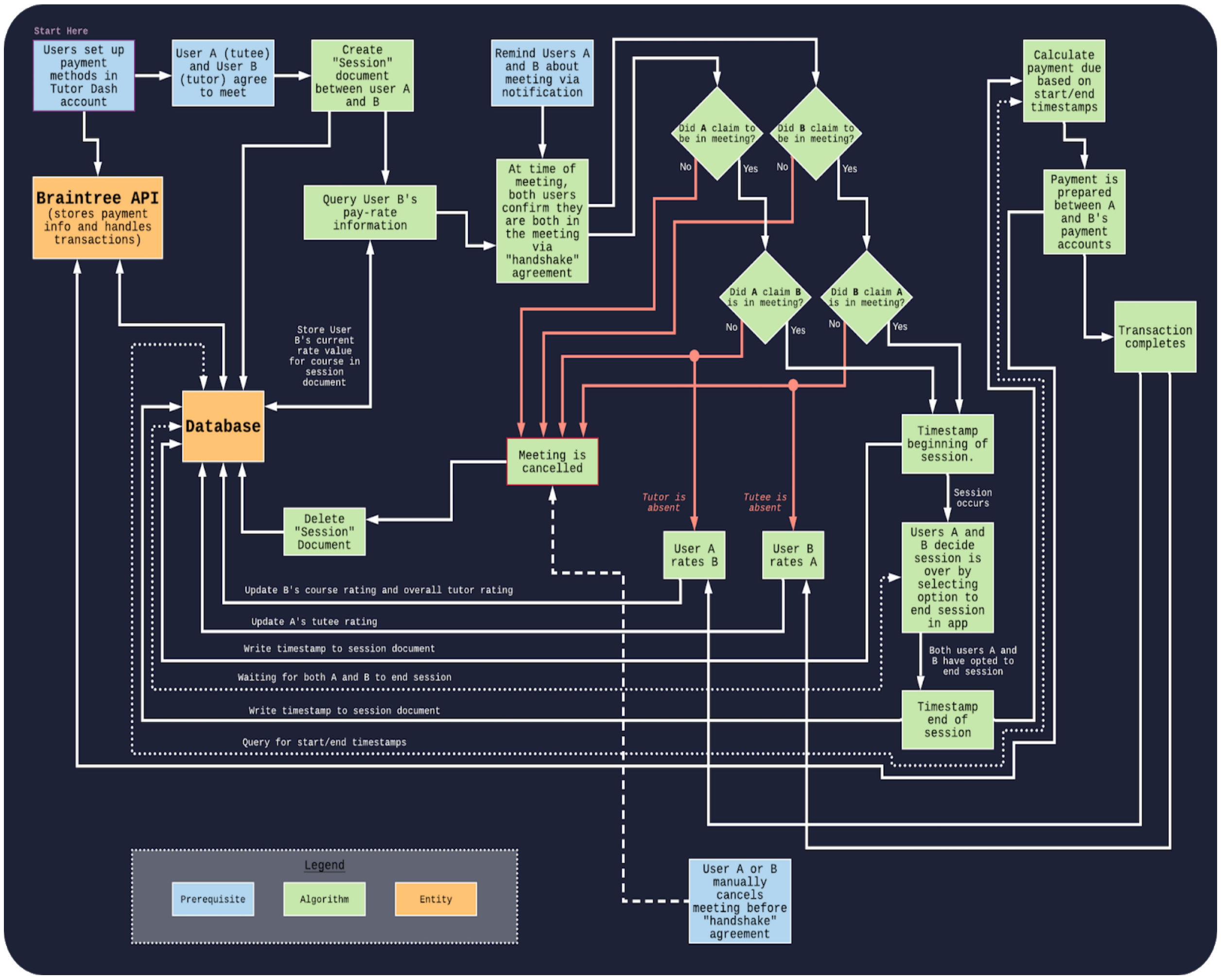

Start Here
Users set up payment methods in Tutor Dash account
User A (tutee) and User B (tutor) agree to meet
Create "Session" document between user A and B
Remind Users A and B about meeting via notification
Calculate payment due based on start/end timestamps
Braintree API (stores payment info and handles transactions)
Query User B's pay-rate information
At time of meeting, both users confirm they are both in the meeting via "handshake" agreement
Did A claim to be in meeting?
Did B claim to be in meeting?
No
Yes
Did A claim B is in meeting?
Did B claim A is in meeting?
Payment is prepared between A and B's payment accounts
Transaction completes
Store User B's current rate value for course in session document
Database
Meeting is cancelled
Timestamp beginning of session.
Session occurs
Tutor is absent
Tutee is absent
Delete "Session" Document
User A rates B
User B rates A
Users A and B decide session is over by selecting option to end session in app
Update B's course rating and overall tutor rating
Update A's tutee rating
Write timestamp to session document
Waiting for both A and B to end session
Both users A and B have opted to end session
Timestamp end of session
Write timestamp to session document
Query for start/end timestamps
Legend
Prerequisite
Algorithm
Entity
User A or B manually cancels meeting before "handshake" agreement

Figure 18. Payment Logic Algorithm Flow

This figure illustrates how payments will be handled in Tutor Dash. To improve security, payment accounts will be handled separately by a reliable and well-documented API called Braintree.

When a tutee is ready to send a session request to a tutor, he/she will be implicitly agreeing on the amount of money that the tutor is charging, so before this request can be sent, a check is performed to make sure enough money is present in the tutee's outgoing payment account. This check will be performed using the Braintree API. Once the session is marked as "finished" by both the tutor and tutee, after both of them have rated one another, the transaction will proceed. The amount that will be taken from the tutee's outgoing payment account and transferred into the tutor's incoming payment account will be calculated based on both the tutor's pay-rate at the time the session request was accepted and the length of the session.

In order to mitigate cancellations and absences, a "handshake agreement" will also be implemented within this algorithm. The ideal case is that both users will attend their meeting on-time, and thus both would complete the "handshake", acknowledging that both users were present and the meeting can proceed as normal. However, this cannot be assumed to happen 100% of the time. If one user is absent from the meeting, and the other user is not absent, then this meeting will fail to proceed. As a result, the user who was absent will be able to be rated poorly by the user who was not absent. If both users are absent from the meeting though, then this event will be handled as a normal cancellation, and neither user will be able to rate the other.

Braintree and Firebase will serve as the fundamental APIs behind this payment handling algorithm. This algorithm requires several instances where data needs to be updated quickly by one user and read instantly by another, so Cloud Firestore will be utilized for this algorithm since the API possesses those capabilities. The parameters for this algorithm will include the tutor's incoming payment account, the tutee's outgoing payment account, the tutor's pay-rate at the time the session was scheduled, and the start/end times of the session.

4.3. Prototype Features and Capabilities

The Tutor Dash prototype will demonstrate many of the same features and capabilities that would be present in the RWP. This prototype will deliver an intuitive and objectively easy-to-use interface that will not require much learning upon using it for the first time. Users can expect a product that will deliver all of the core features present in the RWP. These features include:

- A user-base constrained to ODU students
- Implementation of real-time tutor qualification based on academic history
- Multiple rating systems for both tutors and tutees
- The ability to toggle availability on and off in a similar fashion to Uber
- Allowing any course at ODU to be offered as a tutor-able course in-app
- 24/7 scheduling for both in-person and online meetings
- A notification system that alerts users when other users contacted them
- Automated competitive pay-rate calculations
- A secure method of handling automatic transactions between users
- Implementation of chat and storage of chat history between users

All of these features will be significant in solving the problem. Some features that Tutor Dash will be implementing such as the rating system and scheduling of both online and in-person meetings are nothing new to the market, however, when these features are combined with new features that automate several existing processes like finding other users, setting pay-rates, and getting paid, Tutor Dash succeeds in solving most of the issues that many university students currently face in the private tutoring market.

4.4. Risks and Mitigations

Tutor Dash has several identified risks involved regarding its development. These risks can be categorized into three groups: customer risks, technical risks, and legal risks. Customer risks refer to risks that would directly impact the end-users of Tutor Dash negatively. Technical risks are risks that directly impact the integrity of Tutor Dash's software and/or hardware. Legal risks are risks that the team behind Tutor Dash may face when trying to launch the product into the market. Figures 19 and 20 illustrate all of the identified risks surrounding Tutor Dash.

Probability / Impact	Very Low	Low	Moderate	High	Very High
Very High	T3, T4, L1	T6, L2		C3, C4	
High	T1, C7	C6, T9			
Moderate	C5	T8	C1, C8, T7		
Low	C2, T5		C10, T2		
Very Low		C9			

Figure 19. Risk Matrix Overview

This figure illustrates all of the risks that the Tutor Dash RWP faces. Each of these risks is assigned an id that can be referenced in Figure 20.

[This space intentionally left blank]

ID	RISK DESCRIPTION	MITIGATION(S)
C1	Student finds tutors to be unhelpful	● Rating system ● Payment refunds
C2	Prospective tutors faking their qualificaitons	● Require official transcript from university registrar ● Make tutors only eligible to tutor classes he/she has received a B or higher in
C3	Shortage of tutors	● Give small bonuses to tutors for a limited time (similar to Uber's business model)
C4	Shortage of tutees	● Give free sessions to new users ● Give loyaltee-free sessions for a certain number of usages
C5	Tutor/tutee leaves a false negative review	● Allow users to challenge reviews (requires manual investigation) ● Withhold ratings/reviews until both users agree on justification
C6	Users abuse application; use app maliciously	● Require users to agree to the terms of use agreement ● Blacklist (ban) users who violate the terms of use agreement
C7	Identity theft; non-users impersonate users and/or users impersonate other users	● Re-authentication when navigating to app from outside window ● "Handshake" agreement between users when sessions begin
C8	Participating tutor/tutees don't show up to their scheduled meetings	● Preallocate payments ● Require deposits ● Threat of poor ratings
C9	Users try to book overlapping sessions	● Only allow users to make appointments for times they don't currently have a scheduled session ● Applies to both tutors and tutees
C10	Tutors are not adequately prepared to engage with tutees via web conferencing	● Alert users of the minimum requirements for web conference meetings upon selecting 'web-conferencing' as a tutoring preference.
T1	Payment is not received	● Integrate usage of a 3rd party API designed to handle e-transactions ● Braintree
T2	Difficulty automating the process of reading a submitted transcript	● Define reusable code for general case ● Optimize as more information is discovered
T3	Database server failure	● Use reliable servers maintained by large corporations ● Firebase
T4	Security breach	● Use 3rd party APIs which are already secure
T5	Application is not compatible on all android devices	● Define minimum SDK for weaker hardware phones ● Define normal SDK for normal hardware phones
T6	Network server failure	● Server redundancy
T7	Pay-rate algorithm doesn't calculate competitive rates	● Determine a base pay that will increase/decrease due to various factors ● Compare pay-rates of similarly rated tutors who tutor the same courses
T8	Web-conferencing session is not set up properly	● Use Google Hangouts ● Ask permission from user to schedule events in their calendar via Google API's
T9	Unexpected interruption prohibits online sessions from occurring	● Refund payments in this case as long as both parties arrived to the meeting. ● Use a Google Hangouts, a commonly used web conference tool maintained by a large corporation.
L1	Violating The Family Education Rights and Privacy Act (FERPA)	● No portal access ● Transcripts are analyzed then thrown out ● Users agree to grade disclosure in terms of use agreement
L2	Users use application for illegal activities	● Terms of use agreement ● Reporting features

Figure 20. Risk Matrix Descriptions

This figure describes the risks for the Tutor Dash RWP introduced in Figure 19. Some of these risks such as C3 and C4 will not be mitigated in the prototype since they are mainly concerned with quantity of users.

4.5. Prototype Objectives and Goals

The objectives for the Tutor Dash prototype closely resemble the objectives for the RWP. The prototype will be a centralized platform where tutoring services can be offered and received. This platform will be designed for ODU students with devices running Android Nougat v.7.0 and higher. This prototype will implement many of the features that the RWP implements such as real-time tutor qualification, notifications, automatic pay-rates, and in-app payment transactions.

The main goal of the prototype will be to create a scalable mobile application that could potentially be a viable product in the future. In order for this goal to be achieved, Tutor Dash will need to gain an initial following of repeat users to establish the environment internal to the app. Tutor Dash will be developed initially for a small subset of its target demographic which will be the test environment for the app, but if this prototype application were to become popular among the ODU student body, Tutor Dash would likely be expanded upon in the future in order to complete the additional features that would be present in the RWP.

4.6. Prototype Development Challenges

There will be several challenges concerning the development of the Tutor Dash prototype. The biggest challenge that many of the other challenges stem from will be due to the team's inexperience working with android API's in an unfamiliar programming environment. For example, many of the third-party APIs that Tutor Dash will be implementing such as Google Calendar, Firebase, and Braintree are all new tools that very few team members have ever had exposure to.

Implementing the major automation algorithms that handle setting up web-conferences and pay-rates are also anticipated to be challenging efforts. Since these algorithms rely heavily on unfamiliar third-party APIs, it is expected that the team will spend more time than average developing and optimizing these algorithms. Since the team will be on a tight time-constrained schedule, fully-implementing these algorithms is projected to be difficult to accomplish.

Another challenge that the team will face is the implementation of the "handshake agreement". The reason this is anticipated to be challenging is due to the fact that this algorithm must absolutely be secure. Since this algorithm involves real-world currency, if a vulnerability were to become exploitable post-release, this would reflect poorly on the development team and lead to larger problems.

Many of the other challenges that the team expects to face are in regards to testing. Since Tutor Dash is a user-run application with very little input from administrators, it is important that when testing, there are enough users online to demonstrate the functionality of all of the features included in the app. To do this, several mocked up accounts will need to be created. In order to test the functionality of the app, these accounts will be leveraged during testing and manually controlled, but this is anticipated to consume many hours. Since Tutor Dash is not a trivial application, the presence of any bug could consume an unexpected amount of time, and this could negatively impact the overall quality of the product since it could lead to features being cut from the prototype.

[This space intentionally left blank]

5. Glossary

Course-specific tutoring: Academic assistance services provided for a particular course at a particular university.

DFWI: An acronym that stands for Drop/Fail/Withdraw/Incomplete. This relates to university course incompletion status.

DFWI rates: Represents the ratio of university students who do not complete their courses to students who do complete their courses.

Direct Competitor: Another product or company which is solely involved in the same domain space as Tutor Dash.

Entity: A person, object, or external server that serves as a leveraged functional component of the Tutor Dash product.

FERPA: The Family Educational Rights and Privacy Act is a United States federal law that protects the privacy of educational records.

Indirect Competitor: Another product or company which is involved in the same domain space as Tutor Dash, but doesn't not focus solely in that space.

Tester: Individuals responsible for testing the quality of the software.

Tutee: A university student seeking academic assistance.

Tutor: A university student offering independent tutoring services that are qualified based off previously taken courses.

Uber: A ride-hailing company that offers the Uber mobile app, which you can use to submit a trip request that is automatically sent to an Uber driver near to you, alerting the driver to your location.

6. References

"Academic Tutoring in Comprehensive Universities." Hanover Research, 2014. URL: https://www.hanoverresearch.com/wp-content/uploads/2017/08/Academic-Tutoring-in-Comprehensive-Universities.pdf

"Academics." Old Dominion University, 8 Jan. 2019. URL: www.odu.edu/academics

"Campus Tutoring." Old Dominion University, 19 Jan. 2019. URL: www.odu.edu/success/academic/tutoring#tab125=0

"Course-Specific Tutoring." Old Dominion University, 19 Jan. 2019. URL: www.odu.edu/success/academic/tutoring/course-specific

"Courses of Instruction." Old Dominion University, Feb. 2019. URL: catalog.odu.edu/courses/

Ciscell, Galen, et al. "Barriers to Accessing Tutoring Services Among Students Who Received a MidSemester Warning." ERIC, Pacific Lutheran University - Department of Sociology, 2016. URL: files.eric.ed.gov/fulltext/EJ1114513.pdf

Evans MDR, Kelley P and Kelley J (2017). Identifying the Best Times for Cognitive Functioning Using New Methods: Matching University Times to Undergraduate Chronotypes. Front. Hum. Neurosci. 11:188. doi: 10.3389/fnhum.2017.00188. URL: https://www.frontiersin.org/articles/10.3389/fnhum.2017.00188/full?&utm_source=Email_to_authors_&utm_medium=Email&utm_content=T1_11.5e1_author&utm_campaign=Email_publication&field=&journalName=Frontiers_in_Human_Neuroscience&id=239492

"Facebook - Groups." Facebook Help Center, Facebook, 2019. URL: www.facebook.com/help/1629740080681586?helpref=hc_global_nav

"Find a Local In-Home Tutor Today." HeyTutor, HeyTutor LLC. URL: heytutor.com/

Fry, Natalie. "New Research Reveals That College Students Study Best Later in the Day." NevadaToday, University of Nevada, Reno, 11 Apr. 2017. URL: www.unr.edu/nevada-today/news/2017/best-time-of-day-to-study

"Peer Assisted Learning" BMC Education, 8 March 2006 URL: https://bmcmededuc.biomedcentral.com/articles/10.1186/1472-6920-6-18

Pierce, Dennis. "Supporting Students Beyond Financial Aid", 2016 URL: http://eds.b.ebscohost.com.proxy.lib.odu.edu/ehost/detail/detail?vid=0&sid=d93df6c4-3729-4b62-8d58-95e25c309878%40sessionmgr102&bdata=JnNpdGU9ZWhvc3QtbGl2ZSZzY29wZT1zaXRl#AN=114789419&db=ehh

Qayyum, Adnan. "Student Help-Seeking Attitudes and Behaviors in a Digital Era." International Journal of Educational Technology in Higher Education, vol. 15, no. 1, 2018, doi: 10.1186/s41239-018-0100-7. URL: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-018-0100-7

"Skooli Tutors Online." Skooli Online Tutoring, Skooli, Feb. 2019. URL: www.skooli.com/prices/students

"Student as Peer Tutors" BMC Education, 9 June. 2014. URL: https://bmcmededuc.biomedcentral.com/articles/10.1186/1472-6920-14-115https://bmcmededuc.biomedcentral.com/articles/10.1186/1472-6920-14-115

Topping, J. Keith. "Trends in Peer Learning", 19 Jan 2007 URL: https://www.tandfonline.com/doi/full/10.1080/01443410500345172?scroll=top&needAccess=tru

"Tutor Matching Service - How It Works." Tutor Matching Service, Tutor Matching Service, 2019. URL: tutormatchingservice.com/#/about

"Tutor.com - The Princeton Review." Tutor.com, The Princeton Review, URL: www.tutor.com

"Tutors - Care.com." Care.com, Care.com, Feb. 2019. URL: www.care.com/tutors

"Wyzant." Wyzant Resources, Wyzant Inc., Feb. 2019. URL: www.wyzant.com/howitworks/students