Running Head: Lab II - Tutor Dash Product Specification

# Lab II - Tutor Dash Product Specification

Alexander Wojtowicz

CS411W - Professional Workforce Development II

Old Dominion University - Team Gold

Professor T. Kennedy

17 October 2019

Version 1

**Table of Contents**

**List of Figures**



**List of Tables**

## 1. Introduction

### 1.1 Purpose

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.

The target customer/user base for Tutor Dash will be limited to university students. Tutor Dash will support only a finite number of universities. Any student with a supported university-affiliated email address will be able to use Tutor Dash. For the prototype, only Old Dominion University will be supported, therefore only students attending Old Dominion University will be able to use Tutor Dash.

At a high level, Tutor Dash's process flow will closely resemble that of similar online tutor-hosting platforms such as Tutor.com or Wyzant. A user will first create an account with Tutor Dash. Once they activate their account by confirming their email address, this user is ready to use the application. Users can be classified as tutees, tutors, or both, where tutees are users looking for tutors, and tutors are users looking for tutees. Inside of the main activity of Tutor

Dash, users will be able to search for one another, and if they so desire, they will be able to schedule tutoring sessions with one another. Upon the conclusion of a session, both users will supply a rating for their correspondent based on performance, and the tutee will automatically pay the tutor based on the length of the session and the tutor's predetermined pay-rate. Figure 1 illustrates this process flow.

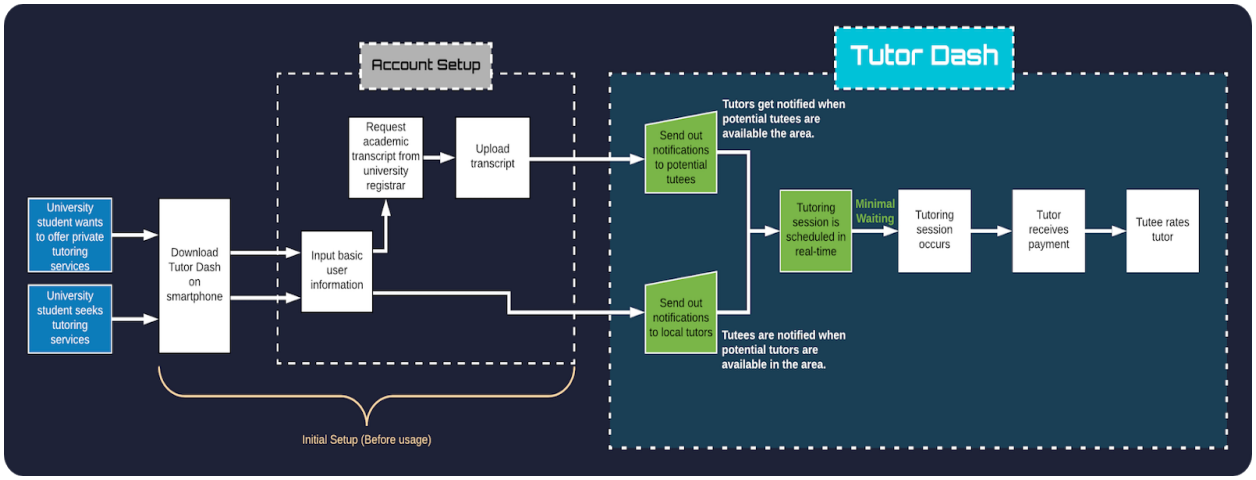


**Figure 1.** Tutor Dash Process Flow

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).

Even though the process flow of Tutor Dash is similar to other options already present on the market, the implementation of the various features that Tutor Dash will offer is what will make it a unique product. Tutor Dash will provide several algorithms that will automate several of the current processes that exist in private tutoring market. Such processes include determining pay-rate, authorizing payment transactions, tutor qualification/verification, finding potentially compatible users, and synchronization of calendar events. Tutor Dash will also utilize real-time capabilities so that users will be able to view and chat with other users in real-time. In addition to

these features, Tutor Dash will also have a built-in system to handle reporting of falsified reviews and blacklisting of malicious users.

Tutor Dash will not be a platform that anyone can use. Only students attending supported universities will be allowed to sign up for user accounts. The objective of Tutor Dash is to make it easier for students to find tutors and/or tutees, so what occurs during the tutoring sessions will not be monitored by Tutor Dash as it is beyond the scope of the application. In addition, due to the time constraints placed on Tutor Dash's development, the graphical component of the app will only have one theme with portrait-style layouts. Table 1 gives a more in-depth look at what kinds of features will be implemented in the Tutor Dash Prototype vs the RWP.

*[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.

**1.2 Scope**

Tutor Dash's primary objective is to provide university students with a resource better-suited for their tutoring needs. Tutor Dash will exist to serve solely as a tool that students will be able to utilize in order to help them find tutors and/or tutees at their own university. This type of application is necessary as it will solve several problems that university students currently face when trying to find tutors and/or tutees. There is currently a lack of centralization in the current private tutoring market, and there are many options that serve as general tutoring platforms. Tutor Dash will exist to serve a specific niche in the market in an attempt to get students more involved. There is also a lack sufficient tutoring services provided by universities. Universities tend to only offer tutoring programs for lower level courses, leaving the majority of their offered courses without a tutoring program. Tutor Dash will solve this problem by providing an option for tutorings to offer course specific tutoring in any course at their respective university.

Tutor Dash will provide benefits to both university students who are tutors and tutees. Not only will Tutor Dash make it easier for students to participate in the tutoring process, but it will also help them receive better grades as they will be more engaged in their course material. This effect would most likely cause DFWI rates to decrease, reflecting positively on the university. The goal of Tutor Dash is to satisfy these benefits by creating long-term customers who regularly engage in the app.

The Tutor Dash prototype will be very similar to the RWP. Refer to Table 1 for a description of the subtle differences. In order to demonstrate Tutor Dash as a proof-of-concept application, all of the core components will be present in the prototype. Users will be able to

create and manage their accounts, specify whether they want to be a tutor, and query for other users in real-time. All of the major algorithms will be fully-functional, and every screen of the user interface will be completed. All of the APIs that would be present in the RWP will also be present in the prototype. The ultimate goal of the prototype is to create a small scale model of the RWP. Ideally, the prototype will be able to scale up to support multiple universities with little to no issues. Figure 2 illustrates the major functional components of both the Tutor Dash prototype and the RWP.

*[This space intentionally left blank]*

### 1.3 Definitions, Acronyms, and Abbreviations

*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.

*Serverless Architecture*: Concerning database interactions over a network server, this type of architecture implies that the server's implementation is invisible (or abstracted) to the team developing the product using the actual server.

*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.

### 1.4 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

“Lab 1 - Tutor Dash Product Description.” Team Gold. 26 Sept. 2019. URL: https://www.cs.odu.edu/~411gold

"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

### 1.5 Overview

This product specification provides the hardware and software configurations, external APIs, capabilities, and features of Tutor Dash. The information provided in the remaining portion of this document includes a detailed description of the hardware and software architectures that Tutor Dash will be built upon, the required functional features and capabilities that will be present in the prototype, and an in-depth look at how the interactions within the mobile application will be handled.

*[This space intentionally left blank]*

## 2. General Description

### 2.1 Prototype Architecture Description

Tutor Dash will be a mobile application for Android running Nougat v.7.0 and higher. The layout of the application will closely resemble that of Uber. Users will be able to sign in and they will first see a screen with a map. This map will have markers for the current user's location, as well as markers for every other user at the current user's school who is currently available. Users will be able to query for other users by typing in their specified query and submitting it. Results will be presented in a scrollable list. Users will be able to chat with other users, and they will also be able to schedule sessions with other users. Essentially, every main part of the application will have its own screen to present the information to the user. Users will also be able to manage their accounts by using the options menu, and this options menu will serve as the primary way of navigating around the app.

The architecture of Tutor Dash will be mostly software-oriented since the hardware component of Tutor Dash only requires an operating system for the application to run. Figure 2 illustrates the major functional components of Tutor Dash. Each of these components can be 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.

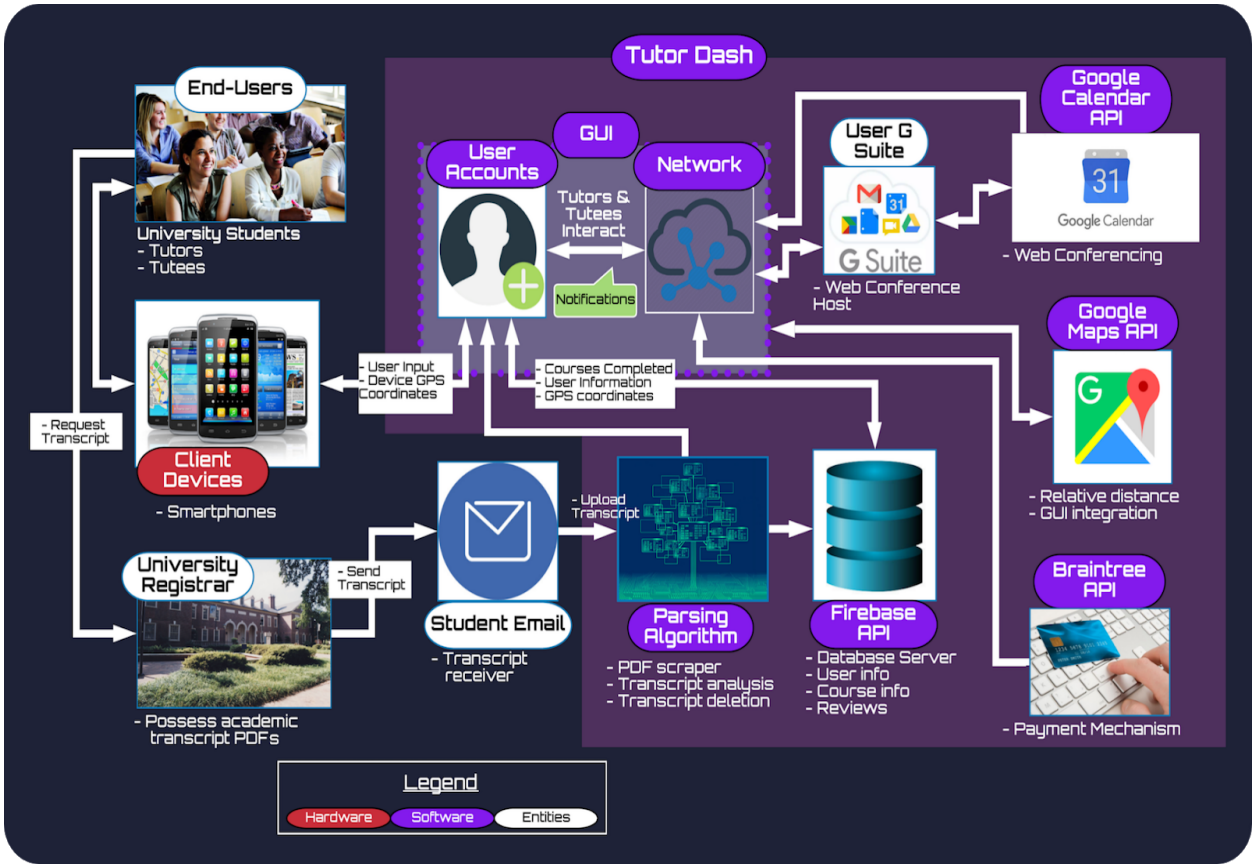


**Figure 2.** Major Functional Component Diagram

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.

Hardware Components:

- Client Devices: Devices will be smartphones running Android Nougat v.7.0. Once the Tutor Dash APK (Android application package) is installed, users are ready to launch Tutor Dash.

*[This space intentionally left blank]*

Entity Components:

- University Registrar: This entity serves as the source of the official academic transcripts for university students. Students will request an academic transcript and receive it from their university registrar.
- Student Email: Transcripts or secure links to download transcripts will be sent to the student's email. Students will download this transcript to their Android device so they can upload it into Tutor Dash.
- User G Suite: Tutor Dash will communicate with user G Suites (particularly Google Calendar) so that sessions scheduled in Tutor Dash can be synchronized to a user's collection of calendar events. If a session is determined to be via web-conferencing, then this Google Calendar event will contain a Google Hangouts link.

Software Components:

- GUI: This component is the wrapper that bundles the data present in the network and current user account. The GUI is important as it is the entire front-end of Tutor Dash, and it is what users will interact with directly.
- User Accounts: Account information will be stored locally and sync up with the database upon login. User accounts hold both public and private user-specific information. Other users will be able to query public information on other user accounts.
- Network: Data will be transmitted through the network (via either network data or wifi). This component is abstracted for Tutor Dash, as it will be adopting a "serverless" architecture with the implementation of Firebase.

- Parsing Algorithm: User will use this algorithm to become qualified to tutor courses. Users will upload their transcripts into this algorithm, and it will decide which courses a user is eligible to tutor. Based on that information, a user will be able to select 1 to all of the available courses presented to them.
- Firebase API: Google's Firebase is a collection of tools that provide a "serverless" architecture for Tutor Dash. Specifically, Firebase Auth will be utilized to handle membership integrity and account management, Firebase Storage will be used to store images for user avatars, and Firebase Firestore will be used to store all user information.
- Google Calendar API: This API will be used as the link between Tutor Dash events and Google Calendar events so that both types of events can be synchronized to the user's Google Calendar.
- Google Maps API: This API will be leveraged to render the interactive map in the main section of Tutor Dash. Location services will also be utilized so that multiple users can be shown on the map in real-time.
- Braintee API: This API will be used to handle payments within the application. Payment methods will be stored in Braintree's "Vault", so Tutor Dash will not keep track of this sensitive information internally.

*[This space intentionally left blank]*

**2.2 Prototype Functional Description**

The entirety of Tutor Dash can be broken down into three functional 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.

The database that Tutor Dash will be using is 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.

The UI/UX will be composed of four distinct phases and two distinct views as illustrated in Figure 3. 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. The UI/UX will serve as the entire front-end of Tutor Dash, and it is the component that users will interact with directly.

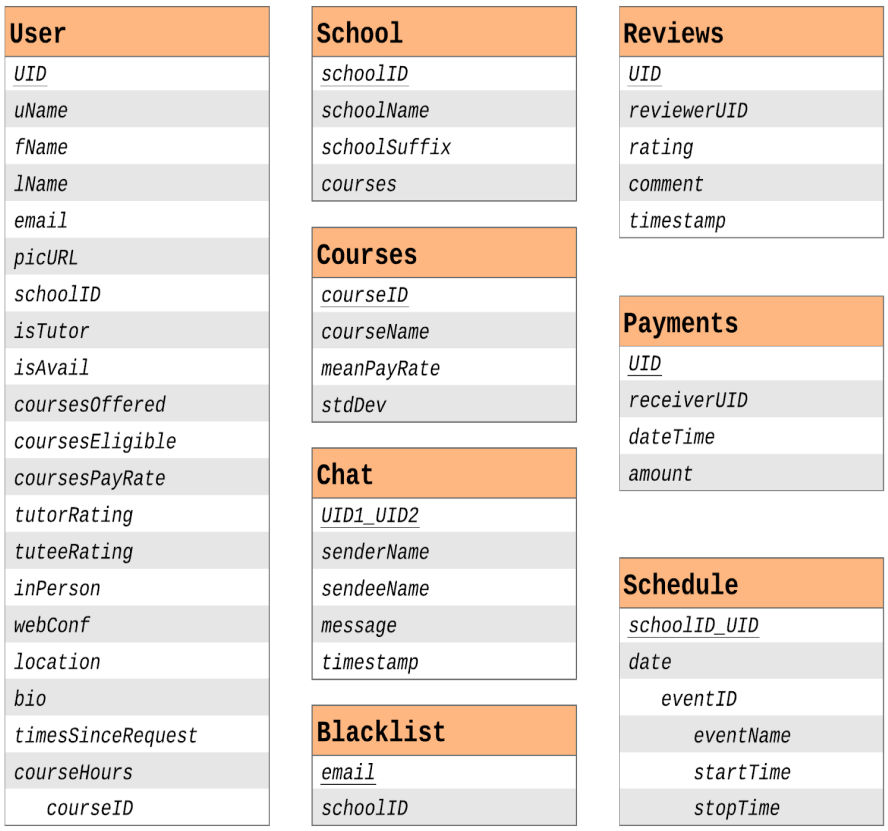


**Table 2.**
Database Schema

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 algorithms will serve as the backend of Tutor Dash. These are also what will make Tutor Dash a unique product with custom features. There are five distinct algorithms that will be implemented within Tutor Dash:

1. PDF Transcript Parser: This algorithm will have two purposes. The first of which will be to parse an official university transcript and determine which courses a user is eligible to tutor based on their grade. The second purpose of this algorithm is to add the selected courses to the database. If a tutor offers a course that no one else at their university offers, then this course has to be added to the list of available courses being offered by tutors at their respective university.

2. Relative Distance Calculator: This algorithm will be used to determine the number of miles away two users are from each other. Based on two users geopoint locations, this algorithm will perform mathematical operations in order to determine how far users are away from the current user. This result will be used in the google map and in the search results view.
3. Automatic Pay-Rate Calculator: Since pay-rates will not be configurable by users, an algorithm that calculates the fairest possible pay-rate will be necessary. This algorithm will take into account several factors such as 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.
4. Session Event Creator/Synchronizer: The purpose of this algorithm will be to create in-app events, and synchronize them with a user's Google Calendar. All events in-app will be placed onto a user's Google Calendar, and this is especially important if the session was scheduled as needing web-conferencing. If this is the case, then the Google Calendar event for this session will contain a Google Hangouts link.
5. Automatic Payment Handler: This algorithm will handle the situations that may occur between the time a session is scheduled and the time it ends. Even though Braintree will be storing user payment methods and handling transactions, payments will be automatic within Tutor Dash, thus an algorithm that responds to user-initiated events is necessary.

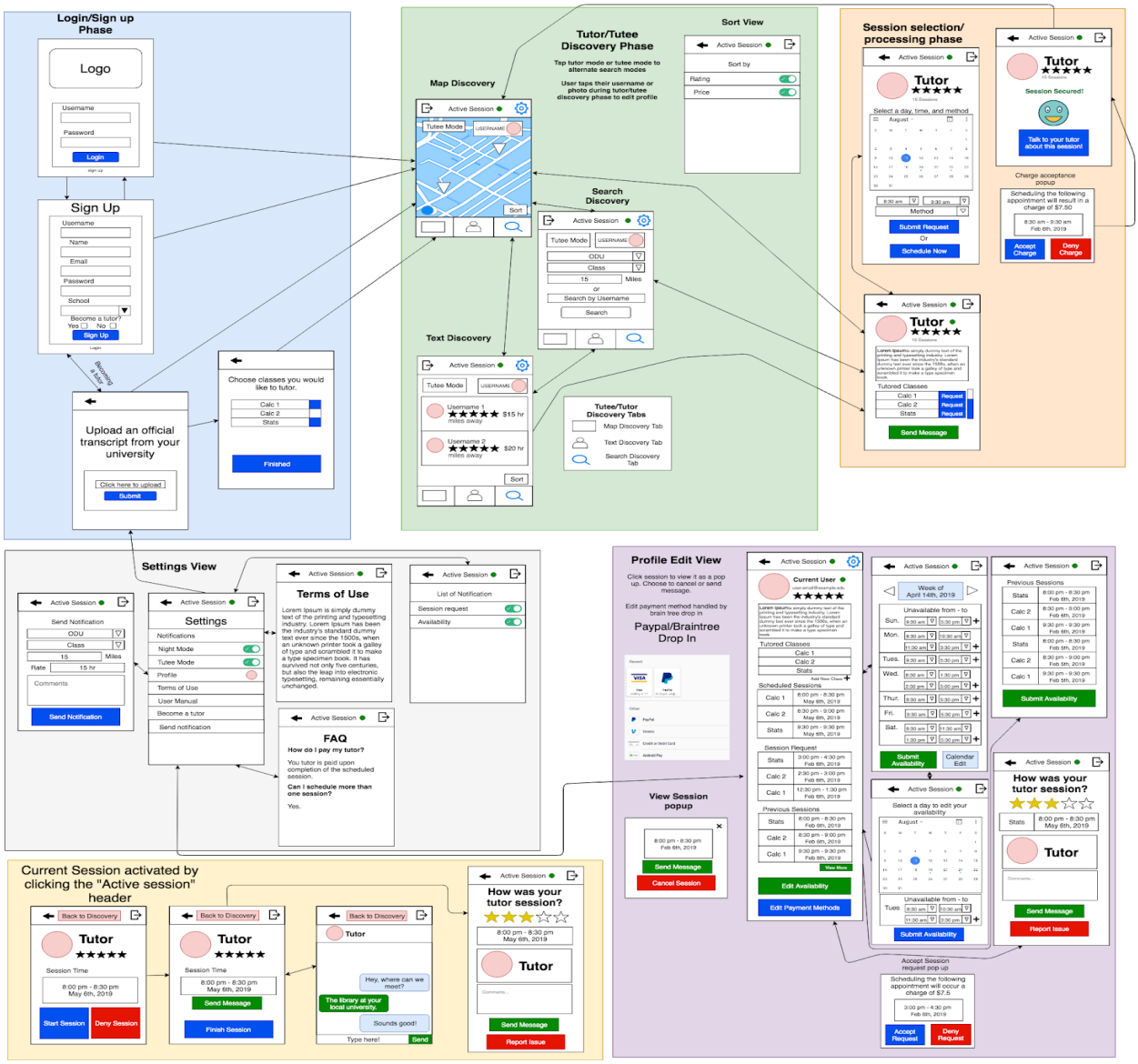


**Figure 3.** UI/UX Full Implementation

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.

*[This space intentionally left blank]*

## 2.3 External Interfaces

Tutor Dash's main external interface will be Firebase. Firebase will allow Tutor Dash to run on a "serverless" architecture since the servers will not be maintained by the team developing Tutor Dash. The only hardware interface concerning Tutor Dash is that of the physical client device. As for software interfaces, Tutor Dash will be leveraging several software APIs to implement the specified features.

### 2.3.1 Hardware Interfaces

The only hardware interface concerning Tutor Dash is that of the client device. For Tutor Dash to run properly, Devices must be running Android Nougat v.7.0 or higher and have GPS location services enabled.

### 2.3.2 Software Interfaces

Several software interfaces will be leveraged in the development of Tutor Dash. Firebase will be used in almost every aspect of the application since data will need to be updated in real-time. For account management, Firebase Auth will be leveraged. This API will provide several boiler-plate helper methods that will make it easier to enforce account integrity. Firebase Storage and Firebase Firestore will be leveraged to store user data.

In addition to Firebase, other APIs will be utilized. PDFBox, will used to parse PDFs. This will make implementing the PDF transcript parsing algorithm easier. Google Maps API will be used to render the google map in the main section of the application where users can see other available users' locations. This API will also server Tutor Dash by providing location services to make distance calculations and the rendering of users on the map easier. Google Calendar API

will be leveraged to synchronize in-app events to users' Google Calendars. This will overall benefit the user experience.

Another service that Tutor Dash may or may not be using is that of SendGrid, Mailgun or JavaMail. These services provide APIs to send emails in a "serverless" architecture. Tutor Dash may use a service such as SendGrid or Mailgun in order to handle password resetting. If none of these services are utilized, then Firebase Auth will be used instead to send reset password emails.

### 2.3.3 User Interfaces

Tutor Dash will be composed of several different layouts customized to the types of information that need to be displayed on the front-end. Users will interact with these screens through their touch-screen Android Devices by clicking on elements. Input will be based on what users enter in fields or what users click on depending on the application of data input.

### 2.3.4 Communications, Protocols, and Interfaces

Tutor Dash's algorithms will all be hosted on the client. Interactions involving database operations will be handled wirelessly via HTTP. The Firebase API handles HTTP callbacks implicitly, so this is the type of communication that Tutor Dash will implement.

*[This space intentionally left blank]*

**3. Specific Requirements (Collaborative)**

*[This space intentionally left blank]*

## Appendix

### A. Algorithm Flow Diagrams:

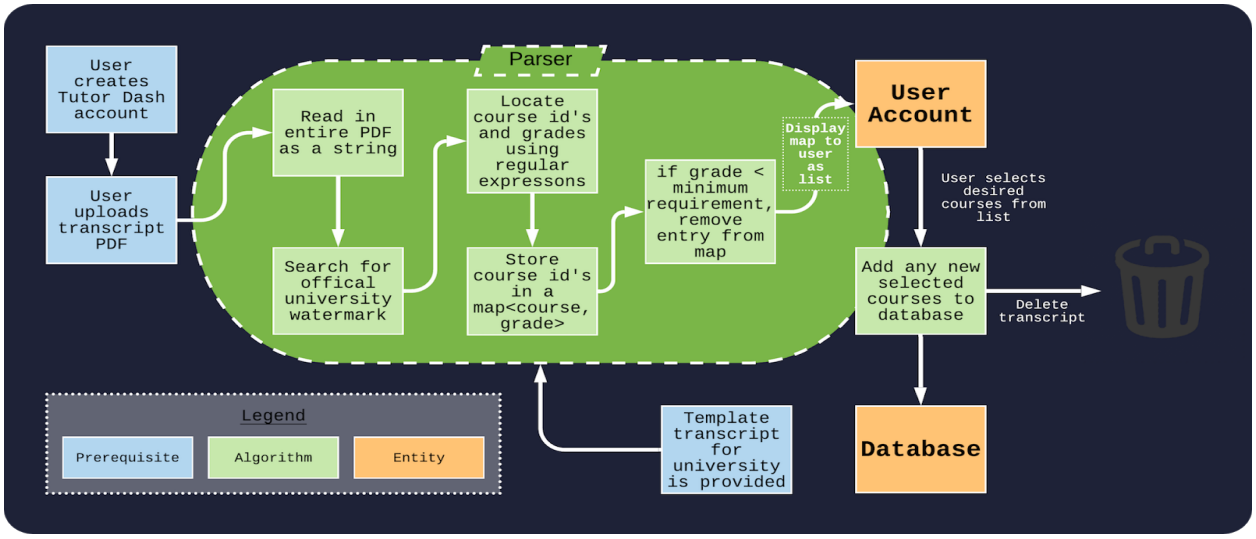


**Figure 4.** PDF Transcript Parser Algorithm Flow

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.

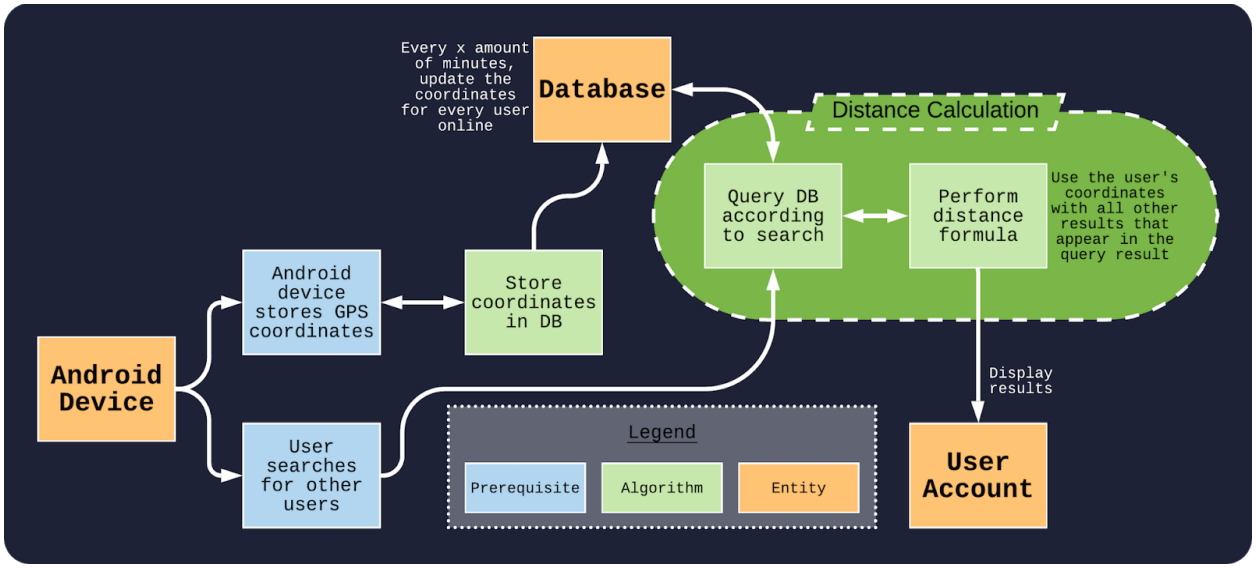


**Figure 5.** Relative Distance Calculator Algorithm Flow

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.

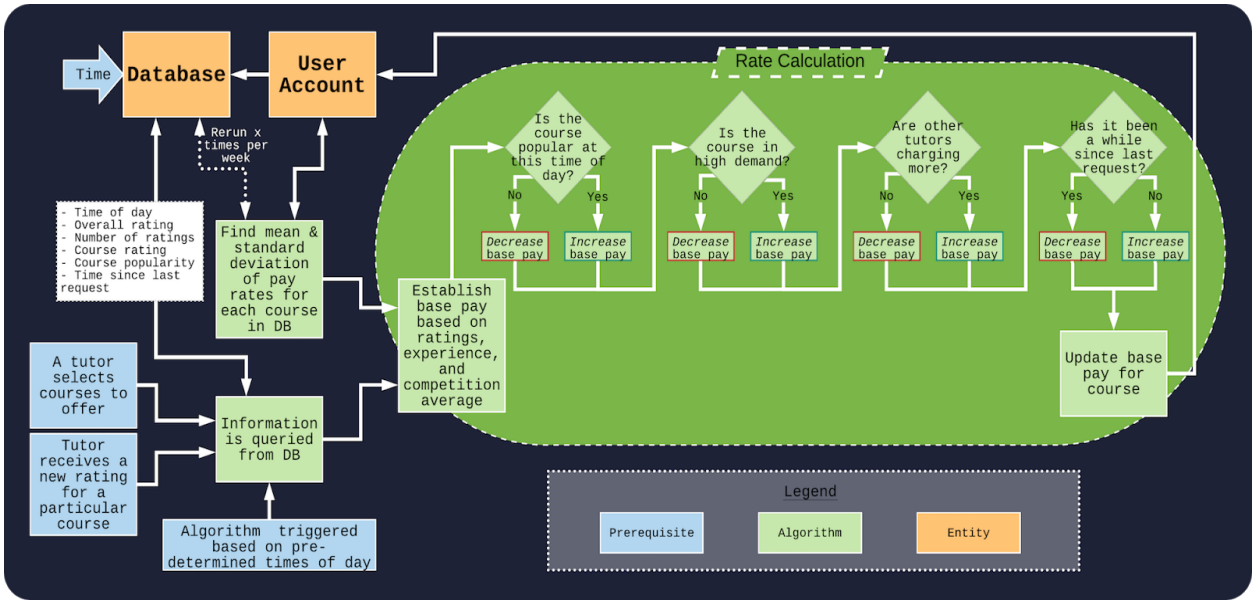


**Figure 6.** Competitive Pay-Rate Algorithm Flow

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.

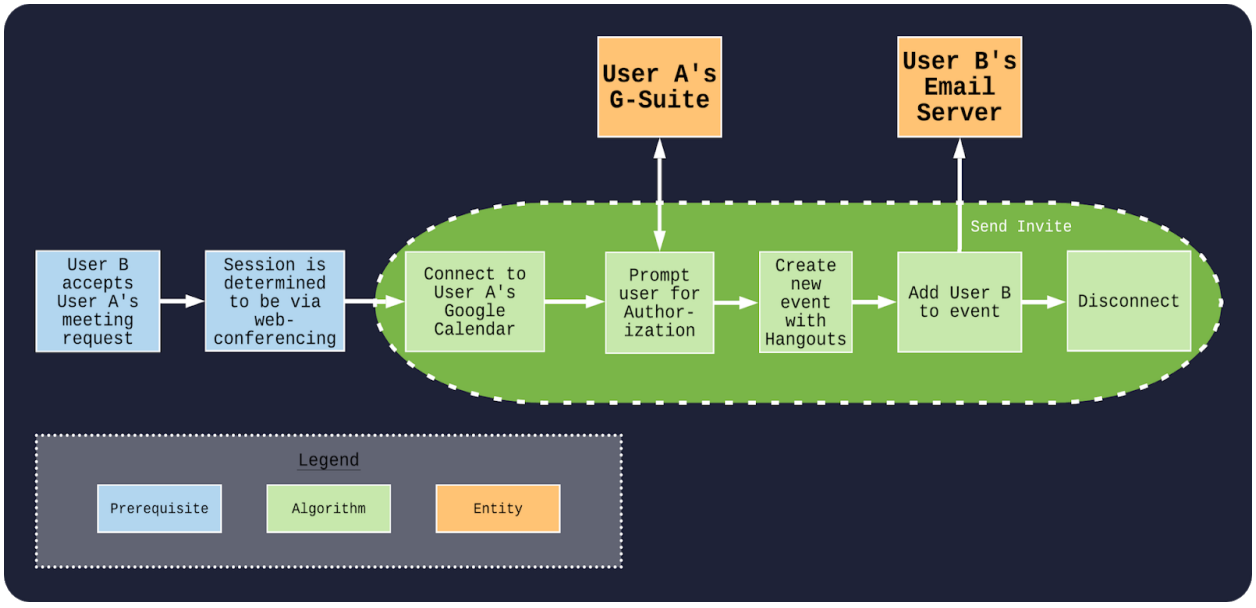


**Figure 7.** Session Event Creator/Synchronizer Algorithm Flow

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.

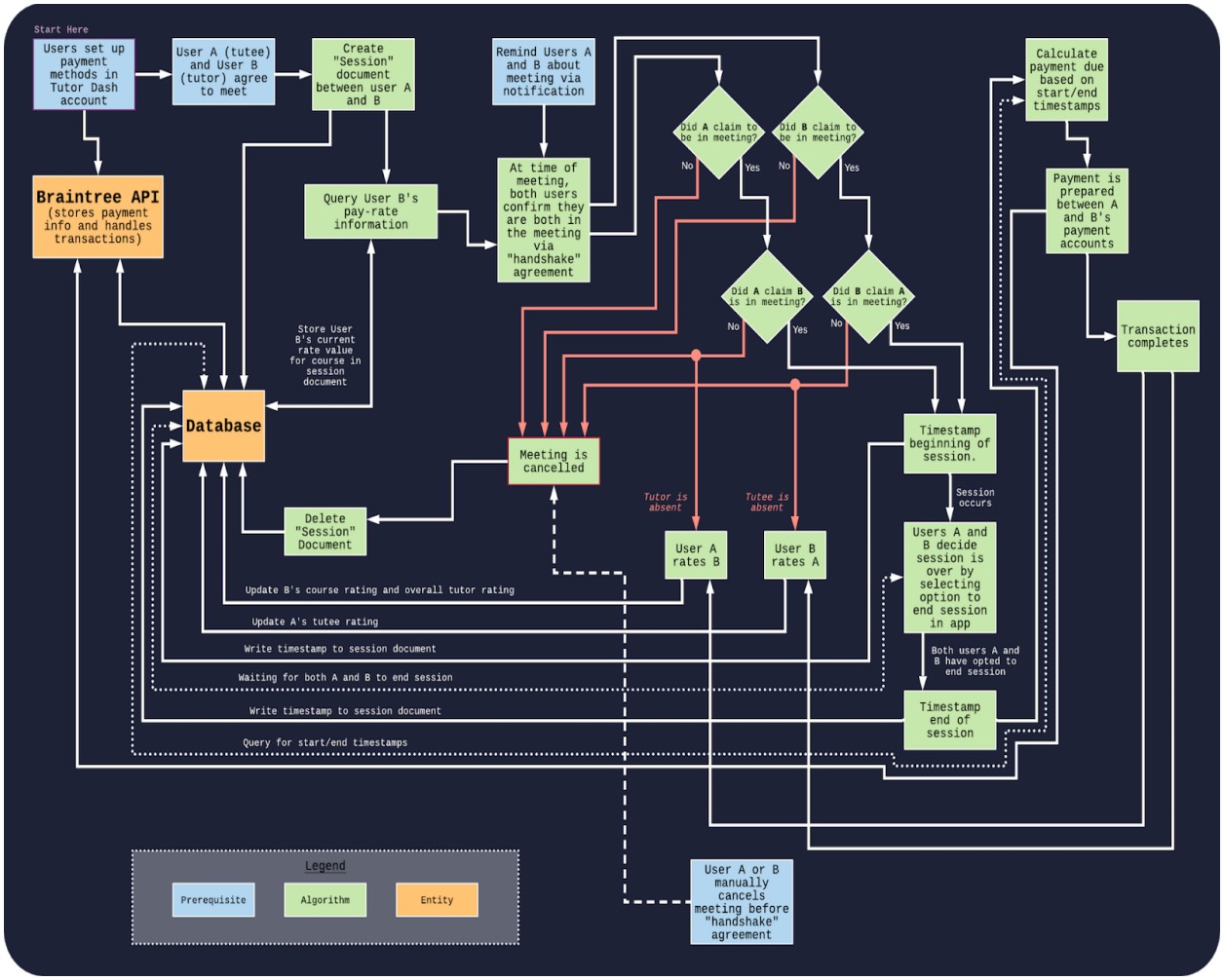


**Figure 8.** Payment Handler Algorithm Flow

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.

## B. UI/UX Wireframes


**Figure 9.** 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.

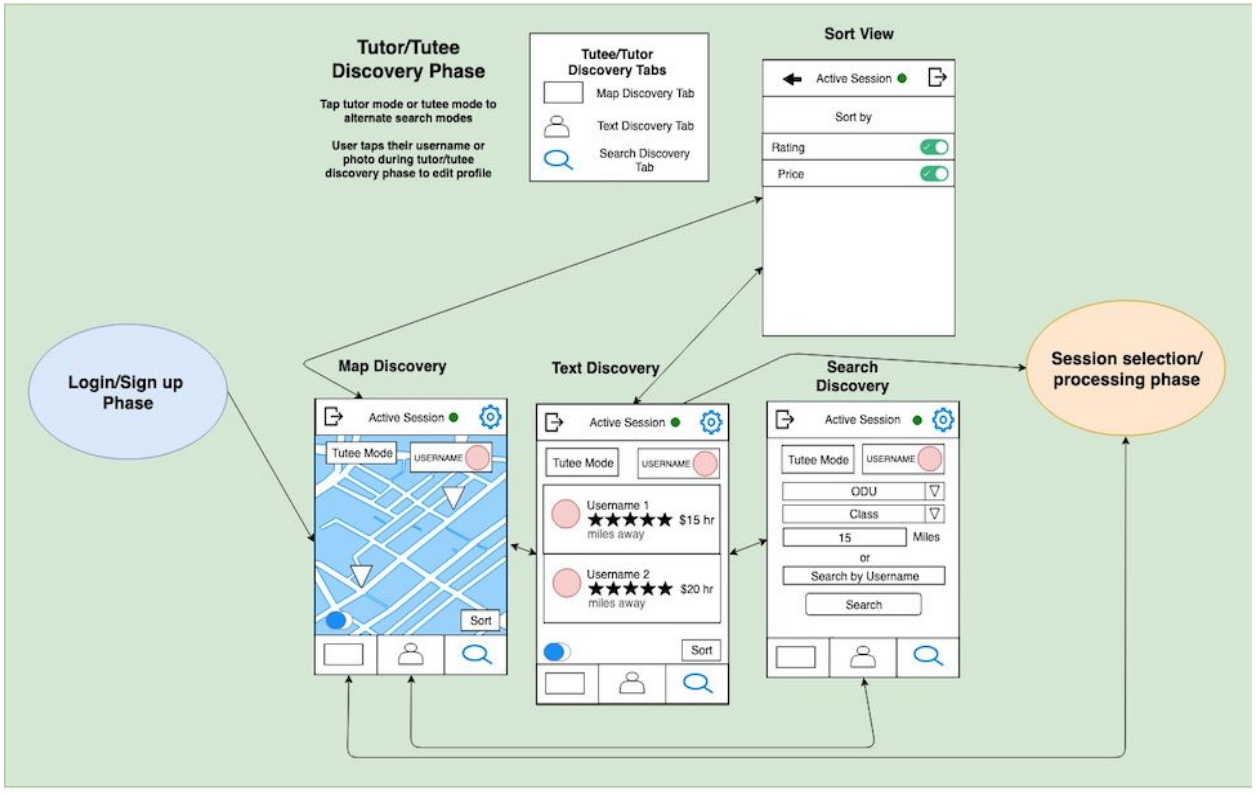


**Figure 10.** UI/UX Phase 2

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.

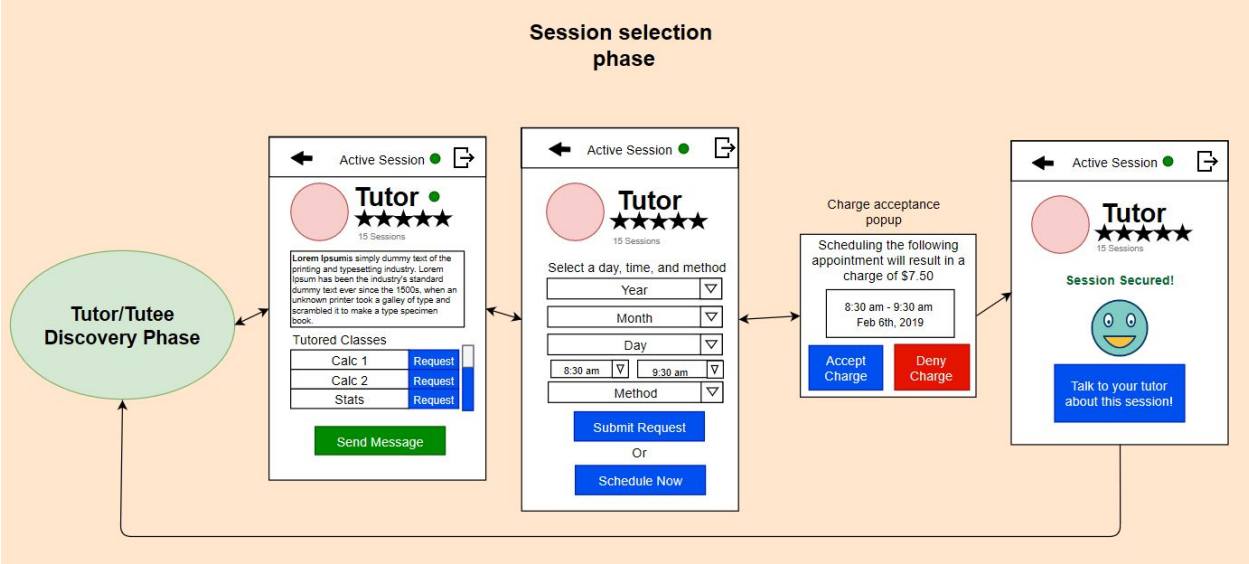


**Figure 11.** UI/UX Phase 3

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.

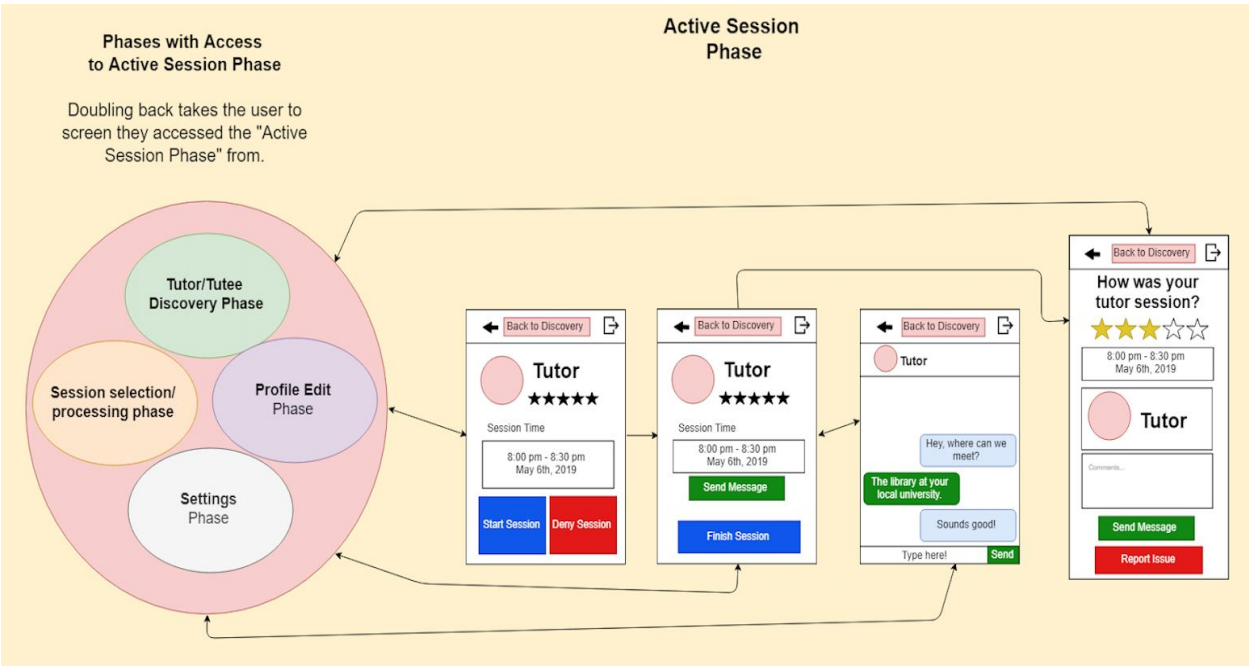


**Figure 12.** UI/UX Phase 4

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.

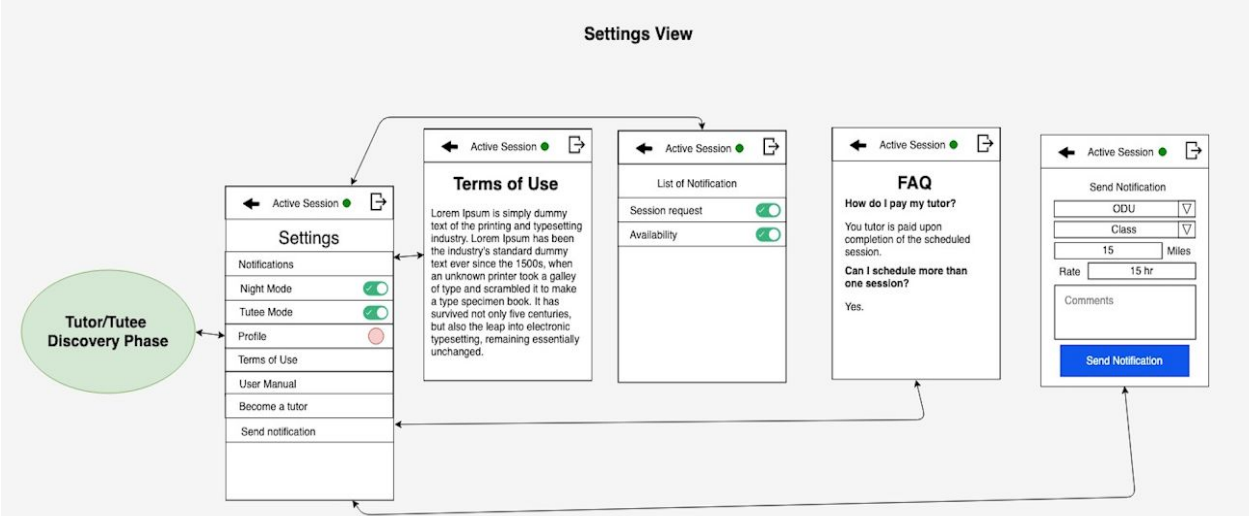


**Figure 13.** UI/UX View 1 - Settings

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.

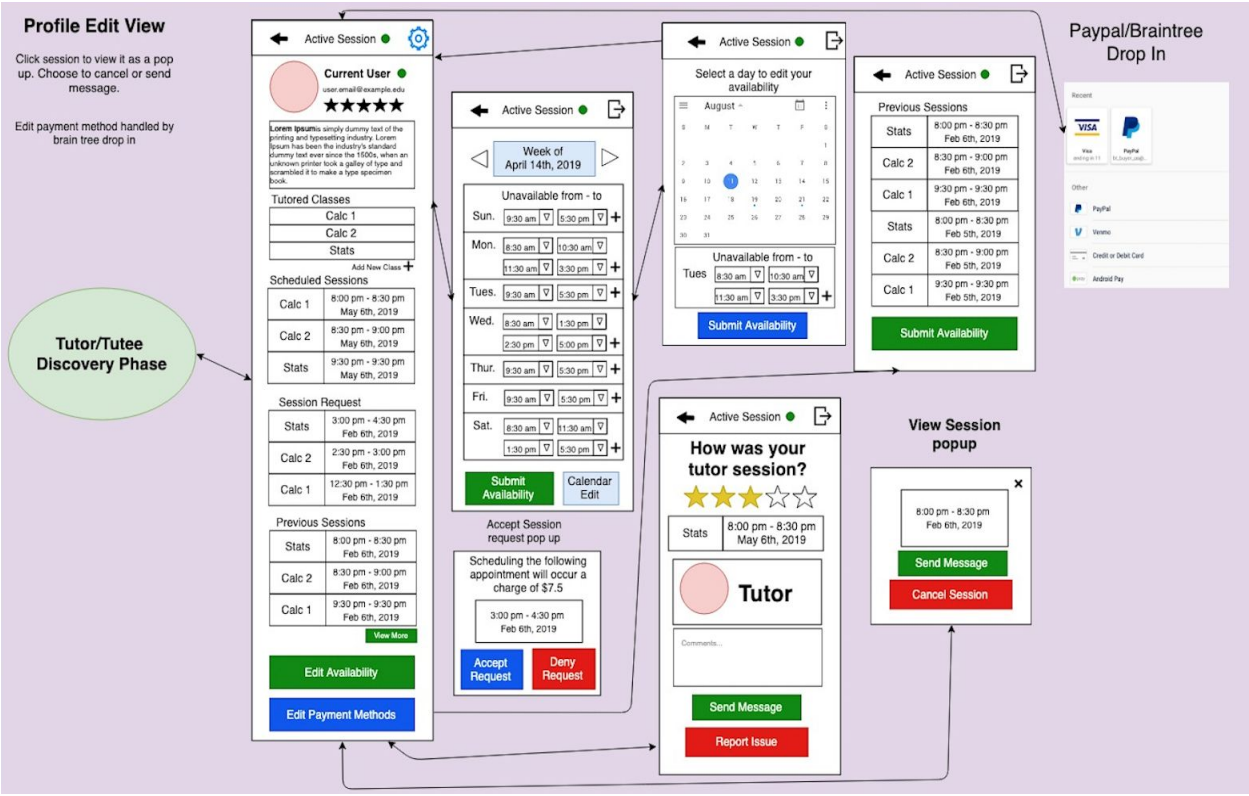


**Figure 14.** UI/UX View 2 - User Profile

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).