3. Specific Requirements

3.1 Functional Requirements

3.1.1 Account Creation (O: Wojtowicz)

Users must have the ability to create accounts within the application. Required information that users will need to fill out will be presented in the application as a form. Accounts will not be activated until users have confirmed their email.

3.1.1.1 Input Validation (O: Wojtowicz)

When a user attempts to create an account, they must be prompted to enter their first name, last name, username, password (twice for integrity purposes), and school. In addition, the user must have the ability to specify if they would like to become a tutor. The following requirements for fields must be non-null values, and they must meet the following conditions:

Passwords must:

1. Be 8 - 20 characters long
2. Contain at least 1 capital letter
3. Contain at least 1 lowercase letter
4. Contain at least 1 numeric digit
5. Cannot contain username or email

Usernames must:

1. Be 6 - 16 characters long
2. Contain only letters or numbers
3. Be unique to every user stored in the database

Emails must be:

1. Follow the regular expression : ^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$
2. Be unique to every user stored in the database
3. End with the appropriate domain extension of the user's selected school (e.g. Old Dominion University emails end with .odu.edu)

3.1.1.2 Population of Options in "School" Field (O: Wojtowicz)

Not every university will be supported in this application. In order to enforce this, the "school" field on the account creation form must be a dropdown menu of options. This menu must be populated at runtime with all of the available schools in the database.

3.1.1.3 Email Confirmation (O: Wojtowicz)

After a user has signed up for an account, an email will be sent to the email address that they provided during the signup phase. This email will contain a link that activates the user's account once clicked. Users will not be able to log into the application until they have activated their account.

3.1.2 Account Management

3.1.2.1 Password Resetting (O: Wojtowicz)

Users must have the ability to securely reset their password. The user must be able to enter the application, click some indicator that navigates them to a password reset activity, and type in their email or username associated with their account. If the email or username is found, then the user will receive an email with a link that allows them to reset their password.

3.1.2.2 Password Changes (O: Wojtowicz)

Users must be able to change their password for their account. Users who are logged in will be able to navigate to somewhere within their account settings and select an option to reset their password. They will be prompted to enter their current password and their new password (twice to enforce integrity).

3.1.2.3 Settings (O: Owings, M: Wojtowicz)

Users must be able to navigate to a settings menu accessed from the Discovery Pages that allows them to take the following actions:

1. Manage account settings
    a. Modify notification alerts to receive greater or fewer alerts than the default amount.
    b. Become a tutor, if they have not already done so.
    c. View the “Terms of Use” agreement.
2. Switch between tutor and tutee mode
3. View their member profile.
4. View the user manual that explains how to use Tutor Dash.

3.1.3 Member Profile View (O: Wojtowicz, M1: Hessefort)

Users must be able to view their member profile in Tutor Dash when they are currently logged in. This view must be accessible from the settings menu, and it will display public information. This information includes:

1. Username
2. First name

3. Last name
4. Bio
5. Calendar (for in-app tutoring sessions)
6. Courses seeking tutoring assistance
7. Courses offering as a tutor
8. Overall tutor and tutee ratings
9. Course-specific ratings for each offered course

The layout for this view should be reusable so that when the current user views other users' profiles, the same layout will render with different user-specific data. The current user must also have the ability to edit their profile. When the current user submits changes to their profile, this data must be updated in real-time to the database.

3.1.3 PDF Transcript Parser (O: Holterhaus)

Users will need to provide their own official university transcript for the parser to organize. The parser will verify that the transcript is indeed valid, and only after doing so will the parser continue searching for valid courses. Courses on the official transcript with a minimum grade of 'B' or higher will be parsed out and added to a list inside the application. Once the parser is complete, the parser will need user input to select which courses pulled out of the transcript will be added to their account for possible tutoring.

3.1.4 Relative Distance Calculator (O: Holterhaus)

The Relative Distance Calculator will need to have the user's exact position. This information will need to be supplied by the user's phone, and updated on the server periodically.

The Google Maps API will then take said information, and plot the distance between the tutor and tutee to give both users a rough estimate of distance.

3.1.5 Alert System (O: Campbell)

Tutor Dash will implement an alert system that must achieve the following:

1. Provide the capability for a tutee to send a notification to all tutors that tutor in a desired course.
2. Provide the capability for a tutor to send a notification to all tutees that are requesting help in a specific course.
3. Provide the capability to receive alerts while application is not running.

3.1.6 Automatic Pay Rate Calculation (O: Ordona, M1: Holterhaus)

Pay-rates for each course are calculated based off various parameters to provide tutors a competitive, fair environment to advertise their services. The Automatic Pay Rate Calculator will need the mean and standard deviation of pay-rates for a specific course, which will be stored inside the database. It will also need to calculate a base rate by using the mean and standard deviation of pay rates for a specific course and adjusting that value based on the user's ratings, experience, and competitive average. The base pay will then be adjusted based off course popularity for a time of day, a course's demand, the average price for other tutors, and whether or not said course has been requested recently. All four of these variables will adjust the base pay-rate on a set value. The Automatic Pay Rate Calculation will also need write permission to the database to update the mean and standard deviation after calculations have been produced. The Automatic Pay Rate Calculation will not drop below or above a certain pay-rate, both values will be adjusted based on customer input and product testing.

3.1.7 Automatic Payment Handling (O: Wojtowicz)

Since payments will be automatic within Tutor Dash, an algorithm that initializes and executes the transactions between a tutor and tutee's payment accounts will be necessary. When both users (a tutor and a tutee) agree on a session, this algorithm will be initialize a transaction between the tutor's incoming payment account and the tutee's outgoing payment account. The mathematical formula that will determine the expected amount that a tutee will owe is (*Tutor's pay-rate*) * (*Expected length of session*) = (*Expected total owed by tutee*). Once the session concludes, the transaction will be executed. However, there are a variety of situations that may occur between the time a session is scheduled, and the time the session concludes. This algorithm will be responsible for handling all of these situations. This algorithm will not handle the rendering of any layouts, but it will be responsible for providing output that other components of the application will use to display certain layouts. Below is a table that shows the situation on the left, and the required outcome on the right:

[This space intentionally left blank]

Situation	Outcome
Tutor accepts a tutee's session request	Both the tutor and tutee's payment accounts are validated to make sure they exist. If either accounts are invalid, DO NOT proceed, but rather, notify both users that the session will not be created. Calculate expected total that the tutee will owe the tutor. Validate that the tutee has at least (*Expected total owed by tutee*) in their payment account. If this is not the case, DO NOT proceed, but rather notify both users that the session will not be created. A SESSION document is created in Firestore. This document has a unique ID, and this ID is inserted in under the "sessions" collection for both users (tutor and tutee). This newly created SESSION document holds the following: ● The tutor's current pay-rate ● The tutor's username ● The tutee's username ● Start date of session ● Start time of session ● Expected length of session.
Tutor cancels before session is set to begin or tutee cancels before session is set to begin.	The SESSION document is deleted. This session id is subsequently removed from both users' document records. Both users are notified of the cancellation.
After X amount of minutes, it is presumed that the tutor is absent from session (After the session is set to begin, only the tutee is present, and thus only the tutee has completed the "handshake")	The SESSION document is deleted. This session id is subsequently removed from both users' document records. Both users are notified of the cancellation. Only the tutee (who was present) must rate the tutor based on performance (see req. 3.1.8).

Situation	Outcome
After X amount of minutes, it is presumed that the tutee is absent from session (After the session is set to begin, only the tutor is present, and thus only the tutor has completed the "handshake")	The SESSION document is deleted. This session id is subsequently removed from both users' document records. Both users are notified of the cancellation. Only the tutor (who was present) must rate the tutee based on performance (see req. 3.1.8).
After X amount of minutes, it is presumed that both the tutor and tutee are absent from session (After the session is set to begin, neither the tutor or tutee are present, and thus neither user has complete the "handshake")	The SESSION document is deleted. This session id is subsequently removed from both users' document records. Both users are notified of the cancellation.
Both tutor and tutee decide to end session earlier or later than expected	A new total amount owed is computed by recomputing the length of the session and multiplying it by the pay-rate stored in the document for that session. This new total amount owed overwrites the old value.
The session ends (that be it on time, earlier than expected, or later than expected)	The transaction with the total amount owed is executed. The total amount owed is withdrawn from the tutee's outgoing payment account, and deposited into the tutor's incoming payment account. The SESSION document is deleted. This session ID is subsequently removed from both users' document records. Both users are notified that the transaction has been completed. Both the tutor and tutee must rate each other based on performance (see req. 3.1.8).

[This space intentionally left blank]

3.1.8 Tutor/Tutee Rating System (O: Owings)

After a session ends, the tutor and tutee must rate each other with up to five stars. If a user does not rate their partner, they will be locked out of the application, and will not be able to schedule any more sessions or search for other users until they submit a rating. A user's rating in tutor mode is separate from their rating in tutee mode. In either mode, a user has the ability to challenge any rating that they believe is unfair. Contested ratings must not be factored into the user's overall rating until a review is conducted. Ratings that are determined to be justified must be factored into the user's overall rating, and ratings that are determined to be unjustified must not be factored into the user's overall rating.

3.1.8.1 Tutee Rating System (O: Owings)

In tutee mode, the user's rating must be determined by averaging all of the ratings they have received as a tutee. Tutors must have the ability to view a tutee's overall rating when partnering with them.

3.1.8.2 Tutor Rating System (O: Owings)

In tutor mode, ratings must be divided into two categories. A rating received as a tutor must be factored into their overall average, and must also be factored into the overall average of ratings they have received as a tutor in the class they were tutoring. Tutees must have the ability to see both a tutor's overall rating and the class-specific rating when partnering with them.

3.1.9 Discovery Pages (O: Ordona)

The Discovery Pages provide users several capabilities to search for courses and tutors. The three pages shall be shown as selectable buttons on a bottom navigation bar.

3.1.9.1 Search Discovery (O: Ordona)

The Search Discovery page allows users to submit or modify a search query for a specific course or tutor. The following functional requirements must be met:

1. Provide the capability to type in an appropriate text box to search for a course.
2. Provide the capability to select from a list of universities via drop down menu.
3. Provide the capability to type in an appropriate text box to search for a tutor.
4. Provide the capability to send the search query via submit button to retrieve results from the appropriate database.

3.1.9.2 Text Discovery (O: Ordona)

The Text Discovery page displays available courses and tutors in a scrollable list. If a search query has been submitted by the tutee, the applicable courses and their associated tutors are displayed. The following functional requirements must be met:

1. Provide the capability to select a course to view specific information.
2. Provide the capability to select a tutor to view specific information.
3. Provide the capability to sort results in the following manner:
    a. Price (Low to High) - Available only if a search query has been sent
    b. Price (High to Low) - Available only if a search query has been sent
    c. Rating (Low to High)
    d. Rating (High to Low)
    e. Distance (Closest to Farthest)
    f. Distance (Farthest to Closest)

3.1.9.3 Map Discovery (O: Ordona)

The Map Discovery page is a visual representation of the user's location. If a search query has not been submitted by the tutee, available tutors within a specified range are shown on the map, with the default range being 10 miles. If a search query has been submitted by the tutee, the applicable courses and their associated tutors are displayed instead. The following functional requirements must be met:

1. Provide the capability to modify the search range of the map.
2. Provide the capability to select users on the map to view specific information.

3.1.10 Chat Messaging (O: Campbell, M1: Wojtowicz)

The chat messaging feature allows user-to-user direct messaging. Any user should be able to initiate a messaging session with any other user by navigating to their profile and clicking on the chat button. Messages should be received from the sender in real-time.

3.1.11 Session Scheduling (O: Wojtowicz)

Tutees will have the ability to send sessions request to tutors. Tutors will not have such ability. Sessions will be tracked within Tutor Dash and stored for X amount of time so that users can view their previous sessions.

3.1.11.1 Request Handling (O: Campbell)

A tutee will be able to send a scheduling request to a tutor in order to set up a tutoring session. The following requirements must be met to schedule a session:

1. Only tutees can send requests to tutors.
2. Only tutors can approve the request.

3. Tutors can reject a request.
4. Tutors can propose changes to the request.
5. The request will contain a date, time, and location.

3.1.11.2 Calendar Synchronization (O: Wojtowicz)

All Tutor Dash sessions will be kept track of within the application and stored within the database. In order to provide a more convenient user experience, these events will be merged onto a user's Google Calendar. The user must authorize this behavior to take place for synchronization to initiate.

3.1.11.3 Online Session Scheduling (O: Hessefort)

The tutors and tutees will be able to both agree to schedule a tutoring session held online. The following functional requirements must be met:

1. Provide the capability to schedule tutoring session meetings through the Google Calendar API.
2. Provide the capability to automatically add scheduled sessions as events to user calendars.
3. Provide the capability to edit and delete scheduled events.

[This space intentionally left blank]

3.1.12 Layout Navigation (O: Campbell, M1: Wojtowicz)

Each screen of Tutor Dash will connect together and flow using the following scheme. On the left-hand side is the name of the screen with an assigned number. On the right-hand side is the set of all screens (screen numbers) that are accessible from the screen on the left-hand side.

1. Login → {2, 4, 14}
2. Sign Up → {1, 3}
3. Transcript Upload → {2}
4. Map Discovery → {1, 5, 6, 7, 11}
5. Text Discovery → {1, 4, 6, 7, 11}
6. Search Discovery → {1, 4, 5, 7, 11}
7. User Profile → {4, 5, 6, 8, 11, 13}
8. Session Scheduling → {7, 9}
9. Active Session → {7, 10}
10. Session Review → {7}
11. Settings → {1, 4, 5, 6, 7, 12}
12. Edit Profile → {11}
13. Chat → {7}
14. Reset Password → {1}

3.2 Performance Requirements

3.2.1 Caching (O: Wojtowicz)

Users should be cached as often as possible to keep the daily number of reads to the database to a minimum. When a logged-in user exits the application, their user data *must* be cached. When a logged-in user leaves the main discovery activity, the collection of all users at their school which was populated when they entered the activity *should* be cached.

3.3 Assumptions and Constraints

3.3.1 Query/Storage Limits (O: Wojtowicz)

Database interactions are constrained to 5Gb of total storage, 50,000 document reads/day, and 20,000 document writes/day.

3.3.2 Time Frame (O: Wojtowicz)

The Tutor Dash prototype must be fully functional by the time the project ends. The allotted development time for the prototype is 15 weeks.

[This space intentionally left blank]

3.4 Non-Functional Requirements

3.4.1 Development Environment/SDK (O: Ordona, M1: Wojtowicz)

The application shall be written in Android Studio (v. 3.4 with target SDK of 28 (minimum of 26))

3.4.2 Google Maps (O: Ordona)

The application shall utilize the Google Maps API for map functionalities.

3.4.3 Firebase (O: Ordona)

The application shall utilize Firebase for database storage and operations. User passwords and other sensitive information will be stored securely using Firebase Auth.

3.4.4 Braintree (O: Ordona)

The application shall utilize Braintree APIs for payment handling. (O: Ordona)

3.4.5 Google Calendar (O: Wojtowicz)

The application shall utilize Google Calendar to handle schedule synchronization and host tutoring sessions via web-conferencing.

3.4.6 Operating System (O: Wojtowicz)

The application shall be runnable on Android devices running Nougat v.7.0 and higher.