**Functional Specification**

**Grinds Web App**

**Group Members**

· **­**Yann Ndjatang: 18337813

· Fawwaz Kekere-Ekun: 18408814

**Table of contents**

**1.Introduction**

· 1.1 Overview

· 1.2 Business Context

· 1.3 Glossary

**2. General Description**

· 2.1 Product / System Functions

· 2.2 User Characteristics and Objectives

· 2.3 Operational Scenarios

· 2.4 Constraints

**3. Functional Requirements**

· 3.1 Registration

· 3.2 Login

· 3.3 Selecting User Instance

· 3.4 Profile

3.5 Searching for tutor

3.6 Paying for Tutor

3.7 Taking a Skill Gap Identification Quiz

3.8 Viewing the data

3.9 Messaging

3.10 Zoom face-to-face lecture

3.11 Rating System

3.12 Logout

**4. System Architecture**

· 4.1 System Architecture Diagram

**5. High-Level Design**

· 5.1 Context Diagram

· 5.2 Data Flow Diagram

· 5.3 Logical Diagram

· 5.4 Use Case Diagram

**6. Preliminary Schedule**

· 6.1 Task List

· 6.2 Gantt Diagram

**7. Appendices**

**1. Introduction**

**1.1 Overview**

The project we are working on is a tutoring web application called Tutor-O made for both tutors and students to use. This application's main goal is to help create a platform on which students can find a tutor for a subject they need extra help with. The primary functionality of this app is to provide a platform with the tools necessary for tutors to aid students to their best of their ability remotely and vice versa for students to find and learn from tutors from the comfort of their homes.

As briefly mentioned before the app will have two different users, “Students” and “Tutors”, with different functionality for each. It will allow for students to set up an account and find a tutor to teach them in a subject of their choice and likewise a tutor will be able to set up an account to be found by students and be able to provide an in-depth learning experience using the features that the app offers.

Our main objective is to create an environment to make the lives of both students and tutors easier. Being available to meet tutors or students in person can be a tough or near impossible task for a lot of people especially during these times, but by having a web app like this in which both parties can meet remotely it takes out the factor of distance or travel issues and provides a more organised platform rather than sending notes via social media.. This app will allow for students to receive extra-curricular assignments and also have a dashboard in which they can see their own personal progress which will be dictated by their tutor.

Our app will allow students to set up an account and then search for tutors through a dashboard. When a student has found a tutor, that tutors services will be blocked by a pay wall that will only be available after a student has set up a payment plan with the tutor. When a tutor has been selected by a student, the tutor will be notified that they have received a new student and be prompted to contact the student.

Our main feature for this application will be the aptitude test that tutors will be able to give to students. This test here will allow tutors to quickly assess a students strong and weak points so that they can give a more personalised learning experience said students. This allows the tutors to easily create a personalised workflow for students which will save the time of learning things that the student is already good at. This will also enable the students to be able to see where they lack and excel at so that they know why their assignments are being given to them.

**1.2 Business Context**

Here are few examples of what could be done with this application in terms of business:

* In relation to our project, a possible business context would be to potentially upload it to the Google Play Store.
* Our app could contain advertisements from other companies which would allow a steady flow of revenue to be generated.
* Schools could integrate customised versions of our app into their network in which teachers and students could use it to interact remotely with each other.

**1.3 Glossary**

**2. General Description**

**2.1 Product/ System Functions**

Here we have the main functions that the app is expected to have by the end of our development cycle. As shown there will be slightly different features depending on what kind of account you make and all these features rely on a user having made an account first. Although this list of features is ideal to a well working and fully operational web app, the features are subject to change at any time over the course of the development.

### Students:

* Go online to the web app and create an account.
* Log into account
* Reset account password, where necessary
* Apply setting to look for tutor of your subject
* Find range of tutors on tutor dashboard
* Pay for tutor
* Check assignments from tutor
* Have interactive in-app messaging with tutor
* Rating system
* User profile
* Log out of account

### Tutors:

* Go online to the web app and create an account.
* Log into account
* Reset account password, where necessary
* Advertise services on tutor dashboard
* Receive payment
* Create aptitude test for students
* Give assignments to students
* Create live online lecture for students
* Have interactive in-app chat with students
* User profile
* Log out of account

**2.2 User Characteristics and Objectives**

The users that we plan to target for this product are students from second level education ranging from 1st year to 6th year. This is because we found from personal experience that there is a greater need for tutors and a big-already established market in this level of education. Students are less likely to have come to grips with proper study routines and may not have the time or opportunity to gather with other students to help each other with work. This app will allow students to get the help that they need from experienced tutors and help them to become independent in terms of studying.

The user interface will allow for a simple, yet appealing user friendly experience. We expect the app to be as simple and straightforward as possible so that students can focus on actually learning about their desired subjects rather than spending time trying to figure out the layout of the web app.

**2.3 Operational Scenarios**

As the structure of Tutor-O has two form of users, there will be distinctive instances of each scenario and despite the fact that they carry out similarly, they are completely separate. Our varieties of user are Student and Tutor.

**User signs up**

**2.4 Constraints**

* Time constraints: Because of the limited time that there is to develop this web app we must make sure to focus on the core functionalities of the product and possibly integrate other less urgent features at a later date.
* Internet constraints: The web app cannot be accessed by users or tested by developers unless there is a stable connection to the internet.

**3.Functional requirements**

**3.1 Registration**

* Description

Registration is the first thing that will need to happen when a student or tutor has decided to create an account on the web app. It will require said user to provide a valid email address, a name and a date of birth. The user will be asked here whether they wish to register as a student or tutor. The user will then be verified by the system sending them a verification email and them clicking a link within that email. It should also be noted that if the domain of the email given does not match our list of acceptable domains then the email will not be accepted. To verify that somebody making a student account is actually a student that will need to sign in with their secondary school email address and likewise the tutors will have to provide a valid degree certificate or teacher email address from the school or institution that they work in to show that they are eligible to to actually tutor students.

* Criticality

This function is of utmost importance due to the fact that without this working the other app functionalities are almost useless as there will be nobody to use said functionalities. It will also be used to make sure that there are no fake users on the website for example, comedy signing up as a tutor to receive payment from students without actually giving any valid tutoring.

* Technical issues

We will need to find a valid way of implementing authentication software into the code and after some research it is most likely that SQLlite authentication will be used to verify students. Pythons Authlib may also be used as an alternative or in conjunction with SQLlite but these are subject to change during development.

* Dependencies with other requirements

None

**3.2 Login**

* Description

Logins come directly after registration and are used to verify that a user is already registered within our database.A user will first be asked whether they wish to login as a student or a tutor. This is so the system knows which database to check the input information against. When a user inputs their information into the required fields. The data will be checked against our stored data and if it happens not to be in our database the user will be rejected and a popup message will tell them that said account cannot be found due to email or password being wrong. There will also be the option to reset a password if a user has forgotten it.

* Criticality

This function is primarily based on security. We will use it to ensure that a random user will not be able to access our web app without going through the registration process first to create an account.

* Technical issues

The data entered must be checked against a vast range of already stored names, email addresses and passwords. This will be done using SQLliteauthentication and python Authlib but is subject to change during the course of development.

* Dependencies with other requirements

A user must have first successfully registered an account with the registration function before being able to successfully login.

**3.3 Selecting User Interface**

* Description

A user will not manually have to select which interface they wish to use as they will be automatically directed to the interface that matches their account. A student will be restricted to using only the student interface whilst the tutor will be restricted to using only the tutor interface.

* Criticality

This step is important so that the web app can run smoothly and that users don’t run into issues such as not being able to access features that are supposed to be available to them or being able to access features that aren't supposed to be available to them.

* Technical issues

This will come down to our program simply checking whether the user chose to login as a student or tutor and then displaying a different interface through the web app based on their selection and account.

* Dependencies with other requirements

User must have successfully logged in as either a student or a tutor.

**3.4 Profile**

* Description

When a user has registered an account their personal profile will be created based on their data. As a student the profile will allow students to check which tutors they are signed up to, what their projected grade is in a subject with a tutor(based on their results that they are getting in thor assessments from their tutors.). Students will have the option to add a picture to their profile. They will also be able access general settings from here and logout. As a tutor the profile will allow tutors to see what students are signed up to them and how well each student is doing with them. Tutors will have the option to add a picture to their profile. They will also be able access general settings from here and logout.

* Criticality

A profile is very simple yet it is fundamental to the user interface as it makes the web app have more of a professional look whilst also giving users a place to access information about their account.

* Technical issues

If a student or tutor is to upload a picture it will take up a lot of space so data will be sted in cloud storage rather than being stored onto our machines database.

* Dependencies with other requirements

User must have successfully logged in as either a student or a tutor.

**3.5 Searching for a Tutor**

* Description

When a student user has created his/her account, they will have access to the ‘Student’ user interface. This involves a homepage with a search bar where they can then search for a tutor. Our search bar will include filters with which a student can search for a tutor based on the subject they’re looking for and the level of the subject. A list of tutors will appear based on the search criterias. Students will then be able to to view the available tutors which contain a rating and student feedback.

* Criticality

This is essential for our web app. This will show a list of tutors. When a student clicks on a tutor’s profile, they will be able to see the tutor’s biography, rating and feedback from other students that had them before

* Technical issues

This search will query the database and return them to the student screen, in a scroll view

* Dependencies with other requirements

Requires Tutors to have made an account and added to the database in order for a student to to search the options

**3.6 Paying a Tutor**

* Description

This function is for when a student a has found a tutor and would like to have him/her as their tutor. The student will be able to briefly message the Tutor about his/her requirements. If the tutor matches their needs, they will be able to pay via online payment

* Criticality

The payment will be going to the tutor. This feature will then unlock all the other features between the tutor and the student.

* Technical issues

We are still unsure of how exactly we will go about creating this feature, as of this moment, we plan on using some sort of Django payment integration

* Dependencies with other requirements

Requires a log in user to have first interact with a tutor

**3.7 Taking a Skill-Gap-Identification Quiz**

* Description

Once a student has acquired a tutor, the tutor will be able to assign a skill gap quiz to

their student. This will permit the tutor to get a better understanding of their student and allow them to identify areas where they need to improve on and create a personalised program for their student.

* Criticality

This will ensure that the student is in good hands and will be getting all the help they need.

* Technical issues

We are still unsure on how exactly we will go about creating this feature, but at the moment we plan on creating some sort of python algorithm.

* Dependencies with other requirements

This requires the student to have acquired a tutor via payment.

**3.8 Viewing the data**

* Description

Once the student has taken the the SGI(Skill-Gap-Identification) quiz. The student

will be able to view the data and see their results.

* Criticality

With a student taking the quiz we must ensure that they’re able to view their results.

This is important for the tutor as this lets them know the strengths and weaknesses of the student.

* Technical issues

We plan to implement this with some fancy visuals done by javascript and react.

* Dependencies with other requirements

This requires the student user to have taken the quiz.

**3.9 Messaging**

* Description

We want the users of our app to have the ability to communicate with one another.

Each user’s messages will be stored and organised, within each individuals inbox.

No searching option is available with messaging. This is mainly how student’s will be

able to communicate with each other.This will include the ability to for each user to send files and images.

* Criticality

It is an important feature as this how tutors will communicate with their students. It will not require their phone number publicly on the web app. They can create email alerts when one sends a message.

* Technical issues

This will be created with django messaging.

* Dependencies with other requirements

Require a log in user with a tutor or student acquired.

**3.10 Zoom face-to-face lecture**

* Description

We plan on providing the ability for a tutor to set up a zoom meeting

* Criticality

This is not a key feature, this is only a feature we have discussed that we may add.

* Technical issues

we are still unsure on how exactly we will go about creating this feature, but at the moment we plan on using the zoom API

* Dependencies with other requirements

To have acquired a student.

**3.11 Rating System**

* Description

Students will be able to rate a tutor based on their services either during or after the

tutoring has been completed.

* Criticality

This is not a key component of our web application. A good user rating can provide a level of reassurance for those who are looking at the same tutor. It also ensures that each user has a good overall experience with the web application.

* Technical issues

We plan on creating this feature with django. We were also considering the possibility

of deactivating a tutor with numerous low ratings.

* Dependencies with other requirements

This requires that student has been tutored by said tutor first along with having exchanged with the tutor for a minimum of 1 month.

**3.12 Logout**

* Description

With this function, a user is provided with the option to sign out of the app, if they

Wish. It is put in place so that no user is confined to any particular device. When they chose to log out, it will not affect their stored user information and they can sign back in at any given time.

* Criticality

With a user having the option to sign in, we must ensure that we provide them with the option of signing out. Also, this ensures that in the situation, where a user may acquire a new device, they’ll be able to successfully log back in without their account being affected.

* Technical issues

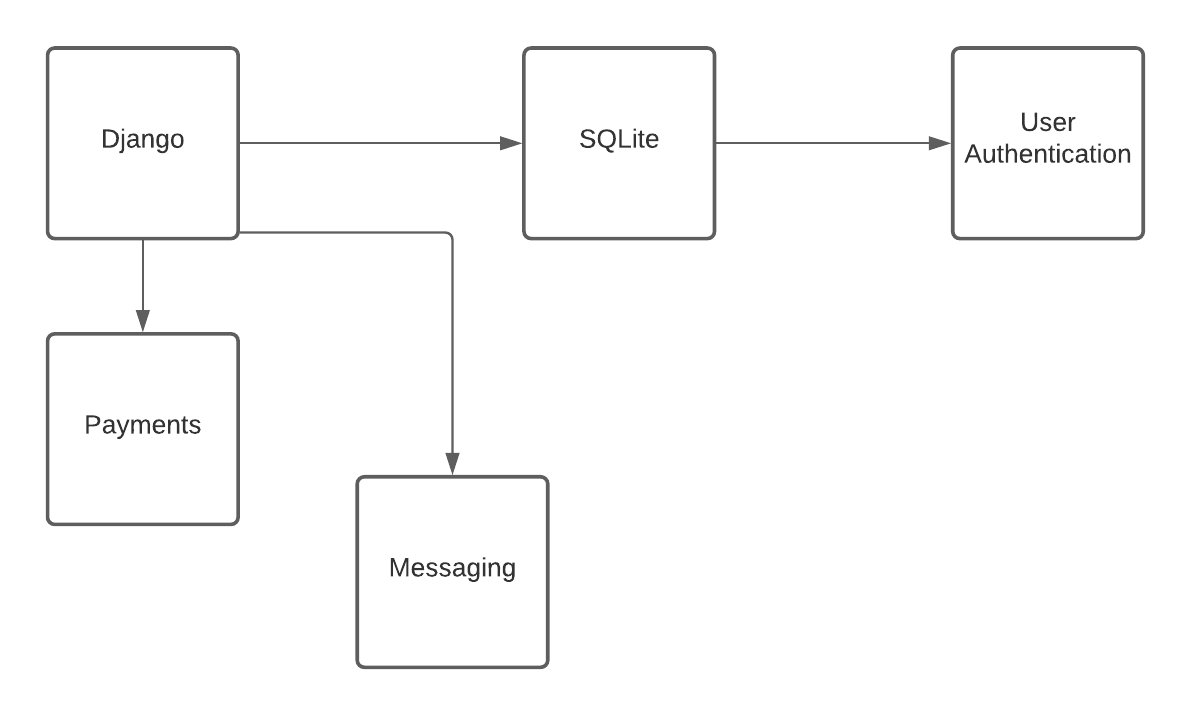
With this option, it means the user will be disconnected

* Dependencies with other requirements

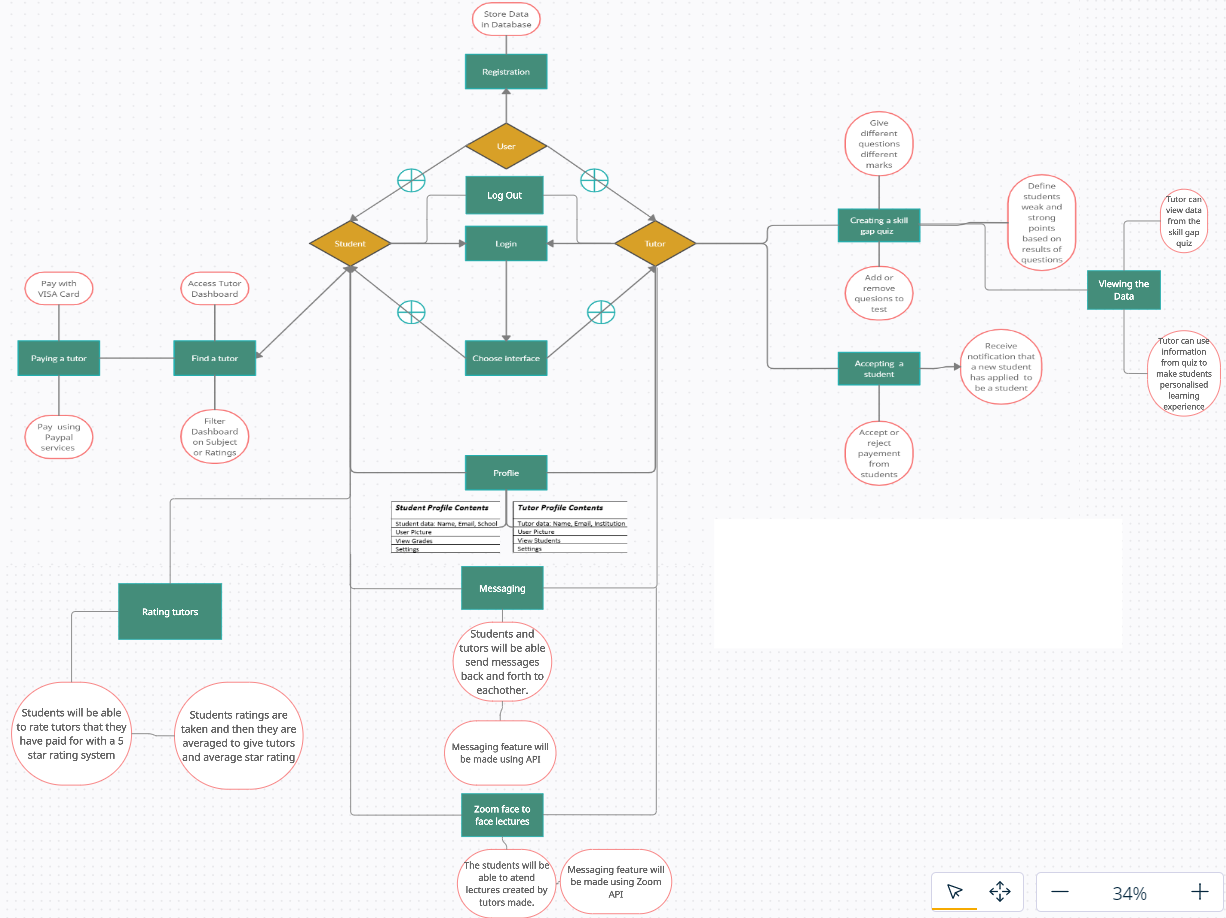
Requires the user to have first been successfully logged in, before logging out.

**4. System Architecture**

* 4.1 System Architecture Diagram

[](https://lucid.app/documents/edit/185320be-2c56-4f97-9b29-e2997ae0ce12/0?callback=close&name=docs&callback_type=back&v=212&s=595.4399999999999)

**5.1Data flow Diagram**

****