Tindents

Project Backlog

Sriharsha Valluripalli
Rahul Balla
Andre Chang
Meriem Bounab
Brent Singh Sharkey

Problem Statement

On a college campus, there are many students that are looking to have supplemental instruction for their coursework. One of the common ways to gain more help is to hire a tutor. Currently, there are many different websites that aim to connect students with tutors that they are able to hire. However, these websites can be overwhelming with many different lists of tutors to choose from, making this process difficult and stressful. The app that we want to develop aims to simplify this process by using a more user-friendly UI that replicates an interface that users are already familiar with. Our app also provides students with tutors that are experts at courses in which they need help.

Background Information

Some of the most popular tutoring apps out there, such as Wyzant, are too broad and not catered towards university students who need help with a particular subject. Also, the UI associated with these apps makes it a little hard for the user to navigate between screens and select the tutors, making the process time consuming. Therefore, there is a need to provide university students with a platform to connect with tutors specific to their needs in a quick and convenient manner. It would be more convenient if we can provide students with an interface they are comfortable with such as viewing a tutor's profile one by one and choosing to like or dislike the tutor by swiping either right or left respectively. The user will also be able to filter tutors by distance, courses they teach, ratings, and price. The interface will make it quick for the students to select tutors and form study groups without having to do a tedious job of navigating between screens.

Environment

Since we will developing an iOS application, we will be using Swift to develop the front-end. For the backend, we will be using a Flask based python server to handle user requests. A MariaDB Database hosted with Amazon AWS will be used to store the student's and tutors details. For the students, details such as their username, password, personal info, and subjects for which they need tutoring will be stored in the database. For the tutors, details such as their username, password, personal info, subjects they can tutor, and their ratings will be stored in the database.

Functional Requirements

Backlog Id	Functional Requirement	Hours	Status
1	As a student, I would like to be able to create a student account	10	Planned for Sprint 1
2	As a student, I would like to filter tutors by rate charged	10	Planned for Sprint 2
3	As a user, I would like to filter tutors by distance	10	Planned for Sprint 2
4	As a student, I would like to have a feed of tutors that I can swipe through	15	Planned for Sprint 1
5	As a student, I would like to swipe right on tutors that I like	5	Planned for Sprint 1
6	As a student, I would like to swipe left on tutors that I don't like	3	Planned for Sprint 1
7	As a student, I would like to see contact information for tutors that I matched with	5	Planned for Sprint 1
8	As a student, I would like to see past tutors that I have matched with	10	Planned for Sprint 2
9	As a student, I would like to be notified of a match with a tutor	10	Planned for Sprint 2
10	As a student, I would like to see the schedule for the tutor I matched with	15	Planned for Sprint 2
11	As a tutor, I would like to be able to create a tutor account	10	Planned for Sprint 1

12	As a student, I would like to be able to login to a student account	5	Planned for Sprint 1
13	As a tutor, I would like to be able to login to a tutor account	5	Planned for Sprint 1
14	As a tutor, I would like to set my hourly charges	5	Planned for Sprint 1
15	As a student, I would like to see the hourly charges of the tutors	5	Planned for Sprint 1
16	As a tutor, I would like to logout of my account	5	Planned for Sprint 1
17	As a student, I would like to logout of my account	5	Planned for Sprint 1
18	As a tutor, I would like to filter students by subjects to study	10	Pushed for Sprint 2
19	As a tutor, I would like to have a feed of students that I can swipe through	10	Planned for Sprint 1
20	As a tutor, I would like to swipe left on students that I don't like	5	Planned for Sprint 1
21	As a tutor, I would like to swipe right on students that I like	3	Planned for Sprint 1
22	As a tutor, I would like to be notified of a match with a student	15	Planned for Sprint 2
23	As a tutor, I would like to see the schedule for the student I matched with	15	Planned for Sprint 2
24	As a student, I would like to see contact information for tutors that I matched with	3	Planned for Sprint 1
25	As a user, I would like to be able to email my match	7	Planned for Sprint 2
26	As a student, I would like to be able to rate tutors	10	Planned for Sprint 2
27	As a tutor, I would like to be able to rate students	5	Planned for Sprint 2
	Total	218	

Non-Functional Requirements

Platform*:* Tindents will be an iOS app available on all apple devices. The app will be divided into a back end and front end section to make it easier to work as a team. Front end will be implemented in Swift while back end will use Flask server and SQL database.

Usability: The interface must be straightforward and easy to use and navigate. The account creation and login page should be simple and allow the user to easily register as either a tutor or student. When a user swipes, the UI should provide clear indication of what direction a user is swiping and all relevant information of the tutor/student should be displayed. The tutor data and rating will be presented in a way that is easy to read and students should find it easy to navigate between different tabs.

Security: The app should protect the user's username/password and also their personal information. Flask's extensions for database access and user authentication come out of the box with measures that protect against common exploits like SQL injection. All the API requests will be authenticated before serving the user's requests.

Reliability: The app should be reliable in terms of only displaying information that is requested by the user. Also, the app should not crash when there is an error in completing the user request or a UI feature has failed to display.

Use Cases

Case 1: Student creates account	
Action	**System Response**
1. Student enters account details	3. Creates an account with the student details and stores in database
2. Student clicks on Register button	4. Redirect user to home screen

Case 2: Filter tutors by price	
Action	**System Response**
1. Student enters price range (min price and max price)	3. API call is generated based on price range entered by user
2. Click on Filter button	4. Receive and parse response from server
	5. Tutors are displayed based on filter criteria

Case 3: Filter tutors by distance	
Action	**System Response**
1. Student enters max distance	3. API call is generated based on price range entered by user
2. Click on Filter button	4. Receive and parse response from server
	5. Tutors are displayed based on filter criteria

Case 4: Students view feed of tutors	
Action	**System Response**
1. Students enter filter criteria	3. API call is generated based on filter criteria
2. Click on search button	4. Response is received and parsed
	5. Display tutors one at a time so that tutors can either swipe right or left
	6. Display next tutor after swiping

Case 5: Swiping right on tutors	
Action	**System Response**
1. Students swipe right on tutors they like	2. Tracks which tutors are liked using boolean values
	3. Displays next tutor

Case 6: Swiping left on tutors	
Action	**System Response**
1. Students swipe left on tutors they dislike	2. Displays next tutor

Case 7: View contact information of tutors	
Action	**System Response**
1. Student clicks on tutor they matched with	2. Displays contact information of tutor

Case 8: View list of matched tutors	
Action	**System Response**
1. Student click on button to see list of matched tutors	2. Generate API call to return list of matched tutors
	3. Receive and parse response from server
	4. Navigate to another screen to display matched tutors

Case 9: Notify students of a match	
Action	**System Response**
	1. Update boolean value to true in database if student and tutor match with each other
	2. Generate notification and display as an alert to student
3. Student clicks on OK	
	4. Navigate to another screen which displays matched tutors

Case 10: View tutors schedule	
Action	**System Response**
1. Student clicks on “View Schedule” to view tutors schedule	2. API call is generated and database is queried to get tutor schedule
	3. Receive and parse response from server
	4. Navigate user to another screen to display tutor schedule

Case 11: Tutor creates account	
Action	**System Response**
1. Tutor enters account details	3. Creates an account with the tutor details and stores in database
2. Tutor clicks on Register button	4. Redirect user to home screen

Case 12: Student login	
Action	**System Response**
1. Student enters credentials	3. Student is authenticated based on entered credentials
2. Student clicks on Login button	4. Redirect user to home screen

Case 13: Tutor login	
Action	**System Response**
1. Tutor enters credentials	3. Tutor is authenticated based on entered credentials
2. Tutor clicks on Login button	4. Redirect user to home screen

Case 14: Tutor sets hourly rate	
Action	**System Response**
1. Tutor goes to profile	4. New hourly rate is updated in database
2. Clicks on Edit profile	5. Redirects user to their updated profile
3. Enters new hourly rate	

Case 15: Student views tutor hourly rate	
Action	**System Response**
1. Student clicks on tutor they want to view	2. Displays detailed information of tutor which includes hourly rate

Case 16: Tutor logout	
Action	**System Response**
1. Tutor clicks on Logout button	2. Redirect user to login screen

Case 17: Student logout	
Action	**System Response**
1. Student clicks on Logout button	2. Redirect user to login screen

Case 18: Filter students by subject	
Action	**System Response**
1. Tutor selects which subject students need tutoring	3. API call is generated based on filter criteria
2. Click on Filter button	4. Receive and parse response from server
	5. Display students who need tutoring in that subject

Case 19: Tutors view feed of students	
Action	**System Response**
1. Tutors enter filter criteria	3. API call is generated based on filter criteria
2. Click on search button	4. Response is received and parsed
	5. Display students one at a time so that tutors can either swipe right or left
	6. Displays next student after swiping

Case 20: Swiping right on students	
Action	**System Response**
1. Tutors swipe right on students they like	2. Tracks which students are liked using boolean values
	3. Displays next student

Case 21: Swiping left on students	
Action	**System Response**
1. Tutors swipe left on students they dislike	2. Displays next student

Case 22: Notify tutors of a match	
Action	**System Response**
	1. Update boolean value to true in database if student and tutor match with each other
	2. Generate notification and display as an alert to tutor
3. Tutor clicks on OK	
	4. Navigate to another screen which displays matched students

Case 23: View student schedule	
Action	**System Response**
1. Tutor clicks on “View Schedule” to view student’s schedule	2. API call is generated and database is queried to get schedule schedule
	3. Receive and parse response from server
	4. Navigate user to another screen to display student schedule

Case 24: View contact information of students	
Action	**System Response**
1.Tutor clicks on student they matched with	2. Displays contact information of student

Case 25: User emails match	
Action	**System Response**
1. User clicks on “Send Email” button	2. Navigates user to another screen where they type the email
3. User clicks on “Send”	4. Sends email to recipient
	5. Returns user to the previous screen

Case 26: Student rates tutor	
Action	**System Response**
1. Student selects how many stars to give to tutor	3. Updates rating for tutor in the database
2. Student clicks on “Assign Rating” button	

Case 27: Tutor rates student	
Action	**System Response**
1. Tutor selects how many stars to give to student	3. Updates rating for student in the database
2. Tutor clicks on “Assign Rating” button	