Full-Stack Web Development Program

CAPSTONE Project Description, Scope & Guidelines

Project Code Name : 2TR-Now (Tutor Now)

Version : Sep 13, 2019 - S

Author : Ramses Trejo

The Idea

Help connect students and tutors anywhere, anytime.

Description

What do you do when you need specific help with a topic that you will have an exam on the next day and there is no one around to ask? or when you have an imminent project delivery and you are stuck with non-working code or when debugging a critical problem in the project and nothing you do seems to help.

2TR-Now solves the problem. A collaborative web-based service that connects students and tutors in real time - safe & effective. Students find available tutors on the platform and connect and interact with them via a web conference. Students have the option to share their screens for providing further context and information about the problem or question at hand.

Requirements

Profiles

- Students must be able to create an account with the following information: username, location, dob, school program, so that tutors can have some context about who they are talking to.
- Tutors must be able to create an account with the following information: username, location, program, specialty, so that the application can be match tutors with requests by students.
- Students and Tutors should be able to add a picture to their profile [Stretch]
- Tutors must provide a criminal record report [Stretch]

Tutoring Sessions

- Student should be able to create a request by using a form, entering the following mandatory information : subject, topic, context, tasks already completed, question to be answered, expected tutoring session duration & language.
- Students should be able to view tutors' username, location, program and specialty before confirming a session.
- Students should be able to rate the answer and support received from the tutor. Rate levels are : Complete, Partial and Not useful.
- Tutors should be able to rate Students in politeness, openness and flexibility.
- Students should be able to see a list of tutoring requests they have made.
- Tutors should be able to see a list of tutoring requests they have assisted.

Search

- Students should be able to reach out to tutors who are currently available in real time. [Stretch]
- Students should be able to search all tutors by criteria like program, specialty, time zone etc. and send a message to chosen tutor. [Stretch]
- Students should be able to use Google Maps to let a student pick out a specific location for looking for tutors, either through searching a location on the map or through typing in a name and choosing from a list (typeahead) [Stretch]

Scheduling

- Students should be able to schedule tutoring sessions with Tutors in the future and get confirmations and reminders for these sessions [Stretch]

Payments

- Students and tutors should be able to agree on a payment type: pay-per-time or pay-per-resolution, before the interaction begins. [Stretch]
- Students should be able to purchase credits using pay-pal or a credit card. [Stretch]

Reviews

- Students should be able to rate and write reviews for tutors, based on the following criteria : communication, quality service and knowledge.

Multi-device support

- Students should be able to access the platform through a mobile device. [Stretch]

Notes

1. Integration with a Web conference tool is not in scope for this project
2. *Requirements marked as [Stretch] are not part of the MVP and they will be optionally implemented by project teams only when MVP requirements have been completed.*

Design Decisions Guidelines

- Focus on the application as a platform to connect students and tutors. It is not necessary to tightly integrate web-conference software, as there are a number of options of conference software in the market.
- UX design should follow best practices but design does not need to be overly stimulating.
- Focus on the problems/pains that are the platform is trying to solve : students get close to immediate help on a school problem and tutors earn extra money by sharing what they know well.

Technology Stack Requirements

Front-end

- React.js
- jQuery / JavaScript ES5
- CSS grid and UI frameworks (bootstrap)
- SASS for styling (optional)
- AJAX (optional)

Back-end

- JavaScript ES6
- Node / Express
- RESTful routes
- Relational Database (MySQL or PostgreSQL)
- NoSQL (MongoDB) (optional)
- Google Cloud for hosting

Collaboration

- git for version control
- Trello for project management (tool)

Modelling

- Adobe XD for Wireframe design
- Draw.io for ERD / Logical model design
- Visual Studio Code or Sublime Text for source code

Project Execution Guidelines

Project Communication

It is highly recommended that project teams practice their scrums at least every 4 days, in order to allow for better communication and tasks follow up. A Scrum Master and a Scrum Product owner should be appointed from the beginning of the project. These roles could be rotated every 1 or 2 weeks for better workload distributor. Scrums should be kept simple and should focus on progress and removal of obstacles:

- What has been accomplished since last Scrum
- What will be worked on next
- What obstacles are being (or might) face

Project Workflows

Team members should identify project milestones and mark deadlines for each milestone on their calendars.
Once the Git repository has been setup, team members should clearly define the workflow for version control that the team will follow. Include creation of branches and steps to resolve conflicts. The idea is to minimize code conflicts and lost code and time due to computer crashes.

Projects Task Distribution

Team members should determine how the project tasks will be distributed for each sprint (agreed-upon & time-boxed iteration for development of pieces of software). It is recommended that high risk tasks and proofs of concept should be tackled first to mitigate risks that could jeopardize the successful completion of the project and requirements are captured in the project management tool as soon as possible.

Project Setup

The following steps are continuation of initial project setup :

1. Setup a requirements management tool (Trello)
2. Create a code repository on GitHub and give access to all team members.
3. Design initial web application architecture
4. Create a project scaffold (from boilerplate code if possible)
5. Create and setup database & initial migration
6. Create seed data for database (should not be included as part of repository). Each team member will have to seed their local database.
7. Define coding guidelines and naming conventions for file names, variables, route names, table names, databases to ensure consistency and minimize confusion while reading other members' code.

Feature Development

Team members work on the development of the web application, adding feature by feature as sprints are completed while observing project milestones. It is recommended that no new features are to be added one week before the last session so that there is enough time for preparing the application demo and minimize the risk of introducing last minute bugs. Account for time that will be dedicated for testing, fixing bugs, refactoring and cleaning code.

Scrum Meetings

During project execution, the instructor will participate in scrum meeting once or twice per iteration and use every opportunity for encouraging the team and fostering a positive team dynamics. Team members direct their own Scrum meetings. For recommendations of Scrum practices, see Project Communication guidelines above.

Project Deployment

Team members should deploy their code often, at least once a week and at the end of each sprint (iteration), to the designated deployment server. It is critical that a production ready instance of the application is ready for Demo Day. Please ensure that every deployment is tested against the requirements (user stories & their acceptance criteria).

Projected Project Plan & Schedule

<table>
<tr><th>WEEK # / ACTIVITIES</th><th>DELIVERABLES</th></tr>
<tr><td>WEEK #0

● Review requirements (epics).Teams will take these epics and they will write well defined user stories that describe requirements and interactions of users and application based on user needs and pain points.
● Setup of git repository and trello project
● Design initial web application architecture with initial technology stack
● Create initial screen designs for 3 pages such as front page, login & registration pages drawn by hand (or with a tool of your preference)</td><td>● Git repository
● List of questions and clarifications
● Overview of web application architecture & technology stack
● Initial wireframes for 3 pages : front page, login & registration</td></tr>
<tr><td colspan="2">SESSION #1 & SESSION #2

● Clarification of requirements with Instructor
● Feedback by instructor on web app architecture & UX design
● Project planning
● Projected Sprint #1 tasks:
○ Refine web app architecture
○ Refine UX design for front-page, sign-in and registration (wireframes)
○ Development of front-page, registration form (start)
○ Design of database tables (start)</td></tr>
<tr><td>WEEK #1

● Setup of your front-end server and database
● Projected Sprint #1/Week #1 tasks:
○ Refine wireframes based on feedback (front-page, sign-in, registration[sign up])
○ Development of front-page (View)
○ Development of registration (View)
○ Development of sign-in (View)
○ Design of ERD/Logical Model for student, tutor, request</td><td>● Sprint #1/Week #1 status report</td></tr>
<tr><td>WEEK #2

● Clarification of requirements with Instructor</td><td>● Refined web app architecture & technology stack
● Refined wireframes (sign in, registration)</td></tr>
</table>

<table>
<tr>
<td>

- Projected Sprint #1/Week #2 tasks:
  - Creation of initial MySQL DDL scripts for student, tutor, request tables
  - Development of registration (Controller) (using registration routes GET/POST)
  - Development of sign-in (Controller)
  - Design student profile view
  - Design tutor profile view
  - Design tutoring request form
  - Prepare demo of registration & sign-in

</td>
<td>

- Drafts of wireframes for student profile, tutor profile
- ERD/Logical model for student, tutor, request
- Initial MySQL DDL Scripts for table creation
- Code & scripts in Version Control
- Deployment of completed code
- Sprint #1/Week #2 status report
- Sprint #1 Demo - Registration & sign-in

</td>
</tr>
<tr>
<td colspan="2">

SESSION #3

- Sprint #1 Demo
- Feedback by instructor on web app architecture, UX and database design
- Clarification of requirements with Instructor
- Retrospective meeting (in-class)
- Projected Sprint #2 tasks:
  - Update & refactor registration implementation & sign-in implementation
  - Refine design of student profile, tutor profile view & tutoring request form
  - Development of student profile, tutor profile & tutoring request form

</td>
</tr>
<tr>
<td>

WEEK #3

- Projected Sprint #2/Week #1 tasks:
  - Refine design student profile view
  - Refine design tutor profile view
  - Refine design tutoring request form
  - Development of student profile (MVC)
  - Development of tutor profile (MV)

</td>
<td>

- Sprint #2 status report

</td>
</tr>
<tr>
<td>

WEEK #4

- Clarification of requirements with Instructor
- Projected Sprint #2/Week #2 tasks:
  - Development of tutor profile (Controller)
  - Development of tutoring request form (MVC)
  - Design list of requests view
  - Design tutoring requests view
  - Design tutor match algorithm/query
  - Prepare demo of Student profile page, Tutor profile page, tutoring request

</td>
<td>

- Sprint #2 status report
- Sprint #2 Demo - Student profile page, Tutor profile page, tutoring request

</td>
</tr>
<tr>
<td colspan="2">

SESSION #4

- Sprint #2 Demo
- Clarification of requirements & feedback with Instructor

</td>
</tr>
</table>

<table>
<tr><td colspan="2">

- Feedback by instructor on list of requests, tutoring requests, tutoring match algorithm
- Retrospective meeting (in-class)
- Projected Sprint #3/Week #1 tasks:
  - Refine design of list of requests view, tutoring requests view, tutor match algorithm/query
  - Development of list of requests view, tutoring requests view, tutor match algorithm/query
  - Updates & refactoring

</td></tr>
<tr><td>

WEEK #5

- Clarification of requirements & feedback with Instructor
- Projected Sprint #3/Week #1 tasks:
  - Refine design of list of requests view
  - Refine design of tutoring requests view
  - Refine design of tutor match algorithm/query
  - Development of list of requests (MVC)
  - Development of tutoring requests (M)
  - Development of tutor match algorithm/query

</td><td>

- Sprint #3 status report

</td></tr>
<tr><td>

WEEK #6

- Clarification of requirements & feedback with Instructor
- Projected Sprint #3/Week #2 tasks:
  - Development of tutoring requests (VC)
  - Refinement of tutor match algorithm/query (request status, associate with selected tutor)
  - Final updates and refactoring
  - Prepare demo of tutoring request page, request status change, associated tutor (+stretch functions)

- Suggested stretch functionality
  - Implementation of students' traits rating for tutoring requests by tutors
  - Generation of notification for tutor when a matching request is received
  - Implementation of rating of tutoring requests by students
  - Implementation of students' academic traits rating for tutoring requests by tutors

</td><td>

- Sprint #3 status report
- Sprint #3 Demo - Tutoring request page, request status change, associated tutor

</td></tr>
<tr><td colspan="2">

SESSION #5

- Final Project Demo

</td></tr>
</table>

- Final Remarks & Recommendations