Sprint 1 Planning Document

TEAM 13 - EDUYA

Tomasz Parzadka, Hans Allendorfer, Weiyi Zhou, Puja Mittal, Ashwin Prasad

SPRINT OVERVIEW

During this sprint, we hope to get the core of our platform up. For the frontend team, this means setting up a responsive design along with some features (profiles, tutor postings). The backend team will be focusing on getting the framework and database set-up to handle account registration and login, as well as setting up functions to drive the planned front-end features.

Scrum Master:

Weiyi Zhou

Meeting Schedule:

MWF 10:30-11:30AM

Risks/Challenges:

As we begin our project, many members aren't fluent in the technologies we are utilizing. Thus, research and time is needed for our member to feel comfortable before implementing.

Additional Notes:

Django uses the MTC structure where models and views are part of helping structure the backend. Contrary to common uses of *models* and *views* in reference to frontend, these are related to our backend in our document. Our MySQL edits are done via Django so we don’t need to explicitly modify the DB each time. Whenever we migrate the changes, Django generates the necessary queries.

USER STORIES

User Story #1:

Register for an account

#	Decription	Team	Owner	Time Est.
1	Setup DB to store account information	Back	Weiyi	15
2	Research Django and prototype	Back	Tomasz	3
3	Ensure that our REACT App will integrate with the Django App	Front	Puja	2
4	Implement email confirmation/error handling	Back	Tomasz	12
5	Research REACT and sending data via forms	Front	Ashwin	8
6	Create registration page	Front	Ashwin	6

Acceptance Criteria:

- Given that the DB is correctly set-up, user information will be stored in manner that is secure, organized, ensures data integrity, and is relatively easy to access.
- Given that registration is successful, their account credentials will be stored in the DB and the user will receive an confirmation email.
- Given that an error in registration occurs, information will not be stored in the DB and an error message will be displayed requiring the user to re-enter correct information.

User Story #2:

Log into my account

#	Decription	Team	Owner	Time Est.
1	Research REACT & Prototype	Front	Puja	7
2	Set up home page	Front	Puja	6

3	Research authentication methods	Back	Weiyi	5
4	Implement proper authentication checks	Back	Weiyi	5

Acceptance Criteria:

- Given our home page is setup correctly, it will function appropriately no matter what device our platform is accessed on.
- Given that our authentication is set up properly, a user will only be able to log in with the correct password.
- Given that our authentication is set-up properly, the plain text password will not be stored or be accessible to outside parties.

User Story #3:

Edit my profile (e.g. add a picture, contact info, personal info, etc.)

#	Decription	Team	Owner	Time Est.
1	Create user profile model	Back	Hans	9
2	Set up user profile page	Front	Puja	5
3	Set up user editing page and ensure it properly edits the DB	Front	Ashwin	6

Acceptance Criteria:

- Given the profile data is stored correctly, that same data is displayed when someone accesses their user profile page.
- Given one's user profile page can be accessed, one can edit their info, updating their user profile page and profile data.
- Given a user has added information to their profile, this information will be private to others, only being able to be seen by the user itself.

User Story #4:

Reset my password

#	Decription	Team	Owner	Time Est.
1	Research Django and prototype	Back	Tomasz	3
2	Emails the user a reset link	Back	Tomasz	12
3	Makes appropriate changes to the database	Back	Weiyi	8
4	Create Password Reset page	Front	Puja	3

Acceptance Criteria:

- Given the email reset link is implemented correctly, only the user will be able to reset their own password.
- Given the reset database function is implemented correctly, the only data reset upon user request will be their own password.
- Given the reset database function is implemented correctly, the new password will be stored appropriately and securely without potential exposure to outside parties.

User Story #10:

Create a posting looking for a tutor with specified subjects and location

#	Decription	Team	Owner	Time Est.
1	Research Django Model/Database functions	Back	Hans	9
2	Create a function that will allow for postings to be created given form data	Back	Hans	6
3	Create form for creating a tutor posting	Front	Ashwin	8

Acceptance Criteria:

- Given that the server properly responds to the data, when I have submitted a form containing data related to a tutor request, I should be redirected to a confirmation page or tutor request page after submission.
- Given that the server properly responds to the data, when I have submitted a form with incorrect or missing data related to a tutor request, I should be redirected to the form screen with error messages regarding the submitted form.
- Given that the server properly handles the valid submitted data, when I submit valid data then I expect the server's database to retain this exact data until further modification.
- Given that the server properly handles the submitted data, when I submit the valid data then I expect this exact data to be displayed on the front-end as submitted.

User Story #11:

Opt to be a tutor and have a public profile with reviews and information

#	Decription	Team	Owner	Time Est.
1	Research Django MVC structure and SQL commands	Back	Hans	9
2	Create a basic tutor profile view	Back	Hans	6
3	Create toggle on the profile screen to opt to be a tutor	Front	Ashwin	2
4	Integrate Material Design Components to be used to style the REACT App	Front	Puja	2
5	Tutor profile card	Front	Puja	6

Acceptance Criteria:

- Given that the student has opted to become a tutor, present them with a form regarding their tutor listing.

- Given that the student has filled out the tutor form, create the tutor listing and store this in the database if data is appropriate and complete. Otherwise, redirect the the student to the form with errors listed.
- Given that the student has successfully opted to become a tutor, display their profile view according to the data they submitted.

REMAINING BACKLOG

1. **Register for an account, storing account information securely in a database.**
2. **Log into my account**
3. **Edit my profile (e.g. add a picture, contact info, personal info, etc.)**
4. **Reset my password**
5. Allow users to navigate between different sections of the website efficiently
6. See a list of available tutors
7. Sort the tutor list by price, class, rating, etc.
8. See the profile for the tutor that posted a listing
9. Leave reviews for tutors including ratings, pricing, and other comments
10. **Create a posting looking for a tutor with specified subjects and location**
11. **Opt to be a tutor and have a public profile with reviews and information**
12. View all current requests for a tutor
13. Filter tutor postings by subject, time, price, rating
14. Add/remove courses I am taking
15. Have course information auto-filled from Purdue databases
16. View course related information (e.g. professor names)
17. View all preexisting tutors, office hours, or SI sessions for each course, scraped from Purdue-based sources
18. View tutor listings for each course
19. View profiles of professors
20. View professor reviews and ratings from RateMyProfessor on their profile
21. Comment on courses/professors, so future students enrolled in that class/with that professor can see a review
22. View previous course exams and lecture notes if publicly available
23. Allow students to see required materials for the course.

As an admin, I would like to:

1. Update course pages manually
2. Edit stored information about tutors and other resources
3. Send reset password emails in case of forgotten passwords