Team 10 Iteration 3 Report

April 12, 2024

Andre Arcaina
Joseph Leung
Tristan Cheng
Felipe Quiroga
Jack Heintz

1. Project topic MEM: Recreation Club Membership

The context of this project is a small amateur club (think salsa dancing or something like that) with a couple of dozen members. Club meets once a week for practice, and they are coached by an amateur coach. Members are free to show up (or not) every time. When they show up, they should pay for each practice, say, $10, but they may also pay up to one month in advance; some discount may be given if so desired. Once a month, the treasurer pays the rent for the hall in which the club meets, and the coach should be paid monthly or biweekly, but only for the practices that she has attended (she has a full-time job which sometimes requires her to be unavailable for the practice). The app should keep track of finances for the club, and for each member, as they sometimes show up for practice but manage to sneak out without paying. It may also send reminders to members about forthcoming practices, both regular and ad hoc ones, and threaten those who are a bit too casual with their payments.

Revised Product Backlog

Requirement					
Story	**Initial Priority**	**Task**	**Estimate (Story Points)**	**Actual Time (Story Points)**	**Status**
Track Club Finances					
Treasurer needs to be able to see the club's current profit	MED	Create the income statement and add the revenue portion (members payments, any other income that may come)	1	4	Finished
		Add the expense portion to the income statement (coach's payments, hall expenses, any other expenses)	1	4	Finished
		Create a list for the year, logging each month's profit to compare changes in revenue /expenses in order to find ways to maximize profit	1	4	Finished

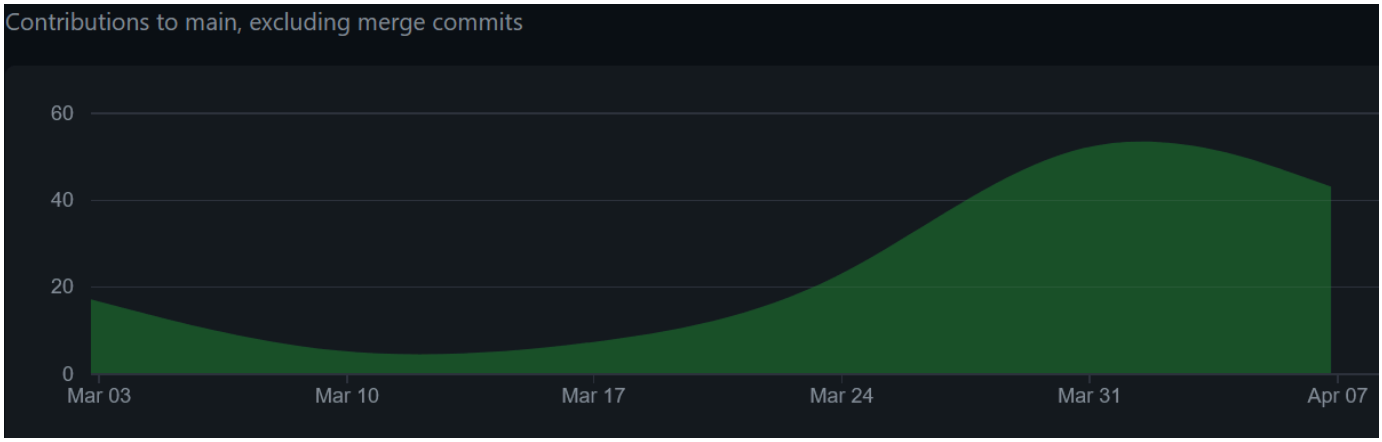
Treasurer needs a log of any unpaid debt from previous months (i.e. any months the rent for the hall was not paid, or not fully paid, and unpaid coach expenses)	HIGH	Create a list to log the total amount of unpaid coach expenses and hall expenses for each month, prioritizing the urgency for payment	2	5	Finished

Log and sort members					
Keep a log of the members	MED	Create a list of the members who have attended the class, and those who will be attending the class Each member will have their name, email, whether they paid or not, membership approval, prepaid fees	3	1	Finished
Sort the members by their frequency of attendance in the club. Provide a discount based on the amount of practices attended.	LOW	Using the list of members, organize the members by the amount of times they have attended practice at the club	2	1	Finished
Sort the members by whether they have paid or not. Provide a discount or charge based on amount of unpaid vs. paid classes	HIGH	Using the list of members, organize the members by the amount of times they paid/not paid	1	1	Finished

User interaction with application					
The coach should be able to communicate to the members about future practices	HIGH	Create a user interface where a message can be sent out to all or some members, with details of future practices or changes to any practices. A reminder message can also be sent out for a member who has skipped a payment	5	3	Finished
The treasurer should be able to manage the coach list and schedule. The coach should be able to manage the members list (i.e. adding or removing)	MED	Create user interfaces for different characters to modify the list of participants. And the corresponding character will receive notification upon changes.	5	2	Finished
Members need to be able to schedule and pay for the practice within the app.	MED	Create a user interface of payment methods or link to a third party payment method. Treasurer and coach should receive notification of the scheduled practice and received payment with user id, class amount and relative coach	5	5	Finished
New members should be able to create an account	HIGH	Create an interface that allows new members to register an account to partake in club activities	3	1	Finished

Members of the club should be able to log into the application	HIGH	Create an interface that allows the members to log into the application and be directed to a UI that corresponds to their role in the club	3	4	Finished

Team Velocity Diagram



Data found

This graph was obtained from the GitHub repository, which shows the number of commits over time. Follow the link to see a detailed history of the repository.

Test Report

User Authentication Tests

Test: Correct information on authentication page
Initial Conditions: Users must be able to login into the system with correct credentials and be found on the database that has all the user information.
Test Inputs: Input member username and password, treasurer username and password, coach username and password.
Expected Result: Given the correct input, expect a different screen for each given role.
Actual Result: Given the correct input, the member, treasurer, and coach can see their own screen.
Passed: Yes.

Test: Incorrect information on authentication page.
Initial Conditions: The user must be able to login into the system with correct credentials, but since the test is based on if the credentials are wrong, there are no initial conditions for this test.
Test Inputs: Input incorrect username and password.
Expected Result: Given the incorrect input, expect an error on the same screen that indicates that there is no found user in the database.
Actual Result: Given the incorrect input, a notification pops up saying that there was no user found.
Passed: Yes.

Finance Tracking Tests

Test: If the finances update properly on the database.
Initial Conditions: The user must have access to the financial section.
Test Inputs: Data inputted into the financial tracker.
Expected Result: Financial data should be saved into the database, user can access the inputted data displayed in proper formatting.
Actual Result: Couldn’t implement.
Passed: N/A.

Test: If the discount feature works as intended for the members.
Initial Conditions: The system has registered users. Users have passed conditions required to receive a discount (membership loyalty, payment streak etc.).
Test Inputs: Must have valid users with more than 1 previous payment.
Expected Result: The user’s payment is discounted automatically after a certain number of payments.
Actual Result: Couldn’t implement.
Passed: N/A.

Test: If the charging feature works as intended for the members.
Initial Conditions: Have members registered in the system. Users saved credit card details for payment.
Test Inputs: User books and pays for classes in advance with their saved credit card.
Expected Result: Class is scheduled in the system for the user, payment is received.
Actual Result: Members are able to book and pay for classes within their dashboard.
Passed: Yes.

Test: Functioning of "review outstanding debts" button.
Initial Conditions: User hasn't paid for their classes before the system was implemented.
Test Inputs: User tries to sign up for another class while they have outstanding debt.
Expected Result: The system doesn't allow users to sign up for class, prompt showing outstanding debt.
Actual Result: Couldn't implement.
Passed: N/A.

Membership Logging Tests

Test: Functioning of "sort the members by attendance" button.
Initial Conditions: Have existing registered users in the database.
Test Inputs: Click on sort button by attendance.
Expected Result: Display users sorted by attendance.
Actual Result: Users can be sorted by Date of Birth, Attendance, Payment Status, and Date Joined.
Passed: Yes.

Test: Functioning of "sort the members by unpaid/paid classes" button.
Initial Conditions: Have existing registered users in the database.
Test Inputs: Click on sort button by unpaid or paid classes.
Expected Result: Display users sorted by unpaid or paid classes.
Actual Result: Display users sorted by Payment Status.
Passed: Yes.

Message Platform Tests

Test: Check if members can send messages and receive announcements/notifications.
Initial Conditions: Member has an active valid account.
Test Inputs: Click on messaging platform and send a message to the coach.
Expected Result: Message received by coach account, both users can view message history.
Actual Result: Message received by coach account, everybody can view history.

Passed: Yes.

Test: Check if the treasurer can send messages and receive announcements/notifications.
Initial Conditions: Treasurer account can access messaging platform.
Test Inputs: Click on messaging platform and send a message.
Expected Result: Message sent to all users.
Actual Result: All users can see messages in the chat, as well as an announcements board.
Passed: Yes.

Test: Check if the coach can send messages/announcements/notifications.
Initial Conditions: Coach has a valid account that can access messaging platforms.
Test Inputs: Click on messaging platform and send a message.
Expected Result: Message sent to intended member/all users.
Actual Result: Message received by members and the treasurer, everybody can view history.
Passed: Yes.

Test: Check if the coach can see reminder messages (if members haven't paid yet).
Initial Conditions: Coach has a valid account.
Test Inputs: Click on individual members to check outstanding balance.
Expected Result: Coach can see if a member has a debt or if they have paid for the class already.
Actual Result: Couldn't implement.
Passed: N/A.

Database Tracking Tests

Test: Check if members are being added to the database (coach management).
Initial Conditions: Existing members signed up and saved in the database.
Test Inputs: Click on view members.
Expected Result: Database shows all registered members.
Actual Result: Shows a table of all registered members to the coach and treasurer.
Passed: Yes.

Test: Check if members are being removed from the database (coach management).
Initial Conditions: Coach/admin has a valid account with management privileges.
Test Inputs: Coach/admin clicks on remove member.
Expected Result: Member is removed from database.
Actual Result: Members can be removed from the database as a coach.
Passed: Yes.