Profimon

Requirements and Specification Document
11/07/2016, version 1

Project Abstract

Profimon is a game that allows you to battle the professors at SFU and collect various badges through these battles. You play as a student who takes a series of courses and battle professors in order to complete your degree. Each user will complete a profile page at the beginning of their journey. Then, in each round, a user can select a course to take and battle the professor in order to obtain a grade for that course. The user has a variety of skills in their arsenal to choose from during these battles. These skills may include studying, reading the textbook, and even cheating on an exam. Some skills expend more stamina than others, forcing the user to mix safe and risky strategies in their quest to obtain a degree. There will also be skills unique to each faculty. A Profimon can fight back with their own set of skills, which can range from asking the student a question during lecture to giving a final exam. A student can collect badges on their way to graduation by earning high grades from a specific group of professors based on their faculty.

Competitive Analysis

Profimon can be most closely compared to the Pokemon main series' gameplay. The battles in Profimon are like Pokemon's because of their turn-based nature where the player chooses which skills to use, and the player's stats determine both the effectiveness of their skills and the professors' moves. This is similar to Pokemon's battle system where one of four moves is chosen each turn. There will also be a limit on how many times each skill can be performed in a single class/battle, similar to the number of Power Points required for each move in Pokemon. The player will be able to increase their stats/power and acquire new skills as they make progress in the game. Like the Pokemon games, the player engages in battles in order to make progress through the game, earning badges along the way and eventually completing the game.

However, the quest structure is different from that of Pokemon. In contrast to Pokemon, the player fights alone as their student avatar. More freedom is granted over which class to take (which resembles the freedom of which job to accept next in the Mystery Dungeon series), and a variety of paths will lead to a winning state of the game. There are also penalties for those who fail courses (i.e. lose the battles), which draws parallels to other roguelike games such as The Binding of Isaac. However, unlike The Binding of Isaac, where players have to start all over upon failing a quest, gameplay is allowed to continue upon failing a course in Profimon (up to a certain number of courses), allowing them to either try again or choose a different course. Players of Profimon will also have the option to retake a successfully completed course (a limited number of times) to improve their overall grade, resulting in greater freedom.

Though users can continue playing the game after having failed a course, having a higher cumulative GPA in Profimon allows the user to upgrade their skills and increase the maximum amount of stamina they can have. This will increase the chances of the player succeeding in later rounds. The ability to upgrade skills is derived from role-playing games such as Fable, where you can spend your points on power-ups that let you do more damage, have more health, etc. The presence of the “stamina” bar also mimics the “mana bar” mechanic commonly seen in RPGs, where a pool of points is spent to use special skills. In our case, most actions will cost stamina. Combining the basic elements of Pokemon’s gameplay with a variety of features from roguelikes and other role-playing games makes Profimon a unique experience specifically tailored towards SFU students.

Target Audience (Customers)

Our target audience consists of SFU students and potential SFU students. They generally range from ages 18 to 30 and are located in the Lower Mainland. Our targets may be of any gender and nationality. These students are looking for a casual gaming experience and enjoy a variety of entertainment. They are somewhat familiar with battle-based games such as Pokemon or Digimon, and most of them will be familiar with the different faculties and course offerings within SFU.

Value to Our Users

Our web app will be a game that provides entertainment value to users. It will have additional appeal to those who attend SFU and are familiar with the courses and instructors, as they will recognize them in the game. The ability to battle profs they recognize may act as a stress reliever for some students. The game will also help students learn more about the different course offerings available at SFU.

User Stories

The following table shows the user stories implemented in iteration 1:

User Story	Sign Up
Description	As a User, I want to sign up to the Profimon webapp by providing a username, email, and password so that I can play Profimon.
Actors	Prospective Users (and Prospective Admins)
Triggers/ Preconditions	Trigger: The user clicks the "Sign Up" button on the login page. Precondition: The user does not have an account yet.
Actions/ Postconditions	Actions: Check that the user does not exist already, the email is unique, and the password is confirmed. Postconditions: If all fields are correctly entered, the sign up is successful (go to the user's profile page). If some fields are not entered correctly, display an error.
Acceptance tests	Sign up is successful when all fields are entered correctly: - Username must be between 6 and 20 characters long (validation). - Password must be between 6 and 20 characters long and contain one letter and number (validation). - Password is confirmed (typed a second time) and must match the previous field (validation). - Email provided is a valid email address (validation). Errors are produced when any field is left empty or violates restrictions. Assertions: When any field constraint is violated, the model tests will be run against - Assert_not user.valid? - Assert for redirection to user_url
Iteration	1

User Story	Login
Description	As a User, I want to login to Profimon using a valid username and password, so that I can play Profimon.
Actors	Users (Regular Users and Admin Users)
Triggers/ Preconditions	Trigger: The user goes to the index page. Precondition: The user is on the index page and has an account.
Actions/ Postconditions	Actions: Check to make sure the username and password matches an entry in the database. Postconditions: If login is successful, redirect to the user's profile page. If login is unsuccessful, display an error.
Acceptance tests	Sign in using a valid username and a corresponding password registered to an account - Username must be entered correctly and must exist in the database.

	- Password must be entered correctly and must match an existing user in the database. - Session ID matches user ID. An error is produced when any field is left empty or does not match any database entry. Assertions: - Assert_equal user.id, session:user_id - Assert redirect to user_url - Assert that one of the objects on the profile page says profile
Iteration	1

User Story	Create Character
Description	As a User, I want to create a character profile so I can start tracking my gaming progress.
Actors	Users (Regular Users and Admin Users)
Triggers/ Preconditions	Triggers: The user clicks the “Create Profile” button from the user page. Precondition: The user has an account and is currently logged into that account.
Actions/ Postconditions	Actions: The user is redirected to the “Create Profile” page. Store the character profile name and faculty field into the database. Postcondition: A new character profile has been created for the user. The user is redirected to the character profile page displaying the character details.
Acceptance tests	The profile fields are entered correctly. - Profile name must not be left empty (pname validate presence true) - Faculty must be chosen from dropdown list (validate presence true) Assertions: - Assert post to profile success - Assert difference for profile count (increased by 1) - Assert redirection to profile_url
Iteration	1

User Story	View Character Profiles
Description	As a User, I want to view character profiles that belong to me.
Actors	Users (Regular Users and Admin Users)
Triggers/ Preconditions	Triggers: The user selects and clicks a character profile from their list of character profiles on the user’s view. Precondition: The user has an account, is currently logged into that account, and has at least one character profile created.
Actions/ Postconditions	Actions: The user is redirected to the character profile page that they selected. Once in the character profile view, the information for that character is displayed. Postcondition: The user is on a character profile page that belongs to them.

	Information about their character is displayed on the view.
Acceptance tests	Profile pages belonging to the user can be reached by the user - Redirected to profile page belong to user - profile user_id = user.id Assertions: - Assert equal profile.user_id = session:user_id - Assert redirection to profile_url
Iteration	1

User Story	Edit Character Profile
Description	As a User, I want to edit my character profiles to change my character profile name and/or faculty.
Actors	Users (Regular Users and Admin Users)
Triggers/ Preconditions	Triggers: The user clicks the “Edit” link on the character profile view. Precondition: The user has an account and is currently logged into that account. The user also has at least one profile created and is currently on the character profile view.
Actions/ Postconditions	Actions: Redirect the user to the edit page for that character. Display the character’s profile name in a text field and the faculty in a drop down menu to allow the user to make changes. Postcondition: The user successfully edits the character’s profile details and the changes are saved. The user is then redirected back to the character profile view.
Acceptance tests	The profile pages belonging to the user can be reached by the user. - Profile.user_id = user.id (make sure user owns the character profile) - Edit (patch) success - Update profile_url, params {pname: “string”, faculty: “string”} - Redirected back to character profile page Assertions: - Assert update success - Assert redirection to profile_url
Iteration	1

User Story	View All Users and Characters
Description	As an Admin, I want to view all the existing users and their characters in the database
Actors	Admin Users
Triggers/ Preconditions	Trigger: The admin user selects “Admin Settings” from the navigation bar. Precondition: The admin user has a valid admin account and is currently logged into an admin account.
Actions/ Postconditions	Actions: The session checks that the user is a valid admin. The admin is redirected to an admin view that displays a table containing all users and characters belonging to each user.

	Postcondition: The admin is on a view where all users and characters are displayed.
Acceptance tests	The admin view displaying all users and characters can be reached by the admin. - Admin Boolean for user must be true - Redirected to admin view - Users' name and characters' name are displayed in admin view Assertions: - Assert is admin (true) - Assert redirection to admin view
Iteration	1

In future iterations, the following user stories will be implemented:

- As a User, I can edit my username, change my email, and/or change my password.
- As a User, I can select a class to enroll a character in in order to earn a GPA and possibly badges in that class.
- As a User, I can select a profile picture for my character.
- As a User, I can use a skill (i.e. study, cheat, etc.) when battling a Profimon in a class.
- As a User, I can view my progress/result in a class.
- As an Admin, I can view all the Profimons.
- As an Admin, I can edit a Profimon's stats.
- As an Admin, I can create a Profimon.
- As an Admin, I can delete a Profimon.
- As an Admin, I can view a list of available skills.
- As an Admin, I can edit the stats of a skill.
- As an Admin, I can add a skill to the list of available skills.
- As an Admin, I can delete a skill.

APIs Used

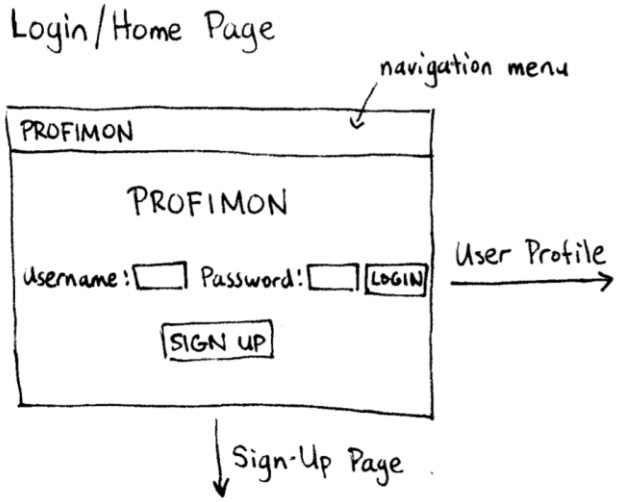
The application will make use of the SFU Course Outlines REST API to retrieve data about the course offerings. To determine the strength of each Profimon, we will use data from the SFU Gatekeeper Course Analysis spreadsheet to calculate a difficulty score for each course. We will then put the difficulty scores onto a Google Sheets spreadsheet and turn it into an API, which our application can read (only the difficulty scores that we calculate will be included in the API, not any of the actual data in the Gatekeeper Course Analysis spreadsheet).

Amount of Work Required

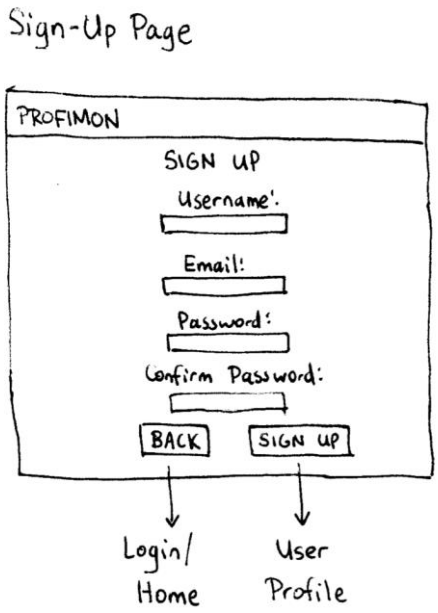
We believe the amount of work required in this proposal is adequate for five group members due to the complexity of implementing certain features in the game. We have identified three epics for the application, and implementing the epic of battling Profimons requires us to not

only build the features and skills available to the user, but also construct the Profimons and determine their different skills and strengths. Building the Profimons will require a similar amount of work as implementing an additional epic. Furthermore, we will need to build a library of skills in order to have skills that are specific to certain subjects/faculties. Due to the number of different subjects/faculties at SFU, we expect this library to be quite large, likely consisting of over 100 different possible skills.

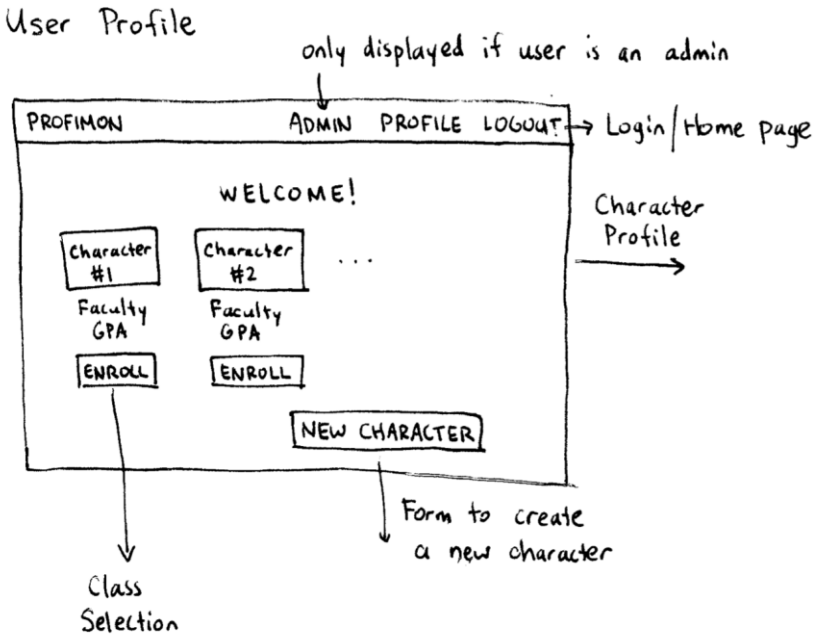
UI Mockups



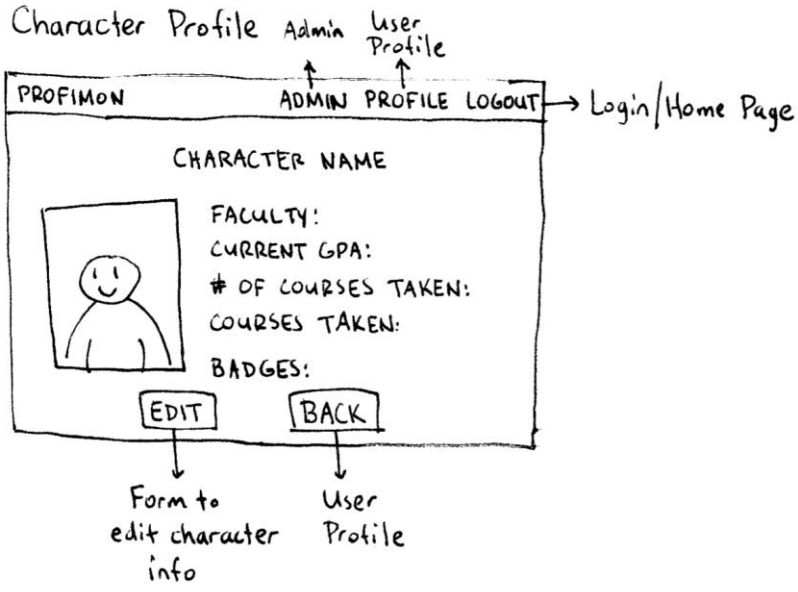
On the login page (or the home page), the user needs to be able to enter in a username and a password, so there will be two text fields to allow the user to do this. Pressing the login button directs the user to the user profile page, while pressing the sign-up link at the bottom directs the user to the sign-up page.



On the sign-up page, the user needs to be able to specify a username, an email, and a password, so there will be text fields to allow the user to input this information.

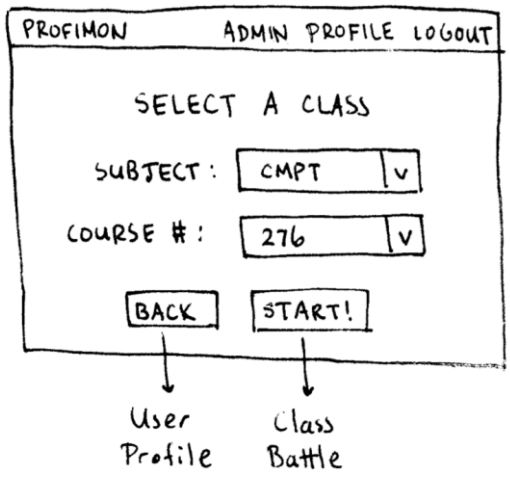


On the User Profile page, we will display all the user's characters, as well as some information about the characters (such as the character's faculty and GPA). The user needs to be able to enroll the character in a class, so there will be an Enroll button under each character (will be implemented in Iteration 2).



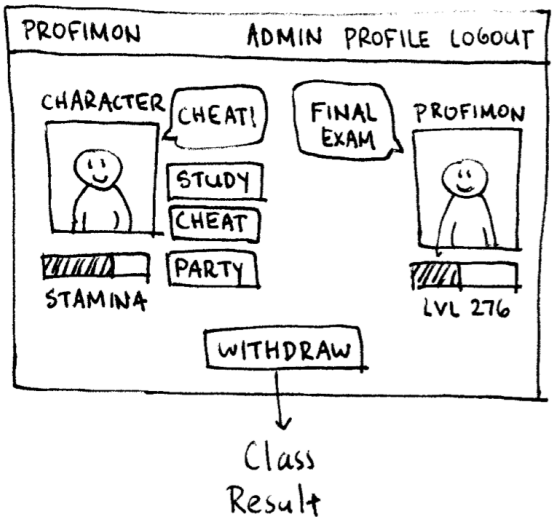
The Character Profile page displays information about a particular character. Users may want to edit information about one of their characters, so an Edit link is provided from this page.

Class Selection

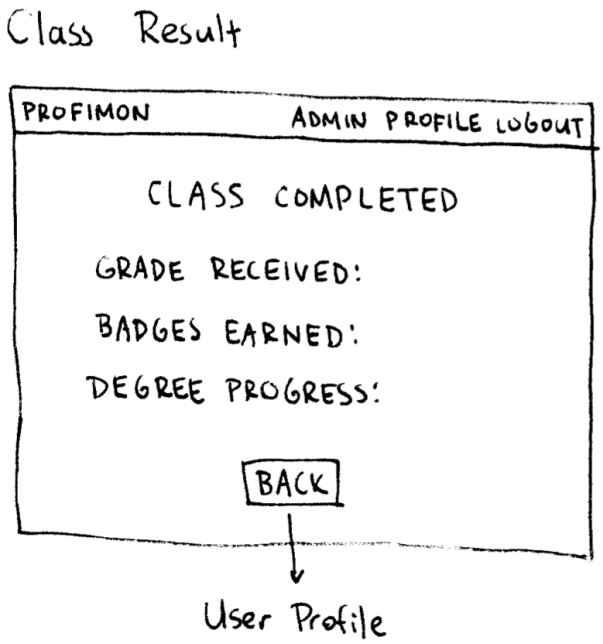


The Class Selection page must allow the user to select a class according to a subject and course number, so two select elements will be displayed on this page.

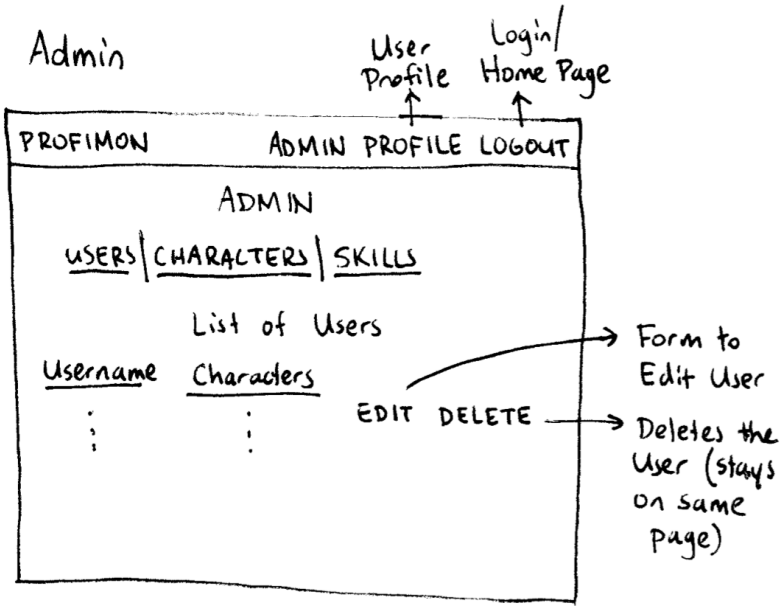
Class Battle



On the Class Battle page, the user selects a skill (i.e. study, cheat, party, etc.) to use during each turn. The Profimon will counter back with his/her own skill. The user's skill choices will be represented by buttons on the screen. Some of these buttons may be disabled if the character runs low on stamina. The battle ends when the user finishes the class (the user's progress in the class is indicated by a progress bar under the Profimon) or if the user presses Withdraw.



Once the user has completed the class, we need to display the user's result in the class as well as the user's overall progress in the game.



Clicking on the admin link in the navigation bar leads the user to the Admin page. Here, the administrator can view detailed information about every user, every character, and every skill currently stored in the game. Administrators will also be able to edit or delete entries, so Edit and Delete links will be provided beside each entry (will be implemented in Iteration 2).