Prodigy (working title)

Requirements

User tasks:

1. Create the list of subjects.
2. Create a number of multiple choice exercises (group of questions) for each subject.
3. Assign introductory and summary text for each exercise.
4. Create questions for each exercise, with one or more answers to each question.
5. Assign a feedback text to various possible answers for a question.
6. Create and add textual theoretical background to be accessible for reference by the student at each subject.
7. Access student specific progress information:
   1. Average score in each completed level
   2. Number of questions completed successfully
   3. Number of questions not completed
   4. Average number of mistakes per question (also by subject)
   5. Time spent in subjects
8. Support the following two ways of submitting answers to a question:
   1. By building a complete answer from the choices in a ordered fashion, starting from the first missing part and until the answer is complete.
   2. By selecting the missing place in the answer to which the user wishes to submit a choice to.
9. Control the progression of students through the syllabus by setting a minimal passing criterion in each level.
10. Allow student to undo a submission in certain scenarios.
11. Consider support for multiplayer gaming.
12. Mark words that appear in the question headline as Bold, Italic, and Underscored.
13. Provide multiple answers to the same question.
14. Provide a template for an answer, from which words are missing.
15. Provide answer that are made of multiple parts, where the order of the parts doesn’t matter.
16. Each level playable at different difficulty levels.
17. Define multiple levels of goals in each level:
    1. Example: By the amount of questions that must be completed / intelligence points gained.
18. Allow use of special game clues.

The following use cases relate to the creation of content and may focus on data addition by files instead of online.

UC1: Create syllabus and subjects

Actor: Teacher

A user selects to create a new syllabus. He enters a name for the syllabus and a short description. Repeats the following for each subject he adds to the syllabus:

1. Enter a name and a short description of the subject.

After all subjects have been defined, the user creates the exercises, or levels that contain questions and determine the progression of a student. He repeats the following for every level created:

1. Enter a name and instructions for the level.
2. Enter an optional preliminary instructive text, a short theoretical background for the level. This text is accessible by the student user at all times during the level.
3. Enter and optional summary text, to appear only when level has ended.
4. Enter the questions of the exercise (described in UC2).

We recognize the fact that this process might be long and error prone, so of course saving every information submitted during is valuable. However, the user must activate, or publish this syllabus before it is available for student users. That operation is final.

UC2: Add Question to Exercise

Actor: Teacher

A user selects an exercise to add questions to (either existing or new).

1. He enters the textual content of the question.
2. In case of entering the content through a written text file, the user declares the answers section using the sentence “answers:”.
3. He enters one or more answers in the following format, each occupying a single line:

A|B|…|N

Where A,B..N are tokens that will be converted to choices for this question, delimited by the character ‘|’ consecutively.

1. In case of entering the content through a written text file, the user declares the dummy section using the sentence “dummies:”
2. He may add additional dummy choices in the following format:

A|Ad

Where the left side token is choice that appears in one or more answers, and the right side token is its associated different dummy token, delimited by the character ‘|’.

1. To supply feedback on possible choice permutations, user must follow these steps:
   1. First declare the feedback section using the sentence “feedback:”.
   2. For each unique permutation, start a new line and repeat:
      1. Enter the permutation in the following format, following the token “choices:”

A|B|…|N

Where A,B..N are tokens that will be converted to choices for this question, delimited by the character ‘|’ consecutively.

* + 1. In the next line, starting with the token “feedback:”, enter the textual feedback.

UC3: Engage Questions in Game

Actor: Student

User selects a question to answer. He presses the engage button. The questions choices appear to the user. The buttons in the choices are expanded and populated with text. Up to 8 unique choices can appear at once. If the question holds more than the maximum, the choices update after the first few have been chosen.

At all times, the on screen choices are taken from the current location in the sequence and onwards, with regards to the answers corresponding the selected sequence. For example, if the question contains one answer <A, B, C, D, E>, the choices shown at the beginning (location at index zero, matching letter A) are {A,B,C,D,E}. At the second location, index 1, the choices are {B,C,D,E} and so on.

Select Choice

The basic logic of selecting a choice.

Repeat process until one of the following happens:

1. A choice sequence has been completed and matches an answer.
2. User has falsely selected choices more than the allowed number.

User clicks on a choice.

If the choice is correct

Append choice to on screen sequence

Increment combo count by 1

Increment number of correct choices by 1

Decrease 1 hit point from question

Increase user hit points by 1

Else

Append the token to the choice sequence but mark as a mistake

Reset user combo to default value

The question does not lose a hit point

If the number of hit points left in the question is lower than the amount required for achieving the minimal goal

Disengage question and lock from user

If the number of available choices is greater than 2

Show

Complete description of a single player game

A player signs in the system. He clicks the button that says ‘start’. Because the player is new, the first section of the game is shown, together with the number of levels. The latest accessible level is presented (which in this case is the first), and its details are shown:

* The introduction
* The instructions
* The difficulty level
* The goal levels that the level can be accomplished at: A goal level sets the minimal amount of intelligence points a user must accomplish in order to reach this level’s goals. For every goal that is reached, the user receives an amount of bonus points and some label, such as “GrammarKing” or perhaps something more specific to the section he is at, such as “The Count of Counting” in the “countables” section. (these may also be unlocked when playing in multiplayer mode)

The player clicks the “Play” button, which starts the game. The level data is loaded at the server and client, and once complete, the game starts at the first question.

The player now sees on screen the following:

* The headline of the question
* The number of intelligence points accumulated in the level so far (starting at zero)
* The

Client Server Communication Protocol

Single Player

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initiator | Key | Value | Destination Process | Return | Notes |
| Client | Play Game | Level Number | Load data, send data to client, start game when client signals that data is loaded and ready |  | Request to start a new game |
| **Client** | **Move To Question Number** | **Question Number** | **Update player location** | **Notify all (multiplayer)** |  |
| Client | Engage Question |  | Validate that question is not previously engaged. Assign Question as occupied by user. | Notify all  (multiplayer) |  |
| Client | Select Choice | Choice Token | Validate selected Token, place token at question answer, at the current user location |  |  |
| Client | Disengage Question |  | Validate that user is able to disengage. | Notify all |  |
| Client | Use Armor |  | Validate action |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Multi Player

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initiator | Key | Value | Destination Process | Return | Notes |
| Client | Join Game | Level Number | Load data, send data to client, start game when client signals that data is loaded and ready |  | Request to start a new game |
| **Client** | **Move To Question Number** | **Question Number** | **Update player location** | **Notify all (multiplayer)** |  |
| Client | Engage Question |  | Validate that question is not previously engaged. Assign Question as occupied by user. | Notify all  (multiplayer) |  |
| Client | Select Choice | Choice Token | Validate selected Token, place token at question answer, at the current user location |  |  |
| Client | Disengage Question |  | Validate that user is able to disengage. | Notify all |  |
| Client | Use Armor |  | Validate action |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |