Prodigy (working title)

Draft 2.0

Prodigy is an online interactive language practice game system, centered on the concept of practice through contextual stories.

Prodigy is a game. Requirements relating to the game derive from us.

Motivation

Build an online system where teachers write and publish various exercises, and students practice them. Engulf these exercises in an enjoyable and rewarding environment for student to play with. Provide an efficient mechanism for teachers to write exercises with great freedom and ease.

Design Requirements

1. Allow player to submit a solution in the following manners:
    a. To an entire exercise
    b. To a paragraph in an exercise
    c. To a part in a paragraph.

    Submitting a solution means the player receives feedback and is not able to modify it later.
2. Allow player to undo a submission in the following manners:
    a. To an entire exercise
    b. To a paragraph in an exercise
    c. To a part in a paragraph.
3. Provide clues and assistance in an exercise. A clue may refer to one or more parts of answer to a paragraph (1), one or more paragraphs in an exercise (2).
    a. Reveal the size interval (in case of multiple solutions) of the part (number of characters) if relevant (concerning exercise formats). (1).
    b. Eliminate one or more parts from the options. (1) and (2).
    c. Reveal the number of missing parts if not indicated. (2).
    d. Place a solution and query the user for its validity.
4. Provide a solution to an exercise in the following ways:
    a. Free text
    b. Selection from available options.
5. Indicate the score a user received in an exercise based of the grading scheme of the writer.
6. Provide different levels of difficulty a user may play at regarding the gameplay of exercises and not the actual content.
7. Consider future implementations of multiplayer online gaming.
8. Support online and offline single player mode.
9. Online single player mode should be considered in the architecture to allow future versions to be playable only online, such as is the case with a HTML version.

Student: Manage the Story Book

A Student must be registered in order to play stories. Once registered, he is supplied with an account, which includes a username, password, and a bookmarking feature to retain references to stories. Stories are only playable through a **Story Book**, which is the bookmarking feature mentioned above, and must be added to one's own for play. Stories are located in the global **Library**. To search for stories, we supply a search mechanism, where stories may be fetched by any textual content that they hold (PI). Stories are found, added to the Story Book, and then played through it.

The following actions are possible from a Story Book:

1. Play a story.
2. Remove a Story from the storybook(PI).
3. Review a finished Story (PI).
4. Review a Story's Writer (PI).

Playing a Story from a Story Book (1) includes:

- Resume a Story from last checkpoint (or possibly from any saved checkpoint)
- Replay a Story from the start
- Replay a previously accomplished Chapter from a story

We thoroughly explain the actual gameplay in the following section.

Use Case: Play Story

Play Story: User selects a story from his **Story Book**. The Story Book contains various stories that were added from **Prodigy Library**. Stories may be ones that have yet to be played, or ones that are currently in progress. If it is one he has previously started and saved his progress, the following scenarios are possible:

1. Resume from last checkpoint. Checkpoints are scattered throughout the paragraphs of a story, and every one saves the exact current game state of the user. The beginning of every chapter (before the first paragraph) is also a checkpoint which is automatically saved. This includes:
   - Health (mistakes left to make)
   - Points earned until checkpoint in current chapter.
   - State of magical items (if used or not).
2. Replay a previously played chapter. Replaying a chapter is made available once one has been completed, and it is played separately from the current story in progress. We shall consider making this option available only once a story has been completed.

Playing a Story

Player receives first paragraph in story (question) and is presented with optional solutions to the question. If the solution is divided into separate parts, the user is required to go through all of them. In other words:

While (EndGame == false)

For each paragraph in story

i. For each part in solution of paragraph
    1. Submit an answer
ii. Submit Paragraph // or automatically submitted when last part inserted
iii. Reward player for each correct part / decrease piece from health with each mistake.

End Game

- If health is depleted
- If all paragraph finished
- If user exited game

A user may receive special items to use in game, which are basically clues:

- Clue 1 – reveal number of characters in one or more parts
- Clue 2 – remove one or more parts that evaluate false in one or more selected parts.
- Clue 3 – system selects one or more random parts as a solution and asks user if this is the correct solution or not (turns paragraph into Yes/No question).
- Clue 4 – allow undo operation of one or more parts.

A teacher requires a platform where he may write playable exercises for students. Teacher also require a tracking mechanism which supplies a means to follow-up on students progression and interact with them. **Note: the current business model dictates that this feature shall only be available to system administrator users, not Teachers.**

Teacher: Manage the Writer's Desk

A teacher must also register to the system before use. Once registered, he is granted a profile which holds his username, password, and access to the **Writer's Desk**. The latter provides means for managing all exercises created, which includes the following:

1. Manage The Story Books. A writer has multiple story books, one book for each Language that the system supports. Each story book holds all stories (pending publication, published, and unfinished). The same functionality that a **student's** story book supplies is available to a writer (limitations may apply). This includes the *creation* and *modification* of all Stories a Teacher holds in his Writer's Desk. A writer may request to cancel publication of a story if it is in a pending state. If a writer wants to remove a published story he must file a request through the system. Deletion of stories is available only if a story is in neither of the two latter cases.
2. Manage individual paragraphs. To support efficient capabilities for presenting a paragraph in more than one exercise. This includes:
    a. Searching for a paragraph by any of its unique features.
    b. Inserting a copy or a reference (requires unique ID) of an existing paragraph into multiple exercises.
    c. Removing all references of to a paragraph.
    d. Modifying a paragraph (includes modification of referenced paragraphs).
3. Publish a Story. When a story is considered complete by its Writer, it is must be authorized by an Admin for it to be available for students. Much like in real life, it must be checked for appropriate content and censored if required.

UC: Create Story

A teacher determines a subject of practice relating to some lesson taught in class. He continues to fill the following details:

- **Name** – the name of the story. Required.
- **Instructions** – the guidance required to fill answer all following questions. Required.
- **Paragraphs** – ordered appearance of individual paragraphs in the story the user is required to complete/answer in order. For each paragraph:
    a. Enter text and mark missing part using <NULL> and </NULL> tags indicating start and end of part, respectively.

b. Now a solution is created with X parts that were tagged.
c. If a teacher requires there to be more than one solution, creates another solution template, holding X missing parts, and enters a sequence of parts that constitute another solution. A part may be empty but the order of each solution must be different. Id the teacher requests to add “dummy” solutions, he repeats the previous process, but indicating that the solution is false (by marking a field for example).

Teacher submits paragraph for validation. If the total amount of text in the paragraph, once complete with an answer, exceeds some threshold, notify. If the total amount of unique parts in the solution exceeds some threshold, notify. Otherwise, an OK is returned, and the teacher proceeds to the next one.

- **Grade** – Once all paragraphs have been submitted, a default equal weight is assigned to all paragraphs. A user may define different weights for individual paragraphs (must be system validated, max =100%), which will determine the “health” a player has in game.
- **Grade Threshold** – usually accompanying a Grade is a minimal passing threshold. This is used to determine the minimal level of perception a student must have in order to allow some sort of progression.

Top level design

Database – holds Story items as XML documents. User data as RDB/FF system.

Login Server – manages access to feature in a system based on user access rights. Manages all access to data and retains state of games user play.

Desktop / HTML game engine component – holds logical rules and visual user interface.

Desktop:

- Game Logic Component – holds state of game and determines result of user triggered moves. A Story is loaded in an instance of this component and played through it. This plays the content of a Story in XML format.
- Server Communication – interacts with Game logic to send progress messages to cloud. Used as checkpoint mechanism and in case some game logic that resides in the cloud must be used to determine game states.
- GUI – represents a story in its current state. Binds to Game engine story and presents alterations in game state.

HTML:

- JavaScript library as the game logical engine.
- HTML5 user interface.