Department of Information Systems

Ben Gurion University of the Negev

PRODIGY

A Computer Game Enhanced E-Learning System

BY

GUY MANZUROLA

ACADEMIC INSTRUCTOR **DR. MEIRAV TAIEB-MAIMON**

Submitted **August 12 2013**

ABSTRACT

In almost all areas of learning, knowledge is acquired through extensive and repetitive practice of the learned material. This is especially true when learning a second language.

A popular and appealing course of providing users with learning material is via digital devices, i.e. “e-learning”. This enables learners to study at a time and pace of their choosing.

However, most if not all of the existing e-learning solutions focus on the needs of educators to impart “selected knowledge” to their students, rather than fulfilling the needs of the learners (Pivec, 2007; McGinnis, Bustard, Black, & Charles, 2008; Jong, Shang, Lee, & Law, 2006).

This leaves the learners to find the necessary motivation to study the content themselves.

The purpose of this project is to develop an e-learning quiz system using design principles from computer games, in order to keep learners engaged and motivated through repetitive practice. This work will focus on English language structure learning content.

TABLE OF CONTENTS

1. LITERATURE REVIEW 6

.1.1 E-Learning: An Introduction 6

1.1.1. What is E-Learning? 6

1.1.2. Problems with Current E-Learning Tools 6

1.2. Enhancing Motivation Through Games 8

1.2.1. Motivation 8

1.2.2. Flow 9

1.2.3. Play 9

1.3. Computer Game Enhanced E-Learning Systems 10

1.3.1. Principles of Instructional Game Design 10

1.3.2. Heuristics for the Design and Evaluation of Instructional Computer Games 12

1.3.3. Designing Video Games for Foreign Language Learning 13

1.4. Evaluating Existing Solutions 14

1.5. Technologies 16

1.5.1. Bottom Tier 16

1.5.2. Middle Tier 17

1.5.3. Client Tier 19

1.5.4. Client and Middle Tier Communication Protocol 20

1.5.5. Development Process 20

1.5.6. Integrated Development Environments (IDE) 21

2. ANALYSIS AND DESIGN 22

2.1. Requirements: The Game Design Document 22

2.1.1. User Interface Mockup 23

2.2. Analysis 28

2.2.1. Gameplay Flow Chart 28

2.2.2. Use Cases 29

2.2.3. Domain model 33

2.3. Software Design 34

2.3.1. Architecture 34
2.3.2. The Database Schema 35
2.3.3. Server API (REST Resources) 37
2.3.4. The Client Application 40
3. RESULTS 44
4. GLOSSARY 46
5. REFERENCES 48

1. LITERATURE REVIEW

It is known that a brain enjoying itself functions more efficiently. In other words, if we enjoy learning, we learn better.

Based on the above, it appears that there is a long term need for a tool that would make the learning process engaging and motivating for the learner.

1.1. *E-Learning: An Introduction*

1.1.1. What is E-Learning?

In essence, **E-Learning is the transfer of instructional material through digital devices** such as mobile phones, tablet computers, laptop and desktop computers, and more .

E-Learning enables learners to study at times and places of their choosing. It may also provide people with the ability to study at their own pace (asynchronous study), and in the order they prefer. It is especially appealing now days, due to the high availability of the instruments required to participate in such activities (Clark & Mayer, 2011; McGinnis, Bustard, Black, & Charles, 2008).

Tools of this type can be used either to complement and support traditional teaching, or as a complete and independent solution that can replace physical teacher-student interaction all together. The latter is much less common, and where found, is mainly used as a training tool for specific skills (e.g. tutorial on a piece of software). The former however is used by numerous institutions, either to consolidate learning material, or to present students with practical exercises relevant to the learnt material.

1.1.2. Problems with Current E-Learning Tools

Though e-learning has great potential to open up a number of different opportunities for learners around the globe (Smith, 2008), most E-Learning systems do not yet have the impact that many believe is possible (McGinnis, Bustard, Black, & Charles, 2008). Without the proper motivation for students to engage in a learning experience, e-learning initiatives following this trend are likely to fail (Smith, 2008). The cause of the problem can be attributed to the following two factors.

Meeting Learners Needs

Current e-learning systems tend to focus on the management and delivery of content (McGinnis, Bustard, Black, & Charles, 2008; Pivec, 2007), which is one of the primary goals of an educator (to instruct) (Prensky, 2002). Thus a situation is created, where the importance of the student within the process is often overlooked; the focus is mainly on the needs of educators, instead of the needs of the learners who are actually using the system to learn.
A learner's needs may be presented by following three demands (McGinnis, Bustard, Black, & Charles, 2008; Pivec, 2007):

1. *Empowerment*. Learners expect to be in control of the learning experience. E-learning systems should thus promote self-directed learning.
2. *Social Identity*. The freedom a learner has in a self-directed learning environment should not come at the expense of a sense of belonging to a social group. Research indicates that such a sense improves motivation and effective learning.
3. *Authentic Learning Experience*. Learners are more engaged and motivated when they are participating in activities to which they may relate.

Lack of Compelling Content

Engaging learners long enough to see them through to the end of a course has become one of the most significant problems faced by e-learning developers. This can be attributed to a lack in one or more of three main issues as suggested by McGinnis et al (McGinnis, Bustard, Black, & Charles, 2008):

1. *Interaction*. Immediate feedback, clear short term goals and a sense of 'flow' through the content is needed for an effective E-Learning experience.
2. *Challenge*. Unchallenging material has been indicated by learners as not motivating, which results in many learners who are reluctant to repeat the experience.
3. *Context*. Students have stressed the importance of being able to appreciate the significance of their current progress in relation to the overall goal of the learning material.

Without compelling content, students are left to find the necessary motivation themselves.

1.2. *Enhancing Motivation Through Games*

It is known that a brain enjoying itself functions more efficiently. In other words, if we enjoy learning, we learn better.

In his paper, "The motivation of Gameplay", Mark Prensky (Prensky, 2002) refers to the relationship between fun and learning, and concludes that, based in part on the works of (Rose & Nicholl, 1998) and (Bisson & Luckner, 1996), fun in the learning process creates relaxation and motivation. He continues to define the benefits of the above consequences as follows: "Relaxation enables learners to take things in more easily; motivation enables them to put forth effort without resentment".

1.2.1. Motivation

Motivation has long been considered an important step in learning (Smith, 2008; Paras & Bizzocchi, 2005; Prensky, 2002). It has always been one of the teacher's primary roles to provide students with the motivation required to stick with the learning process to the end. However, motivating students has also been one of the biggest problems in education over the last half century or so (Malone T. W., 1980; Prensky, 2002).
Without compelling content, students are left to find the necessary motivation themselves. Excluding some rare cases in which students are actually connecting with the learnt material, Prensky (Prensky, 2002) states the following as possible student motives: "students' motives for learning are a mixture of intrinsic goals and extrinsic rewards, combined with psychological factors such as fear and the need to please".

Malone (Malone T. W., 1980) differentiates two kinds of motivation: intrinsic and extrinsic; "In general, an activity is said to be intrinsically motivated if there is no obvious reward associated with the activity. Conversely, an activity is said to be extrinsically motivated if engaging in the activity leads to some external reward like food, money, or social reinforcement. Generally, an external reward is dispensed by a human or mechanical agent in a way that is not "naturally" a part of the rewarded activity".
Malone emphasizes that intrinsic motivation should not be viewed as the absence of rewards, but as "a need for competence and self-, as a search for an optimal amount of psychological incongruity, or the "experience of flow"". At last, Malone states that intrinsically motivated

students are likely to spend more time and effort learning and feel better about what they learn.

1.2.2. Flow

The literature commonly references the Flow Theory by Csikszentmihalyi (Csikszentmihalyi, 1990) within the concept of motivation and learning, and can be regarded as a method for understanding and implementing motivation (Paras & Bizzocchi, 2005). A state of flow can be described as an "Optimal experience, where we feel a sense of exhilaration, a deep sense of enjoyment that is long cherished, does not come through passive, receptive, relaxing times." (Csikszentmihalyi, 1990). Csikszentmihalyi describes that a state of flow in an activity can be achieved through having clear goals, achievable challenges and accurate feedback (McGinnis, Bustard, Black, & Charles, 2008). This is a psychological take that can provide a rational explanation as to why and how games may benefit a learner, since it is generally considered that most games put a person in a state of flow.

1.2.3. Play

Paras and Bizzochi (Paras & Bizzocchi, 2005) state that play, which is the state a person is at when playing games, produces a state a flow. They assert that **"**Just like learning environments shouldn't restrict the learner's ability to more freely construct knowledge, the game environment should not restrict the player's cognitive process, but rather allow the game player to freely make choices that help to reach an end goal**".**

According to Paras and Bizzocchi (Paras & Bizzocchi, 2005) "**Games foster play, which produces a state of flow, which increases motivation, which supports the learning process**".

1.3. Computer Game Enhanced E-Learning Systems

Based on the above, games are inherently motivating for players. We would like then to introduce the same elements that make games motivating, to a learning environment.

1.3.1. Principles of Instructional Game Design

What is it then that makes games so captivating? Malone (Malone T. W., 1980) defines three major categories:

Challenge Many players enjoy playing games because they provide a challenge. In single-player home games, where social or bragging rights motivations are less of an issue, it is a primary source of motivation: "When a person faces a challenge and then overcomes it, that person has learned something. It does not matter if that challenge is in a math textbook or in a computer game. Challenging games can be learning experiences. Players will learn from games, even if that learning is limited to the context of the game, such as how to navigate through the forest, survive a particularly hairy battle, or convince the duke that their intentions with his daughter are honorable." (Rouse, 2005)
Learners have indicated that unchallenging material fails to stimulate them, making the experience unattractive and eventually discouraging progression (McGinnis, Bustard, Black, & Charles, 2008).

Providing players (and learners) with a challenge seems to be the most appealing motivating aspect for playing (and thus learning). Specifically, providing a clear, reachable and appealing goal to a game (Malone T. W., 1980; Rouse, 2005; McGinnis, Bustard, Black, & Charles, 2008; Csikszentmihalyi, 1990).

Fantasy Malone defines a fantasy inducing environment as "one that evokes mental images of things not present to the senses or within the actual experience of the person involved", and states that "fantasies can make instructional environments more interesting and more educational" (Malone T. W., 1980).

Curiosity One of the main motivating forces that propel players through a game is the desire to explore new things (Rouse, 2005). Malone describes curiosity in the context of learning as "the motivation to learn, independent of any goal seeking or fantasy-fulfillment"

(Malone T. W., 1980). In general, the learning environment should be one where "the learner knows enough to have expectations about what will happen, but where these expectations are sometimes unmet.

We can distinguish between the following aspects of curiosity (Malone T. W., 1980):

1. Sensory curiosity – attracting player attention through changes in patterns of audio or visual effects.
2. Cognitive curiosity – the desire to improve one's knowledge structure.

In his book, "Game Design, Theory and Practice", Richard Rouse III (Rouse, 2005) lists a number of reasons players play games. We may augment the above three principles with the following three:

Socialization The origin of games is tied to a social experience, and that is something all game designers must remember. This communal component is central to their appeal: "For most people, the primary reason they play games is to have a social experience with their friends or family." (Rouse, 2005)

Bragging Rights "When players are victorious at a challenging game, they realize they can do something well, probably better than most people, which makes them feel better about themselves." (Rouse, 2005)

Interaction The one thing the art of games can do better than any other art form is providing an interactive experience. **Players play because they want to interact** (Rouse, 2005).

1.3.2. Heuristics for the Design and Evaluation of Instructional Computer Games

In her paper, “Heuristics and usability guidelines for the creation and evaluation of fun in video games” (Federoff, 2002), Melissa Federoff compiles a list of heuristics used to create fun in games.

The list is based mostly on available literature, and augmented with a few heuristics derived from her case study. We shall use some of the entries in the list to evaluate existing solutions, and to provide us with guidelines in creating and evaluating our own solution.

presents some of the key principles from that list.

Element	**Guideline**
Game Interface	A player should always be able to identify their score/status in the game
Game Interface	Use sound to provide meaningful feedback
Game Interface	Provide means for error prevention and recovery through the use of warning messages
Game Interface	Players should be able to save games in different states
Game Mechanics	Feedback should be given immediately to display user control
Game Play	There should be a clear overriding goal of the game presented early
Game Play	There should be variable difficulty level
Game Play	There should be multiple goals on each level
Game Play	Game play should be balanced so that there is no definite way to win
Game Play	The game should give hints, but not too many
Game Play	The game should give rewards
Game Play	There must not be any single optimal winning strategy
Game Play	Design for multiple paths through the game

Table 1 – Design guidelines for creation of fun in video games

1.3.3. Designing Video Games for Foreign Language Learning

We shall complement the above list with a number of key principles that relate to the current work of English language learning, taken from “10 key principles for designing video games for foreign language learning” (Purushotma, Thorne, & Wheatley, 2009):

1. All elements of the game, particularly communication and input mechanisms, should have a playful spirit to them.
   “Having students type out full sentence responses is, naturally, too slow and cumbersome for use in a video game. Thus, most games simply provide students with menus offering choices of preconstructed sentences to choose from by clicking on them with their mouse. Yet, this removes the process of constructing a sentence from the student, making it questionable what, if any, learning of language forms is taking place.”
2. Metalinguistic descriptions and terminology should be presented through optional supporting material, not as part of the core gameplay.
   “It is generally not necessary — or desirable — to place such metalinguistic information as central feature of the game; nor should the game require players to name the grammatical categories and other metalinguistic concepts in order to make progress in the game, so long as they are able to use them successfully. In fact, many of the bestselling language learning titles (like Pimsleur and Rosetta Stone) take the extreme position of forgoing any mention of metalinguistic terms at all.”

1.4. Evaluating Existing Solutions

This section serves two purposes:

1. To conclude whether or not the option of selecting an off the shelf product that answers all of the most important requirements is valid.
2. To verify the uniqueness of our proposed solution with regards to English language structure learning.

To achieve the first goal, we evaluated different kinds of quiz engines and learning management systems. At this end, all solutions evaluated lacked gameplay elements.

To achieve the second goal, we evaluated various English learning games. While all solutions evaluated were targeted to a younger audience, almost all answered all or most of the criterions used to evaluate good games, as noted by (Malone T. W., 1980).

Table 2 presents a Comparison of current solutions and our proposed solution, based on baseline set of design principles and requirements, partially taken from

Guideline	English Learning Games					LLS	LMS		Prodigy
	Alien Language	The Magic Key	Mingoville	Monkey Puzzles	Academy Island	Rosetta Stone	Quizlet	Moodle	
In game score/status	✓	✓	✓	✓	✓	✓	✓	×	✓
Audio feedback	✓	✓	✓	✓	✓	✓	✓	×	?
Error prevention	×	×	×	×	×	✓	×	✓	✓
Save state	×	×	✓	×	✓	✓	×	×	✓
Immediate feedback	✓	✓	✓	✓	✓	✓	✓	×	✓
Clear goal	✓	✓	✓	✓	✓	✓	×	×	✓
Variable difficulty level	×	×	×	×	✓	×	×	×	✓
Multiple level goals	×	×	✓	✓	✓	✓	×	×	✓
More than one way to win	×	×	×	×	×	×	×	×	?
Give few hints	✓	✓	✓	✓	✓	✓	✓	✓	✓
Give rewards	✓	✓	✓	✓	✓	×	×	×	✓
No optimal strategy	×	×	×	×	×	×	×	×	×
Multiple paths through game	×	×	✓	×	✓	×	×	×	?
Socialization	×	×	✓	✓	✓	✓	✓	✓	✓
Multiplayer	×	×	✓	×	×	×	✓	×	×

Table 2 – Comparison of existing solutions

1.5. Technologies

This project shall be implemented as a web-based application. Web-based applications are often multitier applications that divide functionality into tiers. Usually, these tiers are located on different computers, although it is possible that they will all reside on the same computer.

The three tier architecture is the most common one:

1. Bottom Tier: also known as the Data tier. Maintains the application's data, usually in a relational database management system.
2. Middle Tier: implements business logic. As the name implies, this tier serves as a mediator between the application's clients and its data.
3. Top Tier: this is the application's user interface, through which clients interact directly. Also known as the client tier.

There is a variety of technologies available at each tier. While we shall not attempt to cover all available ones (there is little time and even less available space in this paper), we shall briefly discuss the technologies most familiar to us, which are also the ones we shall be comparing. As Steve McConnoll notes in "Code Complete 2" (McConnell, 2004), "Programmers are more productive using a familiar language than an unfamiliar one".

1.5.1. Bottom Tier

A relational database stores data in data tables. To enhance modularity between the tiers, the database is usually available as a separate product, the relational database management system. The RDBMS is usually implemented as a server component, controlling access to the tables themselves. We shall examine three popular free license database management systems.

MySQL Now owned by Oracle, is a multiuser, multithreaded (i.e., allows multiple connections) RDBMS server that uses SQL to interact with and manipulate data. According to (Wikipedia, 2012), it is the world's most used open source RDBMS as of 2008.

Microsoft SQL Server Express Is a free version of Microsoft's extremely popular SQL Server RDBMS. It is designed specifically for small scale applications. Much like the rest of Microsoft's products, it is highly supported and very user friendly. It is however designed to work solely with Windows systems. This edition also limits the size of usable RAM and the size of the database itself.

Oracle Express Edition Was first introduced in 2005, and is offered as a free distribution on Windows and Linux machines. Oracle's superb database capabilities are mostly apparent in this version, but much like Microsoft's slimmed down version, it also limits the number of usable CPUs, RAM and user data.

Table 3 compares the above technologies based on selected criteria. We have selected MySQL as our RDBMS of choice, mainly due to the full set of features and unlimited performance and storage capabilities available with the free edition. Also, due to its popularity, it is highly supported throughout the community.

Feature	**Oracle 11g XE**	**MS SQL Server Express**	**MySQL Community Edition**
CPU usage count	1	1	Unlimited
License	Free	Free	Free
Support	✓	✓	×
GUI Administration	✓	✓	Workbench
Procedures	✓	✓	✓
Functions	✓	✓	✓
Triggers	✓	✓	✓
Windows	✓	✓	✓
Mac OSX	✓	×	✓
Linux	✓	×	✓
Unix	✓	×	✓

Table 3 – Bottom tier technologies comparison chart

1.5.2. Middle Tier

Programming Language

We are admittedly partial towards object oriented programming languages. Objects are reusable software components. Almost every noun can be represented as a software object in terms of attributes and behaviors. Programming in objects is extremely expressive, and can foster modularity, allowing the components to change and evolve with additional requirements and design constraints, with minimal effect on each other. We shall be examining two very popular languages for server side development.

Java Is an object oriented language with syntax similar to C and C++ that was developed by Sun Microsystems, Inc. Java was designed to run on any platform by converting Java source code to byte code, which is then run in each platform within an environment known as a virtual machine. Java is in widespread use for programming Web applications.

C# (pronounced C sharp) Is A general-purpose, object-oriented language and programming environment developed by Microsoft with syntax similar to C, C++ and Java, and it provides extensive tools that aid development on Microsoft platforms.

C# actually has a lot in favor development wise, in our opinion. Perhaps it is because we are more familiar with it. But one cannot deny the simplicity of using a single IDE (Integrated development environment) that supplies every available functionality required, out of the box. Compare this with Java, that due to its open source nature, has multiple IDEs available from different vendors, each supporting a different set of plugins that enhance and ease the development of the different components, such as GUI and Web components.
However, we have chosen Java as our programming language of choice, mainly because we are less familiar with it, but also because it is more commonly used in web based projects.

Application Servers

A web server is a specialized software that responds to client requests by providing it with resources. While the business logic is implemented using one of the programming languages reviewed above, a web server is required to for the client to access and invoke this functionality.

Microsoft Internet Information Services (IIS) Is the third most popular server in the world (IIS, 2012). It is designed for use with Microsoft Windows.

Apache Tomcat Is an open source software implementation of the Java Servlet and Java Server Pages technologies. Tomcat provides a pure java HTTP web server for java code to run. Tomcat is in wide use and very popular amongst web application development. (Apache Tomcat, 2012)

Jetty Is a pure java based HTTP server and Java Servlet container. It is developed as a free open source project as part of the Eclipse Foundation. It is used in numerous large scale projects, and is rapidly gaining popularity over the past years.

Table 4 compares the above based on selected criteria. We have selected Apache Tomcat as our application server of choice, mainly due to our previous familiarity with it, and its extensive community support.

Feature	**Jetty**	**IIS**	**Tomcat**
License	Free	With Licensed Windows System	Free
Web Socket	✓	✓	✓
REST	✓	✓	✓
Servlet Container	✓	×	✓
Support	Medium	High	High

Table 4 – Application server comparison chart

1.5.3. Client Tier

Dealing with web applications, we have a number of choices when implementing the client tier. Both programming languages reviewed at the middle tier have technologies that support running applications written in those languages at the client tier. This allows for faster programming and implementation, and reuse of logic and data structures between tiers, because a person is dealing with one language for both the client and middle tier.
However, the true benefit of web based application is HTML.

HTML Is a special type of computer language called a markup language. It is designed to specify the content and structure of web pages in a portable manner.
HTML enables a developer to create content that will render appropriately across the extraordinary range of devices connected to the internet. This includes smartphones, tablets, laptops, desktops and more. (Deitel, Deitel, & Deitel, 2012)
Now under development, HTML5 is the new emerging version of HTML.
HTML5 references two additional technologies:

JavaScript Is a language that helps one build dynamic web pages and computer applications. It enables one to do the client side programming of web applications.

Cascading Style Sheets (CSS) A format used to specify the presentation, or style, of elements on a web page. (Deitel, Deitel, & Deitel, 2012)

jQuery Currently the most popular of hundreds of JavaScript libraries. jQuery simplifies JavaScript programming by making it easier to manipulate a web page's elements and interact with servers in a portable manner across various web browsers. It provides a library of custom graphical user interface (GUI) controls (beyond the basic GUI controls provided by HTML5) that can be used to enhance the look and feel of your web pages. (Deitel, Deitel, & Deitel, 2012)

1.5.4. Client and Middle Tier Communication Protocol

Representational state transfer (REST) Refers to an architectural style for implementing web services.

The REST protocol follow a synchronous fashion: A request is made by the client, followed by a response from the server (Pull). It is impossible for the server to initiate communication with the client, although some methods simulate such Push capabilities.

JavaScript Object Notation (JSON) An alternative to XML (Extensive Markup Language) for representing data. It is a text-based data-interchange format used to represent objects in JavaScript as collections of name/value pairs. Json notation is identical to declaring objects in JavaScript (and hence the name) meaning that creating objects from JSON is extremely fast and efficient (reading data and loading to memory).

A number of third party libraries (GSON, Jackson) that are capable of converting JSON objects to and from Java objects, extremely simplifying the development of a Java – JavaScript paradigm project.

1.5.5. Development Process

A software development process describes an approach to building, deploying, and possibly maintaining software.

Iterative In an iterative lifecycle process, development is organized into a series of short, fixed-length mini-projects called iterations; the outcome of each is a tested, integrated, and executable partial system. Each iteration includes its own requirements analysis, design, implementation, and testing activities.

Waterfall In a waterfall lifecycle process there is an attempt to define all or most of the requirements before programming, and often, to create a thorough design before programming.

The iterative approach is especially suitable to a project of this nature, since the full set of requirements shall be compiled throughout the development process.

1.5.6. Integrated Development Environments (IDE)

To support development of any software project, although not mandatory, IDE software greatly simplifies the coding process. IDEs are designed to maximize a programmer's productivity, for one by greatly reducing configuration requirements. (Integrated Development Environments, 2012)

Microsoft Visual Studio Is an IDE from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services for all platforms supported by Microsoft Windows. A paid license is required for the full capable version. (Microsoft Visual Studio, 2012)

Eclipse Is a multi-language IDE comprising an extensible plug-in system. It is a free software, highly popular for development of Java applications. (Eclipse (software), 2012)

Intellij Is a commercial Java IDE by JetBrains. Intellij requires a paid license to access the full featured version, which, from experience, supports extremely rapid and fluid coding.

Table 5 compares the above technologies based on selected criteria. We have selected Intellij IDEA as the IDE of choice, mainly due to its extensive set of coding supporting features, ready out-of-the-box.

Feature	**Visual Studio**	**Eclipse**	**Intellij IDEA**
Java Support	×	✓	✓
C# Support	✓	×	×
JavaScript Support	✓	Via Plugin	✓
HTML Support	✓	Via Plugin	✓
License	Paid	Free	Paid

Table 5 – IDEs comparison chart

2. ANALYSIS AND DESIGN

The analysis and design step (or steps) includes gathering the user requirements, analyzing them in terms of software construction and finally designing the final software product.

2.1. Requirements: The Game Design Document

Prodigy presents fill-in-the-blanks, multiple choice type of question, augmented with elements from game design in order to enhance participant motivation. A game consists of an exercise and a player. The game ends when the player has completed all questions, or has lost all health. Collecting as many stars as possible is the goal of the game (as well as not getting hurt).
To start a game the player must first select an exercise. This is accomplished by first selecting a subject-matter and subsequently an exercise. A subject is identified by a name (e.g. present simple, past progressive etc.), an exercise - by a title (e.g. put the verb into the past tense, rewrite into the negative). The player may change the selected subject or exercise at any time.
A game starts with the first **question** displayed, and the first **blank** is automatically selected. To progress to the next question the player must solve the current one. The possible **answers** are displayed for the currently selected blank. A question is a portion of text with one or more missing words or characters (from now on – tokens). The location of each missing word or character (or a consecutive sequence of) is called a blank. Each blank has one or more possible predefined answers. The player may go back to a previously solved question. Blanks may be selected freely – there is no constraint on the order of solving blanks.
An answer may be correct or incorrect. A click on an answer submits it. Submitting a correct answer results in solving a blank, and submitting an incorrect answer results in loosing health, and If a hint is defined to an answer, it is displayed. In any case, submitting an answer results in placing the text inside the blank.
A blank can be either solved or unsolved. While in any of the above states, a blank may be selected or unselected. A selected blank presents an arrow to indicate the user it is the current blank for which answers are displayed.

A question may be solved or unsolved. A solved question means that all of its blanks have been solved – i.e. filled with the missing tokens.

All blanks initially contain a star. To collect a star one must solve a blank on the first attempt. The player gets one shot to collect a star from a blank. A star is lost as the result of a false submission. When the game ends the total grade is calculated as $\frac{|stars\ collected|}{|stars\ in\ exercise|} \times 100$.

A player is initially granted 10 hitpoints. A hitpoint represents a health point, so the health of a player consists of hitpoints. The health is depleted when the amount of hitpoints reaches zero – in this case the player "dies" and the game ends as a failure. The health cannot be incremented. Loosing hitpoints is possible only by submitting false answers.

The **score** keeps track of the points earned per each solved blank. A solved blank grants the user a maximum of 100 points if no mistakes were made. For every incorrect submission, this amount of points is halved. The player may receive a **score-multiplier** if he manages to solve at least two questions without making a single mistake. The level of the multiplier is expressed as an integer larger than 2. When the player is in possession of the multiplier, every correct submission is multiplier by its value, and this amount is added to the score. After receiving the multiplier, any additional question solved increments its value by one (again, under the constrained that no mistakes were made during this process). The multiplier is lost when a mistake is made.

To keep things for interesting, we introduce the **combo**. The combo is a mechanism designed to provide additional feedback when the player is on a row – does not make any mistakes. Every correct submission results in a feedback in the form of a positive word – e.g. Good, Great etc. The combo increments with every correct submission, which results in a different positive word displayed, usually of higher positive context (for example, Good will be displayed before Great). The combo is reset when a mistake is made.

2.1.1. User Interface Mockup

We start off by presenting some rough mockups of the user interface, which were created as part of requirements elicitation (English questions content inspired by (Stannard, 1959), mockups created with Expression Studio).

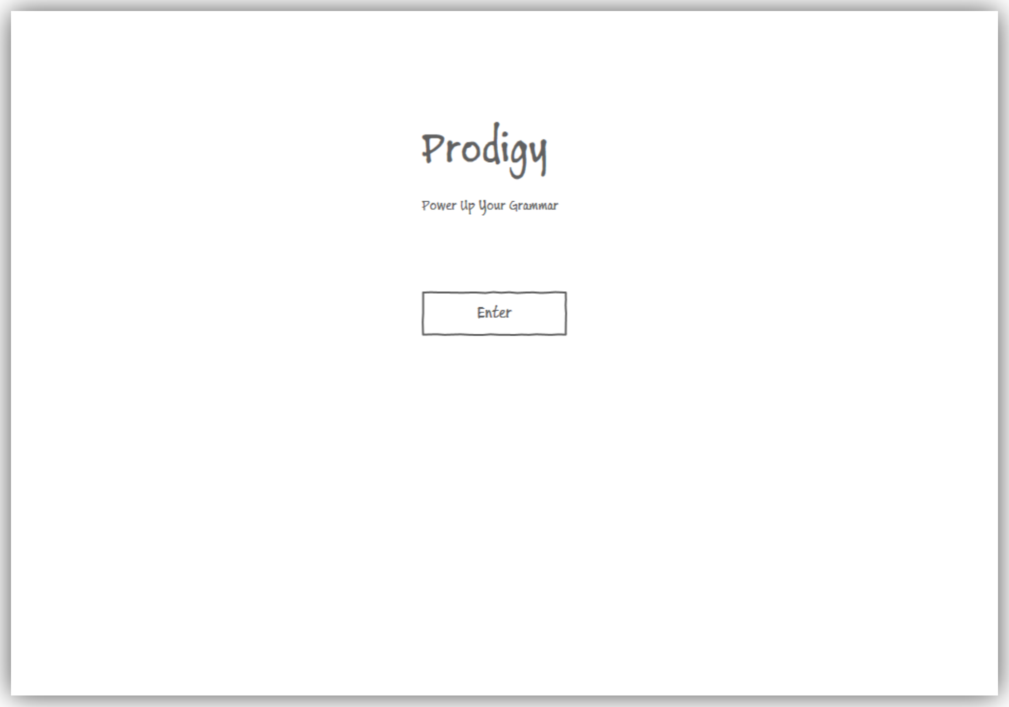


Figure 1 - home screen

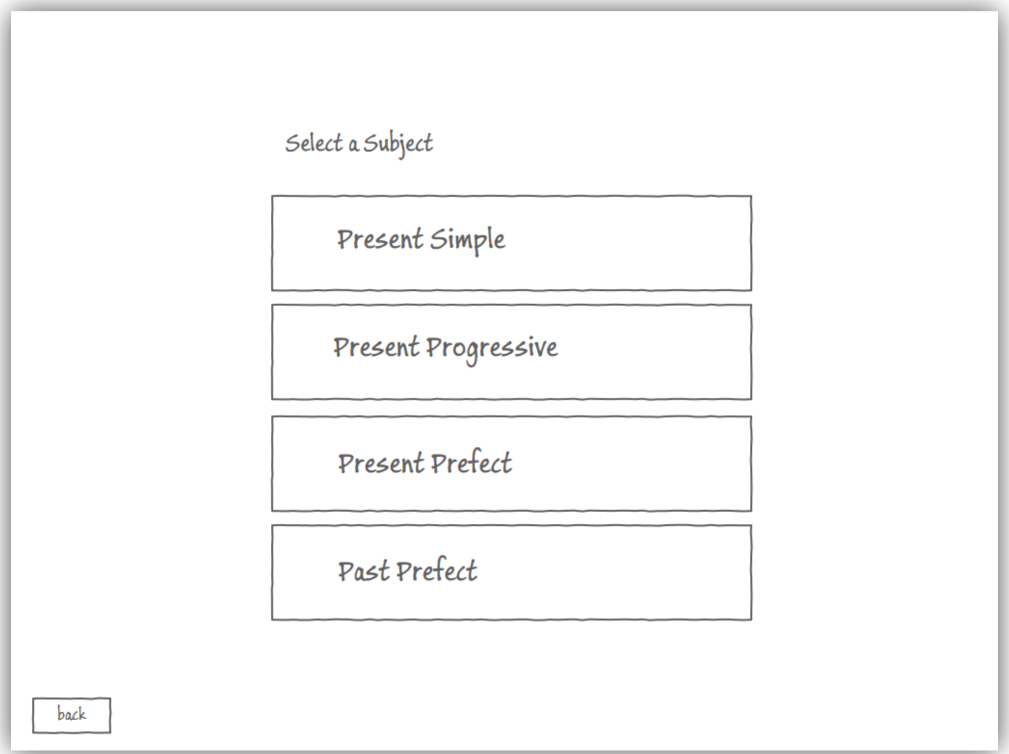


Figure 2 - subject selection screen

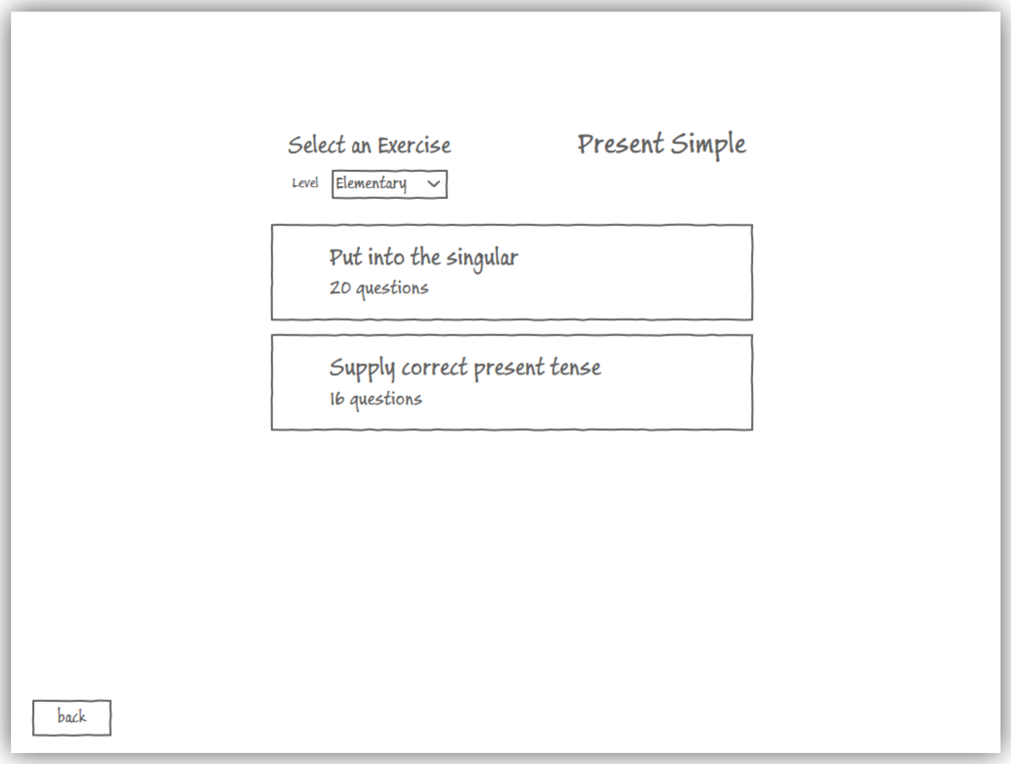
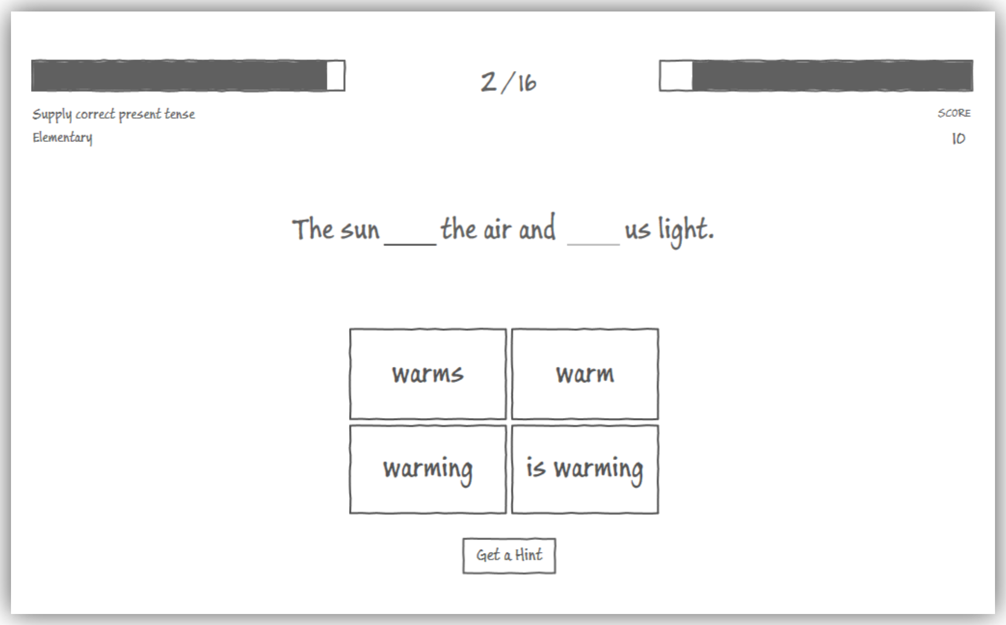


Figure 3 - exercise selection screen



2/16
Supply correct present tense
Elementary
SCORE
10
The sun ___ the air and ___ us light.
warms
warm
warming
is warming
Get a Hint

Figure 4 - second question selected, first blank selected

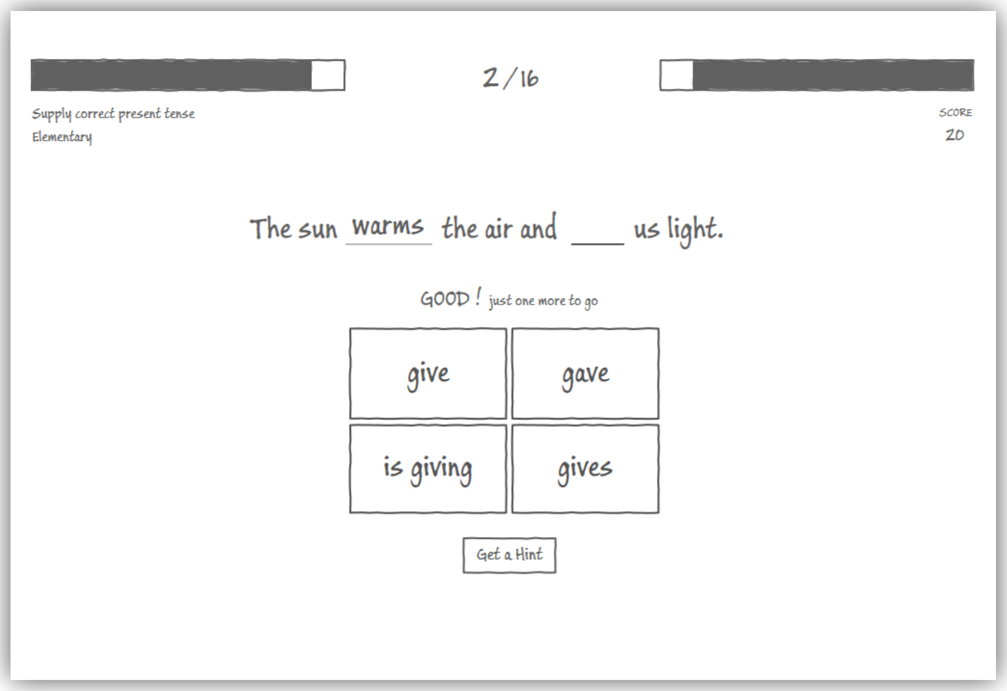

2/16
Supply correct present tense
Elementary
SCORE
20
The sun warms the air and ____ us light.
GOOD! just one more to go
give
gave
is giving
gives
Get a Hint

Figure 5 - correct answer submitted

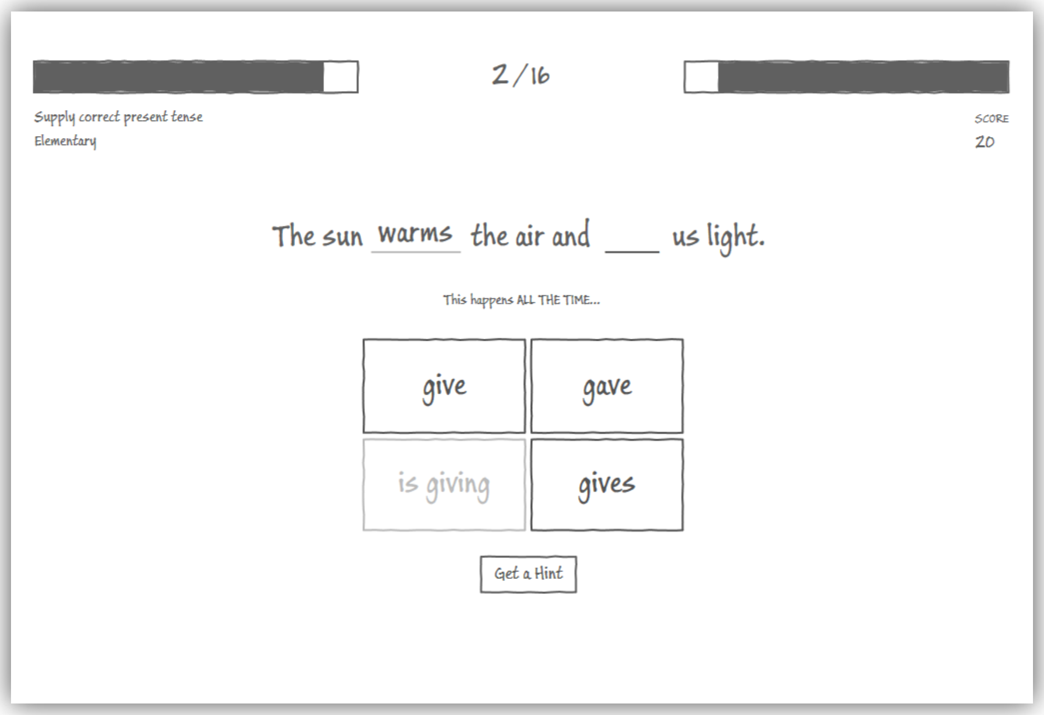

2/16
Supply correct present tense
Elementary
SCORE
20
The sun warms the air and ____ us light.
This happens ALL THE TIME...
give
gave
is giving
gives
Get a Hint

Figure 6 - second blank selected, false answer submitted, feedback shown

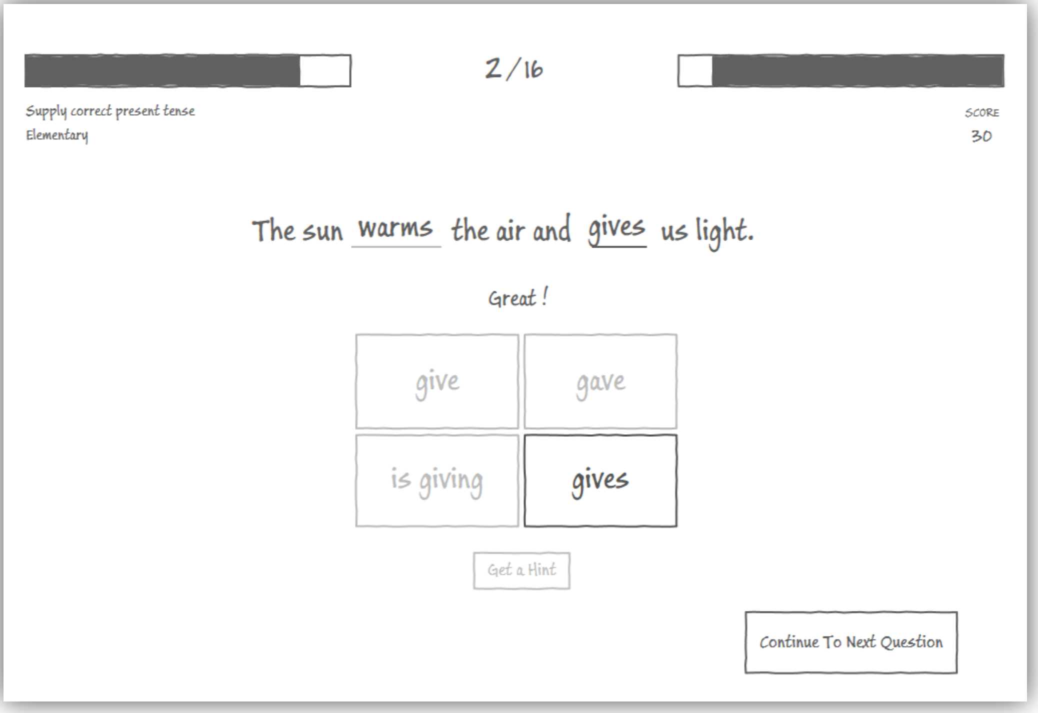

2/16
Supply correct present tense
Elementary
SCORE
30
The sun warms the air and gives us light.
Great !
give
gave
is giving
gives
Get a Hint
Continue To Next Question

Figure 7 - correct answer submitted

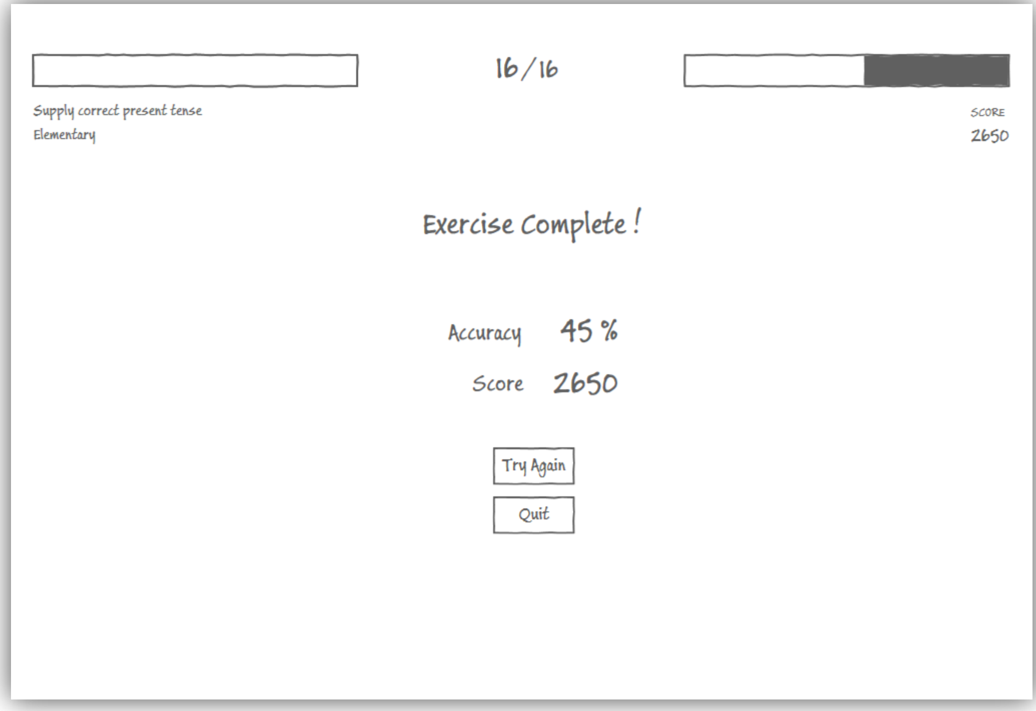


Figure 8 - exercise completed

2.2. *Analysis*

2.2.1. Gameplay Flow Chart

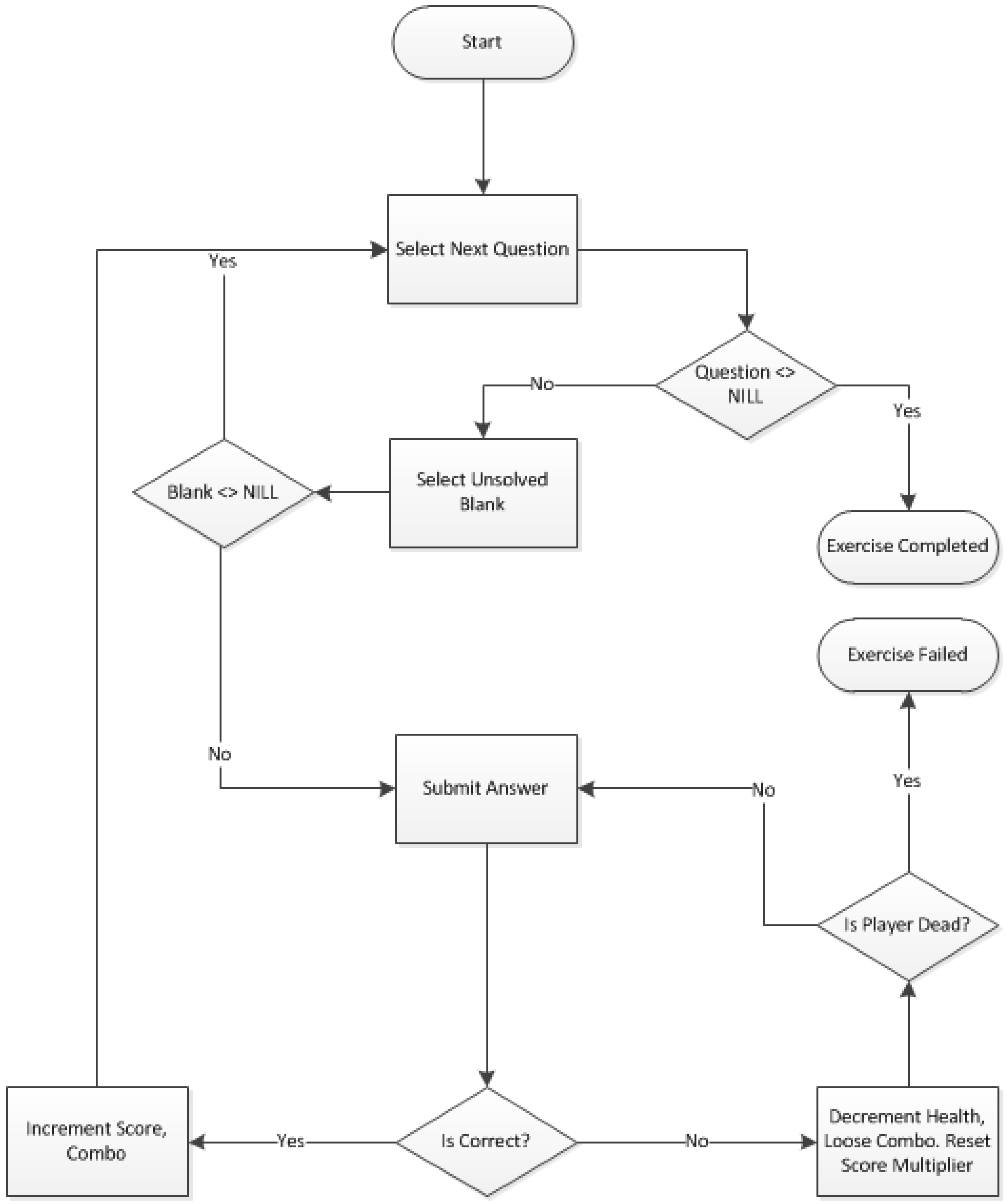


Figure 9 - Gameplay flow chart. Added for additional clarification of requirements

2.2.2. Use Cases

The requirements presented earlier provide us with enough information to elicit use cases in a manner more adequate for implementation. The granularity of the use cases is fine. Basically, one can distinguish two major use cases – **Select a Game** and **Play a Game**. Selecting a game includes use cases 1 and 2, Playing a game includes use cases 3-11.

ID	Name	Short Description	Actor
1	Select a Subject	Select a subject to practice	Player
2	Select an Exercise	Select an exercise from those available to the selected subject	Player
3	Select Next Question	Move to the next unsolved question	Player
4	Select Previous Question	Move back to a solved question	Player
5	Select Blank	Select a blank to fill and view the available answers	Player
6	Submit Answer	Submit an answer as a solution to a blank and receive an appropriate feedback	Player
7	Pause Game	Pause the current game and present the options menu exercise	Player
8	Resume Game	Resumes the game from a paused state	Player
9	Quit Game	Quit game back to exercise selection	Player

Table 6 - use cases

UC1 ***Select Subject***

Actor: Player

Precondition*:* The user clicked the *start* button at the welcome screen.

Main Success Scenario:

The user enters the subject selection screen. He is presented with a list of all available subjects, each identified by a name. The user clicks the desired subject.

Success Guarantee:

The system stores the selected subject, loads the relevant exercises and forwards the user to the exercise selection screen.

Extensions:

The user may return to the welcome screen by clicking the back button.

UC2 ***Select Exercise***

Actor	Player
Cross Reference	UC1
Precondition	User has successfully completed UC1.

Main Success Scenario

The user views all exercises of the previously selected subject. Each exercise is identified by a title. The user clicks an exercise he wishes to play.

Success Guarantee

The selected exercise is loaded and the user is forwarded to game start screen.

Extensions

The user may return to the subject selection screen by clicking the back button.

UC3 ***Select Next Question***

Actor	Player
Precondition	The current question is solved.
Main Success Scenario	User clicks the *next* button.
Success Guarantee	The next question is displayed or an end of exercise screen is loaded.

Extensions

If the current question was last, pressing the next question button ends the game. In this case the exercise-cleared screen is loaded.

UC4 ***Select Previous Question***

Actor	Player
Precondition	The current question is not the first one in the exercise.
Main Success Scenario	The user clicks the *back* button.

Success Guarantee	The previous question is displayed.

UC5 ***Select Blank***

Actor	Player
Cross Reference	UC3, UC4
Precondition	A question is presented.
Main Success Scenario	The user clicks a blank.

Success Guarantee

The arrow icon is points to the selected blank and the multiple choices are displayed.

Extensions

A blank is automatically selected by the system when the user completes UC3 or UC4, or when a blank is solved which is not the last unsolved blank in the question.

UC6 ***Submit Answer***

Actor	Player
Cross Reference	UC3, UC4, UC5
Precondition	An unsolved blank is selected

Main Success Scenario

The user clicks the answer he whishes to submit. The system checks if the answer is true or false.

In case the answer is true:

1. If the blank contained a star
    i. Collect it, and increment number of collected stars
2. If a multiplier is in effect
    i. The points stored at the blank are multiplied by the multiplier value
3. The score is incremented
4. The combo is incremented
5. Show the combo feedback inside the answer button
6. Show the answer text at the blank

7. Pop the score received from the blank
8. If the blank solved is last in question
    i. Unlock next-question button
9. Else
    i. Automatically select the next unsolved blank in the question

In case the answer is false:

1. Reset the multiplier
2. Reset the combo
3. Decrement user health by one hitpoint
4. If user health is zero or lower
    a. Go to UC8
5. Else
    a. Show answer in red
    b. Display the associated hint, if any
    c. Show the answer inside the blank

UC7 ***Pause Game***

Actor	Player
Precondition	A game is in progress
Main Success Scenario	User clicks the pause button during gameplay.
Success Guarantee	A game paused screen is presented.

UC8 ***Resume Game***

Actor	Player
Cross Reference	UC7
Precondition	Pause screen loaded
Main Success Scenario	User clicks the resume button during gameplay.
Success Guarantee	The pause screen disappears and game is resumed

UC9 ***Quit Game***

Actor	Player
Cross Reference	UC9
Precondition	Pause screen loaded
Main Success Scenario	User clicks the quit button
Success Guarantee	user is redirected to exercise selection screen

2.2.3. Domain model

The following UML class diagram captures the entities in play.

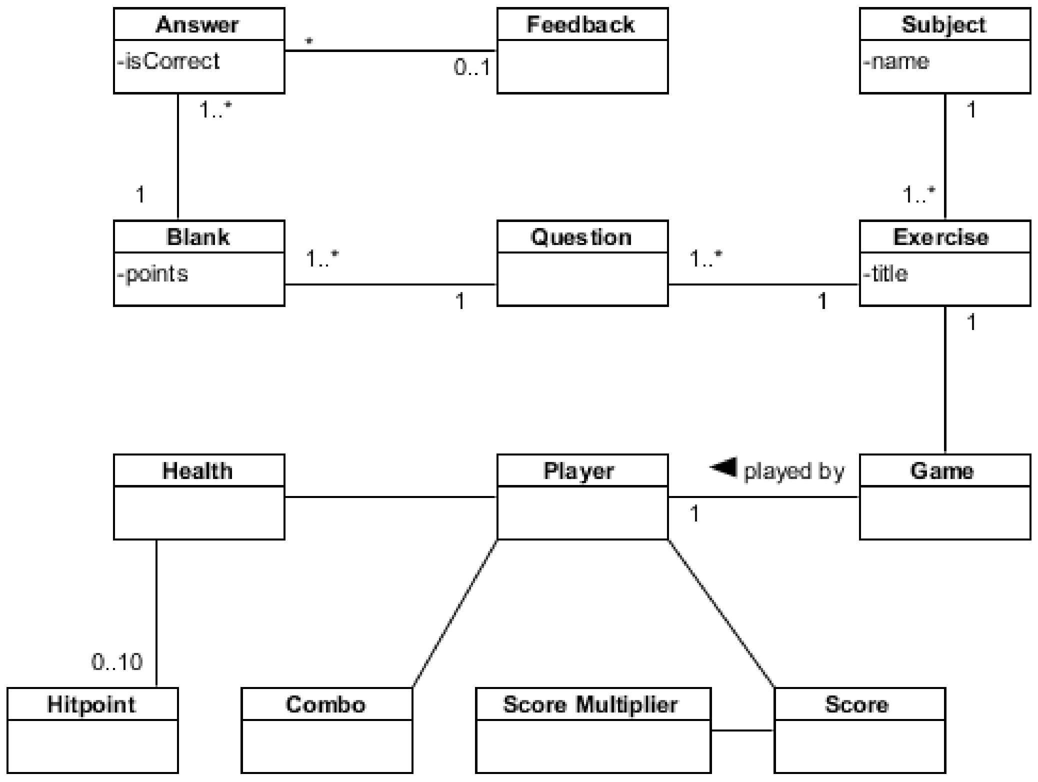


Figure 10 - Domain Model UML class diagram

2.3. Software Design

2.3.1. Architecture

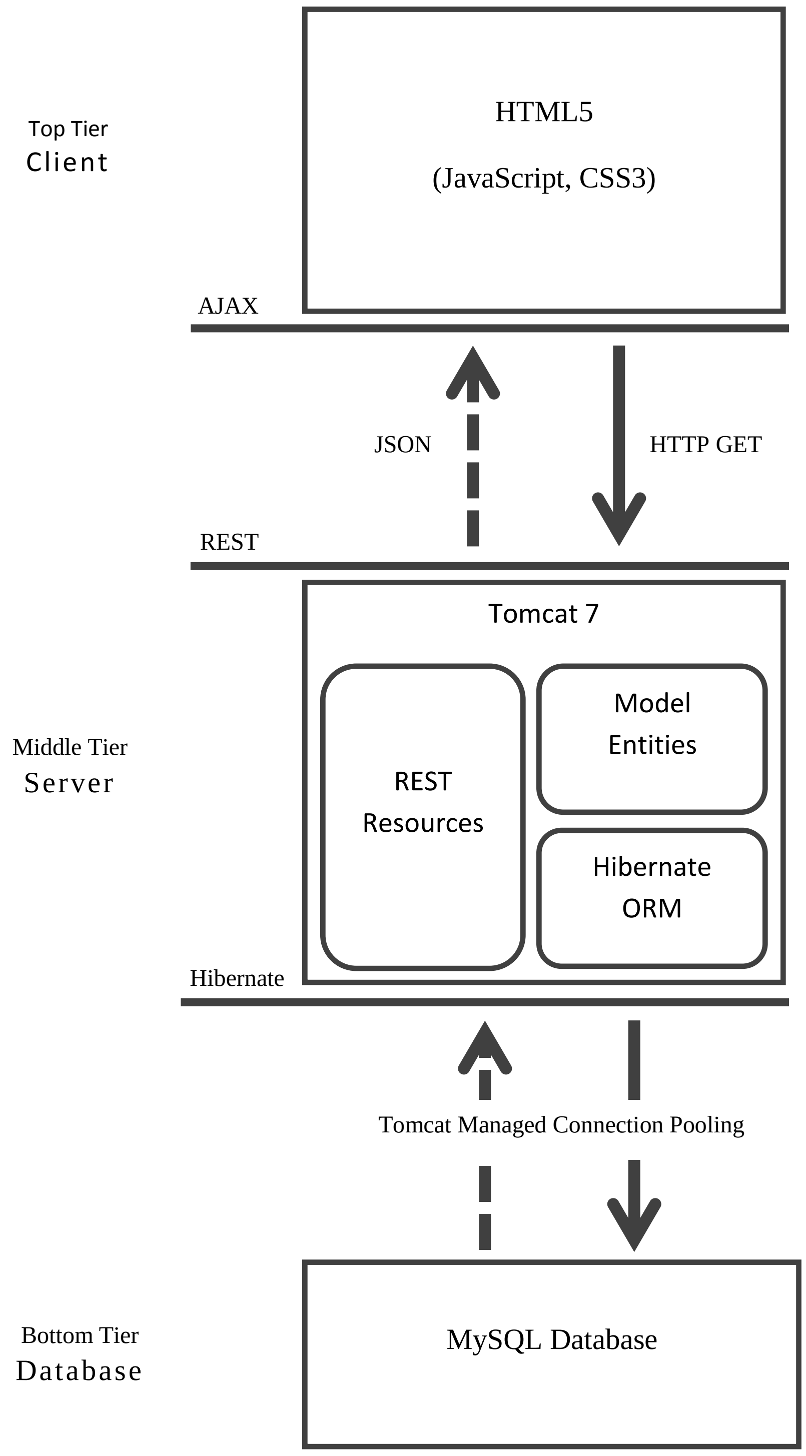


Figure 11 – top level architecture

The above diagram captures the separation of tiers in our system. We can distinguish 3 individual tiers in our system:

Data Tier – A relational database holding persistent data used in our game.

Server Tier – A service oriented module exposing a RESTful API for fetching game data from the Data tier.

Client Tier – This is where the game logic and presentation resides.

Key points in the architecture:

- Reponses generated from REST services invocations are in JSON format.
- All data required to play a game should be fetched beforehand. In other words, the client should not invoke server resources during gameplay. This is to support fast response times.
- The server and client modules are completely decoupled. Each may grow and change independently, under the constraint that the current server API shall not change. This also allows us to change the server/client programming language and platform (e.g. mobile app that uses the same services).

2.3.2. The Database Schema

This data tier comprises of a MySQL database. It persists data required to play a game. Users' actions are not reflected in the database, i.e. no dynamic information is stored during or after a game session. Insertion of data is done using a temporary admin module during development stage. It is not a part of the final product.

Subjects

Column	**Domain**	**Notes**
id	integer	
name	varchar(200)	
number_of_exercises	integer	

Table 7 - Subjects data table

Exercises

Column	Domain	Notes
subject_id	integer	Fk_Exercises_Subjects
id	integer	
title	varchar(200)	
number_of_questions	integer	

Table 8 - Exercises data table

Questions

Column	Domain	Notes
id	integer	
exercise_id	integer	Fk_Questions_Exercises
html	longtext	
idx	integer	Order within exercise

Table 9 - Questions data table

Blanks

Column	Domain	Notes
question_id	integer	Fk_Blanks_questions
idx	integer	Order within question

Table 10 - Questions data table

Answers

Column	Domain	Notes
question_id	integer	Fk_Answers_Questions
blankIdx	integer	FK_Answers_Blanks
token	varchar(100)	
correct	boolean	
feedback	varchar(100) (nullable)	

Table 10 - Answers data table

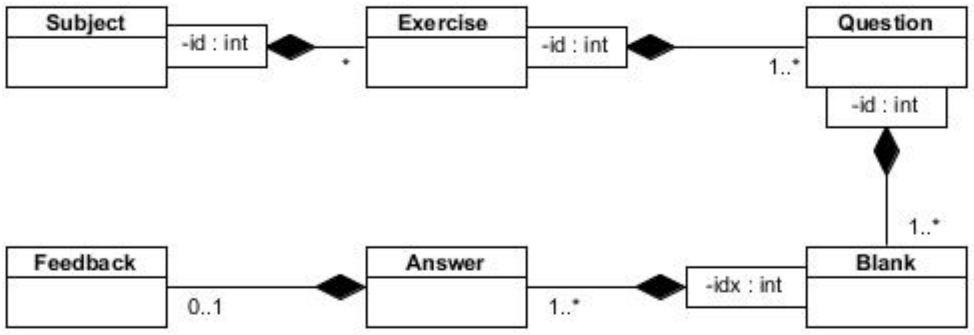


Figure 12 - Database entity relationship in UML class diagram

2.3.3. Server API (REST Resources)

The server tier is implemented following a RESTful architecture. Specifically, we have followed the HATEOS principle (HATEOS, 2013). Basically what this means is that the client application knows only the entry point to the services (a single URL), and all subsequent actions are discovered within resources returned by the services. Specifically, a URL that redirects the client to further actions. The server is a Query server, i.e. it supports only reading of data, no write capabilities. Note that we do not support paging at this step (ability to request *N* number of entries from a specified *offset*).

The following sequence diagram describes the sequence of actions required to select an exercise.

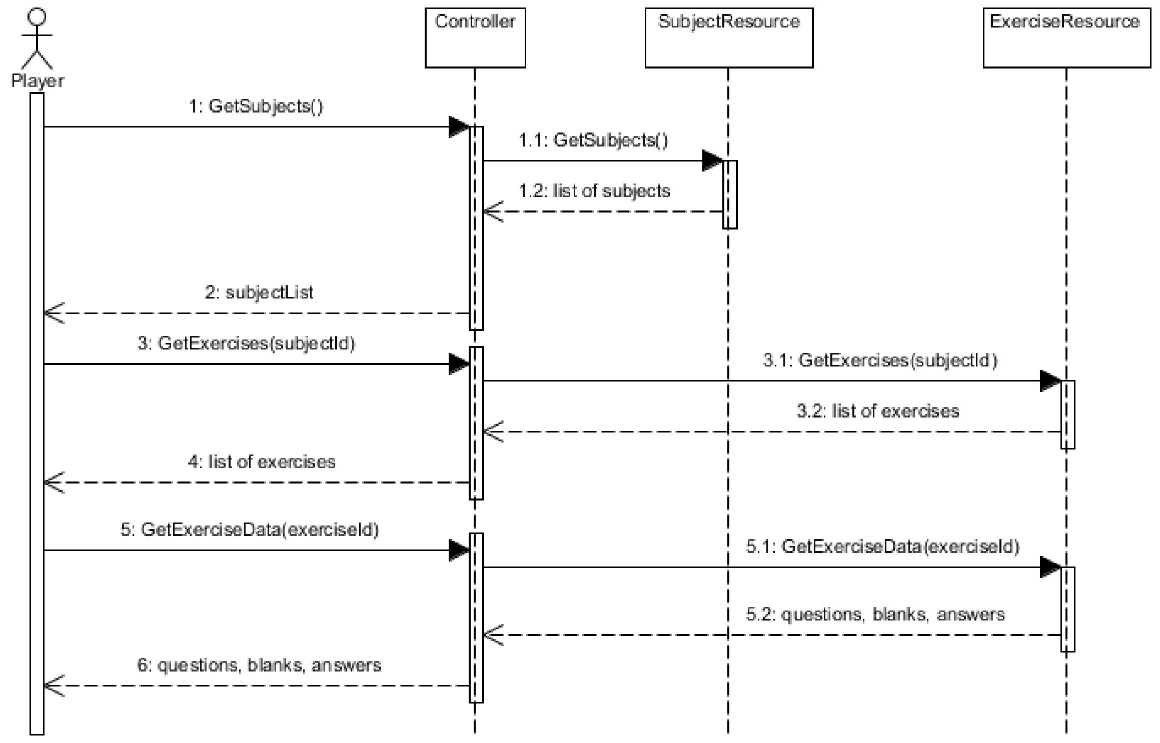

Player
Controller
SubjectResource
ExerciseResource
1: GetSubjects()
1.1: GetSubjects()
1.2: list of subjects
2: subjectList
3: GetExercises(subjectId)
3.1: GetExercises(subjectId)
3.2: list of exercises
4: list of exercises
5: GetExerciseData(exerciseId)
5.1: GetExerciseData(exerciseId)
5.2: questions, blanks, answers
6: questions, blanks, answers

Figure 13 - Sequence Diagram describing interaction with exposed resources

The following presents the objects returned after each request through a URI (in bold). The application has a single entry point, and further links are provided to navigate through the resources. The use cases are presented as headlines.

Get all subjects

Relative URL **/subjects**

```
{
  “subjects”:   [
                  {
                    “name”:               STRING,
                    “numberOfExercises”:  INTEGER,
                    “exercises”:          { “href”: URI  }
                  }
                  . . .           //all subjects
                ]
}
```

Get all exercises that belong to a subject identified by the supplied name

---

Relative URL **/exercises?subject=<NAME>**

```
{
  “subject”:    { “name”:   STRING },
  “exercises“:  [
                  {
                    “title”:                STRING,
                    “difficultyLevel”:      { “name”:   STRING },
                    “numberOfQuestions”:    NUMBER,
                    “href”:                 URI
                  },
                  ...                       //all exercises
                ]
}
```

Get a specific exercise identified by id

Relative Url **/exercises/{id}**

```
{
“exercise”: {
            “subject”:            { “name”:   STRING },
            “title”:              STRING,
            “difficultyLevel”:    { “name”:   STRING },
            “numberOfQuestions”: NUMBER,
            “questions”:   [
                              {
                              “html”:STRING,
                              “blanks”:   [
                                            {
                                            “answers”:  [
                                                          “token”:
                                                          “feedback”:
```

```
                                                        "correct":
                                                        ]
                                              },
                                              // blanks...
                                            ]
                                  },
                                  // questions...
                                ]
                    }
          }
```

2.3.4. The Client Application

Most of this project's work resides in the development of the client application. This is a single app application, developed using HTML5 (and by extension – JavaScript and CSS3). A single HTML document is downloaded to the client browser. It contains references to necessary CSS3 and javascript files. Each such file contains a the logic required at each screen. This allows us to implement and test screens individually, as well as changing the logic with ease. As a consequence, it introduces difficulties in maintenance.
The first function to load is `main`. It manages the screens, using published events, by appending and removing screens from the DOM.

Screens Layout & Implementation

There are a total of 7 separate screens in the game. Every screen inherits from the Entity object, and implements its own logic. The following state machine captures the screens and the events transferring the user from one screen to the next.
Each screen is responsible for a number of use cases:

Welcome

Purpose	Presents the entry point to the system.
Published Events	"Start"

Subject Selection

Purpose	Loads and presents the list of subjects from the server
Published Events	“Subject Selected”

Exercise Selection

Purpose	Loads and presents the list of exercise from the server. Loads the exercise data when a subject is selected
Published Events	“Exercise Selected”

Game

Purpose	Contains the logic required to play a game
Published Events	“Pause”, “Exercise Failed”, “Exercise Cleared”

Paused

Purpose	Presents the user with options to resume or quit the current game.
Published Events	“Quit”, “Resume”

Exercise Failed

Purpose	Presents the user with options to restart the game with the same exercise as before, or quit from game back to exercise selection
Published Events	“Quit”, “Restart”

Exercise Cleared

Purpose	Presents the user with the total grade earned at the game, and options to restart the game with the same exercise as before, or quit from game back to exercise selection
Published Events	“Quit”, “Restart”

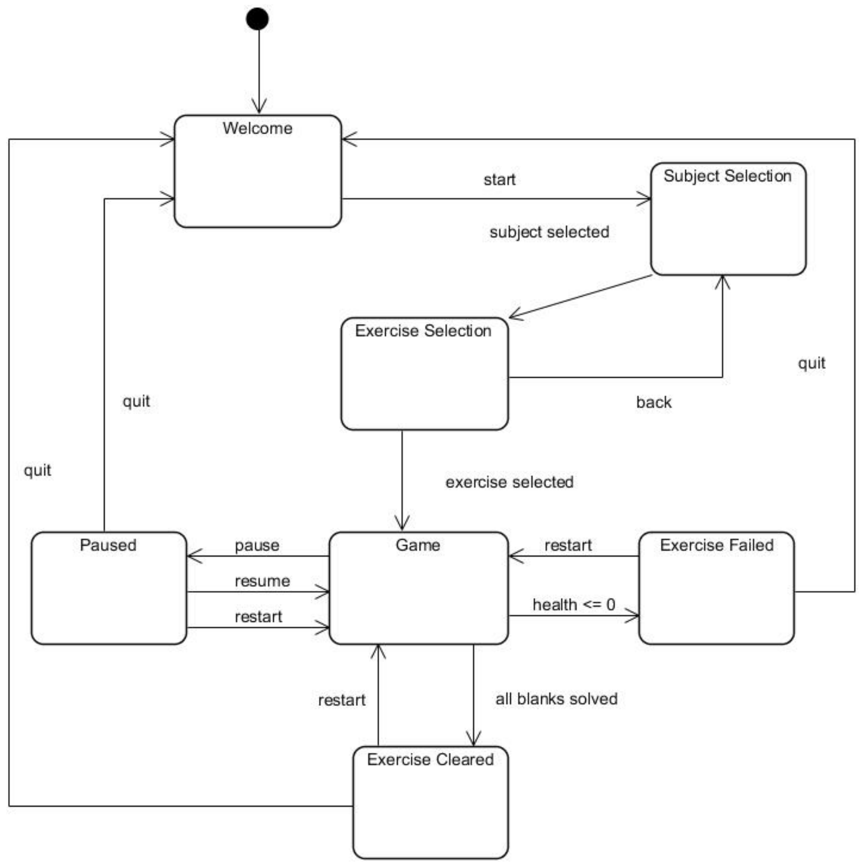


Figure 14 - Client App Screen State Machine

The Entity Implementation & Deriving Functions

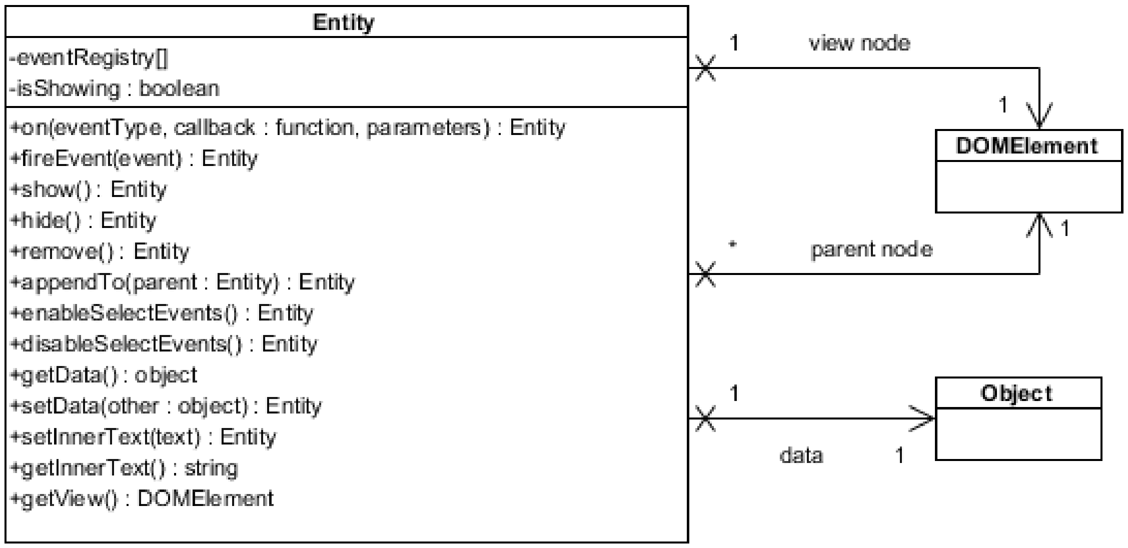
The proprietary Entity object provides an abstraction of drawable elements in the client application. Almost all objects used inherit from the Entity object, by use of Functional inheritance. As part of the operation contract, all functions return *`this`*, unless a function specifically returns a different value, e.g. *`getInnerText`*.

It is important to notice that all "click" events are abstracted by a "select". This eases transition to touch events in later versions.

Main usage:

- ***Augmentation*** Inherit from the Entity and augment it with custom functionality.
- ***Instantiation*** Provide an entity with a DOM element that will present the view, and an optional data object to be used by the software client.
- ***Subscribe to Events*** Use the *on* function and supply it with the name of an event, a callback function and parameters to be used by the callback function.

- ***Fire Events*** Fire events at will using the `fireEvent method.`



Entity
-eventRegistry[]
-isShowing : boolean
+on(eventType, callback : function, parameters) : Entity
+fireEvent(event) : Entity
+show() : Entity
+hide() : Entity
+remove() : Entity
+appendTo(parent : Entity) : Entity
+enableSelectEvents() : Entity
+disableSelectEvents() : Entity
+getData() : object
+setData(other : object) : Entity
+setInnerText(text) : Entity
+getInnerText() : string
+getView() : DOMElement
1
view node
1
DOMElement
1
*
parent node
1
Object
data
1

Figure 15 - Entity Object Class Diagram

3. RESULTS

Overall impressions are extremely positive. Users have expressed delight when using the system, and have complemented the visual design of the application. The most notable reaction from the users is the desire to try and reach the highest score possible. However, making mistakes resulted in a smile and little frustration, due to the visual feedbacks and hints supplied upon them.

A few remarks were made almost unanimously by all users:

- Most users needed instructions regarding the gameplay. When confronted with a question and a blank, it was not obvious that one should press the answer in order to submit an answer. This was mostly noticebale within an older users (40+). Younger users did not ask for instructions, but rather tired until succeeded. This may be attributed to the fact that younger users are more familiar with computer games, and as a result understand that most games must be learned by practice.
- Users have complained that when completing a question, the following one does not automatically load, but the user rather needs to actively press the next button. Also, the appearance and location of the next question button does not reflect its importance.

The application was publicly accessible for two months.

4. GLOSSARY

Alien Language – Alien language is an online Computer Assisted Learning site, designed to support the teaching of modern Foreign Languages to school children 11-14, although it can be used by learners of all ages . http://crossland.co.uk/alienlanguage.co.uk/alienlanguage/index.htm

The Magic Key – The magic key website is aimed at children aged 5-6 years and follows the TV programs in which Kiepper, Biff, Chip, Wilf, Wilma and Floppy go off on their adventures. It covers English sentence and text level work. http://www.bbc.co.uk/schools/magickey/adventures/index.shtml

Mingoville - Mingoville is an interactive English language course that caters to kids of all ages worldwide. Mingoville consists of a wide variety of materials that can be reached directly through the portal. Mingoville is a storybook world of English language learning and interaction. There are 10 missions in the course. Each mission covers a distinct theme and can be worked on independently from the other missions. http://www.mingoville.com/en/demo.html

Monkey Puzzles – Monkey Puzzles is a collection of 8 progressively unlocked mini-games designed to test knowledge as a preparation to the Cambridge English Exams. http://www.cambridgeesol.org/exam-preparation/games/monkey-puzzles.html

Academy Island – Players of Academy Island have to progress through different levels by helping an unknown alien life form use English in a range of situations such as shopping in a bakery and visiting places such as an art gallery and library. http://www.cambridgeesol.org/exam-preparation/games/academy-island-game.html

Rosetta Stone – Rosetta Stone is multiple language learning software, that uses patented, smart speech recognition tools. http://www.rosettastone.com/

Quizlet – Quizlet is an online learning tool. It is based on the concept of Sets, i.e. multiple question-answer pairs that users create and reference alike. http://quizlet.com/

Moodle – Moodle is a popular open source Learning Management Software. http://moodle.com/

5. REFERENCES

Apache Tomcat. (2012, 12 15). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Apache_Tomcat

Eclipse (software). (2012, 12 17). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Eclipse_(software)

IIS. (2012, 12 16). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Internet_Information_Services

Integrated Development Environments. (2012, 12 15). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Integrated_development_environment

Microsoft Visual Studio. (2012, 12 17). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Visual_studio

WebSocket. (2012, 12 17). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/WebSocket

Bisson, C., & Luckner, J. (1996). Fun in Learning: The Pedagogical Role of Fun in Adventure Education. *Journal of Experimental Education*, 109-110.

Clark, R. C., & Mayer, R. E. (2011). *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning.* San Francisco: Pfeiffer.

Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience.* Harper and Row.

Deitel, P., Deitel, H., & Deitel, A. (2012). *Internet & World Wide Web: How to Program.* Prentice Hall.

Federoff, M. A. (2002). *Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games.*

Jong, M., Shang, J., Lee, F., & Law, H. (2006). Learning Online: A Comparative Study of a Situated Game-Based Approach and a Traditional Web-Based Approach. *Technologies for E-Learning and Digital Entertainment, 3942*, 541-551.

Larman, C. (2007). *Applying Uml and Patterns.* New Jersey: Prentice Hall PTR.

Malone, T. W. (1980, August). What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games.

Malone, T. W. (1980). What Makes Things Fun to Learn? Heuristics for Designing Instructional Computer Games.

McConnell, S. (2004). *Code Complete, Second Edition.* Washington: Microsoft Press.

McGinnis, T., Bustard, D. W., Black, M., & Charles, D. (2008). Enhancing E-Learning Engagement using Design Patters from Computer Games. *First International Conference on Advances in Computer-Human Interaction.* Sainte Luce: IEEE Computer Society.

Paras, B., & Bizzocchi, J. (2005). Game, Motivation, and Effective Learning: An Integrated Model for Educational Game Design. *Changing Views - Worlds in Play.* DiGRA.

Pivec, M. (2007). Editorial: Play and learn: potentials of game-based learning. *British Journal of Educational Technology*, 387-393.

Prensky, M. (2002). The Motivation of Gameplay. *On The Horizon*.

Purushotma, R., Thorne, S. L., & Wheatley, J. (2009). *10 Key Principles for Designing Video Games for Foreign Language Learning.*

Rose, C., & Nicholl, M. J. (1998). *Accelerated Learning for the 21st Century.*

Rouse, R. I. (2005). *Game Design Theory & Practice.* Wordware Publishing Inc.

Sloman, M. (2001). *The E-Learning Revolution.* London: CIPD.

Smith, R. (2008). *Motivational Factors in E-Learning.*

Stannard, A. W. (1959). *Living English Structure, A Practice Book for Foreign Students.* Longmans.

Wang, H., & Sun, C.-T. (2001). Game Reward Systems: Gaming Expreiences and Social Meanings. *DiGRA.*

Wikipedia. (2012, 12 4). *MySQL*. Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Mysql