pixelware TRUE DIGITAL IDEAS

Modo Quiz

TECHNICAL DESIGN AND SPECIFICATION

1 Table of Contents

2 Update History:

- 9aug 2014 (Ståle): Minor updates and file copy to project drop box
- 14jan 2015 (Eirik + Board): new special rewards, question updates, distribution + control system and set up panel
- 13feb 2015 (Alexander): formalize verbal agreements from meeting 6feb2015 + various formatting changes
- 14feb 2015 (Alexander): add Technical Considerations section and remove design no longer applicable (design resource takes precedence)
- 17feb 2015 (Alexander): move most of outdated textual spec to the *design resource* (reference in text) as it is much clearer in terms of the flow through the game. Final touches on technology and specifications.

3 Introduction

Pixelware will implement, test and prepare the game described in this text for iOS and Android devices. The implementation will follow the graphical design outlined in the provided resource available at the time of writing at https://projects.invisionapp.com/share/9B1Q3JVTJ. This will be referred to as the *design resource* throughout this text.

The game is a multi-language quiz game with in-app purchasable quiz aids; which is a way for a player to get an edge on their opponents.

Syncrotec will own Modo Quiz in its entirety. Syncrotec is referred to as the Product Owner throughout this document.

3.1 Practices and Code Style

Agile and Lean project management with Kanban and rapid iterations.

The Modo Quiz project will be managed as a Lean project with a Kanban board for action points and User Stories.

We will strive to have a buildable product within the two first weeks of development. Having a buildable product means that it is possible to iterate with feedback from Product Owner, on the topics that needs discussion in order to secure progress.

3.1.1 Expectations and Intent During Iterations

Pixelware will provide the Product owner with regular builds and feedback should be on-point.

When early builds are made available, for review by the Product Owner, Pixelware will indicate what aspects is a matter of discussion. The early builds will not be fully functional or completely designed. In order for rapid iterations to be productive the Product Owner will need to keep the feedback scoped to the build that is presented (i.e. a build that shows the start screen animation feature should not get feedback about the login button not being clickable).

A build will come with a set of bullet points that is a complete set of features or design that is subject for discussion. As we close in on the final build, this list will be removed and at this point any and all feedback is welcome. If ideas are presented that is not a part of the specification, they will receive an estimate (both production time and price) and Product Owner can choose to add the feature to the delivery at an agreed upon price.

Delivery

Delivery will adhere to industry conventions to ensure a maintainable product.

For the code style we will use the existing styles recommended by the respective community. This means, the Unity3d source code will follow Unity3d's conventions. The Web API and Backend will follow the conventions used by Microsoft in its examples and style guides. This ensures ease of transition, if Modo Quiz were to be handed off to other developers, for instance for support and/or maintenance purposes.

Modo Quiz will be a delivery of three parts outlined below in:

- The Modo Quiz Game
- The Modo Quiz API
- The Modo Quiz Backend

These will be packaged as a deliverable Azure WebRole, and Modo Quiz Game packaged for submitting to App Store and Google Play.The Modo Quiz Game

4 Technical Considerations

The following sections are an overview of the chosen technologies and an insight into how they apply to the Modo Quiz product.

The game will be built with configuration in mind. There will be no hardcoded numbers or text in the game itself. This means that change requests such as "we want the time for each question to be 12 seconds instead of 10" is a single change, in a single file, on a single line. The only exception to this rule is changes that would have an impact on the graphical representation of the app (for instance changing the phase from 6 * 3 to 4 * 4, that is not to say it cannot be done (as we will make the *system* flexible, but it would incur costs in relation to design and the implementation of that design).

4.1 The Modo Quiz Game

A modern Unity3d game with high-quality project structure, source code and assets.

Modo Quiz will be implemented with Unity3d, a quickly growing and proven game engine for productive and proficient small teams. The code will adhere to best practice standards for data-driven applications and will use the *Model View ViewModel* (MVVM) design pattern. Furthermore, the project will follow the agreed upon conventions in the Unity3d community, and the code will be documented in-line with XML Documentation Comments (from the C# Programming Guide).

Assets will for the most part be made up of the provided PhotoShop (.psd) files received from the Product Owner. For other assets refer to section; Unity Dependencies and Project Delivery.

4.1.1 Text Assets for Translation

Modo Quiz is a multilingual app, as such all text will be read from a well-defined resource

We will store all text strings in a file that is easily editable by any text editor. We propose the following format, stored in a simple `.txt` file:

```
                                                        > /norwegian.txt

play        =     Spill
buy         =     Kjøp
```

Each language will therefore have an associated resource file, which associates a system key (for instance play) to the actual display text ("Spill").

These files will need to be stored using the Unicode encoding to ensure correct representation of special characters (such as æ, ø and å).

4.1.2 Model View ViewModel

The de-facto standard for data driven user experiences - Ensures high maintainability of product.

The Model View ViewModel pattern separates the concerns of the following questions:

- *Model*: How do we get the data we need?
- *View*: How do we show the data to the user?
- *ViewModel*: What data should we show to the user?

By using this pattern; implementation, maintenance and implementation benefits. Each part of the system has a clear responsibility and a clear and loose coupling to the other parts of the system. Bugs are easily located (i.e. if something is displayed wrong, it's in the *View*, if some data is wrong it's in the *Model* and if some data is missing from the *View* it's the *ViewModel*).

4.1.3 Unity Dependencies and Project Delivery

The ownership of all assets will be transferred to Product Owner upon project completion.

The solution will heavily use the newly released UI system (part of Unity 4.6 as of Nov 2014). Furthermore, if third party assets are used, we will make sure that the commercial nature of Modo Quiz does not violate their licenses. All assets will be contained in the project and will become the property of Product Owner upon final payment.

4.2 The Modo Quiz API

A modern and secure access point for the Modo Quiz Game, and the Modo Quiz Backend

The Modo Quiz Application Programming Interface (API) will be the unified access point for the Modo Quiz games for Android and iOS (and potentially Windows Phone), and the Backend. This will be a RESTful[1] cloud[2] service, prepared for Microsoft Azure for a scalable and persistent solution.

4.2.1 Microsoft Web API

A proven and tested framework for making modern web backends

We will implement the API with Microsoft's *Web API* product. This is an industry standard for making RESTful services with a clear and concise code base. By using established frameworks like this, we make sure that our code is easily understood by new resources (or maintainers) and therefore cheaper to maintain and support.

4.2.2 Entity Framework

Abstraction of concerns relating to the database - The wheel for data storage has been invented.

This is an Object Relational Mapper (ORM) which makes it easy to store objects (such as a *Question*) in a database – and equally easy to retrieve an object from the database.

---

[1] http://en.wikipedia.org/wiki/Representational_state_transfer
[2] http://en.wikipedia.org/wiki/Cloud_computing

Entity Framework is an abstraction layer between the actual database and the code – which means freedom in terms of storage types and security in terms of adhering to best practices for security and safety of transactions with the database.

4.2.3 Domain Model

Modo Quiz is driven by data. Here is an outline of what will be stored in the database.

This is a rough overview of the critical information stored per Domain Object in the Modo Quiz Domain Model.

Upon delivery we will create a complete *Unified Modeling Language* (UML) *Class Diagram*[3] as part of the delivery. It will resemble this outline, but some changes are likely to happen over the course of implementation. The only guarantee is that the domain model will be able to store all data needed to make the game adhere to the behavioral specification in this document (and the implied behavior from the *design resource*).

4.2.3.1 User

- Profile picture
- Display name
- Full name
- Gender
- Birthday
- City
- Country
- Email
- Coins
- Trophies : List<Trophy>
- GameHistory : List<Game>

4.2.3.2 Question

- Language
- Category # dropdown (or override, then admin must accept which will create new category)
- Topic # same as category
- Question id (from excel)
- Question text
- Question image
- Difficulty rank
- Hint
- Alternatives
- QuizAidsUsed
- Contributor : User # the person who added the question
- Administrator : User # the person who authorized the question

[3] http://en.wikipedia.org/wiki/Class_diagram

4.2.3.3 Trophy

- Name
- Progress
- Description

4.2.3.4 QuizAid

- Name
- Price

4.2.3.5 Game

- Challenger : User
- Challengees : List<User>
- Name
- Questions : List<Question>
- QuestionsPerPhase # default 6
- Phases # default 3

4.3 The Modo Quiz Backend

The Modo Quiz delivery will contain a simple, yet functional and powerful, backend where players can contribute new Questions for the game to use.

Modo Quiz Backend version 1 (which will be delivered as part of this project) will allow a Modo Quiz registered user to log in to a contributor site; and contribute a *Question*. This will be done using a form with an interface that should feel familiar to the user (every aspect of the Question Domain Model will be editable by creator, except for internal system IDs). If no design is provided for this, Pixelware will employ a modified version of the industry standard CSS template called Bootstrap with a modified color palette to match the Modo Quiz Game. If a design is provided, Pixelware will need to have some say into what can and cannot be done within the timeline of this project.

When a Question is contributed, an admin will then have to certify that the question is (at the very least):

- Not offensive[4]
- Correct
- Classified correctly (with category and topic)

The creator and the admin will both get a list of close matches in order to warn about possible duplicates. The administrator can change any part of the question. Upon certification, the contributor and the admin will be associated to the Question.

4.3.1 Microsoft MVC 5

Models, Views and Controllers make web applications scalable and maintainable.

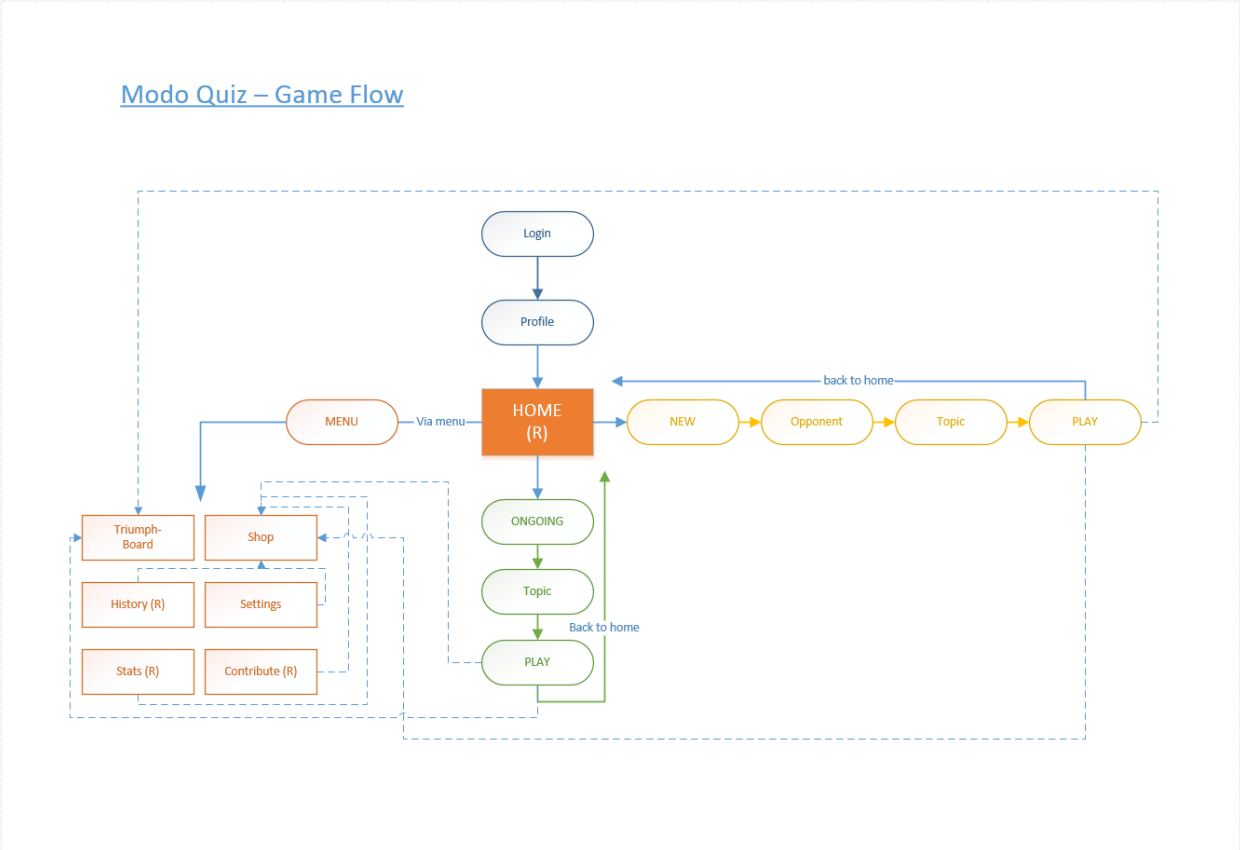
[4] App Store and Google Play both have terms in relation to offensive language, we need to certify that we do not endorse offensive language in any way

The backend will be implemented in http://www.asp.net/mvc/mvc5, which follows the same line of argument as the other technology choices outlined in this document. This framework is an industry standard for making modern and maintainable web applications.

The Model View Controller pattern separates the concerns of the following questions:

- *Model*: How do we get the data we need?
- *View*: How do we show the data to the user?
- *Controller*: What should we show to the user now?

4.4 Overall Quiz Flow Chart



4.5 Design Implementation Notes

Even though there is little interaction design presented, Pixelware will employ experience and create a vivid and exciting experience for the user.

No action should go unnoticed, no reward should pass unseen.

The design outlined in the aforementioned *design resource*[5] does not include any interaction design, nor any animations and transitions. Based on Pixelware's experience with interaction design and games specifically – we will make opinionated suggestions for review. One of the iteration builds will be focused on interaction design feedback, and at this point we can change animations and transitions to match the expectations of Product Owner. Changes in this subject are a timely and costly affair, so we will limit the iterations focused on interaction design to two iterations. Which will result in at most three different designs (our original, revision 1, revision 2) after this an agreement should be made between Pixelware and Product Owner to come to a decision in regards to interaction design, animations and transitions.

4.5.1 Animations

The login screen has an animated logo. We will split the layers of the circular logo and make them rotate in alternating directions (so they align once every spin).

This iconic animation will be repeated as a visual hook for the user to indicate that the game is doing something. This may include retrieving the next question, waiting for payment acceptance etc.

Animations will be made in a reusable fashion, so we can apply any given animation (for instance a shrink, grow, spin etc) to any given visual component.

4.5.2 Screen Transitions

We will make a smooth and natural transition between screens. This is an animation that will be repeated across all screens, so it should be fast, fluid and natural.

There are many sources of inspiration for this, and we will make something that feels good, but maybe also adds a bit flare to the Modo Quiz Game. A subtle change that will differentiate this fun app from the rest.

4.5.3 Interaction

Every click (on a clickable icon or button) should invoke an animation of either size, color or font changes.

This animation will be reused across most graphics that can be clicked; so it should be fast, fluid and natural.

Some buttons will most likely have a special and more exciting animation, such as when using a Modo Quiz, and selecting an *Answer* from a *Question*.

5 Game Design and Game Play

The overall design and flow of the game has been specified and implemented as a mock available at the *design resource*. This resource is agreed to override the previous specification written by Product Owner.

For flow, features and design; the order of precedence is:

- This specification
- The *design resource*
- The previous version of the specification

[5] https://projects.invisionapp.com/share/9B1Q3JVTJ

This means that whenever the three resources are in disagreement - we first check this specification, then the *design resource* and lastly the previous version of the specification.

5.1 Start New Quiz

From the home screen, if "Start New Quiz" is clicked, we'll go to this screen. The Start New Quiz screen has four options:

5.1.1 Challenge A Friend

One-on-one challenge. Follows the Quiz Sequence.

5.1.2 Challenge Friends

Multiple friends can take the same challenge (either found through a text code on Facebook wall, or as an in-app challenge to a friend you already know). Follows the Challenge Quiz Sequence.

5.1.3 Random Challenger

In this mode, the system will try to match you with another player who has also clicked 'random challenger'.

We will set a configurable timeout for this, as a way to tell the user "there is currently no one looking to play against a random challenger".

5.1.4 Find Challenge

This enables you to search for a challenge with an identifier as seen on a friend's Facebook wall. The identifiers should be short and memorable. For example "Challenge by Eirik *Påske 2015*" where the text in *italics* is user defined upon Challenge Quiz creation.

5.2 Quiz

Implementation of the flow and design excellently presented in the design resource

The overall quiz design is well defined in the *design resource*. Therefore, that resource will serve as the overall design and flow of how to start a quiz.

5.2.1 Quiz Sequence

This is the main game page. The description below illustrates the game flow when the player is answering questions. For reference, the rough sequence is:

- Select game mode
    - Random challenger
    - Challenge one friend
    - Challenge many friends
    - Find challenge
- Select opponent(s)
- Choose category and topic
    - NEW: Search through topics via button on the Modo icon in the middle of the page
      Upon click, the lower section (with topics) will slide up, and the list of topics will contain everything that matches the search string entered by the user.
- Start Quiz Sequence (outlined below)

The sequence of a game is as follows:

- A round number is display for 1 second (“Round 1”, “Round 2”, etc.)
- The question and answers is displayed.
- A timer counts down from 10 seconds as soon as the answers are displayed
- When the player’s choose/ tap one of the answers this should be visible as “selected” and indicate a correct or wrong answer.
- If the player have not chosen an answer when there are three seconds left a visible element should be displayed indicating “time is running out”
- If answer is correct
    - Animation of score
- If answer is wrong or time is up
    - When the timer hits 0 a message should say “Time is up”. This is shown for 5 seconds.
- End with a 5 second timer to allow for Modo Shop interaction (if shop is opened, the timer stops and the game continues after shop is closed).

5.2.2 Quiz Aid

Quiz aid is in-game help that makes it easier for the player to select the correct answer. These quiz aid are small icons that are clickable when the player is answering quiz questions. There are three quiz aid alternatives selectable from the quiz game page:

50/50 - When this option is selected, 50% of the wrong answers are removed (made inactive)

More time - When this option is selected, the countdown timer adds 15 seconds.

New question - When this option is selected the current question is skipped and replaced with a new question.

Hint - When this option is selected a text hint is given to the player. This hint is relevant to the question to make it easier to answer.

Quiz aids will be acquired when pressed, and the account will withdraw coins at this point. As such, aids are only clickable when the user has Coins. If the user clicked an aid (in an attempt to buy it) but had insufficient coins, the store will open at the first available time (after timer is out) with an explanation that it’s opened because the player needs more coins in order to complete the intent to get an Aid.

If coins are available, then the Aid icon will play an animation and take effect, then it is set to not-clickable state (grayed out, or transparency).

5.2.3 Challenge Quiz

Modifications of the game flow - many against many - Friends against friends.

When a Game Domain Object has more than one Challengee then that game is considered a Challenge Quiz.

A challenge quiz should be postable to the user’s Facebook wall. This will create an entry on the wall with the identifying name for the quiz. Other friends who join through Facebook will then have to open the app, and

search up a challenge quiz. The user will then use the Find Challenge option in the “Choose your game” screen from the *design resource*.

The only differences from a regular Quiz flow is:

5.2.3.1 Challenge Quiz Sequence

In a challenge quiz, there is only one phase. This means that every user answers all the questions in one go. There is no sync time to wait for the other users to finish.

5.2.3.2 Challenge Score Board

Challenge Quiz will show a High Score List (in accordance with the *design resource*) instead of the classic Scoreboard. When clicking of an entry in the High Score List a Scoreboard will be shown between yourself and the person that was clicked.

In the list, the entry for You is highlighted, but it cannot be clicked.