HÁSKÓLI ÍSLANDS

Final Report
BS in Computer Science

Svavar Árni Halldórsson

Due 5. June 2015

CONTENTS

1 Introduction

This report describes the preparation and execution of my final project for my B.S. in computer science. The project initiated from my interest in web development as well as fascination with the card game Magic the Gathering. The goal of this project was to study different methods in developing websites, setting up databases and overall make a functional and user friendly site. This report describes a simple website where a user can further immerse himself in searching, browsing and studying all the cards available in the game. The premise of what the system can do for the user is a front for a study of different web development techniques and a number of them are used in developing the system itself although the system itself will eventually be made with ASP.NET MVC using Visual Studios. The first part of this report describes the requirement analysis, followed by the preliminary design phase, ending with an overview of the production process itself. The last parts of this report contain the conclusion, word definitions and references.

2 About the System

Magic was the first trading card game produced and it continues to thrive, with approximately twelve million players as of 2011. Magic can be played by two or more players each using a deck of 60+ printed cards or a deck of virtual cards through the Internet Based Magic: The Gathering Online. An organized tournament system and a community of professional Magic players has developed, as has a secondary market for Magic cards. Magic cards can be valuable due to their rarity and utility in gameplay (further information).

This system, called Magic Builder, is meant as a reference guide for Magic players to search and browse cards, view detail information about them and build deck of cards. Magic Builder will allow a user to register an account on favorite cards and build decks on his profile. The user can also view information about Magic and the different card sets that have been published.

3 Competition

3.1 MTG VAULT

This website is an extensive library of Magic The Gathering information. The website mostly centers around viewing decks from other users and sharing your own. The card search is advanced and gives a lot of options. Overall the website works pretty well and is easy to use. The biggest flaws I noticed while researching MtgVault concern the look and feel of the site, it looks dated and unresponsive, when using the card search or clicking items the page executes a full reload and most importantly the page is not scalable and therefore only works on desktop computers. (mtg vault).

3.2 MTG VAULT

Magic builder will feature a lot of the same functions as mtgVault such as searching for cards and building decks, however my goal is to make Magic builder simple, fast and available on all screen sizes.

4 The Average User

perhaps a text about the purpose of this section...

4.1 Background

The average system user will be in the age range 20 - 40, any gender and have an elementary education or more. The user will most likely have good computer experience and have minimal disabilities, although poor eyesight could be a factor.

4.2 System Use

The average user will use the system any day of the week, with possible spikes on the weekends and average use will probably be around once per week. There should be no training necessary and the user should have a very positive attitude towards the system.

4.3 Environment

- Technical Environment
  The system will be used on desktop computers, laptops, tablets or other similar devices.
- Real Environment
  The system will be used in any environment, i.e. at home, work, in school, or other institutions and places.

4.4 Main Goals

The average user will use the system to make an account, search and browse cards and make decks. Possibly connect with other players and share ideas.

5 Requirements

The following is a list of functionalities that the user can do within the system. They are ordered by priority A, B or C. Requirements with priority A are essential, B are features that are important but not critical and C are nice to have features. The numbers in the user stories column refer to user stories with the corresponding numbers.

5.1 Functional Requirements

Nr.	Description	User Stories	Priority (a/b/c)	Status
1	User must be able to register an account.	1	a	
2	User must be able to login and logout of his account.	2,3	a	
3	User must be able to search and browse cards.	4	a	
4	User must be able to select cards and view detailed a information about each on	5	a	
5	User must be able to favorite cards and store them in their own profile.	6	a	
6	User should be able to filter cards by number of variables when searching.	7	a	
7	User should be able to create decks of cards.	8	a	
8	User should be able to view statistics about their decks.	9	b	
9	User should be able to add other users as friends.	10	b	
10	User should be able to share decks with other users.	11	b	
11	User should be able to decide whether their decks are private or public.	12	b	
12	User should be able to comment on their friends decks.	13	b	
13	User should be able to post questions about game decisions to other users.	14	c	
14	User should be able to register and login with Facebook and Google accounts.	15	c	

5.2 Non-functional Requirements

Nr.	Description	Priority (a/b/c)	Status
1	Operational requirements: Automated help/error messages when the user makes a mistake.	b	
2	Should take into account the most widely known web accessibility standards (WCAG 1.0, WCAG 2.0 etc	b	
3	Should support most mobile resolutions and standards.	b	
4	Should account for common disabilities such as poor eyesight.	b	
5	The website should be able to handle 1.000 visitors a at the same time.	b	

6 User Stories

The user stories describe the requirements in an everyday language. They also suggest a reason for wanting to be able to complete the tasks. They are meant to be simple and easy to understand and should make a good checklist when implementing the system. The user stories are ordered by priority, with the red boxes containing the highest priority stories, the yellow the intermediate ones and the green the lowest priority. Although the yellow and green user stories might not be implemented they are nevertheless included in case the organization of the project, or the priority of the user stories, changes in any way.

Number	User stories
1	As a user I want to register an account so that I can use the services provided by the system. *The user inputs a username, password and an email.*
2	As a user I want to login so that I can use services specific to my account. *The user inputs his username and his password.*
3	As a user I want to logout of my account so my account is no longer available. *The user clicks the logout button.*
4	As a user I want to search for cards to be able to browse the cards available. *The user inputs a search string.*
5	As a user I want to be able to view more detailed information about a specific card to know more about it *The user selects a card.*
6	As a user I want to favorite a card so I can browse my favorite cards. *The user selects a card and presses 'favorite'.*
7	As a user I want to be able to use filters when searching for cards to make it easier to find cards of specific type. *The user adjusts the appropriate filters for his search.*
8	As a user a want to be able to create my own decks so I can add cards to them and store them to view later. *The user presses a button to create a deck and selects cards to add to the deck.*
9	As a user I want to be able to view statistics about my decks so I can better see if they correctly made or not. *The user selects a deck.*
10	As a user I want to add other users as friends so I can connect and share with them. *The user adds users by username.*
11	As a user I want to be able to share my decks with other users and see their decks for comparison. *The user automatically shares public decks.*
12	As a user I want to be able to make my decks either private or public so other user can view some but not all my decks. *The user selects either the private or public option in the deck view.*
13	As a user I want to comment on the decks from other users to voice my opinion. *The user writes a comment when viewing decks and posts it.*
14	As a user I want to post questions about game rulings to other users so I can get their opinion. *The user inputs a username and writes a question to send to that user.*
15	As a user I want to be able to register and login with my Facebook and/or Google account for convenience. *The user presses the login with... or register with... buttons.*

7 Usability Goals

I decided to set a few target goals for new users, if more than 80% of them are able to perform the tasks with the given constraints then the system's usability is satisfactory, but of course experienced users should easily be able to outperform the constraints.

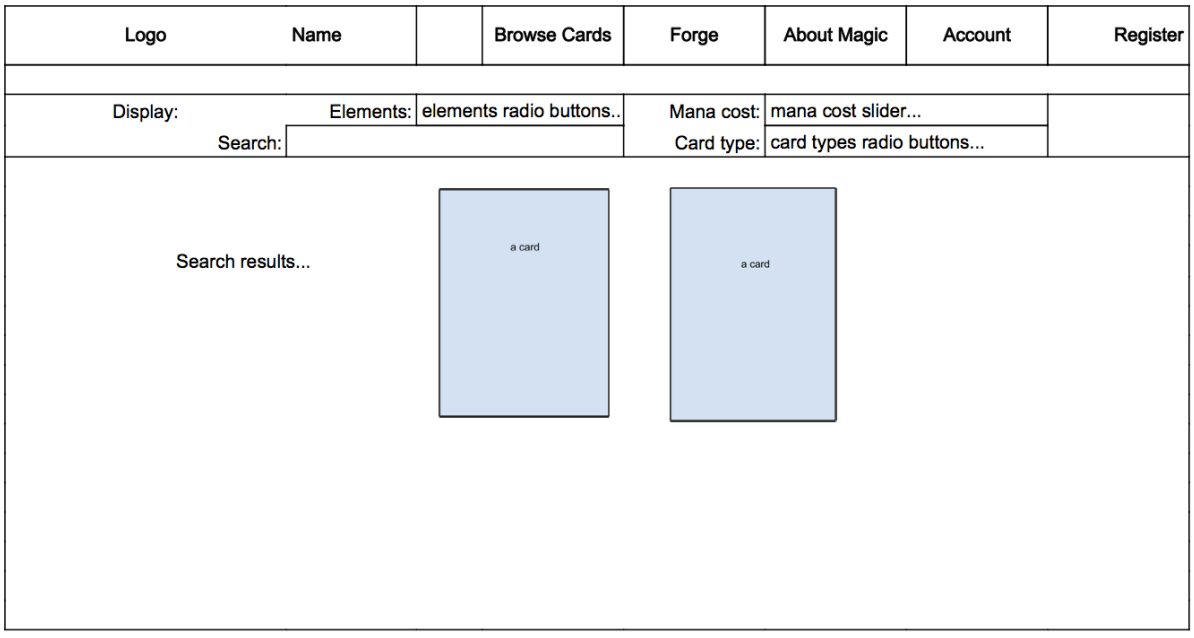
- The user should be able to register an account in under 2 minutes.
- The user should be able to login in less than 20 seconds.
- The user should be able to search for a card in less than a minute.
- The user should be able to create a deck in less than a minute.
- The user should be able to log out in less than 20 seconds.
- The user should be able to get to the homepage in one click from any page.

8 Prototypes

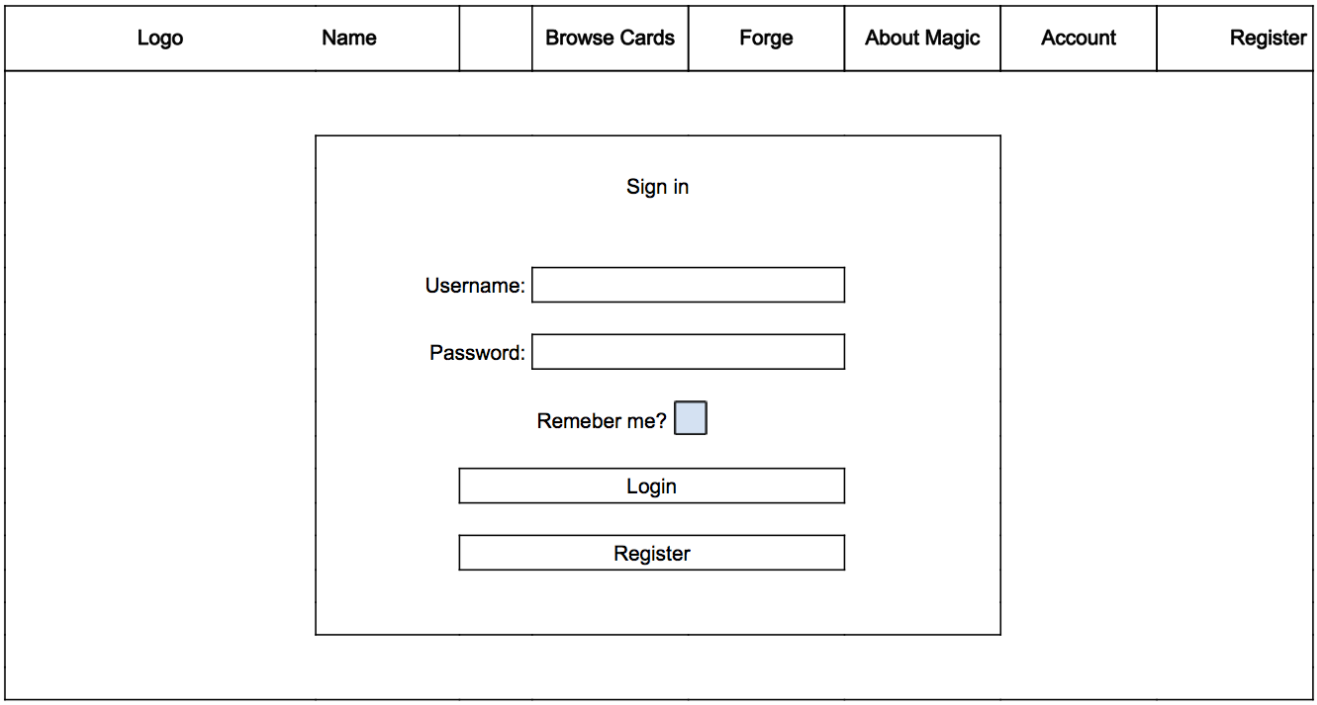
For the first draft layout of the website I made a few wireframes that give a good overview of the website. I also made some graphic design prototypes for reference but they do not necessarily portray the final look of the website since functionality will always be the highest priority.

8.1 Wireframes

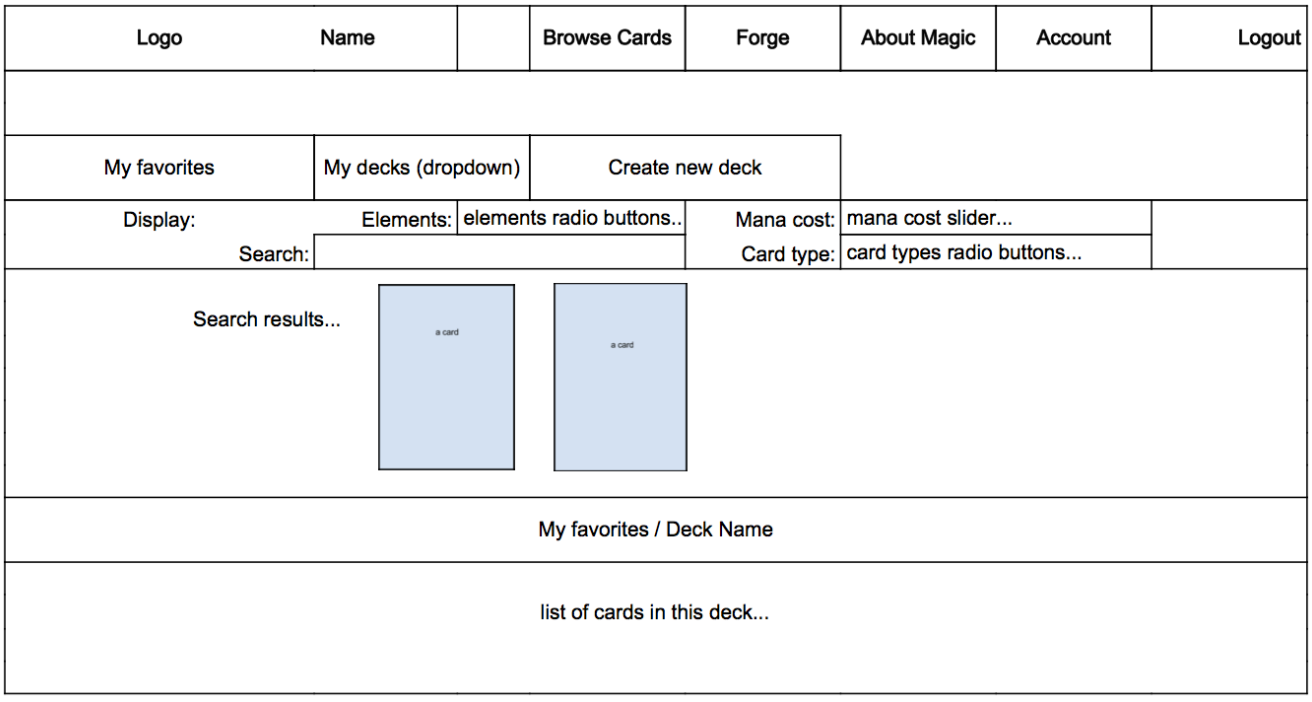
The first wireframe describes the front page of the website. It has the same layout as all other pages on the site but if the user is not logged in he will be prompted with the register option in the top right corner of the page but the logout option otherwise.



The login page will also have the same layout as the other pages but display a login form where the user can insert his login information or press register if the user doesn't have an account. The register page is the same as the login page except it displays a register form.



The forge is only available for logged in user. It has the same layout as other pages but displays the logout option in the top right corner. It gives the user the options to view their favorite cards, create new decks or navigate between already made decks and has the same search as the front page where the user can search for cards and add them to their decks. The current deck is then displayed below.

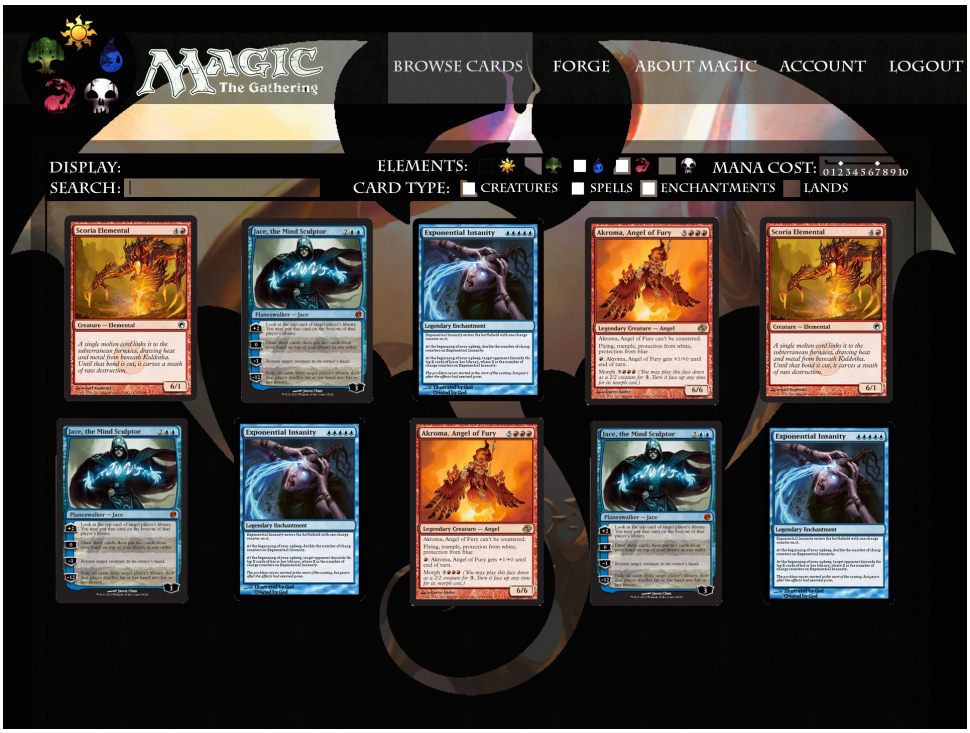


The about magic page will have the same layout as the others with either logout or register in the top right corner, depending on whether the user is logged in or not. The main area will then display a tab bar that a user can use to exchange the information displayed.

Logo	Name		Browse Cards	Forge	About Magic	Account	Register
TAB BAR							
	Basic information		The rules	The sets	...		
text...							

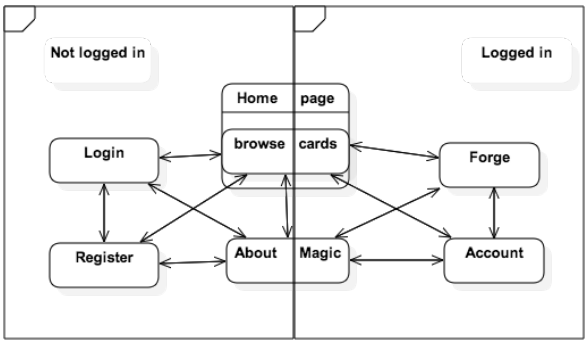
8.2 Graphic Design

These prototypes were mostly made as a reference and were only meant to portray the final design if time allows.



9 Navigation Diagram

This diagram is meant as an overview of the navigation throughout the system. The diagram describes how the user navigates between pages and the two frames shows the difference in navigation when the user is logged in and when he is not.



10 The Database

Magic builder uses two different databases. All card data is pulled from an online database http://api.mtgdb.info/ that serves json data through http GET requests. The database is accessed through a C# driver that gives access to a library of methods that can be used to query the server for data. User data is stored in a SQL database kept locally on the server.

11 Description of Technical Environment

During the programming phase I will explore different ways of developing the system. The following tools have been and will be used during preparation and production of the system.

- All the code behind the system will be written in Visual Studio 2013 in the MVC5 environment.
- For assistance I will use Sublime Text 2 while coding.
- The system will be designed primarily with Google Chrome making use of the built in development tools.
- For javascript debugging I will use jslint.com as well as the development tools in Google Chrome.
- Source control will be provided with git.
- For source control hosting, I will use github.com, and connect to github through the team explorer in Visual Studio.
- I will use bootstrap to make visualisation changes to the website, but also to ensure accessibility and responsiveness of the website.
- The prototypes are implemented in Google Spreadsheet and Photoshop.
- The navigation diagram and class diagram were made in StarUML.
- Google Docs and LaTeX was used to make the reports.

12 Programming Rules

I decided to put forth a few programming rules to follow, for better consistency and readability of code. First I looked at the XHTML standard for HTML and some CSS coding standards. I looked into Microsoft traditions for C# and explored common traditions for JavaScript code. I adjusted the rules for my purposes and they are listed below with a few examples for clarity.

12.1 HTML

- All attributes, events, and tags must be written in lower case.
- All elements must be closed.
- The value assigned to an attribute must be enclosed in quotes.
- All elements must be properly nested.
- Attribute minimization is not allowed (selected must be selected='selected').
- Comment the code.
- Declare the correct doctype.
- Never use inline styles.
- Place external CSS files within the head & put javascript files at the bottom.
- Never use inline javascript.
- Use h1 - h6 tags properly.
- Use labels for all form boxes.
- Use unordered list for navigation.
- Always place the alt attribute for images.

Listing 1: ProgrammingRules/HTMLrules.html

```
<!--
   The follwowing codes shows an example of these rules.
   Lowercase on attributes, events and tags, elements must be properly nested
   & closed and value assigned to an attribute must be enclosed in quotes.
 -->

<table>
  <tr>
    <td>
     The three muskateers
    </td>
  </tr>
</table>
 <ul>
  <li>About us</li>
  <li>Lists</li>
  <li> <a href="www.facebook.com" title="My Page">My Facebook</a> </li>
</ul>
```

12.2 CSS

- When grouping selectors, keep individual selectors to a single line.
- Use classes in selectors but avoid id's for better reusability of code.
- Include one space before the opening brace of declaration blocks for legibility.
- Place closing braces of declaration blocks on a new line.
- Selectors names should start with a lowercase letter.
- Id names should have underscores: #user_image.
- Class names should have hyphens: .profile-image.
- Only use English words.
- Group related items together with comments.
- Class and id names should be descriptive.
- CSS files should be organized using flags.
- Avoid shorthand CSS.
- All CSS should be in an external stylesheet, no inline styles.
- HTML first then CSS.
- Comment the CSS code.
- Use bootstrap whenever possible.
- Avoid extra selectors.

Listing 2: ProgrammingRules/CSSrules.css

```
/*
  The follwowing codes shows an example of these rules.
  Selectors in single line, classes in
  lowercase with dashes & right use of braces.
*/

.selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}

.class2 {

  /* TODO */
}
```

12.3 C#

- Only use English words.
- When code is being indented, the 'tab' button should be used (in Visual Studio 2013 'tab' is saved as four spaces).
- Use predefined type names instead of system type names like Int32, String, Single, UInt64, etc.
- Use camelCasing for method arguments and local variables.
- Use PascalCasing for class names and method names.
- Use noun or noun phrases to name a class.
- Vertically align curly brackets.
- Declare all member variables at the top of a class, with static variables at the very top.
- Commenting Conventions:
  - Place the comment on a separate line, not at the end of a line of code.
  - Begin comment text with an uppercase letter.
  - End comment text with a period.
  - Should be written in English.
- Always specify [HttpPost] or [HttpGet].
- Use a try-catch statement for most exception handling.
- Avoid using abbreviations in names.
- Do not use underscores in identifiers.

- Singular names for enums.
- Do not use the word enum in enum names.
- Do not push code that doesn't compile!
- Avoid complex expressions.
- Consider warnings as errors.
- Set a region around code that is related to one another.
- Use implicitly typed local variables when the variable type is clear.
- Use explicitly typed local variables when the variable type is not clear.

Listing 3: ProgrammingRules/csrules.cs

```
1
2  /*
3  The follwowing code shows an example of these rules.
4  Comment conventions, camelCasing, predefined type names, PascalCasing
5  and aligned curly brackets.
6  */
7
8  public class UserLog
9  {
10     public void Add(LogEvent logEvent)
11     {
12         int itemCount = logEvent.Items.Count;
13         var currentYear = 2015;
14         // ...
15     }
16 }
17
18 // Declare all member variables at the top of a class,
19 // with static variables at the very top.
20
21 public class Account
22 {
23     public static string BankName;
24     public static decimal Reserves;
25
26     public string Number {get; set;}
27     public DateTime DateOpened {get; set;}
28     public DateTime DateClosed {get; set;}
29     public decimal Balance {get; set;}
30
31     // Constructor
32     public Account()
33     {
34         // TODO
35     }
36 }
```

12.4 JavaScript

- In general, use camelCasing for functions, variables and methods. Use PascalCasing for classes and enums. Constant values should be in all caps.
- Always use semicolons.
- Always use 'var' while declaring variables.
- Vertically align curly brackets.
- Declare variables outside of the 'for' statement.
- Never pass a string to SetInterval and SetTimeOut. Instead, pass a function name.
- Javascript files should be stored in an external file (not inline).
- The code should be correctly indented.
- Line length should not exceed 80 characters.
- Variables should be declared before they are used.
- Use === and !== in comparisons instead of == and !=.
- Comment your code.

Listing 4: ProgrammingRules/javascriptrules.js

```
1  // The follwowing code shows an example of these rules.
2  // Declaration of variables & passing a function name.
3
4  var container = document.getElementById('container');
5  for(var i = 0, len = someArray.length; i < len;  i++) {
6     container.innerHtml += 'my number: ' + i;
7     console.log(i);
8  }
9
10 setInterval(someFunction, 3000);
```

13 The Production Process

13.1 How the overall production fared

In the first stages of development the core functionality such as logic and view rendering will be done on the server but when I've finished building the core functions on the server and the user interface is complete I plan to move the view rendering to the browser. To make the page faster and enable asynchronous loading of page elements I will use Javascript AJAX calls to send and receive data from the server. By building the system incrementally I can ensure that the page will work on a large range of devices, including ones with Javascript turned off.

13.2 What went wrong and how it can be done better next time

I initially started building the system using AngularJS along with an online database www.firebase.com. I had some trouble getting things to work, especially concerning the database connections. In the end i decided that AngularJS framework was too large and complex for the simple functionality of Magic Builder so i abandoned that approach in favor of using asp.net MVC5. Having had no experience with both of those systems I spent a lot of time doing tutorials and reading about how things work. I also started from scratch a few times in order to eliminate design errors while building the database and models. Getting the initial system to build, connecting the database and enable-ing user accounts can fail in a number of ways and errors can pop up that can be hard to handle.

14 Requirements Outcome

14.1 What Requirements were met

14.2 What Requirements where left out and why

14.3 What Requirements where not finished and why

15 Usability Goals Outcome

16 Changes from Initial Analysis and Design

16.1 Interface Design

16.2 Class Diagram

17 Word Definitions

- ...

18 References