



HÁSKÓLI ÍSLANDS

---

# Final Report

## BS in Computer Science

---

Svavar Árni Halldórsson

Due 5. June 2015

## CONTENTS

<b>1 Introduction</b>	<b>4</b>
<b>2 About the System</b>	<b>4</b>
<b>3 Competition</b>	<b>4</b>
3.1 mtg vault . . . . .	4
3.2 What can be simulated . . . . .	4
<b>4 The Average User</b>	<b>5</b>
4.1 Background . . . . .	5
4.2 System Use . . . . .	5
4.3 Environment . . . . .	5
4.4 Main Goals . . . . .	5
<b>5 Requirements</b>	<b>6</b>
5.1 Functional Requirements . . . . .	6
5.2 Non-functional Requirements . . . . .	6
<b>6 User Stories</b>	<b>7</b>
<b>7 Usability Goals</b>	<b>8</b>
<b>8 Prototypes</b>	<b>8</b>
8.1 Wireframes . . . . .	8
8.2 Graphic Design . . . . .	10
<b>9 Navigation Diagram</b>	<b>11</b>
<b>10 The Database</b>	<b>11</b>
<b>11 Description of Technical Environment</b>	<b>11</b>
<b>12 Programming Rules</b>	<b>12</b>
12.1 HTML . . . . .	12
12.2 CSS . . . . .	13
12.3 C# . . . . .	14
12.4 JavaScript . . . . .	16
<b>13 The Production Process</b>	<b>17</b>
13.1 How the overall production fared . . . . .	17
13.2 What went wrong and how it can be done better next time . . . . .	18
<b>14 Requirements Outcome</b>	<b>18</b>
14.1 What Requirements were met . . . . .	19
14.2 What Requirements where not finished and why . . . . .	19
<b>15 Usability Goals Outcome</b>	<b>20</b>
<b>16 Changes from Initial Analysis and Design</b>	<b>20</b>
16.1 Interface Design . . . . .	20

16.2 The Navigation Diagram . . . . .	20
<b>17 What was Learned</b>	<b>20</b>
<b>18 Conclusion</b>	<b>21</b>
<b>19 References</b>	<b>22</b>

## 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 a fascination with the card game Magic the Gathering. The goal of this project is 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 system itself will use Magic the Gathering as a premise but the most important thing will be a prototype mini version of a application, with requirement analysis and usability studies and the study of ASP.NET MVC, using Visual Studio. 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 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 and build decks on his own profile.

## 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 WHAT CAN BE SIMULATED

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

To better visualize how the system would be used, I felt that it was important to analyze a typical user of a system of this kind.

### 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 should be able to register an account.	1	a	
2	User should be able to login and logout of his account.	2,3	a	
3	User should be able to search and browse cards.	4	a	
4	User should be able to select cards and view detailed information about each one.	5	a	
5	User should be able to filter cards by number of variables when searching.	6	a	
6	User should be able to create decks of cards.	7	a	
7	User should be able to view statistics about their decks.	8	b	
8	User should be able to add other users as friends.	9	b	
9	User should be able to share decks with other users.	10	b	
10	User should be able to decide whether their decks are private or public.	11	b	
11	User should be able to comment on their friends decks.	12	b	
12	User should be able to post questions about game decisions to other users.	13	c	
13	User should be able to register and login with Facebook and Google accounts.	14	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 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. <i>The user inputs a username, password and an email.</i>
2	As a user I want to login so that I can use services specific to my account. <i>The user inputs his username and his password.</i>
3	As a user I want to logout of my account so my account is no longer available. <i>The user clicks the logout button.</i>
4	As a user I want to search for cards to be able to browse the cards available. <i>The user inputs a search string.</i>
5	As a user I want to be able to view more detailed information about a specific card to know more about it <i>The user selects a card.</i>
6	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. <i>The user adjusts the appropriate filters for his search.</i>
7	As a user I want to be able to create my own decks so I can add cards to them and store them to view later. <i>The user presses a button to create a deck and selects cards to add to the deck.</i>
8	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. <i>The user selects a deck.</i>
9	As a user I want to add other users as friends so I can connect and share with them. <i>The user adds users by username.</i>
10	As a user I want to be able to share my decks with other users and see their decks for comparison. <i>The user automatically shares public decks.</i>
11	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. <i>The user selects either the private or public option in the deck view.</i>
12	As a user I want to comment on the decks from other users to voice my opinion. <i>The user writes a comment when viewing decks and posts it.</i>
13	As a user I want to post questions about game rulings to other users so I can get their opinion. <i>The user inputs a username and writes a question to send to that user.</i>
14	As a user I want to be able to register and login with my Facebook and/or Google account for convenience. <i>The user presses the login with... or register with... buttons.</i>

## 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.

Logo	Name		Browse Cards	Forge	About Magic	Account	Register
Display:	Elements:	elements radio buttons...		Mana cost:	mana cost slider...		
Search:				Card type:	card types radio buttons...		
Search results...			a card	a card			

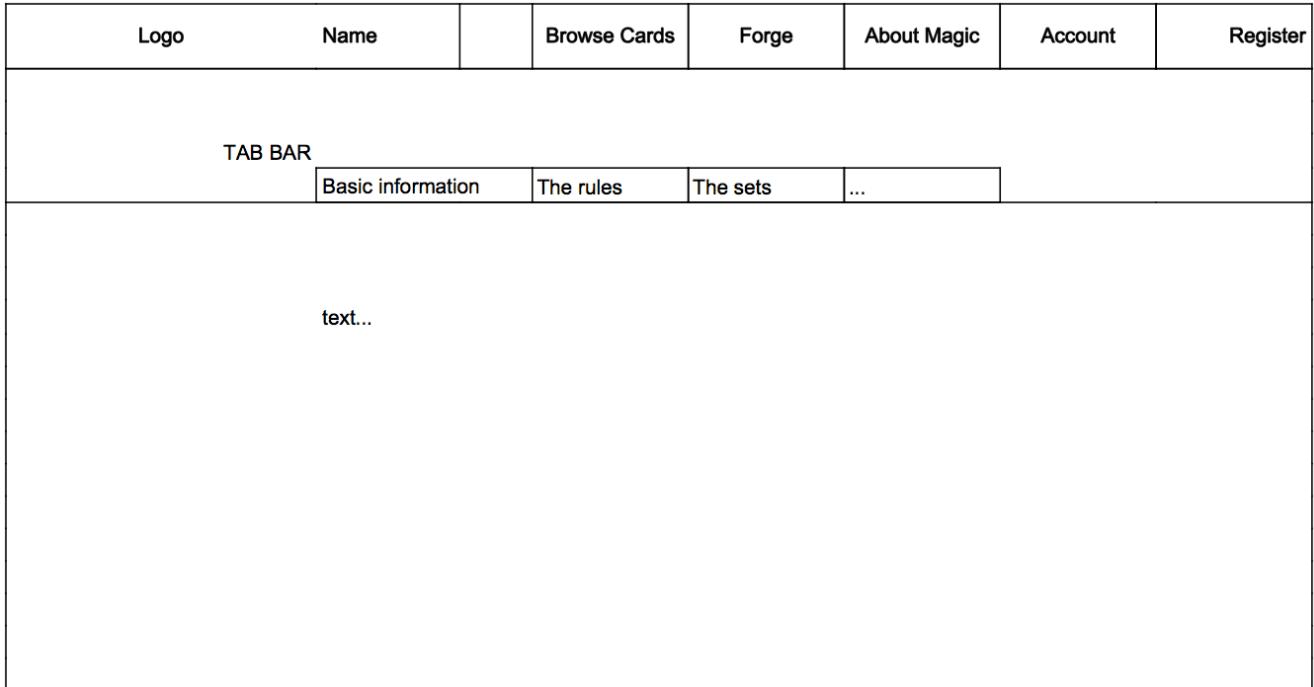
The login page will also have the same layout as the other pages but display a login form where the user can insert his login information or press register if the user doesn't have an account. The register page is the same as the login page except it displays a register form.

Logo	Name		Browse Cards	Forge	About Magic	Account	Register
<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Sign in</p> <p>Username: <input type="text"/></p> <p>Password: <input type="password"/></p> <p><input checked="" type="checkbox"/> Remember me?</p> <p><input type="button" value="Login"/></p> <p><input type="button" value="Register"/></p> </div>							

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.

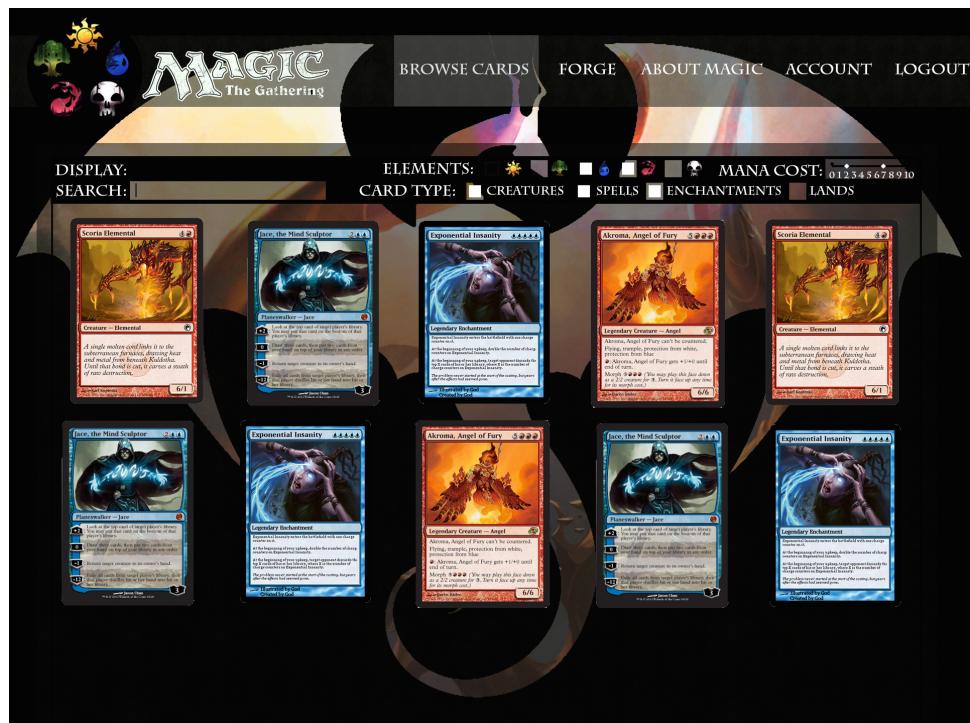
Logo	Name		Browse Cards	Forge	About Magic	Account	Logout
<div style="display: flex; justify-content: space-between;"> <span>My favorites</span> <span>My decks (dropdown)</span> <span>Create new deck</span> </div> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <span>Display:</span> <span>Elements: <input type="radio"/> elements radio buttons..</span> <span>Mana cost: <input type="range"/> mana cost slider...</span> </div> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <span>Search:</span> <input type="text"/> <span>Card type: <input type="radio"/> card types radio buttons...</span> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <span>Search results...</span> <div style="border: 1px solid black; width: 100px; height: 150px; background-color: #e0e0ff; display: flex; align-items: center; justify-content: center;">a card</div> <div style="border: 1px solid black; width: 100px; height: 150px; background-color: #e0e0ff; display: flex; align-items: center; justify-content: center;">a card</div> </div> <div style="text-align: center; margin-top: 20px;"> <span>My favorites / Deck Name</span> </div> <div style="text-align: center; margin-top: 20px;"> <span>list of cards in this deck...</span> </div>							

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.



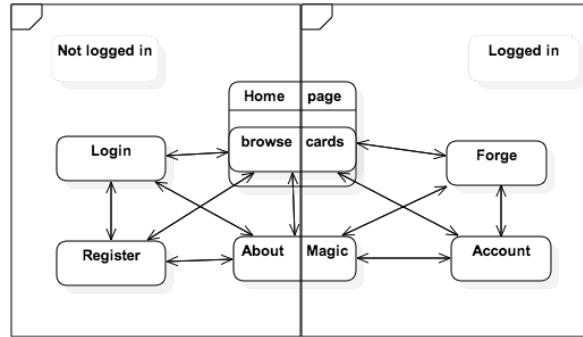
## 8.2 GRAPHIC DESIGN

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 was made in StarUML.
- Google Docs and L<sup>A</sup>T<sub>E</sub>X 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

```
1 <!--
2   The following codes shows an example of these rules.
3   Lowercase on attributes, events and tags, elements must be properly nested
4   & closed and value assigned to an attribute must be enclosed in quotes.
5   -->
6
7 <table>
8   <tr>
9     <td>
10    The three musketeers
11   </td>
12 </tr>
13 </table>
14 <ul>
15   <li>About us</li>
16   <li>Lists</li>
17   <li> <a href="www.facebook.com" title="My Page">My Facebook</a> </li>
18 </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

```
1 /*  
2  The following codes shows an example of these rules.  
3  Selectors in single line, classes in  
4  lowercase with dashes & right use of braces.  
5 */  
6  
7 .selector,  
8 .selector-secondary,  
9 .selector[type="text"] {  
10 padding: 15px;  
11 margin-bottom: 15px;  
12 background-color: rgba(0,0,0,.5);  
13 box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
14 }  
15  
16 .class2 {  
17     /* TODO */  
18 }  
19 }
```

### 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 following 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 following 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 setInteval(someFunction, 3000);
```

## 13 THE PRODUCTION PROCESS

During the production process I worked with many tools for “bootstrapping” or creating new web apps that come prepackaged with everything a particular framework needs. While working with Angular I mostly used the <http://yeoman.io> generator which can bootstrap an Angular project with build tools included such as automated tests, package manager, source control and automated deployment. After getting all of that working and setting up a webapp with some basic functionality such as a card search and some css styling I set off to get the database working, with user and card relations as well as authentication. That is where I had the most problems, getting the database to store card data and connecting that to users didn't work. After some time I decided to move over to ASP.NET MVC5 and started from scratch there.

When I started on ASP.NET I decided to use the code first approach when building databases. I built some model classes for card and deck data and used code-first migrations to generate the database schema straight from the code. After dropping the database and redesigning the model classes a few times I had a working model of my data and a database to connect to. With the database layer ready I created a Repository which contains all the methods which manipulate the database, that way the database layer is completely separated from the logic and the views. By using a clear separation of the three layers replacing a layer becomes easy. After that I needed a way to pass more than one model into my views, such as Magic cards, decks and user data. I created a viewModel class that contains all the data my views need. With that taken care of I created controller endpoints for urls such as /createDeck, /search etc. with corresponding views. For easier styling I added bootstrapLess which is an extended .less version of the popular Bootstrap UI css style framework. Less allows the designer to think more like a programmer while adding styling to pages. Less uses functions, variables and selector nesting for easier development. When the project is built the .less files are converted to .css. Finally, after the core functionality regarding search and decks was completed I added a details pop up view for individual cards and did some refactoring on the code, such as putting repeated html into partial views to reduce clutter in the Razor view pages. I added filters to the search to be able to narrow down card choices and then I deployed the app to a cloud based server hosted with Microsoft.

### 13.1 HOW THE OVERALL PRODUCTION FARED

I initially started building the system using AngularJS along with an online database <www.firebaseio.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 neither 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 enabling user accounts was challenging but after a few tries I got things working.

## 13.2 WHAT WENT WRONG AND HOW IT CAN BE DONE BETTER NEXT TIME

While working on the project I spent a lot of time on setting up new Angular and ASP.NET MVC5 web apps, configuring databases and managing authentication. On my next project I know I will be much better familiarized with the frameworks which will cause me to design the inner workings of the system in accordance with the frameworks. I can than hopefully avoid design errors which led to me starting from scratch a few times. At the start of the project while I worked with AngularJS I had some problems setting up a database connections with [www.firebaseio.com](http://www.firebaseio.com), in the future I will be wary of using bleeding edge technologies such as the Angular and Firebase wrapper AngularFire, lack of examples and documentation proved problematic. After I connected to the online Magic the gathering database with the MtgDb.driver.dll available on <http://api.mtgdb.info/> I noticed that I was getting parse errors on some JSON data so that the search feature was completely broken, fortunately someone had posted details about the error on the projects [github site](#). After cloning the project i was able to locate the problematic code, fix it and rebuild the .dll file. Everything worked like a charm after that.

## 14 REQUIREMENTS OUTCOME

Nr.	Description	User Stories	Priority (a/b/c)	Status
1	User should be able to register an account.	1	a	Finished
2	User should be able to login and logout of his account.	2,3	a	Finished
3	User should be able to search and browse cards.	4	a	Finished
4	User should be able to select cards and view detailed information about each one.	5	a	Finished
5	User should be able to filter cards by number of variables when searching.	6	a	Finished
6	User should be able to create decks of cards.	7	a	Finished
7	User should be able to view statistics about their decks.	8	b	Not finished
8	User should be able to add other users as friends.	9	b	Not finished
9	User should be able to share decks with other users.	10	b	Not finished
10	User should be able to decide whether their decks are private or public.	11	b	Not finished
11	User should be able to comment on their friends decks.	12	b	Not finished
12	User should be able to post questions about game decisions to other users.	13	c	Not finished
13	User should be able to register and login with Facebook and Google accounts.	14	c	Partially done

Nr.	Description	Priority (a/b/c)	Status
1	Operational requirements: Automated help/error messages when the user makes a mistake.	b	Not finished
2	Should take into account the most widely known web accessibility standards (WCAG 1.0, WCAG 2.0 etc	b	Finished
3	Should support most mobile resolutions and standards.	b	Finished
4	Should account for common disabilities such as poor eyesight.	b	Finished
5	The website should be able to handle 1.000 visitors at the same time.	b	Untested

## 14.1 WHAT REQUIREMENTS WERE MET

The main goal of this project was to finish the fundamental requirements and during the implementation learn more about the overall setup of a web application. The initial step was to allow users to register an account, login and log out. The MVC ASP.NET framework in Visual Studio offers a premade authentication which I could use as a base for MagicBuilder. The built in functionality for the project did help in some areas but not in all. Nevertheless, I decided to try to use as much of it as I could since it offers a lot of perks, such as input validation and security for database records. The account requirements were therefore quite straightforward to implement although I needed some research to make my own tables connect with the user table as to make sure that each user had separate decks and saved cards.

The second part of the project became the search. I started by making a simple search where the user could input a search input string and get all cards that contained that particular string in their name. The user is presented with no more than 100 cards that he can browse through. When I got further into the project and got more comfortable with the online database I added a few filters to the search. The filters allow the user to narrow down their search a bit which is much more user friendly. I found it unnecessary to add more filters since the project is only meant as a prototype.

Each card presented by the search also includes a detailed information about itself. Since the cards themselves are sometimes hard to read I felt it was important that the user could view these details in a larger window. I added a flip over effect on the cards that presents a button that the user can press for more details about the card. It provides a nice visual as well as a simple interface for the user. When the user presses the button a window is displayed while the surroundings are blurred so the window becomes very clear and the user is not interrupted by other material. The window displays the card image a bit larger and all relevant data related to that card.

The main reason for making an account based application was to allow the user to create his personal decks and add cards to them. It took a bit of time to get my database tables to work with the user database as well as the online database but when the connection had been properly made I made a tab bar where the user can create and view his decks. I also created a rename and delete deck functions which I felt was important for the usability of the system. The 'add to deck' and 'remove from deck' buttons were placed on the backside of the cards with the 'detail' button for good consistency.

The non-functional requirements were not a high priority. Nevertheless, I tried to keep them in mind as I worked and managed to finish a few of them.

Overall I am very pleased with the progress of the high priority requirements. I feel that the system now offers a good base for a larger project and it should be great foundation to built upon.

## 14.2 WHAT REQUIREMENTS WHERE NOT FINISHED AND WHY

Since this project was meant as a study of basic web development, I decided to focus on finishing the highest priority requirements and try to implement them so as to make the system a good mini version of a system of this type. All lower priority requirements were therefore left out, i.e. all B and C requirements, since all extra time was spent on optimizing the high priority requirements rather than adding more features.

All non-functional requirements were B or moderate priority. Even so, some of them I managed to finish but the help and error messages are not finished. There is some input validation and there aren't many places where the user can make mistakes so these messages should be one of my future goals.

## 15 USABILITY GOALS OUTCOME

The usability goals were put forth to make sure that the features implemented were simple and easy to use. No formal usability test was conducted, although I did test the system informally on another user that had no trouble finishing the task required in the time limit given by the usability goals. I felt it was very helpful to have formulated these goals since it made me think more on how the user would actually use the system and not only on how it would work. I believe that during web development it is especially important to not forget the user and always try to envision the user's responses to the interface and its functionality.

## 16 CHANGES FROM INITIAL ANALYSIS AND DESIGN

### 16.1 INTERFACE DESIGN

My main frame of reference for the interface design were the wireframes. I made the graphic design models more as inspiration but I always suspected that the project's timeframe would not allow for such an extreme design. I also felt it was more important that the website was responsive so I decided to use bootstrap as much as possible. The layout of the page turned out quite similar to the wireframes, although, in the forge where the user can make decks and search for cards, I decided to have the decks content listed above the search and not the otherway around like in the wireframes. Furthermore, some of the top nav bar links have other names than I originally planned, but still the layout is quite similar.

### 16.2 THE NAVIGATION DIAGRAM

Although the navigation diagram was very simple I feel it helped me visualize the overall navigation throughout the website. I decided to keep the navigation as I originally designed although some names changed and I left out the 'about Magic' site since it didn't require any programming and was therefore a bit unnecessary for this project.

## 17 WHAT WAS LEARNED

This project increased my understanding of web development considerably. I feel that I have a much better grasp of the MVC concept and how the three layered design works when working with web applications.

Researching, doing tutorials and building the AngularJS project gave me a lot of experience coding in Javascript, working with Javascript frameworks and manipulating data in the JSON format.

Web programming in ASP.NET was completely new to me so I spent some time reading about how it works and after that came countless tutorials and finally the project itself. I now have a good understanding of how the ASP.NET MVC5 model works, how to code the controllers, work with Razor views to bind data into views and compile HTML, manipulate the configuration files, manage databases, work with source control in Visual studio and deploying the webapp to a live server.

## 18 CONCLUSION

Overall I am very happy with how the project turned out, I accomplished my goals regarding learning the technologies needed to become a more valuable member at work. The webapp is simple and there are still a lot of features I'd like to add, such as layout and style changes, error handling, deck information and probability statistics but the system works and is now running on a live server at <http://magicbuilder.azurewebsites.net/>.

The code is available at <https://github.com/savarhall/MagicBuilder>. I fell like I'm just getting started with the system so I will undoubtedly continue development over the summer and hopefully I can get it to a stage where me and my friends at work can start using it to track scores, look at statistics and other related things during our weekly Magic sessions.

## 19 REFERENCES

Further information about Magic the Gathering, the card game.

mtg vault

<http://api.mtgdb.info/>

<http://yeoman.io>

[www.firebaseio.com](http://www.firebaseio.com)

The Github site for the Magic Database

The official AngularJs documentation

Egghead.io - a source of AngularJS tutorials

Thinkster.io - a source of AngularJS tutorials

asp.NET MVC official tutorials portal

Microsoft azure web services documentation