

Magicurve

UX-Imagine

H.A.D.M Harishchandra	114048U
R.C.S. Manoj	114159K
K.A.N. Maduwanthi	114089U

Faculty of Information Technology

University of Moratuwa

2015

Magicurve

UX-Imagine

H.A.D.M Harishchandra	114048U
R.C.S. Manoj	114159K
K.A.N. Maduwanthi	114089U

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Honors Degree of Bachelor of Science in Information Technology

2015

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of the Students

Signature of the Students

H.A.D.M Harishchandra

R.C.S Manoj

K.A.N Maduwanthi

Date:

Supervised by:

Signature of Supervisor

Dr. Lochandaka Ranathunge

.....

Date:

Dedication

We would like to pleasantly dedicate our project of

“Magicurve”

To

The dean of our faculty

Mr. P.M. Karunaratna

The head of the department of Information technology,

Our project supervisor,

Dr. Lochandaka Ranathunge

And

All the lecturers in our faculty.

Acknowledgements

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to our dean, Mr. P.M. Karunarathna. Also we are thankful to Dr. Lochandaka Ranathunge for his valuable guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We also like to thank the other academic and none academic staff members at the Faculty of Information Technology for helping us in many ways.

Last but not least, we would like to thank our family members who were a great support for us physically and mentally.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

Abstract

‘Magicurve’ is the study of generating web pages based on the handmade sketches. The study also included features which allows users to modify the output using a tools or methods to change the input for the tools. This paper discusses about the advantage of automatic generation of web pages using the image processing technologies instead of traditional way of prototyping and sketching. This study will be very much useful for the user interface designers and the user experience designer to reduce the time and cost taken do a prototype. And also the study will provide an opportunity to simulate web pages for any users who are willing to publish a web site using minimal technical skills.

The study has governed a survey on finding the best set of symbols to be used in shape recognition aiming the designers with the architectural background. The study has divided its system into three main sub systems shape recognition, code generation and output modification interface. These three system are implanted using several technologies including asp.net C# and Javascript. The shape recognition sub system is composed with two approaches, rule based and neural network. The code generation also supports multiple design layouts such as absolute layouts for responsive layouts. The output modification interface is implemented using canvas technology which gives nice user friendly way to interact with the proposed system.

The study has found that most designers tend to draw the input symbols with their own style. Therefore, using a machine learning technique is the best way to provide users more friendly system. The system can be extended to support more design layouts and control types in the future.

The shape recognition subsystem had nearly 80% of accuracy while Code generation had 90% of average and modification interface had 100%.

Table of Contents

1. Introduction.....	1
1.1. Introduction	1
1.2. Background and Motivation.....	1
1.3. Aim and Objectives	3
1.3.1. Aim	3
1.3.2. Objectives	3
1.4. Proposed Solution	3
1.5. Guidelines for the Report	4
2. Review of others' work.....	5
2.1. Introduction	5
2.2. Review of Existing Approaches	5
2.3. Summary	8
3. Technology Adapted.....	9
3.1. Introduction	9
3.2. Technologies We Used.....	9
3.2.1. ASP.Net	9
3.2.2. Aforge.Net	9
3.2.3. JavaScript.....	10
3.3. Summary	11
4. Our Approach.....	12
4.1. Introduction	12
4.2. Users of Magicurve	12
4.3. Features	13
4.3.1. Shape recognition and feature extraction.....	13
4.3.2. Rule Based System	13

4.3.3.	Neural Net Classifier.....	14
4.3.4.	Code Generator	14
4.3.5.	Rich editor.....	14
4.4.	Input and Output.....	15
4.5.	Summary	15
5.	Analysis and Design	16
5.1.	Introduction	16
5.2.	System Architecture	16
5.3.	Shape Recognition.....	17
5.4.	Feature Extraction	17
5.4.1.	Symbol Mapping / Identification	18
5.5.	Customize Rich Tool Interface Development	22
5.5.1.	Data Service	22
5.5.2.	App Controller	23
5.5.3.	Drawing Service.....	23
5.5.4.	UI view + Canvas	23
5.6.	Code Generation.....	24
5.7.	Summary	25
6.	Implementation	26
6.1.	Introduction	26
6.2.	Shape Recognition Module	27
6.2.1.	The Convolution	28
6.2.2.	The Blob Detectors	29
6.2.3.	The edge detectors	30
6.2.4.	The matchers	30
6.2.5.	Shape Checkers	30
6.2.6.	The filter helpers.....	31

6.2.7.	The output	32
6.3.	Customize Rich Tool Interface Development Module	33
6.4.	Code Generation Module	34
6.4.1.	Controls Sorting – Step 1	35
6.4.2.	Division Algorithm – Step 2	36
6.4.3.	Sorting List of Rows – Step 3	36
6.4.4.	Main HTML code generation	37
6.4.5.	Deciding Width of Columns	38
6.5.	Summary	38
7.	Evaluation	39
7.1.	Introduction	39
7.2.	Evaluation Techniques and Result	39
7.2.1.	Shape Recognition and Describing.....	39
7.2.2.	Customize Rich Tool Interface Development.....	40
7.2.3.	Automatic Code Generation	40
7.3.	Summary	40
8.	Conclusion & Further work	41
8.1.	Introduction	41
8.2.	Achievement of Objectives	41
8.3.	Limitations	42
8.4.	Conclusion.....	42
8.5.	Further Work.....	42
8.6.	Summary	42

List of Figures/Tables

Figure 1: Magicurve with Input, Process and Output	3
Figure 2: Users of the Magicurve	12
Figure 3: Rule Based System	13
Figure 4: View of the Neural Net Classifier	14
Figure 5: Top level Architecture of the Proposed System	16
Figure 6: Top Level Design View of Shape Recognition – Module 1	17
Figure 7: Design of the Feature Extractor.....	18
Figure 8: Predefined Symbols.....	19
Figure 9: Two Approaches for Shape Mapping.....	20
Figure 10: Common Properties of a Control.....	21
Figure 11: Top Level Design View of Interface Development – Module II	22
Figure 12: The Data Contract	23
Figure 13: Top Level Design View of Code Generation – Module III	24
Figure 14: The Top Level Implementation	26
Figure 15: Image Processing Layer Implementation	27
Figure 16: The Identity Matrix	28
Figure 17: Processing Convolution Algorithm	29
Figure 18: Shape Checker Structure	31
Figure 19: Filter Helper Implementation	32
Figure 20: Control object for output	33
Figure 21: Sorting Algorithm Based on “Y” Property.....	35
Figure 22: Final Process Pseudo Code.....	37
Figure 23: Test cases for shapes	39

Table of Abbreviations

Abbreviation	Meaning
HTML	Hypertext markup language
UI	User Interface
UX	User Experience
AI	Artificial intelligence

1. Introduction

1.1. Introduction

User interface is the main component of a web application that user interacts. To design user interface, UI designers have to do a heavy lift. There are several steps before the actual html code is developed. First the handmade sketch is drawn. Then the sketch is needed to convert to a wireframe. Therefore, a designer draws a wireframe using a wire framing software. Then the wireframe is converted to a prototype using software like 'Photoshop'. Then only we can generate the html using that prototype. As you can see this takes lot of time and effort. What if he can bring UX designer's imagination from sketch to html design without converting the sketch into wireframe or prototype? What if the he UI designer has so little to do? Therefore, "Magicurve" is the solution we are looking for.

Magicurve, a user friendly tool which converts handmade sketch to html using image processing and certain AI technologies. There are many tools that generates html from the prototype, but the developers start shouting the output because they generate a crappy set of code that developers cannot start work on. Therefore, we intend to output a basic but professional standard code that can be used for further developing. And also this tool will use the built-in web camera of the user's laptop as the input for the desired output, and also some basic customizations on the fly (Of course we can always edit the generated code).

1.2. Background and Motivation

Web application development is the process and practice of developing web application. When we think about technologies those are used in web application development, there are two main categories of coding, scripting and programming for creating web applications [1]. Those categories are client side scripting, coding and server side scripting, coding [1]. User interface designing is the main task of client side developing in web development process. It mainly focuses on the user's experience and interaction [2]. Developing UI involves different kind of developers and they need different skills and developing, designing experiences as well as knowledge to complete this process successfully with meeting relevant goals. Not only that this process may

expensive and costly than other steps. When we consider about user interface creation of web application development process, basically it contains three main steps. Firstly, UX designers have to draw a sketch of the web page. Then it needs to convert into graphic with PSD or any other image format. Finally, they try to develop web page using markup language such as html by looking created PSD graphic prototype. But sometimes this process may take lots of time and waste time because of the bad designing of the user interface. It will also cause to decrease user interaction for web page.

As the main UI designer of the Innovative-e (pvt) Ltd, Mr. Qasid Mukhtar says there are tools like Adobe Muse, Axure or even Photoshop, “they generate HTML from what’s designed, but the HTML they render, if isn't used correctly generates a crappy set of code which breaks the moment you start dev on top of it” [3]. Therefore, it’s very important that what every tool that we are intent to build should generate the code with good syntax.

Another problem is that most of the developers do not have a good eye for the design. Therefore, no matter how much harder they develop the business logic, the software fails because of the user interface. Therefore, for individual contractors need a tool that can generate UI with a little help from a designer. Or they can use this tool to test out different layouts with in milliseconds of time can come up with the best layout they can use. And if we consider about multiplatform mobile development frameworks like apache Cordova or Ionic, the developer can use this tool to generate all the screens and controls at place in no time. And as we are now using project management techniques like Scrum, the requirement changes very rapidly. Therefore, this tool will help to adapt to those changes faster.

With the knowledge we are going to gain knowledge on image processing and AI technologies at our level four subjects it’ll be very interesting to use our knowledge at this challenge, and come up with a professional level tool that can have good business value. Therefore, Magicurve is rapid developing solution with minimum steps of creating user interface than above described existing process.

1.3. Aim and Objectives

1.3.1. Aim

The aim of this project is to develop a user friendly tool for rapid development of web application interfaces by generating html code automatically from sketch input with use of image processing techniques and certain AI technologies.

1.3.2. Objectives

There are set of general objectives of our project.

- Study of the rapid development of web interfaces with saving time.
- Study of technologies such as image processing techniques and other AI techniques that can solve the problem.
- Design a tool that can generate rapid html designs using a provided sketch image.
 - ✓ This tool should be able to manipulate all the controls one by one.
 - ✓ Ability to recognize the controls changes if the same sketch is revised.
 - ✓ Contrast between different versions of the same sketch.
 - ✓ Error handling.
- Evaluation of the proposed solution.

1.4. Proposed Solution

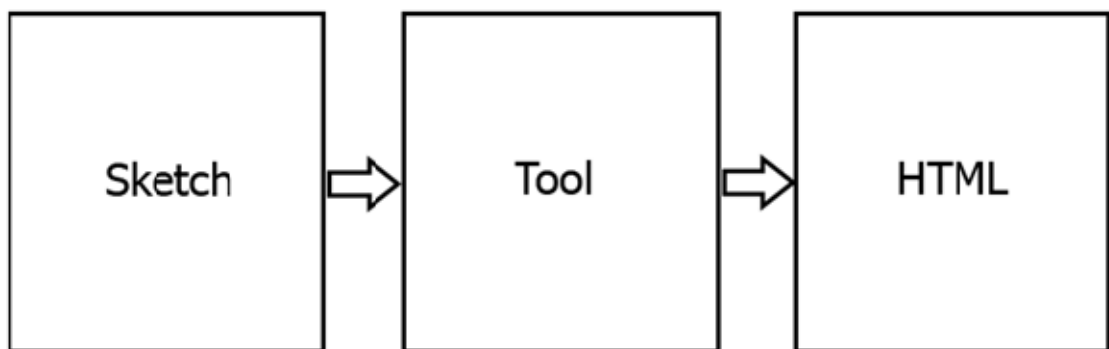


Figure 1: Magicurve with Input, Process and Output

Our solution is used handmade sketch of html pages as input and then tool that we will implement, imports that sketches and process them and finally it will generate html page according to sketch input.

1.5. Guidelines for the Report

Chapter 1 describes the background and motivation, problem in brief and our aim and objectives of doing this project. Literature review is discussed in detail in chapter 2. Chapter 3 addresses the technologies we have adopted and our reasons for the selection of those over others while Chapter 4 is about our approach to the solution. Analysis and design is explained in Chapter 5 whereas Chapter 6 has detailed information on the implementation of each module. Report concludes with the final chapter; Chapter 7 which contains the discussion.

2. Review of others' work

2.1. Introduction

In this chapter we will be discussing about others approaches which have been taken for this particular problem. And also we are describing technologies, algorithms that are used by others that are similar to our approach. By going through this chapter you will be able to understand what are the lacks available in current systems, what are the solutions suggested by the developers to overcome them, how it can be achieved, how the technology is going to be adopted and so on.

2.2. Review of Existing Approaches

User Interface designing and developing in web development is time consuming task because of it has to complete several initial steps. Therefore, we have to research and review other existing approaches and algorithms as aim of our project is to develop a user friendly tool for rapid development of web application interfaces by generating html code automatically from sketch input. This is brief review of similar approaches.

According to our one of the main module of our project, we are requiring to recognize of sketch curves, corners and edge points as specific shapes or unique symbols by using image processing techniques and then map those recognized shapes into html controls based on feature vector. The past researches have been developed several techniques and algorithms with the intention of shape detection such as using generalizes Hough transform [4,5,6]. In these papers, they have provided some alternative interpretations for Hough's method. Further template matching [7,8] etc. named papers also provided important ways of shape detection. Anyhow, both of Hough method and template matching method are sensitive to noise and sampling artifact.

Michael Kass proposed active contour models [9] and Chenyang Xu proposed vector flow model [10] to preventing this previously mentioned problem. But then another limitation can be found in those approaches also. It is that methods suffer from high computation time and complexity. Then after, 2012 Zakaria has proposed method to recognize triangle, square and circle kind of basic shape in an image via algorithm. In his algorithm, he used intensity value from the input image [11]. After that it is

threshold by using Otsu's method which is non parametric and unsupervised method [11] for automatic threshold selection in segmentation. Foreground and background are the two regions which contain in threshold image when the use of Otsu's method. Then median filtering is used to exclude noise. And also with the purpose of detecting and finding edges in an image, they used Sobel operator. It is essential of having method to remove unwanted edge pixels. Therefore, thinning method is applied for that purpose and it help them to increase the false detection [11]. Finally, with the use of compactness of region [11] shapes are detected. In here they have addressed the previously discussed limitation of high computational time problem. Because in this recognition algorithm takes less computational time. But this method also has some limitations such as noises not completely removed, sensitive to noise and lighting conditions [11]. Therefore, in some cases, noise will identify as object in the image. Then output result is not the desired one.

After segmenting and detecting primitive shapes and edges, corners of sketch input, in our case we have to refer and study that how to recognition complex shapes, mapping, describing of recognized shapes or symbols into corresponding html controls or elements by using appropriate AI techniques such as rule based approach, neural network as next module of our project as next step of first module. In 1996 J. A. Landay has been tried to introduce interactive sketching tool called SILK (Sketching Interfaces Like Krazy) [12] that avoids design, implementation and evaluation problems in early stages of UI design. And also it combined more benefits of paper related sketching with current electronic tools. This tool is a recognition system that lets the user to sketch out hard graphical user interface designs. Also it transforms them into more glossed version as the design becomes more stable. There were some techniques and concepts which they have used something interesting. Some of those basic concepts and techniques are sketching in electronic way for user interface design, storyboarding using electronic way, memory design mechanism and page transformation. When these researchers have tried to reach their objectives they have used automatic recognition and other automation mechanisms and algorithms. Even though this system identified and solve the notion of ambiguity in the sketch, there is a limitation of handling of single parts. As an example there is an ambiguity when it comes to identify that is this group of strokes a radio button or a checkbox [12].

We have focused our attention to find existing automatic code generation related approaches also with the purpose of getting idea for our second module of code generation. Then it was very difficult to find similar algorithms and techniques for our requirements. Anyway few years ago Andrew and Sona has published paper with title of “Automatic code generation: model – code semantic consistency” [13]. In this paper they have described benefits of having automatic code generation software. It also indicated modeling experiments with the use of model elements (symbols) and code constructs [13]. Finally, they have analyzed semantic consistency between used model and generated output code [13]. Furthermore, it described more advantages of using automatic code generation for particular achievement such as our approach. They mentioned different ways of performing automatic code generation such as code generation templates [13], high-level graphical notations for modeling [13] and so on. But this paper just explained analyzing and comparing way of generated code with expected goal. They did not provide any algorithm for generating code automatically. In 2011 “Research and Application of Code Automatic Generation Algorithm Based on Structured Flowchart” [14] paper was explained about similar approach by Xiang-Hu Wu and few other researchers. They have proposed an algorithm based on structure identification for structure flowchart [14]. Then exhaustive methods are used to measure the effectiveness of the proposed algorithm [14]. The proposed algorithm is based on mainly with depth-first search algorithm with recursion [14]. First they identified flow chart by using this algorithm and then based on output of that part, it is generated code automatically.

Final module of our approach is implementing user friendly tool with lots of features and integrating other two modules together to achieve final system. Therefore, our final goal is developing a user friendly tool for rapid development of web application interfaces by generating html code automatically from sketch input. When we researched on existing tools similar to our approach we have found that HTML code generator [15]. It is a generic software which also enables user to create web pages in web development process with any level of computer skills without any need of knowing HTML code in less time. Even this tool provides lots of feature for user, it has some limitations such as it doesn’t allow handmade html sketch input into system, doesn’t output w3c valid html source code and it is hard to customize html code in further development.

2.3. Summary

In this chapter we have clearly explained about currently available systems for this problem and lacks available with the current systems. There are advantages in our systems over others approaches.

3. Technology Adapted

3.1. Introduction

This chapter mainly conveys the idea about the technologies we used to implement our project. When we consider about this project we see lots of new technologies which must be combined to implement this system. When comparing to our knowledge we gathered from the references we used simple techniques to prepare our project.

3.2. Technologies We Used

There are three major technologies that we use to implement our project. They are ASP.Net [16] Aforge.Net [17] and JavaScript [18]. Brief descriptions about these technologies are given below.

3.2.1. ASP.Net

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript. We are using web API 2.2 for building rich restful web service to work with JavaScript front end [16]. Web API also gives the extensibility features like dependency injection restful routing. And also ASP.Net which can be implemented with C# gives the ability to work with other necessary technologies like Aforge.Net. As well as we use ASP.Net to support Azure Cloud Hosting [16].

3.2.2. Aforge.Net

AForge.NET is an open source C# framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence-image processing, neural networks, genetic algorithms, fuzzy logic, machine learning, robotics, etc. [17].

We use this framework over OpenCV and other image processing frameworks because it has lots of inbuilt shape recognition features. And also Aforge.Net supports neural networks and other AI features which will be useful for describing the recognized image segments. It can be used with ASP.Net too.

3.2.3. JavaScript

JavaScript is one of the simplest, versatile and effective languages used to extend functionality in websites [18]. Uses range from on screen visual effects to processing and calculating data on web pages with ease as well as extended functionality to websites using third party scripts among several other handy features [18]. New HTML5 features such as the canvas, audio and video elements, as well as supporting HTML5 technologies such as local storage and geo-location, are rather useless without JavaScript [19]. So, it is essentially mandatory for those who are making HTML5-compliant browsers to support JavaScript [19]. Supporting other scripting languages is optional [19]. We are using two main JavaScript library in our project. Those are called AngularJS and ZebraJS.

3.2.3.1. AngularJS

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications [20]. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop [20]. AngularJS is a toolset for building the framework most suited to our application development [20]. It is fully extensible and works well with other libraries [20]. Every feature can be modified or replaced to suit your unique development workflow and feature needs.

3.2.3.2. ZebraJS

Zebra is a new OOP, Rich WEB UI programming philosophy that makes stress to modular software engineering as an opposition to HTML DOM/CSS manipulation with scattered everywhere JavaScript code. Zebra can rely on almost empty HTML as a container for JS code [21]. Zebra core introduces and implements easy and very clear OOP concept. Inheritance, interfaces, method overloading and overriding, constructors, inner classes, packaging and other OOP features can be found there [21]. Following OOP stimulates implementing well designed, supportable, extendible and pretty code [21]. Zebra Rich WEB UI is powerful engine that allows developers to implement any imaginable user interface components [21]. Zebra UI core employs new HTML5 Canvas element where any custom content can be rendered. Having full control over UI components rendering you can get rid of typical annoying WEB UI “puzzling” and heavy CSS/DOM manipulations [21].

3.3. Summary

By reading the content of this chapter you would have got what the technologies used on this development process, what are the specialties we've got and what are the reasons for choosing that one among a bunch of the other available technologies.

4. Our Approach

4.1. Introduction

The previous chapter described different technologies, libraries that have been adopted to achieve our system. By going through the content of this chapter, you may get a clear view that how we are going to achieve our aim and objectives of our project and how we get adopted with the above those technologies to achieve our approach.

4.2. Users of Magicurve

The propose system will be used by both technical and non-technical personal. This system is mainly focused for the UX designers who do not have technical skills for handling high end software like Dreamweaver or sketch3. But persons who have the technical knowledge has the advantage to customize the generated code using rich editor. The programmers also can use the system to do rapid prototyping of web pages or mobile designs.

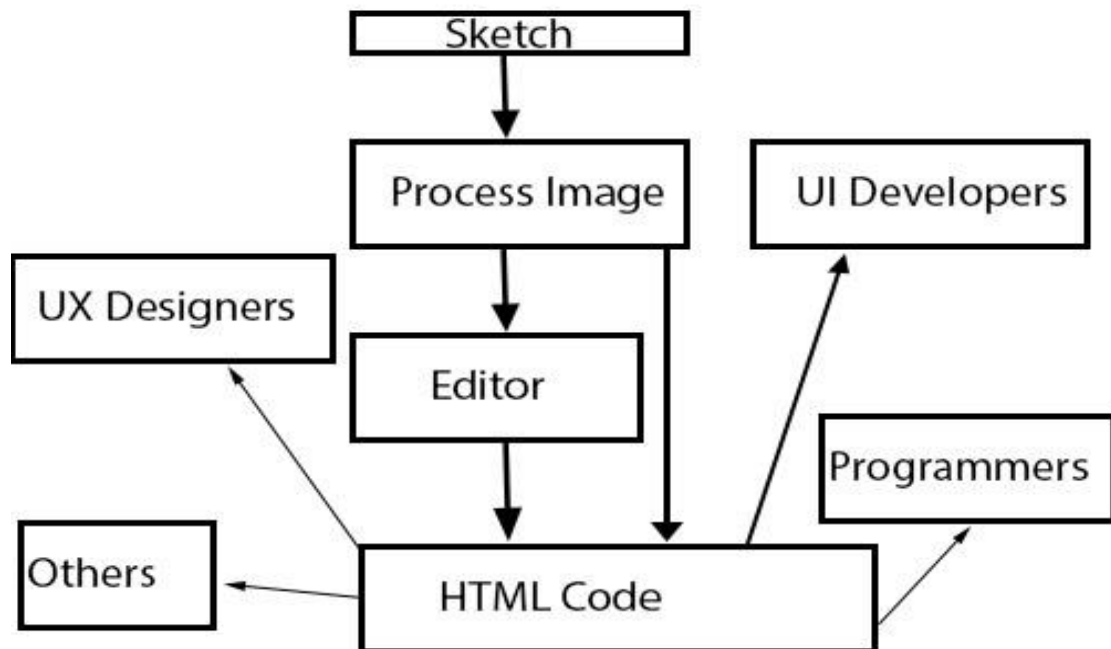


Figure 2: Users of the Magicurve

4.3. Features

4.3.1. Shape recognition and feature extraction

We have used GDI+ interface to access the image's pixel using C#. The Accord-net and Aforge.net frameworks were used to apply the general image processing filters like Gaussian and Threshold. For extracting the features, we had to use pixel based algorithms since the algorithms should have the specific parameters for the image condition of our inputs and outputs.

We had defined fifteen (15) shapes to be recognized using the image processing. The main features extracted from the image were

- Number of Corners
- Number of Horizontal Lines.
- Number of Vertical Lines
- Number of Different Angels between Lines.

Once the sketch image is used to extract above features, the feature vector will be used as an input for the neural network layer or rule based system we have implemented.

4.3.2. Rule Based System

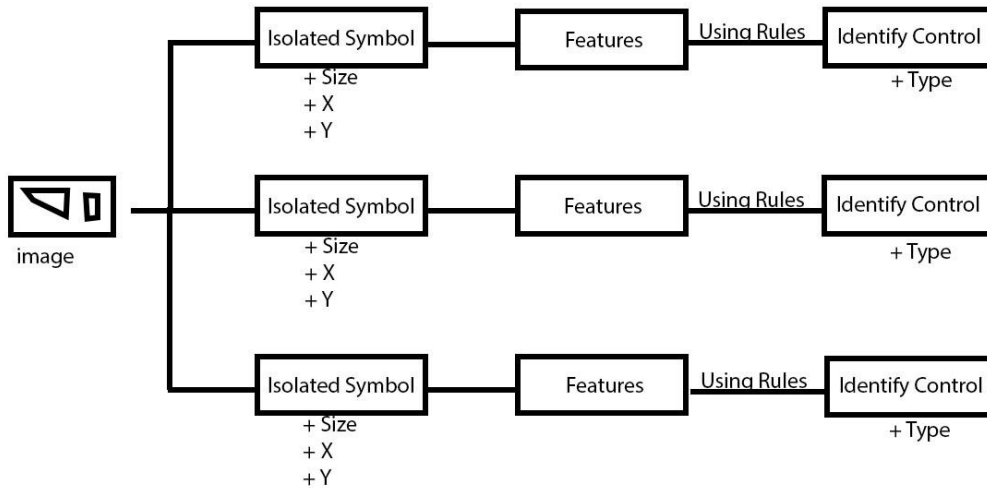


Figure 3: Rule Based System

The rule base system is implemented using pure C# as it basically a set of logical conditions which would match the relevant inputs to a known class. This approach was first one for recognizing the controls. But we decided using a neural net would more appropriate if we were to introduce new symbols.

4.3.3. Neural Net Classifier

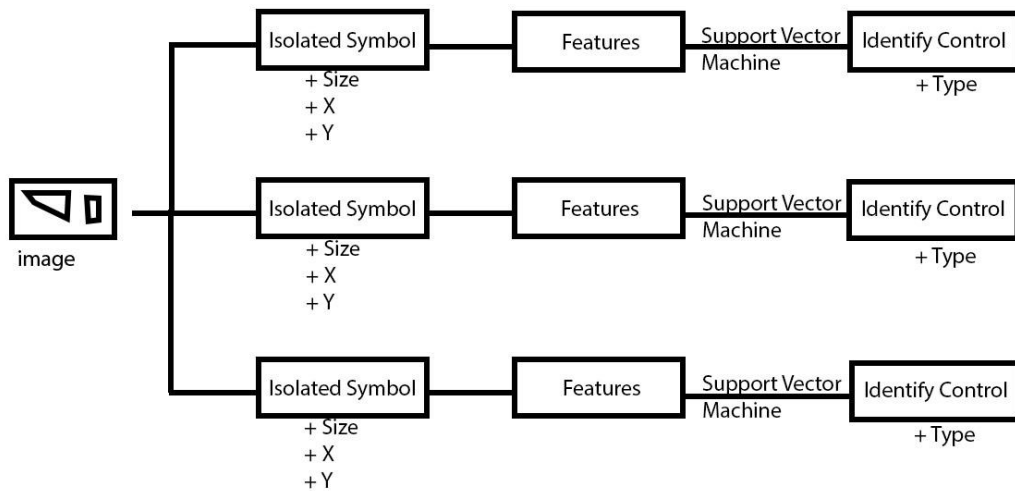


Figure 4: View of the Neural Net Classifier

The classifier is used to recognize the defined controls as output classes and the features extracted from training set as the inputs. This implements with the help of accord-net library. The main advantage here was we can train new symbols to map our controls.

4.3.4. Code Generator

The output from the rule based system /neural net classifier or UI editor would be used as the input for the code generator. This will a simple list of controls with its properties. The code generator uses enhanced algorithms to position the controls send by the first module. The code generator is also implemented in pure C# and the output of the system is the HTML code itself. The code generator is hosted as asp.net web API self-host application so that it can be used independently.

4.3.5. Rich editor

The output from the rule based system /neural net classifier is used as the input for the rich editor this is implemented using javascript. The editor can be used to change existing controls add or remove controls. And to implement this we have used Angularjs for architecture and Zebrajs library for manipulating canvas. The changes done by the user is reflected to the code generator as they request for the html code. The rich editor makes this system a user friendly and usable one.

4.4. Input and Output

Inputs to the system are handmade sketch and user made changes on the rich editor. The outputs are generated html and the live view of the controls in the rich editor.

4.5. Summary

This chapter is detailed on our approach in solving the mentioned problem. Here we have discussed the conceptual design of each module with alternative paths we have taken. The features are described as different modules or subsystems of the full system. The next chapter is about analysis and design of the system.

5. Analysis and Design

5.1. Introduction

In this chapter we mainly focus on the modules that we can divide the whole process. Here we use some appropriate figures and top level design of the system with the purpose of giving correct idea about each module and how these separated modules work together. The interaction and the behaviors of the modules are clearly shown in this chapter.

5.2. System Architecture

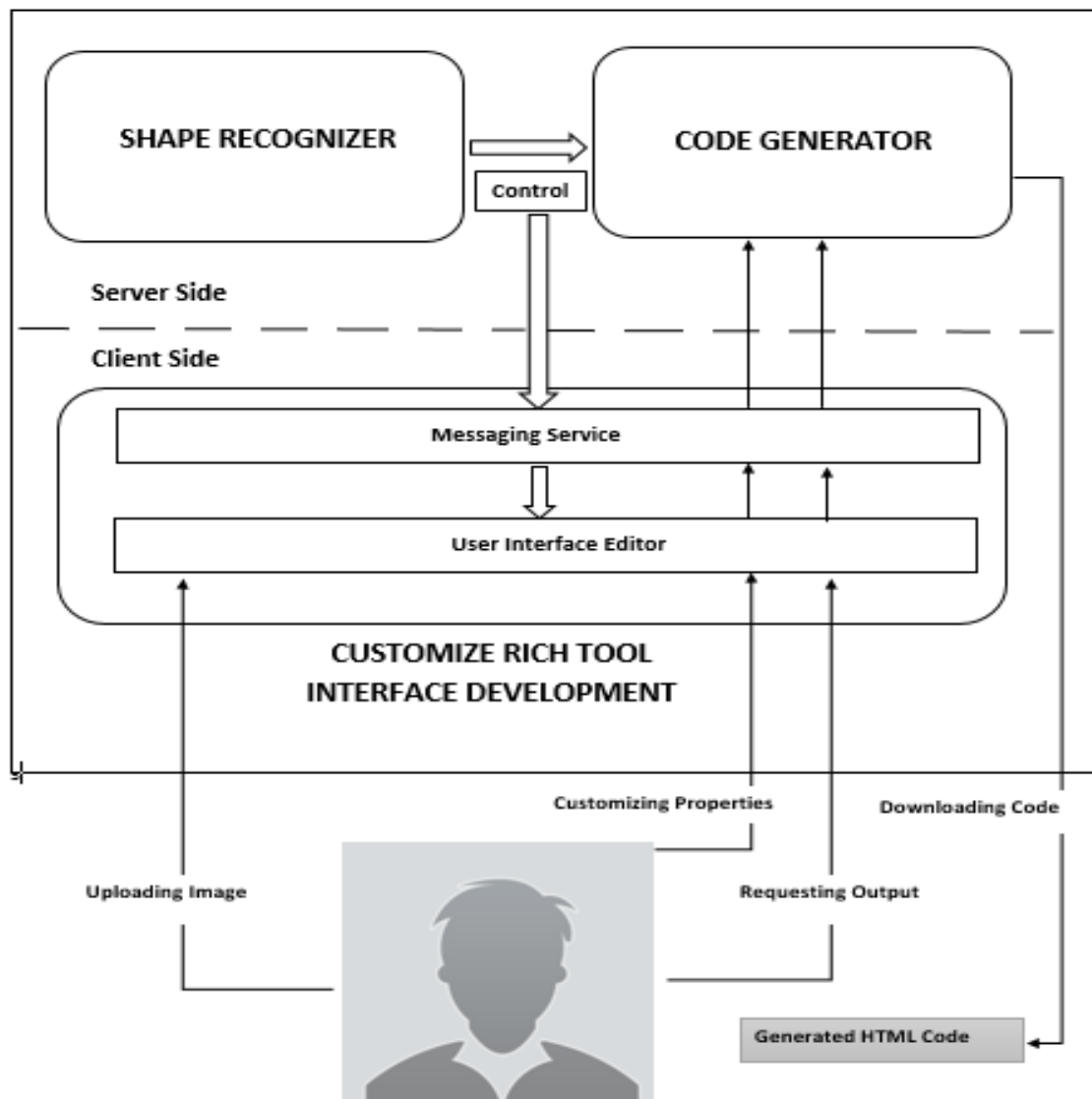


Figure 5: Top level Architecture of the Proposed System

The system is designed with long term goal of publishing as a web application in the cloud. Therefore, the whole application is crafted in a way where all the dependencies can be changed with minor effort. Our system has three main modules as below.

- Feature extraction and shape recognition.
- Rich customization tool interface development.
- Code generation.

An abstract level architecture diagram of our system is given in figure 5.1. The system is a collection of the following modules; its functionality and its interaction with the other modules are explained below.

5.3. Shape Recognition

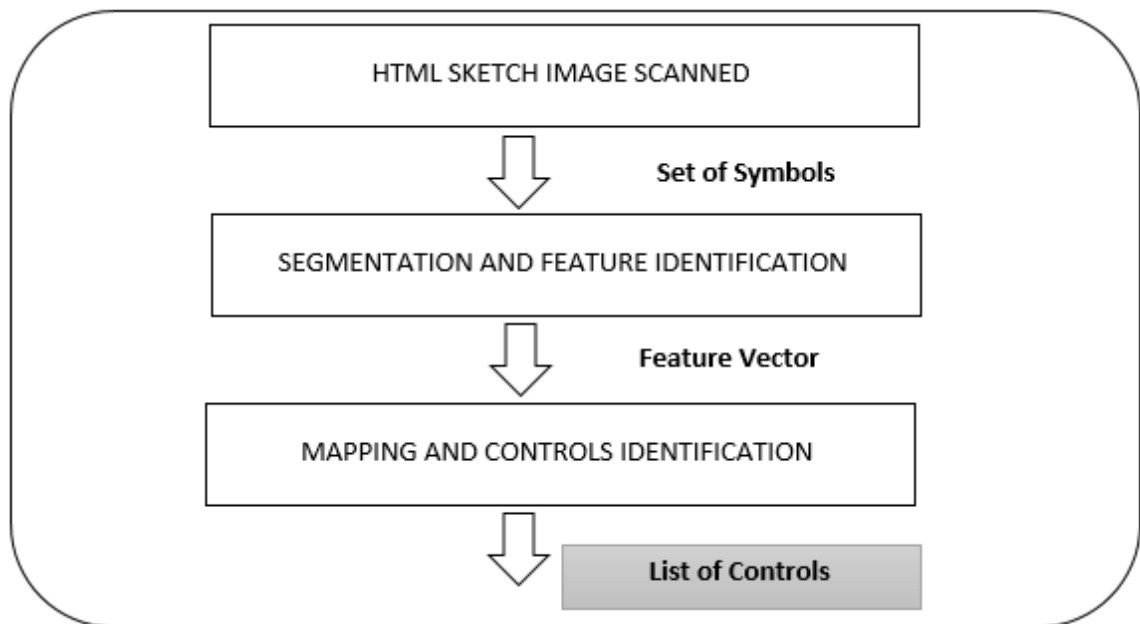


Figure 6: Top Level Design View of Shape Recognition – Module 1

Shape recognition is done via image processing and neural network techniques. Here we have followed two different approaches.

5.4. Feature Extraction

The top level workflow of the first approach is displayed in the figure below. It illustrates the steps followed until feature extraction.

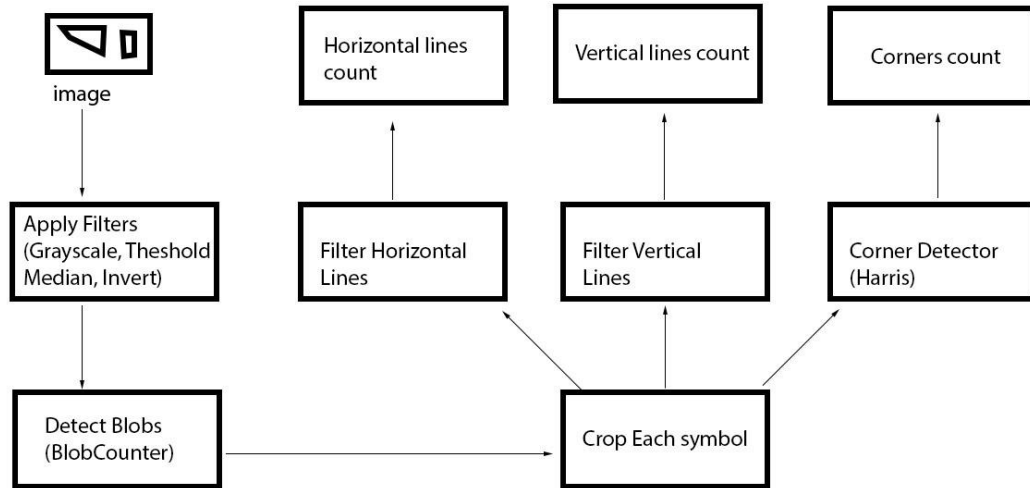


Figure 7: Design of the Feature Extractor

The design of the feature extractor is made to work with controlled level of image quality. Once the image is inserted, the system will apply the imaging filters to the image to make ready the image for blob processing. We had to choose this filters carefully and effectively since most of handmade sketches contains noise and false positives. The algorithms used for those filters would be discussed in the next chapter.

After going through the Blob counter which extracts each counter, we use that output to crop the original image and create an image which contains only single symbol. When the cropped symbol is available we can send copy of that symbol for processing Horizontal lines, Vertical lines, Corners.

Horizontal lines and vertical lines are extracted using a convolution algorithm. And we used Harris Corner detector for extracting the corners. After these features are extracted. We can send these features to the Rule based system or Neural net for determining the type of the symbol.

5.4.1. Symbol Mapping / Identification

We had defined 15 basic symbols which user can use for the input of the system. We also did a survey on finding the best symbols for each controls with maximum user experience.

The symbols defined are shown in figure 5.4. We used rule based approach for identification of the symbols. But to add more shapes or to change the existing shapes it was required to change the existing code again and again. Instead we decided that we

should design a neural network which can be used to train new symbols and change the symbol if user required. Since we are using Asp.net we designed the dependencies to be injected at the run time (DI).

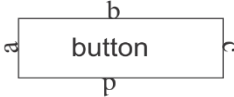
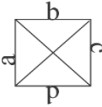
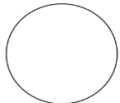
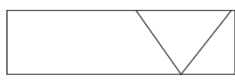









Button		4	2	2
CheckBox		12	2	2
Radio		-	-	-
Combo		9	2	2
TextBox		8	2	3
Label		8	3	2
Paragraph		16	3	3
Image		8	4	4
Password		16	2	5
HyperLink		8	3	3
DatePicker		7	2	2
Iframe		3	2	2
Horizontal Line		-	1	0

Figure 8: Predefined Symbols

The features expected for each control is mentioned in above figure in order of

- Corner Count
- Horizontal Lines
- Vertical Lines

The shape recognizers are designed in a manner that they can be injected as mentioned. Therefore, we can use either UiShapeChecker or SvmShapeChecker.

The Class design diagram for the shape recognizers are as follows.

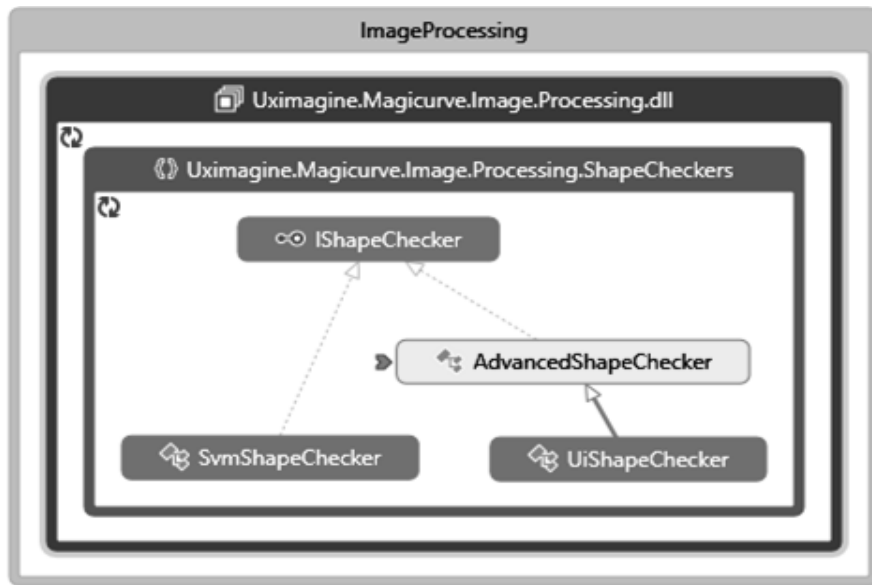


Figure 9: Two Approaches for Shape Mapping

So if the user wants to recognize the shapes using the neural network techniques all we have to do is change the object to be initialize at the runtime of the web site.

As the top level system architecture describes this module will output a set of control objects which has the following format.

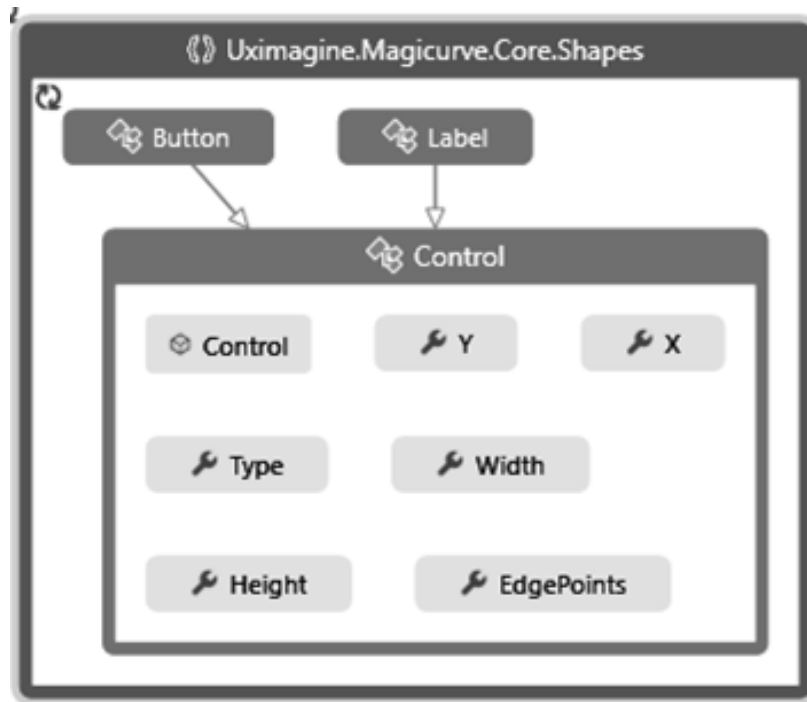


Figure 10: Common Properties of a Control

Each control contained these properties

- Type
- X
- Y
- Width
- Height
- Edge Points

And we used the same design format for the input of both modules UI tool and code generator. Both modules receive these controls as a list of controls. We will discuss the design of the UI tool in the next topic.

5.5. Customize Rich Tool Interface Development

This module is about building an interactive tool for the users to customize the output of the system.

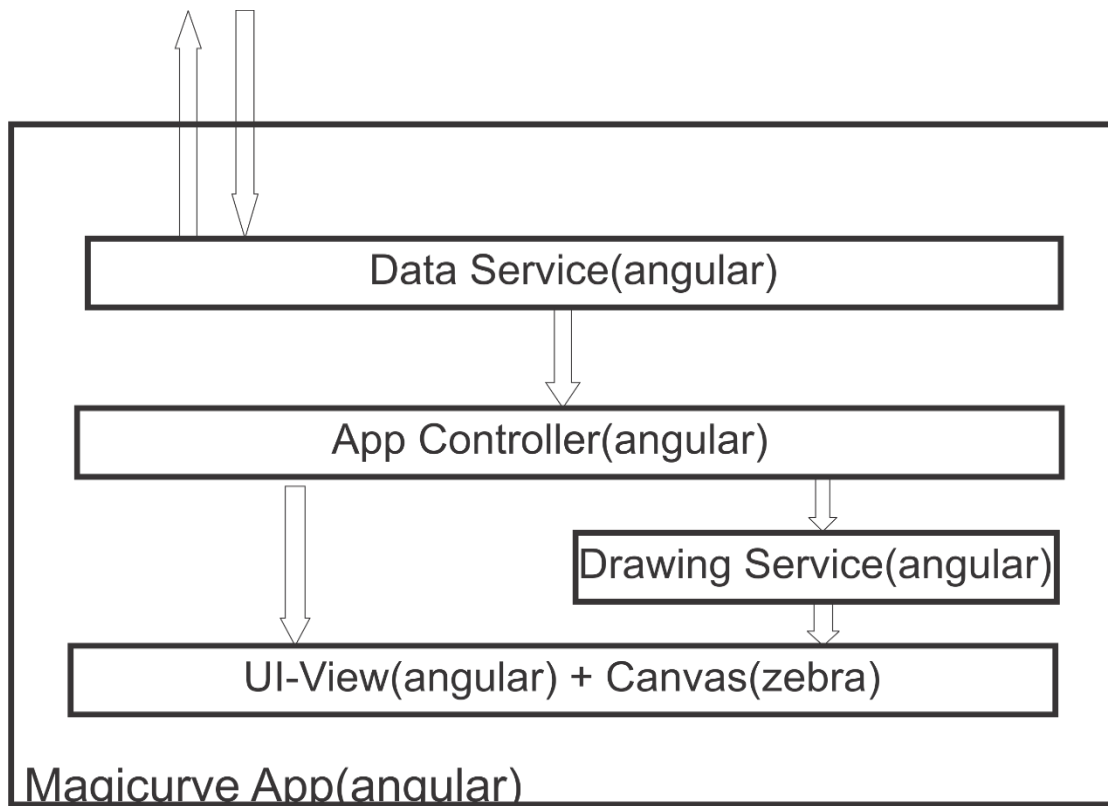


Figure 11: Top Level Design View of Interface Development – Module II

Since we have decided we need to implement a single page application for the maximum user experience we designed the front end using angular and zebras. The angular library does most of the heavy lifting for user interaction. And the zebras is used for canvas manipulation.

5.5.1. Data Service

The data service is the bridge between the back end and the front end. As the back end contains the API for image manipulation and code generation, data service does all the message passing with that API. The data service is typical angular service and the implementation will be discussed in the next chapter.

The input for this module will a JSON with the following design.

```
"Controls": [  
  {  
    "Type": 0,  
    "X": 0,  
    "Y": 0,  
    "Width": 0,  
    "Height": 0,  
    "EdgePoints": null  
  },  
  {  
    "Type": 0,  
    "X": 0,  
    "Y": 0,  
    "Width": 0,  
    "Height": 0,  
    "EdgePoints": null  
  },  
]
```

Figure 12: The Data Contract

There might be addition field included in the JSON. For example, if the control is a button the text for the button is included. This is the contract between back end and the front end of the system.

5.5.2. App Controller

The controller is used for controlling user interactions. The event fired in the main UI will be captured directly by the controller or if the event is fired inside the canvas, then it will be notified to the controller via the drawing service.

5.5.3. Drawing Service

The drawing service will be used to communicate with the canvas in the html page. This service is also an angular service which is one of the few in the Magicurve app.

5.5.4. UI view + Canvas

This is the user interface of the application. The screen shot for this view is available at the appendix. The canvas manipulation is done via the zebrajs library, which has a lot of predefined html controls. And also it supports a lot of events like resizing, arranging, and context menus.

We also use this view to give live update on the generated code on each modification done via the UI. If the user is not interested in editing the generated view, then user can download the generated code via the relevant command. The UI changes will be sent

to the backend via the data service we mentioned above. The code generator will scan the list of changes and generate code again.

The design does not support only to update changes. Instead it takes the full list of controls and regenerate the code for those controls. The ability to contrast between different versions of the code will be added in the future.

5.6. Code Generation

This module is going to describe about how to automatically generate final html code with set of new algorithms. In Shape Recognition module is going to identify shapes of given sketch and map them into defined html controls after this tool receiving sketch input. Then it will pass list of all controls in the html sketch with its properties to the code generator. Top level design view of code generation module can be stepped as below figure 5.9.

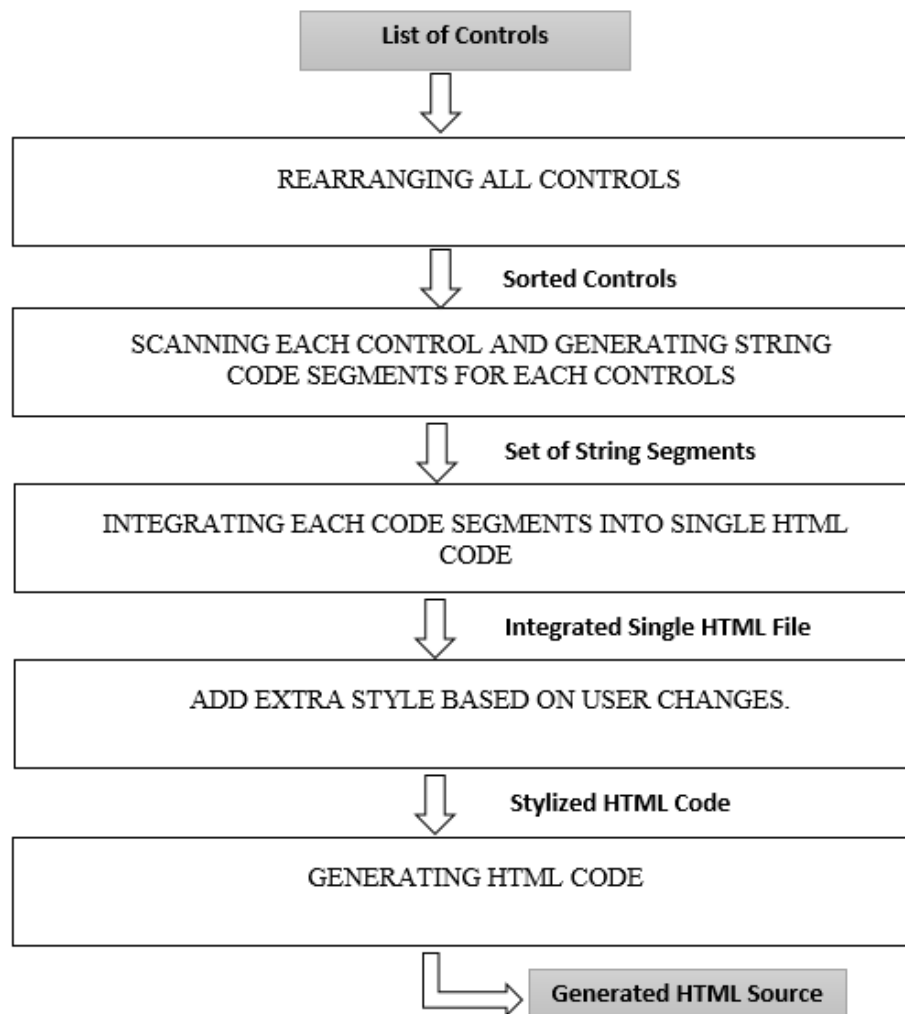


Figure 13: Top Level Design View of Code Generation – Module III

Input set of list of controls of this module are not sorted or ordered when it received to this module. Therefore, as first step, we have used new set of algorithms to sort and arrange those controls into order. After that based on those sorted controls, code generator generates string code segments for each and every controls separately. This component links with next component of integrating all string segments of each controls based on sketch input requirement of user. As well as if user makes changes for controls via pop up dialog, then those changes are added into final code also by calling relevant functions of code generator. Finally, code generator module releases generated html source code with other sub files supported to main source.

5.7. Summary

In this chapter, we have illustrated and explained the top level design architecture of our system and functionalities of each module. You will be able to get a clear idea of our project solution with its separate modules by referring this section. Therefore, we can conclude that this is the precise workflow of this project with the deep analysis and design of solution.

6. Implementation

6.1. Introduction

In this chapter we have provided implementation details of each module that is stated in the design diagram. We have maintained details with the consistency between design and the implementation sections. While we were describing the implementation, we have stated about software, hardware, flowcharts, algorithms, pseudo codes, code segments as per each module in the design.

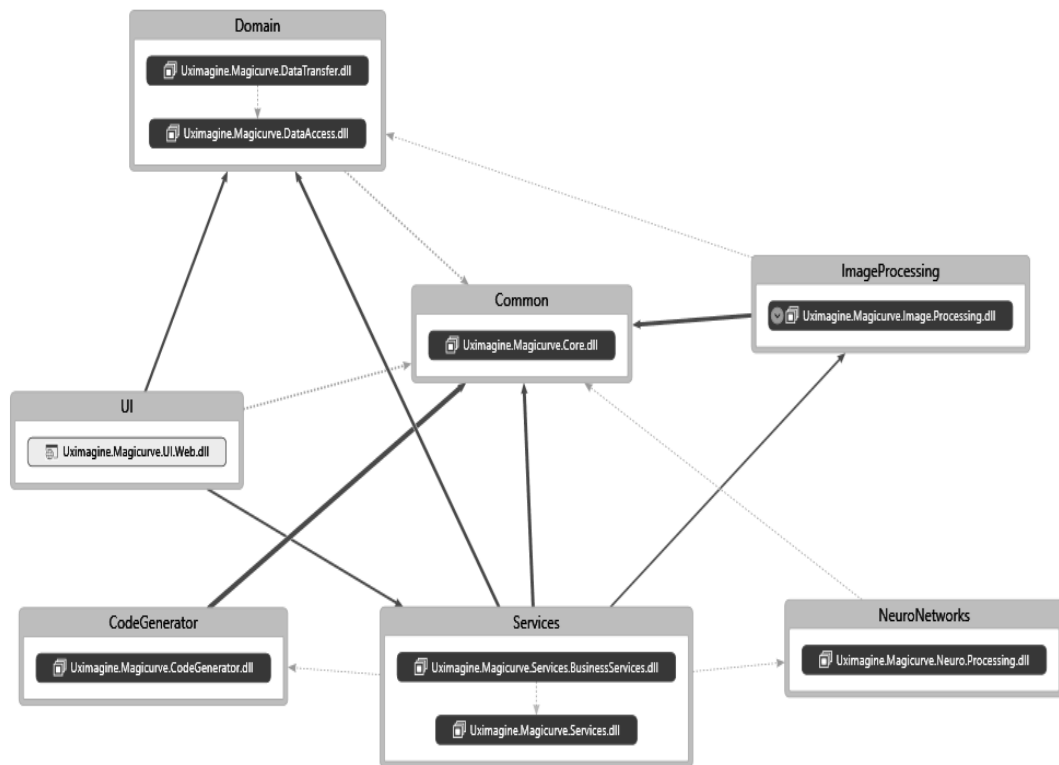


Figure 14: The Top Level Implementation

The application is implemented as an Asp.net web application. The above figure shows the dependencies and the namespaces of each layer. Note that modules we discussed earlier are included with this implementation. They may be implemented as multiple layers in the project. The projects Core, Data Access, Data Transfer, Services and

Business services are support layers in order to function the application as a web site and other nonfunctional requirements.

6.2. Shape Recognition Module

The shape recognition module is fully implemented in Asp.net and C# with help of Accord-net library. As the figure 6.1 shows, the two layers involved with the shape recognition module.

- Uximage.Magicuve.Image.Processing
- Uximage.Magicuve.Neuro.Processing

These two layers are combined in the Business layer when the user initiated processes are requested. Additionally Uximage.Magicuve.Core is referenced in every layer. The figure 6.2 show the implementation of the Image Processing Layer.

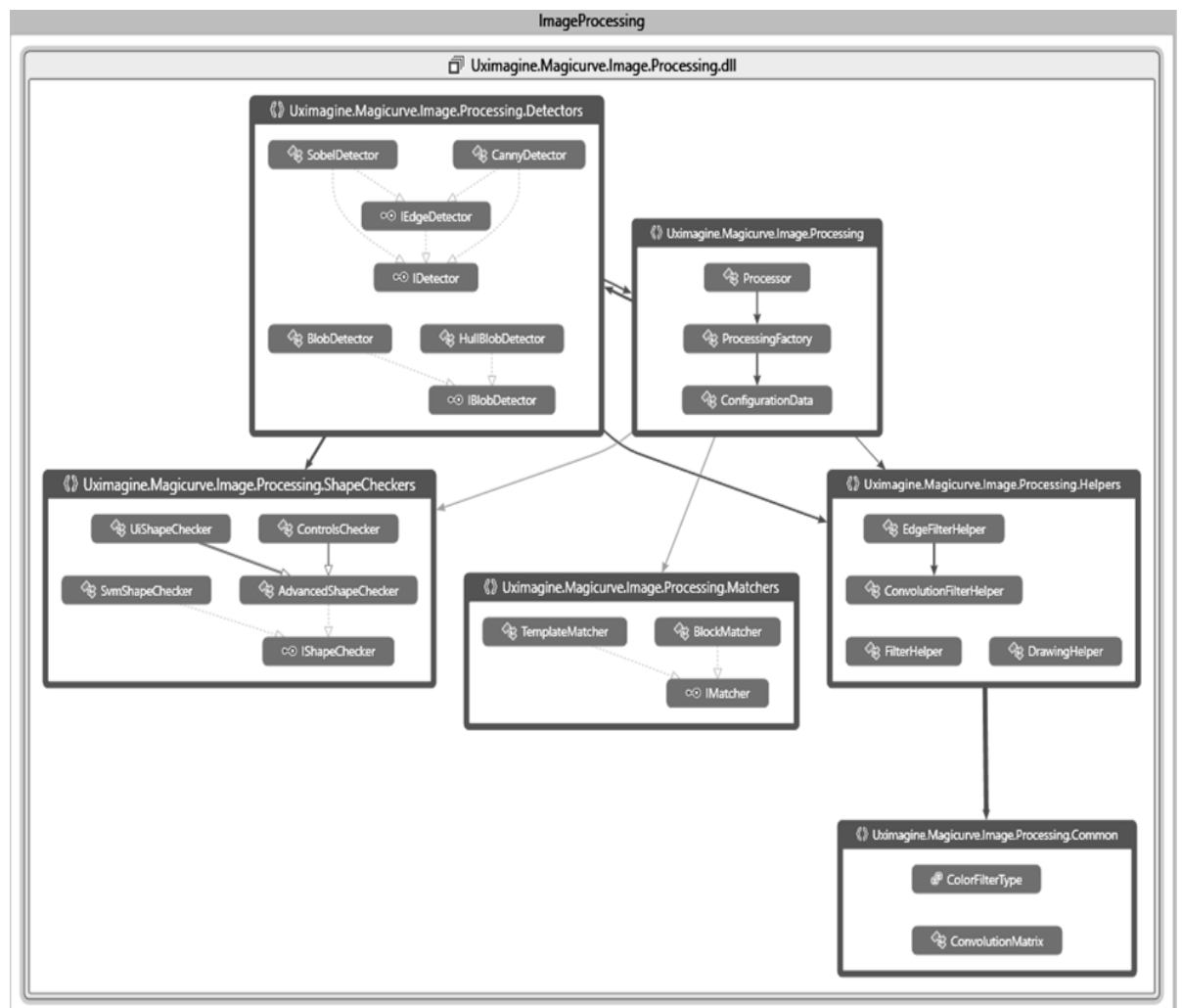


Figure 15: Image Processing Layer Implementation

This layer consists of following type of operators.

- Blob Detectors
- Edge Detectors
- Matchers
- Shape Checkers
- Filter Helpers

6.2.1. The Convolution

The convolution is the concept of merging a kernel or a small matrix with the pixel values of an image. By doing this merging we apply simple filters to the target image. The figure 6-3 show the identity matrix, which give the same image as the output after convolution. We have used this convolution concept for many filters we have implemented in this layer.

The convolution is the concept of merging a kernel or a small matrix with the pixel values of an image. By doing this merging we apply simple filters to the target image. The figure 6-3 show the identity matrix, which give the same image as the output after convolution. We have used this convolution concept for many filters we have implemented in this layer.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figure 16: The Identity Matrix

```

for each image row in input image:

    for each pixel in image row:
        set accumulator to zero
        for each kernel row in kernel:
            for each element in kernel row:
                if element position corresponding* to pixel position then
                    multiply element value corresponding* to pixel value
                    add result to accumulator
                endif
            set output image pixel to accumulator

```

Figure 17: Processing Convolution Algorithm

6.2.2. The Blob Detectors

The blob detectors are used for isolating the symbols. There two different implementation provided by Accord-net is used in this layer. These two implementations are inherited from the IBlobDetector interface which enables to inject the implementation at the runtime. The HullBlobDetector is more accurate when detecting discreet shapes. The Blob detectors output set of Point with structure of following.

```

public struct IntPoint
{
    /// <summary>
    /// X coordinate.
    ///
    /// </summary>
    public int X;
    /// <summary>
    /// Y coordinate.
    ///
    /// </summary>
    public int Y;
}

```

The IntPoint gives the coordinate of each pixel that is detected as an edge point. The edge points are divided into four categories in the blob detector output.

- leftEdge
- rightEdge
- topEdge
- bottomEdge

Each counter that is found by the Blob Detector is outputting a list of IntPoints which are the collection of all the points detected from left, right top and bottom. These points are used in the process of feature extraction in the later process.

6.2.3. The edge detectors

The edge detectors are implemented using convolution concepts. The edge detectors were used in order to remove the thick edges of the images capture and make the images inverted to have black background with single operator.

6.2.4. The matchers

The matchers were implemented in order to recognizing the shape with template or block matching techniques. But the implementation was not suitable for the current application we are developing now. Because the size and the shape can be different from one sketch to another. Therefore, we had to leave behind these implementations.

6.2.5. Shape Checkers

The idea of shape checkers was first arising from the library class AForge.Math.Geometry.SimpleShapeChecker. From that idea we implanted an interface called IShapeChecker with the following properties and behaviors.

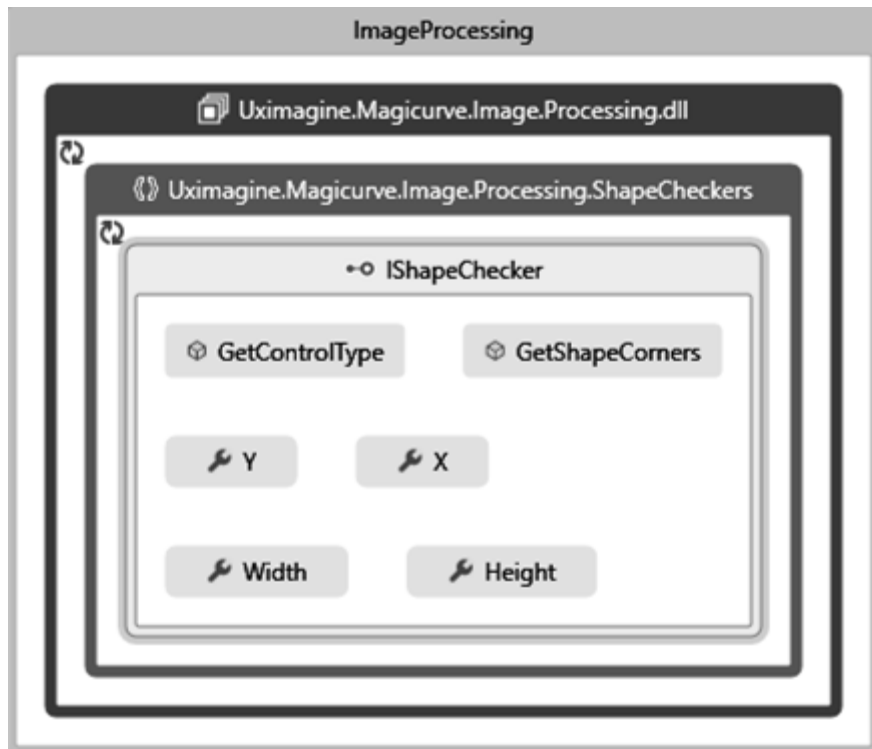


Figure 18: Shape Checker Structure

With this structure we implemented several shape checkers such as `UiShapeCheker` and `SvmShapeChecker` for rule based and neural net approaches. The `UiShapeChecker` class had the following implementation when determining the shapes. This implementation is done with simple logical assumptions like the button is rectangle and the rectangle should have couple of equal sides. And sytem have added a threshold value of 30 pixels in order to tolerate human error. This implantation approach is suitable for known shapes with known properties. But to add new shapes or to change the existing properties of a shape would become very hard with this approach. Still this very a perfect way to recognize shapes with known properties. The `SvmShapeChecker` was based on Support Vector machine and machine learning techniques. The code segment is attached at appendix B.

6.2.6. The filter helpers.

The filter helpers were actually set of image filters implemented in order to apply on Bitmap images.

There were several other filter helpers like Edge filters and convolution filters which was used to apply image filters on the image for operations like horizontal and vertical line detection and edge detection.

The code segment is an example for invert filter implemented with C# GDI+ interface is available at the appendix B.

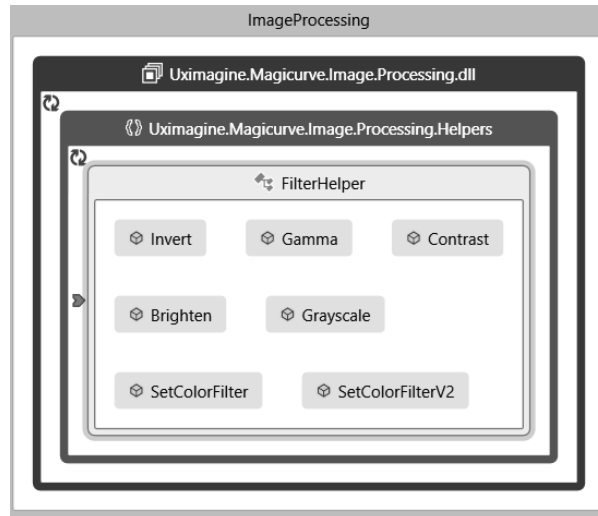


Figure 19: Filter Helper Implementation

6.2.7. The output

The output of this module is list of Control objects. The structure of the Control class is as in the figure 6-12. The X, Y, Height and Width properties are calculated from the bounding rectangle of the shape. The Type is recognized by the Shape checker classes. The Styles are added from code generator and UI tool. The edge points are only used within the Image processing module only.

This is the end of the implementation of Shape recognition module. Next the paper will describe UI tool and code generation.

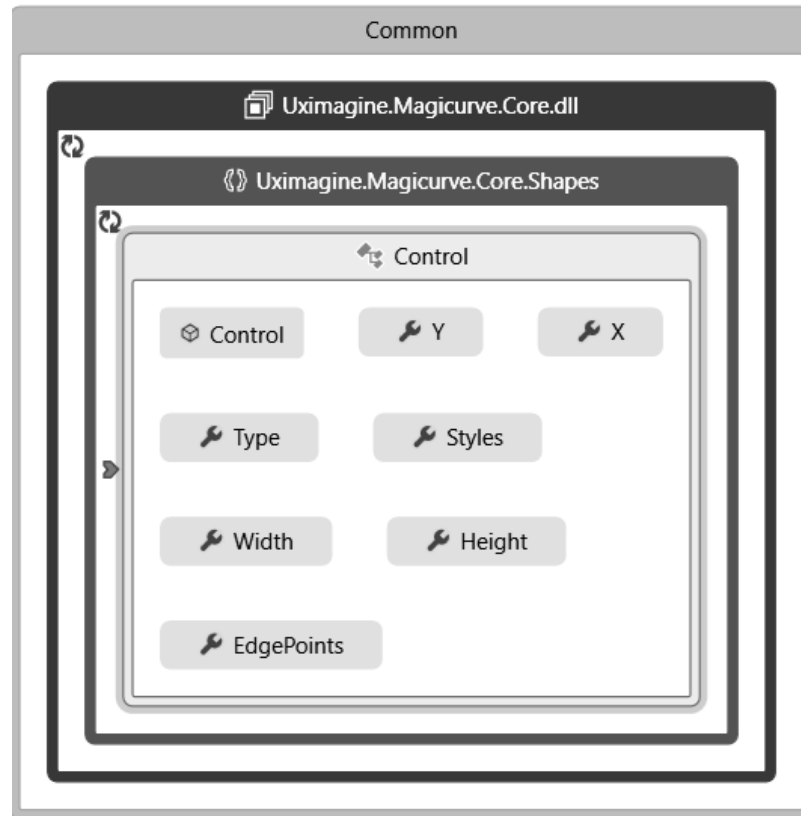


Figure 20: Control object for output

6.3. Customize Rich Tool Interface Development Module

We have defined 15 symbols for 15 HTML control tags. Those 15 controls can be managed in the web design and rest of the html tag controls add to design using tool box according to my implementation. Getting the most appropriate html tag controls, mapping the controls and creating rich customize environment are the major targets of this module. JavaScript framework which is named as Zebrajs is used to generate html tag controls. X and Y coordinates of every control is relatively root element and also they have height and width. Non editable view was created as first part of this module implementation. If the user wants to edit the design, it will pass whole in to shaper pan. Shaper pan give facility to drag and resize the all the html controls in the design panel. User has privilege to change the any css style of any control by right clicking on the relevant control (color, caption etc.). This is all the implementation done under zebrajs and Angularjs framework.

Every control has a unique ID that is related to mapping part. Recognized shapes with controls mapping list is sent as json object which has the attribute “type”. Type has unique ID 0 to 15 and 0 denoted as unrecognized control. This control set received as

input for tool interface developing algorithm. This algorithm shows how to those html controls allocation to design panel and who the parent tag and also how to set a resolution according to image content. The code segment is attached at appendix B.

Algorithm Steps:

1. First control categorizes under header content footer.
2. Every categorize give priority to each html controls.
3. Map to predefined control set.
4. Set resolution according to image height and width.

Sometime image segment components are not actual html controls. Therefore, it should give user to ability of changing it. Our tool provides rich environment to customization for that. All the changeable controls and attribute must be changed in the generating html code also. Angularjs is used to pass the value of the controls. If some modification has been done in design panel, it should be automatically updated generate html file.

6.4. Code Generation Module

First of all, the system need to order the controls before thinking of arranging them into web page. User draws sketch with controls from top to bottom of the page. But after detecting those shapes as controls in shape recognition module, it sends list of controls without any order with relevant feature or property set for each control. Therefore, there should be a mechanism or proper algorithm or set of algorithms to sort those controls with their properties. Even there were existing approaches for automatic code generation, researchers could not find algorithm with can be achieved our target output. Therefore, in this case study have built set of basic algorithms to achieve this purpose. You can clearly see implementation of those algorithms.

6.4.1. Controls Sorting – Step 1

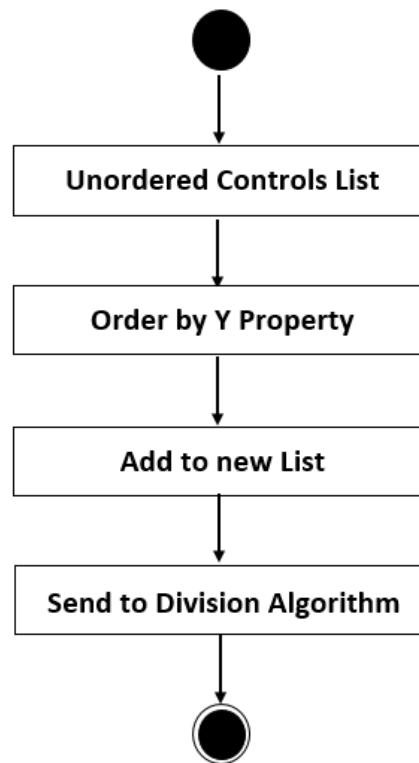


Figure 21: Sorting Algorithm Based on “Y” Property

When this `SortListYProperty()` method is called by parsing list of controls which is output of first module, it reads “y” attribute value of each controls and arranges controls from less y value to high y value from top to bottom. Finally, it returns a new list of controls with all the provided other properties of each controls. Then it passes those returned list into next algorithm of adding div tag for each controls.

6.4.2. Division Algorithm – Step 2

Division algorithm is used to divide relevant row of each ordered controls by providing div tag division based on height value and y property value. This is the way which identify the one control should place in one row and compared to row of previous control what is the row of next control. It decides that whether next control should place in same row or in another new row. It provides new attribute which named rowIndex as a flag for each control to identify its row. Then those decided list of row list with controls, row index values and row heights are inserted into list of row object. Those list of row object is sent to next algorithm to sort it again based on X property value. The code segment is attached at appendix B.

6.4.3. Sorting List of Rows – Step 3

This method gets out one row by row and identify controls per row and then those identified controls are sorted based on X property value of each controls. The code segment is attached at appendix B. Then output of list of rows are sent through main code generation process which is shown below.

6.4.4. HTML code generation

```
start generating main code passing controls list from module 1
    append header and body container with custom css
    get sorted controls based on y from step 1
    then send it into step 2 then that output into step 3
    get final row list controls
    for each row in final row list controls
        append div class "row" for each rows and set row height
        for each Control item in row. Controls
            append div class "col-md-*" for each controls in one row
            switch by item Type
                call each function based on item Type
                then generate code segments
                append each segments
            end switch
        end "col-md-*" div tags
    end inner for each
    end "row" div tags
end outer for each
append all other external scripts and the footer
return generated html into html file
end main code generation
```

Figure 22: Final Process Pseudo Code

6.4.5. Deciding Width of Columns

Automatically Code generation module have implemented according to above shown pseudo code with mentioned set of algorithms. As well as it is used above 6.4.5 logical step to decide col-md-* inside the given pseudo code. The code segment is attached at appendix B.

6.5. Summary

In this chapter, the paper has explained implementation details of each module with the help of flowcharts, pseudo codes, algorithms and code segments. Next chapter is going to describe about evolutions of the given solutions from those implementations.

7. Evaluation

7.1. Introduction

This is the chapter which is describing effectiveness and efficiency of each module of the project in term of graphs, tables with testing results and accuracy of each steps of development process.

7.2. Evaluation Techniques and Result

7.2.1. Shape Recognition and Describing

This module is test in Intel Core i5 2.2 GHz processor and 12 GB of Ram and the input image were in the quality of 96 dpi. The testing was done for each algorithm using unit testing techniques provided in visual studio and nunit framework.

- Testing for Blob Detectors

The blob detection accuracy was 100% with application of correct algorithms and threshold values for the input images. The blob detection was implanted using accord.net technology.

- Testing for UiShapeChecker.

The tests were running in order to detected each control with the input image. The accuracy of the shapes checkers was near 20% for many shapes. However, the controls like Button and radio button was 80% accurately recognized.

```
[TestCase(@"D:/Data/test/inputs/template4.jpg", 100, ControlType.Button)]  
[TestCase(@"D:/Data/test/inputs/image_05.jpg", 25, ControlType.Button)]  
[TestCase(@"D:/Data/test/inputs/image_06.jpg", 25, ControlType.Button)]  
[TestCase(@"D:/Data/test/inputs/image_07.jpg", 25, ControlType.Button)]  
[TestCase(@"D:/Data/test/inputs/radio_06.jpg", 25, ControlType.RadioButton)]  
[TestCase(@"D:/Data/test/inputs/combo_10.jpg", 25, ControlType.ComboBox)]  
[TestCase(@"D:/Data/test/inputs/combo_06.jpg", 25, ControlType.ComboBox)]  
[TestCase(@"D:/Data/test/inputs/combo_07.jpg", 25, ControlType.ComboBox)]  
[TestCase(@"D:/Data/test/inputs/combo_08.jpg", 25, ControlType.ComboBox)]  
[TestCase(@"D:/Data/test/inputs/combo_09.jpg", 25, ControlType.ComboBox)]
```

Figure 23: Test cases for shapes

7.2.2. Customize Rich Tool Interface Development

Testing result shows that 100% accuracy by correctly identifying html controls tag for a produced html controls for a given image. There can be a little amount of processing time to render html tags and time for sending changed html controls to code generator.

7.2.3. Automatic Code Generation

This module tested under set of test cases by providing set of sample data of different web pages. It is tested time to time with 2, 4, 6, 8, 10, 12 controls and check whether those controls are placed correctly in relevant location and how much accuracy in generated html codes. I built normal html pages with position absolute style for all controls as expected output pages for all tests. Then same values of X, Y, Width, Height of sample pages is given as input for testing purpose and compared generated actual output with expected output of corresponding pages. Below table is mentioned result of the evaluation.

Total Number of Controls	2	4	6	8	10	12
Correctly Placed in web page	2	3	5	7	8	10
Incorrectly Placed in web page	0	0	0	1	2	2
Accuracy	100%	100%	100%	87.7%	80%	83.33%
Average Accuracy	91.84%					

Table 1: Testing Results of Code Generation

Accuracy of generating code based on different number of controls in web page can be summarized as shown in above table. As well as the system would be able to reach more than 80% responsiveness of the generated pages with the help of adding bootstrap features.

7.3. Summary

This chapter describes the different ways that have been used to evaluate certain aspects of the system. The output has been quantified where possible and these results are used in the next chapter to draw up a final conclusion.

8. Conclusion & Further work

8.1. Introduction

The previous chapter is mentioned details and methods the system has used in order to evaluate the final output of our system. This chapter is going to focus on giving an overall discussion on the project and the tasks carried out up to this level. In here our aim is to draw a conclusion based on the evaluation chapter discussed above. Limitations of the proposed solution are also discussed. Further work that can be carried out in order to improve the system has also been stated

8.2. Achievement of Objectives

With the progresses of this project the study has been able to gather deep knowledge in areas of technology such as image processing, machine learning, HTML canvas and other javascript frameworks.

We have been able to develop a user friendly interface for magicurve which has the capability to generate html code form sketch image. And also this tool has the ability to manipulate the controls with user interaction. The tool is also capable giving live updates for the code as the user changes the image input of interface changes.

The Code generator is capable of outputting different styles of code like responsive design using bootstrap library or absolute layout design with over 80% accuracy. The code generator is also capable of generate code with additional code changes which user cannot input throw the sketch.

The shape recognition module is capable of recognizing defined shapes with high accuracy using the rule based approach. And the machine learning algorithm is improving its accuracy with the training.

The special feature of this tool is to host this as a web site in a cloud platform. Currently all the modules are integrated and functioning smoothly. The system can say that the project is success with high accuracy.

8.3. Limitations

The system need to high quality images for processing otherwise the accuracy of the tools does not meet the expectations.

8.4. Conclusion

This paper presents a system which can be used to generate html code using hade made sketch. This tool will serve UI and UX designers for saving their valuable time and web developers for rapid prototyping.

8.5. Further Work

Hosting the application in the windows Azure cloud will be taken place in the future and support for generating code for mobile design layout and on screen drawing support will be available in the next development versions.

8.6. Summary

This chapter provides an overview on what has been discussed in this report. It briefly explains the work that has been carried out. Limitations of the solution and the achievement of objectives have been highlighted. A conclusion has been drawn and further work that can be done to improve our proposed solutions has also been discussed.

References

- [1] B. Kohan, "Guide to Web Application Development," 2014. [Online]. Available: <http://www.comentum.com/guide-to-web-application-development.html..>
- [2] "User Interface Design," [Online]. Available: http://en.wikipedia.org/wiki/User_interface.
- [3] Q. Muktar, Interviewee, *Issues in ui automation tools*. [Interview].
- [4] R. O. Duda, P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, 1972.
- [5] D. Ballard, "Generalizing the hough transform to detect arbitrary shapes," Science Department, University of Rochester, NY 14627, U.S.A, 1980.
- [6] D. Shi, L. Zheng, J. Liu, "Advanced hough transforms using a multilayer fractional fourier method," *IEEE Transactions on Image Processing*, vol. 19, no. 1558–1566, 2010.
- [7] J. P. Lewis, "Fast Template Matching," Proc. of Canadian Image Processing and Pattern Recognition Society, Quebec, 1995.
- [8] R. Brunelli, Template matching techniques in computer vision, Wiley: ISBN 978-0-470-51706-2, 2009.
- [9] M. Kass, A. Witkin, D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4 pp.321–331, 1987.
- [10] Chenyang Xu, Jerry L. Prince, "Snakes, Shapes, and Gradient Vector Flow," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 7, no. 3 pp. 359–369, 1998.
- [11] Zakaria Mohd Firdaus, Choon Hoo Seng, and Suandi , "Object Shape Recognition in Image for Machine Vision Application," *International Journal of Computer Theory and Engineering*, vol. 4, no. 1 pp: 76-80, 2012.
- [12] C. J. Alvarado, "A Natural Sketching Environment: Bringing the Computer into Early Stages of Mechanical Design," 2000.
- [13] Andrew J. Korneck, Sona Johri, "Automatic Code Generation: Model – Code Semantic Consistency," CRM Guidant, Inc. St. Paul, MN.

- [14] Xiang-Hu Wu, Ming-Cheng Qu, Zhi-Qiang Liu, Jian-Zhong Li , "Research and Application of Code Automatic Generation Algorithm Based on Structured Flowchart," Journal of Software Engineering and Applications, 2011.
- [15] Adnan Abuarafah, A. Baith Mohamed, Ehab F. Amer, "New Revolution of HTML Code Generators," KING ABDULAZIZ UNIVERSITY-SCIENCE, vol. 20, 2008.
- [16] "Asp.Net," Microsoft, 2015. [Online]. Available: <http://www.asp.net>.
- [17] "Aforge.Net," 2014. [Online]. Available: <http://www.aforgenet.com>.
- [18] "Javascript advantages and disadvantagea," [Online]. Available: <http://www.jsripters.com/javascript-advantages-and-disadvantages/>.
- [19] "advantages javascript scripting languages," [Online]. Available: <http://www.coderanch.com/t/519745/HTML-CSS-JavaScript/advantages-javascript-scripting-languages>.
- [20] "Angular JS," Google, 2015. [Online]. Available: <https://angularjs.org>.
- [21] "Zebra JS," Sandtube, 2012. [Online]. Available: <http://www.lw-zone.org/word/>.

Appendix A

Individual Contribution

H.A.D.M Harishchandra

Before developing the Magicurve system we went to the Architecture Faculty of University of Moratuwa and discussed how we can draw the mockup (sketch design of website) controls. Final decision was getting survey details from 100 student of Architecture Faculty. The 1st thing that we did was brainstorming and got every details as much we could. Then divided system into components and start the development.

In creating the Magicurve system, my contribution is to create single page application using ASP.NET and angularjs. Its goal is to simplify both development and testing of such applications by providing a framework for client-side (MVC) architecture and also I have created a service for message passing to the shape inventory, the canvas and the property customizer. In addition to that I have created all the GUI Component of the project. Before I started to implement the system I gathered all the requirements. After that we analyzed requirement and tried to implement the system according to requirements.

At the beginning, I created a GUI of the system and includes all the prototype of brainstorming before. After that starting to create function for 15 html controllers to render on the design panel. According to requirement implement two function for editable and non-editable view. Editable view is the hardest task in this module and working with the mouse event. After this creation develop a context menu for right click. Finally, I mapped to algorithm and integrate with the json object send by image segmented. Final output in my part is given design of the sketch drawn by client and create rich customaries environment give to user. User changeable view parameter send to code generator part done by myself.

R.C.S Manoj

In my contribution to project magicurve, I have implemented the shape recognition module and the few other integration tasks based on asp.net technology with the help of my team members. After the project proposal was accepted we had to gather some data on which will be the most suitable for symbols to draw the sketch. Until that data is gathered I had to define my own symbols to continue with the project. With the help of my team members we defined 15 symbols for testing purposes.

Initially I had setup the Asp.net project architecture with few sub projects included. And we decided that we should use TFS online for version control and task break down. After setup project structure, we had to read many research papers regarding image processing and shape recognition. The whole team did read the research papers and gathered information on it. And also we search for tools which are very similar to the one we are going to implement. We came across some tool which generate code but not using image processing but drag and drop.

It was up to me to find a good image processing and machine learning framework in order to implement new feature using the existing implementation. I found Afroge.net is good choice to get along with. But latter I found a fork of the same project that is maintained well call accord.net so I switched to that version. It was very interesting when learning machine learning techniques with my own. But I took the image processing course module offered by the faculty which helped me a lot when dealing with the images. I tried to implement most of the image processing algorithms on my own with the parameter changes which are suitable for kind of images I used for testing.

I had to draw my own symbols for testing through I was not a good artist. But that was exactly the point. Do recognize the shapes with the maximum tolerance. First I used the web cam images for testing. Then the images from a quality camera. And finally scanned images with a digital scanner. Images from cameras contained too much noise and some areas had to be manually crop before processing. But in the real implementation the users cannot preprocess the image before submit them to the application. Therefore, it was agreed that we should only accept high quality images from a scanner. And I think by the end of this project we are ready to do our pilot release.

The assigned module for me was automatically generating html code based on output controls list of module 1. I have researched existing ways of doing this by referring research papers. The challenging point was arranging each controls corresponding right place of the generating page. Because the list of control is sent without any order of controls from top to bottom. And other point was making web page responsive way with the size of the screen. But the problem that I faced was difficulty of finding existing algorithms or existing proper methods to achieve target output of my module. Before starting more implementations of this task I started from very simple known ways of doing this. It means, I ordered controls based on its y property value and gave position absolute style for each controls. Then controls were located actual place without any conflict. But I could not achieve responsiveness of generated code. Therefore, I thought further about issue and tried to develop simple algorithm by using and thinking properties of controls in the list such as height, width, y, and x and so on. After getting more time I came up with algorithm which developed to add div tags for all controls as the way responsive of web page is achieved by bootstrap. This algorithm is used to divide whole page into rows and add controls relevant row by adding class “row” for the div. Then I have tested my algorithm by giving sample data manually with several test cases.

As well as I have developed another method to sort controls base on x property and then another method to divide each row into columns. While I were thinking about ways that I can get final code in code generation part same time I thought how It will effect to other two modules also. Because when I develop my part without thinking other external factors such as the way user think and draw sketch and output of the module 1, it will be conflicting between input controls list and generated output. Finally, after I have implemented set of methods with proper algorithms, I finalized all those into one path and combined all into main output generating part by integrating all string segment of codes into one html output into html file. Not only that I have tested each and every algorithm and whole implantation under set of test cases and illustrated all those result with the help of graphical representation such as graphs in the final documentation. End of the days of final preparation of the project, I spent time to understand whole system as one and other module implementations with the help of other members of the project and gained much knowledge related to all areas covered by the project.

Appendix B

```
1 reference | Chandima Ranaweera, 6 days ago | 1 author, 4 changes
protected bool IsButton(List<IntPoint> edgePoints, List<IntPoint> corners)
{
    var isButton = false;
    const double distanceError = 30;

    if (this.CheckIfPointsFitShape(edgePoints, corners))
    {
        if (corners.Count == BUTTON_CORNER_COUNT)
        {
            //get length of each side
            var sides = new float[corners.Count];
            var next = 1;
            for (var i = 0; i < corners.Count; i++)
            {
                if (i == corners.Count - 1)
                {
                    next = 0;
                }

                sides[i] = corners[i].DistanceTo(corners[next++]);
            }

            if (sides[0] - sides[2] < distanceError && sides[1] - sides[3] < distanceError)
            {
                isButton = true;
            }
        }
    }

    return isButton;
}
```

Button Recognition Algorithm

```

public static Bitmap Invert(this Bitmap image)
{
    Bitmap bmap = (Bitmap)image.Clone();

    // GDI+ still lies to us - the return format is BGR, NOT RGB.
    BitmapData data = bmap.LockBits(new Rectangle(0, 0, bmap.Width, bmap.Height),
        ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);

    int stride = data.Stride; // width of single line.
    IntPtr scan0 = data.Scan0; // Pointer to the data.

    unsafe
    {
        byte* ptr = (byte*)(void*)scan0;
        int nOffset = stride - bmap.Width * 3; // calculate the padding.
        int nWidth = bmap.Width * 3; // steps for width * BGR
        for (int y = 0; y < bmap.Height; ++y)
        {
            for (int x = 0; x < nWidth; ++x)
            {
                ptr[0] = (byte)(255 - ptr[0]);
                ++ptr;
            }

            ptr += nOffset; // skip the padding.
        }
    }

    bmap.UnlockBits(data);

    return bmap;
}

```

Figure: Invert filter algorithm

```

function drawMenuBar(menuX, menuY, menuW, menuH) {
    var menu = new zebra.ui.MenuBar
        ({
            "HOME": {
                "Subitem 1.1": null,
                "Subitem 1.2": null,
                "Subitem 1.3": null
            },
            "CONTENT": {
                "Subitem 2.1": null,
                "Subitem 2.2": null,
                "Subitem 2.3": null
            },
            "CONTACT": null
        });
    menu.setBounds(menuX, menuY, menuW, menuH);
    return menu;
}

```

Function for Drawing Menu Bar

```

public List<Row> DivAlgorithm(List<Control> list){
    double defaultHeight = 40;
    double maxHeight;
    List<Row> listOfList = new List<Row>();
    Control minYControl = list[0];
    int rowIndex = 0;
    if (list[0].Height < defaultHeight){
        maxHeight = defaultHeight;
    }else{
        maxHeight = list[0].Height;
    }
    listOfList.Add(new Row(){
        Controls = new List<Control>(){list[0]},
       RowIndex = 0,
        Height = maxHeight
    });
    for (int i = 1; i <= (list.Count) - 1; i++)
    {
        int previousY = list[i - 1].Y;
        int currentY = list[i].Y;
        if (currentY > previousY + maxHeight){
            if (list[i].Height > defaultHeight){
                maxHeight = list[i].Height;
            }
            else{
                maxHeight = defaultHeight;
            }
            listOfList.Add(new Row(){
                Controls = new List<Control>(){list[i]},
               RowIndex = ++rowIndex,
                Height = maxHeight
            });
        }
        else{
            if (list[i].Height > maxHeight){
                maxHeight = list[i].Height;
            }
            listOfList[rowIndex].Controls.Add(list[i]);
            listOfList[rowIndex].Height = maxHeight;
        }
    }
    return listOfList;
}

```

Division Algorithm Code Segment

```

public List<Row> SortListXProperty(List<Row> rowList){
    List<Row> finalizeRowList = new List<Row>();
    foreach (Row row in rowList){
        List<Control> finalizeControllist = new List<Control>();
        var query = from con in row.Controls
                     orderby con.X
                     select con;

        finalizeControllist = query.ToList();
        finalizeRowList.Add(new Row()
        {
            Controls = finalizeControllist,
            Height = row.Height,
            RowIndex = row.RowIndex
        });
    }
    return finalizeRowList;
}

```

Sorting Algorithm Based on X property

```

public int GenerateColSizeAlgo(Control item, double pageWidth)
{
    double value = (item.Width / pageWidth) * 12;
    int colSize = (int)(Math.Round(value));
    if (colSize < 2)
    {
        colSize = 2;
    }
    return colSize;
}

```

Deciding Column Width