CANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report
Version 1

CENG 408
Innovative System Design and Development 2

Design Parser for IOS and Android

Mustafa IPEK 201611031
Vural Can SISMAN 201611055
Erhan YUMER 201611067

Advisor: Assoc. Prof. Dr. Aydin KAYA

Table of Contents

LITERATURE REVIEW

1. ABSTRACT

The area of parsing has been desolate in the last few decades but still stays an open research problem. The aim of the design parser project will be to be able to receive and process our very own markup language in Java language. Many developments have been mentioned in our study, most of them as open-source projects and libraries to help front-end programmers. In this paper, we aimed to summarize the computer science research literature that relates to implementations of parser projects and we surveyed the broad computer science literature to evaluate how this area has been studied from a technological and scientific perspective.

2. INTRODUCTION

In today's world, people are very prone to creating their websites, Android, or IOS apps. Besides, more people mean more websites, more apps and more work to be done. The use of parsing is usually helping frontend programmers in coding and saving time. Therefore, parsing various types of codes has been an attractive and promoter task for our team. On the other hand, there are lots of challenges in parsing projects like collecting info about languages, definition of variables to convert, selecting the labeled data, etc. Due to the many challenging tasks under evaluation, these parsing systems need a huge amount of knowledge including data structures, visual design, and code evaluation technologies.

3. PREVIOUS WORK DONE IN THIS FIELD

Despite design parsing being a desolate field to study in Computer Science, hundreds of works have been done in past years. We decided to look to them as a team if they are an inspiration to our team and we can catch their gap then fill them to make the design parsing field a greater area.

a) <u>jsoup : Parser for Java to Html</u>

Jsoup can do most of the things that we want from a lot of Html pages. Jsoup can make them usable to make applications. For example, let us say we want to convert a website into a mobile application; Jsoup gathers data by parsing the tags on the Html page of the website and fits it in the application. Jsoup is an open-source Html parser library made with Java programming language.

b) Postcss

PostCss is usable as a parser for making styles with JS plugins. Unlike others, PostCss's focus is the design part. PostCss is used by big companies in the market including Wikipedia, Twitter, Alibaba, and JetBrains. One of the PostCss plugins, "The Autoprefixer" is popular in most CSS processors. PostCss takes a CSS file and supplies an API to analyze and change its rules (by converting them into an Abstract Syntax Tree). Then this API can be used by plugins to make useful things, as finding errors automatically.

c) Php Parser

Php Parser has many features like:

- Parsing PHP 5-7 code into an abstract syntax tree(AST).
- Translates an abstract syntax tree(AST) back to PHP code.
- Building infrastructures to traverse and modify ASTs.
- Building and simplifying abstract syntax tree(AST) construction for the code generation.
- Converting abstract syntax tree(AST) to JSON and vice versa.

d) JSX

JSX is an XML-like syntax tool without any described pattern; it is not intended to be implemented by browsers(Chrome and Firefox). It is converted to be used by various preprocessors to convert these tokens. Originally JSX was parsed via a Facebook fork of Esprima which we will mention later.

e) Parsedown

Parsedown's system works to read markdowns exactly as we humans do. First, it looks at the lines. It tries to understand how lines started. Once it recognizes the lines, then it continues to rest. While it is reading, it looks for special characters simultaneously. This feature helps to recognize inline elements.

This approach is called a "line-based" approach. Since the release of Parsedown, other developers who have interest in parser projects have used the almost same approach to develop other markdown parsers in PHP and other languages.

f) Esprima

Esprima is an ECMAScript parser set up in ECMAScript (also known as JavaScript). Esprima has higher performance than its opponents. It is released and sustained by Ariya Hidayat, with the help of his teammates.
Esprima has many features like:

- Logical syntax tree format as standardized by ESTree project.
- Tentative support for JSX, a syntax extension for RN.
- Optional tracking of syntax node location.

Also, Esprima is tested harshly at about 1500-unit tests with full code coverage.

g) Globalize

Globalize has features like formatting numbers and parsing them.

- Allows developers to load as much data as they need.
- Avoids duplicating data.
- It keeps code portable and customizable which means developers can change it according to their will.
- Runs in browsers, also on Node.js, continuously across all the mentioned platforms.

4. CONCLUSION

We looked at many parser open-source projects on the internet and saw their strengths and weaknesses. Besides, we have listed the top 6 parser libraries/frameworks that work as our project. Usually, people choose to read the input character by character, separate by tokenizing, placing them in a tree, and giving the final output to the user.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1. INTRODUCTION

1.1 Purpose

Design Parser for Android and IOS aims to convert HTML like syntax to Java and Swift code. It is a cross-platform mobile code converter. Android and IOS applications UI design are sometimes so hard to develop for responsive applications because there are many phone types and phone sizes all of them have different features. There are several ways of creating responsive design with pure Java and Swift code, but this way is too hard and too long-time coding for developers, so we will create a new HTML like template design language. We will describe all syntax features for it with this syntax programmer will create easy and short time coding.

1.2 Road Map

Our job will have five basic parts:

First, we will create a list with Android, and IOS UI features like Button, Image, View, ListView, etc.

Secondly, we will create a language or syntax for these Android and IOS UI features because we will create a single code which will be written by a programmer who wants to develop a mobile application, and we will convert this one code to Android(Java) and IOS(Swift) platforms code so our syntax should work these two platforms robustly.

Thirdly, we need a parser to convert our syntax to Android(Java) and IOS(Swift) code. This parser will read our syntax character by character and divide it into tokens for put syntax tree which will be created by our coding requirements and also with these tokens we will handle Android and IOS UI features Button, Image, View, etc. We will handle all these different features separately, and this will give us good coding opportunities.

Fourthly, we should create a converter that walks on our syntax tree, which is generated in the third step for converting our syntax to Android(Java) and IOS(Swift) code.

Finally, we will create a CLI(Command Line Interface) to create new projects or run projects, or generate Android and IOS system packages. With this CLI, the programmer can interact with our application.

1.3 Definition

OS	Operating System
UI	User Interface
CLI	Command Line Interface

1.4 Overview

This document is prepared to give details, technical information, and required specifications for this project's purpose.

2. OVERALL DESCRIPTION

2.1 Product Perspective

This product will be designed to produce mobile apps but firstly for the view of apps. Because we can use some IDE or use many programming languages but making mobile apps is often hard to create a responsive view. Some design languages (Html, XML ...) are easy to use for design, and we can inherit our project on them.

Our project will create a responsive app with more than one language, and it will be used with an easy using design language like Html, etc.

2.1.1 Development Methodology

We will consider android tools (Image, Layout, etc.) and IOS tools (TextView, Scroll Bar) to create a design language.
Our designed language should include a layout hierarchy. Every design tool should be in a layout so we can place it on the screen. The code which we will create will be suitable for the mobile environment.
We need a parser to connect this to a developed method, So we will write a parser to convert a design syntax which we create to syntax which is code for a mobile environment.

2.2 User Requirements

Users who will use this technology should have basic knowledge about any design language. Also, users can have some programmer skills. This skill may help them to use the technology efficiently.

2.3 Rules

There is a syntax for our design and runtime programming language. So all users who use this technology should comply with this syntax.

2.4 Problems

The project is a library-like project, and it runs on runtime, so it can be slow or may crash.

3. REQUIREMENTS DEFINITION

3.1 External Interface Requirements

3.1.1 System Interfaces

This part is explained in 3.2 Functional Requirements.

3.1.2 User Interfaces

The software will work actively on all platforms with Java Language installed.
Our program will most likely run on a command prompt.

3.1.3 Hardware Interfaces

The minimum requirement is: Intel® Core™ i3 processor or AMD Phenom X4 processors, almost 100MB of disk space and 4 GB RAM.

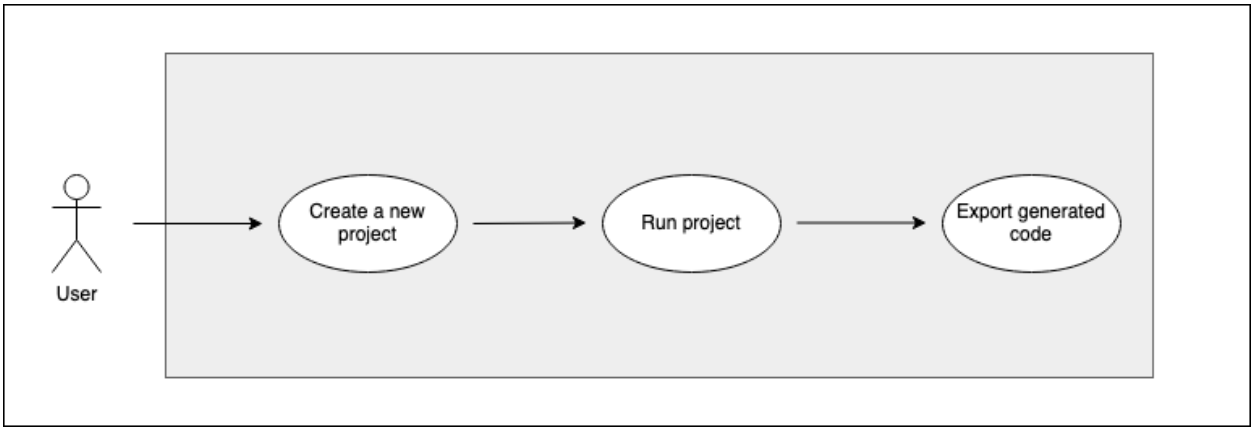
3.1.4 Software Interfaces

The minimum requirement is having Java Language and an IDE. Also, macOS, Windows 7(or later), or Linux.

3.1.5 User CLI (Command Line Interface) Use Case

User can do:

- Create a new project
- Run project
- Export parsed code



3.2 Performance Requirements

The minimum size of system requirements are as follows:

Processors: Intel® Core™ i3 processor or AMD Phenom X4

Disk space: ~100MB

OS: macOS, Linux, and Windows 7 or later

Java versions: Java 8

Compatible tools: Visual Studio Code, Eclipse, Android Studio, Xcode.

3.3 Software System Attributes

3.3.1 Reliability

We did some reliability tests on our software and got almost the same running times at the same input sizes. So, we can say our program is reliable and feasible.

3.3.2 Availability

The program will work on Windows 7 or later, macOS and Linux.

3.3.3 Security

Since our software will be a parser program that translates between languages, there will be no security prevention at release. We may add security options in the future when dealing with server-side code generation.

3.3.4 Maintainability

To increase the stability of the application, the developer team will update project files once a month.

SOFTWARE DESIGN DESCRIPTION (SDD)

1. INTRODUCTION

1.1 Purpose

Our purpose is basically to ease making designs for mobile applications and make them responsive.

1.2 Scope

In mobile development, there are some ways of creating a responsive design, but these are hard and take so long. And mobile phone development should be easy, like creating a web page. There are many ways some tools like react-native, flutter, etc., but using them are a lot of version problems using libraries.

Our scopes are to make a design language like HTML and decide how to create responsive design easily with code in runtime and fit it with our design language and provide users Java or Swift code.

1.3 Glossary

Term	Definition
SDD	Software Design Document
CLI	Command Line Interface
User	People who use the parser system.
UI	User Interface

1.4 Overview of the Document

This document is formed to explain how this project works and why it is designed like that.
We are creating a design language to define common GUI design for ios and android. This language has a syntax, so we describe some rules. To these rules, we will convert this design to code to create a design in runtime for both iOS and Android.

1.5 Motivation

We worked on a few mobile applications, and we had created some mobile apps for several years. Relying on our experience, we have decided to do a project about mobile applications, and we have an issue with the design part. In Android, making designs with XML is not comfortable and not responsive. But with creating design with Java and Swift giving responsive design but it is so hard and writing a lot of code.
We realize that in code, we can make responsive designs and make them easy to design. And we will create a design language that fits with that logic and convert it to Java and Swift code.

2. DESIGN OVERVIEW

2.1 Description of a Problem

Our main issue in the project is to save more time for people who code and make work easier for them. We will define our parser functions and classes, then combine them.

2.2 Technologies Used

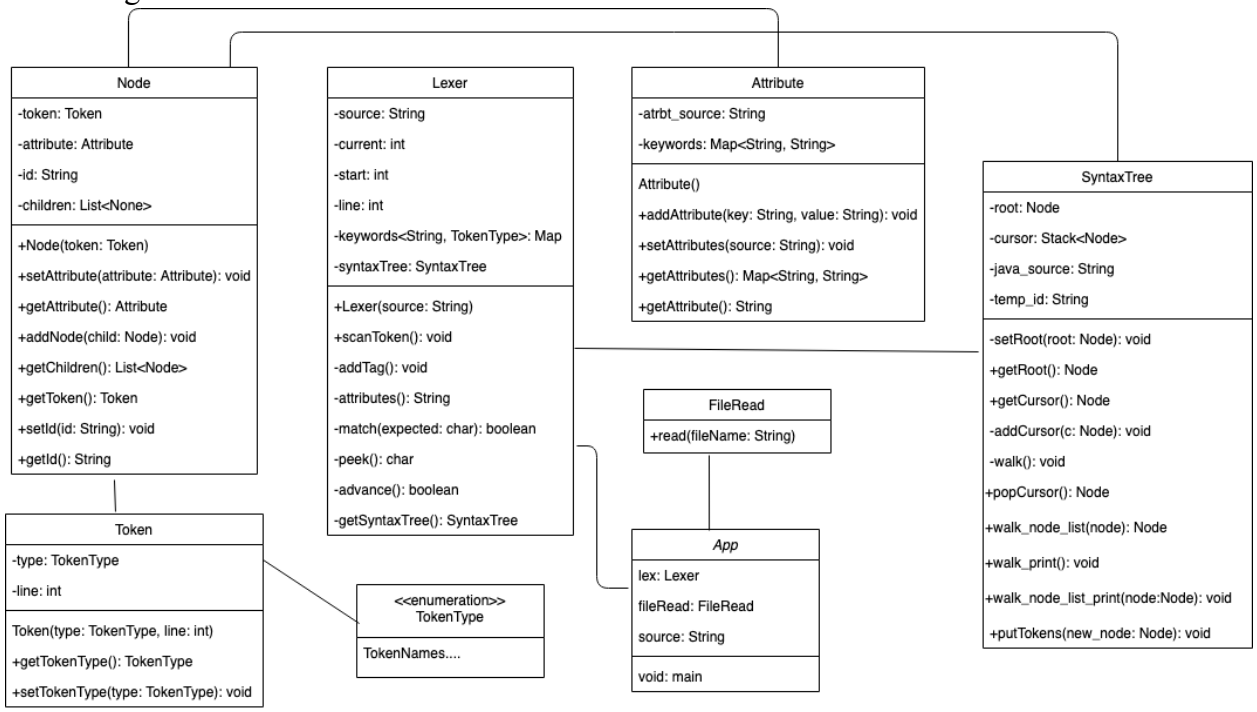
We will use the Java programming language and compile Android codes, Android Studio, and XCode to compile iOS code, and Eclipse IDE to run our program. The project's target platforms will be Windows, Linux, and macOS.

2.3 Architecture Design

2.3.1 Design Approach

At the start of this project, we decided to use the Agile development methodology since this project will be a long one, and we must respond to the changes very quickly and nimbly on this run. Like customers ' changing requests during the project, our decisions to make things better etc. Also, communication and teamwork are very important in agile development methodology and we will use this as an advantage on our side.

- Class Diagram



2.3.2 Architecture Design of Design Parser

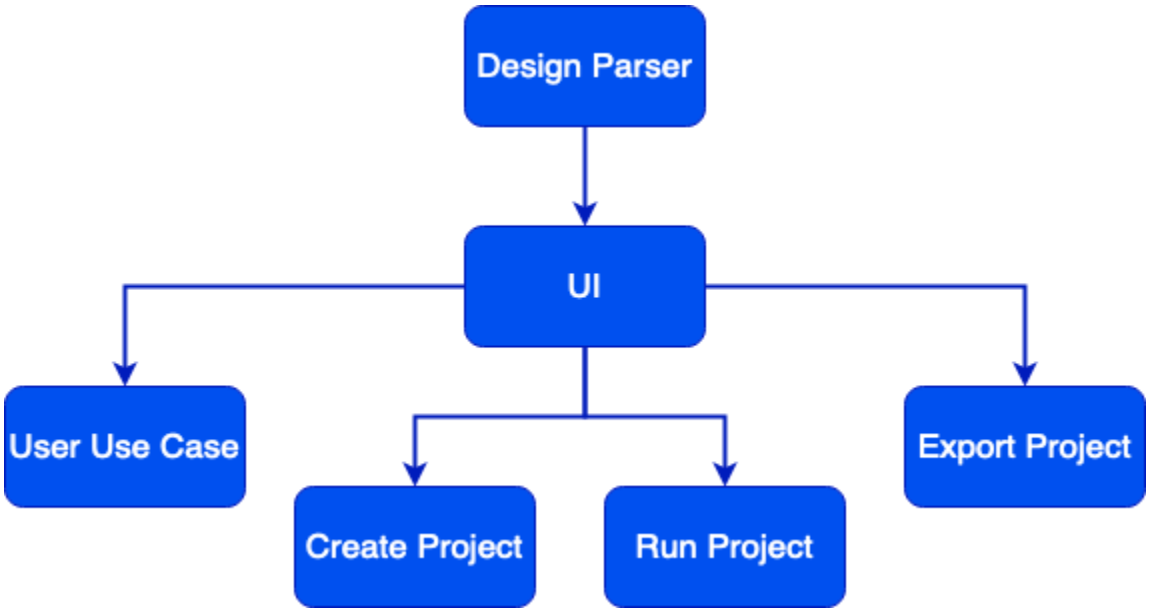
- User Menu

  Summary: The user will be able to use the following commands in the terminal:
  -create_project
  -run_project
  -export_project

  Initial step by step user guide after the program installation on the computer:

1. User opens a terminal window on the operating system they use.
2. Enter "-create_project" to the terminal.
3. Write their code.
4. Enter "-run_project" to the terminal.
5. If they want to export their code to somewhere else, they can enter "-export_project" in the terminal.

3. USE CASE REALIZATION



3.1 External Interface Requirements

- Create Project: Works through terminal command “create_project”. Creates a new project in the specified file for the user.
- Run Project: Works through terminal command “run_project”. Shows in the emulator.
- Export Project: Works through terminal command “export project”. Create a file in the project folder, put the iOS or Android code in it.

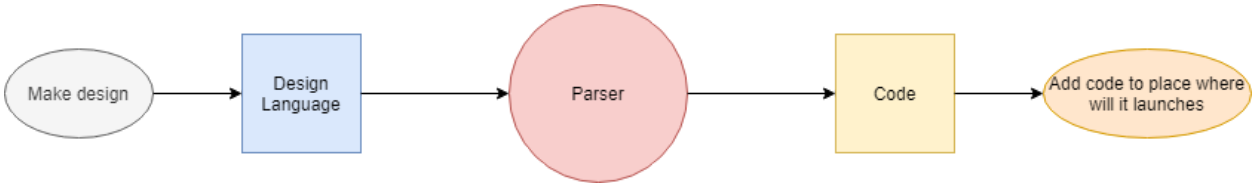
3.1.1 UI Design

In our UI, there is just CLI, and the terminal user can interact with the terminal with CLI. In our CLI there are some features. These are some options, creating a new project, running a project, exporting files for Android or IOS.

We do not require any sub-title since users will just write their code in any text editor and wait for output. In the “Create project” title, some files will be added to the user's working directory. In the “Run project” title, the program will work on an emulator. In the “Export project” title, users will get Java or Swift code. We will show our program's details about what we did in the latest update, the date of the latest update etc.

4. PARSING

Parsing is part of this project. With parsing, we are connecting two parts of the project, which are design language and its code state. If the design code is written correctly (if there is no syntax error), we can convert it to code for mobile apps to launch at runtime.



- **Make Design:** The user makes a design with the design language we have created.
- **Design Language:** Determining syntax and rules.
- **Parser:** The parser converts design code to syntax tree to create runtime code for iOS and Android.
- **Code:** Separate code according to the device. If it's Android, convert it to Java. If it's iOS, convert it to Swift.
- **Add code to the place where it will launch:** Create a CLI to create new projects or run projects.

TEST PLAN

1. INTRODUCTION

1.1 Overview

Our program will work only on a terminal which is our UI as we mentioned in SRS and SDD documents. So, in our program we only have one mode which we will mention as UI.

1.2 Scope

This document includes detailed test plans for the Design Parser project.

1.3 Terminology

Acronym	Definition
UI	User Interface

2. FEATURES TO BE TESTED

In this section, we will look over brief descriptions of all the major features to be tested. For each major feature, we will add a Test Design Specification at the end of the document.

2.1 User Interface

Contains 4 commands that provide access to our system completely.

3. ITEM PASS/FAIL CRITERIA

3.1 Exit Criteria

- Create a new project
- Run Project
- Export parsed code

5. TEST DESIGN SPECIFICATION

User Interface

Sub Features to be Tested

- Creating Project (UI.CP): All users have to create a project to start using parser. After creating a project, a text message “project created” will be shown to screen.

- Running Project (UI.RP): After entering the run project command, a text message “run project completed” will be shown.

- Export Project (UI.XP): A text message “project exported” that notify the user about the project being exported will be shown on the screen.

- Parser Help (UI.HLP): Users can look for help in our system. After entering this command, texts about how to use the parser system and commands will be shown on the screen.

- Parser Runtime (UI.RT): Checks if the parser program can convert source code to iOS and Android code.

- Parser Syntax (UI.SYN): Checks if the user has designed a valid source code to parse.

Test Cases

TC ID	Priority	Scenario Description
UI.CP	High	Enter the “parser-create” command. After entering, the project is created.
UI.RP	High	Enter the “parser-run” command. After entering, the project is runned.
UI.XP	High	Enter the “parser-export” command. After entering, the project is exported.
UI.HLP	High	Enter the “parser-help” command. After entering, a list of commands will appear.
UI.RT	High	Create and save proper design code. Then, the parser should be able to convert code.
UI.SYN	High	Parser controls users’ code for validity at syntax.

6. DETAILED TEST CASES

6.1 UI.CP

TC ID	**UI.CP**
Purpose	Create the project.
Requirements	-
Priority	H
Estimated Time Needed	1 sec
Dependency	-
Setup	Open terminal.
Cleanup	Close terminal or enter “clean” command.

6.2 UI.RP

TC ID	**UI.RP**
Purpose	Run the project.
Requirements	-
Priority	H
Estimated Time Needed	1 sec
Dependency	-
Setup	Open terminal.
Cleanup	Close terminal or enter “clean” command.

6.3 UI.XP

TC ID	**UI.CP**
Purpose	Export the project.
Requirements	6.2
Priority	H
Estimated Time Needed	5 sec
Dependency	-
Setup	Open terminal.
Cleanup	Close terminal or enter “clean” command.

6.4 UI.HLP

TC ID	**UI.HLP**
Purpose	See help commands about parser.
Requirements	-
Priority	H
Estimated Time Needed	5 sec
Dependency	-
Setup	Open terminal.
Cleanup	Close terminal or enter “clean” command.

6.5 UI.RT

TC ID	**UI.CP**
Purpose	Seeing the parser program be able to convert the written code..
Requirements	A saved and running program.
Priority	H
Estimated Time Needed	30 sec
Dependency	-
Setup	Write a proper program.
Cleanup	Delete source code file or close terminal.

6.6 UI.SYN

TC ID	**UI.SYN**
Purpose	See if the program throws error messages.
Requirements	A saved and running program.
Priority	H
Estimated Time Needed	5 sec
Dependency	-
Setup	Write your source code with some syntax errors.
Cleanup	Write over your source code with errors included.

TEST RESULTS

1. INDIVIDUAL TEST RESULTS

TC ID	**Priority**	**Date Run**	**Run By**	**Result**	**Explanation**
UI.CP	H	25.05.2021	Erhan YUMER	Pass	Project is created.
UI.RP	H	25.05.2021	Erhan YUMER	Pass	Project is runned.
UI.XP	H	25.05.2021	Erhan YUMER	Pass	Project exported.
UI.HLP	H	25.05.2021	Erhan YUMER	Pass	A list of commands appeared.
UI.RT	H	25.05.2021	Erhan YUMER	Pass	Parser converted the code.
UI.SYN	H	25.05.2021	Erhan YUMER	Pass	Parser controlled users' code for validity at syntax.

2. SUMMARY OF TEST RESULTS

We have executed 6 high priority test cases and 6 test cases are passed. Exit criteria is met.

Priority	**Number of TCs**	**Executed**	**Passed**
H	6	6	6
M	0	0	0
L	0	0	0
Total	**6**	**6**	**6**

3. EXIT CRITERIA

We have executed all test cases and 100% of high priority test cases are passed. Software development activities are completed within the anticipated cost. Software development activities are completed within the anticipated timeline. Exit criteria is met.

Criteria	**Met or Not**
100% of the test cases are executed.	M
100% of the test cases are passed.	M
100% of High Priority test cases passed.	M
No high priority or severe bugs are left outstanding.	M
Verify if software development activities are completed within the projected cost.	M
Verify if software development activities are completed within the projected timelines.	M

4. KNOWN PROBLEMS

No known problems have been encountered in test results.

5. CONCLUSION OF TEST RESULTS

This section includes the test results of the project “Design Parser for Android and iOS”. The test cases are implemented and 100% of the test cases are completed successfully. Software development activities are completed within the anticipated cost. Current stage of the project is available to use.

CONCLUSION

At the end of CENG407 & CENG408 course, the development stages of a software project, literature review and its importance, moreover, writing Software Requirements Specification and Software Design Documents, were mastered on a real-world project. In this process, Literature Review, Software Requirements Specification, Software Design and Test Plan documents were written. Details of the studies can be examined in these reports.

ACKNOWLEDGEMENT

We are grateful for the guidance we have received from *Assoc. Prof. Dr. Aydin KAYA*. The help we received from him was a great asset to improve this project and ourselves.

REFERENCES

[1] ai. (2017, January 16). *Postcss*. Retrieved from https://awesomeopensource.com/project/postcss/postcss

[2] ai. (2013, September 25). *Postcss/postcss:Transforming styles with JS plugins*. Retrieved from https://github.com/postcss/postcss

[3] nikic. (2017, January 16). *Php Parser*. Retrieved from https://awesomeopensource.com/project/nikic/PHP-Parser

[4] nikic. (2014, September 14). *Nikic/PHP-Parser: A PHP Parser written in PHP*. Retrieved from https://github.com/nikic/PHP-Parser

[5] erusev. (2017, January 16). *Parsedown*. Retrieved from https://awesomeopensource.com/project/erusev/parsedown

[6] erusev. (2013, July 21). *erusev/parsedown: Better markdown Parser in PHP*. Retrieved from https://github.com/erusev/parsedown

[7] ariya. (2017, January 16). *Esprima*. Retrieved from https://awesomeopensource.com/project/jquery/esprima

[8] ariya. (2015, March 11). *Jquery/esprima: ECMAScript parsing infrastructure for multipurpose analysis*. Retrieved from https://github.com/jquery/esprima

[9] scottgonzalez. (2017, January 16). *Globalize*. Retrieved from https://awesomeopensource.com/project/globalizejs/globalize

[10] scottgonzalez. (2014, April 14). *Globalizejs/globalize: A JavaScript library for internationalization and localization that leverages the official Unicode CLDR JSON data*. Retrieved from https://github.com/globalizejs/globalize

[11] Hedley, Jonathan. (2010, June 8). *JSoup Java HTML Parser, with the best of HTML5 DOM methods and CSS selectors.* Retrieved from https://github.com/jhy/jsoup

[12] Facebook. (2019, March 28). *JSX*. Retrieved from https://facebook.github.io/jsx/

[13] IEEE. "IEEE Std 1016-2009 IEEE Standard for Information Technology-System Design-Software Design Descriptions". IEEE Computer Society. July, 20 2009.

[14] CENG408_Design_Parser_SRS_V1.0, December 4, 2020.

[15] CENG408_Design_Parser_SDD_V1.0, January 5, 2021.