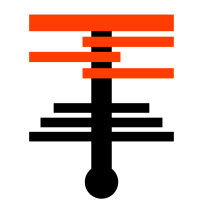


Project Submitted In the Context of the Course   
Info3301 **Software Engineering**



**Developed By  
Jad Mrad  
George Raed**

Lebanese University

Faculty of Sciences I

Department of computer Sciences

2017 - 2018

# Acknowledgements

If any.

# Abstract

The chosen application/domain and the major steps done while achieving the project

# Table of Contents

[Acknowledgements 2](#_Toc533761743)

[Abstract 3](#_Toc533761744)

[Table of Contents 4](#_Toc533761745)

[Table of Figures 5](#_Toc533761746)

[Table of Tables 6](#_Toc533761747)

[Chapter I - Introduction 7](#_Toc533761748)

[I.1. The business domain 7](#_Toc533761749)

[I.2. About the modeled application 7](#_Toc533761750)

[2.1. Users Persona 8](#_Toc533761751)

[2.2. Analysis of the Existing Similar Programs 8](#_Toc533761752)

[I.3. Plan of the document 9](#_Toc533761753)

[Chapter I - Requirement Analysis and Specification 9](#_Toc533761754)

[I.1. Introduction 9](#_Toc533761755)

# Table of Figures

[Figure 1 Programming Language Market Shares - Early 2018 7](#_Toc533770345)

[Figure 2 - Oracle SceneBuilder 1.1 Early Release 9](#_Toc533770346)

[Figure 3 - Gluon SceneBuilder 8 10](#_Toc533770347)

# Table of Tables

[Table 1 Comparison of Existing Similar Programs & Frameworks 8](#_Toc533619573)

1. Introduction
   1. The business domain

This project fits into the software development domain and is specifically directed towards Java programmers working on JavaFx or GUIs in general. We believe this is an ideal community to target as it is sizable with 3 Billion devices running java and its sizable market hare at 22.9% as can be seen in figure 1. We also chose JavaFx as a platform to build upon as it is the most recent officially released framework for creating graphical user interfaces with updates coming frequently and as recent as 2 hours[[1]](#footnote-1) before this report was written.

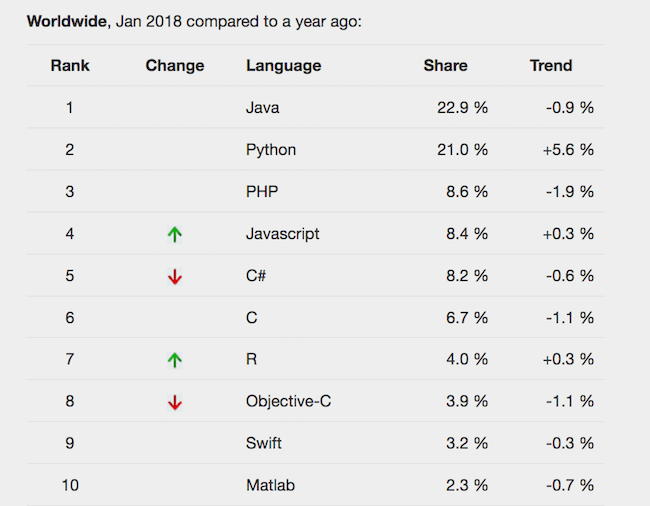


Figure 1 Programming Language Market Shares - Early 2018

In an ever-expanding community we believe to have stumbled upon a requirement that no one has provided yet based on our research. As will be discussed in detail below, any Java developer will in one way or another need our program be-it for large scale collaborations or a simple personal project.

* 1. About the modeled application

The program is intended to be used by anyone from beginners to experienced developers to design and implement JavaFx completely code-free. It can also be utilized by anyone who doesn’t know how to code which is useful on clients during the requirement gathering stage. With many more possibilities, the community needs such a code-free program not to mention the blueprints system that is so far unseen for JavaFx.

* + 1. Users Persona

Some key cases where our product would be optimal:

* Java developers creating GUI assets or prototypes with integrated logic
* Designers setting up a projects layout without requiring any application code
* Newcomers to JavaFx that wish to explore its potential
* Clients to show Software Engineers what they want
* Swing and/or AWS users switching over to JavaFx syntax
  + 1. Analysis of the Existing Similar Programs

There are quite a few programs and frameworks that could be considered alternatives or competitors to ours yet none of them carry the functionalities that EasyFx provides, mainly the handler management blueprints.

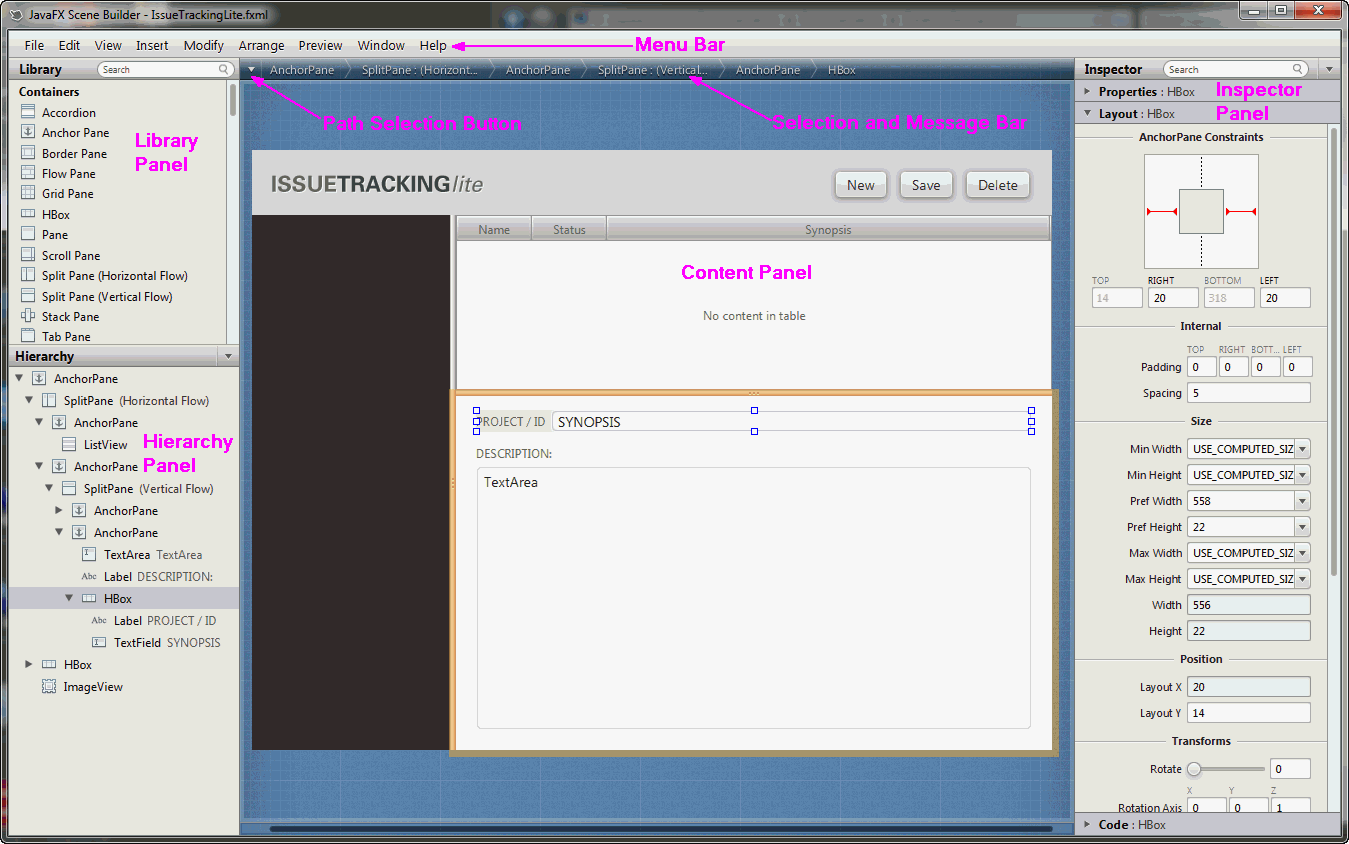


Figure 2 - Oracle SceneBuilder 1.1 Early Release

Oracle began SceneBuilder in late 2013 with the 2.1 release of JavaFX as a means of lightening the load of having to compile and execute code to view UI progress and check for any faults. It came at an early stage of Javas rise to dominate the programming world and thus was unopposed.

Though we view this as a gift to the community from Oracle and a major improvement that certainly made many projects easier, their decision to push their own Markup Language FXML, released in 2011, as the only possible output type is what prompted us to plan and go through with EasyFx.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Last Update | Up to Java version | Accepted Inputs | Output | Blueprints |
| Oracle Scene  Builder 1.1 | October 2013 | Java 7 | None | FXML | No |
| Oracle Scene Builder 2.0 | April 2014[[2]](#footnote-2) | Java 8 | FXML | FXML | No |
| Gluon SceneBuilder | June 2018 | Java 10 | FXML | FXML | No |
| EasyFx | January 2019 | Java 10 | EzML / FXML | JavaFx / EzML / FXML | Yes |

Table 1 Comparison of Existing Similar Programs & Frameworks

As is apparent in Table 1, our only proper competition is the Gluon continuation of Oracle’s SceneBuilder which took over post-2014 releases after Oracle halted development and favored a source-code only release. Though Gluon have been doing a great job maintaining it, they continue to rely solely on FXML and have not provided a way for experienced developers to extract a proper JavaFx variation of their design.



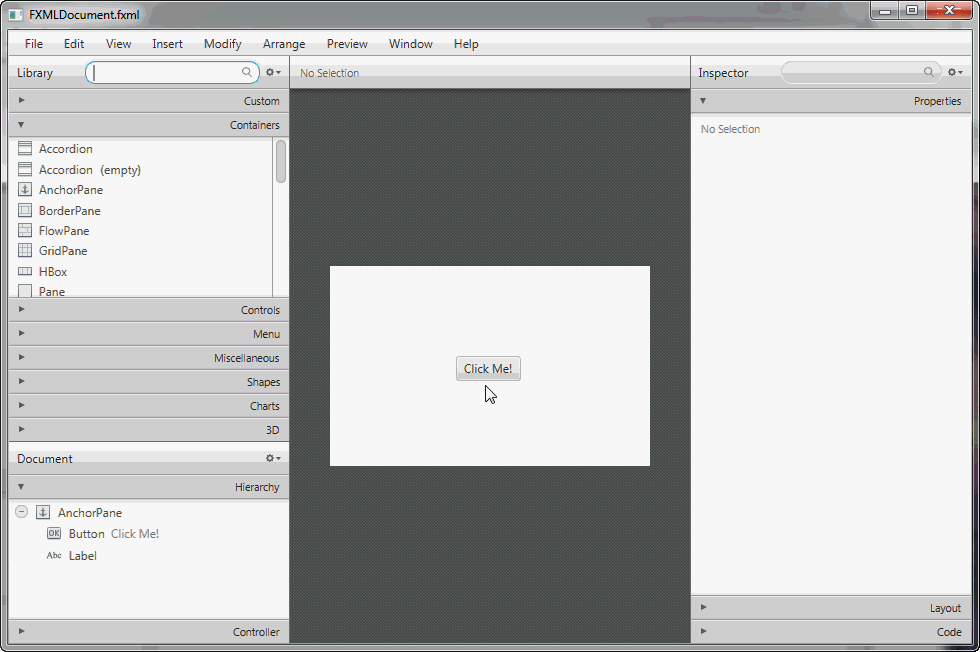


Figure 3 - Gluon SceneBuilder 8

FXML simply takes away the extended libraries and functionalities that make JavaFx unique among its outdated rivals by replacing intricate Object Oriented syntax with a Markup Language. Overall what makes our program diverse is allowing the user a choice of output and integrating an essential part of development, event handling, in the form of blueprints.

* 1. Plan of the document

This document goes into detail on all stages of designing and maintaining this product from the basic requirements engineering process through development, prototyping, testing, and preparation for an official release. Future plans and mappings of projected evolution will also be included as well as extended versions or requirement gathering procedures. All these will be discussed in detail below.

1. Requirement Analysis and Specification
   1. Introduction

The following is a mildly modified condensation of the requirements collected through the initial meeting **[see addendum M1]** and follow-up meeting, phone calls, and questionnaire **[see addendums M2 through M4 and MQ]**. Unnecessary data has been omitted for the sake of simplicity. Note that the client has been simulated to better simulate the requirements gathering process. Our true client remains the Java developer community as a whole.

“EasyFx is a program that makes programming with JavaFx easier and code-free. Users should be able to drag and drop buttons, fields, and shapes onto the canvas. The user can edit the details of these items and the items can then be dragged around to change their position and can also be deleted. Selecting one of them sends the user to a blueprint-like page where he can manage what happens when interacting with each item (hovering, clicking, etc...).

Users can start on an empty canvas, use a pre-built template, or import their own projects that can be FXML or previous projects made on the program. Closing the program will first prompt the user to save their changes or cancel. Saving is done into a custom file type made for the program, as FXML, or a text file holding extracted JavaFx.

All this can be done using a guest account. Users can also register a local account with an email address. This will allow them to save their progress directly onto the local account instead of a file for ease of use. They are prompted upon creating an account to sign up for the newsletter to receive information about future releases. Registered users can access a settings tab to manage their account details, logout, clear data, or sign up/out of the newsletter.

The system should ensure that user data, encrypted locally, can only be accessed by the user himself and no one else. Passwords should never be viewable at the point of entry or any other time. Users shall receive notification of any profile change or unauthorized login via email.

The program itself shouldn’t be very big and should fit in a portable, executable JAR file.”

* 1. Requirements Analysis

The following is the extraction of functional and non-functional requirements from the passage:

* + 1. Functional Requirements

Mandatory…

* + 1. Non-functional Requirements

Mandatory…

* 1. Specification
     1. Use Cases

Mandatory… Give the whole UC diagram then explain and justify all the association in it (extends, includes and inheritance).

* + 1. Use Cases Textual Description and Sequence Diagrams

Good to have it for the couple of main functionalities… optional for the rest.

The following is a sample use case detailed textual description:

* + - 1. “Register User” Use Case:

|  |  |
| --- | --- |
| **Actors** | User |
| **Description** | This use case allows the user to the register in the app so she/he can use its features |
| **Pre-Conditions** | The device is connected to the internet |
| **Post-Conditions** | The user is able to proceed to the app home activity and use its features |
| **Normal Flow** | 1. The user open the application for the first time 2. The user enter her/his information in the registration form 3. The user submit the form |
| **Exceptions** | Registration form input returned validation error  The user phone number already exists in the system  The user email address already exists in the system |

Table 2 DTD of the “Register User” Use Case

You may also give DFD but not necessary at all if irrelevant to put the accent on the data in your app.

* 1. Conclusion

This chapter presented…

Difficulties encountered in gathering the data, if any, and the followed approach to overcome them (ask the client for more details, check and get inspired by existing similar apps, leaving the questions open till being answered in an agile reactive way later on during the development phase…)

1. [JavaFx GitHub Release](github.com/javafxports) [↑](#footnote-ref-1)
2. [Oracle SB 2.0 Release notes](docs.oracle.com/javase/8/scene-builder-2/release-notes) [↑](#footnote-ref-2)