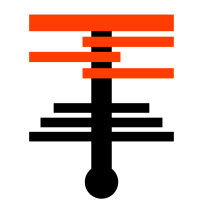


Project Submitted In the Context of the Course   
Info3301 **Software Engineering**



**Developed By  
Jad Mrad  
George Raed**

Lebanese University

Faculty of Sciences I

Department of computer Sciences

2018 - 2019

# Acknowledgements

We would like to express our deep gratitude to Professor Kamal Baydoun, the course instructor, for providing a solid and thorough basis on which to build our project and subsequent report on in addition to guiding us through the software engineering process step by step. Furthermore we would like to acknowledge with much appreciation the support provided by the informatics department lab assistant with special thanks going out to Doreid Dagher for providing access to the labs in which we conducted much needed testing.

# Abstract

The purpose of this project is to provide to the Java community a platform to build JavaFx projects with a degree of flexibility never before seen. EasyFx is a program that allows users to create, in a drag & drop fashion, complex UI with no coding needed. We seek to also solve a problem that so far remained unaddressed which is the lack of a proper visual scripter that provides a choice of output format. To achieve our set goals we set out to get the target functionalities with version 0.x (EZFXLite) then later build from the ground up on our main release in versions 1.x and beyond.

# Table of Contents

[Acknowledgements 2](#_Toc535528291)

[Abstract 3](#_Toc535528292)

[Table of Contents 4](#_Toc535528293)

[Table of Figures 6](#_Toc535528294)

[Table of Tables 7](#_Toc535528295)

[Chapter I - Introduction 8](#_Toc535528296)

[I.1. The business domain 8](#_Toc535528297)

[I.2. About the modeled application 9](#_Toc535528298)

[2.1. Users Persona 9](#_Toc535528299)

[2.2. Analysis of the Existing Similar Programs 9](#_Toc535528300)

[I.3. Plan of the document 11](#_Toc535528301)

[Chapter II - Requirement Analysis and Specification 12](#_Toc535528302)

[II.1. Introduction 12](#_Toc535528303)

[II.2. Requirements Analysis 13](#_Toc535528304)

[2.1. Functional Requirements 13](#_Toc535528305)

[2.2. Non-functional Requirements 14](#_Toc535528306)

[II.3. Specification 14](#_Toc535528307)

[3.1. Use Case 14](#_Toc535528308)

[3.1. Use Cases Textual Description and Sequence Diagrams 15](#_Toc535528309)

[II.4. Addendums 18](#_Toc535528310)

[4.1. Meetings: 18](#_Toc535528311)

[The fictional company InsertCompanyNameHere.inc will henceforth be referenced as ICNH 18](#_Toc535528312)

[The team working on the project EasyFx will henceforth be referenced as “the developers”, “the development team”, and “devs”. 18](#_Toc535528313)

[4.2. Questionnaires: 19](#_Toc535528314)

[II.5. Conclusion 19](#_Toc535528315)

[Chapter III - Application Conception 19](#_Toc535528316)

[III.1. Introduction 19](#_Toc535528317)

[III.2. UML Class Diagram 19](#_Toc535528318)

[III.3. Sequence Diagrams 20](#_Toc535528319)

[III.4. Technical Design 20](#_Toc535528320)

[III.5. State Chart Diagram 20](#_Toc535528321)

[III.6. Maintenance and Delivery 20](#_Toc535528322)

[III.7. Conclusion 20](#_Toc535528323)

[Chapter IV - Application Test 21](#_Toc535528324)

[IV.1. Validation Test 21](#_Toc535528325)

[IV.2. Integration Test 21](#_Toc535528326)

[IV.3. Unit Test 22](#_Toc535528327)

[IV.4. Conclusion 22](#_Toc535528328)

[Chapter V - Conclusion 23](#_Toc535528329)

[V.1. Future Considerations 23](#_Toc535528330)

[Bibliography 24](#_Toc535528331)

[Note about the bibliography [to be deleted from the final document] 24](#_Toc535528332)

# Table of Figures

[Figure 1 Programming Language Market Shares - Early 2018 8](#_Toc536216786)

[Figure 2 - Oracle SceneBuilder 1.1 Early Release 9](#_Toc536216787)

[Figure 3 - Gluon SceneBuilder 8 11](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EasyFX%20Project%20Report.docx#_Toc536216788)

[Figure 4 - Use Case Diagram 14](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EasyFX%20Project%20Report.docx#_Toc536216789)

[Figure 5 - DFD of "Extract" Use Case 16](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EasyFX%20Project%20Report.docx#_Toc536216790)

[Figure 6 - Class Diagram Front End & Login Sequence 21](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EasyFX%20Project%20Report.docx#_Toc536216791)

# Table of Tables

[Table 1 Comparison of Existing Similar Programs & Frameworks 10](#_Toc534153088)

[Table 2 - Priority Description 15](#_Toc534153089)

[Table 3 - DTD of the “Import Project” Use Case 15](#_Toc534153090)

[Table 4 - DTD of the “Extract” Use Case 16](#_Toc534153091)

[Table 5 - DTD of the “Register” Use Case 17](#_Toc534153092)

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 share 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.

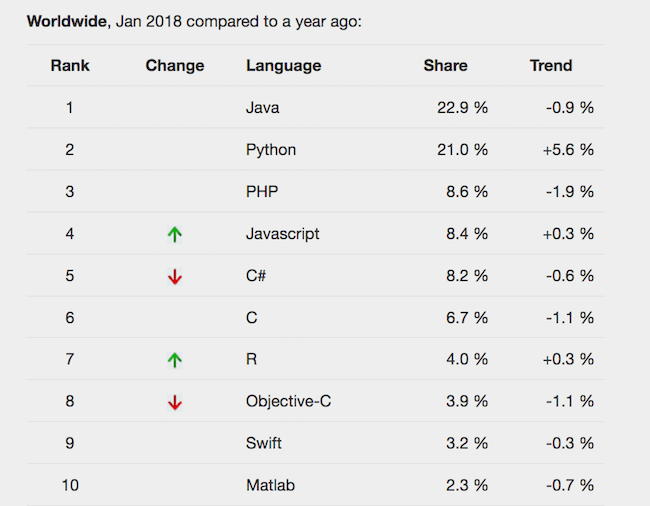


Figure 1 Programming Language Market Shares - Early 2018

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.

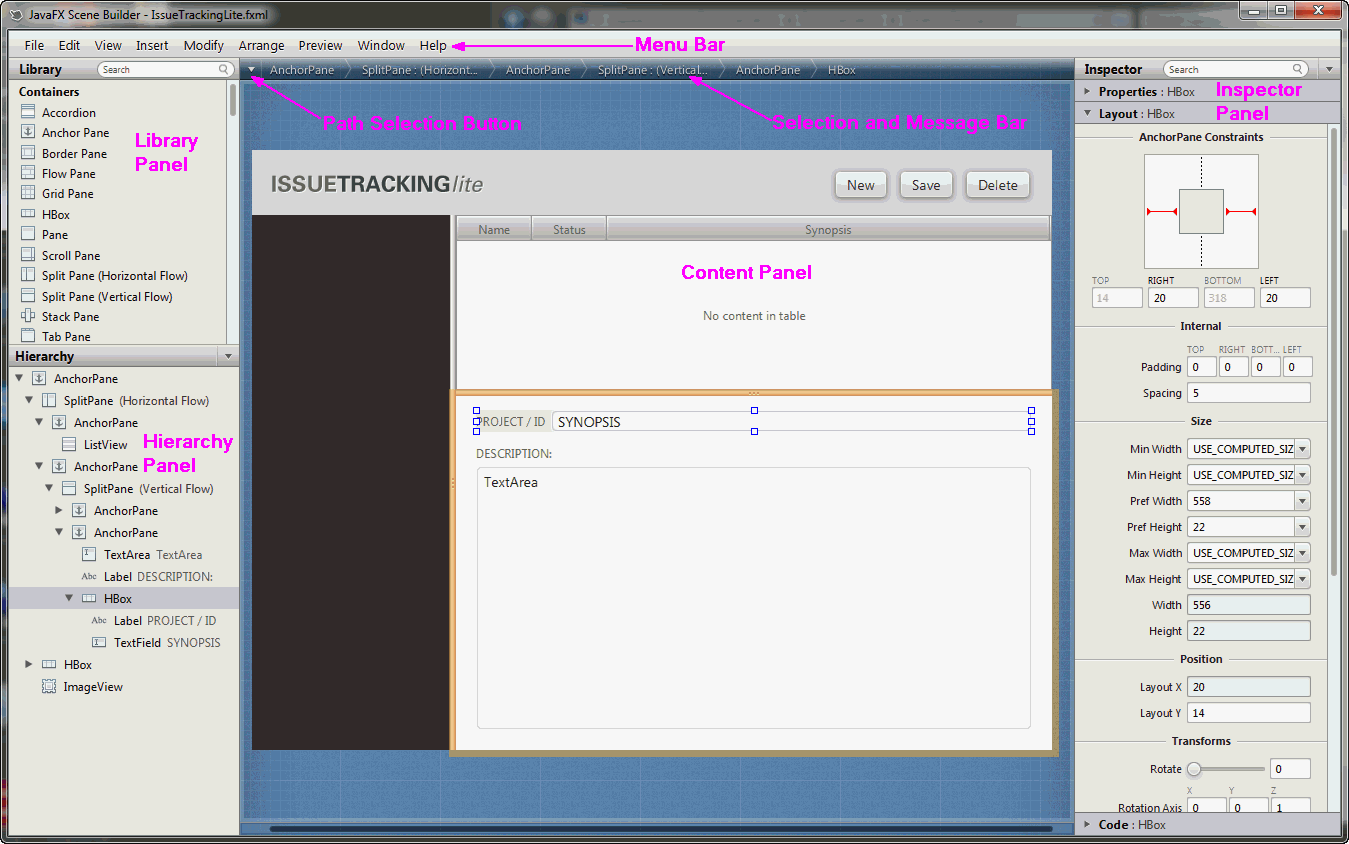


Figure 2 - Oracle SceneBuilder 1.1 Early Release

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.

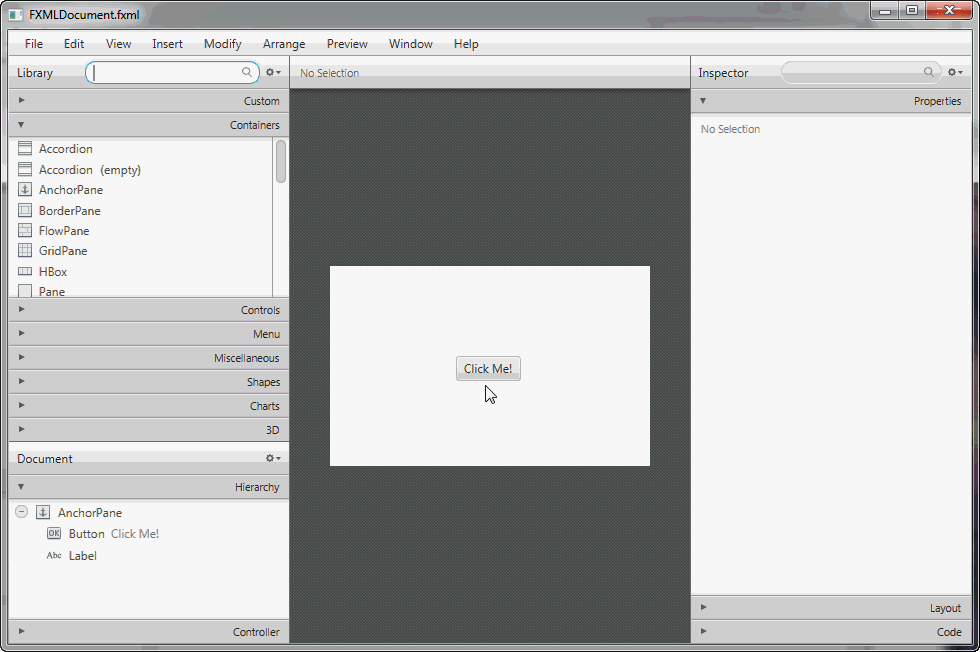
Though we view this as a gift to the community from Oracle and a major improvement that certainly made many projects easier, their decision to push their own Markup Language FXML, released in 2011, as the only possible output type is what prompted us to plan and go through with EasyFx.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 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 the developers over at 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 fictionated 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. They can also switch between a light and dark theme.

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.

Use a custom file extension to save created projects. 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.

“Custom file type made for the program” has been replaced with “EZML”, our program’s own markup language that constructs the UI.

* + 1. Functional Requirements

Guests can:

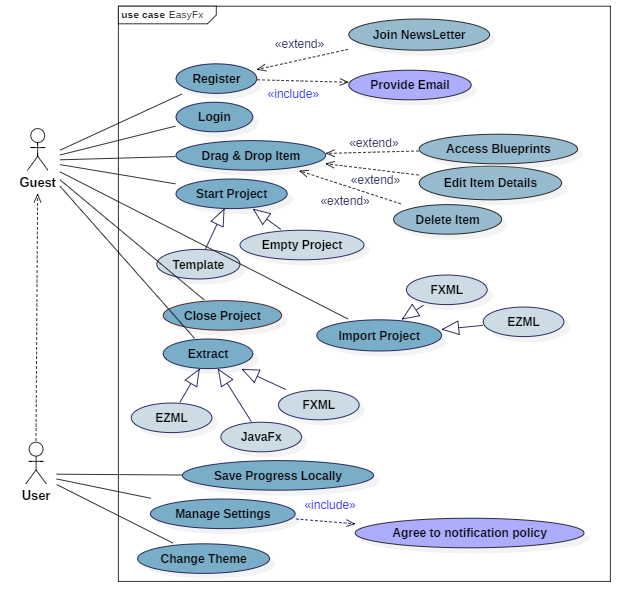
* Register a local account with an email address. **Feasible**
* Drag and drop buttons, fields, and shapes onto the canvas. **Feasible**
* Edit the details of these items. **Feasible**
* Items can then be dragged around to change their position. **Not Feasible [see addendum M3]**
* Delete items. **Feasible**
* Manage events with blueprints functionality. **Feasible**
* Start an empty project. **Feasible**
* Start a template project. **Feasible**
* Import EZML or FXML project. **Feasible**
* Close the program. [Customized closing sequence] **Feasible**
* Save/Extract as EZML, FXML, or JavaFx. **Feasible**

Registered users can:

* Save their progress directly onto the local account. **Feasible**
* When creating an account sign up for the newsletter to receive information about future releases. **Feasible**
* Access a settings tab to manage their account details, logout, clear data, or sign up/out of the newsletter. **Feasible**
* Receive notifications of any profile change or unauthorized login via email. **Not Feasible [see addendum M4]**
* Change the products theme. (light/dark) **Feasible**
  + 1. Non-functional Requirements

The system should:

* Allow users to drag & drop items. **Usability**
* Ensure that user data, encrypted locally, can only be accessed by the user himself and no one else. **Confidentiality**
* Passwords should never be viewable at the point of entry or any other time. **Access Security**
* The program itself shouldn’t be very big and should fit in a JAR file. **Portability**



* Run without using high maintenance servers **Development Requirement**
  1. Specification
     1. Use Case

Figure 4 - Use Case Diagram

|  |  |  |
| --- | --- | --- |
| Priority |  | Details |
| 1 | Cosmetic | - Serves no purpose |
| 2 | Add On | - Extension that runs alongside main functionalities |
| 3 | Neutral | - Secondary functions |
| 4 | Semi Essential | - Required for proper functionality. Failure won’t cause error |
| 5 | Essential | - Failure will most probably cause error state |

* + 1. Use Cases Textual Description and Sequence Diagrams

Table 2 - Priority Description

* + - 1. “Import Project” Use Case:

|  |  |  |
| --- | --- | --- |
| **Number** | #UC\_DTD\_001 | |
| **Name** | Import Project | |
| **Summary** | Extract EZML project from input | |
| **Priority** | 4 | |
| **Post-conditions** | Project is constructed and opened | |
| **Primary Actor** | User | |
| **Trigger** | Import option from File menu | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User browses for target file |
|  | 2 | System validates file extension |
|  | 3 | System constructs UI from project |
| **Extensions** | **Step** | **Branching Action** |
|  | 2a | File extension is invalid |
|  | 2b | System notifies user: “Invalid File Type” |
|  | 2c | Import cancelled |
|  | 3a | System runs into error extracting data |
|  | 3b | User is notified of the error |
|  | 3c | Import cancelled |

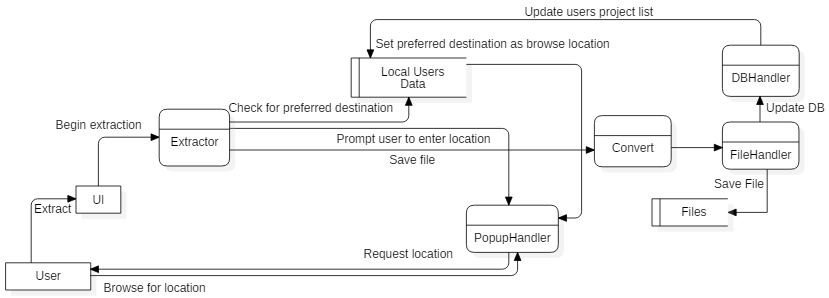
Table 3 - DTD of the “Import Project” Use Case

* + - 1. “Extract” Use Case

|  |  |  |
| --- | --- | --- |
| **Number** | #UC\_DTD\_002 | |
| **Name** | Extract | |
| **Summary** | Export created project into a specific format | |
| **Priority** | 4 | |
| **Preconditions** | **N/A** | |
| **Post-conditions** | User now has a file in a specific format of his creation | |
| **Primary Actor** | User | |
| **Trigger** | Export option from File menu | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User selects save location (Browse) |
|  | 2 | User selects format (EZML, FXML, or raw JavaFx) |
|  | 3 | User enters file name |
|  | 4 | System checks if a file of that name/extension exists in target |
|  | 5 | User provides preferred save location |
| **Extensions** | **Step** | **Branching Action** |
|  | 3a | File with same name exists, prompt user to over-write file |
|  | 3b1 | User agrees. File is over-written |
|  | 3b2 | User denies. Cancel extraction |

Table 4 - DTD of the “Extract” Use Case

Figure 5 - DFD of "Extract" Use Case



* + - 1. “Register” Use Case:

|  |  |  |
| --- | --- | --- |
| **Number** | #UC\_DTD\_001 | |
| **Name** | Register | |
| **Summary** | Register local account and input details | |
| **Priority** | 2 | |
| **Post-conditions** | User has an active local account in the database | |
| **Primary Actor** | User | |
| **Trigger** | Register button from Login scene | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User provides name and email |
|  | 2 | Email is verified (?) |
|  | 3 | User provides a valid password |
|  | 4 | System checks password strength |
|  | 5 | User provides preferred save location |
|  | 6 | System prompts user to join newsletter |
|  | 7 | User accepts |
|  | 8 | User added to news list |
| **Extensions** | **Step** | **Branching Action** |
|  | 2a | System notifies user: “Email is not valid” |
|  | 2b | User prompted to retry step 1 |
|  | 3a1 | System notifies user: “Password is weak” |
|  | 3b1 | Passwords do not match |
|  | 3a2/3b2 | User sent back to step 3 |
|  | 6a | User denies |
|  | 6b | User set to be notified about newsletter on next login |
| **Open Issues** | I001\_1 | Email verification system not confirmed functional yet |

Table 5 - DTD of the “Register” Use Case

* 1. Addendums
     1. Meetings:

The fictional company InsertCompanyNameHere.inc will henceforth be referenced as ICNH

The team working on the project EasyFx will henceforth be referenced as “the developers”, “the development team”, and “devs”.

**“The program”, “The system”, “The framework”, and “The project” are all references to EasyFx.**

* + - 1. M1: Initial conception meeting

Location: Head Office of ICNH

Date: 22/10/2018

Time: 10:00 AM to ~12:00 AM

Attendance:

* EasyFX Team (2/2)
* ICNH CEO, CSO, & IT Manager (3/3)

Agenda Items:

* Discuss EasyFX general functionalities & expectations [Success]
* Plan EasyFX UI design [Partial success, finalization postponed see addendum M2]
* Agree on timeline [Success]

Action Items:

* The developers will provide an in-depth documentation report on the software including all meetings conducted
* The developers are tasked with providing a handy user-manual alongside the product
* The agreed upon functionalities (see II.1) are to be achieved by the devs before the agreed upon dates
* ICNH will provide full access to their own development team for testing purposes
* ICNH will provide limited access to their database facilities for integration purposes
  + - 1. M2: Follow-up in depth meeting

Location: Starbucks Hamra

Date: 28/10/2018

Time: 02:00 PM to ~03:00 PM

Attendance:

* EasyFX Team (2/2)
* Selected ICNH Developers (2/3)
* ICNH Graphic Designer (1/1)

Agenda Items:

* Plan EasyFX UI design [Success, see TODO: Sketch]

Action Items:

* The program will feature a night/dark mode
* The primary colors of the program will be an office grey on red brick tint
* The design will be similar to that of [Name Omitted For Trademark Reasons]
  + - 1. M3: Phone call regarding feasibility

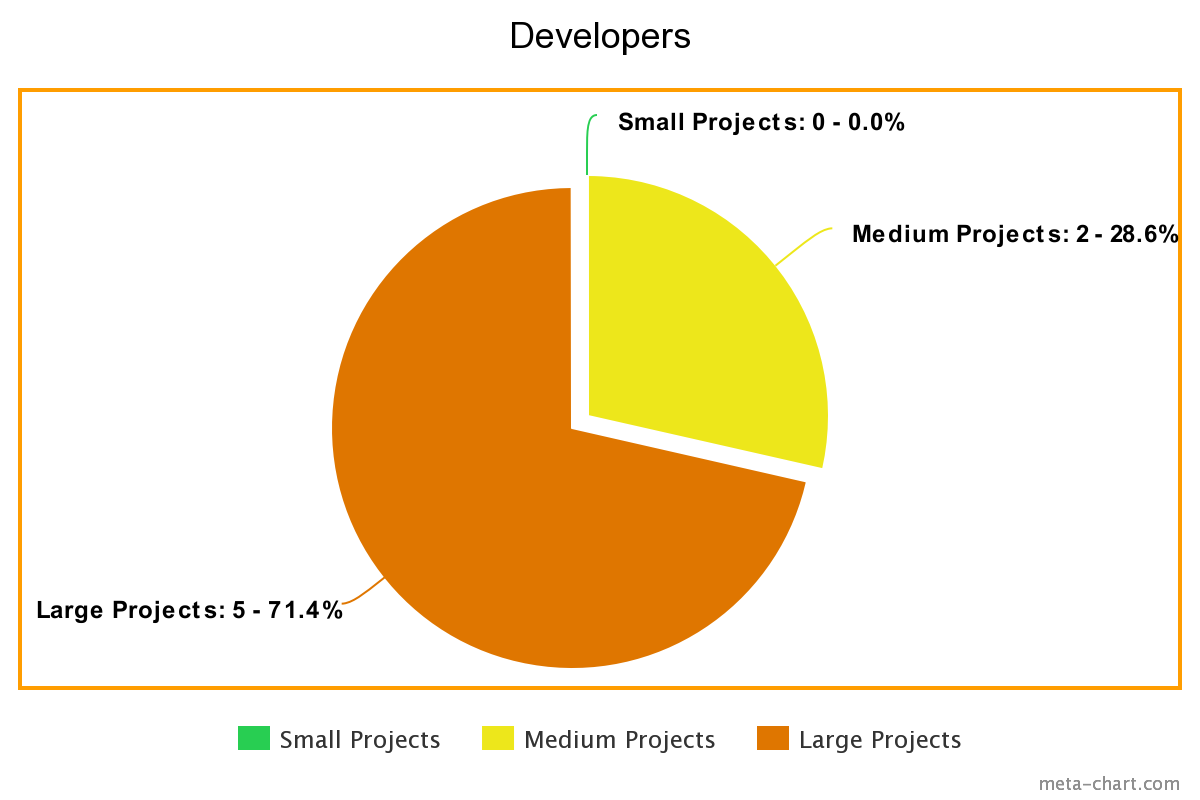
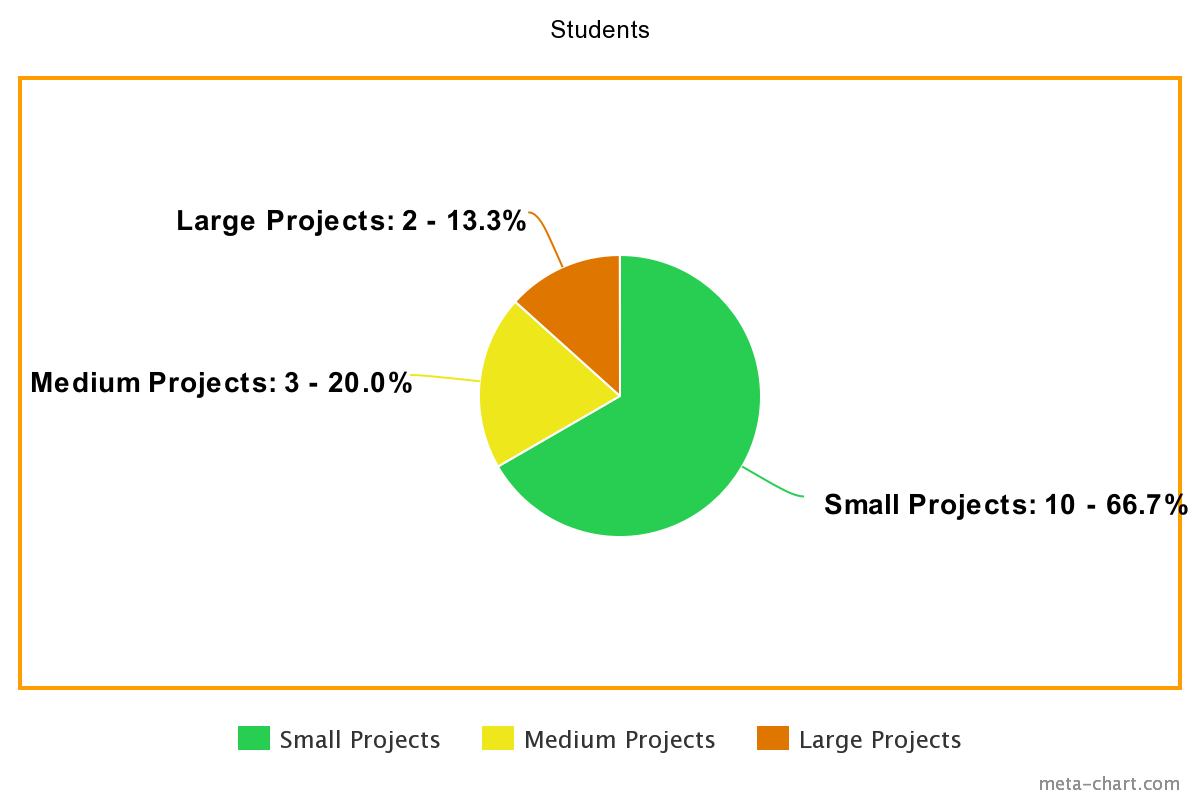
Action Items:

* Drag-ability of nodes outside set containers has been omitted for technical reasons
* More containers were requested to fill drag-ability gap
  + - 1. M4: Final pre-release meeting

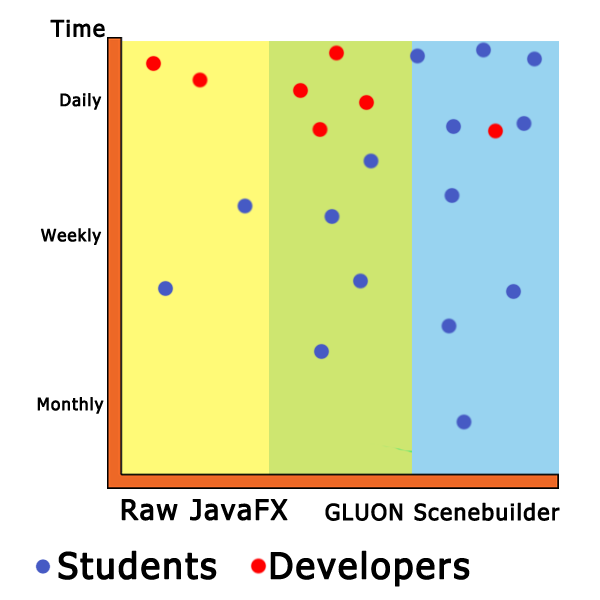
Action Items:

* E-mail configuration removed from deliverable(s) to be added in by ICNH teams later
* Demos tested and approved
* DB integration test run and approved
  + 1. Questionnaires:
       1. MQ: JavaFx developers
* **How often do you develop with JavaFx**
  + Once or twice a month
  + Weekly
  + 2-3 Times a week
  + Daily
* **Do you use any of the following (check all that apply)**
  + Raw JaxaFX
  + Oracle SceneBuilder
  + Gluon SceneBuilder
  + JFoenix Library
* **What scale of projects do you work on?**
  + 100-2000 Lines of code
  + 2000-10000 Lines of code
  + 10000+ Lines
* **Which of the following would you consider yourself**
  + Beginner
  + Student
  + Freelancer
  + Developer (Employed)

MQ was distributed on 22 FX developers of varying backgrounds for more diverse results and to aid comparison. The following is a compilation of these results:



As expected, developers tend to work on larger projects whereas students take up smaller ones, the latter being in their learning phase. With this data in mind we move on to our second input evaluation:



We thus deduce that larger projects require the developers to be working directly on the raw JavaFX code for optimization and flexibility purposes. Based on this assessment we chose to further push EasyFX’s usability as a conversion tool.

* 1. The Design Process

TODO: Hand drawn sketches

* 1. Conclusion

TODO: Conclude

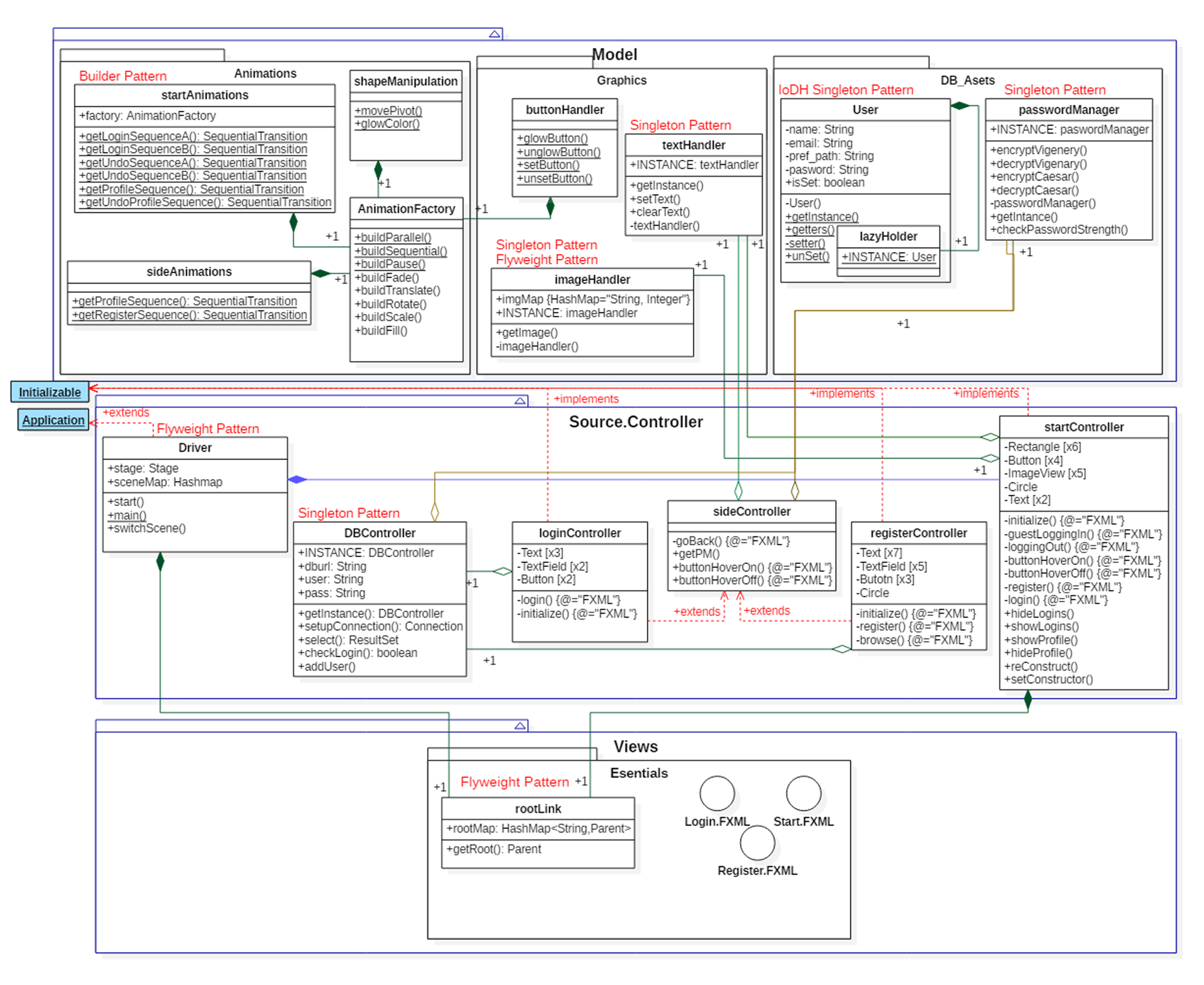
1. Application Conception
   1. Introduction

With the project planned out and a proper foundation set up, next come the steps to bring EasyFx to life. Below is the detailed description of our coding process and how we transformed a shower thought into a usable application.

* 1. UML Class Diagram

Figure 6 - Class Diagram Front End & Login Sequence

* + 1. Front End & Login



TODO: Back end & Builder Class Diagram

* 1. Sequence Diagrams
     1. Initial access sequences

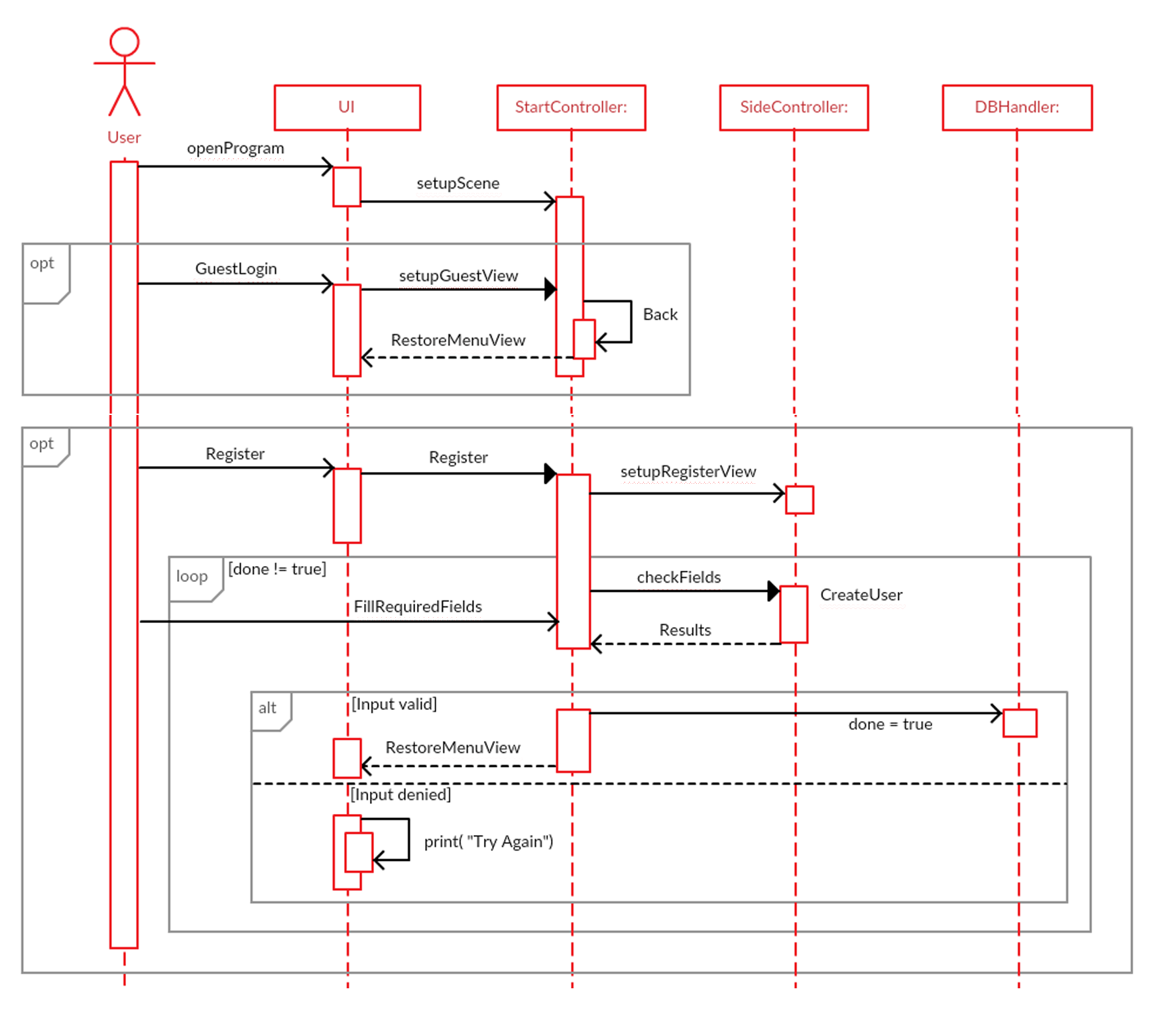


Figure Initial Access Sequences

Figure 7 shows 2/3 of the three access sequences to reach the main builder screen. The standard ‘Login’ was omitted to limit redundancy due to its similarities to ‘Register’.

* + 1. Something else

TODO: secondary & compact sequence diagram to use elements not used above

* 1. Technical Design

TODO: About pieces of code, database choice, web techniques, languages choice…

* 1. State Chart Diagram

.TODO: Maybe

* 1. Maintenance and Delivery

This part is also optional and it is to be done only if you find that you have enough concrete elements in order to realize it.

If you ever want to get started, then give an estimated delivery date; A plan of successive deliveries in case you have several batches.

Also give your plan to support your client by helping him adopt your project, put it into service and then your strategy to help him and keep his service in perfect condition in the future.

* 1. Conclusion

This chapter presented our application design. First, we introduced…

After that, we presented...

In the next chapter, we make a complete feasibility study and draw out the plan of our project realization, risk management and client support and satisfaction strategy...

1. Application Test

In this chapter, propose an appropriate process for the validation of your application, while clarifying the tests to be carried out in order to verify the satisfaction of the needs and the cases of use initially identified.

The titles given here are for information only. You are free to adjust as you seem clearer, relevant or better. You can also reverse the order (from the unit to the validation instead of the reverse) or use subtitles if necessary.

* 1. Validation Test

Give a general idea of ​​the list of validation tests that must be performed to ensure the software will meet the initial needs and will conform to its original specification.

It is also important to specify the list of means (manual tests, automated tests ... which scenarios and datasets) and tools (pre-written and handwritten validation scenarios or pre-recorded scenarios in a validation tool Automated, e.g. selenium ...) that will do the job for you.

* 1. Integration Test

Having chosen software architecture and a set of tools for the development of your application at the specification stage, it is time to imagine which conflicts, defects or errors can occur when integrating the different bricks of your software.

Thus, propose a set of measures and procedures for testing and verification that will help to detect, correct or avoid such failures or unpleasant surprises.

* 1. Unit Test

In case you can identify clever, complicated or hard to implement algorithms, then propose a way to check the completeness and adequacy of the code that will be written. This can be done by carrying the cap "software architect", and specifying, for each functionality or complex algorithm to be tested, the data sets and the extreme values ​​to be considered as input of the unit tests and the corresponding outputs as the algorithm must satisfy.

Be creative to imagine cases that seem unpredictable to your novice developers, even if they have read your specification. Be selective to consider only major features and algorithms that deserve to be tested, while avoiding redundancy and insignificance. Be precise and meticulous in choosing your most relevant test sets and error and fault sensors.

* 1. Conclusion

Give your final word on this part, including the points that still seem vague to you at this stage and that you intend to explore once the development will be launched and then as it advances.

1. Conclusion

About the whole project…

Discussion of the lived experience and the obtained results.

* 1. Future Considerations

Pointers toward future work, especially if you are really willing to implement the proposed application.

## Bibliography

|  |  |
| --- | --- |
| [1] | Summerville, Software Engineering, Pearson, 2014. |

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)