**Module « Windows Development »**

**-**

**Project « Epibubble »**

**Software Architecture Specifications**

**(SAS)**

**\*\*\*LOGO\*\*\***

Table of contents

[Introduction 3](#_Toc483407758)

[1. Project context 3](#_Toc483407759)

[2. Global architecture 3](#_Toc483407760)

[3. Component description 3](#_Toc483407761)

[a. Component 1 3](#_Toc483407762)

[b. Component 2 3](#_Toc483407763)

[c. Component 3 3](#_Toc483407764)

[4. Traceability matrix 3](#_Toc483407765)

# Introduction

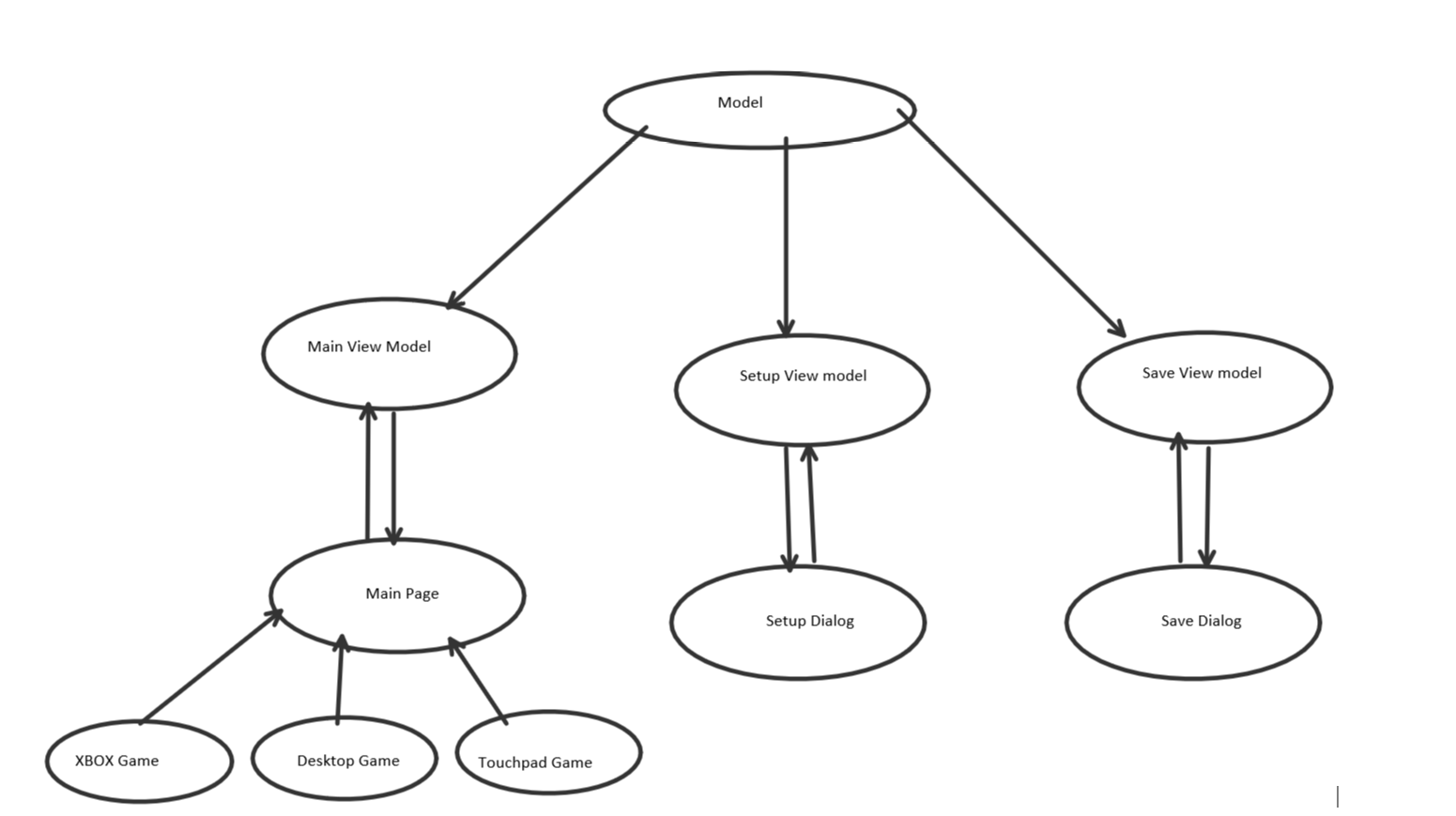
This document is a technical representation of the overall structure of our project. This project aims at introducing or ameliorating our skills in the field of Windows development. The project implemented in this module is known as Epibubble.

# Project context

Epibubble is a bubble game built in C# using the DirectX library and leveraging the power of the MVVM design pattern. Explicit project specifications where specified. Taking into consideration all of the project specifications be it functional or non-functional requirements, here is a detailed documentation of the global architecture of our implementation of Epibubble.

# Global architecture

*The global architecture for our implementation of Epibubble is MVVM. Here is a detailed description of how the components of its MVVM architecture are designed and integrated with each other.*

**

# Component description

## Model

*Programmatic representation of real-world data. This data is input by the user or represented to him through the application. A model is being used by the View model to represent data. It is ignored by the View and in our case, the model includes User information when saving a game, or the game components and their various states like the Arrow, and the Ball.*

## View

*The View is responsible for data representation and user interaction with the game. In our case, the view is the Main page, the Save and Setup dialog and each of the game components.*

*The Main page contains every user option with which the user will interact with the program. Like the setup, save buttons. And depending on which device the game is run; the appropriate game is loaded.*

*Each dialog represents a sub interaction with the game. And is backed by its own view model.*

*Each Gaming component contains methods to handle specific game interactions as needed by the user. And depending on the device which it is one, for example, only desktop game component will handle mouse and keyboard interactions. The other devices like XBOX and Touch screens will handle theirs.*

## ViewModel

The view model is responsible for the overall interactions between the application components. The view model’s properties are bound to the various views and handle interactions requested by the user.

The Main ViewModel is responsible for the main view’s interactions. The setup and save view models are responsible for each of their specific view interactions.

# Traceability matrix

This matrix make the correspondence between components, classes, functions, and requirements developed in the request for proposal:

|  |  |  |  |
| --- | --- | --- | --- |
| **Id requirement** | **Requirement description** | **Component** | **Function / action** |
| *REQ\_XXX\_XXX* | *Description* | *Component X* | *Prototype function and action* |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |