Movie Dubbing Studio Manager

Project Documentation

**Revision History:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Change Description | Issue Date | Author |
| 0.8 | Add/Edit/Delete and listing features added for basic tables.  Workflow of application created.  User login and logout support. | 02/01/2020 | Sevdzhan Alkan |
| 0.9 | Projects list and detail page completed. | 02/02/2020 | Sevdzhan Alkan |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Abstract 4](#_Toc31749346)

[2 Keywords 4](#_Toc31749347)

[3 Introduction 4](#_Toc31749348)

[3.1 Support Pages 4](#_Toc31749349)

[3.2 Main Pages 4](#_Toc31749350)

[4 User Interface 5](#_Toc31749351)

[4.1 Application UI Layout 6](#_Toc31749352)

[4.2 Screens 7](#_Toc31749353)

[4.3 Support Screens 7](#_Toc31749354)

[4.3.1 List Page 7](#_Toc31749355)

[4.3.2 Detail Page 8](#_Toc31749356)

[4.4 Master Screens 8](#_Toc31749357)

[4.4.1 List Page 8](#_Toc31749358)

[4.4.2 Detail Page 9](#_Toc31749359)

[5 What I Skip 11](#_Toc31749360)

[5.1 Form Validation & Data Binding 11](#_Toc31749361)

[5.2 Multi Language Support 11](#_Toc31749362)

[5.3 Material Design For Everywhere 11](#_Toc31749363)

[6 References 11](#_Toc31749364)

# Abstract

This document is describes user interface, using of application and application code of the project. You will see firstly UI and using introductions, then project structure and the application code.

# Keywords

C#, .Net Framework, NInject, Dependency Injection, WPF, Entity Framework, ORM, XAML, Material Design

# Introduction

The project developed for Advanced Software Technologies as a course work project.

The used technologies are listed below.

* C#, this programming language mainly supporting for WPF.
* .Net Framework, its stores useful class collection and also the important part is the WPF app needs to .Net to work and build the project.
* EntityFramework, I used that as ORM framework. Basically, An Entity Framework does; managing Database via your table definitions and generates T-SQL queries via.
* NInject, the library for Dependency Injection.

The application helps to store data about ongoing dubbing projects to manage them. The application is able to store data about movies, persons, tags, publishers, users, genres.

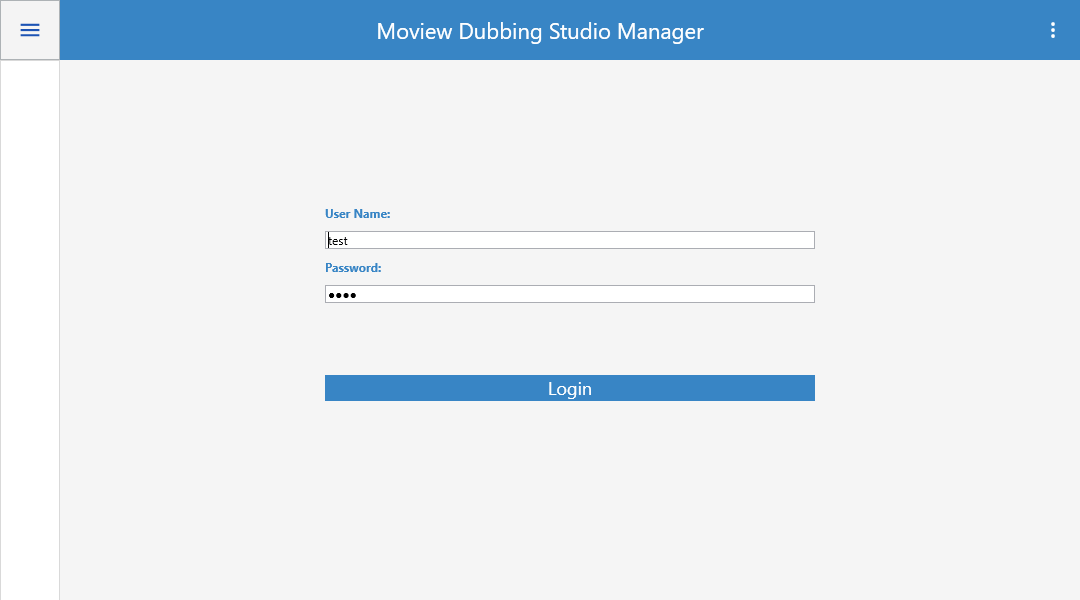
## Screens

Some screens are designed to operate CRUD operation for simple tables, that’s are named as Support Screens. The other screen category is Master Screens, these screens using for CRUD operations like Support Screens. But the differences are; table of the master screen is has one or more than one relation with other tables and that makes master screens to more complex.

**Master Screens are:** Project, Movie

**Support Screens are:** User, Genre, Language, Person, Publisher, Tag

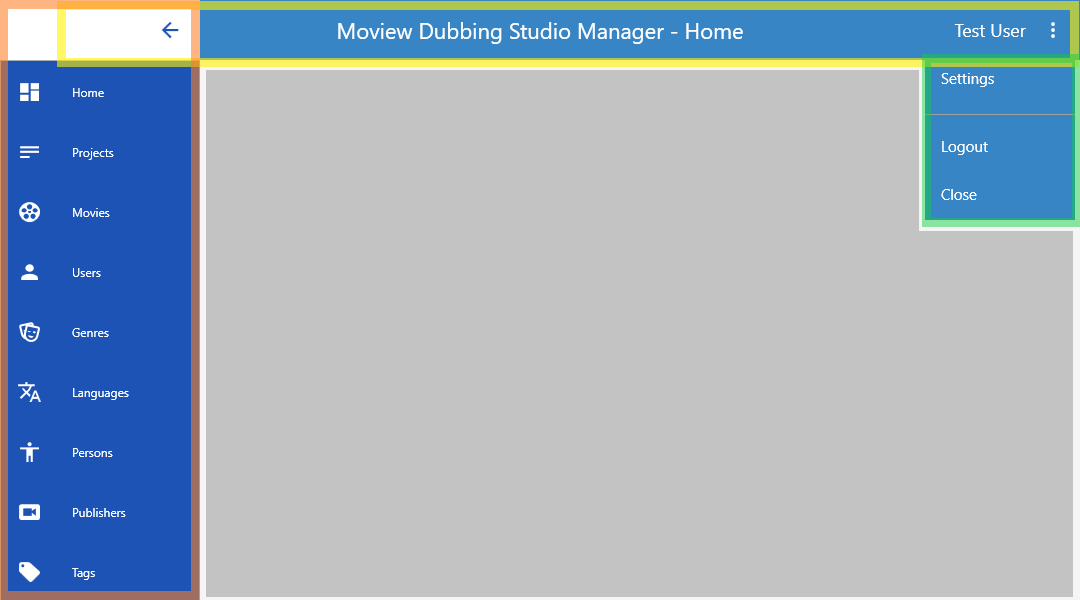
# User Interface

Let’s look at the first page, which is opening when the app launched.

The main windows open when the application launched. If the user not already logged to the application, the main windows load the login page.

The application has only one window, it's working like a single page application. The application loads different pages into the main window up to user inputs.

## Application UI Layout



This is the application User Interface layout. It contains four different area. That’s are;

* Navigation menu. (selected with orange)
* Window border/toolbar. (selected with yellow)
* User menu. (selected with green)
* Page container. (filled with grey)

When user changes the page or the application loads a page up to an operation, the page loads into page container area. The application never created new window, all operations are doing into main window.

**Note:** First two screen, Home and Movie are not finished yet. I was panned locate some charts into Home screen, to be a dashboard. And also, Movie screen is little complex then the support pages because of that reason I couldn’t complete these screen in time.

Each button, which are in navigation menu, its opens related list page. Example: “Users’ button is opens ***UserList*** page. And each list screen have a detail page for add, edit operations.

List pages are maintains listing saved records, deleting selected record and navigation to the detail page.

Detail pages are maintains adding new record to he database or editing operation for existing records in the database.

## Support Screens

### List Page

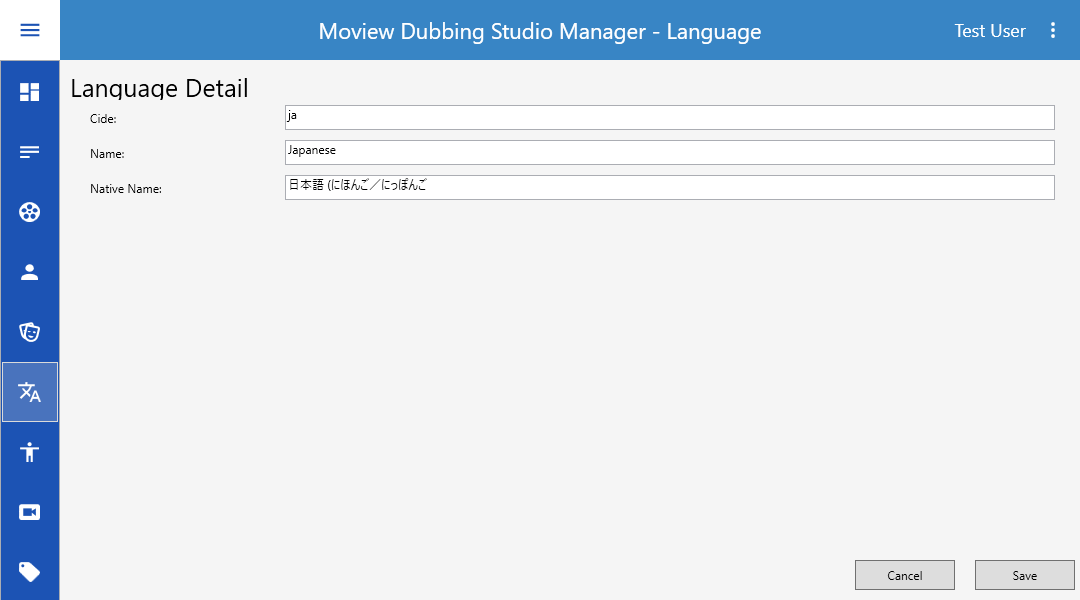
Each screen has a list page, like showed below. List page are contains a button group on top right section and a DataGridView component to list saved records.



Functionalities of the buttons;

* Add: loads detail page with empty input fields.
* Edit: if user already select a row on grid component, the detail page loads with selected record data. Else its works like Add button.
* Delete: if user selected a record on Grid, this button deletes the record from database.

### Detail Page



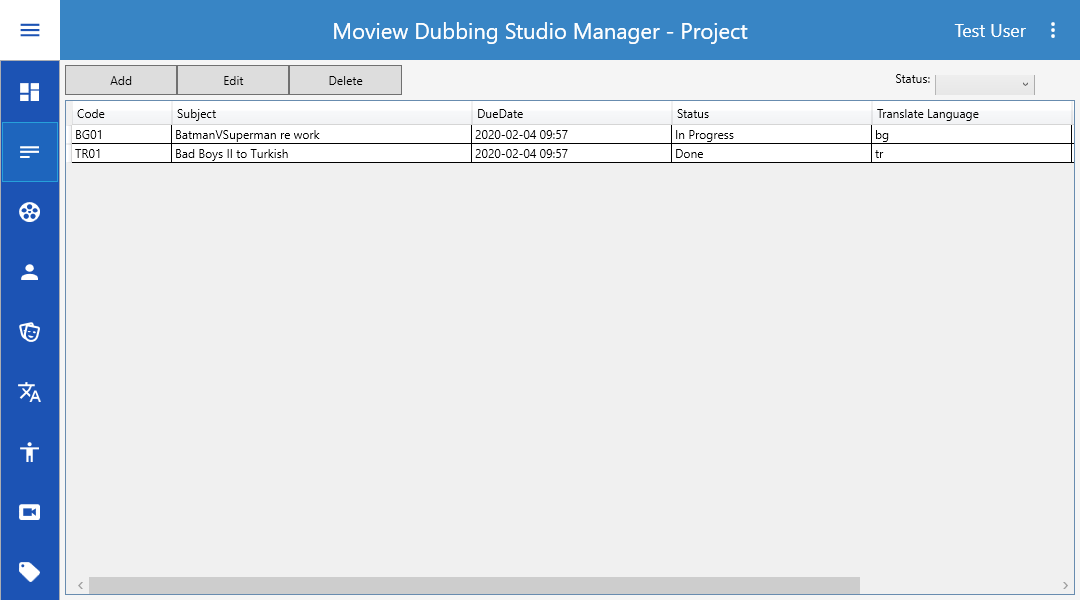
The detail page are contains fields of related database table and buttons for save or cancel adding/editing data.

## Master Screens

On this part is explaining the Project Screens because of Movie Screen aren’t completed.

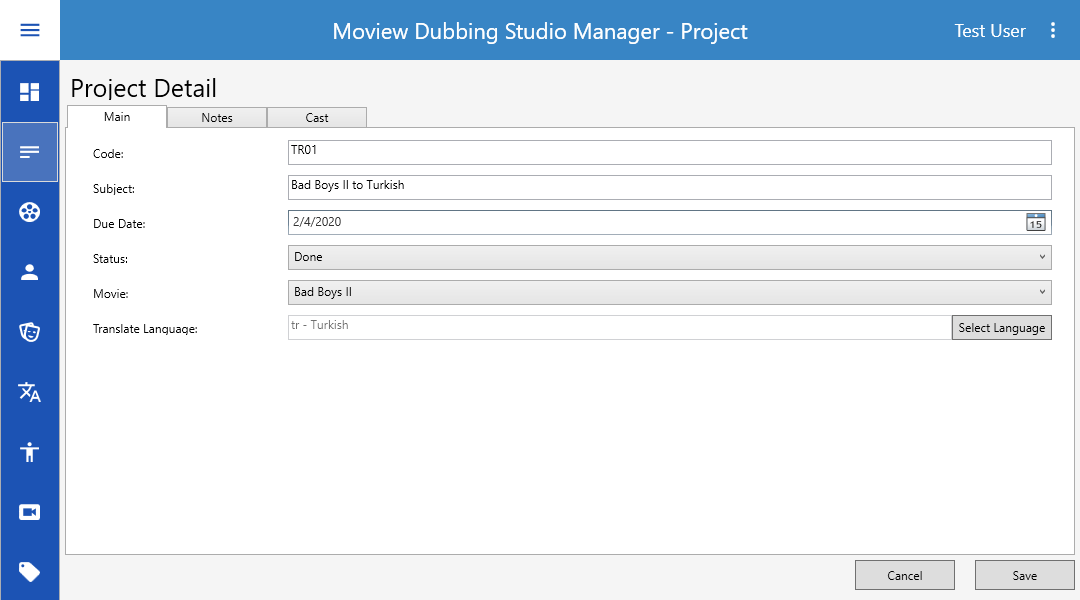
### List Page

List page of the project screen almost same with other screens. Only it has a additional Status filter.

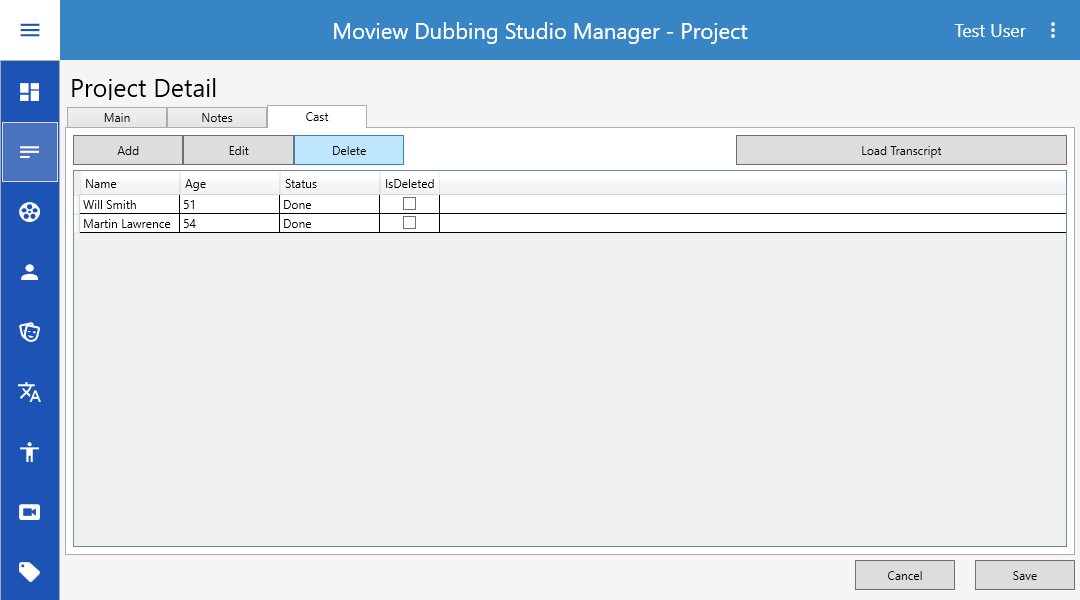


### Detail Page

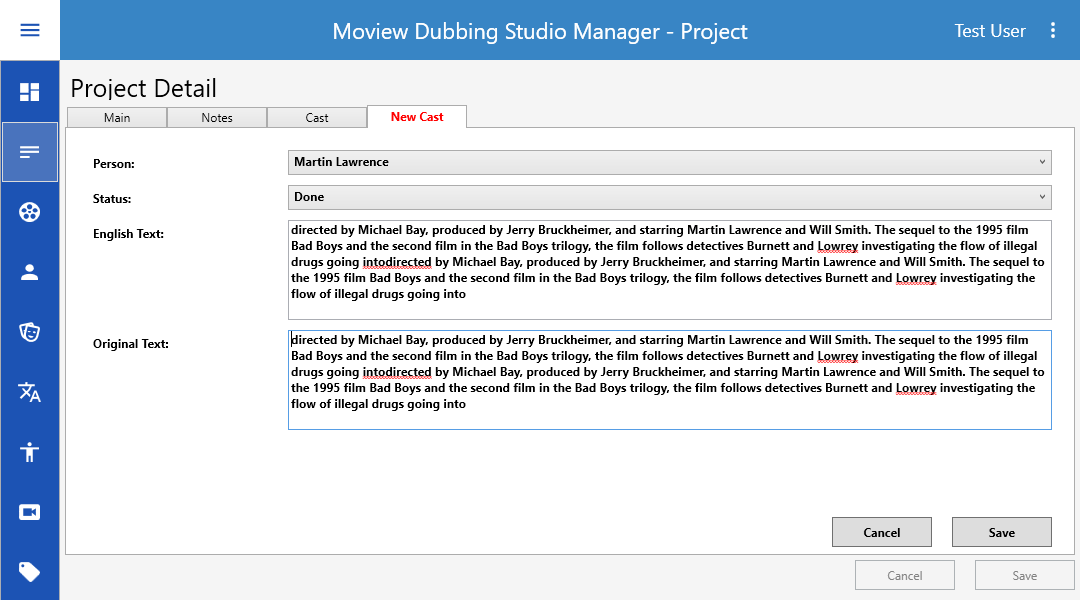
The detail page has a tab control, each tab contains different components. ‘Main’ tab contains fields of the Project table. ‘Notes’ tab contains only Note filed of Project table. ‘Cast’ tab added for ProjectCast table operations. And the last tab is named as ‘Cast Detail’, its reponsable to adding and editing dubbing artists of the project. Also ‘Cast Detail’ tab not visible default, when its needs to use, the application makes it visible.



#### Cast List



#### Cast Detail



# Technical Review

This part of the document are describing how is the code working.

## Application Layout

|  |  |
| --- | --- |
|  | Firstly I want to describe solution and projects. The solution has five different project. That’s are;   * **Core:** Its contains some useful classes to use entire solution. * **Data:** the database logic are located into this project. * **Desktop:** The main project of the solution, WPF application is this project. * **Domain:** database tables(poco classes) are located into this project. * **UI:** That project added to store custom UserControls, like custom grid component. It can useable, but it is not used. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# What I Skip

I skipped some features to add the application. Because these features makes the application more complex or to add them I need more time for implementation.

## Form Validation & Data Binding

WPF and NVVM design pattern supporting databinding and form validation. But its needs more XAML and C# codes to implement.

## Multi Language Support

On real world solutions I think its have to support by application. Each static text of the UI can get from a specific language file. Also this language file can load on login by the user preference. But for this project it is not necessary.

## Material Design For Everywhere

Currently the material design using only main window layout. Its not using on components of UserControls. Again its needs more customization in XAML.

## Better Entity Framework using

Using lazy loading, improvement for linq queries, additional mapping classes for poco classes.

# References

* [C# WPF Tutorial - Multiple Views](https://www.youtube.com/watch?v=xUwk2-_tRzo)
* [C# WPF UI Tutorials: 01 - The Basics](https://www.youtube.com/watch?v=Vjldip84CXQ)
* [Dependency Injection For WPF & MVVM](•%09https:/stackoverflow.com/questions/25366291/how-to-handle-dependency-injection-in-a-wpf-mvvm-application)
* [C# WPF Material Design UI: Navigation Drawer & PopUp Menu](https://www.youtube.com/watch?v=YQ1EJJZBHyE&list=WL&index=11&t=0s)
* [Pluralsight WPF Course](https://app.pluralsight.com/library/courses/wpf-mvvm-in-depth/table-of-contents)