# Contents

[I. Contents 1](#_Toc71193481)

[II. Database connection checking 2](#_Toc71193482)

[III. Adding a new Tax Payer record 2](#_Toc71193483)

[1. Personal Information 2](#_Toc71193484)

[2. Tax Information 3](#_Toc71193485)

[IV. Searching/Browsing the records 4](#_Toc71193486)

[1. Searching by TFN (Tax File Number) 5](#_Toc71193487)

[2. Searching by last name 6](#_Toc71193488)

[3. Browsing all records 7](#_Toc71193489)

[V. Updating an existing record 8](#_Toc71193490)

[VI. Version of control discussion 8](#_Toc71193491)

[VII. Project structure 10](#_Toc71193492)

[1. MVP (Model – View – Presenter) 10](#_Toc71193493)

[2. Gradle 10](#_Toc71193494)

[3. Spring boot for desktop application 11](#_Toc71193495)

[4. Database 11](#_Toc71193496)

# Database connection checking

When application is being launched, there is a checking for database connection. If the database is not available or can not be connected by any reasons. The GUI (Graphic User Interface) cannot show. The following messages can be found in the logs

|  |
| --- |
| *Could not connect MySQL database* |

Graphical user interface, text, application

Description automatically generated

# Adding a new Tax Payer record

When launching the application successfully, the main view will be as following

Graphical user interface, application

Description automatically generated

The fields within the \* character in the form are the required ones that means it need to be filled before saving the record.

## Personal Information

**Tax File Number (TFN)**, **First Name, Last Name, Address** and **Phone Number:** are all required for a Tax Payer record

## Tax Information

**Tax Held** and **Tax To Be Returned** are calculated from the inputs of **Income per Year** and **Deductible Amount,** by clicking on **“Calculate Tax”** button.

The tax is calculated by the ATO appendix being shown as following

Table

Description automatically generated

A sample case before saving will be similar as following

Graphical user interface, application

Description automatically generated

After clicking “Save” button, a popup will show information of successful saving

Graphical user interface, application

Description automatically generated

Clicking on **“Reset”** button will clear the fields in form and make it looks like the first time of launching.

# Searching/Browsing the records

The searching area is located at top right on the main view of the application. It allows to search by TFN, last name and the feature of browsing all records is also supported here by input \* character for searching.

Graphical user interface, application

Description automatically generated

If leaving the text field empty, then click on “Find” button, a message will show as following

Graphical user interface, application

Description automatically generated

## Searching by TFN (Tax File Number)

**Input a TFN** and click on “**Find**” button to search the record, if there is an existing record for the given TFN, the details will be loaded into the fields in main form.

Graphical user interface

Description automatically generated with low confidence

After clicking **“Find”**

**Graphical user interface, application

Description automatically generated**

If the searching could not find any records, it will show as following

Graphical user interface, application, Word

Description automatically generated

## Searching by last name

**Input a last name** and click on “**Find**” button to search the record(s).

**If there is only one record found**, the details information is shown in the main view similarly as following

**Graphical user interface, application

Description automatically generated**

**If there are multiple records found,** after clicking on Find button,a new view will be shown as following

Graphical user interface, application, Word

Description automatically generated

On the table, select one row and then click on **“Load”** button to show the details information on the main view.

Graphical user interface, application

Description automatically generated

## Browsing all records

Input \* character and click on “**Find**”button, a view will show all record in the tables as following

Table

Description automatically generated

# Updating an existing record

To update info of an existing Tax Payer, we need to search the record first, it can be done by searching TFN or last name as mentioned above. After searching and loading information into main view successfully. We can modify the editable information and calculate the tax held and tax to be returned again. Then click on Save button to update into database.

Graphical user interface, application

Description automatically generated

# Version of control discussion

The application can be developed with more features and the bugs fixing activities, a large team with multiple developers can involve making things growing fast. To avoid making the source code suffering and enable the ability of rolling back when a critical issue happen, version of control should be considered here, namely we should use GIT as the VCS (Version of Control System).

GIT will enable the developers working in parallel and independently. It is also easy to trace back the histories of changes and rollback to wherever needed.

We need to have a GIT server available, setting up a new one can be overkilled. It is better to use a wide-known system as [GitHub](https://github.com/) or [Bitbucket](https://bitbucket.org/).

Git branching models should be applied. Each of developer need to work on the feature branch, test carefully the changes and create the Pull Request (PR) to request the reviews the changes from the collaboration team before merging those changes into the main branch (develop or master – it is up to the team’s decision).

Diagram

Description automatically generated with medium confidence

# Project structure

## MVP (Model – View – Presenter)

Graphical user interface, application

Description automatically generated

**com.trs.entity** is the package of **Model (M)** layer where we place the entity classes representing for the database tables.

**com.trs.presenter** is the package of **Presenter (P)** layer where we place the presenter classes being responsible for handling the data from model classes and decide the ways of displaying them into the view layer

**com.trs.ui** is the package of **View (V)** layer where we place the classes of GUI (Graphic User Interface).

## Gradle

Project uses NetBean as the IDEA that helps boosting the GUI implementation and Gradle as the build tool and dependencies management.



## Spring boot for desktop application

This project utilizes the benefit from Spring Boot framework, although it is used mainly for web application, we can take advantage of the Repository layer (I.E. CrudRepository interface) to work with Mysql database without writing the connection string and database queries.

## Database

MySQL 5.7 is installed as a part of Wamp, as the suggestion from the requirement, the credentials are used in the application properties are root/mypassword

Graphical user interface, text, application

Description automatically generated

Using CLI to login by the following command

*mysql -u root -pmypassword*

The database need to be created manually before the application starting up



*Text

Description automatically generated*

After the application is launched, JPA implementation as a part of the **Spring framework helps to generate the corresponding tables automatically. So that, we don’t need to create the table by the SQL query.**

Text

Description automatically generated

Insert some dummy data by the following query

Text

Description automatically generated

We can check the data in tables as following

A picture containing text, scoreboard

Description automatically generated