BAY HOUSE SCHOOL AND SIXTH FORM

Chair of Governors Training Tracker

CG4.2 - Design

John Robinson

09/01/2014

Contents

Design

Justification of Tools

To create the solution to the problem, I will be using a combination of Java and SQL. I will use Java to create a program with a Graphical User Interface which will connect to an SQL database where all my data will be stored. To create the program in Java, I will be using an IDE (Integrated Development Environment) called Netbeans. To manage the database, I will use a tool called MySQL workbench. The server itself will be hosted onsite at college by the technical team and I will install MySQL and the workbench at home.

These two tools will allow me to create and test my program throughout its development stages as well as support me in creating a fully functioning piece of software. As both Netbeans and MySQL workbench have debugging tools, I will be able to use them to my advantage to find out when an error has occurred in my program and how I can go about fixing it. I picked Java itself because I already have experience with the language and learning a new language would take time and would waste development time on my project. I've never used a SQL database before and I wanted to make a program that would be able to store the data in a remote location as opposed to flat file so that if the user wanted to use the program in multiple places, they could do so without having to take the flat file database with them. For instance, if my client wanted to have the program installed on two computers in her house, it would almost be impossible to work with flat file databases as she would have to move the saved filed from one computer to another. If a remote database is used, then my client can use either computer as long as the computer hosting the database is turned on.

I have decided to not use an application like Access or Excel to create my program because Excel would be very limiting in the functionality I could have implemented in my program. If I were to create the program from scratch, I would be able to create and implement whatever features I wanted to as long as I had the knowledge to implement such features. This gives me a larger scope. I could have used Access which gives a wider range of functionality than Excel and as this is a database project, then it would have been acceptable to use Access. The reason why I didn't use Access was because it would cost my client and me money to buy Microsoft Access. As Java and MySQL are free programs and their tools are usually free, this allows me to create a free program.

I have decided to create the system in Java as it will allow me to create a solution to run on a wide range of hardware. There are forks/implementations of MySQL for Linux, Mac and Windows and Java itself runs on Linux, Mac and Windows. This will allow my client to be able to use any operating system without worry of not being able to use the system anymore. I have also decided not to use Excel or Access to create the solution to the problem as these solutions will cost the client money if they do not already have this software. I want my solution to be as costless as possible for myself and my client. Access and Excel also have higher system requirements than Java. This means that my system can run on a wider range of hardware than Access and Excel. Also, Access and Excel are only natively available for Windows and Mac. It can be run on Linux but at great difficulty with many features either missing or not working fully.

Input, Output, Capture and Format

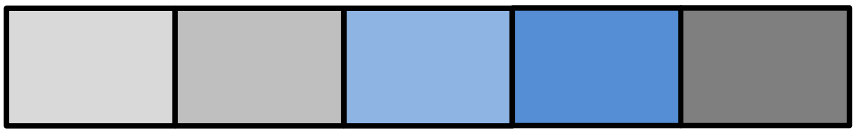
User Interface

Below, I have designed the user interface such at the same designs will be implemented across both the input and output forms.

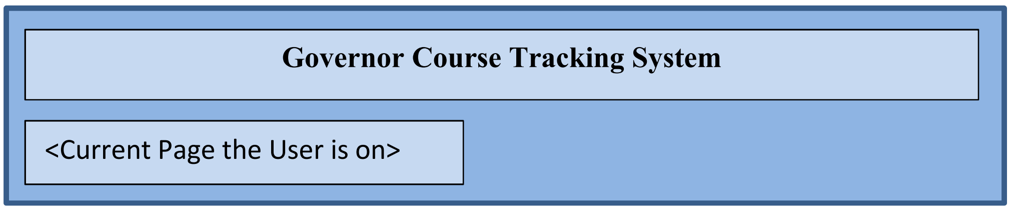
Aesthetics

I have contacted Ann, my client, and she has informed me that there is no logo that I can use for my project. I would have liked to use a logo for the project, but I don't think it's critical in the design and I don't want to spend development time making a logo.

For my program, I aim to use blue and grey colours. Some example of these colours would be:



The reason I have decided to use the colours is because of their simplicity and their contrast. Black text would be easy to ready on any of these colours, with the exception of the darker grey, and they all blend together very well. I have no creative design experience and want to stick to simple colours that have been proven to work with other major desktop applications.


Governor Course Tracking System
<Current Page the User is on>

Above is an example of the header that I will use to repeat throughout my program. It should appear on all my windows and there will be a text box which will inform the user what page they're currently on.

I may want to use a few fonts throughout my program. I am thinking that for headers I will use one font and another, different, font for the main text. This will ensure the program doesn't look bland and the important text is easier to read.

For the main text, I may use "Times New Roman". The reason I have picked this font is because I know that it's a font that is widely used in applications due to its readability. I also know that it's installed by default on almost every computer so the compatibility would be good, too.

This is an example of the font, "Times New Roman".

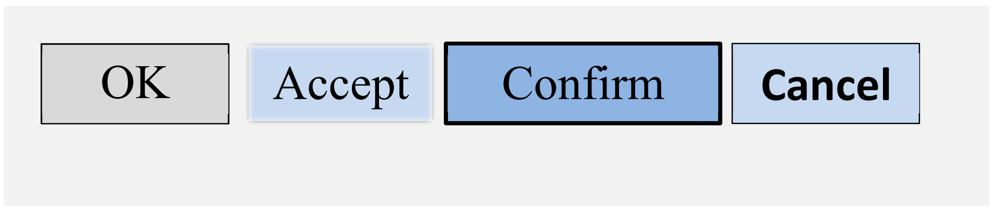
This is an example of the font, "Times New Roman".

For the headings, I may use the font "Cambria". I have picked this font for the fact that it's slightly different to Times New Roman in that the letters are a little bit more spaced out and slightly more defined. I didn't want to go for a cursive font in case it became too hard to read.

This is an example of the font "Cambria".

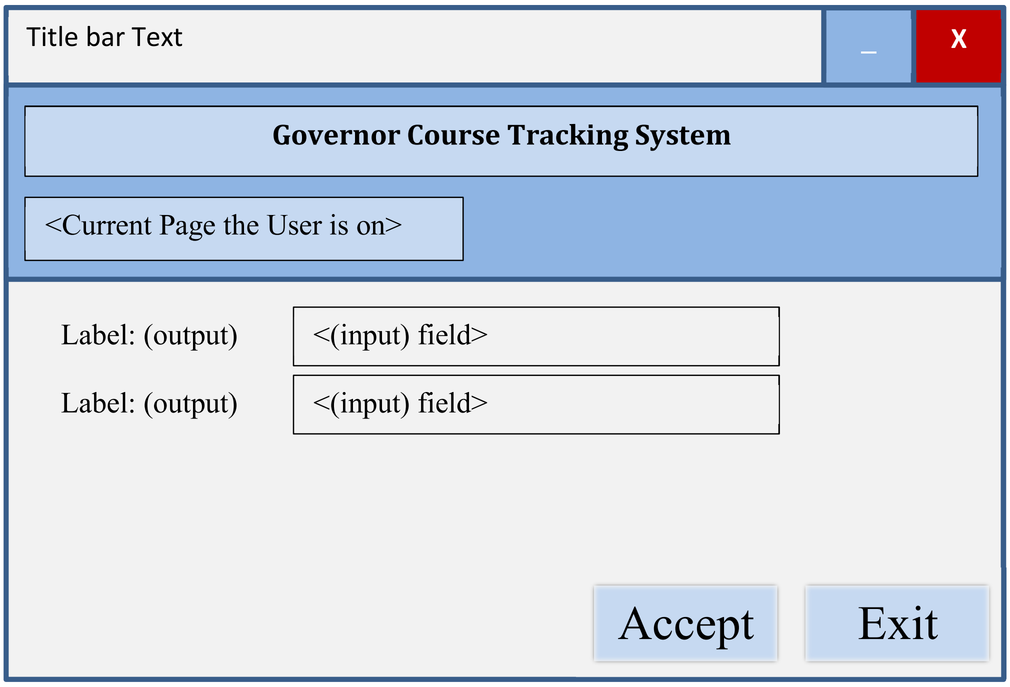
This is an example of the font "Cambria".

Below, I have experimented with a few different button designs. I have tried making the text bold, changing the weight of the outline of the box and different colours. I am in favour of the "Accept" button in that it's easy to read and doesn't stick out too much on the background. As I'm not likely to use a white background, I have decided to model my buttons on a grey background to see how well they contrast. I think I'll use the accept button style.

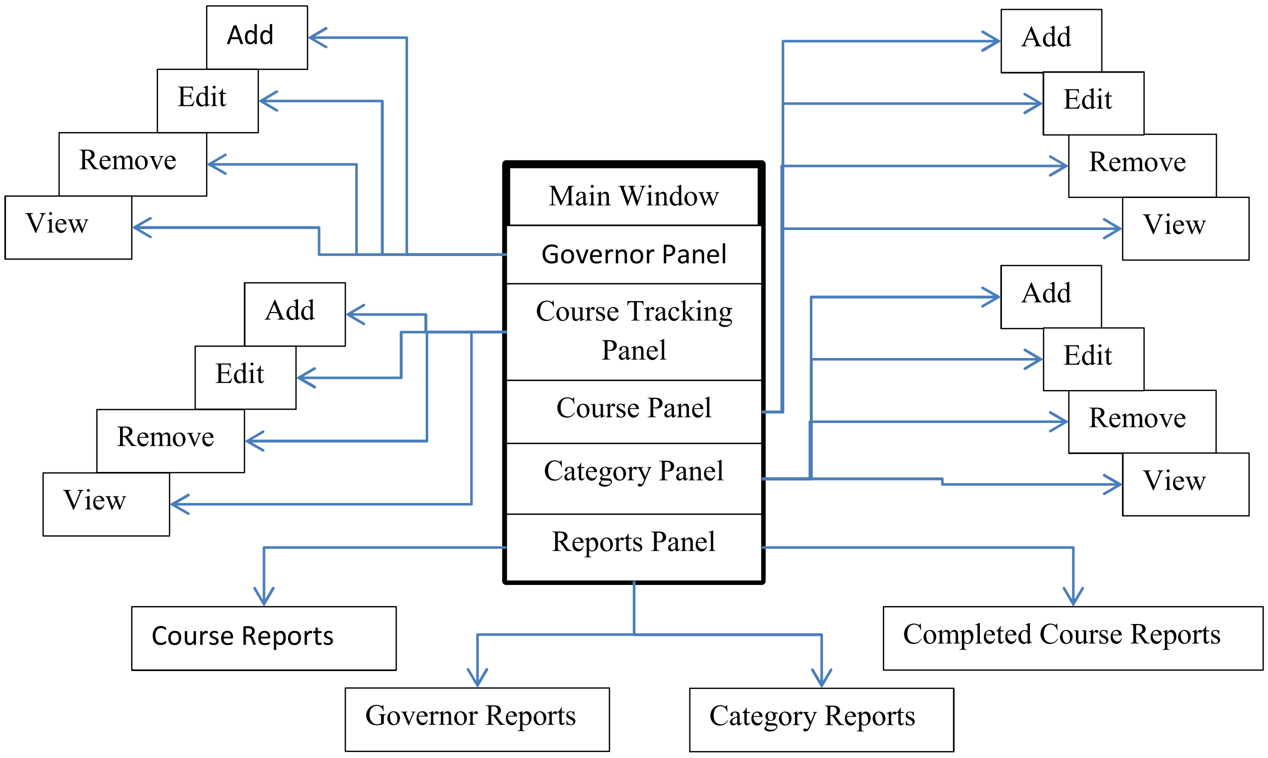


Mock-up

Below I have put together all of the above assets to create what I think the GUI will look like.



Navigation Layout



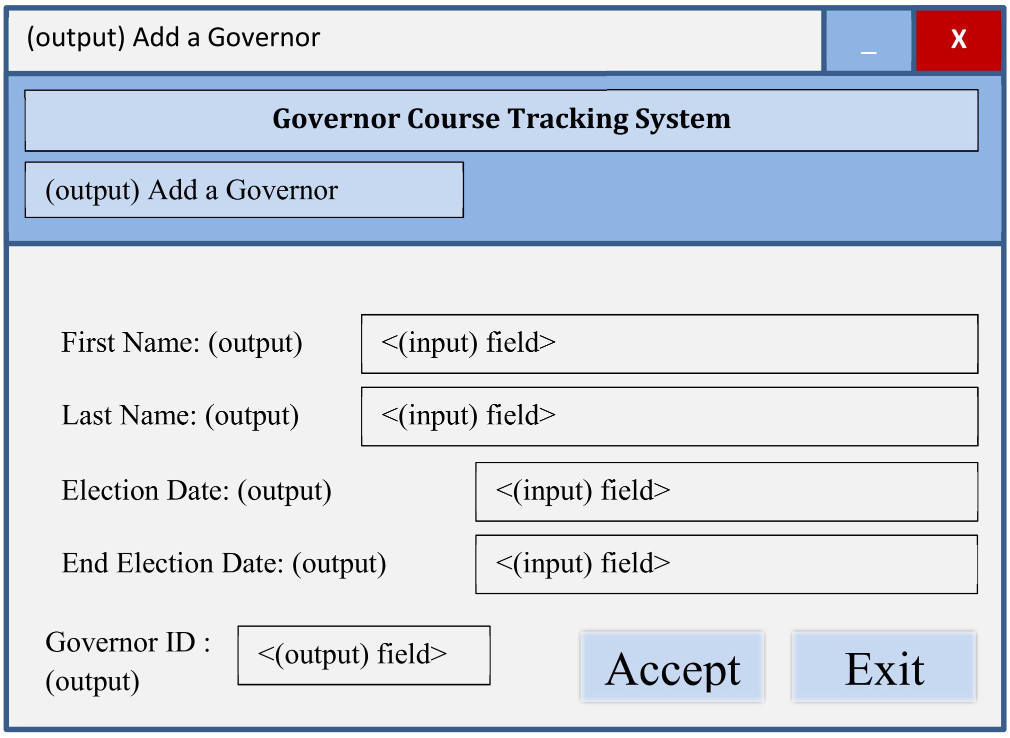
Above I have created a layout of what I think my navigation will look like. The main window will have panels which will have buttons which will connect to their forms. The reason why I have decided to layout the navigation like this is so that the user can access all of the forms in one place. I didn't want to have separate forms for each of the panels in the main window because then the user has to jump through more hoops to get to the form they want to use.

Screens, Reports and Data Capture Forms

In this section, I will look at multiple input and output forms and their designs.

Form Designs

Below is an example of what the Add a Governor form might look like. The buttons are placed in the bottom right of the window and the data entry begins at the top left. This is how the user is likely to begin reading this form and is going to want to enter the data. It's in an intuitive order and helps the user to do their work efficiently. The governor ID button which is returned after the user has added a governor is placed next to the Accept button to easily find.



(output) Add a Governor

_ X

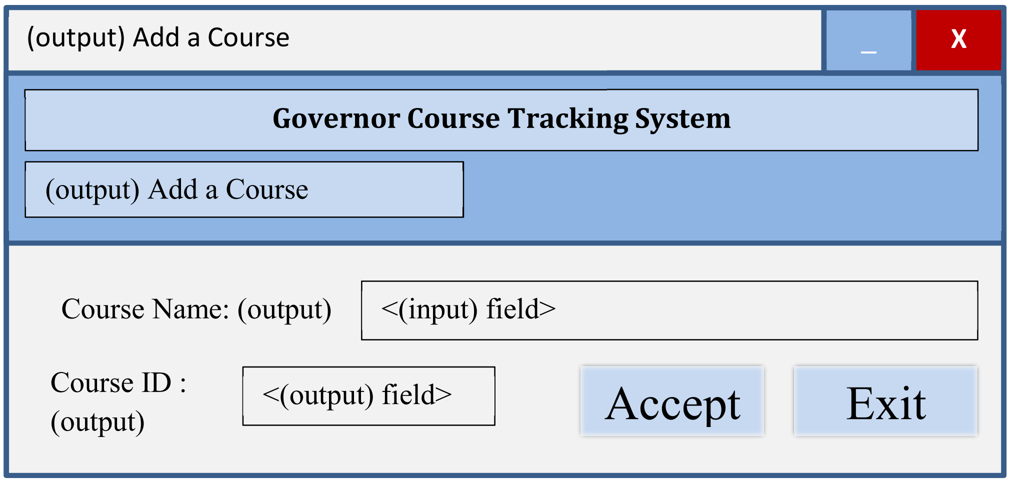
Governor Course Tracking System

(output) Add a Governor

First Name: (output)	<(input) field>
Last Name: (output)	<(input) field>
Election Date: (output)	<(input) field>
End Election Date: (output)	<(input) field>
Governor ID : (output)	<(output) field>

Accept Exit

Below I have designed the form that will be used to add a course to the system. This follows a similar layout to the add a governor form.



(output) Add a Course

_ X

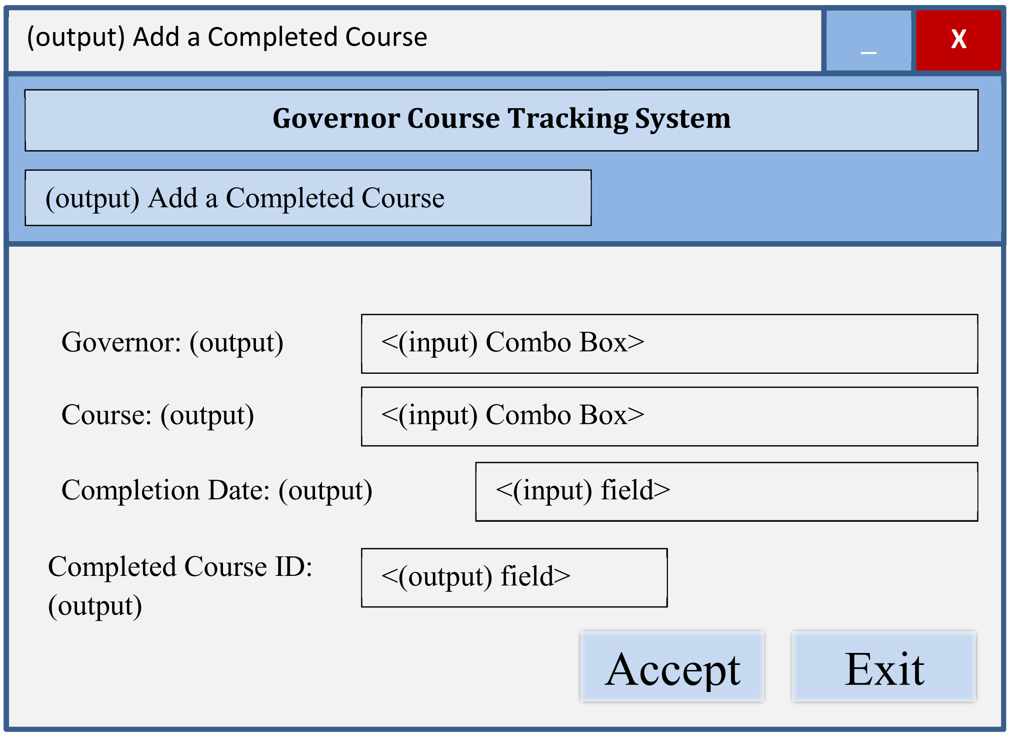
Governor Course Tracking System

(output) Add a Course

Course Name: (output)	<(input) field>
Course ID : (output)	<(output) field>

Accept Exit

Below I have designed the "add a completed course" form. This is one of the most critical forms in the whole program because it allows me to solve the problem I was faced with. I have followed the previous input form designs along with the aesthetics to create a form that holds consistency. Any other input forms in my program will follow the design and layout of the three input forms I have designed.



(output) Add a Completed Course

_ X

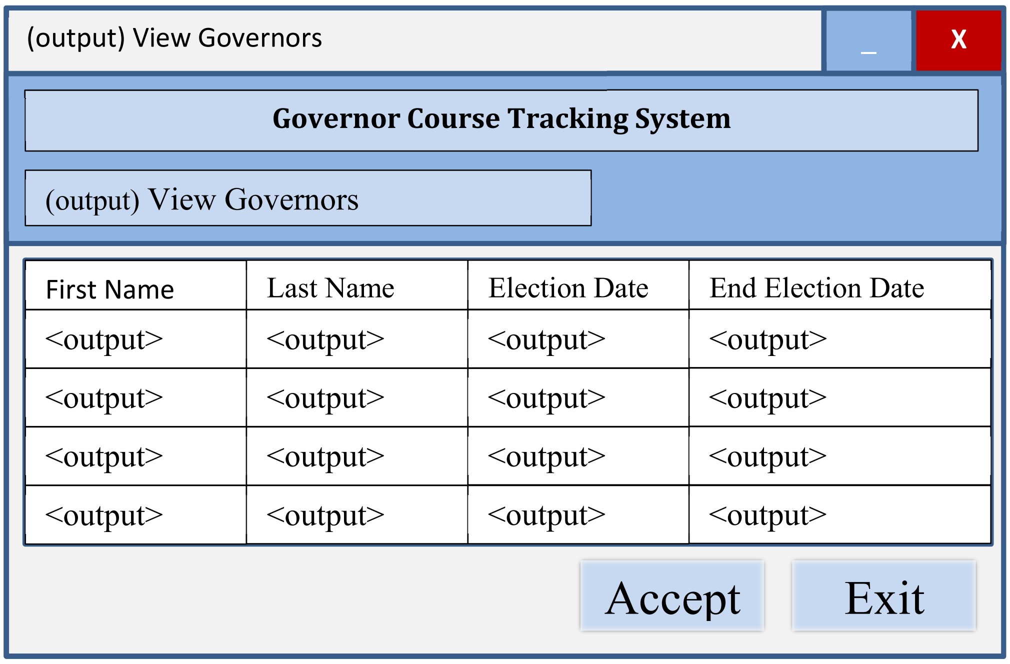
Governor Course Tracking System

(output) Add a Completed Course

Governor: (output)	<(input) Combo Box>
Course: (output)	<(input) Combo Box>
Completion Date: (output)	<(input) field>
Completed Course ID: (output)	<(output) field>

Accept Exit

Below is an example of what my main output form design will look like. All my output forms will look roughly like this with the exception that the field names will change. When I implement a new view form, I will go off this design.



(output) View Governors

_ X

Governor Course Tracking System

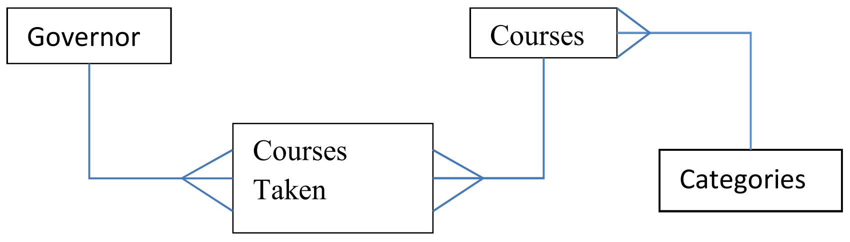
(output) View Governors

First Name	Last Name	Election Date	End Election Date
<output>	<output>	<output>	<output>
<output>	<output>	<output>	<output>
<output>	<output>	<output>	<output>
<output>	<output>	<output>	<output>

Accept Exit

Data Structures and Access

Data Structures



Above I have created a relational database diagram which shows how my tables in my database are linked together.

Data Dictionaries

Below I have taken each one of my tables and split it into a data dictionary to make creating my database a little bit easier.

Data Dictionary of the "governor" Table

Field Name	**Description**	**Data Type**	**Example**
GovID	A unique identifier for each governor.	Auto Value – Integer	1,2,3,4,5
Firstname	The first name of the governor.	String	John
Surname	The last name of the governor.	String	Robinson
ElectionDate	The date that the governor was elected.	Date	17/7/2009
EndDate	The date that the term of the elected governor ends.	Date	17/7/2013

"governor" table: **GovID**, Firstname, Surname, ElectionDate, EndDate.

Data Dictionary of the "coursestaken" Table

Field Name	**Description**	**Data Type**	**Example**
TakenID	The unique ID for each course taken.	Auto Value – Integer	1,2,3,4,5
GovID	A foreign key for the governor ID.	Integer – Foreign	1,2,3,4,5
CourseID	A foreign key for the course ID.	Integer – Foreign	1,2,3,4,5
DateTaken	The date that a course was taken.	Date	17/7/2013

"coursestaken" table: **TakenID**, *GovID*, *CourseID*, DateTaken.

Data Dictionary of the "course" Table

Field Name	**Description**	**Data Type**	**Example**
CourseID	Primary key for the Course.	Auto Value – Integer	1,2,3,4,5
CatID	Foreign key for the category.	Integer – Foreign	1,2,3,4,5
CourseName	The course name.	String	Food preparation

"course" table: **CourseID**, *CatID*, CourseName.

Data Dictionary of the Category Table

Field Name	**Description**	**Data Type**	**Example**
CatID	Unique identifier for the category.	Auto Value – Integer	1,2,3,4,5
CategoryName	The category name	String	Core; Achievement of pupils; Quality of teaching.

"category" table: **CatID**, CategoryName.

Methods of Access

When accessing my database, I will be constructing MySQL queries. As MySQL has many constructs that allow me to access the database and retrieve information, I will be using these to aid my retrieving of data.

When I want to view records from the database, it may be helpful to retrieve how many records I have in a database. To do this, I will use a SELECTE statement and the COUNT function.

I will structure it like this in my database:
SELECT COUNT (*) from <tableName> //count the amount of records in the given table
I think that this will be useful in my program to count the amount of records that I am going to add to a table before adding them. An example of this being used in my program would be if I wanted to create a table of all the categories in the database:
SELECT COUNT (*) from category

If I want to add something to my database, I could use an INSERT statement that would insert a new record into a table. An example of how this would be structured would be:
INSERT into <tableName> (Fieldname1, Fieldname2) VALUES (FieldData1, FieldData2) // Adds a new record into the given table name to populate fields with values
An example of how I could use this in my program would be to add a category for instance:
INSERT into category (CategoryName, CategoryDate) VALUES ('Child Care', '12/12/12')

If I need to find something in my database with a given criteria, I can use the SELECT function with a WHERE statement that will allow me to retrieve the record(s) I require. An example would look like:
*SELECT * from <tableName> WHERE <fieldname> = <fieldvalue> // finds a record in the given table where the given field value matches the value in the field in the database.*
I could use this in my program to find a course from a list that the user wants to edit. Used in this context, it might look like:
*SELECT * from course where CourseID = '3'*

After editing a record, I may need to update it. I can use the UPDATE statement to update a record where a criteria matches in the database. An example of how this statement would look:
UPDATE <tableName> SET <fieldname1> = '<fieldvalue1>', <fieldname2> = '<fieldvalue2>' WHERE <Fieldname3> = <Fieldvalue3> //update the record in the given table and replace the field values in the record with the new given values
For instance, I could use this to update an edited course record like so:
UPDATE course SET CourseName = 'Child Wellbeing', CatID = '4' WHERE CourseID = 3

I may need to delete records from the database. To do this, I can use the DELETE statement. An example of what a DELETE statement would look like:

DELETE from <tableName> WHERE <fieldname> =’<fieldvalue>’ //delete a record from the given table but only where the criteria is met
I might use this statement in my program when the user wants to remove a record from the database like a course. An example of what the statement might look like in my program would be:
DELETE from course WHERE CourseID = ’7’

Validation

Field

For many of the fields in my program, I anticipate that I will implement several types of validation checks. In combo boxes, I will need to implement a presence check to ensure the user has actually selected something from the combo box. Here is some pseudo code that would allow me to check:

```
IF (combo box == empty)
        print “there was an error, please select an item from the combo box”
ELSE
        continue operation
END IF
```

This doesn’t have to be used just on combo boxes, but it could also be used on text fields as well.

As I will be predefining my data types, I won’t need to include any type validation. I will also be converting between data types to insert into my database and converting data from my database to use within my program. This way, there won’t be any errors with data types that the user can induce on their own.

I will have length checks implemented into the format validation as I will be creating a class with string validation as I will have to validate the first and last name of the governors, the course names and category names. If I place this all in one class I won’t have to repeat code throughout my project and thus hopefully create fewer errors.

As I will be getting my user to input dates, it is critical that these dates match up with the dates on the calendar. For example, there it’s a 32nd of any month. Luckily, Java has a function called “IsDate” that will return a Boolean value on whether a user defined date is valid or not. Some pseudo code to display this:

```
myInputDate = dateFromField
isDateValid = IsDate(myInputDate)
IF(isDateValid == true)
        Continue operation
ELSE
        print “The date is not valid, please check to see if the date is valid”
END IF
```

This would allow me to find out if a date that the user has input is valid or not. If it is, then I can continue to place it into my database. Else, I can inform the user.

Test Data and Format Checks

The user will primarily be inputting text into the fields of the program so I need to ensure that the format validation is working and designed well. For instance, I will need to design the fields that it will accept typical, erroneous and boundary data. I will be using format checks to validate multiple things. I will validate the governor's first and last names, the course names, category names as well as dates. Luckily, Java has a format function which will allow me to easily convert a string into a date using the format function. This will also prepare it for being added to the database.

First name validation:

Data Type	Validation	Typical	Erroneous	Boundary
String [10] characters long	Must begin with a capital letter. May only include any number of upper or lower case letters, dashes or apostrophes. Must not be blank.	James	J8mes	Mic-h'a'el

Last name validation:

Data Type	Validation	Typical	Erroneous	Boundary
String [16] characters long	Must begin with a letter. May only include any number of upper or lower case letters, dashes, spaces or apostrophes. Must not be blank.	Robinson	R0bin~so##n	Jack-s 'onhenry

Course name validation:

Data Type	Validation	Typical	Erroneous	Boundary
String [16] characters long	Must begin with a letter. May only include any number of alphanumeric characters. Must not be blank.	Child Care	2Parent Communication #####	Financing 29

Category name validation:

Data Type	Validation	Typical	Erroneous	Boundary
String [16] characters long	Must begin with a letter. May only include any number of alphanumeric characters. Must not be blank.	Legal	Legal Care and Living #200@	Category Number 2

Sample Error Messages

Some of the sample error messages that I will be using may include:

Please ensure all fields are filled out.
Please make sure the <name of field> field is filled out.
Please make sure you have selected a <field name> from the drop down box.
Please make sure that your governor's first name only has up to 10 letters including apostrophes and dashes.
Please make sure that the date is formatted like so: day/month/year

These error messages and others similar to these should be clear enough that the user knows what has gone wrong and that they can fix the issue with maximum guidance.

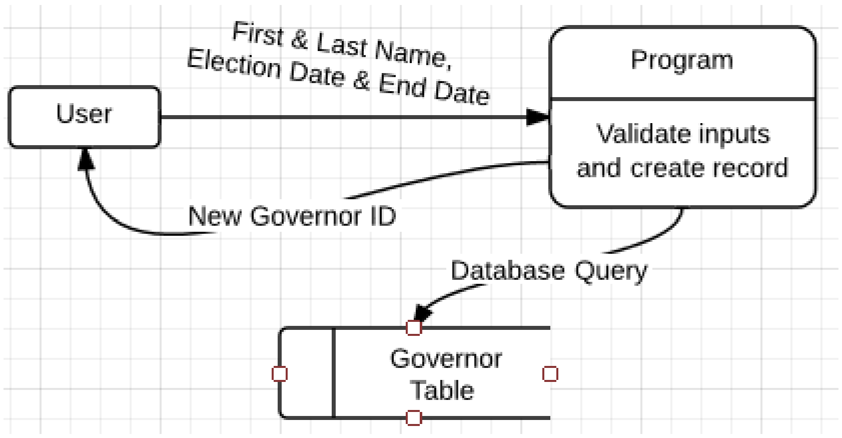
Processing Stages

Data Flow Diagrams

Throughout this section, I will focus on mainly dealing with governor records for the data flow diagrams, but the diagrams for other processes will be very similar with different sets of data being moved around. The data will almost always be moving the same way as with these diagrams.

Adding a Governor

This is the DFD for adding a governor. This same DFD can be applied to adding other items to the databases such as courses and categories.

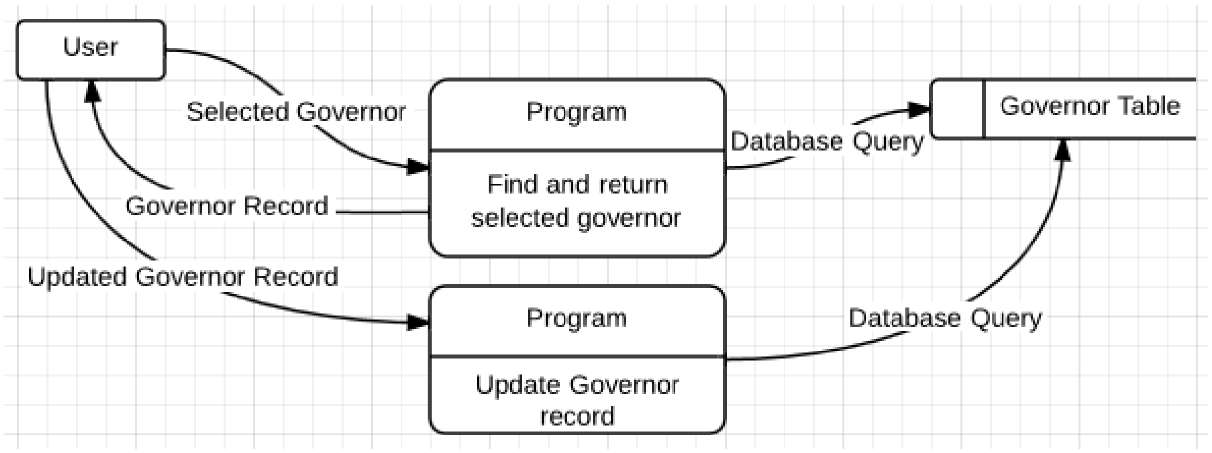


First & Last Name, Election Date & End Date
User
Program
Validate inputs and create record
New Governor ID
Database Query
Governor Table

The user will supply the governor details which is then turned into a record and applied to the database with a database query. The program then returns the governor ID to the user.

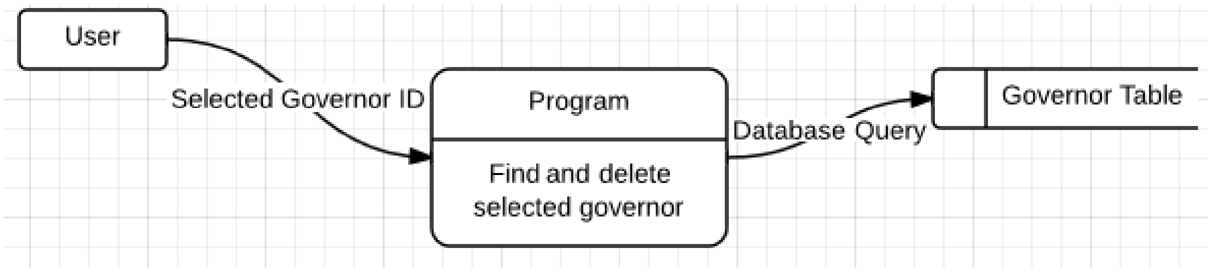
Editing a Governor

This data flow diagram can be used as an example for updating or editing any record in the database.



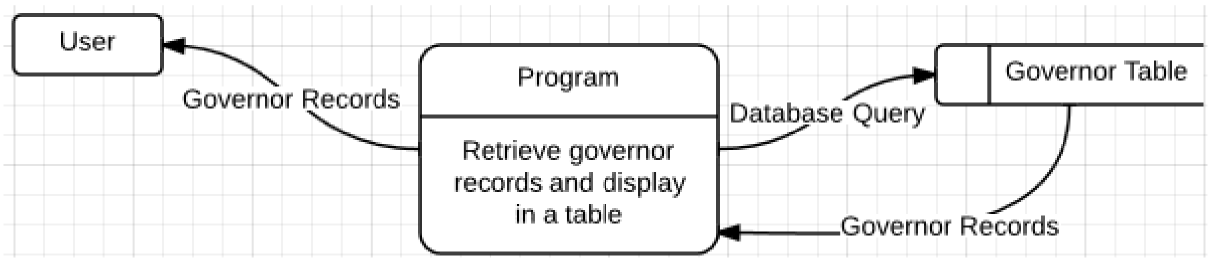
The user will supply the governor from a list, and then the program will retrieve the governor record from the database with a query then return the governor record for editing. Once the editing is completed, the new record can be updated in the database with another query.

Removing a Governor



This DFD displays how a record can be deleted from the database. The database query will contain the governor ID supplied by the user and will remove the governor that is linked to the ID. Other records can be deleted in this way.

Viewing Governors



The DFD here retrieves the governor records from the database and process them for adding to a table to display to the user. This can be applied to multiple different scenarios where the records pulled from the database will be from different or multiple tables at the same time.

Processes

Adding a Record

One of the main problems I will be encountering in my project will be the need to add records to the database. This is the most important process to get working as it will determine my ability to manipulate the data in the database. Without any information in the database, I have nothing for my user to manipulate. An example of a record that I will need to add will be the governor records. I will use adding a governor record as an example.

```
INPUT firstName, lastName, electionDate, endDate
IF (inputs are not blank)
    SQL = “insert into governor table field(firstName, lastName, electionDate, endDate)
    error = SQLExecute
    IF (ERROR)
        DISPLAY error message
    ELSE
        SQL = “get last inserted key”
        key = SQLExecute
        DISPLAY key
    END IF
ELSE
    DISPLAY error message
END IF
```

Removing a Record

Here, my problem is that I need to be able to allow the user to remove records from the database. For instance, if the user wants to remove a course from the database, the user needs to eb able to select that entry into the database and remove it. For instance, the pseudo code may look something like the following:

```
INPUT courseID
IF (input is blank)
    DISPLAY error message
ELSE
    SQL = “remove from course table where course ID matches the input courseID”
    error = SQLExecute
    IF (error)
        DISPLAY error message
    ELSE
        DISPLAY success message
    END IF
END IF
```

This set of code should allow for a record to be removed from the database and that is how I will handle the removing of any record in the database with a similar method.

Editing a Record

Editing a record almost combines the problems of adding a record and removing a record. I need to have an input from the user as to what record from a table they would like to edit, and then allow them to edit it and then update it in the database. So for instance, I could use the example of editing a category and show some pseudo code on how I would go about solving this problem:

```
INPUT categoryID
IF (input is blank)
        DISPLAY error message
ELSE
        SQL = "SELECT from category table where categoryID matches input category ID"
        Error = SQLExecute
        IF (error)
                DISPLAY error message
        ELSE
                DISPLAY success message
                SET categoryName field with categoryName from the record
        END IF
END IF
 //wait for the user to be done editing the fields and click continue
        //in a new method
INPUT new categoryName and old categoryID
IF (inputs are blank)
        DISPLAY error message
ELSE
        SQL = "update table category field (categoryName) with new categoryName where
        categoryID matches input categoryID"
        Error = SQLExecute
        IF (error)
                DISPLAY error message
        ELSE
                DISPLAY success message
END IF
```

This method should allow for the user to edit a record and to have it updated in the database. Any similar problems where a record has to be update, it can be done so by using a method similar to this one. The only thing that would change would be the input data and the manipulation of the new field data with the SQL statement.

Validation

Since I am going to be creating a class to store my validation methods in, all I need to do to validate something is pass the parameters to the method and wait for a return. I will use the validation passing of the governors' first name as a model for creating the following pseudo code:

```
INPUT firstName
IF (input is blank)
    DISPLAY error message
ELSE
    isValid = validateFirstName(firstName)
    IF (isValid = FALSE)
        DISPLAY error message "the validation failed"
    ELSE
        Continue with operation
    END IF
END IF
```

The above code is the first half of the validation process. The second half comes from the actual validation class itself:

```
INPUT firstName
IF (firstName matches Regex validation pattern)
    isValid = TRUE
ELSE
    isValid = FALSE
END IF
RETURN isValid
```

This will be applied to all my validation methods in the validation class and will allow for the same validation to be used across my entire program hopefully removing bugs relating to where I've copy and pasted validation code from one location to another. This ensures that my validation methods are consistent.

Evaluation Criteria

Usability

To test if the program is usable, I will ask a test user with some given test data to sit down and use my program. I will then ask them some questions on how they think the program was suitable and usable. Any feedback here that I am given will be used to discuss how usable I think the program is.

Functionality

To test how functional the program is, I intend to write a test plan that will incorporate a run through of my entire program. I will not test every single feature as the user would be using it in my test runs as that would take an incredibly long time, but I aim to test a run through of the most commonly used features in the program. Outside of the test runs, I will create and conduct tests for the other major and minor functions to ensure that they work.

For the functionality to pass, I need the main purpose of every function that I test to work correctly as intended. Other bugs like spelling errors or display layout errors that don't impact on the functionality of the program can be considered minor bugs and be fixed relatively easily I would hope.

When testing the functionality of the program, I will ensure I use some test data. This test data will include typical, erroneous and extreme data. This will test the validation of the program to make sure the user doesn't enter something that isn't meant to be entered.

Suitability

This is an objective criterion in that what is suitable to one person may not actually be suitable at all to another person. I will have to try to take an unbiased stance and think from the person who is using the programs perspective and decide whether or not the program is actually suitable. Of course, when I get some user feedback I will be able to use what they think about the program to decide if the program is actually suitable.

Original Objectives

At the end of creating my program I will take a look at my original objectives and see how well they are implemented in the program. I will compare features that are implemented in the program and the objective they were designed to meet. If I feel enough objectives are met and that the functions are well enough implemented to meet the objective, then I will consider that objective met. My main objectives are the most critical and I aim to meet all of my main objectives. Secondary objectives are not quite as critical and won't result in a failure state if they're not all implemented.

Overall Success

The overall success of the project will be judged on how well I think the usability, functionality, suitability are met and implemented. Of course, it also depends on my original objectives and whether or not I've successfully met as many as possible to make the project an overall success.

I think that the project will be an overall success if I can confidently say that I can give my program to my client in a usable, functional and suitable state with enough objectives me that the program can actually be used for the intended purpose.