ST JOHN'S COLLEGE

A web based mapped information and reservation system of St John's College

IT PAT, St John's College 2019

System Design Document

Gordon Fleming

Table of Contents:

User Interface Design:

All diagrams/screenshots follow after the GUI descriptions

Description

Booking Tool

- The booking form will allow users to book certain, available time slots that work within the already congested timetable.
- The form has several different drop downs, boxes and fields.
    - Venue
        - Drop down
        - What classroom the user wants to book
        - The contracted classroom names are used
    - Day
        - Drop down
        - The day of the week the user wants to book
        - Only for school days (Mon-Fri)
    - Week
        - 'Blue' or 'Maroon' week
        - Either needs to be selected
        - On load 'Blue' is selected
    - Period
        - Drop down
        - Only Periods 0 to Period 9 are considered
    - Staff
        - Text field
        - The staff members name, (the person who will be in the venue for the specified time, not necessarily the person making the venue booking).
    - Activity
        - Drop down of the subjects that will be taught during the user's specified time.
    - Data
        - HTML date calendar input, for specific day of booking.

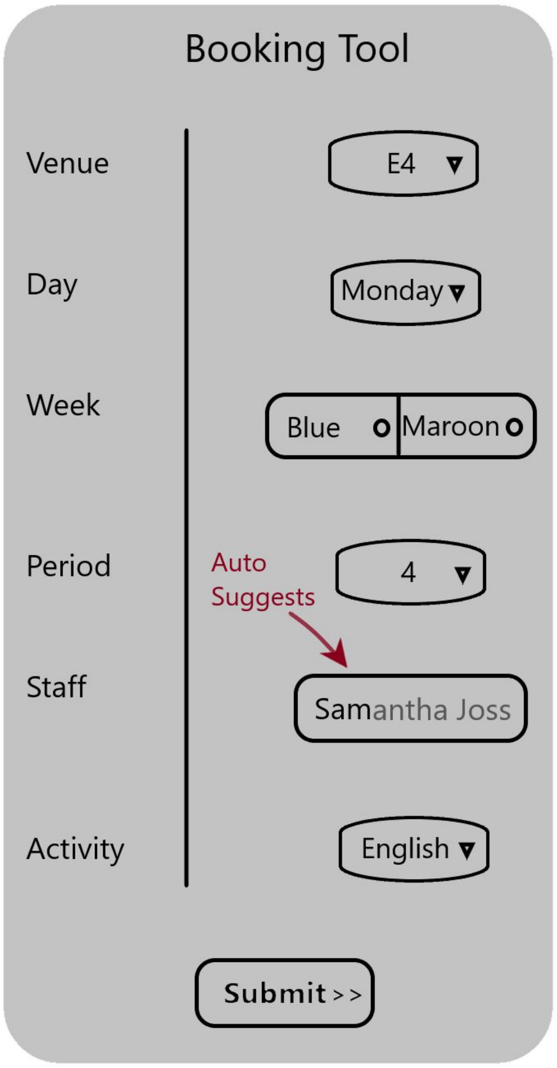
Actions

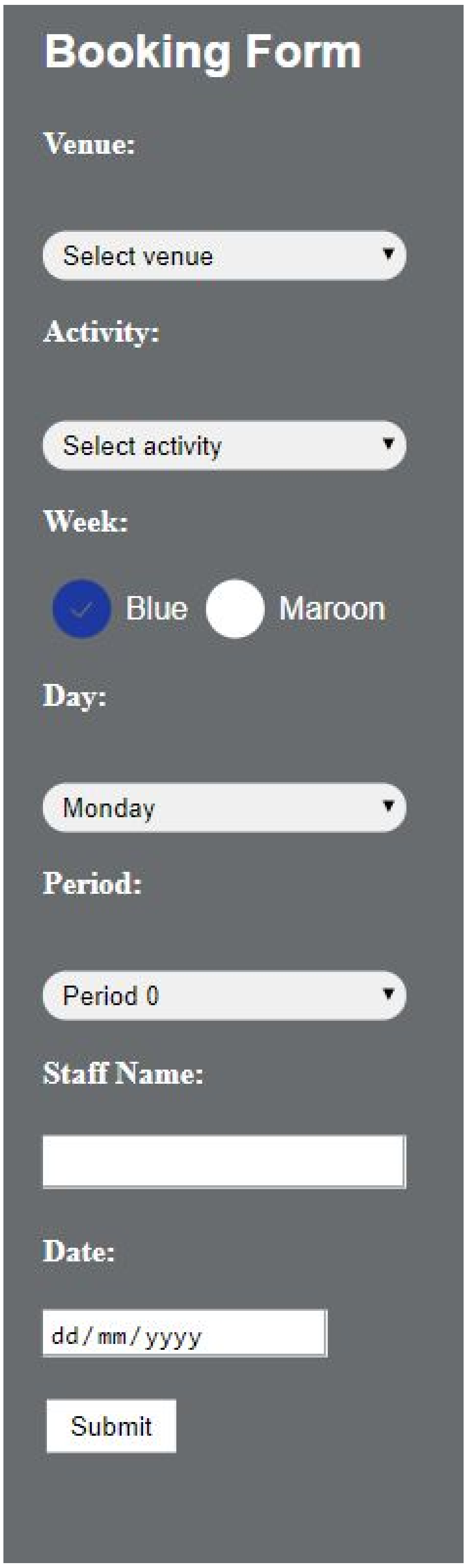
- Once complete and all fields are filled in, the user will need to toggle the submit button, triggering the java 'Booking' servlet, transfering users details into SQL and inserting the data.
- Values will then be inputted to the database's bookings table.
- If the booking is valid and successful the added record will be added to the database and success message shown on the user's screen.
- If unsuccessful, an error message will pop-up to warn the user.
- The added record will be shown on the map, at the times specified in the booking.
- This will in turn inform other staff members of what is happening in the venue, that would be otherwise considered open and free.

Data

- The users data will be inputted into a relevant SQL statement.

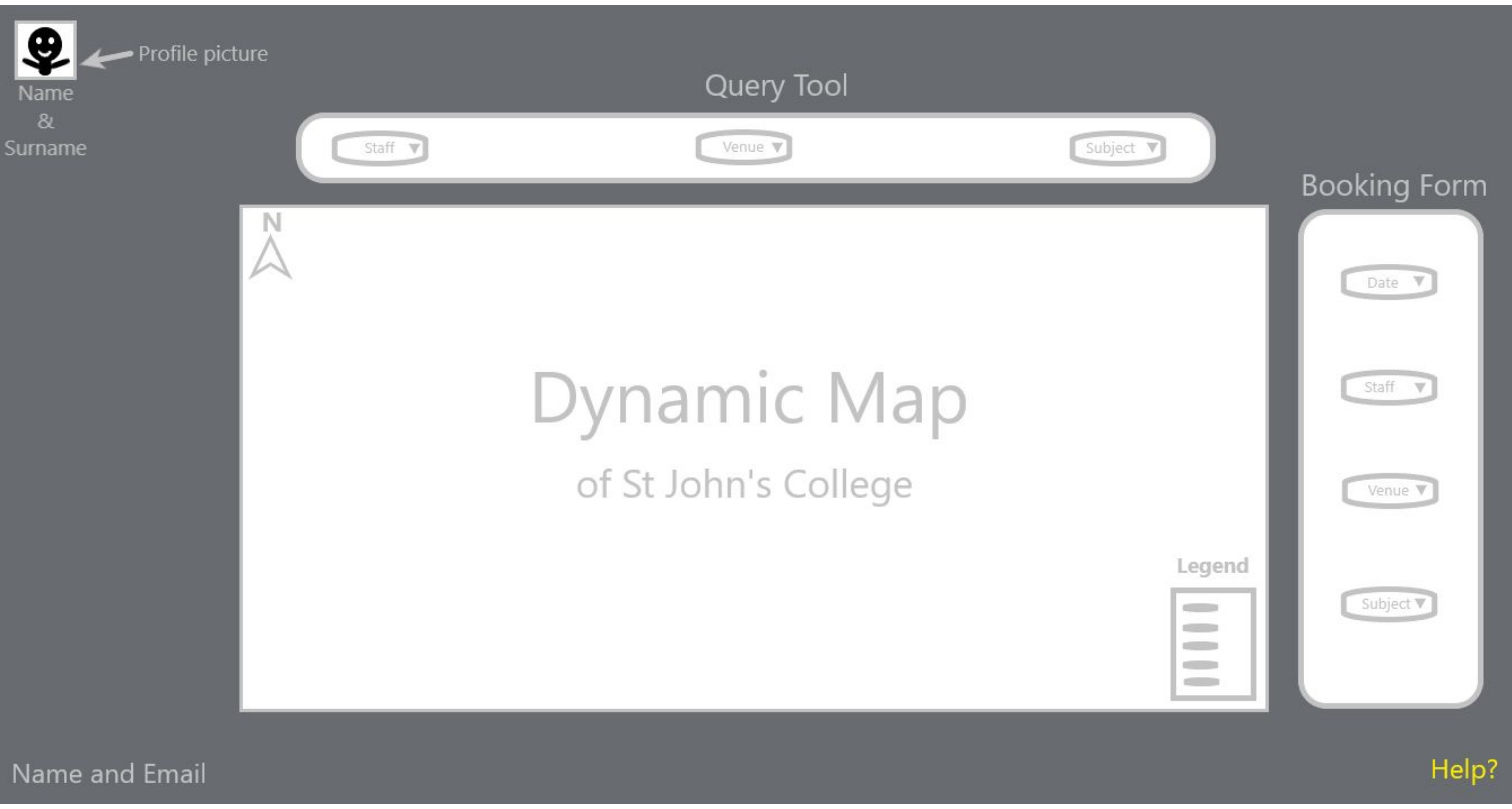
Planned:	**Actual:**



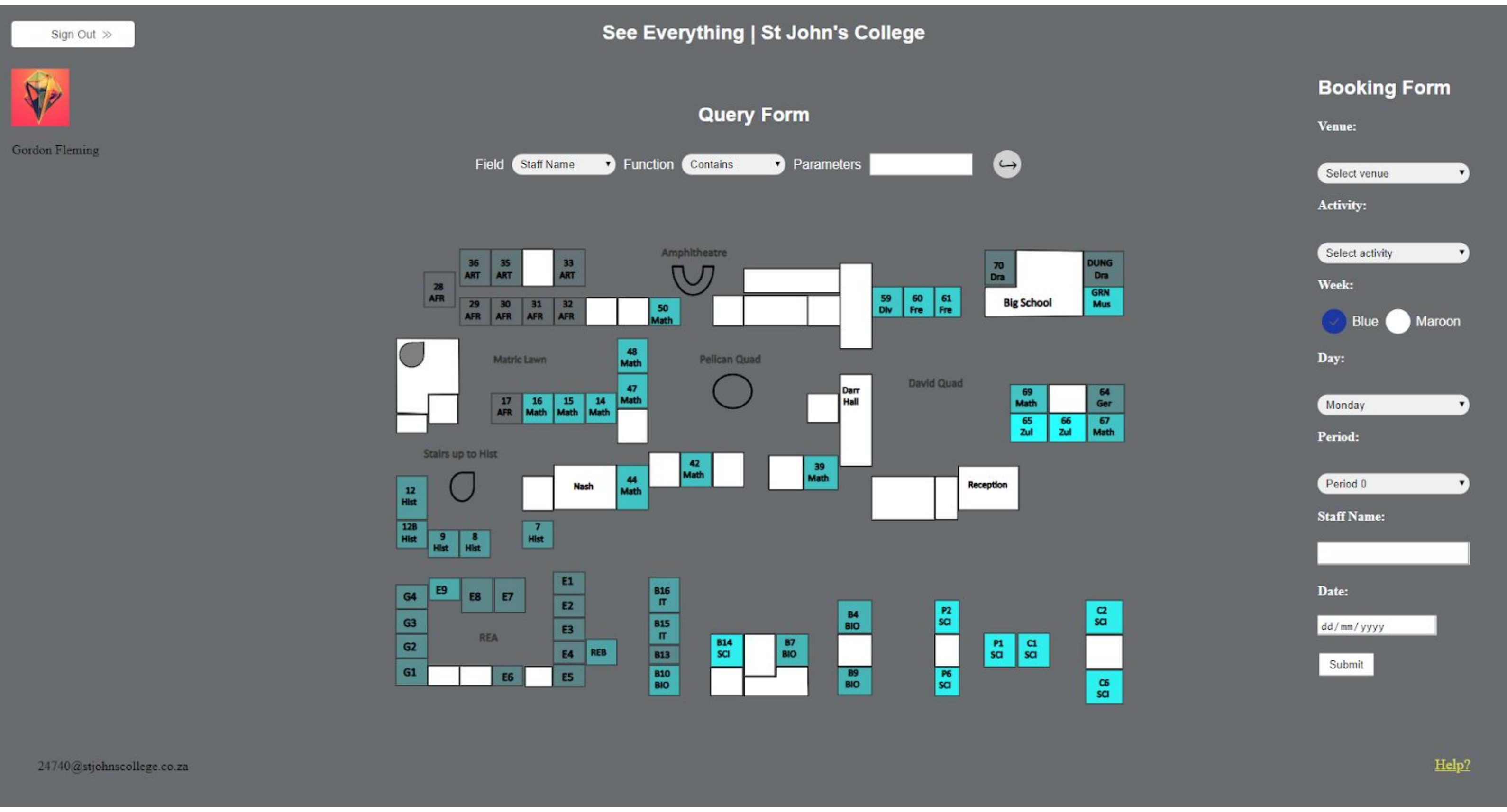


Description	**Overall Layout of the main page** ● The dynamic map, query tool and booking form are all elements that work with each other and communicate amongst each other to produce productive and useful results to the user by communicating through the backend. ● The query tool and booking form are all described at their respective separate descriptions.
Actions	● The 'Help?' will, when toggled, open a new tab in the browser containing an image with help and descriptions of all the elements on the screen. ● When 'Sign out' is toggled using the google api, the user will be disconnected and signed out of their account and redirected to the login screen. ● Actions of both booking forms and query tools are, again described at their respective separate descriptions.
Data	● The map will receive all its data from the connected database and JSON file containing venues, timings and staff names. ● The query and booking form will update and query the database which will in turn change the map. ● All other data for these elements are all described at their relevant screen descriptions… ● Top left user info is static data fetched using google's api.

Planned:

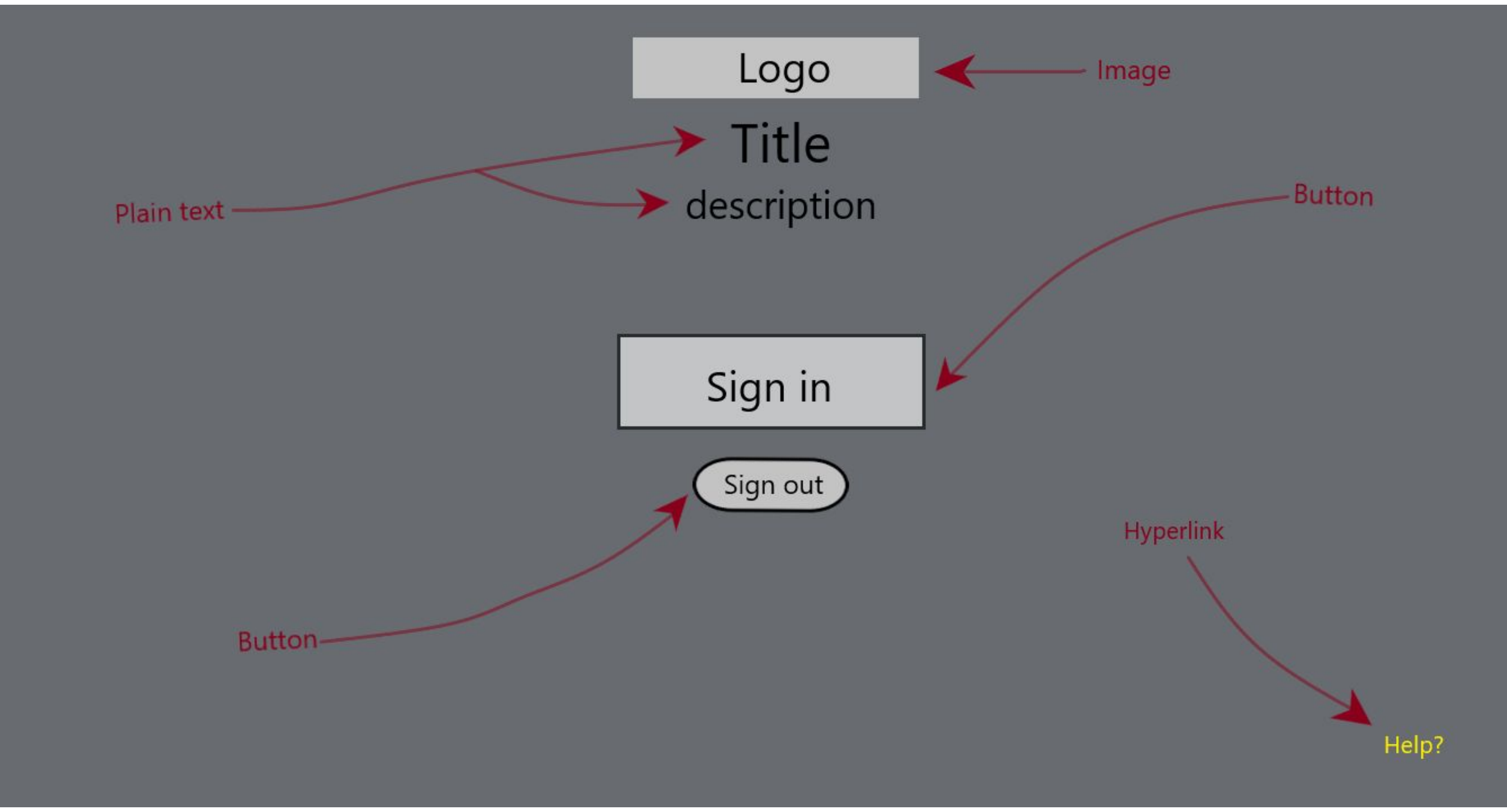


Actual:

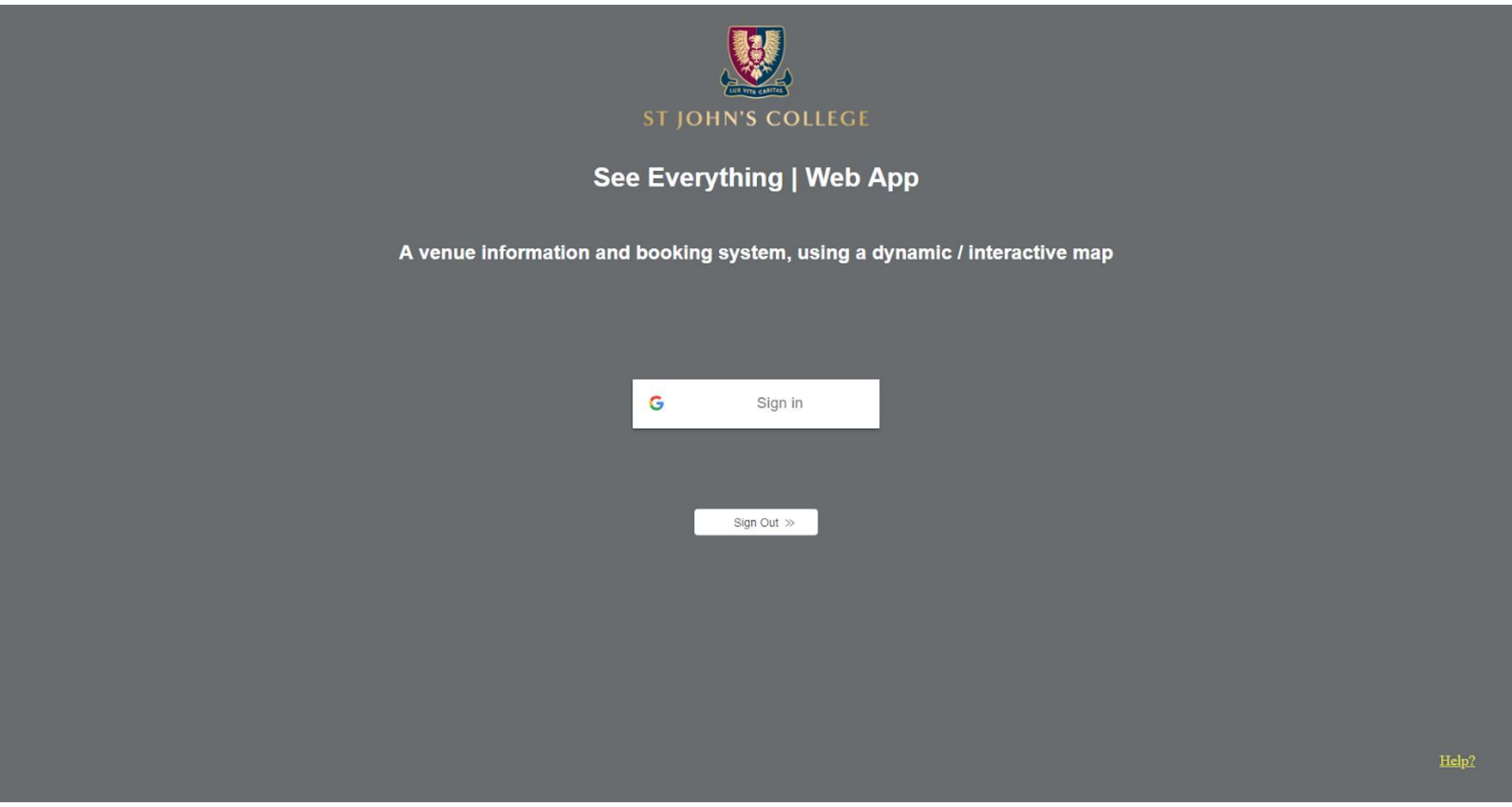


Description	**Login Screen** ● This screen is the first screen the user will encounter when accessing the page. ● Successful login procedure is needed to access the main page. ● The logo and all other text is static and non-interactive.
Actions	● When the 'Sign in' button is toggled a new window will open with google's sign in process. ● Once completed successfully the user will be redirected to the main page. ● If unsuccessful google will return a relevant error message and the window will have to be closed for the user to try again. ● The help hyperlink, located in the bottom right will, when toggled open a new window with an image containing the 'help tips'.
Data	● The login page doesn't access any locally stored or server side stored data except the 'help image' when the hyperlink is toggled. ● The details for login and data needed to validate and verify the users is all managed and stored with google.

Planned:

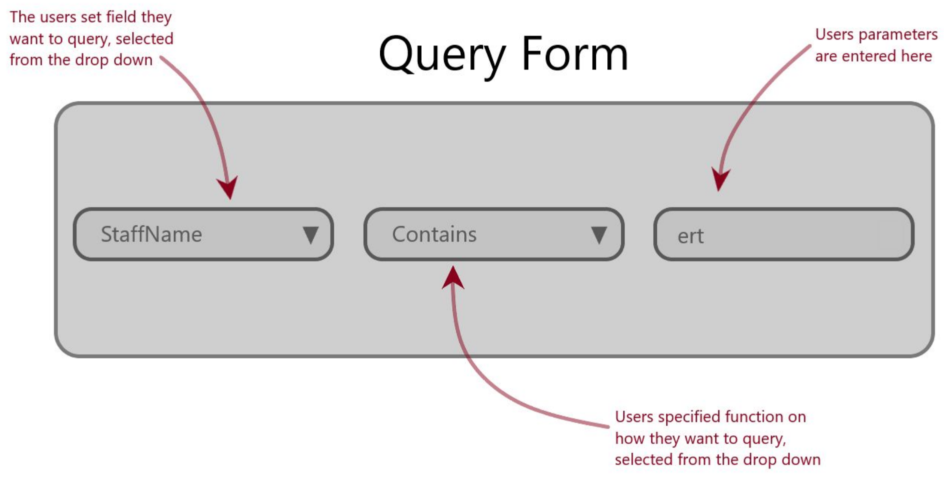


Actual:

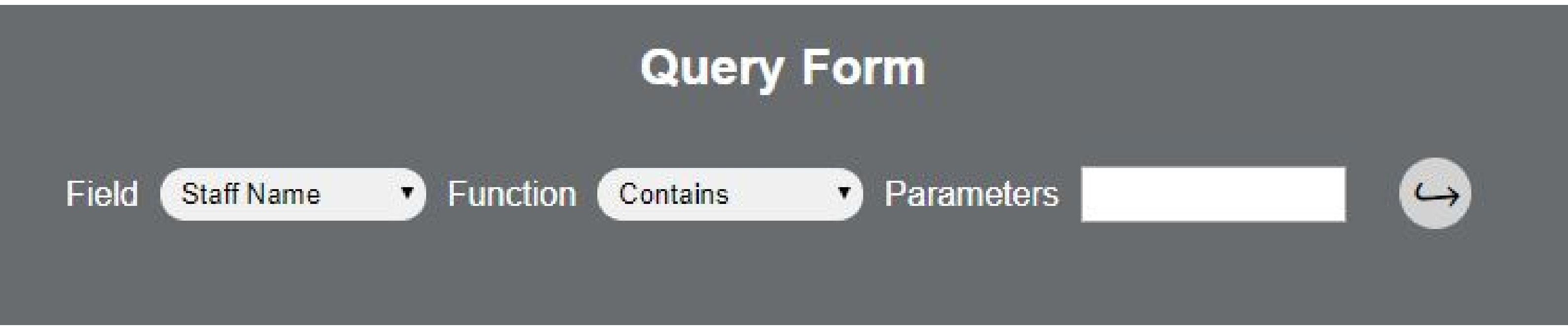


Description	**Query Form** ● Query form located on the main page above the dynamic map. ● The form will be a bar with a 'title' and 3 boxes ● The first box ○ Handles what field the user wants to query ● The second box ○ Handles the type of query / search the user wants ● The third box ○ Handles the users custom parameters ● The query form will not be creating any data, but only accessing the data and querying the specified parameters.
Actions	● The user will then submit their query once finished, triggering a response from the backend. ● Through what the user has requested, a relevant SQL statement will be compiled and results will be shown on the dynamic map once the site has been refreshed.
Data	● The form will access its relevant data from the connected database and JSON file, such as staff names and venue details.

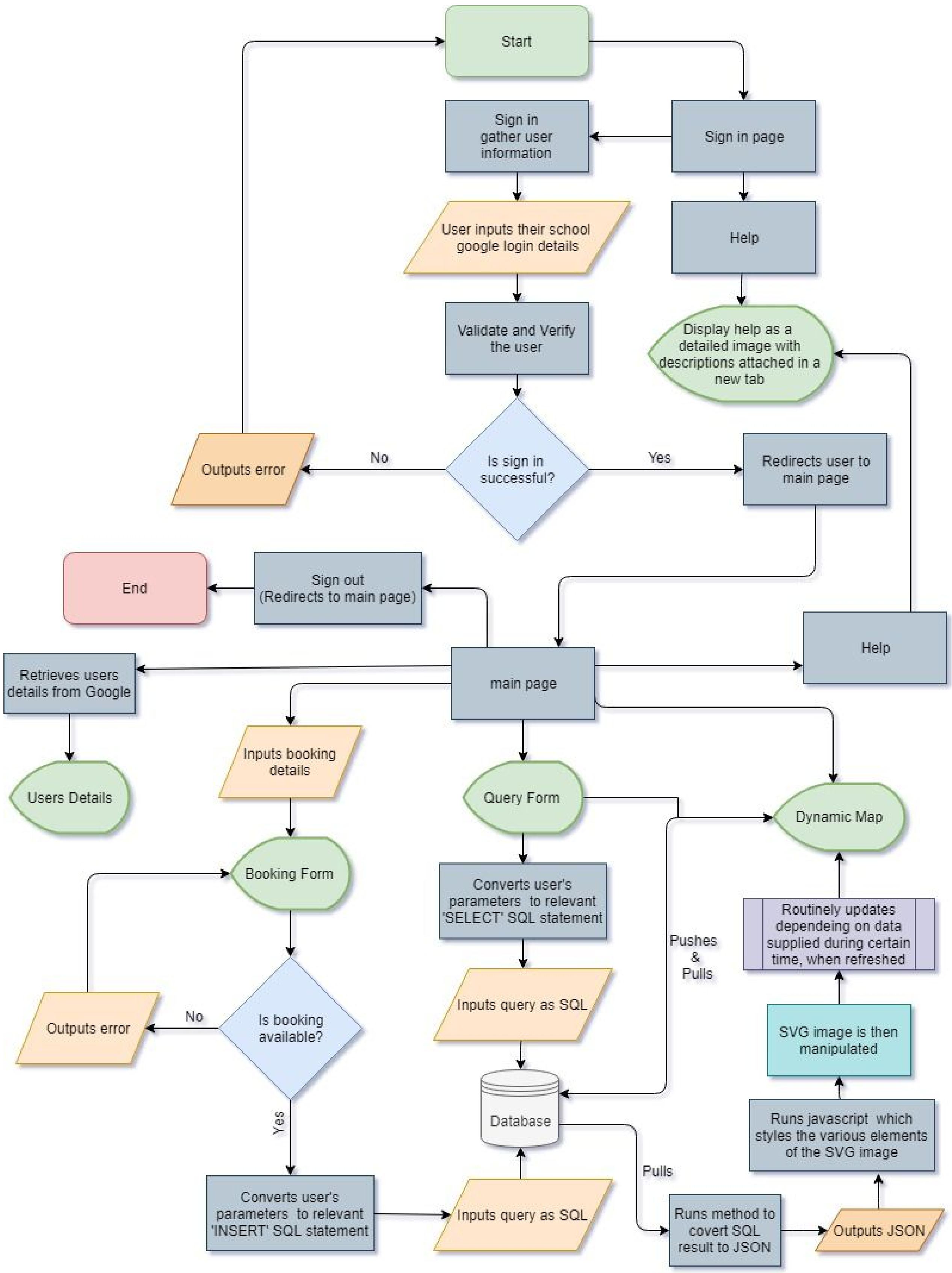
<u>Planned</u>:


The users set field they want to query, selected from the drop down
Query Form
Users parameters are entered here
StaffName ▼
Contains ▼
ert
Users specified function on how they want to query, selected from the drop down

<u>Actual</u>:



Sequencing:


Start
Sign in gather user information
Sign in page
User inputs their school google login details
Help
Validate and Verify the user
Display help as a detailed image with descriptions attached in a new tab
Outputs error
No
Is sign in successful?
Yes
Redirects user to main page
End
Sign out (Redirects to main page)
Help
Retrieves users details from Google
main page
Users Details
Inputs booking details
Query Form
Dynamic Map
Booking Form
Converts user's parameters to relevant 'SELECT' SQL statement
Pushes & Pulls
Routinely updates dependeing on data supplied during certain time, when refreshed
Inputs query as SQL
Outputs error
No
Is booking available?
SVG image is then manipulated
Database
Yes
Runs javascript which styles the various elements of the SVG image
Pulls
Converts user's parameters to relevant 'INSERT' SQL statement
Inputs query as SQL
Runs method to covert SQL result to JSON
Outputs JSON

Explanation:

- ➔ The user will have the login screen displayed, here they can either login or sign out or click on the 'Help' hyperlink.
    - ◆ Login button when toggled will display the google sign in window in a separate tab where the user will enter their details.
        - Only accepts the school's google accounts.
    - ◆ Sign out when toggled will sign the user out and refresh the page.
    - ◆ 'Help' when toggled will open a new window containing an image.
        - 'Help' is described more further down the explanation.

- ➔ If the user is valid, the user will then be redirected to the main page, though if not valid will receive an error and will have to attempt the login again.

- ➔ The main page will then be displayed, here there are multiple functions/screens tied into one. The Booking form, User Details, Query form, Help and the Dynamic Map. To exit the main screen, the user needs to toggle the 'sign out' button.
    - ◆ Users Details
        - User information is shown, this information is fetched from users school google account.
        - Their name and profile image is displayed.
    - ◆ Booking Form
        - User will enter their booking within the form.
        - The activities and venue drop downs are populated with data fetched from their respective tables in the database.
        - Each record is then looped through to create a new drop down option in the HTML front-end.
        - If the users booking is valid, the booking details will be entered into the connected database.
        - Their booking will then be compiled into a relevant 'INSERT' SQL statement.
        - If the users booking is invalid, there will be an error message shown to the user and they will have to change their details so it complies with the database's design.
    - ◆ Query Form
        - The user will enter their search parameters into the Query form.
        - Their constructed query will be converted into a relevant 'SELECT' SQL statement.
        - This will query the database, which will in turn change the maps appearance, by automatically altering the JSON file.

- Dynamic Map (SVG image)
  - Will retrieve all its information and data of what's where at what time from the database and connected JSON files.
  - This will be converted to JSON data in the backend.
  - JSON data will then be read by Javascript and will style the map
  - The actual map and locations of classrooms will be displayed through the SVG symbolic map of St John’s which is made so each class is a seperate object, with all other venues grouped as one object.
  - The database and map locations will all be in communication with each other, in order to return valid and useful results.
- Help (When toggled)
  - Display relevant images of both screens, with help attached.
  - The help will be displayed in an image opened in a new tab in the users browser.
  - The help will explain the various elements on the screen and their respective functionality.

Class Design:

ConQuery
-SQL: String
getResultSet()

ConQuery:

- Stores the ‘String’ variable containing the SQL statement used to extract the relevant information from the database.
- Method, ‘getResultSet()’ connects to the database using the connection previously established in other classes.
- The ‘getResultSet()’ method then executes the query.

BookingControl
#doPost()

BookingControl:
- ➔ Manages all the booking requests from the user.
- ➔ Retrieves all the data values from the front-end (values the user inputted through the booking form) then converts and assigns each value a relevant java variable.
- ➔ Each variable is then ordered and inserted through a connection to the database into the ‘Booking’ table.
- ➔ The class outputs error or success messages for the various scenarios the user may experience.

DBproperty
+urlHome: String +urlSchool: String +urlLaptop: String +DB_Class: String +DB_URL: String

DBproperty:
- ➔ Stores the values of the addresses to the locations of the database, each ‘url’ is defined by which computer I am using to develop.
- ➔ Defines the ‘String’ value of both the JDBC sqlite driver and the database ‘url’ used for connection to the database.

Main

+doGet()
+getJsonObject()

Main:
- ➔ Locates the destination of the JSON file.
- ➔ Prints out the converted data to the separate JSON file in the correct format, from the ‘getJsonObject()’ method.

DbConnection

+DbConnection()
+Connection()

DbConnection:

- ➔ Establishes the connection with the database.
- ➔ Verifies if the connection is successful.

JsonServices
+getFormattedResult()

JsonServices:

- ➔ This class manages all the formatting and converting of data, ‘ResultSet rs’ is past into the methods parameters.
- ➔ The method, ‘getFormattedResult()’ reads through the column names and adds each to a ‘List’ object.
- ➔ Each ‘List’ is then converted to readable JSON data.

Query

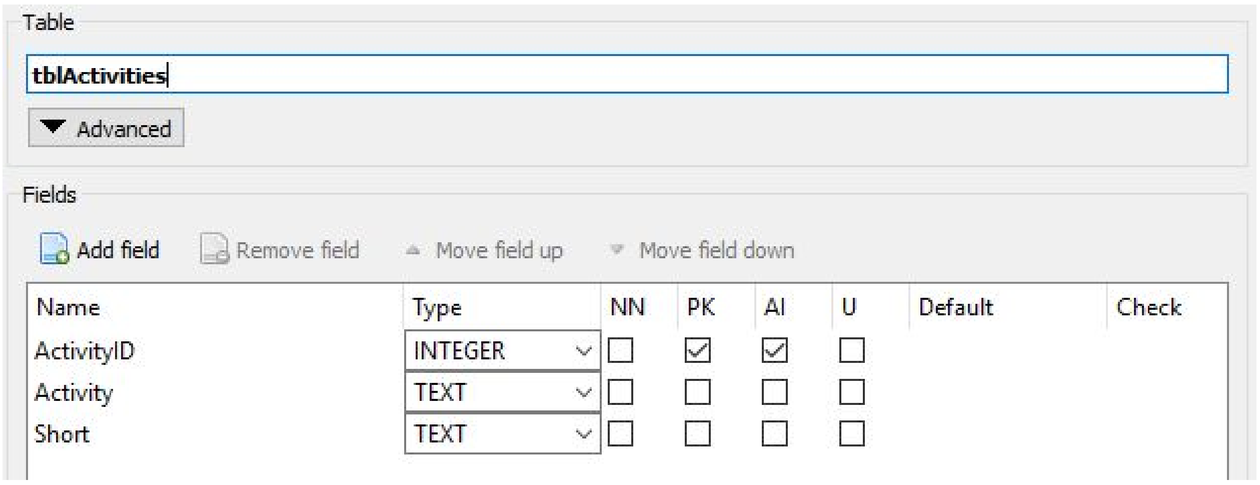
+produceTable()
+doGet()

Query:

- ➔ The class is a servlet which manages the requests passed through the query form.
- ➔ This data in turn influences the results shown on the table.
- ➔ Every time a request is run is searches for results

Persistent Storage Design:

tblActivities: Contains all data regarding venues, each record is auto incremented.



Table

tblActivities

Advanced

Fields

Name	Type	NN	PK	AI	U	Default	Check
ActivityID	INTEGER	☐	☑	☑	☐		
Activity	TEXT	☐	☐	☐	☐		
Short	TEXT	☐	☐	☐	☐		

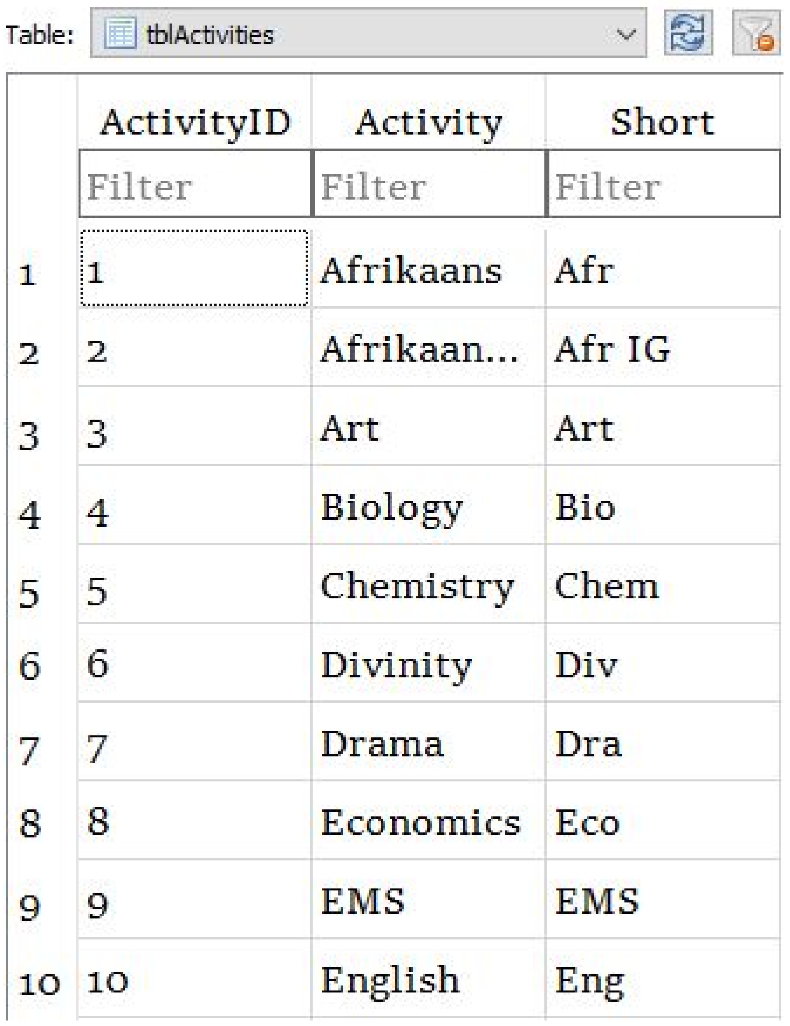
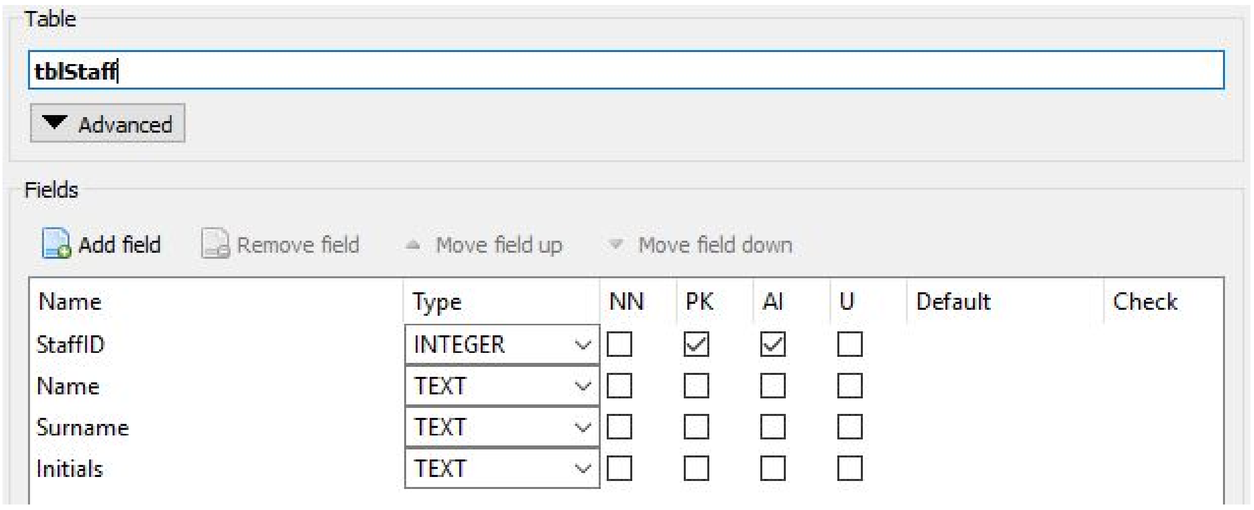


Table: tblActivities

	ActivityID	Activity	Short
	Filter	Filter	Filter
1	1	Afrikaans	Afr
2	2	Afrikaan...	Afr IG
3	3	Art	Art
4	4	Biology	Bio
5	5	Chemistry	Chem
6	6	Divinity	Div
7	7	Drama	Dra
8	8	Economics	Eco
9	9	EMS	EMS
10	10	English	Eng

tblStaff: Contains staff names, surnames and initials, all the records are auto incremented.



Table

tblStaff

Advanced

Fields

Add field Remove field Move field up Move field down

Name	Type	NN	PK	AI	U	Default	Check
StaffID	INTEGER	☐	☑	☑	☐		
Name	TEXT	☐	☐	☐	☐		
Surname	TEXT	☐	☐	☐	☐		
Initials	TEXT	☐	☐	☐	☐		

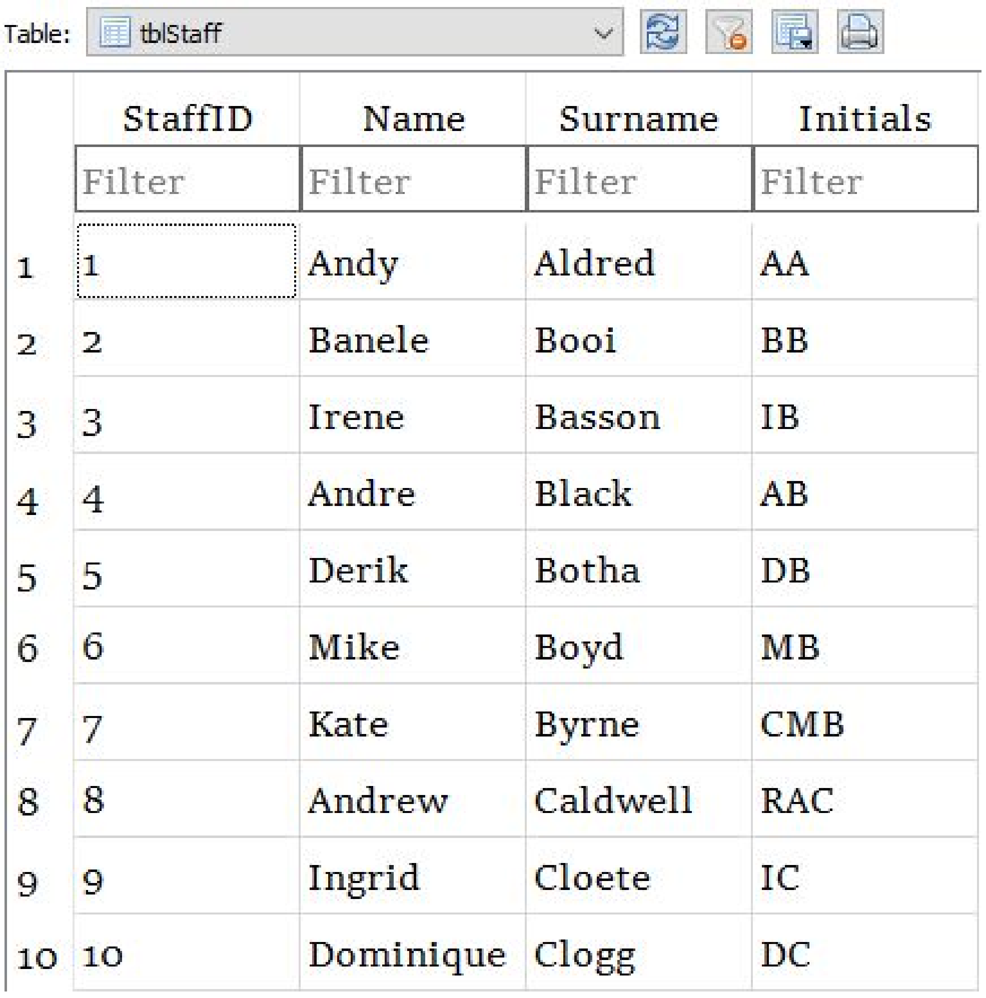
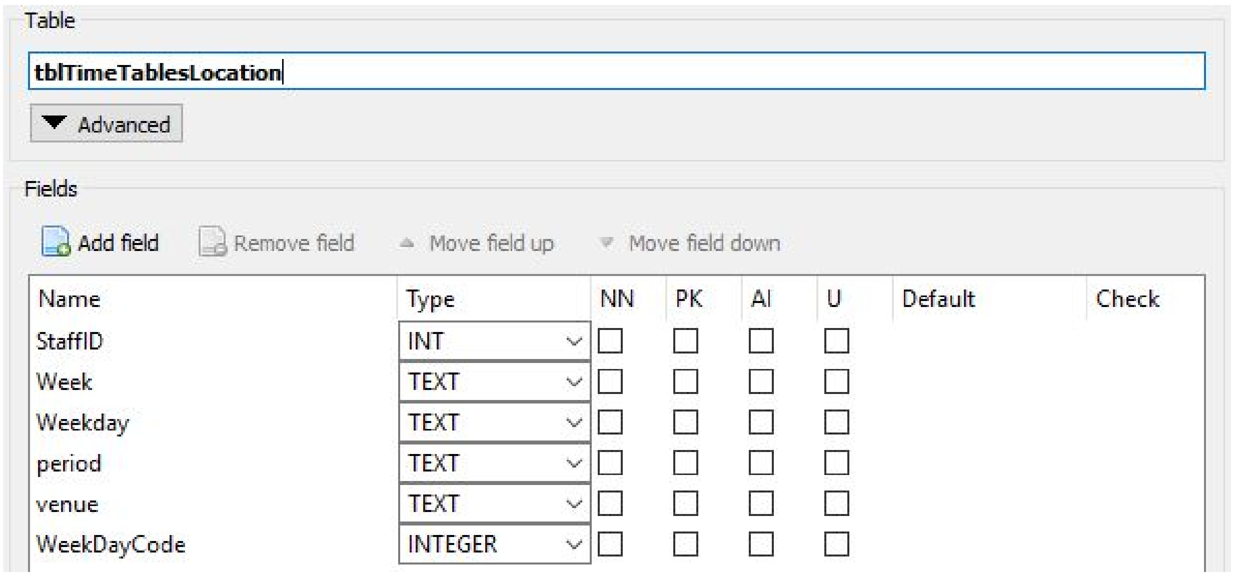


Table: tblStaff

	StaffID	Name	Surname	Initials
	Filter	Filter	Filter	Filter
1	1	Andy	Aldred	AA
2	2	Banele	Booi	BB
3	3	Irene	Basson	IB
4	4	Andre	Black	AB
5	5	Derik	Botha	DB
6	6	Mike	Boyd	MB
7	7	Kate	Byrne	CMB
8	8	Andrew	Caldwell	RAC
9	9	Ingrid	Cloete	IC
10	10	Dominique	Clogg	DC

tblTimeTablesLocation: Contains information of what teacher is where at a specific time.



Table

tblTimeTablesLocation

Advanced

Fields

Add field Remove field Move field up Move field down

Name	Type	NN	PK	AI	U	Default	Check
StaffID	INT	☐	☐	☐	☐		
Week	TEXT	☐	☐	☐	☐		
Weekday	TEXT	☐	☐	☐	☐		
period	TEXT	☐	☐	☐	☐		
venue	TEXT	☐	☐	☐	☐		
WeekDayCode	INTEGER	☐	☐	☐	☐		

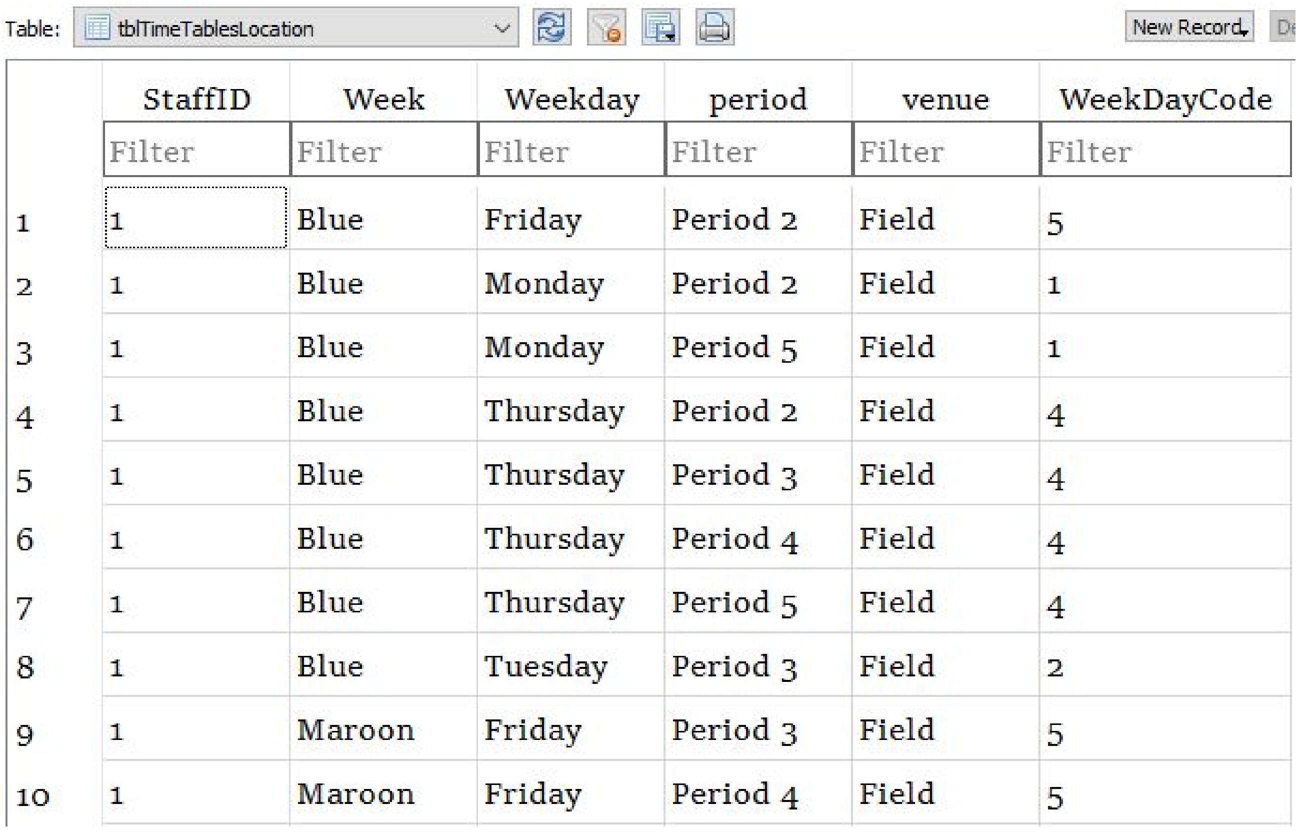


Table: tblTimeTablesLocation New Record

	StaffID	Week	Weekday	period	venue	WeekDayCode
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Blue	Friday	Period 2	Field	5
2	1	Blue	Monday	Period 2	Field	1
3	1	Blue	Monday	Period 5	Field	1
4	1	Blue	Thursday	Period 2	Field	4
5	1	Blue	Thursday	Period 3	Field	4
6	1	Blue	Thursday	Period 4	Field	4
7	1	Blue	Thursday	Period 5	Field	4
8	1	Blue	Tuesday	Period 3	Field	2
9	1	Maroon	Friday	Period 3	Field	5
10	1	Maroon	Friday	Period 4	Field	5

tblTimings: Contains data regarding the times 'Periods' occur in weeks 'Blue' and 'Maroon'.

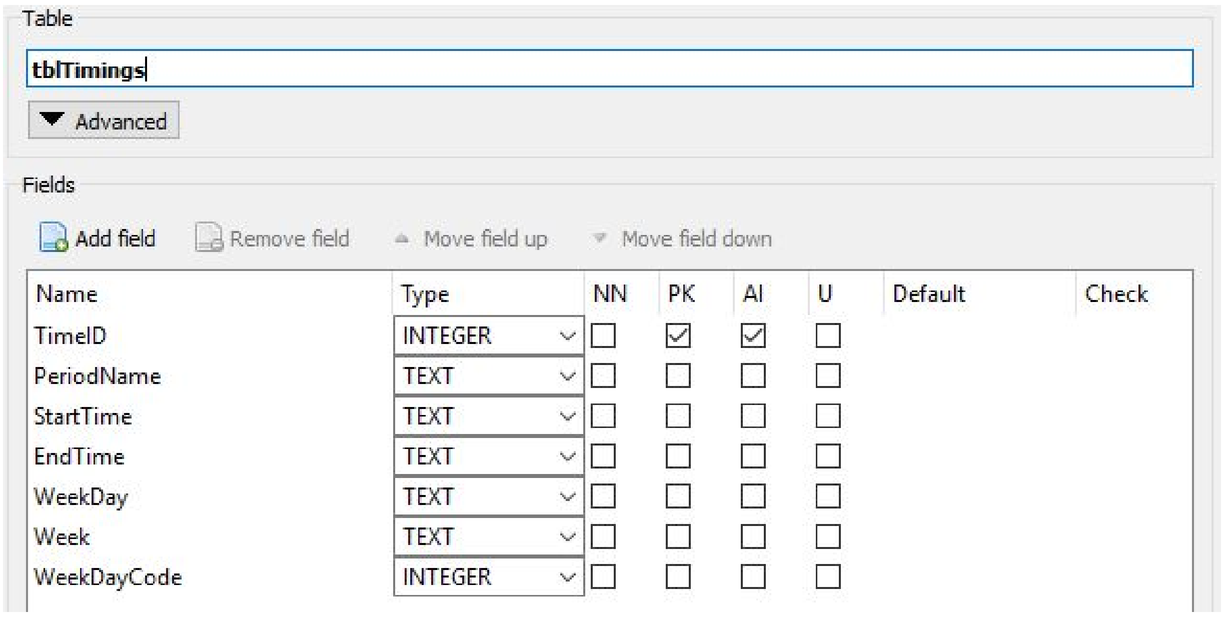


Table: tblTimings

Fields

Name	Type	NN	PK	AI	U	Default	Check
TimeID	INTEGER	☐	✓	✓	☐		
PeriodName	TEXT	☐	☐	☐	☐		
StartTime	TEXT	☐	☐	☐	☐		
EndTime	TEXT	☐	☐	☐	☐		
WeekDay	TEXT	☐	☐	☐	☐		
Week	TEXT	☐	☐	☐	☐		
WeekDayCode	INTEGER	☐	☐	☐	☐		

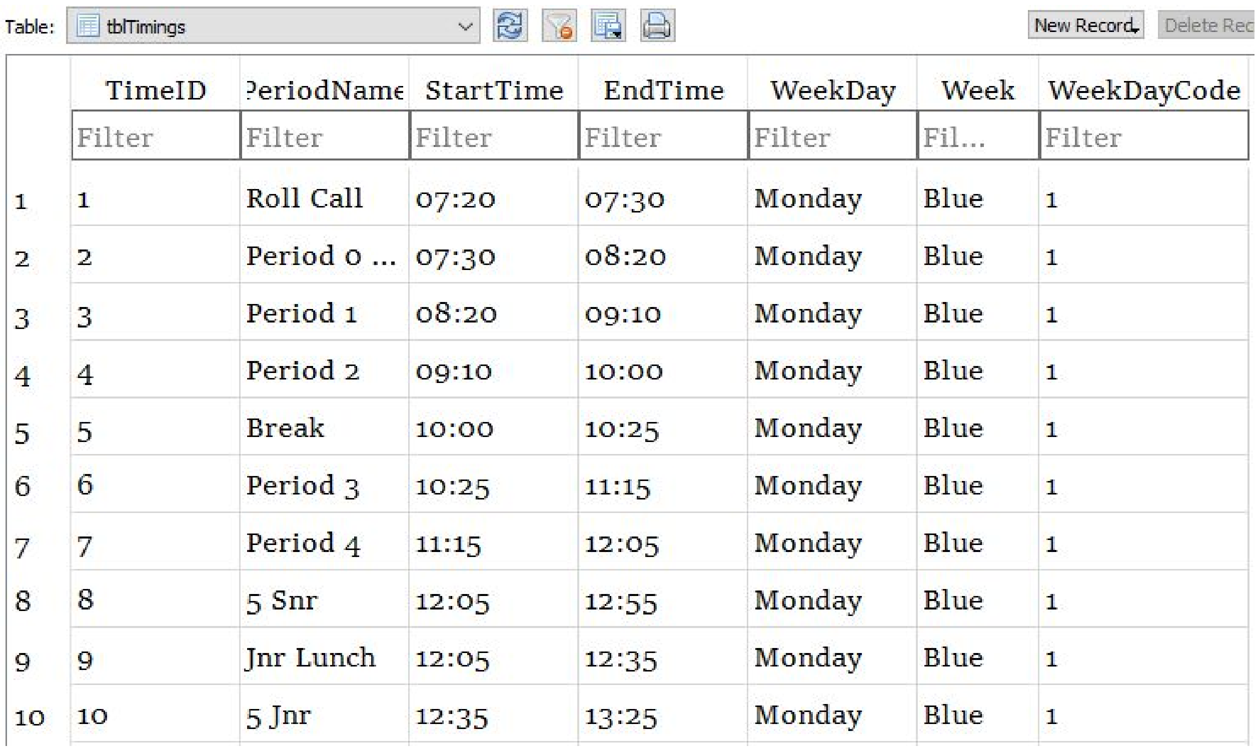
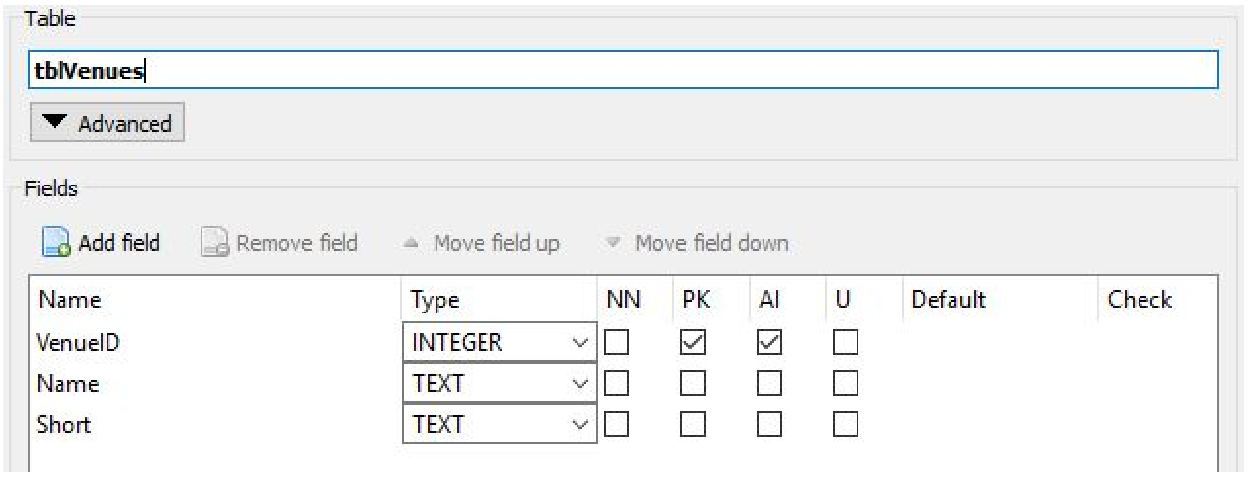


Table: tblTimings

	TimeID	PeriodName	StartTime	EndTime	WeekDay	Week	WeekDayCode
1	1	Roll Call	07:20	07:30	Monday	Blue	1
2	2	Period 0 ...	07:30	08:20	Monday	Blue	1
3	3	Period 1	08:20	09:10	Monday	Blue	1
4	4	Period 2	09:10	10:00	Monday	Blue	1
5	5	Break	10:00	10:25	Monday	Blue	1
6	6	Period 3	10:25	11:15	Monday	Blue	1
7	7	Period 4	11:15	12:05	Monday	Blue	1
8	8	5 Snr	12:05	12:55	Monday	Blue	1
9	9	Jnr Lunch	12:05	12:35	Monday	Blue	1
10	10	5 Jnr	12:35	13:25	Monday	Blue	1

tblVenues: Contains data of all subjects at the school, each record is auto incremented.



Table

tblVenues

Advanced

Fields

Name	Type	NN	PK	AI	U	Default	Check
VenueID	INTEGER	☐	✓	✓	☐		
Name	TEXT	☐	☐	☐	☐		
Short	TEXT	☐	☐	☐	☐		

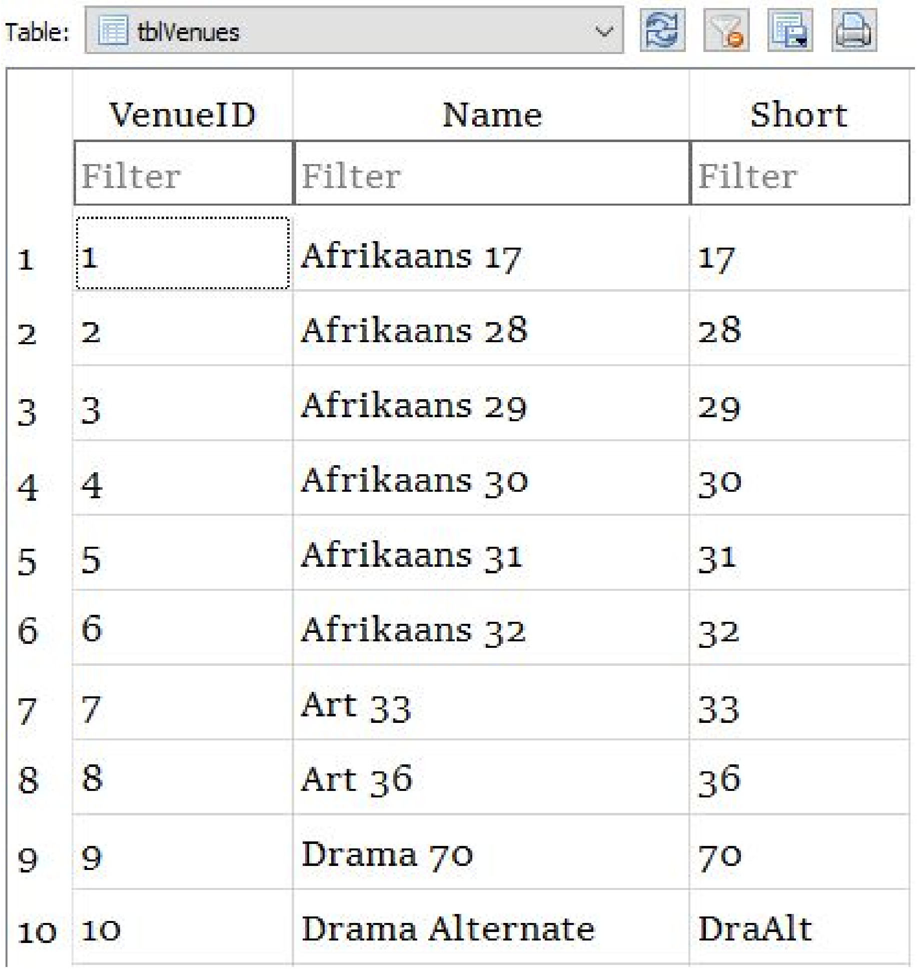


Table: tblVenues

	VenueID	Name	Short
1	1	Afrikaans 17	17
2	2	Afrikaans 28	28
3	3	Afrikaans 29	29
4	4	Afrikaans 30	30
5	5	Afrikaans 31	31
6	6	Afrikaans 32	32
7	7	Art 33	33
8	8	Art 36	36
9	9	Drama 70	70
10	10	Drama Alternate	DraAlt

Explanation of Storage Design:

Brief overview:

The application uses a mix of connections with the database as well as JSON files. The one JSON file contains data of venues and the other one which dynamically changes on the users request contains data of which teacher is in what venue at what time. This file is written on the user's request and is populated from an integrated query in a java class which then extracts the data into the respective JSON file.

I have used sqlite as it is relatively lightweight relational database and there are various accessible open source applications for handling the data. The GUI I used for handling and creating the data is called 'DB browser for sqlite'. The resources needed and the libraries required for handling the sqlite data in java are minimal.

The various fields within my database are all separated into their respective tables. Each table containing data that is related. The database allows for effective querying within my program as each field is uniquely identifiable and effectively organised. Using a database also allows for the booking of venues, the input of bookings can be easily managed through the booking system's link to the database. The validity of the bookings can also be efficiently checked with relevant SQL queries.

Databases in my application provide a reliable system for managing my data and are constructive in producing dynamic JSON files as output as per the users request. The use of JSON data allows my map to change dynamically according to the data read from the JSON file. The data is in such a format as it is the most functional way to structure my data to work coherently with the way my map is built/configured.

Advantages of using a database:

- Easy to update data values, manually as well as automatically.
- Validation of data can be implemented to make sure data input is valid.
- Large amounts of data can be stored and managed/accessed efficiently.
- Maintaining a database is simple and saves time for the user.
- The security of the database restricts access and accessibility to only certain users by assigning security groups and by encrypting the database.
- Connection between the front-end and back-end is secure fast and reliable, through using the correct libraries and connection properties.

Advantages of using JSON file/format:

- Commonly used data format for asynchronous browser-server communication.
- Language independent data format.
- Derived from JavaScript, but allows other programming languages such as Java to generate and parse JSON formatted data.
- Lightweight when compared to other formats such as previously used XML.