ST JOHN'S COLLEGE

Technical Documentation
&
Testing Document

IT PAT, St John's College 2019

Technical & Testing Document

Gordon Fleming

Contents:

Technical Documentation:

Externally sourced code:

Located in class:	What does the code do:	Link:
JsonServices.java	This code converts data retrieved from my database and converts it into a workable JSON format.	shorturl.at/BIT08
Main.java → getJsonObject()	This method outputs and returns the JSON data once it has established a connection to the database.	
Map.js	This javascript was edited from the externally sourced code to suite my conditions. This code displays the relevant JSON data above each of their respective venues.	shorturl.at/cinJL
Google authentication API	Google's API allows me to restrict access to only St John's College members, done through the integration of their javascript in my code.	shorturl.at/bgqM1

Explanation of Critical Algorithms:

JsonServices-

Description:

- ➔ Fetches data from the results of a SQL query
- ➔ Converts this data from the database
- ➔ Writes this data to the respective JSON file
- ➔ JSON file can then be read by the javascript
- ➔ Javascript then displays the data on the map

Why it is critical-

The java class, JsonServices is critical to enabling me to convert data from my SQlite database to JSON data for my SVG map to read. This program runs through each record in my relevant database table and converts it into useful JSON data, which is then interpreted by my javascript code. This links my map to the data contained in my database, it is run whenever the refresh button is toggled.

```
public class JsonServices {
public static List<JSONObject>getFormattedResult(ResultSet rs) throws SQLException{
    List<JSONObject> resList = new ArrayList<JSONObject>();
    try{
      //gets all the column names
      ResultSetMetaData rsMeta = rs.getMetaData();
      //uses the connected JSON library, to be able to it's pre-constructed methods
      int columnCnt = rsMeta.getColumnCount();
      List<String> columnNames = new ArrayList <String>();
      //loops to get all column names
      for(int i=1;i<=columnCnt;i++){
        //adds all retrieved column names to List object
        columnNames.add(rsMeta.getColumnName(i).toUpperCase());
      }
      while(rs.next()){
        //convert each object to a readable JSON object
        JSONObject obj = new JSONObject();
        for(int i=1;i<=columnCnt;i++){
          String key = columnNames.get(i-1);
          String value = rs.getString(i);
          obj.put(key, value);
        }
        resList.add(obj);
      }
    }catch(Exception ex){
      ex.printStackTrace();
    }finally{
      try{
        rs.close();
      }catch(SQLException ex){
        ex.printStackTrace();
```

```
        }
      }
      return resList;
   }
}
```

BookingControl-

Description:

- ➔ Fetches the booking request and checks if all entered data is valid
- ➔ Establishes a connection to the database
- ➔ Assigns each booking field a variable and places each variable into a relevant insert SQL statement
- ➔ The data is then inserted into the database, booking table
- ➔ Booking table is then read and analysed by the map

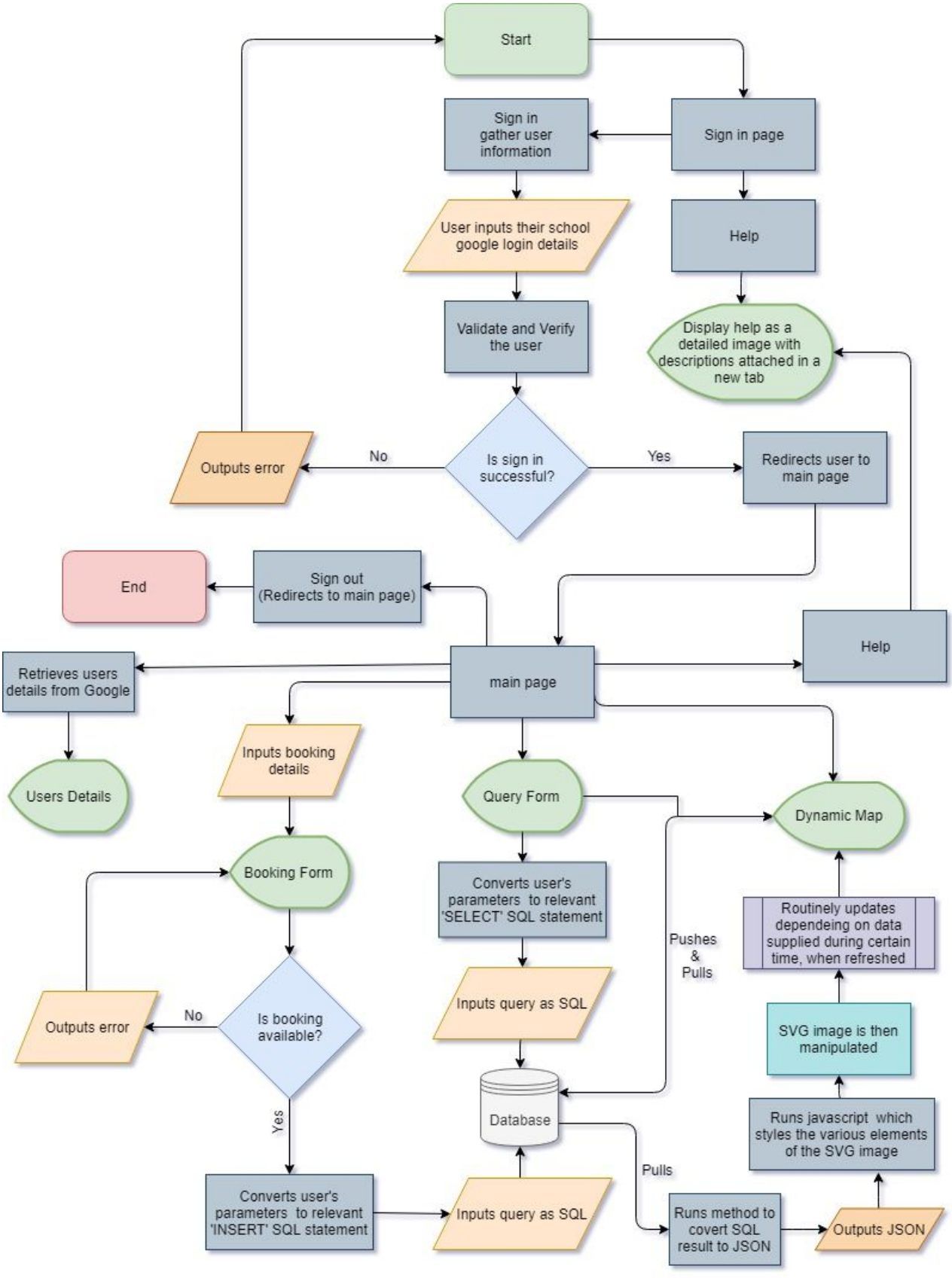
Why it is critical-

The class BookingControl handles all the user's booking requests inputted through the booking form. The java connects the form to the database from the front-end HTML. The form also handles all error responses, if the data is invalid a relevant error message will be displayed to the user. Once the code establishes all data inputted in the form is valid it inserts all the data into the relevant table. This is done through placing each field into a SQL 'insert' statement. The class is vital in allowing the application to handle both permanent data from the database as well as user requested data.

```
public class BookingControl extends HttpServlet { //Servlet for managing booking
requests from the user

 @Override
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

//Receives values and processes the request from the server

  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  //assigns each variable with a fetch value of the same type
  String venue = request.getParameter("venue");
```

```
  String activity = request.getParameter("activity");
  String week = request.getParameter("week");
  String day = request.getParameter("day");
  String period = request.getParameter("period");
  String staffName = request.getParameter("staffName");
  String date = request.getParameter("date");

  //validates the user's given input and gives relevant error message if invalid

  if (venue.isEmpty() || venue.equals("-1") || activity.isEmpty() || activity.equals("-1") ||
week.isEmpty() || day.isEmpty() || period.isEmpty() || staffName.isEmpty() ||
date.isEmpty()) { //Checks for certain invalid values
   RequestDispatcher rd = request.getRequestDispatcher("main.jsp");

   //Outputs error message if invalid data was inputted

   out.print("<style>.alert
{padding-right:10px;padding-left:10px;padding-top:5px;padding-bottom:5px;width:600px
;background-color: #f44336;/* Red */color: white;margin-bottom: 15px;bottom:
0;position: absolute;left:50%;transform: translate(-50%); }"
          + ".closebtn { margin-left: 15px;color: white;font-weight: bold;float: right;font-size:
22px;line-height: 20px;cursor: pointer;transition: 0.3s;}"
          + ".closebtn:hover { color: black;}</style>"
          + "<script>function redirect() {window.location =
\"http://localhost:8080/See-Everything/main.jsp\";}</script>"
          + "<div class=\"alert\"><span class=\"closebtn\"
onclick=\"this.parentElement.style.display='none';redirect();\">&times;</span><strong>E
rror!</strong> The booking was unsuccessful, make sure there are no blanks or invalid
inputs.</div>");
   rd.include(request, response);
  } else {
   try {
    DbConnection dbconn=new DbConnection();                    //Establishes the
database connection
      try (Connection myconnection = dbconn.Connection()) {
         String query = "INSERT INTO tblBooking VALUES (?,?,?,?,?,?,?)";      //Inserts
the users booking with relevant SQL statement
```

```
        try (PreparedStatement ps = myconnection.prepareStatement(query)){
//Generates the sql query
            ps.setString(1, venue);
            ps.setString(2, activity);
            ps.setString(3, week);
            ps.setString(4, day);
            ps.setString(5, period);
            ps.setString(6, staffName);
            ps.setString(7, date);

            ps.executeUpdate();

// executes the statement in the database

            ps.close();
            myconnection.close();
        }
        RequestDispatcher rd = request.getRequestDispatcher("main.jsp");

        //Outputs success message if valid data was inputted

        out.print("<style>.success
{padding-right:5px;padding-left:10px;padding-top:5px;padding-bottom:10px;width:600px
;background-color: green; color: white;margin-bottom: 15px;bottom: 0;position:
absolute;left:50%;transform: translate(-50%); }"
         + ".closebtn { margin-left: 15px;color: white;font-weight: bold;float: right;font-size:
22px;line-height: 20px;cursor: pointer;transition: 0.3s;}"
         + ".closebtn:hover { color: black;}</style>"
         + "<div class=\"success\"><span class=\"closebtn\"
onclick=\"this.parentElement.style.display='none';\">&times;</span><strong>Success!</
strong> Your booking was submitted to the server.</div>");
        rd.include(request, response);

}
  } catch (SQLException ex){
      System.out.println("Error.");
          }
      }
```



*Flow chart helping explain critical algorithms, reference is from my design document.

Advanced Techniques:

- ➔ My project is a web application that makes use of JSP pages (Java Servlet Pages) this allows me to integrate java into my HTML code, further allowing me to create advanced user requests.

- ➔ I have organised and normalised my schools data into proper database tables, which allows me to display accurate results that have relevance to what is actually happening when and where in the school. The data was all adapted from a spreadsheet to a database, this required complex SQL queries as well as complex joins.

- ➔ Everytime the map is updated, a complex SQL query is run which requests all relevant results. These results are converted to JSON data in the back-end to allow the data to be read and inputted into the map through the javascript code.

- ➔ I created an SVG map from scratch, adapted from a PDF map sourced from the school. This map allows me to address each element (Venue) depending on its assigned ID. To be able to achieve this I have had to also use javascript and JSON data, which all works in unison with the rest of my application. This allows me to address each classroom individually.

- ➔ My GUI is user friendly and has various dynamic as well as interactive elements. For example certain buttons change when the mouse is over them. My GUI is slick and the appearance is managed by a totally separate CSS page.

- ➔ The booking form displays an array of results. This is read from a connection to the database. The java embedded in the HTML reads through each record on the table and displays it to the user in a relevant drop down menu.

- ➔ My program makes use of networking and can be hosted on a server for others to access the application.

Testing Documentation:

Test	Inputted value	Normal	Extreme	Erroneous
Login Screen				
1.	Email	User enters their correct school email.	N\A	User enters their personal email
2.	Password	User enters their correct school password.	N\A	N\A
Results for the relevant scenarios				
Normal		success	Erroneous	Error message
Booking Form				
1.	Booking details	User enters all booking details.	N\A	User doesn't fill in all the fields
Normal		success	Error message	Error message
Data is selected through drop downs, radio buttons and set input menus. Defensive programming is used here.				
Query Form				
1.	Query topic and the user's parameters	User enters their staff member or venue from the drop down along with their parameters.	N\A	N\A
Normal		success		
If extreme or/and erroneous data is inputted, no error will occur, rather no response will show in the table.				

*Error message = Set response to the user's input, not an error with the program.

Screenshots of success and error messages for the booking form: