# J2SE Case Study: Job Management System.

The Job Management Service I have developed primarily provides its users with the ability to schedule when they would like to send email messages to other users, for example, a reminder email. I used Gmail’s Simple Mail Transfer Protocol (SMTP) server to send emails.

I have integrated the Quartz API to the prototype with the main aim of allowing users to schedule their emails.

Moreover, the Quartz API provides additional functionality, which I have not covered in the prototype but are areas of improvement/future works to my application, such as unscheduling a job, adding a new job, and removing a job. All these can be done without the need to re-develop the Job Management System.

I have used MySQL as my Data Source, this ensures that the email sending job and all the job-related data is persistent at all times.

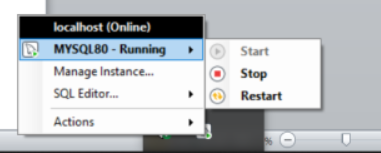
## System Requirements

Java 8+

## System Configurations

Download and install MySQL in case it is not present in your machine. Download MySQL Workbench because it is the Graphical User Interface application that you can use to access the MySQL database server.

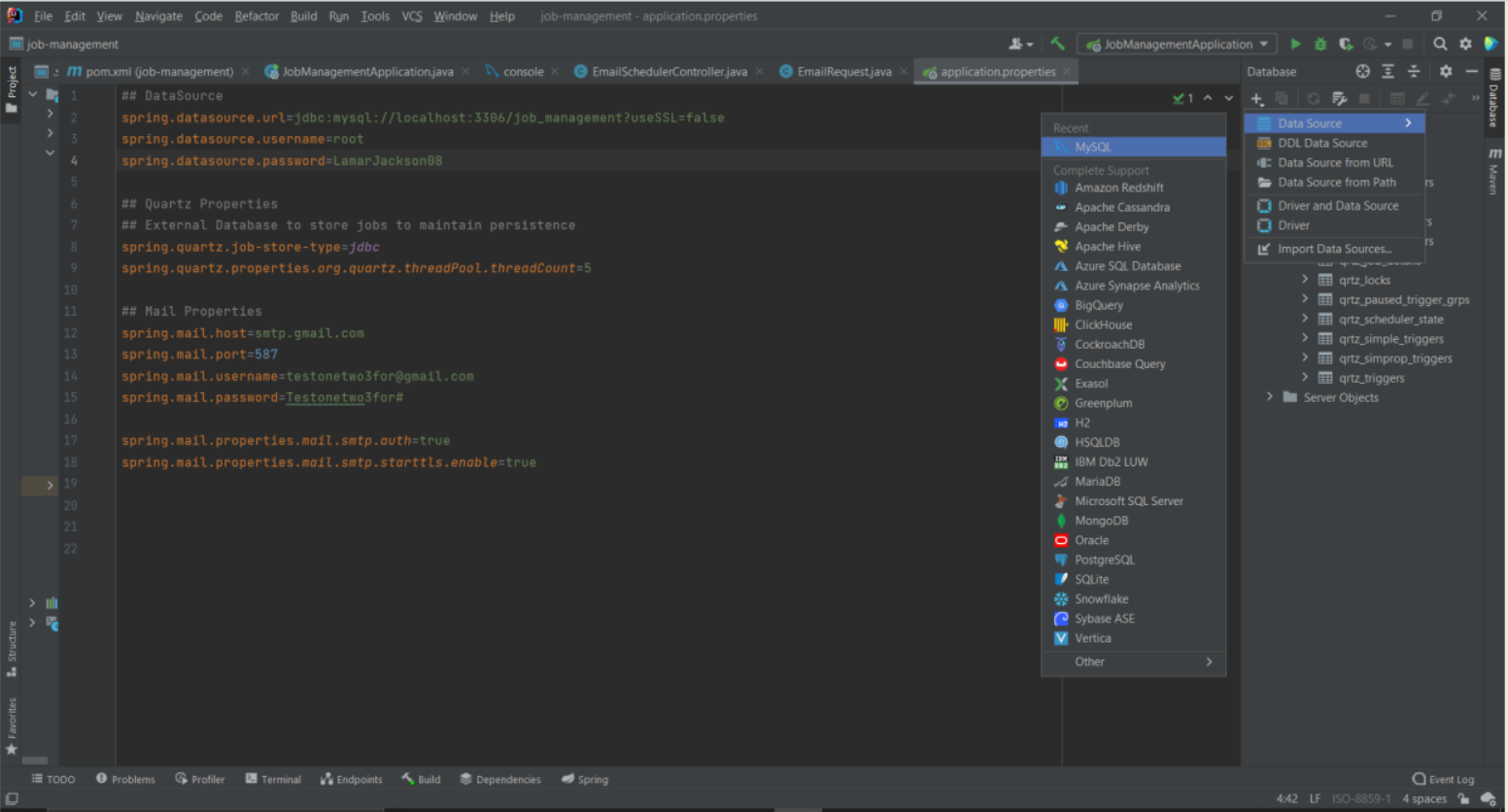
Ensure the MySQL Service is running, for my case I use MySQL Notifier to turn on and turn off the MySQL Service from my Task Bar i.e



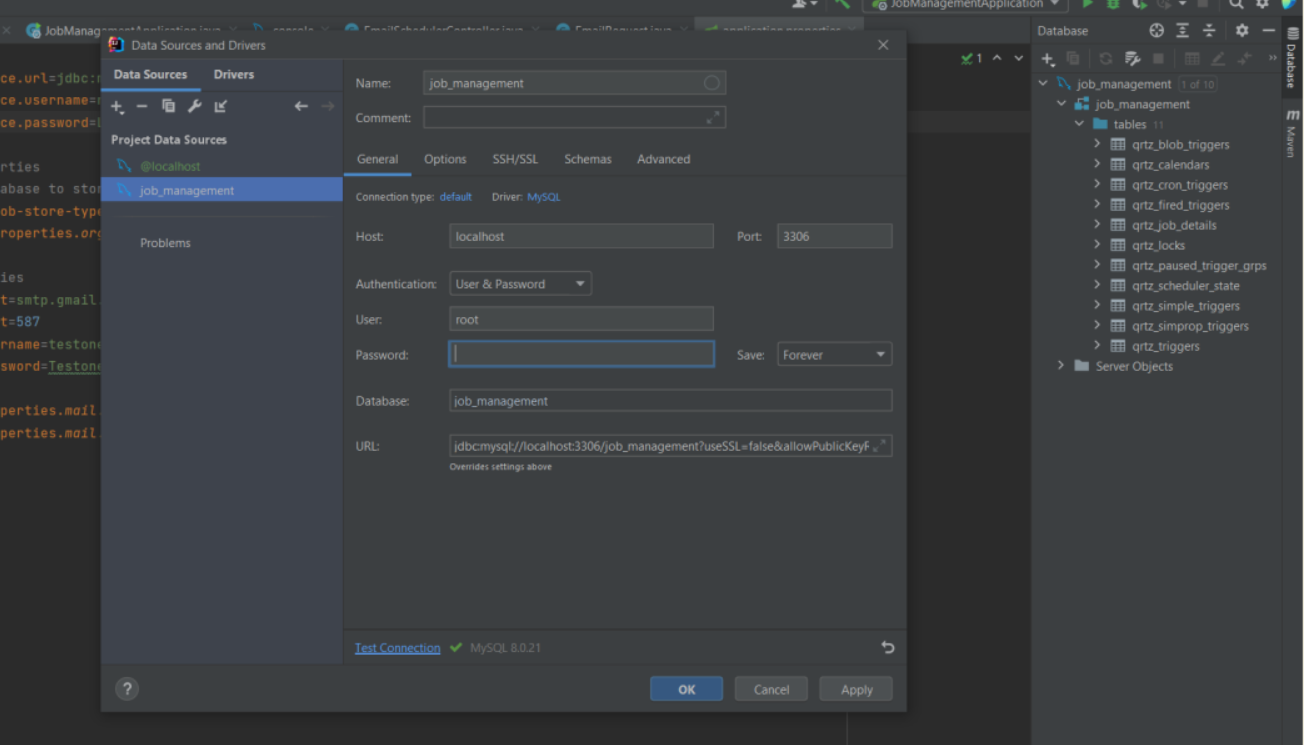
Create a database called ‘job\_management’.

Import the SQL Database that is located in the .zip file with the name ‘Dump20210810.sql’ to your MySQL Workbench under the ‘job\_management’ database.

Once the database is successfully imported, connect the prototype to the database. For my case, I used Intellij IDEA as my IDE of choice. You then click the ‘Database’ icon on the far right, then click the ‘New’ or ‘+’ icon and then select ‘Data Source’. After that click on the ‘MySQL’ option which appears in the dropdown menu. A sample of that procedure is represented in the screenshot below:



A new pop-up window will appear on the screen that will look like this:



In the Name section you can proceed to type the name of the database you imported, in my case the database is called ‘job\_management’. Under the authentication drop-down menu select the ‘User & Password’ option. The next step you then do is to insert your MySQL Database server username and password in the ‘User’ and ‘Password’ input field respectively.

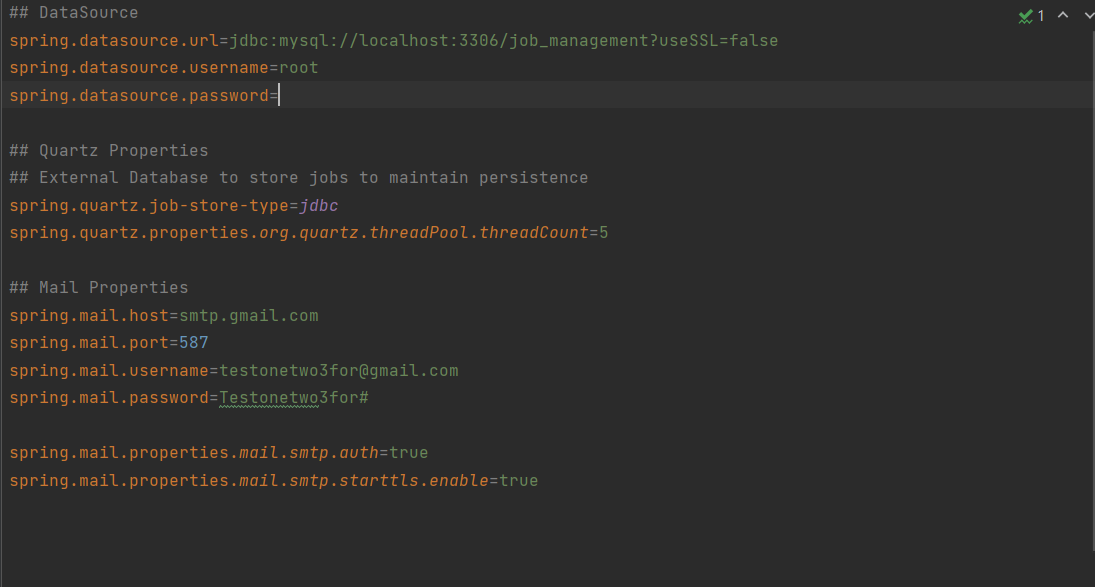
Under the URL input field section you can copy and paste this URL: jdbc:mysql://localhost:3306/*job\_management*?useSSL=false&allowPublicKeyRetrieval=true

The subdomain section of the URL which is labelled in italics is the database name. The next step is clicking the Test Connection link to ensure that there is a successful connection between the database and the IDE.

If the connection is successful then a success message will appear on the screen for a few seconds and then disappear automatically. Click the ‘Apply’ button and then the ‘Ok’ button. You should then be able to access the database and all its tables in the ‘Database section’.

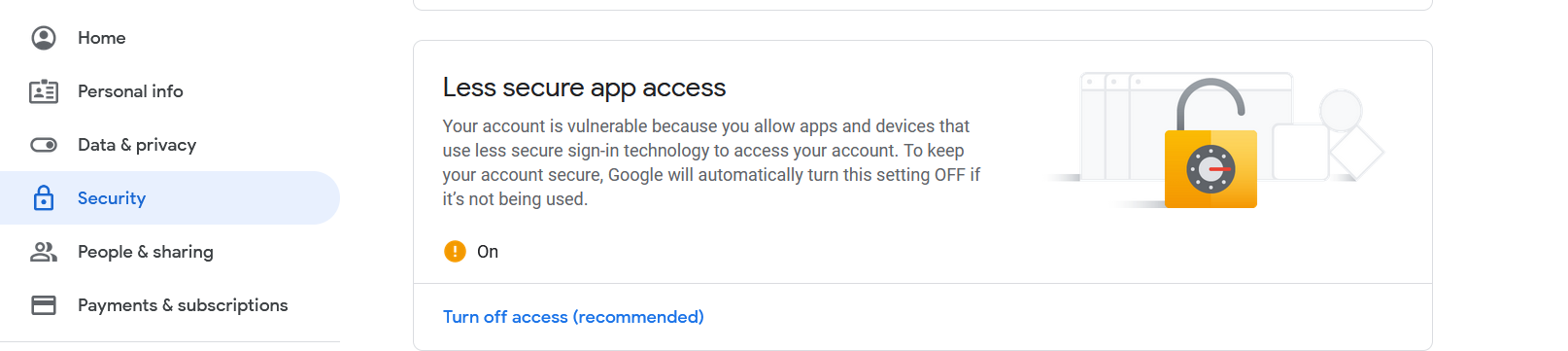
## Application Specific Configurations.

The application uses a dummy Gmail email account that I created. The email address and password are present under the ‘resources/application.properties’ file as present below:



In this section you need enter your MySQL Database server username and password in the second and third line of code respectively after the ‘##DataSource’ comment to be able to successfully compile the code. You can further change the username and password to reflect your email address and its password under the ‘## Mail Properties’. However, I would highly discourage this because you would be required to allow your Gmail account be accessed by less secure applications and devices without proper verification and validation.

Below is the where I made the changes under the Security option in the ‘Manage your Google Account’ section:



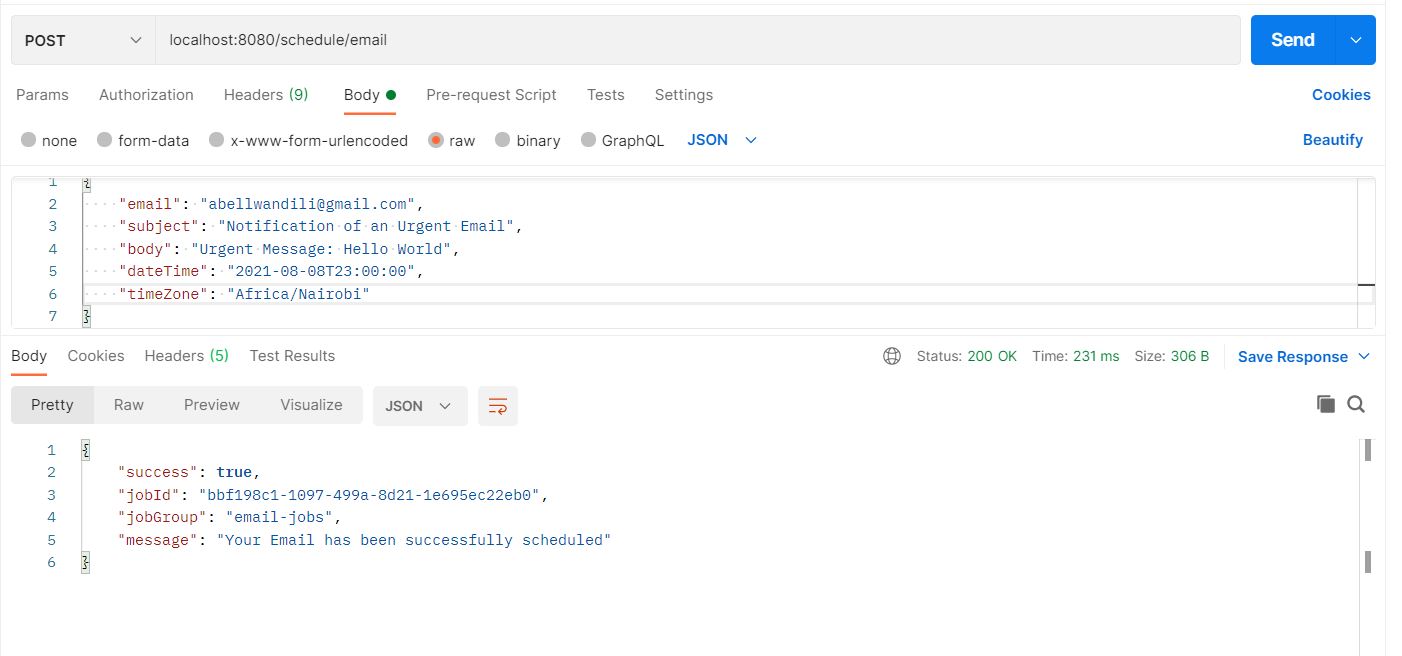
This could be potential security vulnerability not only to your account but to your personal details.

## Testing

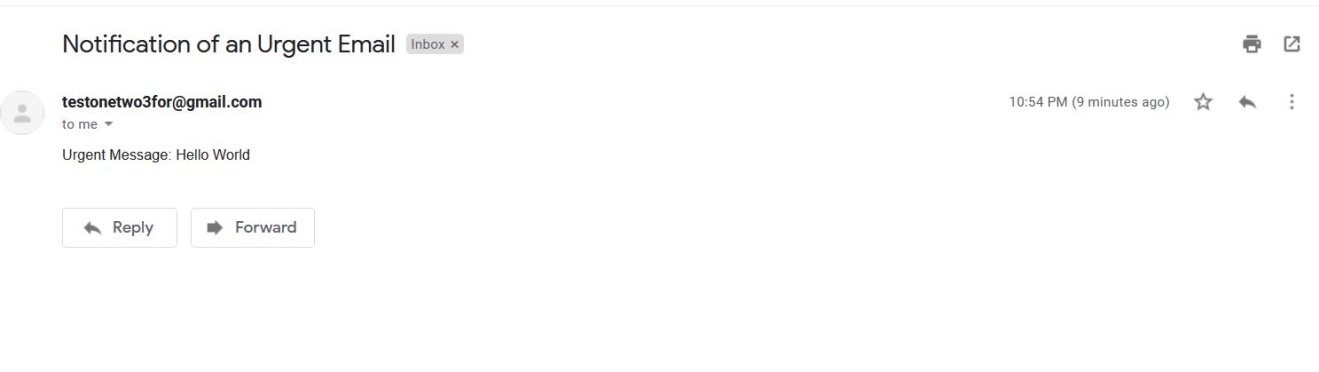
I used Postman to test my REST API and I was able to successfully schedule and send emails. Once you run the application on your IDE, you can access the REST API through this endpoint: **localhost:8080/schedule/email**

The application validates the Request Body to ensure that it has the following fields filled: the email address of the recipient, the subject of the email, the body or message in the email message, the sender’s date, time and timezone that you would like to schedule for the email to be sent to the recipient.

Below is the POST request and response on Postman:

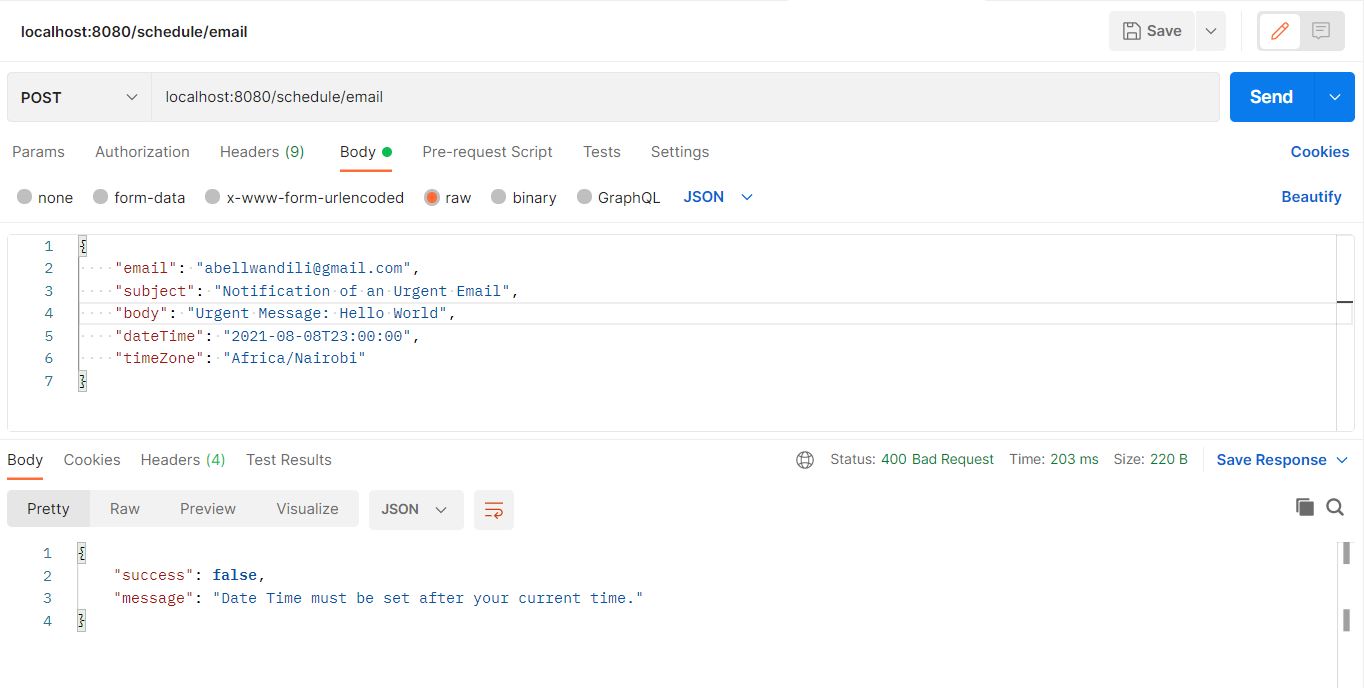


The Response on my email address was as follows:



The ‘dateTime’ format is in the order of ‘Year:Month:DateTtime’ i.e ‘2021-08-10T23:00:00’. The time zone is in the format ‘Continent/City’ i.e ‘Africa/Nairobi’.

The system also validates that the ‘dateTime’ is set to a date and time that should be in the future not. For example the sender cannot schedule an email to be sent at their current time or at a previous date from their current date i.e yesterday or last week. Below is the response error message if a user schedules to send an email on a past date or at their current time.



## Duration

I have spent 9 days working on this case study.