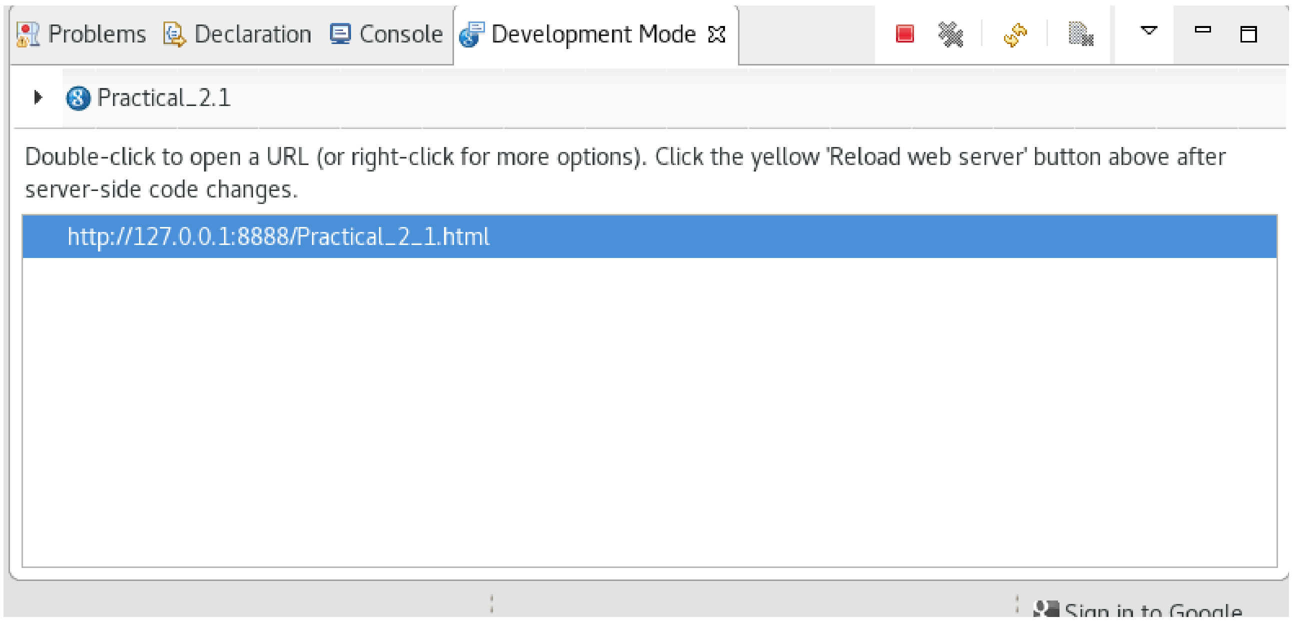
REMOVE PASSWORD AND USER NAME

Access instructions

1. Download and install: Google plugin for eclipse
https://developers.google.com/eclipse/docs/getting_started
Version that worked in labs was 4.4
2. In Eclipse File/Open projects from Filesystem and select project folder for "Import source:"
3. Change private String password = "******"; in MySQL Connection.java class, which is in com.prac.server package to your mysql password. And "private String uname = "XXXX"; to your username
4. From the dropdown of "Run As" Web application (just project name or you can choose Super Dev mode). Tested on Super Dev mode with a firend user gg50.
5. You will be presented by an url, just open it by double click.



Double-click to open a URL (or right-click for more options). Click the yellow 'Reload web server' button above after server-side code changes.

http://127.0.0.1:8888/Practical_2_1.html

Overview

My application uses asynchronous/synchronous callbacks with remote procedure calls. The two interfaces DBConnectionAsync and DBConnection specify the functionality. My server side is implemented in the MySQLConnection class, which implements the DBConnection synchronous functions. The client side is implemented in the three classes Log_in_Screen, RegistrationScreen and PageOne, which serve as RPC providers that query the DB. Inside the client classes I have my GWT GUI components, the nested classes, which implement AsyncCallback and act in response to the server. When different buttons are pressed, new instance of one of the nested classes (callback handlers) is created and remote procedure call is issued, by calling one of the methods in the interfaces, to communicate with the server and the handler listens for response. I have fulfilled all the basic requirements, which can be tested inside the main page (PageOne). Moreover, I have two other pages for login and registration, which also are fully functional. SHARED package is not used.

Code Summary
Inspiration for the overall code structure was drawn from [1]. However, everything required complete modification, apart from the structure and general idea. I have used the other sources in the references to help me tackle with minor, concrete problems. Part of the server side implementation, which involves prepared statements and database connectivity was drawn from the lectures.

Database Modifications
I implemented log in and register extensions so the difference was that I included a password column in the customer table used for logging in.

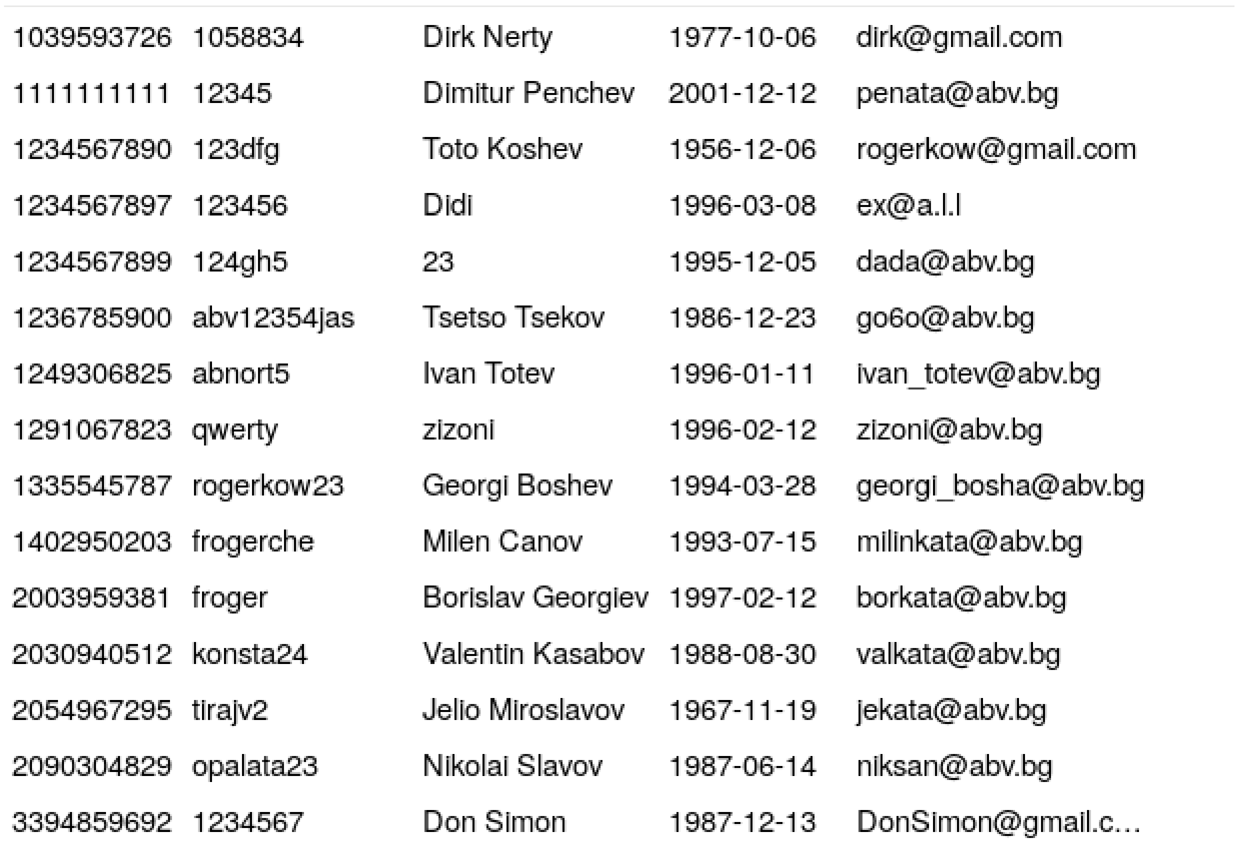
Interface Design, Implementation & Testing
Interface Design
My interface design is fairly simple, because I focus on the functionality. However, I have an initial log in page, from which the customer can either log in or register if not registered in the system. The register page has places to fill information and register and back buttons. To navigate through pages. When you log in you can access the database through the buttons if you desire or purchase a book. I also have a "Back" button from this page to go back to the login screen and log in as another user.
Implementation
My gui implementation is simple. I use labels for info, text boxes for user input and buttons for deployment of user requests. I lay out the structure of the pages nicely by Having a vertical panel in each page and in it I put horizontal panels in which the other widgets are stored. For displaying the database results I have a result horizontal panel in which with two nested for loops I put vertical panels for each column of data to lay out things nicely in the printResult() function. After a new rpc is issued and new data is fetched from db I clear this result panel for the new results. In my PageOne I have a couple of interesting features, one of which is a drop down list with all authors to allow the user to easily pick from which author they want results from. Another helpful feature is the back button. For finding the most popular books I decided to print all books sorted on the number of purchases in descending order, simply because the database does not have a lot of books at the moment and I want to show to the user. That can be changed when there is big number of books. When making a purchase, I have chosen to take a book ISBN, written by the user, because they can easily see the ISBN of books when viewing them with the "View all books" functionality. I decided to only register one phone when the user first registers in the system. In future extension of the system the customer would be able to access their profile and add additional phones. I have also added background text into my text boxes to help the user navigate in the UI. I decided to implement namely log in and register because this seemed the most important of the extensions and is crucial for a functioning on-line book streaming service.

Testing
For testing I have performed multiple tests using the GUI and looking through normal cases as well as corner cases, such as keeping a consistent database, dealing with inappropriate values, corner cases etc. I will show some of the tested functionality that modifies the database below.

User Registration test

DB Before

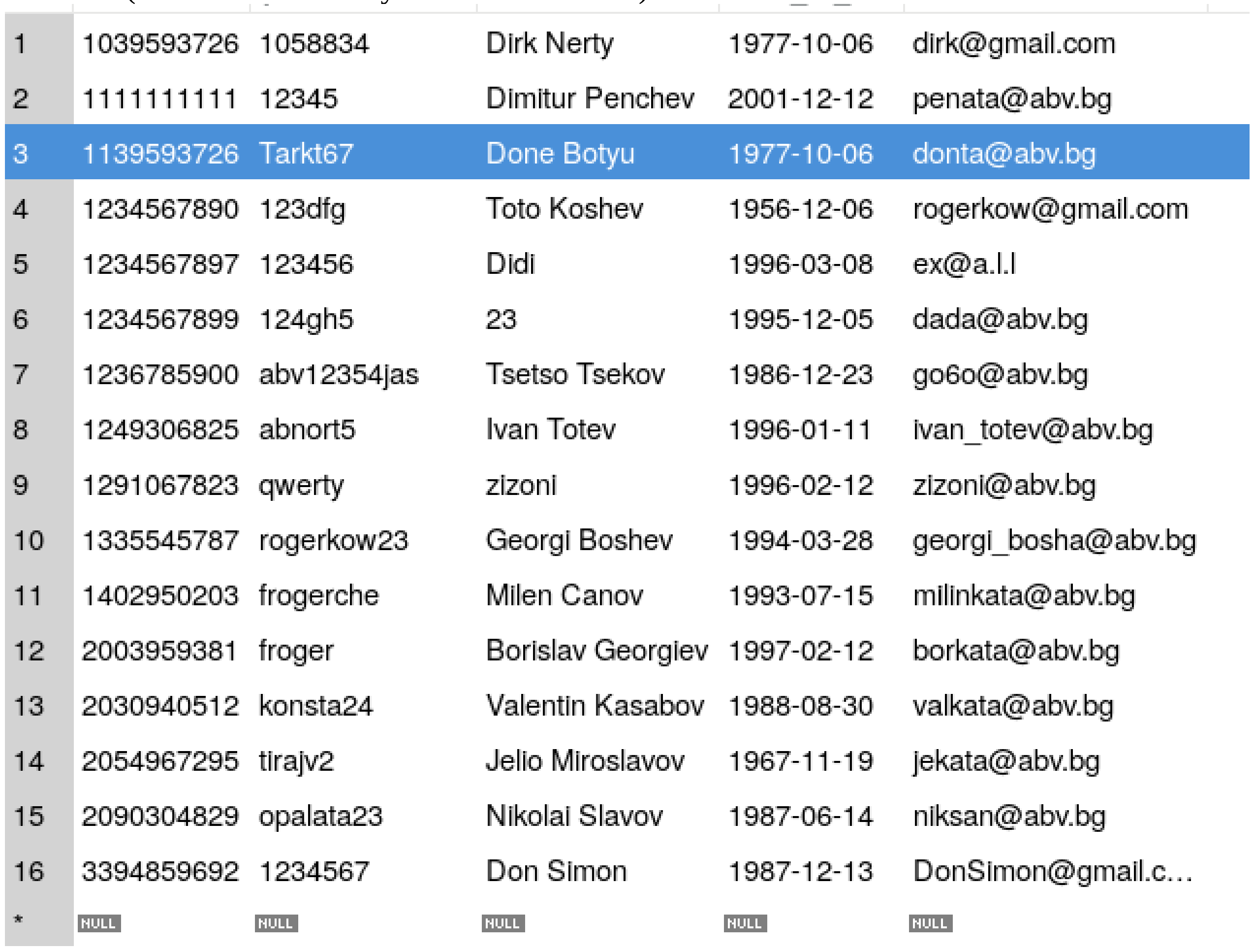
Customer



1039593726	1058834	Dirk Nerty	1977-10-06	dirk@gmail.com
1111111111	12345	Dimitur Penchev	2001-12-12	penata@abv.bg
1234567890	123dfg	Toto Koshev	1956-12-06	rogerkow@gmail.com
1234567897	123456	Didi	1996-03-08	ex@a.l.l
1234567899	124gh5	23	1995-12-05	dada@abv.bg
1236785900	abv12354jas	Tsetso Tsekov	1986-12-23	go6o@abv.bg
1249306825	abnort5	Ivan Totev	1996-01-11	ivan_totev@abv.bg
1291067823	qwerty	zizoni	1996-02-12	zizoni@abv.bg
1335545787	rogerkow23	Georgi Boshev	1994-03-28	georgi_bosha@abv.bg
1402950203	frogerche	Milen Canov	1993-07-15	milinkata@abv.bg
2003959381	froger	Borislav Georgiev	1997-02-12	borkata@abv.bg
2030940512	konsta24	Valentin Kasabov	1988-08-30	valkata@abv.bg
2054967295	tirajv2	Jelio Miroslavov	1967-11-19	jekata@abv.bg
2090304829	opalata23	Nikolai Slavov	1987-06-14	niksan@abv.bg
3394859692	1234567	Don Simon	1987-12-13	DonSimon@gmail.c…

DB after

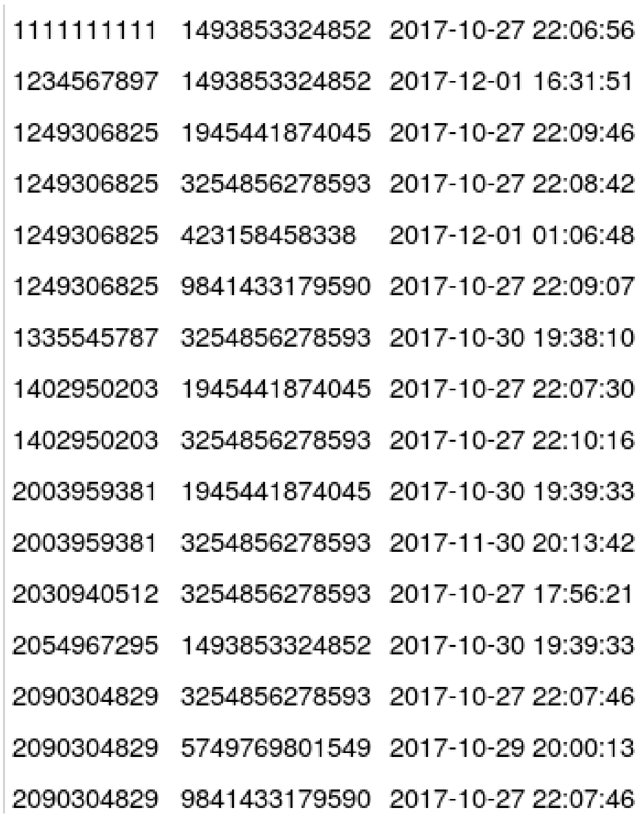
Customer (selected “Done Botyu” is new customer)



1	1039593726	1058834	Dirk Nerty	1977-10-06	dirk@gmail.com
2	1111111111	12345	Dimitur Penchev	2001-12-12	penata@abv.bg
3	1139593726	Tarkt67	Done Botyu	1977-10-06	donta@abv.bg
4	1234567890	123dfg	Toto Koshev	1956-12-06	rogerkow@gmail.com
5	1234567897	123456	Didi	1996-03-08	ex@a.l.l
6	1234567899	124gh5	23	1995-12-05	dada@abv.bg
7	1236785900	abv12354jas	Tsetso Tsekov	1986-12-23	go6o@abv.bg
8	1249306825	abnort5	Ivan Totev	1996-01-11	ivan_totev@abv.bg
9	1291067823	qwerty	zizoni	1996-02-12	zizoni@abv.bg
10	1335545787	rogerkow23	Georgi Boshev	1994-03-28	georgi_bosha@abv.bg
11	1402950203	frogerche	Milen Canov	1993-07-15	milinkata@abv.bg
12	2003959381	froger	Borislav Georgiev	1997-02-12	borkata@abv.bg
13	2030940512	konsta24	Valentin Kasabov	1988-08-30	valkata@abv.bg
14	2054967295	tirajv2	Jelio Miroslavov	1967-11-19	jekata@abv.bg
15	2090304829	opalata23	Nikolai Slavov	1987-06-14	niksan@abv.bg
16	3394859692	1234567	Don Simon	1987-12-13	DonSimon@gmail.c…
*	NULL	NULL	NULL	NULL	NULL

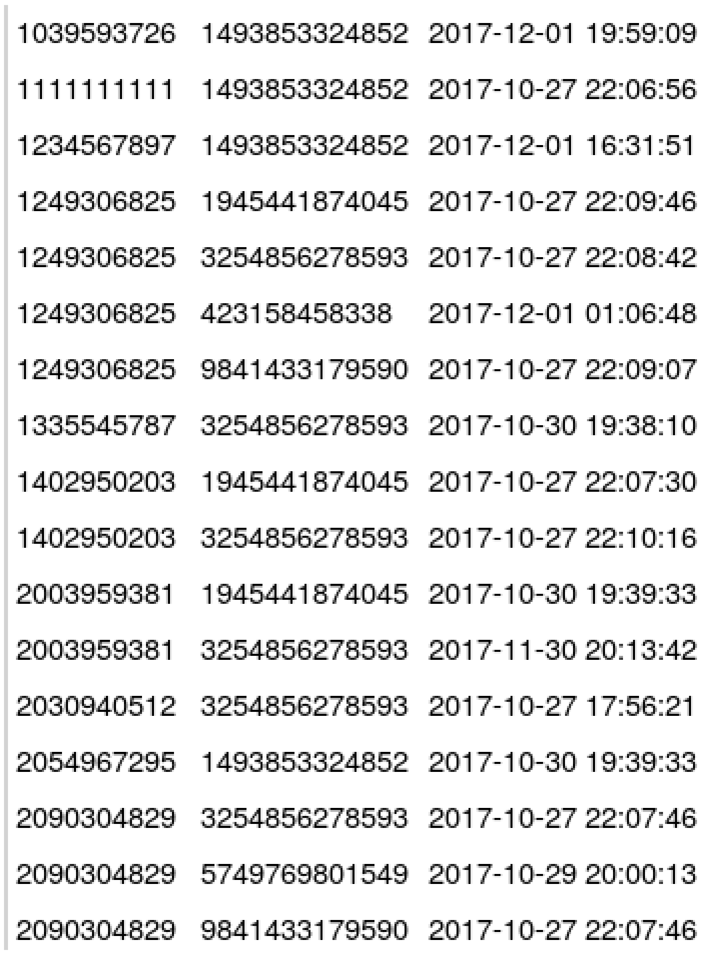
Book Purchase

Before



```
1111111111   1493853324852   2017-10-27 22:06:56
1234567897   1493853324852   2017-12-01 16:31:51
1249306825   1945441874045   2017-10-27 22:09:46
1249306825   3254856278593   2017-10-27 22:08:42
1249306825   423158458338    2017-12-01 01:06:48
1249306825   9841433179590   2017-10-27 22:09:07
1335545787   3254856278593   2017-10-30 19:38:10
1402950203   1945441874045   2017-10-27 22:07:30
1402950203   3254856278593   2017-10-27 22:10:16
2003959381   1945441874045   2017-10-30 19:39:33
2003959381   3254856278593   2017-11-30 20:13:42
2030940512   3254856278593   2017-10-27 17:56:21
2054967295   1493853324852   2017-10-30 19:39:33
2090304829   3254856278593   2017-10-27 22:07:46
2090304829   5749769801549   2017-10-29 20:00:13
2090304829   9841433179590   2017-10-27 22:07:46
```

After



```
1039593726   1493853324852   2017-12-01 19:59:09
1111111111   1493853324852   2017-10-27 22:06:56
1234567897   1493853324852   2017-12-01 16:31:51
1249306825   1945441874045   2017-10-27 22:09:46
1249306825   3254856278593   2017-10-27 22:08:42
1249306825   423158458338    2017-12-01 01:06:48
1249306825   9841433179590   2017-10-27 22:09:07
1335545787   3254856278593   2017-10-30 19:38:10
1402950203   1945441874045   2017-10-27 22:07:30
1402950203   3254856278593   2017-10-27 22:10:16
2003959381   1945441874045   2017-10-30 19:39:33
2003959381   3254856278593   2017-11-30 20:13:42
2030940512   3254856278593   2017-10-27 17:56:21
2054967295   1493853324852   2017-10-30 19:39:33
2090304829   3254856278593   2017-10-27 22:07:46
2090304829   5749769801549   2017-10-29 20:00:13
2090304829   9841433179590   2017-10-27 22:07:46
```

Database Operations

Some of the functions have similar structure so I have tried to modularise everything as much as I can but I have tried to also try/catch aggressively so I can debug easier and see when potential problems are coming from if needed.

getAllBooks - simply constructs a query to get all books from database and iterate over the result set to put the metadata and data in a 2D string list to return to client.

GetBooksByAuthor – Constructs query to select all books by a particular author and again flushes everything into a 2D String list which is returned to the handler.

GetReviewsOfBook – constructs a query to get all reviews of a particular book, queries the database and puts results in a 2D list.

GetPopBooks – queries database to receive books sorted by number of purchases.

PurchBook - purchases a book for the client (session holder) and update database. I keep track of who is the owner of the current session from when the customer logs in by having a field for the user ID in PageOne.java.

GetAuthors – gets the names of all authors from database to put them in a drop down list inside PageOne for user aleviation.

Authenticate – Queries the database to check for user login details. Here I want to mention that using Strings like booleans is not the best choice but that was how I started in this implementation and I wanted to keep it consistent.

Register – This function is a bit long so it can be modularised in the future. It basically adds strings to a list to act as booleans for every piece of data (e.g. Id, password). Then applies appropriate checks to see if data is correct and Id, email and phone are not already taken. I could have written more helpful messages to the customer make the user interface more user-friendly but given more time I would look into that. At the end of the method I have to change two inter-connected tables so I disable auto commit and flush them right one after another, then enable it again.

Note: there may be an overflow if the 10 digits ID is too big, so given more time, I would look into that and maybe change its type in the database.

Evaluation & Conclusion

I think that I have made a reasonable attempt at the practical using GWT in java. I have achieved the basic specifications as well as extended the practical execrise to look closer to a real working online service. I think that this practical and the module so far as a whole helped me a lot in my understanding of databases and will have positive effect on the CS3099 because I am doing back end.

References

https://tenbergen.files.wordpress.com/2016/06/db-access-in-gwt-the-missing-tutorial.pdf [1]
https://stackoverflow.com/questions/5175728/how-to-get-the-current-date-time-in-java [2]
http://www.gwtproject.org/doc/latest/RefWidgetGallery.html [3]
https://stackoverflow.com/questions/10575624/java-string-see-if-a-string-contains-only-numbers-and-not-letters [4]

https://stackoverflow.com/questions/8204680/java-regex-email [5]
https://stackoverflow.com/questions/30507579/how-to-check-correct-date-format-pattern [6]
https://stackoverflow.com/questions/21054744/java-validate-that-a-given-string-represents-a-date-i-e-yyyy-mm-dd [7]