12/30/2021

Luis Davila

The University of texas at el paso | Brain Computation lab

Setting up a remote PostgreSQL database

And The Serendipity App

Contents

[Remote Database Folder (What’s included?) 8](#_Toc98248176)

[1.) App Deployment Folder (What’s included?) 8](#_Toc98248177)

[a. AdvanceSearchWindow (.mlapp file) 8](#_Toc98248178)

[b. HardCodeTable (.xlsx file) 8](#_Toc98248179)

[c. InfoWeWant (.xlsx file) 8](#_Toc98248180)

[d. Main (.mlapp file) 8](#_Toc98248181)

[e. postgresql-42.3.1 (.jar file) 8](#_Toc98248182)

[f. Start (.mlapp file) 8](#_Toc98248183)

[IMPORTANT!!!! 8](#_Toc98248184)

[2.) backupProtocal (.bat file) 8](#_Toc98248185)

[3.) createDummyTable (.m file) 9](#_Toc98248186)

[4.) createLiveTable (.m file) 9](#_Toc98248187)

[5.) createRatTable (.m file) 9](#_Toc98248188)

[6.) displayTables (.m file) 9](#_Toc98248189)

[Introduction 9](#_Toc98248190)

[Things to Download Before Beginning 9](#_Toc98248191)

[Configuring the Firewall 11](#_Toc98248192)

[Configuring PostgreSQL for Remote Access 1](#_Toc98248193)

[Creating A Server and Database 6](#_Toc98248194)

[What’s the recommended password 7](#_Toc98248195)

[recommended server name 10](#_Toc98248196)

[Recommended Database Name 15](#_Toc98248197)

[Creating a Database Connection 17](#_Toc98248198)

[What should I put in my fields? 20](#_Toc98248199)

[Name 20](#_Toc98248200)

[Vendor 20](#_Toc98248201)

[Driver Location 20](#_Toc98248202)

[Database 20](#_Toc98248203)

[Server 20](#_Toc98248204)

[Port Number 21](#_Toc98248205)

[Creating the Live Table In your database (createLiveTable.m) 21](#_Toc98248206)

[Creating Rat Table (createRatDatabase.m) 1](#_Toc98248207)

[Creating a Dummy Table (createDummyTable.m) 1](#_Toc98248208)

[Look at Tables(displayTables.m) 1](#_Toc98248209)

[Establishing Backups (backupProtocal.bat) 2](#_Toc98248210)

[Pro tip 2](#_Toc98248211)

[The Serendipity App 9](#_Toc98248212)

[Start.mlapp 9](#_Toc98248213)

[Callbacks 9](#_Toc98248214)

[SetDriverPathButtonPushed 9](#_Toc98248215)

[SetIPAddressButtonPushed 9](#_Toc98248216)

[LoginButtonPushed 10](#_Toc98248217)

[Properties 10](#_Toc98248218)

[app.IPAddressSet 10](#_Toc98248219)

[app.driverPathSet 10](#_Toc98248220)

[app.driverPath 10](#_Toc98248221)

[Main.mlapp 10](#_Toc98248222)

[CallBacks 10](#_Toc98248223)

[startupFcn 10](#_Toc98248224)

[SearchButtonPushed 10](#_Toc98248225)

[SelectNodulusFileButtonPushed 11](#_Toc98248226)

[SelectRatDropDownValueChanged 11](#_Toc98248227)

[SelectRatDropDown\_2ValueChanged 11](#_Toc98248228)

[SelectRatDropDown\_3ValueChanged 11](#_Toc98248229)

[SelectRatDropDown\_4ValueChanged 11](#_Toc98248230)

[AddDataToDatabaseButtonPushed 12](#_Toc98248231)

[SetTodaysDateButtonPushed 12](#_Toc98248232)

[Port1DropDownValueChanged 12](#_Toc98248233)

[Port2DropDownValueChanged 12](#_Toc98248234)

[Port3DropDownValueChanged 12](#_Toc98248235)

[Port4DropDownValueChanged 13](#_Toc98248236)

[SeeUploadHistoryButtonPushed 13](#_Toc98248237)

[Functions 13](#_Toc98248238)

[results(app) 13](#_Toc98248239)

[organize(app) 13](#_Toc98248240)

[organize2(app) 14](#_Toc98248241)

[orgranize3(app) 14](#_Toc98248242)

[organize4(app) 14](#_Toc98248243)

[checkNamesInAppMatchNamesInFile(app) 14](#_Toc98248244)

[createDatabaseConnection(app) 14](#_Toc98248245)

[Properties 14](#_Toc98248246)

[app.rat1 14](#_Toc98248247)

[app.rat2 14](#_Toc98248248)

[app.rat3 15](#_Toc98248249)

[app.rat4 15](#_Toc98248250)

[app.actual\_table 15](#_Toc98248251)

[app.rattable 15](#_Toc98248252)

[app.password 15](#_Toc98248253)

[app.datasource 15](#_Toc98248254)

[app.username 16](#_Toc98248255)

[app.IPaddress 16](#_Toc98248256)

[app.jarFilePath 16](#_Toc98248257)

[app.files 16](#_Toc98248258)

[app.workingDirectory 16](#_Toc98248259)

[app.info 16](#_Toc98248260)

[app.universalCounter 16](#_Toc98248261)

[app.variableList 16](#_Toc98248262)

[app.universalCurrentNodulusFile 17](#_Toc98248263)

[app.universalVariablesWeWant 17](#_Toc98248264)

[app.rat\_1\_timestamps 17](#_Toc98248265)

[app.rat\_1\_x\_coordinates 17](#_Toc98248266)

[app.rat\_1\_y\_coordinates 17](#_Toc98248267)

[app.rat\_2\_timestamps 17](#_Toc98248268)

[app.rat\_2\_x\_coordinates 17](#_Toc98248269)

[app.rat\_2\_y\_coordinates 17](#_Toc98248270)

[app.rat\_3\_timestamps 17](#_Toc98248271)

[app.rat\_3\_x\_coordinates 18](#_Toc98248272)

[app.rat\_3\_y\_coordinates 18](#_Toc98248273)

[app.rat\_4\_timestamps 18](#_Toc98248274)

[app.rat\_4\_x\_coordinates 18](#_Toc98248275)

[app.rat\_4\_y\_coordinates 18](#_Toc98248276)

[app.driverPath 18](#_Toc98248277)

[app.IPAddress 18](#_Toc98248278)

[app.replaceALLDuplicates 18](#_Toc98248279)

[app.namesFoundUniversal 18](#_Toc98248280)

[app.universalContainer 19](#_Toc98248281)

[AdvanceSearchWindow.mlapp 19](#_Toc98248282)

[Callbacks 19](#_Toc98248283)

[startupFcn 19](#_Toc98248284)

[SearchButtonPushed 19](#_Toc98248285)

[UITableCellSelection 19](#_Toc98248286)

[DeleteSelectedRowButtonPushed 19](#_Toc98248287)

[ExportRowToFileButtonPushed 19](#_Toc98248288)

[UIFigureCloseRequest 19](#_Toc98248289)

[ExportSearchResultToFileButtonPushed 20](#_Toc98248290)

[ImportAndGraphResultsButtonPushed 20](#_Toc98248291)

[AddMoreResultsToFileButtonPushed 20](#_Toc98248292)

[GraphTheseSearchResultsButtonPushed 20](#_Toc98248293)

[Properties 20](#_Toc98248294)

[app.queryResult 20](#_Toc98248295)

[app.index 20](#_Toc98248296)

[app.exportRow 20](#_Toc98248297)

[app.firstOrNot 20](#_Toc98248298)

[app.databaseConnection 21](#_Toc98248299)

[app.username 21](#_Toc98248300)

[app.password 21](#_Toc98248301)

[app.actualTable 21](#_Toc98248302)

[app.queryParts 21](#_Toc98248303)

[app.exportFile 21](#_Toc98248304)

# Restoring A Database from backup

Assuming you have downloaded the backup database from Harvard Dataverse from the following link please follow the following steps to restore it. In case you have not yet downloaded the file it can be found in the link below inside the Database Backup folder which is viewable under tree view. Be sure to unzip the file before proceeding.

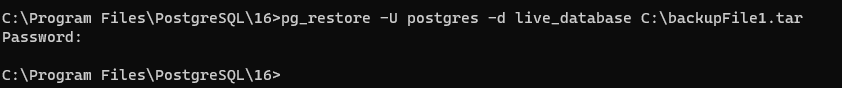
<https://doi.org/10.7910/DVN/QADUKS>

Run the following commands in the PSQL command line which should have been installed when you set up PostgreSQL on your local machine.

A screenshot of a computer program

Description automatically generated

In your machine’s Local Command Prompt run the following commands. Before running the commands ensure to add PostgreSQL 16’s bin file to the command line path.



When prompted for password please enter the password you used when setting up PostgreSQL on your local machine.

# Remote Database Folder (What’s included?)

## App Deployment Folder (What’s included?)

### AdvanceSearchWindow (.mlapp file)

This app will conduct basic searches in the live\_table.

### HardCodeTable (.xlsx file)

This excel file is a representation app.info which is explained later in detail later in this paper.

### InfoWeWant (.xlsx file)

This excel file is a list of variables we want the Serendipity app to keep track of.

### Main (.mlapp file)

This app is the main page of the Serendipity app

### postgresql-42.3.1 (.jar file)

This file is the jar file that the “Start.mlapp” and database connection both need to execute.

### Start (.mlapp file)

# IMPORTANT!!!!

This app is the start page for Serendipity. It is HIGHLY RECCOMMENDED that you leave this file in the App Deployment folder and simply create a shortcut to this file on your desktop. This can be accomplished by right clicking on “Start.mlapp” and clicking the “Create shortcut” option.

### backupProtocal (.bat file)

This program will automatically create backup files in the designated directory. You must specify the directory by editing the file.

### createDummyTable (.m file)

### createLiveTable (.m file)

### createRatTable (.m file)

### displayTables (.m file)

# Introduction

The purpose of this paper is to give a detailed explanation of how to set up a remote PostgreSQL database that the Serendipity App can read from and write to. It will also provide a high-level explanation of how the Serendipity app works, but for a detailed explanation the source code should be referenced. Together with the Serendipity App and a supporting remote database the Big-Data analysis can easily be performed on neatly sorted data.

A Windows computer should be used for this process for best results, but it could be recreated on other computers with some research on the user’s part.

# Things to Download Before Beginning

The following programs should be downloaded before beginning as they will all be used in the process of creating and setting up the remote database.

1. MATLAB
   1. <https://www.mathworks.com/help/install/ug/install-products-with-internet-connection.html>
2. MATLAB App Designer
   1. <https://www.mathworks.com/products/matlab/app-designer.html>
3. MATLAB database toolbox
   1. <https://www.mathworks.com/products/database.html>
4. PostgreSQL
   1. <https://www.postgresql.org/download/>
5. PostgreSQL JDBC Driver file
   1. <https://jdbc.postgresql.org/download.html>

Provided that everything was properly downloaded we can now begin configuring the computer to allow for remote access.

# Configuring the Firewall

It is important to configure the firewall because without doing this your server will not allow for remote connections, and thus no data will be able to be uploaded making it impossible for the project to continue.

1. Open up the windows control panel, you can find this program by clicking on the windows start logo and searching “Control Panel”

Graphical user interface, text, application

Description automatically generated

1. Graphical user interface, text, application

   Description automatically generatedSelect the “System and Security” option and arrive at the following screen
2. Select the “Windows Defender Firewall” Option and arrive at the following screen

Graphical user interface, text, application, email

Description automatically generated

1. Click on Advanced Settings and arrive at the following screen.

Graphical user interface, text, application, email

Description automatically generated

1. Click on “Inbound Rules” and arrive at the following screen. The listed rules might be different in your particular case, but this is no reason for case.

Graphical user interface, text, application

Description automatically generated

1. Click on the “New Rule…” option found under the Actions tab on the right side, and arrive at the following screen

Graphical user interface, application, Teams

Description automatically generated

1. Select the “Port” bubble and click “Next”, arriving at the following screen.

Graphical user interface, application

Description automatically generated

1. Click the “Specific local ports:” bubble and enter “5432” in the text box, which is the default port that PostgreSQL uses. This might change in the future as PostgreSQL releases new updates, PostgreSQL documentation should be referenced for this.

Graphical user interface, application

Description automatically generated

1. Click on next and arrive at the following page

Graphical user interface, text, application, Teams

Description automatically generated

1. Click on the “Allow the connection” bubble and click next, arriving at the following screen.

Graphical user interface, text, application

Description automatically generated

1. Click “Next” without modifying anything, and arrive at the following page

Graphical user interface, application, Teams

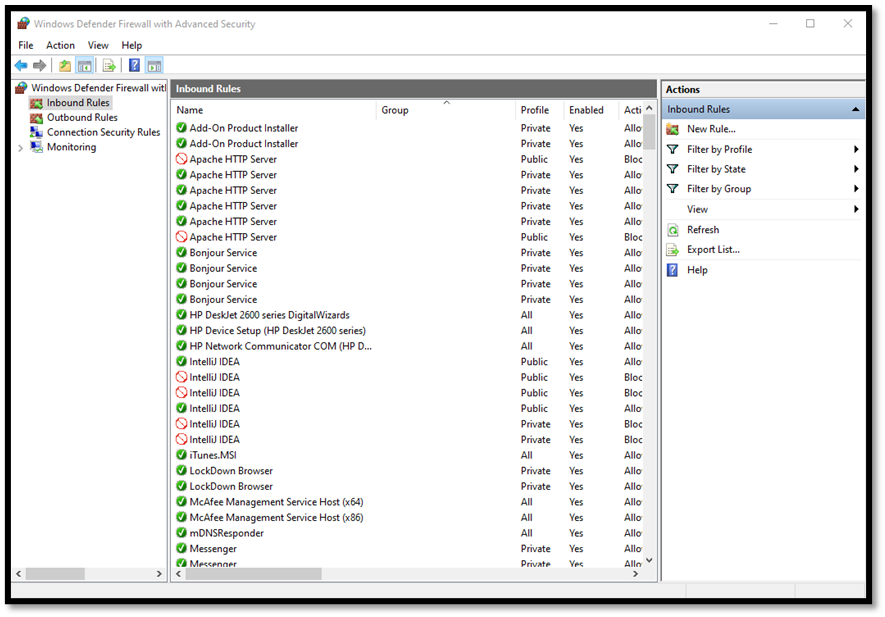
Description automatically generated

1. Fill “PostgreSQL Database Rule” in the “Name” field and enter “Rule that allows outside connections to the PostgreSQL database” in the “Description (optional):” field.

Graphical user interface, text, application

Description automatically generated

1. Click “Finish” and you’ll arrive at the following screen.



1. Your rule will now be listed under Inbound Rules and can be deleted, rewritten, or recreated using the previous steps.

# Configuring PostgreSQL for Remote Access

It is import to configure PostgreSQL for Remote Access because by default it does not allow remote access. Without this configuration nothing will be able to be uploaded to the database, and the project cannot continue.

1. Do not begin this portion before downloading PostgreSQL to your computer, if you have not yet done that please refer to page 2
2. The following steps allow for the server you are creating to be accessed remotely, as by default PostgreSQL doesn’t allow remote connections
3. Click on the windows start tab and search for “sql shell” which should have been downloaded to your computer when PostgreSQL was installed.
4. DO NOT left click on the “sql shell” application, instead right click on it and select the “open file location” option

A screenshot of a computer

Description automatically generated with medium confidence

1. You’ll arrive at a screen that looks like the following

Graphical user interface, text, application

Description automatically generated

1. Once again right click on SQL shell and arrive at the next screen

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

1. Click on “13” and arrive at a screen that should look like the following but may not match exactly. The important part is being able to find the “data” folder. Click on it and look for the pg\_hba.conf file.

Graphical user interface, application

Description automatically generated

1. Open up the file and scroll all the way to the bottom, copy and past the following lines of code.

host all all 0.0.0.0/0 md5

host all all ::/0 md5

1. It should look like the following

Graphical user interface, text

Description automatically generated

# Creating A Server and Database

Creating the Server and Database allows for the data our project is creating to be stored, thus allowing us to access it whenever necessary.

1. The easiest way to do this is through the "pgAdmin” GUI already installed when you installed PostgreSQL
2. Simply search in the windows start for “pgAdmin” and you should see the following

Graphical user interface, application

Description automatically generated

## What’s the recommended password

1. Open up the app, you will be prompted to set a master password, make it something simple, recommended “1234”
2. With everything set up you should see a screen like the following

Graphical user interface, text, application, email

Description automatically generated

1. You will not have as many servers as you see in the photo, you will likely start off with just 1 named “PostgreSQL 13”
2. Right click on the “servers” label, go to create-> server

Graphical user interface, application, Word

Description automatically generated

1. You’ll arrive at the following screen

Graphical user interface, application

Description automatically generated

## recommended server name

1. Fill “lab\_server” into the “Name” field

Graphical user interface, application

Description automatically generated

1. Click on the “Connection” tab and arrive at the following screen

Graphical user interface, text, application, email

Description automatically generated

1. Fill “localhost” in the “Host name/address” field
2. Fill “5432” in the “Port” field
3. Fil “postgres” in the “Maintenance database” field
4. Fill “postgres” in the “Username” field
5. Fill the previously created master password into the “Password” field
6. Click on Save

Graphical user interface, text, application, email

Description automatically generated

1. You’ll now see a new server in your servers group named “lab\_server”

Graphical user interface, text, application

Description automatically generated

1. With this done expand the “lab\_server” tab and right click on “Databases”
2. Go to “Create” -> “Database”

Graphical user interface, application

Description automatically generated

## Recommended Database Name

1. Arrive at the following screen and fill in “Live\_Database” into the “Database” field and click save

Graphical user interface, text, application

Description automatically generated

1. If everything was done correctly then by expanding the “database” menu you should see your newly created database

Graphical user interface

Description automatically generated with medium confidence

1. With all this done your database is now set up

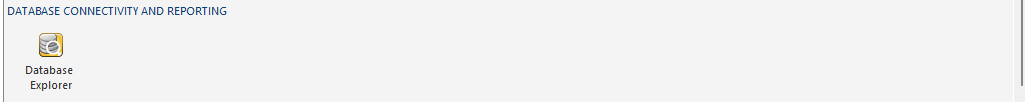
# Creating a Database Connection

Creating the database connection enables the Serendipity app, and other basic connection algorithms to be run. This allows us to actually access the data we have created, enabling the project to move forward.

1. To create a database you must have the MATLAB database toolbox installed, otherwise it is impossible
2. If you do not have it installed, please download it at the following link

<https://www.mathworks.com/products/database.html>

1. With the database tool box installed you can now proceed with making a database connection
2. Begin by opening MATLAB
3. Navigate to the “Apps” tab in the taskbar
4. Select the “Database Explorer”
   1. You may need to select the down arrow if the “Database Explorer” app isn’t immediately seen among the other apps
   2. The “Database Explorer” icon can be seen below



1. With the database explorer app now open go to the “Connections” grouping and select the “Connect” drown down arrow which will open a new menu

Graphical user interface, application

Description automatically generated

1. Navigate down to “New Data Source” and Select the “New JDBC Data Source” option which will automatically open a creation wizard

Graphical user interface, application

Description automatically generated

## What should I put in my fields?

### Name

If you used the recommended database name found in “Recommended Database Name” then you should enter “live\_database” into this field. If another name was entered when creating the database that same name must be used here.

#### Common mistake

If the name of the database and the name of the database connection do not match, none of the code will work

### Vendor

Select PostgreSQL as the vendor.

### Driver Location

In the “Things to Download Before Beginning” section of this paper, you were instructed to download the JDBC driver file. Here you must select that file before proceeding.

### Database

If you used the recommended database name then you should enter “live\_database”, if you named it something else you must use the other name.

#### Common Mistake

If the value entered here does not match the name of the database entered when creating the database then none of the code will work.

### Server

Enter “localhost” here.

### Port Number

By default it should populate 5432, and this should be perfect as long as the ports were never manually changed by someone for security purposes.

1. With all these fields filled in you can now click on the “test” button.
2. A prompt for the password and username will appear

Graphical user interface, application

Description automatically generated

1. Enter “postgres” into the username and “1234” for the password

# Creating the Live Table In your database (createLiveTable.m)

It is important to create the live\_table because this is where everything will be uploaded and sorted. This table allows our project to quickly and efficiently access our data.

1. To create a table in the “lab\_server” database where the information will actually be stored locate the “createLiveTable.m” file
2. Fill in the password, and click run
3. The live\_table is now created and is ready to be written to
4. By default the live\_table has the following columns
   1. subjectID
   2. gender
   3. birthdate
   4. genotype
   5. cagenumber
   6. health
   7. cagemates
   8. experimenter
   9. tasktypedone
   10. notes
   11. intensityofcost1
   12. intensityofcost2
   13. intensityofcost3
   14. intensityofcost4
   15. costprobability1
   16. costprobability2
   17. costprobability3
   18. costporbability4
   19. rewardconcentration1
   20. rewardconcentration2
   21. rewardconcentration3
   22. rewardconcentration4
   23. rewardvolume1
   24. rewardvolume2
   25. rewardvolume3
   26. rewardvolume4
   27. rewardprobability1
   28. rewardprobability2
   29. rewardprobability4
   30. rewardprobability4
   31. mazenumber
   32. approachavoidtimestamp
   33. approachavoid
   34. playstarttrialtone
   35. presentcost
   36. lightlevel
   37. referencetime
   38. videostarttime
   39. feeder
   40. stoptrack
   41. trialname
   42. detectionsettings
   43. trialcontrolsettings
   44. referenceduration
   45. animalid
   46. mazeofferdelivery
   47. nazenoofferdelivery
   48. starttime
   49. recordingafter
   50. recordingduration
   51. trialduration
   52. mazecostoff
   53. coordinatetimes
   54. xcoordinates
   55. ycoordinates
   56. presentcostend
   57. costpresentationfinish
   58. stopincopixrecording
   59. decisionmakingtime
   60. startincopixrecording
   61. date
   62. activezonetimestamp
   63. activezonevalue
   64. presentcosttimestamp
   65. costacknowledgementtimestamp
   66. deliveryacknowledgedtimestamp

It should be noted that if a new table ever has to be created or if the desired variables change they can be changed in this file. PostgreSQL syntax should be studied in order to make well thought out changes.

# Creating Rat Table (createRatDatabase.m)

The rat table is functionally a list of the rats our project is experimenting on. Having this list is important to the project as knowing what rats have what attributes is will be important for our analysis.

1. This file creates a small table named “rattable” in your database that has all the rats currently being experimented on
2. By default, the table “rattable” has the following columns:
   1. name
   2. gender
   3. birthdate
   4. cagenumber
   5. health
   6. genotype
   7. cagemates
3. This file should be edited to match your rats: their age, name, health, and other things you may want to keep track of
4. If you are making edits, be sure to study PostgreSQL syntax to make sure they are done correctly

# Creating a Dummy Table (createDummyTable.m)

The dummy table is important for troubleshooting any updates before actually implementing them on the live table

1. The dummy table is an empty clone of the live\_table, and should be used for any tests before things are executed on the live\_table
2. The dummy table exists on the same database as the live\_table
3. Simply enter the password in the dummyTable.m script where indicated and click run, it should work simply from there

# Look at Tables(displayTables.m)

This file is useful for allowing programmers to look at the data and ensure it is being parsed and sorted correctly. This allows the project to verify it’s results.

1. This function is helpful when you want to make sure things are being uploaded as expected
2. simply fill in the name of your database connection, the password, the username, and the name of the table you want to view where indicated and run the program
3. If the table you want to see is empty nothing will print
4. if the table you want to view has many rows then it might take a minute to run

# Establishing Backups (backupProtocal.bat)

From time to time the database may crash, and thus to not lose any information creating reliable backups is crucial

1. A .bat file has been included that will assist in this, it is named backupProtocal.bat
2. Look for the “ENTER PASSWORD HERE” prompt and enter the password that was created for the database
3. Look for the “ENTER DIRECTORY YOU WANT TO BACKUP TO HERE” prompt and enter the directory

### Pro tip

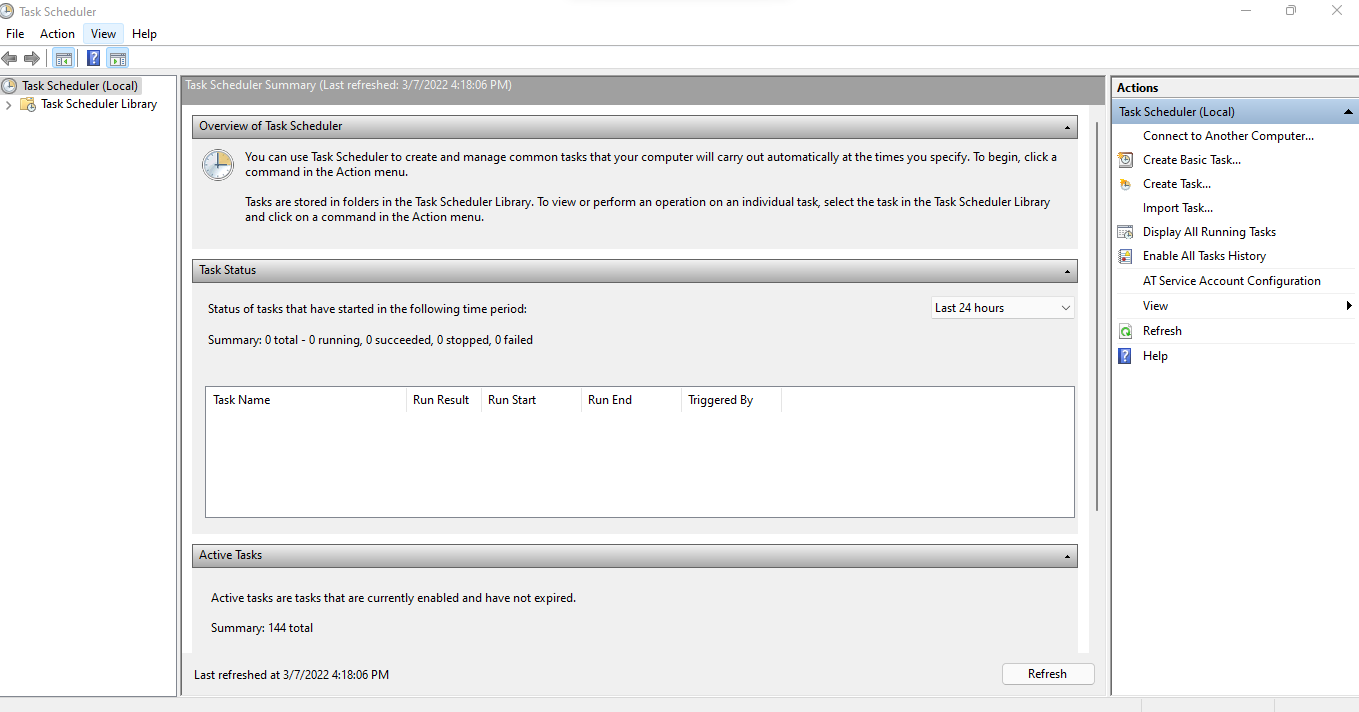
The directory you store the backup in should ideally be backed up on some kind of cloud service like OneDrive, google drive, etc. This ensures that as long as these companies don’t shut down their servers you will always have access to your backups.

1. The backupProtocal.bat file by default creates 5 back .tar files, you can of course modify it to do more, but I would not recommend less
2. With the changes to backupProtocal.bat made, save the file and relocate it to your desktop.
3. Once this file is on the desktop go to the windows start icon and search for “task scheduler”

A screenshot of a computer

Description automatically generated with medium confidence

1. Open task scheduler and arrive at the following page



1. Look for the “Create Basic Task” option and click on it, opening the following task creation wizard

Graphical user interface, text, application

Description automatically generated

1. I recommend naming the task “Backup database Task”, description can be left blank, click next

#### Recommended Task Name

Backup database Task

1. Arrive at the following Screen and Select “daily” then hit next

Graphical user interface, application

Description automatically generated

1. Fill in Today’s Date and 12:00:00 AM in the fields and click next

Graphical user interface, text, application

Description automatically generated

1. Click next again, no need to change anything
2. When you arrive at the following page click on the “Browse” button and select the “backupProtocal.bat” file which should be located on your desktop, Click nextGraphical user interface, text, application

   Description automatically generated
3. Click finish
4. You have now finished creating a backup
5. This backup is establishing a backup of the entire database, so all tables that exist within that database will be preserved by it

# The Serendipity App

The Serendipity app is a useful Graphic User Interface that has built in parsing, graphing, searching, and uploading capabilities. The app is very simple to use and lab-techs with little to no programming experience can upload, graph, and search data with little training. Using this system thousands of rows of data can be uploaded to the remote-database and scripts can be written to analyze all this data for verification purposes.

Before the Serendipity App can be deployed for use however it must first be modified for your specific purposes. What follows is a list of all the functions in the Serendipity App and their general purpose,[[1]](#footnote-1) as well as which functions need to be modified before deployment.

# Start.mlapp

## Callbacks

### SetDriverPathButtonPushed

Pushing this button simply opens a file explorer windows and allows the user to choose the .jar file located in the App deployment folder. This is necessary as this file is used in uploading.

Once a jar file is selected the Serendipity App is told the path was set.

### SetIPAddressButtonPushed

This function tells the computer that the user has entered an IP address.

### LoginButtonPushed

This callback opens the Main.mlapp function allowing the user to begin uploading. If the IP address and driver path are not set then Main.mlapp will not open.

## Properties

### app.IPAddressSet

0 if IP address is not set. 1 if it is set.

### app.driverPathSet

0 if the driver path is not set, 1 if it is set.

### app.driverPath

String representation of the driver path.

# Main.mlapp

## CallBacks

### startupFcn

Upon the opening of any instance of Main.mlapp 3 things occur.

1. The rat table is searched for the default ‘none’ rat and its info are populated into the Serendipity App
2. The IP address and driver path are read from “Start.mlapp”
3. The database connection is uploaded

### SearchButtonPushed

Opens an instance of “AdvanceSearchWindow.mlapp.

### SelectNodulusFileButtonPushed

Pressing this button allows the user to choose a folder full of excel files that will be parsed and uploaded

### SelectRatDropDownValueChanged

When the drop down is activated it reveals a list of rats to the users. Selecting a rat will populate all of that rat’s information into the Serendipity App. This is accomplished by performing a search for that rat in the remote rat table. The rat chosen here should be the rat in maze 1.

### SelectRatDropDown\_2ValueChanged

Performs the same function as “SelectRatDropDownValueChanged”, but for the rat in Maze 2.

### SelectRatDropDown\_3ValueChanged

Performs the same function as “SelectRatDropDownValueChanged”, but for the rat in Maze 3.

### SelectRatDropDown\_4ValueChanged

Performs the same function as “SelectRatDropDownValueChanged”, but for the rat in Maze 4.

### AddDataToDatabaseButtonPushed

The activation of this button prompts the Serendipity app to parse all the excel documents in the Nodulus file specified by the user in “SelectNodulusFileButtonPushed”. Specifically it looks for all the items in the “InfoWeWant.xlsx” document also contained within the “App Deployment” folder, and returns their [row,column] position in an array that other functions can then access to get information out of.

### SetTodaysDateButtonPushed

This button simply get’s the date in MM/DD/YYYY format, and puts in the GUI, so that it can later be read by the parsing. This allows for additional functionality later.

### Port1DropDownValueChanged

This function will import the text in the other Maze’s information boxes, into Maze 1’s information boxes. This function is more a quality-of-life function for any end-users, but also conveniently minimizes human error.

### Port2DropDownValueChanged

This function will import the text in the other Maze’s information boxes, into Maze 2’s information boxes. This function is more a quality-of-life function for any end-users, but also conveniently minimizes human error.

### Port3DropDownValueChanged

This function will import the text in the other Maze’s information boxes, into Maze 3’s information boxes. This function is more a quality-of-life function for any end-users, but also conveniently minimizes human error.

### Port4DropDownValueChanged

This function will import the text in the other Maze’s information boxes, into Maze 4’s information boxes. This function is more a quality-of-life function for any end-users, but also conveniently minimizes human error.

### SeeUploadHistoryButtonPushed

This button will open a new window where the history of all rats uploaded so far into the database will be displayed in a simple table. Information included are the dates that the rats ran the trials, the name of the rats that ran the trials, and what maze the rats were in on that particular day.

## Functions

### results(app)

This function takes the [row, column] positions found in the “AddDataToDatabaseButtonPushed” and then accesses various pieces of information from the excel file named “InfoWeWant.xlsx”. Those pieces of data are then stored into a table that other functions in the app can access when it is time to sort the data into rows and upload them to the remote database. The table is provided in the supplementary materials.

### organize(app)

This function takes all the relevant data for the rat in Maze 1, sorts it into a row and uses sqlwrite to write it to the live\_database table. In the case of duplicate data it prompts the user if they want to replace the old data, or not upload.

### organize2(app)

This function performs the same utility as “organize(app)”, but uses it for the rat in Maze 2.

### orgranize3(app)

This function performs the same utility as “organize(app)”, but uses it for the rat in Maze 3.

### organize4(app)

This function performs the same utility as “organize(app)”, but uses it for the rat in Maze 4.

### checkNamesInAppMatchNamesInFile(app)

Will check to make sure that the names put into the GUI, match the names that are in the user designated file path. If they do not match then a warning dialogue will alert the user, but not stop them from uploading the data. This function does not check for order, simply that all the rats appear.

### createDatabaseConnection(app)

Creates a database connection through code, allows it to be dynamic in case the server IP address changes.

## Properties

### app.rat1

A table representing the data of the rat in Maze 1.

### app.rat2

A table representing the data of the rat in Maze 2.

### app.rat3

A table representing the data of the rat in Maze 3.

### app.rat4

A table representing the data of the rat in Maze 4.

### app.actual\_table

The string representation of the table that all uploads will be directed towards. This table exists within the PostgreSQL database “live\_database” which exists within the PostgreSQL server created in the previous section. Updating this single variable to the name of another table will redirect ALL uploads to that other table. To know more about tables please refer to the “createLiveTable.m” file.

### app.rattable

The string representation of the table containing all of the information with all the rats active in the experiment. Please refer to the “createRatTable.m” file if you would like to know more about the rat table.

### app.password

The string representation of the password to the PostgreSQL server. Changing this password here will change all instances of password throughout the app.

### app.datasource

The string representation of the datasource name. Changing this will change the database that all PostgreSQL commands are executed on.

### app.username

The username of database we are accessing. By default, it is “postgres”.

### app.IPaddress

The IP address of the server, this is dynamic, and it comes from the “Start.mlapp” window.

### app.jarFilePath

The path of the PostgreSQL jar file, this is dynamic and gotten from the “Start.mlapp” window.

### app.files

A list of all the files in “app.workingDirectory”.

### app.workingDirectory

The path of the Nodulus file specified by the user in “SelectNodulusFileButtonPushed”. This folder should be full of excel files.

### app.info

This is a table of all the data parsed from each individual nodulus file. A complete break down of it can be seen in the “HardCodeTable.xlsx” file.

### app.universalCounter

Keeps track of which .xlsx file we are currently looking at in the nodulus file.

### app.variableList

The list of columns in the live\_table. As your own live\_table changes you may need to add, remove, or shift items from this list.

### app.universalCurrentNodulusFile

The current nodulus file that the Serendipity app is parsing.

### app.universalVariablesWeWant

A list of all the variables that the Serendipity parser is looking for. This list is read from “InfoWeWant.xlsx”.

### app.rat\_1\_timestamps

A horizontal string array of timestamps of the rat in Maze 1 during the trial.

### app.rat\_1\_x\_coordinates

A horizontal string array of the rat in Maze 1’s x-coordinates during the trial.

### app.rat\_1\_y\_coordinates

A horizontal string array of the rat in Maze 1’s y-coordinates during the trial.

### app.rat\_2\_timestamps

A horizontal string array of timestamps of the rat in Maze 2 during the trial.

### app.rat\_2\_x\_coordinates

A horizontal string array of the rat in Maze 2’s x-coordinates during the trial.

### app.rat\_2\_y\_coordinates

A horizontal string array of the rat in Maze 2’s y-coordinates during the trial.

### app.rat\_3\_timestamps

A horizontal string array of timestamps of the rat in Maze 3 during the trial.

### app.rat\_3\_x\_coordinates

A horizontal string array of the rat in Maze 3’s x-coordinates during the trial.

### app.rat\_3\_y\_coordinates

A horizontal string array of the rat in Maze 3’s y-coordinates during the trial.

### app.rat\_4\_timestamps

A horizontal string array of timestamps of the rat in Maze 4 during the trial.

### app.rat\_4\_x\_coordinates

A horizontal string array of the rat in Maze 4’s x-coordinates during the trial.

### app.rat\_4\_y\_coordinates

A horizontal string array of the rat in Maze 4’s y-coordinates during the trial.

### app.driverPath

The location of the PostgreSQL jar file. This is read from “Start.mlapp”

### app.IPAddress

The Server IP address. This is read from “Start.mlapp”.

### app.replaceALLDuplicates

This double will be 0 if the user is not uploading duplicates. If the user is uploading duplicates then the Serendipity App will ask them if they want to replace the duplicates, and if the user answers yes then this variable will change to 1 and all deletes and reuploads will be performed without further prompting.

### app.namesFoundUniversal

This is the file path of the user designated file in “SelectNodulusFileButtonPushed”.

### app.universalContainer

This is the MATLAB container object that contains the row/column position of all the items listed in app.universalVariablesWeWant.

# AdvanceSearchWindow.mlapp

## Callbacks

### startupFcn

Reads the name of the database connection from “Main.mlapp”.

### SearchButtonPushed

Performs a PostgreSQL query on the “live\_table”. Will return a table of results that match the parameters and display that table in Serendipity app.

### UITableCellSelection

When the user selects any cell on the table the app will record the [row, column] position of the cell they want, so that other functions can be performed.

### DeleteSelectedRowButtonPushed

Allows the user to delete specific rows, but it’s always simpler and safer to reupload the data then to delete specific rows.

### ExportRowToFileButtonPushed

Exports a specific row to an excel file named “yourExportFile.txt”.

### UIFigureCloseRequest

Closes the app when the “X” button is pushed.

### ExportSearchResultToFileButtonPushed

Allows the user to export their search results into an excel file that they can name.

### ImportAndGraphResultsButtonPushed

Allows the user to take a .xlsx file created by the Serendipity app and create a cost/reward graph.

### AddMoreResultsToFileButtonPushed

Allows the user to append more search results to a previously existing search file. This allows the user to perform ranges.

### GraphTheseSearchResultsButtonPushed

Creates a cost/reward graph based off the user’s latest search.

## Properties

### app.queryResult

A table representation of the results of the user’s latest search.

### app.index

The [row, column] position of the cell that the user selects which triggers the UITableCellSelection callback.

### app.exportRow

The single row the user wants to export.

### app.firstOrNot

This double is a 0 if the user hasn’t previously exported a row, and 1 if they have already exported a row.

### app.databaseConnection

The name of the database connection.

### app.username

The username for the remote database.

### app.password

The password for the database.

### app.actualTable

The name of the current live\_table.

### app.queryParts

All the possible parts of the query, this order is important.

### app.exportFile

The name of the file that the user wants to export their query to.

1. Functions Marked as “MODIFY” need to be modified before the Serendipity App can be deployed, some things needed to be hardcoded in order for them to work [↑](#footnote-ref-1)