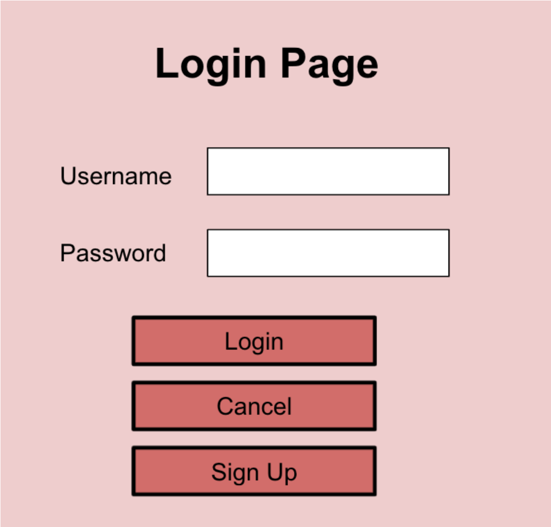
# Criterion B: Design

## Graphical Design

### Figure 1 - Login Page



### Figure 2 - Register Page

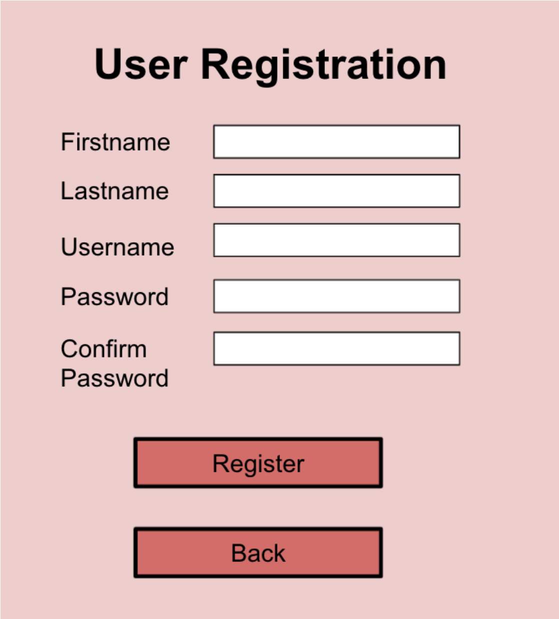


Figure 3 - Home page

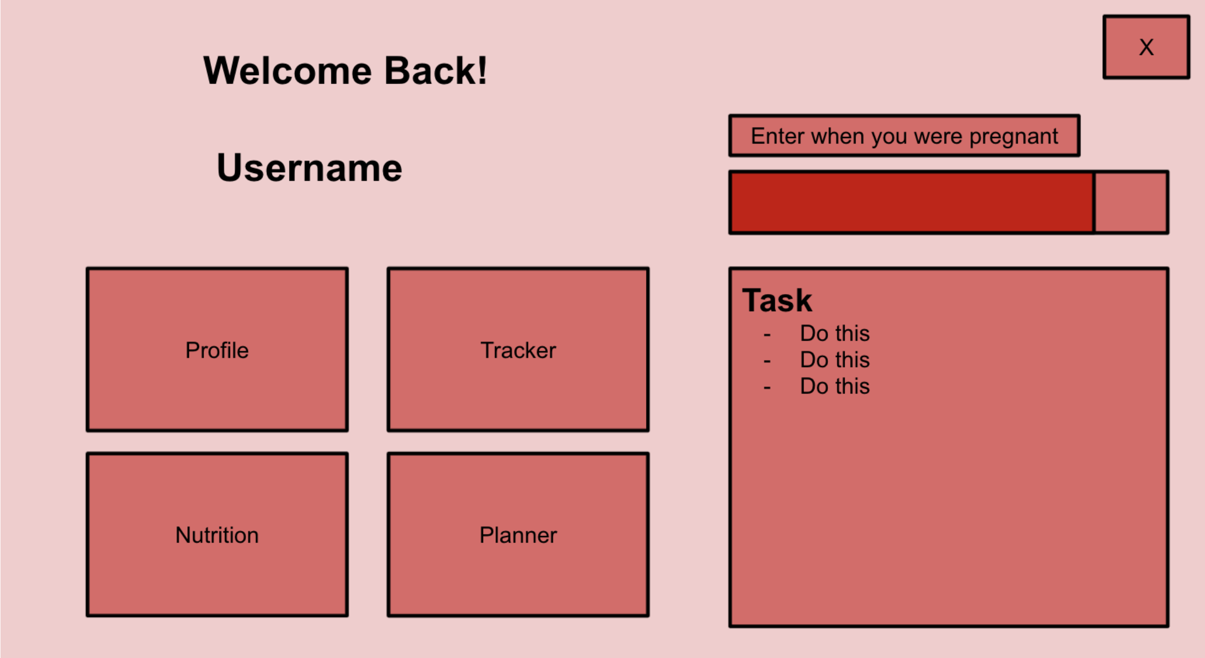
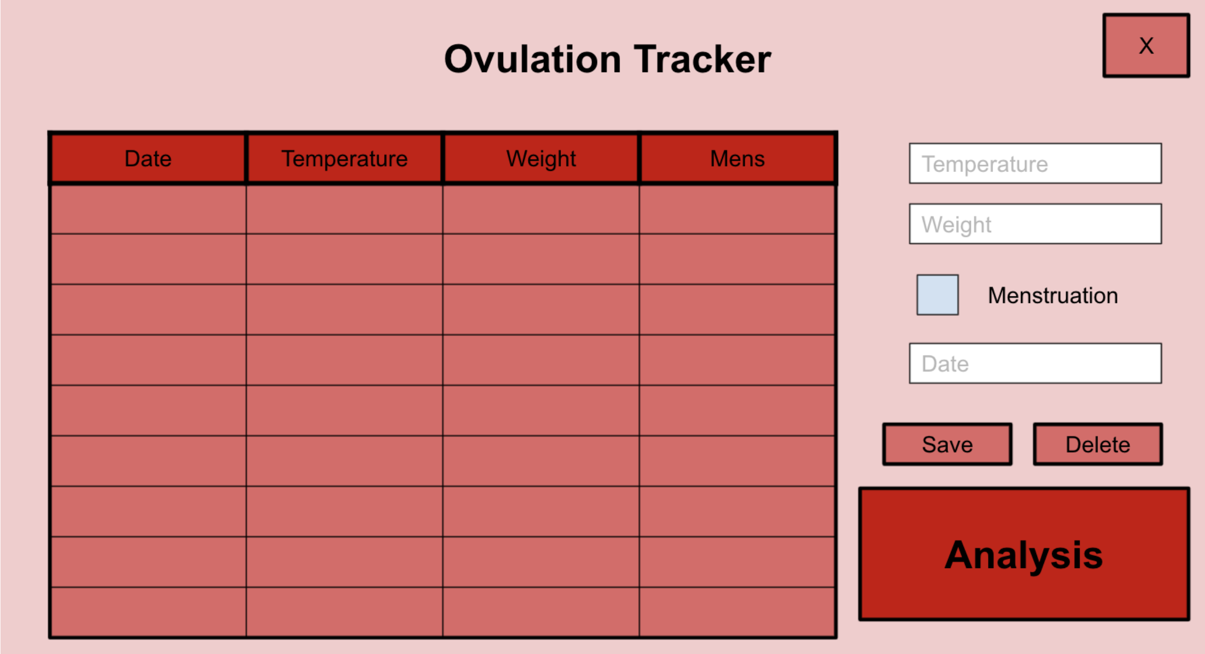
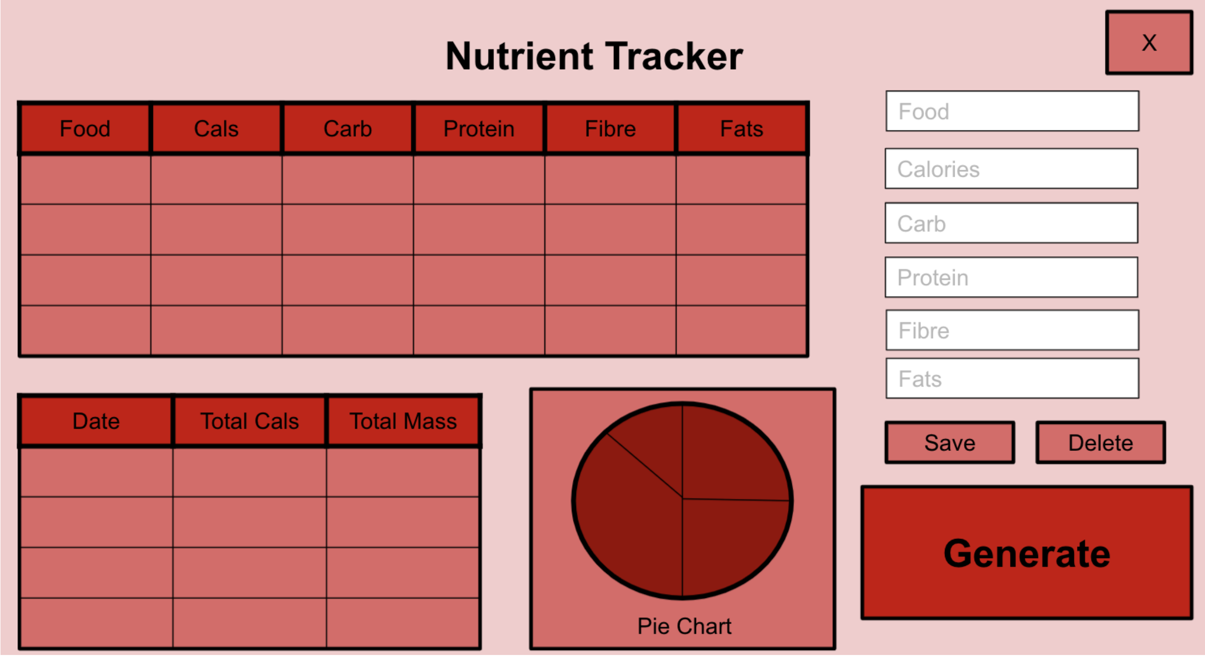


Figure 4 - Ovulation Tracker Page



| Date | Temperature | Weight | Mens |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## Figure 5 - Nutrient Tracker Page

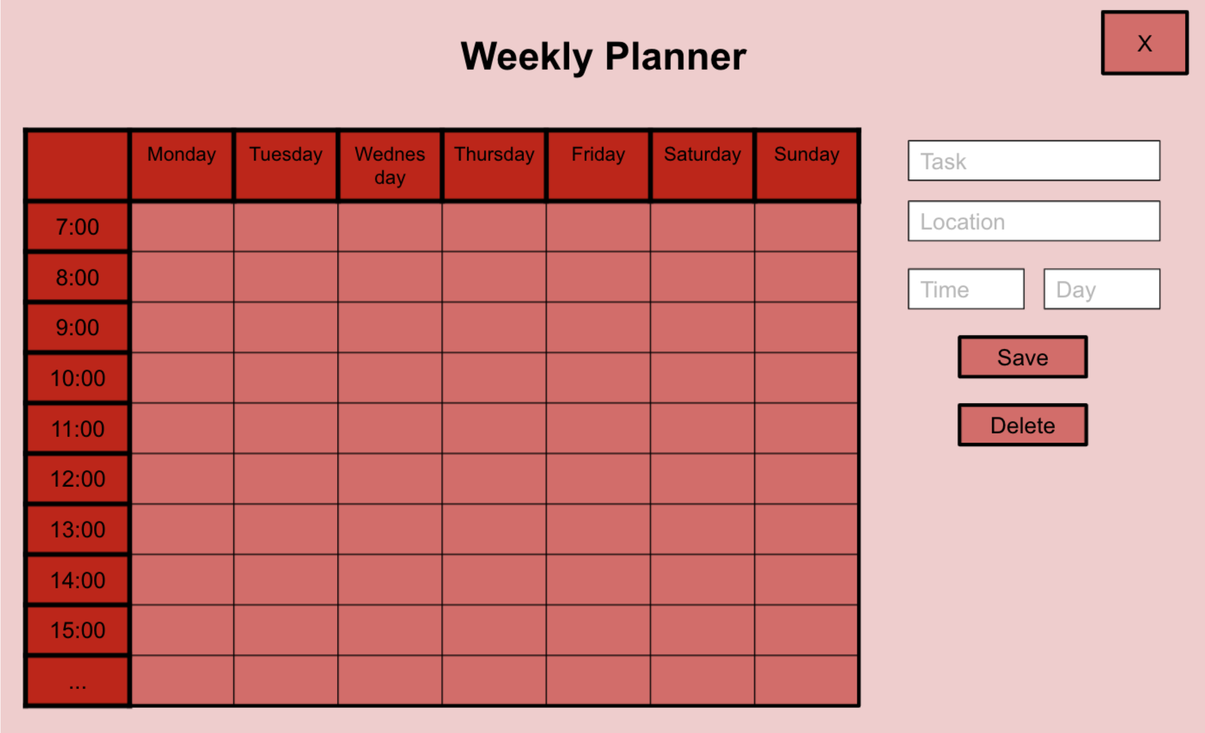


Nutrient Tracker

| Food | Cals | Carb | Protein | Fibre | Fats |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| Date | Total Cals | Total Mass |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

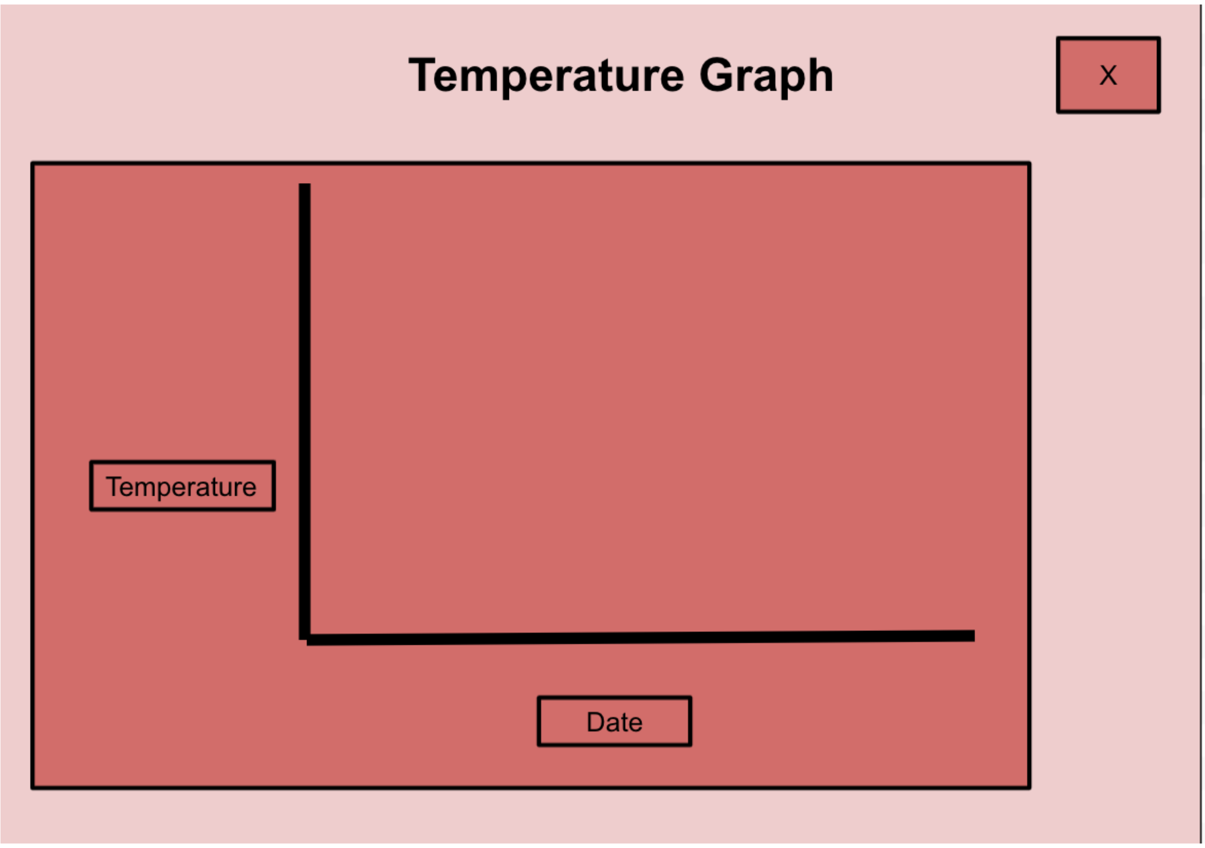
## Figure 6 - Weekly Planner Page



Weekly Planner

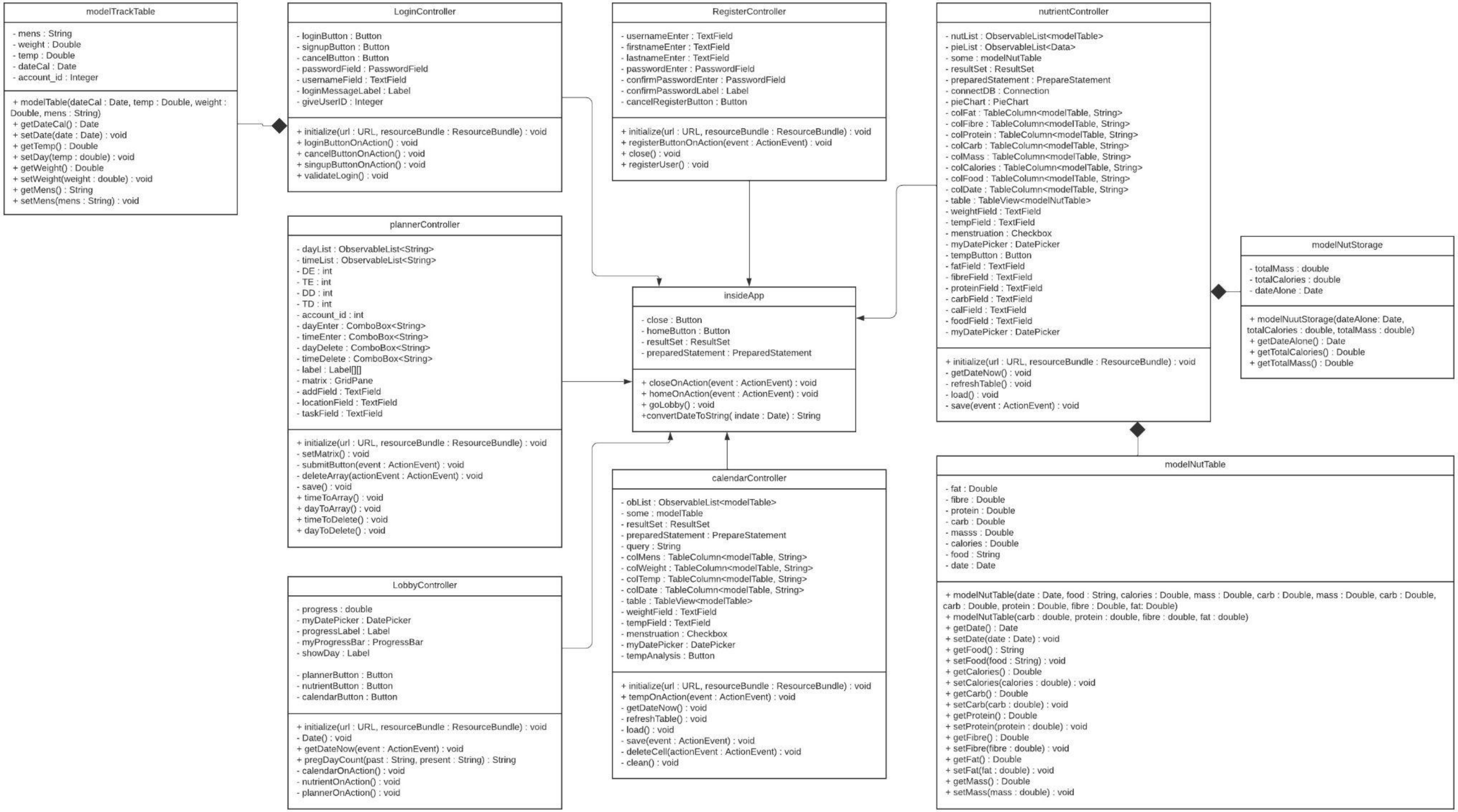
| | Monday | Tuesday | Wednes day | Thursday | Friday | Saturday | Sunday |
|---|---|---|---|---|---|---|---|
| 7:00 | | | | | | | |
| 8:00 | | | | | | | |
| 9:00 | | | | | | | |
| 10:00 | | | | | | | |
| 11:00 | | | | | | | |
| 12:00 | | | | | | | |
| 13:00 | | | | | | | |
| 14:00 | | | | | | | |
| 15:00 | | | | | | | |
| ... | | | | | | | |

### Figure 7 - Temperature Graph Analysis



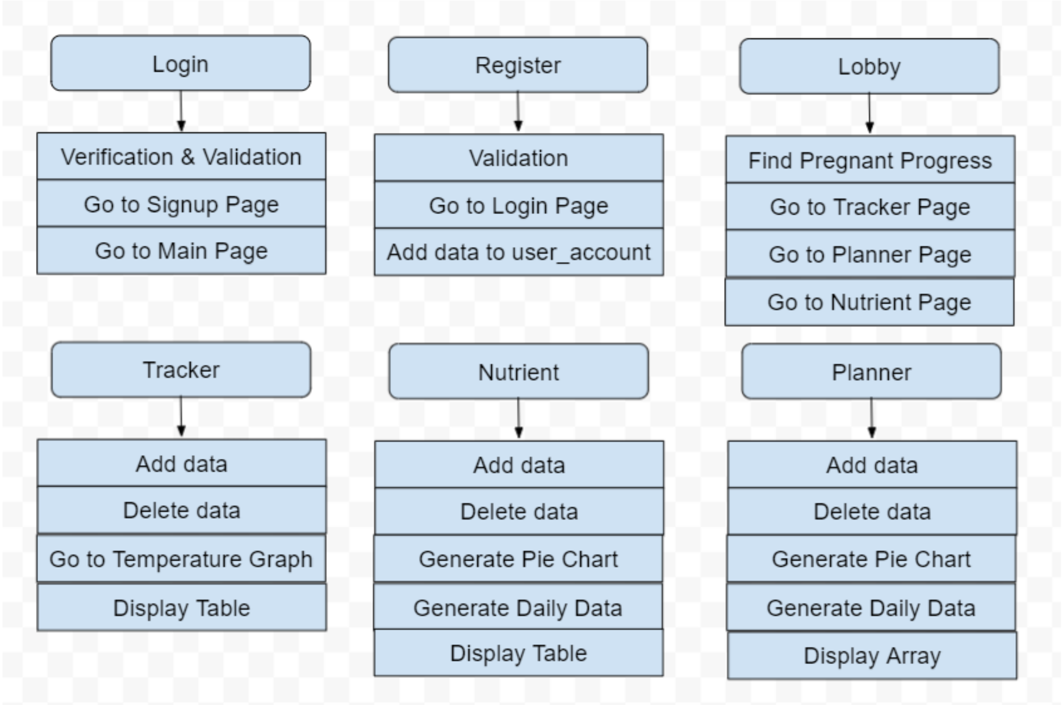
# UML diagram

## Figure 8 - UML diagram showing structure of the program


modelTrackTable
- mens : String
- weight : Double
- temp : Double
- dateCal : Date
- account_id : Integer
+ modelTable(dateCal : Date, temp : Double, weight : Double, mens : String)
+ getDateCal() : Date
+ setDate(date : Date) : void
+ getTemp() : Double
+ setDay(temp : double) : void
+ getWeight() : Double
+ setWeight(weight : double) : void
+ getMens() : String
+ setMens(mens : String) : void
LoginController
- loginButton : Button
- signupButton : Button
- cancelButton : Button
- passwordField : PasswordField
- usernameField : TextField
- loginMessageLabel : Label
- giveUserID : Integer
+ initialize(url : URL, resourceBundle : ResourceBundle) : void
+ loginButtonOnAction() : void
+ cancelButtonOnAction() : void
+ singupButtonOnAction() : void
+ validateLogin() : void
RegisterController
- usernameEnter : TextField
- firstnameEnter : TextField
- lastnameEnter : TextField
- passwordEnter : PasswordField
- confirmPasswordEnter : PasswordField
- confirmPasswordLabel : Label
- cancelRegisterButton : Button
+ initialize(url : URL, resourceBundle : ResourceBundle) : void
+ registerButtonOnAction(event : ActionEvent) : void
+ close() : void
+ registerUser() : void
nutrientController
- nutList : ObservableList<modelTable>
- pieList : ObservableList<Data>
- some : modelNutTable
- resultSet : ResultSet
- preparedStatement : PrepareStatement
- connectDB : Connection
- pieChart : PieChart
- colFat : TableColumn<modelTable, String>
- colFibre : TableColumn<modelTable, String>
- colProtein : TableColumn<modelTable, String>
- colCarb : TableColumn<modelTable, String>
- colMass : TableColumn<modelTable, String>
- colCalories : TableColumn<modelTable, String>
- colFood : TableColumn<modelTable, String>
- colDate : TableColumn<modelTable, String>
- table : TableView<modelNutTable>
- weightField : TextField
- tempField : TextField
- menstruation : Checkbox
- myDatePicker : DatePicker
- tempButton : Button
- fatField : TextField
- fibreField : TextField
- proteinField : TextField
- carbField : TextField
- calField : TextField
- foodField : TextField
- myDatePicker : DatePicker
+ initialize(url : URL, resourceBundle : ResourceBundle) : void
- getDateNow() : void
- refreshTable() : void
- load() : void
- save(event : ActionEvent) : void
modelNutStorage
- totalMass : double
- totalCalories : double
- dateAlone : Date
+ modelNuutStorage(dateAlone: Date, totalCalories : double, totalMass : double)
+ getDateAlone() : Date
+ getTotalCalories() : Double
+ getTotalMass() : Double
plannerController
- dayList : ObservableList<String>
- timeList : ObservableList<String>
- DE : int
- TE : int
- DD : int
- TD : int
- account_id : int
- dayEnter : ComboBox<String>
- timeEnter : ComboBox<String>
- dayDelete : ComboBox<String>
- timeDelete : ComboBox<String>
- label : Label[][]
- matrix : GridPane
- addField : TextField
- locationField : TextField
- taskField : TextField
+ initialize(url : URL, resourceBundle : ResourceBundle) : void
- setMatrix() : void
- submitButton(event : ActionEvent) : void
- deleteArray(actionEvent : ActionEvent) : void
- save() : void
+ timeToArray() : void
+ dayToArray() : void
+ timeToDelete() : void
+ dayToDelete() : void
insideApp
- close : Button
- homeButton : Button
- resultSet : ResultSet
- preparedStatement : PreparedStatement
+ closeOnAction(event : ActionEvent) : void
+ homeOnAction(event : ActionEvent) : void
+ goLobby() : void
+convertDateToString( indate : Date) : String
calendarController
- obList : ObservableList<modelTable>
- some : modelTable
- resultSet : ResultSet
- preparedStatement : PrepareStatement
- query : String
- colMens : TableColumn<modelTable, String>
- colWeight : TableColumn<modelTable, String>
- colTemp : TableColumn<modelTable, String>
- colDate : TableColumn<modelTable, String>
- table : TableView<modelTable>
- weightField : TextField
- tempField : TextField
- menstruation : Checkbox
- myDatePicker : DatePicker
- tempAnalysis : Button
+ initialize(url : URL, resourceBundle : ResourceBundle) : void
+ tempOnAction(event : ActionEvent) : void
- getDateNow() : void
- refreshTable() : void
- load() : void
- save(event : ActionEvent) : void
- deleteCell(actionEvent : ActionEvent) : void
- clean() : void
LobbyController
- progress : double
- myDatePicker : DatePicker
- progressLabel : Label
- myProgressBar : ProgressBar
- showDay : Label
- plannerButton : Button
- nutrientButton : Button
- calendarButton : Button
+ initialize(url : URL, resourceBundle : ResourceBundle) : void
- Date() : void
+ getDateNow(event : ActionEvent) : void
+ pregDayCount(past : String, present : String) : String
- calendarOnAction() : void
- nutrientOnAction() : void
- plannerOnAction() : void
modelNutTable
- fat : Double
- fibre : Double
- protein : Double
- carb : Double
- masss : Double
- calories : Double
- food : String
- date : Date
+ modelNutTable(date : Date, food : String, calories : Double, mass : Double, carb : Double, mass : Double, carb : Double, carb : Double, protein : Double, fibre : Double, fat: Double)
+ modelNutTable(carb : double, protein : double, fibre : double, fat : double)
+ getDate() : Date
+ setDate(date : Date) : void
+ getFood() : String
+ setFood(food : String) : void
+ getCalories() : Double
+ setCalories(calories : double) : void
+ getCarb() : Double
+ setCarb(carb : double) : void
+ getProtein() : Double
+ setProtein(protein : double) : void
+ getFibre() : Double
+ setFibre(fibre : double) : void
+ getFat() : Double
+ setFat(fat : double) : void
+ getMass() : Double
+ setMass(mass : double) : void

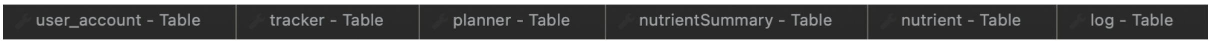
# Function

## Figure 9 - Controller class and its function

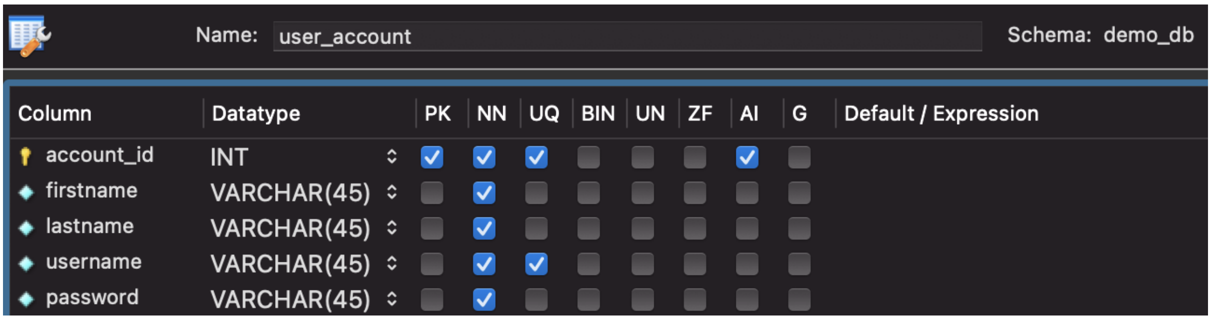


# Database

## Figure 10 - Database table



## Figure 11 - user_account data type and its expression



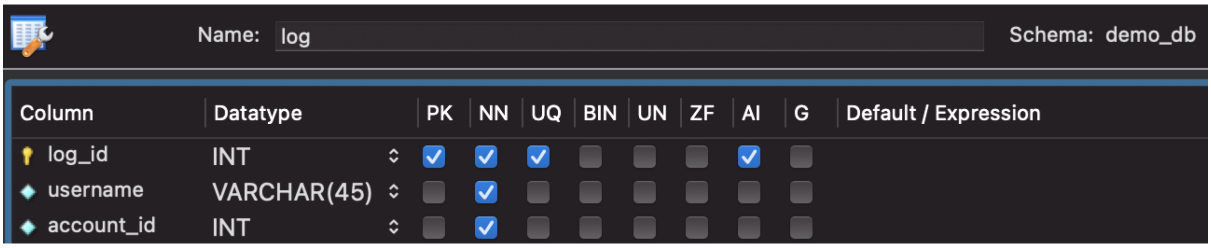
Name: user_account

Schema: demo_db

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| account_id | INT | ☑ | ☑ | ☑ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| firstname | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| lastname | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| username | VARCHAR(45) | ☐ | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| password | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

- NN (Non-Null)
- PK (Primary Key)
- UQ (Unique)
- AI (Auto-Increment)

## Figure 12 - log table data type and its expression

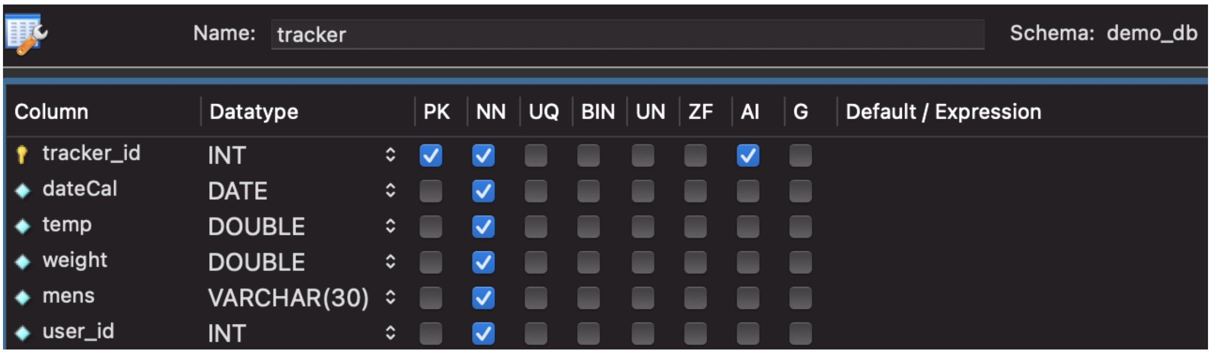


Name: log

Schema: demo_db

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| log_id | INT | ☑ | ☑ | ☑ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| username | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| account_id | INT | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

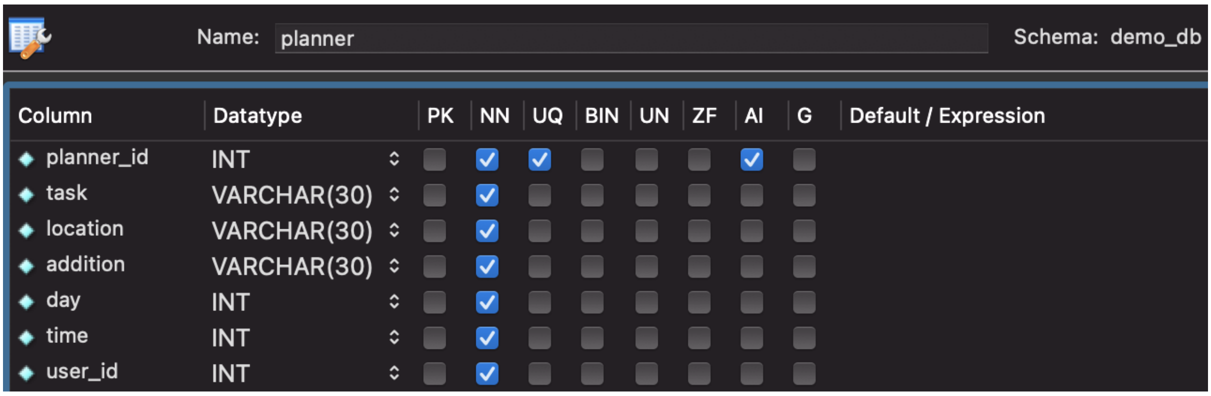
### Figure 13 - tracker table data type and its expression



Name: tracker Schema: demo_db

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| tracker_id | INT | ☑ | ☑ | □ | □ | □ | □ | ☑ | □ | |
| dateCal | DATE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| temp | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| weight | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| mens | VARCHAR(30) | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| user_id | INT | □ | ☑ | □ | □ | □ | □ | □ | □ | |

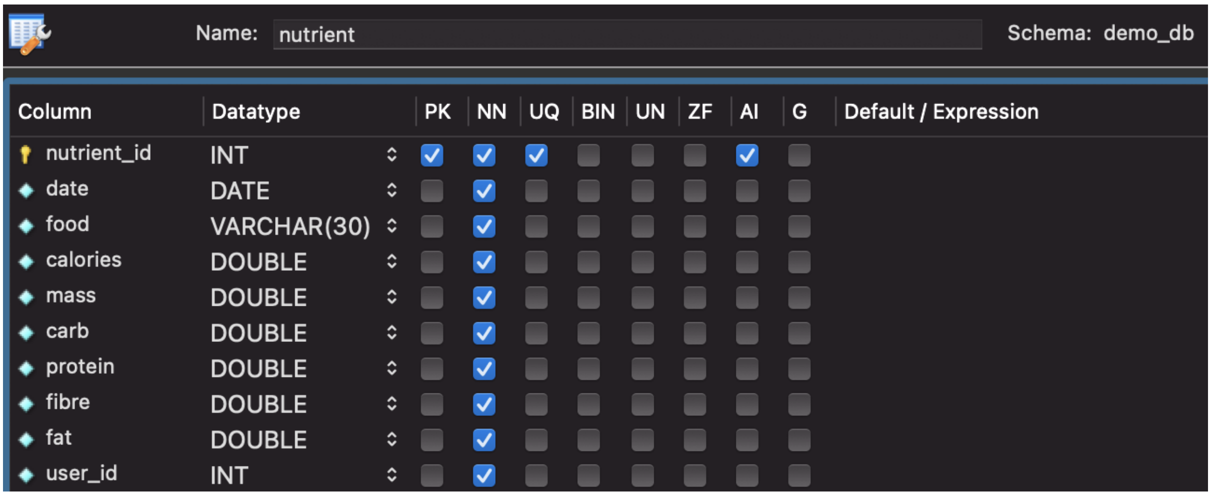
### Figure 14 - planner table data type and its expression



Name: planner Schema: demo_db

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| planner_id | INT | □ | ☑ | ☑ | □ | □ | □ | ☑ | □ | |
| task | VARCHAR(30) | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| location | VARCHAR(30) | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| addition | VARCHAR(30) | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| day | INT | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| time | INT | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| user_id | INT | □ | ☑ | □ | □ | □ | □ | □ | □ | |

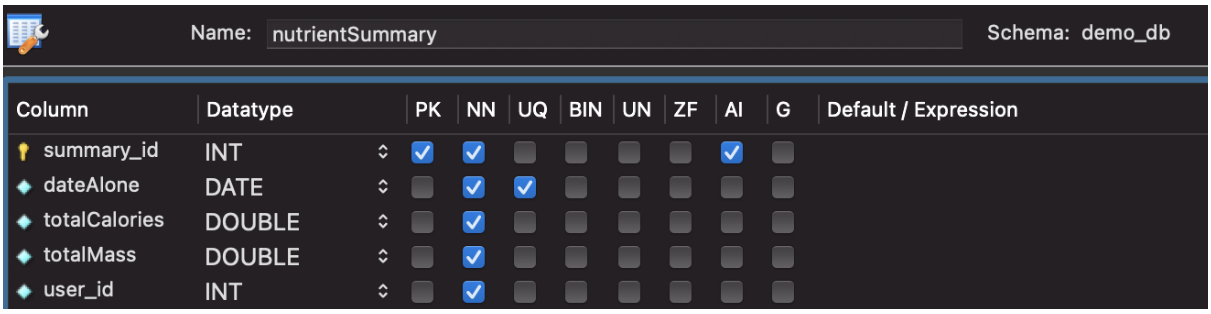
### Figure 15 - nutrient table data type and its expression



Name: nutrient Schema: demo_db

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| nutrient_id | INT | ☑ | ☑ | ☑ | □ | □ | □ | ☑ | □ | |
| date | DATE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| food | VARCHAR(30) | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| calories | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| mass | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| carb | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| protein | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| fibre | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| fat | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| user_id | INT | □ | ☑ | □ | □ | □ | □ | □ | □ | |

### Figure 16 - nutrientSummary table data type and its expression



Name: nutrientSummary Schema: demo_db

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| summary_id | INT | ☑ | ☑ | □ | □ | □ | □ | ☑ | □ | |
| dateAlone | DATE | □ | ☑ | ☑ | □ | □ | □ | □ | □ | |
| totalCalories | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| totalMass | DOUBLE | □ | ☑ | □ | □ | □ | □ | □ | □ | |
| user_id | INT | □ | ☑ | □ | □ | □ | □ | □ | □ | |

# Flowchart

## Figure 17 - Flowchart of how the program work in general

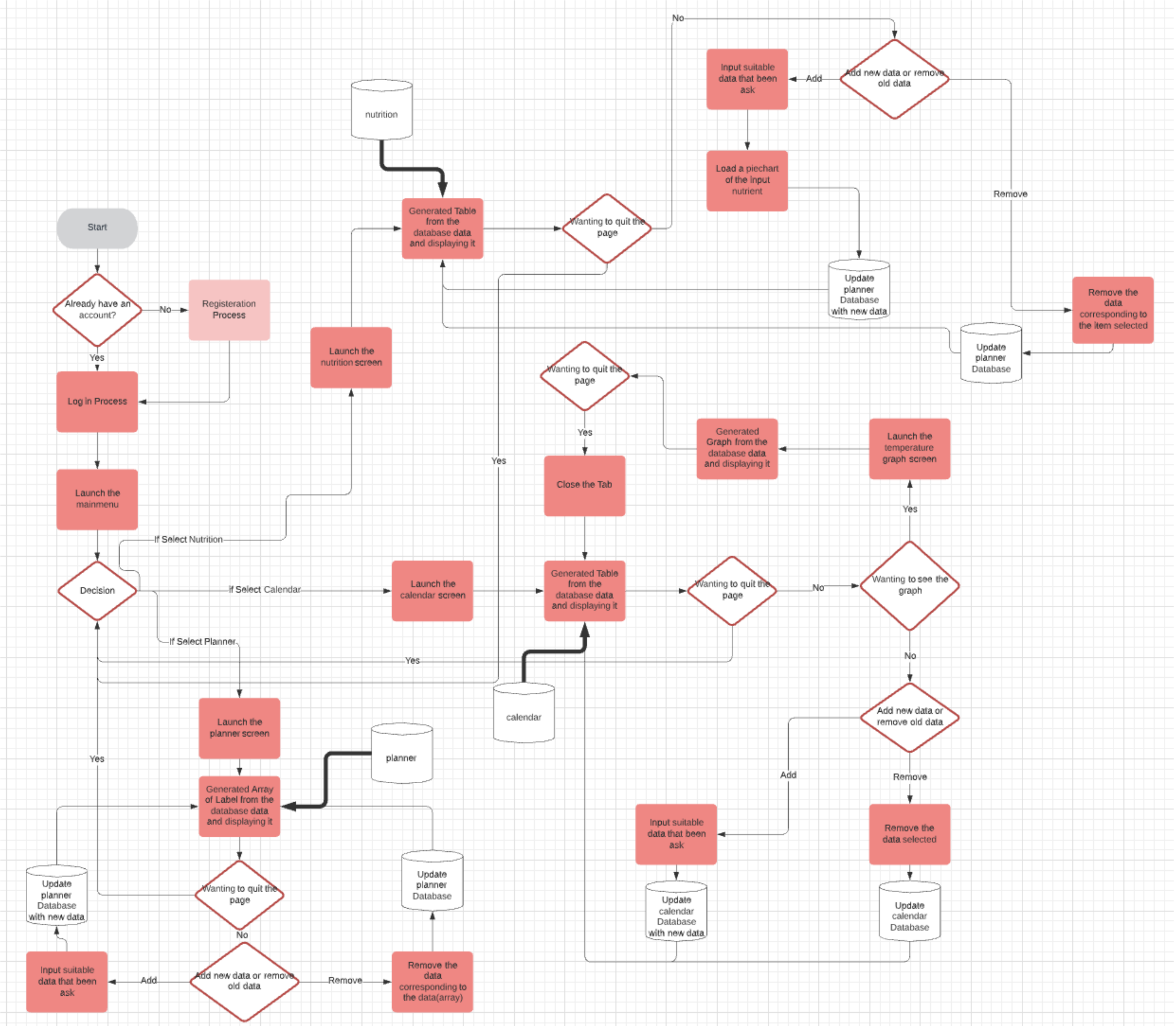
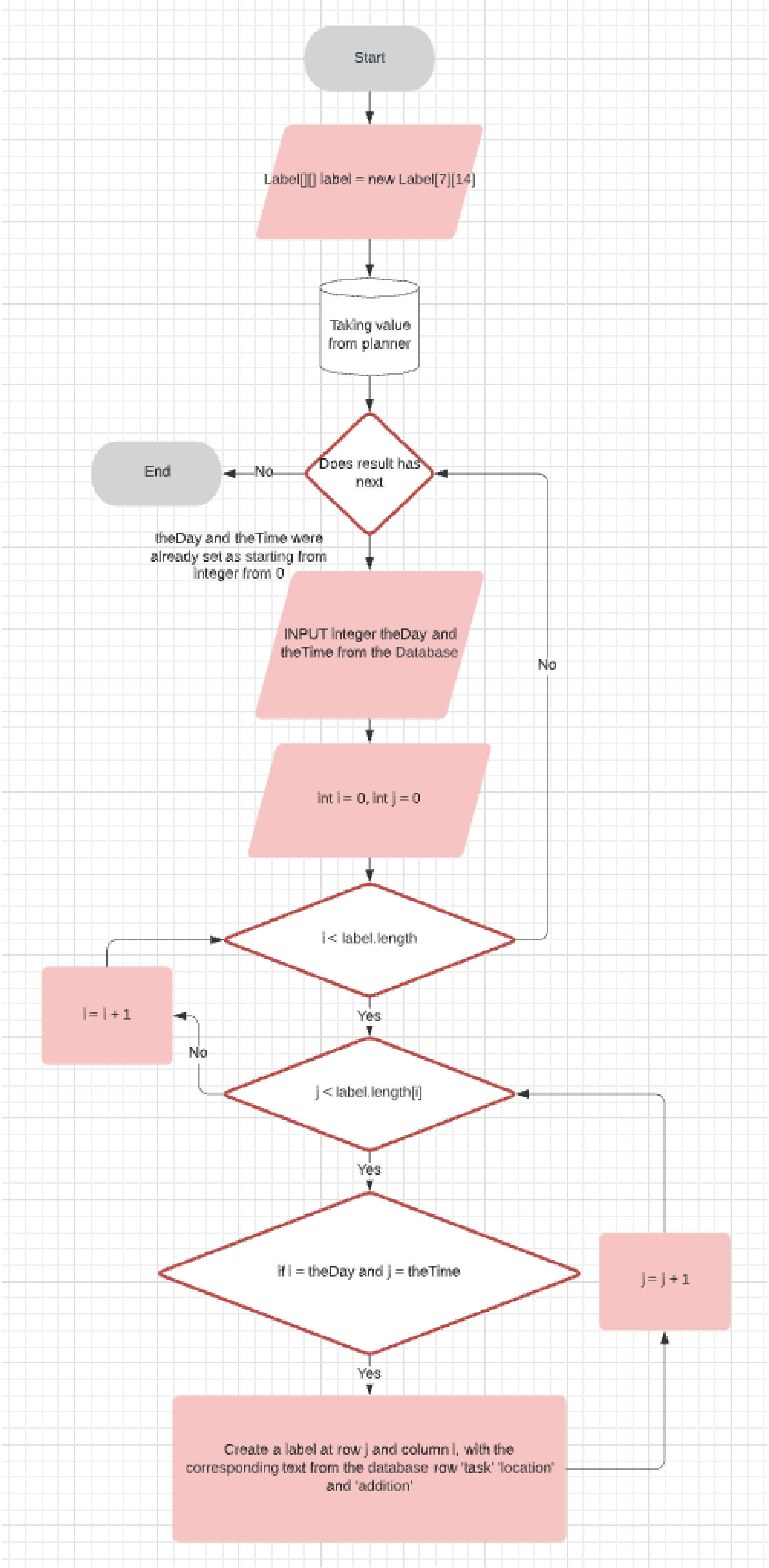

No
Input suitable data that been ask
Add
Add new data or remove old data
nutrition
Load a piechart of the input nutrient
Remove
Generated Table from the database data and displaying it
Wanting to quit the page
Start
Update planner Database with new data
Remove the data corresponding to the item selected
Already have an account?
No
Registeration Process
Update planner Database
Launch the nutrition screen
Yes
Wanting to quit the page
Log in Process
Yes
Generated Graph from the database data and displaying it
Launch the temperature graph screen
Yes
Close the Tab
Yes
Launch the mainmenu
If Select Nutrition
Decision
If Select Calendar
Launch the calendar screen
Generated Table from the database data and displaying it
Wanting to quit the page
No
Wanting to see the graph
If Select Planner
Yes
No
calendar
Add new data or remove old data
Launch the planner screen
planner
Yes
Add
Remove
Generated Array of Label from the database data and displaying it
Input suitable data that been ask
Remove the data selected
Update planner Database with new data
Wanting to quit the page
Update planner Database
Update calendar Database with new data
Update calendar Database
No
Input suitable data that been ask
Add
Add new data or remove old data
Remove
Remove the data corresponding to the data(array)

Figure 18 - Flowchart of multi-array creation (Most complex)


Start
Label[][] label = new Label[7][14]
Taking value from planner
Does result has next
No
End
theDay and theTime were already set as starting from integer from 0
INPUT integer theDay and theTime from the Database
No
int i = 0, int j = 0
i < label.length
i = i + 1
Yes
No
j < label.length[i]
Yes
if i = theDay and j = theTime
j = j + 1
Yes
Create a label at row j and column i, with the corresponding text from the database row 'task' 'location' and 'addition'

# Input/Output

## Figure 19 - Login System

| Input/Output | Content | Data Type | Location | Example |
|---|---|---|---|---|
| Input | Username and Password | String | Text Field | Username: root<br>Password : 1234 |
| Output | Main lobby page | GUI | GUI | [Figure 3] |

## Figure 20 - Add new tracker record

| Input/Output | Content | Data Type | Location | Example |
|---|---|---|---|---|
| Input | Temperature, Weight, Date, Menstruation | Double, Date, Boolean | Text Field, Date Picker, CheckBox | Temperature: 36.753<br>Weight : 62<br>Date: 8/11/2021 |
| Output | “” | “” | Database | Temperature: 36.753<br>Weight: 62<br>Date: 2021-08-11 |
| Output | “” | “” | Display Table | Temperature: 36.8<br>Weight: 62.0<br>Date: 2021-08-11 |

## Figure 21 - Add new planner record

| Input/Output | Content | Data Type | Location | Example |
|---|---|---|---|---|
| Input | Task, Location, Additional Info, Date, Time | String, Integer, Integer | Text Field, ComboBox | Task: Teaching<br>Location: School<br>Additional Info: Biology<br>Date: Monday<br>Time: 7:00 |
| Output | “” | “” | Database | Task: Teaching<br>Location: School<br>Additional Info: Biology<br>Date: 0<br>Time: 0 |
| Output | “” | “” | Display Table | Temperature: Teaching<br>Weight: School<br>Date: Biology<br>Date: Monday Column<br>Time: 7:00 Column |

### Figure 22 - Add new nutrient record

| Input/Output | Content | Data Type | Location | Example |
|---|---|---|---|---|
| Input | Date, Food, Calories, Carb, Protein, Fibre, Fat | Double, Date, String | Text Field | Date: Automatic Today Date<br>Food: Lobster<br>Calories: 50<br>Carb: 10<br>Protein: 8<br>Fibre: 2<br>Fats: 1 |
| Output | Date, Food, Calories, Carb, Protein, Fibre, Fat, Mass | “” | Database | Date: Today Date in “YYYY - MM - DD”<br>Food: Lobster<br>Calories: 50<br>Mass: 21<br>Carb: 10<br>Protein: 8<br>Fibre: 2<br>Fats: 1 |
| Output | Date, Food, Calories, Carb, Protein, Fibre, Fat, Mass | “” | Display Table | Date: Today Date in “YYYY - MM - DD”<br>Food: Lobster<br>Calories: 50<br>Mass: 21<br>Carb: 10<br>Protein: 8<br>Fibre: 2<br>Fats: 1 |
| Output | Carb, Protein, Fibre, Fat | Pie Chart | Pie Chart | Carb: 10<br>Protein: 8<br>Fibre: 2<br>Fats: 1 |

## Program Development Plan

1. GUI setup with Scenebuilder program
    a. Replicate the design from graphical design of this section
    b. Importing some image on to the pane
    c. Name graphical interface item such as button, graph or Text field with appropriate name that can be easily recalled
2. Main Class
    a. Connect to database and link to fxml files(GUI)
    b. Login and Register Functionality
        i. Creation of new data into database
        ii. Verification of new data
        iii. Validation of data login
    c. Lobby Page
        i. Pregnancy calculator
        ii. Progress Bar
    d. Tracker Page
        i. Input data from interface to database
        ii. Putting data from database into interface(table)
        iii. Deleting data from database from interface
        iv. Link to graph page
        v. Display graph with the data from database
    e. Nutrient Page
        i. Input data from interface to database
        ii. Putting data from database into interface(table)
        iii. Deleting data from database from interface
        iv. Concluding table with data from the previous set-up table
    f. Planner Page
        i. Array of pane and text
        ii. Input data from interface to database
        iii. Putting data from database into interface(array)
        iv. Deleting data from database from interface
3. Sub-Classes
4. Bug-Fixing/ Remove Unnecessary
5. Test plan

## Test table

Figure 23 - Test table

| Test Type | Natures of the Test / Expected Outcome |
|---|---|
| (1) The graphical interface of the application like the image should be loaded up | Run the program and observe if the image loads up and that the image that loads up is meant to be there and not similar or overlap from other images. |
| (2) Check if the registration and login function works (This link back to success criteria 1a and 1b) | Insert data into registration and submit it. Then go to the database to check if the value that entered exists and is equal to the one on the database. Then use that data to log in to see if it will validate and send the user to the lobby page. |
| (3) Checking if inserting and deleting data in either array or table work (This link back to success criteria 2, 3 and 4 for entering in data) | Using the assigned button to submit data or delete data, then observe if the data has vanished from the user interface table and the database. |
| (4) Check if the entered data have a suitable value to be stored in a database or the user had entered in the field (This link back to success criteria 1f) | Check if an alert message pops up when the user didn't enter any data or user put it abnormal data |
| (5) Check if the user can see other people data (This link back to success criteria 1b) | Run the program with one account then put in data, switch to another account then see if that data will be able to see from this account. |
| (6) Check if graph analysis for both pie chart and line graph work properly (This link back to success criteria 2d and 3f) | Run the program with some data in the table then use the assigned key button for the graph. Observe the graph if the y axis and x-axis have the designated name and value for each point. Or in the case of a pie chart having the data match the original one. |
| (7) Check if the program home/back button or any button that is designated to other page work | Run the program and through the use of a button check if the destination page is the correct place and make sure that when it goes to the new page the old page exits itself to avoid overlapping. |
| (8) Check if the calculation work and give the right value for the right input (This link back to success criteria 1d, 3d) | Run the calculation algorithm using System.print out.ln() to check the value of data by printing it out before displaying it. |
| (9) Check if resetting the program for the next day would reset the data for that yesterday (This link back to success criteria 3e) | Close the program after entering the data then reset the program and change timezone to tomorrow then run the program to see if the data still there |