**Analysis**

**Introduction**

For my NEA I will be creating an organization app called Pink Plan for students where they can put their social and school life in writing and have the ability plan every day efficiently. It will include weekly timetables where students can add their school timetable or create a study timetable for themselves. It will also include a calendar for upcoming events such as exams or homework being due. Additionally, my application will include daily to do lists. My client can add tasks for each day and set priority levels for each or them and a certain time that task should be completed. There will be a reminder function so my client can remember that they have a task to complete and anti-procrastination messages will help to avoid uncompleted tasks at the end of each day.

**Problem**

My client is a 17-year-old, female, A-Level student named Nebras Shakour. She struggles to stay organised, and procrastinates her schoolwork because of this. After conducting research which I will show below, I discovered Nebras is not the only one with this issue and students ages 15 to 18 also face these problems.

For students there is so much to remember every day and it is easy to forget things or miss homework and important tasks because of this. Life also is not just about school; students must manage being a student and everything else outside of their student life, so things can get complicated. The problem with this is that Students find it hard to stay organized when they have so many things to remember, which was conformed in my research. My app will help students to put every aspect of their life in writing. Every exam, event and step needed for them to be successful will be recorded so nothing is missed. This will be extremely helpful as it leaves no room to forget things and therefore ensures complete organisation. This application will provide better time management skills for young people by allowing them to recognise and understand patterns within their life so they know how much time they need to allocate a task and have a better understanding of what they can complete within a day. This ensures students will not become burnt out. This is when a student does a lot of schoolwork in a short space of time due to procrastination and poor time management and therefore loses motivation to do anything for days or weeks afterwards. It will also make sure students will have a realistic academic and social lifestyle.

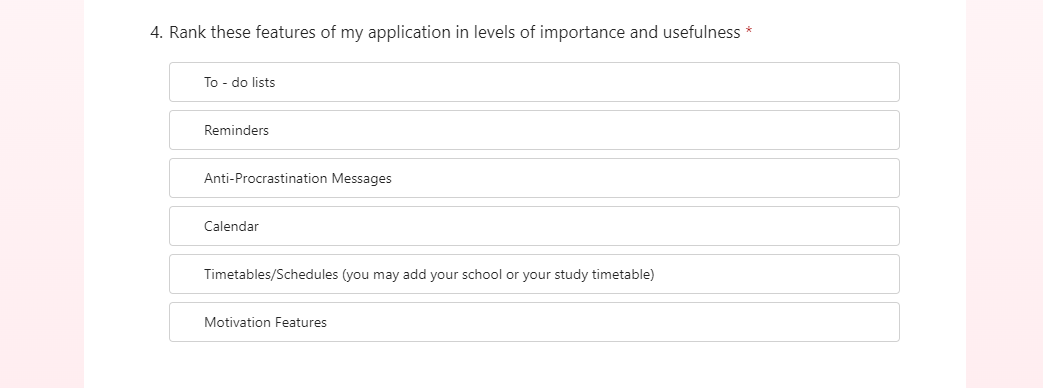
My client is given paper timetables from school, these are easily lost and ruined they're also forgotten quickly. Students can get in trouble for not knowing what lesson they should be in or just for forgetting their timetable resulting in valuable time being spent in detention. Not having your timetable is a major inconvenience for students like my client. It means they must spend time finding out what lessons they have or must recall from memory what lessons they have even though there are many lessons, interventions or appointments it could be. This is even more of an issue for people like my clients with alternating weeks. Having your school timetable on your phone can eliminate these issues as it is highly unlikely to forget your phone in comparison to forgetting a piece of paper with your timetable on it.

I can assume there is less supervision and structure during university in comparison to secondary school as you are an adult then. Students are thrown straight into self-discipline and could quickly realise how easy it is to fall behind and skip lessons when there is no punishment other than the loss of your education. Students can quickly get disorganised this happens as the missed assignments and lessons add up and you will find there is not enough time to do anything. My application will prevent this as younger students who use my application will already be trained before university, on how to stay organised. They also will have previously felt the satisfaction of staying organised and completing things on time which is motivating and will ensure they remain disciplined in their next steps. So, they will not face this issue as when they continue to use it in university anti-procrastination messages and general reminders will keep them on top of work and keep their lives structured.

**Research**

For my application I need to do research on what features people would find useful, such as to do lists, calendars, reminders and more. I will create a form for this which asks people in my target demographic, students ages 15 to 18, to rank these features from most to least valuable. This will give me a better understanding of what is important to students and the features that are not a necessity.

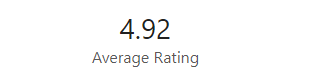
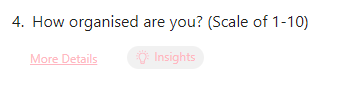
Below is the question I asked for this form. I wanted to see what I should initially focus on for the basic features on my application. The features that are mostly voted first and second are what I will focus on to begin with and those that are mostly voted last, I would use as optional additional features I can add after the main structure.

In this form, I found that 70% of respondents placed to do lists and schedules in their top 3, as this is very much the majority when building my app, I will initially focus on implementing these two features into my app as that is what my target audience are interested in mostly. More importantly my client, Nebras placed to do lists, Anti-procrastination messages and Schedules in her top 3. Taking this into consideration these will definitely be the 3 main features of my app. I will still include the 3 features she placed last in my app however I will not focus on them as much as the top 3. However, I would consider Nebras an anomaly with what she placed last, Reminders, as she gives an explanation as to why they would not be necessary for her, but I can understand why reminders would be essential for another user. For this reason, I will still focus on and ensure that useful reminders are a feature of my app as other users will benefit from this. Below I have shown Nebras’ ranking exactly and her additional comment on the reminder feature.



I need to do research to determine how many people think the idea of my app will be useful to them. I will ask them questions about themselves and how they manage their schoolwork to determine if it aligns with the features I want in my app. I will send my form to students aged 15 to 18. This includes GCSE and A-level students. I believe this demographic of people will benefit from my app the most so this research will help me to make sure.

For the first question the average rating was below half. I asked this question to see how useful and needed my app really is for students and this rating showed me my app was needed most of my respondents are not organised.



For the next question I wanted to see if there were any people using other apps to stay organised so I could see how many competitors were known amongst my local target audience. I found that people did not really use any apps for organisation so there does not seem to be many competitors known amongst people near me. This shows me how necessary my app is for local students as although I personally know of many competitors my audience does not, so they do not yet know how beneficial organisation apps are for students.

In this next question I wanted to see if one of the problems I initially outlined, that I believe my app would solve was true amongst older students who my target audience are. As I previously said paper timetables are easily lost and timetables in general are easily forgotten. I was surprised to see that the average rating of this question was below half as in my other form 70% of respondents wanted the schedule feature but on average, they do not typically forget what lessons they have. This led me to believe that students are interested in the timetable/schedule feature for a different reason.





The next question’s purpose was to determine the benefits of my motivation features. The average score on this was fairly high. However, based on my previous research, the rankings, this average rating was confusing. In the ranking research most respondents placed the motivation feature very low in their ranking. However, they are all quite unmotivated. This makes me believe that the respondents may not think the motivation features will benefit them or they do not understand them. Now, looking back on how I asked the question I do not think I was specific enough as “motivation features” is very broad, to improve in the future I will describe that these motivation features would be messages throughout the day, encouraging users to keep persevering. This likely affected the results of my ranking research as respondents were not given a clear idea of what this feature would consist of.





This question was to ensure that my local target audience would be interested in my app and would use it. The average for this was high. After analysing each individual respondents rating, I noticed that girls mainly rated this question quite high, and boys did the opposite. This led me to believe that females would be the main demographic using my application. So, because of this I will cater features and aesthetics of my application to women and filter my respondents down to just women. In most of my questions males voted against my apps features and against the main colours so my app will consider female students aged 15-18.



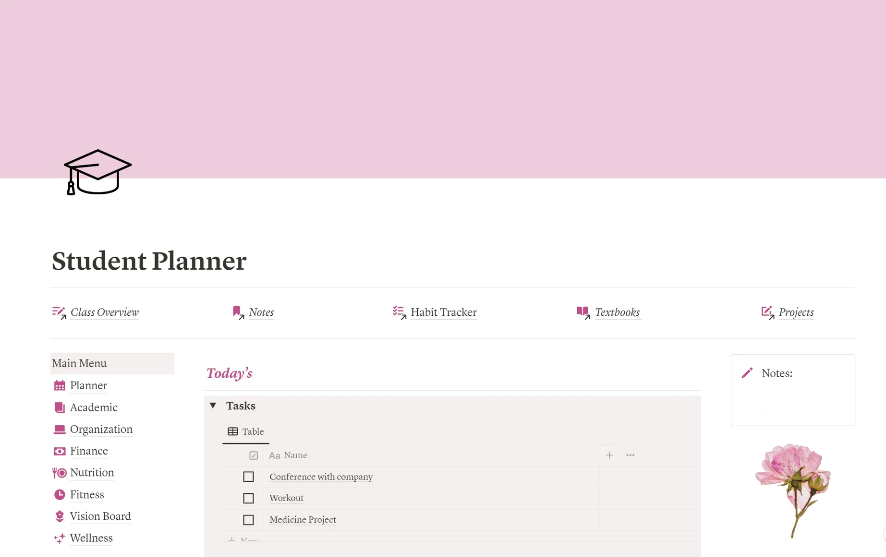


The last 2 questions asked if they like to do lists and schedules. Again, the ratings were high with to do lists having an average of 7.46 and schedules an average of 6.69. This question solidified for me that most of my customers would be women as in the previous form to do lists and schedules appeared to be the favoured features of my application. So, in this form of course I wanted to see if how much people like and would enjoy using these features as I intend to make them the main features of my app. Again, I saw that the respondents who rated these main features the highest were all women. So, I am sure now that I need to cater my app to women.

**Existing solutions**

As a part of my research, I investigated competitors that have the same and similar features as my app.

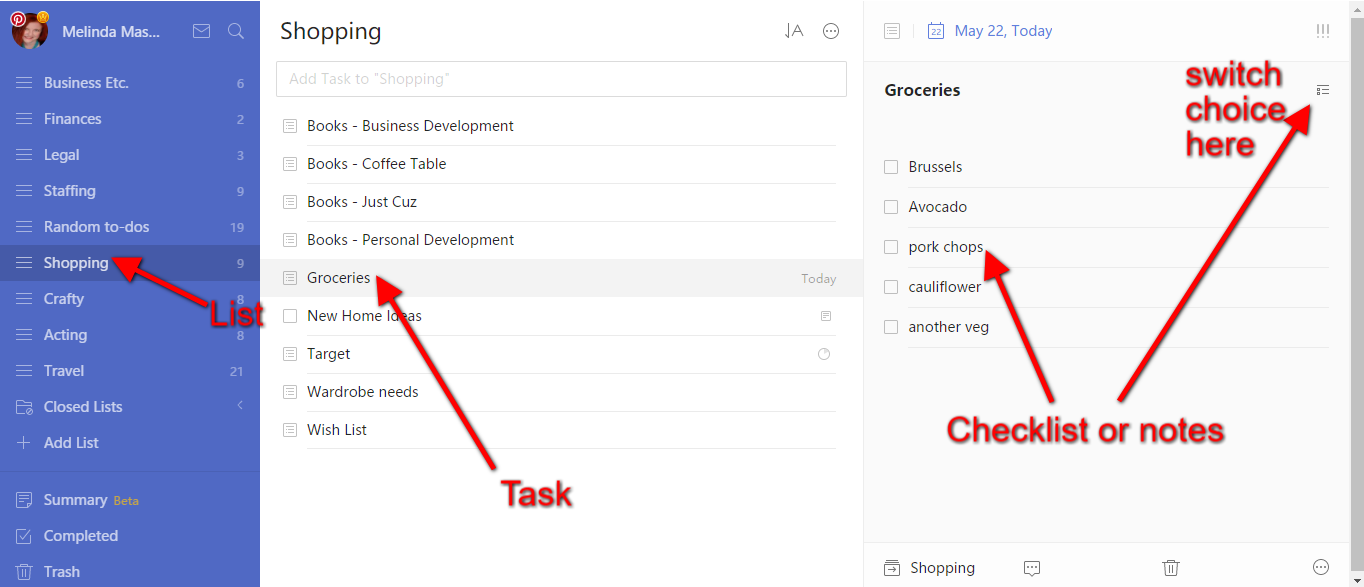
The first competitor is Notion.



Notion is a website that is used for many other things but can be used for organisation like my app. On notion users can create pages and put anything they want on it, for example things to help them stay organised. Users can also make their pages aesthetically pleasing so it suits their taste. For my research I got notion myself and created my own page so I could get an idea of how someone who would use my app would use notion.

In creating my own page, I found that notion has a lot of features, and I believe it stands out so much due to its design. With notion users are given a lot of freedom in terms of design this would be useful to my target as my audience are mainly young girls who enjoy organisation and typically care for the aesthetics of their organisation. However, I did find notion quite difficult to use as I needed a lot of knowledge on how to use the app to be able to use it effectively. It required me to watch multiple tutorials to be able to create a perfect page. So overall notion has allowed me to develop on ideas for the motivation I wish to include in my app as on notion you can add quotes and I want to implement this into my application also; adding your own personal quotes that you can see throughout your day can be inspirational. Additionally given how much I struggled to use notion I want to ensure that my application is easy for users to use and understand and I could potentially add a tour of my app for new users as I think people will be strayed away from my application if they don't understand how to use it and have to do a lot of prior research in order to get the best results out of my app. My app and notion have some of the same features, but I want my app to be very different to notion. To do this I want to make my app easier to use than notion by having built in designs, so users do not need to do it themselves and research how to. Also, the layout of my app will be different as notion is mainly used on devices other than phone as a phone doesn’t provide much space for all the design options notion has.

Another competitor is TickTick.



As a part of my research, I downloaded TickTick myself to see how my client and someone who would use my application would use this app.

Ticktick is a to do list and calendar app. Like notion, ticktick can be used on a mobile device but unlike notion ticktick does not lose as much functionality when on a phone. With ticktick there are a few themes and colour options you can set for the whole app; this would be very useful for my target as it means the app can still be aesthetically pleasing but it does not require as much work as notion does. TickTick has 2 main features, to do lists being the main one and a calendar also being included. I believe ticktick would be beneficial to my audience as it is extremely easy to use and very simple overall. You can easily add a task set priority levels etc. Also, Tick Tick's calendar is connected to a to do list. I believe on a laptop the calendar function is more useful as from the example’s I have seen there is much more detail on a laptop than a phone in the calendar. However, I do think having a calendar connected to a to do list is extremely useful and I will put this in my app. Despite this tick tack would be too simple for my client and target audience as it only has those 2 features. A calendar and a daily to do list. My app will also include timetables/ schedules, motivation and anti-procrastination help. Ticktick only includes the basic functionally of what my app would be, and this would not be enough to satisfy the needs of my client. Although, ticktick ‘s simplicity is also a very beneficial aspect of the app and could even be considered a feature. I want to make my app as simple and easy to use as this app and to do this I will try to implement the tutorial for new users that I previously mentioned.

From my research, these are the main requirements of my solution:

* Allow user to create an account and log in.
* User can make a to do list.
* User can mark tasks as completed.
* Create schedules.
* Recommend activities for a schedule and a to do list.
* Add events to the calendar.
* Review uncompleted tasks.

**Objectives**

Every user must have a Login.

**1.** Users will be asked to create an account

1.1. Users should enter their first name, surname, email, phone number and password.

1.1.1: Users' passwords will be hashed

1.1.2: Users hashed passwords will be stored into the database

1.2. Verification every time a user creates an account:

1.2.1: The first and last name should only include letters 1.2.2: The email must not already exist

1.2.3: Phone numbers should only include numbers and should be 11 characters long

1.3. Users will confirm their account through their email.

1.3.1: Details of the user's information provided will be displayed in the email

1.3.2: They will be directed to simply press “confirm” and will be told to return to the app

1.3.3: Once confirmed user information will be stored into a database.

1.4. New Users will fill out a questionnaire about themselves.

1.4.1: Questions about what the user does for fun and they're age in the questionnaire

1.4.2: All information from the questionnaire will be stored in a database

**2.** Existing users can login

2.1: Users should enter their password and either email address or phone number 2.1.1: Verification every time a user log’s in:

2.1.1.1: The email must already exist in the database

2.1.1.2: The password must match the email

There should be a to do list page.

**3.** Users may add a task

3.1: Once they press the icon to add a task, they will be directed to fill out some information

3.1.1: Users must enter the title of the task for example. Finish NEA analysis 3.1.2: Users may add a description of the task.

3.1.3: Users may add sub tasks for example Finish research, Finish objectives

3.1.4: Users must add a priority level for the task. High, low or medium

3.1.4.1: Task should be coloured based on priority key

3.1.4.2: Tasks should appear on the page in order of priority

3.1.5: Users must add what time they will do the task, add an option for it to be done at any time.

3.1.6: Users must add a difficulty level of the task.

3.1.7: Users may choose to repeat a task every day or week or month etc

3.1.8: Users must press “done" once they've finished created the task

3.1.8.1: Once task is created it should be stored in the database

3.2: Users can tick off completed tasks

3.2.1: Users can press icon next to each task to mark them as completed

3.2.2: The difficulty level will be checked, and appropriate praise will be given to the user

3.2.3: A message should pop up if the difficulty level was hard to say well done and recommend a break or a treat

3.2.3.1: Information from the questionnaire mentioned will be used to recommend activities for their break.

3.2.4: Tasks will not be completely erased but moved to a “completed tasks section”.

3.2.4.1: Users will be given the option to erase these tasks at the end of each day

3.2.4.2: Tasks in the completed section can be moved back to the main to do list at any time

Built in calendar.

**4.**Users can add an event to any day

4.1: Once they press the icon to add an event users will be prompted to fill out some details about the event

4.2: Users must add a date of the event. 4.3: Once they finish adding the event a small dot will appear on the day they have selected to signify that something is happening that day.

4.4: Users must enter the title of the event for example. Programming Exam 4.4.1: Users may add a description of the event.

4.5: Users may add an emotion associated with the event

4.5.1: The emotion will be stored with the event in a database. When the user is reminded about the event the emotion will be checked, and the correct words of encouragement will be offered.

4.6: Users may add what time specifically the event is

4.6.1: Users may choose to repeat the event every day or week or month etc

4.7: Users must press “done" once they've finished created the task

4.7.1: The event should then be stored in the database

**5.** To do list and calendar should be linked

5.1: On the calendar when a specific day is clicked. Users should be directed to the to do list page of that day which will also display all the events on that day.

**6.** Users should be able to add their Schedules/timetables

6.1: When Users are getting set up with my app they will be asked if they have any schedules or timetables, they would like to add

6.2: On the timetable page there will be an Icon to add a new timetable so users can add as many as they would like

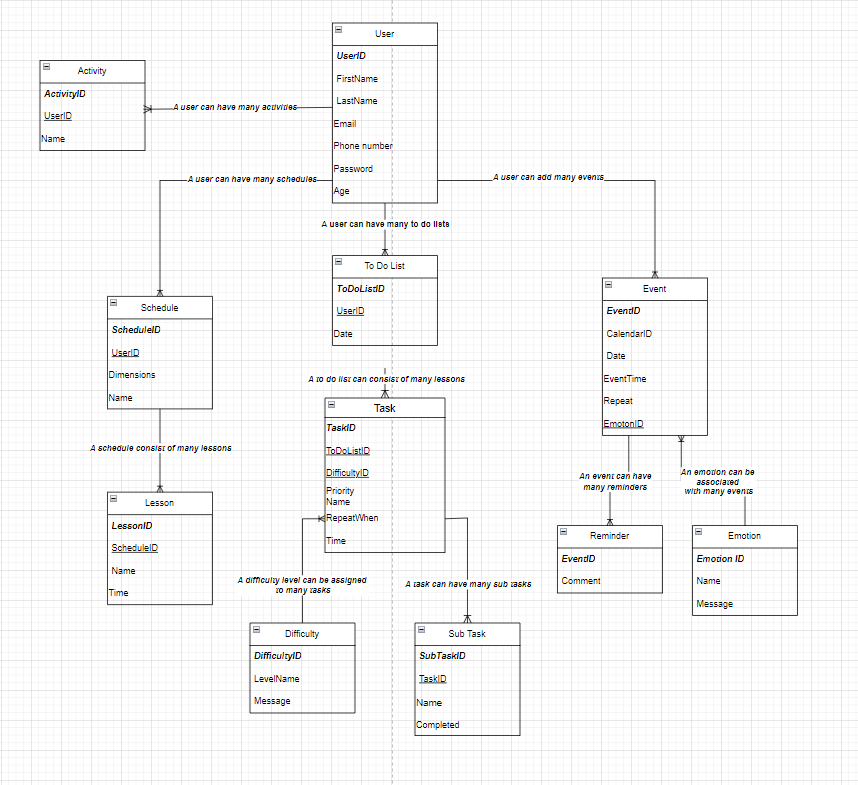
6.3: Users will be asked the dimensions required for their timetable(eg,4x4)

6.4: Users will be able to insert as much detail of their timetable as they would like by pressing any square within the table

6.5: Users must press “done” when they have made their timetable to ensure it is stored in the database

**Models**

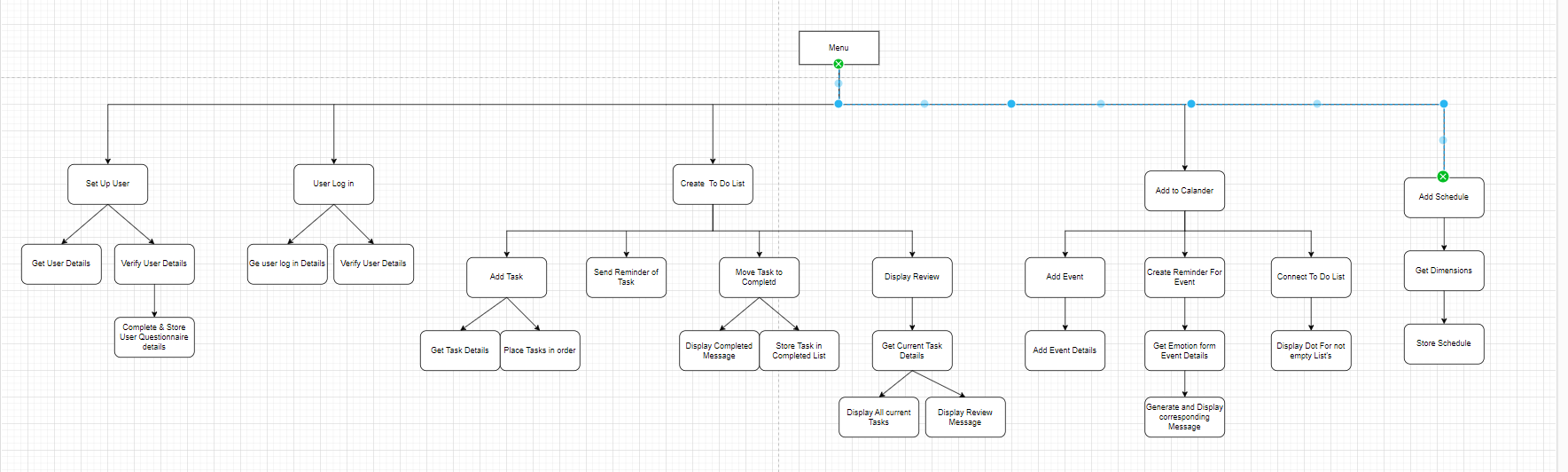
Entity Relationship Diagram



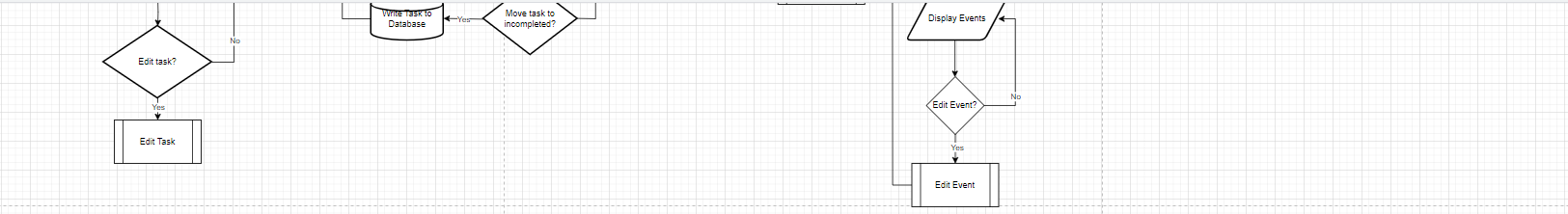
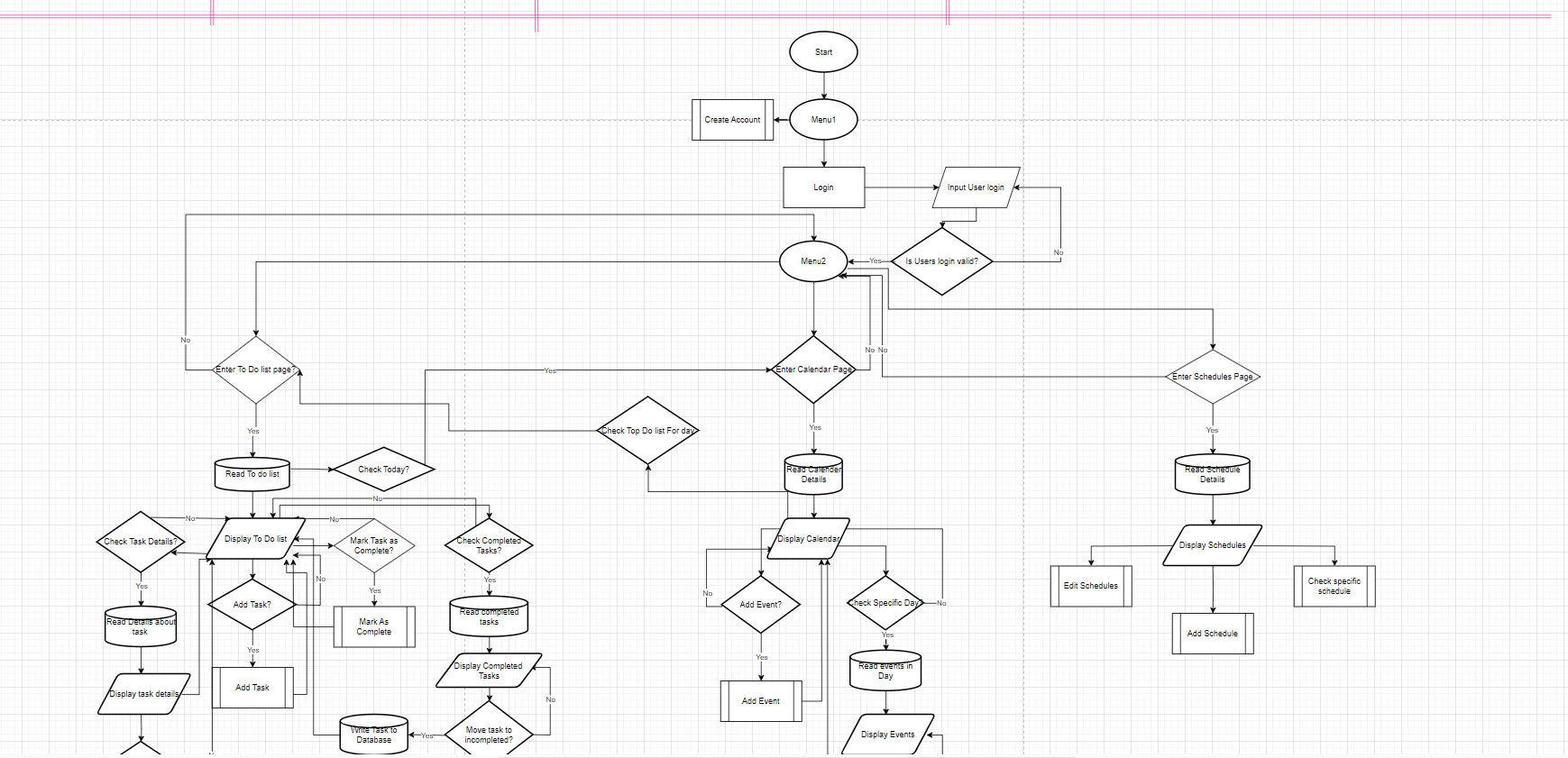
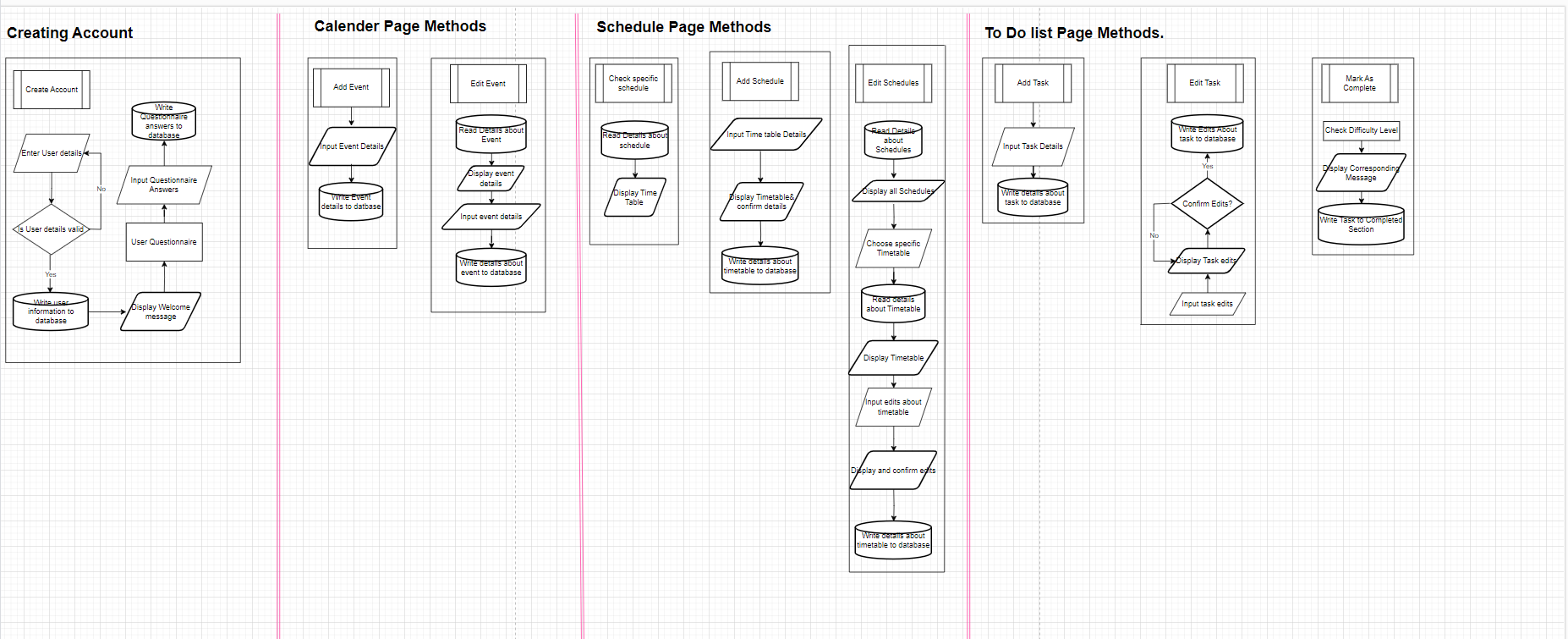
As most of my entities are related to the user a began by making the User table. A user can have many activities, these are received in the questionnaire when a user signs up. An activity is something a user enjoys doing for example reading. The reason the activity has its own table is because each user enters 3 activities so I could not simply put the activity name in the users table as its more efficient to represent this as a one-to-manty relationship. The user can have many schedules/timetables. For example, they could alternate between 2 different timetables every week. These schedules consist of many lessons. The lessons are small portions of the entire schedule which when put together make the whole schedule. The user may also have many to do lists, one for each day, which are made up of many different tasks. All tasks are assigned a difficulty level and can consist of many subtasks. The user can attend many events such as an exam and hey can set many different reminders for this exam. A reminder a month before and one a week before etc. A user can associate each event with one emotion, chosen out of 10. This emotion can be used to describe the users feeling towards many different events.

**Design**

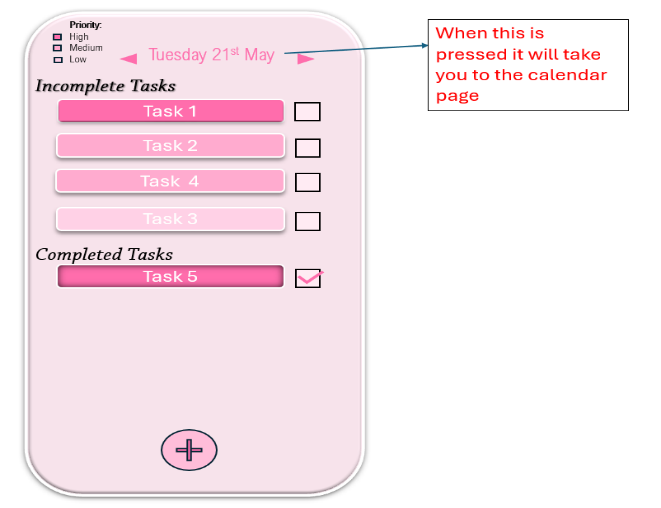
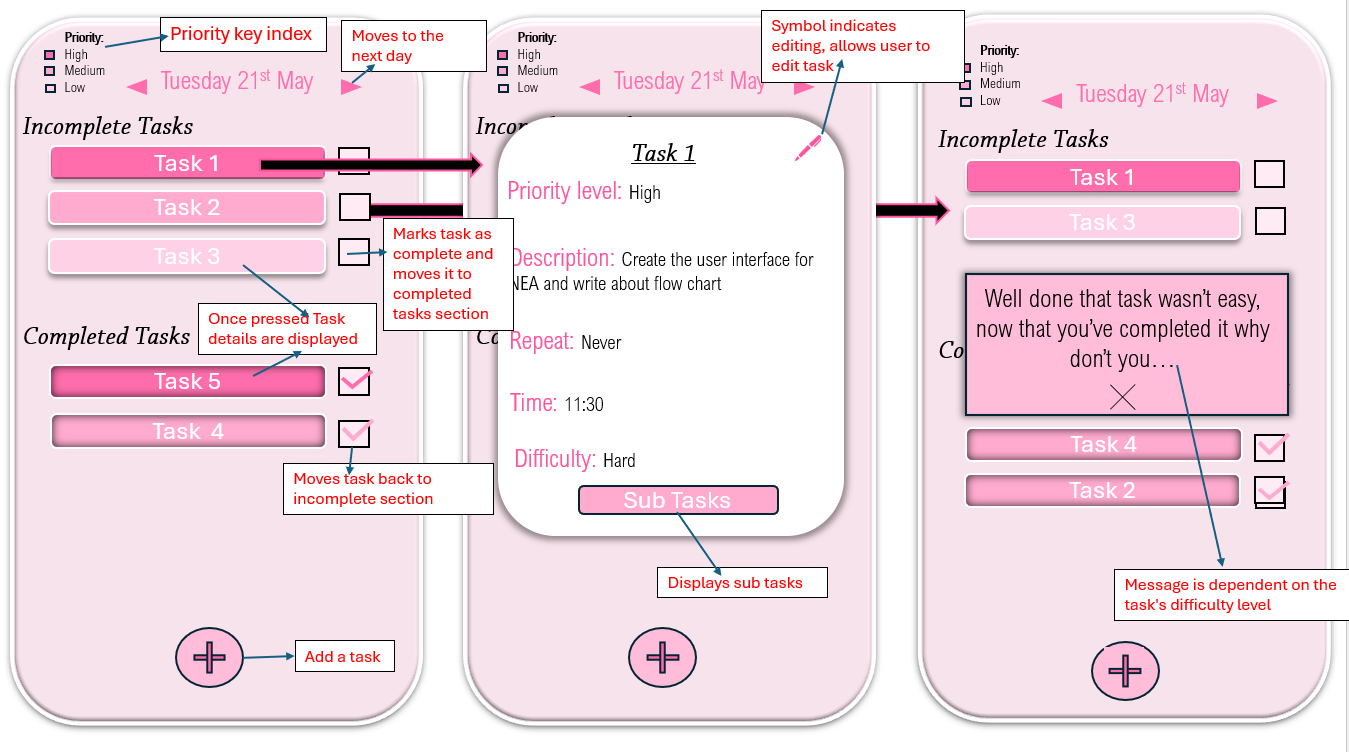
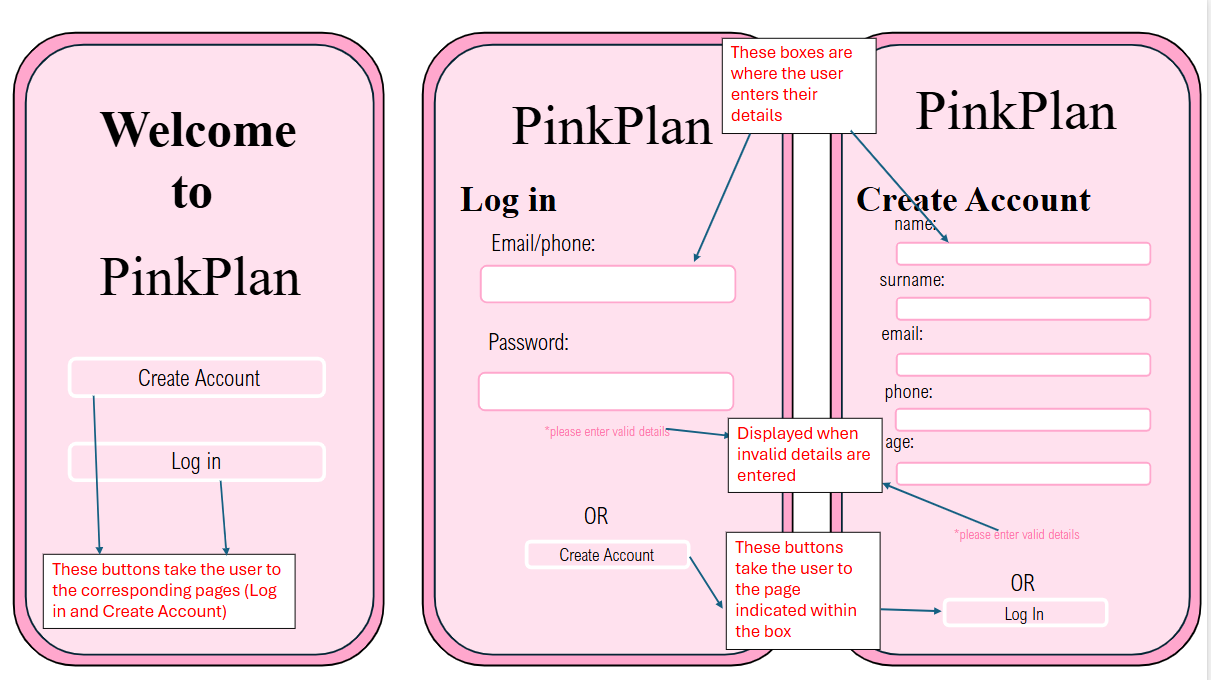
Hierarchy Chart

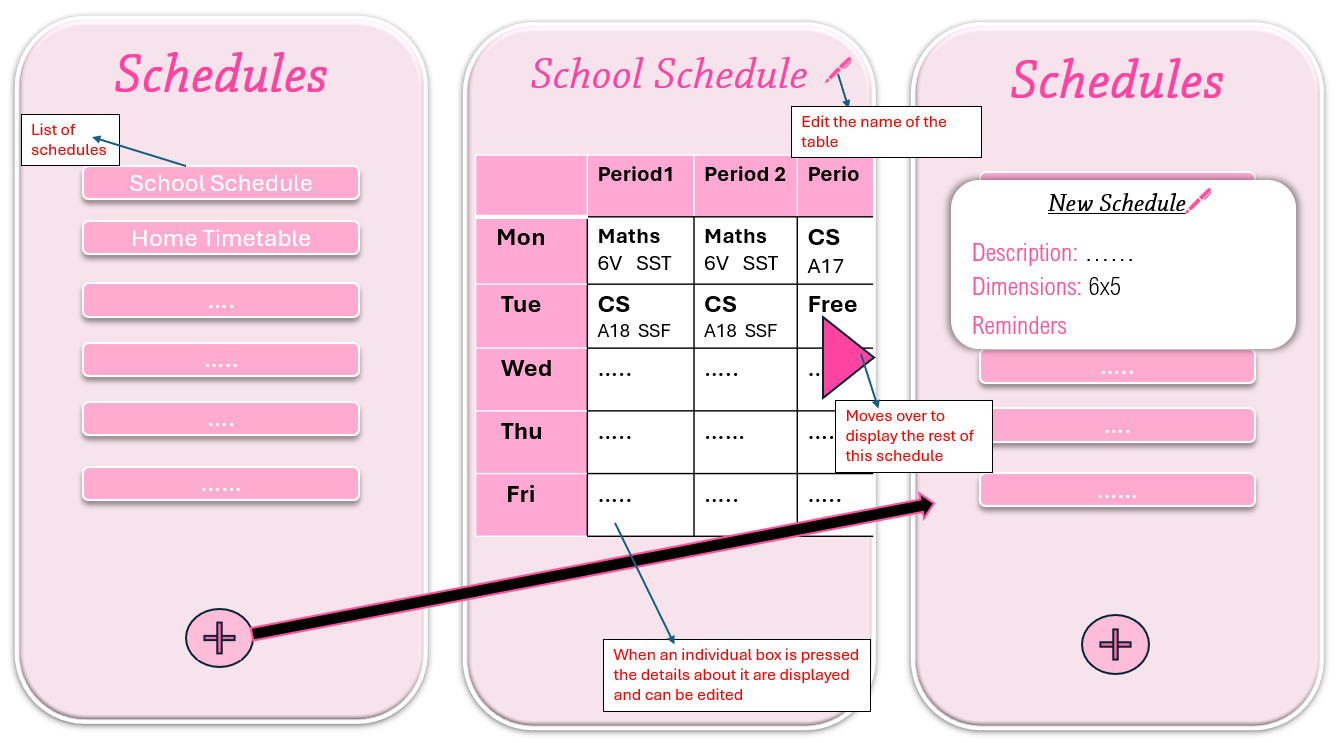
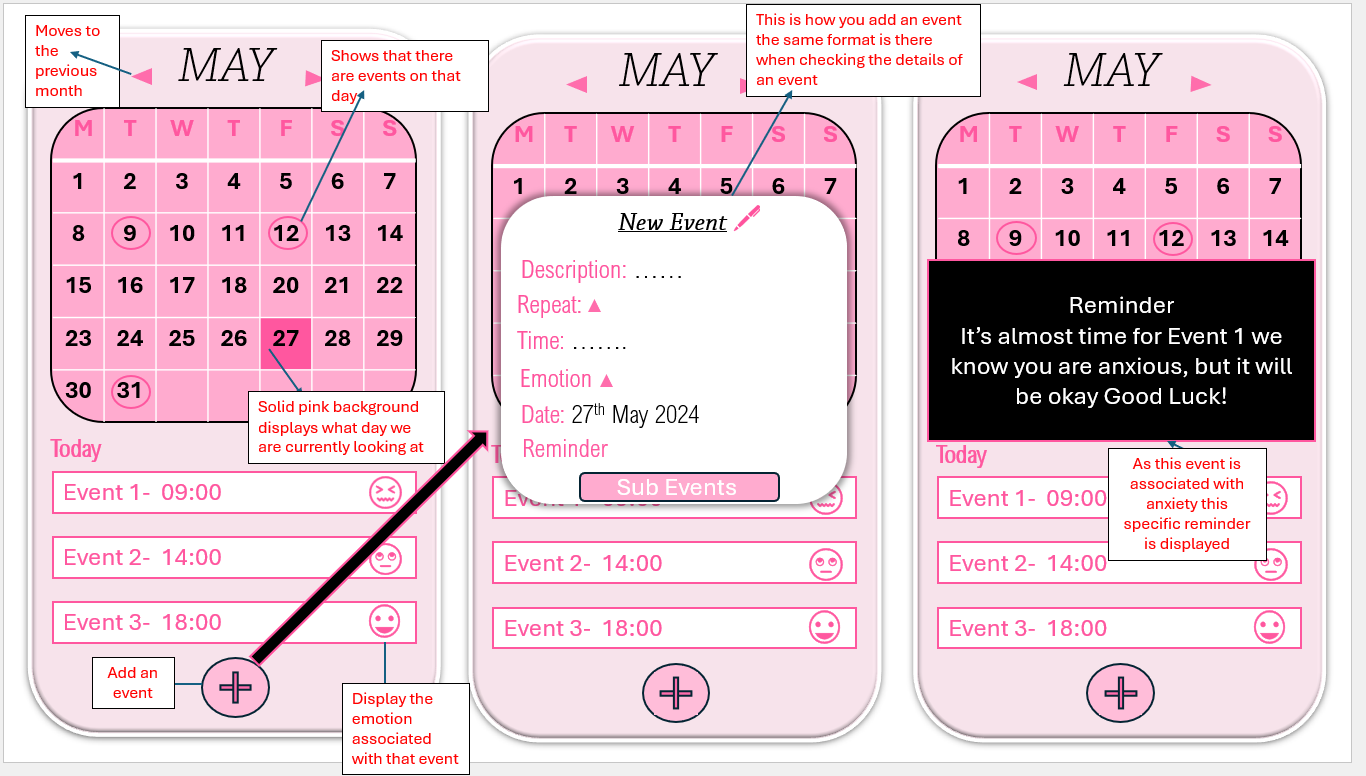


My hierarchy chart shows all the subprograms within my app. Firstly, a user can set up their account on the sign in page within this the user inputs the details I mentioned in my first objective, once the details have been verified the user will complete a questionnaire about themselves. Then the user can log into their account where their log in details will be compared with the ones they signed up with. The to do list page will act as a home page for the app. Displaying all the user's tasks for the day and tasks they have completed. On the to do list page users can add tasks and mark tasks as completed. At the end of each day, they can review all the tasks that they have completed and reflect on the ones they did not complete. Within the to do list page there will be ways to access all the other areas within the app. At the top of the to do list the date will be displayed. This will be a button that can take the user to the calendar where they can add events, create reminders for these events and see all the days that have events and tasks as these days will be marked. From the to do list subprogram users can choose to access the schedule page where they can add many different schedules.

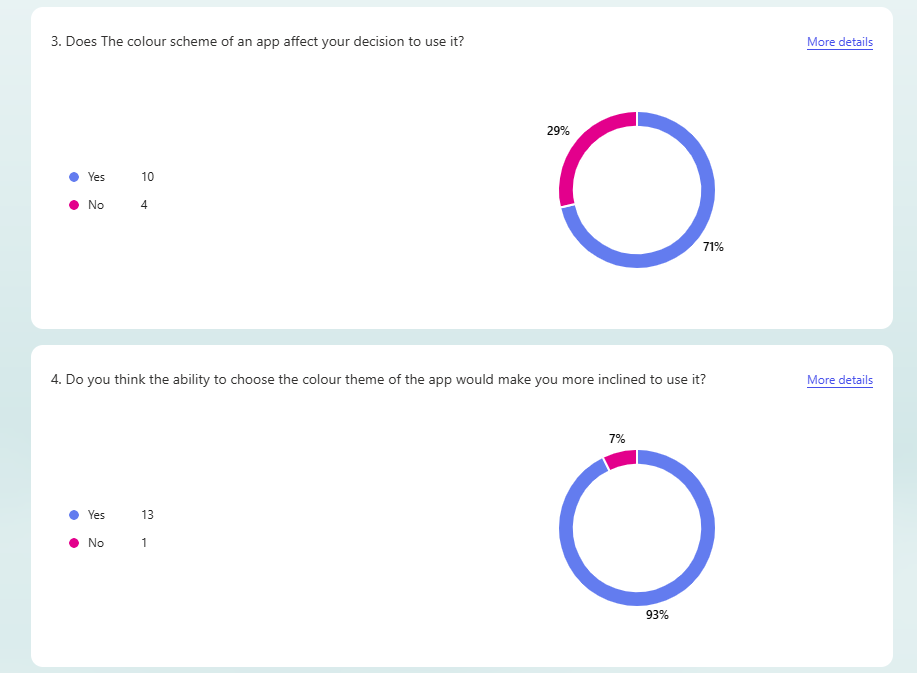
Flow Chart

The flowchart above further expands on my hierarchy chart. It begins with a subprogram where users can create an account. The user can enter their details and if they are valid their information will be stored in the database. A welcome message is displayed, and the user then fills out a questionnaire, details of this questionnaire are stored in the database. The user then logs in and their details are checked for validity, if the password is stored in the database or not. The user then enters the menu where they can open the to do list, calendar or schedule page. If the user opens the to do list page the to do list is read and the to do list is displayed. The user can choose to check details of tasks or view the completed tasks. Both involve reading from the database and displaying necessary information. When viewing task details, you can choose to edit these. The user inputs their edits and can confirm these, once confirmed the new task details replace the old ones in the task table. Users can also move tasks to the completed section. Within this subprogram the difficulty level of the task is read, and the appropriate message is displayed. This task is then written to the completed table in the database. Users may also add a task in this subroutine the user is prompted to input details about the task and these details are written to the task table. Users can click the date on the to do list to view further details about this day. Clicking this button takes users to the calendar page. You can add and edit events. All functions involve inputting details and writing these to the database. To view specific days the events in the selected date are read and displayed. In the schedule page stored schedules are read and displayed. Users can choose to add specific schedules. This subroutine takes user inputs, confirms them with the user and stores them into the database. To check specific schedules the subroutine reads details about the selected schedule from the database and display these. Lastly users may edit their schedules. The subprogram reads details about the schedule and displays the schedule. The user can then enter their edits, these are confirmed with the user and are updated in the database. The updated schedule is displayed.

**Interface design (HCI)**

I have decided to design my app like this because based off my research below the colours and aesthetics of an app is important to my target audience. So, it was important to heavily use the colour voted most appealing to my audience, and to make it aesthetically pleasing. It’s also important for my design to be quite self-explanatory so users won't have an issue understanding how to use my app so unlike notion my competitor they won't have any trouble using the application and therefore won't be inclined to not use it. To make my design self-explanatory I used common symbols and clear titles. For example, the plus symbol commonly indicates the addition of a new feature for example a new schedule in an app. Also, the pen symbol I use on the pages dedicated to adding a schedule or adding or editing a task in a to do list is commonly used to represent something you can edit within an app.

The data blow shows that 71% of my target audience, students aged 15 to 18, and my client Nebras believes the colour scheme influences their decision to use an app. This confirmed that the colour design is important. 93% of my target audience said if they could choose the colour scheme in an app, they would be more inclined to use it. So, I allowed my target audience and client to vote which colour design is the best.



I carried out a survey asking people what colours are more appealing for my app. The colours surveyed were pink, purple, blue, black and red. Overall Pink received the highest rating with an average of 7.14/10. Blue was 2nd, purple third, red fourth and black last. From this survey I have concluded that the original colour theme for my application will be pink. Initially I was going to make the original design the regular black and white. However, through my research I asked my target audience and client if the colour of an application affects their decision to use it. 70% of respondents said it does so for this reason I will use pink as my original design.



Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name | Field Name | Data Type | Field Length | Constraint | Description |
| tblUser | UserID | INT | 4 | Primary Key | Unique User Identifier,  Users are ordered so first userID = 0001 |
|  | UserFname | CHAR | 20 | Not Null | Users First Name |
|  | UserSname | CHAR | 20 | Not Null | Users Surname |
|  | Password | VARCHAR | 20 | Not Null | Users Password to get into the app |
|  | Phone | INT | 11 | Not Null | Users Phone Number |
|  | Email | VARCHAR | 20 | Not Null | Users email |
|  | Age | INT | 2 | Not Null | Users Age |
|  | Verified | BOOLEAN | 5 | Not Null | If the users account has been verified or not |
| tblQuestionnaire | LeisureID | INT | 10 | Primary Key | Unique activity identifier |
|  | Residence | CHAR | 20 | Not Null | Users' country of residence |
|  | YearGroup | VARCHAR | 20 | Not Null | The year group the user is in |
| tblActivity | ActivityID | INT | 10 | Primary Key | Unique identifier for the user’s activities |
|  | UserID | INT | 4 | Foreign Key | Connects the |
|  |  |  |  |  |  |
| tblToDoList | TodoID | INT | 10 | Primary Key | Unique list identifier |
|  | UserID | INT | 4 | Foreign Key | Identifies what user owns the todo list |
|  | CalendarID | INT | 5 | Foreign Key | Allows the to do list to be connected to the calendar |
|  | Date | DATE |  | Not Null | Date to do list is to be completed on |
| tblTask | TaskID | INT | 6 | Primary Key | Unique task identifier |
|  | TodoID | INT | 10 | Foreign Key | Identifies what to do list the task belongs to |
|  | ActivityID | INT | 10 | Foreign Key | Identifies the activity from the questionnaire associated with the task |
|  | DifficultyID | INT | 10 | Foreign Key | Identifies the difficulty level of the task |
|  | Priority | CHAR | 20 | Not Null | Priority level of the task |
|  | RepeatWhen | VARCHAR | 20 | Not Null | When the task should be repeated |
|  | Time | VARCHAR | 20 | Not Null | What time the task should be done |
|  | Completed | Boolean | 5 | Not Null | If the task is completed or not |
| tblSubTask | SubTaskID | INT | 20 | Primary Key | Unique subtask identifier |
|  | TaskID | INT | 6 | Foreign Key | Identifies what task the subtask belongs to |
|  | Name | VARCHAR | 20 | Not Null | Name/ description of subtask |
|  | Completed | Boolean | 5 | Not Null | If the subtask is completed or not |
| tblCompletedTask | TaskID | INT | 10 | Foreign key | To access the details about the task |
| tblDifficulty | DifficultyID | INT | 10 | Primary Key | Unique difficulty level Identifier |
|  | LevelName | CHAR | 20 | Not Null | Name of difficulty level |
|  | Rank | INT | 1 | Not Null | Level of difficulty in comparison to other levels |
|  | ActivityID | INT | 4 | Not Null | Activity ID so appropriate leisure can be recommended to user |
|  | Message | VARCHAR | 100 | Not Null | Message displayed when task is completed |
| tblCalendar | CalendarID | INT | 10 | Primary Key | Unique Calendar Identifier |
|  | UserID | INT | 4 | Foreign Key | Determines which user owns that calendar |
| tbEvent | EventID | INT | 10 | Primary Key | Unique event identifier |
|  | CalendarID | INT | 10 | Foreign Key | Identifies what calendar & user the event belongs to |
|  | Name | VARCHAR | 20 | Not Null | Name of the event |
|  | Date | DATE |  | Not Null | Date of the event |
|  | EventTime | VARCHAR | 20 | Not Null | Time of the event |
|  | Repeat | VARCHAR | 20 | Not Null | When the event is repeated |
|  | EmotionName | INT | 10 | Foreign Key | Users' emotion associated with the event |
| tblEmotion | EmotionName | VARCHAR | 20 | Primary Key | Name of Emotion, unique identifier |
|  | Message | VARCHAR | 100 | Not Null | Message displayed when reminding user about event |
|  | Message2 | VARCHAR | 100 | Not Null | Message Displayed once event has passed |
| tblSchedule | ScheduleID | INT | 10 | Primary Key | Unique schedule id |
|  | Name | VARCHAR | 100 | Not Null | Name of the schedule, can describe what it’s for |
|  | UserID | INT | 4 | Foreign Key | Indicates which user the schedule belongs to |
|  | ActivityID | INT | 10 | Foreign Key | Gets the users activity table to be able to recommend activities in their free time |
|  | Dimensions | VARCHAR | 5 | Not Null | Dimensions of the schedule/timetable |
| tblBox | BoxID | INT | 10 | Primary Key | Unique identifier for a box within the timetable |
|  | Description | VARCHAR | 50 | Not Null | Everything written inside the box |
|  | ScheduleID | INT | 10 | Foreign Key | Indicates what schedule the specific box belongs to. |

IPSO

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | PROCESS | STORAGE | OUTPUT |
| User enters their details (name, email, password etc) | Check if details are valid. Correct length, valid email domain etc. Then create a new user. | Store to User table in database | Display welcome back message |
| User inputs answers to the questionnaire |  | Store to the Questionnaire table in database, include users ID as a foreign key | Carry out tour of the application. |
| User inputs login details, email phone number and password | Check if login details are valid and match | Temporarily stores the details entered so they can be compared to the previously provided user information | Displays welcome message |
| User Inputs details about a new task, name, priority, subtasks, difficulty... | Check if all necessary details have been provided/ filled out | Stores the new task in tblTask | Displays the newly created task in the to do list. |
| User enters task edits | Shows user the edits they made to the task and checks if the edits are correct | Stores the edits by replacing the old details about the task with these new ones | Displays the edited task in the users to do list` |
| User clicks the tick box next to a task to mark it as complete | Check the difficulty level that has previously been associated with this task | Stores the task to the completed task table. | Tick box next to task now has a tick within it to illustrate that is has been completed. Task is now in the completed tasks list. |
| User inputs the details of a schedule/timetable | Table is confirmed with the user by presenting their inputs back to the. | Stored in tblSchedule in the database | Table is displayed and user is prompted to add more details such as specific lessons. |
| User selects specific schedule to edit and inputs their edits | Edits are displayed back to the user | Edits are stored in replacement of what is currently stored in the database for that schedule | The edited table is displayed to the user |
| User inputs details about an upcoming event in their calendar | Details are confirmed with the user by presenting their inputs back to them. Checks if all necessary details have been provided | Event is written to the database and is stored in the calendar. | Circle is displayed around the date of the event to show there is an event on that day. |
| User inputs password to create an account | Hashing algorithm converts password into a value | Value stored in database so it can be used for comparison when the user logs in |  |

Designing SQL Statements

**Selecting the user id**

SELECT UserID FROM tblUser

WHERE Email = email

‘email’ is the email the user used to log in to their account. This SQL Statement selects the user ID that is stored with the email the user logged in with to retrieve the primary key, the user ID.

**Selecting the password**

SELECT Password FROM tblUSER

WHERE Email = email

This statement selects the hash value of the password that is stored with the email the user entered to log in. This is used when the user's password input is hashed with the same algorithm and the 2 numbers, are compared to confirm the user entered the correct password to log in.

**Inserting the users sign up information into the database.**

INSERT INTO tblUSER (Fname,Sname,Email,Phone,Password)

VALUES (fName,sName,email,phone,pass)

This inserts the users' details into the database. The first line specifies the fields the data should be stored in so in the second line I used different names to define the data or values that will be stored in those fields to avoid confusing myself when programming with these. I will do this throughout my program for example I will name the users original plain text password differently to the hashed version of their password. So, while I am programming with these variables, I can understand what state the data is in.

**Inserting the userID into other tables as a foreign key**

INSERT INTO tblActivitiy (UserID)

VALUES (userID)

The primary key of the user table, UserID connects the user to all their to do lists, schedules and activities. So, this is used as a foreign key in almost all the tables in my database. I use this foreign key to identify that unique user's data which ensures they see their own data (to do lists and schedules) when they open their account.

**Selecting the priority level of tasks**

INSERT INTO tblTask(ToDoID)

Values(todoID)

SELECT Priority FROM tblTask

WHERE ToDoID = todoID

AND UserID.tblToDoList = userID.tblToDoList

The tasks are ordered based on their priority level so when displaying the users to do list I select the priority levels of the tasks that belong to that user. I select the priority level using “WHERE ToDoID = todoID” to ensure the correct to do list and tasks are being displayed. Then using “WHERE UserID.tblToDoList = userID.tblToDoList” UserID is the foreign Key in the to do list table that holds the corresponding users ID.

Selection of hardware and software

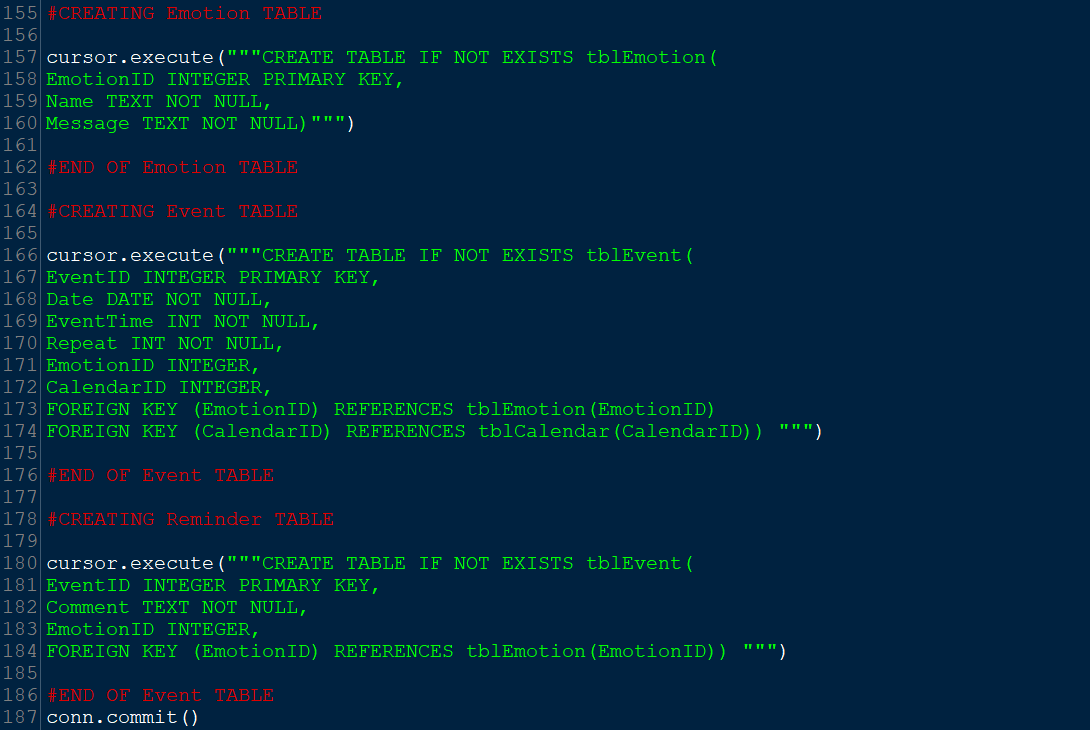
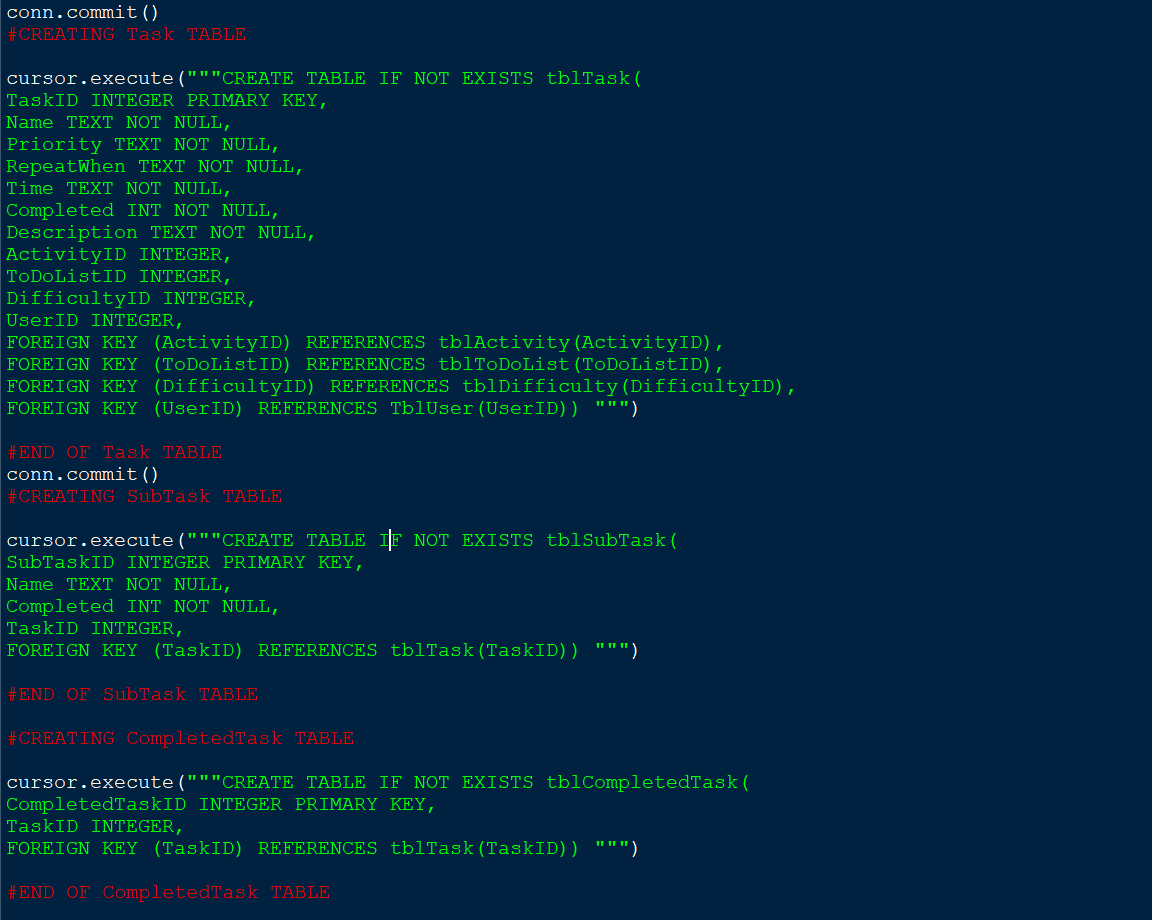
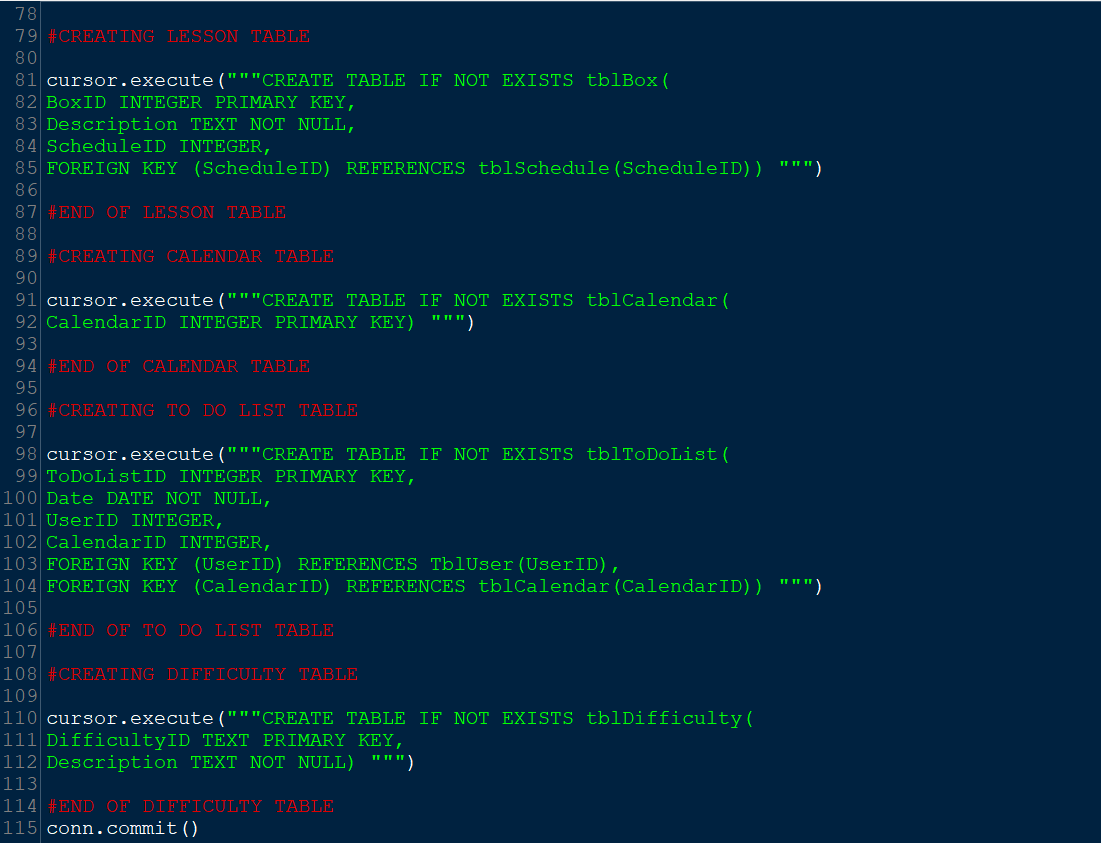
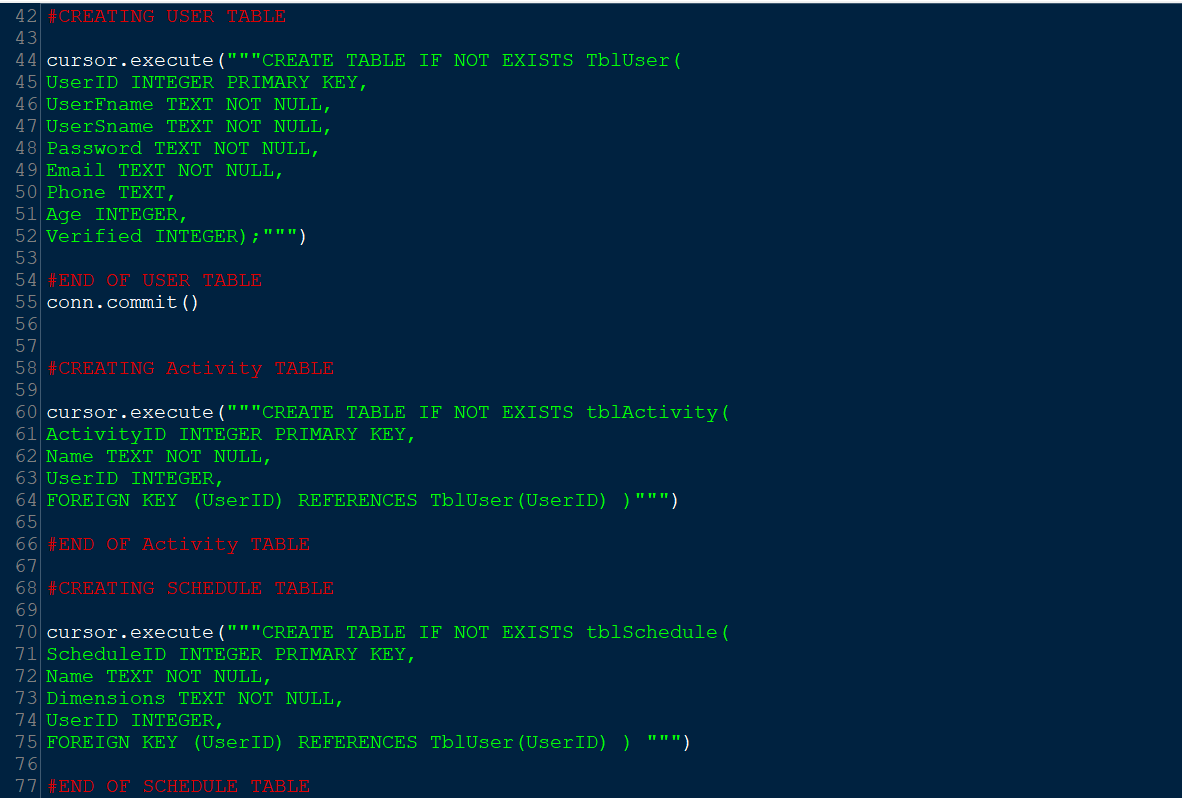
I decided to use python to create my app, I am very familiar with python making it the most efficient programming language to use as I will not need to spend time learning the basics and python can still effectively create my app. The Tkinter library will be used to create the front end as it allows me to have full control of the colours and overall appearance of my app which plays a large part in the selling point of PinkPlan. Tkinter uses quite a basic format for creating widgets buttons and labels which will be easy for me to memorise and adapt to suit each aspect of my app. This allows for a lot of repetition for example the different frames I use for functions such as task details or add task will be the same size meaning I do not need to recreate these frames; I can use them in multiple subprograms. Tkinter uses get functions so I may store the inputs into widgets as variables that I can then use in algorithms such as my hashing algorithm where I use my previous python knowledge to create this. Using this library is favourable for my users as it allows me to create a simple design making it easy for people to use my app and optimize the benefits of PinkPlan. Tkinter uses a lot of repetition visual studio code will be the best software to code my app on as it allows me to select multiple elements with the same name and edit them all at the same time.

I will use the SQLite3 library to crate my database as I am familiar with the format SQL statements are written in so it will be easy to translate the SQL Statements, I have designed into the same format that the SQLite3 library uses. When researching the different SQL libraries, I could use with python such as PostgreSQL, MySQL and SQLite. SQLite was the easiest to understand and can store all my user's information clearly. I comprehended how to program with SQLite and understand the purpose of features such as cursor and conn.

**Implementation**

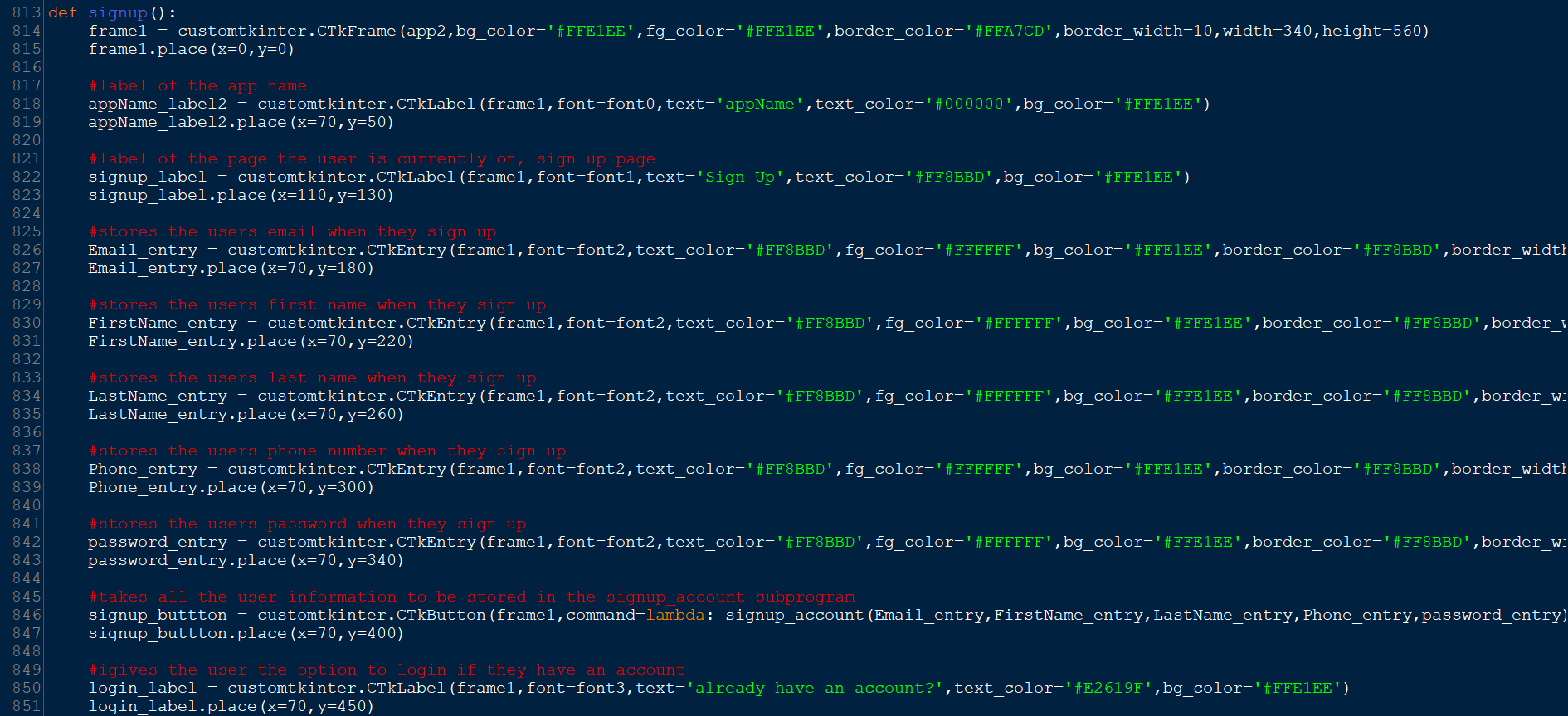
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Test Num |  | Test Input | Expected output | Actual output | comment |
| Users Can Create Account and Enter Details | 1 |  | [‘testing1@gmail.com](mailto:‘testing1@gmail.com)  Yetunde  Adeniji  07383441179  Test1’ | ‘Account Created’  Input displayed as stored in user table |  |  |
|  | 2 | Phone Number must be <11 character. | 07123456789012 | ‘Error’- 'Your phone number should be 11 integers' |  | There is an if statement using the len function to check if the length is valid |
|  | 3 | Phone Number Must only contain letters | 07A12345678 | ‘Error’- 'Enter a valid phone number’. |  | There is an if statement using the isdigit function |
|  | 4 | Name Should Include letters only | Y3tund3 | 'Error'-'Enter a real name' |  | There is an if statement using the is alpha function. |
|  | 5 | Can not be blank | ‘ ‘ | ‘Error’-’Enter all details’ |  | An if statement prints this error if even one field is blank |
|  | 6 | Can not use same email as another user | 'testing1@gmail.com’ | ‘Error’-’Account Already exists’ |  |  |
| User can enter activities in the questionnaire | 7 |  | ‘Ice Skating’’Watching Shows’  ‘Going on walks. | Activities will be shown when information stored in database is printed |  | The Store activity subprogram stores the activities into the database when Add activity is pressed |
| Entering an age in the questionnaire | 8 |  | 17 | Age will be shown when the information stored in the user table is printed |  | The Store age subprogram stores the age into the database using an update statement when the enter button is clicked. |
| Users must enter a valid password when logging in | 9 | Password must be 10 characters or less | ‘sparkling\_Donkey1’ | ‘Error Password must be 10 characters or less |  |  |
|  | 10 | Passwordsare hashed to produce a value when account is created | ‘IceCream8’ | Account Created and numbr is generated |  | ‘IceCream8’ hashes to 32 |
|  | 11 | If a hashed password is the same as the ones already stored the number is changed | ‘IceCream8’ | Account Created and 33 |  | IceCream8 hashes to 32 but because 32 is already stored 1 is added |
| User can enter email and password to log in | 12 |  | [‘testing1@gmail.com’](mailto:‘testing1@gmail.com’)  ‘Test1’ | Successfully Logged in |  | Email is compared with what’s stored in the database password is hashed and compared. |
|  | 13 | The user can’t log in with the wrong email | [‘WrongPass@gmail.ccom’](mailto:‘WrongPass@gmail.ccom’)  ‘Test1’ | ‘Email does not exist’ |  |  |
|  | 14 | The user can’t log in with the wrong Password | [‘testing1@gmail.com’](mailto:‘testing1@gmail.com’)  ‘wrongPass’ | ‘Invalid Password’ |  | The input is hashed and compared with the number stored in the database |
|  | 15 | The user can not log in with blank entries | ‘Empty  ‘Test1’ | ‘Enter all Data’ |  |  |
| The user can switch between the log in and sign-up page | 16 |  | Click ‘Create Account’ on log in page | Sign up page appears |  | The button calls the sign-up subprogram |
|  | 17 |  | Click ‘login’ on sign up page | Log in page appears |  | The button calls the login subprogram |
| A key is displayed at the top of the to do list | 18 |  | Click  ‘High’ | Popup-These tasks are Very important’. |  | Messagebox function displays this when button is pressed |
|  | 19 |  | Click  ‘Medium’ | Popup-These tasks are mildly important’. |  | A subprogram is called when button is pressed |
|  | 20 |  | Click ‘Low’ | Popup-These tasks are not very important’. |  |  |
| Users can open a calendar | 21 |  | Click the date | Pop up calendar |  |  |
| User Can add a task to the to do list | 22 | When add task is pressed the user can enter details | Click +New Task | Pop up displaying information user can add |  | Add task subprogram is called the placeholder text’s prompt what the user should enter |
| The task is coloured based on its priority level | 23 | Tasks that are high priority are hot pink | Create Task With priority level ‘High’ |  |  | select statement fetches the already stored tasks and displays the tasks |
|  | 24 | Tasks are stored in the database | Select all the tasks from the Task table | List of all tasks including ‘Maths HW’ |  |  |
|  | 25 | Tasks that are medium priority are pink. | Create Task with priority level ‘Medium’ |  |  | An if statement is used to display the correct colours |
|  | 26 | Tasks that are low priority are baby pink. | Create Task with priority level ‘Medium’. |  |  |  |
| The User can check the details of a task when they click on it | 27 |  | Click on ‘TestNEA’ | Pop up of task details |  | A select statement gets all the information about the task and displays it |
| The task is removed from the incomplete task section when complete task is pressed | 28 |  | Click Complete Task | TestNEA is removed from the incomplete Tasks section |  |  |
|  | 29 | Completed Tasks are deleted from the task table | Click Complete Task | TestNEA is not in the list of tasks stored in the database |  |  |
| The x button takes the user back to the page they were previously on | 30 | The user can go from adding a task to the to do list page | Click X on the add task pop up | The to do list page is displayed |  | The x button calls the to do list subprogram, so it appears like they go back |
|  | 31 | The user can go from checking task details to the to do list page | Click X on the task details pop up. | The to do list page is displayed |  |  |

**Creating Database Using SQLite3**

 I used SQLite3 to create my database as this was easy to implement and I found the SQL queries easy to execute, my database reflects my entity relationship diagram. However, when coding I made a change to my database which I will discuss in my evaluation. I also used my data dictionary when selecting all the data types for my database. I began with creating the user table as this table holds the data that separates all the data held in the other tables by which user they belong to. Then added the user id as a foreign key in all the necessary tables connected to the user table. The table box represents the lesson table in my entity relationship diagram I changed the name as my users’ schedule may also consist of activities.

**Objective 1.1. Users should enter their first name, surname, email, phone number and password. Users Passwords will be hashed and stored into the database.**

User interface for objective 1.1

The above screenshot shows me creating the graphical user Interface for the sign-up page. As I previously conducted a questionnaire about the aesthetics of my app it is important all my colours matched this and my design. For this reason, I used Tkinter for my interface as it allows me to have full control of the colour of every part of my app and I am previously familiar with this library. I use the lambda function, so I don't need to globalise all the user's inputs. It allows me to pass the inputs through to the next subprogram so they can be stored, and the password can be hashed.

**Entry Widget:** I use the **CTkEntry** function to collect user inputs. For example, **Email\_entry** is used when the user inputs their email to sign up. A place holder text is used to prompt the user's input. Throughout my code I often globalise these variables as I store the user inputs in separate subprograms.

**Button Widget:** I use the **CTkButton** function as an enter button throughout my program. When a user presses these buttons, they are taken to a separate subprogram called using command = “subroutine name”. For example, the **signup\_button** command is **signup\_account**. So, then the signup\_button is pressed the signup\_account subprogram is called and carried out.

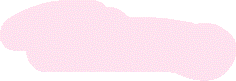
**Label Widget:** I use the **CTkLabel** to display titles and information. For example, **signup\_label** is used to display the title of the page, so the user knows the function.

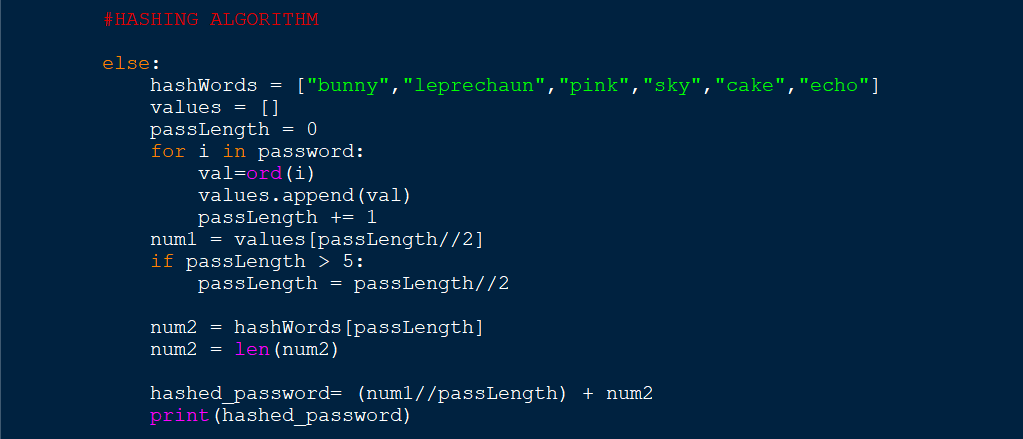
**OUTPUT:**

Screens screenshot of a login screen

AI-generated content may be incorrect.1.1.1: Users' passwords will be hashed

PinkPlan



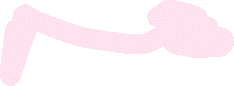
Once a user clicks” create account” and all their sign-up information and it is all valid, the password is hashed. The program iterates through each character in the user's plaintext password. For every character, the program will add the associated Unicode value to an empty array called **values** and then records how long the password is by incrementing the variable “pass length” by 1. Afterwards the middle number in the array of Unicode values is selected and stored in a variable. If the password length is greater than 5 it is divided by 2. A random word is selected from the list of hash words using an index of the new, halved password length. The length of this word is then stored. Finally, the hashed value is created by dividing the middle value in the list of Unicode values by the length of the user's password then adding the length of the random word selected from the hash Words.

**OUTPUT:**

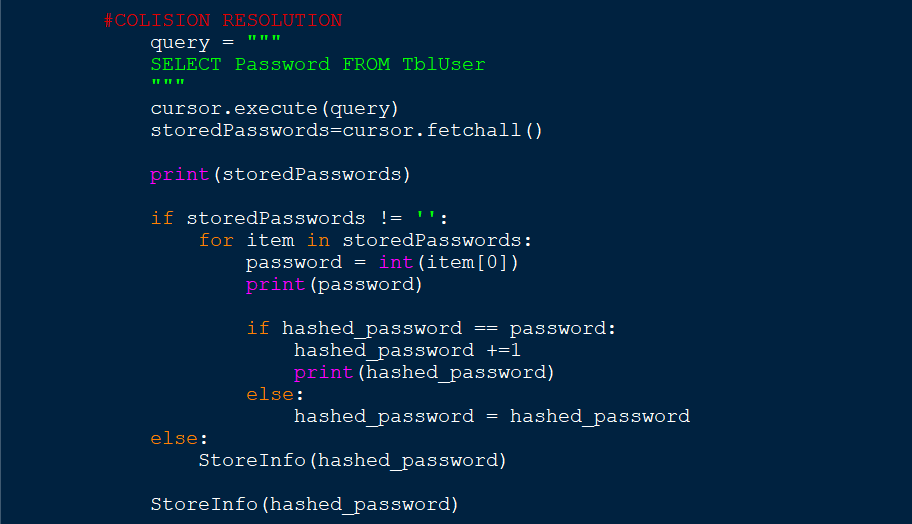
A screenshot of a computer screen

AI-generated content may be incorrect.

PinkPlan

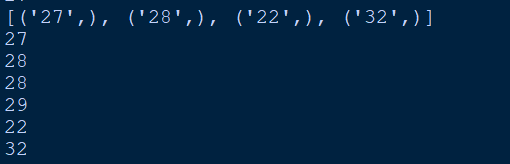


Test1 hashes to 27

If 2 of my user's plain text passwords create the same hash value, I have collision resolution to ensure that no same 2 numbers will be stored in the user table. This is done by selecting all the passwords currently stored in the User table in my database and storing this as a variable, **storedPasswords**. A for loop increments through items in this variable and compares the user's hashed password with all the currently stored passwords. If the users hash value is the same as any in the database, the program adds 1 to the password. The loop continues and adds 1 until the user's hashed password is different to all the ones currently stored in the database.

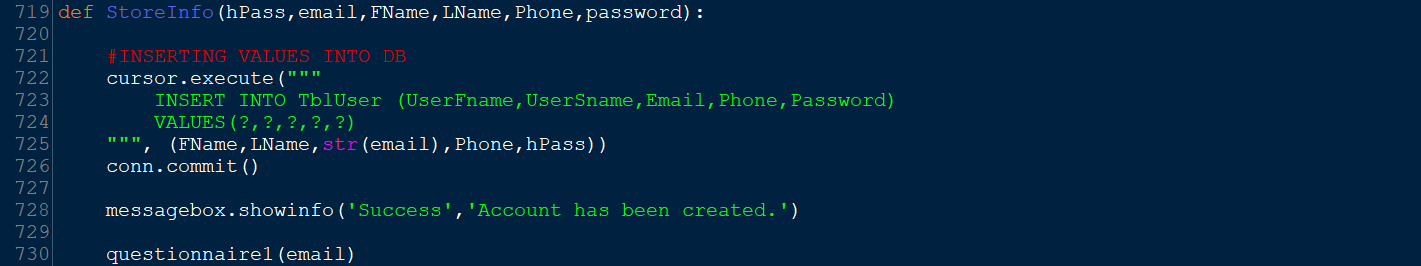
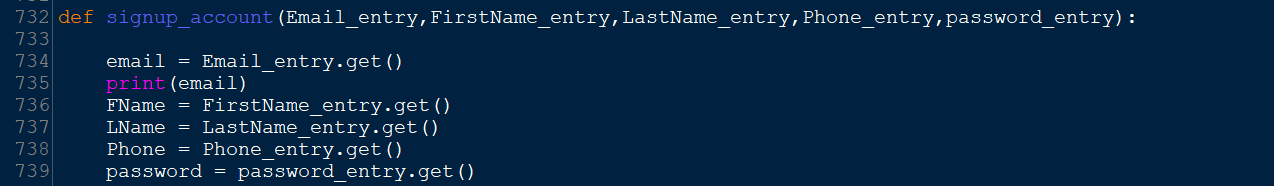
**OUTPUT:**

**A screenshot of a sign up form

AI-generated content may be incorrect.**

Previously I stored the password Test1 into the database and it hashed to 27. I also added another account with the same password, and it hashed to 28 due to the collision resolution. I added 2 accounts with random passwords hat hashed to 22 and 32 then added another account with the same password Test 1 as shown above so you can see the collision resolution recognised 28 is also taken so it hashes to 29.

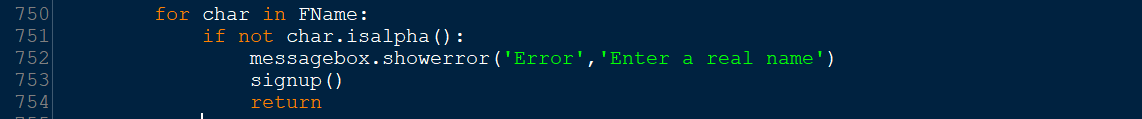
1.1.2: Users hashed passwords will be stored into the database

I store all the user's information once it has been checked. Using an insert statement, I commit this to the database and a new user is created. When the user is created it is allocated a primary key, my first users key will be 1 and then 2 and so on. This key is a foreign key in almost every table in my database and makes it easy for me when checking my database to see what information belongs to what user.

At the end of my program, I print the information currently stored in every table in the database. This is what is stored in the user table.

**Objective 1.2. Verification every time a user creates an account.**

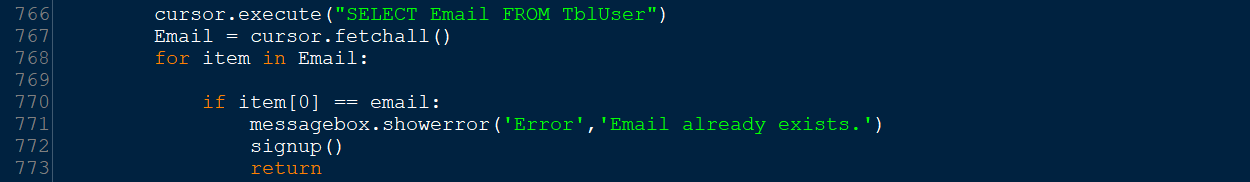
1.2.1: The first and last name should only include letters

The code iterates through every character in the users first name and last name and checks if all characters are letters using the isalpha function an error message is displayed if any characters are not letters.

**OUTPUT:**

A screenshot of a computer screen

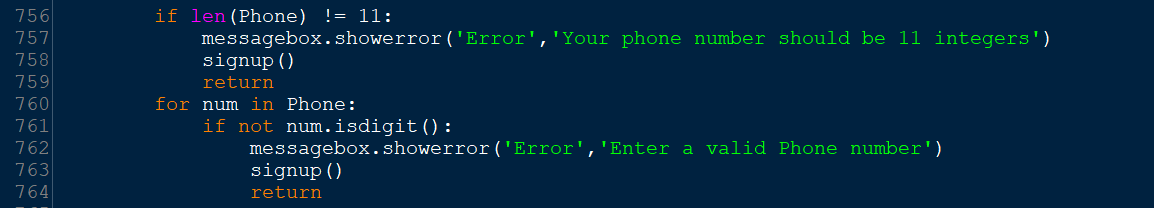
AI-generated content may be incorrect.

1.2.2: The email must not already existThe code above checks that all fields required have been filled using an if statement to check if any are blank. If all data has been entered it fetches all the emails currently in the database using a SELECT statement. For every email in the list of emails already stored in the database the if statement within this for loop compares each email stored with the user’s email. If they are the same it tells the user the email already exists within the database, so they cannot create an account with it. Then the subprogram where users can enter data is called again so the user can change their email or log in.

A screenshot of a phone

AI-generated content may be incorrect.

1.2.3: Phone numbers should only include numbers and should be 11 characters long

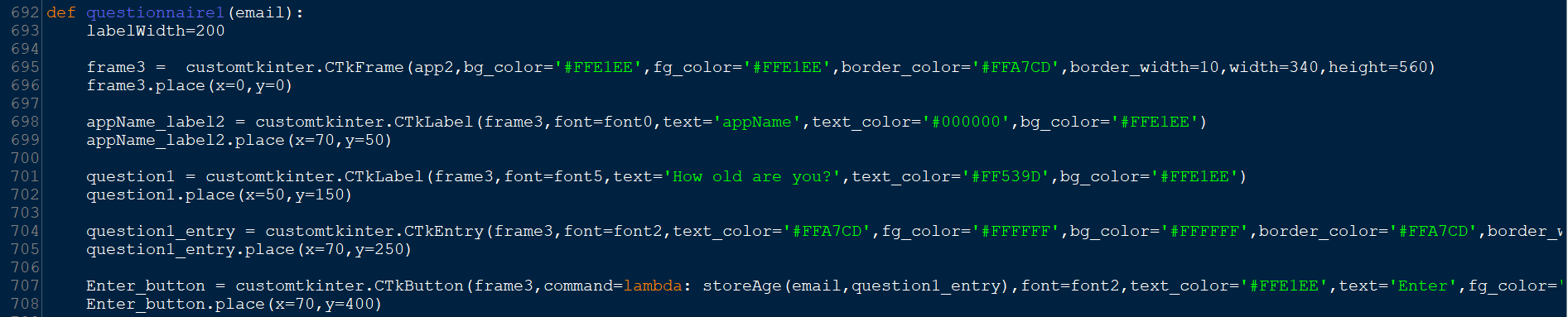
The code iterates through every character in the user's phone number and checks if all characters are numbers using the isdigit function. A message error is displayed if characters are not all numbers.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Objective 1.4. New Users will fill out a questionnaire about themselves.**

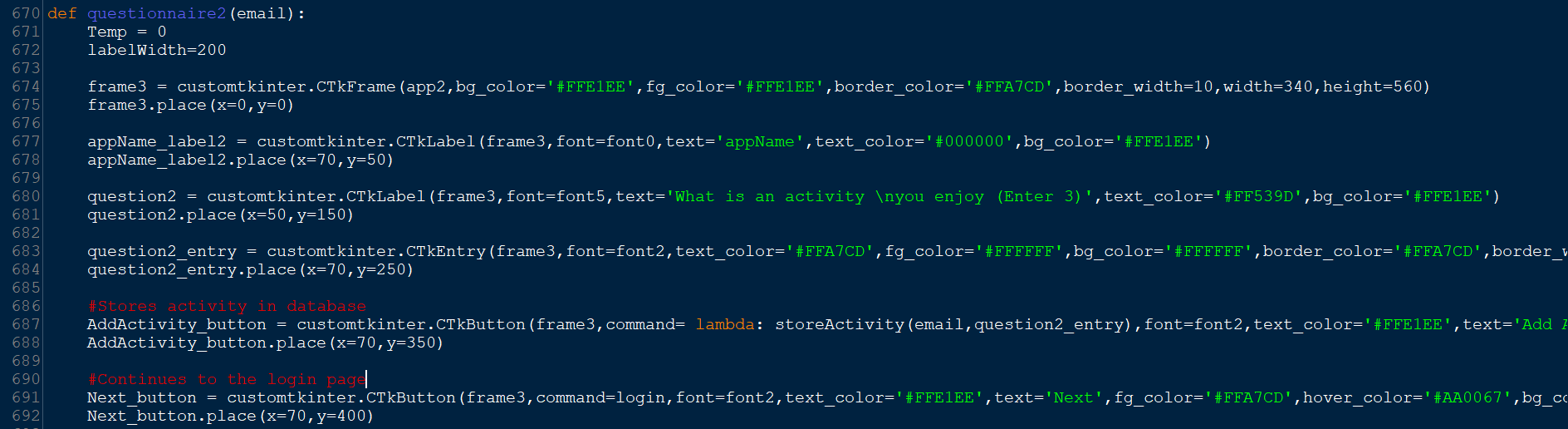
User interface for objective 1



A screenshot of a pink box

AI-generated content may be incorrect.

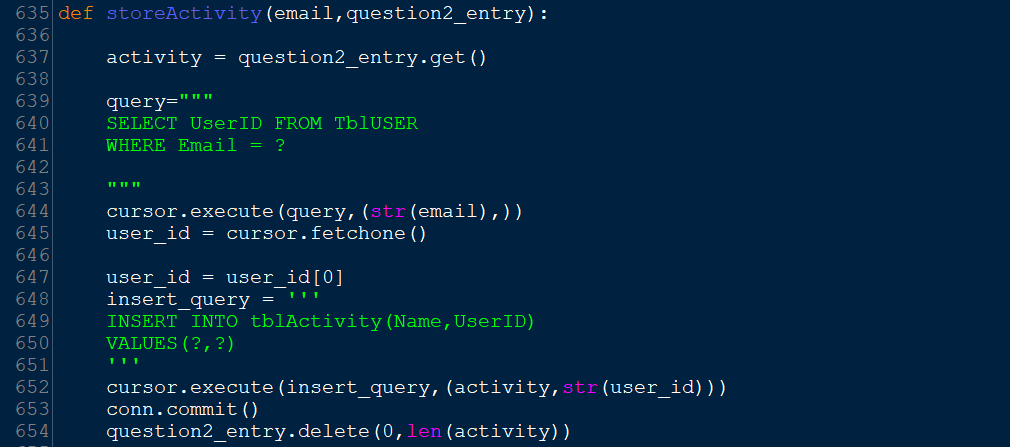
Users can enter activities in the questionnaire.

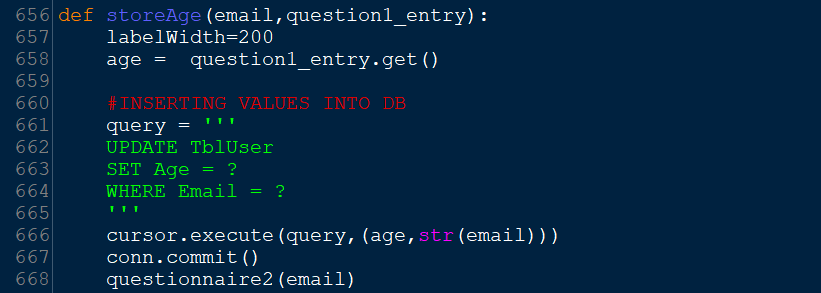


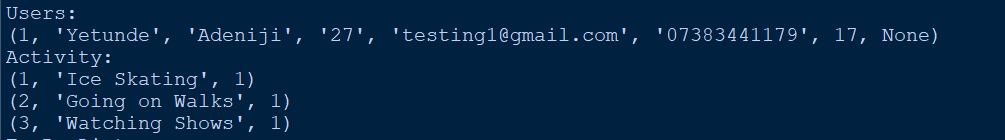
1.4.2: All information from the questionnaire will be stored in a database

A screenshot of a pink screen

AI-generated content may be incorrect.

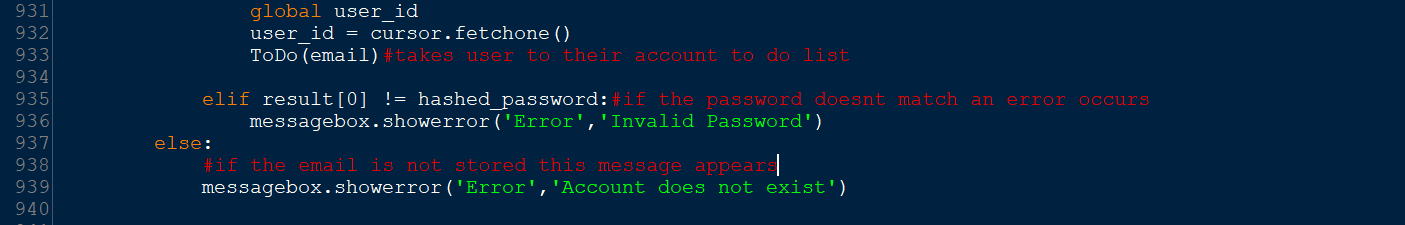
The above screenshot is from the **storeActivity** subprogram previously mentioned. In the while loop every time the user presses add activity, they are taken to the storeActivity program where the activity is stored in the Activity table. Previously my plan was to store this in the questionnaire table as my entity diagram will show however when programming I realised storing the Activities in a table like this would be easier to access for the schedule page in future. To store the activity, I first selected the **UserID** from the User table that corresponds to the users email they entered when they signed up. It then stores both the user ID and the activity name into the activity table. These changes are committed, and the activity entry box is cleared.

As I have previously stored information in the User table I decided to use an UPDATE statement instead of an INSERT statement. Like the activities the users age was also going to be in the questionnaire table. However, it was also much easier to put this information in the user table.



**2.1: Users should enter their password and email address to login**

User interface for objective 1.1

Here I used a select statement to get the password in the user table that corresponds with the email the user inputted. The password they input when logging in is hashed using the same algorithm from the sign-up section and compared with this password. If the passwords are the same the user is taken to the questionnaire otherwise an error message is displayed, and the user can enter the correct password. If the email they entered is not in the database, the user is told the account does not exist.

A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

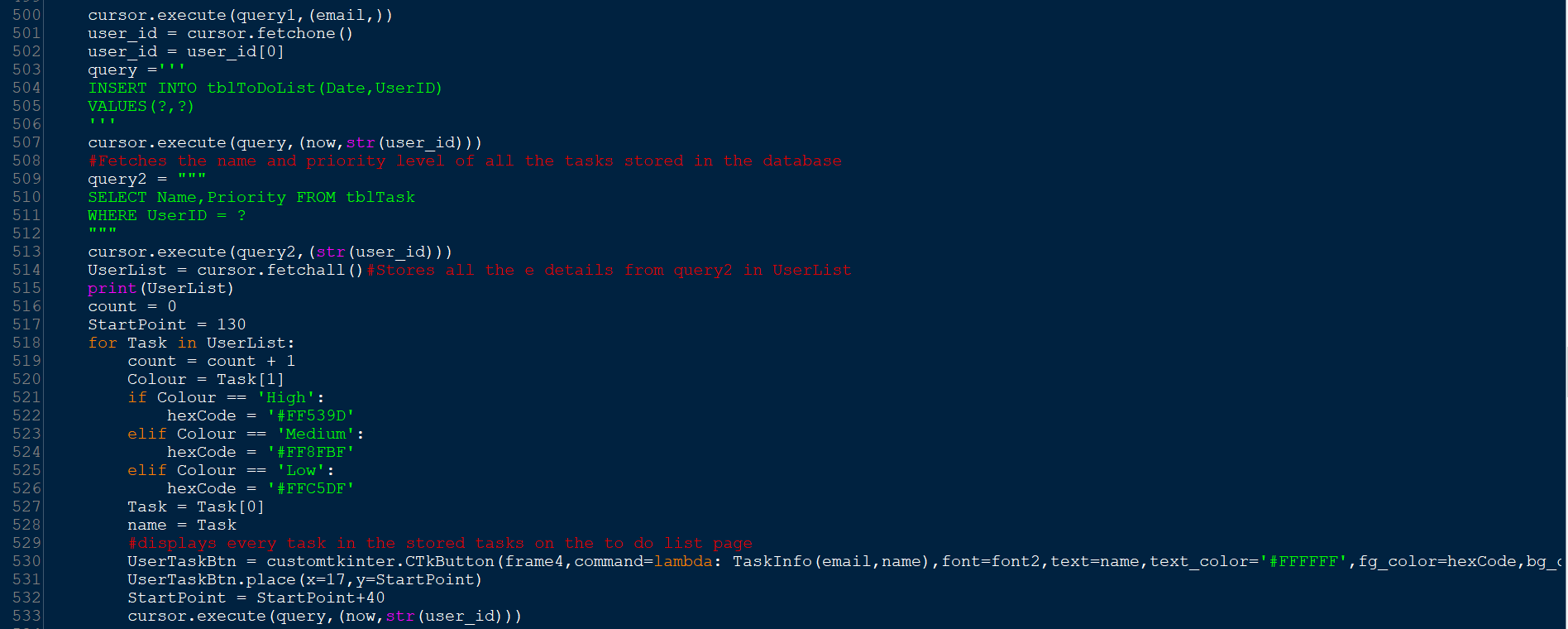
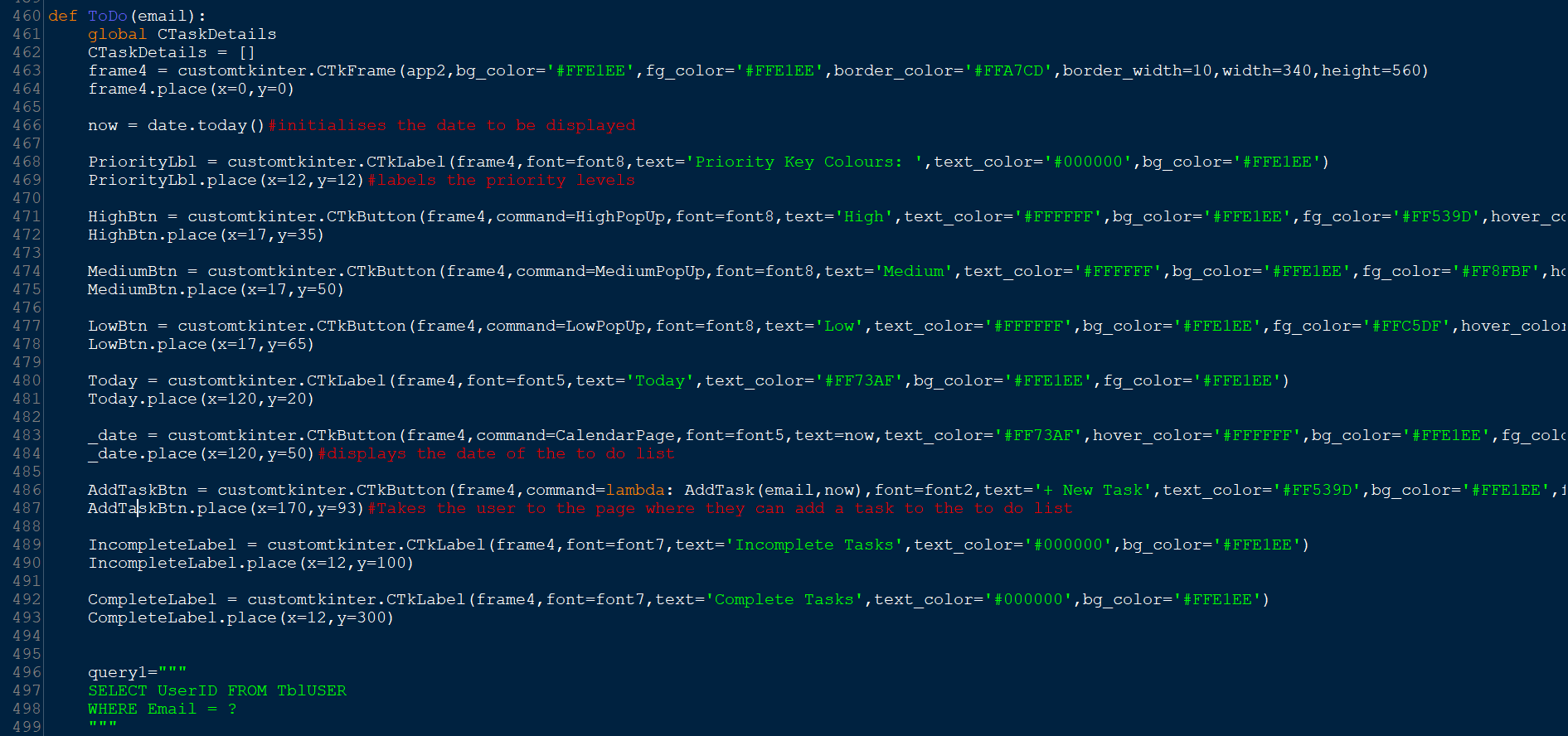
AI-generated content may be incorrect.

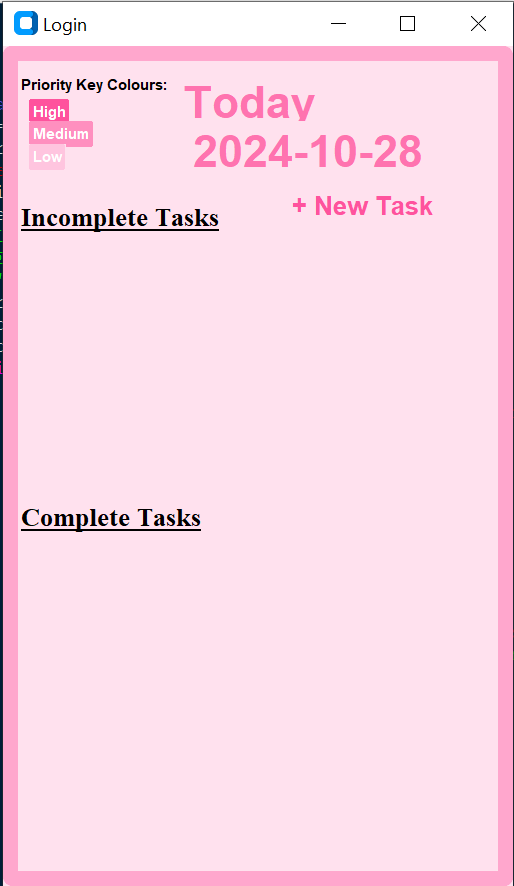
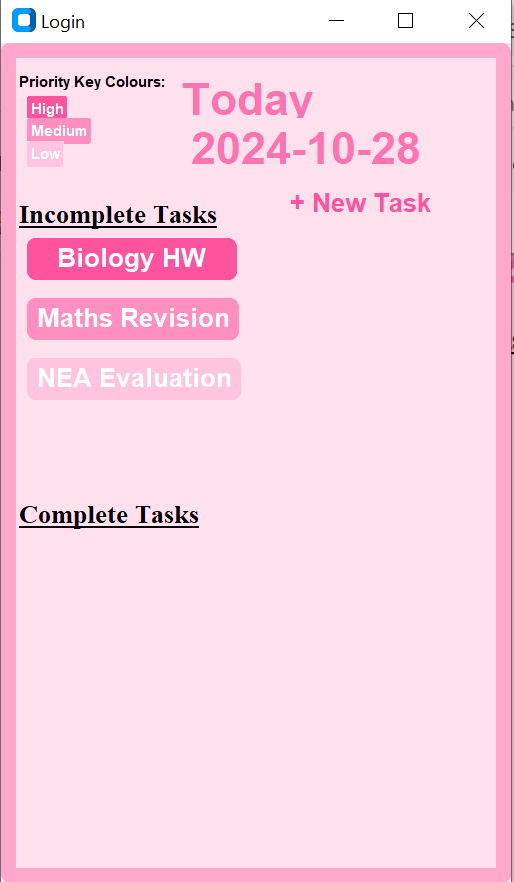
**A screenshot of a computer screen

AI-generated content may be incorrect.**

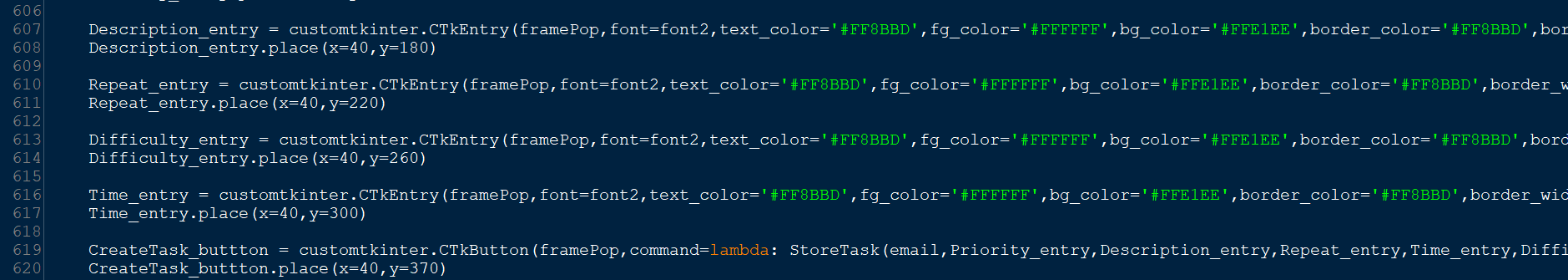
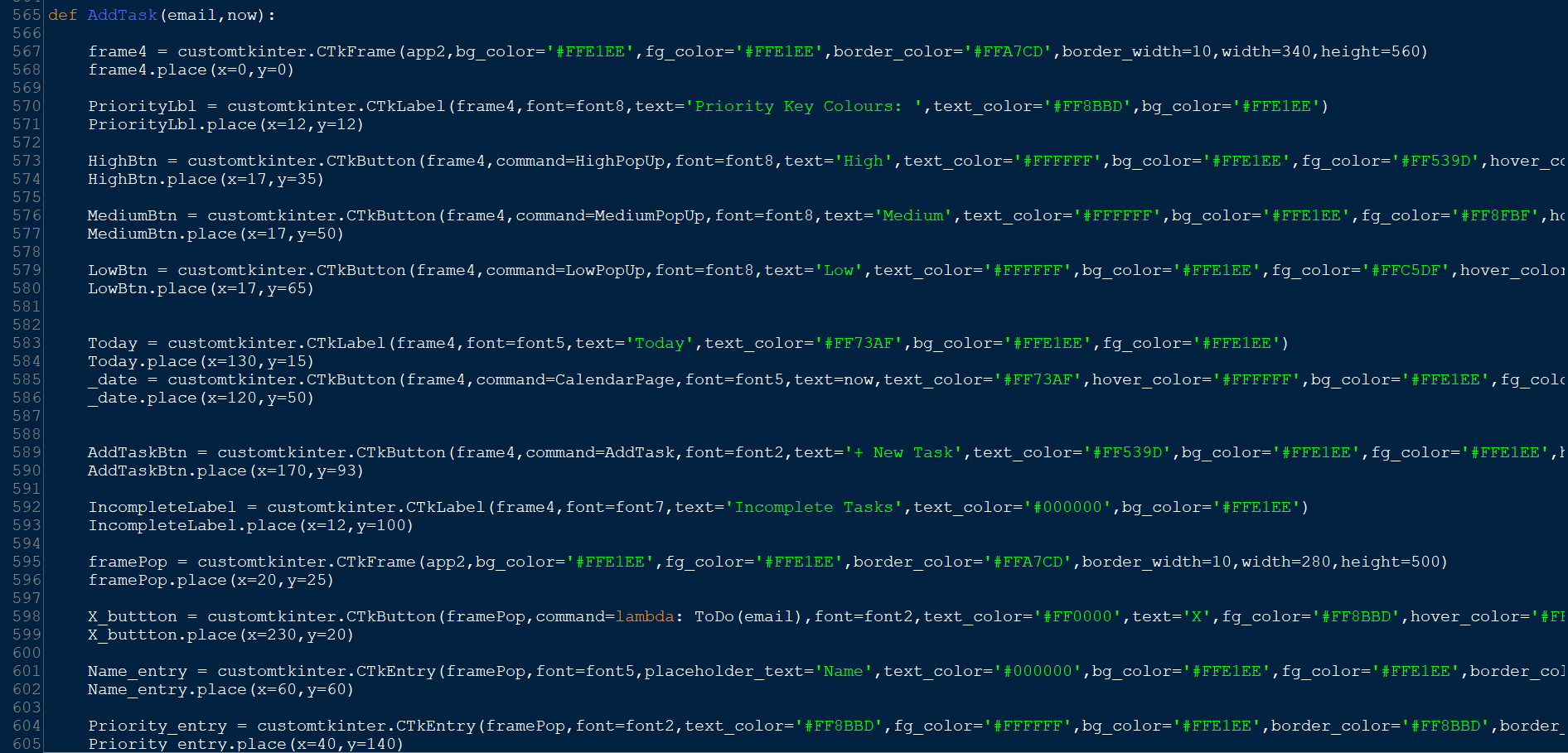
**To do list page**

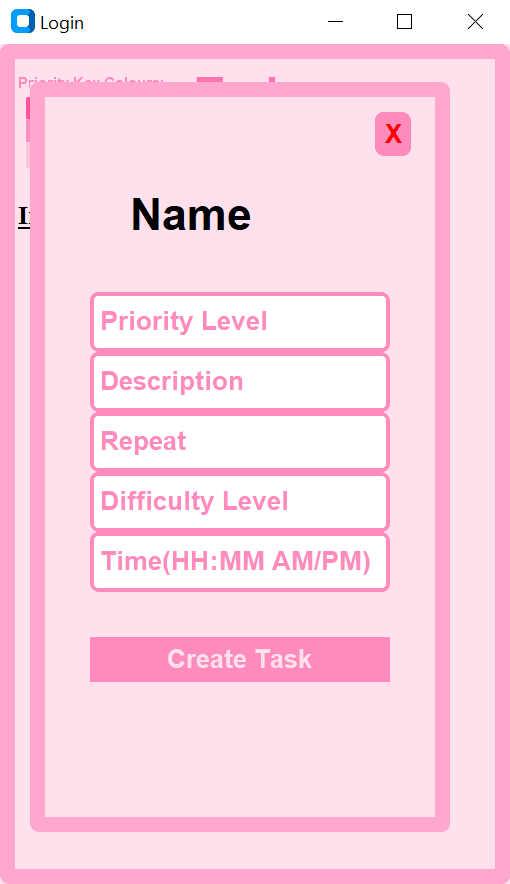
User Interface for to do list.

I use a select statement to get the list of already stored Tasks in the database. It fetches the name of the task and the priority level and stores this in the **UserList variable.** For every **Task** in the UserList The priority level is stored in the variable Colour on line 520. If the Colour is a certain priority the task will be displayed as a certain colour as shown on line 530 where the **fg\_color** is set to the colour corresponding with the priority level. This is stored in the variable **hexCode.** The variable, **name** is set to the first item in **Task** (Task[0]). This is displayed as the text of the task as shown in line 530 where text is set to name.

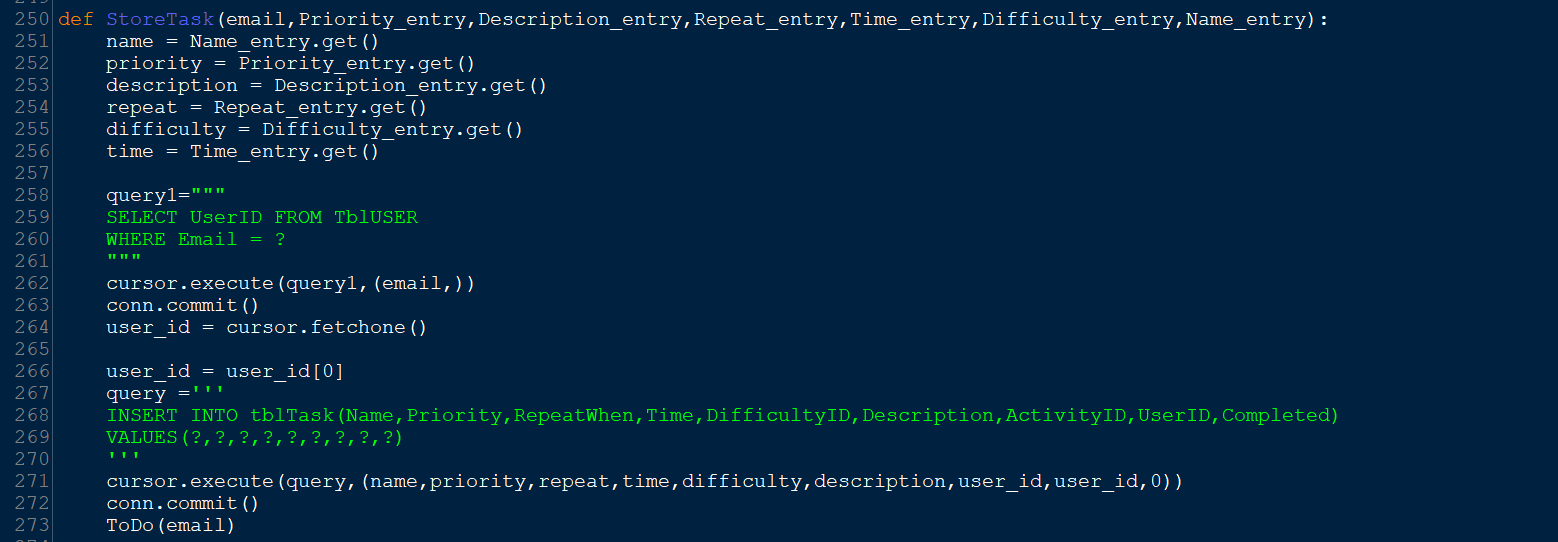
 

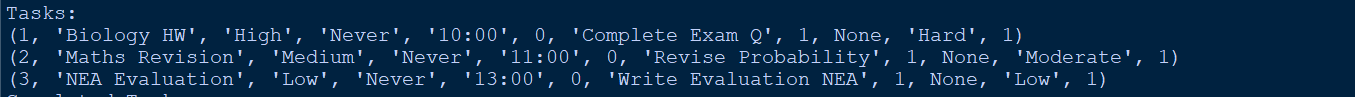
**3. Users may add a task**

The entry widgets allow the user to enter all the information about the task. When they press the create task button the task is stored in the **StoreTask** subprogram.

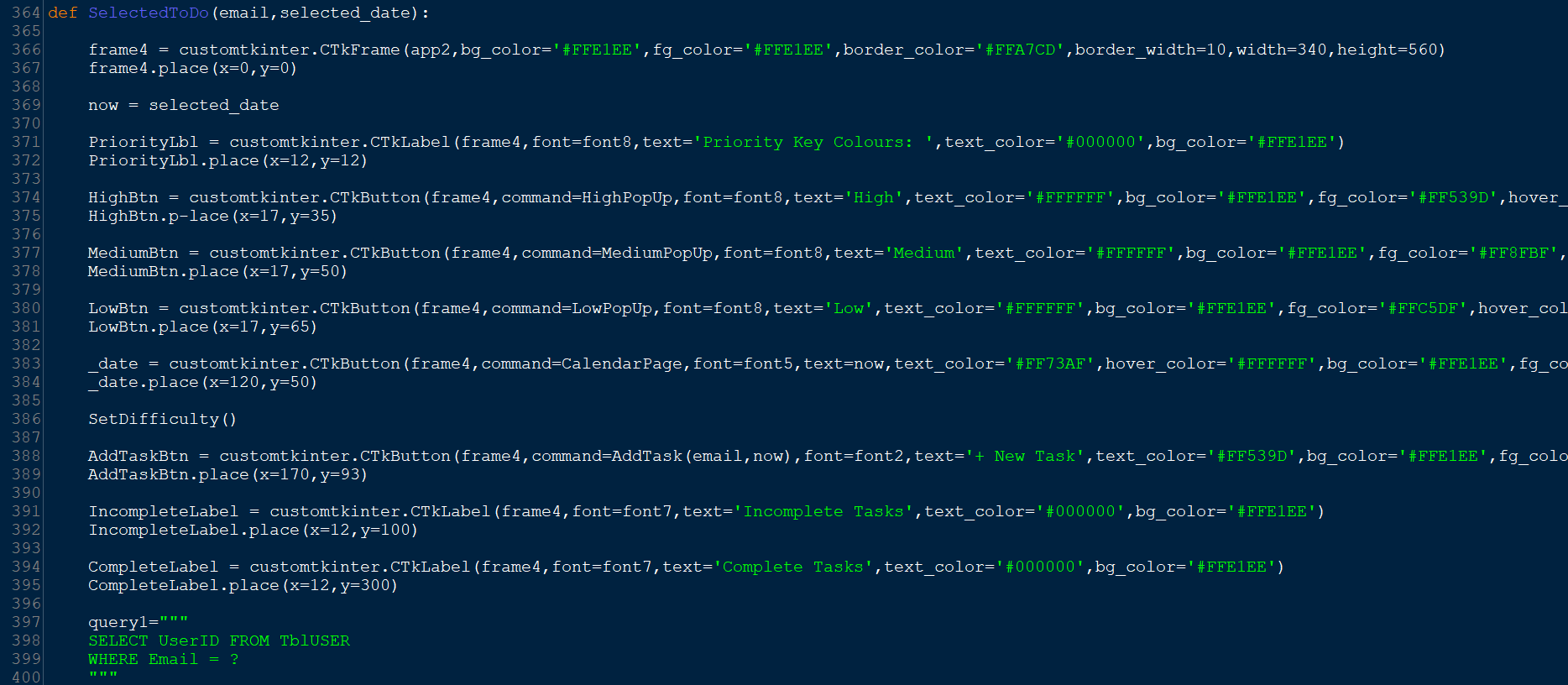
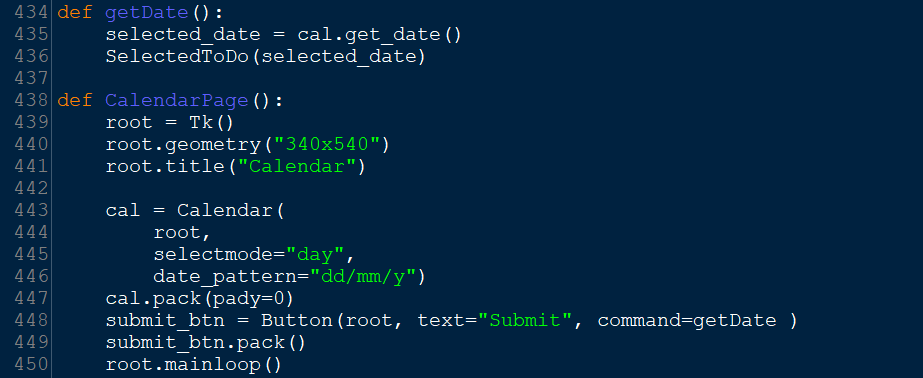


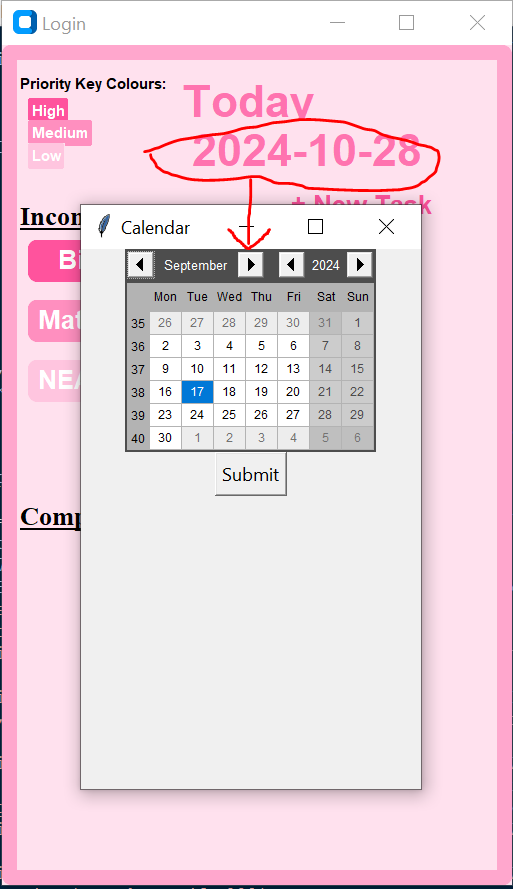
3.1.8.1: Once task is created it should be stored in the database

From lines 251-256 I store all the user's inputs into variables that are shorter and easy to code with. Then I store these values into the Task table in the database. The user is then taken to the to do list page where the task will be displayed.

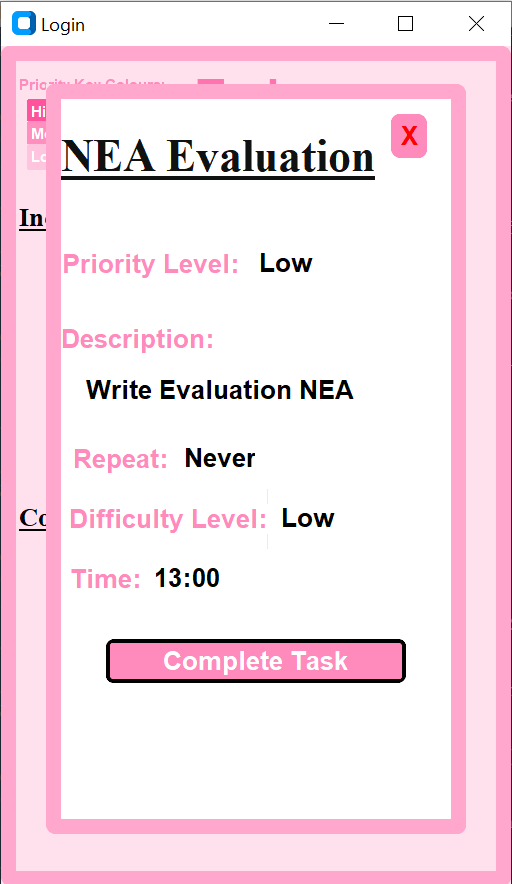
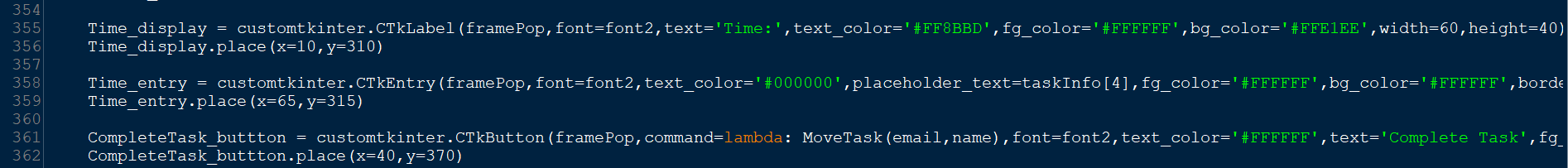
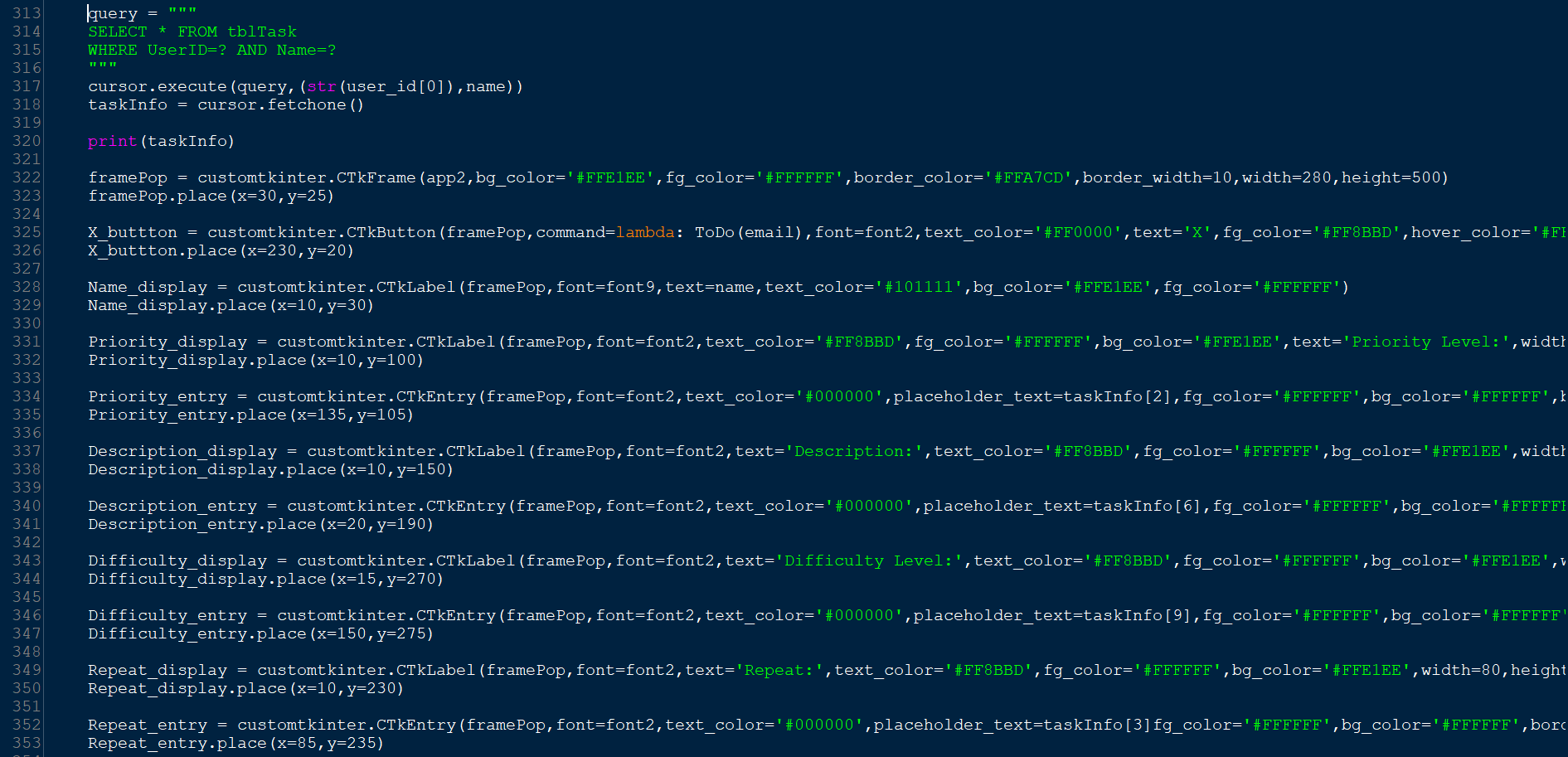


**5. To do list and calendar should be linked**





3.2.1: Users can press icon next to each task to mark them as completed



I used a select statement to select all the information about the task that is clicked. This information is stored into the variable **taskInfo**. This array is used to fetch the stored user inputs to be displayed when the TaskInfo subprogram is called.

**Watch Test Video (Screen Pal Version)**

0:00 – Testing a name with numbers/ special characters.

1:10 – Testing a phone number greater than 11 integers.

1:40 – Testing phone number with letters in it.

2:10 – Testing a password greater than 10 characters.

2:30 – Testing hashing algorithm.

2:36 – Completing questionnaire, Inputting age.

2:50 - Completing questionnaire, Inputting Activities

3:16 – Testing signing up with the same email as another user.

3:42 – Testing empty entry box.

4:12 – Testing wrong email when logging in.

4:33 – Testing wrong password when logging in.

4:46 – Testing logging in successfully.

4:54 – Testing hashing collision resolution.

5:57 – Testing priority key pop ups.

6:14 – Test pop up calendar

6:26 – Testing adding new tasks.

6:50 – Tasks are coloured based on priority.

6:52 – Task details are displayed when the task is clicked.

8:26 – The task is removed from the incomplete section when complete task is clicked. The task ID is stored in the complete tasks table.

**Watch Test Log In (Screen Pal Version)**

**Evaluation**

My code meets only some of its requirements. On its own it works well however considering my original objectives the code does not meet as many requirements as I would like. The objectives I did meet where achieved to a good standard allowing the app to be quite functional. Users can create accounts and log in to their accounts. When they create an account, there is a hashing algorithm in place for security. Users may make to do lists. The tasks are coloured based their priority level. When creating tasks users can input information about the task. They can check this information when they click the task in the to do list. I conducted a survey to measure the overall success of programme my programme.

**Objective 1 - Users will be asked to create an account.**

When the program first begins the sign-up page appears. Users must enter their email, first and last name, phone number and a password to create an account. I used if statements to ensure that the user enters all the data, their name is only letters and the phone number is valid, 11 integers and the password is less than 10 characters. When all the information is checked the user’s information is taken to another subprogram so the password can be hashed. Originally, I made the entry widgets for the users sign up detail's global variables. However, for complexity I removed these global variables and passed the users details to the next subprogram to be stored using the lambda function. When I first tried to enter parameters when I called the subprogram ‘signup\_account’ (the subroutine that hashes the user's password’). Errors occurred as the signup\_account subroutine was called immediately when the program ran. So, I added lambda to stop this.

I use this lambda function throughout my program to pass user information's through to different subprograms. Mainly the users email when they log in as this is used in most select statements to get the users information. Once the user enters valid information to sign up the sign-up account subprogram hashes the user's password. My algorithm works well to hide my user's passwords however when improving my program, I want to be able to apply them to any password as mine only works for passwords 10 characters or less meaning that users are limited when creating passwords. This may be mitigated by making the words in the hash Words array cycle over, so I don't need to add more words to the list.

The sign-up page was quite effective, and no errors occur for all types of user inputs. The design of this page is slightly different to the one I created in the design section however all the colours match. **Objective 1.3** says the user verifies their account through email. As I am not able to make an automated email this objective could not be met. Another option would have been for me to email users myself however this would be ineffective and time consuming, so I decided not to execute this objective.

**Objective 1.4** is the questionnaire that users fill out once they sign up. This objective was met successfully as the users age gets stored and they can enter activities they enjoy which are stored in the database along with the user ID. However, the user is only supposed to enter 3 activities, there are no measures in place to prevent the user from entering more than or less than 3. I attempted to use a while loop to prevent the user from entering any less than or more than 3 but this caused the whole program to freeze. In future I believe a count of every time an activity is stored could be used to ensure the user only enters 3 activities.

**Objective 2 - Existing users can login**

This objective was met successfully. When users log in with incorrect inputs errors are displayed and when they log in with the correct information, they are taken to their to do list.

**3 - Users may add a task**

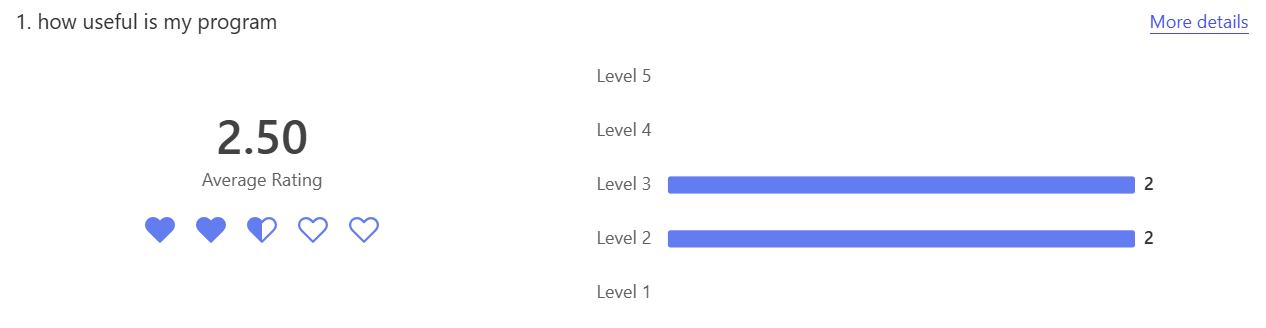
This objective was met fully, users can add a task, and it is stored in the database. The tasks are displayed on the to do list and are coloured using the priority key. This is exactly the function I was trying to achieve. The tasks name, description, priority and difficulty level, time and repeats are stored in the database. Each task also has the users ID and its own ID within the database.

The only difference is there are no tick boxes next to each incomplete task. When the task is clicked all the information displayed and at the bottom the user can select ‘Complete Task’. My plan was for the task to be removed from the incomplete task section and moved into the complete task section. However, only the first part worked. When the user click’s complete task the task is no longer displayed in the incomplete task section. It is also deleted from the Tasks table in the database. The Task ID is then stored in the completed task section. I was going to use this to select all the task information where the task id matches the completed task ID however, I had already deleted the task from the Task table in the database so if I were to select the task details with the matching completed task ID this would return nothing. To fight this error, I stored the task details in a set before I delete this information from the Task table. This also gave me errors and I realise this is because I did not use a specific index when accessing information in the task details set.

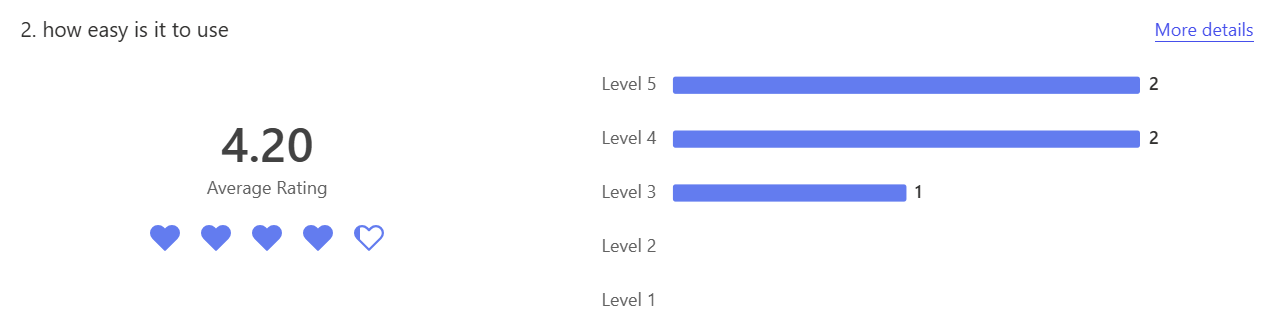
**4 - Built in calendar**

The built-in calendar code works, and it relates to the to do list page as to access it there is a button embedded within the subprogram that displays the date. This takes the user to a calendar. The calendar is displayed however it does not match my original design. Using tkinter I thought I would have a lot of control over the design of different aspects of my application. When researching how to make and embedded calendar I realised this would not be possible in tkinter and so I still made a calendar with none of the colours and other design features I planned. Another issue is the calendar does not have its main functionality which was for users to add events to any day. I tried to change this function so users can change the date of the to do lists and add tasks to a certain date. To properly execute this function, I would need to first select the to do list ID of the selected date and display all the tasks with the same to do list ID stored in the database.

**Feedback from Users**

Out of 5 my users rated on average that my program usefulness is 2.5/5. The highest rating was 3 and the lowest 2. This did not surprise me as my program does not meet all the previously established objectives which decreases its functionality. The main objectives met are not ones that benefit the user such as the sign up and the log in page. So, it makes sense my users rated its usefulness low as the to do list is only part of the intended function and not all of it.

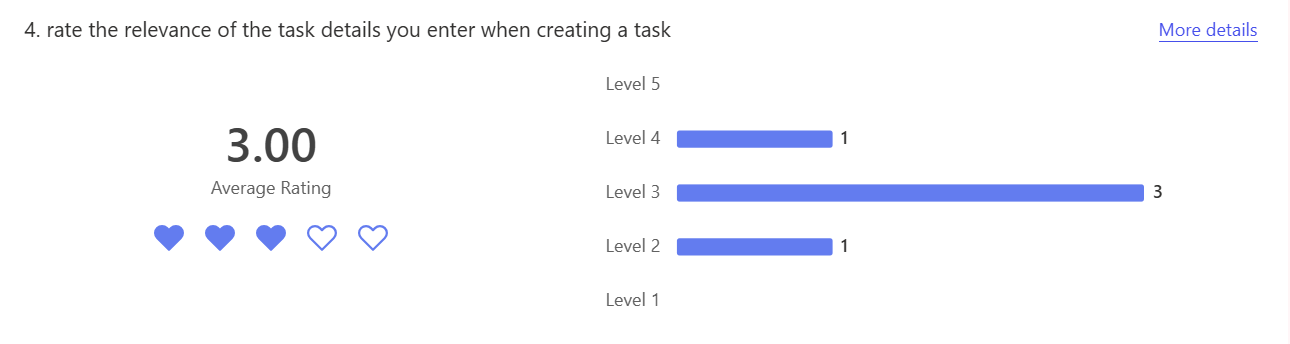
To improve my program and increase the rating. Completing the built-in calendar would add to my program’s function. The users being able to add events and the idea of corresponding emotions and reminders would have been very beneficial to my users with all the exams my target audience would have. To do this I would place an add event button on the to do list page which has a similar structure to the add task function. This would work given that the user can select specific dates on the calendar pop up.

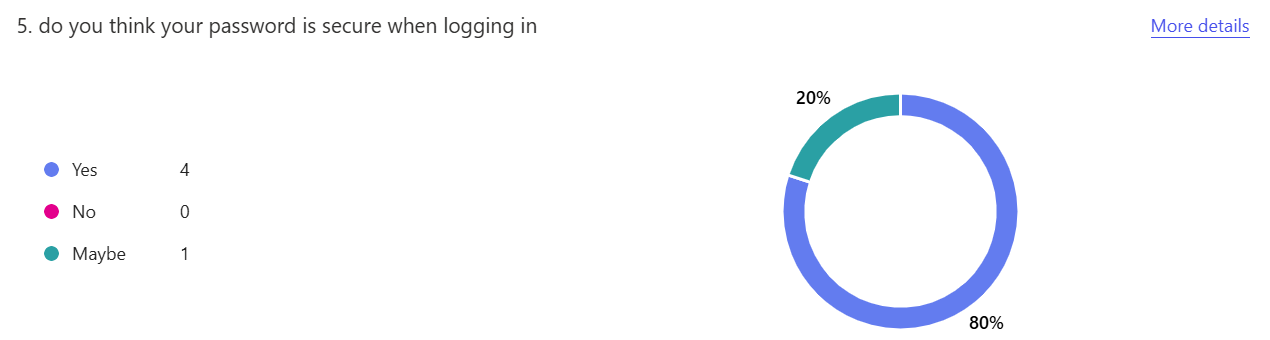
The rating on how easy my app is to use scored quite highly with most users rating this 4 or 5 out of 5. The user that rated this 3/5 indicated the low rating was due to the design of my application.

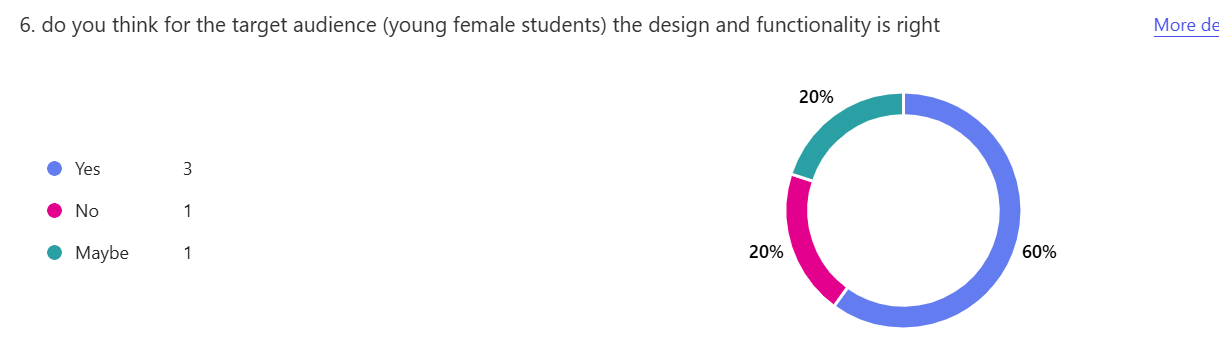
The same user rayed the design of my app low.

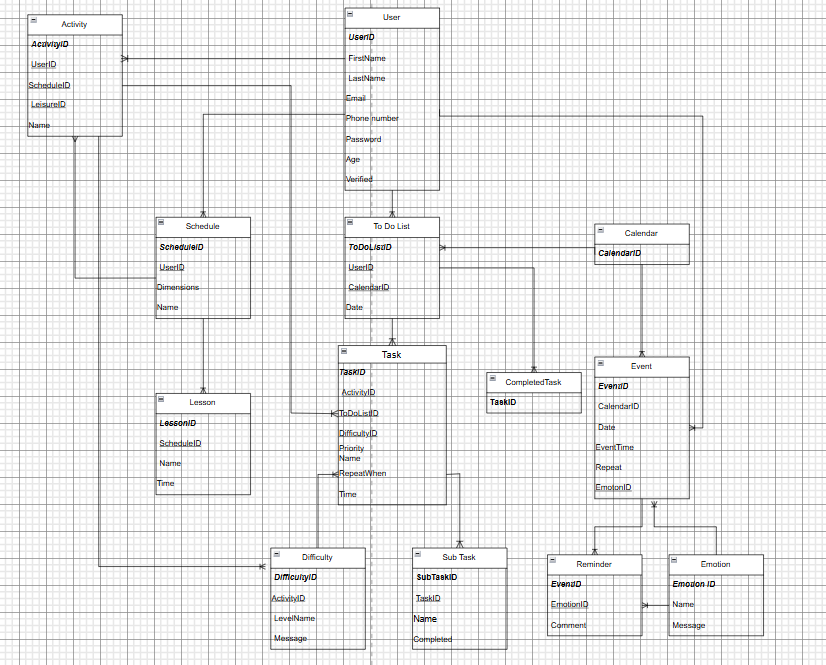
This user is dyslexic, and the intense number of pink colours makes the sign-up page and to do list page especially difficult to read. The user however says the log in page was the easiest to use. When comparing the sign up and login pages I see the login page involves bold black writing where the user needs to enter information. This is what makes it clear for the user to input the necessary information. Whereas the signup page has pink writing that matches the border of the sign-up frame. Although it may be aesthetically pleasing the pink writing is much less clear than the bold black writing. If the whole program used black writing to make important information stand out users like this one would be able to enjoy my app. To amend this, I can change the hexadecimal of certain parts of my code to black so they stand out and my user can read them. For example, all the place holder text on the sign-up page entry widgets can be changed to a black colour. This is the design on the log in page and would make my app easier for dyslexic users to use.

Another user says they would not download my app as it hurts their eyes. I believe this is also due to the intense amount of pink throughout the code. To fix this I can add variety to the colours I use so it is not too intense for my users, and they can enjoy the actual function of my app without the design preventing them. Previously I conducted a survey asking users if they prefer to choose the colour of an app or just use the original one and 95% of respondents said they do so to improve my app I will add the choice of other colour themes that had a high average rating also so my app is more flexible and easy to use.

The average rating on the relevance of all the task details required was 3/5. I asked my users this question as I was unsure all those details about the task were necessary and this rating is to confirm. As the average rating was not very high, I do not believe the user needs to enter all those details about a task. For example, the time of the task being stored as one of the details is not completely necessary. Perhaps if the time was used to notify and remind the user before their task needs to be complete, entering the time would be useful. However, as there won't be notifications sent to users the time is quite irrelevant. The users being required to enter this unnecessary information is a waste of storage and therefore ineffective for my program and the space could be used for more informative details.

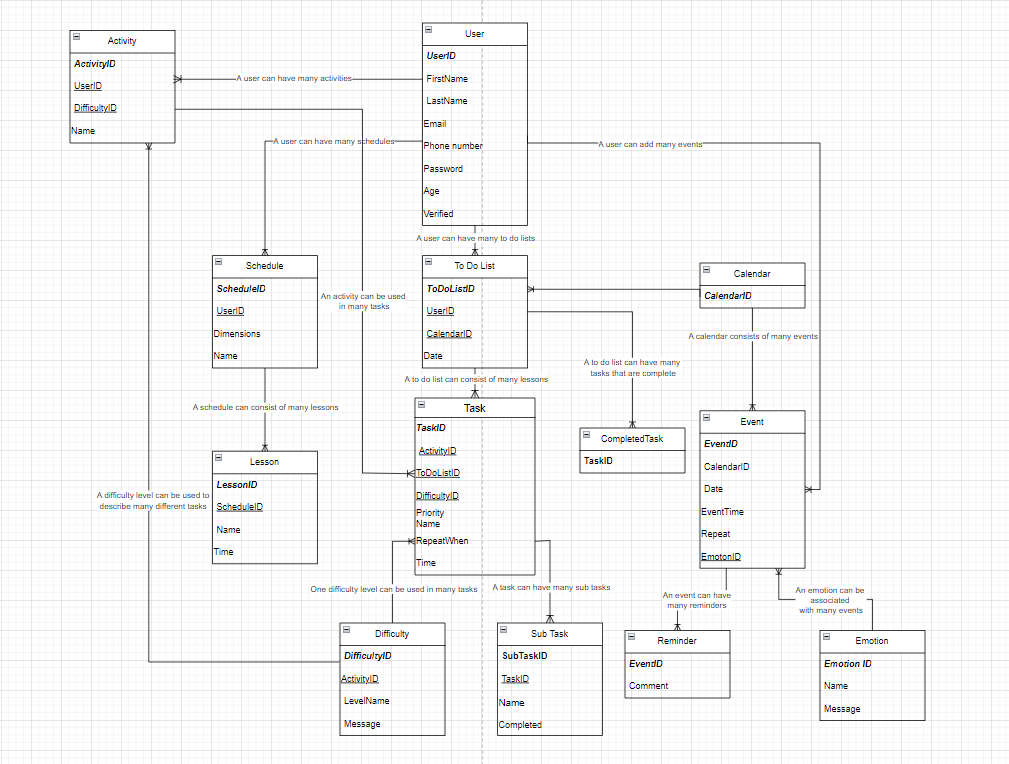
80% of users believed the passwords are secure when they are logging in. This is mainly because the passwords are hashed with a random algorithm, I made myself. This securely stores the passwords an makes them untraceable in the database. However, 20% of users said the password is maybe secure. When users are logging in the password is not displayed as asterisks, this leaves room for shouldering. If the password was displayed as asterisks when the user enters their password the password would be completely secure as the hashing algorithm makes the passwords untraceable in the database however when logging in the user's password can easily be observed. This measure removes this problem.

60% of users agree my programs design and function is right for my target audience I previously mentioned in my analysis. However, 40% of users were unsure or did not agree. Again, this is likely due to features I mentioned before, not all young girls favour the colour pink and so my app would only be perfect for a very niche audience which limits my users. To broaden the amount of people who can enjoy my app I would add a variety in colour and design as I previously mentioned in question 3. Also, my app does not meet all objectives so it is only partially beneficial to my target audience, young female students as with many exams adding events would help users with organisation.



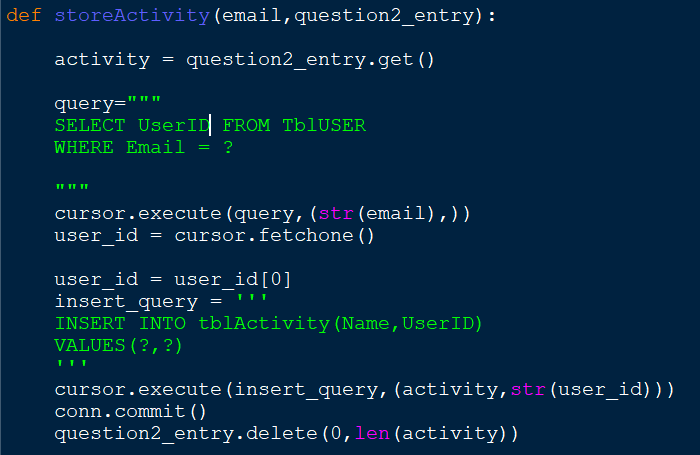
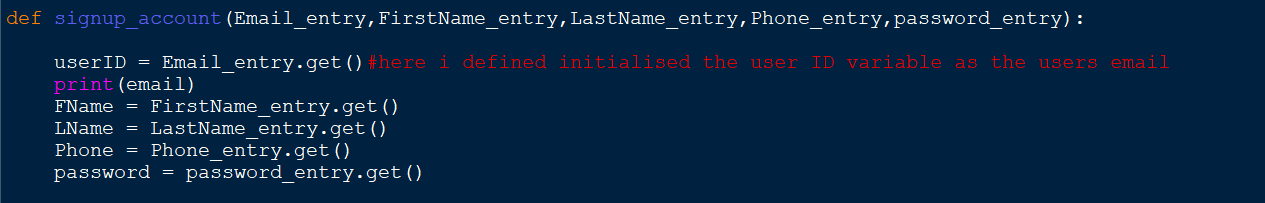
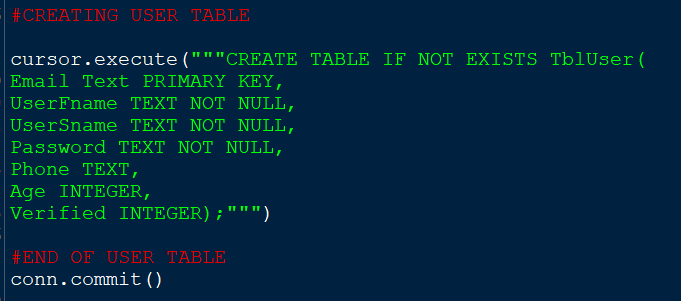
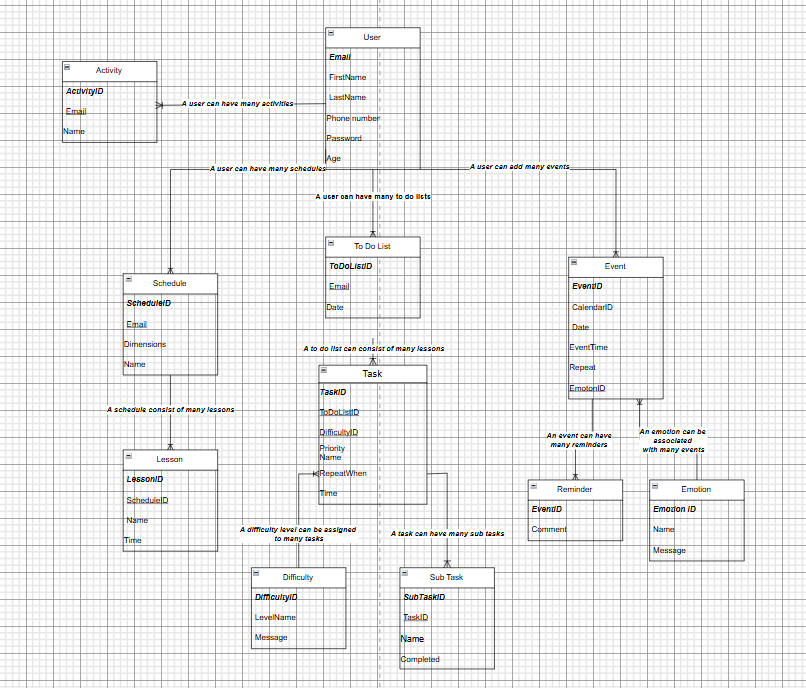
When making my database originally, I had a questionnaire table. This was going to hold information about the user including the LeisureID which is the unique identifier for the questionnaire. Every activity was to be stored in the activity table which would be directly connected to the questionnaire table. One questionnaire to many activities. The Activity table and User table would have been connected through the questionnaire table. Although this connection was reasonable it was difficult to code with, and I found having no questionnaire table most effective.

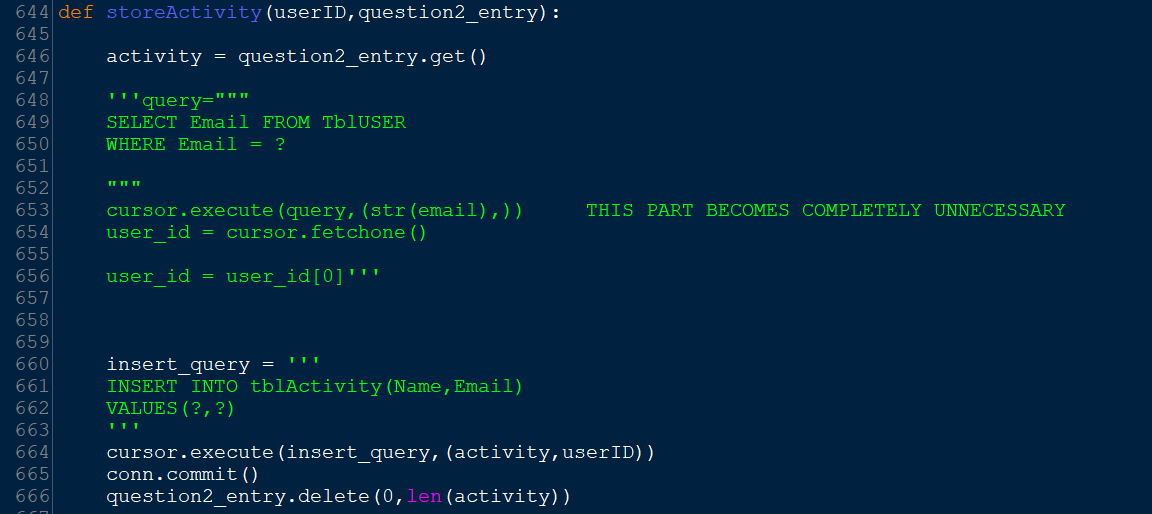
I edited my database to have no questionnaire table so in replacement every time a user adds an activity the User ID is stored in the database along with the activity, so each activity is connected to the user it belongs to. I edited my entity relationship diagram to reflect this:



In my entity relationship diagram, I displayed how the User can have many schedules and to do lists. However, I had an issue as the user and calendar had a one-to-one relationship and for it to be fully normalised I need only one-to-many relationships. So, I added events as users can have many events throughout the calendar ‘s year and calendars can have many events stored within them. Also, because I wanted the to do list page and calendar page to be connected a there are daily to do lists which are illustrated on the calendar, I added that calendars can have many to do lists. I also struggled to connect the questionnaire to other entities as I intend to use the user's activities to recommend what they should do in their free time when making a schedule and as a reward when they complete a task in there to do list. So, to connect the questionnaire to the to do list and schedule I added one to many relationships between the questionnaire and the schedule and the task.

In my entity relationship diagram and my database, I use the user ID which is a number assigned to users that indicates when they created their account in relation to other users, however when the user creates an account, they enter an email and users cannot have the same email as each other. Therefore, the email is also a unique entity identifier so to reduce unnecessary select statements like the one below and reduce the size of my database. To correct this, I will remove the user ID column from my entity relationship diagram and database and replace this with the user's email in my program I will define the variable user\_id as the users email using the get function rather than the user ID so I can then use this in comparative statements when checking users information for example checking what user owns a to do list. Below are my changes:

I had multiple cases like the one above where I selected the users email to select the users ID. Making the email the primary key simplified sections of code like this one a lot. Below is how this specific part of my program looks after this change.

From lines 648 to 656 the code is completely unnecessary. When the user logs into their account successfully the email they use is passed through each subprogram. As the email is now the primary key, I do not need to select anything from the user table as the user's email has already been passed through the subprograms from when they logged in. I will only need to select the users email before they log in to insert their age this will still be simpler as there's no need to first select the users email as this is now the primary key. This change adds to the efficiency of my code

**References**

**30th September – Login System with MYSQL -** [**https://www.youtube.com/watch?v=AIKpNSu6980**](https://www.youtube.com/watch?v=AIKpNSu6980)

**30th September – Tkinter Beginner Course - https://www.youtube.com/watch?v=ibf5cx221hk**