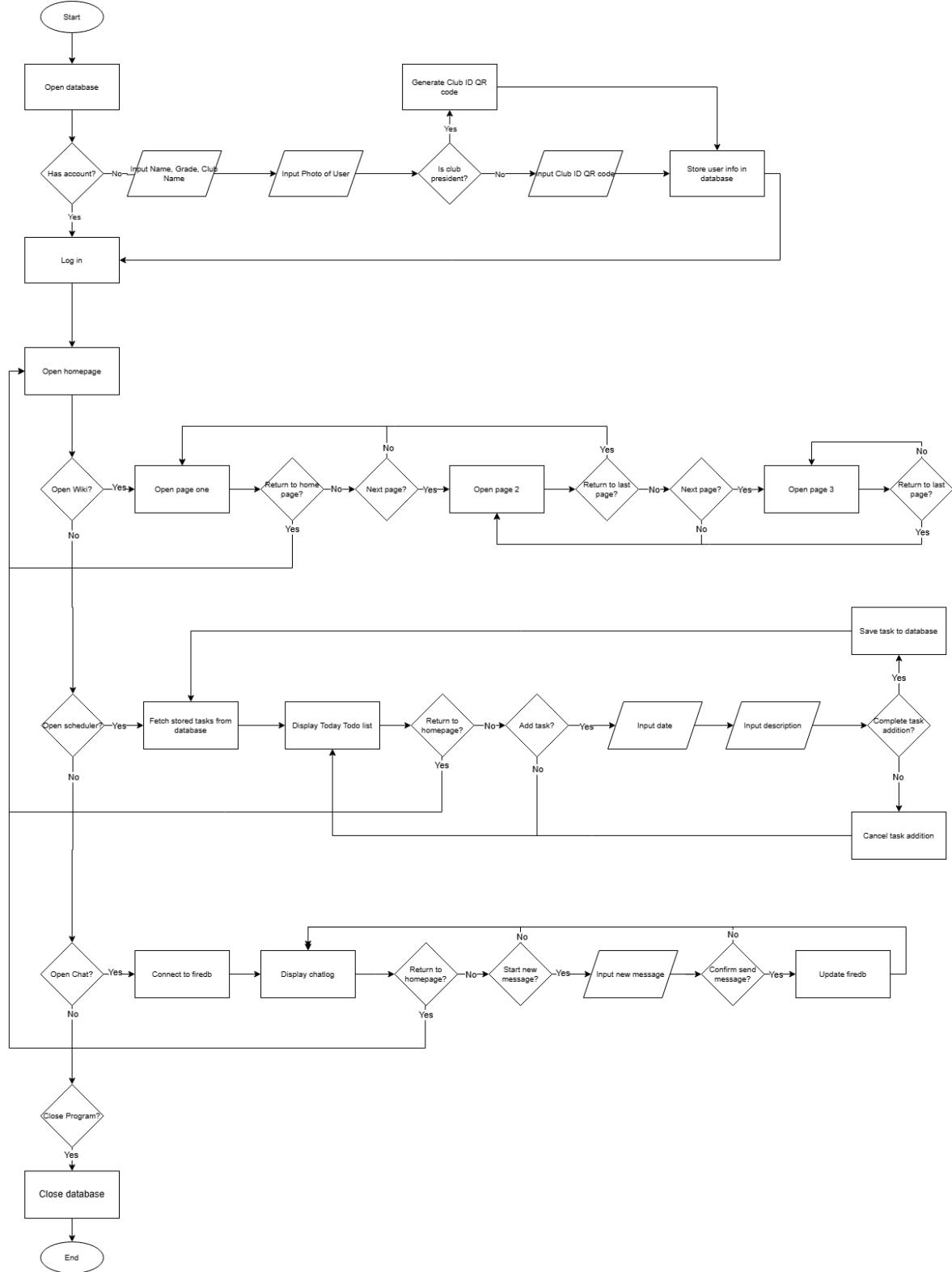


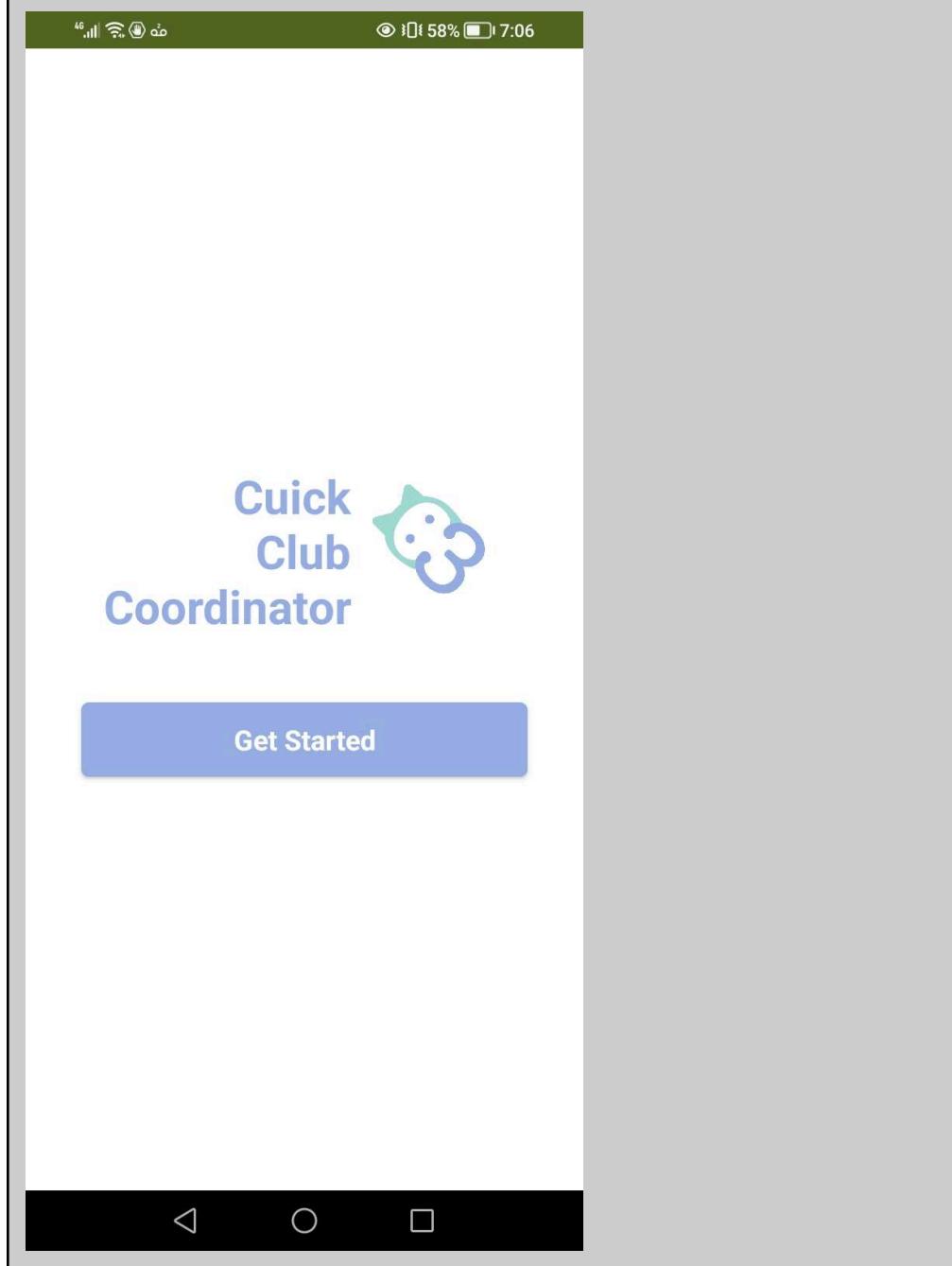
1. Flowchart



2. Screen UI and background elements + Code + Screenshot

Screen1		
User Interface	Name/Purpose	
Buttons	Start If pressed, sends the user to registration or login depending on if the user has an account.	
Images	Logo Displays an image of the app logo.	
Labels	Name Displays the name of the app.	Label2 Provides spacing.
TinyDB	TinyDB1 The main database of the app. Stores the user account information used.	

Screen1



Screen1

The Scratch script consists of three main sections:

- A global variable `account_exist` is initialized to `false`.
- When `Screen1` initializes, it sets `global account_exist` to `to` call `TinyDB1`.`.GetValue` with tag `"account_exist"` and valueIfTagNotThere `false`.
- When the `Start_btn` is clicked, it checks if `global account_exist` is `true`. If `false`, it opens the `Login` screen; otherwise, it opens the `Register` screen.

Register

	The main database of the app. Stores the users new account information.								
Textbox	First_name_entry	Last_name_entry							
	Provides a spot for the user to input their first name.	Provides a spot for the user to input their last name.							
Datepicker	Datepicker1								
	Allows the user to select the day of their birth.								
Password Textbox	Password_entry								
	Allows the user to enter their password.								
Notifier	Notifier1								
	Display warning messages if registration is incomplete.								
Sensors	Clock1								
	Formats datepicker instants for storage and display.								

Register

The image displays two side-by-side screenshots of a mobile application interface for user registration. Both screenshots show a top status bar with signal strength, battery level at 58%, and the time 7:06.

Screenshot 1 (Left):

- Welcome new user!**
- Registration**
- First Name
- Last Name
- Password
- Date of Birth Choose Date
- Next**

Screenshot 2 (Right):

- Welcome new user!**
- Registration**
- First Name
- Last Name
- Password
- Date of Birth
- Next**

Both screenshots include a black navigation bar at the bottom with three white icons: a triangle pointing left, a circle, and a square.

```
when Next_btn .Click
do
  if is empty First_name_entry .Text
    then call Notifier1 .ShowMessageDialog
        message "Please Enter Your First Name"
        title "Warning"
        buttonText "I Understood"
  else if is empty Last_name_entry .Text
    then call Notifier1 .ShowMessageDialog
        message "Please Enter Your Last Name"
        title "Warning"
        buttonText "I Understood"
  else if is empty Password_entry .Text
    then call Notifier1 .ShowMessageDialog
        message "Please Enter A Password"
        title "Warning"
        buttonText "I Understood"
  else if DatePicker1 .Text = "Choose Date"
    then call Notifier1 .ShowMessageDialog
        message "Please Your Birthdate"
        title "Warning"
        buttonText "I Understood"
  else
    initialize local user_name to join upcase First_name_entry .Text
    upcase Last_name_entry .Text
    in
      call TinyDB1 .StoreValue
        tag user_name
        valueToStore get user_name
      call TinyDB1 .StoreValue
        tag password
        valueToStore Password_entry .Text
      call TinyDB1 .StoreValue
        tag birthday
        valueToStore DatePicker1 .Text
    open another screen screenName Register2
```

Register

When the user submits their registration, the program will validate the user's input. If all input is valid, the program stores it into the account database in the correct format. After, the program sends the user to registration2. If the input is invalid, the program will call the notifier to give a warning..

Register2					
User Interface	Name/Purpose				
Buttons	Take_photo_btn Activates the camera for the user to take a photo.	Complete_btn If pressed, check that all fields have been completed. If complete, send the user back to screen1. Otherwise, call the notifier.			
Image	Image1 Empty image box used to display the photo the user takes/uploads. Default set to the app logo.				
Labels	Title Displays the text "Verify yourself".	Label5 Provides spacing for the title.	Label2 Provides spacing for the image.	Label3 Provides spacing between camera button and image picker.	Label4 Provides spacing between complete button and image buttons.
Media	Imagepicker1 Allows the user to select an image stored on the device.	Camera1 Allows the app to use the user device's camera.			
TinyDB	TinyDB1 The main database of the app. Stores the user's photo file path.				

Register2

Notifier	Notifier1 Display warning messages if registration is incomplete.				
----------	--	--	--	--	--

4G 58% 7:07

Verify Yourself



Take Photo Upload Photo

Complete Registration



4G 58% 7:07

Verify Yourself



Take Photo Upload Photo

Complete Registration



Register2

```
initialize global Photo_taken to false

when Take_photo_btn .Click
do call Camera1 .TakePicture

when Camera1 .AfterPicture
image
do call TinyDB1 .StoreValue
tag " photo_id "
valueToStore get image
set Image1 . Picture to get image
set global Photo_taken to true

when ImagePicker1 .AfterPicking
do call TinyDB1 .StoreValue
tag " photo_id "
valueToStore ImagePicker1 . Selection
set Image1 . Picture to ImagePicker1 . Selection
set global Photo_taken to true

when Complete_btn .Click
do if get global Photo_taken
then call TinyDB1 .StoreValue
tag " account_exist "
valueToStore true
open another screen screenName Screen1
else call Notifier1 .ShowMessageDialog
message " Please verify your identity with a photo! "
title " Warning "
buttonText " Understood "
```

This code updates the image to display whatever photo the user selects or takes. When the user completes this

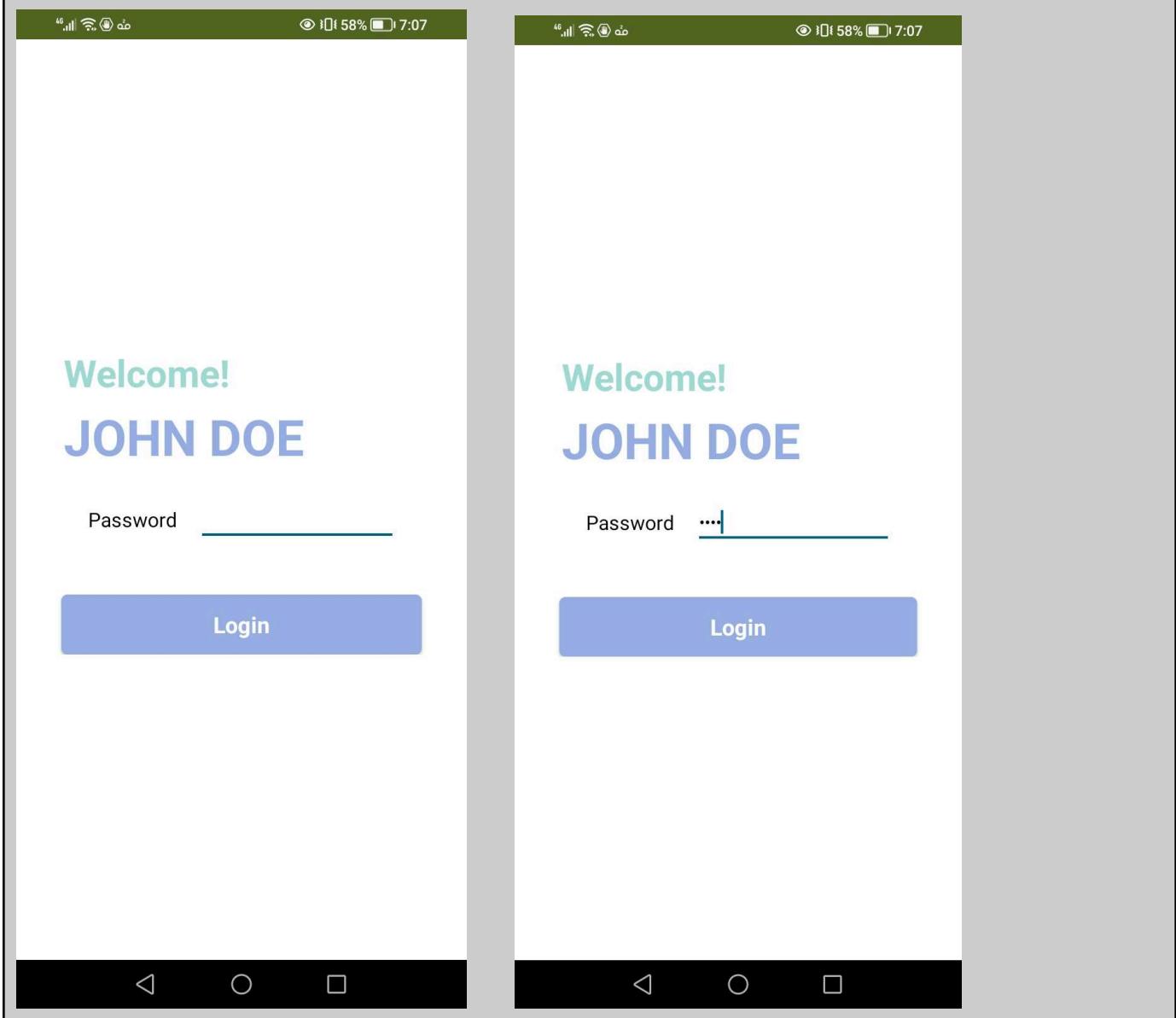
Register2

registration, the program will validate the input. If the input is valid, the photo file path is stored in the database and the user is redirected to screen1 of the program. If the input is invalid, the program calls the notifier to give a warning.

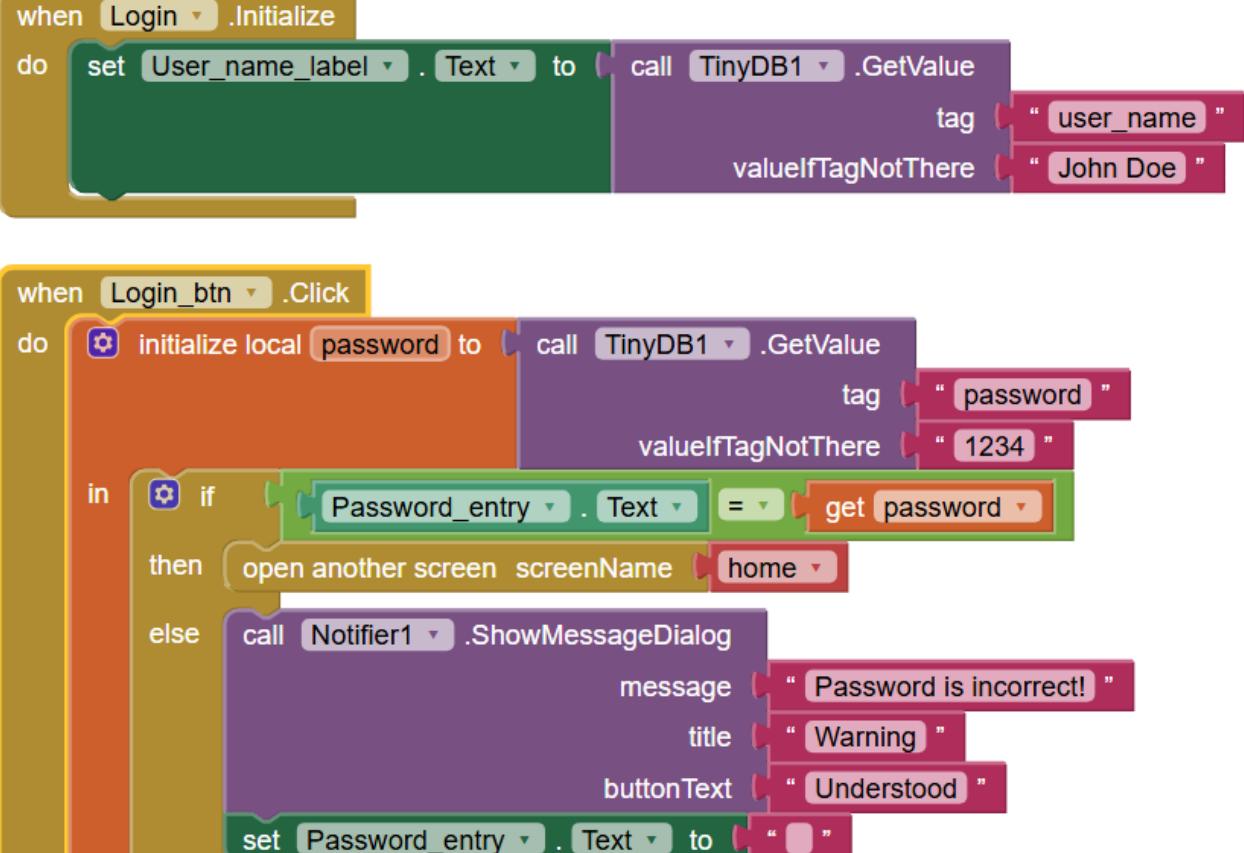
Login

User Interface	Name/Purpose					
Buttons	Login_btn If pressed, check if the user password is correct. If correct, send the user to home. If incorrect, call the notifier.					
Password Textbox	Password_entry Allows the user to enter their password.					
Labels	Welcome_label Displays the text "Welcome!"	User_name_label Displays the user's name	Label9 Provides spacing for the login entry and header	Password_label Displays the text "Password".	Label6 Provides spacing between the Password label and entry	Label5 Provides spacing between the login button and the password entry field
TinyDB	TinyDB1 The main database of the app. Stores the user's account information used in verifying the login.					
Notifier	Notifier1 Display warning messages if login is incorrect.					

Login



Login



```
when [Login] initialized
do
  set [User_name_label v] to [Text]
  call [TinyDB1 v].GetValue
    tag ["user_name"]
    valueIfTagNotThere ["John Doe"]

when [Login_btn] clicked
do
  initialize local [password] to [1234]
  call [TinyDB1 v].GetValue
    tag ["password"]
    valueIfTagNotThere ["1234"]
  if [Password_entry v].Text = [get password]
    then open another screen [home]
  else
    call [Notifier1 v].ShowMessageDialog
      message ["Password is incorrect!"]
      title ["Warning"]
      buttonText ["Understood"]
    set [Password_entry v].Text to [ ]

```

When the user presses the login button, the program gets the inputted password and checks it against the account password stored in the database. If the password is correct, the program sends the user to the home screen. If not correct, the program calls the notifier to send a warning.

Home

User Interface	Name/Purpose			
Buttons	Title	Chat_btn	Planpedia_btn	Schedule_btn

Home				
	Non functional button used for the rounded border aesthetics like a label. Displays the text "Home".	Send the user to the chat screen.	Send the user to the plantpedia screen.	Sends the user to the schedule screen.
Image	Logo Displays the app logo.			
Labels	Label4 Provides spacing for the header.	Label1 Provides horizontal spacing between navigation buttons	Label2 Provides vertical spacing between navigation buttons	

Home

4G WiFi Battery

⌚ 30% 58% 7:07

Home

Planpedia

Schedule

Team Chat

Log Out



Home

```
when Chat_btn .Click
do open another screen screenName Chat

when Schedule_btn .Click
do open another screen screenName Schedule

when Planpedia_btn .Click
do open another screen screenName Planpedia

when Logout_btn .Click
do open another screen screenName Screen1
```

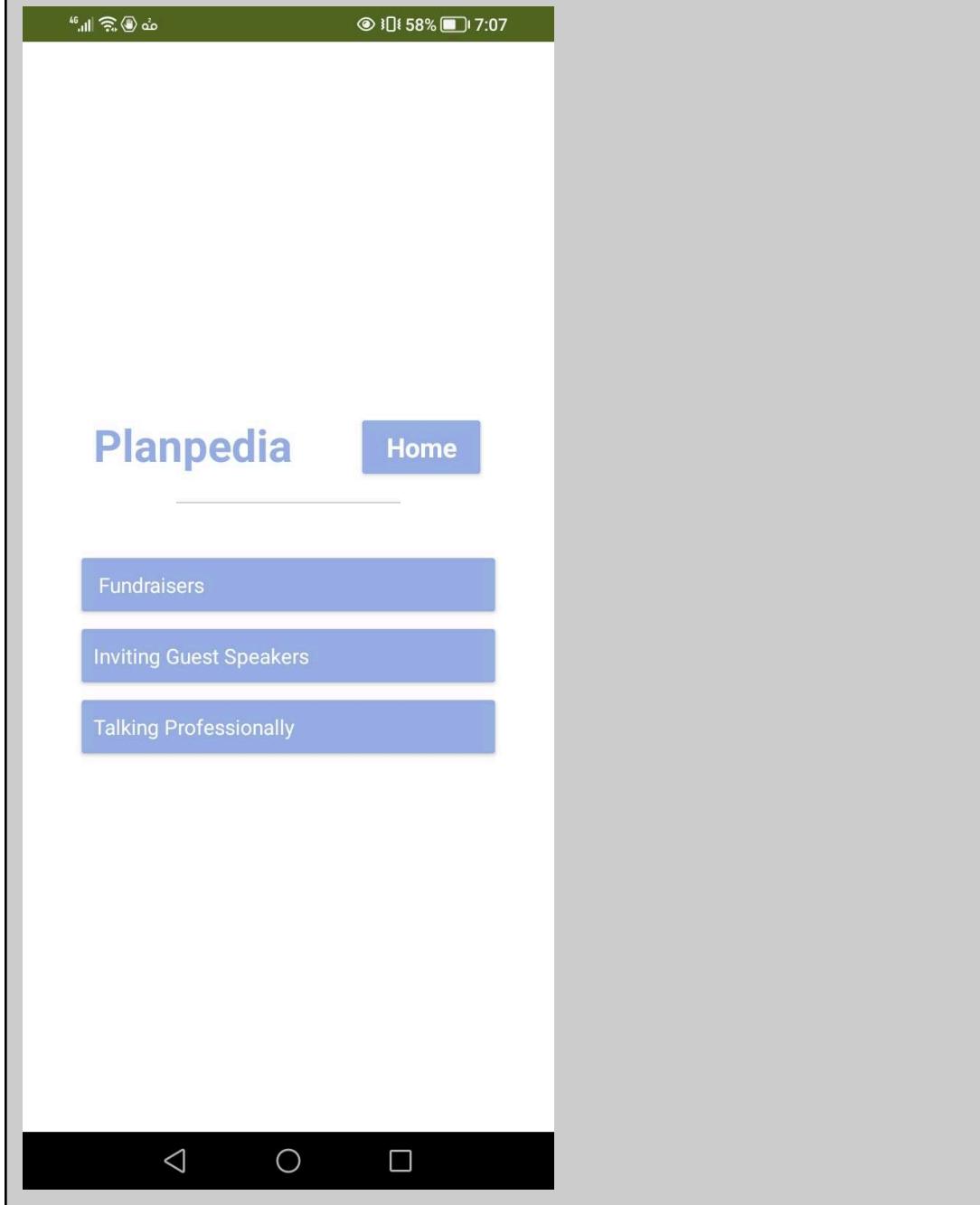
Depending on the button pressed, the program sends the user to the appropriate screen.

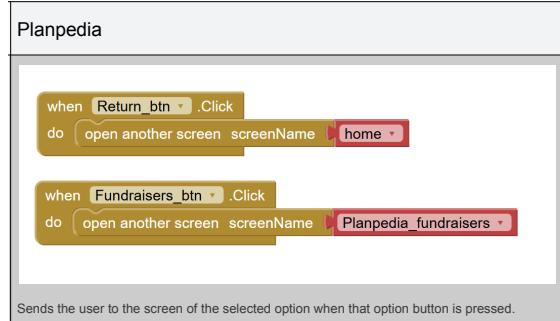
Planpedia

User Interface	Name/Purpose			
Buttons	Return_btn Sends user to the home screen	Fundraisers_btn Sends users to the Planpedia fundraiser article screen.	Guest_speak_btn Sends users to the planpedia guest speaker article. (Does not exist yet)	Talking_btn Send users to the plantpedia public speaking article. (Does not exist yet)
Password Textbox	PasswordTextbox 1 Non Functional text box used to place an aesthetic			

Planpedia

	line.			
Labels	Title Displays the text “Plantpedia”.	Label1 Provides spacing for header		





Planpedia_fundraiser												
User Interface	Name/Purpose											
Buttons	Return_btn Sends user to the plantpedia hub screen											
Password Textbox	PasswordTextbox 1 Non Functional text box used to place an aesthetic line.											
Labels	Title Displays the text "Fundraiser".	Subheading1 Subheader for introduction.	Paragraph1 Introduction article.	Subheading2 Subheader for product.	Paragraph2 Product article.	Subheading3 Subheader for finance.	Paragraph3 Finance article.	Subheading4 Subheader for human resources.	Paragraph4 Human resources article	Subheading5 Subheader for publicity.	Paragraph5 Publicity articles.	Label2 Bottom spacing for the text articles.

The image displays two side-by-side screenshots of a mobile application interface, likely from an Android device, showing a fundraising guide. Both screens have a top status bar with signal strength, battery level at 58%, and the time 7:08.

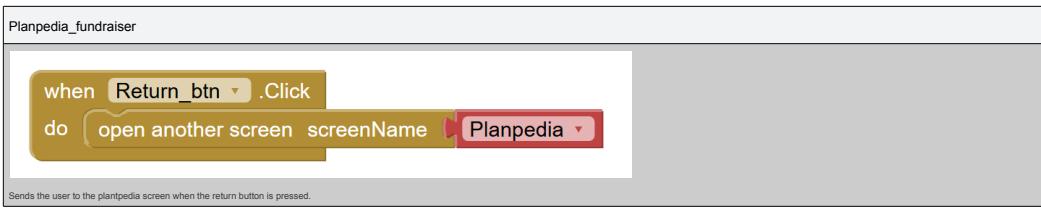
Left Screen Content:

- Section Headers:** "Fundraisers" and "Return".
- Introduction:** A brief paragraph explaining that fundraising is a vital part of running a healthy club and providing a guide to running a product-based fundraiser.
- Product:** A section suggesting cold food products or small trinkets as items to sell.
- Finance:** A section discussing the cost of goods sold, per-unit cost, and a 10% rule of thumb for profit.
- Human Resources:** A section noting the need for club members to prepare the product and staff the selling booth.
- Marketing:** A section on advertising, mentioning physical locations like sign-wavers and posters, and online platforms like social media.

Right Screen Content:

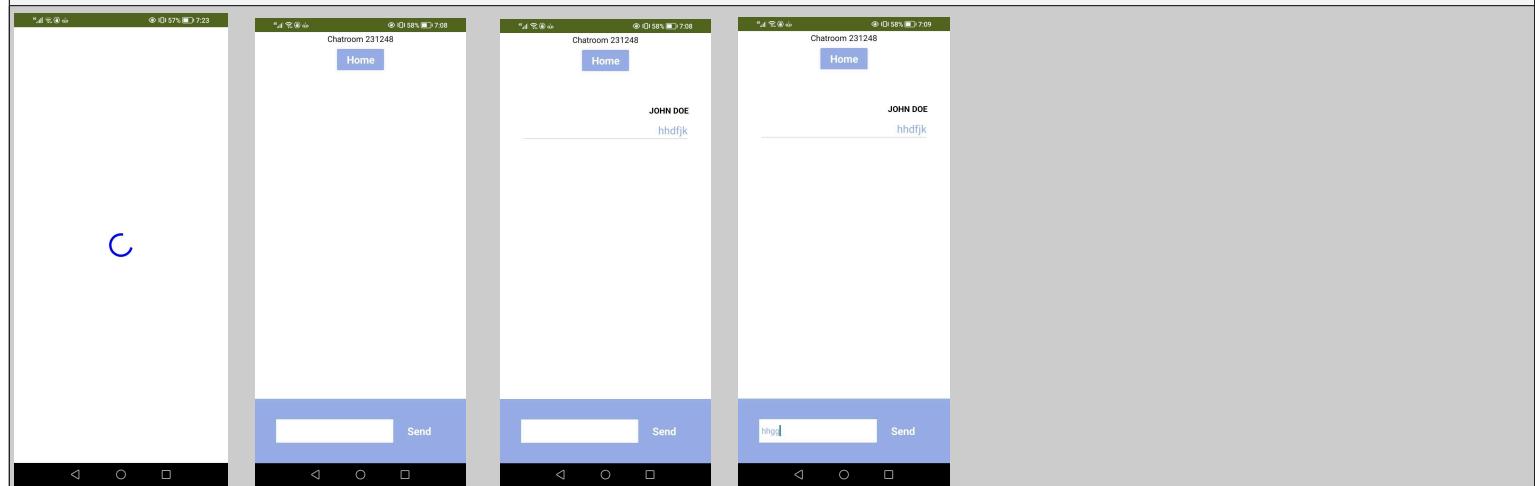
- Section Headers:** "Fundraisers" and "Return".
- Text:** A detailed paragraph about budget allocation, starting with identifying per-unit costs, calling suppliers, and checking online resources for material costs. It suggests overestimating costs by 10% and adding a 1-5 dollar markup to the sale price.
- Human Resources:** A section reiterating the need for club members to prepare the product and staff the booth.
- Marketing:** A section reiterating the need for both physical and online advertising efforts.

The bottom of each screenshot shows a black navigation bar with standard Android icons for back, home, and recent apps.



Chat			
User Interface	Name/Purpose		
Buttons	Return_btn Sends user to the home screen	Send_btn Displays the message that a user types.	
TinyDB	TinyDB1 Fetches the user's chat ID data.		
Circular progress	Circularprogress1 A spinning wheel displayed while loading the chat system.		
Textbox	Textbox6 Displays the message a user sends.	Textbox5 Displays the message a user sends.	Msg_entry Allows the user to type in a message.
Labels	Id_display_label Displays the ID of the current chat.	Label6 Empty text box used to display the user's name when they send a message.	Label5 Empty text box used to display the user's name when they send a message.

Chat



C

Chat

when Return_2m → Click

do Open another screen [screenName: home]

when Chat_2 Initialize

do Initialize local [chat_id] to call TinyDB1 GetValue

tag: chat_id

valueIfTagNotThere: Does_not_exist

in

if

compare test: get_chat_id

with: Does_not_exist

then open another screen [screenName: Open_new_chat]

else set id_display_other.Text to join [Chatterbox: get_chat_id]

set CircularProgress1.Visible to false

set VerticalManagement2.Visible to true

end

when Send_2m → Click

do if

not [Label0.Visible: true]

set Label0.Visible to true

set Label0.Text to call TinyDB1.GetValue

tag: user_name

valueIfTagNotThere: John Doe

set TextBox0.Text to Tag_entry.Text

set TextBox0.Visible to true

else if

Label0.Visible: true

set Label0.Visible to true

set Label0.Text to call TinyDB1.GetValue

tag: user_name

valueIfTagNotThere: John Doe

set TextBox0.Text to Msg_entry.Text

set TextBox0.Visible to true

set Msg_entry.Text to "

When the screen initializes, the program checks if the user is in a chat room via the chat_id tag in the database. If the user is not in a chatroom, the user is redirected to the open_new_chat screen. If the user is already in a chat, the program hides the spinning loading wheel and reveals the chatroom UI elements. The user can now interact with the program via two buttons.

The return button will send the user back to the home screen. The send button will get the text stored in the msg_entry written by the user and paste it into the chatroom along with the user's name stored in the database.

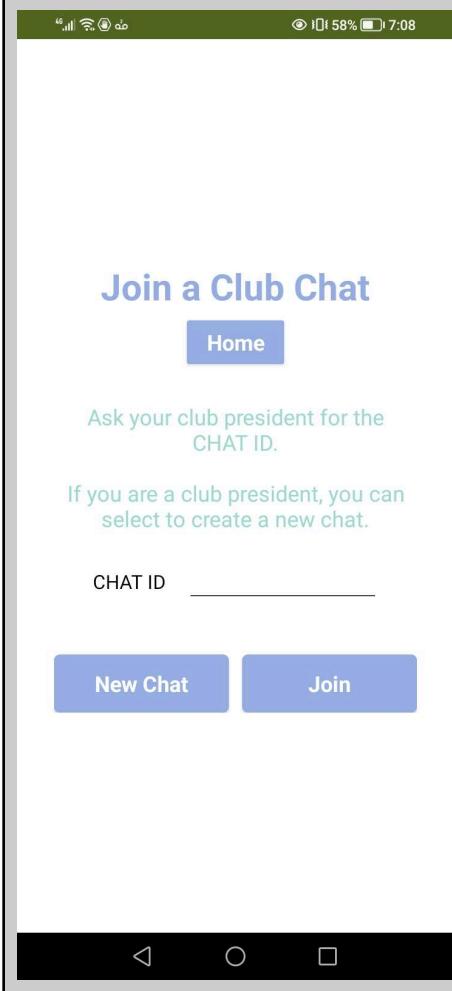
Open_new_chat

User Name/Purpose

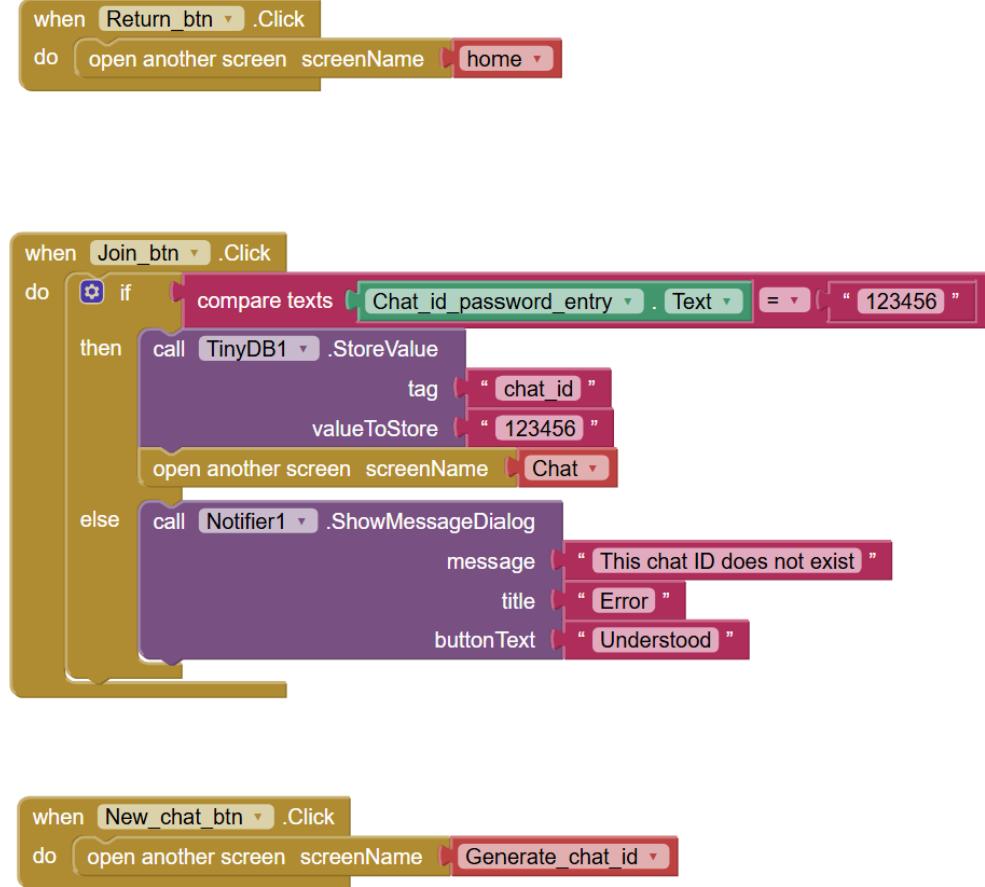
</

Open_new_chat								
Interface								
Buttons	Return_btn Send the user to the home screen.	New_chat_btn Sends the user to the Generate_chat_id screen.	Join_btn Stores the given chat ID and redirects the user to the chat screen.					
TinyDB	TinyDB1 Stores the entered chat ID into the user's profile.							
Notifier	Notifier1 Warns the user if the chat ID is invalid.							
Password Textbox	Chat_id_password_entry Provides a textbox for the user to enter their Chat ID.							
Labels	Title Displays the text "Join a Club Chat".	Label11 Provides title spacing.	Info_label Displays instructions on how to use the screen.	Label9 Provides spacing for the entry field.	Chat_id_label Displays the text "CHAT ID".	Label6 Provides spacing.	Label5 Provides spacing.	Label10 Provides spacing.

Open_new_chat



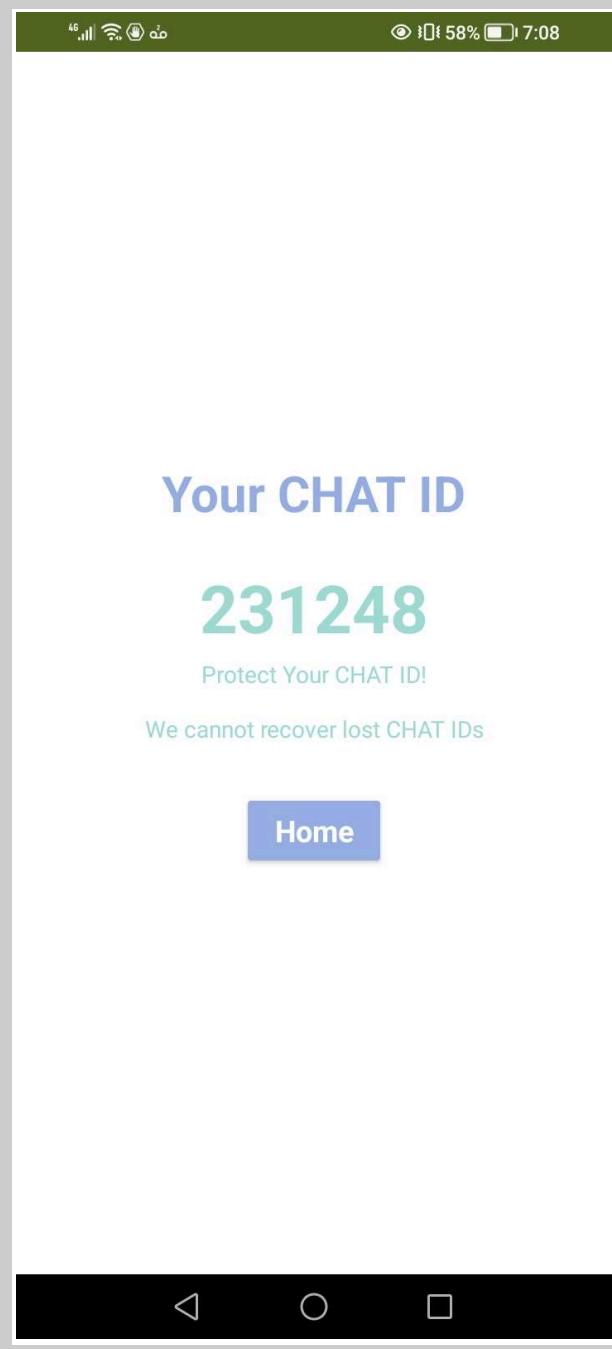
Open_new_chat



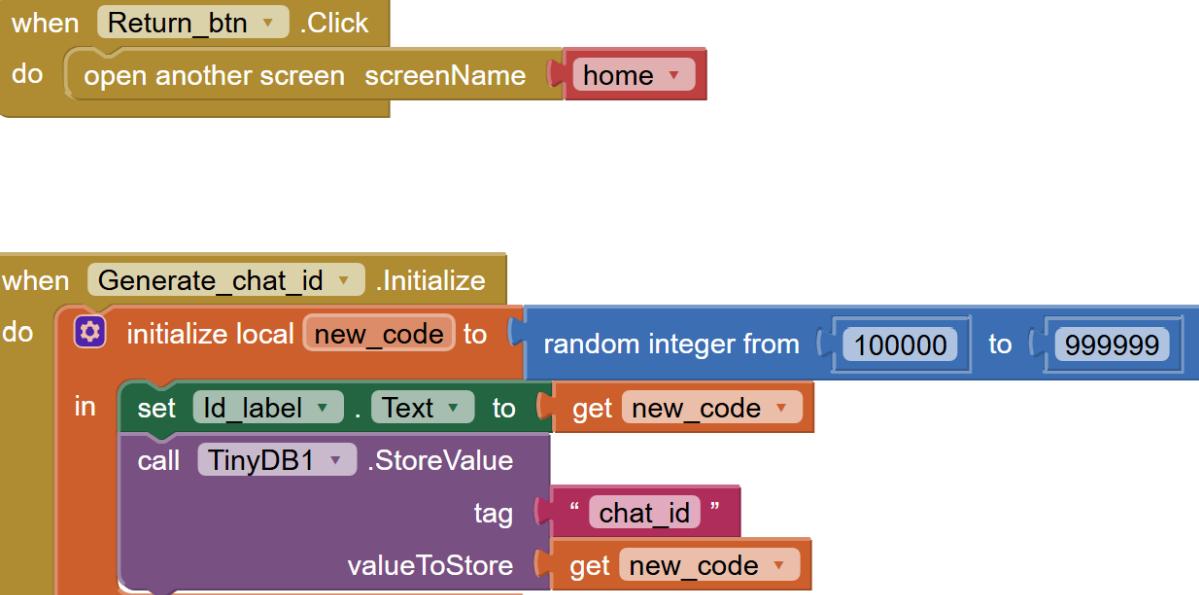
The user can press the return button and the program will send the user back to the home screen. The user can also press the new chat button and get redirected to the generate_chat_id screen. Otherwise, the user can input a chat ID into a text field. When the user presses the join button, the program will get the imputed 6-digit value and check if the ID is valid; the default chat ID I set was 123456. If the chat ID is valid, the program stores the ID into the database. If not, the notifier is called to give a warning.

Generate_chat_id					
User Interface	Name/Purpose				
Buttons	Return_btn Send the user to the home screen.				
TinyDB	TinyDB1 Stores the generated chat ID into the user's profile.				
Labels	Title Displays the text "Your CHAT ID".	Id_label Displays the generated chat ID.	Disclaimer_label Warns the user that the chat ID is not recoverable.	Label11 Provides spacing.	Label15 Provides spacing.

Generate_chat_id



Generate_chat_id



The Scratch script consists of two main sections:

- when Return_btn .Click:** Triggers a "do" block that opens the "home" screen.
- when Generate_chat_id .Initialize:** Triggers a "do" block that:
 - Initializes a local variable "new_code" to a random integer between 100000 and 999999.
 - Sets the text of the "Id_label" to the value of "new_code".
 - Stores the value of "new_code" under the tag "chat_id" in the TinyDB1 database.

When the screen is initialized, the program pseudo randomly generates an integer from 100000 to 999999. The generated number is then displayed as the text of the ID label and stored to the database as the user's chatroom. The user can then press return to get sent to the home screen.

Schedule

User Interface	Name/Purpose				
Buttons	Return_btn Send the user to the home screen.	Clear_s1_btn Clears the entry fields and tinydb tags of all schedule1 data.			
TinyDB	TinyDB1 Fetches and stores the user's schedule data.				
Textbox	Schedule1	Schedule2	Schedule3		

Schedule					
	Allows the user to write schedules for a day.	Allows the user to write schedules for a day.	Allows the user to write schedules for a day.		
Datepicker	Datepicker1 Allows the user to pick the date for schedule1.	Datepicker2 Allows the user to pick the date for schedule2.	Datepicker3 Allows the user to pick the date for schedule3.		
Timepicker	Timepicker1 Allows the user to pick the time for schedule1.	Timepicker2 Allows the user to pick the time for schedule2.	Timepicker3 Allows the user to pick the time for schedule3.		
Sensor	Clock1 Converts timepicker and datepicker instants into strings for storage and display.				
Labels	Title Displays the text "Schedule".	Info_label Displays instructions for how to use the feature.	Label11 Provides spacing.	Label14 Provides spacing.	Label15 Provides spacing.

Schedule

4G WiFi Battery 58% 7:08

Schedule

Home

Add notes and dates! All notes will be saved.

Date

Time

Clear

Date

Time

Clear

Date

Time

Clear

4G WiFi Battery 58% 7:08

Schedule

Home

Add notes and dates! All notes will be saved.

Jun. 12, 2025

4:08:00 p.m.

Clear

Date

Time

Clear

Date

Time

Clear



Schedule

```
when Return_btn .Click
do open another screen screenName home
```

When the return button is pressed, the program sends the user to the home page.

```

when Schedule .Initialize
do
  set Schedule1 .Text to call TinyDB1 .GetValue
    tag "schedule_text_1"
    valueIfTagNotThere "N/A"
  initialize local schedule_date_1 to call TinyDB1 .GetValue
    tag "schedule_date_1"
    valueIfTagNotThere "N/A"
  in
    if get schedule_date_1 ≠ "N/A"
      then set DatePicker1 .Text to call Clock1 .FormatDate
        instant call Clock1 .MakeInstantFromMillis
          millis get schedule_date_1
          pattern "MMM d, yyyy"
  initialize local schedule_time_1 to call TinyDB1 .GetValue
    tag "schedule_time_1"
    valueIfTagNotThere "N/A"
  in
    if get schedule_time_1 ≠ "N/A"
      then set TimePicker1 .Text to call Clock1 .FormatTime
        instant call Clock1 .MakeInstantFromMillis
          millis get schedule_time_1
          pattern "HH:mm:ss"
  set Schedule2 .Text to call TinyDB1 .GetValue
    tag "schedule_text_2"
    valueIfTagNotThere "N/A"
  initialize local schedule_date_2 to call TinyDB1 .GetValue
    tag "schedule_date_2"
    valueIfTagNotThere "N/A"
  in
    if get schedule_date_2 ≠ "N/A"
      then set DatePicker2 .Text to call Clock1 .FormatDate
        instant call Clock1 .MakeInstantFromMillis
          millis get schedule_date_2
          pattern "MMM d, yyyy"
  initialize local schedule_time_2 to call TinyDB1 .GetValue
    tag "schedule_time_2"
    valueIfTagNotThere "N/A"
  in
    if get schedule_time_2 ≠ "N/A"
      then set TimePicker2 .Text to call Clock1 .FormatTime
        instant call Clock1 .MakeInstantFromMillis
          millis get schedule_time_2
          pattern "HH:mm:ss"
  set Schedule3 .Text to call TinyDB1 .GetValue
    tag "schedule_text_3"
    valueIfTagNotThere "N/A"
  initialize local schedule_date_3 to call TinyDB1 .GetValue
    tag "schedule_date_3"
    valueIfTagNotThere "N/A"
  in
    if get schedule_date_3 ≠ "N/A"
      then set DatePicker3 .Text to call Clock1 .FormatDate
        instant call Clock1 .MakeInstantFromMillis
          millis get schedule_date_3
          pattern "MMM d, yyyy"
  initialize local schedule_time_3 to call TinyDB1 .GetValue
    tag "schedule_time_3"
    valueIfTagNotThere "N/A"
  in
    if get schedule_time_3 ≠ "N/A"
      then set TimePicker3 .Text to call Clock1 .FormatTime
        instant call Clock1 .MakeInstantFromMillis
          millis get schedule_time_3
          pattern "HH:mm:ss"

```

Schedule

When the screen is initialized, the program checks if any existing data for the text strings, date instants, or time instants for any of the three schedule slots is in the database. If there exists data, the data is displayed on the UI elements.

```

when Schedule1 .TextChanged
do
  call TinyDB1 .StoreValue
    tag "schedule_text_1"
    valueToStore Schedule1 .Text

when DatePicker1 .AfterDataSet
do
  set DatePicker1 .Text to (call Clock1 .FormatDate
    instant
    pattern "MMM d, yyyy")
  call TinyDB1 .StoreValue
    tag "schedule_date_1"
    valueToStore (call Clock1 .GetMillis
      instant)
      DatePicker1 .Instant

when TimePicker1 .AfterTimeSet
do
  set TimePicker1 .Text to (call Clock1 .FormatTime
    instant)
    TimePicker1 .Instant
  call TinyDB1 .StoreValue
    tag "schedule_time_1"
    valueToStore (call Clock1 .GetMillis
      instant)
      TimePicker1 .Instant

when Clear_s1_btn .Click
do
  set Schedule1 .Text to [ ]
  call TinyDB1 .ClearTag
    tag "schedule_text_1"
  set DatePicker1 .Text to [ Date ]
  call TinyDB1 .ClearTag
    tag "schedule_date_1"
  set TimePicker1 .Text to [ Time ]
  call TinyDB1 .ClearTag
    tag "schedule_time_1"

```

The image shows four separate Scratch script blocks stacked vertically. Each block begins with a 'when' event (Schedule1.TextChanged, DatePicker1.AfterDataSet, TimePicker1.AfterTimeSet, or Clear_s1_btn.Click) and contains a 'do' block with multiple steps. The first block stores the value of the Schedule1 text input into the database with tag 'schedule_text_1'. The second block formats the date from the DatePicker1 and stores it into the database with tag 'schedule_date_1'. The third block formats the time from the TimePicker1 and stores it into the database with tag 'schedule_time_1'. The fourth block clears all three text inputs and removes the associated data tags from the database.

Detects changes to the schedule 1 text, timepicker, and datepicker to store into the database. When the clear schedule 1 button is pressed, it clears the text fields of the data and removes the data associated with schedule 1

Schedule

from the database.

```

when Schedule2 .TextChanged
do call TinyDB1 .StoreValue
    tag "schedule_text_2"
    valueToStore Schedule2 . Text

when DatePicker2 .AfterDateSet
do set DatePicker2 . Text to call Clock1 .FormatDate
    instant DatePicker2 . Instant
    pattern " MMM d, yyyy "
call TinyDB1 .StoreValue
    tag "schedule_date_2"
    valueToStore call Clock1 .GetMillis
        instant DatePicker2 . Instant

when TimePicker2 .AfterTimeSet
do set TimePicker2 . Text to call Clock1 .FormatTime
    instant TimePicker2 . Instant
call TinyDB1 .StoreValue
    tag "schedule_time_2"
    valueToStore call Clock1 .GetMillis
        instant TimePicker2 . Instant

when Clear_s2_btn .Click
do set Schedule2 . Text to " "
call TinyDB1 .ClearTag
    tag "schedule_text_2"
set DatePicker2 . Text to " Date "
call TinyDB1 .ClearTag
    tag "schedule_date_2"
set TimePicker2 . Text to " Time "
call TinyDB1 .ClearTag
    tag "schedule_time_2"

```

Schedule

Detects changes to the schedule2 text, timepicker, and datepicker to store into the database. When the clear schedule 2 button is pressed, it clears the text fields of the data and removes the data associated with schedule 2 from the database.

```
when Schedule3 .TextChanged
do call TinyDB1 .StoreValue
    tag "schedule_text_3"
    valueToStore Schedule3 .Text
```

```
when DatePicker3 .AfterDataSet
do set DatePicker3 .Text to call Clock1 .FormatDate
    instant DatePicker3 .Instant
    pattern "MMM d, yyyy"
call TinyDB1 .StoreValue
    tag "schedule_date_3"
    valueToStore call Clock1 .GetMillis
        instant DatePicker3 .Instant
```

```
when TimePicker3 .AfterTimeSet
do set TimePicker3 .Text to call Clock1 .FormatTime
    instant TimePicker3 .Instant
call TinyDB1 .StoreValue
    tag "schedule_time_3"
    valueToStore call Clock1 .GetMillis
        instant TimePicker3 .Instant
```

```
when Clear_s3_btn .Click
do set Schedule3 .Text to " "
call TinyDB1 .ClearTag
    tag "schedule_text_3"
set DatePicker3 .Text to "Date"
call TinyDB1 .ClearTag
    tag "schedule_date_3"
set TimePicker3 .Text to "Time"
call TinyDB1 .ClearTag
    tag "schedule_time_3"
```

Schedule

Detects changes to the schedule 3 text, timepicker, and datepicker to store into the database. When the clear schedule 3 button is pressed, it clears the text fields of the data and removes the data associated with schedule 3 from the database.