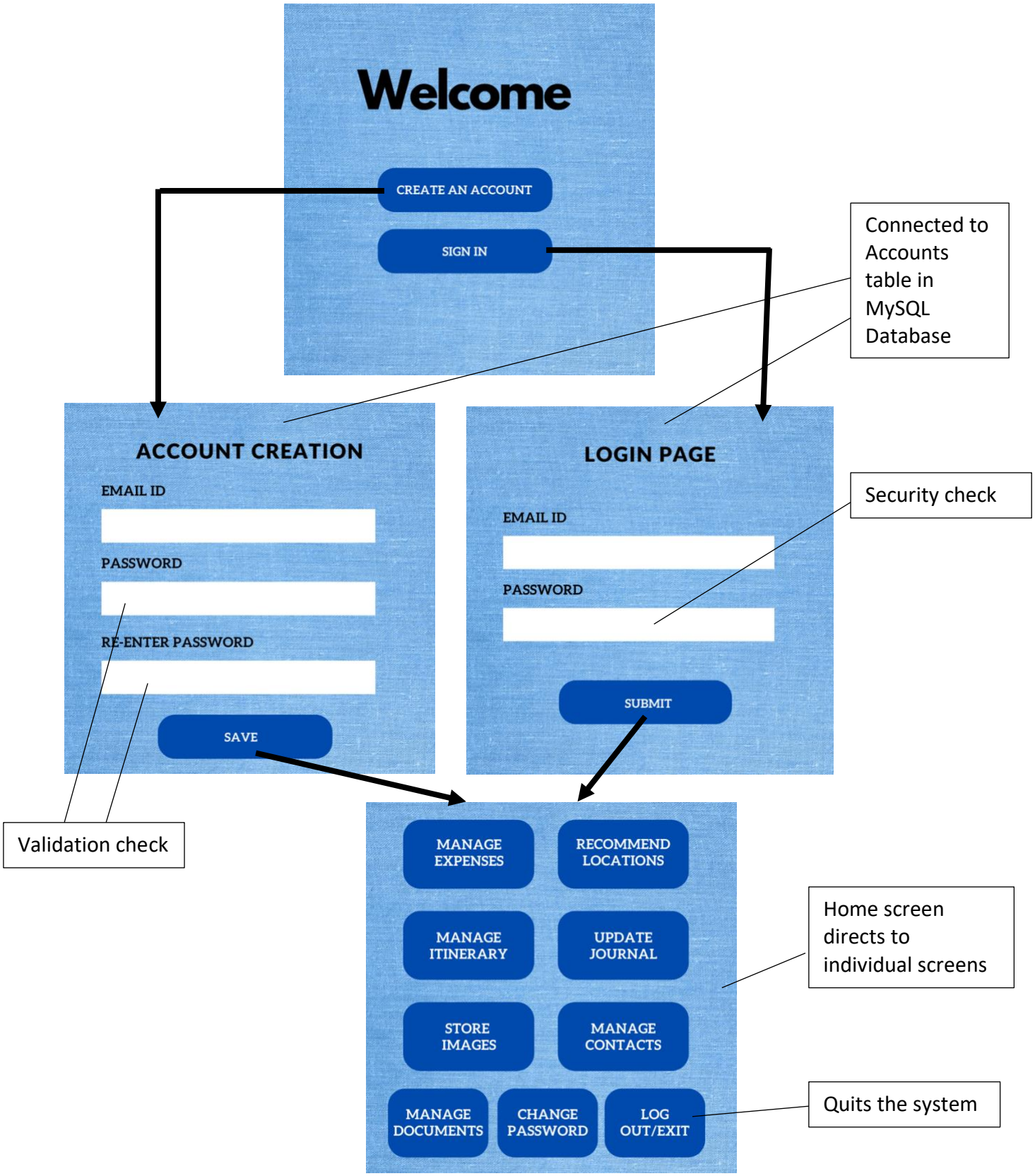
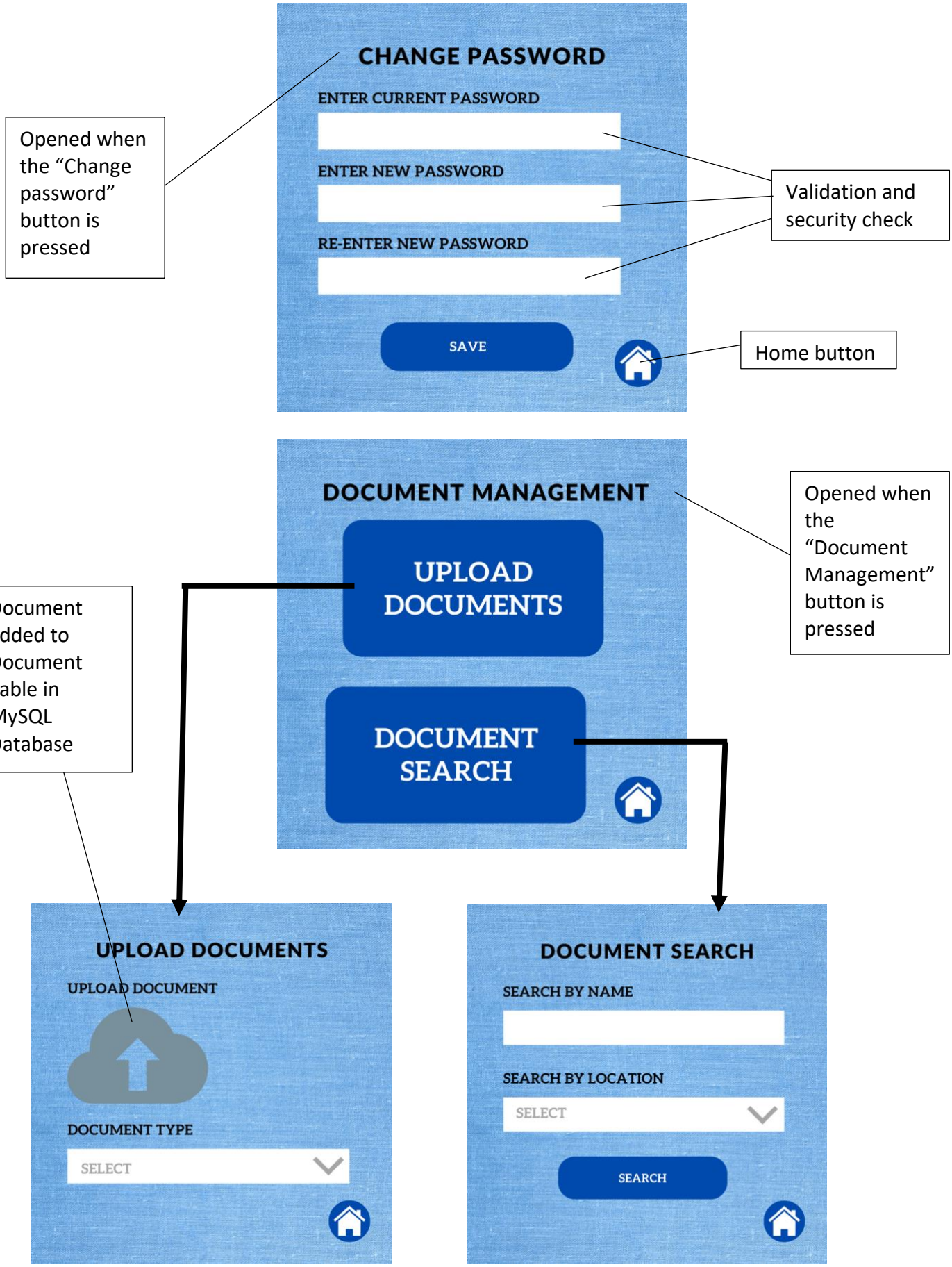
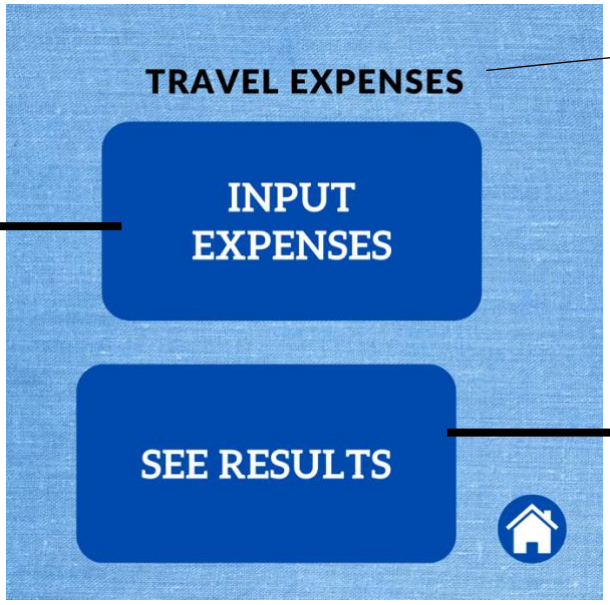
Section B: Design

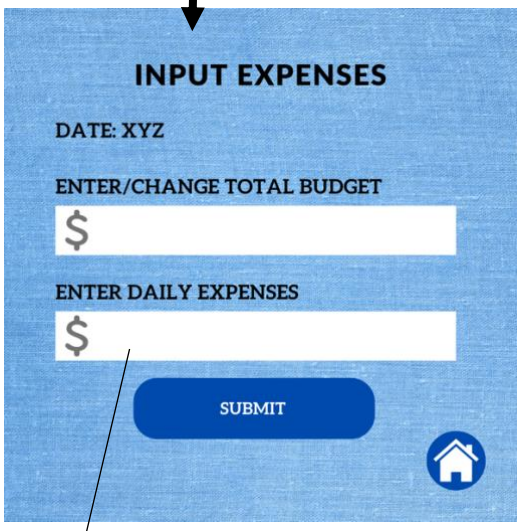
Prototype




CHANGE PASSWORD
ENTER CURRENT PASSWORD
ENTER NEW PASSWORD
RE-ENTER NEW PASSWORD
SAVE
Opened when the "Change password" button is pressed
Validation and security check
Home button
DOCUMENT MANAGEMENT
UPLOAD DOCUMENTS
DOCUMENT SEARCH
Opened when the "Document Management" button is pressed
Document added to Document Table in MySQL Database
UPLOAD DOCUMENTS
UPLOAD DOCUMENT
DOCUMENT TYPE
SELECT
DOCUMENT SEARCH
SEARCH BY NAME
SEARCH BY LOCATION
SELECT
SEARCH

Opened when the “Travel Expenses” button is pressed

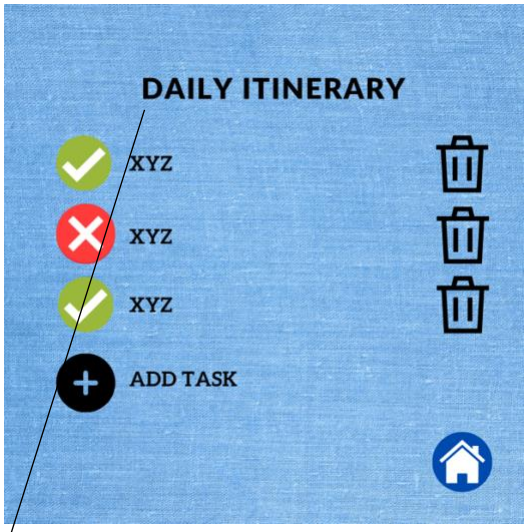




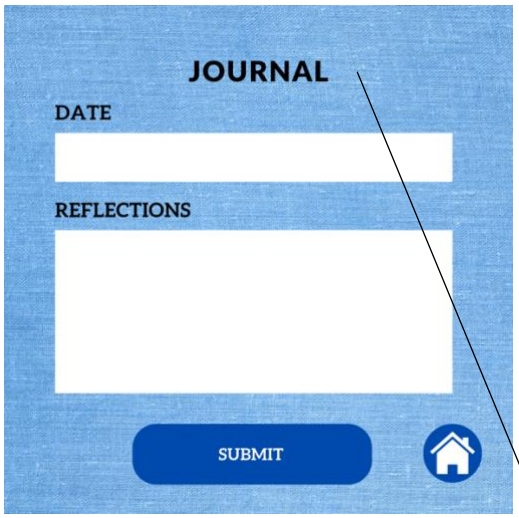
Expenses Table in MySQL Database updated daily with inputted data

Recommendations based on hotspots in a 5km square around the location, taken from the Google Maps Database

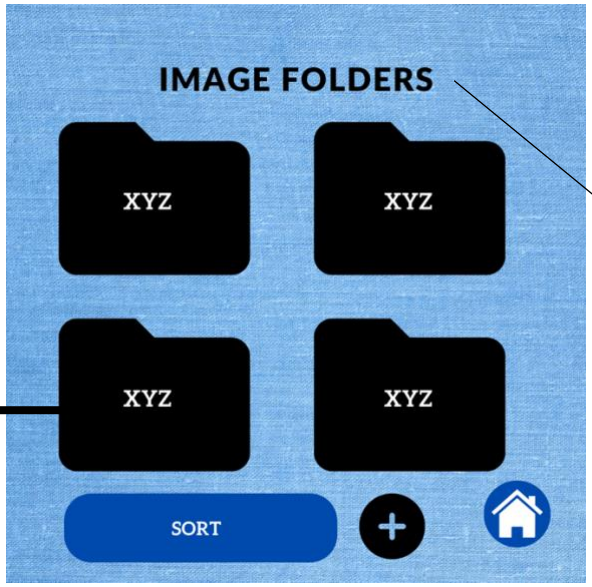
Opened when the “Location Recommendations” button is pressed



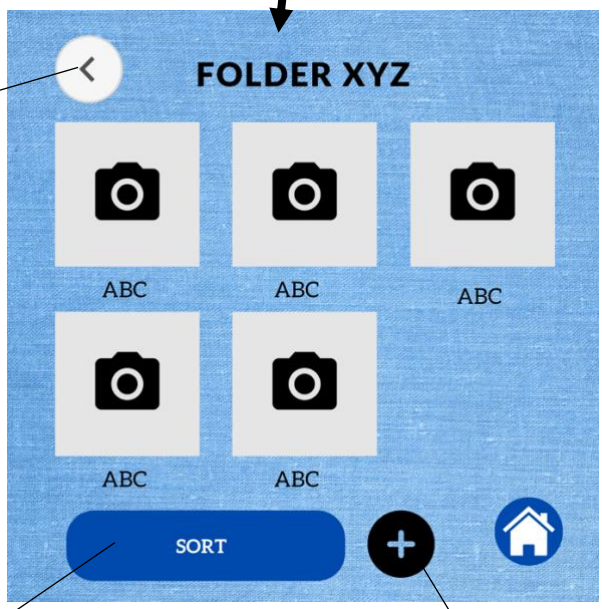
Opened when the "Manage Itinerary" button is pressed



Opened when the "Update Journal" button is pressed



Opened when the "Store images" button is pressed

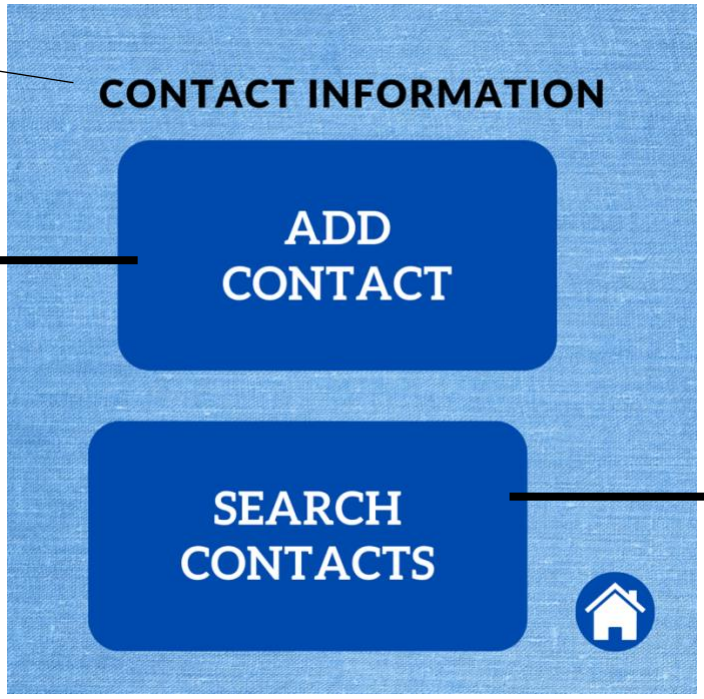


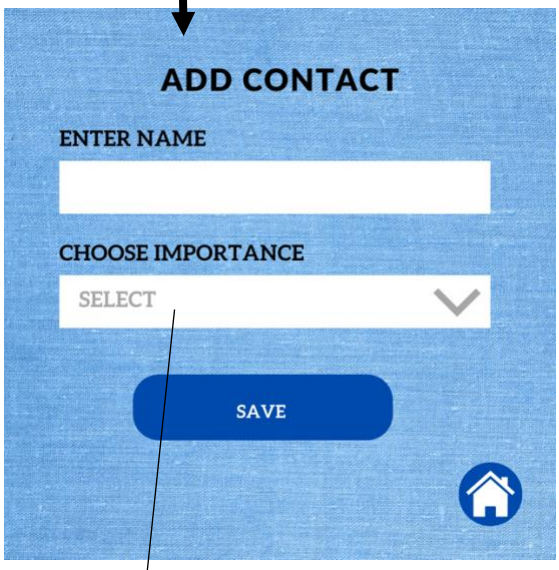
Back button

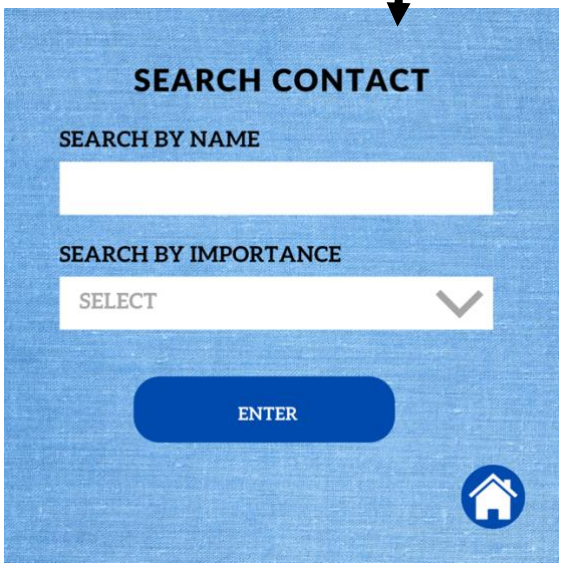
Sorts by name

Add new image

Opened when the "Manage Contacts" button is pressed

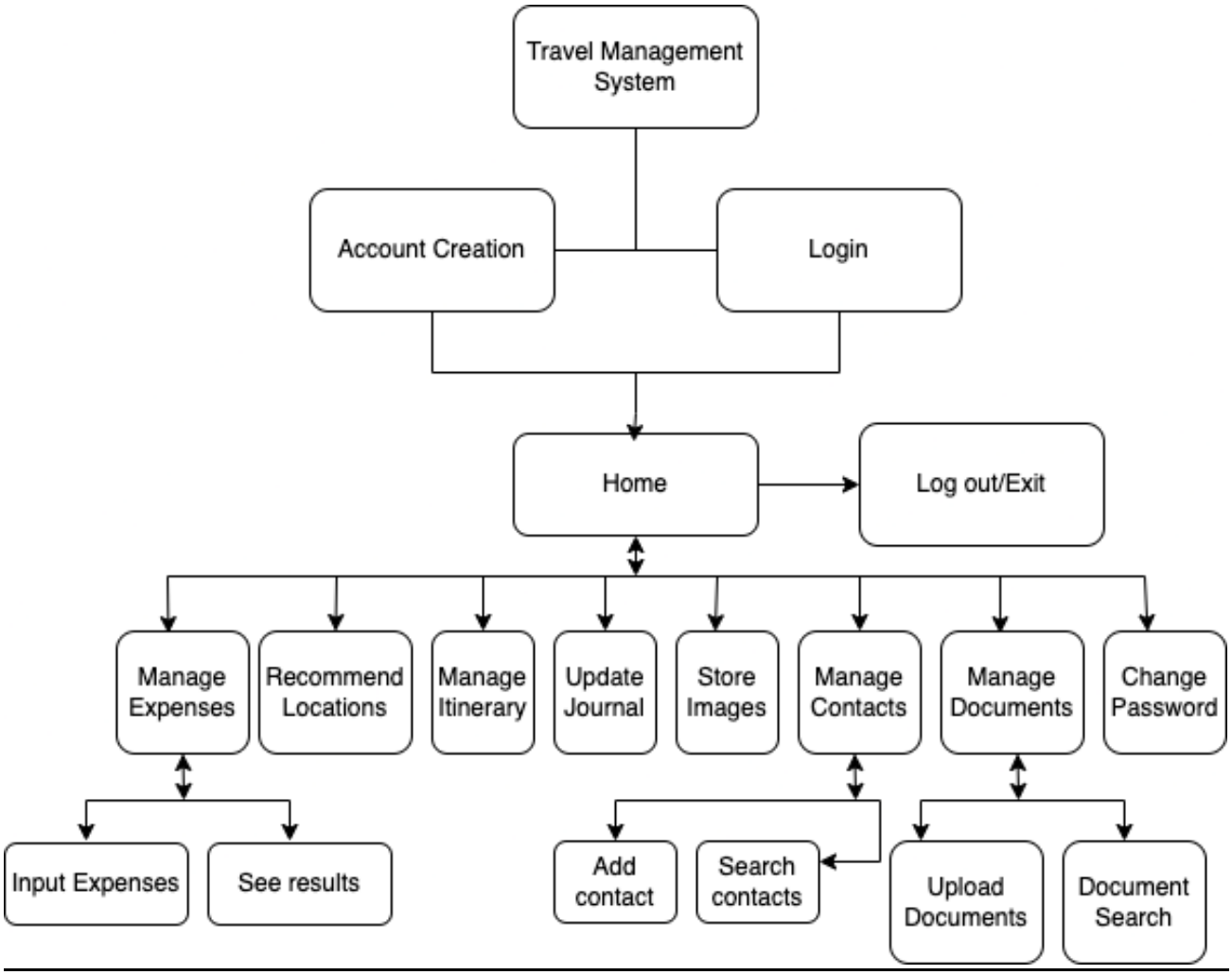




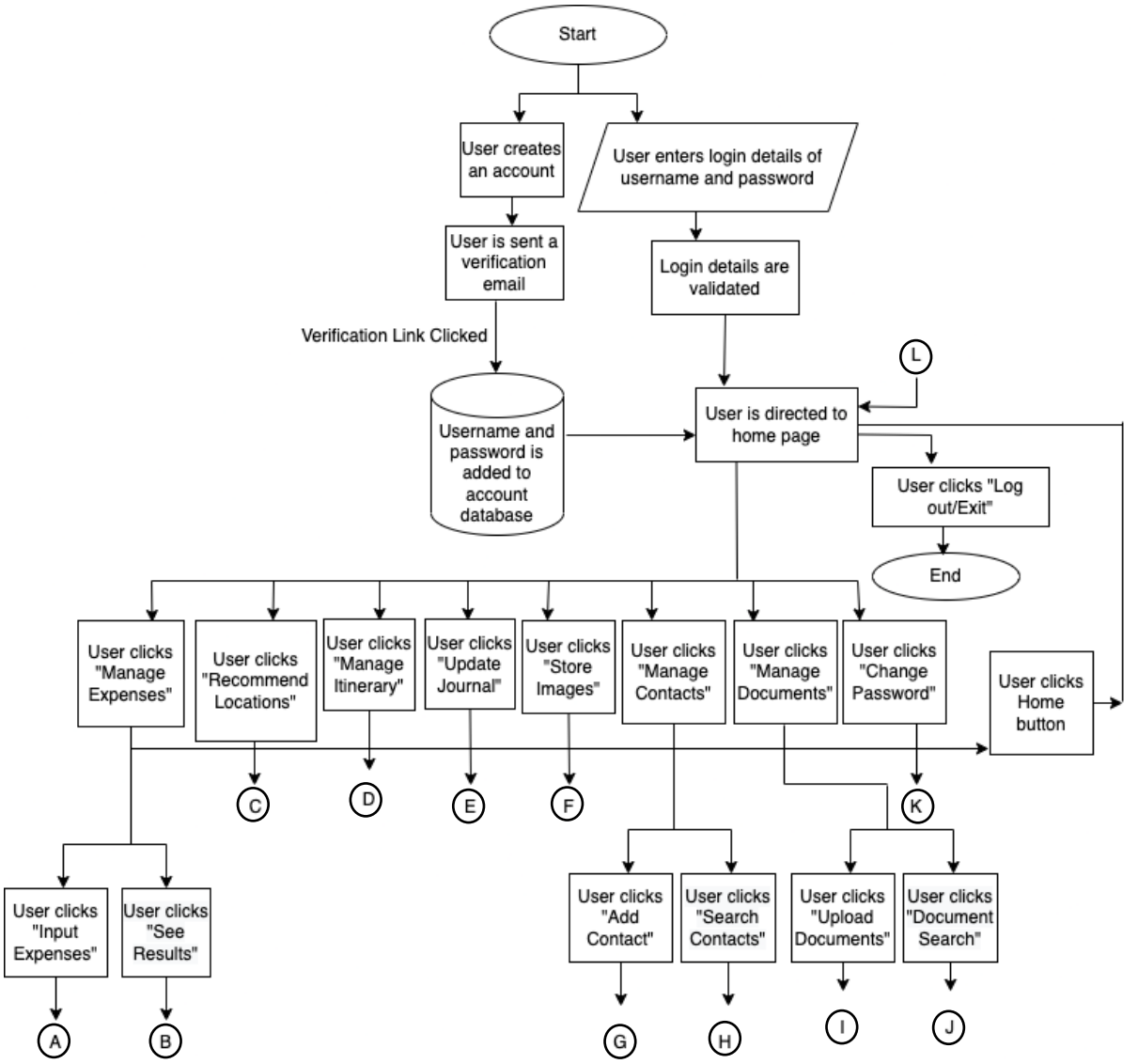


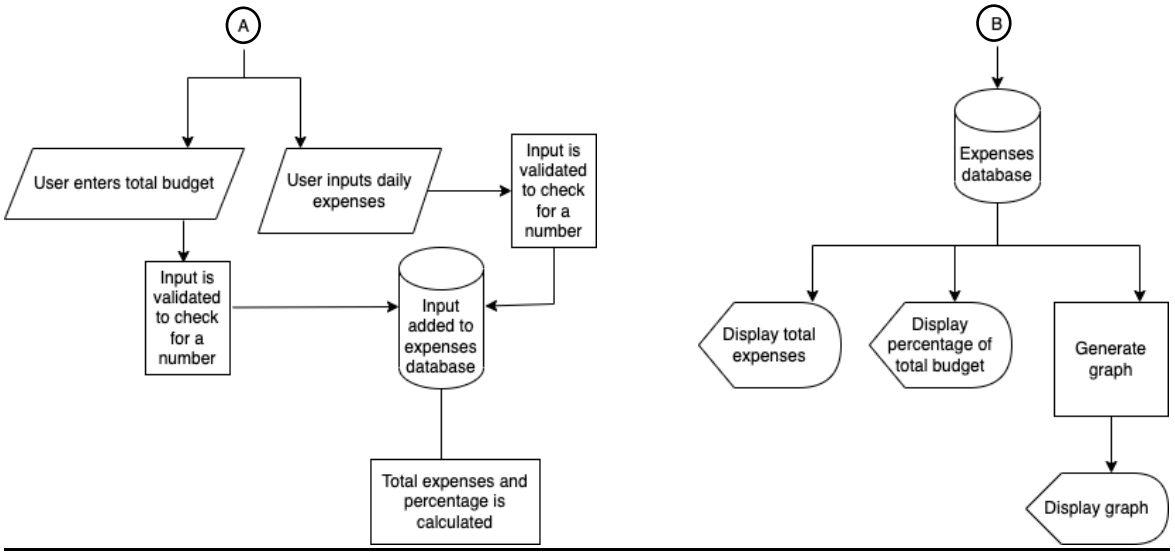
Importance can be high, medium or low

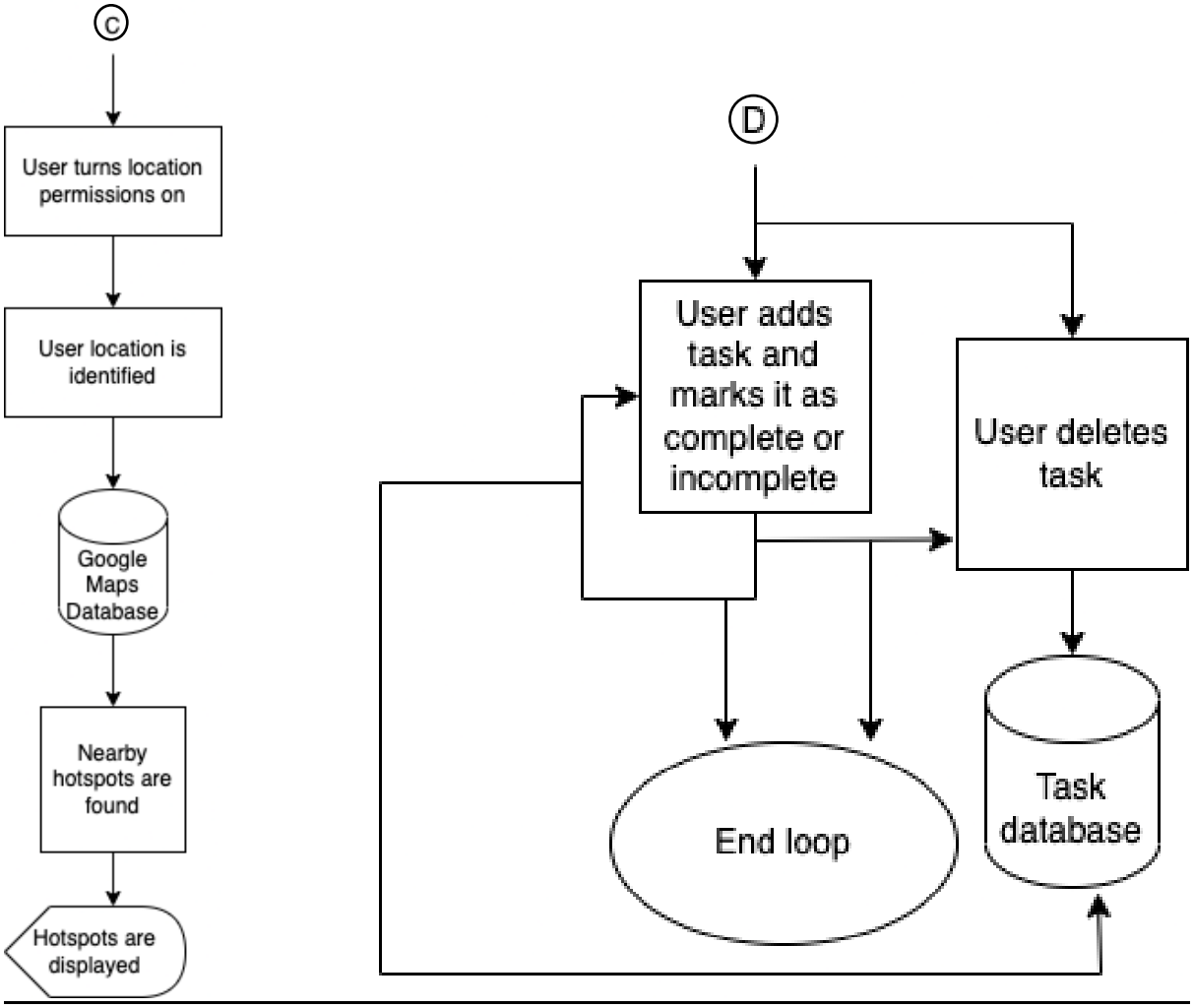
Structured Diagram

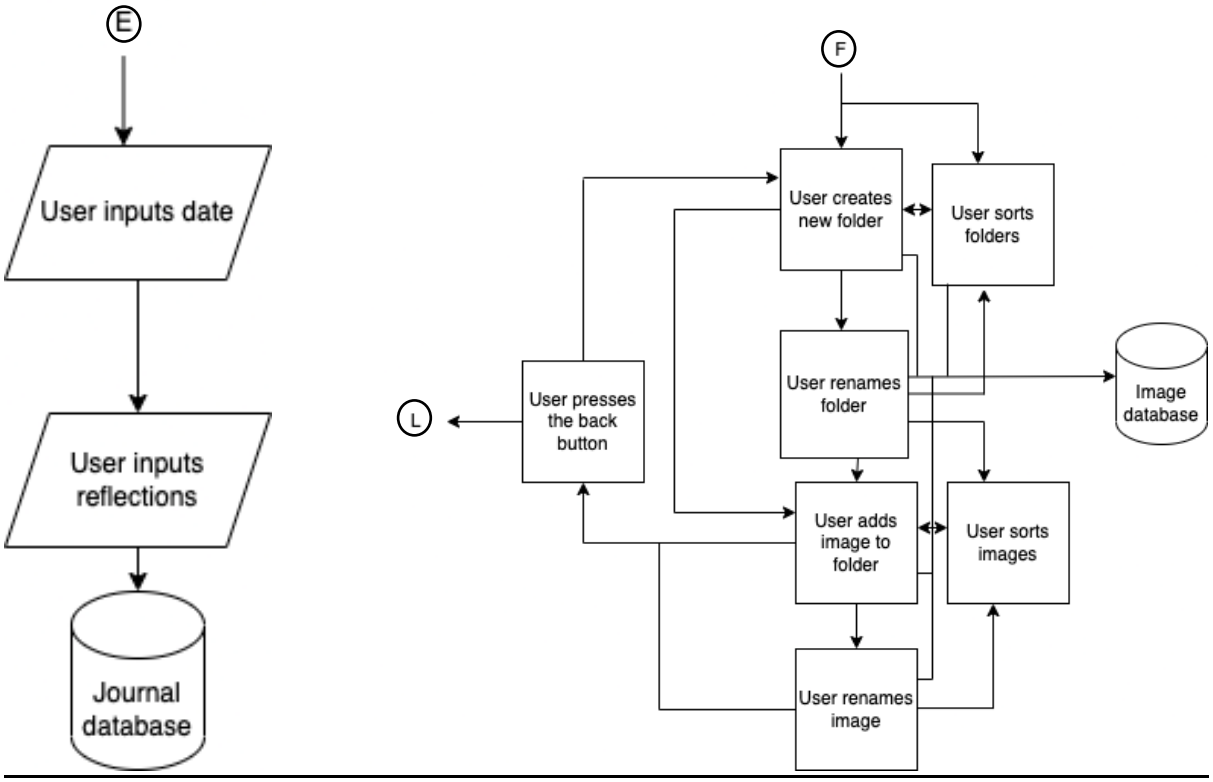


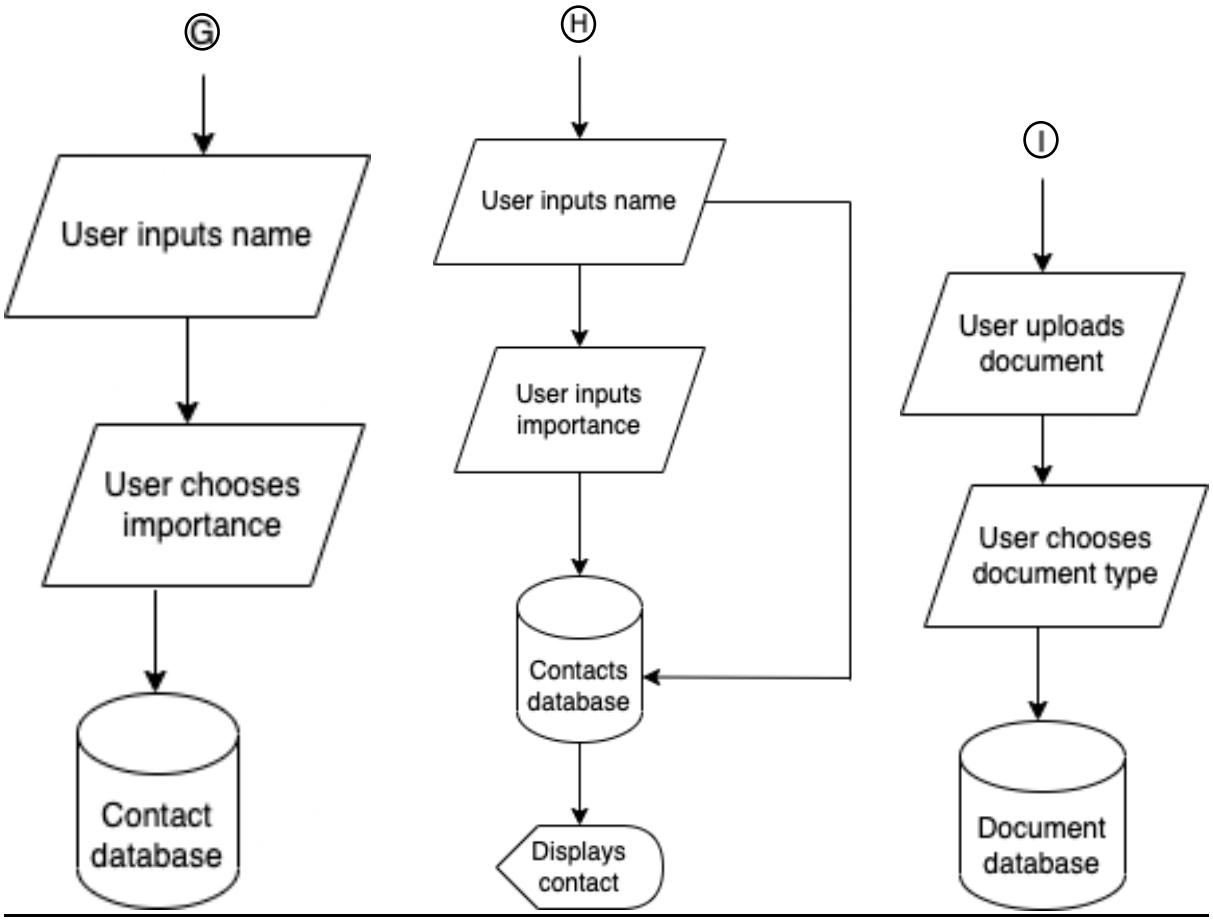
System Flowcharts

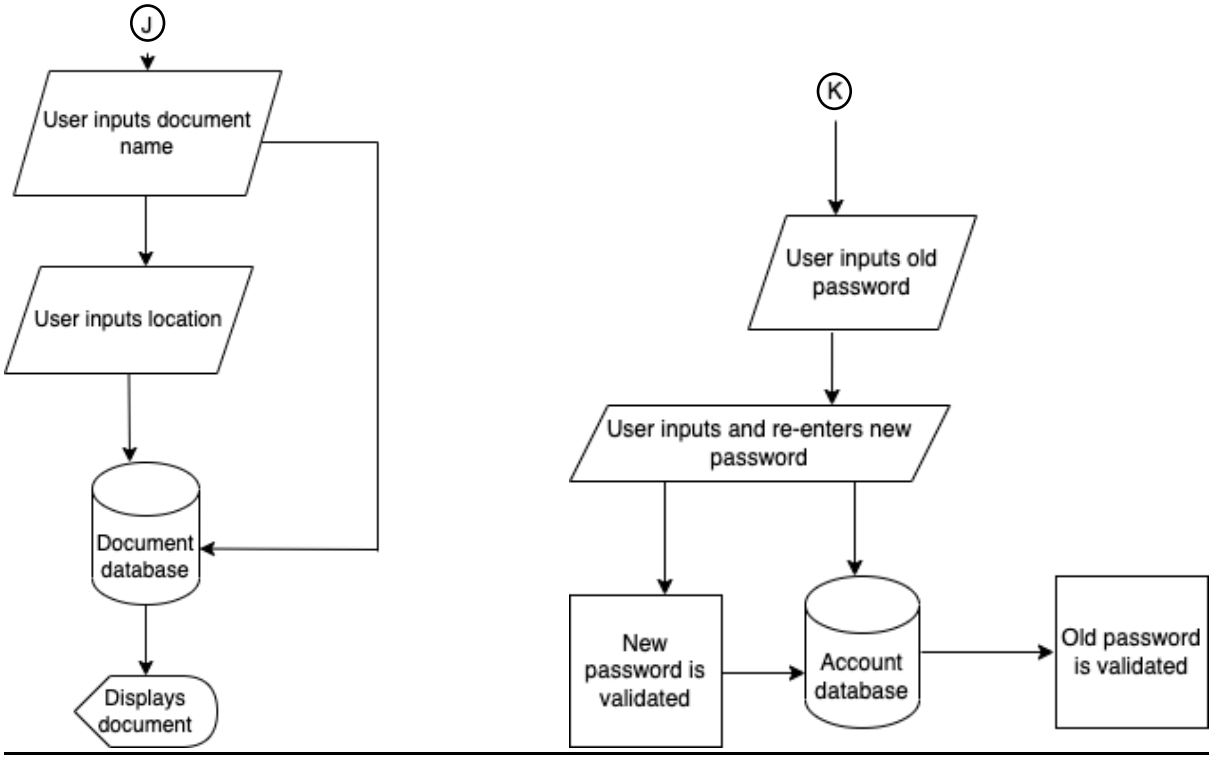





C
User turns location permissions on
User location is identified
Google Maps Database
Nearby hotspots are found
Hotspots are displayed
D
User adds task and marks it as complete or incomplete
User deletes task
End loop
Task database


E
User inputs date
User inputs reflections
Journal database
F
User creates new folder
User sorts folders
User renames folder
Image database
L
User presses the back button
User adds image to folder
User sorts images
User renames image


G
User inputs name
User chooses importance
Contact database
H
User inputs name
User inputs importance
Contacts database
Displays contact
I
User uploads document
User chooses document type
Document database


J
User inputs document name
User inputs location
Document database
Displays document
K
User inputs old password
User inputs and re-enters new password
New password is validated
Account database
Old password is validated

Relationship Tables

Account Table
Purpose: To store the details of all accounts created.

Field Name	Data Type	Description
id	Integer	Stores a unique id generated for every new account created.
username	String	Stores the username entered by a user who creates an account.
password	String	Stores the password entered by a user who creates an account. Password is updated if user changes password.

Expenses Table
Purpose: To store the information of all expenses made by the user.

Field Name	Data Type	Description
id	Integer	Stores a unique id generated for every new account created.
budget	Double	Stores the value of the travel budget entered by the user.
dailyExpenses	Array (double)	Stores the daily expenses entered by the user at a unique index.
totalExpenses	Double	Stores the total expenses calculated on a daily basis by aggregating the daily expenses entered by the user.

Task Table
Purpose: To store the tasks entered by the user as part of their travel itinerary.

Field Name	Data Type	Description
id	Integer	Stores a unique id generated for every new account created.
tasks	Array (string)	Stores each task added by the user at a unique index. Updated if a task is added or deleted.
taskStatus	Array (boolean)	Stores whether each of the corresponding tasks in the tasks array are complete or incomplete based on the checkmark.

Journal Table
Purpose: To store the dates and reflections of the user in the digital journal.

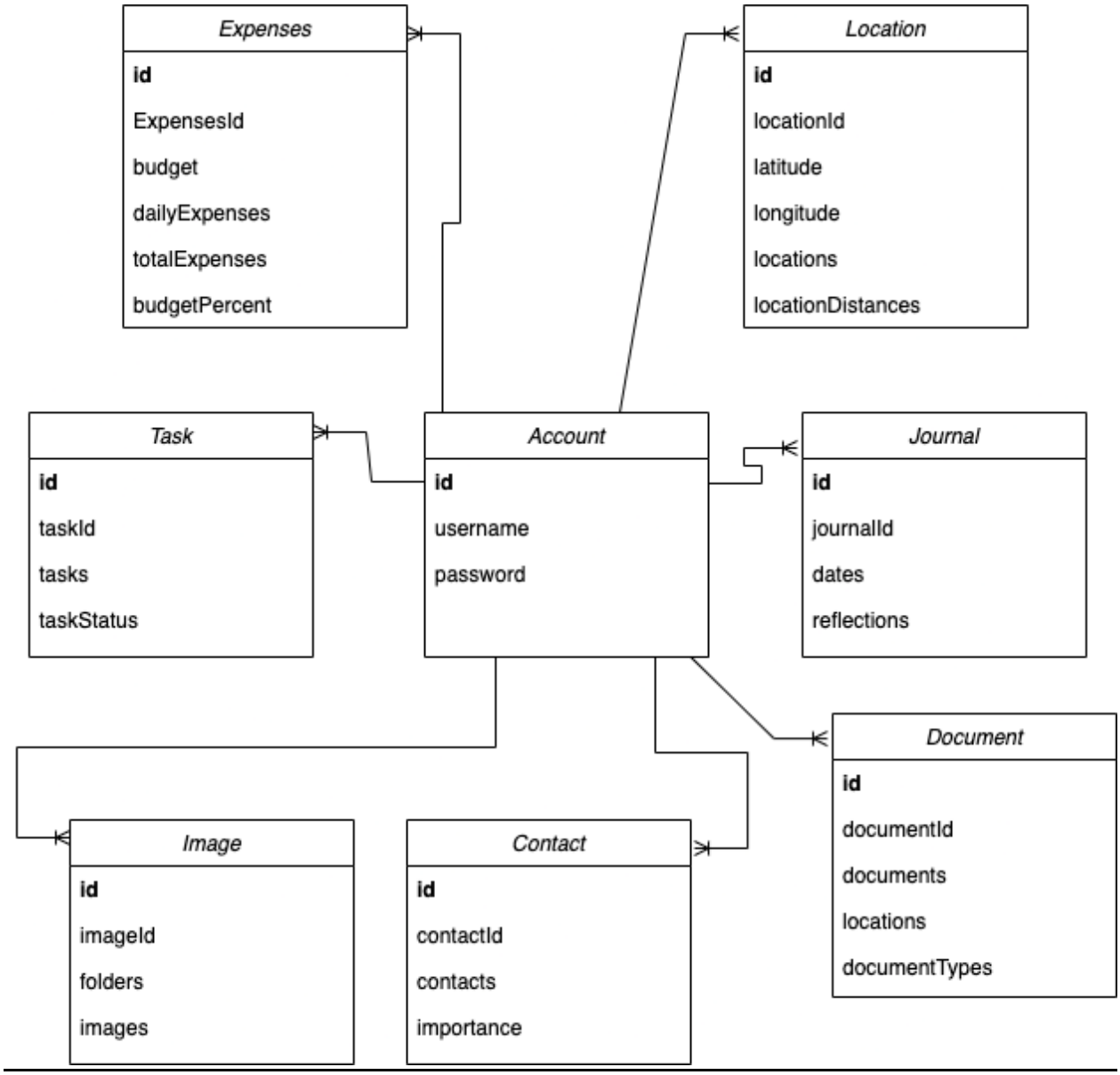
Field Name	Data Type	Description
id	Integer	Stores a unique id generated for every new account created.
dates	Array (String)	Stores the date of each reflection entered by the user at a unique index.
reflections	Array (String)	Stores the corresponding reflections entered by the user for each date in the dates array.

Contact Table

Purpose: To store the contacts and their importance inputted by the user.

Field Name	Data Type	Description
id	Integer	Stores a unique id generated for every new account created.
contacts	Array (integer)	Stores each name of a contact entered by a user at a unique index.
importance	Array (integer)	Stores the corresponding importance to each contact in the contacts array based on input from the user. 0 = low importance; 1 = medium importance; 2 = high importance.

Entity Relationship Diagram



Pseudocode

The pseudocode below creates an outline for the processing of the main functions of the travel management system.

Generate ID

```
generate_id()
     random = new SecureRandom()
     id = random.nextInt(10000)
     return id
```

Calculate total expenses

```
calculate_expenses()
     connection = Expenses.connect()
     if (connection)
          loop for int i = 0 to i = dailyExpenses.length() - 1
               totalExpenses = totalExpenses +
               dailyExpenses[i]
          end loop
          return totalExpenses
     end if
     else
          return null
     end else
```

Calculate the percentage of total expenses out of the travel budget

```
calculate_percentage()
     connection = Expenses.connect()
     if (connection)
          percentage = (totalExpenses/budget)*100
          return percentage;
     end if
     else
          return null
     end else
```

Finds the hotspots and distances of nearby locations

```
locate_hotspots()
     latitude = Settings.location[0]
     longitude = Settings.location[1]
     connection1 = GoogleMapDB.connect()
     connection2 = Location.connect()
     if (connection1 & connection2)
          loop for int i = latitude - 0.045 to i = latitude +
          0.044 with i = i + 0.001
```

```
                loop for int i = longitude – 0.045 to i =
                longitude + 0.044 with i = i + 0.001
                     if (hotspotFound())
                          locations.add(hotspot)
                          locationDistances.add(hotspotDistance
                          )
                     end if
                end loop
           end loop
           hotspots = new double[][]
           hotspots.add(locations, locationDistances)
           return hotspots
      end if
      else
           return null
      end else
```

Sort image/document names

```
import Arrays

sort_images()
     connection = Images.connect()
     if (connection)
          loop for int i = 0 to i = images.length() - 1
               Arrays.sort(images[i])
          end loop
          return images
     end if
     else
          return null
     end else
```

Rename an image/document

```
rename_image(image_original; image_new)
     connection = Images.connect()
     if (connection)
          loop for int i = 0 to i = images.length() - 1
               if (image_original.equals(images[i])
                    images[i] = image_new
               end if
          end loop
          return images
     end if
     else
          return null
     end else
```

Delete an image/task

```
delete_image(image)
     connection = Images.connect()
     if (connection)
          loop for int i = 0 to i = images.length() - 1
               if (image.equals(images[i])
                    image[i].remove(image)
               end if
          end loop
          return images
     end if
     else
          return null
     end else
```

Test Plan

		Success Criteria	Incorrect Data/Procedure	Correct Data/Procedure
1	**a**	Allow the user to create an account with their email and a secure password	Password: abc1234 Click → Weak password	Password: 8bg2aI1^x Click → Login successful
	b	Send the user a verification email upon account creation	Click "Create account" → if email not received → Unsuccessful	Click "Create account" → if email received → Successful
	c	Allow the user to change their password	Enter details and click "Change password" → Log in with new password → If login unsuccessful → Unsuccessful	Enter details and click "Change password" → Log in with new password → If login successful → Successful
2	**a**	Allow the user to login to their account if the username and password entered is correct	Create account with email joseph34578@gmail.com and password "8bg2aI1^x" → Login with account details → If login unsuccessful → Unsuccessful	Create account with email joseph34578@gmail.com and password "8bg2aI1^x" → Login with account details → If login successful → Successful
3	**a**	Allow the user to input their total travel budget, number of travel days and daily expenses	Daily expenses: 430, 340, 282.87 Budget: 1000 Number of days: 5 Click "save" → If message indicates data is not successfully stored → Unsuccessful	Daily expenses: 430, 340, 282.87 Budget: 1000 Number of days: 5 Click "save" → If message indicates data is successfully stored → Successful
	b	Calculate their total travel expenses	Daily expenses: 430, 340, 282.87 Click → If total expenses is not 1052.87 → Unsuccessful	Daily expenses: 430, 340, 282.87 Click → If total expenses is 1052.87 → Successful
	c	Show the user the percentage of their budget they have exhausted	Budget: 1000 Daily Expenses: 450, 50, 401.5 Click → If user does not receive notification → Unsuccessful	Budget: 1000 Daily Expenses: 450, 50, 401.5 Click → If user receives notification → Successful
4	**a**	Allow the user to open and access Maps through the system	Click "open maps" → If maps not opened → Unsuccessful	Click "open maps" → If maps opened → Successful

5	**a**	Allow the user to input their daily itinerary	Click “add task” → Enter “homework”, “mall” → Log out → Log in → Click “Update itinerary” → If tasks “homework”, “mall” are missing → Unsuccessful	Click “add task” → Enter “homework”, “mall” → Log out → Log in → Click “Update itinerary” → If tasks “homework”, “mall” are missing → Unsuccessful
	b	Allow the user to save and update their itinerary	Click “add task” → Enter “homework” → Log out → Log in → Click “Update itinerary” → If task “homework” is missing → Unsuccessful	Click “add task” → Enter “homework” → Log out → Log in → Click “Update itinerary” → If task “homework” is present → Successful
6	**a**	Allow the user to add, delete and edit the journal daily	Click “Update journal” → Enter date and add sample reflection “abc” → Repeat for next day → Log out → Log in → If journal reflections not added → Unsuccessful	Click “Update journal” → Enter date and add sample reflection “abc” → Repeat for next day → Log out → Log in → If journal reflections added → Successful
	b	Allow the user to mark each write-up with a specific date	Click “Update journal” → Enter date and add sample reflection “abc” → Log out → Log in → If reflection not mentioned with specified date → Unsuccessful	Click “Update journal” → Enter date and add sample reflection “abc” → Log out → Log in → If reflection mentioned with specified date → Successful
	c	Allow the user to view the journal	Click “View Journal” → If journal not shown → Unsuccessful	Click “View Journal” → If journal shown → Successful
7	**a**	Allow the user to click an image using the system	Click “Click Image” → If Photo Booth does not open → Unsuccessful	Click “Click Image” → If Photo Booth opens → Successful
8	**a**	Allow the user to add important contact details	Add contact → Log out → Log in → If contact not found → Unsuccessful	Add contact → Log out → Log in → If contact found → Successful
	b	Allow the user to mark certain contacts with high importance, some with moderate importance and others with low importance	Add contact with high importance → Add contact with low importance → Log out → Log in → If contacts not mentioned with corresponding importance → Unsuccessful	Add contact with high importance → Add contact with low importance → Log out → Log in → If contacts mentioned with corresponding importance → Successful