

|  |
| --- |
| University of Greenwich |
| Mobile Application Design and Development |
|  |

|  |
| --- |
| 11-21-2022 |

# Contents

[Contents 1](#_Toc119948278)

[I. Introduction 2](#_Toc119948279)

[II. Checklist 2](#_Toc119948280)

[III. Reflection 3](#_Toc119948281)

[IV. Application Evaluation 3](#_Toc119948282)

[1. Human Computer Interaction 3](#_Toc119948283)

[Consistency 5](#_Toc119948284)

[Shortcuts 6](#_Toc119948285)

[Informative feedback 6](#_Toc119948286)

[Dialouge 7](#_Toc119948287)

[Error handling 8](#_Toc119948288)

[2. Security 8](#_Toc119948289)

[3. Ability of the app to run on a range of screen sizes 9](#_Toc119948290)

[4. Future developments 10](#_Toc119948291)

[V. Implementation 11](#_Toc119948292)

[1. Enter details of trips 11](#_Toc119948293)

[2. Store, view and delete trip details or reset the database 13](#_Toc119948294)

[3. Add expenses to a trip 14](#_Toc119948295)

[4. Search 15](#_Toc119948296)

[5. Upload details to a cloud-based web service 16](#_Toc119948297)

[6. Create a cross-platform prototype of the app 16](#_Toc119948298)

[7. Implement persistence 19](#_Toc119948299)

[8. Additional feature 19](#_Toc119948300)

[VI. Code listing 21](#_Toc119948301)

[Constant 21](#_Toc119948302)

[Main fragment 21](#_Toc119948303)

# List of Figures

[Figure 1. Trip illustrated with circle image 5](#_Toc119948304)

[Figure 2. Search bar 5](#_Toc119948305)

[Figure 3. Example 6](#_Toc119948306)

[Figure 4. System uses search icon 7](#_Toc119948307)

[Figure 5. Using home icon shortcut 7](#_Toc119948308)

[Figure 6. Empty list 8](#_Toc119948309)

[Figure 7. Confirmation dialog when adding new trip 9](#_Toc119948310)

[Figure 8. Error message 9](#_Toc119948311)

[Figure 9. Responsive in design 11](#_Toc119948312)

[Figure 10. Enter details of trips 13](#_Toc119948313)

[Figure 11. Error message 13](#_Toc119948314)

[Figure 12. Confirmation delete trip 14](#_Toc119948315)

[Figure 13. Home screen 14](#_Toc119948316)

[Figure 14. Data stored in SQLite database 15](#_Toc119948317)

[Figure 15. Trip detail 16](#_Toc119948318)

[Figure 16. Expense stored in SQLite database 16](#_Toc119948319)

[Figure 17. Search 17](#_Toc119948320)

[Figure 18. Add trip Ionic 18](#_Toc119948321)

[Figure 19. Error message ionic 18](#_Toc119948322)

[Figure 20. Insert done 19](#_Toc119948323)

[Figure 21. List trip ionic 19](#_Toc119948324)

[Figure 22. Details of trip 20](#_Toc119948325)

[Figure 23. Index DB 20](#_Toc119948326)

[Figure 24. Translator 21](#_Toc119948327)

# List of Tables

[Table 1. Checklist 3](#_Toc119948328)

# I. Introduction

In this project, the author develops an application called MExpense - an application that helps employees to record details of their expenses during trips and then upload them to the server. The author has to develop 2 different applications: native application and hybrid application. Native application uses Java as the backend programming language, while cross-platform apps use the Ionic framework for development. This whole project is developed by the author by his own efforts. This report consists of 5 parts:

* Checklist: A brief table containing a checklist of features the author has implemented.
* Reflection: The author reflects on how the application was developed. The author also provides lessons learned, discusses the strengths and weaknesses of the entire project, and possibilities for improvement.
* Application Evaluation: Author's rating of his app in terms of: Human Computer Interactoin, Security, Ability of the app to run on a range of screen sizes and future developments.
* Implementation: The author screenshots his application and explains the functions that have been developed
* Code Listing: The code that the author has written.

# II. Checklist

Table 1. Checklist

|  |  |
| --- | --- |
| Feature | Implementation |
| a) | Fully implemented |
| b) | Fully implemented |
| c) | Fully implemented |
| d) | Implemented but just allow for searching with name |
| e) | Not implemented |
| f) | I have created a cross-platform prototype of app |
| g) | I have implemented persistence |
| h) | Translator was implemented |

# III. Reflection

The development of the application went on with some difficulty, however, in the end, the application was completed, meeting most of the requirements. Users can enter details of trip (with validation) and add expenses to a trip. Users can search for trips by name and the system can store, view and delete trip details or reset the database. Besides, the author has taken advantage of ML Kit - Google's library to use Machine learning models. The translator has been used in the article, to help users convert English into German, Arabic and Korean.

The author used fragments to build the layouts for the application, managing them through the navigation graph. This is a little different from what the author was taught by his instructor, so the development was not smooth and was a bit difficult. The author also references some user interfaces on the net to apply to his design. However, because the application development time takes up too much time, the author does not have enough time to take care of the interface.

Regarding security, the author finds that his application is not really secure. The author has not researched thoroughly about data security methods in Android programming.

Through this project, the author learned how to develop android application using android studio and java. He also learned how to use fragment, view binding to replace normal findViewById usage. He reached out to popular machine learning libraries used in android programming. Besides, the author has also applied some design rules learned in Human Computer Interaction course (HCI).

# IV. Application Evaluation

## 1. Human Computer Interaction

The author has applied some of the knowledge he has learned in HCI to design UI/UX for the application. For example, the author has used attention and memory in cognitive psychology. When talking about attention, the author used circle images to illustrate the trips. Besides, the author also uses the names of the trips to annotate them. This makes the user more focused on the list of trips.

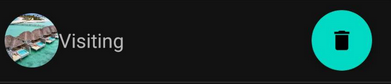


Figure 1. Trip illustrated with circle image

The title of the page has a larger font size, bolder, making users not confused between different pages. Moreover, the author has set the text color to contrast with the background color. Intuitively, it makes the text noteworthy. The author annotated each data field with a label above to let the user know which information they are going to input. All the labels have a bigger size, bold. If one object is physically larger, brighter, bolder or more visually intense than surrounding ones, then it is more likely to attract attention. Furthermore, when adding/updating a trip or an expense, a white-board with text fields are placed in the center of the page, and it is contrast with the background color. This helps users easily focus on the place they fill information.

In order for users to know where to look for trips, the author has used word suggestions to make it easier for users to use, thereby increasing the user experience.



Figure 2. Search bar

When it comes to the memory theory, the author used icons that is easy to remember things when performing tasks. He used icon **trash** to illustrate delete function.

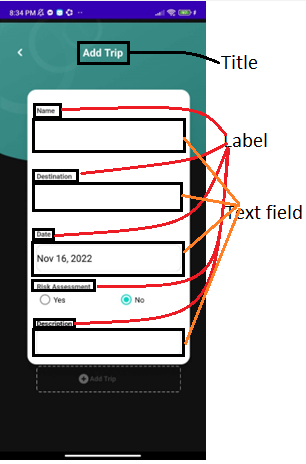


Figure 3. Example

Besides, the author also evaluate his app follows Shneiderman’s Eight Golden Rules (Wong, 2020). These are:

### Consistency

The system uses familiar icons, colors, menu hierarchy, calls to action, and user flow when designing similar scenarios and action sequences. For example, the application uses the same search icon as in many other systems like Google, Microsoft, etc. This represents design consistency. Besides, it helps users not to confuse the functions of the button when using the application.





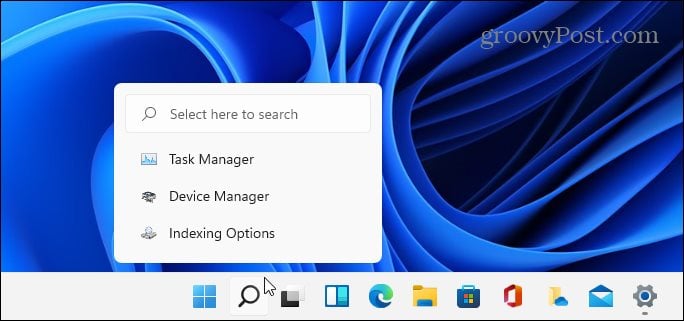


Figure 4. System uses search icon

### Shortcuts

The system the author designed provides options for users to complete their work faster. For example, when the user is on the translator screen, if the user wants to return to the main screen, they just need to click the Home icon in the upper right corner of the screen instead of having to press the back button repeatedly. This helps users to navigate and use faster and easier.

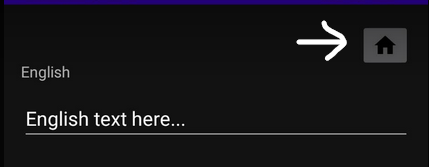


Figure 5. Using home icon shortcut

### Informative feedback

The system indicates that the system's status is occurring. A good example of this is when the list of trips or expenses is empty, the system will notify the user of the lists. the book is empty, instead of not giving a misleading message to the user. This is to explain to the user that this could be a system error, or that the trip/expenses list is empty.

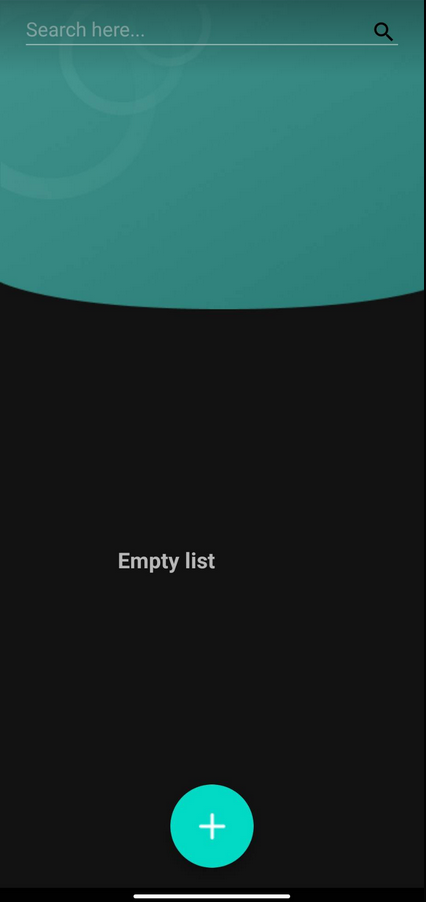


Figure 6. Empty list

### Dialouge

The system offers dialogue windows to help users confirm the information when adding a new trip. This helps them avoid adding trips with incorrect information.

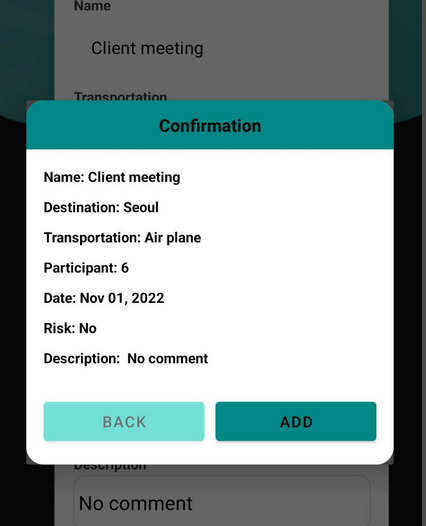


Figure 7. Confirmation dialog when adding new trip

### Error handling

When users encounter errors, such as not filling in the fields completely, the user will be notified that they must complete the fields. This is because there are many fields of information in the application that are indispensable, and it is not acceptable to ignore it. Therefore, the use of a gentle message to display the error is necessary to warn the user.



Figure 8. Error message

Besides, there are 3 more rules: Permit reversal of actions, Support internal locus of control, and Reduce short-term memory load.

## 2. Security

As for security, the application that the author developed has almost no security method. Therefore, this application is very vulnerable, and it causes damage to the user's personal information. In the long run, users will no longer use products without protection. Enhancing the app's security helps the author maintain user trust and device integrity.

First, the author can build a login functionality for the user. This will prevent strangers from being able to see the contents inside. Besides, encrypting passwords with Bcrypt also prevents hackers.

As the payment for various types of expenses is more developed, the author's application can be linked with banks for payment. However, when connecting between the author's application and the banking application, important data will be exchanged. To ensure this permission is not used out of control, the author needs to make sure that the connected application is trusted. By declaring a separate signature for that permission and when using it, check if they match or not.

Missing cases in input validation is one of the most common security issues affecting applications. Therefore, input data validation is essential. Choosing type-safe languages tends to reduce the likelihood of input validation problems.

In the scope of the application, the author implements only one security measure, which is input validation.

## 3. Ability of the app to run on a range of screen sizes

The author has deployed the application on 2 devices: Xiaomi 2201117SG and Android Emulator Pixel 4 API 30. Xiaomi 2201117SG has a resolution of 1080x2400. Meanwhile, the Android Emulator Pixel 4 API 30 has a size of 1080x2280. The author's application is capable of running on the two devices above. Furthermore, to support the deployment of the application on a variety of devices with different resolutions, the author designed the layout of the application to be responsive and adaptive. For example, when designing the interface, the author used Constraint layout. This will help the elements in the application be automatically scaled to the screen size based on the constraints that have been installed.

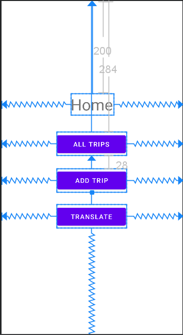


Figure 9. Responsive in design

## 4. Future developments

Although the functions of the application have been basically completed as required (except for the e function), the application has a lot of potential for development. When it ccomes to **function e**, the author's current level of programming and knowledge cannot tackle the **function e**. Therefore, the author needs to focus on developing and training his ability to solve difficult functions (eg **function e**).

In terms of UI/UX, the application does not have a good interface. The author has not taken advantage of the empty spaces of the UI. Besides, he has not arranged the elements properly, making his application unattractive to users, thereby making the user experience bad. Therefore, in the future, the author needs to improve his ability to design interfaces, as well as his ability to research and refer to interfaces to make his application eye-catching, attractive to users, thereby attracting more users.z`

Regarding security, the author needs to implement more security measures for the application to be deployable in practice. For example, using Internal Storage for Sensitive Data helps prevent sensitive information from being exposed. Besides, encrypting data on external storage is also one of the necessary measures when the internal memory capacity of Android devices is limited. Since data on external storage media can be accessed directly by both the user and other applications on the device, it is important that you store that data in an encrypted format. One of the most popular encryption algorithms used by developers today is AES, which stands for Advanced Encryption Standard, with a key size of 256 bits.

Last but not least, the application has many potential functions that can be deployed and developed. For example, to improve the user experience, the author can develop a function to forecast the weather at the destination of the user's trip. This will help them prepare the necessary items when they arrive at the new location. Another function that is equally important is the function of integrating payment of all kinds of expenses with e-wallets and banking applications. This will help users save time when paying expenses.

# V. Implementation

## 1. Enter details of trips

This is the page to enter details about a trip. The information includes **Name, Destination, Date, Risk Assessment and Description**. The trip name and destination are dropdown selection, date is a date picker dialog, risk assessment has 2 radio buttons and description is the text field.

Information fields are checked to see if they are left blank. Once the above information has been filled in and checked, click the "**Add Trip**" button below a pop-up window with detailed information about the trip will appear. Users will check this information and they will be allowed to go back and change any information they want. However, when the information is not filled in completely, there will be messages to the user reminding them which fields are allowed to be omitted and which fields are required.

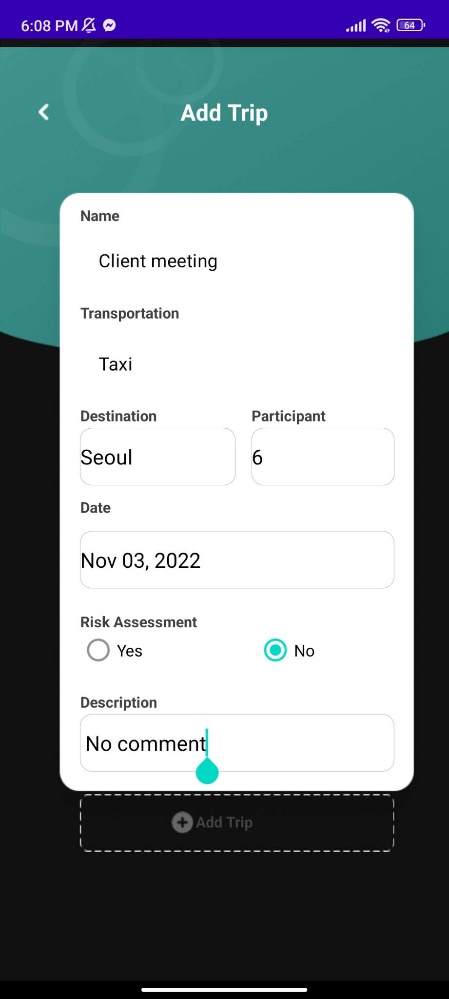
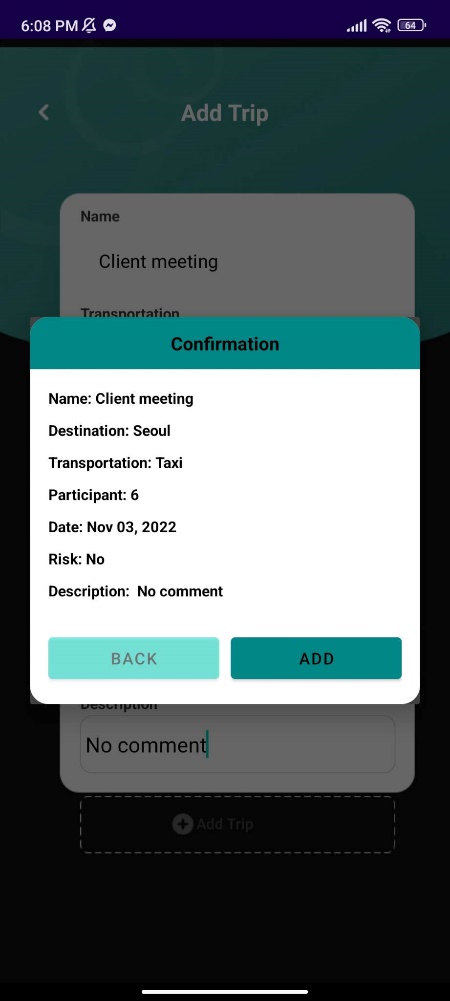
 

Figure 10. Enter details of trips

The information that is required: **Name, Destination, Date, Transportation, Participant, and Risk Assessment**. Whereas, **Desciption** is optional. For example, when users do not fill in the **Name** field, there will be a error message that be thrown like this:



Figure 11. Error message

In the application, the author set the **Name** and **Transportation** are lists, and it is predefined by the author. When finish the process, the system navigates the screen to Home screen, which is the place that all the trips are listed.

## 2. Store, view and delete trip details or reset the database

This is where users can keep track of their entire trip booking history. The author has stored them in SQLite database and displayed them using RecycleView. Right next to each trip, the author puts a delete button so the user can delete it. When performing deletion, the system will display a popup dialog to ask if the user really wants to delete this trip. If so, the system will delete the trip from the database.

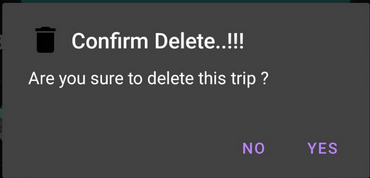


Figure 12. Confirmation delete trip

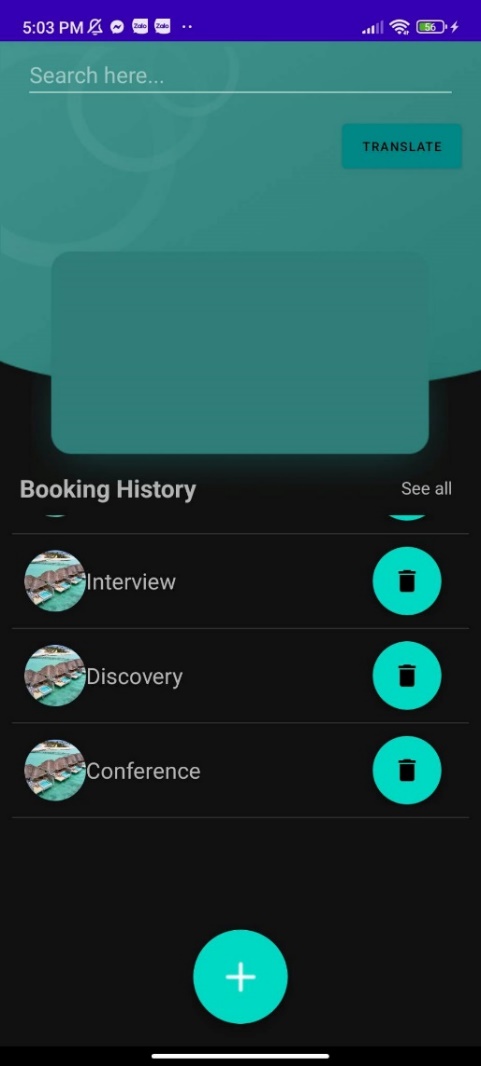


Figure 13. Home screen

When users want to see the details of a trip, they just need to touch the trip they want. Then the "**Add trip**" screen once again appeared, but this time with changes. Label is changed from "**Add trip**" to "**Trip detail**"; pre-filled fields. If the user wants to edit any information, just retype the information and then click the "**Save**" button right below. The data will be edited and updated automatically in the SQLite database. If the user doesn't want to edit, they can click the back arrow in the upper left corner of the screen to go back to the old page.

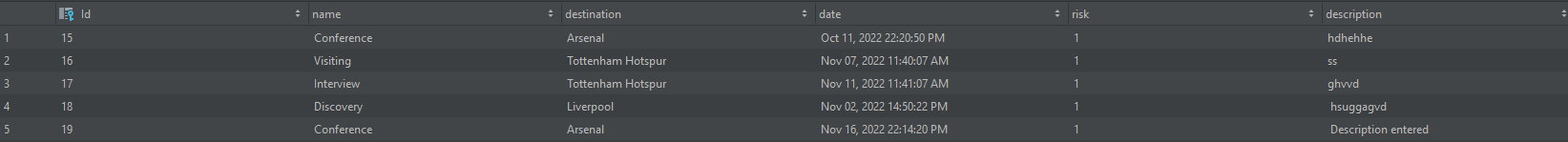


Figure 14. Data stored in SQLite database

## 3. Add expenses to a trip

Once the trip creation is completed, the user can add expenses to the trip. Here, the user will enter information about the type of the expense, the amount, the time of the expense and the comment. In the fields above, **type, amount,** and **time** are required, while **comments** are optional. These expenses of trip are also stored in the SQLite database. Once added, the system will navigate back to the screen showing the list of expenses for the trip. Note: users can add multiple expenses for one trip. For example, a trip may include 2-3 expenses. The time of expenses belong to the date of the trip. For example, if the date of the trip is 16th November, then, all the time of expense would belong to that date.

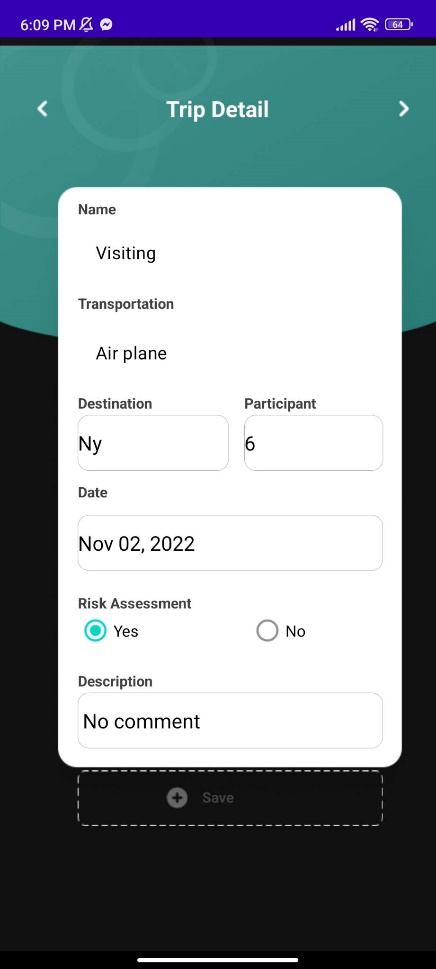


Figure 15. Trip detail

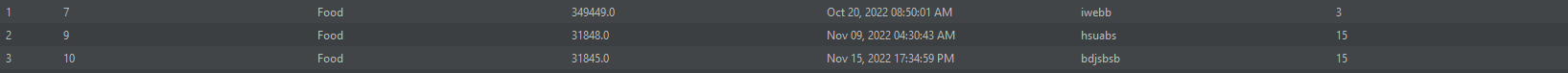


Figure 16. Expense stored in SQLite database

## 4. Search

The users are allowed to search for a trip based on its name in the database. Users just need to enter the name in the search bar, then the trips with the corresponding name will appear. Users only need to enter the first few letters and the corresponding named trips will appear.

In the example illustrated below, the author entered the keyword "in" into the search bar, then the system automatically gave trips whose name contains the keyword that the author entered before. there. In this case, the returned results are Visit**in**g and **In**terview respectively.

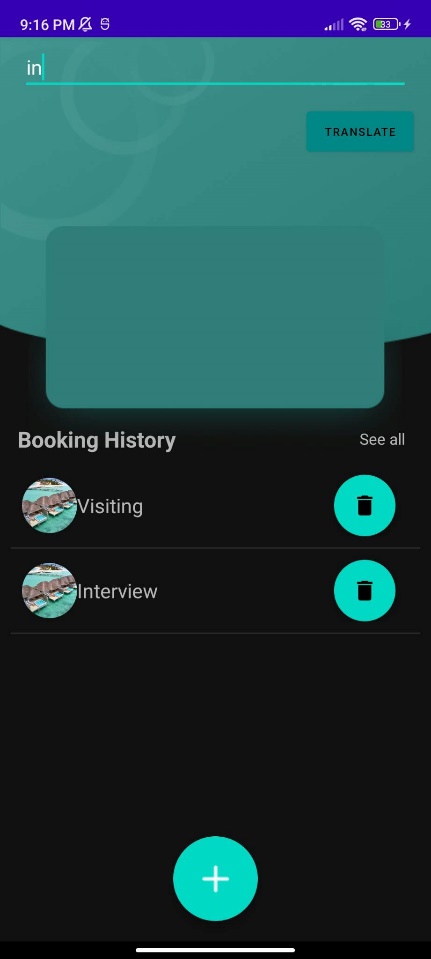


Figure 17. Search

## 5. Upload details to a cloud-based web service

Unfortunately, the author does not implement this functionality in his application.

## 6. Create a cross-platform prototype of the app

The author has successfully deployed cross-platform applications. Here users can perform functions such as adding a new trip, updating them and deleting it. This app is deployed using ionic.

The function of adding a trip is similar to section **IV.1**. Here, users need to enter information fields including **Name**, **Destination**, **Date**, **Risk Assessment** and **Description**. In which, **Name**, **Destination, Date** and **Risk** **Assessment** are required fields. **Description** is an optional information field. The trip name and destination are dropdown selection, date is a date picker dialog, risk assessment has 2 radio buttons and description is the text field.

If the user does not fill in the required information fields, the system will appear a window informing that the trip cannot be added because the above fields have not been filled in. If they add successfully, a window pops up informing the user that the trip has been added to the database.

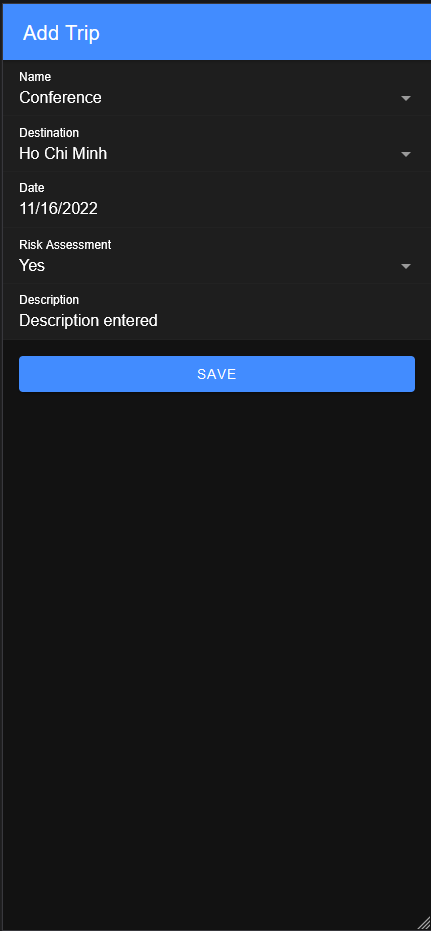


Figure 18. Add trip Ionic

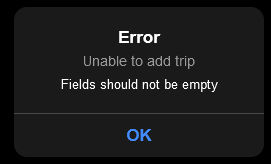


Figure 19. Error message ionic



Figure 20. Insert done

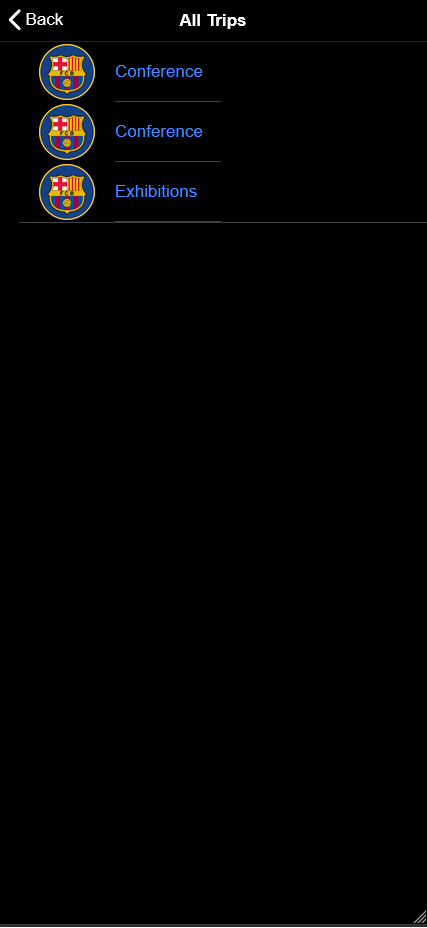


Figure 21. List trip ionic

The system will navigate the application to the screen showing the list of trips. The author captioned the trip by name. Users click on each trip to see detailed information about the trip. Besides, users can update or delete trips.

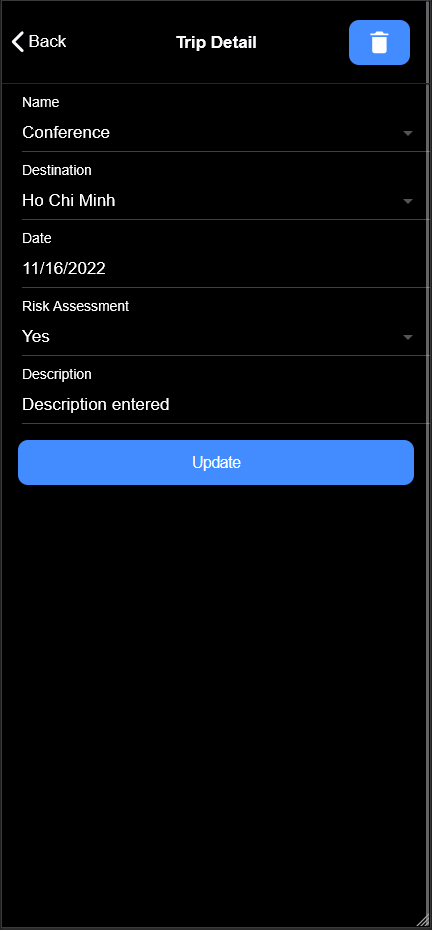


Figure 22. Details of trip

If the user wants to delete a trip, they need to click on the button with the trash can icon at the top right of the screen. When they click on it, the trip will be deleted and the system will navigate to the page containing the list of trips.

## 7. Implement persistence

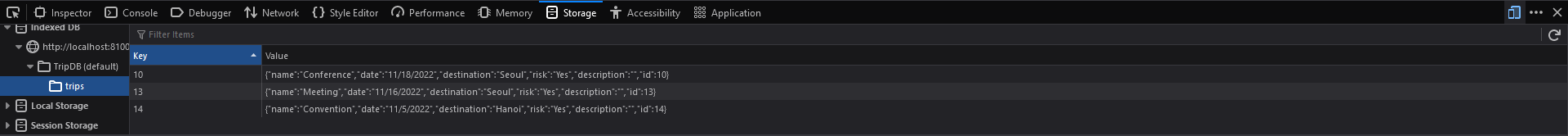


Figure 23. Index DB

All the data are stored in the index DB – which means the data are stored on web browser.

## 8. Additional feature

The author has built extra translation functionality for his application. This feature is really useful for people when they go abroad for business, or travel. Hiring your own interpreter is very expensive. So having a "language assistant" on the user’s side is really a trusted companion of the user. All users need to do is type what they want to translate, and then press the button to translate into different languages. Within the scope of this project, the author has given the user to translate English into 3 different languages: German, Arabic, and Korean. These are 3 languages that are widely used around the world. After clicking the buttons, the translated sentences will be displayed below. An example is illustrated by the figure below.

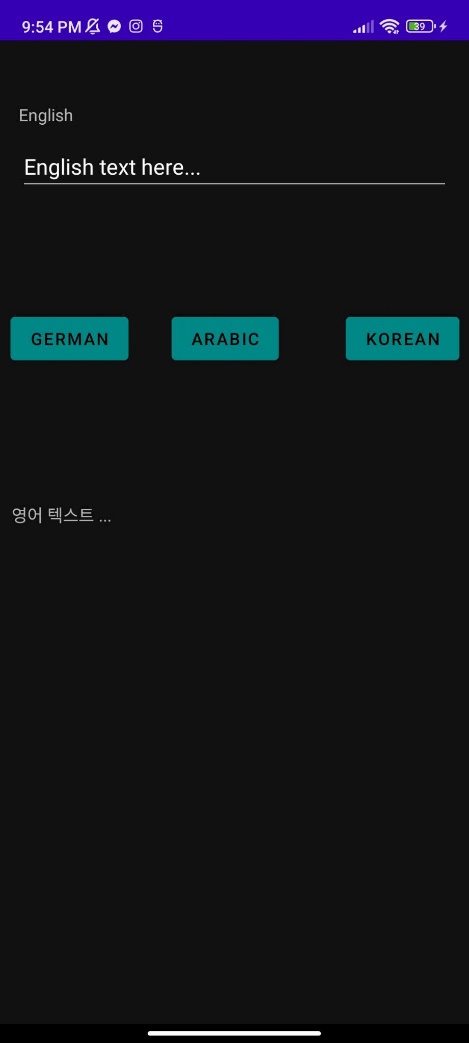
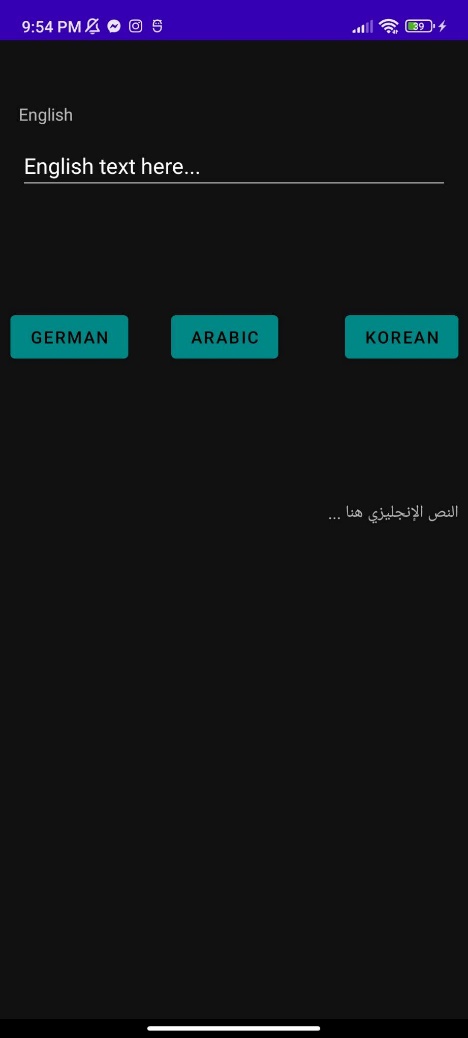
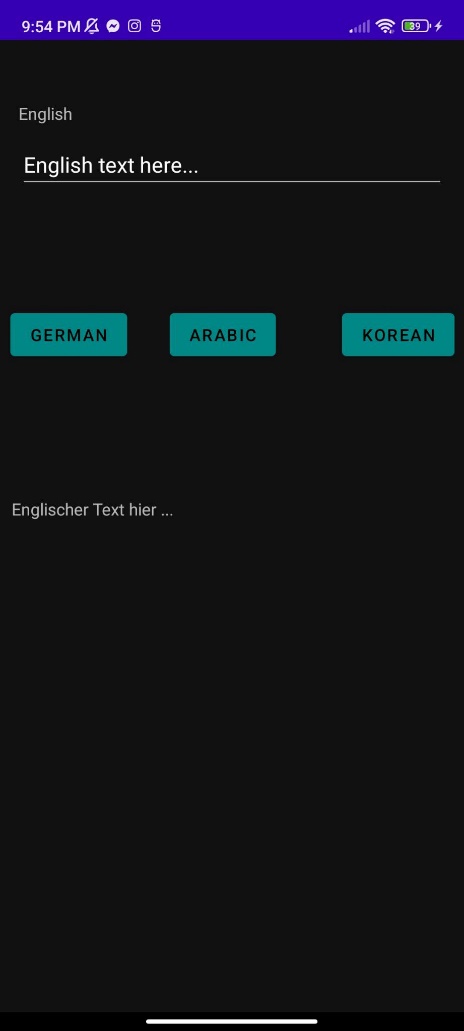
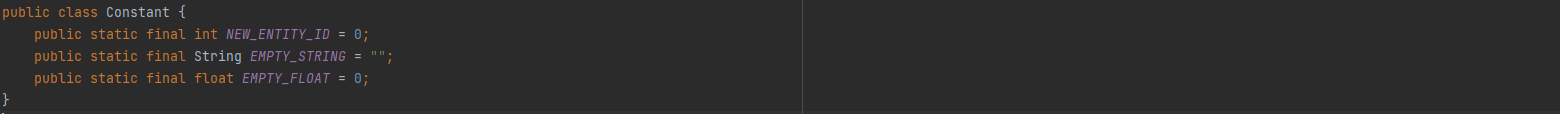
  

Figure 24. Translator

# VI. Code listing

## Constant



## Main fragment

