**M-Expense: Expense Management App**

Often employees are able to claim expenses for travel that they have to undertake in order to do their jobs. For instance, an employee who usually works in London may have to travel to Plymouth for a meeting. They could claim for the transport costs (e.g., underground, train and bus) and also for lunch and refreshments or accommodation.

You are to create a mobile app for use by employees that they can use to record details of their expenses and then upload them to their employer’s server. The app is to be called **M-Expense** and will allow employees to record trip and expense details and do searches. The features the app should support are given below:

* Features **a)** to **e)** are to be implemented as a **native Android app coded in Java**.
* Features **f)** and **g)** are to be implemented as a **hybrid app coded using cross platform** frameworks like Xamarin, React Native, Flutter etc.
* Feature **g)** can be implemented as **either or both** additionsto thenative android app or cross platform app.

Your final app is the culmination of all your hard work on this course, which should become a strong addition to your programming portfolio. You should produce an app that is well-designed, robust and useful. The GUI design should be clean, simple to navigate, and operate smoothly without sluggishness or crashes. The app should not require instructions or a manual to use.

**Part A – Implementation (80%)**

**Description of the application**

1. **Enter details of trips (10%):**

Note that users must be able to enter all of the following fields. ***Required*** fields mean that the user must enter something in this field; otherwise they will be prompted with an error message.  ***Optional*** fields mean that the user can enter something if they wish, but they will not get an error message if nothing is entered.

The user should be able to enter:

* Name of trip (e.g. “conference”, “client meeting”, etc.) – *Required field*
* Destination - *Required field*
* Date of the trip - *Required field*
* Requires risk assessment (i.e. “Yes” or “No”) - *Required field*
* Description – *Optional field*
* Two or more other fields of your own invention – be creative!

The app will check the input and if the user doesn’t enter anything in one of the required fields, the app should display an error message.

Once the details have been accepted by the app (e.g. no required fields were missing), it should display the details back to the user for confirmation and allow them to go back and change any details that they wish.

1. **Store, view and delete trip details or reset the database (10%)**

All the details entered by the user should initially be stored on the device in an SQLite database.

The user should be able to **list all** the **details of all trips** that have been entered into the app, edit or delete individual trips, and delete all the details from the database.

1. **Add expenses to a trip (10%)**

Employees will be able to enter details about expenses.  They should be able to select a trip and then add the following details:

* Type of expense (e.g. “travel”, “food”, “other” - *Required field*
* Amount of the expense - *Required field*
* Time of the expense - *Required field* *(ideally this should default to the current date)*
* Additional comments - *Optional field*

The user must be able to enter multiple expenses for a single trip.

The app should store all details entered on the device in an SQLite database.

It should be possible for a user to select a trip and display all expenses.

1. **Search (10%)**

The user should be able to search for a trip in the database by name. At its simplest, this could mean entering the name and displaying the first trip that matches.  Ideally the user should be able to enter the first few letters of the name and display all matching trips.

Advanced search options will allow searching for all trips with the following criteria: name, destination and/or date. It should be possible for a user to select an item from the resulting search list and to display its full details.

1. **Upload details to a cloud-based web service (10%)**

The user should be able to upload all the details of expenses stored on the mobile device to a cloud-based web service. It should be possible for all expenses to be uploaded at once. YOU DO NOT NEED TO WRITE the web application as the URL of the web service will be made available to you by your instructors.

The web service will accept the data and reply with a message which your app should display. Ideally your app should format the message so that is easy for the user to read.

**The format in which the data must be uploaded**

The data you upload to the web service should be sent in an HTTP POST message.  The HTTP POST message will contain one field called **jsonpayload**. The value of **jsonpayload** is a string in JSON format.

You have some flexibility over how you design the JSON however the following are essential:

* The JSON must be valid. See <http://www.w3schools.com/js/js_json_syntax.asp> for details on JSON syntax.
* You must include a field with the name "userId" which has a value of your university user id e.g.  "userId":"wm123"
* The trip details must be represented as an array with the name "detailList"
* Each element in the array "detailList" represents an expense about which details are being uploaded.
* Each expense in "detailList" must at a minimum have a field called "name" whose value is the trip name or the brief description of the trip.
* Below is an incomplete example of what the JSON might look like

{"userId":"wm123",

"detailList":[

{"name":"Android Conference",   *other fields*},

{"name":"Client Meeting",   *other fields*}]}

**The format of the response**

The web service will respond with an HTTP response containing JSON which represents a reply to your upload.  The JSON reply will contain the following fields:

|  |  |
| --- | --- |
| **field name** | **value** |
| **uploadResponseCode** | A code indicating whether or not your upload was successful.  Values are either SUCCESS or FAILURE.  If the code is FAILURE the message field will contain more information about what went wrong. |
| **userid** | The same userId that was included in the upload message |
| **number** | The number of records uploaded |
| **names** | The names of the records uploaded |
| **message** | A message giving more information about what happened.  Particularly important if the uploadResponseCode is not SUCCESS |

An example response is given below.  Note that the order of the fields may vary

{"uploadResponseCode":"SUCCESS",

"userid":"wm123",

"number":2,

"names":"Android Conference, Client Meeting",

"message":"successful upload – all done!"}

Your app should display the data sent in the response.  You could just display the raw JSON message, but it would be better to format the output to make it more readable for the user.

1. **Create a cross-platform prototype of the app (10%)**

Implement as much as you can of features a) using cross platform frameworks.

1. **Implement persistence using cross platform (10%)**

Implement as much as you can of features b) using cross platform frameworks.

1. **Add additional features to either or both the Android or cross platform version of the app (10%)**

**Features a) to g) are the core requirements for the app.** If you have implemented these and want to add some additional features, then you may. Any enhancements should be implemented **in addition to** **NOT instead of** the core requirements.  The idea is that these features **stretch** your skills, so be prepared to do your own research and feel free to show off!  You can think of your own enhancements.  Here are some possible examples:

* Allow photos taken by the camera to be added to the data stored
* Pick up the location automatically from the user's location
* Show the location of where an expense incurred on a map
* Make use of an external web service other than the one provided for the coursework.
* Anything you can think of – again be creative!

M-Expense: Ứng dụng quản lý chi phí

Thông thường, nhân viên có thể yêu cầu chi phí đi lại mà họ phải trả để thực hiện công việc của mình. Chẳng hạn, một nhân viên thường làm việc ở London có thể phải đến Plymouth để họp. Họ có thể yêu cầu thanh toán chi phí vận chuyển (ví dụ: tàu điện ngầm, xe lửa và xe buýt) cũng như chi phí ăn trưa và giải khát hoặc chỗ ở.

Bạn phải tạo một ứng dụng dành cho thiết bị di động để nhân viên sử dụng mà họ có thể sử dụng để ghi lại chi tiết về chi phí của họ và sau đó tải chúng lên máy chủ của chủ lao động. Ứng dụng này được gọi là M-Expense và sẽ cho phép nhân viên ghi lại chi tiết chuyến đi và chi phí cũng như thực hiện tìm kiếm. Các tính năng mà ứng dụng nên hỗ trợ được đưa ra dưới đây:

Các tính năng từ a) đến e) sẽ được triển khai dưới dạng ứng dụng Android gốc được mã hóa bằng Java.

Các tính năng f) và g) sẽ được triển khai dưới dạng ứng dụng kết hợp được mã hóa bằng cách sử dụng các khung nền tảng chéo như Xamarin, React Native, Flutter, v.v.

Tính năng g) có thể được triển khai dưới dạng một hoặc cả hai phần bổ sung cho ứng dụng Android gốc hoặc ứng dụng đa nền tảng.

Ứng dụng cuối cùng của bạn là đỉnh cao của tất cả nỗ lực của bạn trong khóa học này, ứng dụng này sẽ trở thành một phần bổ sung mạnh mẽ cho danh mục lập trình của bạn. Bạn nên tạo ra một ứng dụng được thiết kế tốt, mạnh mẽ và hữu ích. Thiết kế GUI phải rõ ràng, dễ điều hướng và hoạt động trơn tru mà không bị chậm hoặc treo. Ứng dụng không nên yêu cầu hướng dẫn hoặc hướng dẫn sử dụng.

Phần A – Thực hiện (80%)

Mô tả của ứng dụng

Nhập chi tiết chuyến đi (10%):

Lưu ý rằng người dùng phải nhập được tất cả các trường sau. Các trường bắt buộc có nghĩa là người dùng phải nhập nội dung nào đó vào trường này; nếu không họ sẽ được nhắc với một thông báo lỗi. Các trường tùy chọn có nghĩa là người dùng có thể nhập nội dung nào đó nếu họ muốn, nhưng họ sẽ không nhận được thông báo lỗi nếu không nhập gì.

Người dùng sẽ có thể nhập:

Tên chuyến đi (ví dụ: “hội nghị”, “cuộc họp khách hàng”, v.v.) – Trường bắt buộc

Điểm đến - Trường bắt buộc

Ngày của chuyến đi - Trường bắt buộc

Yêu cầu đánh giá rủi ro (nghĩa là “Có” hoặc “Không”) - Trường bắt buộc

Mô tả – Trường tùy chọn

Hai hoặc nhiều lĩnh vực khác do chính bạn phát minh – hãy sáng tạo!

Ứng dụng sẽ kiểm tra đầu vào và nếu người dùng không nhập bất kỳ thứ gì vào một trong các trường bắt buộc, ứng dụng sẽ hiển thị thông báo lỗi.

Khi các chi tiết đã được ứng dụng chấp nhận (ví dụ: không có trường bắt buộc nào bị thiếu), ứng dụng sẽ hiển thị lại các chi tiết đó cho người dùng để xác nhận và cho phép họ quay lại và thay đổi bất kỳ chi tiết nào họ muốn.

Lưu trữ, xem và xóa chi tiết chuyến đi hoặc đặt lại cơ sở dữ liệu (10%)

Tất cả các chi tiết do người dùng nhập ban đầu phải được lưu trữ trên thiết bị trong cơ sở dữ liệu SQLite.

Người dùng sẽ có thể liệt kê tất cả các chi tiết của tất cả các chuyến đi đã được nhập vào ứng dụng, chỉnh sửa hoặc xóa các chuyến đi riêng lẻ và xóa tất cả các chi tiết khỏi cơ sở dữ liệu.

Thêm chi phí cho chuyến đi (10%)

Nhân viên sẽ có thể nhập chi tiết về chi phí. Họ sẽ có thể chọn một chuyến đi và sau đó thêm các chi tiết sau:

Loại chi phí (ví dụ: “đi lại”, “ăn uống”, “khác” - Trường bắt buộc

Số tiền chi phí - Trường bắt buộc

Thời điểm chi phí - Trường bắt buộc (lý tưởng là trường này nên mặc định là ngày hiện tại)

Nhận xét bổ sung - Trường tùy chọn

Người dùng phải có thể nhập nhiều chi phí cho một chuyến đi.

Ứng dụng sẽ lưu trữ tất cả các chi tiết được nhập trên thiết bị trong cơ sở dữ liệu SQLite.

Người dùng có thể chọn một chuyến đi và hiển thị tất cả các chi phí.

Tìm kiếm (10%)

Người dùng có thể tìm kiếm chuyến đi trong cơ sở dữ liệu theo tên. Đơn giản nhất, điều này có thể có nghĩa là nhập tên và hiển thị chuyến đi đầu tiên phù hợp. Lý tưởng nhất là người dùng có thể nhập một vài chữ cái đầu tiên của tên và hiển thị tất cả các chuyến đi phù hợp.

Tùy chọn tìm kiếm nâng cao sẽ cho phép tìm kiếm tất cả các chuyến đi với các tiêu chí sau: tên, điểm đến và/hoặc ngày. Người dùng có thể chọn một mục từ danh sách tìm kiếm kết quả và hiển thị đầy đủ chi tiết của mục đó. Phần A – Thực hiện (80%)

Mô tả của ứng dụng

Nhập chi tiết chuyến đi (10%):

Lưu ý rằng người dùng phải nhập được tất cả các trường sau. Các trường bắt buộc có nghĩa là người dùng phải nhập nội dung nào đó vào trường này; nếu không họ sẽ được nhắc với một thông báo lỗi. Các trường tùy chọn có nghĩa là người dùng có thể nhập nội dung nào đó nếu họ muốn, nhưng họ sẽ không nhận được thông báo lỗi nếu không nhập gì.

Người dùng sẽ có thể nhập:

Tên chuyến đi (ví dụ: “hội nghị”, “cuộc họp khách hàng”, v.v.) – Trường bắt buộc

Điểm đến - Trường bắt buộc

Ngày của chuyến đi - Trường bắt buộc

Yêu cầu đánh giá rủi ro (nghĩa là “Có” hoặc “Không”) - Trường bắt buộc

Mô tả – Trường tùy chọn

Hai hoặc nhiều lĩnh vực khác do chính bạn phát minh – hãy sáng tạo!

Ứng dụng sẽ kiểm tra đầu vào và nếu người dùng không nhập bất kỳ thứ gì vào một trong các trường bắt buộc, ứng dụng sẽ hiển thị thông báo lỗi.

Khi các chi tiết đã được ứng dụng chấp nhận (ví dụ: không có trường bắt buộc nào bị thiếu), ứng dụng sẽ hiển thị lại các chi tiết đó cho người dùng để xác nhận và cho phép họ quay lại và thay đổi bất kỳ chi tiết nào họ muốn.

Lưu trữ, xem và xóa chi tiết chuyến đi hoặc đặt lại cơ sở dữ liệu (10%)

Tất cả các chi tiết do người dùng nhập ban đầu phải được lưu trữ trên thiết bị trong cơ sở dữ liệu SQLite.

Người dùng sẽ có thể liệt kê tất cả các chi tiết của tất cả các chuyến đi đã được nhập vào ứng dụng, chỉnh sửa hoặc xóa các chuyến đi riêng lẻ và xóa tất cả các chi tiết khỏi cơ sở dữ liệu.

Thêm chi phí cho chuyến đi (10%)

Nhân viên sẽ có thể nhập chi tiết về chi phí. Họ sẽ có thể chọn một chuyến đi và sau đó thêm các chi tiết sau:

Loại chi phí (ví dụ: “đi lại”, “ăn uống”, “khác” - Trường bắt buộc

Số tiền chi phí - Trường bắt buộc

Thời điểm chi phí - Trường bắt buộc (lý tưởng là trường này nên mặc định là ngày hiện tại)

Nhận xét bổ sung - Trường tùy chọn

Người dùng phải có thể nhập nhiều chi phí cho một chuyến đi.

Ứng dụng sẽ lưu trữ tất cả các chi tiết được nhập trên thiết bị trong cơ sở dữ liệu SQLite.

Người dùng có thể chọn một chuyến đi và hiển thị tất cả các chi phí.

Tìm kiếm (10%)

Người dùng có thể tìm kiếm chuyến đi trong cơ sở dữ liệu theo tên. Đơn giản nhất, điều này có thể có nghĩa là nhập tên và hiển thị chuyến đi đầu tiên phù hợp. Lý tưởng nhất là người dùng có thể nhập một vài chữ cái đầu tiên của tên và hiển thị tất cả các chuyến đi phù hợp.

Tùy chọn tìm kiếm nâng cao sẽ cho phép tìm kiếm tất cả các chuyến đi với các tiêu chí sau: tên, điểm đến và/hoặc ngày. Người dùng có thể chọn một mục từ danh sách tìm kiếm kết quả và hiển thị đầy đủ chi tiết của mục đó.

Tải chi tiết lên dịch vụ web dựa trên đám mây (10%)

Người dùng sẽ có thể tải tất cả các chi tiết về chi phí được lưu trữ trên thiết bị di động lên dịch vụ web dựa trên đám mây. Tất cả các chi phí có thể được tải lên cùng một lúc. BẠN KHÔNG CẦN VIẾT ứng dụng web vì URL của dịch vụ web sẽ được cung cấp cho bạn bởi những người hướng dẫn của bạn.

Dịch vụ web sẽ chấp nhận dữ liệu và trả lời bằng một thông báo mà ứng dụng của bạn sẽ hiển thị. Lý tưởng nhất là ứng dụng của bạn nên định dạng thông báo để người dùng dễ đọc.

Định dạng mà dữ liệu phải được tải lên

Dữ liệu bạn tải lên dịch vụ web phải được gửi trong thông báo HTTP POST. Thông báo HTTP POST sẽ chứa một trường có tên là jsonpayload. Giá trị của jsonpayload là một chuỗi ở định dạng JSON.

Bạn có thể linh hoạt trong cách thiết kế JSON, tuy nhiên, những điều sau đây là cần thiết:

JSON phải hợp lệ. Xem http://www.w3schools.com/js/js\_json\_syntax.asp để biết chi tiết về cú pháp JSON.

Bạn phải bao gồm một trường có tên "userId" có giá trị id người dùng trường đại học của bạn, ví dụ: "userId":"wm123"

Chi tiết chuyến đi phải được biểu diễn dưới dạng một mảng có tên "detailList"

Mỗi phần tử trong mảng "detailList" biểu thị chi phí về những chi tiết nào đang được tải lên.

Mỗi chi phí trong "detailList" tối thiểu phải có một trường gọi là "tên" có giá trị là tên chuyến đi hoặc mô tả ngắn gọn về chuyến đi.

Dưới đây là một ví dụ không đầy đủ về giao diện của JSON

{"userId":"wm123",

"danh sách chi tiết":[

{"name":"Hội nghị Android", các trường khác},

{"name":"Họp khách hàng", các trường khác}]}

Định dạng của phản hồi

Dịch vụ web sẽ phản hồi bằng phản hồi HTTP chứa JSON thể hiện phản hồi cho nội dung tải lên của bạn. Câu trả lời JSON sẽ chứa các trường sau:

tên trường

giá trị

tải lênMã phản hồi

Mã cho biết quá trình tải lên của bạn có thành công hay không. Các giá trị là THÀNH CÔNG hoặc THẤT BẠI. Nếu mã KHÔNG LỖI, trường thông báo sẽ chứa thêm thông tin về những gì đã xảy ra.

tên người dùng

UserId tương tự đã được bao gồm trong thông báo tải lên

con số

Số lượng bản ghi được tải lên

tên

Tên của các bản ghi được tải lên

tin nhắn

Một thông báo cung cấp thêm thông tin về những gì đã xảy ra. Đặc biệt quan trọng nếu uploadResponseCode không THÀNH CÔNG

Một phản ứng ví dụ được đưa ra dưới đây. Lưu ý rằng thứ tự của các trường có thể thay đổi

{"uploadResponseCode":"THÀNH CÔNG",

"userid":"wm123",

"số 2,

"names":"Hội nghị Android, Cuộc họp Khách hàng",

"tin nhắn":"tải lên thành công – tất cả đã hoàn tất!"}

Ứng dụng của bạn sẽ hiển thị dữ liệu được gửi trong phản hồi. Bạn chỉ có thể hiển thị thông báo JSON thô, nhưng sẽ tốt hơn nếu định dạng đầu ra để giúp người dùng dễ đọc hơn.

Tạo nguyên mẫu đa nền tảng của ứng dụng (10%)

Triển khai càng nhiều càng tốt các tính năng a) bằng cách sử dụng các khung nền tảng chéo.

Thực hiện kiên trì sử dụng đa nền tảng (10%)

Triển khai càng nhiều càng tốt các tính năng b) bằng cách sử dụng các khung nền tảng chéo.

Thêm các tính năng bổ sung cho một trong hai hoặc cả phiên bản Android hoặc đa nền tảng của ứng dụng (10%)

Các tính năng từ a) đến g) là các yêu cầu cốt lõi đối với ứng dụng. Nếu bạn đã triển khai những tính năng này và muốn thêm một số tính năng bổ sung thì bạn có thể. Bất kỳ cải tiến nào cũng nên được triển khai ngoài KHÔNG thay vì các yêu cầu cốt lõi. Ý tưởng là những tính năng này nâng cao kỹ năng của bạn, vì vậy hãy chuẩn bị để thực hiện nghiên cứu của riêng bạn và thoải mái thể hiện! Bạn có thể nghĩ về những cải tiến của riêng bạn. Dưới đây là một số ví dụ có thể:

Cho phép thêm ảnh do máy ảnh chụp vào dữ liệu được lưu trữ

Nhận vị trí tự động từ vị trí của người dùng

Hiển thị vị trí phát sinh chi phí trên bản đồ

Sử dụng một dịch vụ web bên ngoài khác với dịch vụ được cung cấp cho khóa học.

Bất cứ điều gì bạn có thể nghĩ đến - một lần nữa hãy sáng tạo!

Tạo nguyên mẫu đa nền tảng của ứng dụng (10%)

Triển khai càng nhiều càng tốt các tính năng a) bằng cách sử dụng các khung nền tảng chéo.

Thực hiện kiên trì sử dụng đa nền tảng (10%)

Triển khai càng nhiều càng tốt các tính năng b) bằng cách sử dụng các khung nền tảng chéo.

Thêm các tính năng bổ sung cho một trong hai hoặc cả phiên bản Android hoặc đa nền tảng của ứng dụng (10%)

Các tính năng từ a) đến g) là các yêu cầu cốt lõi đối với ứng dụng. Nếu bạn đã triển khai những tính năng này và muốn thêm một số tính năng bổ sung thì bạn có thể. Bất kỳ cải tiến nào cũng nên được triển khai ngoài KHÔNG thay vì các yêu cầu cốt lõi. Ý tưởng là những tính năng này nâng cao kỹ năng của bạn, vì vậy hãy chuẩn bị để thực hiện nghiên cứu của riêng bạn và thoải mái thể hiện! Bạn có thể nghĩ về những cải tiến của riêng bạn. Dưới đây là một số ví dụ có thể:

Cho phép thêm ảnh do máy ảnh chụp vào dữ liệu được lưu trữ

Nhận vị trí tự động từ vị trí của người dùng

Hiển thị vị trí phát sinh chi phí trên bản đồ

Sử dụng một dịch vụ web bên ngoài khác với dịch vụ được cung cấp cho khóa học.

Bất cứ điều gì bạn có thể nghĩ đến - một lần nữa hãy sáng tạo!