# IN721 2017 Main Project: Complete Mobile Application

For this project, you will design, test and implement a complete mobile application. The project will be performed in phases, as described below. Penalties will be imposed for any missed deadlines.

Your goal is to create an app, which, if uploaded to the Google Play Store, would be downloaded and used by real people.

## Core requirements

Your project must:

1. Be selected from one of the options given below.
2. Be implemented in Android, targeting a typical smart phone running API 18: Android 4.3 (Jelly Bean) or higher (tablet version **not** required).
3. Fulfil all agreed functional requirements (see below).
4. Have a user interface designed in accordance with correct Human Computer Interaction principles.
5. Be implemented with elegant, correctly structured, fully commented code.
6. Incorporate feedback from formal task-based user testing (see below).

## Application Options

Your application is to be selected from one of the following. In all cases, you may assume that internet access will always be available. Applications that use location will be tested with mock locations.

1. **Shopping Wish List**: Allow your users to take pictures of items they see while out and about, which they think they might wish to purchase in the future. They need to be able to save information about vendor and price, to add notes, and to peruse their wish list later in an organised way.
2. **Outdoor Activity Weather Checker**: Trampers, mountain bikers and other outdoor enthusiasts need to be able to check upcoming weather conditions as they plan their routes for the day. Build an application that lets them do this efficiently. Consider very carefully the appropriate interface for specifying a route.
3. **Parking Locater:** When you drive to a new city, you want to be able to quickly and efficiently find public parking. Build an application to provide this service. It should locate appropriate available parking options dynamically and provide the user with the necessary information to find them. (Some parking data services cover only a limited area. This is fine; simply indicate, in your proposal, the area to which you will restrict use.)
4. **Language Translator:** When travelling, it is often essential to be able to get a translation of the local language. Build an app that allows the user to translate **individual words** from various languages into English. This app must support at least two of: Spanish, French, German and Italian. For those languages, it must provide audio to demonstrate correct pronunciation. You may support any other languages that you wish, with audio optional. If you support a language that requires an alternative keyboard, make sure to design clear UI for access.
5. **Commute Checker:** Allow your users to check for any reported traffic incidents before they set out on a drive. Your app should be useful for a short drive around town or a long trip. Again, consider very carefully the appropriate interface for efficiently specifying a route.
6. **What To Do Around Here:** Build an app that lets people find out what interesting tourism sites and activities are nearby when travelling. Your application should adjust dynamically to the user's location. Provide as much information as possible, but make sure it's easy to read, navigate and understand.
7. **What's Happening There:** Let your users check on the current situation in their travel destinations by providing an easily browsable and readable summary of relevant up-to-date news reports based on a provided location.
8. **Nearby Pictures:** Build an app that searches for and displays images taken at the user's current location. This app must update dynamically, have a very carefully designed interface to allow convenient **vertical** **and** **horizontal** navigation, and must source images from at least three different web services.

## Phases & Due Dates

All deliverables must be uploaded to your IN721 repo by the indicated due dates.

|  |  |  |
| --- | --- | --- |
| **Component** | **Comments** | **Due Date** |
| Project proposal.  Word document(s) or pdf unless specified otherwise. | * Statement as to which of the available projects you intend to implement. * Complete proposed **formal** functional requirements for your project. Please separate “The User Shall…” and “The System Shall…” portions into their own tables. If you propose either too little or too much functionality, we will negotiate an adjustment. This is the spec against which your Functionality and Robustness mark will be determined. * A list of all web services you intend to use (if any) with a working example URL from each. * A list of required on-board hardware, if any. (That is, must the phone have a gyroscope, or a temperature sensor, front and back cameras, etc.) * Complete wireframes for your project. These should be hand-drawn on paper. Upload images or scans. | Friday, 26 May, 5.00 pm |
| Testable beta | * A completely running version of your app for formal task-based user testing. There will be time during class to do user testing; you may wish to do more outside of class to gather more data. Attendance at class on Wednesday 7, June is mandatory. Come with your working beta and a prepared task list for user-testing. | Wednesday, 7 June, 11.00 am. |
| User-testing report | * Detailed report of user testing protocol including number of subjects run. * Detailed description of issues raised during user testing and specifics of action taken to resolve each issue. | Friday, 16 June, 5.00 pm |
| Release version of application | * Final project files | Friday 16, June 5.00 pm |

## Web Services

Some of the easy-to-use web services don't provide very high quality information. This is fine for the purposes of this project. You can fulfil the web services requirement of this assignment via any publically accessible remote service, without penalty. All of the following, for example, are acceptable (and there are many others).

|  |  |
| --- | --- |
| **Service** | **Find Information At** |
| Traffic data | msdn.microsoft.com/en-us/library/jj136866.aspx |
| Language translation | www.frengly.com |
| Dictionary | www.dictionaryapi.com |
| Weather data | openweathermap.org/api  www.wunderground.com/weather/api |
| Travel and tourism sites | www.tixik.com/info/api/ |
| Geocoding (i.e. to get the lat lng from a place name or address) | nominatim.openstreetmap.org |
| News | http://open-platform.theguardian.com/access/ |
| Parking data | www.parkwhiz.com  www.parkme.com |
| Maps | GoogleMaps |
| Images | https://www.flickr.com/services/api/ |

## Marking Schedule

|  |  |
| --- | --- |
| **Component** | **Weight** |
| User-testing report | 10% |
| Look and feel | 10% |
| Interface design/usability | 20% |
| Functionality & robustness | 30% |
| Code elegance | 30% |

<http://www.tixik.com/info/api/>

API Key: 20170524734838062865655

<http://www.tixik.com/api/nearby?lat=-45.8787605&lng=170.5027976&limit=10&key=20170524734838062865655>

Google Places

**https://maps.googleapis.com/maps/api/place/textsearch/json?location=-45.8787605,170.5027976&radius=1000&type=attraction&key=AIzaSyBOm2SY3mY9Awm8qmxoIMWT236M3bGZB8k**

**https://maps.googleapis.com/maps/api/place/textsearch/json?query=restaurants+in+Dunedin&key=AIzaSyBOm2SY3mY9Awm8qmxoIMWT236M3bGZB8k**

https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photoreference= &key=AIzaSyBOm2SY3mY9Awm8qmxoIMWT236M3bGZB8k