Timetable app with student-generated module feedback

PAVELS BARTUSS
B.A. (Mod.) Business and Computing
Final Year Project April 2013
Supervisor: Owen Conlan

School of Computer Science and Statistics
O'Reilly Institute, Trinity College, Dublin 2, Ireland
DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

Name

Date
Acknowledgements

I would like to thank Owen Conlan for agreeing to supervise my project and support that he provided. Vladislav and Patrick for pointing out things that needed improvement. An engineer student that I met in LG12 who logged into my application in order for me to see how it behaves with different timetable for the first time (sorry, I cannot remember your name).

Thank to Google that cached thousands of questions and answers, and strangers who submitted them.
Table of Contents

Abstract ................................................................................................................. 5

1. Introduction ........................................................................................................ 6
   Motivation ........................................................................................................... 6
   Research question and objectives .................................................................... 7
   Structure ............................................................................................................. 7

2. System Analysis ................................................................................................... 8
   Features ............................................................................................................. 8
   App 1 ............................................................................................................... 8
   App 2 ............................................................................................................... 9
   App 3 .............................................................................................................. 10
   Summary .......................................................................................................... 11

3. Design .................................................................................................................. 13
   Goals and Constrains ........................................................................................ 13
   Parts of an Application ....................................................................................... 13
   User Interface ..................................................................................................... 13
   Web Scrapping Method ...................................................................................... 16

4. Implementation ................................................................................................. 18
   Software ............................................................................................................. 18
   Per Day View ..................................................................................................... 18
   Modules Management ........................................................................................ 21
   Feedbacks ........................................................................................................... 22
   Internal Data Storage ......................................................................................... 23
   Web Scrapping ................................................................................................. 23

5. Evaluation ............................................................................................................ 26
   The Application .................................................................................................. 26
   Comparison ....................................................................................................... 27
   Use Cases .......................................................................................................... 28

6. Conclusion ........................................................................................................... 31

References ............................................................................................................. 32

Appendixes ............................................................................................................. 34
Abstract

Trinity College offered online personal timetable for quite long time. Most recently, students were introduced to my.tcd.ie website, which offered wide variety of college related services that students could access anytime implying that they have Internet access. This included timetable with even higher options for customization than ever. The process to view the timetable is simple and requires as much as three clicks and an input of account details. Trinity College has put some effort to support mobile devices, for example, Apple mobile devices were allowed to connect to the Wi-Fi infrastructure of the college and it was promised that in near future Android devices would be able to do the same. At the time of this research, my.tcd.ie website did not provide mobile version as the result it is not user-friendly if accessed from smartphone or other mobile device. To top it off it takes about 40 seconds to pull information from the database (not including navigation through website). All this in era of the smartphones, a Swiss knife of electronic devices, in a pocket within the reach of hands that grants access to nearly unlimited source of information in a few seconds but not TCD timetable. This paper will analyse existing timetable apps and go over the process of creating one, specifically designed to extract timetable from my.tcd.ie, with some additional functionality.
Chapter 1

Introduction

In Ireland, smartphone ownership was as high as 71% at the end of 2012 (gaumina.ie, 2012). Android, an open source Linux based operating system owned by Google, was realised in late 2008. Soon it became a leading mobile operating system on the market. By the end of 2012, over 60% of all smartphones were running on Android (idc.com, 2012). There are about 800,000 applications available on the official app store. Number of tools available to create applications for android, some of which are meant to be used by people with little programming knowledge such as drag-n-drop style building of the app (appinventor.org) or more traditional ones using software development environments.

Motivation

Throughout four years at Trinity College, I have observed different ways that students use to keep track of their timetable. Some were printing off the timetable and highlighting the modules that they do. Sometimes they would just download the .pdf file consisting the timetable and store it on their phone/laptops. Others used third party timetable applications on their mobile devices and manually inserted each lecture. Taking picture with a camera or a screenshot and storing it on the device was an option too. The online timetable provided by the college looked like a perfect solution but the accessibility did not favour it. I, myself, trying few ways and while at the end they all provide what is needed in one way or another, they have their own downside and sometimes are tedious to start using them. The growing popularity of the smartphones gave me an idea to merge the use of these devices and personalised timetable provided by the college into one thing that would not require constant Internet connection and will be accessible within few seconds. An additional function of an ability to exchange thoughts, tips and generally useful information about the modules, among the students taking those modules, came from the class meetings that we had during the academic years, when juniors were invited to ask us questions they would like to. I decided to implement it on an Android platform
because I had experience in Java programming and I was familiar with XML, which can be used to layout the structure of the interface.

Research Question
Create an offline Android timetable app by web scraping the my.tcd.ie website and provide additional functionality.

Research Objectives

Analyse – existing timetable apps available on the market, by comparing their pros and cons. Identifying key features.

Design – timetable app with an option to extract timetable from my.tcd.ie website. Add an ability to exchange module feedbacks between students.

Implement – Web scraping mechanism on an Android based mobile device. It should support login into secure website (https) and storing extracted data on the device. Finally, a method to exchange feedbacks between students.

Validate – Test the application. Provide use cases of the system. Compare implemented application with existing solutions. How successfully does it meet the objectives?

Structure
In chapter two of the paper I will go over few timetable application that are currently available on the Play Market (official Android app market), analyse them and compare their features. In chapter three design and architecture of my own application. Chapter 4 will list tools and describe software solutions that were used to meet the design specifications and support functionality of the application. In chapter 5, I will evaluate the implementation using use case scenarios, my own view as a student and comparison with existing solutions. Chapter 6 of this paper will conclude the research and give an insight of what can be improved in future work.
Chapter 2

Analysis

There are many timetable applications available in the market. Most of them are free but some are paid ones. For the purpose of this project, I will chose free applications targeted specifically at the students. Key features that I will be looking at are as follows:

- **User Interaction.**

How exactly user can navigate through the application. How easy is commonly viewed information accessed. Overall how intuitively navigation is.

- **What information is stored and how it is displayed.**

Apart from obvious time and lecture title, can user add in which room/building the lecture is going to be? Lecturers name. Any extra information. Is all data displayed if so how it is structured? Is it easy to read?

- **Performance**

Start-up time. Empty timetable performance. How timetable behaves with data stored in it?

- **Extra features not directly related to the main task.**

App 1

There are applications designed for specific colleges but unfortunately, most of them have no manual insertion and require login details therefore I was not able to test them on a full scale. Such is the case with unofficial Dublin Institute of Technology Timetable app. From its description, one can learn that it serves similar purpose as the app I am going to implement. On launch, it asks the user for login details, presumably used to access online college timetable, thankfully this dialog can be closed after which I was able to enter main part of the application. The app provides two options, to extract timetable into the application and/or download pdf file into the device in both cases data is stored locally, therefore independent from the server uptime. One can set the range of the timetable to cover semester or specific week range before downloading. It uses tabs for navigation; the buttons for weekdays Monday to Friday are at the top of the screen. It
seems that app provides no gesture/swiping support. The input dialogs are plain and simple. It is difficult to comment on the performance, since timetable cannot be populated manually, nor do I have access to the DIT timetable service to check how long it takes to extract timetable. While timetable is empty there are no complains towards the performance. There are no extra features, the app serves its solo purpose of a timetable.

App 2

Another application that is going to be analysed is College Timetable published by AndroidEire. App is not linked to any particular college, as the result timetable must be filled manually. When launched application shows a list of weekdays Monday to Sunday and a settings button. By selecting a day from the list user will be presented with lecture list for that particular day and an add button at the bottom of the screen. There is no way to switch between days without going back to launch screen. There are few options to select in the setting screen. Upon adding a lecture, the data entered by the user is divided into three parts – Subjects, Rooms and Lecturers. All three are included in Settings screen for simpler management, for example, instead of changing room for each lecture in the timetable one can do it from Rooms section in the settings and changes will be applied to all of them. Interesting to note that delete any entry in any of the three sections has no effect on the timetable itself. Along with those three buttons, there are Delete Classes button, Tasks, Save, Open and Share timetable. It is possible to export timetable into a single file and store it in desired location including cloud, like Dropbox, so that after reinstalling there will be no need to manually insert all lectures again. In the task section, one can add particular tasks that will work as a reminder; unfortunately, it seems like it has no push notifications therefore it is up to the user to look up what tasks are yet to be completed or when is the deadline for them. Task section is not linked to the Subjects list, as the result students will have to type in Subject for each task, it would be logical to add drop-down list of the Subjects that are already in the timetable. Overall, it is an interesting add-on for the timetable but current state requires some improvements. The theme of the app is in very bright colour patterns nothing like default themes that come with Android operating system, which may not suit some of the users. Furthermore, alert dialogs use native Android theme, which looks out of place. All input dialogs are well structured; it is easy for the user to switch between different input areas. The College
Timetable has no gesture or swiping navigation; on screen buttons or physical buttons of the device (back button) are used for this purpose. There are no performance issues, even with timetable consisting 20+ lectures everything works smoothly and app launches within seconds.

App 3

A+ TimeTable added by timesoft is a third application to be analysed. It has no link to any college, so the only way to fill the timetable is to do it manually. Similarly, to DIT timetable app it uses tabs for navigation, by default ranging Monday to Friday but can be altered to begin with Sunday or to include weekends. It has a week view, a table that shows all lectures for whole week. To enter week view one can press middle, on screen, button at the bottom of the screen or swipe left/right while viewing any particular day but in order to leave this view one needs to tap back button. From user perspective, it is a strange choice, a lot of application that use tabs for navigation adopt swiping as a mean for switch between those tabs, for example, going from Monday to Tuesday tab would require user to swipe right to left. The week view is not a tab; it is a separate screen with relatively different purpose. It is possible to switch from tab view to “simple view” which removes tab buttons on the top and adds left\right buttons. With this setting enabled swiping no longer opens week view, in fact it does nothing, which again is a strange choice. Populating a timetable is an easy task, there is an option to copy same lecture from one day to another, which may simplify tedious task. One is asked to enter subject, room, lecturer, select time and a colour that will reflect this lecture on the week view screen. There is no connection between entered subjects - each lecture in the timetable treated as a separate entity, as the result, changing lecturers’ name for one of the subjects in the timetable will not have any effect on other subjects with the same lecturer. For unknown reasons add subject dialog does not disappear nor are the input fields nullified when save button is clicked, although, small notification appears that process was successful. Same as in previous application user can export created timetable in this case with two options – a picture (jpg file) of a week view or as a file of a daily schedule that later can be imported if needed. The interface elements are of a softer colour pattern and resembling native Android theme. This app has a built in advert bar that appears above navigation buttons. Instead of having predefined space for the advert, tab buttons take its
place and since from all user interface elements advert bar loads last (sometimes taking few seconds to load) buttons then get repositioned below it. This leads to accidental clicking on the ads if user decides to switch to a different day of the week right after the application loads. Is it a design flaw or done intentionally, remains a question, either way it damages usability of the program. There are no performance issues apart from minor animation glitches when entering settings or tapping subject on week view. This animation has no real purpose and can be easily omitted without any loss. In the menu one can find an item called “Read”, it reads out the timetable subjects depending on selected day. It uses built in Talk Back function of the phone that is supposed to improve accessibility for blind or low-vision people. Unfortunately, it does not recognise time, for example, 01:00 will be read as “one hundred hours” along with that it reads title of the subject and its lecturer. If the user has enabled Talk Back function in the phone settings, it will read the text without the need to press anything else within the app, which makes Read button obsolete. The application loads and works well when empty and with 20+ lectures in the timetable.

Table below summarises findings.

<table>
<thead>
<tr>
<th></th>
<th>Navigation</th>
<th>Displayed Information</th>
<th>Structure of timetable</th>
<th>Extra Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIT Timetable</td>
<td>Buttons</td>
<td>Time; Subject (including type, i.e. lecture/tutorial etc); Lecturer; Room;</td>
<td>Tabs;</td>
<td>Download timetable as pdf file;</td>
</tr>
<tr>
<td>College Timetable</td>
<td>Buttons</td>
<td>Time; Subject; Lecturer; Room;</td>
<td>Separate views for each weekday;</td>
<td>Tasks; Exporting timetable;</td>
</tr>
<tr>
<td>A+ TimeTable</td>
<td>Buttons/ partial swiping</td>
<td>Time; Subject; Lecturer; Room;</td>
<td>Tabs + week view; Colour differentiates subjects;</td>
<td>Exporting timetable; Week view;</td>
</tr>
</tbody>
</table>
All three analysed apps share very similar design of the single frame of lecture within timetable. Beginning and end time are on the very left side. Lecture title takes up most space – usually positioned right after the time or middle of the frame in significantly larger font size. There is some variety of lecturers’ name and room placement in relation to other objects. In two of the cases room and lecturers’ name is below the title, to the left and right side of the frame. If room or name line are too long to fit into one line, Student Timetable app enlarges the frame and splits the text, in other words each textbox has a strict width size but flexible height. Lecturer title treated in the same manner. It loses its neat look but this way it gets rid of the problem of overlapping text that A+ TimeTable encounters. The latter does not restrict width of the Room and Lecturers’ name textbox, in situation when both hold more than few words text overlaps making it impossible to read. It allows two lines of text for lecture title, if that is not enough second line ends with “…”. The A+ timetable uses native time picker dialog, which is the best choice as users are most likely familiar with it from using it in other applications. Student timetable uses much simpler version of time picker dialog with counter, plus and minus signs. It is slightly slower as one cannot scroll through the time selection instead, user would need to click or tap and hold respective sign.
Chapter 3

Design.

Goals and Constraints
Allow students to create their own timetable on an Android device by populating timetable manually.
Allow safe logging into my.tcd.ie and a method to extract data from the timetable.
Extracted data must be stored on the device without future need for Internet connection unless user decides to extract timetable again.
For an easier management, some fragments of the data should be stored separately.
Module<->Lecture seems to have the strongest link, provide additional screen to manage the list of modules and its lecturers.
Allow students to exchange feedback on modules that they take. Provide some sort of database that will store and pull feedback on request.
In order for an app to be beneficial for the user, the time it takes to view the timetable must be shorter than using browser to view it on my.tcd.ie.

Parts of the application
Per day view. Tabs that host each day. Buttons and swiping support to switch between days.
Menu. Should have Extract timetable button, which will open login dialog. Module management section consists list of modules and lectures. Add new item into timetable for manual insertion.
Feedback section, most likely called from the module management section. Must use some kind of identifier to differentiate each module.
Context menu, depending on the section of the app should allow user to edit/delete items.
Login dialog, request users account details in order to login into my.tcd.ie. Upon login, display processing dialog.
Add new item dialog, depending on the section of the app, ask for user input – module title, lecturer, start/end time and day of the week.

What information is going to be displayed and user interface issues raised with it

The idea is to extract the information from the my.tcd.ie website. It was an essential step to analyse the website and based on the finding and features discussed in chapter two - design timetable. On the website, timetable presented as a table: time on the left, days on the top. Each cell in the table is either empty or holds particular set of information about lecture. This information include: range of academic weeks selected lecture lasts; Module code; Size of the group; Group – usually an empty field; Activity – lecture/tutorial/lab; Room – including name of the building in brackets; Sometimes lecturers name. Picture on right shows an example of a single cell.

When user puts mouse cursor on top of the cell additional textbox appears with following information: Module – full title of the module without its code; Date – day of the week; Time – in a 24-hour format, beginning and end of the lecture. Here is the picture of how it may look.

It was obvious that not all of this information is required in day-to-day timetable. At least four parts of this information were vital – time, module title, date (day of the week) and lecturer - all four appear in all cases of the existing system analysis and it is simply what any timetable consist. Many modules share similar or even identical words in the title
such is the case with most of the first year modules, when they begin with the word “Introduction to”. I decided that module code should be included in its full title, as quite often students recognise modules by their codes. By adding the code to the beginning of the title, students may easily identify what module they have next just by glancing at it in the app. Week range, size and group seemed unnecessary in the timetable for various reasons. First, both size and group are very inconsistent, in many cases they are omitted from the timetable on my.tcd.ie, appearing as a blank line. As a student, I cannot find any case in which size of the group would be of any usefulness to a student. My assumption is that group is for the lab or tutorial activities but even then, this line is empty at least in the timetables that I have seen. Either my.tcd.ie is not supplied with enough information to fill this line or it is not used for other reasons. The academic year is divided into two terms. First term begins at week 5 and ends at week 16, second term ranges from 21 to week 32. By default timetable on the website shows first term of the year. The apps must allow student to choose for which term he or she wants to get timetable. If user decides to get timetable for second term and some module lasts two terms, the week range in the timetable still displays Wks 5-10, 12-16. There is no reason to include this information in the timetable, as it provides no value to the end user in day-to-day usage. Another aspect that must be taken into account is that module titles and rooms sometimes consist rather long string. Previously seen solutions of the way to position this information may not fit in this situation.

Each view of the weekday will be identical in terms of layout. There are four pieces of information to be displayed in each lecture slot. As mentioned above, extra space must be given to few of them. I have decided to put module title, lecturers’ name and room on a separate line, given maximum length space for all three. For an easier readability, title must be highlighted presumably with a bigger font size. For a better look, each lecture slot will be of a fixed height size – lines that do not fit will be shortened and displayed in full if selected. This should not raise any accessibility issues as modules can still be identified by their code or by the part of the title that will be displayed. All lecture slots in a single day view have to be sorted by time in ascending order.

Module management section adopts similar look of a single day view but consist less information. Each slot will be unique in the list and will include full module title with
lecturers’ name. Consider possibility to add unique colour for each module. This
technique observed in one of the existing solutions that was analysed in chapter two.
From a personal experience, it seems to make process of identifying particular module a
lot easier and overall adds a better look for the app. User will be asked to fill this section
first, before filling the timetable. Any changes in this section will be automatically
reflected in the timetable itself. It should make timetable management an easier task –
each time user wants to add new lecture he or she will not need to type module title or
lecturers name, they will appear in the drop-down menu. Similarly, if one needs to
change lecturers name or title single editing in this section will change all entries in the
timetable.
Feedbacks are going to be stored in a SQL database. To enter this section the mobile
device has to be connected to the Internet. It will present list of feedback with their
ratings in descending order. Upon receiving particular amount of negative votes,
feedback will be automatically removed. Moreover, students will be able to report
responses that they deem to be inappropriate. Each entry will be of a strict format, student
could comment on a list of given topics all related to the module. The list will include
lecturers, tutorials, materials, examination and workload. The list is not completed and
might be expanded in future.

Web scrapping method
My.tcd.ie website provides no API and no direct access to the database, so the only
possible way to get data from the website is by web scrapping. It is a technique of
extracting the information from the website.
To scramble the data from the web page, the app needs a method capable of connecting
to a website, sending HTTP POST & GET requests to it, and view the page source
(HTML), i.e. store the page as a text object. The POST method request the target website
to accept the data in our case there are at least two pieces of data to send – user account
details (username and password) and possibly term selection on the timetable page.
HTTP GET method, on the other hand, used to retrieve the data from the specified URL.
Since the webpage requires login, it holds session for each such successful attempt. The
session kept alive using cookies. Cookies are small files stored by web browser that can
identify the user. Therefore, the web scrapping method must be able to store at pass
cookies along with the connection attempts. There are number of HTML parsing libraries
available for Java that support similar behaviour but not all of them are capable of
working on an Android platform. The most noticeable libraries are – HTMLUnit nd
JSoup. For the purpose of this application, the process of scrapping the webpage should
be a quick and non-resources consumable, as it will run on a mobile device with a limited
processing power. Upon implementation, further research/testing will be required to
decide which of the available libraries can work on a given platform and best suits the
needs of the system.

By looking at the source of the my.tcd.ie timetable, it becomes obvious that additional
formatting is required. The pop-up text field mentioned above is a part of the Java Script
and embedded into a HTML tag. HTML parsers extract data based on those tags and its
elements, and is not design to select pieces of information within the elements’ attributes.

For example, a part of a cell in the timetable may look like this:

\<td class="fullBorder" rowspan="2" onmouseover="\"tooltip('Module: COMPUTER GRAPHICS\r\nDate: Monday \r\nTime: 15:00 - 16:00')\" \r\n\"\>…\>.

Using HTML parser one can extract “onmouseover” and get the underlined string but
only highlighted text within the given string is meaningful for our purpose. The
information displayed in the pop-up text field is vital for the timetable app, as the result
there is a need to implement method to format the raw string and get required data.
Chapter 4

Implementation

In this chapter, I will first go over the software used in the process of implementing the application and detailed description of each part of the app listed in the design section.

Software

For this project, I will be using Eclipse IDE with Android Development Tools (ADT) plugin as a primary software. ADT extends Eclipse by allowing creation of android-based projects, also comes with XML editor, debug and testing Android SDK package that can emulate mobile device so there is no need for a physical devices. Additionally, I will be using Opera browsers’ built-in Dragonfly developer and debugging tool, specifically Network Inspector that logs all HTTP requests and responses to help navigate through my.tcd.ie website. Few external libraries will be added as project development goes on.

Main Screen

Timetable day view

I began by created main activity that would host daily view of the timetable. Android by default provides class called TabHost, a container that holds buttons for navigation and a desired layout for the body of the tab content. It does not come with swiping support so it had to be added as a separate method. I have created five tabs labelled Monday (Mon) to Friday (Fri). Each tab hosts vertical list of items in this case lecture slots. All items in this list have identical layout consisting five text fields - time of the beginning of the lecture and its finish time, lecturers’ name, module title and place where this lecture is going to be – as well as a colour bar on the left. The position of the elements declared in XML so that none of the text overlap, any text that is too long to fit into single line is shortened to “…” and displayed in full if tapped. In other words, each element is aware of the existence of other elements positioned on the same y-coordinate. I have tried to utilise
given space to a maximum extend at the same time highlighting the most important parts of the information. The default look of a single lecture slot shown in the picture below.

Room/Building can take all space up-to the “Start” and Lecturer can extend all the way to the Finish time. Module title text can take all space to the end of the screen.

I added simple method that would detect current day of the week using Java Calendar class at the time of an app launch. Based on the current day it would automatically switch to the corresponding tab. When student will open the app, quite likely he or she would like to check today’s lectures, using this method user will not need to switch tabs. If current day is Saturday or Sunday app will switch to Monday.

At this stage, there was only one way to switch between tabs using buttons on the top of the screen. The android provides gesture recognition class that I decided to try. In order to use it one need to define gestures using third party application and import file with defined gestures into the project. It allows pre-define relatively complex gestures, which it is capable to recognise but at some performance trade-offs. It took seconds to recognise simple left to right/right to left gestures making it frustrating to use and definitely not suitable for this type of application, so I looked for another method. This time it was Motion Event detector, which describes motions in terms of axis values. For detecting simple swiping, I needed to get beginning x-axis position of a motion event and its end value as well as velocity of the movement. From the user perspective – when finger touches the screen and realises, and how quickly the movement was. After getting these values and comparing them to minimum motion distance and velocity, it is a straightforward task to tell if it was left to right or right to left motion, based on which tab switches to previous or next one.
Menus
The context menu appears when user taps and holds on the item, it gives two option to choose from Edit and Delete.
I added few items into main menu (options) that is called by user by either pressing physical button of the device or three-dot button on the screen, depending on the device. At first, I made all items to appear in the drop-down menu, the list included Modules, Add Lecture, Extract Timetable, Delete All, Help and About. After some testing, I realised that at least two items, Modules and Add Lecture, were used the most and it would be a better idea to let access them within a single click rather than forcing user to enter Menu first. Besides, Module part of the application plays important role and users must be aware of it existence. Android lets menu items to be placed on the action bar at the very top of the application next to its title (if present). Placing a Modules button in that location seemed like a best option. It both catches user attention and easily accessible at any time. I tried placing Add Lecture button next to the Modules but it made action bar crowded with text and possibly confuse users due to their relatively similar meaning, so I looked for a better solution. The ListView class that hosts lectures slots has method that allows adding a fixed footer at the end of the list. I made simple button with ‘+’ sign of the same width and height as a lecture slot and added it to the footer. Users have two option for adding lectures selecting it from the menu or tapping a button at the end of the lecture list. If there were many lectures on a given day, one could select it from menu without the need to scroll all the way down or tap a ‘+’ button right away if list fits the screen.

Dialogs
Since user adds all modules and lectures before filling the timetable, add lecture dialog lets user to pick module from drop-down menu. Days of the week provided in drop-down list too. The user selects time using default time-picker dialog that he or she should be
familiar with. The only text input that is required is a building/room where lecture takes place.

To extract timetable user needs to select appropriate option from the menu. The login dialog warns that current timetable will be over-written, has two text input fields for username and password, and two check boxes Michaelmas term and Hilary term respectively. Few alert dialogs can be triggered if operation was unsuccessful or due to wrong input. While extracting the timetable, it displays simple processing dialog.

When application launched for first time, it welcomes user with a short instruction on how to use the application, explaining each part of it. This welcome/help dialog can be called again by selecting ‘Help’ from the menu. Even through app is straightforward to use, the main reason for this dialog is to address the fact that Modules must be filled out first. However, it also gives quick overview of other functions.

**Module Management section**

For consistency, this section took similar look as a single day view. It is a vertical list of items, which contains slots for each module, its lecture and colour bar associated with it. Each slot has a pre-defined layout in XML file that by default looks like this:

Upon adding new module, student will decide which colour he or she wants to associate with it. I have added a colour selector dialog, which is not a part of my work. It is an Android open source project, widely available to the public (java2s.com). It lets user to pick colour from a wheel that follows RGB model and then returns selected colour as a hexadecimal value.
This section has similar user interaction in terms of a context menu and adding new entries as a main section of the app except for one difference. When user single clicks on the module and module title begins with its code (e.g. CS4051), it will open a feedback section for selected module.

**Feedback section**

Here students can leave their feedbacks for modules that they take. Each module has its own rating, all students can vote once but they can always change their mind. The voting should help students in finding most useful responses. When student adds new feedback, he is presented with a drop-down list of topic to choose from. They include Lectures, Tutorials, Assignments, Tests, Examination, Workload, Materials, and General. These topics bring in some structure into each feedback, which makes it easier to read and gives a better look, compared to a chunk of a plain text. Along with voting students may report feedback that they deem to be inappropriate by tapping and holding on the feedback. Each field is optional. Here is a screenshot of how single feedback may look:
The feedbacks are stored in MySQL database hosted on digiweb.ie. The database consists of two tables. Table one-called modules with a primary key module code.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Collation</th>
<th>Attributes</th>
<th>Null</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>varchar(6)</td>
<td>latin1_swedish_ci</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second table called comments holds feedbacks (Comment as a text), module code (foreign key), date with time (including seconds) as a primary key, rating as integer and a report field as tinyint (Boolean).

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Collation</th>
<th>Attributes</th>
<th>Null</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>text</td>
<td>latin1_swedish_ci</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vote</td>
<td>int(11)</td>
<td>latin1_swedish_ci</td>
<td>No</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>varchar(6)</td>
<td>latin1_swedish_ci</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>datetime</td>
<td>latin1_swedish_ci</td>
<td>No</td>
<td>0000-00-00 00:00:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report</td>
<td>tinyint(1)</td>
<td>latin1_swedish_ci</td>
<td>No</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The app creates a thread that handles the connection and queries with the database using MySQL JDBC Connection library.

**Internal Storage**

Android supports few options for storing data. These options vary from internal to external and affected by the size and/or type of the data. The data that has to be stored is well structured and require some space, and meant to be stored on the device. I decided to use SQLite database.

For this purpose, I have created two classes, one responsible for initialising the database and tables, and another one that will be responsible for the queries. The application requires two tables to store modules and lectures. Modules table store title of the module, lecturer’s name and a colour. All fields are of a text type. Lecture table stores module, day (integer), start and finish time of the lecture as a text.

The application may execute number of queries during its run-time. It needs to insert data into both tables, update it if user edits something, delete and load data from the database. I have also created third table that stores users’ votes from feedback section. All feedbacks that are stored in MySQL database (external) are anonymous; there is absolutely no link to any particular user or a device as it may have raised privacy issues. In order to limit one vote per user it had to be recorded somewhere.
Web scraping.

The login, extraction and formatting of the timetable from my.tcd.ie are included in the method of the main class. Android does not allow network connection from the main UI thread of the program so I used AsyncTask class that has similar behaviour of a thread in Java. It performs the operation in background and publishes the result back to the UI thread. I was choosing between two HTML parser frameworks HTMLUnit and JSoup. HTMLUnit is pretty much text based browser that simulates web browser behaviour, whereas JSoup is slightly simpler in this matter. I looked through many documentation and examples on how both used for similar purpose, and found that it is very difficult to set up HTMLUnit to work on the Android platform. In fact, quite few reported that they were not able to make it to work and switched to other solutions (stackoverflow.com, 2012). On the other hand, there were many successful attempts to use JSoup on Android so I tried it. JSoup uses CSS like selector to query over the data.

When user presses login in the extract timetable dialog the AsyncTask executes (assuming user supplied correct inputs). It attempts to connect to my.tcd.ie for the first time, the attempt will be unsuccessful since it requires cookies but it acquires them during this attempt as well as a key passed along with user details upon login. It connects to my.tcd.ie for second time, now with cookies and a key. JSoup by default is supposed to follow redirecting but for unknown reasons it does not in this case, so it has to scrap the page and look for a redirect link. It repeats similar operation looking for the links that lead to the timetable. The timetable on my.tcd.ie by default shows first term. If user selected first term in login dialog, it then stores timetable into temporary document file and queries it, looking for specified tags and attributes. It extracts tag with attribute ("td[onmouseover]") and the text stored in that table cell. The sample of a result after this stage is in the Appendix 6.

In normal web browser, to get timetable for second term, one would click “Go” button, select range of weeks or dates and the timetable will update. The URL in the browser address field will not change and remain https://my.tcd.ie/urd/sits.urd/run/SIW_XTTB.start_url?<...>, the update is called by AJAX (Asynchronous JavaScript and XML) and only affects one part of the web page -
the timetable. JSoup is not capable of interacting with Java Script elements, i.e. the dialog where user enters week range or date range.

Using Dragonfly (webpage development tool), I looked at the request that was sent to update the timetable. The POST request to update the timetable is sent to the https://my.tcd.ie/urd/sits.urd/run/SIW_XTTB_1.update_timetable? Along with the week range, dates, timetable view style, academic year and three more keys. The response is a HTML table with timetable for specified dates. To do this using JSoup, I scrap the web page for the keys, then specify the parameters (academic year, week range etc.) and create the URL that updates the timetable. The result is split into relevant pieces and automatically inserted into the SQLite database. While this process lasts user sees basic loading dialog. The username and password is destroyed as soon as AsyncTask commits.
Chapter 5

Evaluation

Application
Let’s begin with the fact that all planned objective were implemented and work as intended. I tried to combine the best UI features of the systems that were analysed and bring additional improvements. I addressed most of the performance issues that I had an effect on but some parts of the app depend on third parties that I cannot change.

Even students without access to the network can use the application – the timetable can be populated manually. However, the main benefit of the app is the option to extract it from my.tcd.ie, removing the hassle of typing each lecture and related information. The extraction process, which each student, who wants to use it, will do at least twice during academic year (once for each term), takes approximately 40 seconds to complete for first term and approximately extra 15 seconds for second term. The table below gives a comparison of time it takes to extract timetable for the first term on different devices:

<table>
<thead>
<tr>
<th>Device</th>
<th>HTC One X</th>
<th>Samsung Galaxy Note II</th>
<th>Samsung Galaxy S2</th>
<th>LG Nexus 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average time</strong></td>
<td>40.3 seconds</td>
<td>39.5 seconds</td>
<td>41.2 seconds</td>
<td>38.6 seconds</td>
</tr>
</tbody>
</table>

Each test repeated three times, average time presented in the table. All tests undertaken on the same Wi-Fi network. In all cases, there were 14 lectures (six different modules, same account) extracted from my.tcd.ie. It becomes clear from the table that performance is not caped by the hardware of the device. The slight fluctuation in numbers is more of an observational error. Furthermore, comparing it with time it takes to open timetable on a computer using different web browser proves the point. Clicking “View My Timetable” (link that will open timetable on my.tcd.ie) on a computer web browser (3 tries on 3 different browsers Opera 12.14, IE10, and Chrome 26.0.1410.43 m) takes on average 36 seconds to load the timetable. This time only reflects the final step (from 4 to 5) in a
chain: 1. Opening my.tcd.ie -> 2. Login -> 3. Clicking “My Timetable” -> 4. Clicking “View My Timetable” -> 5. Timetable - whereas table above includes all steps from 1 to 5. It is safe to say that most of the processing time is connecting/opening timetable on my.tcd.ie and small fraction of that time is extracting, formatting storing the data. After all, student would not need to do it every time he or she wants to check the timetable.

According to the devices default network bandwidth-monitoring tool, single extraction of the timetable (14 lectures, same account as in above testing) uses about 24 KB of data, which is negligible, compared to average data usage of less than 2 GB (fiercewireless.com, 2012).

The feedback section has small delays when user votes. Each time a vote is given, the app makes a connection with the database, updates the vote and pulls new data from it. This allow user to see the result of his vote immediately but at the trade-off of small non-responsive time of the application. I have used Simple Adapter (a class provided by the Android) for inserting data into the view, which works perfectly fine with static data like timetable but may not be the best option for the feedback section. This adapter cannot update the information within the view, as the result it creates new instance of itself and then populates it. When user scrolls down through number of feedbacks and votes for one of them, I have to store position of the first visible feedback and focus on it after the vote cast. If only half of the first visible feedback was on the screen, the user will see it ‘jumping’ to the beginning of that feedback, as it cannot store precise position in the scroll view.

Comparing to existing solutions

The application that I have created aimed at Trinity college students but students outside of it or without the Internet access could use it. DID Timetable app is design purely for its student and has no such option. Compared to other timetable apps analysed in chapter two, my solution fully support both swiping and button type navigation. In the UI design, I have used native theme, so none of the elements of the application ‘stand out’ and should be familiar to Android users. There are no performance issues apart from relatively long web scrapping operation, which is beyond of my control.
Use cases and sequence diagrams.
For this section, I will name user Joe.

**Check Timetable**

*Assumptions:* Joe filled out timetable beforehand.

*Steps:* Joe decides to check what his next lecture is. He takes out the phone, unlocks it, clicks on the TCD Timetable app icon and looks at the timetable. Then Joe decides to check at what time he starts on Wednesday. He clicks on Wednesday Button at the top of the screen and looks at the time of the first lecture in the list.

**Sequence Diagram:** Appendix 1

**Insert Module**

*Assumption:* App is running, main screen.

*Steps:* Joe wants to add new module. He clicks on the Modules button on the top of the screen. In the Modules section, Joe clicks on the button with a plus sign. In the dialog, Joe types in Modules tittle and lecturers’ name. He also picks colour for this module. Joe clicks add button, looks at the result and presses back button.

**Sequence Diagram:** Appendix 2

**Insert Lecture**

*Assumption:* App is running, main screen. Joe added some or all of the modules in Modules section.

*Steps:* Joe clicks on the button with a plus sign on it. He selects module title from the drop-down menu. Joe types in Building and room, where lectures will be taking place. Joe selects day from the drop-down list. He clicks on the starting time button and selects time. He does the same for the finish time. Finally, Joe clicks add button and looks at the result.

**Sequence Diagram:** Appendix 3
Extract Timetable (successful case)

Assumption: App is running, main screen. Joes’ phone connected to the network.
Steps: Joe clicks on the Menu button. From the Menu he selects Extract Timetable. In a dialog that appeared, he enters his account details, puts a tick in front of the Michaelmas term and presses Login. Joe sees progress dialog. After it disappears, he sees his timetable for Michaelmas term.
Sequence Diagram: Appendix 4

Extract Timetable (not so successful case)

Assumption: App is running, main screen. Joes’ phone connected to the network.
Steps: Joe clicks on the Menu button. From the Menu, he selects Extract Timetable. In a dialog that appeared, Joe, by mistake, enters wrong account details, puts a tick in front of the Michaelmas term and presses Login. Joe sees an alert message that something went wrong, suggesting that he might have entered wrong account details.
Sequence Diagram: Appendix 4

Reading Feedbacks

Assumption: App is running, main screen. Joe filled out his timetable. There is at least one feedback in the database. Joes’ phone connected to the network.
Steps: First Year student Joe, in his spare time, decides to read feedbacks for module that he is doing. He clicks on Modules on the top of the screen and selects the one he is willing to find out about. Joe reads feedbacks that previous year students left and learns about a book in the library that explains the topic he did not understand. He also learns that non-mandatory tutorials cover exam-like question that last year students found very useful.
Sequence Diagram: Appendix 5
**Adding Feedbacks**

*Assumption:* App is running, main screen. Joe filled out his timetable. Joes’ phone connected to the network.

*Steps:* Joe, in his spare time, decides to leave a feedback for module that he just finished. He clicks on Modules on the top of the screen and selects the one he is willing to leave feedback. He clicks on Add Feedback button and picks number of topics from drop-down menu. He writes about final assignment, that it took much longer to finish than he and his classmates expected. Tutorial question that were not marked are the part of upcoming assignment. Finally, Joe clicks submit button.

*Sequence Diagram:* Appendix 5

**Voting for Feedback**

*Assumption:* App is running. Joe already entered Feedback section of the app. Joe filled out his timetable. There is at least one feedback in the database. Joes’ phone connected to the network.

*Steps:* After reading few feedbacks, Joe finds few of them particularly close to his thoughts and he was actually planning to write the same. Instead of adding similar feedback, Joe decides to vote for the existing one. Therefore, Joe presses upwards-looking arrow and increments the vote by one. Since all other feedbacks had equal votes, the one he voted for gets to the top of the list.

*Sequence Diagram:* Appendix 5
Chapter 6

Conclusion
I believe this project was successful. All objectives that were set at the beginning accomplished in full. As a student, I would definitely use this application. It lacks some of the extra features offered by other timetable apps but it does not require TCD students to input everything manually. Some of the features can be implement in future work along with improvements.
To fix short delays in the feedback section, a buffer can hold votes and insert them in bulk rather than one at the time. Right now, the app makes direct connection with the database, which raises some security issues. Someone may decompile the app (which is not hard) and find login details for it. The account used in the app has limited access. For this reason, there is no way to edit or delete feedbacks for students but it is not a solution to the problem. In future, one should create a web service to manage the connection with the database (for example a PHP script, with predefined queries). The feedbacks also raise some ethical issues. Students can report inappropriate feedbacks but who should have the rights to moderate them?
One of the apps had task in the timetable, which unfortunately did not work properly. I think it is a good add-on for timetable app. Student should be able to add personal short notes for each lecture in the timetable, with optional push-notification about their due date.
The web scrapping method is not reliable. If they decide to change the timetable on the web, the app may not be able to extract it. The best option would be if there was an API provided by the my.tcd.ie administrators but it is unlikely to happen. Due to time constrains I was not able to test the app on a wide scale, as the result it has a potential to fail for some courses if their timetable is drastically different.
I enjoyed working on this project and looking forward to improve this app in my spare time, before realising it into the app market.
References


java - Getting HtmlUnit to run under Android - Stack Overflow. 2013. java - Getting HtmlUnit to run under Android - Stack Overflow. [ONLINE] Available at:

Appendices

Appendix 1 – Sequence diagram

Checking Timetable

User → TCD Timetable → Event Handler → SQLite

- User starts the app
- TCD Timetable: Open Current Day View, Creates Tabs with Day View, Get Current Day
- Event Handler: Get Lectures for Current Day, Current Day Lectures
- SQLite: Inserts Data
- User selects Wednesday
- TCD Timetable: Show Current Day
- Event Handler: Switch to Wednesday Tab, Get Lectures for Wednesday, Wednesday Lectures
- SQLite: Inserts Data
- User shows Wednesday Tab
Appendix 2 - Sequence diagram

Add Module

User

Clicks Add Button

TCD Timetable

Clicks on Colour Select Button

Clicks on Back Button

Event Handler

Selects Desired Colour

Inserts Data

Enters New Module Details, Clicks Add

Modules

Presses Back Button

New Module Data

Changes Colour of the Button

SQLite

Start New Intent Modules

Shows Modules Section View

Inserts Data

Get Add Module Dialog

Get Modules

Creates Add Module Dialog

Select Colour

Creates Colour Picker Dialog

Show Dialog

Show Colour Picker Dialog

Colour

Show Modules Section

Close Colour Picker Dialog

Finish Modules Intent

Close Dialog

Resume Main Activity

Insert New Module

Inserts Data
Appendix 3 - Sequence diagram
Appendix 4 - Sequence diagram
Appendix 5 - Sequence diagram
Appendix 6 – Sample output of the extraction method

<table>
<thead>
<tr>
<th>PD</th>
<th>TID</th>
<th>Application</th>
<th>Tag</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
<tr>
<td>..</td>
<td>794</td>
<td>818</td>
<td>com.st</td>
<td>System.out</td>
</tr>
</tbody>
</table>

Appendix 7 – Add New Lecture dialog

Add Lecture

Module: CS4051 HUMAN FACTORS

Room/Building: [ ]

Day: Monday

From: [ ] To: [ ]

Add Lecture Cancel
Appendix 8 – Add New Module Dialog

Appendix 9 – First launch welcome screen (same as Help)
Appendix 10 – Login dialog, extraction.

Login to my.tcd.ie

Current timetable will be overwritten!

Username

Password

☐ Michaelmas Term: Weeks 5-16  ☐ Hilary Term: Weeks 21-32

Cancel  Login

Appendix 11 – Context options

Options

Edit

Delete