A Multi-Disciplinary Mobile Applications Project Course at the Graduate Level

Jonathan Rose[1], **Braiden Brousseau**[1], **and Alexandra Makos**[2]
[1]Department of Electrical and Computer Engineering, University of Toronto, Ontario, Canada
[2]Ontario Institute for Studies in Education, University of Toronto, Ontario, Canada

Abstract— *We describe five years of experience and learning with a graduate project course that brings together students from many disciplines – health, education, music, science, etc. – with graduate-level programmers from Computer Engineering and Computer Science. The goal of the course is to create a prototype of a mobile application in the field of the non-programmer. The course provides experience in cross-discplinary ideation, interaction and communication, and has resulted in a number of exciting ideas. This paper describes the structure of the course as it has evolved, with commentary on what works and does not, including assignments, group forming, and project milestones. We then describe a few of the specific project outcomes, give a general sense of the remainder, and describe issues and future directions for the course.*

Keywords: Projects, Multi-Disciplinary, Mobile, Wearable

1. Introduction

In 2007, mobile technology took a revolutionary leap with the introduction of the iPhone, and the subsequent opening of the programmability of that phone to anyone who could program a computer. Mobile and wearable technology bring together a powerful internet-connected computer with novel sensors and output devices that have enabled incredibly creative new applications in almost every field of endeavour, giving rise to perhaps one of the greatest surges of creativity in human history. The ubiquity of these smart mobile devices means that many people are inspired to conceive their own ideas for new applications, but only those skilled in the art of programming can create them. However, people without programming skills do have great expertise and ideas that could drive new applications if they were somehow enabled to make them. Conversely, the set of ideas from those in the programming disciplines is limited by their area of expertise, which is often purely technical.

Our notion, begun in 2010, was to bring together students at the graduate level in many disciplines, together with graduate-level programmers to work collaboratively to create prototype applications in the field of the non-programmer. Our assumption was that students at the graduate level in various disciplines would bring a level of expertise understanding and insights from their field, to drive new applications that would either aid them in their research, or do something novel in their general field. We also assumed that graduate-level programmers would have the skills and drive to create a high-quality prototype in the short time frame of a single-semester course.

We also saw the course as an experiment in inter-disciplinary project work that would benefit students of both types by exposing everyone to issues in inter-disciplinary communciation and engineering project work.

The results have been delightful and inspiring, rising to the level of pointing out new directions for research. There have been students from many disciplines taking the course - medicine, music, psychology, anthropology, biomedical engineering, education and many more - together with many students from computer engineering and computer science. Universities have a unique ability to create experiences such as this; one reason for writing this paper is to describe how relatively easy it is mount this kind of course, and to encourage others to do so.

This paper is organized as follows: the next section describes the two fundamental natures of students taking the course, their requirements, and the implicit bargain they make in undertaking the project. The subsequent sections describe the structure of the course - the project structure, lecture content, and assignments. We then describe several specific projects as example outcomes, and give a general account of the larger number of projects over the five years. Finally, we describe several issues and our plans for dealing with them in future iterations of the course.

2. Student Types and the Bargain

The fundamental assumption of the course is that there are two types of students: those who bring programming skills and experience (called, quite naturally, *Programmers*) and those who bring expertise and knowledge from another discipline. The latter we call *Appers*, rather than the more obvious but negatively connotated ‘non-programmers’. Early on, by the second lecture, students must declare whether they are a Programmer or an Apper, as they will follow two different paths. It is also possible for a student to declare as both, if that they possess the requirements of both.

We require that a Programmer has fairly extensive course-based and project-based software experience. Our experience has found that those with limited programming backgrounds (perhaps having taken just one or two introductory programming courses) are unable to both learn the mobile programming environment and make a meaningful contribution to the project. For this reason we ask the Programmers to describe their relevant course work and project experience as part of their first assignment. The criteria for a Programmer are:

1) Two introductory programming courses, including one that describes data structures and algorithms. The latter course should not be theory-only, but include practical programming assignments.
2) Several courses in the areas of operating systems, graphics, databases or software engineering. These kinds of courses often have intensive programming experiences in which students have to work on large amounts of code in a short amount of time - crucibles for good programming experience.
3) Project experience in which the Programmer has personally created a relatively large amount of code in a functional programming language - preferably 2000 lines or more. This experience can be in prior course projects or capstone design projects, or in internship/work experiences.

If the student's qualifications appear to be insufficient, then we have a conversation to establish a clearer understanding of their background. If the background is not strong enough, we ask the student to leave the course.

A key requirement of the Appers is that they bring expertise from a specific discipline. An easy example would be a medical doctor taking a graduate degree, who clearly brings the expertise of medical practice and perhaps even a specialty. Another example is a graduate student in an education research program who brings expertise in the understanding how people learn. There are many more good examples from graduate students in music, psychology, and civil engineering; several of these will be describe in the examples section below. We have allowed Appers in one graduate field to make a 'claim' to expertise in another field - for example, an Electrical Engineer who had spent a good part of his life composing music wanted to drive an application in the music composition field. Upon discussion, he was able to substantiate this claim with sufficient experience. On occasion, Appers arrive in the course, but are unable to make a clear claim to a specific area of expertise. Our experience has shown that these students will both have trouble attracting programming partners (an important part of the course process described below), and that if they do, that the resulting proposal and project is weak and un-directed; we now move to suggest that these students leave the course. (In both cases, asking students to leave is not a pleasant experience for them, but it is far better than the struggles we have seen in previous years when they stay).

The expertise of the Apper is an essential element of the course, as is the high-level programming capability of the Programmer(s). We view the course as a *bargain* between these two parties - the Apper brings multiple years of expertise in a field providing knowledge and insight that lay people do not have, and sometimes access to sophisticated facilites that can aid the project. An example of the latter is an Apper in rehabilitation sciences who has access to a rehabilitation measurement lab and equipment that could be used to measure the accuracy of a smartphone-based instrument. The Programmers give, in return, their labour, programming skill and technical insights as to what is possible, feasible and achievable.

This gives rise to one of the key rules of the project: the application must be within the field of the Apper, and not something else for which they have an idea, *but no expertise*. This rule is rather important, as some number of Apper candidates arrive in the course with ideas for applications quite outside of their field – perhaps wanting to make a game or some kind of social networking app unrelated to their field. While these might be good ideas, without the backing expertise there is less likliehood of novelty and authoritative insights.

3. Project Structure

The central goal of the course is to form an inter-disciplinary team and to create a prototype of a novel mobile application in the field of the Apper. The timeline of the 14 week course is given in Table 1; this section will describe the core components of each activity.

Week(s)	Project Activity
1-2	Introduction & Team forming
3	Idea Forming
4	Approval-in-Principle
5	Written Proposal/Plan Due
6	Proposal/Plan Presentations (extra lecture)
7	Reading Week - no class, work though!
8	User Experience Lecture
9-10	Spiral 2 Presentation & Demo
11-12	Spiral 4 Presentation & Demo
13-14	Final Presentation & Demo

Table 1: Project Timeline

3.1 Team Forming and Vetting of Students

Perhaps the most difficult and chaotic part of the course is the process of forming teams. After some experimentation, we have landed on specifying that a team should consist of one Apper and two Programmers. It makes sense to have one domain expert in a project; on occasion a sginle student has served as both Apper and Programmer. We have found that the typical amount of programming work is extensive and likely too much for one person, and hence the two Programmers. Two programmers also serve to help each other learn any new material in the event that one gets stuck. The scope of the projects involves such a large range of the computer engineering stack that multiple programmers can complement each other's specific expertise in the various areas such as web/server, UI, optimization, and computer vision. On the other hand group dynamics become more difficult with three Programmers, and so we allow this only under exceptional circumstances (typically if another group disintegrates and a Programmer is looking for a home, we will assign that person to a group with particularly ambitious programming needs).

One explicit rule of the course is that a student can only stay in the course if they successfully become part of a team. This requires a 2:1 ratio of Programmers to Appers, which can only be in-directly enforced by the rule; however our experience has been that this ratio has always roughly been present, even at the beginning of the course. Previous research suggests that allowing students to choose their own teams results in choices that are based on a mixture of predictions of success (i.e. choosing people who will help them succeed) and finding partners who are self-similar [1], the latter limiting the former. There is one extra constraint in our projects that influences us to allow student-led group selection: we want everyone to be excited by the topic of the project they are working on, and so want the team forming to include this freedom of choice. So, a Programmer who is keen on doing a sports-oriented application can freely gravitate towards an Apper in Kiniesology, rather than someone in the Education program.

For students to find out about potential partners, the very first assignment for both Programmers and Appers requires them to both describe themselves in writing, and to make a video - both of which are posted on a course-accessible website. In both, the Programmers are asked to describe their background, their software project experience, and what kind of project they are interested in. This is used by the instructors and TAs to ensure that their software capability is sufficient to withstand the stress of the course, as their part of the 'bargain' described in Section 2. It is also used by the Appers to evaluate the capabilities and communication skills of potential partners.

In the document and video, the Appers are asked to define their area of expertise, and to float ideas of what they have been thinking about as potential applications. We assume that every Apper has already pondered this question, as that is the reason they were taking the course! The instructor and TAs use this information to determine if there is a clear definition of expertise (which is the Apper's side of the bargain).

Since a large part of the course consists of communication that is both written and verbal, we feel that having students displays their communication capability early on is important.

In the 2015 year we took a new approach to forming teams, encouraging Programmers to find a compatible programming partner first (who they'd like to work with, and who has similar areas of interest when seeking a topic and Apper). After forming a pair, the team would interact with Appers to find their third partner. We provided an extra week for this pairing step to take place.

In previous years we have finalized the team-forming process by having a separate class get-together, in the evening to close on finding a team. At this event we have the Appers give one minute summaries of their area/ideas, verbally, and then have the Programmers informally talk to potential partners. In the current year with the extra time available, many of the groups were formed prior to this point in time, and so the extra time was used to have the un-attached Appers make 3-minute pitches to the un-attached Programmers. This latter process was possible as there were only a 5 un-attached Appers at that point, and three pairs of Programmers. This did mean that two Appers were unable to continue in the course.

In addition to the viewing of the written documents and video (which are due within the first week of the course), several other opportunities are given for the students to get to know each other. The last parts of the first and second lectures, are used for the Appers and Programmers to introduce themselves and talk about their background and ideas. We use these very informal statements as opportunities for class-wide discussion of ideas, leading into the idea creation phase, next.

3.2 Idea Creation and Approval-in-Principle

As the teams are forming, and once they are formed, there is much informal discussion of project topics. As discussed above, the topic must be in the field of the Apper. As this is a graduate course, the scope can include mobile applications that might help the Apper in their graduate-level research (for example help collect data among patients in a novel way) *or* be something that would augment their field more generally (such as a teaching aid in education). The Apps may well be something that could be commercialized, but this possibility of a research/field mandate sets the scope to be super set of those applications that are of commercial value.

Teams are encouraged to propose ideas via email to the instructor and TAs to get rapid feedback. The projects proposed must be of sufficient technical depth; this is sometimes a function of the Programmers' capabilities. This rule prevents simple information-based apps that would be the mobile equivalent of a document. There must be something that has some learning/challenge/effort at the graduate level of the Programmers. The instructor reserves the right of approval on this basis, and it is done informally via email. Some of the 'ideation' is driven by the content of the first four lectures, which describe the capabilities of mobile and wearable devices, and give examples using those capabilities, as described below in Section 4. The key milestone is to received 'approval-in-principle' on the topic and rough scope, and is due in week four of the course.

At this point (and the next) in the process, it is crucial to spot and correct especially fuzzy thinking - it is not uncommon for there to be an unfocused idea that is too broad and lacking in clarity. If left uncorrected, the final project has typically been poor. Sometimes, the act of enforcing focus reveals an inability to do so, and it is better for the student to leave the course at this point. Other times students given clear feedback on the need for focus and clarity return with far superior ideas-in-principle.

3.3 Proposal/Plan

Given approval-in-principle, the team is now asked to write a short proposal and plan of app structure/work. A written proposal/plan is due in week five, and is presented in week six.

The proposal reiterates what the project is, and its motivation. The plan gives a rough design of the overall structure of the work, and a set of milestones to be achieved of the ensuing six weeks. For the structure, we encourage students to present a block diagram of the entire functionality of the App and any connected hardware or software. The Appers are typically unfamiliar with the notion of a high-level block diagram; it is also surprising that a subset of the Programmers are also unfamiliar. Hence we review the concept, and illustrate it with a discussion on what would be involved in a hypothetical application's block diagram.

One of the most exciting parts of the course comes in the ideation phase: because many Apps are connected to us as humans (being portable), we can all fairly easily imagine ourselves using a proposed application, and to ponder whether it will work, and what else it might do. We call this process 'living within' the proposed Application. That is, we mentally place ourselves within the context of using the App, and see if there are new and related ideas that could bring more capabilty/functionality. To teach this notion, we have the class as a unit *live* in a particular example; an App that measures the ability of a person to balance, as a metric of sobriety. We then ask how this functionality might be used - by police, bartenders, spouses, etc. and how that might work. Over the years of the course, some wonderful inspirations have resulted from this process, some of which are described in Section 6.

The proposal presentation is limited to six minutes - every presentation in the course has a similarly short time limit; not only to allow the acommodation of up to 20 projects, but to ensure brevity and clarity. The class as a whole along with the TAs and instructors respond by providing feedback and asking questions. We also provide written feedback along with the grade for the proposal/plan. This point in time is also key for providing projects with a lack of focus or incorrect scope the needed guidance.

For the plan, we advocate making use of the general engineering method known as the *agile* or *incremental* or *spiral* method: *the idea is to make the simplest evocative prototype as soon as possible, and then iterate on it, adding improvements.* The weeks from the proposal/plan week are numbered starting from week 0 in the plan. We suggest selecting clear target of functionality for a prototype be ready as soon as week 1 is over, and call that the 'Spiral 1' target. Subsequent weeks are Spiral 2, 3 and so on. As such, the plan isn't required to have much detail as the subsequent goals are more readily identified as time progresses, as per the spiral/agile method [2].

An important part of the proposal presentation is from the Apper, who typically gives the introduction and motivation sections; we also ask the Apper to describe what their role will be in the project execution, and what else they will bring. Two examples of this are the use of laboratory facilities that can help in measurements the project/App makes, or the application of their expertise in music analysis.

We believe that one of the key reasons for the success of the projects in this course is the requirement that students present, in the class, their progress and a demonstration of the partially-working App, as of the Spiral 2 and Spiral 4 deadlines, prior to the final presentation and demo. This forces work to happen at regular pace - hard deadlines and a public presentation are very focusing!

Since it takes roughly four hours to do the proposal/plans for 15-20 projects, and our lecture is only 2 hours/week, we have added in an extra lecture for the proposal week to launch everyone at the same time.

Finally, at the proposal stage, we ask the team to give a name to their project/app. This name becomes the label by which we refer to the project, and very quickly becomes the identity of the team.

3.4 Execution and Demonstrations

The final six weeks of the course are devoted to the work to build the prototype, and the series of interim and then final presentations/demos. The actual work goes on outside of class, and the students must meet regularly to make that happen. As we typically have on the order of 15-20 projects, it takes one entire lecture to get through ten of these presentations, including time for feedback and questions. The Spiral 2 and 4 presentations are required to be 5 minutes long, and are similar in structure: a quick reprise of goal, a description of the progress made, a demonstration, and then a plan for what the next demo will bring. The class and instructors respond with questions, feedback and suggestions. These cover everthing from helping to solve technical challenges (or warning of some to come), re-directing the goals in the face of new issues or exciting discoveries, as well as feedback and suggestions on the quality of the presentation itself. The presentation is graded and given additional written feedback.

The final presentation and demonstration happens in the last two weeks. The students are given a longer presentation time (8 minutes), and asked to make a presentation that is self-contained. You can view the videos of many of the final presentations from 2013 [3], 2014 [4] and 2015 [5].

4. Lecture Content

The lecture content is confined largely to the first five lectures. Some of the content is the project organizational structure, described above. The key material is a description of the capabilities of mobile technology, and examples of their use in applications. These applications include prior projects in the course, and interesting and innovative applications from the literature and commercial world. The series of lectures can be found by going to the link in this reference [6], and clicking on the 'Content' link.

The stunning capabilities of mobile technology arise from the miniaturization of integrated circuit and sensor technology - the one portable device contains a powerful computer, a high-speed connection to the internet, a high-resolution screen, high-quality sound output, and a series of cheap but accurate sensors: the accelerometer, gyroscope, magnetometer, microphone, light sensor, proximity sensor, GPS and barometer. Remarkably, the sensors are sampled at rates ranging from 10 to 100 samples/second, providing very frequent measurements of acceleration, angular motion, light, magnetic fields, air pressure and more. The processor glues all of these sensors and outputs together with all of the algorithms ever developed in computing. Generally speaking, most people do not know about all of these different sensors; they may have used them inadvertently without even knowing that they exist. The details and example uses of these sensors, described in detail in Lectures 1, 2 and 3 are a key source of ideation.

There are also some extraordinary examples of the use of these sensors - for example, the digital signal processing algorithms developed in [7] are able to infer a human's heart rate by analyzing video of a person's face – the invisible-to-the-human-eye change in colour with each heartbeat can be detected with under 20 seconds of video. This *super-human* capability is inspiring, and we suspect there are many of these that can be used in future-looking applications.

More recently there has been a surge of smaller wearable devices that are wirelessly connected to the smartphones. Thes have sensors in smaller, more portable/wearable packages. One low-cost example is the $29 'Sensor Tag' from Texas Instruments [8], which is a bluetooth-connected set of buttons, accelerometer, gyroscope, magnetometer, humidity sensor, ambient temperature sensor, and directional temperature sensor. Another example is the 'Node' sensor from Variable Inc., which includes the accelerometer, gyroscope and magnetometers in the base unit, and can add two other sensors either end - temperature, weather, colour, gas and many others [9]. Even smaller are the bluetooth-connected trackers that can indicate the presence or absence of anything they are attached to – for example the TrackR device [10].

5. Assignments

The first month of the course is taken up with two key tasks: the Programmers and Appers need to come up to speed on the mobile programming environment, and, as described above, form teams and generate ideas for projects. In the following sections we describe the assignments, which can be viewed under the Assignments link of [6].

5.1 Programmer Assignments

For the Programmers, this part of the course is like many other programming courses in which they learn about a new kind of programming paradigm/environment and infrastructure by doing assignments that take them from the simple basic capabilities to the complex. The challenging part is that they have to do this in one month, rather than 3 months for a typical undergraduate course. We rely on the fact that these Programmers are at the graduate level, and have experience and sophistication and so are able to pick up the material more quickly. The four assignments described below cover the basics that almost every App in the course will need:

1. Development Environment & Simple Widgets. This is the basic environment set-up, creating the first simplest program ('hello world') and making a slightly more complicated App that receives user input, and produces a few outputs. At this point the Programmers have to have chosen which environment they will use for the course - typically Android, but some choose iOS. This choice is important, as they will have to find a programming project partner who wants to work in the same environment. For learning resources we use both the online Android documentation and one of two book series [11] [12]. All the assignments give sections of the texts to read.

2. Containers, Fragments, Select, Lists and Files. This assignment teaches the basic methods to display lists of items - a very common attribute of many Apps - and how to store information and retrieve it from files. It also teaches how users provide various kinds of input,and conveys the basics of the view hierarchy of both Android and iOS. Students who are unfamiliar with event-driven systems will be exposed to various aspects of that concept.

3. Location, Motion Sensors and Image Capture. This assignment gives exposure to several of the key capabilities of modern mobile devices - determining the geopgraphic location of the phone, measuring its motion through the use of the accelerometer, and capturing images from the camera. Many applications make use of one or more sensors.

4. Threads, Databases and Network Connections The final assignment deals with creating a local SQL database, and spawning threads to perform tasks independent of the basic User Interface thread. These are important as many apps require databases, and often perform compute or network tasks that would slow down the user interface if they were not spawned as a separate thread.

5.2 Apper Assignments

The goals of the assignments for the Appers is to expose them to the capabilities of mobile technology, to explore what has already been done in mobile in their field, to give them experience in the user interface design, to be exposed to language and concepts of one essential capability/discipline of computer engineering, and to practice being creative in the mixed milieu of technology and their field. It is worth noting that in the first version of the course in 2011, we had the Appers try to learn a simple form of programming, based on a visual programming framework called Google App inventor that itself was based on the MIT Scratch environment [13]. We found that the Appers with no programming background were easily able to do very simple apps with a few buttons that cause actions. However, once the concepts of variables and loops were in play, they quickly became lost. In subsequent years we focused the Appers towards understanding capabilities of mobile devices, understanding computer concepts and creativity exercises. The four current assignments for Appers are:

1. Connecting Your Field to the Mobile Devices Field. Here the Appers are asked to survey the landscape of mobile applications in their own field - they are to look at descriptions of five different applications (available on an App Store or described in a paper) in their field and write a 100 word summary of it. Then, they are to acquire one of these Apps (one that they can run on their own mobile device) and to write a more extensive review of it, including its motivation, why it is interesting, how it works and suggestions on how to improve it. This is important so that the student can get a sense of the prior art, and when it comes to inventing their own ideas, not replicate what has been done.

2. App Design Principles, Mockingbird & Practice. One part of the project that all Appers can participate in actively, is the user interface look and feel. There exist a number of different free, online tools for both drawing and linking different screens of an App [14][15] [16] These tools are quite easy for anyone to learn, as they are similar to regular picture-drawing programs. The assigment asks the Appers to first read about some basic principles of user interface design (provided by Apple [17] and Android [18]) and to practice building simple designs using one of the above tools. Then, they are asked to create the complete user interface design and flow/links between screens for a specific application. At the same time, to get the 'creative' juices flowing, the assignment asks them to invent a new application based on a specific novel capability - in this case the ability to listen to a conversation among multiple people and determine what fraction of the conversation each person takes up.

3. Understanding Parts of the Canvas. The third assignment makes use of the fact that the Apper will have formed a team with two other Programmers at this point. The assignment gives a set of basic technologies/capbilities present in all modern networked computers – search, databases, signal processing, optimization, and internet communication – and asks the Apper to choose one of these. They must then spend time with their programming partners to learn from them the basics of this capability, and then to explain what they've learned in their own words. After that they are asked to do their own independent research to augment what they've learned and to write that up at a later time. The goal here is to have the Apper cross over into the technical realm to a certain depth - to understand *what* is being achieved in that realm, but not really *how* it is done, as that is typically too complicated. It also sets up a pathway of dialogue between the partners in the team. Our expectation that dialogue in the other direction (from Apper's field to the Programmers) happens naturally as part of the project.

4. Creativity, Sensors and You. The goal of this assignment is to stimulate the creation of ideas, connecting to some of the early lectures on capabilties of smartphones. The Appers are asked to invent ideas for interesting Apps in their field that make use of some of the smartphone sensors - the accelerometer, gyroscope, barometer, camera, light sensor, proximity sensor, humidity sensor, etc. It also asks them to consider the amount of processing that might be needed in their idea - for example, if they use a single picture, they'd need to count the number of pixels to be processed in an image, or much more for a video. The second part of the assignment suggests several new sensors that may appear in the future (a gesture sensor, an ultrasound imager, an emotion sensor, a blood pressure sensor, and a brain activity sensor) and asks the student to come up with ideas in their field making use of these.

6. Outcomes

The course has been running for five years, with the fifth year in flight as this paper is written. In total, including this year, there will have been 95 completed projects with 76 Appers and 187 Programmers. The projects have been in many of the areas that the University covers in its graduate program, including Aerospace, Anthropology, Biomedical Engineering, Drama, Education, General Medicine, Industrial Engineering, Library Science, Music, Museum Studies, Nusring, Pharmacy, Physiotherapy, Psychology, Rehabilitation Science, and Surgery. It has been thrilling to watch the teams come together and create novel and interesting applications in these fields. It is clear that, in the best projects, there is a great deal of inter-disciplinary learning as the Programmers become familiar with the basic concepts and language of their Apper's discipline. For many of the Appers it is their first experience in a serious engineering project, and their exposure to the concepts of agile/spiral development, and the complexity of software has been a formative experience for them. The experience in the course will enable them to more easily collaborate with engineering development in the future, and possibly make them more likely to realize their ideas in collaboration with an engineering team. It is also clear, from the increase in quality of the presentations over the four opportunities in the course, that the strong emphasis on communication engenders significant improvements in many of the student's ability to communicate.

To illuminate some of the outcomes, we describe three of the projects in more detail below, and provide a table summary of several more after that.

6.1 iAnkle

The goal of the iAnkle project [19] was to help a person with an injured ankle (either a break or sprain) recover. This is done by both measuring the stability of the ankle and prescribing exercises to progressively improve the stability. The Apper is a licensed Physiotherapist who was completing a Master's degree in public health. The basic idea, conceived by the Apper, was to use the accelerometer in a phone (tucked in a sock) to measure how much the patient would 'wobble' when doing various kinds of balance exercises, such as standing on one foot. The prototype both quantified the wobble through measurement, and attempted to prescribe successively harder exercises as progress was made. The measurement with the phone's accelerometer was a proxy for far more expensive force plates that are used in experimental physiotherapists labs, and the Apper, with the help of the Programmers, was able to bring that to a low-cost prototype that has the potential to be used by anyone with a smartphone. This very exciting application continues as a research project, with the Apper heavily involved.

6.2 Baton

The Apper in the Baton project [20] is an experienced high school teacher who was taking a Master's degree in Education. He sought to find a better pathway of communication between a teacher and a class of students during a class discussion. Rather than simply putting up one's hand to make a contribution, the students were to use a smartphone application to indicate both an interest in communicating, and also the nature of the contribution - either building on a previous comment, contradicting it, or moving to a different topic. The teacher's receiving application (connected through a server) would be notified of who was involved, and how long they had been waiting to contribute. If you watch the video available under the link [20] you'll see the Apper and Programmers give an eloquent demonstration of the concept.

6.3 Mindful Me

The Apper in the Mindful Me project [21] was a Ph.D. student in Psychology, with a focus on the treatment of addiction (to alcohol, drugs, tobacco, etc.). A standard way that therapists ask a patient to help themselves is to ask them to write into a journal a description of their cravings – when they happen and under what circumstances. This is used in a process of reflection and understanding, part of the process of resisting a relapse. The Apper's idea was to make this journal into a smartphone application, to gain ease of use and more privacy (as no-one would ask someone typing on a smartphone what they are doing, whereas they might ask about writing into a journal). A phone could also more easily record time, and location (using the GPS) and possibly other things about the state of the patient. During the inital proposal and spiral 2 presentations a very interesting thing happened - it became clear that a phone could not only act as a recorder of events, it could become an actor that actually influenced events. Consider an alcoholic who appears to be moving towards his favourite bar (as monitored by the GPS) or a drug addict heading towards the place where she purchases her favourite drug. A phone could notice this and try to intervene! This raised the very interesting question of what such an intervention might be – such as the phone playing a song, speaking a mindfulness text, or calling a friend or sponsor. This is one example of how 'living in' an App brought forth an exciting idea.

6.4 Summary of Selected Projects

Table 2 gives a short list of several other projects - their name, year, and very short description. Longer descriptions of each of these (a final written report and a video presentation) can be found by looking under each year's website archive, which can all be found at the bottom of the page of reference [6].

Year	Name	Description
2011	BrainEX	Brain exercise to combat dementia
2011	Wound Capture	Record wound assessment and treatment
2011	Whimper	World noise mapping project
2012	EYEdentify	Game helps autistic children learn emotions
2012	DriveMod	Monitor and measure car driving quality
2012	SurgicalBlackBox	Surgery video & data review/annotation
2013	Mobile Stage	Augmented reality interactive theatre
2013	LunchTime	Helping children learn to tell time
2013	SnapNDose	Proper dosing of children's medicine
2013	NewCanuck	Helping immigrants learn local culture
2014	Speech Coach	Helping speakers give better talks
2014	Surgical Trainer	Measuring surgeon's movements
2014	Critter	Social Connections with Virtual Pets
2014	MyAlly	Helping stressed/suicidal teenager's
2015	flapCheck	Monitoring recovery from plastic surgery
2015	Peptiblocks	Game to find good protein folding
2015	PUPL	Pupil light reaction measurement

Table 2: Selected Projects

7. Issues and Future Offerings

We have been very pleased and excited by the various outcomes of the course, but there are a number of issues with this kind of course that we list below, together with some thoughts as to how we will revise the course to deal with them.

One key isssue relates to the Appers in the course – they are required to do a fair bit of work at the beginning in terms of the assignments described in Section 5, and the ideation and proposal phases of the project. However, after that, the amount of work and contribution by the Apper varies widely. In the best cases, the Appers engage fully, doing the UI design, active testing and feedback of various versions of the software, comparison with laboratory measurements, and sometimes even some programming. In other cases, there is less active involvement and the Programmers have largely taken over even some of the specification work. It is difficult to prevent the latter, but it can be mitigated by careful vetting of the Appers early on in the course, to ensure that they have sufficient expertise and capability to contribute; we have noticed a correlation between the expertise level of the Appers and their ability to contribute.

A second issue is the quality of the feedback that we provide the students. While there is quite a bit of feedback given verbally after each presentation when urgently needed, and some written feedback, we feel that students could benefit from more private and individual feedback and guidance. This arises because of the very un-directed nature and broad nature of the work in the project – for some students, this is one of their very few experiences of this kind (rather real-world) work. In a sense, each project is like a miniature master's thesis, and there is room for far more constructive feedback. With the scale of students in the course it is difficult to provide this level of feedback. We are planning to re-think methods of providing better feedbak.

A third issue is recruiting of students. The University of Toronto is one of the largest campuses in North America, with 65,000 students total and 16,000 graduate students, in many different departments. In the past we have 'marketed' the course by speaking with Associate Chairs of Graduate Studies in many departments and faculties, and conveying email advertisements. We have begun to feel that there are many more highly qualified expert Appers who would be interested in the course, who have not heard about it. Our plan going forward is to offer seminars in different departments to describe the course and outcomes related to that field.

8. Conclusions

We have described a novel graduate course that brings together students from many disciplines to prototype mobile and wearable applications in those different fields - working collaboratively with graduate-level Programmers. The outcome has been a very exciting experience for both the students, the instructor and the teaching assistants, with many novel ideas explored. We have also found that a great deal of integrative learning takes place, along with interdiscplinary thinking. There is also good experience in communication across disciplines. This paper (and associated web sites) describe the nature and structure of the course, which could easily be taught in any graduate-level University with programming and other disciplines. Our hope is that this document could enable it to happen many times over!

9. Acknowledgements

The authors would like to thank the many students who took this course and worked hard to get a great result. We are also grateful to Google and Huawei for donations of smartphones, and Arshia Tabrizi, esq., for a donation to support the course.

References

[1] P. Hinds, K. Carley, D. Krackhardt and D. Wholey, "Choosing Work Group Members: Balancing Similarity, Competence, and Familiarity," Journal of Organizational Behavior and Human Decision Processes", Vol. 81, No. 2, pp. 226–251, 2000.

[2] B. Nejmeh and D. Weaver, "Leveraging scrum principles in collaborative, inter-disciplinary service-learning project courses," IEEE Frontiers in Education Conference (FIE), pp.1-6, Oct. 2014.

[3] (2013) ECE 1778 Final Presentations 2013. [Online]. Available: http://uoft.me/1778-2013

[4] (2014) ECE 1778 Final Presentations 2014. [Online]. Available: http://uoft.me/1778-2014

[5] (2014) ECE 1778 Final Presentations 2015. [Online]. Available: http://uoft.me/1778-2015

[6] (2015) ECE 1778 Course Website 2015. [Online]. Available: http://uoft.me/1778-2015front

[7] M. Poh, D. McDuff, and R. Picard, "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation," Opt. Express 18, 10762-10774 (2010).

[8] TI Sensor Tag [Online]. Available: http://www.ti.com/sensortag

[9] Variable Node Device [Online]. Available: http://variableinc.com/products/

[10] TrackR Device [Online]. Available: https://www.thetrackr.com

[11] M. Murphy, "The Busy Coder's Guide to Android Development," [Online]. Available: https://commonsware.com/Android/

[12] J. Nutting, F. Olsson, D. Mark and J. LaMarche, "Beginning iOS 7 Development, Exploring the iOS SDK," Apress 2014.

[13] MIT Scratch [Online]. Available: https://scratch.mit.edu

[14] Go Mockingbird UI design tool [Online]. Available: https://gomockingbird.com

[15] Moqups UI design tool [Online]. Available: https://moqups.com

[16] Pencil UI design tool [Online]. Available: http://pencil.evolus.vn/Stencils-Templates.html

[17] Designing for iOS [Online]. Available: http://tinyurl.com/pxxh66k

[18] App Structure [Online]. Available: http://developer.android.com/design/patterns/app-structure.html

[19] N. Shah, L. Carvalho, and I. So, iAnkle video and report [Online]. Available: http://uoft.me/iAnkle

[20] Z. Teitel, V. Chen, F. Zhao, Baton video and report [Online]. Available: http://uoft.me/Baton

[21] E. Guy, S. Puri, and S. Chen, Mindful Me video and report [Online]. Available: http://uoft.me/MindfulMe