MANCHESTER 1824
The University of Manchester

Mark It: A Digital Solution for Laboratory Marking

Third Year Individual Project – Final Report

April 2019

Enrique Tasa Sanchis

9673164

Supervisor:

Dr Peter N. Green

Contents

Total word count: 10,139

This is a software project. There are plenty references to code in this report, but for clarity's sake, when possible, the functioning of the code has been explained rather than explicitly shown in the body of the report. Annexes O and P at the back of the report include all project code with explanatory comments. A showcase of the application's function can be found in Annex A.

Abstract

A key component of any engineering student's university journey is attendance to laboratory sessions, practical exercises designed to immerse the student in a hands-on task that requires the application of the theoretical knowledge acquired in lectures. In many cases these exercises are assessed, often not only by lecturers but also by university staff or PhD students acting as demonstrators, using paper mark schemes. These documents are a source of inconvenience – they are often misplaced or lost, handwriting on them is not always perfectly legible or understandable and the student is unable to receive feedback on their achieved score given as their result is inputted into a computer and their mark scheme archived. This project consists of the development of a software tool to replace these mark schemes, as well as a complementary online cloud-based backend to allow for its deployment in real-life situations within universities and other education institutions, eliminating errors in the marking process and allowing students to consult a breakdown of their marks and potential comments written by demonstrators. The use of this tool will make the entire process of marking student's laboratory work more streamlined and reliable for demonstrators and clearer and more transparent for students.

Glossary

CSV	Comma Separated Values
UI	User Interface
UX	User Experience
SDK	Software Development Kit
API	Application Programming Interface
GUI	Graphical User Interface
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
Key-Value pair	Data structure similar to an array where a value is accessed by not by position within a structure but by association with a given codeword (key)
Life-cycle	Refers to the programmatical stages in the life of an instance within a program. In reference to an activity, a life cycle defines creation, GUI layout, data loading, user interaction, data saving and destruction.
Kernel	Core operating system of a computer or computing device. Handles low-level applications, peripherals and events.
Cloud	Within this document, refers to any online-server based instance of a computing resource. This might be a database, a server or any other kind of internet-connected resource.

1. Introduction

Evaluating undergraduates' work is a key aspect in the life of academics – be it professors, lecturers, researches or PhD students[1]. It is also a crucial element in a student's academic life as it ultimately determines their grade for a certain unit, year or degree. In recent years there has been a push to make this process more open to the people involved: more specifically, putting pressure on lecturers to be more transparent about the specifics of their mark schemes and the grades awarded to students. Because of this ubiquity in academic life, as well as its importance as a concept, marking is a process that needs to be regularly reviewed and improved in order to guarantee its accuracy and fairness.

This project originally aimed to provide an improvement to this process through a digital implementation of the currently paper-based marking system, focusing on taking an existing application developed in previous years and expanding it to become a realistically more usable solution. The aim of the project remained similar throughout the year, but a shift in focus was necessary when it was realised that the application that had been developed in previous years was not suitable for rework and rather had to be rewritten from the ground up. The project changed from an addition to what had been developed previously to the creation of a completely new system with the inspiration of what had been done in the past, without using any of the previously developed codebase.

The use of a digital solution rather than a paper-based one has many benefits ranging from the environmental to the practical, but it is the latter that especially concerns this project: paper-based mark schemes have often been a source of problems for university staff. Illegible handwriting, transcription errors and the inability to record and store concrete feedback for students have all caused issues in the past, with students often disputing their awarded grade with little or no tools for lecturers to determine whether the claim was justified – paper mark schemes did not include space for comments and were not archived for later consult due to time and space constraints. A digital solution eliminates all transcription problems while allowing examiners to leave specific feedback for students and providing a simple and cost-effective way of archiving all information relating to the laboratories for later consultation.

This idea was taken up by George Wellard during the 2017 – 2018 academic year, with Dr Peter N. Green as his supervisor in the context of a 3rd year project[1]. The result of this project was *MarkIt*, a prototype Android application that was able to showcase the

[1] Herein referred to collectively as university staff

potential of a reimagined marking system and provide a demonstration of how that system could work for a number of pre-loaded mark schemes. The 2018 – 2019 continuation of the project focused creating a solution that would incorporate the features needed to make it a product that could be used during the laboratories as a substitute for the old marking system – concurrent use by various examiners, cloud data store and further application functionality.

1.1. Project Aim

The primary aim of this project is to provide a fully working solution to build on the concepts developed previously on *MarkIt.* This project aims to take a prototype application and develop the code and the IT infrastructure around it to transform it into a real-world usable tool. This entails development of new infrastructure like a cloud backend with capabilities for data storage but also the rewriting of the user facing application in order to adapt it to a client-server model with the cloud-stored data. Furthermore, the project aims to provide a way for users to load mark scheme data into the cloud from a PC in order to configure the mark schemes that the app will handle.

1.2. Project Objectives

There are two sets of concrete objectives that were defined for this project: base objectives, goals that are the basis for the project to be deemed as completed; and stretch objectives, goals that would take *MarkIt* further in providing a complete solution but that will only be attempted once the base objectives are completed.

The objectives presented in this report differ from the ones that were originally proposed for this project. This is due to the change in project philosophy and content halfway through the project timeframe – when it was realised that a completely new system had to be developed, it became obvious that objectives relating to the previously developed codebase had to be scrapped and new objectives setting goals for the new set of software had to be drawn up. Nevertheless, this situation was accounted for, and the objectives hereby presented are realistic and deviate as little as possible from what was originally intended. The original objectives can be found in annex B.

1.2.1. Base Objectives

- Implement a cloud-based storage system for the data collected by the application

- Ensure that various users can access the data simultaneously, reading/writing at the same time
- Implement a system to allow users to save mark schemes in the cloud and use them repeatedly
- Implement a feature to allow users to consult student's grades for the current laboratory in real time
- Provide a way for lecturers or unit leaders to download grade data in a format compatible with BlackBoard uploads

1.2.2. Stretch Objectives

- Implement a feature to allow users to consult student's grades for all previous laboratories
- Allow users to define mark scheme structures in a desktop application and save them in the cloud
- Allow users to define mark scheme structures in the mobile app and save them in the cloud

2. Background

The following section composes a collection of contextual information necessary for the understanding of this project, with sections about the academic framework that *MarkIt* attempts to tackle as well as technical context about the tools and infrastructure with which *MarkIt* has been built.

2.1. Microcontroller Engineering II

In order to develop an application like *MarkIt,* it was important to find a real-world example of a unit that ran laboratory sessions to examine where the shortfalls of the marking process were and what could be done specifically to better that process. The unit which served as this framework for this project was Microcontroller Engineering II (ME2), EEEN20019; a unit taught by Dr Peter N Green since 2004. ME2 runs two laboratory sessions for each student each semester. It is important to understand the laboratory procedure and marking process in order to understand the development of this project.

In ME2 laboratory sessions, students are tasked with programming C/C++ tasks and running them on one of the school's supplied microcontroller boards[2] – students must illuminate a certain LED, make an ADC work or take a reading of room temperature with a sensor and then display it in a 7-segment display. These sessions have run without issue, yet it was obvious to the university staff that the process of marking them could be improved upon, not to solve any flagrant problem, but to make the experience of marking and being marked better.

The marking process involves the student being evaluated and a member of university staff evaluating their work. Students sit at their PCs and read from a laboratory script[3] what the task they are required to do is, what the parts and sections of the task are and how many marks correspond to each part. Students are left to work on their solution and raise their hand to request an evaluation when they complete a certain part (*e.g.*, in a lab were a student is to take a temperature reading with an onboard sensor and display it as a decimal value on a 7-segment display, the student will be marked separately for setting up the sensor and display, reading and storing a value from the sensor and displaying that value on the display). The evaluation process itself consists of the member of staff observing the work of the student and recording the mark achieved in a printed paper mark scheme. The paper marks schemes are then transcribed into a computer were the student

[2] Further information about the boards can be found in annex C
[3] Further information regarding the laboratory contents can be found in annex D

mark is recorded against their student ID [4]. These grades are then uploaded to BlackBoard, and eventually, students get notified of their achieved grade.

Hence, this project was to be based around the workflow and mark schemes of ME2 and attempt to provide a solution that was as adaptable as possible – in order to provide the tool for as many units as possible – but focuses on covering the needs of ME2 specifically.

2.2. Mark It – 2017/18 edition

MarkIt [5]was originally developed by George Wellard during the 2017 - 2018 academic year, under the supervision of Dr Peter N Green [1]. The application was developed in the scope of a 3rd year project and further work was done during the summer in developing user guides and documentation. The purpose of the application was to eliminate the need for paper marking sheets during laboratories for the Microcontroller Engineering II subject by making the entire marking process digital through an Android application.

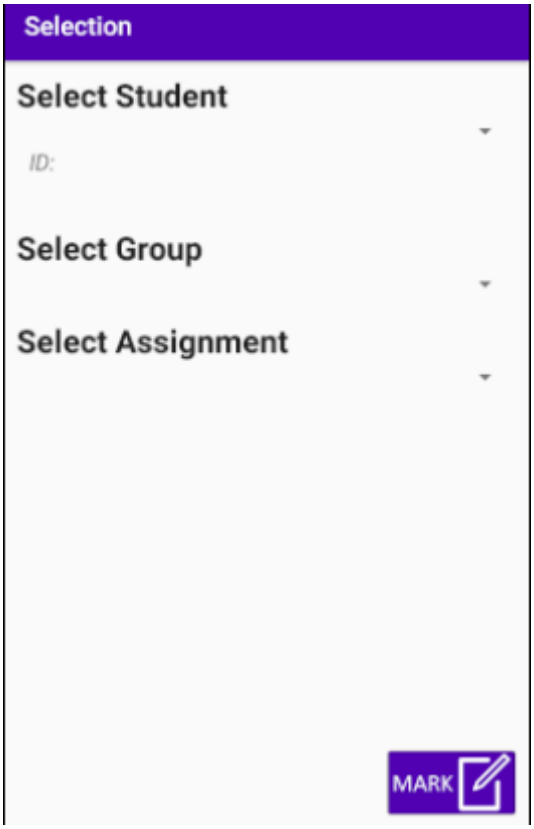


Figure 1 - A screenshot of MarkIt 2017/18

In order to do this, a solution was developed that included the following features:

- Creation of a mark scheme via a Windows desktop app (written in C#), which would then be loaded into the application to provide a mark scheme
- The input of an excel sheet containing the names and student IDs of students to be marked
- An Android app with a user interface allowing for marking of student's work, with features such as:
    - Selection page to determine which student was being marked and what assignment was being completed
    - Marking page which showed the tasks to be completed by the student and allowed the marker to select the correctly fulfilled ones
- Option to add comments to students marks as feedback
- The output of grade data as a CSV file to then be uploaded to BlackBoard

[4] A sample mark scheme can be found in annex E
[5] The user manuals developed for MarkIt 2017/18 can be found in annex F

The project was successful in developing an application that could perform the task of marking students' work against a preloaded mark scheme, but it did not provide a full solution that could be implemented in laboratories for a variety of reasons. Most notably, the application did not provide support for various examiners marking at the same time, given that it saved marks data locally on the device it was running on. As having various examiners at the same time is an option that most unit leaders opt for, this made it difficult to use *MarkIt* in real-world laboratory sessions.

A further discussion of MarkIt 2017/18 and its impact on this project is given under Project Method, in section 3.4.

2.3. Android

Android is a mobile operating system (OS) developed by Google, based on the Linux kernel [2]. It was launched in September 2008 and is now the OS with the biggest number of installs on devices worldwide [3]. It can be executed on tablets and mobile phones and variants of it exist for TVs, smartwatches and laptops.

2.3.1. The Android platform

Development of Android applications is done via the Android Software Development Kit (SDK) [4]. The SDK essentially is an API for the OS and allows developers to build apps and functionalities for devices running Android. The SDK includes a debugger for the OS, libraries to interact with the machine running Android and documentation. Development of applications is done in Android Studio, an Eclipse-like IDE designed by JetBrains under the supervision of Google [5]. Layout files for Graphical User Interfaces (GUIs) are written in XML and applications are written in Java although recently Google has launched support for Kotlin.

2.3.2. The architecture of Android

Like any other operating system, Android was built with a specific architecture in mind. Although a full exploration of the architecture of the OS is beyond the scope of this project and report, it is important to mention certain aspects of it for the reader to understand parts of the development process of MarkIt.

In Android, users interact with **activities** [6]. These are the most basic building blocks of an application within Android. They are composed of a Java class that defines their behaviour and an XML file that defines their layout. Activities are managed by a part of

the OS called the activity manager, to which the developer has access through system calls, to create activities or switch from one to another. A real-world example of the use of activities is apparent to anyone who has ever interacted with a phone application: one activity will be used to control the main functionality while another will provide the user with a way of changing the settings.

Figure 2 - Applications contain activities which contain fragments

Activities, in turn, launch **fragments** [7], which are a sort of mini-activities, or small code snippets that take up a section of the screen and execute a certain piece of code.

They were developed to provide a way for developers to reuse small fragments of code that implement a certain action on screen. Fragments are managed by a fragment manager, which the developer can only access to create, destroy or switch fragments. Fragments must always launch from an activity and are linked to it – if the parent activity is closed, all its fragments will also be destroyed.

Other elements of the Android environment, such as Toolbars, Drawer Layouts, EditTexts or spinners are specific to the OS and can be discussed in other sections of this report, although their use tends to be more specific and therefore will probably be inconsequential in the discussion of the project.

2.4. Data storage in mobile applications

Data and its manipulation are the basis of most digital applications, and *MarkIt* is no different. A basic understanding of data storage in software as a general concept and in mobile applications, in particular, is necessary to understand the development of this project.

Android applications like *MarkIt* (and more generally, most applications) require access and use of non-volatile memory – a way of storing data beyond a power off of the device they run on. Android provides two ways of storing data on a device's physical storage. The first one is through internal memory file storage [8], where a programmer can write certain parameters to a file and let the OS handle the specifics of the storage of the file beyond location within directories. The second one is through a local databasing system based on

Structured Query Language (SQL) schemes – a relatively simple and well-tested system for database storage that is an industry standard within software development. This memory management scheme is further discussed below.

The approach to memory management during this project involved one further step in complication: the memory would be non-volatile but would not be stored locally on the device running it, but rather on a server in the cloud. Due to the specification of the Android SDK, an Android application is unable to directly query an SQL based cloud server and needs a middle translator application (RESTful API) running on the server that provides the requested information as a JSON file. Both concepts are further explained below, as they are central to the development of *MarkIt*.

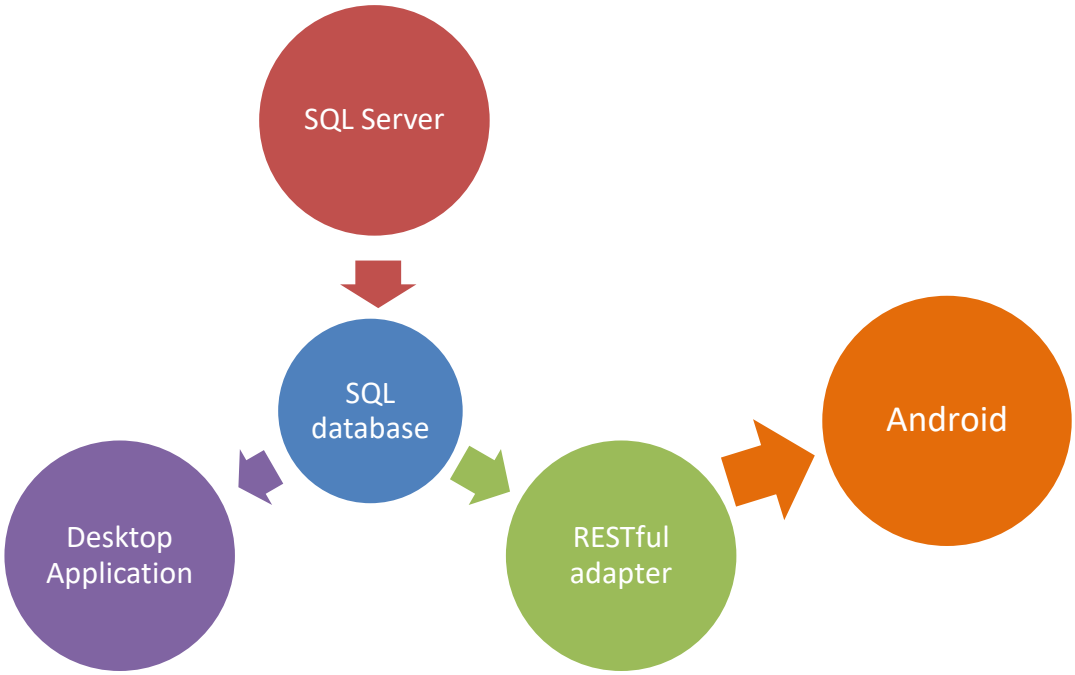


Figure 3 - data architecture for MarkIt

2.4.1. Relational Databases and Sequential Query Language (SQL)

Relational databases are a type of digital database first proposed in the 1970s that has now become the standard way of storing data in databases worldwide [9]. Their basic principle of operation is simple, similar to what a user of Microsoft Excel might be used to interacting with: a database is a collection of tables, each of which holds information within columns and rows. Each column designates an attribute of the data it holds (a name, a date, etc...) and each row holds an entry of information with has these

attributes. In this way, in a table about customers of a shop, columns might include customer's names, customers' customer ID and customer's birthday and each row would represent a certain customer (e.g., "Valentino Rossi"-> row 1 column 1, "4646464" -> row 1 column 2, "16/02/1978" -> row 1 column 3).

The special thing about relational databases is that they allow certain attributes to be shared amongst tables, so that an attribute, for example, "customer ID", might be present in various tables, and so different pieces of information about a customer can be saved in various different tables but the information can all be accessed by knowing a certain attribute, for the sake of our example, that customer ID.

SQL is a programming language that has become the standard way of managing and querying relational databases [9]. As a language, it allows users to inquire for data in a syntax similar to that of common English. In its basic form, for example, it allows users to select certain attributes of data from a table based on a filter: a user can ask to know all the customer names and IDs of people born in March. It also provides statements to add or modify table elements or to create tables in the first place. SQL has become so omnipresent that relational databases have come to be known as SQL databases, which is the nomenclature used in the rest of this report.

2.4.2. RESTful APIs and JSON files

As mentioned previously, the Android SDK specification explicitly forbids applications from directly connecting to a database server (be it cloud-based or of any other kind) and retrieving data [4]. Due to this limitation, an Android application will not be able to use SQL to retrieve information from a database, but will only be able to send a message to the database server making a certain request. It is then the responsibility of the server to physically implement the logic and SQL infrastructure to serve that request, package that information in a certain way that the Android OS will understand and send it back to the requesting app.

This logic that runs in the database server and takes care of the two-way "translation" between the Android app and the database server is called a RESTful API, a standard name for HTTP-based servers that handle data requests. In the specific case of Android, the SDK and documentation mandate that information passed into the application must come packaged as a JSON file [10], a file format similar to XML that codifies information as key-value pairs, in a way that is readable to both humans and

machines through code.

A RESTful API would have to be used for the cloud backend of MarkIt and JSON would be the file format in which the application would receive cloud data. Further exploration of these concepts can be found in “Project Method”, section 3.

2.5. Cloud services

Having explored the functioning of memory within Android, and how the OS handles data stored in the cloud, the services that offer that cloud storage must be examined as well. With the proliferation of the Internet in the 21st century and the explosion in use of mobile devices after the introduction of smartphones in the early 2010s, a need for cloud services emerged. This term encompasses many things: from data services where developers can store their application and user data online to processing and machine learning power hosted in the web for companies to make use of; there is a growing industry in providing computer related services on the cloud.

This trend has been noticed by many technological companies. 10 years ago, a developer who wanted to set up a cloud service would have had to purchase hosting from a web provider, set up the data storage infrastructure within the server and then implemented logic to interact with client applications. With the emergence of cloud services, this process has been simplified greatly, offering developers a quick and easy way to get the resources they need to build applications. There are many companies that offer hosting of SQL and non-SQL databases, authentication and log-in infrastructure, virtual machines and even programmable chatbots amongst many other IT services. Developers pay a fee to have access to these services and interact with them as they would with a code library – there is documentation available that explains how to make use of the services and tutorials that help explain the functionality they offer.

Further exploration of the cloud service offers available and the providers that were considered and chosen for this project is available in section 3.5.

3. Project Method

The following section explains the approach taken towards completing the project, detailing the procedure and practices that have been undertaken in order to deliver the Android application Mark It.

3.1. Literature review: Android, Java and databases

Work on the project began with a literature review, in order to come to terms with the basics of Java, a programming language which was completely new to the student; Android, another completely new platform and the current state of databases, SQL and cloud services, something with which the student was more familiar given that he had some previous experience in developing this kind of code during his industrial placement.

A basic understanding of Java as a programming language was obtained by reading "Sams Teach Yourself Java in 24 Hours" [11] and attempting to make connections between the student's previous knowledge of C++ and C# and the newly learnt language of Java. Once it was deemed that the knowledge of Java gained was enough to allow for an understanding of basic code and development of small applications, the focus shifted to applying this knowledge to the specifics of Android development. This was important given that although Android applications are written in Java, it is the architecture and workings of the Android OS that are crucial for the development of apps: Java as a language is merely the tool used to interact with the OS's SDK.

Lastly, once background reading had been completed on the topics of Java and its relation to Android and the SDK, the investigation of data-basing systems began, with reading about "standard" SQL databases and the alternatives that have emerged over the years [12]. These are data-basing schemes like *MongoDB*, *non-SQL systems* or *Rel,* although it was concluded that due to its simplicity and ubiquity it would be better for this project to be based on traditional SQL.

3.2. Developing a sample application

Having investigated the background and context of the tools and frameworks that were going to be used in this project, attention shifted to attempting to familiarize the student with said tools – once the theoretical learning had been done, it was important to attempt to apply it in a realistic scenario. In this way, the student set off to develop a sample Android application in order to become familiarised with Android Studio and writing Java

code in an Android SDK setting.

The result of this effort was the application *Wheel of Life*[6], an app developed during the Oxford University Hackathon 2018 in November 2018. The student took part in the Hackathon as a participant and led a project involving developing an Android application to aid, measure and control the mental health of university students in the UK. The application's functionality involved recording information about a student's day and mood and displaying it in a visual way to attempt to identify trends in what made students feel better or worse during their university lives.

Although the application was limited in many senses, it proved to be a great learning experience in development of Android apps, allowing the student to become familiar with concepts such as activities or fragments, ideas core to the architecture and working of any Android program.

3.3. Existing codebase and project

Having acquired a relative familiarity with the tools and elements that would be the core of the project, the focus of the project moved to review the state of *MarkIt* as an application, as there was an already existing codebase from the 2017/18 project. Work began with reading the final report submitted by George Wellard in April 2018, with special consideration to the specifics of the results presented: what exactly had been produced in the previous years? What did the application do exactly, and how did it do it? Furthermore, what had worked for Mr Wellard and could be applied again and what had he pointed out as problematic and should be taken special care of?

The report answered many of these questions while providing a solid foundation in understanding what the finished application at the end of the 2018 academic year achieved to do. Mr Wellard spent time in his report going over the architecture of Android, which pointed to the importance of understanding the OS as a working system before launching into development; pointed to the idea that thorough testing of the application's functionality was crucial in ensuring a stable software product and regretted not taking enough time before commencing development to plan certain aspects of the application, in particular classes and data structures. This information was extremely valuable for the student, who had "expert advice" in how to go about development, advice that was applied in many cases further on in the project.

[6] A showcase of the *Wheel of Life* application can be found in annex G

The time then came to examine the software project and the codebase that Mr Wellard had provided, as well as the functioning app on an Android device. The application proved to work as described in the report on the testing device: it provided an interface to mark a laboratory exercise selected from a pre-loaded list with a certain example student as the markee. Nevertheless, a close examination of the underlying code was less technically positive in the context of this project. Mr Wellard's application met the requirements of the project he embarked on: it provided an application that could mark students on pre-loaded data schemes on the device – yet it did not fulfil the requirements of what the student wanted *MarkIt* to be after this year's project. *MarkIt 2017/18* was written in a situation-specific way, making it ideal to showcase the concept of an application to mark students' work but architecturally unable to provide a full solution that could be used in a real-life situation.

This discovery was unfortunate in the context of this project: what had started as a task to adapt an already existing application to store data in the cloud was now becoming a project in which an application would have to be built from the ground up, with special care taken to ensure its architecture was adapted for a cloud data storage scheme and could be used in a real-world scenario by real-world users. This, however, was not disheartening, given that the student welcomed the challenge of building an entire application and realised the potential for efficiency in writing a completely new codebase rather than having to refactor a previously existing one.

3.4. Finding the optimal cloud provider

The options for an Android developer seeking a cloud framework on which to base an application's operation are relatively limited, especially when compared to those faced by a developer working with a desktop platform. This is due to constraints set by the Android SDK and specification, which forbid Android apps to directly connect to a SQL database through a connection string or similar. Rather, the specification mandates for an adapter to be built - a server-side application which provides a translating service between an Android device and the SQL server. The Android app can the query that service (a RESTful based API) which in turn queries the SQL server database and provides data to the original querying device.

Because of these constraints, time had to be spent comparing services that could provide the platform and logistics to support development: specifically, platforms that would provide hosting of a SQL database and either a way to create the RESTful adapter service

to query said database or provide an already built API.

There are many options like these available to developers, although the most robust ones tend to the offered by the bigger technology companies - e.g., Google, IBM, Amazon or Microsoft. This is due to the sheer size and experience of these companies when it comes to this kind of industry – the biggest technology companies have the biggest server farms, the knowledge of how to develop useful tools to run on those servers, etc… Of the four mentioned companies, research yielded a portfolio of 3 different offer philosophies.

IBM, a company that has usually been associated with appealing to businesses and enterprises, maintains that philosophy in their cloud service, IBM Cloud Products. On top of the basic cloud services like SQL database hosting or networking services, they offer AI and machine learning tools to help small businesses. Due to this business focus, this offer was rejected as a viable option, as other options out there could tailor better to this project's needs.

Amazon recently launched Amazon Web Services (AWS) and boast a full suite of cloud tools, ranging from database hosting to business and security applications. Nevertheless, the student had prior experience in working with AWS as he had been involved in attempting to add cloud capabilities to the application developed at the University of Oxford Hackathon 2018[7]. The experience with AWS had been cumbersome, mostly due to the service's poor documentation and lack of tutorial information. Due to this awkward experience, AWS was rejected – having access to good documentation is a crucial part of developing software and AWS's faults were deemed too cumbersome.

This left two final candidates: Google's Firebase service and Microsoft's Azure Cloud. Firebase is a service launched in 2011 and acquired by Google in 2014. Owned by the makers of Android, Firebase provides closer integration with the SDK and Android's development tools, offering easy authentication options, quick code samples and even pre-written widgets for developers to use. Nevertheless, it only allows for the storage of database data in a non-SQL format. Their service is based on non-SQL protocols: data is stored as Key-Value pairs in JSON format, which can be queried and manipulated in a way similar to SQL but entail certain significant differences. This made the service unsuitable for this project, given the project's needs heavily rely on relational data - all mark scheme and marks info would be related to a common student ID.

[7] Further information can be found in section 3.2.

On the other hand, Microsoft's Azure Cloud service was launched in 2010. It is a suite of cloud services amongst which a developer can find SQL data storage. This already made it the ideal service for this project's needs, yet upon further investigation, it was realised that Azure natively provides programmers with a RESTful API to interface between their cloud service and a developer's mobile application.

Once the cloud service was selected, considerations had to be made regarding cost and resources. Although Azure has a cost depending on data stored and queries executed, the student had £150 Azure credit given to him at the Oxford University 2018 Hackathon by representatives of Microsoft, a budget which was calculated to be more than ample to suit the project's needs. In this way, the cloud service provider for *MarkIt* was selected, and development could continue to its next phase.

3.5. Planning the database structure of MarkIt

Following the advice received by the project's supervisor and the student that developed the first version of MarkIt in 2017/18, the student was careful to precisely plan as many aspects of the application as possible before implementation. This process started with careful plans about the application's database structure.

Database normalisation is the process by which a is designed to operate as efficiently as possible [13] bearing in mind the use that the database will have and the requirements of the interface with the database (in this case, the SQL language). Normalisation defines various tiers of database efficiency, each with a set of organisational requirements. In this way, a database can be defined as complying to 1NF (first normal form), 2NF, 3NF, 4NF or 5NF; each of them a tier with requirements. A 2NF table will have been organised to 1NF standard and then went through a second normalisation process to go up to 2NF level. The use and, specifically, the extent of normalisation is a source of debate in the software development community, as many argue that the effort of normalising a database is too large to compensate for the effort that the exercise needs. Nevertheless, there is an informal consensus that the first two levels of normalisation make a database more workable and efficient and are therefore worth putting time into them.

In this way, after sketching the data needs and structures for the application, a 2NF normalised database structure was planned out – with a total of 7 tables that defined information about students, laboratory groups and locations, unit information, mark schemes and collections of final grades. 2NF normalisation meant that the database

aligned with the following criteria:

- No duplicate columns could be found on one table
- Separate tables were created for each set of data and each row could be identified by one unique column (primary key)
- Subsets of data that applied to multiple rows of a table were removed and placed in separate tables
    - Relationships between these new tables and their predecessors were created

By the end of this planning process, the data structure of the database – and therefore, the application – was defined and ready to be deployed and manipulated. This initial planning was to be a great advantage in the later stages of the application, as an organised database was easier to query and interact with. The final data structure can be found in annex H.

3.6. Developing a cloud-based sample application

Having gained the knowledge to develop Android applications, chosen a cloud provider and determined a certain database structure, a second sample application was developed, this time to get to grips with the working of Microsoft's Azure Cloud and the specifics of connecting to the service from an Android.

The application that was developed was *Karanta!*[8] an application meant to help users keep records of the books they are reading and pick which books to read next. The application allowed users to store a list of the books they meant to read, mark books from that list as already read and give them a rating and add comments. Another feature made the application randomly select a book from the upcoming list for the reader to read next.

All application data was stored in an SQL database hosted in Azure, and interfacing with said database was done through an Azure service called "Mobile App Easy Tables". Further information about the specifics of these schemes is below, in section 3.6.1.

This exercise might seem redundant in hindsight – yet it was extremely valuable for the student. Although marketed as extremely simple and convenient to use, Azure is a service that requires certain technical know-how and developing *Karanta!* forced the student to develop a solid foundation in knowing how to interact with the Azure portal (Microsoft's

[8] A showcase of *Karanta!* can be found in annex I

interface to set up and manage the services Azure offers). Furthermore, the student developed methods to read, write and modify elements in a cloud database, which were built with modularity in mind to allow them to be slightly modified and used in the source code for MarkIt.

3.6.1. Putting the cloud infrastructure in place[9]

It is important to explain how the specifics of setting up the cloud infrastructure function in order to understand the entirety of the project. When logging into the Azure portal, users are presented with a dashboard of all the resources that they have acquired (initially, none) and a toolbar of all the resources that are available to them.

In order to set up the database, a user must acquire and configure a server. This server represents an online server and possesses all the characteristics of an ordinary server: users can configure a firewall, install protocols and applications on it, etc… In this case, the student set up the firewall to allow IP connections 000.000.000.000 – 255.255.255.255 (effectively disabling the firewall to enable testing from any location) and named the server “universityofmanchester”. Creating an SQL database was then easily done with the created server as the host – the SQL database “markit” was accessible and hosted in the “universityofmanchester server”.

The next step entailed the creation of the RESTful API to create an adapter between the Azure-hosted database and the querying Android device. Azure names these services “mobile apps” and also requires the developer to create them and link them to a server and a data source, in this case, the “markit” database. Upon creation of the mobile service, Azure gives the developer a downloadable Visual Studio project that must be run and deployed from the Visual Studio program within a Windows computer – this launches a website which serves as the RESTful adapter, as the Android device queries the site which in turn queries the database and sends a reply.

Although this process might seem relatively straightforward, the documentation provided by Azure was outdated in the naming of one service and made this part of the development a longer process than intended. Although this wasn’t technically difficult, that small inconvenience proved the importance of well-documented software, as a lot of time was wasted due to an overlooked detail in the documentation.

[9] A more detailed explanation of the steps taken to set up the cloud server can be found in annex J

3.7. Planning the software architecture of MarkIt

Once again following the advice learnt from the previous year's project, the student took an amount of time to plan out the construction of code for the final development of the *MarkIt* application. Most of the planning entailed sketches of UI but also careful thinking about where in the application's flow and lifecycle were queries to the cloud going to be made, how was data going to be handled internally and, very importantly, what the data containers for the information downloaded for the cloud were going to be.

3.8. Writing MarkIt

By this point, the student had attained both the knowledge and the digital framework to be able to commence development of the final application.

Over the course of two months, the student proceeded to develop all components of the *MarkIt* application, beginning with the development of simple, one paged-activities and building upon those sections of code to produce the application showcased in Section 4. The application was built with modularity and scalability in mind – having experienced the issues that building a rigid application could bring first hand with the experience of *MarkIt* 2017/18, the ability for the application to grow and adapt to new needs or ideas was a core belief in developing the software.

The final application encompasses 3 Android activities of varying complexity – the most simple activities download information from the cloud for the user to select which students and mark schemes must be marked; the most complex bring up a toolbar which holds various fragments which the user can iterate through and use concurrently in order to mark a student.

The process of building the application was frustrating at times as although the student felt confident in his programming abilities learning a completely new platform like Android poised a significant challenge. All of the preparation, learning and planning made the experience of writing the application easier and more efficient, but there were elements of the project that could not be planned for or surpassed until the final app was developed – truly understanding the architecture and life cycle of Android and its activities proved to be the most difficult task in the project. The code written to manipulate how the application interacted with the end user is relatively simple, but the underlying concepts of how elements of the OS and user experience come together to create an application were much easier learnt theoretically than applied practically.

When developing the application, the student relied at times upon the help of the online community at Stack Overflow [14][15][16]. This Q&A website provided an excellent platform for the student to interact with more experienced Android developers and ask questions about the specifics of developing under the Android SDK – more knowledgeable developers answered his questions in a way that allowed him to understand not only how to solve the problem at hand but also the underlying reason why there was a problem at all.

In the end, all these difficulties all proved to be surmountable, and although the final application might have been finished a few weeks later than originally intended it is a robust and solid product that solves the problem it sought to tackle and can realistically be used in a real-world scenario.

3.9. Writing *MarkIt* Desktop companion

Nearing the end of the project, having finished the core application of *MarkIt* for Android, the student turned his attention to providing a solution in order to allow users to upload mark scheme information to the cloud, preferably in the form of a desktop application.

The student had gained ample experience in writing C# applications during his industrial year working as a C# developer for ALFATEC SISTEMAS, S.L., a software development company in Spain. This greatly simplified the process of writing a desktop app for Windows computers and allowed the student to deploy his ideas for a desktop companion app relatively quickly.

The application developed is described in detail in section 4.2, and development of it took place in Microsoft Visual Studio as a native C# application. Most of the work done in this round of development went into understanding the ways in which C# code could be used to read and take information from Excel files, as this was the medium selected for mark scheme information to be fed into the app.

Connection with Azure's cloud was much simpler in the desktop app than with Android, given that the student only needed to supply connection strings and log-in credentials in the C# code and could query the database directly in SQL and receive the information required without the need for an adapter.

3.10. Testing

The student finalised work for the project by extensively testing the two applications

developed. Having had experience as a developer during his industrial year, he had first-hand experience of how critical proper testing was in software development projects and carried out scrupulous testing procedures for both the Android and Windows applications. The tests that were carried out ranged from application-specific tests such as empty fields when attempting to submit mark schemes to the cloud to industry-standard testing like an Android monkey test[10]. The results of this testing are showcased in sections 4.1.4 and 4.2.3 respectively.

[10] Random stream of user input such as taps, clicks and swipes fed to the application, further information in annex L

4. Results

This section showcases the results of this project, in this case, the two completed applications developed to provide physical mediums to create mark schemes and mark students as well as an explanation of the cloud back-end powering the system. Given the nature of this project as a software development based task, the results are presented in a manner similar to an application specification.

In order to understand the functionality of the app, it is convenient to explore the structure of a mark scheme. A mark scheme is composed of a parent component called an “Assignment”, which englobes a certain amount of “Sections”. Each section holds up to 5 “parts”, which are the smallest units that an examiner can check as completed or incomplete. Parts have marks associated with them that get cumulatively added and become the marks for a section. The sum of the marks for all the sections in an assignment yields the assignment mark.

4.1. Mark It Android App

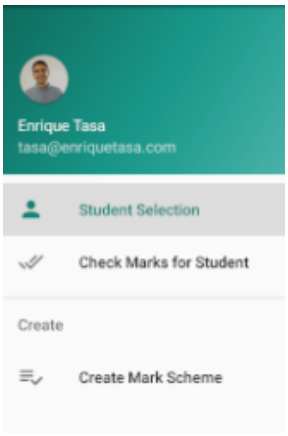


Figure 4 – the navigation drawer in MarkIt

The main component of the MarkIt solution is an Android app that can be used to select a student to mark, complete a mark scheme and submit it to the cloud for saving. It also allows for consultation of a student’s grade for a certain completed assignment.

The application has been designed as a navigation drawer application (fig. 4), so a swipe from the left side of the screen or a tap of the hamburger button at the top left corner will showcase a drawer that allows the user to switch from one activity to another. The application is split into 3 activities.

4.1.1. Student selection activity

This is the activity that the user sees first as they open the application (fig.5). It has a simple design of various text boxes and dropdown menus. The user can select what unit is this laboratory for, what specific assignment is going to be marked, what laboratory group is being marked and what student ID is going to be marked.

The dropdown selection menus populate with information from the cloud, downloading specific ranges of data depending on what the user selects. In this way, the user will need to select what unit is the laboratory for before being prompted to select a specific mark

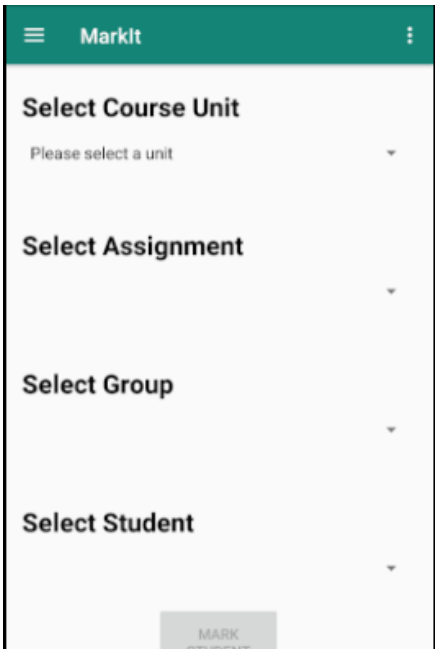


Figure 5 - the view that starts MarkIt

scheme to use and the mark schemes that will be available will only be the ones submitted for that unit. In the same way, only the laboratory groups and students that have been enrolled in said unit for said laboratory will appear as selectable for the examiner.

Once all the information regarding what student and mark scheme is inputted, the examiner is able to click a “Mark Student” button, which will take them to the next activity.

This activity is relatively simple and was easy to develop once an understanding of the transactions of data between device and cloud were understood.

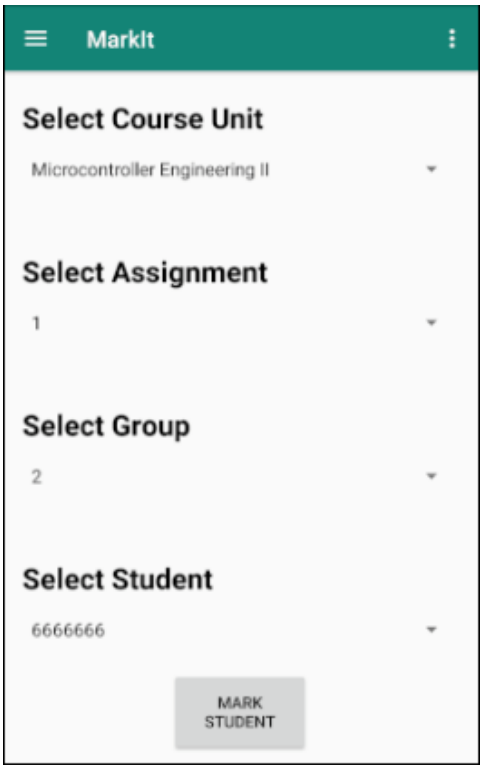


Figure 6 - student selection activity with completed fields

Data transactions between Android and the cloud

Various sections in this report dive deep into the cloud, the SQL databases that power this application and how Android relates to them. However, it is also interesting to explore how the data transactions are made specifically and programmatically.

As discussed, Android is unable to communicate with a database via pure SQL and must rely on an API to make data transactions. In the case of this application, said API is provided by Microsoft’s Azure Cloud, and the functions used for data transactions come from that API.

The model used in Azure’s API is very interesting from a programming point of view – in order to facilitate the management of data within developer’s code, the makers of the API avoided making programmers deal with databases directly. In this way, programmers do not deal with column names, rows or SQL rules; rather manipulating standard classes that are passed to the cloud for storage and can be recovered from the cloud via queries. A code example is shown in fig.7

```
1.  void refreshUnitDropDown(String field, String condition){
2.
3.      Log.d(TAG, "Refreshing course unit dropdown");
4.      // This is how a basic query is executed, in this case all units are retrieved
5.      mClient.getTable(LabGroup.class).where().field(field).eq(condition)
6.              .execute(new TableQueryCallback<LabGroup>() {
7.
8.          // Listener that automatically gets set for the result of the transaction with Azu
    re
9.          @Override
10.         public void onCompleted(java.util.List<LabGroup> result, int count, Exception exce
    ption, ServiceFilterResponse response) {
11.
12.             if (exception == null) {       // if no exceptions, transaction succesful
13.
14.                 // Do something if you wish, var "result" contains your data
15.                 List<String> unitSpinnerList = new ArrayList<>();
16.                 unitSpinnerList.add("Please select a unit");
17.
18.                 Log.d(TAG, "Dropdown content successfully retrieved from cloud");
19.
20.                 for (LabGroup unit : result) {
21.                     if (!(unitSpinnerList.contains(unit.getLabGroupUnit()))) {
22.                         unitSpinnerList.add(unit.getLabGroupUnit());
23.                     }
24.                 }
25.
26.                 AddListToDropdown(unitSpinnerList);
27.
28.             }
29.
30.             else {
31.                 Log.d(TAG, "Exception found: " + exceptio
    n.getMessage());
32.
33.                 }
34.             }}
35.     });
36. }
```

Figure 7 – code example of API use

4.1.2. Student marking activity

The second activity in the application perhaps is the central one. The “Student select activity” sends the information selected by the user about the unit, mark scheme and student to this activity via an Android construct built for data sharing between activities called a “bundle”. The student marking activity then unpacks the bundle and downloads the mark scheme information for the required mark scheme from the cloud, presenting it to the user as seen in fig. 8.

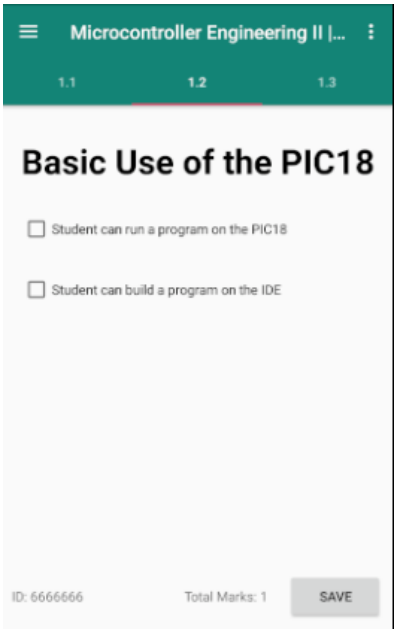


Figure 8 - showcase of marking app

Each section in the selected assignment is displayed on one tab of the activity, with the user being able to swipe to navigate between them. Each section then displays the section title and a series of checkboxes with the parts of the section, as well as the student ID of the student being assessed. Each of these tabs holds an Android fragment that takes the mark scheme information downloaded by the activity.

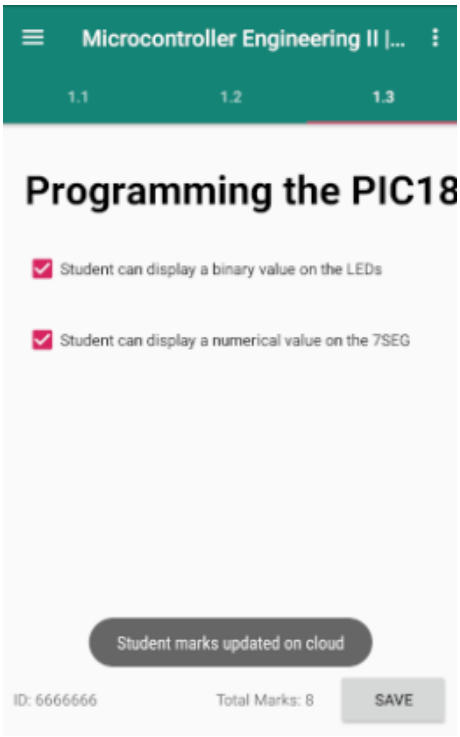


Figure 9 - Showcase of the marking activity with checked parts and upload confirmation message

The mark student activity then allows the user to check the checkboxes to determine whether a certain part of the mark scheme has been completed. On checking or unchecking the checkbox, the total marks label displayed at the bottom of the activity gets updated to display the current amount of marks the student has for this certain mark scheme. A “save” button allows the user to save the results of the mark scheme into the cloud, with a message coming up to confirm that cloud upload has been successful.

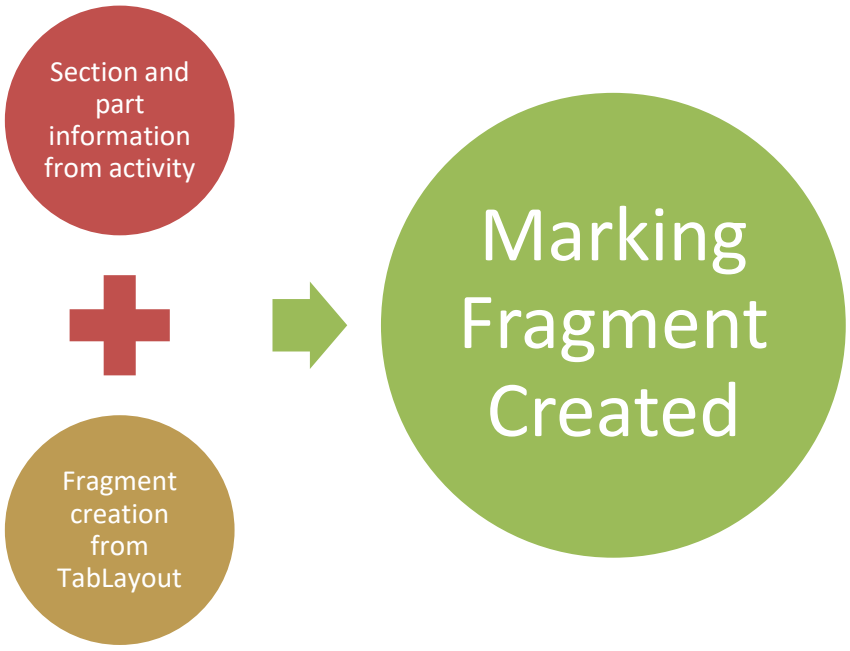


Figure 10 - fragment creation architecture

4.1.3. Grade consultation activity

The final activity in the application can be accessed via the navigation drawer at the left side of the screen. Its purpose is to enable users to consult what grades the cloud database holds for a certain student regarding a certain assignment.

The activity has a simple design which is similar to the student selection activity: it holds a selection dropdown for student ID and assignment as well as a button to consult the database for the information about grades. The app can be seen in fig.11.

When the button is clicked text appears below it to indicate the number of marks received by each student, allowing the user to have quick access to that information. If there is more than one record of a student being marked for an assignment in the cloud database, the application will showcase a log of the instances on marking in chronological order.

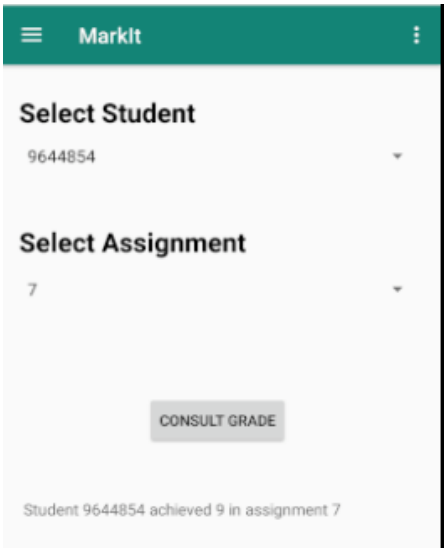


Figure 11 – showcase of grade consultation activity

4.1.4. Testing

In order to ensure application stability and user experience quality, the application was tested extensively. Two types of tests were performed: software automated tests run with the aid of the Android Studio IDE and real-world issue-based tests thought up by the student (what does the application do if there is no internet connection? How does it behave if a user deliberately tries to interfere with its operation?). The results of these tests can be found in annex L.

Overall, the application passed most tests, and only small changes were needed to be made to fix the bugs found during the testing procedures.

4.2. Mark It Windows App

A companion desktop application to finalise development of a full solution was developed after completion of the Android app. The functionality of the Windows application was twofold: it had to provide a way to download student-grade data as a CSV file and allow for uploading of mark schemes to the cloud.

```
<assignment>
    <assignmentNumber>1</assignmentNumber>
    <assignmentName>Getting to grips with the PIC18</assignmentN
    <assignmentAvailableMarks>9</assignmentAvailableMarks>
    <assignmentAuthor>Tasa</assignmentAuthor>
    <LabGroupLocation>Barnes Wallis PC Cluster</LabGroupLocation
    <LabGroupUnit>Microcontroller Engineering II</LabGroupUnit>
    <LabGroupNumber>1</LabGroupNumber>
    <LabGroupStudentID>9673164</LabGroupStudentID>
    <markScheme>
        <section>
            <sectionNumber>1</sectionNumber>
            <sectionName>Attendance</sectionName>
            <sectionAvailableMarks>1</sectionAvailableMarks>
            <sectionAuthor>Tasa</sectionAuthor>
```

Figure 12 - XML mark scheme format

The developed application accomplishes both these tasks. It is a simple, C# Windows application that is composed of two tabs: “Download grades” and “Upload mark schemes”. The download tab allows for selection of

an assignment and a download location, the upload sections allows the user to find an XML file that defines a mark scheme and upload that information to the cloud.

4.2.1. Upload mark scheme tab

The uploading tab serves the purpose of allowing lecturers or unit leaders to upload the mark schemes they have defined for a certain laboratory to the cloud infrastructure of MarkIt, pre-loading the information for the Android app to display, manage and change it.

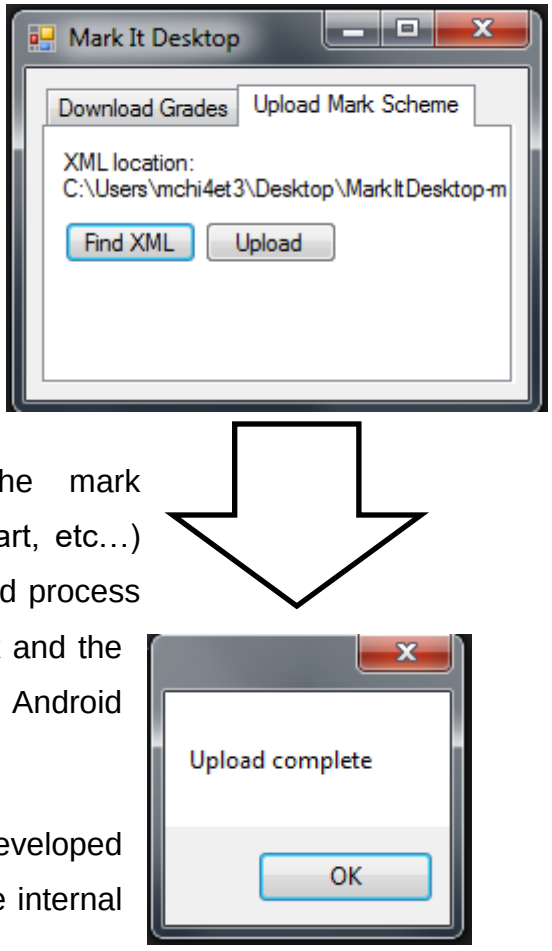


Figure 13 – desktop upload and message box

Designed as simply as possible in the context of this utility application, the tab has one text box and two buttons: one to find the XML file that defines the content of the mark scheme[11](assignment name, marks available per section/part, etc…) and one to upload the selected file to the cloud. If the upload process finalises successfully, the user is notified via a message box and the content of the mark scheme can then be checked on the Android application.

In order to implement this functionality, methods had to be developed to specify how to respond to GUI events and to deal with the internal logic needed to make the upload work. Specifically, a class was developed to decode the input XML in order to separate the assignment, section and part information as well as a class to deal with the decoded data and upload it to the correct database in the cloud.

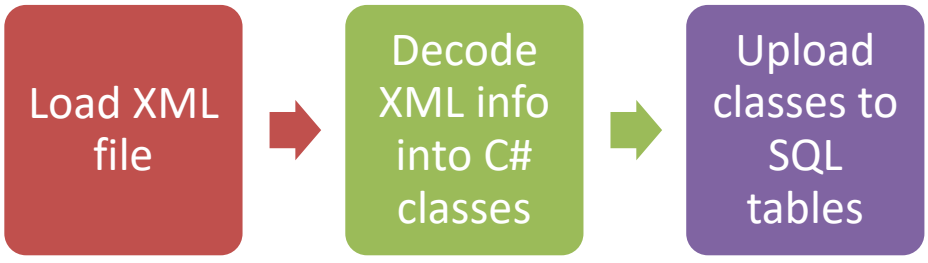


Figure 14 - XML load, decode and upload logic

[11] Sample XML file can be found in appendix K

4.2.2. Download grade tab

The second tab in the application serves the purpose of downloading a CSV file with student ID information and the corresponding grade achieved by that student in a specific assignment. The tab contains a dropdown menu to select the assignment to download data for, a label that displays the download location for the CSV file and two buttons: one to define the download location and another to commence the download. The interface is shown in fig.15.

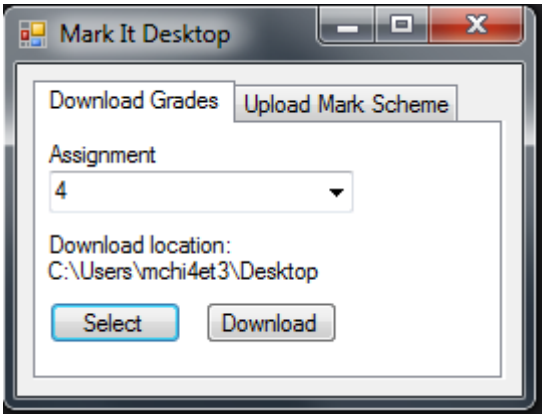


Figure 15 - Interface of download part of application

In order to implement this functionality, a class to query the online database for the information regarding student IDs and marks achieved had to be created, and methods had to be developed to write into the CSV file in the correct format.

4.2.3. Testing

In a similar vein to the Android app, extensive testing was conducted on the application to ensure its stability and correctness. Due to its limited size, testing procedures were quicker to implement, although they yielded interesting results – especially in how to notify the user about what problem was causing the application to malfunction specifically.

In order to solve these issues, a class was developed to act as a diagnostic tool to the application itself, allowing users to be notified about what exactly the problem causing malfunction was: be it the cloud connection, the XML file or a wrongful combination of selection parameters.

The results of these tests can be found in annex L.

4.3. Cloud Back-End

Although a relatively simple component of the developed system, it is important to at least mention and briefly explain the composition of MarkIt's cloud back-end.

The main component of the back-end is the SQL database where all application data is stored. The database resides on a server within the student's Azure account, has 7 tables and is normalised to 2NF. There are no formal rules for the database and it can be queried via a connection string (IP and authentication data, used by the Windows application) or

through a restful API provided by Azure (used by the Android app).

The RESTful API is a service that Azure calls “Mobile Apps” and is connected to the SQL database through a configuration parameter in the Azure portal. It supports communication between the Android application and the database server and is hosted at markituom.azuresites.com

5. Conclusion

Having examined a background to the project, the methods used to deliver on the objectives set out to achieve and the final results delivered, this conclusion provides a review of the objectives and some final comments by the student.

5.1. Objectives review

Having set out a collection of base and stretch objectives at the start of the project, these can now be examined in light of the delivered application.

5.1.1. Base Objectives

- **Implement a cloud-based storage system for the data collected by the application**
  This objective was successfully achieved in the implementation of a SQL database system with Microsoft's Azure cloud as the provider. Connections were made from mobile and desktop devices, ensuring the functioning and flexibility of the developed system.

- **Ensure that various users can access the data simultaneously, reading/writing at the same time**
  The use of a service like Azure has ensured that multiple users will be able to use the applications at the same time – just like the service provided an API to interact with the SQL database, they provide infrastructure to ensure that various users will be able to use the service at once. Due to the service tier selected on Azure portal, the maximum amount of simultaneous users is 5. The last user to submit a certain mark for a student will be the one that is recorded as the official mark for that student.

- **Implement a system to allow users to save mark schemes in the cloud and use them repeatedly**
  In saving the mark scheme data permanently in a cloud SQL database, the mobile application can provide marking activities for any available mark scheme in the cloud, making mark schemes reusable objects.

- **Implement a feature to allow users to consult student's grades for the current laboratory in real time**
  The grade consultation activity provides a framework to consult the grade a certain student has achieved for a set assignment. Furthermore, it provides a history of all the marks received, in case changes had happened in marking assignments.

- **Provide a way for lecturers or unit leaders to download grade data in a format compatible with BlackBoard uploads**
  The *MarkIt* desktop application allows for the download of grade data in CSV file that can then be uploaded directly to Blackboard

5.1.2. Stretch Objectives

- **Implement a feature to allow users to consult student's grades for all previous laboratories**
  The same activity which provides support for real-time laboratory grade consultation allows for users to select any laboratory assigned to a certain student ID for consultation.

- **Allow users to define mark scheme structures in a desktop application and save them in the cloud**
  The *MarkIt* desktop companion application provides a tool to define a mark scheme's sections, parts and marks – and then provides the functionality to upload the mark scheme to the cloud and save it for later repeated use if so required.

- **Allow users to define mark scheme structures in the mobile app and save them in the cloud**
  This objective was not met. Although it was a stretch objective, and therefore considered a useful "extra", the student did not have time to implement this activity due to the unexpected re-writing of all of the application. In the navigation drawer for the Android application, a setting has been added to make the creation of an activity to implement this functionality in the future.

5.2. Final comments

This project set out to develop a solution, in the widest sense of the word. The ultimate goal of the project was not to develop an application that would simply work or deliver on a certain set of objectives relating to the cloud, but to deliver a digital package that would, if used, do away with a problem. This is the power of new technologies in the form of applications and smartphones: for the first time in history, humans are able to write software to concretely and efficiently address daily issues and inconveniences, carrying the device that runs said software in their pocket.

MarkIt is an application with a lot of room for improvement, but it achieved what it set out to do: albeit simple and perhaps plain in its user experience, it is a solution that can be

introduced into the workflow of university laboratories right away. It can be used simultaneously by various users, lecturers can upload mark scheme data and download mark data to put into BlackBoard and it is intuitive enough that it does not require a lot of training to utilise. In this regard, this project and MarkIt can be considered to have been successful.

The student emerges from this project a somewhat competent Android developer, as well as a more experienced general software developer. Widening the portfolio of languages and platforms that he had interacted with, the student was faced with new challenges and difficulties, having to learn and grow as an engineer in order to solve them. Furthermore, in regard to employment opportunities, the student can confidently claim that he can develop full-stack cloud applications, and have MarkIt as an example of a project which involved most aspects of what can be expected of a software developer: from UI design to algorithm implementation, from database management to improvements to user experience.

Nevertheless, there are things that can be improved about the application, and there were things that could have been done better in the project itself. *MarkIt* has not been extensively tested in Android devices outside the one the student had access to during the project, nor has it been tested in a real-world laboratory situation. As confident as the student is in having delivered a solution, it is obvious that with real-world use issues and necessary improvements would appear.

The project itself, although having delivered what it promised, was at many times inconsistent. Project and time management could have been more efficient: periods of focus and work alternated with weeks of not paying the project much attention and in general most of the work that involved actual programming of the application happened in the later months of the project. An earlier start in development and a more consistent work schedule would have probably yielded a more pleasant experience for the user and a more pleasant time for the student in working in the project.

In conclusion, this project was successful in delivering what it promised to deliver, delivering a ready-to-use application with room for technical improvement. The student has gained relevant experience and learnt valuable skills and lessons while enjoying work on his 3rd year project.

6. References

[1] **Wellard, G.**; **Green, P.N.**, "*App for Lab Support"*, 3rd Year Project Report, University of Manchester.

[2] **Google, Inc.**, *"Android Documentation",* documentation for operating system, Alphabet Inc., https://developer.android.com/docs

[3] **Global Stats: Stat Counter**, *"Operating System Market Share Worldwide, July 2017 – July 2018",* Global Statistics, http://gs.statcounter.com/os-market-share#monthly-201707-201807-bar

[4] **Google, Inc.**, *"Android SDK Tools",* Alphabet, Inc., https://developer.android.com/studio/releases/sdk-tools

[5] **Google, Inc.**, *"Android Studio Download",* Alphabet, Inc., https://developer.android.com/studio

[6] **Google, Inc.**, *"Android Activities",* Alphabet, Inc., https://developer.android.com/reference/android/app/Activity

[7] **Google, Inc.**, *"Android Fragments",* Alphabet, Inc., https://developer.android.com/reference/android/app/Fragment

[8] **Google, Inc.**, *"Data Storage in Android",* Alphabet, Inc., https://developer.android.com/training/data-storage/files

[9] **Microsoft Corporation**, *"Structured Query Language",* https://docs.microsoft.com/en-us/sql/odbc/reference/structured-query-language-sql?view=sql-server-2017

[10] **Crockford, D**, *"Introducing JSON",* https://www.json.org/

[11] **Cadenhead, R,** *"Sams Teach Yourself Java in 24 Hours",* Sams Publishing

[12] **Date, C,** *"An Introduction to Database Systems",* Pearson

[13] **Darwen, H., Date, C.J., Fagin, R.,** *"A normal form for preventing redundant tuples In relational databases",* University of Warwick

[14] **Tasa, E.** *"How to implement Fragment-Activity communication by overwriting*

Interface?", Stack Overflow, https://stackoverflow.com/questions/55281865/how-to-implement-fragment-activity-communication-by-overriting-interface

[14] **Tasa, E.** *"Fragment Pager Adapter only displaying last fragment created",* Stack Overflow, https://stackoverflow.com/questions/55196401/fragment-pager-adapter-only-displaying-last-fragment-created

[14] **Tasa, E.** *"getView() returns null in second activity",* Stack Overflow, https://stackoverflow.com/questions/55010516/getview-returns-null-in-second-activity

8. Annexes

Annex A Application showcase

Annex B Draft Introduction, first round of project objectives

Annex C Data sheet for Nucleo Board

Annex D Laboratory manual for Microcontroller Engineering II

Annex E Sample mark scheme for Microcontroller Engineering II

Annex F Documentation for MarkIt 2017/18

Annex G Showcase of *Wheel of Life* application

Annex H Data Structure for MarkIt

Annex I Showcase of *Karanta!* application

Annex J Explanation of how to set up an Azure server, SQL database and Mobile App

Annex K Sample Mark Scheme in XML format

Annex L Testing

Annex M Risk Assessment

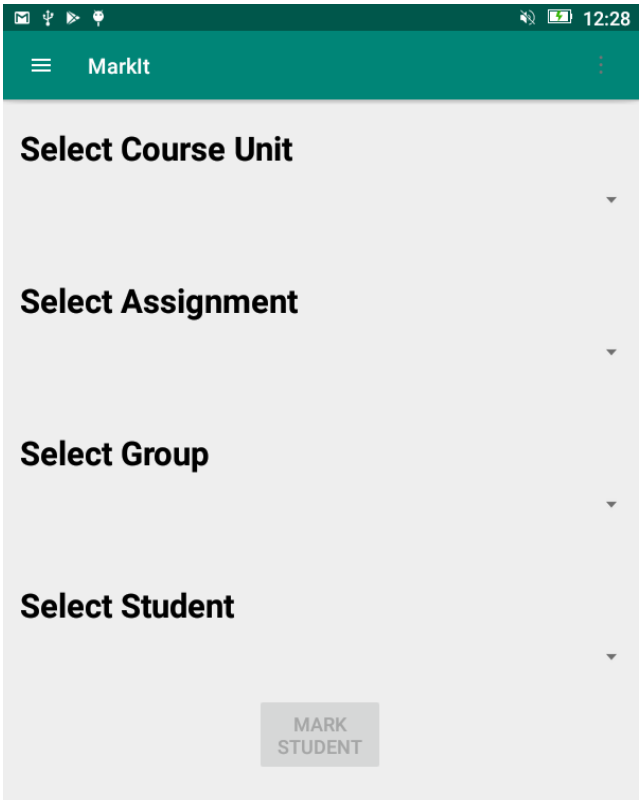
Annex N Project Plan

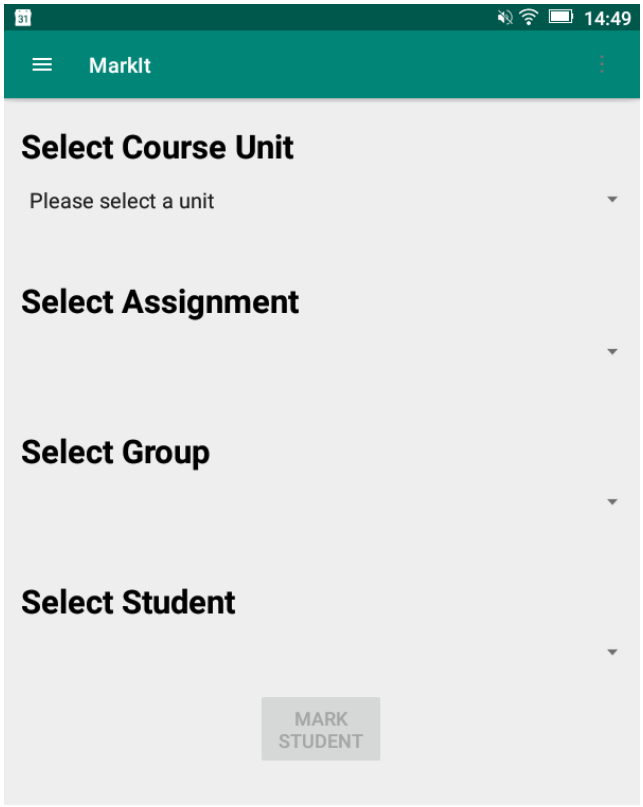
Annex O Application code for MarkIt Android

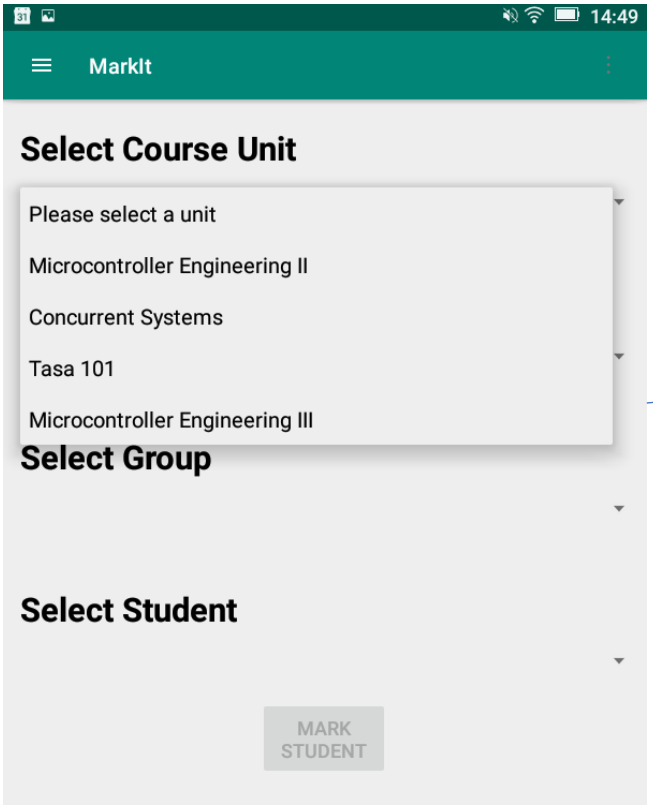
Annex P Application code for MarkIt Windows

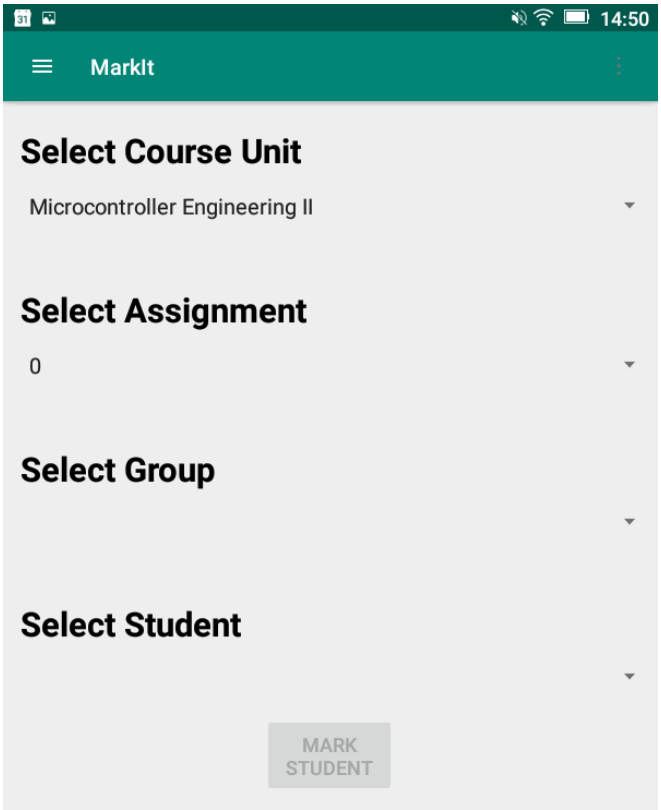
Annex A

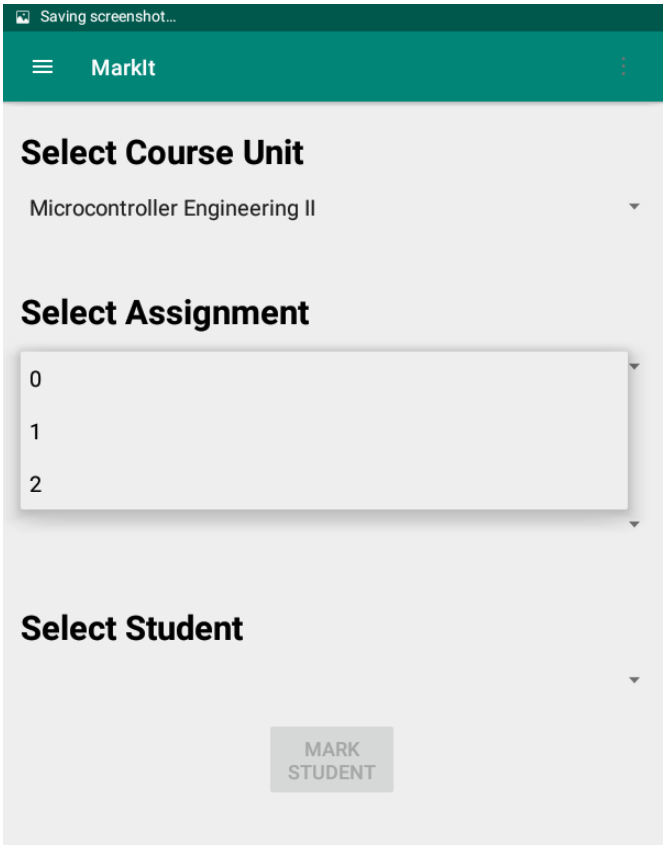
Application showcase

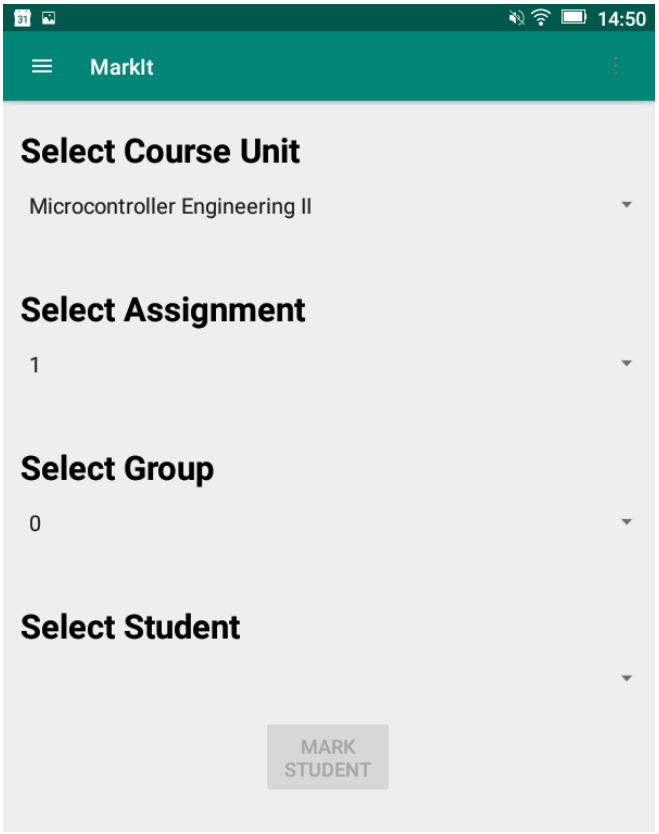

12:28
MarkIt
Select Course Unit
Select Assignment
Select Group
Select Student
MARK STUDENT

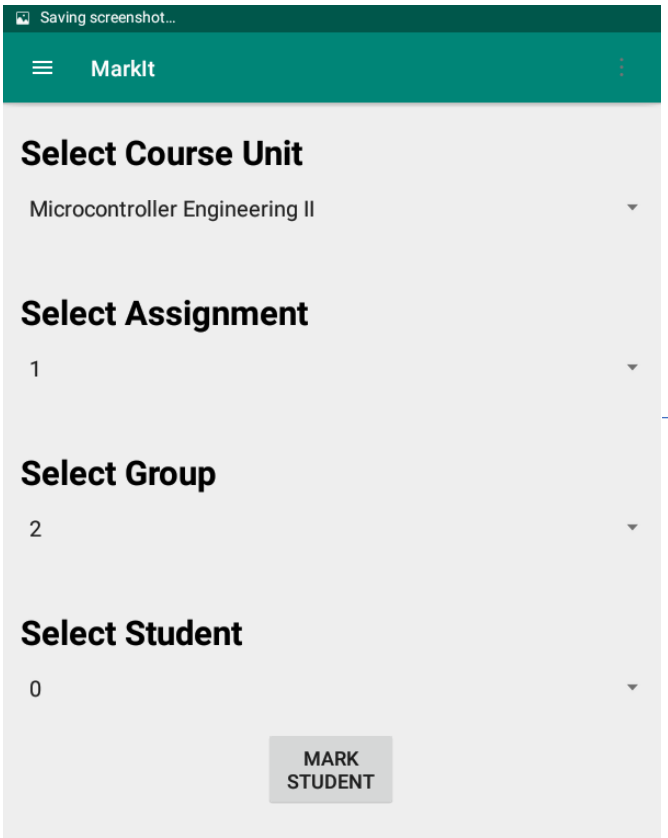

14:49
MarkIt
Select Course Unit
Please select a unit
Select Assignment
Select Group
Select Student
MARK STUDENT

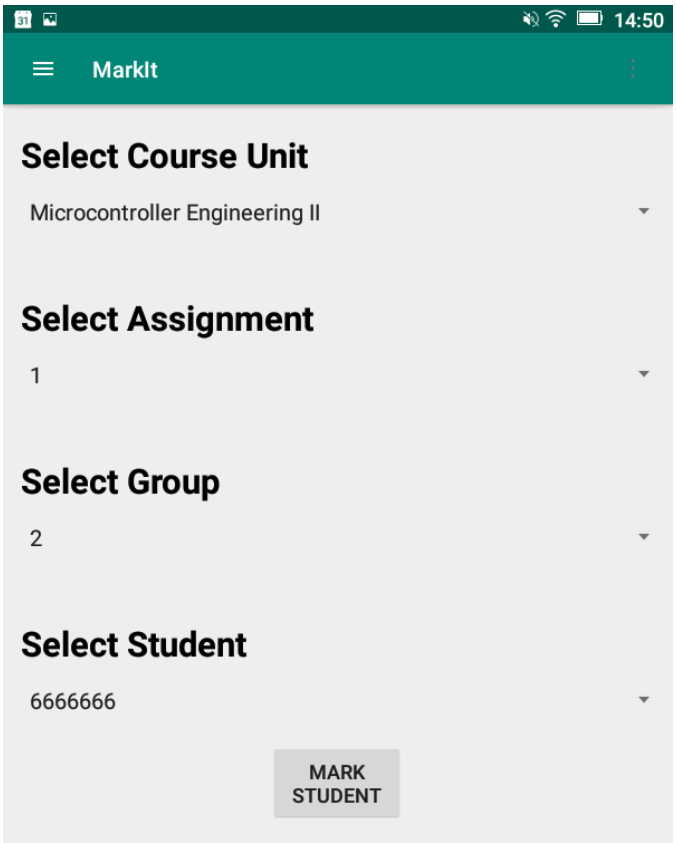

14:49
MarkIt
Select Course Unit
Please select a unit
Microcontroller Engineering II
Concurrent Systems
Tasa 101
Microcontroller Engineering III
Select Group
Select Student
MARK STUDENT


14:50
MarkIt
Select Course Unit
Microcontroller Engineering II
Select Assignment
0
Select Group
Select Student
MARK STUDENT

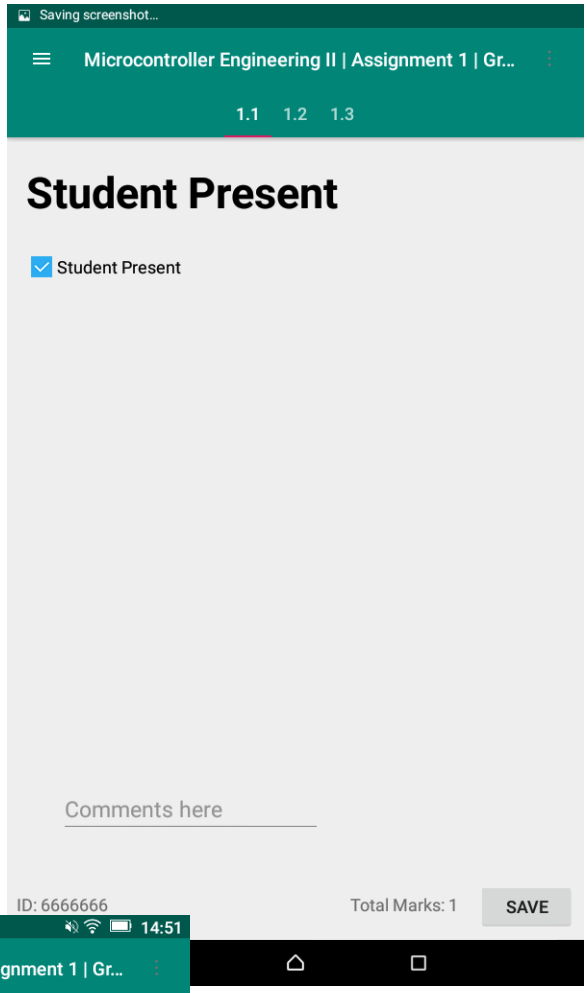

Saving screenshot...
MarkIt
Select Course Unit
Microcontroller Engineering II
Select Assignment
0
1
2
Select Student
MARK STUDENT

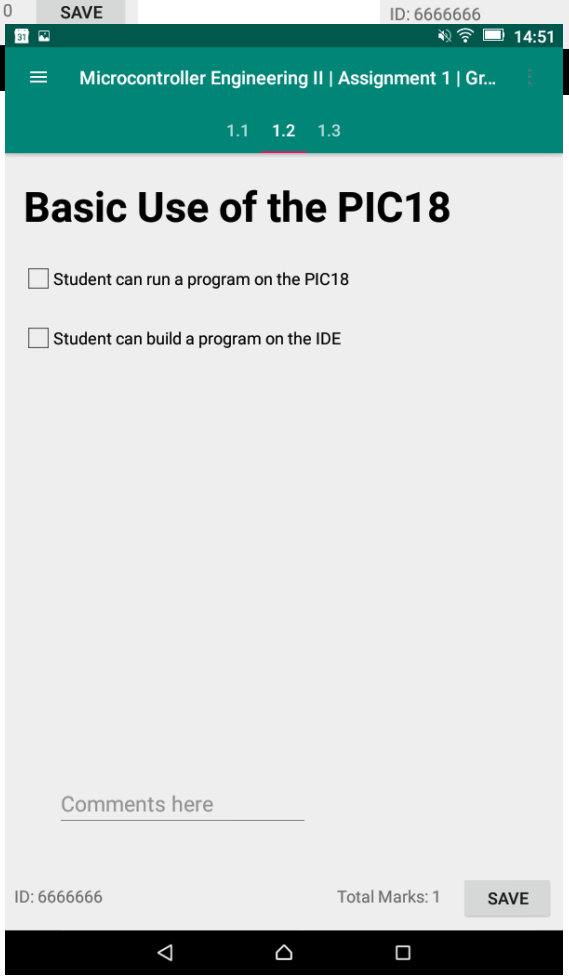

14:50
MarkIt
Select Course Unit
Microcontroller Engineering II
Select Assignment
1
Select Group
0
Select Student
MARK STUDENT

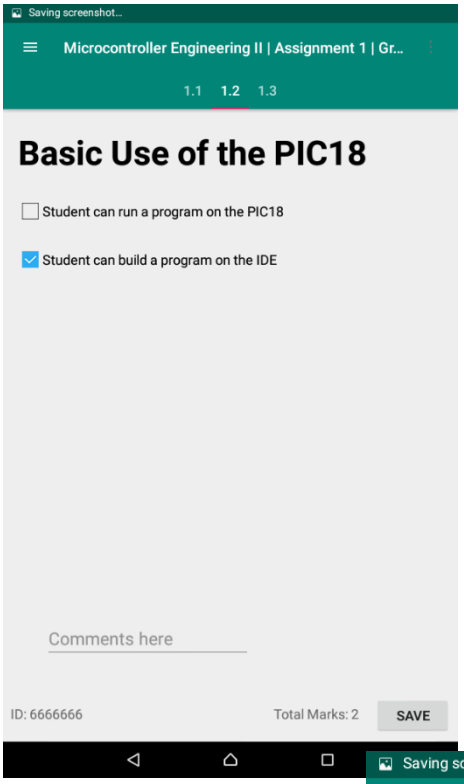

Saving screenshot...
MarkIt
Select Course Unit
Microcontroller Engineering II
Select Assignment
1
Select Group
2
Select Student
0
MARK STUDENT


14:50
MarkIt
Select Course Unit
Microcontroller Engineering II
Select Assignment
1
Select Group
2
Select Student
6666666
MARK STUDENT

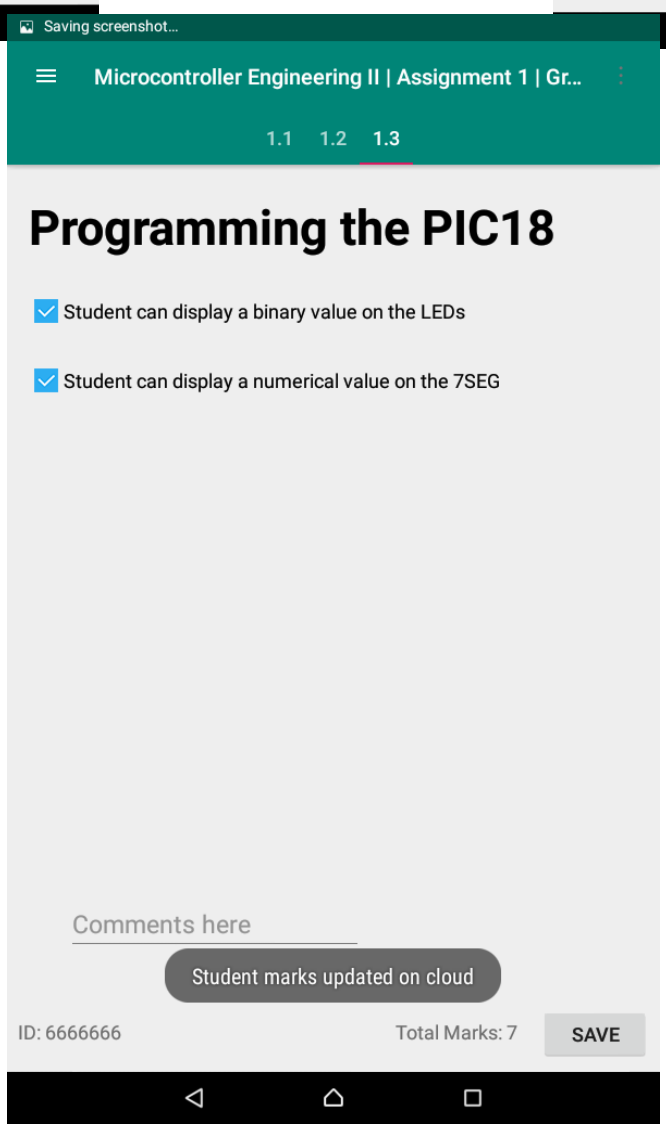

Saving screenshot...
Microcontroller Engineering II | Assignment 1 | Gr...
1.1 1.2 1.3
Student Present
Student Present
Comments here
ID: 6666666
Total Marks: 0
SAVE

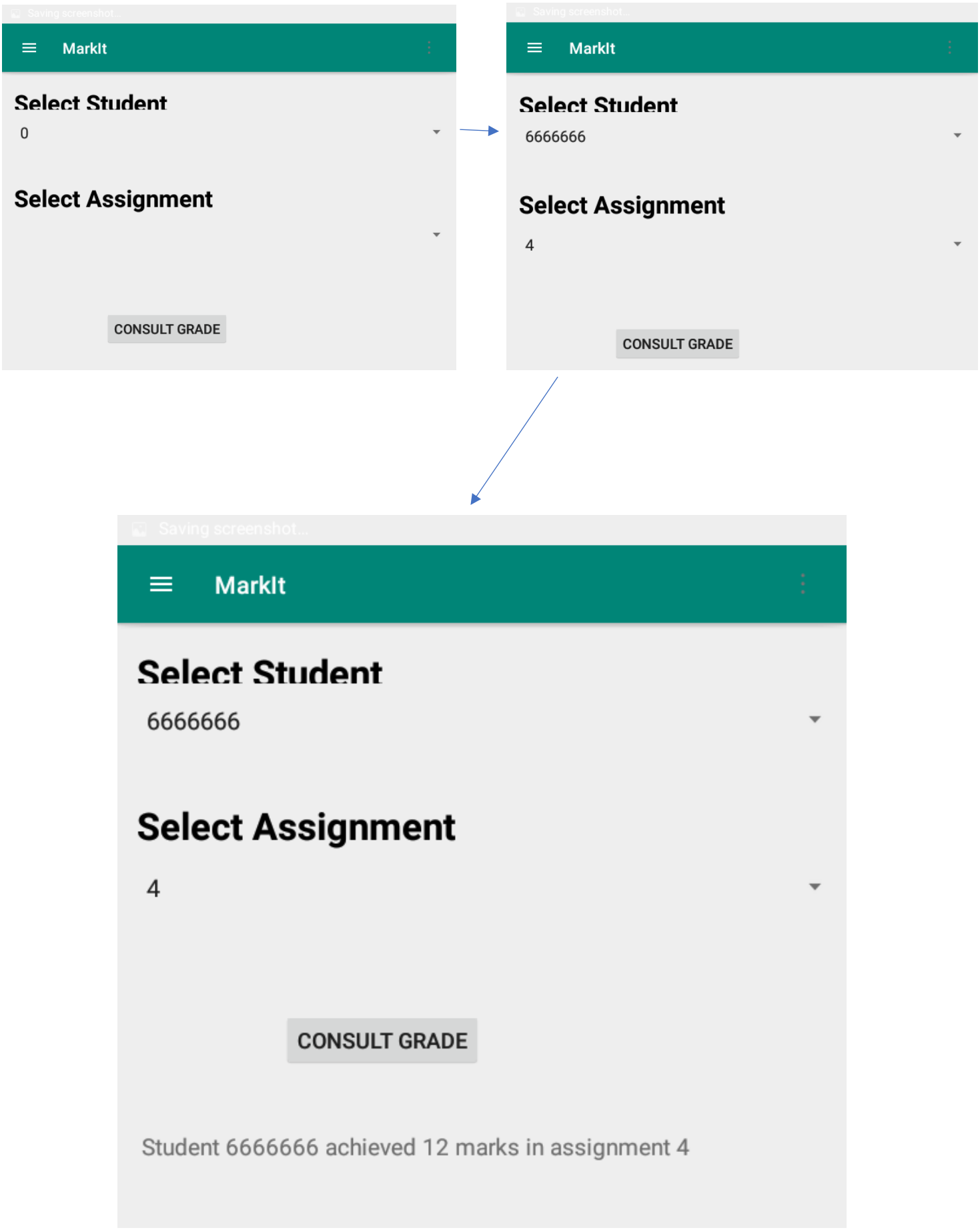

Saving screenshot...
Microcontroller Engineering II | Assignment 1 | Gr...
1.1 1.2 1.3
Student Present
Student Present
Comments here
ID: 6666666
Total Marks: 1
SAVE

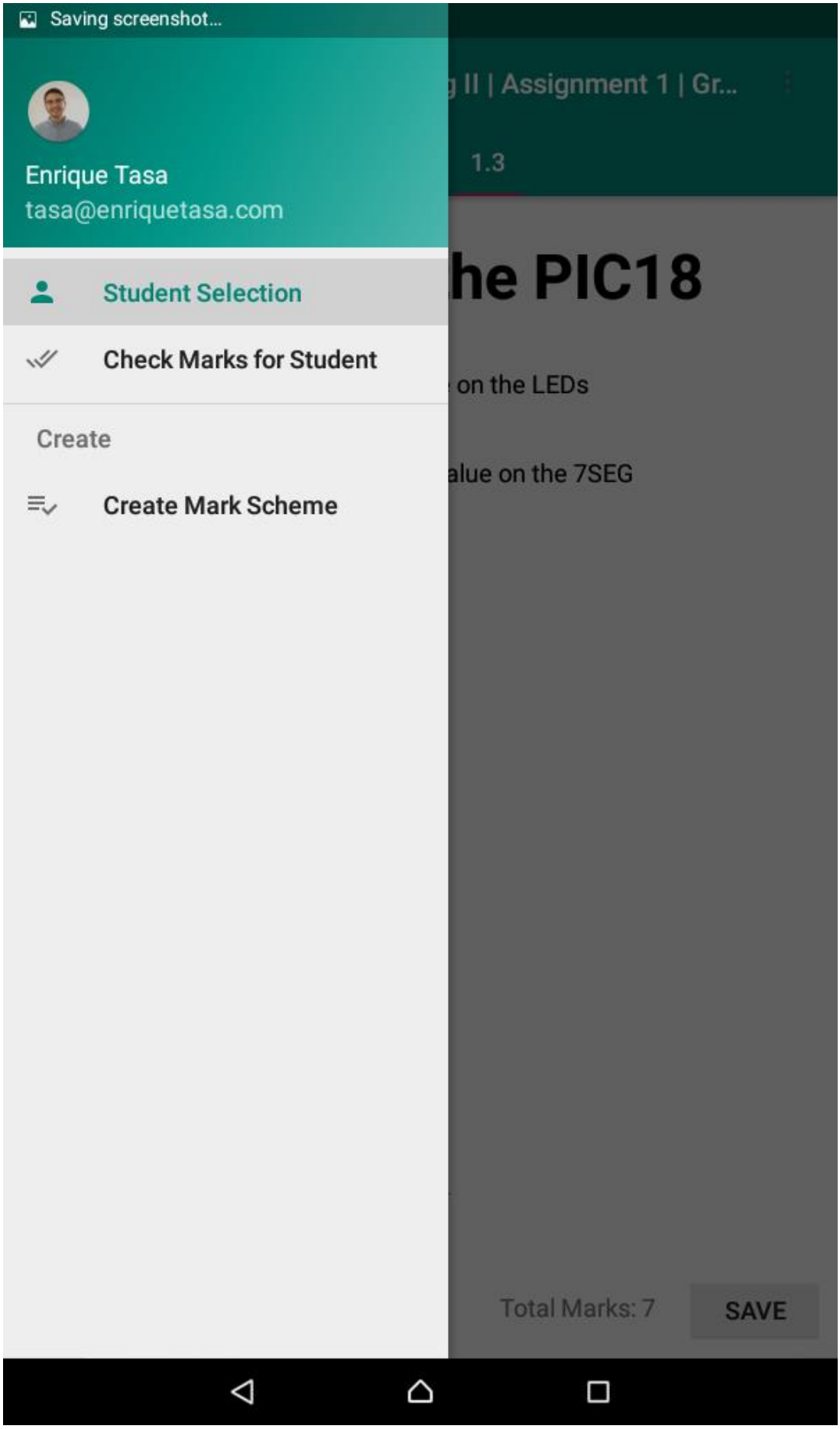

14:51
Microcontroller Engineering II | Assignment 1 | Gr...
1.1 1.2 1.3
Basic Use of the PIC18
Student can run a program on the PIC18
Student can build a program on the IDE
Comments here
ID: 6666666
Total Marks: 1
SAVE

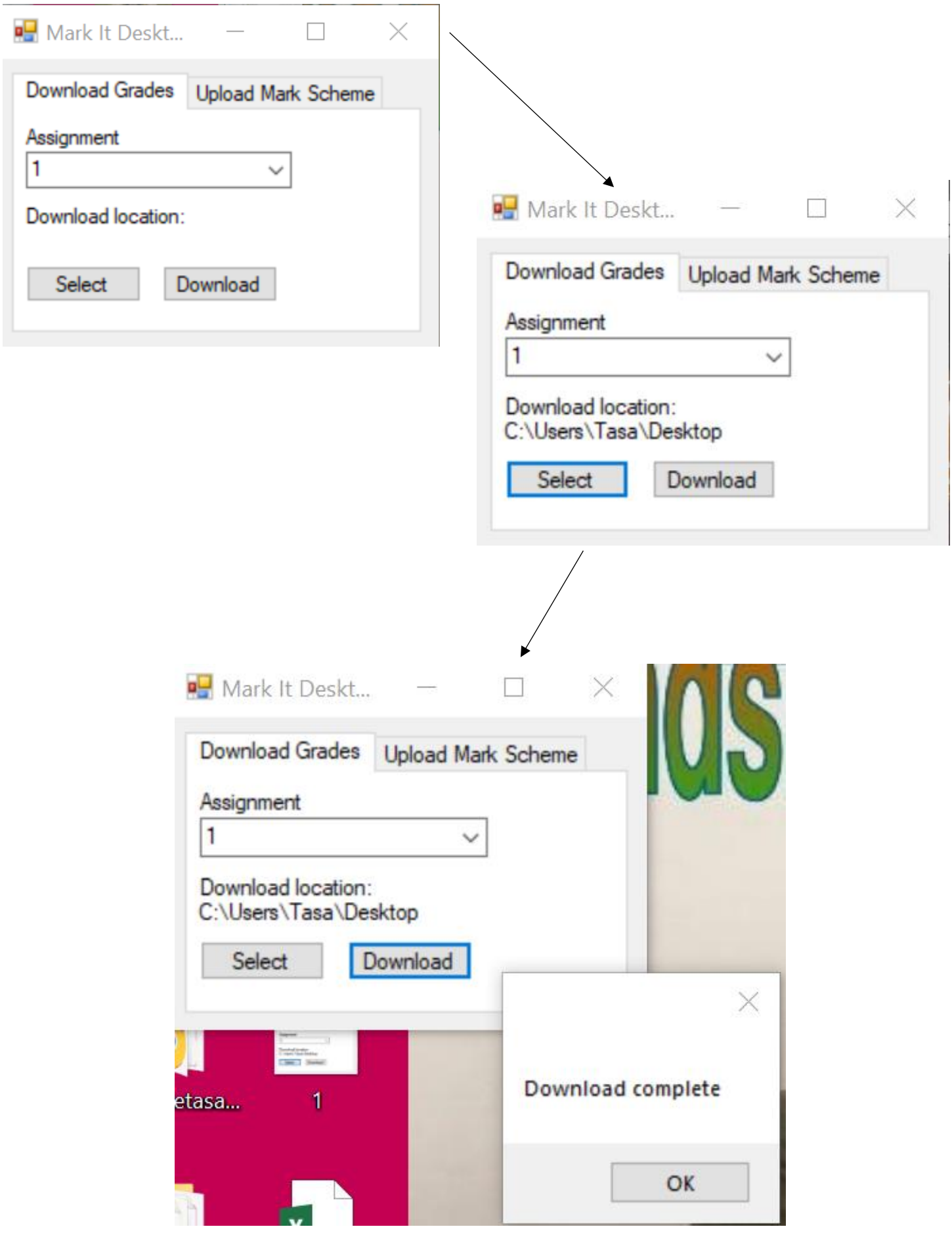

Saving screenshot...
Microcontroller Engineering II | Assignment 1 | Gr...
1.1 1.2 1.3
Basic Use of the PIC18
Student can run a program on the PIC18
Student can build a program on the IDE
Comments here
ID: 6666666
Total Marks: 2
SAVE

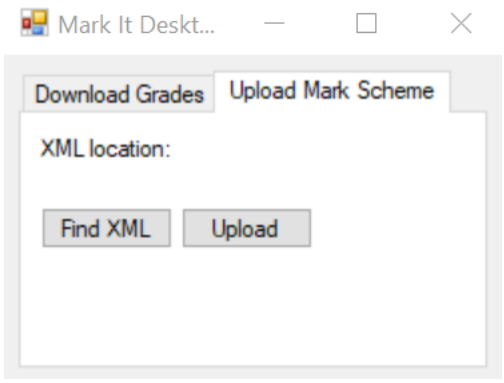

Saving screenshot...
Microcontroller Engineering II | Assignment 1 | Gr...
1.1 1.2 1.3
Programming the PIC18
Student can display a binary value on the LEDs
Student can display a numerical value on the 7SEG
Comments here
ID: 6666666
Total Marks: 2
SAVE

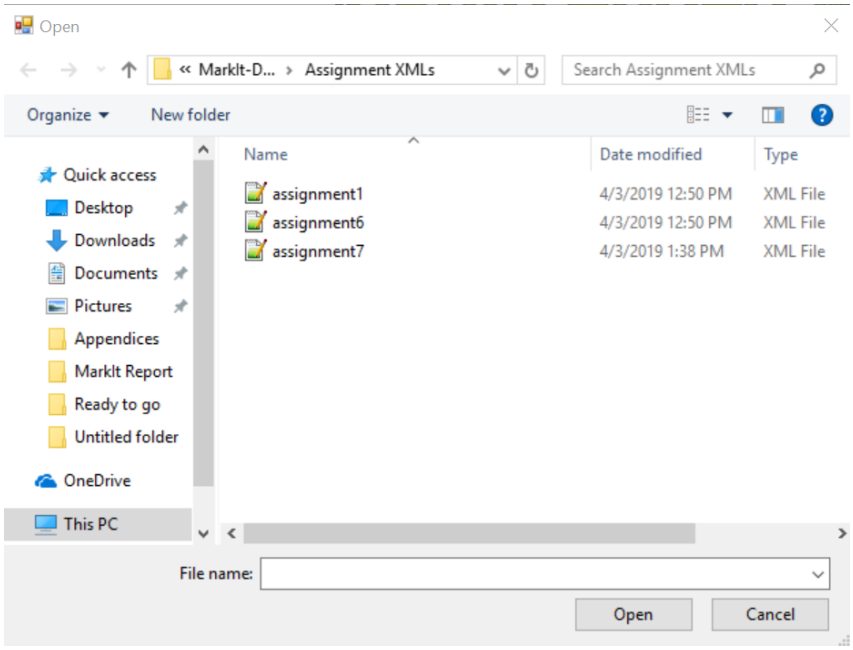

Saving screenshot...
Microcontroller Engineering II | Assignment 1 | Gr...
1.1 1.2 1.3
Programming the PIC18
Student can display a binary value on the LEDs
Student can display a numerical value on the 7SEG
Comments here
Student marks updated on cloud
ID: 6666666
Total Marks: 7
SAVE

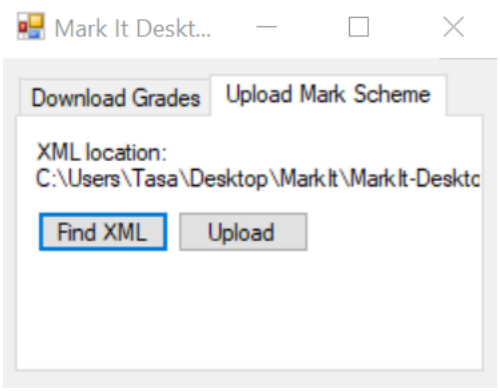

Saving screenshot...
MarkIt
Select Student
0
Select Assignment
CONSULT GRADE
Saving screenshot...
MarkIt
Select Student
6666666
Select Assignment
4
CONSULT GRADE
Saving screenshot...
MarkIt
Select Student
6666666
Select Assignment
4
CONSULT GRADE
Student 6666666 achieved 12 marks in assignment 4

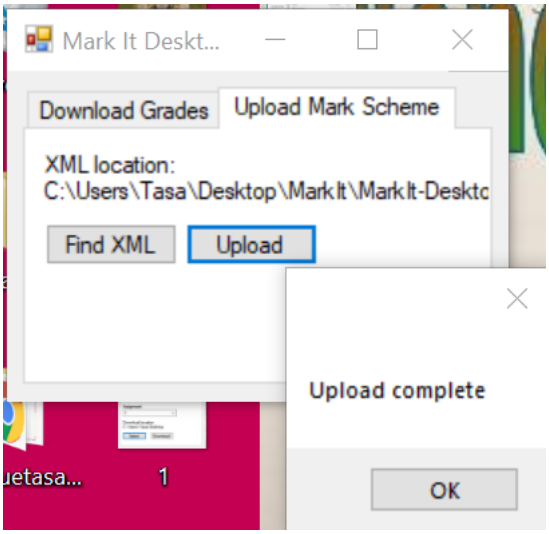

Saving screenshot...
Enrique Tasa
tasa@enriquetasa.com
Student Selection
Check Marks for Student
Create
Create Mark Scheme
II | Assignment 1 | Gr...
1.3
he PIC18
on the LEDs
alue on the 7SEG
Total Marks: 7
SAVE

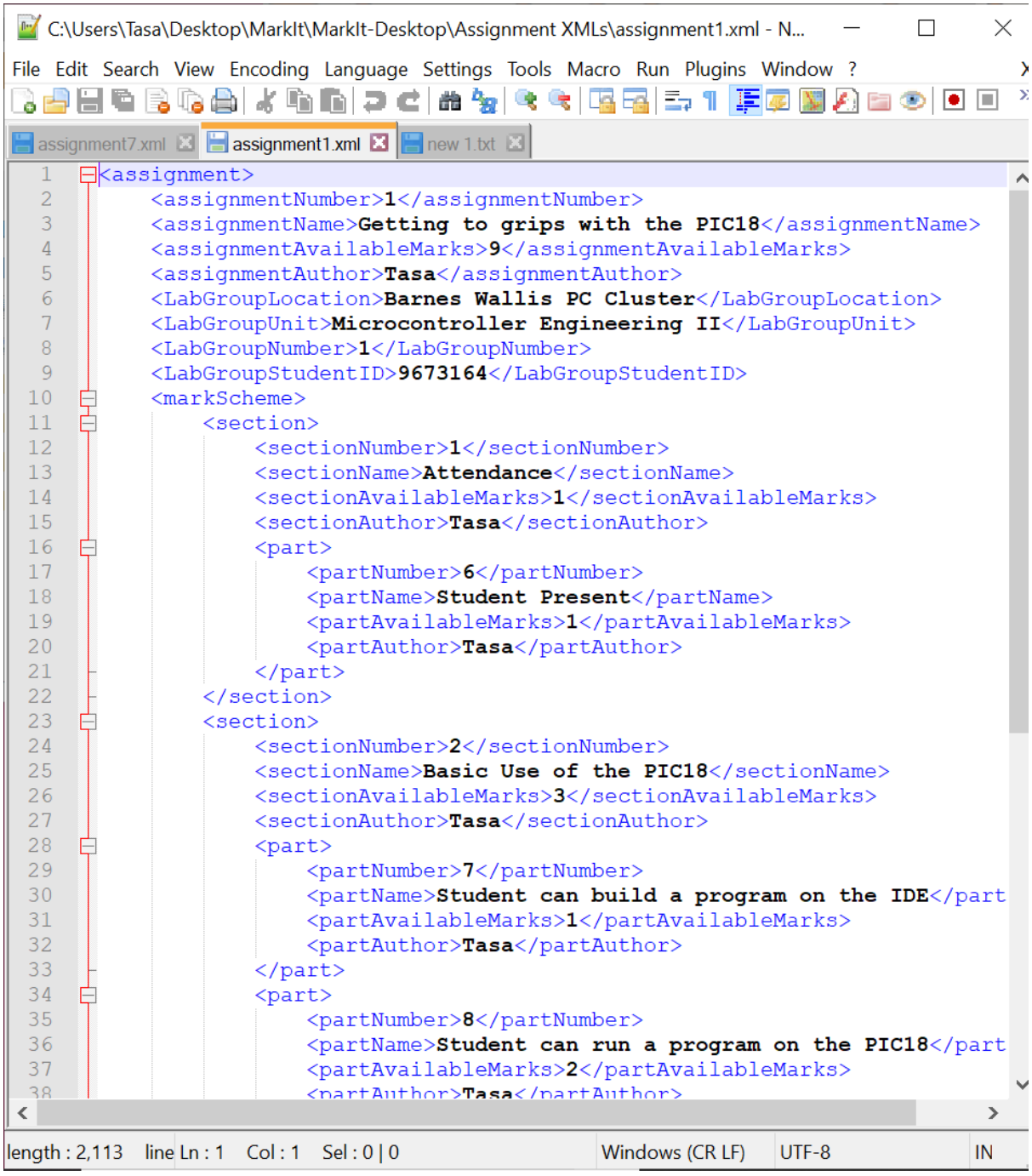

Mark It Deskt...
Download Grades
Upload Mark Scheme
Assignment
1
Download location:
Select
Download
Mark It Deskt...
Download Grades
Upload Mark Scheme
Assignment
1
Download location:
C:\Users\Tasa\Desktop
Select
Download
Mark It Deskt...
Download Grades
Upload Mark Scheme
Assignment
1
Download location:
C:\Users\Tasa\Desktop
Select
Download
etasa...
1
Download complete
OK


Mark It Deskt...
Download Grades
Upload Mark Scheme
XML location:
Find XML
Upload


Open
« MarkIt-D... › Assignment XMLs
Search Assignment XMLs
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
Appendices
MarkIt Report
Ready to go
Untitled folder
OneDrive
This PC
Name
Date modified
Type
assignment1
4/3/2019 12:50 PM
XML File
assignment6
4/3/2019 12:50 PM
XML File
assignment7
4/3/2019 1:38 PM
XML File
File name:
Open
Cancel


Mark It Deskt...
Download Grades
Upload Mark Scheme
XML location:
C:\Users\Tasa\Desktop\MarkIt\MarkIt-Desktc
Find XML
Upload


Mark It Deskt...
Download Grades
Upload Mark Scheme
XML location:
C:\Users\Tasa\Desktop\MarkIt\MarkIt-Desktc
Find XML
Upload
Upload complete
OK
uetasa...
1

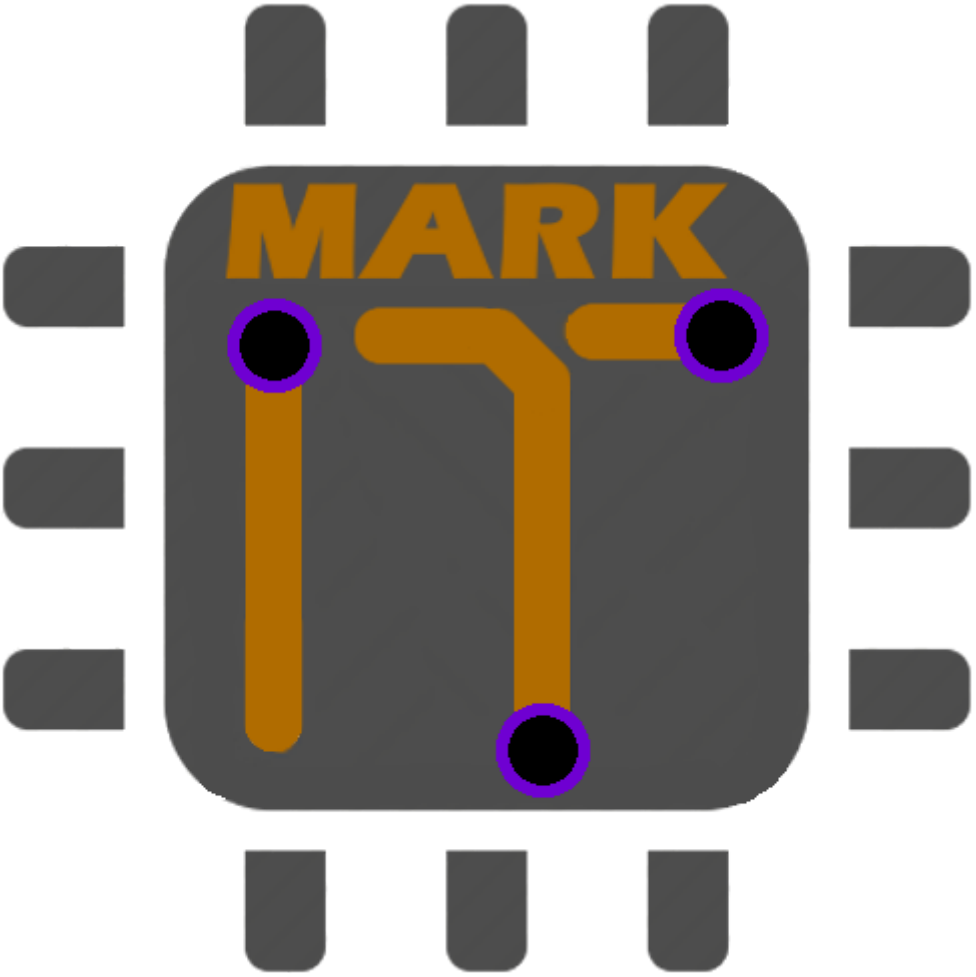

C:\Users\Tasa\Desktop\MarkIt\MarkIt-Desktop\Assignment XMLs\assignment1.xml - N...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
assignment7.xml
assignment1.xml
new 1.txt
<assignment>
<assignmentNumber>1</assignmentNumber>
<assignmentName>Getting to grips with the PIC18</assignmentName>
<assignmentAvailableMarks>9</assignmentAvailableMarks>
<assignmentAuthor>Tasa</assignmentAuthor>
<LabGroupLocation>Barnes Wallis PC Cluster</LabGroupLocation>
<LabGroupUnit>Microcontroller Engineering II</LabGroupUnit>
<LabGroupNumber>1</LabGroupNumber>
<LabGroupStudentID>9673164</LabGroupStudentID>
<markScheme>
<section>
<sectionNumber>1</sectionNumber>
<sectionName>Attendance</sectionName>
<sectionAvailableMarks>1</sectionAvailableMarks>
<sectionAuthor>Tasa</sectionAuthor>
<part>
<partNumber>6</partNumber>
<partName>Student Present</partName>
<partAvailableMarks>1</partAvailableMarks>
<partAuthor>Tasa</partAuthor>
</part>
</section>
<section>
<sectionNumber>2</sectionNumber>
<sectionName>Basic Use of the PIC18</sectionName>
<sectionAvailableMarks>3</sectionAvailableMarks>
<sectionAuthor>Tasa</sectionAuthor>
<part>
<partNumber>7</partNumber>
<partName>Student can build a program on the IDE</part
<partAvailableMarks>1</partAvailableMarks>
<partAuthor>Tasa</partAuthor>
</part>
<part>
<partNumber>8</partNumber>
<partName>Student can run a program on the PIC18</part
<partAvailableMarks>2</partAvailableMarks>
length : 2,113 line Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF) UTF-8 IN

Annex B

Draft Introduction

Draft Introduction

Mark It App - 3rd Year Project



Enrique Tasa Sanchis
9673164
University Of Manchester
November 2018
Supervisor: Dr Peter N Green

Contents

Introduction and background

This third year project involves taking on the development of further features for *Mark It*, an Android application developed by George Wellard as his third year project in the previous academic year. More specifically, this year's development focus is the implementation of a cloud-based data architecture to bring the application's features to a level that allows it to be used during laboratories for Microcontroller Engineering II.

Mark It was developed during the 2017 - 2018 academic year, under the supervision of Dr Peter N Green. The application was developed in the scope of a 3rd year project and further work was done during the summer in developing user guides. The purpose of the application is to eliminate the need for paper marking sheets during laboratories for the Microcontroller Engineering II subject by making the entire marking process digital through an Android application.

In order to do this, a solution was developed that included the following features:

- Creation of a mark scheme via a Windows desktop app (written in C#), which is then uploaded to the application to provide a framework for marking
- Input of an excel sheet containing the names and student IDs of students to be marked
- User interface allowing selection of student being evaluated and input of marks out of a maximum taken from the mark scheme
- Option to add comments to students marks
- Output of students/mark data as a csv file to then be uploaded to BlackBoard

The project was successful in developing an application that could perform the task of marking students work, but it did not provide a full solution that could be implemented in laboratories, as various technical challenges came up. Most notably, various examiners do marking at the same time, and given that the application saved data locally on the device it was running on, it was impossible for it to be used effectively in labs.

Because of this, this year the aim of the project is to implement a cloud based data storage architecture to allow multiple user simultaneous interaction with the database of student marks, so that examiners can use the application in the real labs. In the following pages a description of the concrete aims and motivation for the project is described, as well as a detailed breakdown of the specific objectives for the year and an update on the current state of the project.

Aims

The aim of this year's project is to develop the current functionality of the application further to include cloud based data storage and simultaneous access to the marks data, in order to support various simultaneous sessions of the application being run by the various demonstrators present at a lab. Once this has been achieved, further aims include the development of more features such as a way for students to consult their marks or the cloud save of mark schemes for repeated use.

Motivation

The motivation behind this project is to make the marking process for laboratories more efficient and effective. By eliminating paper mark schemes and transitioning to a fully digital solution, we can ensure that there will be less mistakes in accurately recording marks (as demonstrators need only to input a number to an app and get visual confirmation, rather than reading other people's handwriting and passing values to an excel sheet), we can provide a simple way for students to quickly consult their marks by asking a demonstrator who only needs to pull up an application to provide the recorded marks and we can make the whole process more efficient and environmentally friendly as less paper is printed and transported around.

Furthermore, if development goes well, the application could become a framework for marking that could be extrapolated to other subjects or situations.

Objectives

There are two sets of concrete objectives that have been defined for this project: base objectives, goals that are seen as the basics for the project to be deemed as completed; and stretch objectives, goals that would take Mark It further in providing a complete solution but that will only be attempted once the base objectives are completed.

Base Objectives

- Implement a cloud-base storage system for the data collected by the application
- Ensure that various users can access the data simultaneously, reading/writing at the same time
- Allow users to define mark scheme structures in the mobile app
- Allow users to have saved mark schemes in the cloud

Stretch Objectives

- Implement a system to allow users to save previous mark schemes and use them repeatedly
- Implement a feature to allow users to consult student's grades for the current laboratory in real time
- Implement a feature to allow users to consult student's grades for the all previous laboratories
- Make a web interface which includes all current mobile app functionality
- Rewrite the application's "Assignment" and "Referrals" classes

Literature review

The only existing literature concerning Mark It is the project report submitted by the student in charge of the project last year. In it, a comprehensive exploration of the whole project is presented, going from the motivation for the creation of the app to the steps taken in the development of the base code.

Other than the project report, various documentation for technologies related with the development have been consulted so far, specifically documentation for Google Firebase, Google Flutter, Android and the Android Development Studio and the Java programming language. The documentation has been consulted in order to clarify or expand on the existing code.

Current state of the project

Up to now, most of the work done for the project has been compromised of planning and preparation. The student has not yet worked on the Mark It code in order to implement any features. This is not ideal, and the student realises that it is now time for actual development work to begin.

The reasons for this delay mostly concern the workload of the student: in the first weeks of the semester the student started thinking of ideas for the project and planning on how to execute them, and when the time to begin making them happen has come, coursework deadlines have stopped the student from devoting the time needed to the project.

Realising this is the case, the student now aims to organise his time better, and seeing that his last deadline is in a week, aims to begin development of the new features imminently.

Annex C

Nucleo Board User Manual

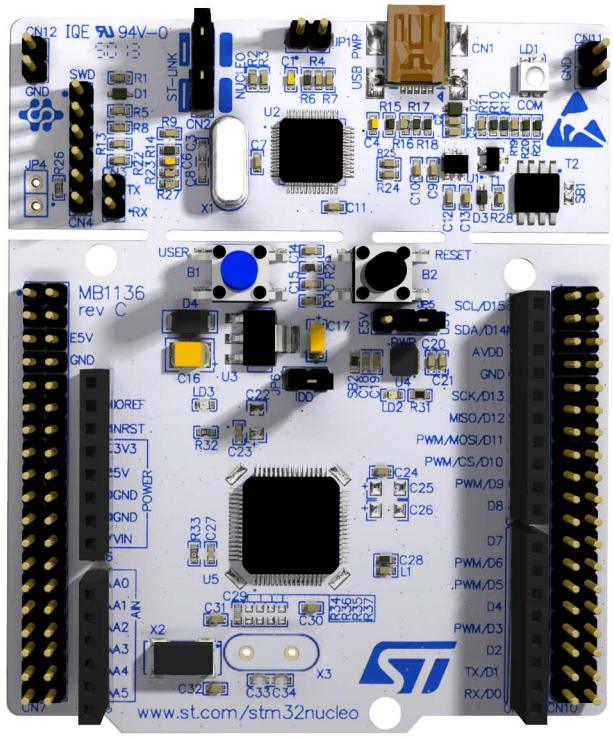
UM1724
User manual

STM32 Nucleo boards

Introduction

The STM32 Nucleo board (NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F103RB, NUCLEO-F302R8, NUCLEO-F303RE, NUCLEO-F334R8, NUCLEO-F401RE, NUCLEO-F411RE, NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE, NUCLEO-L476RG) provides an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller lines, choosing from the various combinations of performance, power consumption and features. The Arduino™ connectivity support and ST Morpho headers make it easy to expand the functionality of the Nucleo open development platform with a wide choice of specialized shields. The STM32 Nucleo board does not require any separate probe as it integrates the ST-LINK/V2-1 debugger/programmer. The STM32 Nucleo board comes with the STM32 comprehensive software HAL library together with various packaged software examples, as well as direct access to mbed online resources at *mbed.org*.

Figure 1. STM32 Nucleo board (1)



1. Picture not contractual.

Contents

List of tables

List of figures

1 Ordering information

Table 1 lists the order codes and the respective targeted MCU.

Table 1. Ordering information

Order code	Targeted MCU
NUCLEO-F030R8	STM32F030R8T6
NUCLEO-F070RB	STM32F070RBT6
NUCLEO-F072RB	STM32F072RBT6
NUCLEO-F091RC	STM32F091RCT6
NUCLEO-F103RB	STM32F103RBT6
NUCLEO-F302R8	STM32F302R8T6
NUCLEO-F303RE	STM32F303RET6
NUCLEO-F334R8	STM32F334R8T6
NUCLEO-F401RE	STM32F401RET6
NUCLEO-F411RE	STM32F411RET6
NUCLEO-L053R8	STM32L053R8T6
NUCLEO-L073RZ	STM32L073RZT6
NUCLEO-L152RE	STM32L152RET6
NUCLEO-L476RG	STM32L476RGT6

The meaning of NUCLEO-TXXXRY codification is as follows:

- TXXX describes the STM32 MCU product line
- R describes the pin count (R for 64 pins)
- Y describes the code size (8 for 64K, B for 128K, C for 256K, E for 512K, G for 1MB, Z for 192K)

The order code is printed on a sticker placed at the top or bottom side of the board.

2 Conventions

Table 2 provides the conventions used for the ON and OFF settings in the present document.

Table 2. ON/OFF conventions

Convention	Definition
Jumper JP1 ON	Jumper fitted
Jumper JP1 OFF	Jumper not fitted
Solder bridge SBx ON	SBx connections closed by solder or 0 ohm resistor
Solder bridge SBx OFF	SBx connections left open

We refer to “STM32 Nucleo board” and “STM32 Nucleo boards” in this document for all information that is common to all sale types.

3 Quick start

The STM32 Nucleo board is a low-cost and easy-to-use development platform used to quickly evaluate and start a development with an STM32 microcontroller in LQFP64 package.

Before installing and using the product, please accept the Evaluation Product License Agreement from *www.st.com/epla*.

For more information on the STM32 Nucleo boards and to access the demonstration software, visit *www.st.com/stm32nucleo.*

3.1 Getting started

Follow the sequence below to configure the STM32 Nucleo board and launch the demo software:

1. Check the jumper position on the board, JP1 off, JP5 (PWR) on U5V, JP6 on (IDD), CN2 on (NUCLEO) selected.
2. For correct identification of all device interfaces from the host PC, install the Nucleo USB driver available on *www.st.com/stm32nucleo*, prior to connecting the board
3. Connect the STM32 Nucleo board to a PC with a USB cable ‘type A to mini-B’ through USB connector CN1 to power the board. The red LED LD3 (PWR) and LD1 (COM) should light up. LD1 (COM) and green LED LD2 should blink.
4. Press button B1 (left button).
5. Observe how the blinking of the green LED LD2 changes according to clicks on button B1.
6. The demo software and several software examples on how use the STM32 Nucleo board features are available on *www.st.com/stm32nucleo*.
7. Develop your own application using the available examples.

3.2 System requirements

- Windows PC (XP, 7, 8)
- USB type A to Mini-B USB cable

4 Features

The STM32 Nucleo boards offer the following features:

- STM32 microcontroller with LQFP64 package
- Two types of extension resources
  - Arduino Uno Revision 3 connectivity
  - STMicroelectronics Morpho extension pin headers for full access to all STM32 I/Os
- mbed-enabled[a]
- On-board ST-LINK/V2-1 debugger/programmer with SWD connector
  - selection-mode switch to use the kit as a standalone ST-LINK/V2-1
- Flexible board power supply
  - USB VBUS
  - External VIN (7V<VIN<12V) supply voltage from Arduino connectors or ST Morpho connector
  - External 5V (E5V) supply voltage from ST Morpho connector
  - External +3.3V supply voltage from Arduino connector or ST Morpho connector
- Three LEDs
  - USB communication (LD1), user LED (LD2), power LED (LD3)
- Two push buttons: USER and RESET
- LSE crystal:
  - 32.768kHz crystal oscillator (depending on board version)
- USB re-enumeration capability: three different interfaces supported on USB
  - Virtual Com port
  - Mass storage
  - Debug port
- Comprehensive free software HAL library including a variety of software examples
- Supported by wide choice of Integrated Development Environments (IDEs) including IAR, Keil, GCC-based IDEs

4.1 Hardware configuration variants

The board can be delivered with different configurations of the oscillator of the target MCU. For all the details concerning High Speed oscillator configurations refer to *Section 5.7.1*. For all the details concerning Low speed oscillator configurations refer to *Section 5.7.2*.

a. See *http://mbed.org/*

5 Hardware layout and configuration

The STM32 Nucleo board is designed around the STM32 microcontrollers in a 64-pin LQFP package.

Figure 2 shows the connections between the STM32 and its peripherals (ST-LINK/V2-1, pushbutton, LED, Arduino connectors and STMicroelectronics Morpho connector).

Figure 3 and *Figure 4* show the location of these features on the STM32 Nucleo board.

Figure 2. Hardware block diagram

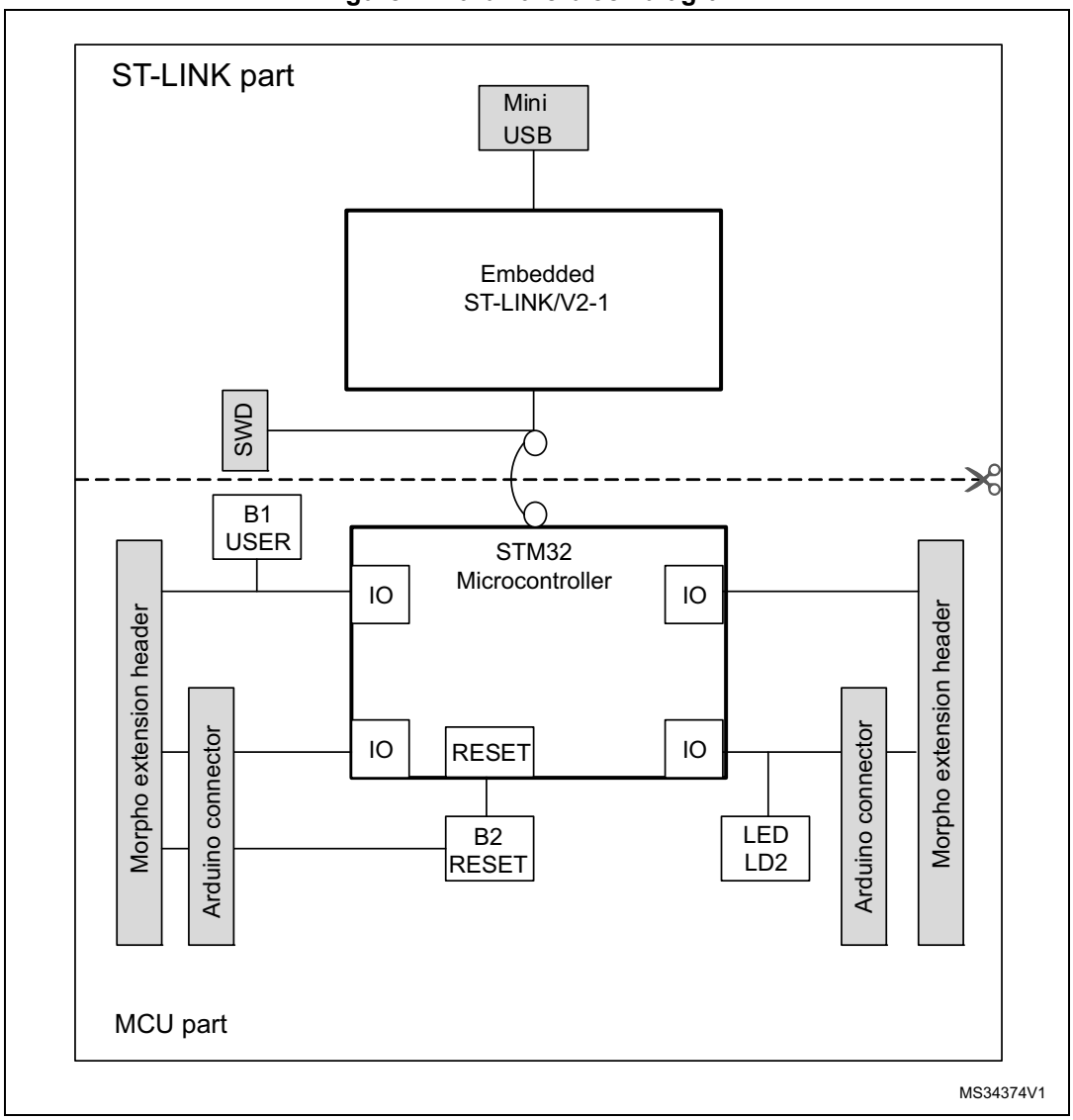
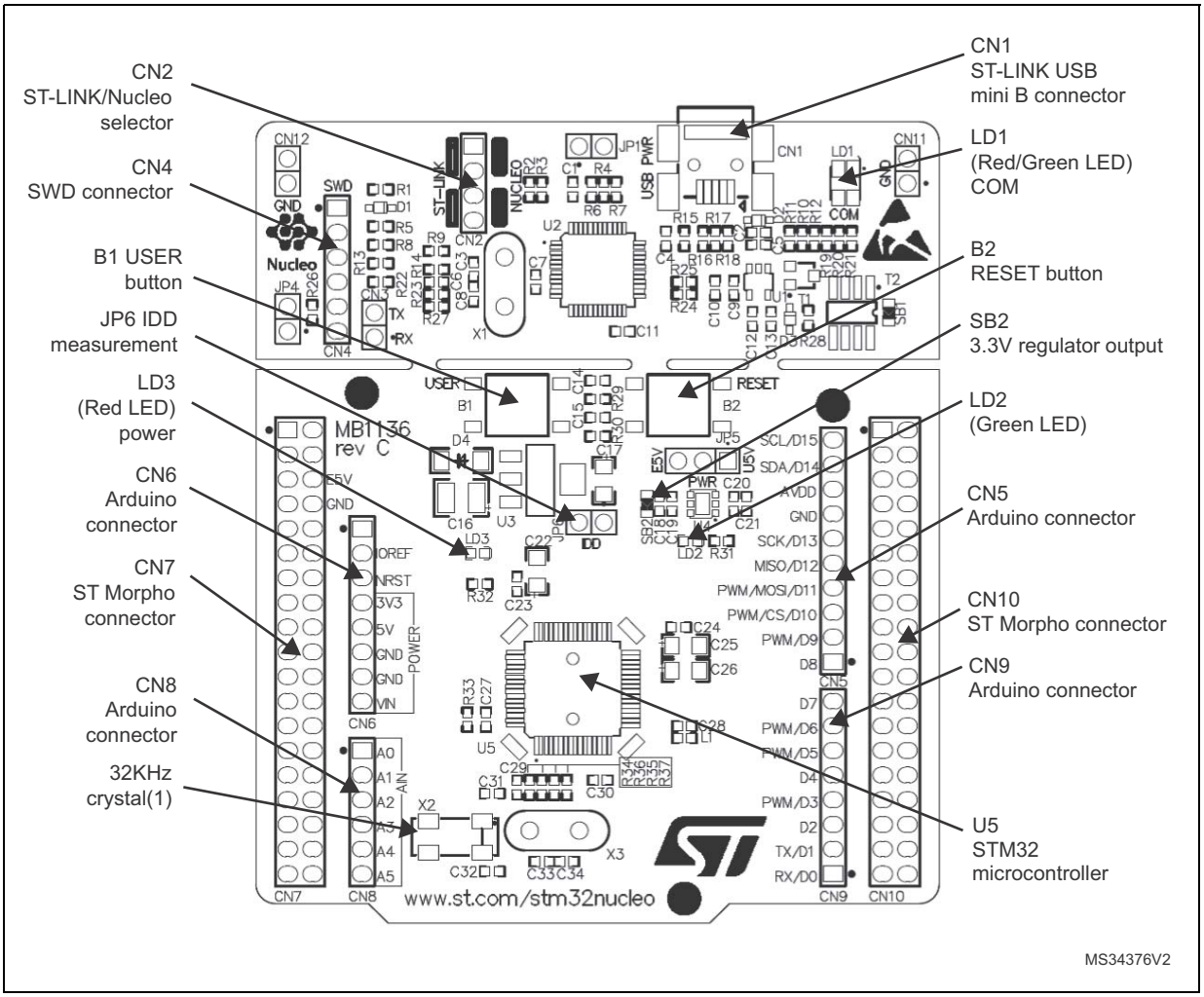
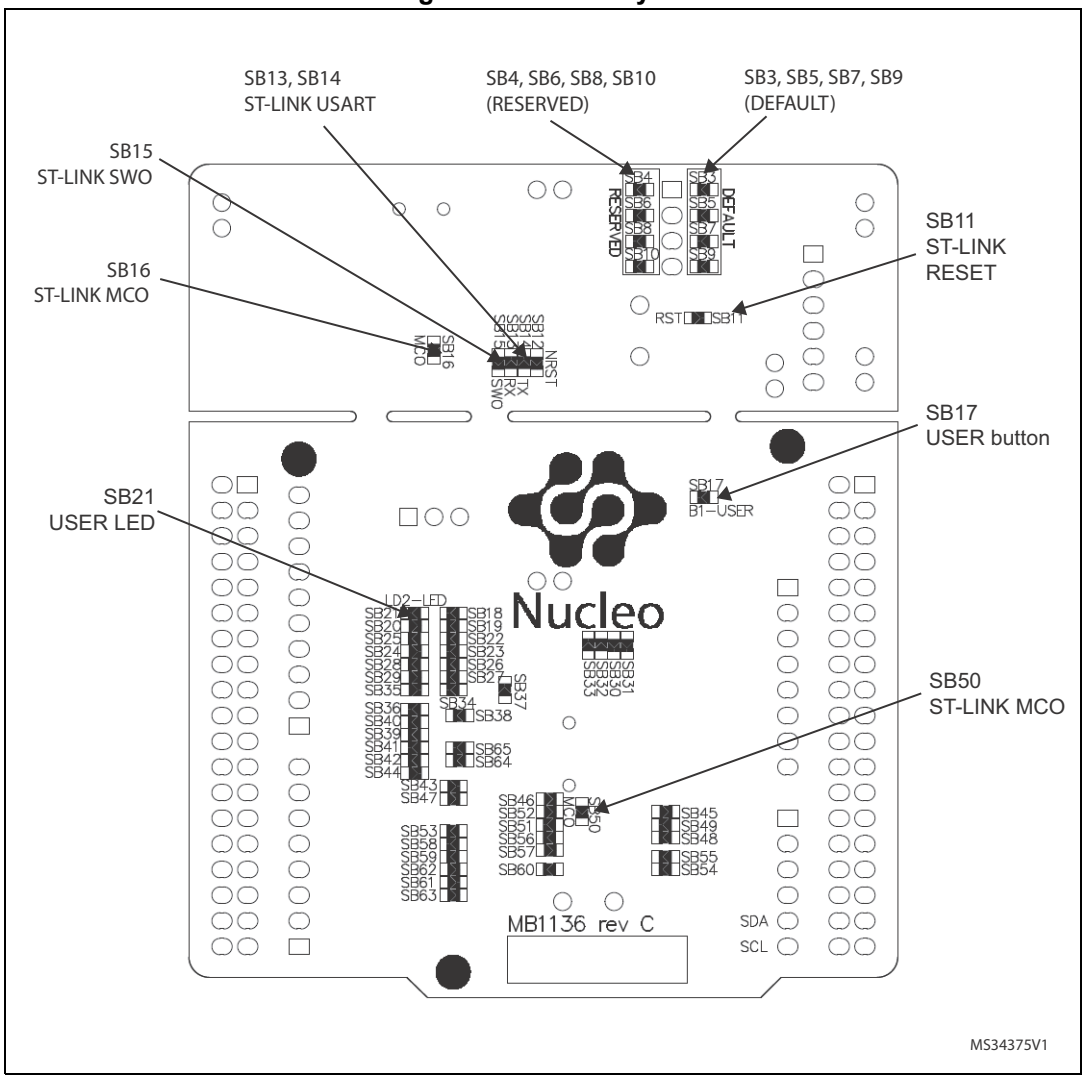


Figure 3. Top layout



1. Crystal may be present or not depending on board version, refer to *Section 5.7.2*

Figure 4. Bottom layout



5.1 Cutable PCB

The STM32 Nucleo board is divided into two parts: ST-LINK part and target MCU part. The ST-LINK part of the PCB can be cut out to reduce the board size. In this case the remaining target MCU part can only be powered by VIN, E5V and 3.3V on STMicroelectronics Morpho connector CN7 or VIN and 3.3V on Arduino connector CN6. It is still possible to use the ST-LINK part to program the main MCU using wires between CN4 and SWD signals available on STMicroelectronics Morpho connector (SWCLK CN7 pin 15 and SWDIO CN7 pin 13).

5.2 Embedded ST-LINK/V2-1

The ST-LINK/V2-1 programming and debugging tool is integrated in the STM32 Nucleo boards.

The ST-LINK/V2-1 makes the STM32 Nucleo boards mbed enabled.

The embedded ST-LINK/V2-1 supports only SWD for STM32 devices. For information about debugging and programming features refer to *UM1075 - ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32, User manual*, which describes in detail all the ST-LINK/V2 features.

The changes versus ST-LINK/V2 version are listed below.

- New features supported on ST-LINK/V2-1:
  - USB software re-enumeration
  - Virtual com port interface on USB
  - Mass storage interface on USB
  - USB power management request for more than 100 mA power on USB
- Features not supported on ST-LINK/V2-1:
  - SWIM interface
  - Minimum supported application voltage limited to 3 V
- Known limitation:
  - Activating the readout protection on ST-Link/V2-1 target prevents the target application from running afterwards. The target readout protection must be kept disabled on ST-Link/V2-1 boards.

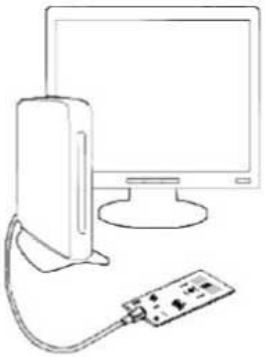
There are two different ways to use the embedded ST-LINK/V2-1 depending on the jumper states (see *Table 3* and *Figure 5*):

- Program/debug the MCU on board (*Section 5.2.2*),
- Program/debug an MCU in an external application board using a cable connected to SWD connector CN4 (*Section 5.2.4*).

Table 3. Jumper states

Jumper state	Description
Both CN2 jumpers ON	ST-LINK/V2-1 functions enabled for on board programming (default)
Both CN2 jumpers OFF	ST-LINK/V2-1 functions enabled for external CN4 connector (SWD supported)

Figure 5. Typical configuration



Hardware requirements:
- USB cable type A to mini-B
- computer with Windows XP, 7, 8

Development toolchain:
- IAR EWARM
- Keil MDK-ARM
- GCC-based IDE

MS19052V4

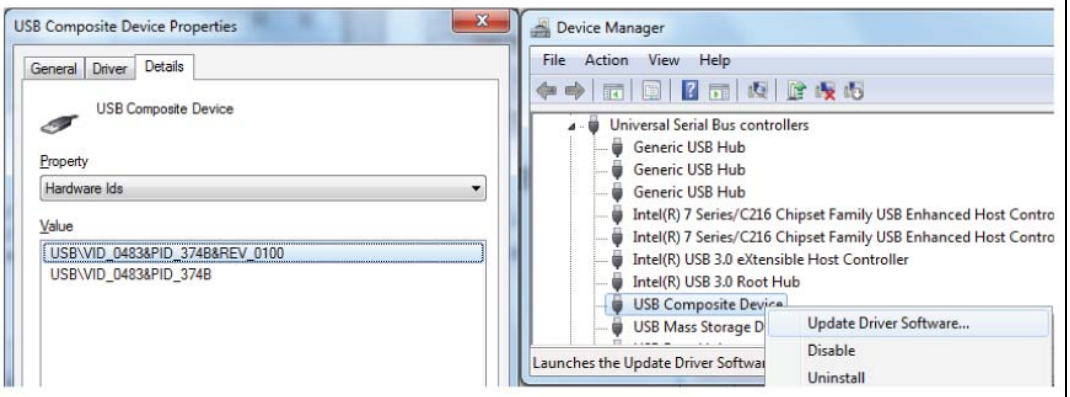
5.2.1 Drivers

The ST-LINK/V2-1 requires a dedicated USB driver, which can be found on *www.st.com* for Windows XP, 7, 8. On Windows XP the ST-LINK/V2-1 driver requires WinUsb to be installed before using the ST-LINK/V2-1 (either available from Microsoft website or included in the USB driver for ST-LINK/V2 for XP).

In case the STM32 Nucleo board is connected to the PC before the driver is installed, some Nucleo interfaces may be declared as “Unknown” in the PC device manager. In this case the user must install the driver files (*Figure 6*), and from the device manager update the driver of the connected device.

Note: Prefer using the “USB Composite Device” handle for a full recovery.

Figure 6. Updating the list of drivers in Device Manager



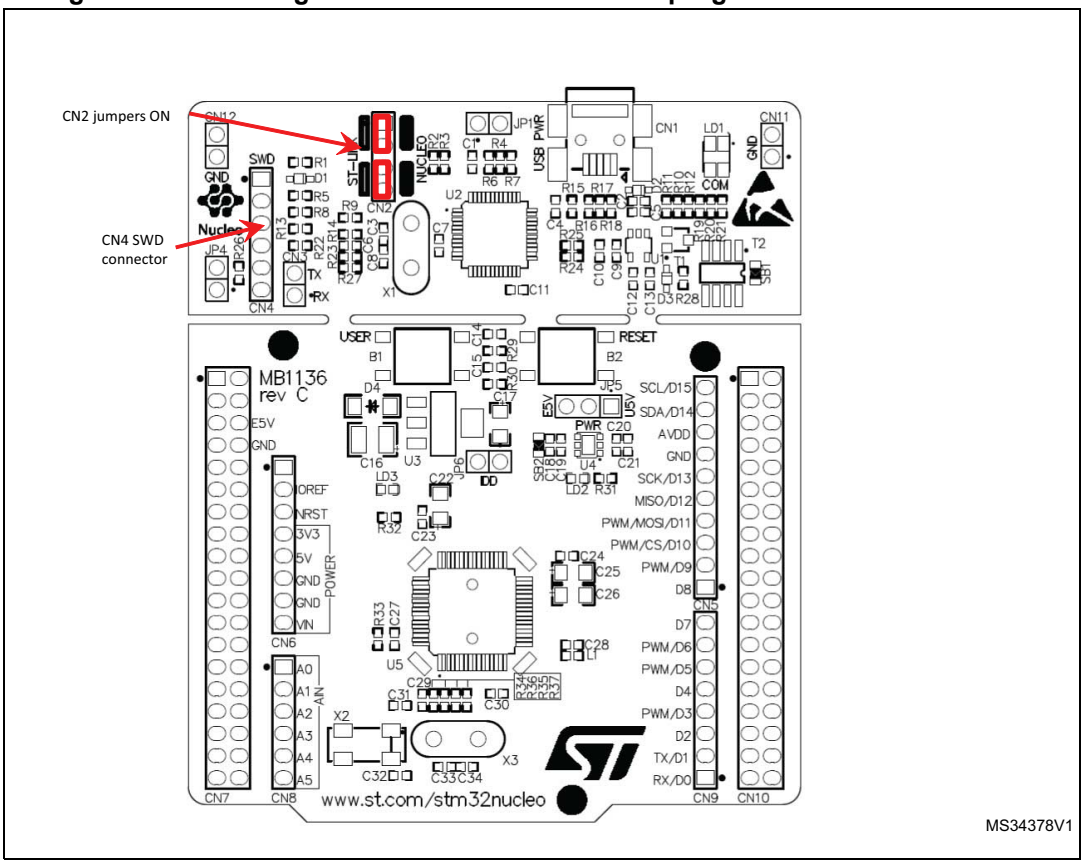
5.2.2 ST-LINK/V2-1 firmware upgrade

The ST-LINK/V2-1 embeds a firmware upgrade mechanism for in-situ upgrade through the USB port. As the firmware may evolve during the life time of the ST-LINK/V2-1 product (for example new functionality, bug fixes, support for new microcontroller families), it is recommended to visit *www.st.com* before starting to use the STM32 Nucleo board and periodically, in order to stay up-to-date with the latest firmware version.

5.2.3 Using the ST-LINK/V2-1 to program/debug the STM32 on board

To program the STM32 on the board, plug in the two jumpers on CN2, as shown in red in *Figure 7*. Do not use the CN4 connector as this could disturb the communication with the STM32 microcontroller of the STM32 Nucleo board.

Figure 7. Connecting the STM32 Nucleo board to program the on-board STM32



5.2.4 Using ST-LINK/V2-1 to program/debug an external STM32 application

It is very easy to use the ST-LINK/V2-1 to program the STM32 on an external application. Simply remove the two jumpers from CN2 as illustrated in *Figure 8*, and connect your application to the CN4 debug connector according to *Table 4*.

Note: SB12 NRST (target MCU RESET) must be OFF if you use CN4 pin 5 in your external application.

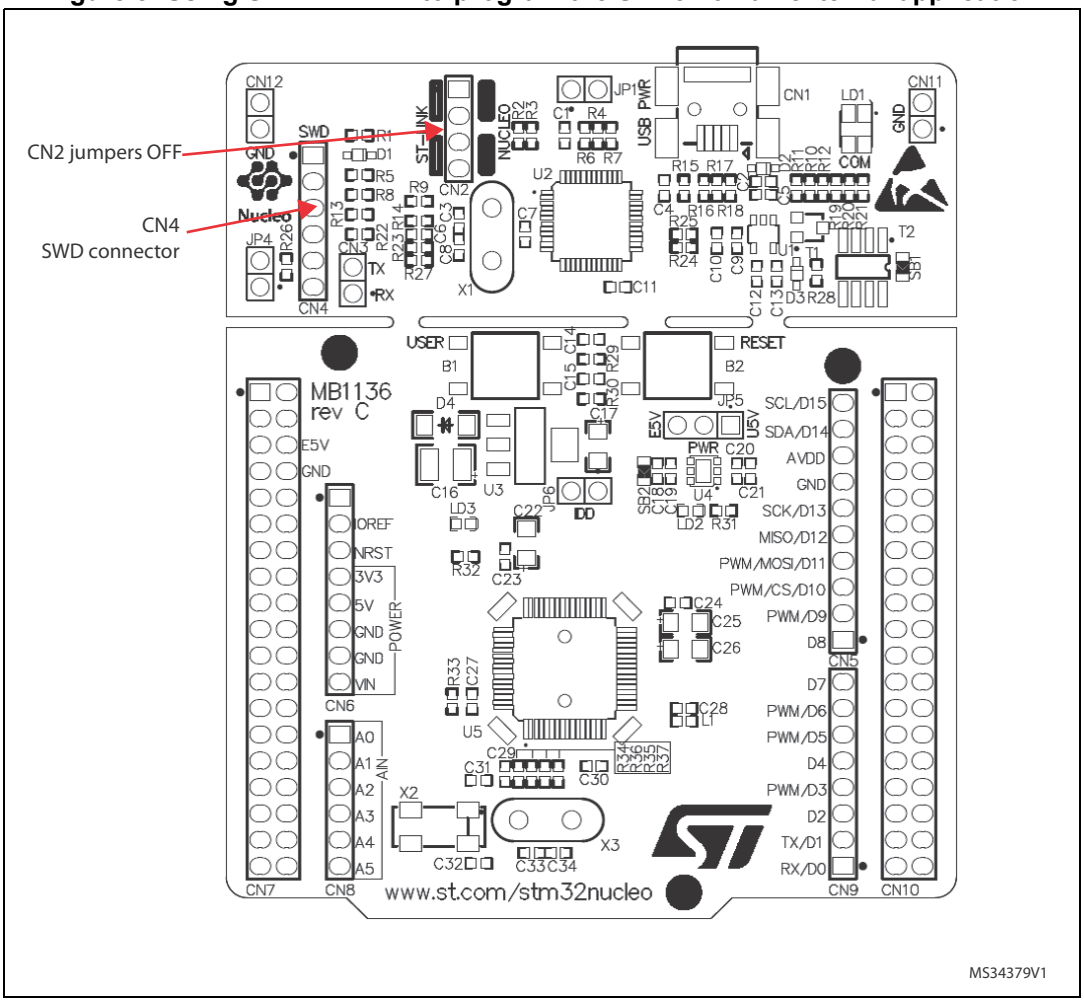
Table 4. Debug connector CN4 (SWD)

Pin	CN4	Designation
1	VDD_TARGET	VDD from application
2	SWCLK	SWD clock
3	GND	Ground
4	SWDIO	SWD data input/output

Table 4. Debug connector CN4 (SWD) (continued)

5	NRST	RESET of target MCU
6	SWO	Reserved

Figure 8. Using ST-LINK/V2-1 to program the STM32 on an external application



5.3 Power supply and power selection

The power supply is provided either by the host PC through the USB cable, or by an external Source: VIN (7V-12V), E5V (5V) or +3V3 power supply pins on CN6 or CN7. In case VIN, E5V or +3V3 is used to power the Nucleo board, using an external power supply unit or an auxiliary equipment, this power source must comply with the standard EN-60950-1: 2006+A11/2009, and must be Safety Extra Low Voltage (SELV) with limited power capability.

5.3.1 Power supply input from the USB connector

The ST-LINK/V2-1 supports USB power management allowing to request more than 100 mA current to the host PC.

All parts of the STM32 Nucleo board and shield can be powered from the ST-LINK USB connector CN1 (U5V or VBUS). Note that only the ST-LINK part is power supplied before the USB enumeration as the host PC only provides 100 mA to the board at that time. During the USB enumeration, the STM32 Nucleo board requires 300 mA of current to the Host PC. If the host is able to provide the required power, the targeted STM32 microcontroller is powered and the red LED LD3 is turned ON, thus the STM32 Nucleo board and its shield can consume a maximum of 300 mA current, not more. If the host is not able to provide the required current, the targeted STM32 microcontroller and the MCU part including the extension board are not power supplied. As a consequence the red LED LD3 remains turned OFF. In such case it is mandatory to use an external power supply as explained in the next chapter.

When the board is power supplied by USB (U5V) a jumper must be connected between pin 1 and pin 2 of JP5 as shown in *Table 7*.

JP1 is configured according to the maximum current consumption of the board when powered by USB (U5V). JP1 jumper can be set in case the board is powered by USB and maximum current consumption on U5V doesn't exceed 100mA (including an eventual extension board or Arduino Shield). In such condition USB enumeration will always succeed since no more than 100mA is requested to the PC. Possible configurations of JP1 are summarized in *Table 5*.

Table 5. JP1 configuration table

Jumper state	Power supply	Allowed current
JP1 jumper OFF	USB power through CN1	300 mA max
JP1 jumper ON		100 mA max

Warning: If the maximum current consumption of the NUCLEO and its extension boards exceeds 300 mA, it is mandatory to power the NUCLEO using an external power supply connected to E5V or VIN.

Note: In case the board is powered by an USB charger, there is no USB enumeration, so the led LD3 remains set to OFF permanently and the target MCU is not powered. In this specific case the jumper JP1 needs to be set to ON, to allow target MCU to be powered anyway.

5.3.2 External power supply inputs: VIN and EV5

The external power sources VIN and EV5 are summarized in the *Table 6.* When the board is power supplied by VIN or E5V, the jumpers configuration must be the following:

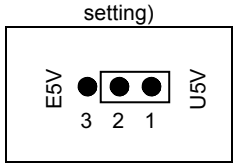
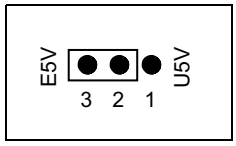
Jumper on JP5 pin 2 and pin 3

Jumper removed on JP1

Table 6. External power sources

Input power name	Connectors pins	Voltage range	Max current	Limitation
VIN	CN6 pin 8 CN7 pin 24	7 V to 12 V	800 mA	From 7 V to 12 V only and input current capability is linked to input voltage: 800 mA input current when Vin=7 V 450 mA input current when 7 V<Vin (< or =) 9 V 250 mA input current when 9 V<Vin (< or =) 12 V
E5V	CN7 pin 6	4.75 V to 5.25 V	500 mA	-

Table 7. Power-related jumper

Jumper	Description
JP5	U5V (ST-LINK VBUS) is used as power source when JP5 is set as shown below (Default setting) 
	VIN or E5V is used as power source when JP5 is set as shown below. 

Using VIN or E5V as external power supply

VIN or E5V can be used as external power supply in case the current consumption of NUCLEO and extensions boards exceeds the allowed current on USB. In this condition it is still possible to use the USB for communication, for programming or debugging only, but it is mandatory to power supply the board first using VIN or E5V then connect the USB cable to the PC. Proceeding this way ensures that the enumeration occurs thanks to the external power source.

The following power sequence procedure must be respected:

1. Connect the jumper between pin 2 and pin 3 of JP5.
2. Check that JP1 is removed.
3. Connect the external power source to VIN or E5V.
4. Power on the external power supply 7 V< VIN < 12 V to VIN, or 5 V for E5V.
5. Check that LD3 is turned ON.
6. Connect the PC to USB connector CN1.

If this order is not respected, the board may be supplied by VBUS first then by VIN or E5V, and the following risks may be encountered:

1. If more than 300 mA current is needed by the board, the PC may be damaged or the current supply can be limited by the PC. As a consequence the board is not powered correctly.
2. 300 mA is requested at enumeration (since JP1 must be OFF) so there is risk that the request is rejected and the enumeration does not succeed if the PC cannot provide such current. Consequently the board is not power supplied (LED LD3 remains OFF).

5.3.3 External power supply input: + 3V3

It can be of interest to use the +3V3 (CN6 pin 4 or CN7 pin 12 and pin 16) directly as power input for instance in case the 3.3 V is provided by an extension board. When NUCLEO is power supplied by +3V3, the ST-LINK is not powered thus the programming and debug features are unavailable. The external power sources +3.3V is summarized in the *Table 8.*

Table 8. +3.3V eternal power source

Input power name	Connectors pins	Voltage range	Limitation
+3V3	CN6 pin 4 CN7 pin 12 and pin 16	3 V to 3.6 V	Used when ST-LINK part of PCB is cut or SB2 and SB12 OFF

Two different configurations are possible to use +3V3 to power the board:

- ST-LINK is removed (PCB cut), or
- SB2 (3V3 regulator) & SB12 (NRST) are OFF.

5.3.4 External power supply output

When powered by USB, VIN or E5V, the +5V (CN6 pin 5 or CN7 pin 18) can be used as output power supply for an Arduino shield or an extension board. In this case, the maximum current of the power source specified in *Table 6* needs to be respected.

The +3.3 V (CN6 pin 4 or CN7 pin 12 & 16) can be used also as power supply output. The current is limited by the maximum current capability of the regulator U4 (500 mA max).

5.4 LEDs

The tricolor LED (green, orange, red) LD1 (COM) provides information about ST-LINK communication status. LD1 default color is red. LD1 turns to green to indicate that communication is in progress between the PC and the ST-LINK/V2-1, with the following setup:

- Slow blinking Red/Off: at power-on before USB initialization
- Fast blinking Red/Off: after the first correct communication between the PC and ST-LINK/V2-1 (enumeration)
- Red LED On: when the initialization between the PC and ST-LINK/V2-1 is complete
- Green LED On: after a successful target communication initialization
- Blinking Red/Green: during communication with target
- Green On: communication finished and successful.
- Orange On: Communication failure

User LD2: the green LED is a user LED connected to Arduino signal D13 corresponding to MCU I/O PA5 (pin 21) or PB13 (pin 34) depending on the STM32 target. Please refer to *Table 10* to *Table 19*.

- When the I/O is HIGH value, the LED is on.
- When the I/O is LOW, the LED is off.

LD3 PWR: the red LED indicates that the MCU part is powered and +5V power is available.

5.5 Push buttons

B1 USER: the user button is connected to the I/O PC13 (pin 2) of the STM32 microcontroller.

B2 RESET: this push button is connected to NRST, and is used to RESET the STM32 microcontroller.

Note: The blue and black plastic hats that are placed on the push buttons can be removed if necessary, for example when a shield or when an application board is plugged on top of NUCLEO. This will avoid pressure on the buttons and consequently a possible permanent target MCU RESET.

5.6 JP6 (IDD)

Jumper JP6, labeled IDD, is used to measure the STM32 microcontroller consumption by removing the jumper and by connecting an ammeter.

- Jumper ON: STM32 microcontroller is powered (default).
- Jumper OFF: an ammeter must be connected to measure the STM32 microcontroller current. If there is no ammeter, STM32 microcontroller is not powered.

5.7 OSC clock

5.7.1 OSC clock supply

There are four ways to configure the pins corresponding to external high-speed clock external high-speed clock (HSE):

- **MCO from ST-LINK**: MCO output of ST-LINK MCU is used as input clock. This frequency cannot be changed, it is fixed at 8 MHz and connected to PF0/PD0/PH0-OSC_IN of STM32 microcontroller.

  The following configuration is needed:

  – SB54 and SB55 OFF
  – SB16 and SB50 ON
  – R35 and R37 removed
- **HSE oscillator on-board from X3 crystal (not provided):** for typical frequencies and its capacitors and resistors, please refer to STM32 microcontroller datasheet. Please refer to the AN2867 for oscillator design guide for STM32 microcontrollers.The X3 crystal has the following characteristics: 8 MHz, 16 pF, 20 ppm, and DIP footprint. It's recommended to use 9SL8000016AFXHF0 manufactured by Hong Kong X'tals Limited.

  The following configuration is needed:

  – SB54 and SB55 OFF
  – R35 and R37 soldered
  – C33 and C34 soldered with 20 pF capacitors
  – SB16 and SB50 OFF
- **Oscillator from external PF0/PD0/PH0**: from an external oscillator through pin 29 of the CN7 connector.

  The following configuration is needed:

  – SB55 ON
  – SB50 OFF
  – R35 and R37 removed
- **HSE not used**: PF0/PD0/PH1 and PF1/PD1/PH1 are used as GPIO instead of Clock

  The following configuration is needed:

  – SB54 and SB55 ON
  – SB16 and SB50 (MCO) OFF
  – R35 and R37 removed

There are two possible default configurations of the HSE pins depending on the version of NUCLEO board hardware.

The board version MB1136 C-01 or MB1136 C-02 is mentioned on sticker placed on bottom side of the PCB.

The board marking MB1136 C-01 corresponds to a board, configured for HSE not used.

The board marking MB1136 C-02 (or higher) corresponds to a board, configured to use ST-LINK MCO as clock input.

Note: For NUCLEO-L476RG the ST-Link MCO output is not connected to OSCIN to reduce power consumption in low power mode. Consequently NUCLEO-L476RG configuration corresponds HSE not used.

5.7.2 OSC 32 kHz clock supply

There are three ways to configure the pins corresponding to low-speed clock (LSE):

- **On-board oscillator:** X2 crystal. Please refer to the AN2867 for oscillator design guide for STM32 microcontrollers.with the following characteristics: 32.768 kHz, 6 pF, 20 ppm, and SM308 footprint. It is recommended to use ABS25-32.768KHZ-6-T manufactured by Abracon corporation.
- **Oscillator from external PC14**: from external oscillator through the pin 25 of CN7 connector.

  The following configuration is needed:
  - SB48 and SB49 ON
  - R34 and R36 removed
- **LSE not used**: PC14 and PC15 are used as GPIOs instead of low speed Clock.

  The following configuration is needed:
  - SB48 and SB49 ON
  - R34 and R36 removed

There are two possible default configurations of the LSE depending on the version of NUCLEO board hardware.

The board version MB1136 C-01 or MB1136 C-02 is mentioned on sticker placed on bottom side of the PCB.

The board marking MB1136 C-01 corresponds to a board configured as LSE not used.

The board marking MB1136 C-02 (or higher) corresponds to a board configured with on-board 32kHz oscillator.

The board marking MB1136 C-03 (or higher) corresponds to a board using new LSE crystal (ABS25) and C26, C31 & C32 value update.

5.8 USART communication

The USART2 interface available on PA2 and PA3 of the STM32 microcontroller can be connected to ST-LINK MCU, STMicroelectronics Morpho connector or to Arduino connector. The choice can be changed by setting the related solder bridges. By default the USART2 communication between the target MCU and ST-LINK MCU is enabled in order to support Virtual Com Port for mbed (SB13 and SB14 ON, SB62 and SB63 OFF). If the communication between the target MCU PA2 (D1) or PA3 (D0) and shield or extension board is required, SB62 and SB63 should be ON, SB13 and SB14 should be OFF. In such case it possible to connect another USART to ST-LINK MCU using flying wires between Morpho connector and CN3. For instance on NUCLEO-F103RB it is possible to use USART3 available on PC10 (TX) & PC11 (RX). Two flying wires need to be connected as follow:

- PC10 (USART3_TX) available on CN7 pin 1 to CN3 pin RX
- PC11 (USART3_RX) available on CN7 pin 2 to CN3 pin TX

5.9 Solder bridges

Table 9. Solder bridges

Bridge	State (1)	Description
SB54, SB55 (X3 crystal)(2)	OFF	X3, C33, C34, R35 and R37 provide a clock as shown in *Chapter 7: Electrical schematics* PF0/PD0/PH0, PF1/PD1/PH1 are disconnected from CN7.
	ON	PF0/PD0/PH0, PF1/PD1/PH1 are connected to CN12. (R35, R37 and SB50 must not be fitted).
SB3,5,7,9 (DEFAULT)	**ON**	Reserved, do not modify.
SB4,6,8,10 (RESERVED)	OFF	Reserved, do not modify.
SB48,49 (X2 crystal)(3)	OFF	X2, C31, C32, R34 and R36 deliver a 32 kHz clock. PC14, PC15 are not connected to CN7.
	ON	PC14, PC15 are only connected to CN7. Remove only R34, R36
SB17 (B1-USER)	**ON**	B1 push button is connected to PC13.
	OFF	B1 push button is not connected to PC13.
SB12 (NRST)	ON	The NRST signal of the CN4 connector is connected to the NRST pin of the STM32 MCU.
	OFF	The NRST signal of the CN4 connector is not connected to the NRST pin of the STM MCU.
SB15 (SWO)	ON	The SWO signal of the CN4 connector is connected to PB3.
	OFF	The SWO signal is not connected.
SB11 (STM_RST)	**OFF**	No incidence on STM32F103CBT6 (ST-LINK MCU) NRST signal.
	ON	STM32F103CBT6 (ST-LINK MCU) NRST signal is connected to GND.
SB1 (USB-5V)	OFF	USB power management is functional.
	ON	USB power management is disabled.
SB2 (3.3 V)	ON	Output of voltage regulator LD39050PU33R is connected to 3.3V.
	OFF	Output of voltage regulator LD39050PU33R is not connected.
SB21 (LD2-LED)	ON	Green user LED LD2 is connected to D13 of Arduino signal.
	OFF	Green user LED LD2 is not connected.
SB56,SB51 (A4 and A5)	ON	PC1 and PC0 (ADC in) are connected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST Morpho connector CN7. Thus SB46 and SB52 should be OFF.
	OFF	PC1 and PC0 (ADC in) are disconnected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST Morpho connector CN7.
SB46,SB52 (I2C on A4 and A5)	OFF	PB9 and PB8 (I2C) are disconnected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST Morpho connector CN7.
	ON	PB9 and PB8 (I2C) are connected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST Morpho connector CN7 as I2C signals. Thus SB56 and SB51 should be OFF.

Table 9. Solder bridges (continued)

Bridge	State (1)	Description
SB45 (VBAT/VLCD)	ON	VBAT or VLCD on STM32 MCU is connected to VDD.
	OFF	VBAT or VLCD on STM32 MCU is not connected to VDD.
SB57 (VREF+)	ON	VREF+ on STM32 MCU is connected to VDD.
	OFF	VREF+ on STM32 MCU is not connected to VDD and can be provided from pin 7 of CN10
SB62, SB63 (USART)	ON	PA2 and PA3 on STM32 MCU are connected to D1 and D0 (pin 7 and pin 8) on Arduino connector CN9 and ST Morpho connector CN10 as USART signals. Thus SB13 and SB14 should be OFF.
	OFF	PA2 and PA3 on STM32 MCU are disconnected to D1 and D0 (pin 7 and pin 8) on Arduino connector CN9 and ST Morpho connector CN10.
SB13, SB14 (ST-LINK-USART)	OFF	PA2 and PA3 on STM32F103CBT6 (ST-LINK MCU) are disconnected to PA3 and PA2 on STM32 MCU.
	ON	PA2 and PA3 on STM32F103CBT6 (ST-LINK MCU) are connected to PA3 and PA2 on STM32 MCU to have USART communication between them. Thus SB61,SB62 and SB63 should be OFF.
SB16,SB50(MCO)(2)	OFF	MCO on STM32F103CBT6 (ST-LINK MCU) are disconnected to PF0/PD0/PH0 on STM32 MCU.
	ON	MCO on STM32F103CBT6 (ST-LINK MCU) are connected to PF0/PD0/PH0 on STM32 MCU.

1. The default SBx state is shown in bold.
2. Default configuration depends on board version. Please refer to chapter 5.7.1 for details
3. Default configuration depends on board version. Please refer to chapter 5.7.2 for details.

All the other solder bridges present on the STM32 Nucleo board are used to configure several IOs and power supply pins for compatibility of features and pinout with STM32 MCU supported.

All STM32 Nucleo boards are delivered with the solder-bridges configured according to the target MCU supported.

5.10 Extension connectors

The following figures show the signals connected by default to Arduino Uno Revision 3 connectors (CN5, CN6, CN8, CN9) and to STMicroelectronics Morpho connector (CN7 and CN10), for each STM32 Nucleo board.

Figure 9. NUCLEO-F030R8

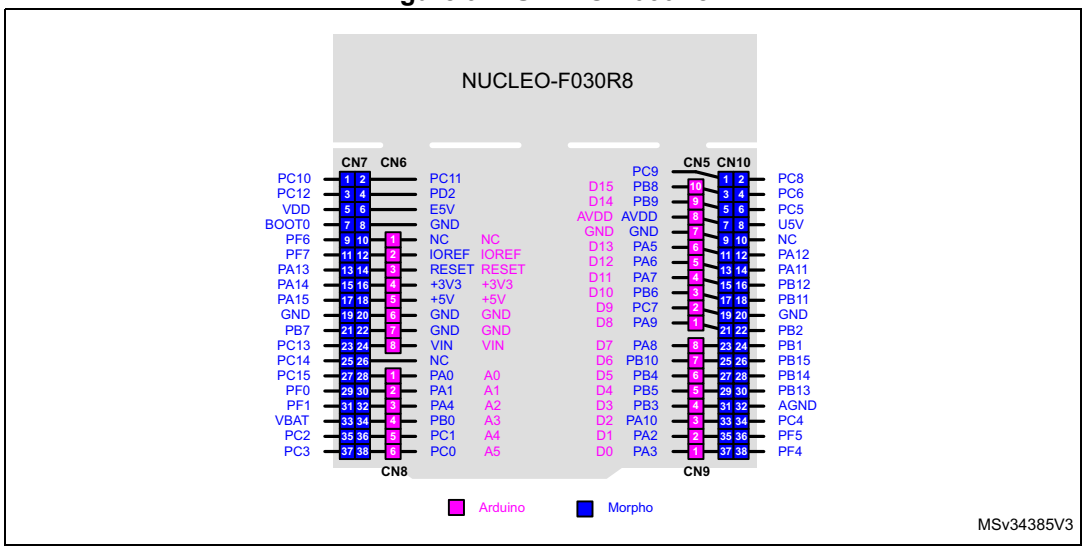


Figure 10. NUCLEO-F070RB

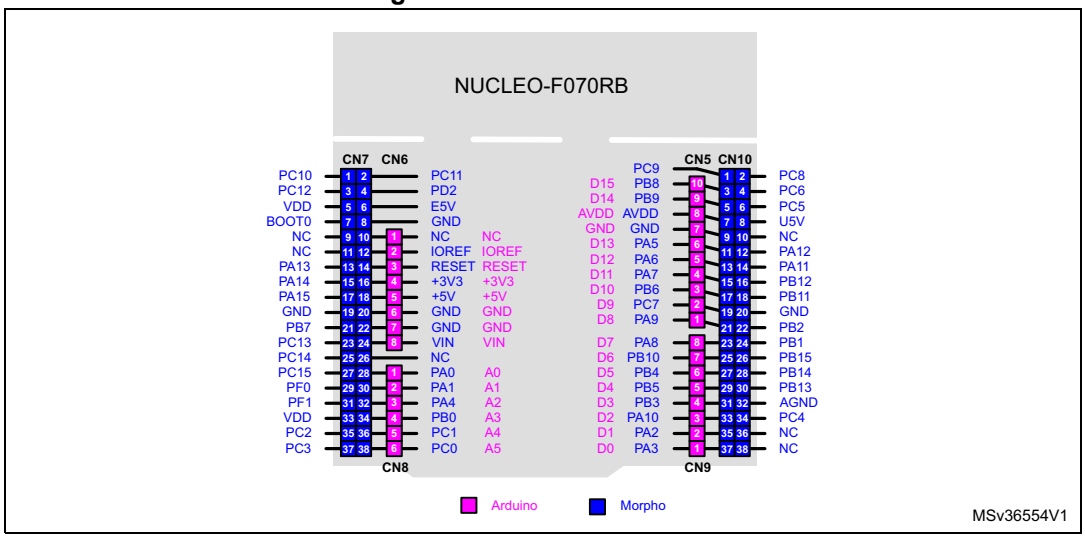


Figure 11. NUCLEO-F072RB

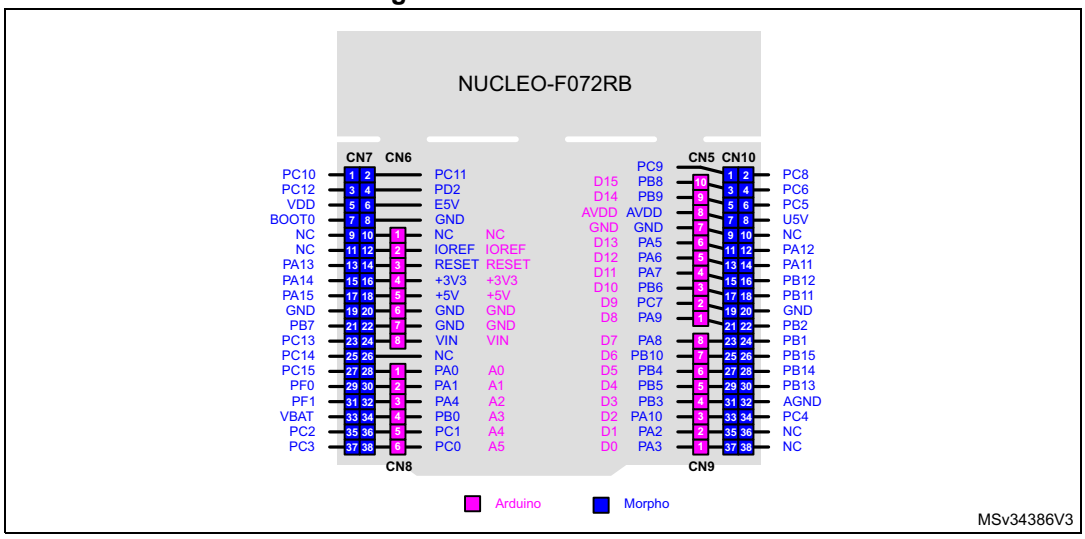


Figure 12. NUCLEO-F091RC

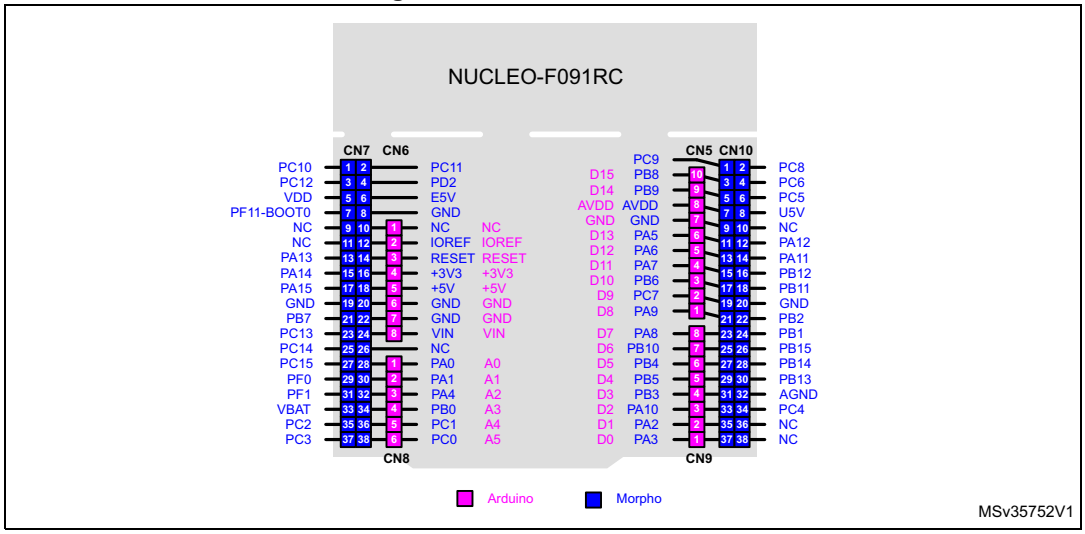


Figure 13. NUCLEO-F103RB

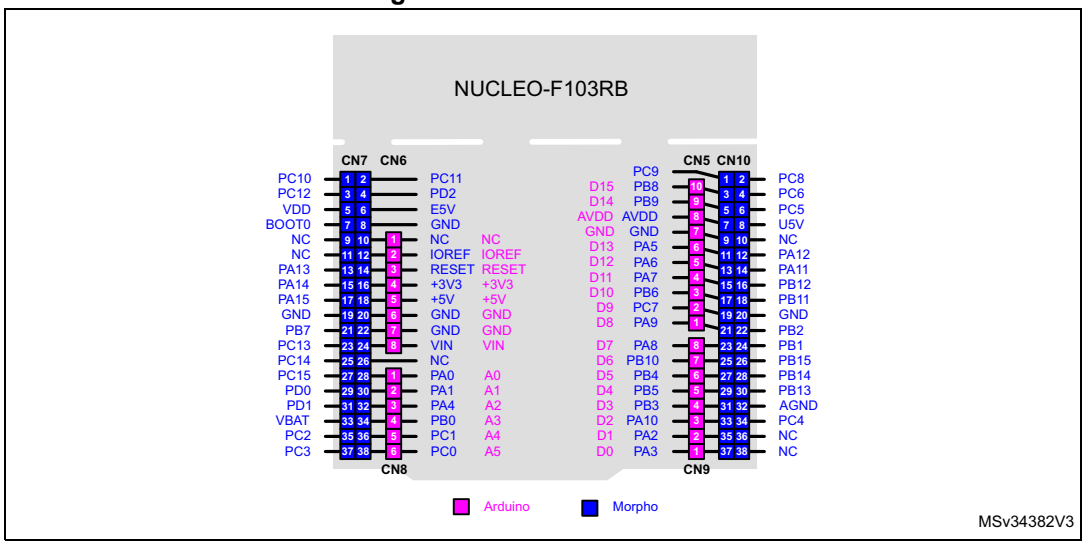


Figure 14. NUCLEO-F302R8

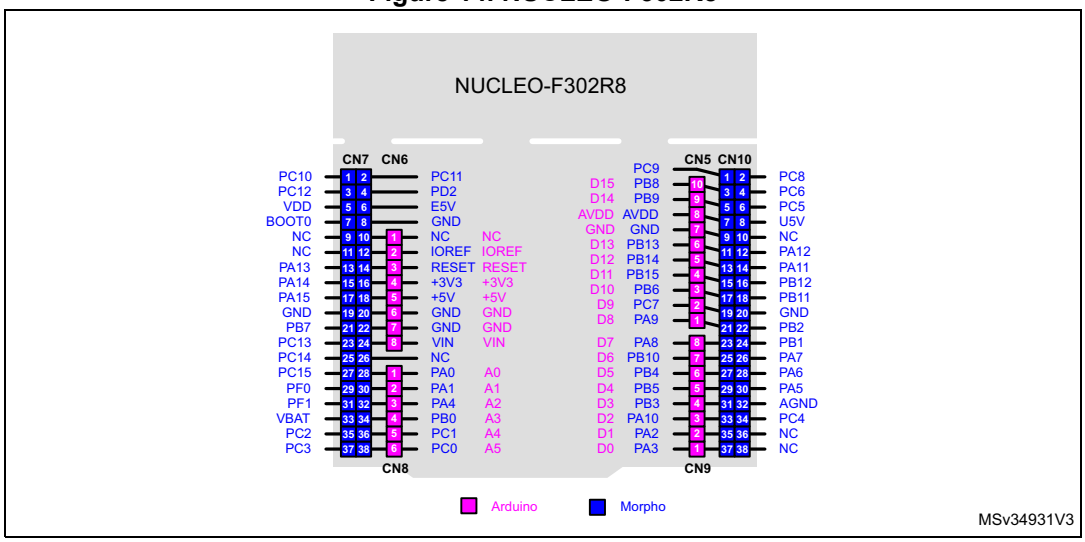


Figure 15. NUCLEO-F303RE

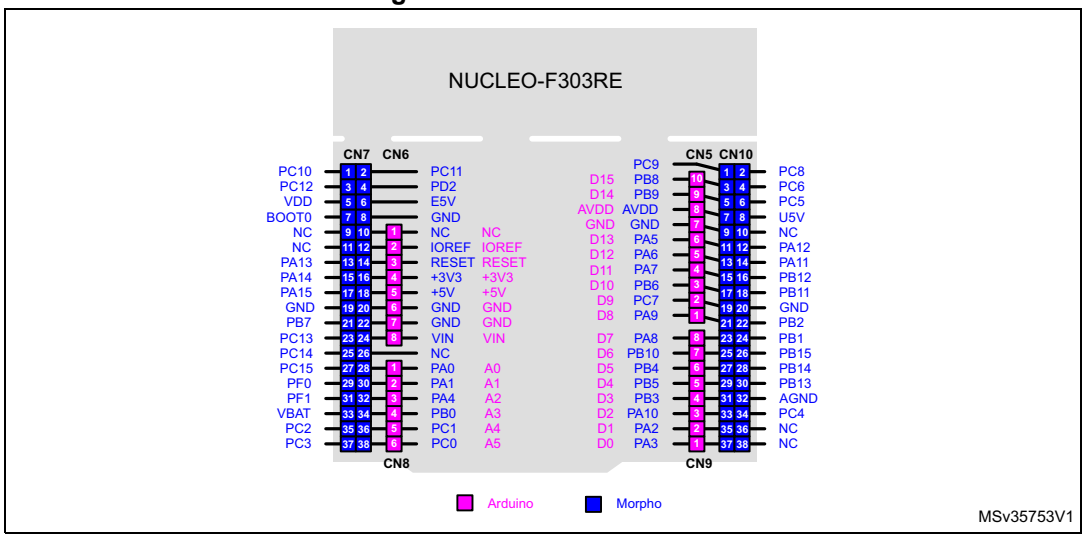


Figure 16. NUCLEO-F334R8

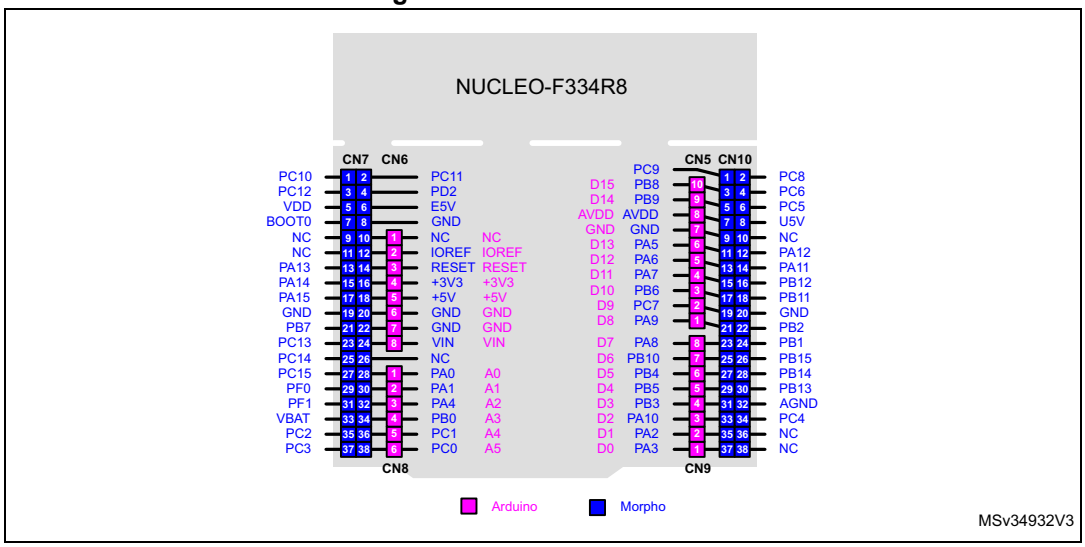


Figure 17. NUCLEO-F401RE

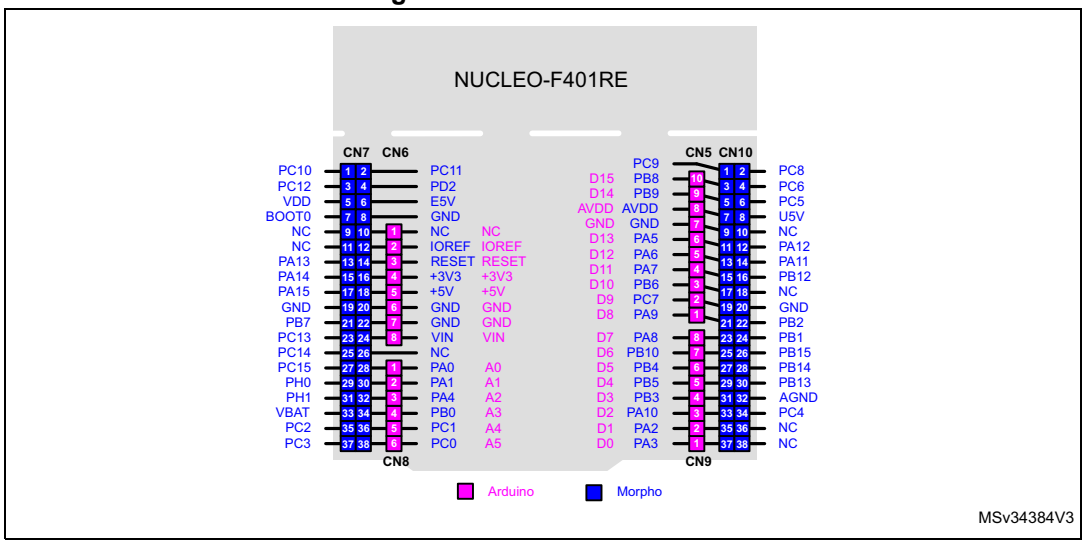


Figure 18. NUCLEO-F411RE

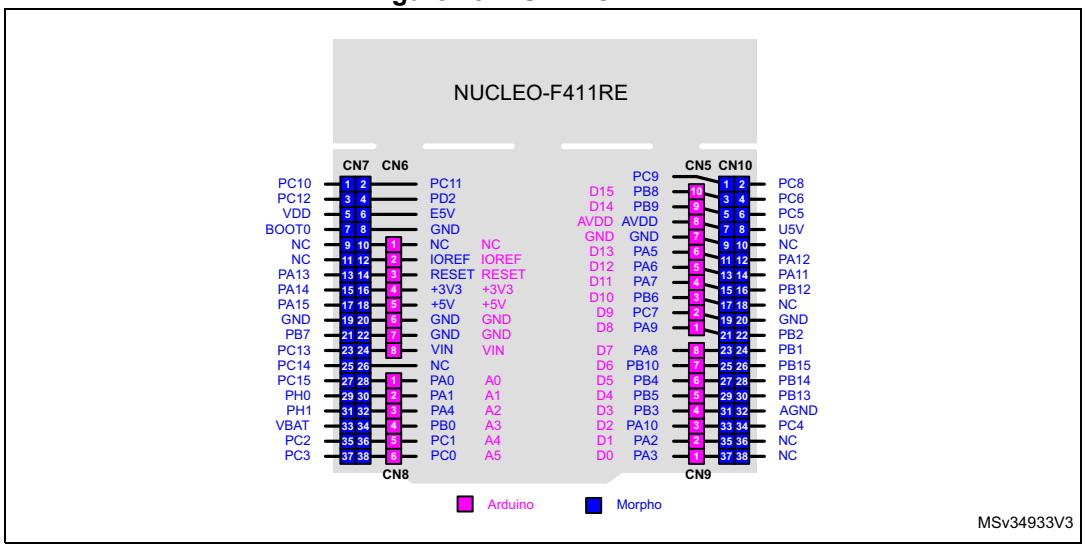


Figure 19. NUCLEO-L053R8

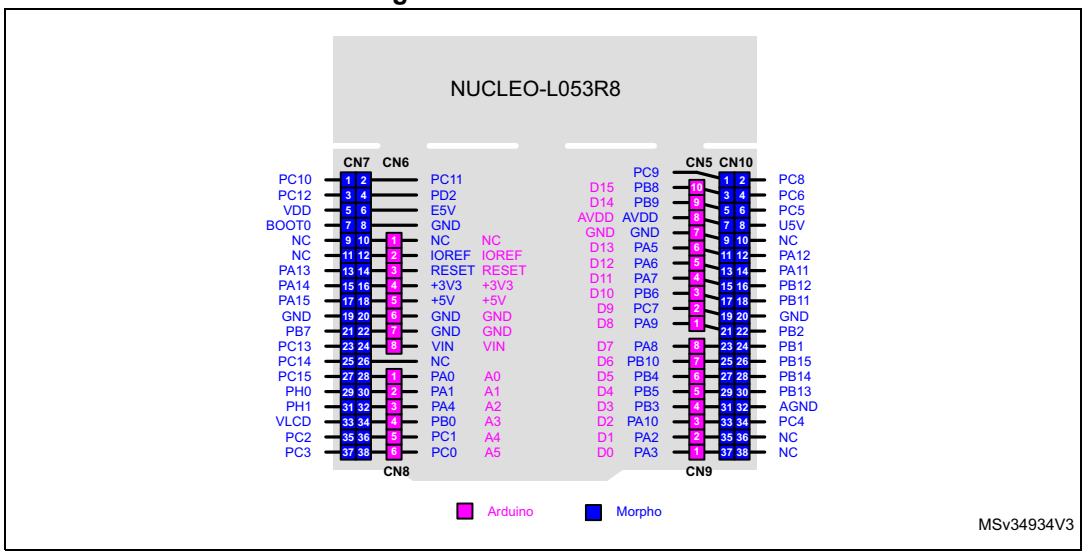


Figure 20. NUCLEO-L073RZ

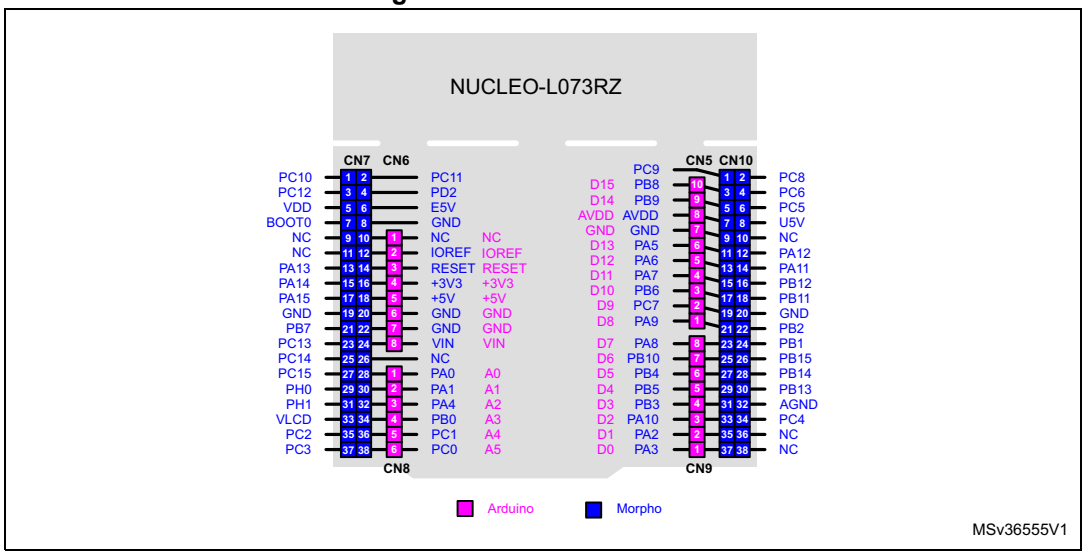


Figure 21. NUCLEO-L152RE

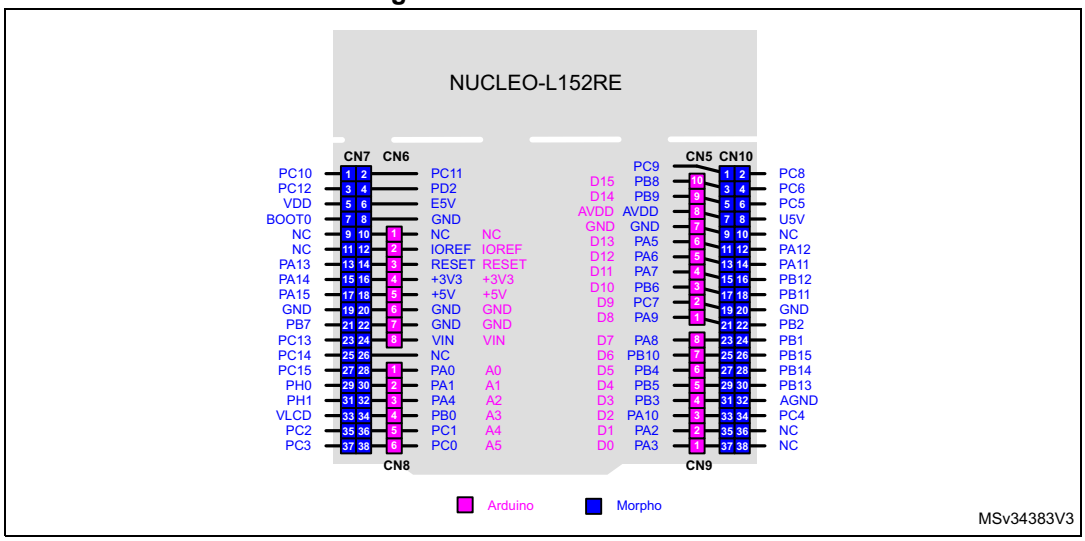
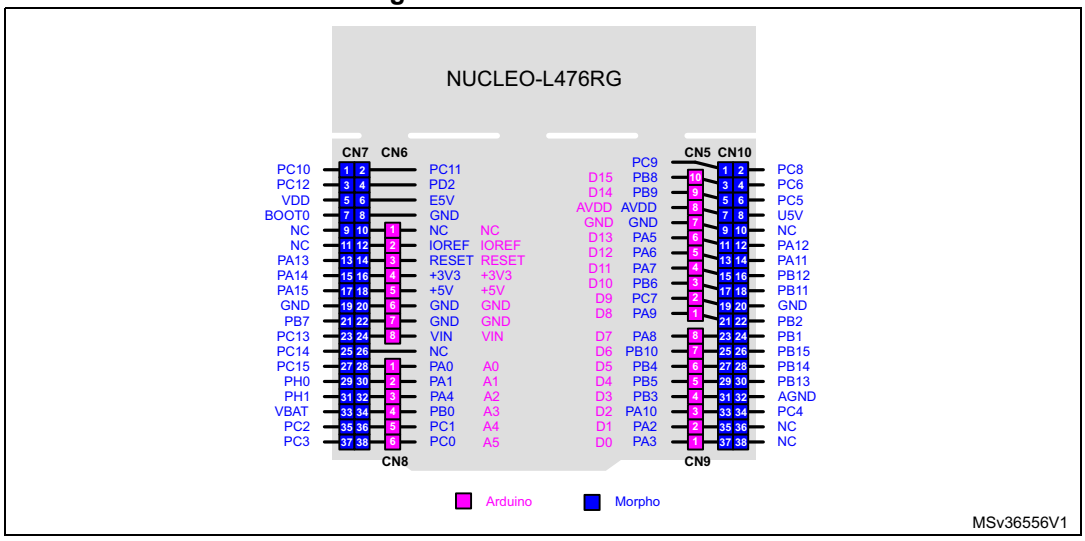


Figure 22. NUCLEO-L476RG



5.11 Arduino connectors

CN5, CN6, CN8 and CN9 are female connectors compatible with Arduino standard. Most shields designed for Arduino can fit to the STM32 Nucleo boards.

The Arduino connectors on STM32 Nucleo board support the Arduino Uno Revision 3.

For compatibility with Arduino Uno Revision 1, apply the following modifications:

- SB46 and SB52 should be ON,
- SB51 and SB56 should be OFF to connect I2C on A4 (pin 5) and A5 (pin 6 of CN8).

Caution: The IOs of STM32 microcontroller are 3.3 V compatible instead of 5 V for Arduino Uno.

Table 10 to *Table 19* show the pin assignment of each main STM32 microcontroller on Arduino connectors.

Table 10. Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)

Table 10. Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC (continued)

CN No.	Pin No.	Pin name	MCU pin	Function
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM16_CH1N or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3(2)
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2(3)
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.
2. Warning: PWM is not supported by D6 on STM32F030 and STM32F070 since the timer is not available on PB10.
3. Warning: PWM is not supported by D3 on STM32F030 and STM32F070 since timer is not available on PB3.

Table 11. Arduino connectors on NUCLEO-F103RB

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_0
	2	A1	PA1	ADC_1
	3	A2	PA4	ADC_4
	4	A3	PB0	ADC_8
	5	A4	PC1 or PB9(1)	ADC_11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM3_CH2 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for detail.

Table 12. Arduino connectors on NUCLEO-F302R8

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 Power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 Analog	1	A0	PA0	ADC_IN1
	2	A1	PA1	ADC_IN2
	3	A2	PA4	ADC_IN5
	4	A3	PB0	ADC_IN11
	5	A4	PC1 or PB9(1)	ADC_IN7 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN6 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PB13	SPI2_SCK
	5	D12	PB14	SPI2_MISO
	4	D11	PB15	TIM15_CH2 or SPI2_MOSI
	3	D10	PB6	TIM16_CH1N or SPI2_CS
	2	D9	PC7	-
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM16_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

Warning: **PWM is not supported by D9 on STM32F302 since the timer is not available on PC7.**

Table 13. Arduino connectors on NUCLEO-F303RE

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 Power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 Analog	1	A0	PA0	ADC1_IN1
	2	A1	PA1	ADC1_IN2
	3	A2	PA4	ADC2_IN1
	4	A3	PB0	ADC3_IN12
	5	A4	PC1 or PB9(1)	ADC12_IN7 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC12_IN6 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 Digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 Digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX

1. Please refer to *Table 9: Solder bridges* or details.

Table 14. Arduino connectors on NUCLEO-F334R8

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_IN1
	2	A1	PA1	ADC1_IN2
	3	A2	PA4	ADC2_IN1
	4	A3	PB0	ADC1_IN11
	5	A4	PC1 or PB9(1)	ADC_IN7 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN6 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM16_CH1N or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

Table 15. Arduino connectors on NUCLEO-F401RE, NUCLEO-F411RE

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_0
	2	A1	PA1	ADC1_1
	3	A2	PA4	ADC1_4
	4	A3	PB0	ADC1_8
	5	A4	PC1 or PB9(1)	ADC1_11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC1_10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM1_CH1N or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

Table 16. Arduino connectors on NUCLEO-L053R8

Connector No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM12_CH2 or SPI1_MOSI
	3	D10	PB6	SPI1_CS
	2	D9	PC7	TIM12_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM12_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

Warning: **PWM is not supported by D10 on STM32L053 since the timer is not available on PB6.**

Table 17. Arduino connectors on NUCLEO-L073RZ

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM22_CH2 or SPI1_MOSI
	3	D10	PB6	SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

Warning: **PWM is not supported by D10 on STM32L073 since the timer is not available on PB6.**

Table 18. Arduino connectors on NUCLEO-L152RE

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM11_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

Table 19. Arduino connectors on NUCLEO-L476RG

CN No.	Pin No.	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC12_IN5
	2	A1	PA1	ADC12_IN6
	3	A2	PA4	ADC12_IN9
	4	A3	PB0	ADC12_IN15
	5	A4	PC1 or PB9(1)	ADC123_IN2 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC123_IN1 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Please refer to *Table 9: Solder bridges* for details.

5.12 STMicroelectronics Morpho connector

The STMicroelectronics Morpho connector consists in male pin headers (CN7 and CN10) accessible on both sides of the board. They can be used to connect the STM32 Nucleo board to an extension board or a prototype/wrapping board placed on top or on bottom side of the STM32 Nucleo board. All signals and power pins of the MCU are available on STMicroelectronics Morpho connector. This connector can also be probed by an oscilloscope, logical analyzer or voltmeter.

Table 20 to *Table 26* show the pin assignment of each main MCU on STMicroelectronics Morpho connector.

Table 20. STMicroelectronics Morpho connector on NUCLEO-F030R8

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	**Name**	**Name**	**Pin No.**	**Pin No.**	**Name**	**Name**	**Pin No.**
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	PF6	-	10	9	GND	-	10
11	PF7	IOREF	12	11	PA5	PA12	12
13	PA13	RESET	14	13	PA6	PA11	14
15	PA14	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13(3)	VIN	24	23	PA8	PB1	24
25	PC14(3)	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PF0	PA1	30	29	PB5	PB13	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	PF5	36
37	PC3	PC0 or PB8(4)	38	37	PA3	PF4	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for detail

Table 21. STMicroelectronics Morpho connector on NUCLEO-F070RB

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PF0	PA1	30	29	PB5	PB13	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VDD	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for detail

Table 22. STMicroelectronics Morpho connector on NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F303RE, NUCLEO-F334R8

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)(2)	GND	8	7	AVDD	U5V(3)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(4)	RESET	14	13	PA6	PA11	14
15	PA14(4)	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PF0	PA1	30	29	PB5	PB13	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(5)	36	35	PA2	-	36
37	PC3	PC0 or PB8(5)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. CN7 pin 7 (BOOT0) can be configured by engi byte as PF11 on NUCLEO-F091RC.
3. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
4. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommended to use them as IO pins if ST-LINK part is not cut.
5. Please refer to *Table 9: Solder bridges* for detail.

Table 23. STMicroelectronics Morpho connector on NUCLEO-F103RB

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PD0	PA1	30	29	PB5	PB13	30
31	PD1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. The default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5 V
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommended to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for detail

Table 24. STMicroelectronics Morpho connector on NUCLEO-F302R8

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PB13	PA12	12
13	PA13(3)	RESET	14	13	PB14	PA11	14
15	PA14(3)	+3V3	16	15	PB15	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PA7	26
27	PC15	PA0	28	27	PB4	PA6	28
29	PF0	PA1	30	29	PB5	PA5	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for details.

Table 25. STMicroelectronics Morpho connector on NUCLEO-F401RE, NUCLEO-F411RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for detail

Table 26. STMicroelectronics Morpho connector on NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VLCD	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for detail

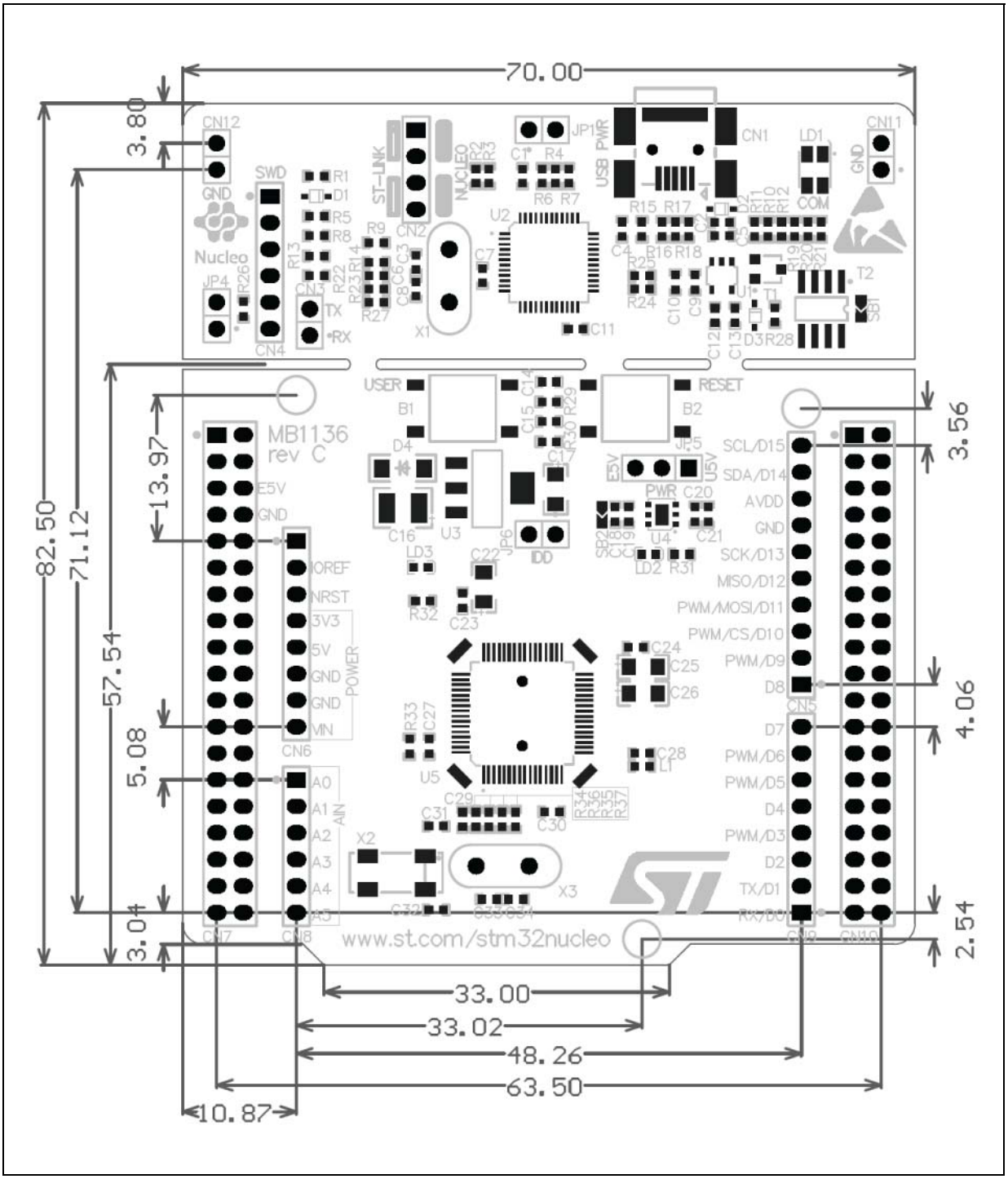
Table 27. STMicroelectronics Morpho connector on NUCLEO-L476RG

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Please refer to *Table 9: Solder bridges* for detail

6 Mechanical drawing

Figure 23. STM32 Nucleo board mechanical drawing



7 Electrical schematics

Figure 24 to *Figure 27* show the electrical schematics of the STM32 Nucleo board.

Figure 24. Electrical schematics (1/4)

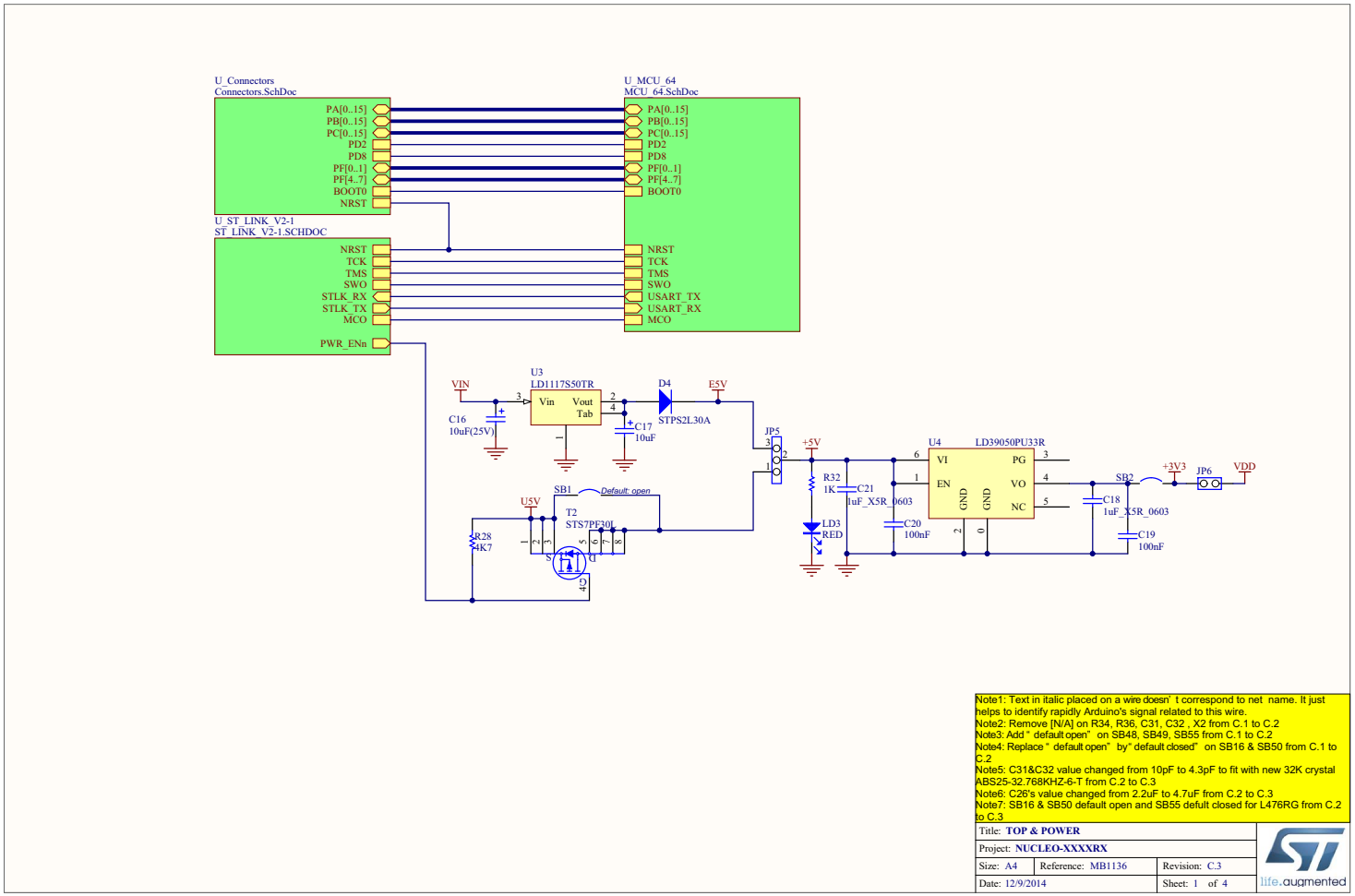


Figure 25. Electrical schematics (2/4)

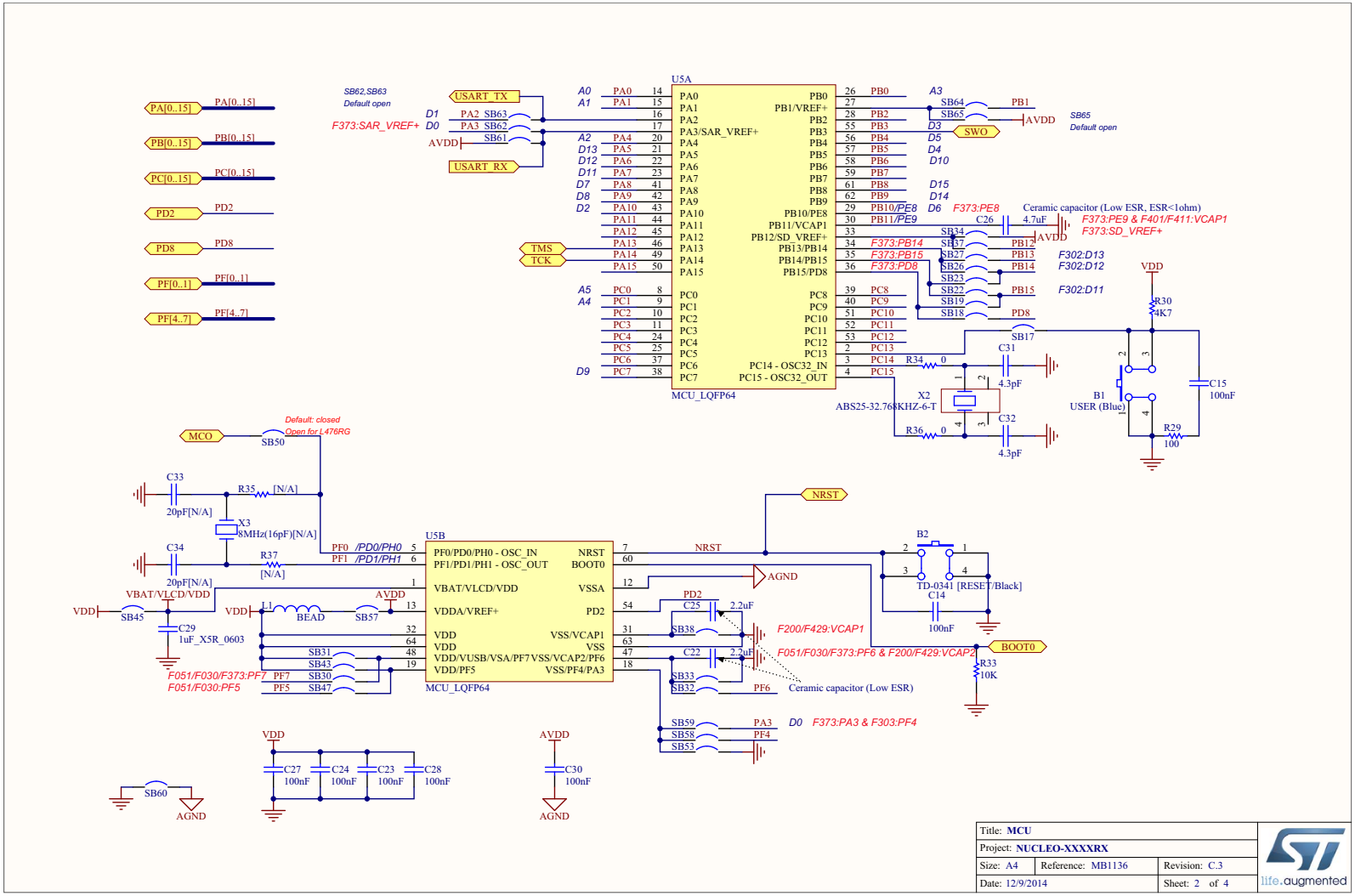


Figure 26. Electrical schematics (3/4)

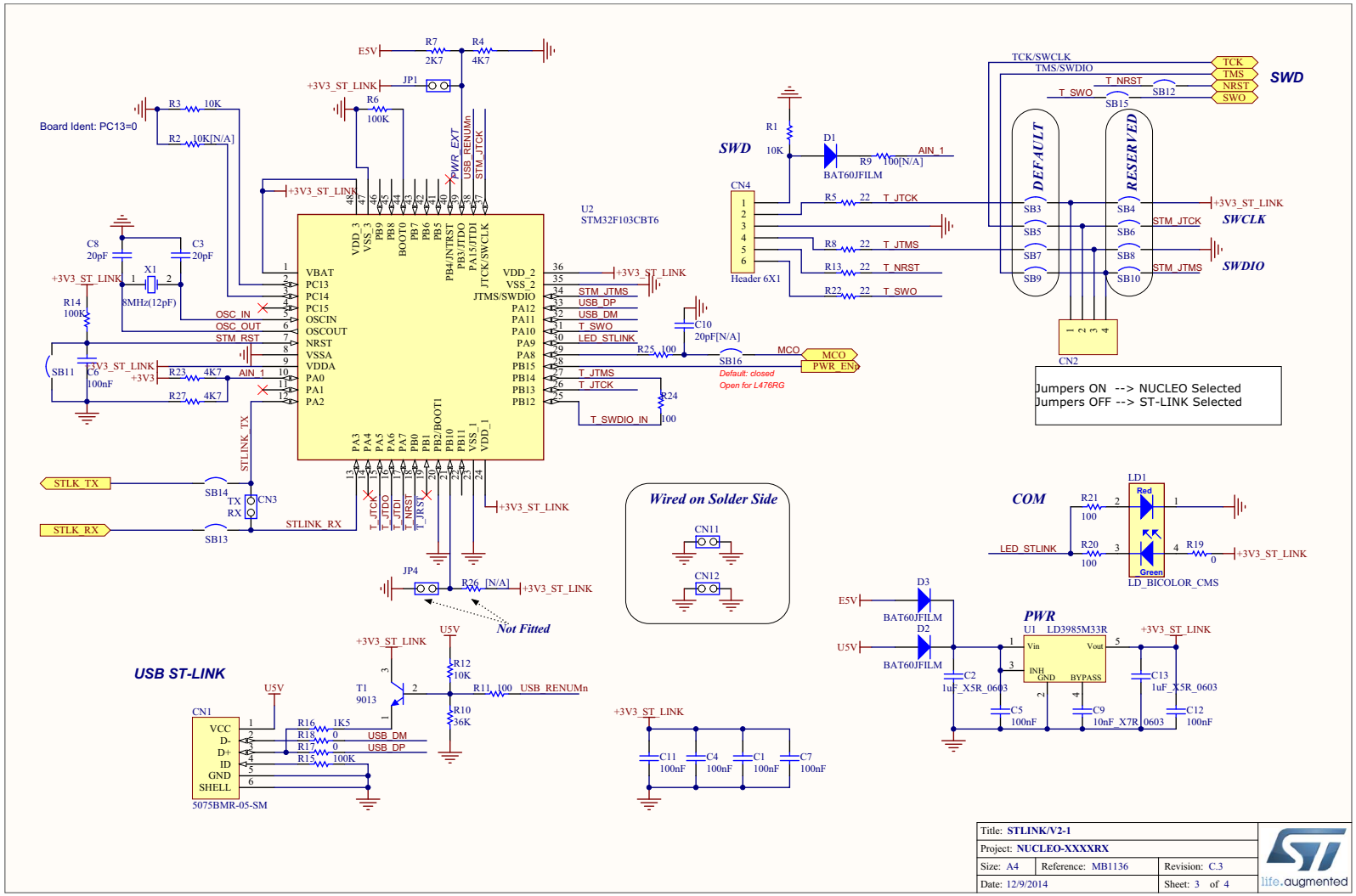
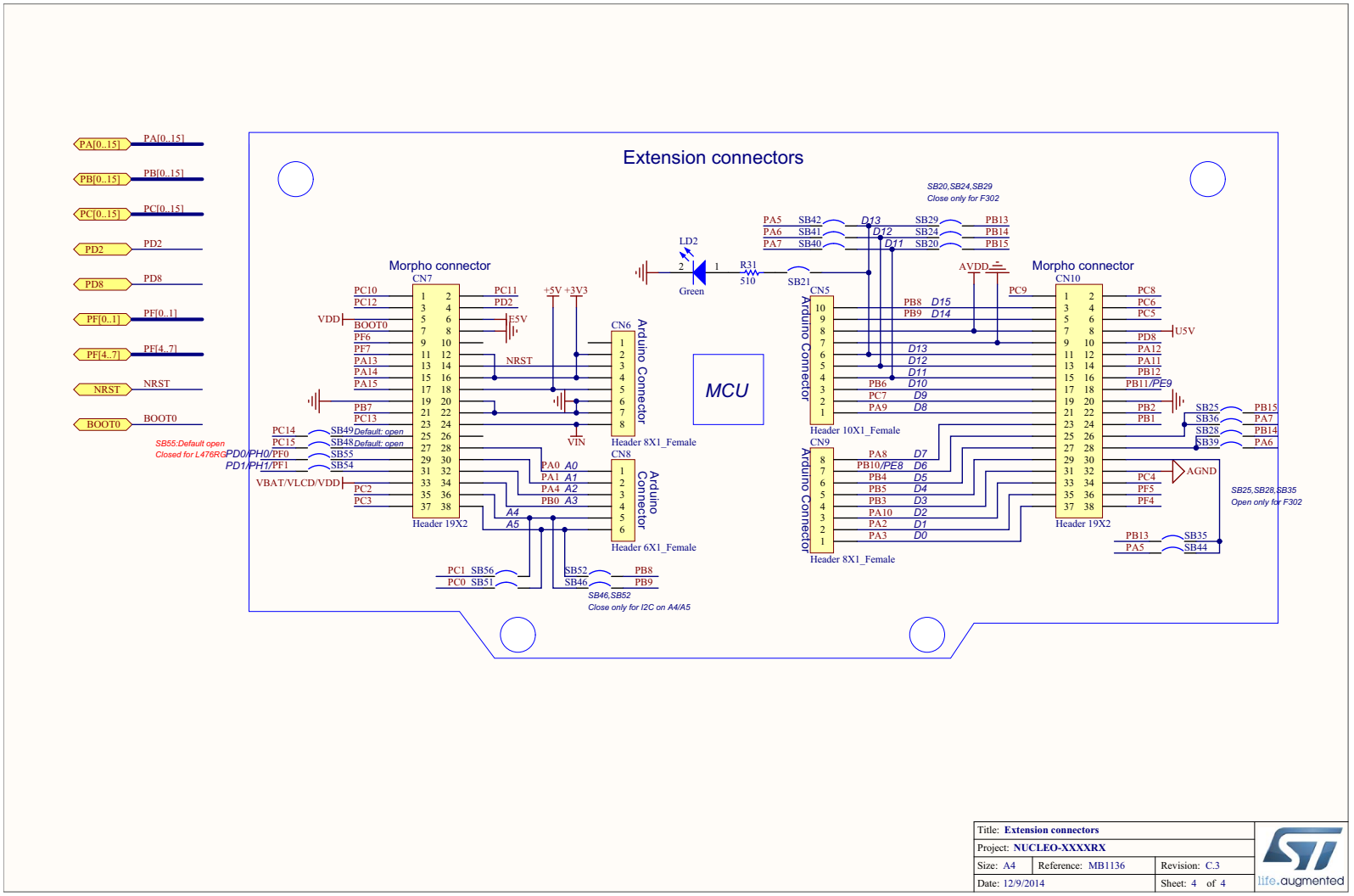


Figure 27. Electrical schematics (4/4)



8 References

1. UM1075 - ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32, User manual

9 Revision history

Table 28. Document revision history

Date	Revision	Changes
10-Feb-2014	1	Initial release.
13-Feb-2014	2	Updated *Figure 1*, *Chapter 4* and *Table 9*.
11-Apr-2014	3	Extended the applicability to NUCLEO-F302R8. Updated *Table 1: Ordering information*, *Section 5.11: Arduino connectors* and *Section 5.12: STMicroelectronics Morpho connector*. Updated *Figure 1*
10-June-2014	4	Updated the board figure: *Figure 1*. Updated HSE and LSE configuration description: *Section 5.7.1*, *Section 4* and *Section 5.7.2*. Extended the applicability to NUCLEO-F334R8, NUCLEO-F411RE and NUCLEO-L053R8.
20-June-2014	5	Updated the electrical schematics figures: *Figure 24*, *Figure 25*, *Figure 26* and *Figure 27*. Refer to the AN2867 for oscillator design guide for STM32 microcontrollers in *Section 5.7.1: OSC clock supply* and *Section 5.7.2: OSC 32 kHz clock supply*.

Table 28. Document revision history

Date	Revision	Changes
30-Sept-2014	6	Extended the applicability to NUCLEO-F091RC and NUCLEO-F303RE; Updated *Table 1: Ordering information*; Updated *Table 10: Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC*; Updated *Table 22: STMicroelectronics Morpho connector on NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F303RE, NUCLEO-F334R8*; Updated *Figure 5: Typical configuration*; Added *Figure 12: NUCLEO-F091RC*; Added *Figure 15: NUCLEO-F303RE*; Updated *Section 5.7.2: OSC 32 kHz clock supply*; Updated *Figure 24: Electrical schematics (1/4)* ,*Figure 25: Electrical schematics (2/4)*;
19-Jan-2015	7	Extended the applicability to NUCLEO-F070RB, NUCLEO-L073RZ and NUCLEO-L476RG; Updated *Table 1: Ordering information*; Updated *Section 5.2: Embedded ST-LINK/V2-1*; Updated *Section 5.7.1: OSC clock supply*; Added *Figure 10: NUCLEO-F070RB*; Added *Figure 20: NUCLEO-L073RZ*; Added *Figure 22: NUCLEO-L476RG* Updated *Table 10: Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC* Added *Table 17: Arduino connectors on NUCLEO-L073RZ* Added *Table 19: Arduino connectors on NUCLEO-L476RG* Added *Table 21: STMicroelectronics Morpho connector on NUCLEO-F070RB* Updated *Table 26: STMicroelectronics Morpho connector on NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE* Added *Table 27: STMicroelectronics Morpho connector on NUCLEO-L476RG* Updated schematics from *Figure 24: Electrical schematics (1/4)* to *Figure 27: Electrical schematics (4/4)*

Annex D

Latest Laboratory Manual for Microcontroller Engineering II

The University of Manchester

Microcomputer Engineering 2

Assignment Manual 2017/2018

Dr. Peter Green
School and Electrical and Electronic Engineering
University of Manchester
peter.n.green@manchester.ac.uk

Part 1: Introduction

Microcontroller Engineering II
Overview of Assignments

0. Polite Notice

This overview contains important information. You are expected to read it!

1. Introduction

This unit aims to help you develop the knowledge and skills to program simple embedded systems in the C programming language. This is a practical objective, so the module is supported by **four laboratory-based assignments**. The four assignments make up the coursework for this unit.

Each assignment requires a range of activities:

(i) Independent study of lecture notes, exercise sheet questions and answers, and technical documentation
(ii) The completion of assessed preparatory work *before* the scheduled laboratory session.
(iii) Laboratory work involving the development of C programs based on specifications contained in the Assignment Scripts included in this document.

1.1 Laboratory Organisation

You will attend four laboratory sessions – **one per assignment**. The class is split into 4 groups (named MCE1, MCE2, MCE3 and MCE4) and each group usually attends laboratory sessions every two weeks, once laboratories begin in week 3. Note that there are no laboratory sessions scheduled for week 7.

You will work with another student in each laboratory. Pairs are allocated by staff and cannot be changed. The assignment of students to groups and pairs can be found on Blackboard, as can the laboratory timetable.

Each laboratory session lasts for one hour and fifty minutes and takes place in the Barnes Wallis Building.

1.1.2 Equipment

The equipment to be used in the laboratory assignments is:

- A laboratory PC
  Your pair will be allocated a PC in the Barnes Wallis cluster and you will use this machine in all 4 assignments.
- An ST Nucleo board (Nucleo F401RE)
  You will each be given a Nucleo board when you return your PIC microcontroller board[1]. Details of when Nucleo boards will be distributed will be posted on Blackboard. This is likely to be at the start of week 2.
- An mbed application shield
  Each pair will be given an application shield at the same time as the Nucleo boards are distributed[2].

1.1.3 Marks

The coursework is worth 20% of the total unit mark. The mark allocation for the individual coursework components is detailed in the table below:

Coursework component	Mark as a fraction of the total unit mark	Takes place
Assignment 1	3%	Weeks 3 and 4
Assignment 2	5%	Weeks 5 and 6
Assignment 3	7%	Weeks 8 and 9
Assignment 4	5%	Weekes 10 and 11

1.1.2 Missed Laboratories and Deadlines

If you think you have a **legitimate** reason for missing a laboratory session, contact your **personal tutor** to explain the situation. If your tutor agrees that your reason is legitimate, **then he/she should send me an email detailing why you need to reschedule your lab session.** However, on-line preparatory work must be completed by the original deadline.

1.1.3 Items to bring to laboratory sessions

You MUST bring Nucleo board, USB cable and Application Shield to each of the four laboratory sessions that you attend. You will not be able to compete the laboratory

[1] The return of a PIC board does NOT apply to direct entry students.
[2] There is a world-wide shortage of these boards at the moment. The School order 300 in April of this year. We received 140 with a promise of the remainder in July. They have yet to arrive and there are no available from other sources.

session without them. You are also advised to bring a calculator, a copy of the lecture notes, a copy of the laboratory script, paper and a pen/pencil.

1.1.4 Laboratory Rules

Pen drives/other removable media may **NOT** be used in any laboratory session. Programs written before the start of the session should be stored on your P drive, and downloaded at the start of the session. Alternatively, you may arrive early and copy your own pre-written code from a pen-drive **before** the session begins.

If removable media are used during the session, they will be confiscated until the end of the session. **Failure to hand over such items upon request will lead to zero marks being awarded for the Assignment.**

All other rules regarding conduct in laboratories in general, and the Barnes Wallis cluster in particular, must be observed.

1.2 Preparatory Work

Each of the four laboratory assignments has an associated Blackboard assessment, consisting of a number of simple multiple choice questions. The Blackboard assessment will be released to each group at 9.00 am, 5 days before the relevant laboratory session takes place. The deadline for completion of the assessment is 17.00 on the day **before** the associated laboratory session.

You should keep a copy of your answers for use in the laboratory session. The assessment will be marked on-line.

1.3 Learning Outcomes

The Learning Outcomes for each assignment are included in its Assignment Script. However, there is a generic Learning Outcome that will be assessed for each assignment. This concerns the ability **to explain how a program works and to answer questions about its operation**. More details are given in the following section.

1.4 Assessment of Programs

The mark that you are awarded for each of the unit's four assignments is the sum of the marks awarded for:

- The preparatory work (on-line assessment and pseudo-code, where appropriate). This will be worth up to 10% of the assignment's marks.
- The programs that you develop during the assignment. This will account for up to 90% of the assignment's marks. See the individual assignments for detailed marks breakdowns.

Your programs will be marked **during** the laboratory session by a demonstrator. **It is your responsibility to ensure that your programs are marked during the laboratory session by asking a demonstrator to assess your work. Requests for marking made at the end of the session will incur a penalty of 20%. No marking will be undertaken more than 10 minutes after the end of the session.**

Assessment of Programs – General Rules

Full marks will be awarded:

(i) If the program in question executes correctly according to the textual specification,

(ii) AND the developers can give a **convincing** explanation of the program's operation, and answer questions about it.

If the developers of a program can demonstrate its correct execution, but **cannot** satisfactorily explain its operation or answer questions about it **or about relevant background material**, a mark of **zero** will be awarded. A proportion of the marks for a program may be awarded if it is only partially correct, so long as a convincing description of the operation of the program can be given.

When a program is marked, each student in a pair will be expected to explain which part of the program they wrote. **Failure to do this will result in each students' maximum mark being capped at 50%.**

Each assignment typically requires you to write a sequence of 3 or 4 short, related C programs. The later programs in the sequence are based on the earlier ones, but are more complex and are worth more marks.

You will receive a single mark for the most complex program that you manage to complete and demonstrate. However, you will be expected to demonstrate working versions of simpler programs too, and **your overall mark will be reduced by 30% if you cannot do this**.

1.4.1 Uploading Programs

You are required to upload to Blackboard any program that you have had marked. This is for the purposes of plagiarism checking and to help settle any dispute about

the award of marks. Programs should be uploaded during the laboratory session or at the latest, 30 minutes after the end of the laboratory session. **Programs uploaded after this time cannot be used in disputes about marks.**

In the event of problems with Blackboard, you should email your programs to: peter.n.green@manchester.ac.uk.

All programs should include the following information in comments at the start of the program: your name, student id, lab group, program title, and date.

1.4.2 The Last Week of the Semester

There may be assessment sessions during the last week of the semester. If you leave Manchester earlier than the end of week 12, you will not be able to attend these sessions and you will forfeit the marks associated with them (i.e. you will be awarded a mark of 0).

1.4.3 Plagiarism

Plagiarism in programming assignments is very easy to detect, even if function and variable names have been changed. Do not be tempted to cheat and copy someone else's work – you will not gain any understanding or expertise, and you will be caught. The School's disciplinary procedures will be invoked against **ANYONE** involved in acts of plagiarism. See your course documentation for further details.

1.5 General Advice

Most inexperienced programmers find programming a difficult[3] and frustrating experience, but one which is ultimately very rewarding. In recognition of this difficulty, **you should expect to spend between six and eight hours on each of the four assignments**. This includes the scheduled laboratory session. The time that you spend working on the assignment before the laboratory session should include the following:

- Thoroughly reading the assignment script.
- Thoroughly reading the relevant lecture notes, specified in the assignment script.
- Attempting problems on the exercise sheets, as specified in the assignment script.

[3] In fact, most programmers find programming difficult, full stop. It is simply that the more experienced you become, the more you become accustomed to the pain.

- Reading any technical documentation indicated by the assignment script. This will include the microcontroller's data sheet [1], reference manual [2], the HAL/LLL document. All are available on the unit's Blackboard pages.
- Completing the preparatory work.
- **Developing and testing prototype versions of the programs described in the laboratory script. You should try to ensure that you have completed at least the first part of an assignment before the start of the corresponding laboratory session.**
- Asking questions after lectures or by email, if appropriate.

These activities will significantly enhance your ability to complete the assignments successfully, and will also help with your preparation for the written and practical examinations in January.

1.6 Contacting Me

I am happy to be contacted via email about any aspect of the unit. It is usual for me to receive very high volumes of email from the class, given the nature of the unit, but I try to respond within 24 hours. To help me be responsive, I would ask you to do the following:

(i) Always start the subject of the email with MCE2, for example:

Subject: MCE2 Query about assignment 1

This is necessary as I use an email filter to separate out emails related to this unit from the rest of my mail. If you don't do this, I may miss your message and you may not get a response to your question.

(ii) ALWAYS include your group and pair number in your emails. If you don't, I'll simply reply by asking for this information.

2. Program Development

In order to develop a program that meets its requirements, it is necessary to have a **clear understanding of the program's specification**, and to ***design* a solution**. This involves deciding upon what **components** are needed in the program. Components in a C program are functions and data structures, whereas in a C++ program, they are classes and objects. It is then necessary to determine how the components work together e.g. which functions call which other functions, the data that is passed between functions, which functions have access to a particular data structure etc etc.[4] Finally it is necessary to decide how the components work. In the case of functions, this involves designing or choosing an algorithm. Hence a design for a C program should, in general, include:

- A list of function prototypes (see the script for Assignment 2).
- C language declarations of any complex data structures (e.g. stacks, buffers etc).
- A module chart showing the function-call structure of the program (see Assignment 2).
- A description of the operation of each function in terms of flow-charts or pseudo-code (see the scripts for Assignments 1 and 3).

Examples of these different elements can also be found in the lecture notes. After the design has been completed, the program must be implemented i.e. the actual code must be written. Note carefully how much **thinking** needs to be done before the coding stage is reached. Experience shows that starting to code without adequate preparation almost always leads to programs that do not work.

You should take steps to back-up your programs, in particular those that actually work! As you will see, some of the later assignments make use of code developed in earlier assignments. Do not simply edit the earlier program, make a copy and edit that. Much of the code that you develop in this module will be useful to you in other contexts (e.g. the Embedded Systems Project, your third year project etc etc.), so be careful to preserve your investment of time and effort.

[4] The choice of components and the design of their interactions is an iterative process. Do not assume that the first solution that comes into your mind will be the best – why should it be?

3. Programming Style

Embedded systems programming is an important part of many engineering enterprises, who always require code to be understandable, reusable and extendable. Hence, you are expected to adopt a mature, professional style of C programming. This was introduced by example in your first year course and this will be continued in the example programs presented in this unit. However, the key guidelines are as follows:

(i) Use meaningful names for functions, variables and types.

(ii) Use indentation with control constructs i.e. the statements 'belonging' to functions, if, while, for etc. Statements within such a construct should be indented by 3 or 4 spaces with respect to the if, while, for etc statement.

(iii) Make good use of comments, but avoid comments that state the obvious e.g.

```
x = 0; // x is set to 0
```

(iv) Do not use 'magic numbers' in your code. 'Magic numbers' are constant values used in expressions, as function parameters without any explanation. For example:

```
g_force = 9.8*mass;
circumference = 6.28*radius;
```

include the 'magic numbers' 9.8 and 6.28. Instead you should define these constants at the start of the program via

```
#define G 9.8
#define PI 3.142
```

or

```
const float g = 9.8, pi = 3.142;
```

and then use the appropriate name in the statements to calculate acceleration and circumference. Basically, application constants and system constants (e.g. fixed addresses) should be defined as constants at the start of the program (or in a header file).

(v) Use appropriate control constructs e.g. if you need a loop to execute a fixed number of times, use a for loop, NOT a while loop with a counter that you create.

(vi) Use functions liberally, including 1 line functions, where it makes sense to do so. See, for example, the functions switch_on_LED and switch_off_LED in Example Program 1.

(vii) Do not use extern variables in place of auto or static variables[5]. Many of you will consider doing this. **Don't**.

(viii) White space (blank lines, tabs, spaces etc) are all good within reason, but do not over-use them.

(ix) Avoid redundant or unreachable code. Redundant code has no effect on the overall computation, whereas unreachable code can never be executed (e.g. code that immediately follows a while(1) loop).

Simple message: if in doubt about how to format a program, use the style presented in the example programs.

YOU WILL BE PENALISED IF YOU DO NOT FOLLOW THESE GUIDELINES. THE PENALTIES WILL INCREASE AS THE UNIT PROGRESSES. IN THE WORST CASE, STAFF WILL DECLINE TO MARK PROGRAMS THAT ARE DIFFICULT TO READ BECAUSE THE ABOVE GUIDELINES HAVE NOT BEEN FOLLOWED.

[5] `extern` variables are declared outside of functions and have the scope and lifetime of the whole program (following their declaration). `auto` and `static` variables have the scope and lifetime of the function in which they are declared. `static` variables also have the scope of the function in which they are declared, but have the lifetime of the whole program. Practically, this means that `auto` variables 'loose' their values at the end of 'their' function, whereas `static` variables retain their values between calls to 'their' function. `static` variables can be useful in interrupt service routines (ISRs). See your earlier C course notes, or Kernighan and Ritchie [].

4. Microcontroller and IO Board Familiarisation

Input and output (IO) via peripheral devices and external hardware are important parts of embedded systems programming. In order to be able to perform IO, it is necessary to understand the detail of the development board that is being used.

As part of the process of learning about the School's boards, completing the following tables[6] is a helpful task, and you are encouraged to do this before you start the assignments for this course unit. Alternatively you can complete the parts of the table relevant to each assignment as part of the preparatory work for that assignment. Either way, you will end up with a reference that will be useful to you during the Embedded Systems project, and may also be helpful in third and fourth year projects.

4.1 Port and Pin Table

To complete the table you need to:

(i) Consult the STM32 Nucleo Board User Manual – pages 53 and 54 which refer to our Nucleo board (Nucleo F401RE). You also need to consider the schematics on pages 56-59.
(ii) Consult the mbed Application shield schematics.
(iii) Consult the Nucleo board connector diagrams
(ii) Consult the STM32F401RE Datasheet

All of these documents are available on Blackboard in Laboratories/Technical Documentation/ST Documentation. Summaries of the relevant information can be found in Laboratories/Technical Documentation/UoM Documentation

[6] Electronic copies are available on Blackboard.

4.1.1 Device Port and Pin Table

Application Shield (AS) Device	AS Connector	Description	Nucleo Port	Nucleo Pin
Speaker				
Blue LED				
Green LED				
Red LED				
Potentiometer 1				
Potentiometer 2				
Joystick right				
Joystick up				
Joystick centre				
Joystick left				
Joystick down				
XBEE[7] nReset				
XBEE Status				
XBee RX				
XBee Tx				

[7] XBee is a radio module standard. The AS does not contain an XBee module although there is a header for connecting one, just above the speaker.

Application Shield (AS) Device	AS Connector	Description	Nucleo Port	Nucleo Pin
nCS				
LCD A0				
LCD SCK				
LCD Reset				
LCD MOSI				
Temperature Sensor				
Accelerometer				

Note that not all of these devices are considered in this unit.

Part 2: The Assignments

Microcomputer Engineering II

Assignment 1: Digital I/O and Time Delays

1. Objectives/Learning Outcomes

- To gain familiarity with using the ARM Keil µVision Integrated Development Environment.
- To write C programs for the NBµC that perform pin-level input and output. You will use the Low Layer Libraries to complete this assignment.
- To use the `SysTick` core peripheral to program time delays.

2. Introduction

In the module Microcomputer Engineering I, you wrote assembly language programs to read switch inputs and drive LED outputs using the general purpose digital I/O (input/output) ports of the PIC18F8722 microcontroller. In these programs you were reading and writing the logic values of the IO pins of the microcontroller. **This is known as digital I/O.** In this laboratory session you will **writing C programs** for the Nucleo board (NB)/Application Shield (AS) combination. Specifically, the programs will involve reading the AS's joystick, which is connected to multiple GPIO pins, and driving LEDs and the loudspeaker on the AS.

3. Background Information

Lectures 3 and 4 – more detailed information in the enhanced notes at ……
Example program 1
Ref manual
Data sheet
LL doc

3. Preparatory Work

In order to successfully complete this assignment you should:

- Watch the on-line videos in Microcontroller Engineering 2/Laboratories Technical Documentation/UoM Documents/'How to' Videos - particularly

'How to Program, Debug, Use Watch Windows/Logic Analyser'. This shows you how to build[8] a project and download it from the PC to the Nucleo board.

- You should also go to Microcontroller Engineering 2/Laboratories Technical Documentation/UoM Documents/EEE STM32 Manual for additional details on creating, building and debugging projects and running code. There is also additional information about the NBμC and the various approaches to programming it.
- Read the notes for lectures 1-4.
- Read the Example Program 1 document (Microcontroller Engineering 2/Study Material/Example Programs)
- Download Example Program 1 from Blackboard (Microcontroller Engineering 2/Study Material/Example Programs)
- Build, download and run Example Program 1 on your Nucleo board.
- Complete and submit the on-line preparatory work.
- Complete *at least* the program of Part 1 *before* your laboratory session.

Relevant lecture notes	**Relevant Exercise Sheet Quest-ions**	**Relevant Example Programs**	**Relevant Sections of Reference Manual RM0368[9]**	**STM32F401xD/E Data Sheet**	**Relevant Sections of Software Document UM1725 HAL/LL document**	**Location of Blackboard Preparation Quiz**
Lectures 1, 2, 3 and 4	Ex Sheet 1	Example 1	Chapter 8 Try to get a broad under-standing of contents so you know where to look for inform-ation when you need it. Read Letcure 3 first and use Chapter 8 to provide clarification where needed	Section 3.27 (GPIOs) Section 4: Pin-out of LQFP64 package used on Nucleo board (Figure 12), and Tables 7 and 8. Section 6.3.16 IO Port Characte ristics (not very importan t for this assignme nt)	Chapter 78 Don't try to read it all! Use the list on pages 1786 and 1787 to understand which GPIO functions are available, and check out the ones used in Example Program 1.	Labor-atories /Assignme nt 1/On-line quiz

Table 1: Supporting Information for Assignment 1

[8] 'Building' means compiling the source code and linking the resulting object code with library and other code, to produce an **executable file** that is ready to be downloaded (**'flashed'**) to the Nucleo board.

[9] You should use this chapter to clarify your understanding of Lecture 3 and Example Program 1. You don't need to read it in depth.

4. The Assignment

There are 3 major, and 1 minor, parts to this assignment.

4.1 Part 0 Warm-up

You must demonstrate Example Program 1 running on a Nucleo board at the start of the laboratory session.

4.2 Part 1 Joystick and LEDs

Re-implement Example Program 1 so that moving the joystick to position p ($p \in$ {north, south, east, west}) causes LED c to illuminate ($c \in$ {red, green, blue}). When the joystick is released, the LED should be turned off.

The values of p and c to be used in your program will be specified at the start of your laboratory session. In developing the program before the laboratory, you should choose an arbitrary combination of p and c.

You should use the function `SystemClock_Config` from Example Program 1 to configure the clock. You will also use this function in ALL other programs that you write in this unit.

4.3 Part 2 SysTick, Joystick and LEDs

Extend the program of part 1 so that when the joystick is in position p_1, the LED of colour c will flash at a frequency of f_1 Hz. When the joystick is in position p_2, the LED of colour c will flash at a frequency of f_2 Hz. When the joystick is in any other position, the LED should be switched off.

p_1, f_1 and p_2, f_2 will be specified at the start of your laboratory sessions.

4.4 Part 3 Final Part

This will be provided in during laboratory session if you complete Part 2 in time. Each group will have a different final part but It will typically involve SysTick, the joystick and the loudspeaker.

5. Approach

Although the programs described in the previous section are relatively simple, they contains enough complexity to require a staged approach to its development. In other words, rather than trying to develop the complete program from scratch, it is advisable to attempt it in a number of simple stages.

Considering the specification of the program, it is clear that the programs involves 3 key tasks:

- Reading the joystick state
- Turning on the appropriate LED
- Flashing the LED at a specified frequency

If the complete program is written from scratch and contains errors (highly likely), it will be difficult to determine whether the error/errors lies in:

- The internal logic of the program
- The reading of the joystick
- The control of the LED
- Some/all of the above…

Hence you are required to follow the sequence of stages described below:

(i) Write a program that simply turns on one of the 3 coloured LEDs on the application board
(ii) Modify (i) so that the coloured LED is on only when the joy stick is in the 'fire' position (i.e. it is in the centre and pressed).
(iii) Write a program to flash the coloured LED at a frequency of 5 Hz.

Warning: if you choose not to follow the above instructions and simply try to write the program of part 2, you will a mark of 0 if the program does not work. If you require help from the demonstrators and/or staff you will be required to follow the above instructions.

Programming Constraints:

You should:

- Follow the programming guidelines presented in Part 1, Section 3 of this document.
- Use functions that separate hardware dependencies from application logic, as discussed in Lecture 4.
- You should NOT use binary strings – either use decimal or hexadecimal.

6. Information to be Specified at the Start of a Lab Session

The following parameters will be specified at the start of your laboratory session. You should use the specified set of parameters in the programs that you write.

The joystick to position p ($p \in$ {north, south, east, west})
The LED c to illuminate ($c \in$ {red, green, blue}).
Frequencies f_1 and f_2, and joystick positions p_1 and p_2.

7. Assessment and Marking

The maximum mark for this assignment is 10. The preparatory work is worth 10% of the assignment mark. The mark awarded will be based on the progress that you make with respect to the sequence of stages in section 4.2. See the table below.

Program	Mark
Part 0	1
Part 1	3
Part 2	6
Part 3	9

Table 3 Mark allocation

Note that these marks are ***not*** cumulative. For example, if you progress as far as successfully completing Part 2, you will be awarded 6/9 irrespective of whether you completed Parts 0 and 1 correctly.

You are advised to write each program and to ensure that it works before moving on to the next in the sequence. However, you need only have your final program marked – the one with the highest marks.

Please note carefully:

- If your program works, but does not use the parameters specified at the start of your laboratory session, your mark will be reduced.
- If your program works but you cannot answer simple questions about how it works, or relevant background information (including clock configuration), you will be awarded 0/10.
- You will be penalised if you do not write the programs consistent with the Programming Constraints above.

If you do not complete the program of Part 2 at the start of Section 4 by the end of your laboratory session, you are strongly encouraged for finish it in your own time in order to achieve the necessary level of proficiency.

Issue 1.0
P.N. Green 06.10.17

Microcomputer Engineering II
Assignment 2: Programming the ADC

1. Objectives/Learning Outcomes

- To program the STM32F401 analogue-to-digital converter (ADC) peripheral to convert analogue voltages from the Application Shield (AS). You are required to use the ST Low Level Library functions that support the operation of the ADC[10].
- To use the LCD (Liquid Crystal Display) on the AS.

2. Introduction

This assignment concerns both the ADC and the use of the LCD on the AS.

2.1 Analogue-to-digital converter (ADC) peripheral device

Many embedded systems are required to process analogue input signals, that is, signals that are continuous in time and amplitude. Examples include input from pressure and temperature sensors, and from microphones. In order for such signals to be processed by an embedded computer system, they must be sampled – converted into a sequence of discrete, digital samples. This sequence of samples approximates the analogue input signal, but is discrete in time and amplitude. Many embedded systems contain a peripheral device known as an analogue-to-digital converter (ADC) whose purpose is to sample an analogue signal and produce a numerical sample value.

The STM32F401 microcontroller contains an on-chip ADC peripheral that is capable of converting analogue input voltages to a digital representation in 12, 10 8 or 6 bits at sampling rates of over 60 kHz. An analogue input multiplexor enables the STM32F401 to select one of 19 voltage sources for conversion.

On the AS, 2 potentiometers are connected to 2 of the ADC multiplexor inputs. The AS board does not provide external reference voltages and consequently the ADC must be configured with $V_{REF+} = V_{DD}$ (+3.3V) and $V_{REF-} = V_{SS}$ (0V) respectively.

To configure the ADC, the following parameters must be specified (as a minimum):

- The ADC clock
- Resolution
- Data alignment

[10] The source code for these functions can be found in

- Trigger source
- ADC mode
- The channel to be sampled.
- Whether or not interrupts are enabled
- The reference voltages (positive and negative)

More details on the above and a discussion of how to use LLL functions to obtain samples from the ADC can be found in lecture 6. Note that additional parameters must be set if complex ADC modes are used.

3. Preparatory Work

In order to successfully complete this assignment you should:

- Read the notes for lectures 5 and 6.
- Consult the sources of information shown in the table below.
- Read the Example Program 2 document (Microcontroller Engineering 2/Study Material/Example Programs)
- Download Example Program 2 from Blackboard (Microcontroller Engineering 2/Study Material/Example Programs)
- Build, download and run Example Program 2 on your Nucleo board/AS.
- Complete and submit the on-line preparatory work.
- Complete *at least* the program of Part 1 *before* your laboratory session.

Relevant lecture notes	Relevant Exercise Sheet Quest-ions	Relevant Example Programs	Relevant Sections of Reference Manual RM0368[11]	STM32F40 1xD/E Data Sheet	Relevant Sections of Software Document UM1725 HAL/LL document	Location of Blackboard Preparation Quiz
Lectures 5 and 6	Ex Sheet 2	Example 2	Chapter 11 Try to get a broad under-standing of contents so you know where to look for information when you need it. Read Lecture 6 first and use Chapter 11 to provide clarification where needed	Section 6.3.20 on ADC character-istics for this assignment)	Chapter 70 Don't try to read it all! Use the list on pages 1786 and 1787 to understand which ADC functions are available, and check out the ones used in Example Program 2.	Laboratories /Assignment 2/On-line quiz

Table 1: Supporting Information for Assignment 2

[11] You should use this chapter to clarify your understanding of Lecture 6 and Example Program 2. You don't need to read it in depth.

You should also consult the document 'Using the LCD of the Application Shield' which is available on Blackboard in Laboratories/Technical Documentation/UoM Documents.

4. The Assignment

The software that is to be developed in this assignment is described in the following sub-sections.

4.0 Part 0 Warm-up

You must demonstrate Example Program 2 running on a Nucleo board at the start of the laboratory session.

4.1 Basic use of the ADC

Modify the example program so that it works with the **right-hand potentiometer**.

4.2 ADC in Continuous Mode

Modify program 4.1 so that it no longer controls LED brightness. Instead, it should:

- Configure the ADC to operate in continuous mode
- Display the current sample value on the LCD as an integer number.

The potentiometer p to be sampled by the program ($p \in$ {left-hand, right-hand}) and the resolution of the ADC r ($r \in$ {6 bit, 8 bit, 10 bit, 12 bit}) will be specified at the start of your laboratory session.

4.3 ADC and Graphics

This will be provided during laboratory session if you complete Part 4.2 in time. Each group will have a different final part but it will involve drawing a cursor object on the LCD and moving it with a potentiometer.

5. Approach

You should write **at least** the following functions to support the code that you develop:

- `void ADC_Continuous_Config(void)`
- `void Right_Hand_Pot_Config(void)`
- `void Left_Hand_Pot_Config(void)`
- `void Display_Sample_LCD(uint32_t sample);`

You may, of course, write additional functions that you believe to be useful.

6. Information to be specified at the start of a Lab Session

- The potentiometer p to be sampled by the program (p∈{left-hand, right-hand})
- The resolution of the ADC r (r∈{6 bit, 8 bit, 10 bit, 12 bit})

7. Assessment and Marking

The maximum mark for this assignment is 10. The preparatory work is worth 10% of the assignment mark. The mark awarded will be based on the progress that you make with respect to the sequence of stages in section 4.2. See the table below.

Program	Mark
4.0	1
4.1	3
4.2	6
4.3	9

Table 2 Mark allocation

Note that these marks are ***not*** cumulative. For example, if you successfully complete program 4.3 you will be awarded 7/9 irrespective of whether you completed 4.0-4.2 correctly.

You are strongly advised to write each of the 4.0-4.3 and to ensure that each one works before moving on to the next in the sequence. However, you need only have your final program marked – the one with the highest marks.

Please note carefully:

- You will lose marks if you do not write the functions specified in section 5.
- If your program works, but does not use the combination of potentiometer and resolution specified at the start of your laboratory session, you will be awarded 0/10.
- If your program works but you cannot answer simple questions about how it works, you will be awarded 0/10.

If you do not complete the program 4.3 by the end of your laboratory session, you are strongly encouraged for finish it in your own time in order to achieve the necessary level of proficiency.

Issue 1.0
P.N. Green 24.10.17

Microcomputer Engineering II
Assignment 3: Interrupts and Timers

1. Objectives/Learning Outcomes

- To gain experience of writing programs with interrupts in general and timer interrupts in particular. The potentiometers (via the ADC), the joystick and the LCD display will also figure in this assignment.

2. Introduction

At a low level, software interacts with peripheral devices using *polling* or *interrupts*. To illustrate the differences between these 2 programming techniques discussed in lecture 7, consider writing a program to meet the following specification.

> *The program should implement a cycling decimal count (0, 1, 2, …, 8, 9, 0, 1, …), where the count increments every 2 seconds. A momentary press of push-button switch PB2 should toggle between enabling and disabling the display of the current count value on the 7-segment display U1.*

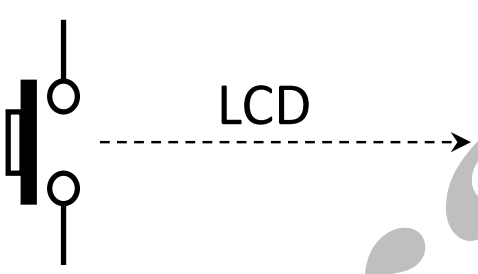
A polling technique would repeatedly check the push button. The pseudo-code program below illustrates how this task may be implemented by *polling* the state of the PB2 push-button switch.

```
while (true) {
    count = count + 1;
    if (count == 10)
      count = 0;
    display count on LCD
    delay for 2s
    if (js_central()){ // polling
      if (count displayed)
        erase count
      else
        display count
}
```

The problem with this approach is that the state of the push-button is only read every two seconds and so it is likely that a momentary press will be missed. The program clearly does not meet the stated specification.

An alternative approach is for the push-button to generate an *interrupt*, and for the associated interrupt service routine to toggle between enabling and disabling the 7-segment display. The pseudo-code program using this approach is shown below, where the function **`ISR`** is invoked when PB2 is pressed.

```
while (true){
    count = count + 1
    if (count == 10)
          count = 0;
  display count
    delay for 2s
  }
```




```
ISR {
if (count displayed)
   erase count
else
   display count
}
```

The ISR is executed immediately PB2 is pressed, irrespective of where in the main program this occurs. Hence the program meets its specification.

3. Preparatory Work

In order to successfully complete this assignment you should:

- Read the notes for lectures 5-10.
- Consult the sources of information shown in the table below.
- Read the Example Program 3 document (Microcontroller Engineering 2/Study Material/Example Programs)
- Download Example Program 3 from Blackboard (Microcontroller Engineering 2/Study Material/Example Programs)
- Build, download and run Example Program 3 on your Nucleo board/AS.
- Complete and submit the on-line preparatory work.
- Complete *at least* the program of Part 1 *before* your laboratory session.

Relevant lecture notes	Relevant Exercise Sheet Quest-ions	Relevant Example Programs	Relevant Sections of Reference Manual RM0368[12]	STM32F40 1xD/E Data Sheet	Relevant Sections of Software Document UM1725 HAL/LL document	Location of Blackboard Preparation Quiz
Lectures 5 and 6	Ex Sheet 2	Example 2	Chapter 11 Try to get a broad under-standing of contents so you know where to look for information when you need it. Read Lecture 6 first and use Chapter 11 to provide clarification where needed	Section 6.3.20 on ADC character-istics for this assignment)	Chapter 70 Don't try to read it all! Use the list on pages 1786 and 1787 to understand which ADC functions are available, and check out the ones used in Example Program 2.	Laboratories /Assignment 2/On-line quiz

Table 1: Supporting Information for Assignment 2

4. The Assignment

The software that is to be developed in this assignment is described in the following sub-sections.

4.0 Part 0 Warm-up

You must demonstrate Example Program 3 running on a Nucleo board at the start of the laboratory session.

4.1 ADC Interrupts

Write a program that uses the potentiometer/ADC, operating in **single channel single mode with interrupts** to select a number from 0 to 300 (0 in the left-most position, 300 in the right-most position). As the potentiometer is rotated, the value between 0 and 300 should be displayed on the LCD.

When the desired number is displayed, moving the joystick into position p_1 inputs the number and freezes the display until joystick position p_2 is selected. The system should not respond to potentiometer changes between the joystick being moved into p_1 and then into p_2 During this interval, the screen should continue to display the input value.

[12] You should use this chapter to clarify your understanding of Lecture 6 and Example Program 2. You don't need to read it in depth.

The joystick should generate interrupts when moved into p_1 and p_2.

4.2 Timer Interrupts

Extend the program of Section 4.1 so that the input value x s selected by the potentiometer is used to configure a timer interrupt from TIM2. At the end of this period a message should be displayed on the LCD e.g. end of period.

4.3 A Kitchen Timer

The program of Section 4.2 models kitchen timers found on cookers, microwave ovens etc. Extend the program of Section 4.2 by adding graphics and sound to indicate the end of the 'cooking' period. The choice of outputs is up to you, but the technical skill of the programming will be taken into account when awarding marks, as will be the aesthetics of the outputs.

5. Approach

You should write **at least** the following functions to support the code that you develop:

- `void ADC_IRQ_Handler(void)`
- `void TIM2_IRQ_Handler(void)`

You may, of course, write additional functions that you believe to be useful. You are required to place all code relating to TIM2 in a file called Timer2.c, which will have an accompanying header file Timer2.h.

6. Information to be specified at the start of a Lab Session

- The potentiometer p to be sampled by the program ($p \in$ {left-hand, right-hand})
- The resolution of the ADC r ($r \in$ {6 bit, 8 bit, 10 bit, 12 bit})
- Positions p_1 and p_2.

7. Assessment and Marking

The maximum mark for this assignment is 10. The preparatory work is worth 10% of the assignment mark. The mark awarded will be based on the progress that you make with respect to the sequence of stages in section 4.2. See the table below.

Program	Mark
4.0	1
4.1	3
4.2	5
4.3	9

Table 2 Mark allocation

Note that these marks are ***not*** cumulative. For example, if you successfully complete program 4.3 you will be awarded 7/9 irrespective of whether you completed 4.0-4.2 correctly.

You are strongly advised to write each of the 4.0-4.3 and to ensure that each one works before moving on to the next in the sequence. However, you need only have your final program marked – the one with the highest marks.

Please note carefully:

- You will lose marks if you do not write the functions specified in section 5.
- If your program works, but does not use the combination of potentiometer and resolution specified at the start of your laboratory session, you will be awarded 0/10.
- If your program works but you cannot answer simple questions about how it works, you will be awarded 0/10.

If you do not complete the program 4.3 by the end of your laboratory session, you are strongly encouraged for finish it in your own time in order to achieve the necessary level of proficiency.

Issue 1.0
P.N. Green 1.11.17

Microcomputer Engineering II

Assignment 4: Programming the USART

1. Objectives and Learning Outcomes

- To program the STM32F401's USARTs peripheral to transmit and receive data over an asynchronous serial communications link.
- To gain a basic familiarity with ARM's `mbed` platform for high-level embedded system development. In particular, parts of the `mbed` class library will be used.

2. Introduction

This assignment is concerned with serial communication using the STM32F401 USART (Universal Synchronous/Asynchronous Receiver Transmitter) peripheral. This is frequently used to transfer data between microcontrollers/microprocessors and external systems (e.g. other microcontrollers/microprocessors, sophisticated sensors, external memories etc). The data to be transferred consists of one or more bytes (i.e. multiple bits), and in serial communication, the bits of these bytes are transmitted and received consecutively (i.e. one-at-a-time).

2.1 Connectors

In order to complete this assignment, you will need to connect pins from the Nucleo board Morpho connectors (see the updated presentation of board schematics - Laboratories). Connectors will be available from the Electronics workshop (Sackville C31) at times to be announced on Blackboard.

3. Preparatory Work

- Consult the sources of information shown below.
- Complete and submit the preparatory work.
- Complete at least program 4.1 before your laboratory session.

4. The Assignment

The overall aim of this assignment is gain experience of using mbed and asynchronous serial communications via USARTs.

4.1 Basic Asynchronous Communication

The objective of this part of the assignment is to achieve simplex communication between 2 Nucleo boards/Application Shields (called a board combination – BC – below). Hence one BC will play the role of transmitter, and the other will act as the receiver. Use the basic **mbed** transmitter and receiver programs provided along with the document Example Programs 4 to get started. These will require modification as you are expected to transmit a counting sequence, not a string (as in the Example Program).

Use the two serial programs given in Section ??? of the Example Program 4 document as a basis for developing **mbed** to send a counting sequence between sender and receiver boards. A counting sequence CS is defined via:

- CS = n, 2n, 3n…. where n will be specified at the start of your laboratory session.
- The time between successive transmissions Δt sec i.e. the transmission of sequence elements i and i + 1 should be t s apart i.e. if i us transmitted at t_i, j should be transmitted at $t_j = t_i + \Delta t$. Δt will be specified at the start of your laboratory session.

Both the sender and receiver should be configured to operate with bit rate B (specified at the start of the lab). CS should be displayed on the LCD screens of both the transmitter and receiver BCs.

4.2 Half-Duplex Communications

Use COPIES of the code **YOU** developed in Section 4.1[13] to implement the following functionality. The sender will send a counting sequence as in part 4.1 When the count reaches the value C (to be specified at the start of your laboratory session), the 'receiver' sends a frame to the 'transmitter'. When the 'transmitter' receives this frame, it stops transmitting.

Both transmitter and receiver programs should use serial interrupts (i.e. have attached callbacks). Marks will be deducted if they do not.

4.3 Additional Functionality

COPY **YOUR** programs from 4.2 and develop them, so that when the 'transmitter' receives a frame from the 'receiver' it stops transmission and sounds a note (frequency specified at the start of the lab) on the loudspeaker for 10 s. When the 10 s has elapsed, the sound is switched off and the transmitter program waits for the user to move the joystick into position p (specified at the start of your lab session) whereupon it restarts transmission of the sequence from the minimum value in the counting sequence i.e. the count restarts.

[13] Copying other people's code constitutes academic malpractice. DON'T do it.

There are no marks awarded for joystick interrupts in this assignment, so you may choose to use them or not. **Both transmitter and receiver programs should use serial interrupts (i.e. have attached callbacks). Marks will be deducted if they do not.**

6.4 Information to be Specified at the Start of a Lab Session

- n – the counting sequence multiplier.
- The time between successive transmissions Δt sec.
- C – terminating count value.
- The bit rate B.

6.5 Assessment and Marking

The maximum mark for this assignment is 10. The preparatory work is worth 10% of the assignment mark. The mark awarded will be based on the progress that you make with respect to the sequence of stages in section 4.1. See the table below.

Program	Mark
6.1	3
6.2	7
6.3	9

Table 3 Mark Allocation

Note that these marks are ***not*** cumulative. For example, if you successfully complete program (iii), you will be awarded 8/9 irrespective of whether you completed parts (i) and (ii) correctly.

PROGRAM 3 WILL NOT BE MARKED UNLESS YOU HAVE SUCCESSFULLY IMPLEMENTED PROGRAM 1 AND/OR PROGRAM 2, HAD THEM MARKED AND OBTAINED FULL MARKS FOR THEM.

Please note carefully:

- If your program works, but does not use the combination of parameters specified at the start of your laboratory session, you will be awarded 0/10.
- If your programs for parts (ii) and (iii) do not use interrupts, you will be not receive full marks.
- **If you do not use separate .c and .h files for the joystick and the loudspeaker code, you will not receive the marks for these features even if your code works.**
- If your program works but you cannot answer simple questions about how it works, you will be awarded 0/10.

Annex E

Sample Mark Scheme

Laboratory 2

Date 07/11/2017
Time
Demonstra inaki

						Part 0	Part 1	Part 2		Part 3		All		
Last Name	**First Name**	**MCE2 Gro**	**Pair No**	**PC**	**Present (Y/N)**	**Part 0 demo-ed (1 mark)**	**Correct operation (2 marks)**	**Display anything on LCD (1 marks)**	**correctly display ADC output (2marks)**	**Draws static cursor (2 mark)**	**Cursor movement controlled by ADC (1 mark)**	**Program style - marks reduced for bad style as in demo guidelines up to a max of 3**	**LabTotal**	**Student Id**
Golan	Kevin	3	21					y	y	y	y		9	9942991
Saravanan	Balaji	3	21					y	y	y	y		9	9945169
Burrell	Mark	3	22					y	y	y	y		9	9920670
Dyson	Elliot	3	22					y	y	y	y		9	10095649
Liu	Yixing	3	23					y	y	y	y		9	10242792
Zhang	Runyu	3	23					y	y	y	y		9	9955033
Adibe	Adaobi	3	24							y	y		9	9884036
Craiu Mulle	Martin	3	24							y	y		9	9907098
Adeshina	Victor	3	25					y	y					9787368
Du	Zhenyu	3	25					y	y					9679685
Hao	Rong	3	26					y	y					9943496
Harris	Hayyan	3	26					y	y					9974902
Ajibola	Azeez	3	27											9896657
Denker	Arda	3	27											9980188
Alwarthan	Sarah	3	28					y	y					9757158
Liu	Wenxing	3	28					y	y					10251834
Sheikh	Zanib	2	24					y	y	y	y			

Laboratory 2

Date 07/11/2017
Time
Demonstrator Inaki

					Part 0	Part 1	Part 2		Part 3		All	
Last Name	First Name	MCE2 Group	Student Id	PC	Present (Y/N)	Part 0 demoed (1 mark)	Correct operation (2 marks)	Display anything on LCD (1 marks)	Correctly display ADC output (2marks)	Draws static cursor (2 marks)	Cursor movement controlled by ADC (1 mark)	Program style - marks reduced for bad style as in demo guidelines up to a max of 3	LabTotal

These are the criteria

Annex F

MarkIt 2017/18 User Guides

User Guide

MARKIT

Victor Adeshina

The University of Manchester

Table of Contents

1. Introduction

The purpose of this user guide is to provide a step-by-step instruction on how MarkIt application.

1.1 About MarkIt Application

MarkIt is an Android application designed for the purpose of marking tasks. These tasks are pre-defined and are usually assigned specific marks.

In this day and age, the ubiquity of electronic devices and the internet means that it would be remiss not to be on par with the rapid changes, especially in the educational sector. This application has been designed to provide a more efficient and reliable way of marking students compared to paper-based system.

In many accessed activities, markers are usually expected to grade students using a paper-based system. The issue is that sometimes, the marks get mixed up and comments may be unavailable due to the limited space for writing. This application also provides the opportunity to reduce financial expenses on paper as well as a positive effect on the environment.

The application aims at providing an automated solution to marking students during an accessed activity. The application provides more room for comments to back up grades given, and a reliable storage to ensure that marks can be referred back to.

The application relies on a few assumptions and requirements for it to function effectively which have been listed below:

1. Inputs: The application requires two input files. One of the input files contains the name of the students, their student identification number and the group may have been assigned to. The second input files contains the tasks (“Assignments”) that the students will be assessed on.

2. Format of input files: The format of the input file is a .csv file that contains all the relevant information about the students. The “Assignment” input file is created by a Desktop application where individual tasks are assigned specific marks.

3. Assumptions: The application assumes that the two input files discussed above are provided before-hand. There is usually an undebatable answer to the task that is expected.

Sample usage of the MarkIT Desktop Application

An ideal setting for this application to be used will be in a Microcontroller Engineering II laboratory session at The University of Manchester. In these sessions, students are given specific tasks and are marked based on a certain criterion. In the picture below, an example criterion has been given below.

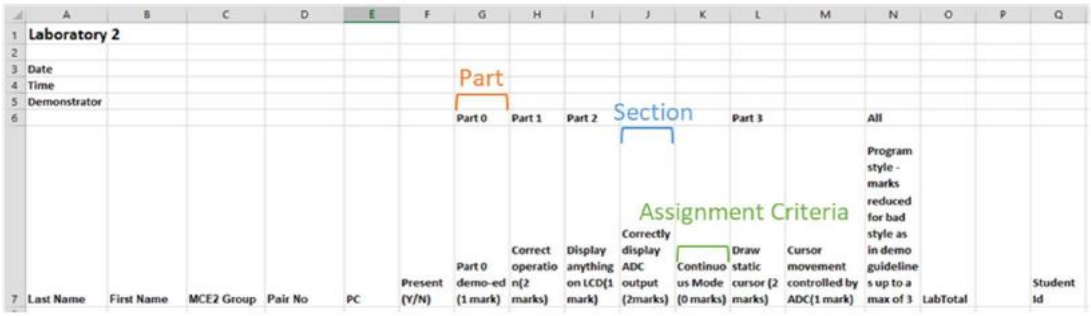


Figure 1: Showing a sample criterion

From the image above, the assignment contains 4 parts (0 - 3). Within these parts are the sections and, each section holds a certain criterion. A typical Microcontroller Engineering II laboratory session will include an assignment with around 9 sections, split into 3 or 4 parts. As the course develops, the number of parts may change.

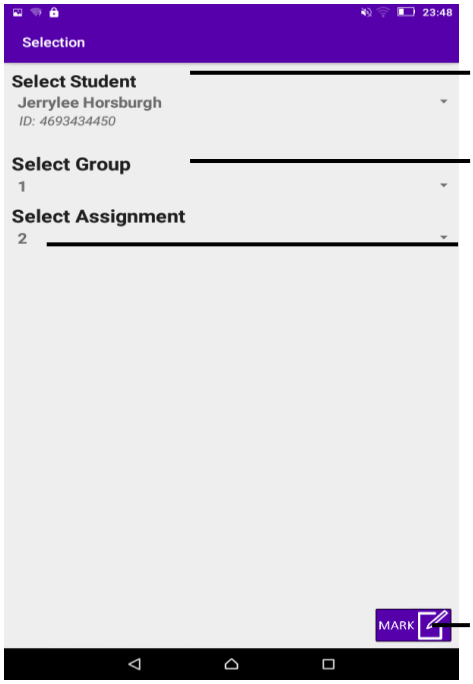
In these sessions, every student belongs to a particular overall group. For example, out of 250 students, 60 students belong to ‘Group 1’, another 60 students would belong to ‘Group 2’ and so forth.

All students would be required to complete a task that involves programming of a microcontroller with an expected result. For example, a task could be to ‘display a number on the microcontroller’. In these type of sessions, there is usually an answer that is expected, and in this scenario, that would be to see a number displayed on the microcontroller. Furthermore, this task would represent a specific mark.

When a student is unable to answer a particular question for any reason, a comment by the marker could be written. The comment section is also available for further explanation regarding why the student has been given a certain mark. This is where the ‘Comment’ feature of the application comes in. It provides the opportunity for the marker to provide a brief summary of reasons behind the mark given if necessary. The comment is also compiled alongside the marks assigned to the students in the output of the application.

2. Key Features of the app

This image below displays the first activity the user comes across while using the application. It contains a total of three spinners, which decide what information will be displayed on the next activity.



'Select Student' spinner contains the list of students with their respective student ID.

'Select Group' spinner contains the list of groups that the students have been assigned to.

'Select Assignment' spinner contains the list of assignments available that determine the questions the students will be marked on.

'MARK' button allows the user to proceed to the next activity.

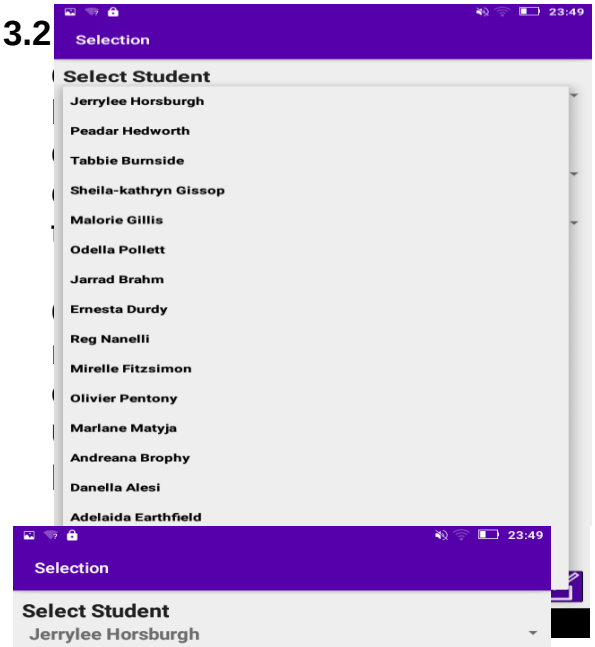
3. Getting started – Step-by-step instructions on using the app

Activity 1

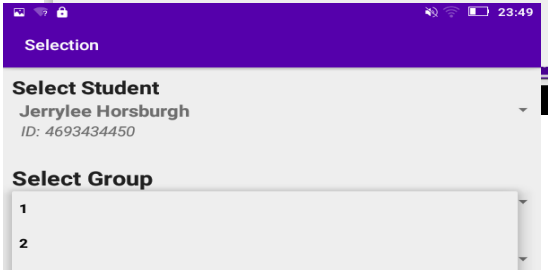
Once the app has been opened, the user must select an option from the three spinners available in the Selection activity. The images below show the display of the three spinners after they have been clicked.

3.1 Select Student

3.2

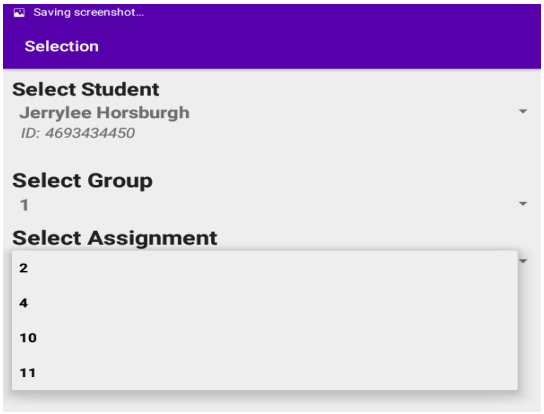


Once the **Select Student** has been selected, a list of student names appears, and further names can be seen by simply scrolling up.



The **Select Group** determines what students appear in the Select Student spinner. Every student is assigned to a

3.3 Select Assignment

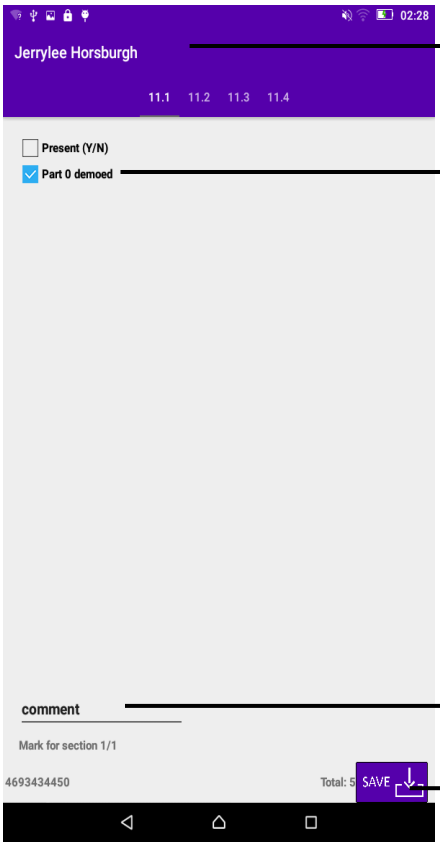


The **Select Assignment** spinner determines what questions the students will be marked on. It determines what the user views on the next activity after the **MARK** button has been pressed.

Activity 2

After the entire selection process has been completed, the user must click on **MARK** button to proceed to the next activity. This is where the actual marking of the student begins. The image below shows a typical view of the next activity. In the image below, *Select Student: Jerrylee Horsburgh, Select Group: 1, Select Assignment: 11* based on the images above.

3.4 Activity 2



The name of the student being marked.

The specific questions per section are available with select boxes. Each select box for its respective questions weighs a certain number of marks which is automatically calculated and added at the bottom of the screen.

The comment section is available for explanation.

The **SAVE** button ensures that the current changes made are saved.

After every section has been marked, a final save button can be seen at the same location on the last section as the image above. Once the button is clicked, the marks, comments and overall grade of that specific student is stored. The user can navigate back to the initial activity at the start of the application using the 'back' button on the Android tablet and repeat the process for a new student.

4. Technical Note

- This application was developed using Android Studio. Java programming language was used within this IDE for the application.
- The application relies on input file provided by a desktop application.
- The output data from MarkIt is stored in a .csv file.

USER GUIDE

MARKIT FORM CREATOR

Victor Adeshina

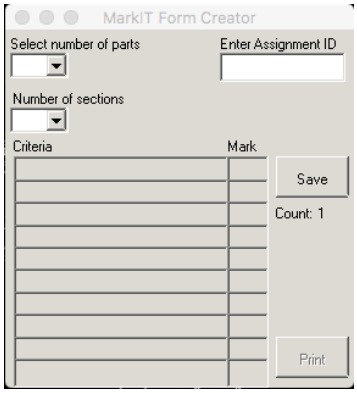
The University of Manchester

Introduction

The MarkIT Form Creator is a desktop application designed to support the MarkIT Android Application. It provides one of the input files required for the MarkIT tablet app to function effectively.

This application is supposed to assign specific marks to certain tasks that students must complete during a contact session such as a laboratory. The application also organises the tasks under different groups depending on how the tasks are setup.

Key Features of the Application



- Select number of parts: Used to decide the sub-division of the tasks. Currently available for a maximum of 10 parts.
- Number of sections: Used to decide the number of tasks per sub-division. Currently available for a maximum of 10 parts.
- Enter Assignment ID: This represent the number that the user chooses to represent the task as.
- Save: Used to save the data provided.
- Print: Used to export a .csv file based on the input data given to the Form Creator.

Sample usage of the MarkIT Desktop Application

An ideal setting for this application to be used will be in a Microcontroller Engineering II laboratory session at The University of Manchester. In these sessions, students are given specific tasks and are marked based on a certain criterion. In the picture below, an example criterion has been given below.

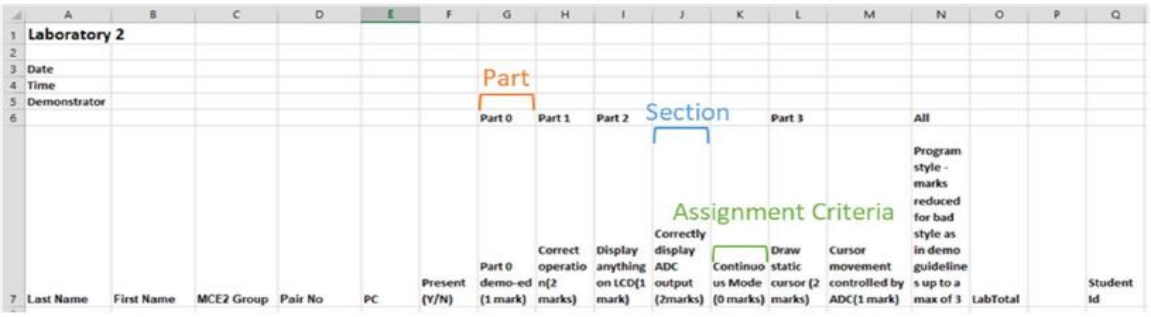


Figure 1: Showing a sample criterion

From the image above, the assignment contains 4 parts (0 - 3). Within these parts are the sections and, each section holds a certain criterion. A typical Microcontroller Engineering II

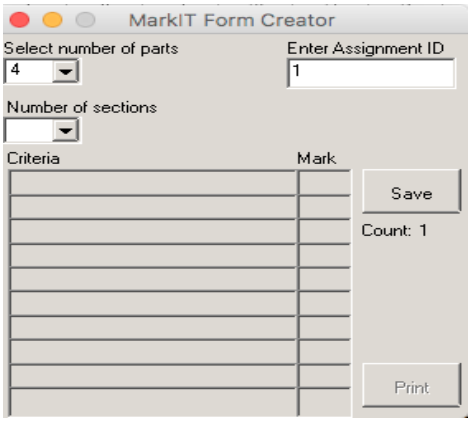
laboratory session will include an assignment with around 9 sections, split into 3 or 4 parts. As the course develops, the number of parts may change.

Step-by-step Guide

Once a typical criterion to the one in the image above has been created, the following steps could be taken to put these values into the MarkIT Form Creator. Please note that the Assignment ID solely depends on what the user has decided to name it and does not affect the output beyond a numerical representation of the overall task.

1. Selecting number of parts and sections

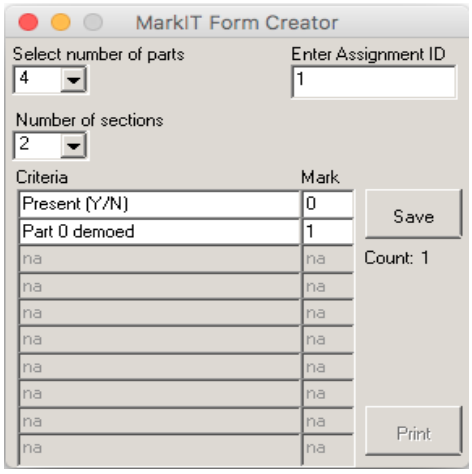
The number of parts is selected based on the way the examiner has decided to sub-divide the tasks. For example, in Figure 1 above, four parts have been labelled. (Part 0 – Part 3). As shown by the side, the number of parts selected on the application is 4. This number does not change automatically as you move from one part to the next.



The 'number of sections' represent every individual task the students will be tested on. The individual sections belong to specific groups; therefore, the user needs to constantly change the number of sections depending on the 'Part' the sections belong to.

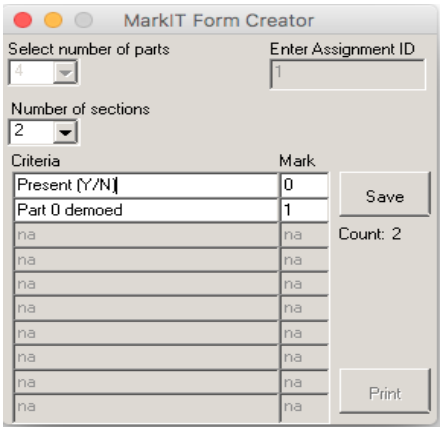
You can know the 'Part' number you are in depending on the 'Count' display in the application. For example, as we are currently on Part 0 in the application and this is the first 'Part', the 'Count' number is '1'. If we were to proceed to 'Part 1', the 'Count' number would be '2'.

Following the example in Figure 1, we will now add details to the 'Criteria' section of the application for Part 0. Since we are in Part 0, there will be only 2 sections based on the example.



2. Save:

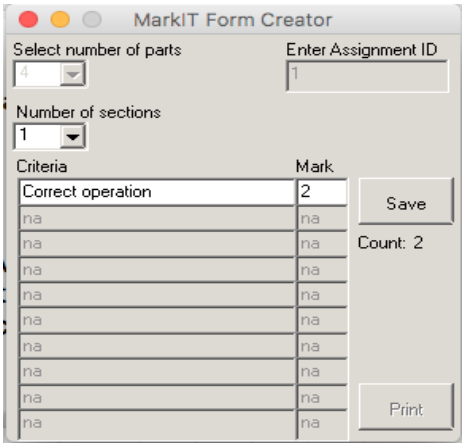
After the details have been inserted for the first part, the 'Save' button is pressed. The 'Save' button ensures that the number of Parts cannot be changed and that the 'Count' value as discussed earlier is incremented. This helps the user know that they can proceed to the next 'Part'.



3. Addition of information for other parts:

After the 'Save' button has been pressed, the user can change the number of sections if necessary, to suit the new 'Part' that is to be inputted for.

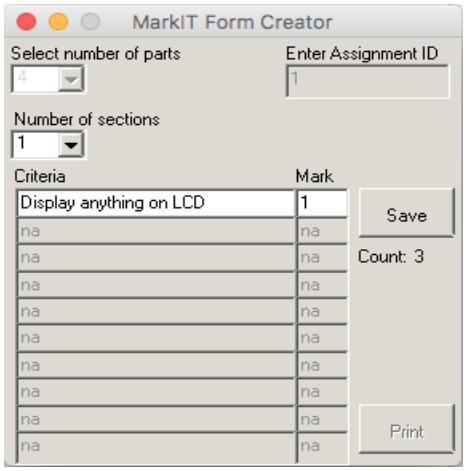
Following the example in Figure 1, Part 1 only has 1 section, therefore, the next step would be to change the 'number of section' spinner to 1 and add the relevant information to that section.



4. Inputting data (1):

Once again, the 'Save button' is pressed to move to the next 'Part', the Count value goes up to 3 and the number of sections can be adjusted once again.

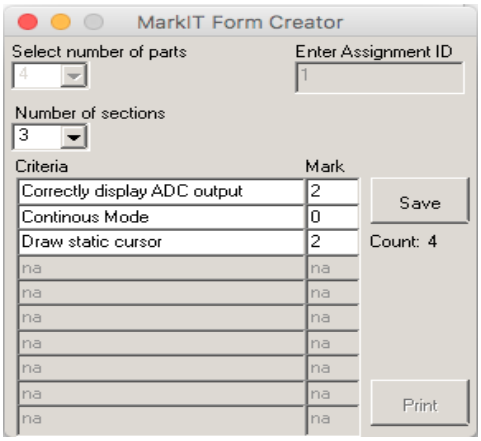
Following the example in Figure 1 and the steps taken so far, we would be in 'Part 2' of the example (which is represented by a count value of 3 on the app), which requires '1' to be selected as the 'number of sections' because it only has one section.



5. Inputting data (2):

Once again, the 'Save button' is pressed to move to the next 'Part', the Count value goes up to 4 and the number of sections can be adjusted once again.

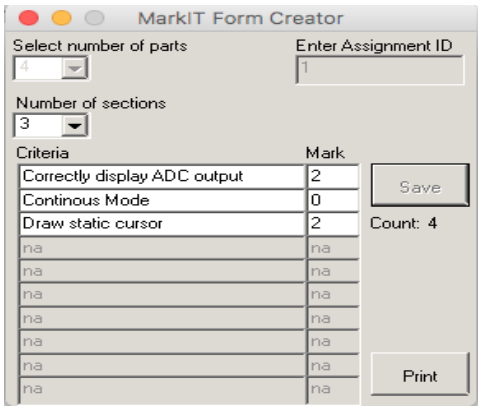
Following the example in Figure 1 and the steps taken so far, we would be in 'Part 3' of the example (which is represented by a count value of 4 on the app), which requires '3' to be selected as the 'number of sections' because it only has one section.



6. Final Step:

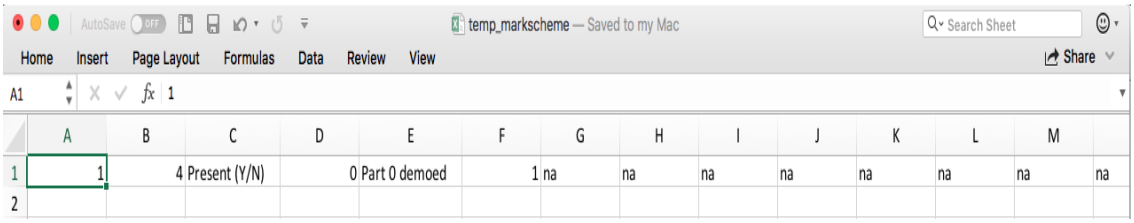
Now that all the available parts have been completed, the 'Save button' is pressed and the 'Count' value is 4. Since this aligns with the 'Select number of parts', the 'Save' button is greyed out and the 'Print' button becomes available for clicking.

Clicking on the 'Print' button will enable us obtain a .csv file that would be necessary as the input to the MarkIT Android Application.



7. Output:

The output file would be a .csv file as discussed earlier with all the information that has be put into the desktop application. The information is stored horizontally and so it is difficult to get the screenshot of the entire document. However, this is a screenshot of the first part of the document, showing the first information that was put into the application for 'Part 0'.



Annex G

Wheel of Life Application Showcase

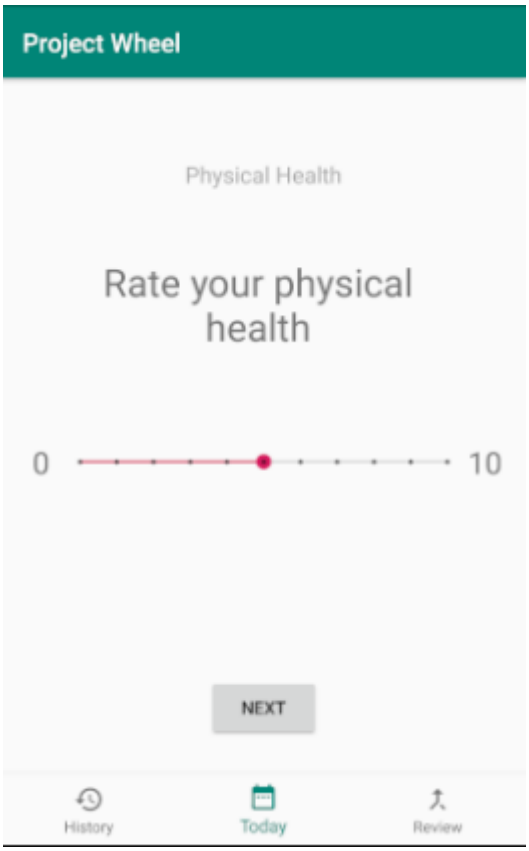


Figure 1

This application was developed as part of the University of Oxford Hackathon 2018. It is a simple application to monitor student mental health while at university.

It is an Android app and was developed as the first sample/training application in preparation for development of MarkIt.

Although it involved cloud store of data it was done on Amazon Web Services, an experience that was vital in the choice to use a different service for the application in this project.

The application only had two activities: one to gather student data (fig.1) and another to display the data in a radar graph format for students to evaluate their mental health and experience (fig.2).

Figure 2

Annex H

Initial Proposed Data Structure for MarkIt

STUDENTFIRSTNAME STUDENTLASTNAME STUDENTID

STUDENT TABLE

LABGROUPLOCATION LABGROUPUNIT LABGROUPASSIGNMENTID LABGROUPNUMBER LABGROUPSTUDENTID LABGROUPDATE

LABGROUP TABLE

MARKSCHEMEASSIGNMENTNAME MARKSCHEMEAUTHOR MARKSCHEMEASSIGNMENTAVAILABLEMARKS MARKSCHEMEASSIGNMENTID

MARK SCHEME TABLE

SECTIONNAME

SECTIONAUTHOR SECTIONASSIGNMENTID SECTIONAVAILABLEMARKS SECTIONID

SECTION TABLE

PARTNAME PARTASSIGNMENTID PARTAVAILABLEMARKS PARTSECTIONID PARTPARTID

PART TABLE

SECTIONAUTHOR	SECTIONASSIGNMENTID	SECTIONPARTID	SECTIONID	SECTIONSTUDENTID	SECTIONSTUDENTMARKS

SECTION MARKS TABLE

TOTALMARKSACHIEVED	TOTALASSIGNMENTID	TOTALSTUDENTID

TOTAL MARKS TABLE

Annex I

Karanta Application Showcase

Karanta! was an application developed as part of the learning and training done in preparation for the development of MarkIt. It was the first application developed where Microsoft's Azure Cloud was powering the back-end and cloud storage, and constituted first contact with this service and its APIs.

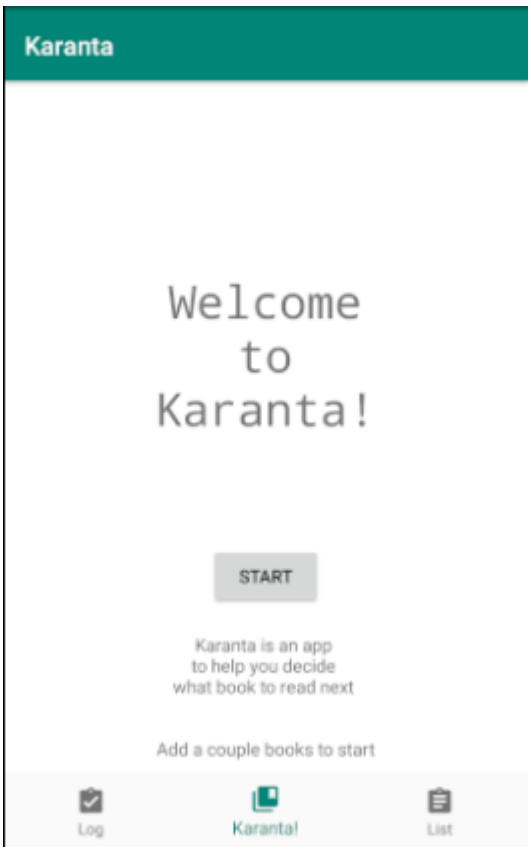


Figure 1 - Karanta's landing page

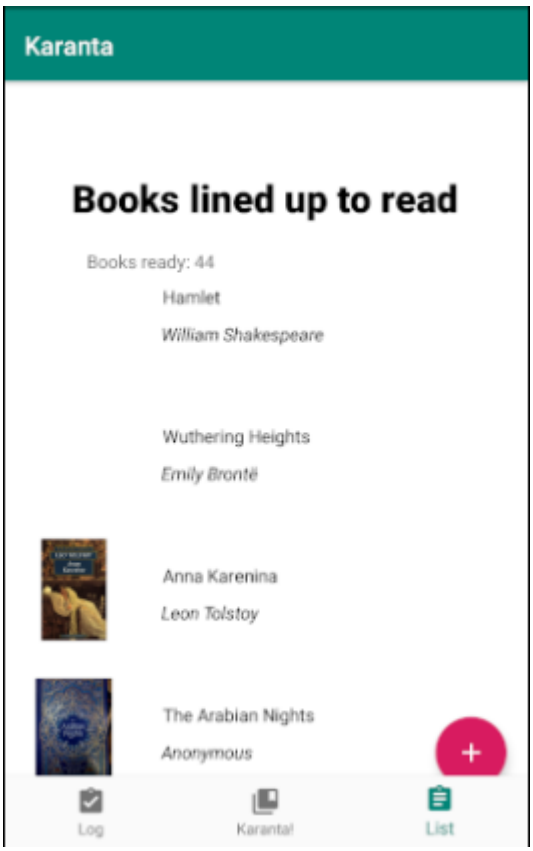


Figure 2 – Karanta included a log of books lined up to read

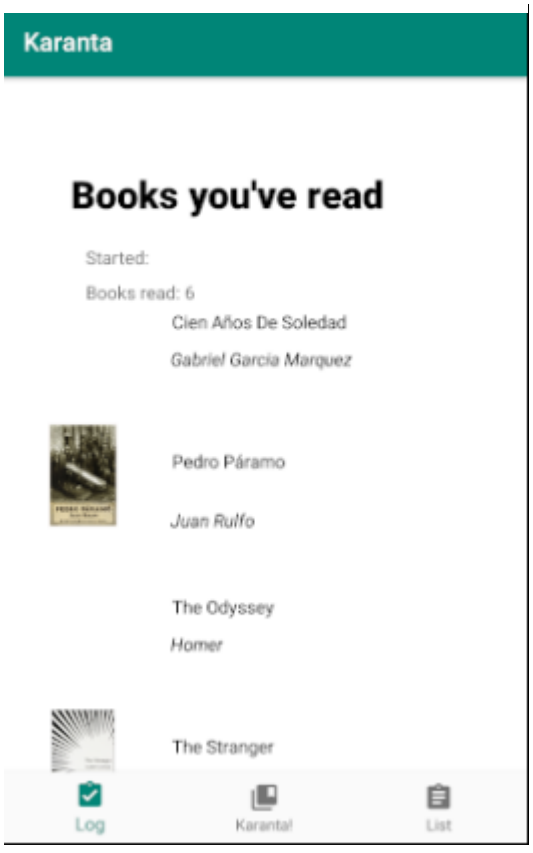


Figure 3 – Karanta's log activity

The application was composed of 3 activities: a log of the books the user had read (fig.2), a list of books the user wanted to read in the future (fig.3) and finally an activity that would randomly select the next book for the user to read (fig.1).

Annex J

Creating an Azure Service

The process to create an SQL database on Microsoft's Azure Cloud that an Android device can access is not as straight forward as one might think. The following document proposes a step-by-step guide on how such an instance is created.

1. **Creation of an SQL Server**

   The developer attempting to create an SQL database must first instantiate an SQL server within the Azure structure. In order to do this, the developer must provide a unique server name as well as authentication options in order to be able to access the server once it is created safely. An example of the creation interface is shown in fig. 1

2. **Creation of SQL database within the recently created server**

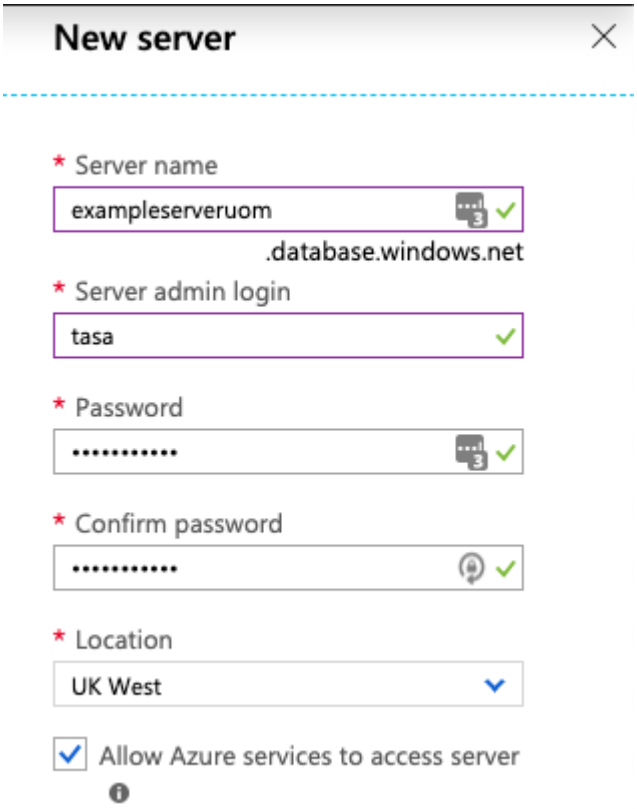


Figure 1

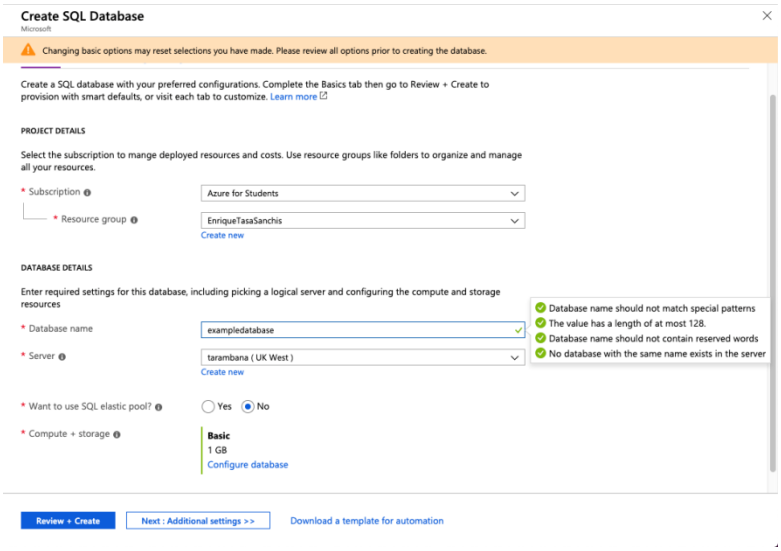


Figure 2

Fig. 2 showcases the interface provided by Azure for a developer to create an SQL database. Developers are prompted to set a name, choose a server where the database will be hosted (must have been created previously, see 1.), a subscription to Azure under which the service is purchased and finally a storage size.

3. Creating a “Web App” in order to have a RESTful API

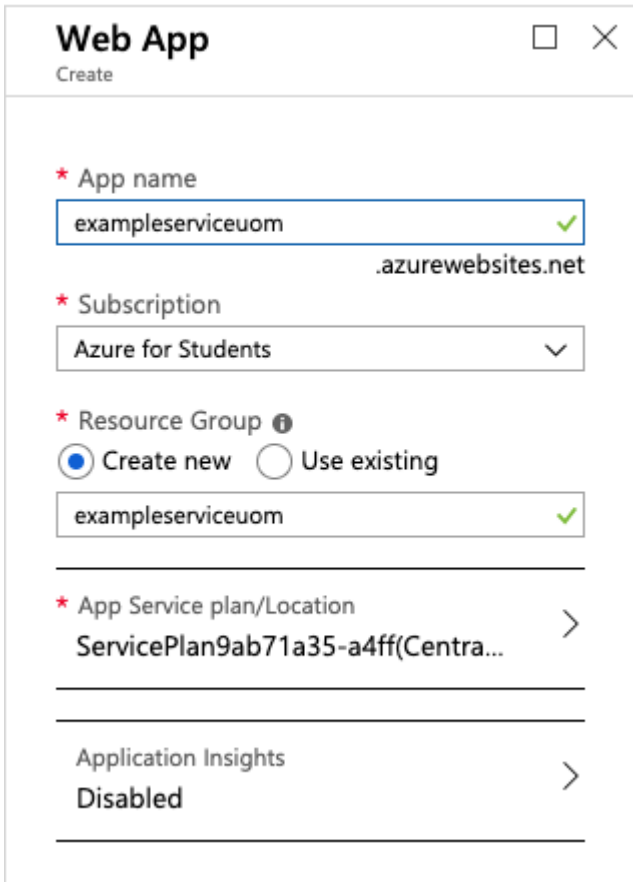


Figure 3

Once the database has been created and hosted on a server, the developer must create a “web app” or mobile service in order for Azure to provide an API and connection between Android apps and the cloud database. The developer must provide a name, a subscription and a resource group (fig.3) name and then connect the web app to a SQL database in the data connections section (fig.4).

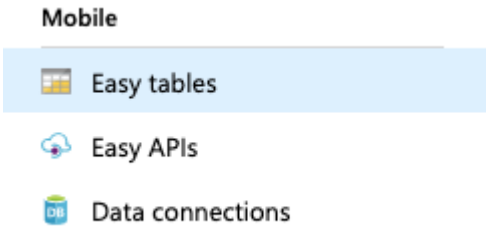


Figure 4

Annex K

Sample XML Mark Scheme

```
<assignment>
	<assignmentNumber>1</assignmentNumber>
	<assignmentName>Getting to grips with the PIC18</assignmentName>
	<assignmentAvailableMarks>9</assignmentAvailableMarks>
	<assignmentAuthor>Tasa</assignmentAuthor>
	<LabGroupLocation>Barnes Wallis PC Cluster</LabGroupLocation>
	<LabGroupUnit>Microcontroller Engineering II</LabGroupUnit>
	<LabGroupNumber>1</LabGroupNumber>
	<LabGroupStudentID>9673164</LabGroupStudentID>
	<markScheme>
		<section>
			<sectionNumber>1</sectionNumber>
			<sectionName>Attendance</sectionName>
			<sectionAvailableMarks>1</sectionAvailableMarks>
			<sectionAuthor>Tasa</sectionAuthor>
			<part>
				<partNumber>6</partNumber>
				<partName>Student Present</partName>
				<partAvailableMarks>1</partAvailableMarks>
				<partAuthor>Tasa</partAuthor>
			</part>
		</section>
		<section>
			<sectionNumber>2</sectionNumber>
			<sectionName>Basic Use of the PIC18</sectionName>
			<sectionAvailableMarks>3</sectionAvailableMarks>
			<sectionAuthor>Tasa</sectionAuthor>
			<part>
				<partNumber>7</partNumber>
				<partName>Student can build a program on the
IDE</partName>
				<partAvailableMarks>1</partAvailableMarks>
				<partAuthor>Tasa</partAuthor>
			</part>
			<part>
				<partNumber>8</partNumber>
				<partName>Student can run a program on the
PIC18</partName>
				<partAvailableMarks>2</partAvailableMarks>
				<partAuthor>Tasa</partAuthor>
			</part>
		</section>
		<section>
			<sectionNumber>3</sectionNumber>
			<sectionName>Programming the PIC18</sectionName>
			<sectionAvailableMarks>3</sectionAvailableMarks>
			<sectionAuthor>Tasa</sectionAuthor>
			<part>
				<partNumber>9</partNumber>
				<partName>Student can display a binary value on the
LEDs</partName>
				<partAvailableMarks>2</partAvailableMarks>
				<partAuthor>Tasa</partAuthor>
			</part>
			<part>
				<partNumber>10</partNumber>
				<partName>Student can display a numerical value on the
7SEG</partName>
				<partAvailableMarks>3</partAvailableMarks>
				<partAuthor>Tasa</partAuthor>
			</part>
		</section>
	</markScheme>
</assignment>
```

Annex L

Testing

Testing procedures for software applications can have two types: static and dynamic. Static testing is defined as a check of the syntactical correctness of code, it is the type of testing that is done by an IDE continuously – every time a line of code is written, the IDE will check for the correctness of that code against the specification and documentation and notifies the user if there is an error of some sort.

Dynamic testing is defined as the tests that take place one an application has been compiled and running. There are many types of dynamic testing, from automated tests that can be programmed with the aid of IDEs to physical testing done by users of the application attempting to misuse it in one way or another.

All the applications developed as a part of this project have been tested both statically (by Android Studio or Visual Studio while development was ongoing) or dynamically once the application had been compiled and was being tested in hardware. The following sections illustrate the testing procedures undertaken in the development of MarkIt Android and MarkIt Desktop once the projects were complete, the dynamic testing undertaken to ensure code safety and stability of the solution.

Mark It Android

The test cases put forward for the Android application were limited in their quantity: due to the design of the application, it had been made very difficult for the user to encounter a case in which the application failed in a way that would bring malfunction. Rather, the application was built to be unresponsive in case of an error occurring – this means the application's main screen will not allow the user to go forward unless all the parameters in the cloud and within it are working (drop boxes will not refresh unless data is downloaded correctly and thus will not allow the user to go forward in the application's operation, etc). In this way, the testing done for the Android application was limited to a monkey test and tests regarding issues external to the application's code itself such as issues with internet connections and simultaneous access of the database.

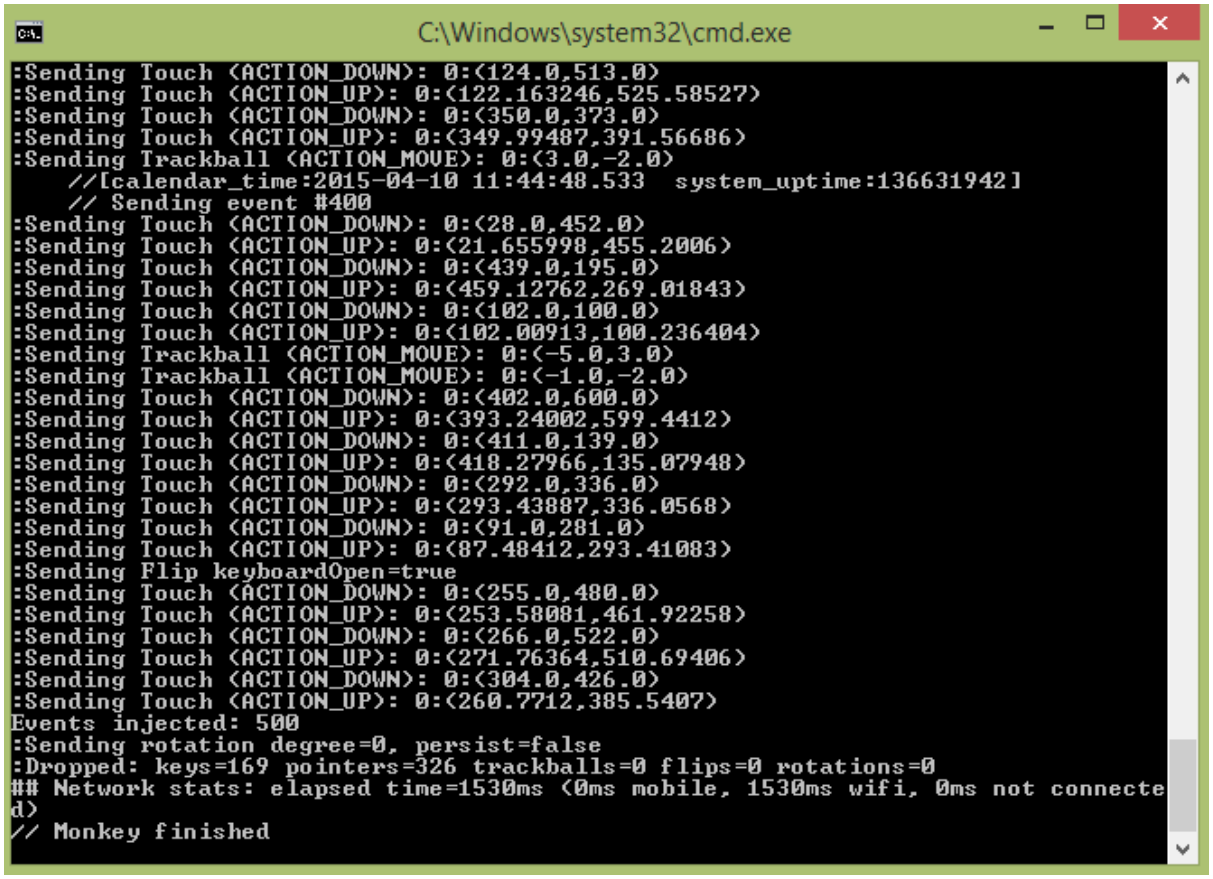
Test case: Android monkey test

The first test case brought forward is a standard when developing mobile applications: the monkey test. A monkey test consists of feeding the device running the developed applications a long stream of completely random user inputs, ranging from swipes to taps scrolls or button presses. This allows developers to see how the application responds to a brute-force style test case and what potentially dangerous elements are present in a UI that could modify the application's functionality.

The monkey test is usually performed through a programme designed to do it automatically, which conveniently is installed with Android Studio. The student, therefore, used the integrated tools to perform a monkey test on the Android application.

The results of the test confirmed what the student suspected: given that the application had been designed so that events happened sequentially (specifically, in order to access any information, a user must select from drop boxes that only populate once the previous one has been selected) the

application isn't prone to malfunctioning or changes in the database via random clicks, as it is difficult that a non-human user would figure out the sequence needed to unlock the application's functionality. In this way, the design of the application deals with the issues that could be brought up by a monkey test.



This terminal window shows the procedure undertaken during a Monkey Test, and how a set of random touches, taps and system events are fed into the device.[1]

Test case: No internet connection on Android device

A second test case was put forward where the device running the application was not connected to the internet. Originally, the application's behaviour in response to this was not acceptable – given that nothing would happen as no information could be downloaded from the cloud, but the user was not notified of any issues.

The application was modified to showcase a toast informing the user of an issue with the internet connection and asking them to connect to the internet and retry. Although this is not an automated solution, it is the best the application can do to get back to a working state – ask the user to connect to the internet. Fig. 1 shows the notification shown to the user.

[1] Image taken from tutorialspoint.com, used in researching Monkey Test procedure

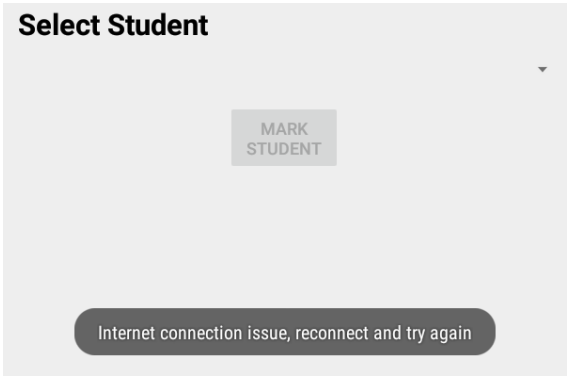


Fig.1

Test case: Simultaneous users accessing records on one student

The last test case that was brought forward for the Android application entailed checking that simultaneous use of the application in various devices was viable. This was especially important not only as a test for the functioning of the application but also as a test of the features that the student wanted to include in the app.

In this way, the application was installed on 2 Android devices and a marking session was started with identical student IDs and assignment numbers. The result of the test was that the system saves both marking sessions in the database and will output the latest one as the final valid mark for a student. This is due to the design of the application especially in grade download but also due to Azure's cloud capabilities which optimise simultaneous connections to an SQL database.

The student, therefore, relied on Azure's infrastructure for the correct functioning of the SQL connection and had to write code to output the correct (latest) grade for a given student.

Mark It Windows

Test case: Invalid file location selected for download or upload

The user of the Windows application is free to select any file location available to them via the FileSelectDialog box that appears when the user clicks the file selection button. The program does not restrict the user selection in any way, but the Windows API provides a warning signal and stops the functioning of the program in response to the user selecting an invalid location.

In this way, this test passes because the OS the application runs on provides a safety net to stop users from selecting an invalid file location. The student, therefore, relies on the OS's handling of errors rather than having provided an explicit solution for it.

Test case: Attempt to overwrite an already existing file

When downloading the CSV file containing the student ID and grade information, the user might select to overwrite a file that had been downloaded previously, and the application will use the same file location and name the downloaded file identically. This would normally create an overwrite error. Again, the Windows OS API provides an error handling mechanism that this application takes advantage of.

Before overwriting a file, the OS will notify the user that this is about to happen and will pop up a message box requiring user confirmation in order to go forward. Once again, the student relies on the OS's function to provide for this error's handling.

Test case: Issues in reading XML file

The Windows application requires reading of an input XML file containing the Mark Scheme information (as well as student, lab group and assignment number data) in order to upload to the online database. A test case was brought forward simulating an invalid XML or one that was missing information.

Initially, the application had no manner of dealing with such an issue and would upload the parts that it could to the cloud, leaving certain parts of a mark scheme's information in the database unfilled. This had to be rectified in order to ensure that the information was always correctly and fully uploaded.

The student made the procedure of uploading each section of the mark scheme dependent on the previous part's upload: by modifying the methods that carry out the upload to return a Boolean value that indicated success or failure, the student was able to only proceed to the next step of uploading a database once the previous one was completed successfully. Furthermore, functionality was added to advise the user on which exact part of the XML had prompted the failure and asked the user to rectify it and try again. A sample of the code is seen below.

```
1.  if (XMLManager.GetMarkSchemeInfo(pathToXML, ref markSchemeInfo))
2.  {
3.      if (XMLManager.GetLabGroupInfo(pathToXML, ref labGroupInfo))
4.      {
5.          assignmentID = markSchemeInfo.markSchemeAssignmentID;
6.
7.          if (XMLManager.GetMarkSchemeSectionsInfo(pathToXML, assignmentID, ref markS
    chemeSectionsInfo))
8.          {
9.              if (XMLManager.GetMarkSchemePartsInfo(pathToXML, assignmentID, ref mark
    SchemePartsInfo))
10.             {
11.
12.             }
13.             else
14.             {
15.                 MessageBox.Show("Error in reading parts from XML, check file and re
    try");
16.             }
17.         }
18.         else
19.         {
20.             MessageBox.Show("Error in sections parts from XML, check file and retry
    ");
21.         }
22.     }
23.     else
24.     {
25.         MessageBox.Show("Error in reading lab group information from XML, check fil
    e and retry");
26.     }
27. }
```

```
28. else
29. {
30.     MessageBox.Show("Error in reading mark scheme information from XML, check file
    and retry");
31. }
```

Test case: Database upload failure

When working with databases it is common to experience problems in connecting to them, to experience network issues or to have problems with authentication. This application would be no different, and a mechanism had to be provided to solve these issues.

A test case was brought forward to test the application's behaviour in case of an internet connection failure or a database connection failure. Originally, the app responded by not uploading the mark scheme but also not notifying the user of any issues.

The application was modified to go about the mark scheme upload in a manner similar to that of the XML file reading: only proceeding to the next step of upload once the previous one had completed successfully (checking that this was so via the return of Boolean values). User messages were also added so that the user was notified of the error that had occurred and was prompted to solve it and retry.

Annex M

Risk Assessment

MANCHESTER 1824
The University of Manchester

General Risk Assessment Form

Date: (1)	**Assessed by:** (2)	**Checked / Validated* by:** (3)	**Location:** (4)	**Assessment ref no** (5)	**Review date:** (6)
2018/10/17	**Enrique Tasa**		**Computer Clusters UoM**		**2019/05/10**

Task / premises: (7)

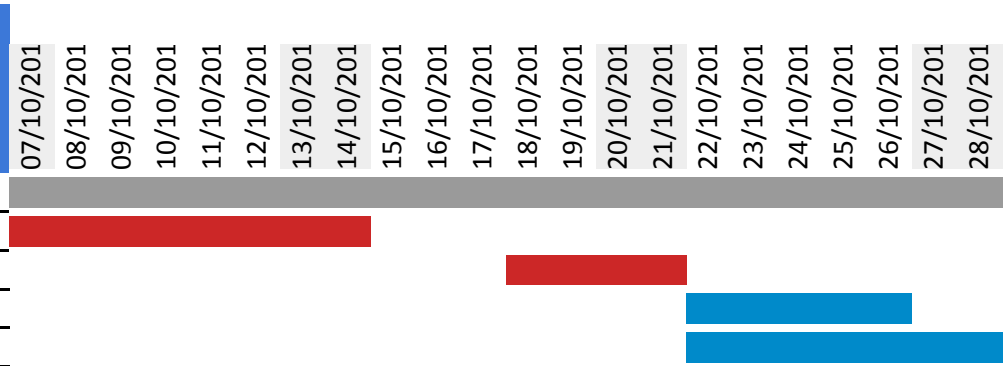
Developing an Android Application (Programming on a computer)

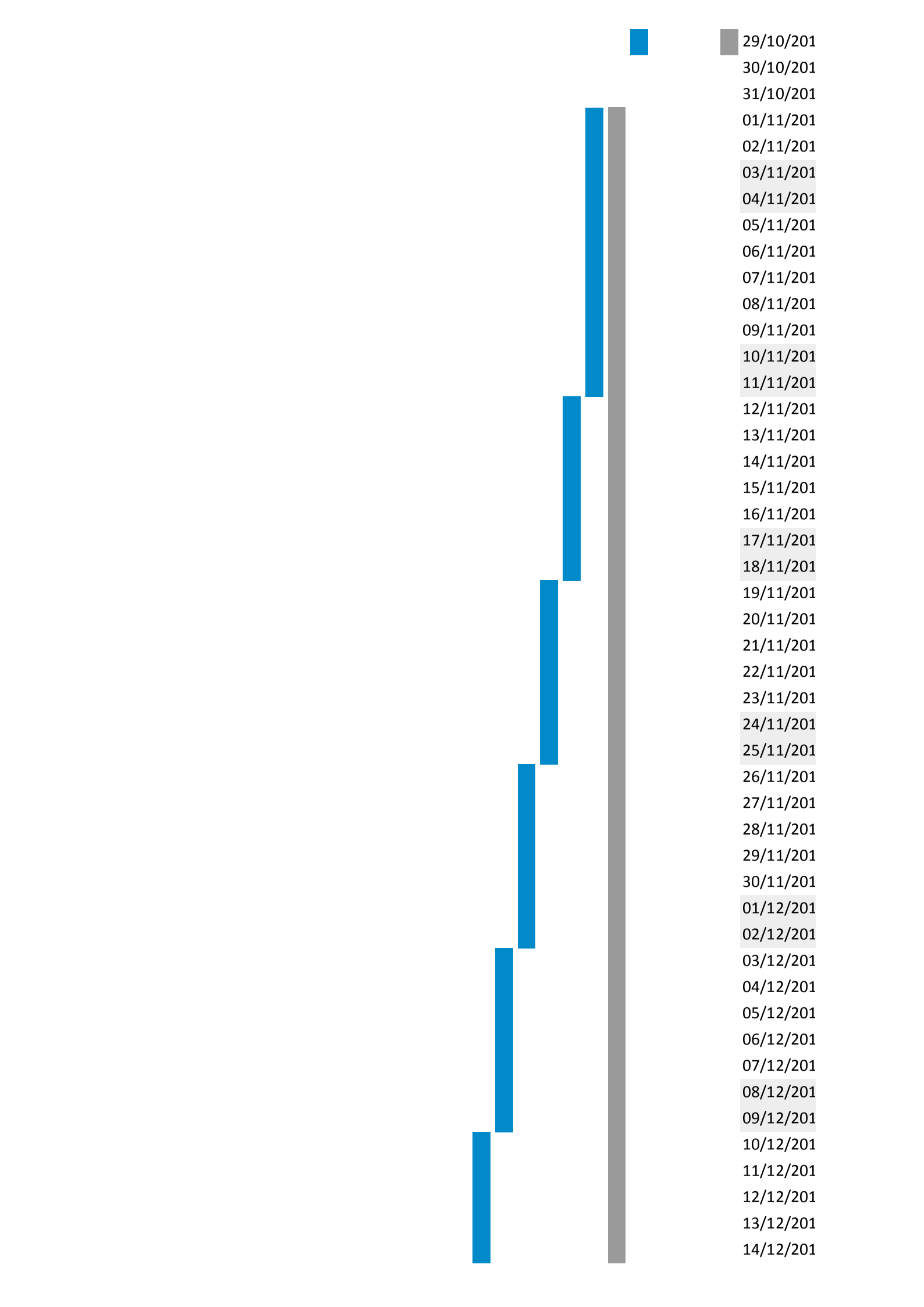
Activity (8)	**Hazard** (9)	**Who might be harmed and how** (10)	**Existing measures to control risk** (11)	**Risk rating** (12)	**Result** (13)
Programming on a computer sitting at a desk	Eye strain	Student Enrique Tasa, eye strain	Reduced blue-light mode on screens, blue-light filtering lenses on student's glasses, frequent breaks	LOW	A
Programming on a computer sitting at a desk	Ergonomic, posture issues	Student Enrique Tasa, muscle strains and cramps	Student to take regular (every 20 minutes) breaks and include walks and stretches into work routine. Desk and computer positioning favours correct posture.	LOW	A
Working with computers	Electrical	Student Enrique Tasa	Student to use correct connectors and insulated cables, not bring drinks near electrical equipment and ask for the help of a member of staff if needed to move or connect/disconnect appliances	LOW	T

Annex N

Gantt Chart for Project

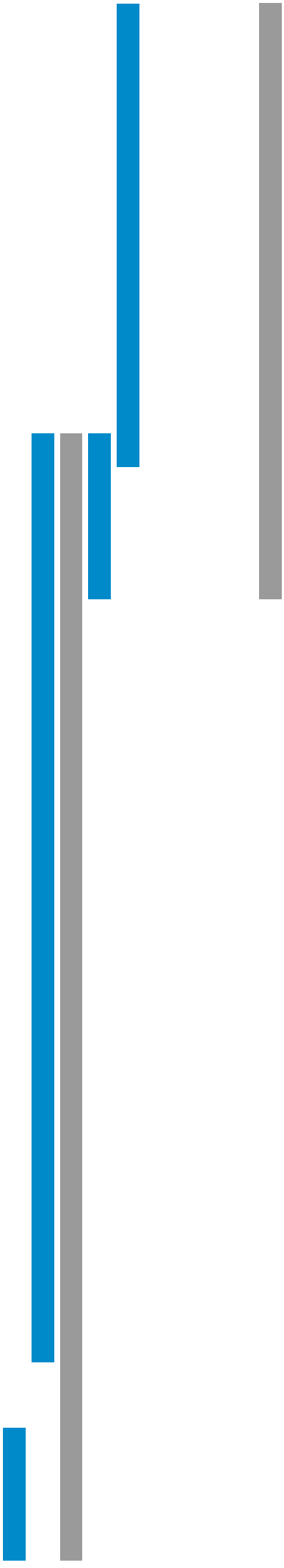
Project Tasks			
Name	Start	Due	Estimated Hours
Planning and research			15
Project Proposal	07/10/2018	14/10/2018	1
Risk Assessment	18/10/2018	21/10/2018	1
Project Plan	22/10/2018	26/10/2018	3
Becoming familiar with Java - Writing an app	22/10/2018	29/10/2018	10
Base Objectives			99
Cloud save of marks data	01/11/2018	11/11/2018	20
Real time grade consultation	12/11/2018	18/11/2018	15
Form creator rewrite	19/11/2018	25/11/2018	20
From creator cloud upload	26/11/2018	02/12/2018	15
Template mark scheme save	03/12/2018	09/12/2018	10
Simultaneous users access cloud data	10/12/2018	14/12/2018	10
Contingency (10% of total section time)	15/12/2018	27/12/2018	9
January exams and preparation	28/12/2018	25/01/2019	-
Stretch Objectives			121
Various lecture templates	26/01/2019	01/02/2019	10
"Assignments" class rewrite	03/02/2019	17/02/2019	10
"Referrals" class rewrite	03/02/2019	17/02/2019	10
Student consult service	17/02/2019	24/02/2019	20
Web interface	24/02/2019	01/04/2019	60
Contingency (10% of total section time)	01/04/2019	05/04/2019	11
Deliverables			40
Final Report	01/04/2019	28/04/2019	30
Oral Examination & Poster	01/05/2019	12/05/2019	10




29/10/201
30/10/201
31/10/201
01/11/201
02/11/201
03/11/201
04/11/201
05/11/201
06/11/201
07/11/201
08/11/201
09/11/201
10/11/201
11/11/201
12/11/201
13/11/201
14/11/201
15/11/201
16/11/201
17/11/201
18/11/201
19/11/201
20/11/201
21/11/201
22/11/201
23/11/201
24/11/201
25/11/201
26/11/201
27/11/201
28/11/201
29/11/201
30/11/201
01/12/201
02/12/201
03/12/201
04/12/201
05/12/201
06/12/201
07/12/201
08/12/201
09/12/201
10/12/201
11/12/201
12/12/201
13/12/201
14/12/201


15/12/201
16/12/201
17/12/201
18/12/201
19/12/201
20/12/201
21/12/201
22/12/201
23/12/201
24/12/201
25/12/201
26/12/201
27/12/201
28/12/201
29/12/201
30/12/201
31/12/201
01/01/201
02/01/201
03/01/201
04/01/201
05/01/201
06/01/201
07/01/201
08/01/201
09/01/201
10/01/201
11/01/201
12/01/201
13/01/201
14/01/201
15/01/201
16/01/201
17/01/201
18/01/201
19/01/201
20/01/201
21/01/201
22/01/201
23/01/201
24/01/201
25/01/201
26/01/201
27/01/201
28/01/201
29/01/201
30/01/201

31/01/201
01/02/201
02/02/201
03/02/201
04/02/201
05/02/201
06/02/201
07/02/201
08/02/201
09/02/201
10/02/201
11/02/201
12/02/201
13/02/201
14/02/201
15/02/201
16/02/201
17/02/201
18/02/201
19/02/201
20/02/201
21/02/201
22/02/201
23/02/201
24/02/201
25/02/201
26/02/201
27/02/201
28/02/201
01/03/201
02/03/201
03/03/201
04/03/201
05/03/201
06/03/201
07/03/201
08/03/201
09/03/201
10/03/201
11/03/201
12/03/201
13/03/201
14/03/201
15/03/201
16/03/201
17/03/201
18/03/201



19/03/201
20/03/201
21/03/201
22/03/201
23/03/201
24/03/201
25/03/201
26/03/201
27/03/201
28/03/201
29/03/201
30/03/201
31/03/201
01/04/201
02/04/201
03/04/201
04/04/201
05/04/201
06/04/201
07/04/201
08/04/201
09/04/201
10/04/201
11/04/201
12/04/201
13/04/201
14/04/201
15/04/201
16/04/201
17/04/201
18/04/201
19/04/201
20/04/201
21/04/201
22/04/201
23/04/201
24/04/201
25/04/201
26/04/201
27/04/201
28/04/201
29/04/201
30/04/201
01/05/201
02/05/201
03/05/201
04/05/201

05/05/201
06/05/201
07/05/201
08/05/201
09/05/201
10/05/201
11/05/201
12/05/201

Annex O

MarkIt Android Code

Android Mark It Code

Mark It Android UI Layout XMLs

MainActivity XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/app_bar_main">

    <TextView
        android:id="@+id/SelectUnitTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_marginStart="16dp"
        android:layout_marginTop="24dp"
        android:text="@string/SelectUnitTV"
        android:textColor="@android:color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Spinner
        android:id="@+id/SelectUnitSp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_margin="16dp"
        android:ellipsize="marquee"
        android:textAlignment="inherit"
        android:textColor="@color/colorAccent"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="@+id/SelectUnitTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectUnitTV" />

    <TextView
        android:id="@+id/SelectAssignmentTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="60dp"
```

```
        android:text="@string/SelectAssignmentTV"
        android:textColor="@android:color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="@+id/SelectUnitTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectUnitSp" />

    <Spinner
        android:id="@+id/SelectAssignmentSp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_margin="16dp"
        android:textColor="@android:color/black"
        android:ellipsize="marquee"
        android:textAlignment="inherit"
        android:textStyle="bold"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="@+id/SelectAssignmentTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectAssignmentTV" />

    <TextView
        android:id="@+id/SelectGroupTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="60dp"
        android:text="@string/SelectGroupTV"
        android:textColor="@android:color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="@+id/SelectAssignmentTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectAssignmentSp" />

    <Spinner
        android:id="@+id/SelectGroupSp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_margin="16dp"
        android:ellipsize="marquee"
        android:textAlignment="inherit"
        android:textStyle="bold"
        android:textSize="20sp"
        android:textColor="@android:color/black"
        app:layout_constraintStart_toStartOf="@+id/SelectGroupTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectGroupTV" />

    <TextView
        android:id="@+id/SelectStudentTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="60dp"
        android:text="@string/SelectStudentTV"
        android:textColor="@android:color/black"
        android:textSize="25sp"
```

```
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="@+id/SelectGroupTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectGroupSp" />

    <Spinner
        android:id="@+id/SelectStudentSp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_margin="16dp"
        android:ellipsize="marquee"
        android:textAlignment="inherit"
        android:textStyle="bold"
        android:textSize="20sp"
        android:textColor="@android:color/black"
        app:layout_constraintStart_toStartOf="@+id/SelectStudentTV"
        app:layout_constraintTop_toBottomOf="@+id/SelectStudentTV" />

    <Button
        android:id="@+id/MarkStudentBtn"
        android:layout_width="120dp"
        android:layout_height="72dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="8dp"
        android:enabled="false"
        android:text="@string/Mark_Student_Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/SelectStudentSp" />

</android.support.constraint.ConstraintLayout>
```

Mark Student XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_mark_student"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".MarkStudent">

    <android.support.design.widget.CoordinatorLayout
        android:layout_width="match_parent"
    android:layout_height="match_parent">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.design.widget.AppBarLayout
```

```
            android:id="@+id/appbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/AppTheme.AppBarOverlay">

            <android.support.v7.widget.Toolbar
                android:id="@+id/toolbarMark"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary"
                app:layout_scrollFlags="scroll|enterAlways"
                app:popupTheme="@style/AppTheme.PopupOverlay"
                app:title="@string/app_name">


            </android.support.v7.widget.Toolbar>

            <android.support.design.widget.TabLayout
                android:id="@+id/sliding_tabs"
                android:layout_width="match_parent"
                android:layout_height="50dp"
                app:tabMode="fixed" />

        </android.support.design.widget.AppBarLayout>

        <android.support.v4.view.ViewPager
            android:id="@+id/container"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_above="@id/SaveBtn"
            android:layout_alignParentStart="true"
            android:layout_alignParentLeft="true"
            android:layout_marginStart="0dp"
            android:layout_marginLeft="0dp"
            android:layout_marginBottom="-61dp"
            app:layout_behavior="@string/appbar_scrolling_view_behavior" />

        <Button
            android:id="@+id/SaveBtn"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentBottom="true"
            android:layout_marginStart="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="10dp"
            android:layout_marginBottom="10dp"
            android:paddingRight="10dp"
            android:text="@string/Save_Button_Text" />

        <TextView
            android:id="@+id/TotalMarksTV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginRight="11dp"
            android:layout_marginBottom="25dp"
            android:layout_toLeftOf="@+id/SaveBtn"
            android:gravity="center"
            android:hint="@string/Total_Marks_Text"
```

```
            android:textSize="14sp" />

        <TextView
            android:id="@+id/StudentIDTV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentBottom="true"
            android:layout_marginStart="10dp"
            android:layout_marginLeft="100dp"
            android:layout_marginEnd="66dp"
            android:layout_marginRight="0dp"
            android:layout_marginBottom="25dp"
            android:layout_toStartOf="@+id/TotalMarksTV"
            android:hint="@string/Student_ID_Hint"
            android:text="@string/ID_Tag_Text"
            android:textSize="14sp" />

    </RelativeLayout>


    </android.support.design.widget.CoordinatorLayout>

    <android.support.design.widget.NavigationView
android:id="@+id/nav_view"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:layout_gravity="start"
android:fitsSystemWindows="true"
app:menu="@menu/activity_main_drawer"
app:headerLayout="@layout/nav_header_main"/>

    </android.support.v4.widget.DrawerLayout>
```

Grade Consultation XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/activity_grade_consult"
    tools:context=".GradeConsult">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.design.widget.AppBarLayout
            android:id="@+id/appbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
```

```
            android:id="@+id/toolbarGC"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.AppBarLayout>

    <TextView
        android:id="@+id/GCSelectStudentTV"
        android:layout_width="wrap_content"
        android:layout_height="30dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_marginStart="16dp"
        android:layout_marginTop="84dp"
        android:text="@string/SelectStudentTV"
        android:textColor="@android:color/black"
        android:textSize="25sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

        <Spinner
            android:id="@+id/GCSelectStudentSP"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentLeft="true"
            android:layout_margin="16dp"
            android:layout_marginStart="4dp"
            android:layout_marginTop="60dp"
            android:textColor="@android:color/black"
            android:textSize="20sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/GCSelectStudentTV"
            app:layout_constraintTop_toBottomOf="@+id/GCSelectStudentTV" />

        <TextView
            android:id="@+id/GCSelectAssignmentTV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentLeft="true"
            android:layout_marginTop="52dp"
            android:text="@string/SelectAssignmentTV"
            android:textColor="@android:color/black"
            android:textSize="25sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/GCSelectStudentSP"
            app:layout_constraintTop_toBottomOf="@+id/GCSelectStudentSP" />

        <Spinner
            android:id="@+id/GCSelectAssignmentSp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentLeft="true"
            android:layout_margin="16dp"
            android:layout_marginTop="28dp"
            android:ellipsize="marquee"
```

```
            android:textAlignment="inherit"
            android:textColor="@android:color/black"
            android:textSize="20sp"
            android:textStyle="bold"

app:layout_constraintStart_toStartOf="@+id/GCSelectAssignmentTV"
            app:layout_constraintTop_toBottomOf="@+id/GCSelectAssignmentTV"
/>

        <Button
            android:id="@+id/ConsultGradeBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="120dp"
            android:layout_marginTop="88dp"
            android:enabled="true"
            android:text="@string/Consult_Grade_Button"
            android:visibility="visible"

app:layout_constraintStart_toStartOf="@+id/GCSelectAssignmentSp"
            app:layout_constraintTop_toBottomOf="@+id/GCSelectAssignmentSp"
/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/StudentGradeInfo"
        app:layout_constraintStart_toStartOf="@id/ConsultGradeBtn"
        app:layout_constraintTop_toBottomOf="@id/ConsultGradeBtn"
        android:layout_marginTop="50dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:textSize="15sp" />

    </android.support.constraint.ConstraintLayout>

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:menu="@menu/activity_main_drawer"
        app:headerLayout="@layout/nav_header_main"/>

</android.support.v4.widget.DrawerLayout>
```

Mark Student XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/constraintLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <!-- Dummy item to prevent AutoCompleteTextView from receiving focus --
>
```

```
    <LinearLayout
        android:layout_width="0px"
        android:layout_height="0px"
        android:focusable="true"
        android:focusableInTouchMode="true" />

    <!-- :nextFocusUp and :nextFocusLeft have been set to the id of this
component
    to prevent the dummy from receiving focus again -->
    <AutoCompleteTextView
        android:id="@+id/autotext"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:nextFocusLeft="@id/autotext"
        android:nextFocusUp="@id/autotext" />

    <TextView
        android:id="@+id/section_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/activity_horizontal_margin"
        android:layout_marginTop="@dimen/activity_vertical_margin"
        android:layout_marginEnd="@dimen/activity_horizontal_margin"
        android:layout_marginBottom="@dimen/activity_vertical_margin"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="@+id/constraintLayout"
        tools:layout_constraintLeft_creator="1"
        tools:layout_constraintTop_creator="1" />

    <EditText
        android:id="@+id/CommentsET"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="56dp"
        android:layout_marginTop="788dp"
        android:ems="10"
        android:hint="@string/Comments_Hint"
        android:inputType="text"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <CheckBox
        android:id="@+id/markCB1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="@+id/sectionTitleTV"
        app:layout_constraintTop_toBottomOf="@+id/sectionTitleTV" />

    <CheckBox
        android:id="@+id/markCB2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="@+id/markCB1"
        app:layout_constraintTop_toBottomOf="@+id/markCB1" />

    <CheckBox
        android:id="@+id/markCB3"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="@+id/markCB1"
        app:layout_constraintTop_toBottomOf="@+id/markCB2" />

    <CheckBox
        android:id="@+id/markCB4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="@+id/markCB1"
        app:layout_constraintTop_toBottomOf="@+id/markCB3" />

    <CheckBox
        android:id="@+id/markCB5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="@+id/markCB1"
        app:layout_constraintTop_toBottomOf="@+id/markCB4" />


    <TextView
        android:id="@+id/sectionTitleTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="140dp"
        android:textColor="@android:color/black"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Mark It Main Activity Logic Java Code

```
package com.tarambana.markit;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;

import com.microsoft.windowsazure.mobileservices.MobileServiceClient;
import
com.microsoft.windowsazure.mobileservices.http.ServiceFilterResponse;
import com.microsoft.windowsazure.mobileservices.table.TableQueryCallback;
import com.microsoft.windowsazure.mobileservices.table.query.QueryOrder;
import com.tarambana.markit.DataContainers.LabGroup;

import java.net.MalformedURLException;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

    static public String TAG = "TASA_LOG ";

    // Object declared to interact with Azure cloud
    private MobileServiceClient mClientAzureConnection;

    int failCountAzureConnection = 0;

    boolean firstTimeUnitDropdownRefreshed = true;
    boolean firstTimeAssignmentDropdownRefreshed = true;
    boolean firstTimeGroupDropdownRefreshed = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        try {
            Log.d(TAG, "MainActivity: attempting to connect to Azure
site");

            mClientAzureConnection = new MobileServiceClient(
                    "https://bookchoice.azurewebsites.net",
                    this
            );
```

```
            Log.d(TAG, "MainActivity: success in connecting to Azure");

        } catch (MalformedURLException e){
            Log.d(TAG, "MainActivity: malformed URL in connection to Azure
site");
            Toast.makeText(getApplicationContext(), "Internet connection
issue, reconnect and try again", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }

        Log.d(TAG, "MarkStudent: bundle retrieved correctly in 2nd
activity");
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.getMenu().getItem(0).setChecked(true);

        Log.d(TAG, "MainActivity: toolbar, drawer and navigation view setup
correctly");

        // Define what happens when the button is clicked (open MarkStudent
activity, sending it selected student and assignment data)
        Button MarkStudentBtn = (Button) findViewById(R.id.MarkStudentBtn);
        MarkStudentBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Spinner unitSpinner = (Spinner)
findViewById(R.id.SelectUnitSp);
                Spinner assignmentSpinner = (Spinner)
findViewById(R.id.SelectAssignmentSp);
                Spinner groupSpinner = (Spinner)
findViewById(R.id.SelectGroupSp);
                Spinner studentSpinner = (Spinner)
findViewById(R.id.SelectStudentSp);

                // Intent built and keys added to it to specify the course,
assignment, group and student ID selected for marking. This data will reach
the newly started activity and will be used there.
                Intent intent = new Intent(getApplicationContext(),
MarkStudent.class);
                Bundle selectionInfo = new Bundle();
                selectionInfo.putString("unit",
unitSpinner.getSelectedItem().toString());
                selectionInfo.putInt("assignment",
(int)assignmentSpinner.getSelectedItem());
                selectionInfo.putInt("group",
(int)groupSpinner.getSelectedItem());
                selectionInfo.putInt("studentID",
```

```
(int)studentSpinner.getSelectedItem());
                intent.putExtras(selectionInfo);
                Log.d(TAG, "MainActivity: going to MarkStudent activity
with assignment " + (int)assignmentSpinner.getSelectedItem());
                startActivity(intent);
            }
        });

        SetUpSpinners();

        refreshUnitDropDownWithCloudData("deleted","false");
    }

    // This method will set up the dropdown menus and refresh them
sequentially when the user selects data from them
    private void SetUpSpinners() {
        final Spinner unitSpinner = (Spinner)
findViewById(R.id.SelectUnitSp);
        final Spinner assignmentSpinner = (Spinner)
findViewById(R.id.SelectAssignmentSp);
        final Spinner groupSpinner = (Spinner)
findViewById(R.id.SelectGroupSp);
        final Spinner studentSpinner = (Spinner)
findViewById(R.id.SelectStudentSp);

        // When a value is selected in the spinner
        unitSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                // If this is not the first time a value is selected,
download the data (first time happens when UI is drawn, this is just a way
of guaranteeing user input in populating these)
                if (!firstTimeUnitDropdownRefreshed){
                refreshAssignmentDropDownWithCloudData("LABGROUPUNIT",
unitSpinner.getSelectedItem().toString());
                }

                firstTimeUnitDropdownRefreshed = false;
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        assignmentSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                if (!firstTimeAssignmentDropdownRefreshed){
                refreshGroupDropDownWithCloudData("LABGROUPASSIGNMENTID",
assignmentSpinner.getSelectedItem().toString());
              }

              firstTimeAssignmentDropdownRefreshed = false;
            }
```

```
            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        groupSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                if (!firstTimeGroupDropdownRefreshed){
                refreshStudentDropDownWithCloudData("LABGROUPNUMBER",
"LABGROUPASSIGNMENTID", groupSpinner.getSelectedItem().toString(),
assignmentSpinner.getSelectedItem().toString());
                }

                firstTimeGroupDropdownRefreshed = false;
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        studentSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                Button markStudentButton = (Button)
findViewById(R.id.MarkStudentBtn);
                markStudentButton.setEnabled(true);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });
    }

    // This method downloads a "LabGroup" class with the information
required from the Azure SQL database, and allows that class to be inspected
and manipulated for use of the data it contains. The other methods below
with similar names work the same but download different types of data, and
thus different classes
    void refreshUnitDropDownWithCloudData(String field, String condition){

        Log.d(TAG, "MainActivity: populating course unit dropdown with
cloud data");

        // This line is Azure's adaptor's way of executing an SQL query:
the LabGroup class downloaded contains the information where the passed
field meets the passed condition

mClientAzureConnection.getTable(LabGroup.class).where().field(field).eq(con
dition)
                .execute(new TableQueryCallback<LabGroup>() {
```

```
                    // Listener for the result of the transaction with
Azure
                    @Override
                    public void onCompleted(java.util.List<LabGroup>
resultLabGroup, int count, Exception exception, ServiceFilterResponse
response) {

                        // If there's no exception and all goes well, set
up the spinner from the result of the transaction
                        if (exception == null) {

                            List<String> unitSpinnerList = new
ArrayList<>();
                            unitSpinnerList.add("Please select a unit");

                            Log.d(TAG, "MainActivity: course unit spinner
data successfully retrieved from Azure");

                            for (LabGroup unit : resultLabGroup) {
                                if
(!(unitSpinnerList.contains(unit.getLabGroupUnit()))) {

unitSpinnerList.add(unit.getLabGroupUnit());
                                }
                            }

                            ArrayAdapter<String> adapter = new
ArrayAdapter<>(
                                    getBaseContext(),
android.R.layout.simple_spinner_item, unitSpinnerList);


adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                            Spinner unitSpinner = (Spinner)
findViewById(R.id.SelectUnitSp);
                            unitSpinner.setAdapter(adapter);
                            failCountAzureConnection = 0;
                        }
                        else if (failCountAzureConnection < 3){
                            Log.d(TAG, "MainActivity: reattempting course
unit data download");

refreshUnitDropDownWithCloudData("deleted","false");
                            failCountAzureConnection++;
                        }
                        else {
                            Log.d(TAG, "MainActivity: exception found: " +
exception.getMessage());
                            Toast.makeText(getApplicationContext(),
"Internet connection issue, reconnect and try again",
Toast.LENGTH_LONG).show();
                        }
                    }
                });
    }

    void refreshAssignmentDropDownWithCloudData(String field, String
condition){

        Log.d(TAG, "MainActivity: populating assignment dropdown with cloud
```

```
data");


mClientAzureConnection.getTable(LabGroup.class).where().field(field).eq(con
dition)
                .orderBy("LABGROUPASSIGNMENTID",
QueryOrder.Ascending).execute(new TableQueryCallback<LabGroup>() {

            // Listener that automatically gets set for the result of the
transaction with Azure
            @Override
            public void onCompleted(java.util.List<LabGroup> result, int
count, Exception exception, ServiceFilterResponse response) {

                if (exception == null) {

                    List<Integer> assignmentSpinnerList = new
ArrayList<>();
                    assignmentSpinnerList.add(000000000);

                    Log.d(TAG, "MainActivity: assignment spinner data
successfully retrieved from Azure");

                    for (LabGroup unit : result) {
                        if
(!(assignmentSpinnerList.contains(unit.getLabGroupAssignmentID()))) {

assignmentSpinnerList.add(unit.getLabGroupAssignmentID());
                        }
                    }

                    ArrayAdapter<Integer> adapter = new ArrayAdapter<>(
                            getBaseContext(),
android.R.layout.simple_spinner_item, assignmentSpinnerList);


adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                    Spinner assignmentSpinner = (Spinner)
findViewById(R.id.SelectAssignmentSp);
                    assignmentSpinner.setAdapter(adapter);
                }

                else {
                    Log.d(TAG, "Exception found: " +
exception.getMessage());
                }
            }
        });
    }

    void refreshGroupDropDownWithCloudData(String field, String condition){

        Log.d(TAG, "MainActivity: populating lab group dropdown with cloud
data");


mClientAzureConnection.getTable(LabGroup.class).where().field(field).eq(con
dition)
                .orderBy("LABGROUPNUMBER",
QueryOrder.Ascending).execute(new TableQueryCallback<LabGroup>() {
```

```
            // Listener that automatically gets set for the result of the
transaction with Azure
            @Override
            public void onCompleted(java.util.List<LabGroup> result, int
count, Exception exception, ServiceFilterResponse response) {

                if (exception == null) {

                    List<Integer> groupSpinnerList = new ArrayList<>();
                    groupSpinnerList.add(000000000);

                    Log.d(TAG, "MainActivity: lab group spinner data
successfully retrieved from Azure");

                    for (LabGroup unit : result) {
                        if
(!(groupSpinnerList.contains(unit.getLabGroupNumber()))) {
                            groupSpinnerList.add(unit.getLabGroupNumber());
                        }
                    }

                    ArrayAdapter<Integer> adapter = new ArrayAdapter<>(
                            getBaseContext(),
android.R.layout.simple_spinner_item, groupSpinnerList);


adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                    Spinner groupSpinner = (Spinner)
findViewById(R.id.SelectGroupSp);
                    groupSpinner.setAdapter(adapter);
                }

                else {
                    Log.d(TAG, "Exception found: " +
exception.getMessage());
                }
            }
        });
    }

    void refreshStudentDropDownWithCloudData(String field1, String field2,
String condition1, String condition2){

        Log.d(TAG, "MainActivity: populating student dropdown with cloud
data");


mClientAzureConnection.getTable(LabGroup.class).where().field(field1).eq(co
ndition1).and().field(field2).eq(condition2)
                .orderBy("LABGROUPSTUDENTID",
QueryOrder.Ascending).execute(new TableQueryCallback<LabGroup>() {

            // Listener that automatically gets set for the result of the
transaction with Azure
            @Override
            public void onCompleted(java.util.List<LabGroup> result, int
count, Exception exception, ServiceFilterResponse response) {

                if (exception == null) {
```

```
                    List<Integer> studentSpinnerList = new ArrayList<>();
                    studentSpinnerList.add(000000000);

                    Log.d(TAG, "MainActivity: student spinner data
successfully retrieved from Azure");

                    for (LabGroup unit : result) {
                        if
(!(studentSpinnerList.contains(unit.getLabGroupStudentID()))) {

studentSpinnerList.add(unit.getLabGroupStudentID());
                        }
                    }

                    ArrayAdapter<Integer> adapter = new ArrayAdapter<>(
                            getBaseContext(),
android.R.layout.simple_spinner_item, studentSpinnerList);


adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                    Spinner studentSpinner = (Spinner)
findViewById(R.id.SelectStudentSp);
                    studentSpinner.setAdapter(adapter);
                }

                else {
                    Log.d(TAG, "Exception found: " +
exception.getMessage());
                }
            }
        });
    }

    // System required method
    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    // System required method
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    // System required method
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();
```

```
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // This method determines what happens when something is selected in
the navigation drawer
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {

        int id = item.getItemId();

        if (id == R.id.student_selection) {
            Log.d(TAG, "MainActivity: student selection activity selected
in navigation drawer");
            item.setChecked(true);
        } else if (id == R.id.make_mark_scheme) {
            Log.d(TAG, "MainActivity: make mark scheme activity selected in
navigation drawer");
            item.setChecked(true);
        } else if (id == R.id.check_marks) {
            Log.d(TAG, "MainActivity: grade consult activity selected in
navigation drawer");
            item.setChecked(true);
            Intent myIntent = new Intent(this, GradeConsult.class);
            this.startActivity(myIntent);
        }

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

Mark Student Logic Java Code

```
package com.tarambana.markit;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.design.widget.TabLayout;
import android.support.v4.view.GravityCompat;
import android.support.v4.view.ViewPager;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.microsoft.windowsazure.mobileservices.MobileServiceClient;
import
com.microsoft.windowsazure.mobileservices.http.ServiceFilterResponse;
import
com.microsoft.windowsazure.mobileservices.table.TableOperationCallback;
import com.microsoft.windowsazure.mobileservices.table.TableQueryCallback;
import com.microsoft.windowsazure.mobileservices.table.query.QueryOrder;
import com.tarambana.markit.Adapters.SectionsPagerAdapter;
import com.tarambana.markit.DataContainers.MarkSchemePart;
import com.tarambana.markit.DataContainers.MarkSchemeSection;
import com.tarambana.markit.DataContainers.Student;
import com.tarambana.markit.DataContainers.StudentTotalMarks;
import com.tarambana.markit.DataContainers.localAssignment;
import com.tarambana.markit.Fragments.MarkStudentFragment;

import java.net.MalformedURLException;
import java.util.HashMap;
import java.util.Map;

public class MarkStudent extends AppCompatActivity implements
MarkStudentFragment.MarksToActivityPasser,
NavigationView.OnNavigationItemSelectedListener {

    static String TAG = "TASA_LOG ";

    int failCountAzureConnection = 0;

    // Object declared to interact with Azure cloud
    private MobileServiceClient mClientAzureConnection;

    public localAssignment currentActiveAssignment = new localAssignment();

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mark_student);
```

```
        try {
            Log.d(TAG, "MarkStudent: attempting to connect to Azure site");

            mClientAzureConnection = new MobileServiceClient(
                    "https://bookchoice.azurewebsites.net",
                    this
            );
            Log.d(TAG, "MarkStudent: success in connecting to Azure");

        } catch (MalformedURLException e) {
            Log.d(TAG, "MarkStudent: malformed URL in connection to Azure
site");
            Toast.makeText(getApplicationContext(), "Internet connection
issue, reconnect and try again", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }

        // Set up toolbar, drawer and navigation bar for UI
        Toolbar toolbarNavBar = (Toolbar) findViewById(R.id.toolbarMark);
        setSupportActionBar(toolbarNavBar);

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_mark_student);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbarNavBar,
R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.getMenu().getItem(0).setChecked(true);

        // Retrieve information sent by MainActivity in bundle
        Bundle receivedInfoFromMainActivity = getIntent().getExtras();
        String unitSelected =
receivedInfoFromMainActivity.getString("unit");
        int assignmentSelected =
receivedInfoFromMainActivity.getInt("assignment");
        int groupSelected = receivedInfoFromMainActivity.getInt("group");
        int studentIDSelected =
receivedInfoFromMainActivity.getInt("studentID");
        Log.d(TAG, "MarkStudent: bundle retrieved correctly in 2nd
activity");

        // Set toolbar title
        toolbarNavBar.setTitle(unitSelected + " | Assignment " +
assignmentSelected + " | Group " + groupSelected);
        Log.d(TAG, "MarkStudent: toolbar string set");

        // Set student ID
        TextView studentIDTV = (TextView) findViewById(R.id.StudentIDTV);
        studentIDTV.setText("ID: " + studentIDSelected);

        currentActiveAssignment.setStudentID(studentIDSelected);
        currentActiveAssignment.setAssignmentNumber(assignmentSelected);

        // Download section information from cloud
        getSectionInfoFromCloud("SECTIONASSIGNMENTID", assignmentSelected,
```

```
studentIDSelected);

        // Start marks tag with 0 marks
        UpdateTotalMarksTag(currentActiveAssignment);
    }

    // This method downloads a "MarkSchemeSection" class with the
information required from the Azure SQL database, and allows that class to
be inspected and manipulated for use of the data it contains. The other
methods below with similar names work the same but download different types
of data, and thus different classes
    void getSectionInfoFromCloud(final String field, final int condition,
final int studentID) {

        Log.d(TAG, "MarkStudent: downloading section information from
Azure");

        // This line is Azure's adaptor's way of executing an SQL query:
the MarkSchemeSection class downloaded contains the information where the
passed field meets the passed condition
        mClientAzureConnection.getTable(MarkSchemeSection.class)
                .where().field(field).eq(condition)
                .orderBy("SECTIONID", QueryOrder.Ascending)
                .execute(new TableQueryCallback<MarkSchemeSection>() {

            @Override
            public void onCompleted(java.util.List<MarkSchemeSection>
resultSection, int count, Exception exception, ServiceFilterResponse
response) {

                // If there's no exception and all goes well, set up the
spinner from the result of the transaction
                if (exception == null) {

                    Log.d(TAG, "MarkStudent: section data successfully
retrieved from Azure");

                    for (MarkSchemeSection sectionDownloaded :
resultSection) {

currentActiveAssignment.setSectionIDSectionName(sectionDownloaded.getSectio
nID(), sectionDownloaded.getSectionName());
                    }

                    getPartInfoFromCloud("PARTASSIGNMENTID", condition,
studentID);

                } else if (failCountAzureConnection < 3) {
                    Log.d(TAG, "MarkStudent: Exception found: " +
exception.getMessage());
                    Log.d(TAG, "MarkStudent: reattempting section data
download");
                    getSectionInfoFromCloud(field, condition, studentID);
                    failCountAzureConnection++;
                }
                else {
                    Log.d(TAG, "MarkStudent: 4th time an exception has been
found: " + exception.getMessage());
                }
            }
        });
```

```
    }

    void getPartInfoFromCloud(final String field, final int condition,
final int studentID) {

        Log.d(TAG, "MarkStudent: downloading part information from Azure");

        // This query must download
        mClientAzureConnection.getTable(MarkSchemePart.class)
                .where().field(field).eq(condition)
                .orderBy("PARTASSIGNMENTID", QueryOrder.Ascending)
                .execute(new TableQueryCallback<MarkSchemePart>() {

                    @Override
                    public void onCompleted(java.util.List<MarkSchemePart>
resultPart, int count, Exception exception, ServiceFilterResponse response)
{

                        if (exception == null) {

                            Log.d(TAG, "MarkStudent: part data successfully
retrieved from Azure");

                            for (MarkSchemePart partDownloaded :
resultPart) {

currentActiveAssignment.setPartIDPartName(partDownloaded.getPartID(),
partDownloaded.getPartName());

currentActiveAssignment.setPartNamePartMark(partDownloaded.getPartName(),
partDownloaded.getPartAvailableMarks());

currentActiveAssignment.setPartIDSectionID(partDownloaded.getPartID(),
partDownloaded.getPartSectionID());

currentActiveAssignment.setPartIDPartCorrect(partDownloaded.getPartID(),
false);
                            }

                            getStudentInfoFromCloud("STUDENTID",
studentID);

                        } else {
                            Log.d(TAG, "MarkStudent: exception found: " +
exception.getMessage());
                        }
                    }
                });
    }

    void getStudentInfoFromCloud(String field, int condition) {

        Log.d(TAG, "MarkStudent: downloading student information from
Azure");


mClientAzureConnection.getTable(Student.class).where().field(field).eq(cond
ition)
                .orderBy("STUDENTID", QueryOrder.Ascending).execute(new
TableQueryCallback<Student>() {
```

```
            @Override
            public void onCompleted(java.util.List<Student>
resultStudentInfo, int count, Exception exception, ServiceFilterResponse
response) {

                if (exception == null) {

                    Log.d(TAG, "MarkStudent: student data successfully
retrieved from Azure");

                    for (Student student : resultStudentInfo) {
                        // Set up all of the student information in the
local assignment

currentActiveAssignment.setStudentFirstName(student.getStudentFirstName());

currentActiveAssignment.setStudentLastName(student.getStudentLastName());

currentActiveAssignment.setStudentID(student.getStudentID());
                        currentActiveAssignment.setStudentMarks(0);
                    }

                    // Once all data has been downloaded, it can be passed
to and used by the fragments that will populate the tab layout
                    setUpTabLayoutFragments();
                    setUpSaveButton();

                } else {
                    Log.d(TAG, "MarkStudent: exception found: " +
exception.getMessage());
                }
            }
        });
    }

    // This method adds or updates elements in the SQL database with the
information passed to it via a class
    void addItemsToTable(localAssignment assignmentToSave){

        Log.d(TAG, "MarkStudent: adding item to SQL database");

        final StudentTotalMarks studentMarks = new StudentTotalMarks();

studentMarks.setStudentTotalMarksStudentID(assignmentToSave.studentID);

studentMarks.setStudentTotalMarksAssignmentID(assignmentToSave.assignmentID
);

studentMarks.setStudentTotalMarksAchieved(assignmentToSave.getStudentMarks(
));


mClientAzureConnection.getTable(StudentTotalMarks.class).insert(studentMark
s, new TableOperationCallback<StudentTotalMarks>() {

            @Override
            public void onCompleted(StudentTotalMarks entity, Exception
exception, ServiceFilterResponse response) {

                // Always check for exceptions/completion
                if (exception == null){
```

```
                    Log.d(TAG, "MarkStudent: SQL insert successful");
                    studentMarks.totalMarksAchieved = 0;
                }

                else{
                    exception.printStackTrace();
                }
            }
        });
    }

    // This method implements an interface: a construct used to pass data
between fragments and their containing activity
    @Override
    public void sendDataToActivity(HashMap<Integer, Boolean>
partIDPartCorrect){
        for (Map.Entry<Integer, Boolean> entry :
partIDPartCorrect.entrySet()){
            if
(currentActiveAssignment.partIDPartCorrect.containsKey(entry.getKey())){

currentActiveAssignment.partIDPartCorrect.put(entry.getKey(),
entry.getValue());
            }
        }

        UpdateTotalMarksTag(currentActiveAssignment);
    }

    // This method implements all the necessary logic to launch a tablayout
of various fragments within this activity
    private void setUpTabLayoutFragments() {

        Log.d(TAG, "MarkStudent: setting up tab layout, fragments");

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        ViewPager viewPager = (ViewPager) findViewById(R.id.container);
        TabLayout tabLayout = (TabLayout) findViewById(R.id.sliding_tabs);

        setSupportActionBar(toolbar);

        viewPager.setOffscreenPageLimit(5);

        // More information on the adapter can be found in its class
        SectionsPagerAdapter adapter = new
SectionsPagerAdapter(getSupportFragmentManager(), currentActiveAssignment);
        Log.d(TAG, "MarkStudent: tablayout adapter built");

        viewPager.setAdapter(adapter);
        Log.d(TAG, "MarkStudent: adapter set");

        tabLayout.setupWithViewPager(viewPager);
        Log.d(TAG, "MarkStudent: view pager set and complete, tablayout is
functional");

    }

    // This method sets up the save button in order to allow for data
saving
    private void setUpSaveButton(){
```

```
        Button saveButton = (Button) findViewById(R.id.SaveBtn);
        saveButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                if(saveStudentMarksInCloud(currentActiveAssignment)){
                    Toast.makeText(getBaseContext(), "Student marks updated
on cloud", Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(getBaseContext(), "Student marks not
saved", Toast.LENGTH_SHORT).show();
                }
            }
        });

    }

    // This method implements the logic to prepare the data to be saved in
the cloud and call the Azure adapter function to save it
    public boolean saveStudentMarksInCloud(localAssignment
assignmentToSave){
        Log.d(TAG, "MarkStudent: saving mark data");
        for (Map.Entry<String, Integer> entry :
assignmentToSave.partNamePartMark.entrySet()){
            if
(assignmentToSave.partIDPartCorrect.get(getKeyForValueInHashMap(assignmentT
oSave.partIDPartName, entry.getKey()))){
                assignmentToSave.studentMarks +=
assignmentToSave.partNamePartMark.get(entry.getKey());
            }
        }
        addItemsToTable(assignmentToSave);

        assignmentToSave.studentMarks = 0;

        return true;
    }

    // This method updates the total marks counter on the UI
    private void UpdateTotalMarksTag(localAssignment assignment){
        int marksCounter = 0;
        for (Map.Entry<String, Integer> entry :
assignment.partNamePartMark.entrySet()){
            if
(assignment.partIDPartCorrect.get(getKeyForValueInHashMap(assignment.partID
PartName, entry.getKey()))){
                marksCounter +=
assignment.partNamePartMark.get(entry.getKey());
            }
        }

        TextView totalMarksCounter = (TextView)
findViewById(R.id.TotalMarksTV);
        totalMarksCounter.setText("Total Marks: " + marksCounter);
    }

    Integer getKeyForValueInHashMap(Map<Integer, String> inputHashMap,
String value) {

        Integer keyToReturn = 0;
```

```
        if(inputHashMap.containsValue(value))
        {
            for (Map.Entry<Integer, String> entry :
inputHashMap.entrySet())
            {
                if (entry.getValue().equals(value))
                {
                    keyToReturn = entry.getKey();
                }
            }
        }
        return keyToReturn;
    }

    // System required method
    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_mark_student);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    // System required method
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    // System required method
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // This method determines what happens when something is selected in
the navigation drawer
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {

        int id = item.getItemId();

        if (id == R.id.student_selection) {
            Log.d(TAG, "MarkStudent: student selection activity selected in
```

```
navigation drawer");
            item.setChecked(true);
            Intent myIntent = new Intent(this, MainActivity.class);
            this.startActivity(myIntent);
        } else if (id == R.id.make_mark_scheme) {
            Log.d(TAG, "MarkStudent: make mark scheme activity selected in
navigation drawer");
            item.setChecked(true);
        }
        else if (id == R.id.check_marks) {
            Log.d(TAG, "MarkStudent: grade consult activity selected in
navigation drawer");
            item.setChecked(true);
            Intent myIntent = new Intent(this, GradeConsult.class);
            this.startActivity(myIntent);
        }

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_mark_student);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

Mark Student Fragment Java Code

```
package com.tarambana.markit.Fragments;

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

import com.tarambana.markit.DataContainers.localSection;
import com.tarambana.markit.R;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MarkStudentFragment extends Fragment {

    public localSection currentActiveSection = new localSection();

    final static String TAG = "TASA_LOG ";

    // Declaration and instantiation of the interface
    private MarksToActivityPasser dataPasserToActivity;

    public interface MarksToActivityPasser {
        public void sendDataToActivity(HashMap<Integer, Boolean>
partIDPartCorrect);
```

```
    }

    public MarkStudentFragment() {
        // Required empty public constructor
    }

    public static MarkStudentFragment newInstance(Bundle bundleReceived) {
        MarkStudentFragment fragment = new MarkStudentFragment();
        // Retrieve the bundle sent in a format compatible with the
fragment concept: as arguments
        fragment.setArguments(bundleReceived);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (getArguments() != null) {
            Log.d(TAG, "MarkStudentFragment " + this.getId() + " : getting
information from arguments");

currentActiveSection.setAssignmentID(getArguments().getInt("assignmentID"))
;

currentActiveSection.setSectionID(getArguments().getInt("sectionID"));

currentActiveSection.setSectionName(getArguments().getString("sectionName")
);

            for (int i = 0; i < getArguments().size() - 4; i++){
                if (getArguments().getInt("partID" + i) != 0){

currentActiveSection.setPartIDPartName(getArguments().getInt("partID" + i),
getArguments().getString("partName" + i));

currentActiveSection.setPartNamePartMark(getArguments().getString("partName
" + i), getArguments().getInt("partMarks" + i));

currentActiveSection.setPartIDPartCorrect(getArguments().getInt("partID" +
i), false);
                }
            }
        }
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof MarksToActivityPasser){
        dataPasserToActivity = (MarksToActivityPasser) context;
            Log.d(TAG, "MarkStudentFragment " + this.getId() + " : data
passer instantiated");
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_mark_student,
container, false);
```

```
        return rootView;
    }

    // This method is called every time the fragment is instantiated in UI
and sets up the UI section of its function: checkboxes are responsive,
etc...
    @Override
    public void onStart() {
        super.onStart();

        Log.d(TAG, "MarkStudentFragment " +
this.getFragmentManager().findFragmentById(R.id.markCB1) + " : fragment
started");

        TextView sectionTitle = (TextView)
getView().findViewById(R.id.sectionTitleTV);

        List<CheckBox> checkBoxesToActivate = new ArrayList<>();

checkBoxesToActivate.add((CheckBox)getView().findViewById(R.id.markCB1));

checkBoxesToActivate.add((CheckBox)getView().findViewById(R.id.markCB2));

checkBoxesToActivate.add((CheckBox)getView().findViewById(R.id.markCB3));

checkBoxesToActivate.add((CheckBox)getView().findViewById(R.id.markCB4));

checkBoxesToActivate.add((CheckBox)getView().findViewById(R.id.markCB5));

        SetUpCheckBoxReactions();

        sectionTitle.setText(currentActiveSection.getSectionName());

        int i = 0;
        for (Map.Entry<String, Integer> entry :
currentActiveSection.partNamePartMark.entrySet()) {
            if (entry.getKey() == null){

            } else{
            checkBoxesToActivate.get(i).setText(entry.getKey());
            checkBoxesToActivate.get(i).setVisibility(View.VISIBLE);
            i++;
            }
        }
    }

    // This method sets up all the checkboxes in the marking UI to respond
to user clicks, etc...
    void SetUpCheckBoxReactions(){

        final CheckBox checkBox1 = (CheckBox)
getView().findViewById(R.id.markCB1);
        final CheckBox checkBox2 = (CheckBox)
getView().findViewById(R.id.markCB2);
        final CheckBox checkBox3 = (CheckBox)
getView().findViewById(R.id.markCB3);
        final CheckBox checkBox4 = (CheckBox)
getView().findViewById(R.id.markCB4);
        final CheckBox checkBox5 = (CheckBox)
getView().findViewById(R.id.markCB5);
```

```
        checkBox1.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked){

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox1.getText().toString()), true);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }

                else {

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox1.getText().toString()), false);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);

                }
            }
        });

        checkBox2.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {

                if (isChecked){

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox2.getText().toString()), true);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }

                else {

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox2.getText().toString()), false);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }
            }
        });

        checkBox3.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked){

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox3.getText().toString()), true);
```

```
dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }

                else {

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox3.getText().toString()), false);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }
            }
        });

        checkBox4.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked){

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox4.getText().toString()), true);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }

                else {

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox4.getText().toString()), false);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }
            }
        });

        checkBox5.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked){

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox5.getText().toString()), true);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
ect);
                }

                else {

currentActiveSection.setPartIDPartCorrect(getKeyForValueInHashMap(currentAc
tiveSection.partIDPartName, checkBox5.getText().toString()), false);

dataPasserToActivity.sendDataToActivity(currentActiveSection.partIDPartCorr
```

```
ect);
                }
            }
        });
    }

    // Utility method to get the key that corresponds to the input value in
a given hashmap
    Integer getKeyForValueInHashMap(Map<Integer, String> inputHashMap,
String value) {

        Integer keyToReturn = 0;

        if(inputHashMap.containsValue(value))
        {
            for (Map.Entry<Integer, String> entry :
inputHashMap.entrySet())
            {
                if (entry.getValue().equals(value))
                {
                    keyToReturn = entry.getKey();
                }
            }
        }
        return keyToReturn;
    }
}
```

Mark Student Adapter Java Code

```
package com.tarambana.markit.Adapters;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import com.tarambana.markit.DataContainers.localAssignment;
import com.tarambana.markit.Fragments.MarkStudentFragment;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

// Note that the maximum amount of sections allowed per assignment is 5
public class SectionsPagerAdapter extends FragmentPagerAdapter {

    public localAssignment currentActiveAssignment;

    public SectionsPagerAdapter(FragmentManager fm, localAssignment
inputAssignment) {
        super(fm);
        this.currentActiveAssignment = inputAssignment;
    }

    // This method populates the tab layout
```

```
    @Override
    public Fragment getItem(int position) {

        MarkStudentFragment fragment = new MarkStudentFragment();

        if (position == 0) {
            fragment =
MarkStudentFragment.newInstance(SplitAssignmentIntoSection(currentActiveAss
ignment, position + 1));
        } else if (position == 1){
            fragment =
MarkStudentFragment.newInstance(SplitAssignmentIntoSection(currentActiveAss
ignment, position + 1));
        } else if (position == 2){
            fragment =
MarkStudentFragment.newInstance(SplitAssignmentIntoSection(currentActiveAss
ignment, position + 1));
        } else if (position == 3) {
            fragment =
MarkStudentFragment.newInstance(SplitAssignmentIntoSection(currentActiveAss
ignment, position + 1));
        } else {
            fragment =
MarkStudentFragment.newInstance(SplitAssignmentIntoSection(currentActiveAss
ignment, position + 1));
        }
        return fragment;
    }

    @Override
    public int getCount() {
        return currentActiveAssignment.sectionIDSectionName.size();
    }

    // This method returns the page title for each page of the tab layout
    @Override
    public CharSequence getPageTitle(int position) {

        switch (position) {
            case 0:
                return (currentActiveAssignment.assignmentID + ".1");
            case 1:
                return (currentActiveAssignment.assignmentID + ".2");
            case 2:
                return (currentActiveAssignment.assignmentID + ".3");
            case 3:
                return (currentActiveAssignment.assignmentID + ".4");
            case 4:
                return (currentActiveAssignment.assignmentID + ".5");
            default:
                return null;
        }
    }

    // This method takes an input assignment class and splits it into the
various sections that are required fro each of the pages (fragments) in the
tab layout
    Bundle SplitAssignmentIntoSection(localAssignment inputAssignment, int
sectionID){
        Bundle bundleToReturn = new Bundle();
```

```
        bundleToReturn.putInt("assignmentID",
inputAssignment.getAssignmentNumber());
        bundleToReturn.putInt("sectionID", sectionID);
        bundleToReturn.putString("sectionName",
inputAssignment.getSectionIDSectionName().get(sectionID));

        for(int i = 1; i < inputAssignment.partIDSectionID.size(); i++){

                List<Integer> partIDList =
getAllKeysForValueInHashMap(inputAssignment.partIDSectionID, sectionID);

                for (int j = 0; j < partIDList.size(); j++){

                    bundleToReturn.putInt("partID" + j, partIDList.get(j));
                    bundleToReturn.putString("partName" + j,
inputAssignment.getPartIDPartName().get(partIDList.get(j)));
                    bundleToReturn.putInt("partMarks" + j,
inputAssignment.getPartNamePartMark().get(inputAssignment.getPartIDPartName
().get(partIDList.get(j))));
                }
        }

        return  bundleToReturn;
    }

    // Utility method to get the key that corresponds to the input value in
a given hashmap
    List<Integer> getAllKeysForValueInHashMap(Map<Integer, Integer>
inputHashMap, int value)
    {
        List<Integer> keyToReturn = new ArrayList<>();

        if(inputHashMap.containsValue(value))
        {
            for (Map.Entry<Integer, Integer> entry :
inputHashMap.entrySet())
            {
                if (entry.getValue().equals(value))
                {
                    keyToReturn.add(entry.getKey());
                }
            }
        }
        return keyToReturn;
    }

}
```

Grade Consult Activity Logic Java Code

```
package com.tarambana.markit;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.microsoft.windowsazure.mobileservices.MobileServiceClient;
import
com.microsoft.windowsazure.mobileservices.http.ServiceFilterResponse;
import com.microsoft.windowsazure.mobileservices.table.TableQueryCallback;
import com.tarambana.markit.DataContainers.StudentTotalMarks;

import java.net.MalformedURLException;
import java.util.ArrayList;
import java.util.List;

import static android.view.View.VISIBLE;

public class GradeConsult extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

   final static String TAG = "TASA_LOG ";

    // Object declared to interact with Azure cloud
   private MobileServiceClient mClientAzureConnection;

   boolean firstTimeStudentDropdownRefreshed = false;

   int failCountAzureConnection = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grade_consult);

        try {
            Log.d(TAG, "GradeConsult: attempting to connect to Azure
site");

            mClientAzureConnection = new MobileServiceClient(
                    "https://bookchoice.azurewebsites.net",
                    this
```

```
            );

            Log.d(TAG, "GradeConsult: success in connecting to Azure");

        } catch (MalformedURLException e){
            Log.d(TAG, "Malformed URL in connection to Azure site");
            Toast.makeText(getApplicationContext(), "Internet connection
issue, reconnect and try again", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }

        // Set up toolbar, drawer and navigation bar for UI
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarGC);
        setSupportActionBar(toolbar);

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.activity_grade_consult);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.getMenu().getItem(0).setChecked(true);

        final Spinner assignmentSpinner = (Spinner)
findViewById(R.id.GCSelectAssignmentSp);
        final Spinner studentSpinner = (Spinner)
findViewById(R.id.GCSelectStudentSP);
        final Button checkMarksBtn = (Button)
findViewById(R.id.ConsultGradeBtn);

        checkMarksBtn.setEnabled(true);
        checkMarksBtn.setVisibility(VISIBLE);

        checkMarksBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                GetStudentGrade("TOTALSTUDENTID", "TOTALASSIGNMENTID",
studentSpinner.getSelectedItem().toString(),
assignmentSpinner.getSelectedItem().toString());
            }
        });

        studentSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                if (firstTimeStudentDropdownRefreshed){
                String studentID =
studentSpinner.getSelectedItem().toString();
                refreshAssignmentDropDown("TOTALSTUDENTID", studentID);
                firstTimeStudentDropdownRefreshed = false;

                }

                else firstTimeStudentDropdownRefreshed = true;
```

```
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        refreshStudentDropDown("TOTALSTUDENTID", "*");

    }

    // This method downloads a "Student" class with the information
required from the Azure SQL database, and allows that class to be inspected
and manipulated for use of the data it contains. The other methods below
with similar names work the same but download different types of data, and
thus different classes
    void refreshStudentDropDown(String field, String condition){

        Log.d(TAG, "GradeConsult: downloading student data");

        try{

mClientAzureConnection.getTable(StudentTotalMarks.class).execute(new
TableQueryCallback<StudentTotalMarks>() {

                    // Listener that automatically gets set for the result
of the transaction with Azure
                    @Override
                    public void
onCompleted(java.util.List<StudentTotalMarks> result, int count, Exception
exception, ServiceFilterResponse response) {
                        if (exception == null) {

                            List<Integer> studentSpinnerList = new
ArrayList<>();
                            studentSpinnerList.add(000000000);

                            Log.d(TAG, "GradeConsult: student data download
successful");

                            for (StudentTotalMarks student : result) {
                                if
(!(studentSpinnerList.contains(student.getStudentTotalMarksStudentID()))) {

studentSpinnerList.add(student.getStudentTotalMarksStudentID());
                                }
                            }

                            ArrayAdapter<Integer> adapter = new
ArrayAdapter<>(
                                    getBaseContext(),
android.R.layout.simple_spinner_item, studentSpinnerList);


adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                            Spinner studentSpinner = (Spinner)
findViewById(R.id.GCSelectStudentSP);
                            studentSpinner.setAdapter(adapter);
                            failCountAzureConnection = 0;
```

```
                        } else if (failCountAzureConnection < 3) {
                            Log.d(TAG, "GradeConsult: Exception found: " +
exception.getMessage());
                            Log.d(TAG, "GradeConsult: reattempting student
data download");
                            refreshStudentDropDown("TOTALSTUDENTID", "*");
                            failCountAzureConnection++;
                        } else {
                            Log.d(TAG, "GradeConsult: exception found: " +
exception.getMessage());
                            Toast.makeText(getApplicationContext(),
"Internet connection issue, reconnect and try again",
Toast.LENGTH_SHORT).show();

                        }
                    }
                });
        }
                catch (Exception e){
                    e.printStackTrace();
                }
    }

    void refreshAssignmentDropDown(String field, String condition){

        Log.d(TAG, "GradeConsult: downloading assignment data");
        // This is how a basic query is executed, in this case all IDs

mClientAzureConnection.getTable(StudentTotalMarks.class).where().field(fiel
d).eq(condition).execute(new TableQueryCallback<StudentTotalMarks>() {

            // Listener that automatically gets set for the result of the
transaction with Azure
            @Override
            public void onCompleted(java.util.List<StudentTotalMarks>
result, int count, Exception exception, ServiceFilterResponse response) {

                if (exception == null) {

                    List<Integer> assignmentSpinnerList = new
ArrayList<>();
                    assignmentSpinnerList.add(000000000);

                    Log.d(TAG, "GradeConsult: assignment data download
successful");

                    for (StudentTotalMarks unit : result) {
                        if
(!(assignmentSpinnerList.contains(unit.getStudentTotalMarksAssignmentID()))
) {

assignmentSpinnerList.add(unit.getStudentTotalMarksAssignmentID());
                        }
                    }

                    ArrayAdapter<Integer> adapter = new ArrayAdapter<>(
                            getBaseContext(),
android.R.layout.simple_spinner_item, assignmentSpinnerList);
```

```
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
                    Spinner assignmentSpinner = (Spinner)
findViewById(R.id.GCSelectAssignmentSp);
                    assignmentSpinner.setAdapter(adapter);
                }

                else {
                    Log.d(TAG, "GradeConsult: exception found: " +
exception.getMessage());
                }
            }
        });
    }

    void GetStudentGrade(String field1, String field2, final String
condition, final String condition2){

        Log.d(TAG, "GradeConsult: downloading grade data");
        // This is how a basic query is executed, in this case all IDs

mClientAzureConnection.getTable(StudentTotalMarks.class).where().field(fiel
d1).eq(condition).and().field(field2).eq(condition2).execute(new
TableQueryCallback<StudentTotalMarks>() {

            // Listener that automatically gets set for the result of the
transaction with Azure
            @Override
            public void onCompleted(java.util.List<StudentTotalMarks>
result, int count, Exception exception, ServiceFilterResponse response) {

                if (exception == null) {

                    List<String> outputInfo = new ArrayList<>();

                    Log.d(TAG, "GradeConsult: grade data download
successful");

                    for (StudentTotalMarks grade : result) {
                        outputInfo.add("Student " + condition + " achieved
" + grade.getStudentTotalMarksAchieved() + " marks in assignment " +
condition2);
                        }

                        String text = "";
                        TextView gradeDisplayTV = (TextView)
findViewById(R.id.StudentGradeInfo);

                        for (String line : outputInfo){
                        text = text + line + "\n";
                        gradeDisplayTV.setText(text);
                        }
                }
                else {
                    Log.d(TAG, "GradeConsult: exception found: " +
exception.getMessage());
                }
            }
        });
    }
```

```
    // System required method
    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.activity_grade_consult);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }
    // System required method
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    // System required method
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // This method determines what happens when something is selected in
the navigation drawer
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();
        if (id == R.id.student_selection) {
            Log.d(TAG, "GradeConsult: student selection activity selected
in navigation drawer");
            item.setChecked(true);
            Intent myIntent = new Intent(this, MainActivity.class);
            this.startActivity(myIntent);
        } else if (id == R.id.make_mark_scheme) {
            Log.d(TAG, "GradeConsult: make mark scheme activity selected in
navigation drawer");
            item.setChecked(true);
        }
        else if (id == R.id.check_marks) {
            Log.d(TAG, "GradeConsult: grade consult activity selected in
navigation drawer");
            item.setChecked(true);
        }
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.activity_grade_consult);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }}
```

Mark It Android Data Containers

MarkScheme class

```
package com.tarambana.markit.DataContainers;

public class MarkScheme {

    @com.google.gson.annotations.SerializedName("id")
    private String mID;

    @com.google.gson.annotations.SerializedName("MarkSchemeAssignmentID")
    public int markSchemeAssignmentID;

    @com.google.gson.annotations.SerializedName("MarkSchemeAssignmentName")
    public String markSchemeAssignmentName;


@com.google.gson.annotations.SerializedName("MarkSchemeAssignmentAvailableM
arks")
    public int markSchemeAssignmentAvailableMarks;

    @com.google.gson.annotations.SerializedName("MarkSchemeAuthor")
    public String markSchemeAuthor;
}
```

MarkSchemeSection class

```
package com.tarambana.markit.DataContainers;

public class MarkSchemeSection {

    @com.google.gson.annotations.SerializedName("id")
    private String mID;

    @com.google.gson.annotations.SerializedName("SectionAssignmentID")
    public int sectionAssignmentID;

    @com.google.gson.annotations.SerializedName("SectionID")
    public int sectionID;

    @com.google.gson.annotations.SerializedName("SectionName")
    public String sectionName;

    @com.google.gson.annotations.SerializedName("SectionAvailableMarks")
    public int sectionAvailableMarks;

    @com.google.gson.annotations.SerializedName("SectionAuthor")
    public String sectionAuthor;

    public int getSectionID() {
        return sectionID;
    }

    public String getSectionName() {
        return sectionName;
    }
}
```

MarkSchemePart class

```
package com.tarambana.markit.DataContainers;

public class MarkSchemePart {

    @com.google.gson.annotations.SerializedName("id")
    private String mID;

    @com.google.gson.annotations.SerializedName("PartAssignmentID")
    public int partAssignmentID;

    @com.google.gson.annotations.SerializedName("PartSectionID")
    public int partSectionID;

    @com.google.gson.annotations.SerializedName("PartPartID")
    public int partID;

    public int getPartID() {
        return partID;
    }

    public void setPartID(int partID) {
        this.partID = partID;
    }

    @com.google.gson.annotations.SerializedName("PartName")
    public String partName;

    @com.google.gson.annotations.SerializedName("PartAvailableMarks")
    public int partAvailableMarks;

    @com.google.gson.annotations.SerializedName("PartAuthor")
    public String partAuthor;

    public int getPartSectionID() {
        return partSectionID;
    }

    public String getPartName() {
        return partName;
    }

    public int getPartAvailableMarks() {
        return partAvailableMarks;
    }
}
```

LabGroup class

```
package com.tarambana.markit.DataContainers;

import java.util.Date;

public class LabGroup {

    @com.google.gson.annotations.SerializedName("id")
```

```
    private String mID;

    @com.google.gson.annotations.SerializedName("LabGroupStudentID")
    public int labGroupStudentID;

    @com.google.gson.annotations.SerializedName("LabGroupNumber")
    public int labGroupNumber;

    @com.google.gson.annotations.SerializedName("LabGroupLocation")
    public String labGroupLocation;

    @com.google.gson.annotations.SerializedName("LabGroupAssignmentID")
    public int labGroupAssignmentID;

    @com.google.gson.annotations.SerializedName("LabGroupUnit")
    public String labGroupUnit;

    @com.google.gson.annotations.SerializedName("LabGroupDate")
    public Date labGroupDate;

    public int getLabGroupStudentID() {
        return labGroupStudentID;
    }

    public int getLabGroupNumber() {
        return labGroupNumber;
    }

    public int getLabGroupAssignmentID() {
        return labGroupAssignmentID;
    }

    public String getLabGroupUnit() {
        return labGroupUnit;
    }
}
```

Student class

```
package com.tarambana.markit.DataContainers;

public class Student {

    @com.google.gson.annotations.SerializedName("id")
    private String mID;

    @com.google.gson.annotations.SerializedName("StudentLastName")
    public String studentLastName;

    @com.google.gson.annotations.SerializedName("StudentFirstName")
    public String studentFirstName;

    @com.google.gson.annotations.SerializedName("StudentID")
    public int studentID;

    public String getStudentLastName() {
        return studentLastName;
    }
```

```
    public String getStudentFirstName() {
        return studentFirstName;
    }

    public int getStudentID() {
        return studentID;
    }

}
```

StudentSectionMarks class

```
package com.tarambana.markit.DataContainers;

public class StudentSectionMarks {

    @com.google.gson.annotations.SerializedName("id")
    private String mID;

    @com.google.gson.annotations.SerializedName("SectionStudentID")
    public int sectionStudentID;

    @com.google.gson.annotations.SerializedName("SectionAssignmentID")
    public int sectionAssignmentID;

    @com.google.gson.annotations.SerializedName("SectionPartID")
    public int sectionPartID;

    @com.google.gson.annotations.SerializedName("SectionID")
    public int sectionSectionID;

    @com.google.gson.annotations.SerializedName("SectionStudentMarks")
    public int sectionStudentMarks;

    @com.google.gson.annotations.SerializedName("SectionAuthor")
    public String sectionAuthor;

    public String getSectionAuthor() {
        return sectionAuthor;
    }

    public void setSectionAuthor(String sectionAuthor) {
        this.sectionAuthor = sectionAuthor;
    }

    public int getSectionStudentMarks() {
        return sectionStudentMarks;
    }

    public void setSectionStudentMarks(int sectionStudentMarks) {
        this.sectionStudentMarks = sectionStudentMarks;
    }

    public int getSectionSectionID() {
        return sectionSectionID;
    }

    public void setSectionSectionID(int sectionSectionID) {
        this.sectionSectionID = sectionSectionID;
```

```
    }

    public int getSectionPartID() {
        return sectionPartID;
    }

    public void setSectionPartID(int sectionPartID) {
        this.sectionPartID = sectionPartID;
    }

    public int getSectionAssignmentID() {
        return sectionAssignmentID;
    }

    public void setSectionAssignmentID(int sectionAssignmentID) {
        this.sectionAssignmentID = sectionAssignmentID;
    }

    public int getSectionStudentID() {
        return sectionStudentID;
    }

    public void setSectionStudentID(int sectionStudentID) {
        this.sectionStudentID = sectionStudentID;
    }
}
```

StudentTotalMarks class

```
package com.tarambana.markit.DataContainers;

public class StudentTotalMarks {

    @com.google.gson.annotations.SerializedName("id")
    private String mID;

    @com.google.gson.annotations.SerializedName("TotalStudentID")
    public int totalMarksStudentID;

    @com.google.gson.annotations.SerializedName("TotalAssignmentID")
    public int totalMarksAssignmentID;

    @com.google.gson.annotations.SerializedName("TotalMarksAchieved")
    public int totalMarksAchieved;

    public int getStudentTotalMarksStudentID() {
        return totalMarksStudentID;
    }

    public void setStudentTotalMarksStudentID(int
studentTotalMarksStudentID) {
        this.totalMarksStudentID = studentTotalMarksStudentID;
    }

    public int getStudentTotalMarksAssignmentID() {
        return totalMarksAssignmentID;
    }

    public void setStudentTotalMarksAssignmentID(int
```

```
studentTotalMarksAssignmentID) {
        this.totalMarksAssignmentID = studentTotalMarksAssignmentID;
    }

    public int getStudentTotalMarksAchieved() {
        return totalMarksAchieved;
    }

    public void setStudentTotalMarksAchieved(int studentTotalMarksAchieved)
{
        totalMarksAchieved = studentTotalMarksAchieved;
    }
}
```

localAssignment class

```
package com.tarambana.markit.DataContainers;

import java.util.HashMap;

public class localAssignment {

    public int assignmentID;

    // Section
    public HashMap<Integer, String> sectionIDSectionName = new HashMap<>();

    // Part
    public HashMap<Integer, String> partIDPartName = new HashMap<>();
    public HashMap<String, Integer> partNamePartMark = new HashMap<>();
    public HashMap<Integer, Integer> partIDSectionID = new HashMap<>();
    public HashMap<Integer, Boolean> partIDPartCorrect = new HashMap<>();

    // Student
    public int studentID;
    public String studentFirstName;
    public String studentLastName;
    public int studentMarks;

    public localAssignment(){

    }

    public int getAssignmentNumber() {
        return assignmentID;
    }

    public void setAssignmentNumber(int assignmentNumber) {
        this.assignmentID = assignmentNumber;
    }

    public HashMap<Integer, String> getSectionIDSectionName() {
        return sectionIDSectionName;
    }

    public void setSectionIDSectionName(Integer sectionID, String
sectionName) {
        this.sectionIDSectionName.put(sectionID, sectionName);
    }
```

```
    public HashMap<Integer, String> getPartIDPartName() {
        return partIDPartName;
    }

    public void setPartIDPartName(Integer partID, String partName) {
        this.partIDPartName.put(partID, partName);
    }

    public HashMap<String, Integer> getPartNamePartMark() {
        return partNamePartMark;
    }

    public void setPartNamePartMark(String partName, Integer partMark) {
        this.partNamePartMark.put(partName, partMark);
    }

    public void setPartIDSectionID(Integer partID, Integer sectionID) {
        this.partIDSectionID.put(partID, sectionID);
    }

    public void setStudentID(int studentID) {
        this.studentID = studentID;
    }

    public void setStudentFirstName(String studentFirstName) {
        this.studentFirstName = studentFirstName;
    }

    public void setStudentLastName(String studentLastName) {
        this.studentLastName = studentLastName;
    }

    public int getStudentMarks() {
        return studentMarks;
    }

    public void setStudentMarks(int studentMarks) {
        this.studentMarks = studentMarks;
    }

    public void setPartIDPartCorrect(Integer partID, Boolean correct) {
        this.partIDPartCorrect.put(partID, correct);
    }
}
```

localSection class

```
package com.tarambana.markit.DataContainers;

import java.util.HashMap;

public class localSection {

    public int assignmentID;
    public int sectionID;
    public String sectionName;
    public HashMap<Integer, String> partIDPartName = new HashMap<>();
    public HashMap<Integer, Boolean> partIDPartCorrect = new HashMap<>();
```

```
    public HashMap<String, Integer> partNamePartMark = new HashMap<>();

    public localSection(){

    }

    public void setAssignmentID(int assignmentID) {
        this.assignmentID = assignmentID;
    }

    public void setSectionID(int sectionID) {
        this.sectionID = sectionID;
    }

    public String getSectionName() {
        return sectionName;
    }

    public void setSectionName(String sectionName) {
        this.sectionName = sectionName;
    }

    public void setPartIDPartName(Integer partID, String partName) {
        this.partIDPartName.put(partID, partName);
    }

    public void setPartNamePartMark(String partName, Integer partMark) {
        this.partNamePartMark.put(partName, partMark);
    }

    public void setPartIDPartCorrect(Integer partID, Boolean correct) {
        this.partIDPartCorrect.put(partID, correct);
    }
}
```

Annex P

MarkIt Windows Code

Mark It Windows Code

Main form UI Design

```
namespace MarkItDesktop
{
    partial class pathBtn
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.selectPathBtn = new System.Windows.Forms.Button();
            this.assignmentLbl = new System.Windows.Forms.Label();
            this.assignmentCB = new System.Windows.Forms.ComboBox();
            this.downloadLbl = new System.Windows.Forms.Label();
            this.downloadBtn = new System.Windows.Forms.Button();
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.XMLocationLbl = new System.Windows.Forms.Label();
            this.uploadBtn = new System.Windows.Forms.Button();
            this.findXMLBtn = new System.Windows.Forms.Button();
            this.tabControl1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.SuspendLayout();
            //
            // selectPathBtn
            //
            this.selectPathBtn.Location = new System.Drawing.Point(6, 136);
            this.selectPathBtn.Name = "selectPathBtn";
            this.selectPathBtn.Size = new System.Drawing.Size(99, 33);
            this.selectPathBtn.TabIndex = 0;
```

```
            this.selectPathBtn.Text = "Select Path";
            this.selectPathBtn.UseVisualStyleBackColor = true;
            this.selectPathBtn.Click += new
System.EventHandler(this.selectPathBtn_Click);
            //
            // assignmentLbl
            //
            this.assignmentLbl.AutoSize = true;
            this.assignmentLbl.Location = new System.Drawing.Point(2, 12);
            this.assignmentLbl.Name = "assignmentLbl";
            this.assignmentLbl.Size = new System.Drawing.Size(93, 20);
            this.assignmentLbl.TabIndex = 1;
            this.assignmentLbl.Text = "Assignment";
            //
            // assignmentCB
            //
            this.assignmentCB.FormattingEnabled = true;
            this.assignmentCB.Location = new System.Drawing.Point(6, 35);
            this.assignmentCB.Name = "assignmentCB";
            this.assignmentCB.Size = new System.Drawing.Size(226, 28);
            this.assignmentCB.TabIndex = 2;
            //
            // downloadLbl
            //
            this.downloadLbl.AutoSize = true;
            this.downloadLbl.Location = new System.Drawing.Point(2, 81);
            this.downloadLbl.Name = "downloadLbl";
            this.downloadLbl.Size = new System.Drawing.Size(147, 20);
            this.downloadLbl.TabIndex = 3;
            this.downloadLbl.Text = "Download location: ";
            //
            // downloadBtn
            //
            this.downloadBtn.Location = new System.Drawing.Point(123, 136);
            this.downloadBtn.Name = "downloadBtn";
            this.downloadBtn.Size = new System.Drawing.Size(99, 33);
            this.downloadBtn.TabIndex = 4;
            this.downloadBtn.Text = "Download";
            this.downloadBtn.UseVisualStyleBackColor = true;
            this.downloadBtn.Click += new
System.EventHandler(this.downloadBtn_Click_1);
            //
            // tabControl1
            //
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.Location = new System.Drawing.Point(12, 12);
            this.tabControl1.Name = "tabControl1";
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(354, 230);
            this.tabControl1.TabIndex = 5;
            //
            // tabPage1
            //
            this.tabPage1.Controls.Add(this.downloadLbl);
            this.tabPage1.Controls.Add(this.downloadBtn);
            this.tabPage1.Controls.Add(this.selectPathBtn);
            this.tabPage1.Controls.Add(this.assignmentLbl);
            this.tabPage1.Controls.Add(this.assignmentCB);
            this.tabPage1.Location = new System.Drawing.Point(4, 29);
            this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
```

```
this.tabPage1.Size = new System.Drawing.Size(311, 180);
this.tabPage1.TabIndex = 0;
this.tabPage1.Text = "Download Grades";
this.tabPage1.UseVisualStyleBackColor = true;
//
// tabPage2
//
this.tabPage2.Controls.Add(this.XMLocationLbl);
this.tabPage2.Controls.Add(this.uploadBtn);
this.tabPage2.Controls.Add(this.findXMLBtn);
this.tabPage2.Location = new System.Drawing.Point(4, 29);
this.tabPage2.Name = "tabPage2";
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(346, 197);
this.tabPage2.TabIndex = 1;
this.tabPage2.Text = "Upload Mark Scheme";
this.tabPage2.UseVisualStyleBackColor = true;
//
// XMLocationLbl
//
this.XMLocationLbl.AutoSize = true;
this.XMLocationLbl.Location = new System.Drawing.Point(6, 16);
this.XMLocationLbl.Name = "XMLocationLbl";
this.XMLocationLbl.Size = new System.Drawing.Size(105, 20);
this.XMLocationLbl.TabIndex = 6;
this.XMLocationLbl.Text = "XML location:";
//
// uploadBtn
//
this.uploadBtn.Location = new System.Drawing.Point(115, 71);
this.uploadBtn.Name = "uploadBtn";
this.uploadBtn.Size = new System.Drawing.Size(99, 33);
this.uploadBtn.TabIndex = 7;
this.uploadBtn.Text = "Upload";
this.uploadBtn.UseVisualStyleBackColor = true;
this.uploadBtn.Click += new System.EventHandler(this.uploadBtn_Click);
//
// findXMLBtn
//
this.findXMLBtn.Location = new System.Drawing.Point(10, 71);
this.findXMLBtn.Name = "findXMLBtn";
this.findXMLBtn.Size = new System.Drawing.Size(99, 33);
this.findXMLBtn.TabIndex = 5;
this.findXMLBtn.Text = "Find XML";
this.findXMLBtn.UseVisualStyleBackColor = true;
this.findXMLBtn.Click += new System.EventHandler(this.findXMLBtn_Click);
//
// pathBtn
//
this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(378, 254);
this.Controls.Add(this.tabControl1);
this.Name = "pathBtn";
this.Text = "Mark It Desktop";
this.tabControl1.ResumeLayout(false);
this.tabPage1.ResumeLayout(false);
this.tabPage1.PerformLayout();
this.tabPage2.ResumeLayout(false);
this.tabPage2.PerformLayout();
this.ResumeLayout(false);
```

```
        }

        #endregion

        private System.Windows.Forms.Button selectPathBtn;
        private System.Windows.Forms.Label assignmentLbl;
        private System.Windows.Forms.ComboBox assignmentCB;
        private System.Windows.Forms.Label downloadLbl;
        private System.Windows.Forms.Button downloadBtn;
        private System.Windows.Forms.TabControl tabControl1;
        private System.Windows.Forms.TabPage tabPage1;
        private System.Windows.Forms.TabPage tabPage2;
        private System.Windows.Forms.Label XMLocationLbl;
        private System.Windows.Forms.Button uploadBtn;
        private System.Windows.Forms.Button findXMLBtn;
    }
}
```

Main form logic C#

```
using MarkItDesktop.Data_Containers;
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;

namespace MarkItDesktop
{
    public partial class pathBtn : Form
    {
        DatabaseEnquerier databaseConnection;
        String pathToCSV, pathToXML;
        int assignmentID;

        public pathBtn()
        {

            InitializeComponent();
            DatabaseEnquerier enquerier = new DatabaseEnquerier();
            this.databaseConnection = enquerier;
            RefreshAssignmentComboBox();
        } // On initialisation, set a global communicator with the database and get
the combobox populated

        private void RefreshAssignmentComboBox()
        {
            assignmentCB.DataSource = databaseConnection.GetAssignmentListFromCloud();
        }

        private void selectPathBtn_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog dialog = new FolderBrowserDialog();

            if (dialog.ShowDialog() == DialogResult.OK)
            {
                pathToCSV = dialog.SelectedPath;
                downloadLbl.Text = "Download location:\n" + dialog.SelectedPath;
            }
        } // Gets path to download location, sets it in UI

        private void downloadBtn_Click_1(object sender, EventArgs e)
        {
            Dictionary<int, int> studentIDStudentGradeData =
databaseConnection.GetStudentGradesFromCloud((int)assignmentCB.SelectedItem);
            var filepath = pathToCSV + "\\downloadedAssigment" +
assignmentCB.SelectedItem + ".csv";
            using (StreamWriter writer = new StreamWriter(new FileStream(filepath,
            FileMode.Create, FileAccess.Write)))
            {
                writer.WriteLine("StudentID,StudentMarks");

                foreach (var line in studentIDStudentGradeData)
                {
                    writer.WriteLine(line.Key + "," + line.Value);
                }
            }

            MessageBox.Show("Download complete");
```

```
        } // Uses methods from the enquirer class to download the data, writes it to a
CSV and saves it in specified location

        private void findXMLBtn_Click(object sender, EventArgs e)
        {
            OpenFileDialog dialog = new OpenFileDialog();

            if (dialog.ShowDialog() == DialogResult.OK)
            {
                pathToXML = dialog.FileName;
                XMLocationLbl.Text = "XML location:\n" + dialog.FileName;
            }
        } // Allows user to select path to XML to upload

        private void uploadBtn_Click(object sender, EventArgs e)
        {
            XMLReader XMLManager = new XMLReader();

            MarkScheme markSchemeInfo = new MarkScheme();
            LabGroup labGroupInfo = new LabGroup();
            List<MarkSchemeSection> markSchemeSectionsInfo = new
List<MarkSchemeSection>();
            List<MarkSchemePart> markSchemePartsInfo = new List<MarkSchemePart>();

            if (XMLManager.GetMarkSchemeInfo(pathToXML, ref markSchemeInfo))
            {
                if (XMLManager.GetLabGroupInfo(pathToXML, ref labGroupInfo))
                {
                    assignmentID = markSchemeInfo.markSchemeAssignmentID;

                    if (XMLManager.GetMarkSchemeSectionsInfo(pathToXML, assignmentID,
ref markSchemeSectionsInfo))
                    {
                        if (XMLManager.GetMarkSchemePartsInfo(pathToXML, assignmentID,
ref markSchemePartsInfo))
                        {

                        }
                        else
                        {
                            MessageBox.Show("Error in reading parts from XML, check
file and retry");
                        }
                    }
                    else
                    {
                        MessageBox.Show("Error in sections parts from XML, check file
and retry");
                    }
                }
                else
                {
                    MessageBox.Show("Error in reading lab group information from XML,
check file and retry");
                }
            }
            else
            {
                MessageBox.Show("Error in reading mark scheme information from XML,
check file and retry");
            }
```

```
            if (databaseConnection.UploadMarkScheme(markSchemeInfo))
            {
                if (databaseConnection.UploadLabGroup(labGroupInfo))
                {
                    if
(databaseConnection.UploadMarkSchemeSections(markSchemeSectionsInfo))
                    {
                        if
(databaseConnection.UploadMarkSchemePart(markSchemePartsInfo))
                        {
                        }
                        else
                        {
                            MessageBox.Show("Error in part upload, retry");
                        }
                    }
                    else
                    {
                        MessageBox.Show("Error in section upload, retry");
                    }
                }
                else
                {
                    MessageBox.Show("Error in lab group upload, retry");
                }
            }
            else
            {
                MessageBox.Show("Error in mark scheme upload, retry");
            }

            MessageBox.Show("Upload complete");
        }
    }
} // Takes XML, reads and stores its data using XMLReader methods, uploads that
information using enquirer methods. Safety built in via if/else boolean statements
```

XML Reader C# Class

```
using MarkItDesktop.Data_Containers;
using System;
using System.Collections.Generic;
using System.Xml;

namespace MarkItDesktop
{
    // This whole class is a utility to read the specific XML files created as
templates. The files can be modified by the user and fed into this app which will read
through this class and upload to the cloud
    class XMLReader
    {
        public bool GetMarkSchemeInfo(String pathToXML, ref MarkScheme
inputMarkScheme)
        {
            try
            {
                XmlDocument doc = new XmlDocument();
                doc.Load(pathToXML);

                // Finds XML tags with that name
                XmlNodeList elemList = doc.GetElementsByTagName("assignmentName");
                for (int i = 0; i < elemList.Count; i++)
                {
                    // Gets inner text and saves it into programmatical data structure
for later use
                    inputMarkScheme.markSchemeAssignmentName = elemList[i].InnerText;
                }
                XmlNodeList elemList1 = doc.GetElementsByTagName("assignmentNumber");
                for (int i = 0; i < elemList1.Count; i++)
                {
                    inputMarkScheme.markSchemeAssignmentID =
int.Parse(elemList1[i].InnerText);
                }
                XmlNodeList elemList2 =
doc.GetElementsByTagName("assignmentAvailableMarks");
                for (int i = 0; i < elemList2.Count; i++)
                {
                    inputMarkScheme.markSchemeAssignmentAvailableMarks =
int.Parse(elemList2[i].InnerText);
                }
                XmlNodeList elemList4 = doc.GetElementsByTagName("assignmentAuthor");
                for (int i = 0; i < elemList4.Count; i++)
                {
                    inputMarkScheme.markSchemeAssignmentAuthor =
elemList4[i].InnerText;
                }
                XmlNodeList elemList5 =
doc.GetElementsByTagName("assignmentLabGroup");
                for (int i = 0; i < elemList4.Count; i++)
                {
                    inputMarkScheme.markSchemeAssignmentAuthor =
elemList4[i].InnerText;
                }

                return true;
            }
            catch (Exception e)
```

```
            {
                return false;
            }
        } // Reads XML and extracts Mark Scheme info into a data structure passed by
reference

        public bool GetLabGroupInfo(String pathToXML, ref LabGroup inputLabGroup)
        {
            try
            {
                XmlDocument doc = new XmlDocument();
                doc.Load(pathToXML);

                XmlNodeList elemList = doc.GetElementsByTagName("LabGroupLocation");
                for (int i = 0; i < elemList.Count; i++)
                {
                    inputLabGroup.labGroupLocation = elemList[i].InnerText;
                }
                XmlNodeList elemList1 = doc.GetElementsByTagName("LabGroupNumber");
                for (int i = 0; i < elemList1.Count; i++)
                {
                    inputLabGroup.labGroupNumber = int.Parse(elemList1[i].InnerText);
                }
                XmlNodeList elemList2 = doc.GetElementsByTagName("LabGroupStudentID");
                for (int i = 0; i < elemList2.Count; i++)
                {
                    inputLabGroup.labGroupStudentID =
int.Parse(elemList2[i].InnerText);
                }
                XmlNodeList elemList4 = doc.GetElementsByTagName("LabGroupUnit");
                for (int i = 0; i < elemList4.Count; i++)
                {
                    inputLabGroup.labGroupUnit = elemList4[i].InnerText;
                }
                XmlNodeList elemList5 = doc.GetElementsByTagName("assignmentNumber");
                for (int i = 0; i < elemList4.Count; i++)
                {
                    inputLabGroup.labGroupAssignmentID =
int.Parse(elemList5[i].InnerText);
                }

                return true;
            }
            catch (Exception e)
            {
                return false;
            }
        } // Reads XML and extracts Lab Group info into a data structure passed by
reference

        public bool GetMarkSchemeSectionsInfo(String pathToXML, int assingmentID, ref
List<MarkSchemeSection> inputMarkSchemeSectionList)
        {
            try
            {
                XmlDocument doc = new XmlDocument();
                doc.Load(pathToXML);

                XmlNodeList elemList = doc.GetElementsByTagName("section");
                for (int i = 0; i < elemList.Count; i++)
                {
                    inputMarkSchemeSectionList.Add(new MarkSchemeSection());
```

```
                }

                for (int i = 0; i < elemList.Count; i++)
                {
                    inputMarkSchemeSectionList[i].sectionAssignmentID = assingmentID;

                    XmlNodeList sectionNumber =
doc.GetElementsByTagName("sectionNumber");
                    for (int j = 0; j < sectionNumber.Count; j++)
                    {
                        inputMarkSchemeSectionList[j].sectionSectionID =
int.Parse(sectionNumber[j].InnerText);
                    }

                    XmlNodeList sectionName = doc.GetElementsByTagName("sectionName");
                    for (int j = 0; j < sectionName.Count; j++)
                    {
                        inputMarkSchemeSectionList[j].sectionName =
sectionName[j].InnerText;
                    }

                    XmlNodeList sectionAvailableMarks =
doc.GetElementsByTagName("sectionAvailableMarks");
                    for (int j = 0; j < sectionAvailableMarks.Count; j++)
                    {
                        inputMarkSchemeSectionList[j].sectionAvailableMarks =
int.Parse(sectionAvailableMarks[j].InnerText);
                    }

                    XmlNodeList sectionAuthor =
doc.GetElementsByTagName("sectionAuthor");
                    for (int j = 0; j < sectionAuthor.Count; j++)
                    {
                        inputMarkSchemeSectionList[j].sectionAuthor =
sectionAuthor[j].InnerText;
                    }
                }

                return true;
            }
            catch (Exception e)
            {
                return false;
            }
        } // Reads XML and extracts Mark Scheme Section info into a data structure
passed by reference

        public bool GetMarkSchemePartsInfo(String pathToXML, int assingmentID, ref
List<MarkSchemePart> inputMarkSchemePartsList)
        {
            try
            {
                XmlDocument doc = new XmlDocument();
                doc.Load(pathToXML);

                XmlNodeList amountOfParts = doc.GetElementsByTagName("part");
                foreach (var part in amountOfParts)
                {
                    inputMarkSchemePartsList.Add(new MarkSchemePart());
                }
                int i = 0;
                int j = 0;
```

```
                XmlNodeList sectionList = doc.GetElementsByTagName("section");
                foreach (XmlNode section in sectionList)
                {
                    XmlNodeList partsWithinSection = section.SelectNodes("part");

                    foreach (XmlNode part in partsWithinSection)
                    {
                        inputMarkSchemePartsList[j].partSectionID =
int.Parse(sectionList[i].FirstChild.InnerText);
                        inputMarkSchemePartsList[j].partAssignmentID = assingmentID;

                        XmlNodeList partNumber =
doc.GetElementsByTagName("partNumber");
                        for (int partNumberIterator = 0; partNumberIterator <
partNumber.Count; partNumberIterator++)
                        {
                            inputMarkSchemePartsList[partNumberIterator].partPartID =
int.Parse(partNumber[partNumberIterator].InnerText);
                        }

                        XmlNodeList partName = doc.GetElementsByTagName("partName");
                        for (int partNameIterator = 0; partNameIterator <
partName.Count; partNameIterator++)
                        {
                            inputMarkSchemePartsList[partNameIterator].partName =
partName[partNameIterator].InnerText;
                        }

                        XmlNodeList partAvailableMarks =
doc.GetElementsByTagName("partAvailableMarks");
                        for (int partMarksIterator = 0; partMarksIterator <
partAvailableMarks.Count; partMarksIterator++)
                        {

inputMarkSchemePartsList[partMarksIterator].partAvailableMarks =
int.Parse(partAvailableMarks[partMarksIterator].InnerText);
                        }

                        XmlNodeList partAuthor =
doc.GetElementsByTagName("partAuthor");
                        for (int partAuthorIterator = 0; partAuthorIterator <
partAuthor.Count; partAuthorIterator++)
                        {
                            inputMarkSchemePartsList[partAuthorIterator].partAuthor =
partAuthor[partAuthorIterator].InnerText;
                        }
                        j++;
                    }
                    i++;
                }

                return true;
            }
            catch (Exception e)
            {
                return false;
            }
        } // Reads XML and extracts Mark Scheme Part info into a data structure passed
by reference
    }
}
```

Data Enquirer C# Class

```
using MarkItDesktop.Data_Containers;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Text;

namespace MarkItDesktop
{
   class DatabaseEnquerier
   {
      public List<int> GetAssignmentListFromCloud()
      {
         List<int> assignments = new List<int>();

         try
         {
            // SQL connection info
            SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
            builder.DataSource = "tarambana.database.windows.net";
            builder.UserID = "tasa";
            builder.Password = "2019Green!";
            builder.InitialCatalog = "bookChoice";

            using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
            {
               // Open connection and build query
               connection.Open();
               StringBuilder sb = new StringBuilder();
```

```
        sb.Append("SELECT[TotalAssignmentID]");
        sb.Append("FROM[dbo].[StudentTotalMarks]");
        String sql = sb.ToString();

        using (SqlCommand command = new SqlCommand(sql, connection))
        {
          // execute query command, take result and read it
          using (SqlDataReader reader = command.ExecuteReader())
          {
            while (reader.Read())
            {
              String data = reader["TotalAssignmentID"].ToString();
              if (!assignments.Contains(int.Parse(data)))
              {
                assignments.Add(int.Parse(data));
              }
            }
          }
        }
      }
    }
    catch (SqlException e)
    {
      Console.WriteLine(e.ToString());
    }

    return assignments;
} // returns the list of assignments from the cloud to populate the dropbox

public Dictionary<int, int> GetStudentGradesFromCloud(int assignmentID)
{
```

```
Dictionary<int, int> studentIDStudentGrade = new Dictionary<int, int>();

try
{
   SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
   builder.DataSource = "tarambana.database.windows.net";
   builder.UserID = "tasa";
   builder.Password = "2019Green!";
   builder.InitialCatalog = "bookChoice";

   using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
   {
      connection.Open();
      StringBuilder sb = new StringBuilder();
      sb.Append("SELECT [TotalMarksAchieved], [TotalStudentID]");
      sb.Append("FROM[dbo].[StudentTotalMarks]");
      sb.Append("WHERE [TotalAssignmentID] = ");
      sb.Append(assignmentID);
      sb.Append("ORDER BY [createdAt] DESC");
      String sql = sb.ToString();

      using (SqlCommand command = new SqlCommand(sql, connection))
      {
         using (SqlDataReader reader = command.ExecuteReader())
         {
            while (reader.Read())
            {
               if
(!studentIDStudentGrade.ContainsKey(Convert.ToInt32(reader["TotalStudentID"])))
                  {
                  studentIDStudentGrade.Add(Convert.ToInt32(reader["TotalStudentID"]),
```

```
                        Convert.ToInt32(reader["TotalMarksAchieved"]));
                    }
                }
            }
        }
    }
    catch (SqlException e)
    {
        Console.WriteLine(e.ToString());
    }

    return studentIDStudentGrade;
} // returns studentID-studentGrade pairs in order to write into CSV file

public bool UploadMarkScheme(MarkScheme markSchemeToUpload)
{
    try
    {
        SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
        builder.DataSource = "tarambana.database.windows.net";
        builder.UserID = "tasa";
        builder.Password = "2019Green!";
        builder.InitialCatalog = "bookChoice";

        using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
        {
            using (SqlCommand command = new SqlCommand())
            {
                command.Connection = connection;
                command.CommandType = CommandType.Text;
```

```
                command.CommandText = "INSERT INTO dbo.MarkScheme
(MarkSchemeAssignmentName, MarkSchemeAuthor, MarkSchemeAssignmentAvailableMarks,
MarkSchemeAssignmentID) VALUES (@MarkSchemeAssignmentName, @MarkSchemeAuthor,
@MarkSchemeAssignmentAvailableMarks, @MarkSchemeAssignmentID)";

                command.Parameters.AddWithValue("@MarkSchemeAssignmentName",
markSchemeToUpload.markSchemeAssignmentName);

                command.Parameters.AddWithValue("@MarkSchemeAuthor",
markSchemeToUpload.markSchemeAssignmentAuthor);

                command.Parameters.AddWithValue("@MarkSchemeAssignmentAvailableMarks",
markSchemeToUpload.markSchemeAssignmentAvailableMarks);

                command.Parameters.AddWithValue("@MarkSchemeAssignmentID",
markSchemeToUpload.markSchemeAssignmentID);


                connection.Open();

                command.ExecuteNonQuery();

                return true;

            }

        }

        catch (SqlException e)

        {

            Console.WriteLine(e.ToString());

            return false;

        }

    } // SQL insert of the Mark Scheme


    public bool UploadMarkSchemeSections(List<MarkSchemeSection>
markSchemeSectionsToUpload)

    {

        try

        {

            SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();

            builder.DataSource = "tarambana.database.windows.net";

            builder.UserID = "tasa";
```

```
            builder.Password = "2019Green!";
            builder.InitialCatalog = "bookChoice";

            foreach (MarkSchemeSection section in markSchemeSectionsToUpload)
            {
               using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
               {
                  using (SqlCommand command = new SqlCommand())
                  {

                        command.Connection = connection;
                        command.CommandType = CommandType.Text;
                        command.CommandText = "INSERT INTO dbo.MarkSchemeSection (SectionAuthor,
SectionName, SectionAssignmentID, SectionAvailableMarks, SectionID) VALUES (@SectionAuthor,
@SectionName, @SectionAssignmentID, @SectionAvailableMarks, @SectionID)";
                        command.Parameters.AddWithValue("@SectionAuthor", section.sectionAuthor);
                        command.Parameters.AddWithValue("@SectionName", section.sectionName);
                        command.Parameters.AddWithValue("@SectionAssignmentID",
section.sectionAssignmentID);
                        command.Parameters.AddWithValue("@SectionAvailableMarks",
section.sectionAvailableMarks);
                        command.Parameters.AddWithValue("@SectionID", section.sectionSectionID);

                        connection.Open();
                        command.ExecuteNonQuery();
                        connection.Close();
                     }
                  }
               }
               return true;
            }
            catch (SqlException e)
```

```
    {
        Console.WriteLine(e.ToString());
        return false;
    }
} // SQL insert of the Mark Scheme Sections

public bool UploadMarkSchemePart(List<MarkSchemePart> markSchemePartsToUpload)
{
    try
    {
        SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
        builder.DataSource = "tarambana.database.windows.net";
        builder.UserID = "tasa";
        builder.Password = "2019Green!";
        builder.InitialCatalog = "bookChoice";

        foreach (MarkSchemePart part in markSchemePartsToUpload)
        {
          using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
          {
             using (SqlCommand command = new SqlCommand())
             {


                    command.Connection = connection;
                    command.CommandType = CommandType.Text;
                    command.CommandText = "INSERT INTO dbo.MarkSchemePart (PartAuthor,
PartName, PartAssignmentID, PartAvailableMarks, PartSectionID, PartPartID) VALUES (@PartAuthor,
@PartName, @PartAssignmentID, @PartAvailableMarks, @PartSectionID, @PartPartID)";
                    command.Parameters.AddWithValue("@PartAuthor", part.partAuthor);
                    command.Parameters.AddWithValue("@PartName", part.partName);
```

```
                    command.Parameters.AddWithValue("@PartAssignmentID",
part.partAssignmentID);
                    command.Parameters.AddWithValue("@PartAvailableMarks",
part.partAvailableMarks);
                    command.Parameters.AddWithValue("@PartSectionID", part.partSectionID);
                    command.Parameters.AddWithValue("@PartPartID", part.partPartID);

                    connection.Open();
                    command.ExecuteNonQuery();
                    connection.Close();
                }
            }
        }
        return true;
    }
    catch (SqlException e)
    {
        Console.WriteLine(e.ToString());
        return false;
    }
} // SQL insert of the Mark Scheme Parts

public bool UploadLabGroup(LabGroup labGroupToUpload)
{
    try
    {
        SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
        builder.DataSource = "tarambana.database.windows.net";
        builder.UserID = "tasa";
        builder.Password = "2019Green!";
        builder.InitialCatalog = "bookChoice";
```

```
            using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
            {
               using (SqlCommand command = new SqlCommand())
               {
                  command.Connection = connection;
                  command.CommandType = CommandType.Text;
                  command.CommandText = "INSERT INTO dbo.LabGroup (LabGroupAssignmentID,
LabGroupLocation, LabGroupNumber, LabGroupStudentID, LabGroupUnit) VALUES
(@labGroupAssignmentID, @labGroupLocation, @labGroupNumber, @labGroupStudentID,
@labGroupUnit)";
                  command.Parameters.AddWithValue("@labGroupAssignmentID",
labGroupToUpload.labGroupAssignmentID);
                  command.Parameters.AddWithValue("@labGroupLocation",
labGroupToUpload.labGroupLocation);
                  command.Parameters.AddWithValue("@labGroupNumber",
labGroupToUpload.labGroupNumber);
                  command.Parameters.AddWithValue("@labGroupStudentID",
labGroupToUpload.labGroupStudentID);
                  command.Parameters.AddWithValue("@labGroupUnit",
labGroupToUpload.labGroupUnit);

                  connection.Open();
                  command.ExecuteNonQuery();
                  return true;
               }
            }
         }
         catch (SqlException e)
         {
            Console.WriteLine(e.ToString());
            return false;
         } // SQL insert of the Mark Scheme LabGroup
      }
   }}
```

Mark It Windows C# Data Containers

Mark Scheme Class

```
using System;

namespace MarkItDesktop.Data_Containers
{
    class MarkScheme
    {
        public int markSchemeAssignmentAvailableMarks { get; set; }

        public String markSchemeAssignmentName { get; set; }

        public String markSchemeAssignmentAuthor { get; set; }

        public int markSchemeAssignmentID { get; set; }

        public MarkScheme()
        {

        }
    }
}
```

Mark Scheme Section Class

```
using System;

namespace MarkItDesktop.Data_Containers
{
    class MarkSchemeSection
    {
        public String sectionAuthor { get; set; }

        public String sectionName { get; set; }

        public int sectionAssignmentID { get; set; }

        public int sectionAvailableMarks { get; set; }

        public int sectionSectionID { get; set; }

        public MarkSchemeSection()
        {

        }
    }
}
```

Mark Scheme Part Class

```
using System;

namespace MarkItDesktop.Data_Containers
{
    class MarkSchemePart
    {
        public String partAuthor { get; set; }

        public String partName { get; set; }

        public int partAvailableMarks { get; set; }

        public int partAssignmentID { get; set; }

        public int partSectionID { get; set; }

        public int partPartID { get; set; }

        public MarkSchemePart()
        {

        }
    }
}
```

LabGroup Class

```
using System;

namespace MarkItDesktop.Data_Containers
{
    class LabGroup
    {

        public int labGroupStudentID { get; set; }

        public int labGroupNumber { get; set; }

        public String labGroupLocation { get; set; }

        public int labGroupAssignmentID { get; set; }

        public string labGroupUnit { get; set; }

        public DateTime labGroupDate { get; set; }

    }
}
```