School of Electronic Engineering and Computer Science

MSc Computing and Information Systems
Project Report 2019

Student Feedback System

Yordan Kyosev

Queen Mary
University of London

August 2019

Acknowledgement

I would like to thank my supervisor Dr Fabrizio Smeraldi and his colleague Dr Maryam Abdollahyan for their advice and thoughtful recommendations. Moreover, I would like to thank my tutor Dr Tony Stockman and all members of the School of Electronic Engineering and Computer Science at Queen Mary about their invaluable guidance and efforts. Finally, I am grateful to my family and friends for their positivity and continuous support throughout the entire postgraduate course.

Abstract

In an effort to collect valuable student feedback at Queen Mary University of London, the report describes the development of a student feedback system built around Raspberry Pi microcomputer. The aim of this project is to present an alternative way of collecting student feedback. The system is a standalone device, which would be located in university premises. Students would be able to provide their feedback on their lecture/seminar experience. The report studies the importance of feedback in higher education and reviews other student feedback approaches before providing a detailed description of the hardware and software development and implementation of system. Considering the strengths, limitations and room for improvements, it appears the Raspberry Pi student feedback system would be a useful addition to current mechanisms of gathering student feedback in a move to improve the quality of education.

Table of Contents

Table of Figures

Introduction

This report describes the development of a prototype system for collecting student feedback built using a Raspberry Pi microcomputer. The student feedback system consists of a standalone device, located in a lecture or seminar room. A sign by the device would read: “How was your lecture/seminar experience today?”. Students would be able to provide feedback by pressing buttons on the device. There are three buttons labelled “Bad”, “OK” and “Great”, which students could press depending on their lecture or seminar experience. The device would save records on its memory but data would be sent to a server as well. The server would be accessible via a web application on the university local network. Users connected to the secure university network would have access to all the records. Feedback information would be accessible and three functions are available to search by date, device location and feedback answer.

The report introduces what feedback is, its importance and role in the world of business. Different factors influenced higher education institutions, which started to use various marketing techniques in the hope of winning more students. In this way, collecting student feedback became as valuable as in business. Several ways of obtaining student feedback and their uses are described. The paper provides a detailed hardware and software implementations of the Raspberry Pi student feedback system, followed by testing explaining issues and solutions throughout the project. The features, strengths, limitations and possible further works of the built feedback system are discussed.

Motivation

As a student at Queen Marry University of London, I would like my opinion to reach teachers and staff and assist in improving both students and staff experience. I believe the Raspberry Pi feedback system would be a good addition to the current student feedback mechanisms. Using the student feedback system together with other methods of collecting feedback could assist in evaluating student feedback and benefit both students and the university.

In addition, I have chosen this project as it involves programming, which would be a great preparation for a career in web development. The Raspberry Pi, running its own version of Linux, is a great educational platform to improve my coding skills. The hardware implementation in itself is an interesting area I would like practise further.

Literature review

In the business community, feedback becomes an asset, which enterprises can readily utilise and adjust to customers' and/or clients' demands. It is also a powerful and extremely versatile tool in informing the long-term decision-making in a business.

Clearly, businesses cannot exist without customers and learning their opinions on products and services has always been sought after. Today, business consultants regularly use expressions like "customer focus", "customer care", "customer satisfaction" and more and the trend of debating customers' complains and experience has been intensifying since the 1980s. Customers gradually became seen as not only the business' payers, but also as those who would benefit from products and services. It is this line of thought, which brought the ultimate definition of "customers" to, for instance, hospitals and universities and their patients and students respectively. Customers talk about their experience with others and receiving their feedback, being positive or negative, can help organisations improve the quality of their services (Barlow and Moller, 1996).

A customer experience rating for a service could be very subjective as it is based on personal perception. Similarly to a services business, education institutions would receive students' feedback, which could be subjective, as students would judge their experience in different ways. A service-oriented business is different from a product-based business due the latter normally involving the customer in the "production" cycle of tangible goods generally accessible to the public, whilst the former would rely heavily on the level of service it is expected to achieve without being able to demonstrate it to the "world" in any physical form. Making use of customer feedback is therefore a good opportunity for an educational organisation to see how it performs at any moment in time, especially in the absence of a production process of tangible goods.

Marketing literature constantly suggests to businesses to maintain good relationships with customers and pay attention to their opinions. Marketing concepts were adopted by universities because they started to perceive the students as their clients, especially following the proliferation of institutions offering various forms of higher education, and the influx of international students, both of which intensified competition amongst universities in the UK and Europe. Although, there is an agreement that a "customer – provider" relationship in education is not like any other as seen in other commercial environments, undoubtedly it still does exist. Furthermore, there has been a lot of focus on customer choice and how universities need to provide courses, which would appeal to students rather than universities teaching what they believe is important (Pitman, 2000).

Another factor that causes students to feel like customers is the substantial tuition fees they need to pay for education. For many years, this has been the case in the United States. In recent years, there has been a significant increase in costs students were expected to pay in the United Kingdom as well. Individuals who pay for a service are called customers and considering the vast amounts of money paid, they would expect a corresponding high quality of education (Eagle and Brennan, 2007).

Today, a university degree could not be enough to secure the best jobs. It might be necessary for students to undertake further studies to improve career prospects. Many education institutions see the opportunity of repeated business with students. They have established marketing departments to aid bringing more students. As institutes have started using more marketing tools, students accepted their new roles as customers. Since 1960s, governments introduced legislations and trade practices about consumer rights to ensure consumers are protected against dishonest providers of products and services. These circumstances created new relations between higher education institutions and students, where students are perceived in some sense as consumers also. At this point, models used at the marketplace were introduced to students and they began to express their opinions on courses, teaching methods and facilities (Svensson and Wood, 2007).

However, some researchers critique the collection of student feedback and its uses. They question students' understanding on educational quality. For example, it is not clear how students in their first year at university would be able to judge the quality of education considering their limited experience and understanding. Student feedback could also be biased. Students could express concerns towards their short-term goals of completing chosen courses successfully, rather than providing feedback focused on a quality education in the long-term (Svensson and Wood, 2007).

Higher education institutions analyse student opinions in order to be more competitive and recruit more students. Generally, student feedback could be useful to education institutions for three main reasons: showing teaching effectiveness for the purpose of salary increases and promotions, giving reviews to teachers in order to improve quality of instructions and assisting other students choose the right university and course (Cohen, 1980).

Feedback collection

Although, universities use a variety of formal and informal mechanisms to obtain student feedback, informal ways of collecting feedback between students and teachers have decreased over the years in the UK. This was caused by a change in the student-staff ratio, which made the informal collection of student feedback less effective. Due to quality assurance implications, more formalised ways of collecting student feedback were established. There has been a growing competition in higher education with the introduction of ranking tables, key performance indicators and consumer choice statistics comparing service standards between universities. Organisational changes of courses into modular forms also led to further decrease in the possibility of informal communications between students and teachers (Brennan and Williams, 2004).

Different types of qualitative and quantitative mechanisms to gather student feedback exist. For universities to be more effective, a variety of mechanisms are used. The following mechanisms are used regularly in collecting student feedback: questionnaires, student-staff liaison committees, discussion groups, lectures, tutorials and informal meetings (Brennan and Williams, 2004).

Questionnaires, a formal way of collecting feedback, are the most commonly used in higher education. They could be used with all students, regardless of their course and level. Most questionnaires do not take lots of time for completion and can give qualitative and quantitative data. However, appropriate knowledge in design and analysis of questions is required to create an effective questionnaire. Otherwise, reviewers would spend more time on analysis, which can increase costs, lead to low response rates and difficulties in making decisions on unclear results (Brennan and Williams, 2004). In 2005, the first National Student Survey questionnaire was introduced to students in the UK. It asks questions in various areas such as teaching, learning opportunities and resources, course organisation, etc. The aim of the survey is to collect feedback regarding student courses, teaching quality, student experience in higher education (Fry, Ketteridge and Marshall, 2010).

Another popular method of collecting student feedback are the student-staff liaison committees. These are formal meetings between student representatives and staff giving an opportunity for continuing discussions. During the meetings, different ideas can be discussed and decisions could be taken quickly providing immediate changes based on student feedback. A lack of representatives' visibility among other students is the main limitation of this method. If representatives do not have enough opportunities to engage with students or they lack motivation, it would be difficult for students to express their concerns to the representatives (Brennan and Williams, 2004).

One more mechanism used to obtain qualitative student feedback is discussion groups. Staff are not represented in the discussion groups and they are not used by a large number of institutions. This method could be useful for already known issues and/or "one-off" meetings, which need to be discussed between students to reach an agreement. Discussions are focused on a specific issue, which does not represent the whole student population. Meetings are held outside of class, which can be time consuming (Brennan and Williams, 2004).

Lectures and seminars are places, where student feedback is gathered as well. Although some students might not feel comfortable with sharing their views on teaching quality and overall experience in front of teachers, some actions could be taken immediately and issues dealt with quickly. There is a risk for the discussion being dominated by vocal students, which lead to ineffective communication (Brennan and Williams, 2004).

For more personal student experience, one-to-one tutorials and meetings could be arranged. This gives students the ability to discuss sensitive and confidential matters with staff. As feedback is based on personal account rather than group facts, it might be difficult to gather information, which could be a useful source to all staff (Brennan and Williams, 2004).

Each of the mechanisms for collection of student feedback collection have its advantages and disadvantages. Depending on institutional requirements, different techniques or a mixture of them could be used. For example, questionnaires would be suitable for a university wide satisfaction survey on student equipment. However, an informal meeting could be used on a small group of students to gather feedback on teaching material of a module.

Feedback can be collected in traditional way using pen and paper, but it would be more time consuming for students and universities to provide reviews, group and evaluate results respectively. According to a study comparing paper and online student surveys at university, it appeared that online surveys were cheaper and response time was higher than paper ones. Some of the main reasons for the cost efficiency were lower mailing costs and no costs on data entry. Moreover, online responses were more detailed than paper feedback provided (Ballantyne, 2004).

Student feedback systems

A customer feedback system assists businesses to improve their performance. The system collects user feedback, which is analysed using different techniques. The data gives valuable information about the strengths and weaknesses of organisations. The student feedback system works in the same way, which would give universities an opportunity to analyse feedback received. Some types of student feedback systems are described below.

A popular in-class feedback system used by many institutions is the classroom response system or also called a “clicker” system. Every student is given a handheld device with buttons, which transmits student replies to a computer displaying results immediately in the classroom. Clickers could be used as a teaching tool encouraging students to take part in-class exercises and discussions. Moreover, students can participate in classroom feedback surveys using clickers. Results would be publicly displayed and discussed between teachers and students. According to Robson and Johnson (2008), a number of studies found that classroom response system is very effective in creating interactive classes and providing immediate feedback.

For example, a teacher can ask a question, analyse responses from students and figure out whether or not students understand a lecture topic. If students do not have a good understanding of the material, the teacher can improve topic presentation, add more detailed notes or spend more time on the difficult parts of the syllabus. Hence, clickers could be a useful tool to both students and teachers. The literature indicates that classroom response systems promote learning and focus on student engagement. However, some studies do not find clickers effective and note that further research is needed to understand the strengths and weakness of the technology (Fies and Marshall, 2006).

Another approach to collect student feedback is through classroom assessment techniques (CATs). Traditionally, CATs used pen and paper method by giving evaluation forms to students. CATs could help instructors in assessing and improving teaching strategies. There are many available CATs for evaluating and improving teaching. An example of CATs is the Teaching

Strategy Matrix in which instructors provide the teaching strategies used in a module and students rate them. Results would show the most effective teaching strategy used to achieve the learning outcomes according to students' votes. Later, electronic tools were combined with CATs to enhance design, delivery and evaluation of results. This is how technoCATs were created allowing student feedback to be submitted electronically, in-class or online. In this way, less in-class time would be used and potentially more students could provide feedback as they have the freedom of using the system in their own time as well (Lieberman et al, 2001).

A different system, developed by the University of Illinois and able to gather student feedback, is EON (Evaluation ONline). It is an online feedback system, which allows students to provide midterm and end of term feedback. Every student has login details to enter the system and leave a review. Students are sent reminders if they have not left feedback. Feedback is summarised in easy to read forms to assist results evaluation. Teaching instructors can see results immediately after the end of the evaluation period. This gives teachers the opportunity to act on feedback relating to quality of teaching (Bullock, 2003).

The systems above are able to collect student feedback in different ways to the Raspberry Pi student feedback system presented in the next section. The clicker system is mainly used in-class to assist lectures promoting interactive lectures and active student engagement. Its purpose is to be part of the teaching process rather than gathering student ratings. CATs can receive student feedback during class or online (technoCATs), same way as EON.

On the other hand, the Raspberry Pi system would collect student feedback after classes or seminars. The system would not directly support the teaching process during classes but it could have an effect on it subsequently if changes are introduced based on student feedback. Student feedback could be accessed immediately as in the systems above. Feedback could only be provided pressing buttons on the standalone device but other options as leaving feedback online would not be available. In order to improve student feedback analysis, it is also possible to combine the Raspberry Pi feedback system with other systems similar to the ones discussed above.

Hardware

The project involves the use of a microcomputer called Raspberry Pi. The system would record feedback when buttons linked to the Raspberry Pi computer are pressed. Here are the main hardware components:

- Raspberry Pi 3 Model B+ – currently, this is not the newest model as Raspberry Pi 4 was released on 24th June 2019. However, the model used in the project is equipped with a quad-core processor and built-in Wi-Fi. This model would be in production until 2023 and would be more than capable of running the required software for the project. Raspberry Pi is developed by the Raspberry Pi Foundation in the UK. According to Robert Mullins, a trustee at the Foundation, the main purpose of the computer is to get more people into programming and creating different projects rather than being just another device to consume content from (Andrews, 2013).
- 32 GB SD card – the card would play the role of a hard drive for the Raspberry Pi. The card size should be sufficient for the operating system and project software. After an installation of the operating system and software running the student feedback system, used space on the SD card would be about 5GB. Therefore, there would be plenty of free space on the SD card for saving student feedback records.

- USB keyboard & Mouse – these peripherals would be needed to set up the Raspberry Pi and code the software on it. However, they would not be part of the user feedback system, as user would only operate the device by pressing the push buttons.
- Push buttons – Tactile push buttons would be used for the project. The system would record student feedback when buttons are pressed. Moreover, they would be used as switches when building electrical circuits.
- Breadboard (generic) – It allows building electronic circuits and push electronic components into the holes. The breadboard consists of a grid of holes in which components could be easily added or removed. In this way, there would be no need to conduct soldering and/or desoldering, which makes the breadboard reusable. The holes are spread out at 2.54mm intervals from one another. Below the grid, there are a series of electrical contacts. The LED lights, push buttons and resistors would be the components located on the breadboard. Wires would connect the breadboard and the Raspberry Pi together (Upton and Halfacree, 2014).
- Jump wires – the wires are required to connect GPIO pins on the Raspberry Pi to components on the breadboard. For the purpose of the project male to female and male to male wires are used.
- LEDs – there will be three LED lights, which would light up when a button is pressed. In this way, the user would be notified when a record is noted by the student feedback system.

- Resistors – they would be required to build electrical circuits. It is always recommended to use resistors, which reduce the electrical current and prevent damaging the Raspberry Pi or components connected to it.
- Charger – a 2.5A charger is required to power the Raspberry Pi 3 Model B+ for proper performance of the microcomputer and peripherals.

The main hardware components used to build the student feedback system could be seen in Figure 1 below. Starting from the top, left to right, a charge, male to female jump wires, a resistor, a push button and LED lights are used in the project. The bottom part of Figure 1, left to right, shows the breadboard and Raspberry Pi 3 Model B+.

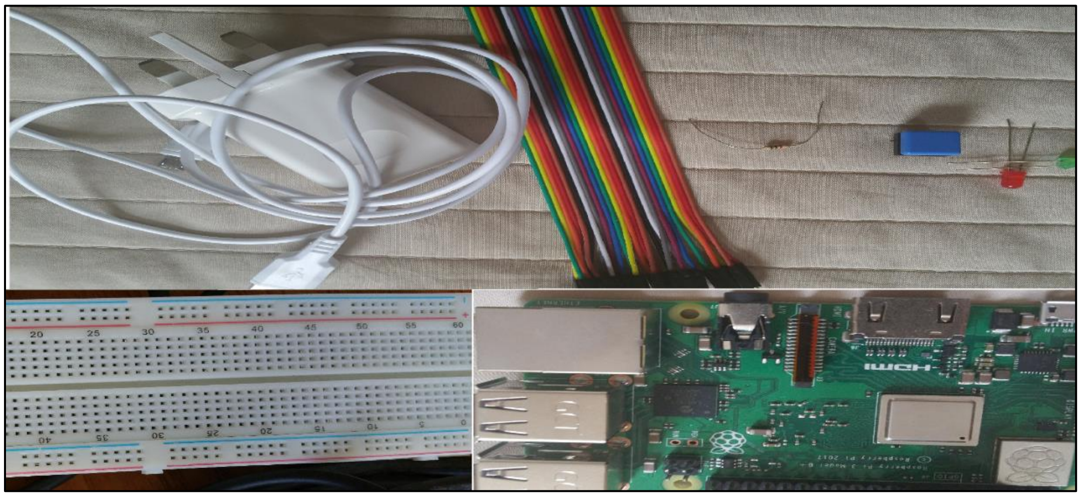


Figure 1. Hardware components

As a result of its specification, the Raspberry Pi Model 3 B+ is a computer, which could be used in a number of different developments. It has Bluetooth, WLAN, camera, video and other ports and features. Not all ports and their available functions were needed for student feedback system.

For the purpose of the project, one of the most powerful features of the Raspberry Pi is required. This is its GPIO (General Purpose Input Output) header. It allows the device to connect to the push buttons on the breadboard and power the LED lights. The Raspberry Pi and the breadboard are connected together using the male to female jump wires.

Figure 2 below shows the layout of the GPIO 40-pin header:

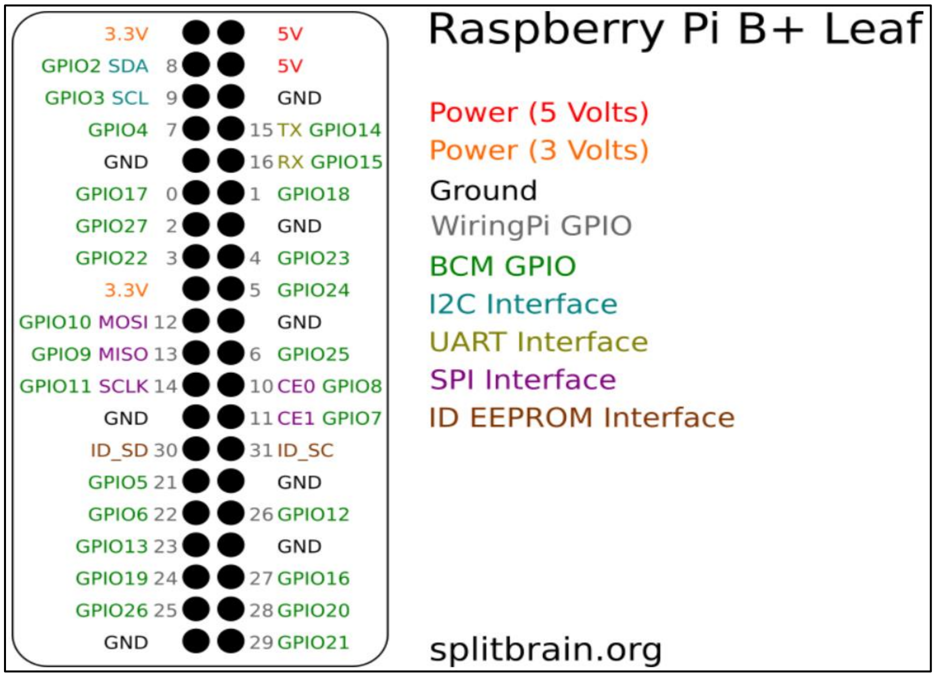


Figure 2. Raspberry Pi 40 Pin GPIO Header, GitHub, 2015

Pins can have two different numbers allocated to them. These are the BOARD number or BCM (Broadcom) number. Only one convention can be used in a project. There is no “right” or “wrong” numbering but some peripherals would specifically require BCM numbering. The numbering in Figure 2 is in BCM order. The same ordering is chosen for this project. Older versions had 26 GPIO pins, while the Raspberry Pi Model 3 B+ model has 40 GPIO pins. There are two 5V and 3V3 pins, eight Ground pins for circuits and two advanced ID EEPROM pins. The rest are GPIO general purpose input/output pins. Depending on their type, pins have different purposes.

Electricity flows in a circuit. It could be described as a loop or path as well. A switch is used to create or break a circuit. The buttons on the breadboard would be the physical switches (McManus and Cook, 2013).

Three LED lights would be used in the student feedback system. LED (light-emitting diode) lights are an example of output devices, which light up when electricity goes through them. Current is allowed to go only in one direction. LED lights are available in a variety of colours (Philbin, 2014). Three different LED colours are chosen for the project: red, yellow and green. LED lights have two legs and their length is different. The longer leg is Positive and the shorter one is Ground. Another visual indicator to distinguish between the two is the body of the LED light. The bottom part of the body is rounded apart from the one side, which is flat. This points out which leg is Ground. This is a useful indicator as legs could be cut depending on the needs of the project.

A resistor plays a very important role when creating electrical circuits. It is used to limit the current a voltage would pass through to a connected device in a circuit. For the project, three resistors are used for every connection between the Raspberry Pi GPIO pins and LED lights. The resistors would limit the supply of current to the LED lights. If the LED light requests more current than the GPIO is able to supply, this could cause damage to the GPIO pin or even the Raspberry Pi. On the other hand, the LED light could burn out by too much current flowing to it. There are different types of resistors, which are measured in Ohms. A resistor with more Ohms would offer larger resistance. The resistors used in the project are 1k Ohms. A colour coding is used to differentiate between different types, as there are no signs on the resistors. Colour of four, five or six bands exists and colours are along the body of the resistor. In order to recognise the type of the resistor, a person should know how to read the colour codes. Another option is to use a multimeter (McManus and Cook, 2013).

For the project, there will be three buttons, three LED lights and three resistors placed on the breadboard, which would be connected to the Raspberry Pi using male to female and male to male jump wires. A button, a LED light and a resistor would create one circuit. Three circuits would be created on the breadboard. When a button is pressed, a circuit would be created. If the button is not pressed, the switch is open and the current would not flow.

When connecting a LED light and a resistor together, it is advisable to use the right resistor by calculating the resistance. This could be achieved using the following formula:

$$\boldsymbol{R = \frac{(V - F)}{I}}$$

R is the resistance. V is the working voltage of a device or the LED light in this case. F is forward voltage of LED light, which is normally a constant and I is the maximum forward current of the LED light. R is measured in Ohms, V and F in volts and I in amps. The desired current in a typical LED light is 25 mA where 1 amp is 1000 mA. Voltage V is 3.3 V and forward voltage F is 1.7 V. Using the above formula: (3.3 – 1.7) / 0.025 = 64. Therefore, the smallest resistor, which is suitable to protect the LED light should be 64 Ohms. A resistor smaller than 64 Ohms would not be useful and the LED light would burn out. The nearest resistor existing is 68 Ohms. However, as noted above, the resistors, used in the project and provided by the Electronics Teaching laboratory at the university, are 1k Ohms. Usually, a resistor with a higher value could prevent the LED to light or make it dimmer. Nevertheless, LED lights perform satisfactory with 1k Ohms resistors (Upton and Halfacree, 2014).

Three tactile push buttons would be used in the project. Three colours, red, green and yellow are used to match the colours of the LED lights. Buttons have four legs and when a button is pressed the legs are connected in a circuit. When the button is not pressed, the current does not flow in the circuit, as it is not closed.

Components, which form the circuits, are grouped together on the breadboard. A red LED light with a resistor and a red button would be in one circuit. A yellow LED light, a resistor and a yellow button would be in another circuit. The last circuit includes a green LED light, a resistor and a green button. In this way, a button would be a switch creating a circuit and controlling a LED light of the same colour.

The breadboard used in the project has two power rails on each side. The two rails are identical and are not connected to each other. One rail has two holes in a row, one positive and one negative, labelled with red and blue lines respectively. There are two metal straps, which run vertically. In this way, components placed on the same horizontal line would be connected together. Each rail is divided in the middle into two sections. This leaves 25 holes in a column for each section. Between the two rails, a grid of holes is located. Inside the holes there are small metal clips, which make contact with the wires and legs of attached components. Metal clips are electronically connected to each other by row. The grid has ten holes on each row. However, the grid is split into two sections leaving five holes for each row in a section. Both sections are isolated from one another. Rows of the grid for the two sections are labelled with numbers going from zero to sixty. Columns of the grid for sections one and two are labelled from A to E and from F to J respectively. In this way, the location of components could be precisely specified. For example, a wire could be placed in a hole B5 and the location could be easily found.

The hardware setup on the breadboard for the three circuits is explained below. Furthermore, all connections on breadboard could be seen in Figure 3:

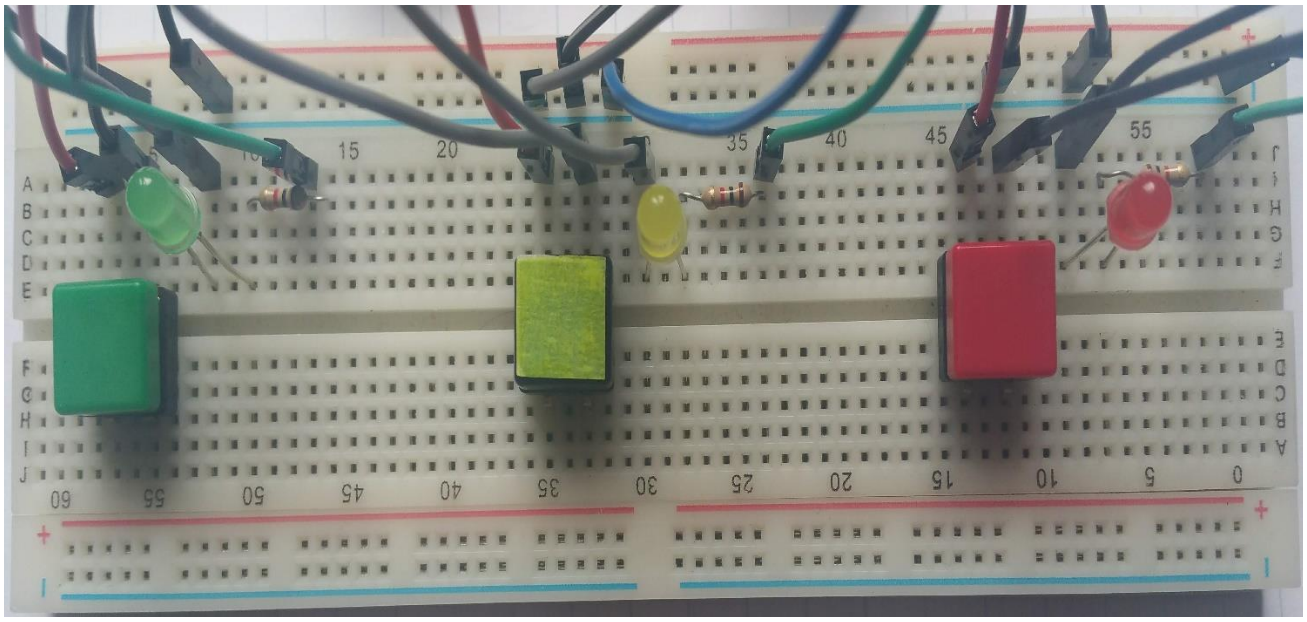


Figure 3. Breadboard connections

The power rail on the breadboard is split into two sections. For the first circuit, a wire goes to a Ground pin on the Raspberry Pi and its other end connects to a hole on the power rail. This connect all components, which are connected to this column of the power rail section. For the other two circuits another Ground pin on the Raspberry is wired to the second section of the power rail. Therefore, all wires and components on this column of the power rail would be connected together.

The three circuits are built in the same way. In order to illustrate the connections between the Raspberry Pi and the hardware components on the breadboard, a circuit diagram is shown below:

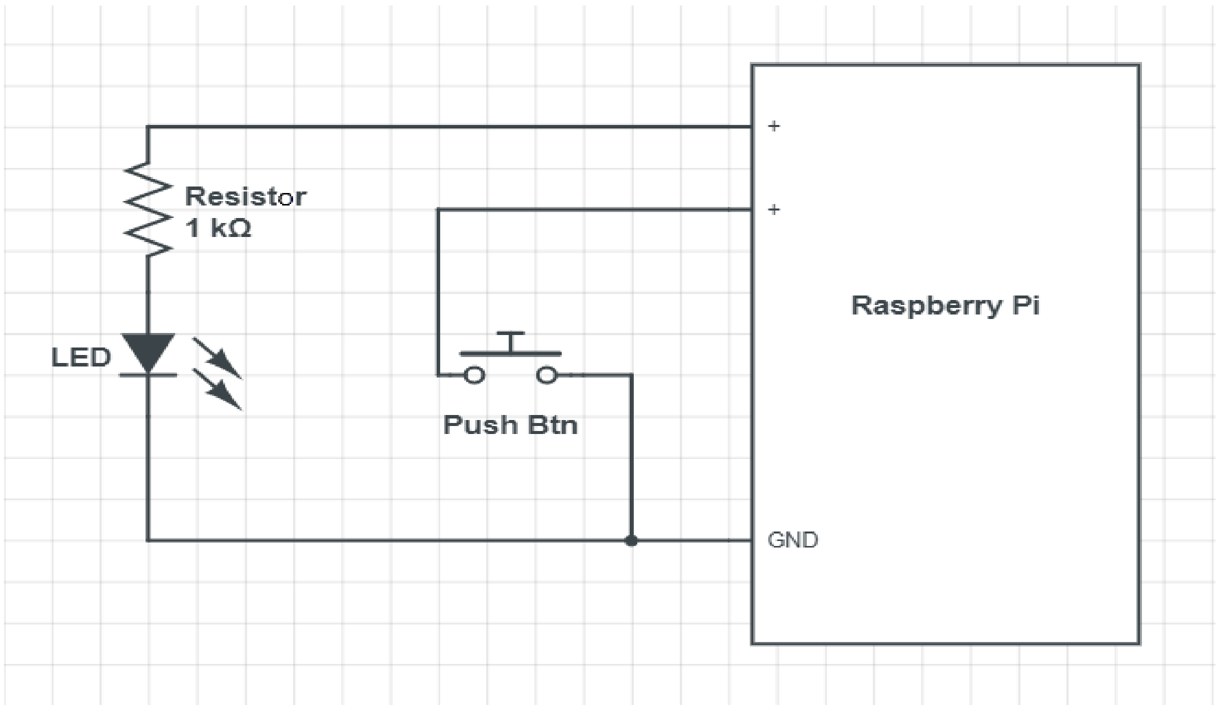


Figure 4. Circuit Diagram

The push button is connected to the Raspberry Pi, where one leg is linked to a GPIO pin and the other leg is attached to a Ground pin. The connection between the Raspberry Pi and LED light is built in a similar way. The negative leg of the LED light is wired to the Ground pin. However, a resistor is placed between the positive leg of the LED light and the GPIO pin.

Figure 4 shows how the circuit functions. The GPIO pin, linked to the button, is set as an input pin and the GPIO pin, wired to the LED light is an output pin. When the button is pressed down, an electric circuit is completed and a signal is sent to the input pin. The Raspberry Pi is programmed to wait for a button press detection from the input pin and influence the other GPIO pin connected to the LED light. Once a button press is identified, the GPIO pin linked to the LED light would output 3.3 volts to turn on the LED light. In this way, the LED light would be switched on while a button is pressed down and turned off when the button is released.

The lengths of the resistors and LED lights are adjusted using a side cutter. The sides of the resistors are reduced, which is making the layout looks cleaner. More importantly, it would not be as easy as before to remove resistors by mistake from the holes. The legs of the LED lights are cut in order to make them level with the buttons. This is a design step, which would be useful during the creation of a box. All components placed on the breadboard and connected to Raspberry Pi are shown in Figure 5 below:

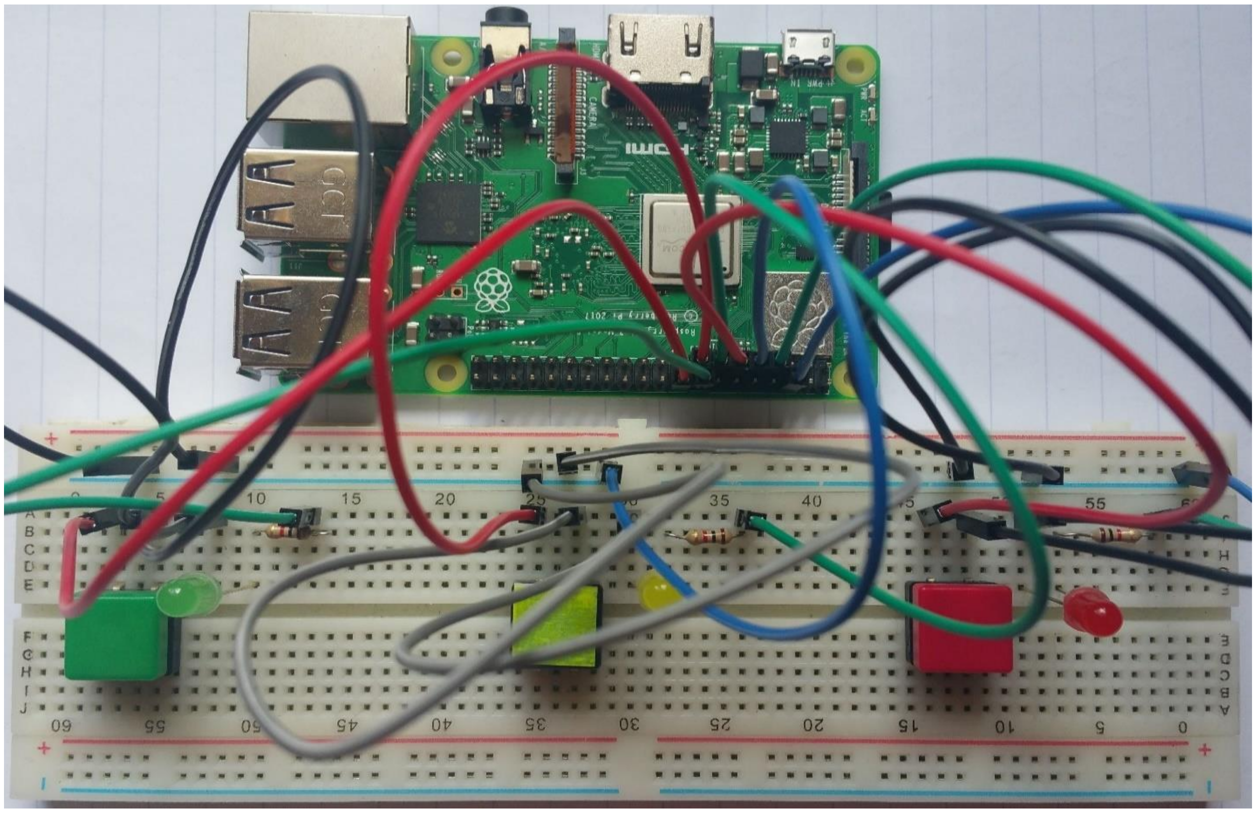


Figure 5. Student feedback system

Once all components are connected, they would be housed in a box, which would represent a stand-alone prototype of the student feedback system. The box would be created from cardboard, as it is easy to bend and manipulate. However, metal strips are added to reinforce the structure. Holes would be cut around the buttons and the LED lights. There would be labels below the buttons. At the back of the box, holes would be made to allow access to the micro USB port for power supply and other ports, which need to be available. Further space inside the box would be left to access the USB ports. After taking into consideration the functionality and hardware components of the feedback system, a box was designed. The box is sprayed with grey metallic paint as well. Its dimensions could be seen in Figure 6 below:

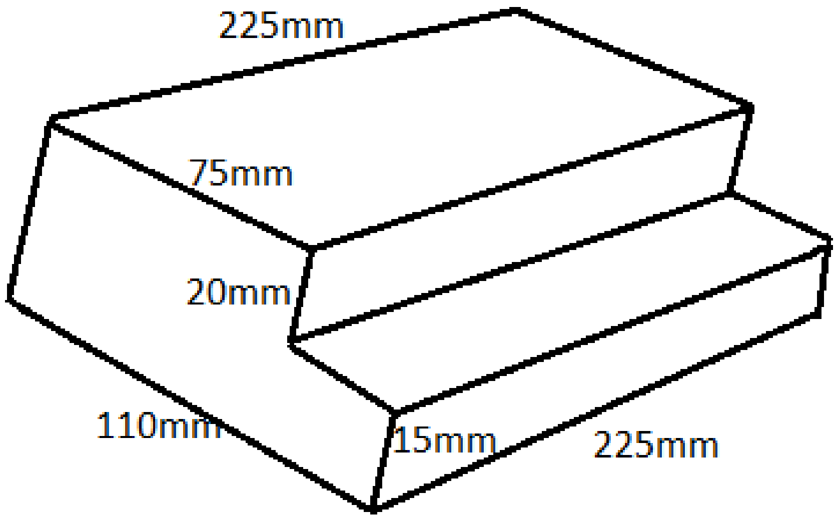


Figure 6. Box dimensions

The box, in which hardware components would reside, is made of cardboard as this is a prototype for presentation purposes only. Ideally, the box could be made of more stable materials such as hard plastic or even metal. A 3D printer could be used for production. This could be done at later stage if the student feedback system is to be implemented within the existing system and processes of collecting student feedback at the university. In addition, a production unit would replace the breadboard with a through-hole board which would allows to solder components on the board. The prototype of the student feedback system could be seen in Figure 7 below:

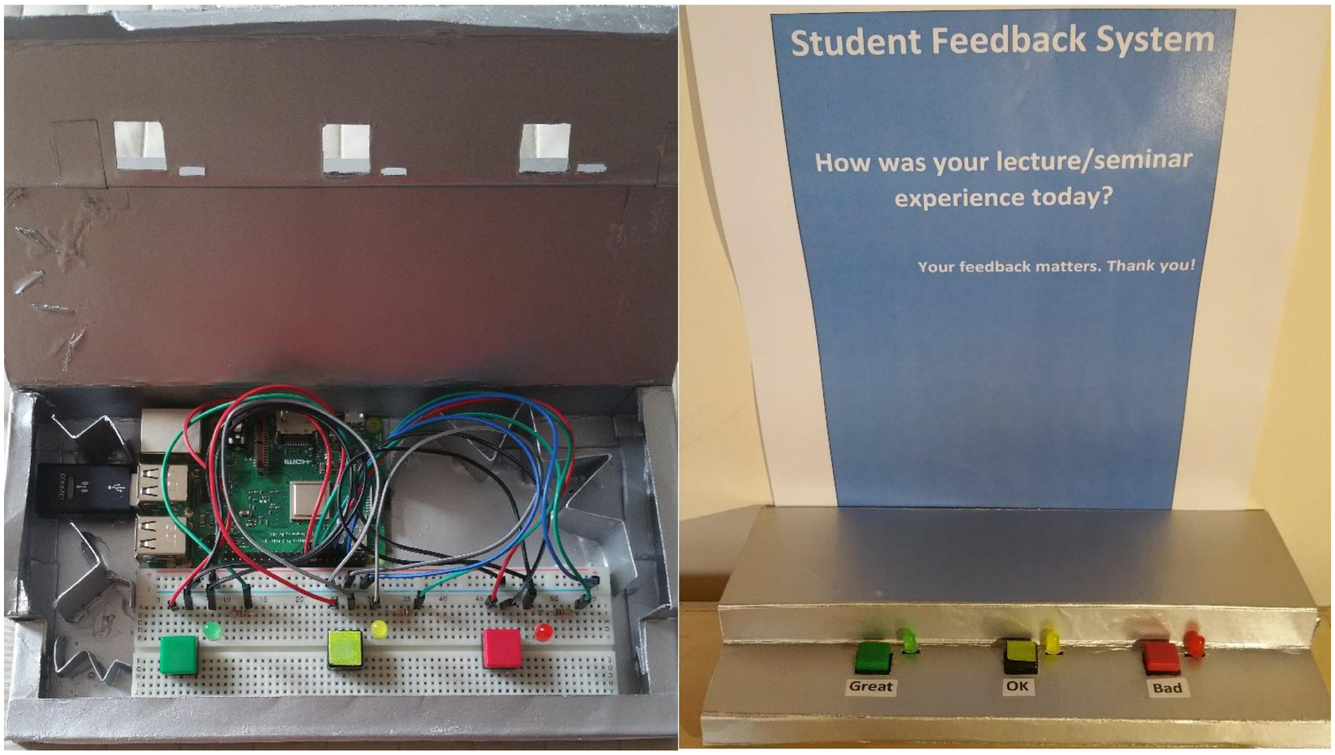

Student Feedback System
How was your lecture/seminar experience today?
Your feedback matters. Thank you!
Great
OK
Bad

Figure 7. Student feedback system with box opened (left) & closed (right)

User Interaction Design

There are fundamental design principles, which need to be considered. They are applied to the design of the student feedback system. According to Norman (2013), these design principles are discoverability, feedback, conceptual model, affordance, signifier, mapping and constraint.

Discoverability is concerned with the visibility of the functional parts of the feedback system. The functional part of the system are the buttons, which are easy to see.

Next principle to look into would be feedback. It gives information on the new state when an action has been executed. When one of the buttons is pressed, a LED light would glow red, green or yellow depending on which button is pressed. This gives an indication that the student feedback system is working and recording feedback.

Another principle is the conceptual model. It is the idea of how an object or system works. The feedback system is a device whose functions are to record and send student feedback to a server. The device in Figure 6 has only three buttons so the user would think this is the only way to operate the device. There are no other visible switches or buttons, which might affect user's conceptual model.

Affordance is another of Norman's design principles. It is related to the possible actions a user can take considering the object/system itself. The student feedback device has LED lights next to the buttons and labels suggesting button functions. Hence, a novice user could easily understand the purpose of the device is to collect feedback.

Signifiers communicate all possible actions to the user. In this case, the shape of the buttons are flat meaning they could only be pressed down.

Mapping principle shows the connection between the elements of a system or object. The student feedback system has a LED light next to each button. This mapping indicates those two parts are connected.

The last design principle is constraint, which is the opposite of affordance. Constraints limit the user and his/her possibilities to use an object or system. The feedback system has buttons and LED lights on one side and no other switches on the box. For this reason, a limited number of functions could be executed by the user.

Overall, by applying Norman's design principles to the student feedback system, a student, who has not previously used the device, should be able to easily figure out how it works.

Software

Software running on both the Raspberry Pi and the server would be created specifically for the needs of the student feedback system. Firstly, a server is created. It would serve two different types of clients. One, users who connect through a web application to get access to records on server and two, Raspberry Pi device(s) sending student feedback.

Initially, a research was conducted to find the most suitable environment for the project. It was decided to use Python 3 as a programming language for the software running on the Raspberry Pi as it has very clean, easy to learn and read syntax. In addition, Python is the preferred programming language by the Raspberry Pi Foundation and there are a large number of Python projects on its official website. More importantly, Python development environment, IDLE, comes preinstalled with the operating system. With regard to the server, Node.js was chosen. It is an open-source JavaScript runtime environment, which has been released initially about ten years ago. It runs on one process so there is no need to create a new thread for every request. Hence, resources are efficiently utilised. This could be useful if large number of clients send requests. In this case, the server would not run out of threads and make new clients wait until a thread is available. Node.js allows using JavaScript outside of a browser. This is one of its main advantages as it allows running the same programming language at the frontend and backend development (Node.js, 2019a).

Node.js uses Chrome V8 engine, which is able to run on any browser and it is system independent. It is written in C++ and continuously evolving to improve the performance of the Web and Node.js ecosystem. From 2009, V8 has a JIT (just-in-time) JavaScript compiler to make execution faster. At start, it might take some time, but once the process is done, the performance would be better than an interpreted code (Node.js, 2019b).

Node.js would be installed on Windows operating system. A Windows Installer was downloaded from www.nodejs.org website. The current and most stable version at the time of installation was 10.16.0 LTS. An important tool, which comes with Node.js installation, is npm (Node Package Manager). It is a command line client where developers could install and publish packages of code. Moreover, it is the world's largest registry of useful packages. This gives access to libraries with reusable code without reinventing the wheel. In this way, a collection of packaged modules would be available for the project (npm, 2019a).

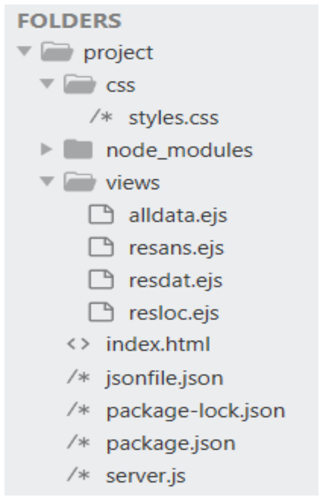
During the development process, additional modules were installed. They are required to create the server, which receives student feedback records and has other functions allowing users to connect via a browser. In addition, these modules were added to the list of dependencies in ***package-lock.json*** file, which would be explained further below. The following modules were used in the project:

- Express – it is a Node.js web application framework, which allows the development of web and mobile applications. Its main abilities are creating routes and responding to HTTP requests. It is possible to use Express with other frameworks as well. Its features would be used in the server-side part of the project (npm, 2019b).
- body-parser – this module is used to parse the body of requests. body-parse is used in combination with Express. It allows incoming requests to be formatted and transformed in a readable way. It supports URL encoded bodies JSON data, which would be useful for the needs of the project (npm, 2019c).

- fs – a module allowing synchronous and asynchronous access and interaction with the file system. Various methods within the module offer different file interactions (Node.js, 2019c).
- json-query – The module allows querying JSON objects. It is used in functions, which retrieve student feedback records within the .json file containing all the data (npm, 2019d).
- EJS – It is a templating language, which allows generating HTML with JavaScript (Ejs, 2019). It is used to pass JavaScript variables such as search results of student feedback records to the .ejs view pages in the *views* folder.

Implementation

The server and web application codes are in a folder named ***project***. The folder has other subfolders and files, which are shown in Figure 8 below:



```
FOLDERS
▼ project
  ▼ css
      /* styles.css
  ▶ node_modules
  ▼ views
      alldata.ejs
      resans.ejs
      resdat.ejs
      resloc.ejs
    <> index.html
    /* jsonfile.json
    /* package-lock.json
    /* package.json
    /* server.js
```

Figure 8. Project folders and files

The main file is ***server.js*** (Appendix 1). It starts the application when "*node server.js"* is typed in the Command Prompt. It contains all modules required to set up a configuration and run the web application, which listens to port 3030. Moreover, there are functions, which search through the student feedback records. In this way, users are able to search by date, feedback answer and location. The file defines routes to pages in the ***views*** folder. The ***views*** folder is required when using EJS view engine as it contains all .ejs files. These files are ***alldata.ejs***, ***resans.ejs***, ***resloc.ejs*** and ***resdat.ejs*** and their codes are shown in Appendixes 2, 3, 4 and 5 respectively. The file ***alldata.ejs*** shows all records of student feedback. ***resans.ejs***, ***resloc.ejs*** and ***resdat.ejs*** files show search results by feedback answers, locations and dates respectively. They are similar to HTML files but add EJS engine functionality for using variables from the server. The pages are accessible from the ***index.html*** file.

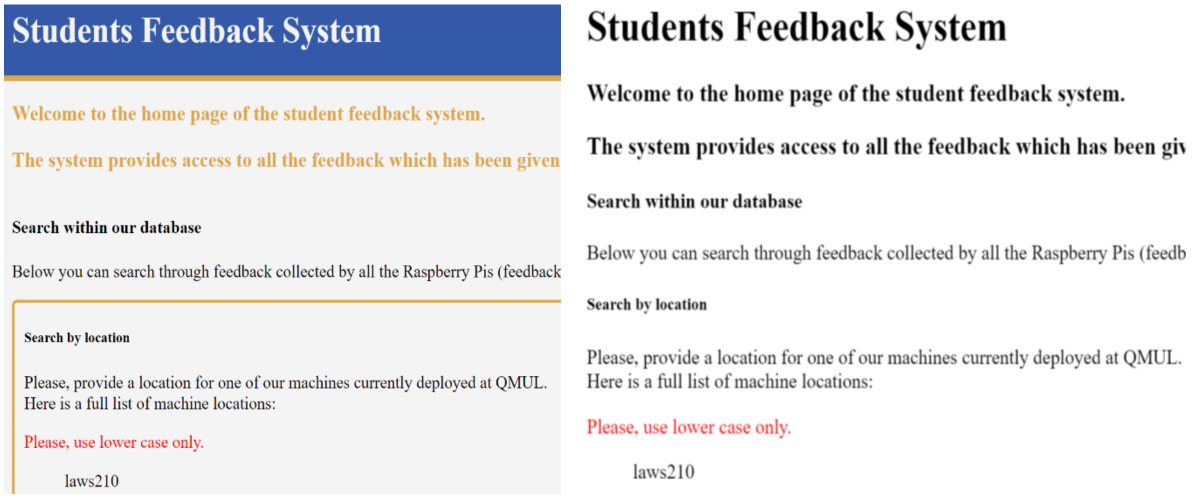
index.html is the home page of the student feedback system, which is accessible via a browser. The page introduces the student feedback system. In addition, it provides three search options and explains how to use them. Buttons link the page to files in the ***views*** folder. Appendix 6 shows the code of ***index.html***.

node_modules folder consists of modules, which could be used in the project. The folder is located in the current folder of the project so modules are installed locally. When packages are installed locally, a ***package.json*** file is created. It consists of project name, a description and other fields, which are shown in Appendix 7.

Another automatically generated file is ***package-lock.json***. It contains all dependencies for the project. The file monitors the versions of the packages installed and required by the project. This allows the project to be replicated exactly in the same way if ran on a different machine or packages are updated (Node, 2019d).

jsonfile.json contains all student feedback records sent by the Raspberry Pi device. Users are able to access the records on the file using the web application. The application allows the user to see all the records and search inside the file by location, date or feedback answer. Please, refer to Appendix 8, which shows file structure.

styles.css file could be found in ***css*** folder. CSS (Cascading Style Sheets) is used to improve the appearance and design of the pages accessible via the browser. The file contains styles, fonts, colours and layout of all pages. CSS code of ***styles.css*** file could be found in Appendix 9. Figure 9 below shows part of the home page of the student feedback system with and without CSS styling.


Students Feedback System

Welcome to the home page of the student feedback system.

The system provides access to all the feedback which has been given

Search within our database

Below you can search through feedback collected by all the Raspberry Pis (feedback

Search by location

Please, provide a location for one of our machines currently deployed at QMUL. Here is a full list of machine locations:

Please, use lower case only.

laws210

Students Feedback System

Welcome to the home page of the student feedback system.

The system provides access to all the feedback which has been giv

Search within our database

Below you can search through feedback collected by all the Raspberry Pis (feedb

Search by location

Please, provide a location for one of our machines currently deployed at QMUL. Here is a full list of machine locations:

Please, use lower case only.

laws210

Figure 9. Design - with (left) and without (right) CSS

Before programming any software on the Raspberry Pi, the operating system had to be downloaded and installed on the SD card. It is called Raspbian and can be ran on all Raspberry Pi models. A beginners' installer called NOOBS was downloaded as a Zip archive. Its files were extracted on the formatted SD card. The first time the operating system is booted, a welcome page appears and settings about time, password and Wi-Fi would need to be set up.

Once the operating system is installed, commands were run to ensure the system and its packages are up to date. One of the reasons to keep a device updated is security. Known vulnerabilities are fixed by an update to minimise risks of exploiting them. Another important reason for regular system updates is to solve areas where bugs occur. In this way, functionality is improved (Raspberrypi.org, 2019a).

Before starting the Python development environment, IDLE, the terminal emulator is used to install additional Python dependencies needed for the purpose of the project. The installation of additional Python packages is achieved by using pip. It is a package manager for Python, which comes with Raspbian by default. pip could install modules for Python 2, whereas pip3 is able to install modules for Python 3. Packages are installed from PyPi (the Python Package Index). PyPi is a repository of software, which is supported by the Python community. It consists of software packages developed for the Python programming language (Raspberrypi.org, 2019b). In order to achieve the desired functionality of the Raspberry Pi and connected hardware components, the following Python modules are used:

- RPi.GPIO – this module is included with Raspbian and controls the Raspberry Pi GPIO channels. It is required to provide control over the GPIO pins connected to the hardware components on the breadboard as buttons and LED lights.
- time – this module is responsible for different time-related functions. time.sleep() is the function, which is required from the module. It is able to stop an execution of a calling thread for specified period of time in seconds (Python, 2019a).

- csv – the module allows reading and writing data in CSV (Comma Separated Values) format. This functionality would be used by the Raspberry Pi to record student feedback in a .csv file.
- datetime – this module has features, which allow both dates and times manipulations. It is used to provide the exact time for each record of student feedback gathered (Python, 2019b).
- moment() – another module for formatting and manipulating dates. It would show the date of receiving student feedback.
- os.path – the module allows access to files on the operating system and the use of pathnames. For the purpose of this project, it would be used to check whether a file exists on the system (Python, 2019c).
- requests – This is one of the most downloaded modules. It allows the use of HTTP requests. It would be used in the project to send requests and pass parameters (Python-requests, 2019).

The software on the Raspberry Pi, allowing the student feedback system to function properly, is in ***mypi.py*** file (Appendix 10). The program is started on the terminal emulator by typing “python mypi.py”. When the program is running, the system is able to register student feedback, save the records on the Raspberry Pi and send them to the server as well. The program recognises which and when a button is pressed so a LED would light up to inform the user that the system is working and the feedback is registered. The file contains the above modules, variables for the buttons and LED lights connected to the GPIO pins and functions, which record student feedback depending on the button pressed. A record would contain a date, time and review answer, which are all saved on a .csv file on the Raspberry Pi. This file is called ***data.csv*** and could be seen in Appendix 11. Before saving a record, the program would check whether the .csv file exists. If the file is not present, the system would create one and save the record. Otherwise, the system would append the record to the existing file and its records.

When a student presses a button on the feedback system, a record is saved on the ***data.csv*** file. In addition, data flows to the server, using a HTTP GET request, where it is saved on ***jsonfile.json*** file. The server would send a response to the Raspberry Pi to acknowledge a record is received. A user is also able to access the server using the web application. The user sends HTTP GET request for the server to search the feedback records, based on parameters provided. The server would send the requested records to the user, which are rendered using the .ejs files described above. Therefore, data flows between the server and users as well.

Testing

The student feedback system was tested carefully to ensure software and hardware components function as intended. Tests were carried out at different stages of the project. If the system was malfunctioning or a fault was found, it was fixed before continuing to the next step of development. The main issues and their solutions during the project are described below.

A few problems occurred during the programing of the software on the Raspberry Pi. Initial testing of the software showed registering numerous records when holding a button pressed down for a longer period. This was fixed by using a “GPIO.add_event_detect” with “bouncetime”. Using event detection, the software recognises the rising or falling edges of the GPIO pins. The “bouncetime” parameter is able to block registering multiple button presses for a specified period in milliseconds.

Another issue occurred while saving the student feedback records on the .csv file. Every time the software on the Raspberry Pi was started, older records on the .csv file were deleted and only the recent session of feedback records was saved. The file was loaded with “with open()” function where the file name and other parameters are added in the brackets. It was found that an additional optional parameter, which sets the mode in which the file is loaded, could be added. Mode “a” for appending to end of the file was chosen to avoid erasing older data on file.

Additional issue relating to the .csv file was found. It was established that records were saved to the .csv file once the program was interrupted (Ctrl + C). However, this could be a problem if the Raspberry Pi loses power. It would cause data loss as output is internally buffered before being saved to the .csv file. Hence, a “flush()” command was added. It forces the system to write to the file every time new data entry is presented.

A problem concerned with saving student feedback records on the server side arose. It was found that the server would save only the first feedback record and the following records would be ignored. Moreover, the following connection error would appear on the Raspberry pi: “requests.exceptions.ConnectionError: ('Connection aborted.', BadStatusLine("''",))”. It was established that there was no response back to the Raspberry Pi once the student feedback record is received. This caused the device to keep on waiting and loading until it timed out. The solution included adding two lines of code: “res.status(200);” and “res.end();” to the end of the function, which adds the records on the server. In this way, with the first line of code, the server would send a response to the Raspberry Pi that records are successfully received (200, “OK”) and the second line of code would end the response session.

Discussion

The student feedback system performs according to its requirements set at the beginning of the project. Having researched other feedback systems and literature on student feedback in higher education, it could be stated that the system in this project has its advantages and disadvantages.

The system collecting the student feedback system would be accessible to all employees and students at university. In the world of business, data is received by the market research department, which transfers findings to individuals in other departments. Those individuals are making the decisions so customer feedback should be in their hands to ensure effective use of information (Reidenbach and Goeke, 2006). In the same way, providing information to students and members of staff could improve teaching quality, as they would be able to make informed decisions. Moreover, making data available to all interested parties would enhance its importance.

According to a report by Morgan et al. (2005), providing customer feedback more frequently would lead to more people within the business realising its vital role. Data dissemination should be vertical and horizontal where horizontal refers to information spread throughout the organisation, and vertical means distributing data across functional areas. Moreover, recipients should trust the information and its accuracy. If data recipients do not believe data is valid and reliable, they would not be likely to use it when making decisions. As a result, making student feedback available throughout the university would increase its importance and the possibility of it landing in the right hands to make effective decisions. Student feedback data should be considered accurate and reliable, as it is read-only. Only members of staff in the IT department have access to the server and student feedback data. Other members can see the records using the web application but cannot manipulate it in any way. Other options are to use the search by date, location and feedback answer functions in the web application. This could increase the validity and credibility of the records in recipients' eyes.

It is often discussed what time is best to collect student feedback. Most student feedback is gathered at the end of a module. At this point, it would be too late to take decisions, which could benefit the current year students attending the module. Subsequently, students could decide to ignore or complete module questionnaires without spending adequate time to produce useful information leading to future improvements. In addition, module questionnaires could be sent to students in the middle of the project. In this way, if necessary, actions could be taken to benefit students taking the module. However, it could be argued that giving students too many opportunities to provide feedback might be an overkill for both students and staff. The frequency in which questionnaires are administered, could introduce changes to their complexity. Simpler questionnaires should be presented to students, who are expected to provide more regular feedback. One of the strengths of the student feedback system presented in this paper is that student feedback records are immediately available for interested parties to see, analyse and base decisions upon them. Therefore, by providing immediate feedback, students would feel their feedback is valued and could make a difference in improving their experience (Brennan and Williams, 2004).

The student feedback system was completed and tested after the end of the semester. Hence, there were no lectures or other classes and the system was not able to collect real world data. Actual student feedback would have been valuable to see whether any trends could be found. Student feedback might show that particular teachers receive more positive or negative feedback and then investigate the reasons. Students' experience during lectures/seminars could be affected by other factors as well. For example, if found that negative feedback is given more often in a particular room, the room condition and equipment could be inspected.

The student feedback system does not relate student feedback gathered to teachers' timetables. Hence, trends cannot be shown immediately and the usefulness of records seems restricted at first, as further analyses and sorting need to be carried out. This could be seen as a limitation. However, teachers' timetables and identities should be kept confidential. Results of teacher performance would be of interest only to authorised individuals such as head of department and members of human resources. Once analysed, student feedback could serve different purposes. Teachers could be interested in changing their approaches depending on feedback. Head of departments might be interested in average results, which are compared between courses, programmes and faculties.

The student feedback system asks one question: "How was your lecture/seminar experience today?". However, it would be difficult to define the factors affecting students' experience based on a single question. As pointed above, numerous causes are possible such as room temperature, malfunctioning equipment, teaching quality, topic was not interesting, tired students, etc. Therefore, asking a more specific question would be more beneficial. For example, customer surveys in the business world have shown that it is important to ask the right questions in order to measure customer experience effectively. Typically, survey questions are created by a group of managers from different departments. Managers should work together to come up with questions, which are dedicated to a specific objective. In this way, businesses would be able to utilise customer feedback in a better way and use it in their future organisational improvement plans (Reidenbach and Goeke, 2006). In the same way, student feedback system could focus on a question related to a particular area of interest. Once results are obtained and analysed, an action could be taken. In addition, questions on the device(s) could be changed as often as required.

The size of feedback gathered by the student feedback system could be tracked as well. It could be investigated whether there was an initial increase in feedback and was it effectively used by teachers and management at the university. According to a study conducted by Lang and Kersting (2006), students provided feedback over a period of four semesters, where it was found that student feedback had significant positive effect on teaching in the short – term. Observations showed that there was an initial motivation in instructors, which increased positive student feedback ratings. As all teachers improved their teaching strategies, a loss of interest in taking part of the experiment was noticed. As a result, there was an increase in negative ratings by students. The research explains, at first, regular student feedback led to "work harder" attitude by teachers, which then changed to "work smarter" at the end of period observed. However, it has been noted that student feedback and ratings remain controversial and might not be the best way to measure teachers' effectiveness due to student bias based on grading, workload and gender.

The system allows students to provide feedback anonymously by simply pressing a button. This could be useful to students who have privacy concerns. According to a leading research publication, customers are becoming more concerned about their privacy, which is leading to a reduction in quality of feedback (Forbes, 2018). Privacy issues might have the same effect on student feedback. For example, often survey questionnaires are sent for every module to students' emails requesting feedback. Surveys might state reviews would not be disclosed to teachers, but students would still be concerned about their privacy. Students could believe providing negative feedback might put them in disadvantageous position thereafter. As a result, they might not be willing to provide an honest review or could decide not to leave feedback at all. However, the student feedback system gives them a chance to leave their honest reviews without the risk of exposing their confidentiality. In this way, student feedback would be more accurate.

Student feedback is a valued component of assessing teaching quality and university performance, even though it should not be the only tool for evaluation. Irrespective of how good a teaching method is, there is always a room for improvement. In order to make student feedback an import part of the evaluation process, students should be asked the right questions (Seldin, 1989).

Conclusion

Regardless of the limitations of the student feedback system, built around Raspberry Pi, it would be a valuable tool for measuring and improving quality of teaching at the university. It could gather student feedback effectively, store records in the device and server to ensure safe data storage without unauthorised access. The paper analysed the system, its uses, strengths and limitations. Evaluations based on its student feedback records, could be combined with other tools to assist the university in improving educational quality and students' overall experience. This report discussed the importance of student feedback and various ways to obtain student feedback. The system could support the university to improve its student feedback collection. In this way, the university shows students' opinions are appreciated and it is willing to hear about students' needs and experience. It could be concluded that the Raspberry Pi student feedback system could have beneficial effects on teaching quality.

Further work

The student feedback system presented in this report could be improved by carrying out further developments. As described above, the system is able to ask only one question. However, a touch screen could be connected to the Raspberry Pi. In this way, more features could be added to the system. For example, a whole survey containing various questions could be developed. In this way, more detailed answers would be gathered.

Currently, student feedback records are stored on a .json file. However, if the above improvements are undertaken, the system could require a SQL or NoSQL database with more complex relationships between records.

Additional features could also be added to the web application. For example, students could be allowed to provide feedback using the web application as well as the Raspberry Pi device. In addition, data visualisation, charts and graphs could be used to present records clearly and efficiently. Access to teachers' timetables could be allowed for authorised individuals. However, a login page would be required. Different types of reports could be created and sent by the system automatically on a regular basis to interested parties.

The above features are only examples of improvements, which could be introduced. Other functions and features could be added or removed depending on the requirements of the institution and its users.

References

Andrews, C. (2013). *Easy as Pi [Raspberry Pi]. Engineering & Technology*, 8(3), pp.34-37.

Ballantyne, C. (2004). *Online or on paper: An examination of the differences in response and respondents to a survey administered in two modes. Australasian Evaluation Society International Conference* 13-15

Barlow, J. and Moller, C. (1996). *Complaint Is a Gift: Using Customer Feedback As a Strategic Tool.* 2nd ed. Berrett-Koehler.

Brennan, J. and Williams, R. (2004). *Collecting and using student feedback - a guide to good practice | Higher Education Academy. [online]* Available at: https://www.heacademy.ac.uk/knowledge-hub/collecting-and-using-student-feedback-guide-good-practice [Accessed 12 Jul. 2019].

Bullock, C. (2003). *Online Collection of Midterm Student Feedback. New Directions for Teaching and Learning,* 2003(96), pp.95-102.

Cohen, P. (1980). *Effectiveness of student-rating feedback for improving college instruction: A meta-analysis of findings. Research in Higher Education*, 13(4), pp.321-341.

Eagle, L. and Brennan, R. (2007). *Are students customers? TQM and marketing perspectives. Quality Assurance in Education*, 15(1), pp.44-60.

Ejs. (2019). *EJS -- Embedded JavaScript templates.* [online] Available at: https://ejs.co/ [Accessed 22 Jul. 2019].

Fies, C. and Marshall, J. (2006). *Classroom Response Systems: A Review of the Literature. Journal of Science Education and Technology*, 15(1), pp.101-109.

Forbes (2018). *Council Post: The Death Of The Traditional Feedback Survey.* [online] Available at: https://www.forbes.com/sites/forbesagencycouncil/2018/10/11/the-death-of-the-traditional-feedback-survey/#65dc1e4e2403 [Accessed 27 Jul. 2019].

Fry, H., Ketteridge, S. and Marshall, S. (2010). *A handbook for teaching and learning in higher education.* New York: Routledge.

GitHub. (2015). *splitbrain/rpibplusleaf.* [online] Available at: https://github.com/splitbrain/rpibplusleaf [Accessed 18 Jul. 2019].

Lang, J. and Kersting, M. (2006). *Regular Feedback from Student Ratings of Instruction: Do College Teachers Improve their Ratings in the Long Run?. Instructional Science*, 35(3), pp.187-205.

Lieberman, D., Bowers, N. and Moore, D. (2001). *Use of Electronic Tools to Enhance Student Evaluation Feedback. New Directions for Teaching and Learning*, 2001(87), p.45.

Morgan, N. A., Anderson, E. W. and Mittal, V. (2005) '*Understanding Firms' Customer Satisfaction Information Usage', Journal of Marketing*, 69(3), pp. 131–151. doi: 10.1509/jmkg.69.3.131.66359.

Node.js. (2019a). *Introduction to Node.js*. [online] Available at: https://nodejs.dev/introduction-to-nodejs [Accessed 21 Jul. 2019].

Node.js. (2019b). *The V8 JavaScript Engine*. [online] Available at: https://nodejs.dev/the-v8-javascript-engine [Accessed 21 Jul. 2019].

Node.js. (2019c). *The Node.js fs module*. [online] Available at: https://nodejs.dev/the-nodejs-fs-module [Accessed 22 Jul. 2019].

Node.js. (2019d). *The package-lock.json file.* [online] Available at: https://nodejs.dev/the-package-lock-json-file [Accessed 21 Jul. 2019].

Norman, D. (2013). *The design of everyday things, revised and expanded edition.* New York: Basic Books.

npm. (2019a). *npm About*. [online] Available at: https://www.npmjs.com/about [Accessed 21 Jul. 2019].

npm. (2019b). *express*. [online] Available at: https://www.npmjs.com/package/express [Accessed 22 Jul. 2019].

npm. (2019c). *body-parser*. [online] Available at: https://www.npmjs.com/package/body-parser [Accessed 22 Jul. 2019].

npm. (2019d). *json-query*. [online] Available at: https://www.npmjs.com/package/json-query [Accessed 22 Jul. 2019].

Philbin, C. (2014). *Adventures in Raspberry Pi*. Chichester: Wiley.

Pitman, T. (2000). *Perceptions of Academics and Students as Customers: A survey of administrative staff in higher education. Journal of Higher Education Policy and Management*, 22(2), pp.165-175.

Python. (2019a). *time — Time access and conversions — Python 3.7.4 documentation.* [online] Available at: https://docs.python.org/3/library/time.html [Accessed 23 Jul. 2019].

Python. (2019b). *8.1. datetime — Basic date and time types — Python 2.7.16 documentation.* [online] Available at: https://docs.python.org/2/library/datetime.html [Accessed 23 Jul. 2019].

Python-requests. (2019). *Requests: HTTP for Humans™ — Requests 2.22.0 documentation.* [online] Available at: https://2.python-requests.org/en/master/ [Accessed 23 Jul. 2019].

Raspberrypi.org. (2019). *Installing Python packages - Raspberry Pi Documentation.* [online] Available at: https://www.raspberrypi.org/documentation/linux/software/python.md [Accessed 23 Jul. 2019].

Raspberrypi.org. (2019a). *Updating and upgrading Raspbian - Raspberry Pi Documentation.* [online] Available at: https://www.raspberrypi.org/documentation/raspbian/updating.md [Accessed 22 Jul. 2019].

Reidenbach, R. and Goeke, R. (2006). *Competing for customers and winning with value: Breakthrough Strategies for Market Dominance*. ASQ Quality Press.

Robson, D. and Johnson, M. (2008). *Clickers, Student Engagement and Performance in an Introductory Economics Course: a Cautionary Tale*. Computers in Higher Education Economics Review, 20.

Seldin, P. (1989). *Using student feedback to improve teaching. New Directions for Teaching and Learning*, 1989(37), pp.89-97.

Svensson, G. and Wood, G. (2007). *Are university students really customers? When illusion may lead to delusion for all! International Journal of Educational Management*, 21(1), pp.17-28.

Upton, E. and Halfacree, G. (2014). *Raspberry Pi User Guide*. 2nd ed. John Wiley & Sons Ltd.

Appendixes

Appendix 1 – *server.js*

```
File: C:\Users\pc\Desktop\project\server.js
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
var fs = require('fs');
var jsonQuery = require('json-query');
var moment = require('moment');
var conte;
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use('/css',express.static(__dirname+ '/css'));
app.set('view engine', 'ejs');


    //Date search function
function search_date(datee){
    var result = jsonQuery('feedback[*date='+datee+']', {data: conte}).value;
    return result;
}
    //Location search function
function search_location(locat){
    var result = jsonQuery('feedback[*location='+locat+']', {data: conte}).value;
    return result;
}
    //Answer search function
function search_answer(answ){
    var result = jsonQuery('feedback[*answer='+answ+']', {data: conte}).value;
    return result;
}

app.get('/', function(req, res) {
    res.sendFile(__dirname +"/index.html");
});

app.get('/alldata', function(req, res) {
    res.sendFile(__dirname +"/index.html");
    var file_cont = fs.readFileSync('./jsonfile.json');
    conte = JSON.parse(file_cont);
    var data = JSON.stringify(conte, null,2);
    res.render('alldata', {answer: conte});
});

app.get('/add', function(req, res) {
```

```
                // http://localhost:3030/add?location=laws210&answer=ok     Add a record
   var d = new Date();
   var date = moment().format('L');
   var time = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
   var location=req.query.location;
   var answer=req.query.answer;
   var file_cont = fs.readFileSync('./jsonfile.json');
   conte = JSON.parse(file_cont);
   conte.feedback.push({location:location, date:date, time:time,answer:answer});
   var data = JSON.stringify(conte, null,2);
   fs.writeFileSync('./jsonfile.json', data);
   res.status(200);
   res.end();
});

app.get('/resloc', function(req, res) {
                              // http://localhost:3030/resloc?location= search by location
   var file_cont = fs.readFileSync('./jsonfile.json');
   conte = JSON.parse(file_cont);
   var search = req.query.location;
   var result = search_location(search);
   var jsonstr = JSON.stringify(result, null,2);

   if (jsonstr.length>2) {
           res.setHeader('Content-Type', 'text/html');
           res.render('resloc', {answer: result});
   }else {
           res.status(404).send("Nothing was found");
   }
});

app.get('/resans', function(req, res) {
                                     // http://localhost:3030/resans?answer=   search by answer
   var file_cont = fs.readFileSync('./jsonfile.json');
   conte = JSON.parse(file_cont);
   var search = req.query.answer;
   var result = search_answer(search);
   var jsonstr = JSON.stringify(result, null,2);

   if (jsonstr.length>2) {
           res.render('resans', {ans: result});
   }else {
           res.status(404).send("Nothing was found");
   }
});

```

```
app.get('/resdat', function(req, res) {
                                            // http://localhost:3030/resdat?datetime= search by date
    var file_cont = fs.readFileSync('./jsonfile.json');
    conte = JSON.parse(file_cont);
    var search = req.query.date;

    var result = search_date(search);
    var jsonstr = JSON.stringify(result, null,2);

    if (jsonstr.length>2) {
                res.render('resdat', {dateans: result});
    }else {
                res.status(404).send("Nothing was found");
    }
});

app.listen(3030, function() {
    console.log('Server is runnung!');
 });

```

Appendix 2 – *alldata.ejs*

```
File: C:\Users\pc\Desktop\project\views\alldata.ejs
<!DOCTYPE html>
<html>
<head>
      <title>Download</title>
      <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="../css/styles.css">
</head>
<body>
  <header>
    <div class="container">
      <h1>Students Feedback System</h1>
    </div>
  </header>

  <div class="container">
      <h3><p>All Records</p></h3>
      <a href="/" class="button">Home</a><br>
  </div>

  <div class="container">
    <p>
     <% for(var i=0; i < answer.feedback.length; i++) { %>

     <p><%- answer.feedback[i].location %>,
      <%- answer.feedback[i].date %>,
      <%- answer.feedback[i].time %>,
      <%- answer.feedback[i].answer %></p>

      <% } %>
    </p>
  </div>

  <footer></footer>

</body>
</html>
```

Appendix 3 – *resans.ejs*

```
File: C:\Users\pc\Desktop\project\views\resans.ejs
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Search by Answer</title>
  <link rel="stylesheet" href="../css/styles.css">
</head>
<body>
      <header>
    <div class="container">
      <h1>Students Feedback System</h1>
    </div>
  </header>

  <div class="container">
    <h3>Search by Answer</h3>
    <a href="/" class="button">Home</a><br>
  </div>
  <br>

  <div class="container">
    <table border="1">
      <tr>
        <td>Location</td>
        <td>Date</td>
        <td>Time</td>
        <td>Answer</td>
      </tr>
      <% for(var i=0; i < ans.length; i++) { %>
      <tr>
        <td><%- ans[i].location %></td>
        <td><%- ans[i].date %></td>
        <td><%- ans[i].time %></td>
        <td><%- ans[i].answer %></td>
      </tr>
      <% } %>
    </table>
  </div>

<br>
<footer></footer>
</body>
</html>
```

Appendix 4 – *resloc.ejs*

```
File: C:\Users\pc\Desktop\project\views\resloc.ejs
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Search by Location</title>
  <link rel="stylesheet" href="../css/styles.css">
</head>
<body>

  <header>
    <div class="container">
      <h1>Students Feedback System</h1>
    </div>
  </header>

  <div class="container">
   <h3>Search by Location</h3>
   <a href="/" class="button">Home</a><br>
   </div>
   <br>

   <div class="container">
   <table border="1">
    <tr>
      <td>Location</td>
      <td>Date</td>
      <td>Time</td>
      <td>Answer</td>
    </tr>
    <% for(var i=0; i < answer.length; i++) { %>
    <tr>
     <td><%- answer[i].location %></td>
     <td><%- answer[i].date %></td>
     <td><%- answer[i].time %></td>
     <td><%- answer[i].answer %></td>
    </tr>
    <% } %>
   </table>
  </div>
  <br>
  <footer></footer>
</body>
</html>
```

Appendix 5 – *resdat.ejs*

```
File: C:\Users\pc\Desktop\project\views\resdat.ejs
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Search by Date</title>
  <link rel="stylesheet" href="../css/styles.css">
</head>
<body>
      <header>
    <div class="container">
      <h1>Students Feedback System</h1>
    </div>
  </header>

  <div class="container">
    <h3>Search by Date</h3>
    <a href="/" class="button">Home</a><br>
  </div>
  <br>

  <div class="container">
    <table border="1">
      <tr>
        <td>Location</td>
        <td>Date</td>
        <td>Time</td>
        <td>Answer</td>
      </tr>
      <% for(var i=0; i < dateans.length; i++) { %>
      <tr>
       <td><%- dateans[i].location %></td>
       <td><%- dateans[i].date %></td>
       <td><%- dateans[i].time %></td>
       <td><%- dateans[i].answer %></td>
     </tr>
     <% } %>
   </table>
  </div>

  <br>
  <footer></footer>
</body>
</html>
```

Appendix 6 – *index.html*

```
File: C:\Users\pc\Desktop\project\index.html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Home Page</title>
  <link rel="stylesheet" href="./css/styles.css">
</head>
<body>
  <header>
    <div class="container">
      <h1>Students Feedback System</h1>
    </div>
  </header>
  <section id="intro">
    <div class="container">
      <h3><p>Welcome to the home page of the student feedback system.</p> <p>The system provides access to all the feedback which has been given by QMUL students using our Raspberry Pi devices deployed around university premises.</p>
      </h3>
    </div>
 </section>
  <div class="container">
    <h4>Search within our database</h4>
    <p>Below you can search through feedback collected by all the Raspberry Pis (feedback machines) by location, feedback answer and date. </p>
  </div>
  <div class="container">
    <section class="boxes">
      <h5>Search by location</h5>
      <p>Please, provide a location for one of our machines currently deployed at QMUL. <br> Here is a full list of machine locations:</p> <p style="color:red;">Please, use lower case only.</p>
      <ul>laws210</ul>
      <ul>bancroft115</ul>
      <ul>itl01</ul> <br>
      <form action="resloc" method = "get">
        Location:<br>
        <input type="text" name="location" placeholder="Enter location here">
        <input type="submit" class="button" value="Search">
      </form>
    </section>
  </div>
  <br>
  <div class="container">
    <section class="boxes">
      <h5>Search by answer</h5>
      <p>In this section you can pick a feedback option and search by it.</p>
      <p>Please, select one of the below option and click on the "Search" button.</p><br>
      <form action="resans" method="get">
        <input type="radio" name="answer" value="bad"> Bad<br>
        <input type="radio" name="answer" value="ok"> OK<br>
        <input type="radio" name="answer" value="great"> Great<br>
        <br>
        <input type="submit" class="button" value="Search">
      </form>
    </section>
  </div>
  <br>
  <div class="container">
    <section class="boxes">
      <h5>Search by date</h5>
      <p>A search by date is available below as well.</p>
      <p style="color:red;">Please, enter dates below in the following format: mm/dd/yyyy, e.g. "06/26/2016".</p>
      <form action="resdat" method = "get">
        Date:<br>
        <input type="text" name="date" placeholder="Enter date here">
        <input type="submit" class="button" value="Search">
      </form>
    </section>
  </div>
  <br>
  <div class="container">
    <section class="boxes">
      <p><b>If you need access to all our of records collected from all machines at QMUL, please use the button below:</b></p>
      <button onclick="doDownload()" class="button">Get Data</button>
      <script>
        function doDownload() {
          window.open("/alldata");
        }
      </script>
    </section>
  </div>
  <br>
  <footer></footer>
</body>
</html>
```

Appendix 7 – *package.json*

```
File: C:\Users\pc\Desktop\project\package.json
{
  "name": "project",
  "version": "1.0.0",
  "main": "server.js",
  "dependencies": {
    "chart.js": "^2.8.0",
    "express": "^4.17.1",
    "ejs": "^2.6.2",
    "json-query": "^2.2.2",
    "jquery": "^3.4.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}

```

Appendix 8 – *jsonfile.json*

File: C:\Users\pc\Desktop\project\jsonfile.json

```
{
  "feedback": [
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:26",
      "answer": "ok"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:27",
      "answer": "ok"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:31",
      "answer": "bad"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:33",
      "answer": "great"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:36",
      "answer": "ok"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:47",
      "answer": "great"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:52",
      "answer": "bad"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:3:54",
      "answer": "ok"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:4:1",
      "answer": "ok"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:4:6",
      "answer": "great"
    },
    {
      "location": "itl01",
      "date": "07/29/2019",
      "time": "15:4:8",
      "answer": "ok"
    }
  ]
}
```

Appendix 9 – *styles.css*

```
File: C:\Users\pc\Desktop\project\css\styles.css
body{
	background: #f4f4f4;
	font: serif;
 }

.container{
  width:85%;
  margin:auto;
  overflow:hidden;
}

 header{
	background: #3459A9;
	font-family: serif;
	color: #f4f4f4;
	min-height:20px;
	border-bottom:#E0A642 5px solid;
}

header .container h1{
	color: #f4f4f4;
}

#intro{
	color: #E0A642;
}

.boxes{
	border-style: solid;
	border-color: #E0A642;
	border-width: 2px;
	border-radius: 5px;
	padding: 10px;
	padding-top: 1pt;
}
footer{
	background: #3459A9;
	color: #f4f4f4;
	min-height:50px;
	border-top:#E0A642 5px solid;
	position: relative;
}

.button{
    border: none;
    color: white;
    background-color:#3459A9;
    font-size: 14px;
    padding: 2px 5px;
    border-radius: 3px;
}
```

Appendix 10 – *mypi.py*

```
#mypi.py
import RPi.GPIO as GPIO
from time import sleep
import csv
import datetime
import os.path
import requests

GPIO.setmode(GPIO.BCM)
sleepTime = .1
#GPIO of BAD Button and LED
lightPin1 = 4
buttonPin1 = 17
GPIO.setup(lightPin1, GPIO.OUT)
GPIO.setup(buttonPin1, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.output(lightPin1, False)
#GPIO of OK Button and LED
lightPin2 = 27
buttonPin2 = 22
GPIO.setup(lightPin2, GPIO.OUT)
GPIO.setup(buttonPin2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.output(lightPin2, False)
#GPIO of GREAT Button and LED
lightPin3 = 23
buttonPin3 = 24
GPIO.setup(lightPin3, GPIO.OUT)
GPIO.setup(buttonPin3, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.output(lightPin3, False)


def badBtn(a):
	button_time = datetime.datetime.now()
	data.writerow([button_time,"1","0","0"])
	csv_file.flush()
	print(button_time, "Bad")
	sending1 = requests.get("http://192.168.1.5:3030/add?location=itl01&answer=bad")

def okBtn(a):
	button_time = datetime.datetime.now()
	data.writerow([button_time,"0","1","0"])
	csv_file.flush()
	print(button_time, "Ok")
	sending2 = requests.get("http://192.168.1.5:3030/add?location=itl01&answer=ok")

def greatBtn(a):
      button_time = datetime.datetime.now()
      data.writerow([button_time,"0","0","1"])
      csv_file.flush()
      print(button_time, "Great")
      sending3 = requests.get("http://192.168.1.5:3030/add?location=itl01&answer=great")

# check if file data.csv exists
file_exists = os.path.exists("/home/pi/Desktop/data.csv")
with open('data.csv','a') as csv_file:
      data = csv.writer(csv_file)
      headers = ['Time', 'Bad','Ok','Great']
      writer = csv.DictWriter(csv_file, delimiter=',', lineterminator='\n', fieldnames = headers)

      # if True, do not put headings again as they were already added at file creation
      if not file_exists:
            data.writerow(['Time', 'Bad','Ok','Great'])

    # Now loop creating new entry on the csv file every time a button is pressed
      try:
          GPIO.add_event_detect(17, GPIO.RISING, callback = badBtn, bouncetime= 400)
          GPIO.add_event_detect(22, GPIO.RISING, callback = okBtn, bouncetime= 400)
          GPIO.add_event_detect(24, GPIO.RISING, callback = greatBtn, bouncetime= 400)
          while True:
            GPIO.output(lightPin1, not GPIO.input(buttonPin1))
            GPIO.output(lightPin2, not GPIO.input(buttonPin2))
            GPIO.output(lightPin3, not GPIO.input(buttonPin3))
            sleep(.1)

            input_bad = 0
            input_ok = 0
            input_great = 0
            if input_bad == True:
                  GPIO.output (17, GPIO.LOW)
            elif input_ok == True:
                  GPIO.output(22, GPIO.LOW)
            elif input_great == True:
                  GPIO.output(24, GPIO.LOW)

      finally:
        GPIO.output(lightPin1, False)
        GPIO.output(lightPin2, False)
        GPIO.output(lightPin3, False)
        GPIO.cleanup()
```

Appendix 11 – *data.csv*

Time	Bad	Ok	Great
29-07-19 15:03:26	0	1	0
29-07-19 15:03:28	0	1	0
29-07-19 15:03:32	1	0	0
29-07-19 15:03:34	0	0	1
29-07-19 15:03:37	0	1	0
29-07-19 15:03:48	0	0	1
29-07-19 15:03:53	1	0	0
29-07-19 15:03:55	0	1	0
29-07-19 15:04:02	0	1	0
29-07-19 15:04:06	0	0	1
29-07-19 15:04:09	0	1	0
29-07-19 15:04:17	0	1	0