Offline collaborative learning approach for remote Northern territory students

Dai, Haixiao; Nguyen, Phong Lam; Kutay, Cat

Published in:
Interactive Technology and Smart Education

DOI:
https://doi.org/10.1108/ITSE-05-2022-0063

E-pub ahead of print: 09/09/2022

Document Version
Peer reviewed version

Citation for published version (APA):
Offline Collaborative Learning Approach for Remote Northern Territory Students

A Learning Box has been build based on minicomputer and a web learning management system. We present different options to create such a system and discuss various approaches for data syncing. The structure of our final setup is a Moodle LMS on a Raspberry Pi which provides a WiFi hotspot. We worked with lecturers from X University who work in remote Northern Territory regions to test this and provide feedback. We also considered suitable data collection and techniques that can be used to analyse the available data to support learning analysis by the staff. This research focuses on building an asynchronous hardware and software system that can store data on a local device until able to share. It was developed for staff and students at university who are using the limited internet access in areas such as remote Northern Territory. This system can asynchronously link the users’ devices and the central server at the university using unstable internet. Digital learning systems are crucial for education and data collected can analyse students learning performances to improve support. The resultant system has been tested in various scenarios to ensure it is robust to ensure students submissions are collected. Furthermore issues around student familiarity and ability to use online systems has been considered due to early feedback.

**Implications for research:**
- Monitoring asynchronous collaborative learning systems through analytics can assist students learning in their own time.
- Learning Hub can be easily set up and maintained using micro-computers now available
- Phone interface is sufficient for learning when supporting video and audio submissions

**Implications for practice or policy:**
- Digital learning can be implemented in an offline environment by using a Raspberry Pi as LMS server.
- Offline collaborative learning in remote communities can be achieved by applying asynchronized data syncing techniques.
- Asynchronized data syncing can be achieved by using change logs and incremental syncing technique.

*Keywords: Learning System, Data Science, Analytics, Collaborative Learning, Offline-First, Asynchronous Syncing*

**Introduction**

In regional Northern Territory, only 58.7% households reported they have internet access in 2016, which was far below 78.8% Australian average (Information Decisions, 2016). That means, internet connection is a major problem for people there and this affects equality of accessing learning resources for tertiary students in regional Northern Territory. Those who can access the internet in remote area do so through: satellite, connections which are vulnerable to bad weather; sometimes microwave links but these cannot reach long distance without repeaters; and very rarely, cables installed in the area, such as for mining companies.
Then they need to install and pay for a wifi hotspot in their home or community centre for educational use when studying. Their ability to make regular payments also may be irregular due to low income. Hence even those with access will find it intermittent or need to travel to link in to an existing system.

Tertiary educational organisations can adapt existing learning management system to remote users. With such a system, students can access learning materials, submit assignments, and collaborate with others and education providers can share multimedia learning materials. Traditionally learning management systems focus on online collaborate learning. Even with the use of progressive web apps, the unit contents need to be downloaded by the students before moving offline, and then collaboration only occurs when they have internet access. In an offline environment there can be synchronization issues between students' learning data and the lecturer's course information unless this is developed with the context in mind. This becomes a challenge for education organisations delivering courses in remote regions where internet resource is limited.

The purpose of this study is to build a collaborative learning system that can work within an offline environment and share data asynchronously with other devices when it can access the internet. The system uses an open-source digital learning software called Moodle (Modular Object Oriented Developmental Learning Environment), and a server hardware that was built based on Raspberry Pi. This study also investigated different techniques to synchronize data to manage the database of the learning management system asynchronously in an environment that internet is only accessible in a specific location at specific time. Thus, students in remote Northern Territory regions can store their submissions and record of their work on the device offline and share their information and collaborative with other local students and with lecturers when they become online.

**Background**

Moodle is an open-source e-learning platform that allows educators, administrators, and students to build customised learning environments using a single, robust, stable, and integrated framework (Olama et al. 2014). By installing the entire Moodle system on portable computing units such as Raspberry Pi or mobile phones, students and lecturers can access the learning platform anywhere, anytime. However, previously learning management system with Raspberry Pi have only been used in a teaching environment where students are co-located with staff. For other work outside of the classroom, such as preparing learning materials and grading assignments, lecturers require an environment with some internet access to develop material.

There are various challenges to designing such a system, having to consider all user scenarios and complications that can arise. Hence this project used an iterative development mode, and many options were investigated. For instance, Moodle provides a mobile app that works asynchronously however cannot be used until each part of the unit has been opened online. Since we cannot rely on students having the same phone for the duration of study, this would not work. This led to the idea of having the entire LMS on a Raspberry Pi server and use a hotspot for any students in the local area to share. However, whether using a service worker or other method, the synchronisation of data is not a trivial task (Balakumar & Sakhidevi 2012). As this was one of the major hurdles in this project, we consider here what the issues are in this area.
**Synchronisation**

For synchronisation the challenge is handling database conflicts between databases in different devices, such as the various students, and the lecturer. In addition, there are four points that need to be considered for synchronisation: minimise storage; minimize the amount of data communicated during synchronisation between a server and a client; an independent server to manage desynchronization; and have appropriate techniques to handle conflicts. Aiming to solve these problems, Kim (2006) proposed a database synchronization framework for small databases embedded in mobile devices. In his framework, he added two fields to each table in one the database: timestamp field to record the time a record change happened, and operation field to record what operation was executed on that record, which should be one of delete, update or insert. When a database is about to synchronize with another database, the server will export all changes to a list and send the list to another server, the receiver server will then compare each timestamp in its database with incoming list and update data accordingly, using the operation field. Kim also built a workflow to detect conflicts and divided conflicts into three groups: insertion conflict, deletion conflict and update conflict. However, Kim’s framework was mainly focused on the software level, it lacked details about the hardware level, which was the one of the reasons of our research that to investigate how the data syncing framework can be embedded to hardware to provide a seamless experience for user, as well as the power and resources usage of the embedded system that will be the best suited for the unique environment in the Northern Territory. In addition, Kim’s framework requires each mobiles devices to be configured to enable the syncing framework, which is not ideal giving the limited internet and various device types in the context. The embedded system we developed will only require one configuration for each system (Raspberry Pi), and the system will allow multiple students to connect at the same time regardless the types of their devices, without configuring every student’s device.

Another database synchronization framework is the Synchronization Algorithms based on Message Digest (SAMD) which resolves problems in database syncing between the central server and mobile devices. This “Message digest consists of a unidirectional hash function that maps a message of a random length to a fixed-length hash value.” The Synchronization Algorithms based on Message Digest framework has the following steps:

**Step 1:** Synchronize the Mobile Client Data Table (MCDT) with Mobile Client Message Digest Table (MCMCT) that is stored in the server.

**Step 2:** Synchronize Data Server Data Table (DSDT) with Database Server Message Digest Table (DSMDT) that is stored in the server. In this step, only records that have same ids with the requested mobile device in DSMDT will be selected to synchronise with DSDT to reduce unnecessary synchronizations and therefore reduce resource usage and increase speed of synchronization.

**Step 3:** Compare the MCMCT with the DSMDT to make decisions on synchronizations. If the MVD value of a record in MCDT is identical with the MDV value the DSMDT then there is no need to synchronize this record. While if the MDV values of a record in MCDT is different from the MDV value in DSMDT, a value field from that record in MCDT will be set to indicate that this record has been modified and needs to be synchronized.

**Step 4:** Filter out all records in the MCMCT and the DSMDT that have modified value set and execute synchronization for these values. Once synchronization finish, reset the modified field. (Alhaj et al., 2013; Choi et al., 2010; Domingos et al., 2014; Faiz & Shanker, 2016; Kalyanakumar & Sangeetha 2015; Sathya & Ayyapan, 2014; Singh & Hasan 2019)

The SAMD syncing technique has been widely investigated and implemented in recent research, however we consider this syncing is too heavy and it is resource intensive (Kekgathetse, M & Letsholo, K 2016). Hence, we want to minimise the size of the system and resources needed in the system due to limited internet connection and computing resources on Raspberry Pi, SAMD will not be ideal in this case. The system developed in this research is less resource intensive than SAMD since devices will be communicating with Restful APIs and uses less internet resources since it only transfers a JSON file, not database tables.
Another technology to build offline apps and sync data is Progressive Web Apps (PWA). PWA implements a set of APIs called Service Workers, which allow progressive web app to perform background tasks, including cache and preload assets of the app when the device is connected to the internet therefore enable offline compatibility of the app and makes the app work offline (Tandel & Jamadar 2018; Roumeliotis & Tselikas 2022). When running offline, progressive app can store user data and changes in database on local machine with indexed database, when device becomes online, the app will send locally cached data to the server. If change was made for the same record on different devices, the app will always choose the change that was made newer to overwrite the change that was made older. This is what we want in our system which will resolve conflicts during data syncing process. However, Moodle system is a huge system that has more than 500MB of assets and resources. If we turn Moodle into a progressive web app, users in remote areas will need to preload and cache 500MB of data on each device through internet to be able to run the system. As there are very limited internet resources in remote Northern Territory regions, we want to minimise the data transferred through internet in remote regions, it is not the best practice for us to turn Moodle into a progressive web app. In this case, we want to find other solutions that allow us pre-install the system on each device before we send the system to students in remote regions and only sync changed data instead of the entire system.

The final alternative method we found uses an audit log (Gudakesa et al., 2014). Instead of using message digest to record database changes, an audit log can be used to store database changes without hash function. Audit log can be created by adding three triggers to each table in the MySQL database for Insert, delete and update operation respectively, then triggers will detect every such query that is executed in the database and record the change that the query made to the database and the query itself. In audit log synchronization framework, all databases must have similar structure in the initial stage before synchronization. Once synchronization starts, the one server will send the audit log of its database to another server. And then the server will compare timestamp of each record in the audit log from sender with the current time on the server. If the audit log is newer, then execute SQL query to apply changes according to audit log. Although this framework can be used in asynchronous database synchronization, however, this framework doesn’t involve conflicts detecting and resolving which is crucial in asynchronous database synchronization. The syncing framework developed in this research is built based on the idea of having audit log, with a touch to resolve conflicts during synchronization. It has the advantage of audit log that is light weight, requires less resources and minimum changes to the database, as well as it’s highly tailored for the Northern Territory environment and have conflicts resolving techniques built in.

**Approach to problem**

**Research method**

In this project, we use Design-Based Implementation (DBIR) which is a systematic approach that requires researchers work with stakeholders closely to design research and investigate the processes of implementation iteratively, in order to find solutions of important matters in the local context (Martin et al., 2019). In DBIR researchers find appropriate research method for different stages and circumstances in the research lifecycle. Originated from Design-Based Research, DBIR focuses on how to design the process of implementation that addresses multiple challenges within a complex system, which is different from the focus of Design-Based Research which is to explore how a specific intervention works to address a particular issue. DBIR enables possibilities for multiple perspectives, as practitioners and researchers jointly negotiate research agendas and elaborate local, practical theories of action, by bringing stakeholders into the research process to make sure the usability of the research innovations in real business cases (LeMahieu et al., 2017).
We use three stages in each iteration cycle. The first stage of the cycle is design. In this stage, we held meetings with the main stakeholder to gather requirements, discuss issues that need to be resolved during this cycle, then chose appropriate techniques to conduct experiments as to who this might be solved. The last stage of the cycle is to test the final implementation before handing over the product for use each cycle. In this stage, we gather results from our experiments, and make relevant conclusions which are then discussed with our stakeholders, so that we can have better understanding of the issue and improve in the next cycle. With the DBIR approach, we conduct different experiments with different methods in different environments and find the best solution in the relevant environment.

Study population and background

Four academic staff at X University who work with remote students agreed to participate this research and implement this system with their work at the beginning of this research. However, as our research progressed, one participant withdrew from the research as they were hesitant to engage given a perception of possible poor influence of new technology on remote students and learning outcomes. The second participant is a casual staff at X University, and we have had irregular contact with this participant due to their other work commitments.

From our research into exiting systems in the domain and discussions with the initial four lecturers, we developed the following user stories to describe the desired system that was to be developed. From our research such a system does not exist but there are components from other systems that can be combined to enable us to provide these:

- User can improve learning by sharing their material with other students in their community
- User is regularly out of internet range but still wants to be able to submit and access material
- Both the new system and the existing CDU system should be intelligible to the user
- User can upload submissions as audio and video as well as text
- User will mostly be on mobile devices

The third and fourth participant have tested the developed system and provided helpful feedback, extending the user stories to be considered in the last section on improvements.

System Components

Internet emulation

The aim is to provide an emulation of the online LMS through a synchronising offline system. In the designed system, a Raspberry Pi broadcasts a WIFI signals that allow remote students to connect to the Raspberry Pi so they can access the learning system that is installed there. At the same time the Raspberry Pi needs to access the internet if available, so the only realistic option is the USB mobile dongle for data sharing, because it is difficult to set up WIFI hotspot and connect to a WIFI at the same time on a Raspberry Pi, as there is only one WIFI hardware module on Raspberry Pi. Both hotspot and WIFI connection require a WIFI module respectively and they cannot function on the same hardware. This led us to use a USB mobile dongle option. In this study, we tested Telstra 4GX USB modem, which is a USB mobile dongle, with Telstra 4G mobile plan.
System architecture

All the learning and teaching actions will be performed by Moodle on the Raspberry Pi exactly as on the normal online LMS platform from university. Whenever the Learning Box is powered on, it checks for the Internet connection to the cloud server which is accessible by training provider (see Figure 1). The database syncing will be automatically performed to push the student submissions from local Raspberry Pi to cloud server and pull all the changes in courses or learning materials that have been made by lecturer on online server platform from cloud server to local Learning Box. In the background, students’ data will be collected such as access time, learning duration, attendance, etc to analysis the users’ behavior and improve effectiveness of the lecturer’s support of students.

In terms of communication our final system has three major components: Moodle learning management system, Raspberry Pi remote server and the central university server (See Figure 2).
The Learning Box is the client part of the system that will be used by students and lecturers when they run workshops in remote regions. Each device (Raspberry Pi) runs an Ubuntu server as operating system. A Moodle learning management system will be installed and deployed on the Learning Box. This forms the client part of our system.

The Raspberry Pi device will create a local network with a WIFI hotspot, users can use their own devices such as mobile phones or tablets to connect to the Raspberry Pi with this WIFI hotspot. Once users connect to the Raspberry Pi, they will be able to use the learning system by entering the pre-defined URL of the learning system using web browsers. Figure 3 below shows the complete design of the internal system of client.

![Figure 3 Interior of the learning box](image)

Central Server

The Central Server at the university is a physical server that running on Ubuntu system, it will process data that received from clients, then send processed data back to clients. This will be accessed by on-campus lecturers. The central server controls the data syncing processes, as well as broadcasting updates to the software system for all clients. When a client wants to sync data, the client will connect to the central server with a RESTful API. Once the central server receives a request from a client, the central server will then execute relevant process based on the type of the request.

Once all syncing processes finish, the server embeds results to a response in JSON format and sends the response back to client. Backend software is built with Node.JS framework to control all data syncing processes and handle communications between the client and database.

Synchronisation system developed

An LMS data syncing framework has two major parts: database syncing and file syncing. We discuss the data syncing first. Adapting some of the approaches mentioned above, we introduced a last_sync variable for every record in the database and compare this variable with the variable of the same record on other devices, and we always use the record with the latest last_sync variable to overwrite the older record. Therefore, we can always keep data in the system updated and accurate and resolve conflicts when syncing. By resolving conflicts through comparing the time of last changes and time of last sync, this algorithm is universal and can apply to any system that is built with a relational database.

Prerequisite for data syncing
Our proposed data syncing framework considers other factors relating to how information can be edited and modified by everyone in the system. This information requires incremental data syncing and conflicts handling to minimise data usage and prevent data loss, which starts with the insertion action.

**Data creation conflict:** First, we investigated Moodle database structure, then we found that Moodle uses auto incremental integers ids as primary keys for records in the database. When we have different devices make changes asynchronously, devices may create same id number for different records at the same time. For instance, student A posts a first discussion to forum on device A, then student B posts his first discussion to same forum on device B. As both posts are the first post, so the id number will be 1 for both of posts, but certainly they are different posts, from different people, with different contents.

When device B trying to sync data with device A, then conflict occurs as there are two different records but with the same unique id and primary key. To make sure each record has a universal unique id as primary key across all devices, we need to disable auto incremental feature for id field in each table in the database.

**ID creation:** In this research, we investigated the Unix timestamp (Unix time) which is the number of seconds that have elapsed since the Unix epoch, which is 00:00:00 UTC on 1 January 1970. Unix time only contains numbers, which can be read and processed by Moodle system natively without changing source code or database structure, which is ready-to-use for our system. However, there are still chances that the Unix time will generate duplicate IDs because Unix timestamp is uniquely identified by seconds, when to records are being created at the same time, they will have the same Unix timestamp therefore causes collision.

At this stage, our system is focusing on tertiary students in remote Northern Territory regions, where only has a small number of students and lecturers. As a result, it is very unlikely that students and lecturers will post information to the system at the same time within one second, therefore there is a very small chance to get duplicate Unix timestamp in the system. Considering pros and cons of Unix timestamp, we decided to use Unix timestamp as unique identifiers for all records that need to be synced.

**Implement new ID:** There are two ways to implement Unix timestamp as unique identifiers: by modifying Moodle source code or using SQL triggers. As we do not want to touch Moodle source code to be able to get software upgrades in the future, we will investigate SQL triggers. SQL triggers allow us to perform actions to the record that being inserted, updated or deleted, before or after the action being executed. In our system, we create a trigger for each table that contains data need to be synced, the trigger will be fired before a record is inserted to the database. The trigger will first check if the inserted record has an ID, if the record has an ID, then ID should not be generated and replaced by timestamp, as the ID is already assigned and need to be kept. If the record does not have an ID, then the trigger will generate a timestamp and assign this timestamp as the ID for this new record, and the ID will also be the primary key.

The setup of the synchronisation was designed with the following components:

**Setup Server:** In our data syncing framework, a syncing server is required to handle syncing requests and process data. In our system, we use Node.js web framework to build the server, and use Express to build REST APIs as endpoints for devices to send requests. All requests and responses are in JSON format. We now present the client and server syncing algorithm.
Setup Remote Device: To be able to sync data automatically, the device (Raspberry Pi) needs to check internet status constantly. Once internet access becomes available, the device will send syncing requests with data that needs to be synced to the server. First, we created a shell script to check internet connection status, this script will start running after the raspberry pi is powered on and Ubuntu system is booted. Then the script will check internet connection by pinging the Google domain every second, once the script is able to solve the domain and receive response from Google, then an internet connection is available and a JavaScript script will be executed, that contains data syncing algorithm. By introducing an hour threshold before future sync requests, we can reduce unnecessary data exchange, as we only have small number of students that will be using this system and the frequency of the data change is very low. As a result, the system will consume less battery power and mobile data, which is beneficial in remote Northern Territory environment where both electricity and internet is difficult to obtain.

Database synchronisation steps

To verify the system, we conducted both thought experiments and physical tests with users at remote locations. We explain here the step-by-step process.

When the central server receives a POST request from a device that coming into data syncing endpoint, the server will first check if the request body is empty. If the request is empty, that means the POST request is invalid and the server will send 400 status code back to the device indicates that the request is invalid. If the body of the request is not empty, the server will then repeat the same process as done on the client, and iterates through all tables to select records in each table that have time_modified values bigger than last_sync variable stored in last_sync.log file. Then the server will combine all records from the same table to a JSON array and combine all JSON arrays to a final nested JSON array.

Once final JSON array has been constructed, the server will send response to the device with status code 200 and the final JSON array in the body. We designed the system to send response to the device before proceeding to the data syncing process because the device is still waiting for response from the server at this stage, and we want to send response to the device as soon as possible so that the device can disconnect from the server and continue its own data syncing processes. This will reduce the risk that data syncing process being disrupted by losing internet connections due to unstable internet connections.

Next, the server will start to iterate every JSON object in every JSON array received in the initial request body. For each JSON object, the server will perform a SELECT SQL query using the ID of that object to find out if the record exists in the same table in the server database. If no result found using the ID, then the server will execute an INSERT SQL query to insert this record to the database. If the record exists, the server will compare the time_modified attribute of the record from device’s request with the time_modified attribute of the record from the server’s database. If the time_modified attribute of the record from device’s request is bigger than the time_modified attribute of the record from server’s database, that means the device’s record is newer then the server’s record, the server’s side record needs to be updated. In this case, the server will perform an UPDATE SQL query to update the record on the server side with the record received from the device.

When performing INSERT or UPDATE queries, the server will accept the value time_modified variable in the record received from the device to perform queries instead of using current timestamp, this will make sure the time_modified variable of the record is
consistent across all devices therefore make the comparison of records easier. Once all records in the request body received from device has been iterated and processed, the server will update its last_sync variable that stored in last_sync.log file with the current time in Unix time stamp format. Till now, all processes in database syncing algorithm on the server side are finished.

Then once the remote device receives a response from the server, the device will first check the status code of the response. If the status code is not 200, the device will finish the data syncing programme and record the message in the response body in a log file on the device. This will help us to indicate the problem and optimise data syncing programme in the future. If the status code is 200, then the device will check if the body of the response is empty. If the body is empty, this means all data is up-to-date and no data syncing necessary. If the body is not empty, the device will then execute the same data syncing algorithm we have on the server.

If the time_modified attribute of the record from server’s request is bigger than the time_modified attribute of the record from device’s database, that means the server’s record is newer then the device’s record, the device’s side record needs to be updated. In this case, the device will perform an UPDATE SQL query to update the record on the device side with the record received from the server. When performing INSERT or UPDATE queries, the device will accept the value time_modified variable in the record received from the server to perform queries instead of using current timestamp, this will make sure the time_modified variable of the record is consistent across all devices therefore make the comparison of records easier. Once all records in the request body received from server has been iterated and processed, the device will update its last_sync variable that stored in last_sync.log file with the current time in Unix time stamp format. Till now, all processes in database syncing algorithm on the device side are finished. Figure 4 illustrates all processes within the data syncing framework.

Figure 4 Database Syncing Algorithm
File Syncing

When database syncing on the device is finished, the device will then try to sync files with the server. In the Moodle system, all user data are stored under a moodledata folder under /var/www/. Moodle will store every user data file using hashed file name and keep the hash value of that file in the database to avoid conflict when two different files have same name.

Since the files are student submission and may be submitted a few times from different box locations as students move around, we want to keep every user file to avoid file syncing conflicts. When one file exists on one device, we want to copy that file to every other device that doesn’t have that file during file syncing. To implement, we use rsync, a “Linux-based tool that can be used to sync files between remote and local servers” (Rsync, 2018). Rsync allows us to compare directories on devices and central server, find missing files and copy missing files from one device to another.

As the ability and availability of file syncing depend on the internet accessibility of the device (Raspberry Pi), the rsync will only be executed on the device when internet is available. After device finishes database syncing processes, the same script that wrote in JavaScript will then execute rsync command. The device will start push local changes to the central server, in this case, the device will be the master and the central server will be the slave, which means all files exist on the device but missing on the central server will be copied to the central server.

In the meantime, the device will pull changes from server to the device, in this case, the central server will be the master and the device will be the slave, which means all files exist on the central server but missing on the device will be copies to the device. Those two processes will be executed at the same time synchronously to minimum the time of execution and reduce the chance of syncing process being interrupted by internet disruptions.

Once all database syncing processes and file syncing processes finish, the script on the device (Raspberry Pi) will return a status message. If any error happens during the syncing processes, the script will return an error and record the stack trace and the time of the error in a log file named error.log on the device, under the same directory of the syncing script. This will allow us to debug and resolve the problem in the future. If no error happens during the syncing processes, the device will then update it’s last_sync variable in last_sync.log file to the time that the syncing process finished and return finish message. If synchronisation fails due to disconnection this will restart when next connected, the change is not saved until completed.

Improvements to design and testing

During the development of the system, various changes were requested, as both the lecturers began to understand and appreciate the opportunities provided by the Learning Box, and the system was tested sometimes remotely when going to community for other unit delivery.

Accessing system

The Raspberry Pi access required the user to switch their phone to the WiFi hotspot address, which was not easy to explain. Then they may wish to disconnect to get information of the
internet when online. In addition to above components, the client part also has a mobile app that allows users to connect/disconnect to the learning system WIFI hotspot by on-click and show connection status of the learning system. Once a user connects to the learning system WIFI hotspot, this app will open learning system webpage automatically in a web browser. The tiny app is installed on the users system during orientation, and is available for download from the server site if they change phones. This will not be a large data pull.

Also, users will be given a Learning Box id when installing the mobile app. If they have one this can be re-entered when re-installing. All user ids will be shared across all Learning Boxes as students move around the region.

**Audio and Video tutorials**

We found the students were having issues in submitting on the existing university LMS system, partly due to the fact they often were working on a phone with no laptop and hence little access to writing. Hence the proposal for oral submissions was welcomed by the lecturing staff. However, this meant users needed to understand the options available for video and audio editing on their phones. Short tutorials in video were used to provide this introduction to existing applications that users could install, and the mobile applications included on a link internal to the Moodle LMS on the Raspberry Pi.

**Power supply**

A user story that was developed later was the need to consider running the Learning Box without external power at times. As it is a portable device that allows students on a local network to connect to a learning management system within this device, this was important. The Learning Box consist of a Raspberry Pi, a mobile broadband board, and a cooling fan. We therefore investigate how much power this box consumes so that we can choose appropriate power source for this device.

To test the power usage of the device (Raspberry Pi), we connected the device to a power source. The temperature of the room where the experiment executed was settle to 32 degrees Celsius with the humidity around 70%, which is the actual weather condition that students in remote Northern Territory regions likely to be in. The power supply was an Australian standard domestic power supply with 240 AC, 50 Hz.

First, we want to test the power usage of the device in idle mode. Idle mode means the device is powered on, student is connected to the device with student’s phone or computer, but students is not performing any task or operation. In addition, there is no service running except core system services. In the idle mode, the rate of energy transfer was around 4.8W to 5W and the electric current was around 0.02 ampere to 0.03 ampere (See Figure 5a).
Then we investigated the power consumption when the system is in use. In this case, we activated the WIFI hotspot on the Raspberry Pi device, the cooling fun was fully functioning, and one user connected to the hotspot using the user’s own device and performed basic operations within the Moodle system, including browsing different pages, downloading course materials and uploading files. When the system in use, the rate of energy transfer was around 7W to 8W, and the electric current was around 0.03 ampere to 0.03 ampere. The highest power consumption was recorded when user downloading a large file from the system, the highest rate of energy transfer was 8.1W and the highest electric current was 0.03 ampere (See Figure 5b). From these results, we concluded that the Raspberry Pi device consumes very little power of electricity, therefore a power bank with 10000mha of battery capacity (eg the 3pin LiPo PiJuice Battery) will be enough to supply the device for 24 hours until the user can find a power source to recharge the power bank.

Analysis of System Use

The final step was to conduct a data analysis of the students learning so that when it goes out to communities, we will be able to monitor and advise students and lecturers. We looked at what data will be possible to collect from users that would be useful for educational analysis (see Table 1) and consider what available analysis tools are in an LMS and how to generate these for the lecturers. This affected also what data tables we needed to synchronise. The following data is collected from the Moodle system running in the students’ browser on the WIFI link to the Learning Box.

Number of Views: Moodle has a built-in log, which records all user activities. This includes activity description, activity type, time of the activity, IP address, origin etc. In this log, we can sort out all page view activities of each user, therefore we can understand how frequent a user view specific page, course material or the learning system in general. This data can help us to determine students’ activity, engagement, and effort made in their study and can give us insight on each student’s learning progress. In addition to page views, we can also find out times of login for each student, this can allow us to monitor the students’ effective attendance.

Time spent in the system: In addition to the standard Moodle log, we can also collect how long does each student spend in this system, by introducing a plugin called Course Dedication. This plugin can estimate dedication time of participants within a course, by recording the time between two clicks from the user. If the time elapsed between two clicks is less than a pre-defined threshold, then the system will consider the user is using this system continuously during this time and count this amount of time in the total time the user spent in this system. If the time elapsed between two clicks is greater than threshold, the system will
consider the user has left the system during this period. By recording individual’s time spent in this system, we can get more accurate insight on each student’s ability to dedicate time on learning the course.

**Number of Downloads:** In Moodle standard log, each file downloading operation will be recorded in the log. To calculate total number of downloads of each student for each resource, we can download the entire log from each Learning Box in excel format, then perform excel command to calculate the total number of downloads. The number of downloads can also indicate each student’s engagement of the course.

**Interactions With Peers:** Interactions with peers is another variable that can be used to assess student’s engagement. To collect interactions with peers’ data, we can calculate the total number of each student’s responses to other students’ postings. This data can be extract using Moodle built-in log and analytics tool. The more responses a student posts, more active that student is when interacting with fellow classmates.

**Student Learning Outcome:** Student learning outcomes can be assessed in various ways. The most direct way is using student’s grades. In the Moodle system, grade centre allows lecturers to see and summarise each student’s results for each assignment, this can be used to indicate how well that student perform in this course. This is handled on the server and synced to the remote Learning Boxes.

<table>
<thead>
<tr>
<th>No.</th>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Total login frequency in LMS</td>
<td>Adding up the number of individual student’s login time into the LMS</td>
</tr>
<tr>
<td>2.</td>
<td>Time spent in the system</td>
<td>Calculating the total amount of time spent between login and logout</td>
</tr>
<tr>
<td>3.</td>
<td>Number of downloads</td>
<td>Adding up the numbers of course materials downloaded</td>
</tr>
<tr>
<td>4.</td>
<td>Interactions with peers</td>
<td>Counting the total number of student’s postings responding to peers</td>
</tr>
<tr>
<td>5.</td>
<td>Number of exercise performed</td>
<td>Counting number of exercise a student has done</td>
</tr>
<tr>
<td>6.</td>
<td>Number of forum posts</td>
<td>Counting a number of posts a student has contributed in the discussion forums</td>
</tr>
</tbody>
</table>

Table 1: Variables that can be collected in Moodle System (Mwalumbwe & Mtebe, 2017)

Once the data required for our analytics is collected, we pre-process the data:

**Create summarization tables:** Because there are hundreds of tables in Moodle database, it is necessary to create a table to summarize all required information for data analytics. In this study, we created a table named mdl_summarization which summaries each student’s learning activities and outcomes.
**Transform the data:** Data analytics with data mining algorithms require specific format for data input. The last step of data pre-processing is to transform pre-processed data to format that can be recognised by data mining software and algorithm.

Using these types of analysis, we can propose which variables to monitor our students learning in future. However, as a precaution we cannot rely on such relationships being causal rather than just correlated hence this requires ongoing implementation research with the lecturers.

**Results**

The research aimed to satisfy various user stories that we gathered from literature and the student needs at our university. We provide here the outcome of the development iterations towards these requirements

- User can improve learning by sharing their material with other students in their community
  Sharing is enabled both within a community on one Learning Box hub, or synchronised across communities where possible. This will encourage others in the community to engage as they can access their colleagues work
- User is regularly out of internet range but still wants to be able to submit and access material
  Offline use of the system is achieved with support for users to login to the Learning Box wifi and then view the material that has been downloaded to date, including other students work at times, and submit their own material. The range of the Learning box will depend on obstacles in the community environment.
- Both the new system and the existing CDU system should be intelligible to the user
  Working on a mobile has made the system more accessible to our remote students, and videos have been selected and included in the learning material to support their use of the system on mobile. An extra app was developed as the initial login to assist novice users
- User can upload submissions as audio and video as well as text
  The access to other forms of media for submissions is important as we upgrade our learning requirements to suit our remote students’ needs and skills
- User will mostly be on mobile devices
  We used existing PWA software for mobile development and this can also be used on tablets and PC for the larger screen.

Technology testing found the system will work offline and will synchronise when back online taking into consideration the user cases that were identified, such as users moving between communities or boxes. With minimal load requirements for each box, the synchronisation time was not an issue, and the time for upload in synchronisation depends on students’ material. If synchronisation fails due to disconnection this will restart when next connected, the change is not saved until completed.

**Future work**

We are looking to work with our students in remote Northern Territory regions to participate in the next stage of researching our system, so that we can collect data we need from students and conduct data analytics based on proposed data analysing techniques and algorithms. After students using our system, surveys will be sending out to students to complete in the
next stage and interview process will also start. By getting students’ feedback we can have insights on what level has our system been affected students learning experience in remote Northern Territory regions. In addition, we also need to get students’ feedbacks on their general opinion through surveys and interviews on how we can improve our system, especially in terms of user experience and interactions. Furthermore, we should also ask education institutions and lecturers to provide their feedback on the support for teaching aspects such as collaboration, during the next stage of this research.

Conclusion

This research investigated an offline approach for collaborative learning targeting students in remote Northern Territory regions where internet resources are limited. An offline compatible digital learning system has been built, which contains Raspberry Pi, solar power bank and Telstra 4GX mobile dongle for hardware, Ubuntu for operating system, Moodle for learning management system, MYSQL database and an asynchronous data syncing framework for software. Although we were unable to conduct user testing and collect data from users, however, various data collecting and analysing techniques have been assessed and we have a clear picture on what data we can collect within this system and how to analyses these data after we collect these data.

References


https://doi.org/10.1007/s10758-014-9226-4


doi:https://doi.org/10.1002/j.1681-4835.2017.tb00577.x

Sathya, D & Ayyapan, G 2014, 'A Secured Message Digest for Efficient Database Using Improving