COMP9900 - Information Technology Project
Intelligent Assistant for electronic Logbook in Radiology training
T3, 2019

Submitted by

Team – Cheesecake

Ajinkya Biswas (Developer)	z5196315@student.unsw.edu.au
Daniel Hocking (Developer)	z5184128@student.unsw.edu.au
Ghazali Hali (Developer)	z5220911@student.unsw.edu.au
Smrita Pokharel (Scrum Master)	z5125134@student.unsw.edu.au

Contents

Introduction

This report is part of the submission requirements for course COMP9900. It describes the implementation details of ‘Project 4: Intelligent Assistant for Electronic Logbook in Radiology Training’.

The solution aims to provide assistance to medical students, specializing in cardiology, nuclear medicine or radiology, in maintaining an electronic logbook of CT Coronary Angiography (CTCA) cases performed by the students as part of their training. The logbook follows standards prescribed by the Conjoint Committee for the Recognition of Training in CT Coronary Angiography. The software application provides support for both certification and recertification pathways that a student can follow. Training requirements that are set by the Conjoint Committee are configured in the system to help guide the progress of the student.

A project proposal document was submitted which outlined the design and approach to the project and acts as a precursor to this document. Most of the goals set in the proposal document have been met however, as is true with any journey, a few circumstances, technical and otherwise, were faced and dealt with amicably with guidance of the excellent support staff of the course.

The application has been developed using agile methodology where the project was divided into three sprints, each of which was composed of two weeklong development, testing and deployment activities. Project demonstration was conducted with supervising staff periodically in the middle of each sprint and their feedback was incorporated to improve the outcomes of the sprints. In addition, the team also conducted 3 meetings per week. Two were face to face and one was over a conference call. The meetings followed scrum specified agenda. An electronic tool, Trello, was used to keep track of each sprint and its associated activities. For source code versioning, GitHub was employed.

The rest of the document details the scope of the project, architecture and implementation specifics as well as challenges and learning of the team on this development journey.

System Overview

The application is implemented using cloud-based infrastructure, Google Cloud Platform. All software components are deployed on a virtual machine environment (Google Compute Engine). The database is PostgreSQL.

The main interface of the application is web-based. The front and backend of the application is developed using Flask, which is a web framework written in Python programming language. The application is integrated with Dialogflow, which is a service provided by Google for developing a conversational interface like chat-bots. This serves as an additional interface for users to interact with the system.

System Architecture

The system will be described using a methodology called the 4+1 architectural view model. It is a modelling technique to describe the system from multiple viewpoints which are described below. Some aspects of the methodology are modified to suit the needs of the project.

1. Deployment view: describes the mapping of the software onto the hardware/infrastructure.
2. Logical view: major functional features of the application are described. This view is combined with use-case view and actions that can be performed by specific users are explained in this section.
3. Development view: describes the organization of the software in its development environment. Modules and libraries that are utilized in the development of the application are described in this section.
4. Data view: describes the data elements that are managed by the system

Deployment view

The software is deployed on 'Compute Engine' which is Google's offering for a virtual machine with persistent storage on Google Cloud Platform. The system has an external endpoint exposed and an internet domain name is registered for the solution logbooks.sydney.

The solution employs Docker for containerization. The images for docker are stored in the storage bucket on GCP whose name is mentioned in the diagram below. The setup guide, in a later section, provides more details.

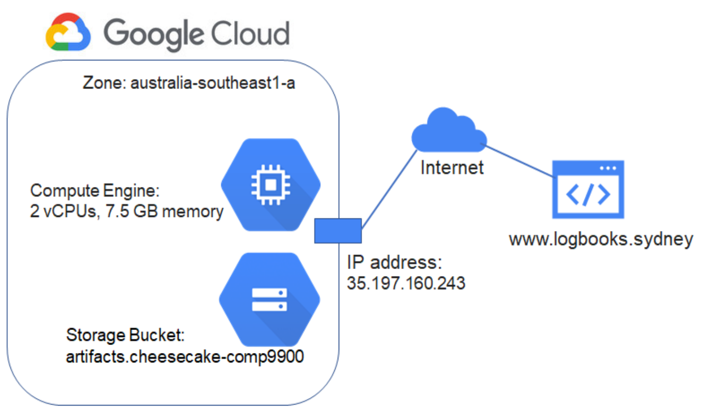
 Google Cloud Zone: australia-southeast1-a Compute Engine: 2 vCPUs, 7.5 GB memory Storage Bucket: artifacts.cheesecake-comp9900 IP address: 35.197.160.243 Internet www.logbooks.sydney	The virtual machine specifications are: Type: n1-standard-2 Zone: australia-southeast1-a CPU: 2 vCPUs Intel Broadwell, RAM 7.5 GB Operating System: Debian 9

Figure 1: Cloud deployment and domain

Logical view

Major functional features of the application are described below. They are categorized into three layers: interface, use case and back end layers. The components that make up each layer are depicted in the diagram below and explained in this section.

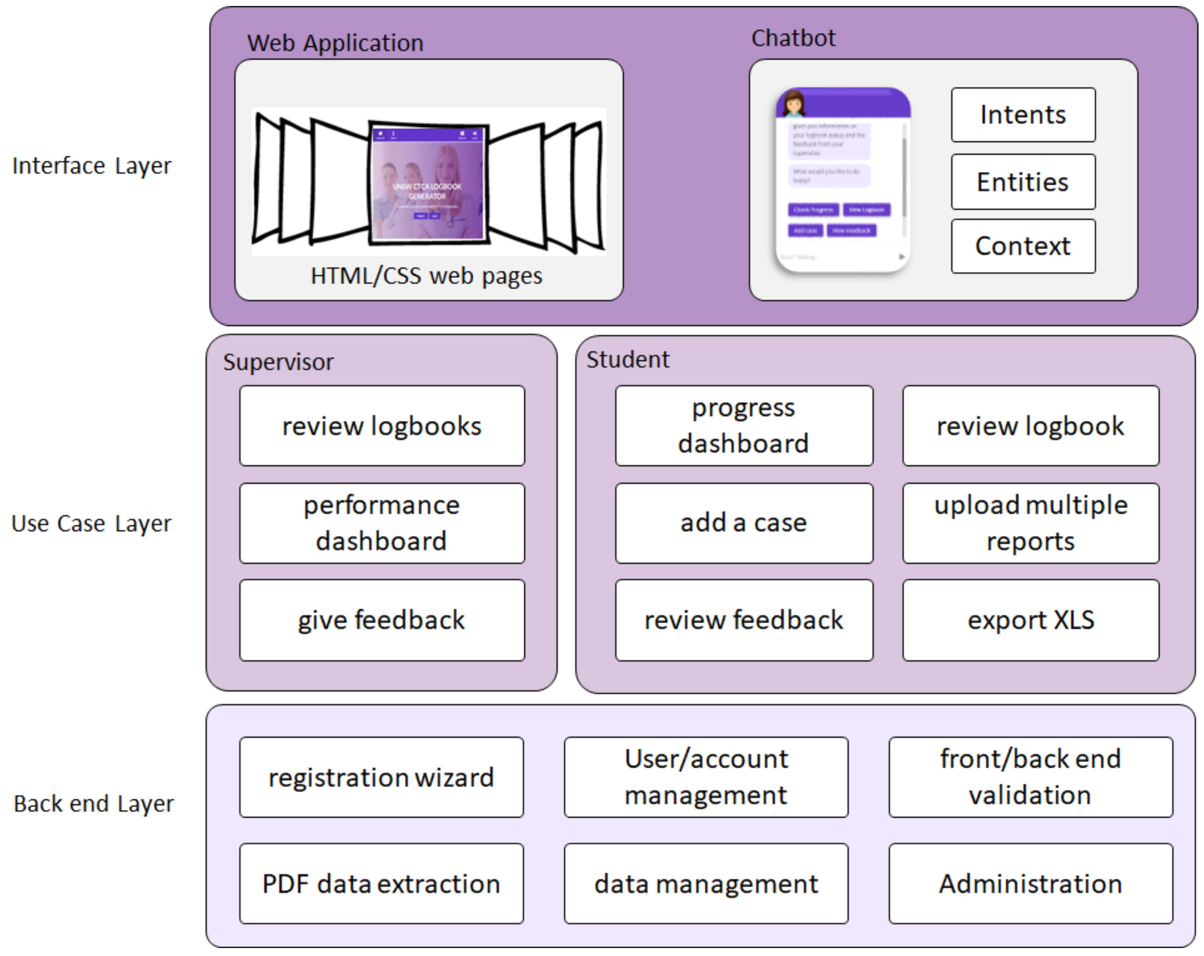


Figure 2: Major functional features of the application

Interface layer

- Web interface.The primary interface of the application is a web application accessible over a web browser. This was chosen as the medical reports that are used to add data to the logbook reside primarily on desktop or laptop machines. The HTML pages are augmented with CSS and javascript to give a responsive experience to the user. Each page is also embedded with an iFrame that enables access to the chatbot which is the secondary interface of the application.
- Chatbot. The chatbot acts as a guide to the user and is able to handle frequently asked questions, webpage usage questions and also serves as the main prompt for user to take action on. The chatbot is aware of the context in which it is operating, that is information of the user, their stats and the current webpage they are on are accessed by the chatbot. This is combined with natural language processing (NLP) capabilities enabled by utilizing Google's DialogFlow API.

Use Case Layer

There are two main types of users: Students and Supervisors. A user account can only be associated with one of these types. User account creation and registration wizard are described in the backend layer. Once a user is created and registered, upon logging in they are presented with web pages and functionality that is relevant to their respective types. Major functional features associated with each type of user are described below:

Student:

- Progress Dashboard. In the main landing page for a user, once logged in as a student a dashboard is presented. It gives a summary of the students' progress to date in terms of number of cases logged and their type. It also indicates feedback received from the supervisor. Both charts and tables are used to present information in a concise and easily understandable manner. Targets for number of cases to be logged are also highlighted. These targets are set based on recommendations from training requirements set by the conjoint committee.
- Review logbook. A student can review the entries made in their own logbook. The logbook type varies depending on the type of training the student is pursuing. The number of cases logged are presented as a run chart to help students gauge their progress over a year. Generally, logbooks are maintained over a period of three years, which is the standard duration of training. The logbook entries are presented as a spreadsheet on the webpage which can be edited by the student.
- Add cases and upload reports. This is the main feature of the application. A student can either enter data for an individual case through a webform or upload diagnosis reports which are parsed, and data is extracted by the system. The user can upload multiple reports at a time and verify the extracted data before saving it in their logbook.
- Export logbook. The system allows users to export data saved in their logbooks in the form of a Microsoft Excel file of type XLS. The exported file follows the template set by the conjoint committee for training. The excel file contains two tabs, a summary tab and data tab. This excel file is supposed to be submitted by the student as part of the application for registration by the CTCA authority.
- Review feedback and comment. A supervisor can from time to time provide feedback to the student. A student can review their specific feedback, acknowledge it and also provide counter comments if any.

Supervisor:

- Performance Dashboard: The supervisor, once logged in to the system is presented with a dashboard that provides an aggregated overview of performance of students that are under his supervision. Count of students with respect to the the training path they are on is presented alongside the total number of cases logged per month over the past year. The supervisor dashboard also presents charts indicating top performing students and those that might need attention. Student performance in this case is judged by the number of cases logged and recency of their activity.
- Review logbooks. The supervisor can review logbooks of all the students that are under his supervision. The supervisor also has the rights to update and export these logbooks.

- Feedback. The supervisor can give one on one feedback to a student. The system maintains the date and time of the feedback and also indicates if the student has acknowledged it or not. The supervisor can also

Backend layer

Functional features of the system that we have categorized as the backend are described below:

- **User account management:** Registration of a user is done with email id set as the user identifier in the system. A confirmation email is sent to the user to activate their account.
- Registration wizard. A guided step by step wizard is built to enable newly created accounts register them successfully with the system. Registration allows capturing important information about the user to determine their role in the system. A user chooses the type of training they are on, start date of the training and also chooses their supervisor from a list of supervisors registered in the system.
- **Data extraction:** Medical reports in the form of PDFs are uploaded in the system to enable extraction of key fields used to populate a logbook. The user can upload multiple files at a time and once the files are parsed the extracted values are presented in a tabular form onscreen for the user to verify, update if necessary and save the records.
- **Data management:** The system persists user's data, their logbook entries and progress statistics in a relational database. It is further detailed in the data view section of this document. The PDF documents are not stored in the system.
- **Validation:** Input from the user is checked and verified both at the user interface level and also from the back end. Validations include password strength, data type checking and correct format relevant to the type of logbook entry.
- **Administration:** The system administration involves setting up the application, managing docker images and configuration of the virtual machine. These activities are described in the technical manual section of this document.
- **Authentication:** The webservices that have been exposed in both backend and frontend are authenticated using HMAC which is a type of Message Authentication Code which has a hash function and a cryptographic key.

Development view

In this view of the system we have highlighted various software components that make up the system. The system is primarily developed in Python 3.7 and uses a lightweight web application framework Flask and containerization using Docker to handle server management, all code written for the project is contained in the VM_setup directory, within this there are two main components:

Backend: used to store, retrieve and manipulate the data of the application

- **Models:** they provide an interface between the database and the app, they are managed by SQL Alchemy ORM and contained in api/models sub-directory
- **API endpoints:** allow the frontend to access functionality provided by the backend, they expose URL with specific methods that will respond with JSON formatted data, these are contained in the api/views/main.py file
- **Other:** helper and other functionality contained in api/views/helpers.py and some other files like api/core.py and api/dialogflow_controller.py

Frontend: retrieves data from the backend to build interfaces for the user

- **Models:** most data are managed by the backend, however there is still a need to manage users and roles on the frontend this is contained in app/models/user.py
- **Controllers:** send requests to the backend to get data to use in templates, take use submissions and send data to the backend. The controllers used for student pages are found under app/main while the ones for the supervisor are under app/supervisor
- **Views:** they use Jinja2 as the template engine to build a view that is mostly HTML, but allows for data to be inserted from the controller part of the application, they are found in the app/templates folder
- **Assets:** CSS/JS files are contained in the app/assets folder, they are included from the app/assets.py file
- **Other:** other helper files can be found under the app directory, for example the helpers.py and decorators.py files

The https_container directory is a boilerplate NGINX docker container used as a reverse proxy, it allows for the production server to use HTTPS without any changes to the setup on the local environment, to make use of it you would need to register a domain and ensure the correct ports have been opened on the servers firewall, it is not recommended for testing purposes.

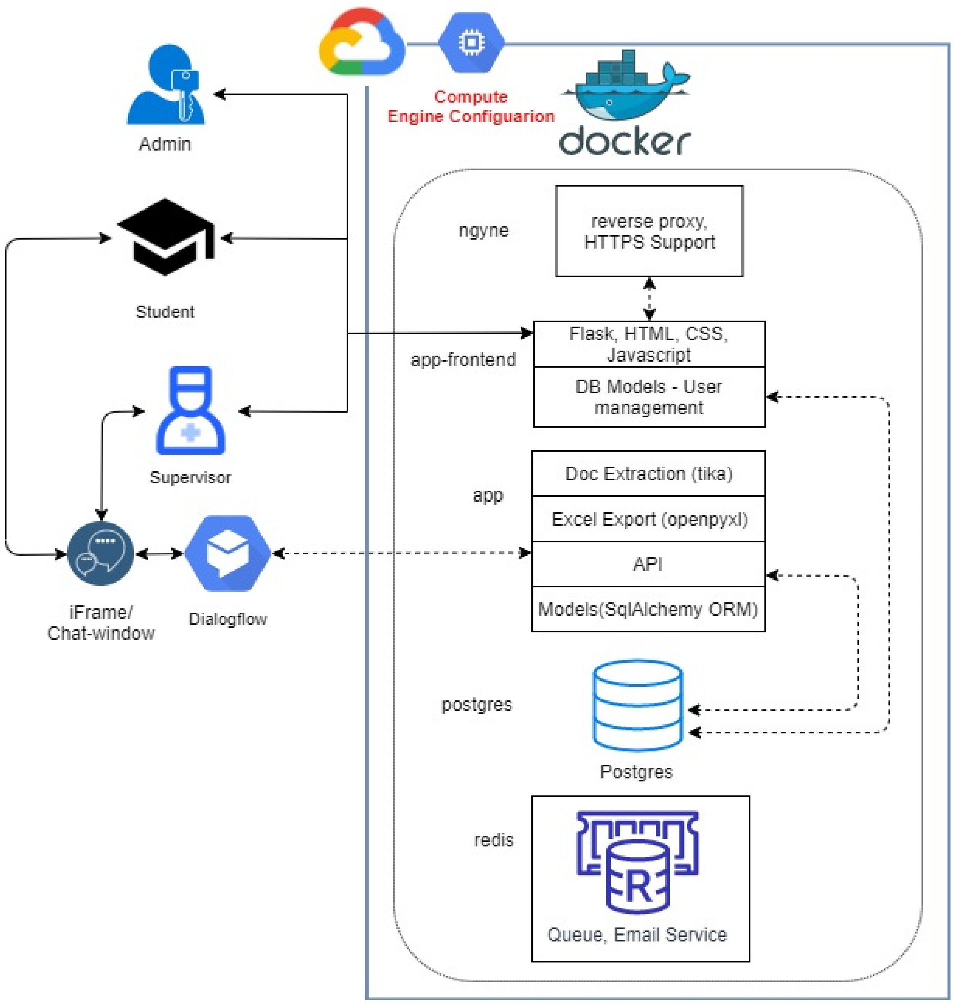


Figure 3: Development view

Services

- Google Cloud Platform (https://console.cloud.google.com/)
  Compute engine, Storage, VPC network
- Dialogflow (https://dialogflow.com/)

Software

- Docker (https://www.docker.com/): has become an industry standard for providing a lightweight container-based virtualisation system
  It has a community edition that is free and open source (under https://en.wikipedia.org/wiki/Apache_License#Version_2.0)
- PostgreSQL (https://www.postgresql.org/): used as the database server
  Free and open source https://www.postgresql.org/about/licence/

Frameworks

- Flask Boilerplate (https://github.com/tko22/flask-boilerplate): used by the backend to help structure the API endpoints
  It uses the MIT license (https://github.com/tko22/flask-boilerplate/blob/master/LICENSE) which is a very permissive and commercial friendly license agreement
- Flask-base (https://github.com/hack4impact/flask-base): used by the frontend to help structure the code, it also provided a good starting point with the registration and login system already implemented
  It uses the MIT license (https://github.com/hack4impact/flask-base/blob/master/LICENSE.md) which is a very permissive and commercial friendly license agreement

Libraries

- Flask (https://www.palletsprojects.com/p/flask/): a lightweight web framework
  License can be found at https://www.palletsprojects.com/license/ it is fairly permissive and doesn't pose problems
- SQL Alchemy (https://www.sqlalchemy.org/): provide an object relational mapper (ORM) for improved database access
  Uses the MIT license (https://en.wikipedia.org/wiki/MIT_License) which is a very permissive and commercial friendly license agreement
- OpenPyXL (https://openpyxl.readthedocs.io/en/stable/): read and write Excel files
  Uses the MIT license (https://en.wikipedia.org/wiki/MIT_License) which is a very permissive and commercial friendly license agreement
- Tika (https://github.com/chrismattmann/tika-python): text extraction from PDF documents
  Uses the Apache 2.0 license (https://github.com/chrismattmann/tika-python/blob/master/LICENSE.txt) which is a permissive and commercial friendly license agreement

Data view

The Entity relationship diagram for the solution is given below.

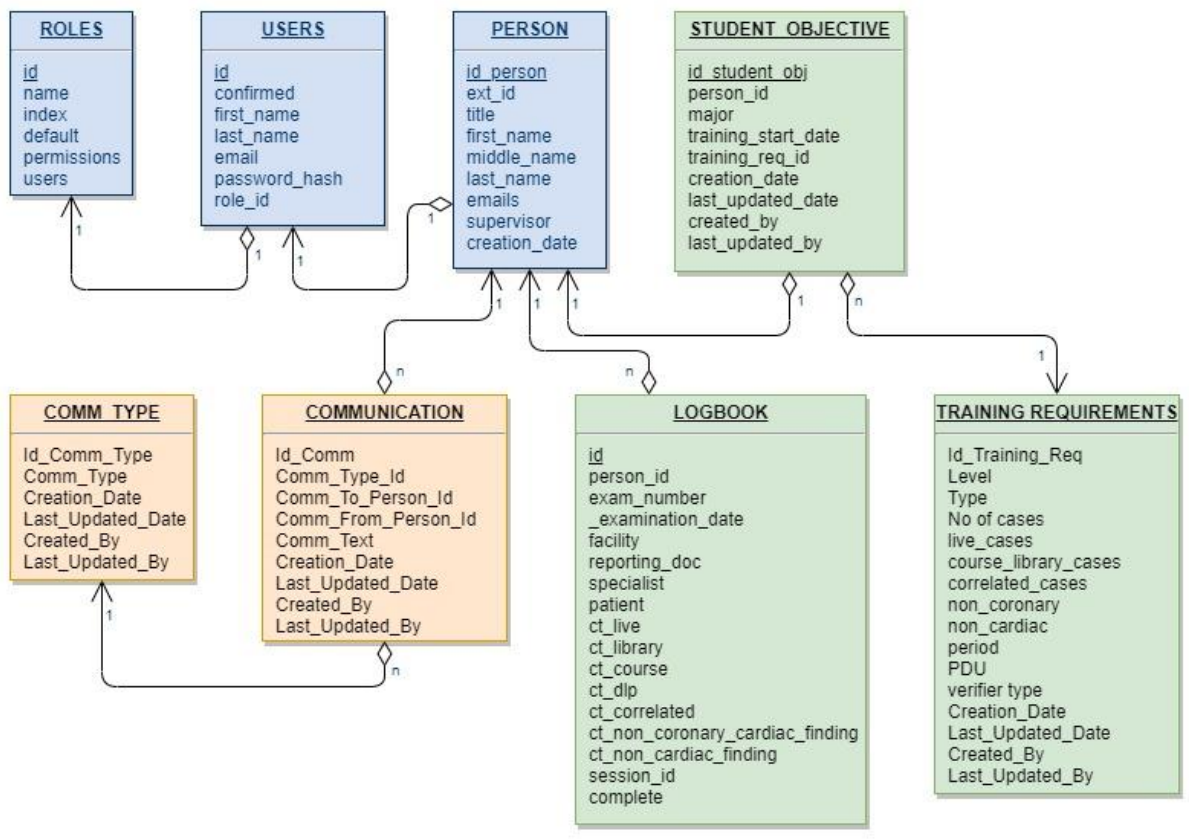


Figure 4: Data entities and their relationships

Description:

Entity	Description
Roles	Stores different user roles – student, supervisor, admin and their access levels in application
Users	Stores user log in information including password in hashed form
Person	Stores person details. It stores student's supervisors also
Student Objective	Stores student's target certification details, training start date
Logbook	This is the table where all logbook entries are stored
Training Requirements	CTCA has certain requirements for each type of certifications. This table stores that information
Comm_Type	Supervisors can provide feedback to students. This table stores what type of feedback is mentioned
Communication	This table stores the actual feedback from supervisor to students

Implementation challenges

Cloud environment setup

The project requirements had suggested that a cloud-based environment should be used. The team had little experience with the technology. However, the knowledge gap was quickly overcome with accelerated training of the Google cloud platform (GCP) and practice using free trial services. Once a virtual machine (compute engine) was setup, the team experimented with Cloud SQL service, which is a fully managed database service that makes it easy to setup, maintain and administer a relational database on GCP. However, because of overhead costs and the fact that our application did not require extensive managed services we opted to install a local PostgreSQL instance on the virtual machine.

Machine Learning for document text extraction

In our initial design proposal, we had suggested use of machine learning model to extract annotated text from CTCA reports. Our research into the problem led us to Google image recognition api, which is a service available on GCP that uses computer vision-based algorithms to extract text from images (OCR) and has support for extracting dense text from PDF documents as well as image files. A prototype was implemented, and we were able to successfully extract raw text from the PDF report however the prototype made it evident that Google's Vision API for 'Document Understanding' has a lot of limitations. Firstly, this offering on GCP is currently in public beta and only a small subset of the features is available. Secondly the cost of usage is per API call, which makes it very expensive for large number of documents. Another limitation was around security. Any document that is to be consumed by the API has to be available on a storage bucket and tagged as public. GCP had suggested a few workarounds for this, but it required more processing overheads.

The biggest challenge however was not converting a pdf document to text, but rather converting text to data, i.e., once we have a series of text strings, converting them into key value pairs that can be stored in a structured database to enable the rest of the application to process this data. The standard way to achieve this is to annotate the text with named entities. We further experimented with Stanford's CoreNLP solution, which is an open source library and contains a number of packages for natural language analysis. It converts text into lists of sentences and words and enables its built-in annotators to recognize parts of speech, named entity recognition, dependency parsing etc. The out of the box features was not suitable for our task at hand, where we needed specific information for logbook entry like radiation dosage, facility information, concurring doctor names and the like. One approach which was tried unsuccessfully was adding a custom annotator on top of Stanford's library.

After a discussion with the course mentor it was agreed to abandon this approach and instead come up with a rule-based model to convert text to data. This entailed writing XX rules based on scanning preceding and succeeding tokens of text and storing those that match the rule criteria. This approach is sufficient for the project at hand; however, its main limitation is that it will work only on the few types of reports that were provided as samples.

User interaction design

The requirements for the project had suggested creation of a 'conversational agent' as the primary interface for the application. The conversational agent or chatbot would populate logbook entries after a question/answer dialogue with the user. We found this approach to be a little cumbersome in cases where multiple documents have to be uploaded at a time or multiples updates are needed to be done on logbook entries.

We considered usability heuristics for user interface design, which are a set of 10 rules of thumb by Jakob Nielson to guide implementation of interaction design. In particular we wanted our design to be minimalist and provide a familiar interface for the users, relying totally on a conversational agent would not have matched users intentions and expected behaviour of the application.

We opted for the web interface to be the primary interface for the user and envisioned the conversational agent to be a user guide and help the user in navigating the web interface. The chatbot is present on each screen and provides useful information for the user in enabling them to effectively use the application.

Technical Manual (setting up *logbooks.sydney*)

Cloud environment setup (GCP)

- Start by creating a Compute Engine VM instance: https://console.cloud.google.com/compute/instances
- Change the name to something appropriate
- Change the region/zone to australia-southeast1-a
- Machine configuration to n1-standard-1 or n1-standard-2

After the VM has finished being created set the IP to static:
https://console.cloud.google.com/networking/addresses/list
Then add a new firewall rule: https://console.cloud.google.com/networking/firewalls/list
Allow from all source IP: 0.0.0.0/0
For the following protocols/ports: tcp:5000,8080

Local environment setup (Linux)

The application has been tested using Debian 9 however it should be possible to run it using many other OS configurations

If you are running Windows, it is recommended that you setup Debian 9 in a virtual machine using VirtualBox

- If after you first install the OS sudo is not found then refer to: https://unix.stackexchange.com/questions/354928/bash-sudo-command-not-found
- Install Docker, instructions can be found at: https://docs.docker.com/install/linux/docker-ce/debian/
- Install Docker Compose, instructions can be found at: https://docs.docker.com/compose/install/
- If Git isn't installed, then do so using: sudo apt-get install git
- If you are using a virtual machine you may need to set up port forwarding to provide outside access using SSH or to the web app, if you are using VirtualBox:
- Find the IP of guest (this is the VM itself): ip addr show
- Find the IP of VirtualBox from host: ipconfig
- Go into settings for the VM and add a port forwarding rule under Network tab (port 22/5000)

Setting up the application

- Use git to clone the repo into your current directory: git clone https://github.com/comp3300-comp9900-term-3-2019/capstone-project-cheesecake.git
- Navigate into the directory containing VM_setup:
  cd VM_setup
- Now run the following commands (sudo may be required):
  docker-compose build
  docker-compose up -d
- By default, the backend will use port 5000, and the frontend will use port 8080

Application Management

The following containers are created:

Name	**Port**	**Purpose**
app	5000	Backend server of app
app-frontend	8080	Frontend server of app
postgres	5432	Database server
vm_setup_redis_1	6379	Redis server for queue

If you have changed the requirements.txt file or the Dockerfile then you must rebuild the image:

- docker-compose build
- docker-compose up -d

If you change the models, then you must recreate the database:

- docker-compose down
- docker-compose up -d

You can check the status of the containers using:

- docker ps

You can get the logs for a container using:

- docker logs [container-name]

Connect to a container using ssh:

- docker exec -i -t [container-name] /bin/bash

Remove no longer used images to free up space:

- docker system prune

If you would like to view emails (generally used for registration or password reset) then you will need to login to mailtrap: https://mailtrap.io

- username: daniel.hocking+cheesecake@gmail.com
- password: cheesecake123

User Manual (guide for using the application)

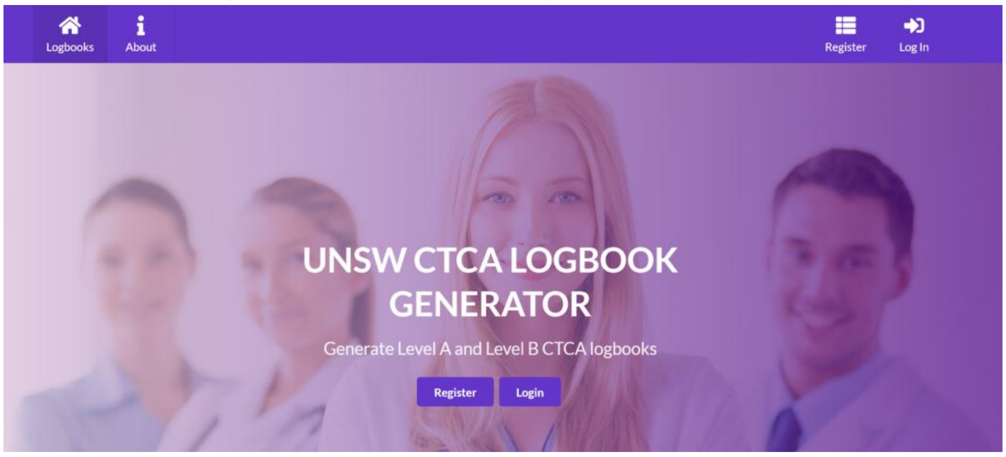
Main landing page

This section contains the instructions to access and navigate the application.
To access this application, user need to open any web browser and go to https://logbooks.sydney/
The webpage will appear as below.

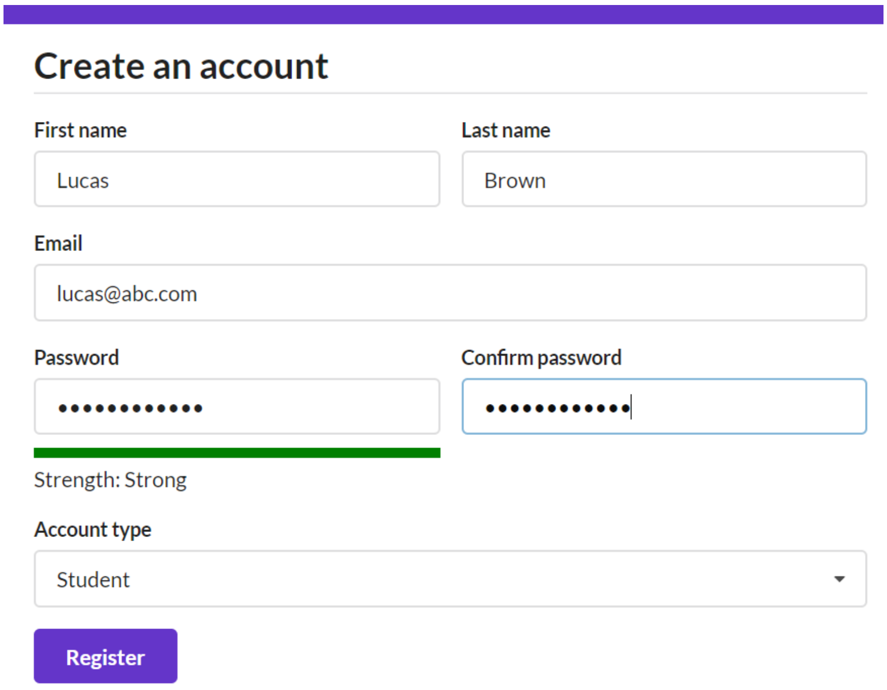
Register

Users can register themselves in the application by performing the below steps.

1. Click on either "Register" option on the top right corner or middle of the page.


Logbooks
About
Register
Log In
UNSW CTCA LOGBOOK GENERATOR
Generate Level A and Level B CTCA logbooks
Register
Login

2. Fill all details. Make sure to provide a strong password for account security and select account type as Student or Supervisor and click "Register" button at the bottom. The below example is how to register as a Student. Similar way user can register as Supervisor.



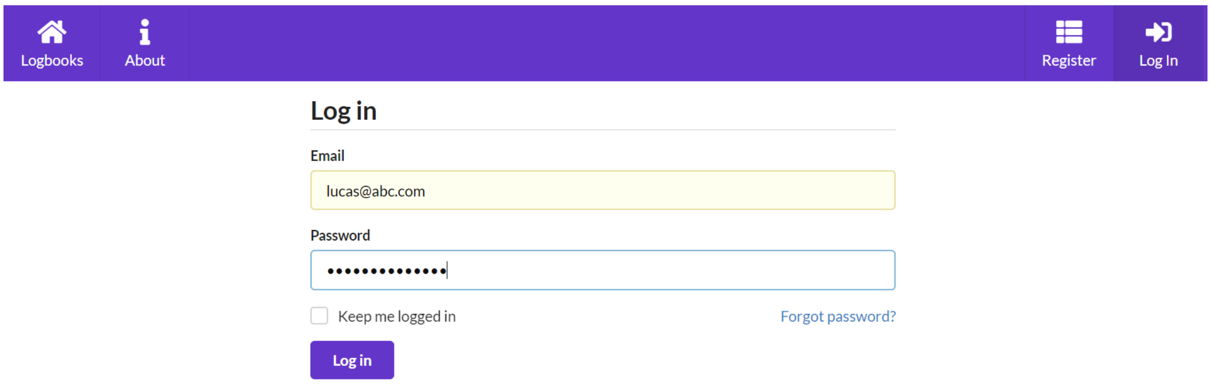
Create an account

First name: Lucas
Last name: Brown
Email: lucas@abc.com
Password: ••••••••••••
Confirm password: ••••••••••••
Strength: Strong
Account type: Student

3. User will receive a confirmation email where an account activation link will be present. Click on the link to activate account.

Log In

1. After user confirms from email, he/she can log in to the system by clicking on the "Log In" option at the top-right corner or middle of the webpage.
2. Log in using email and password what was used to register.



Log in

Email: lucas@abc.com

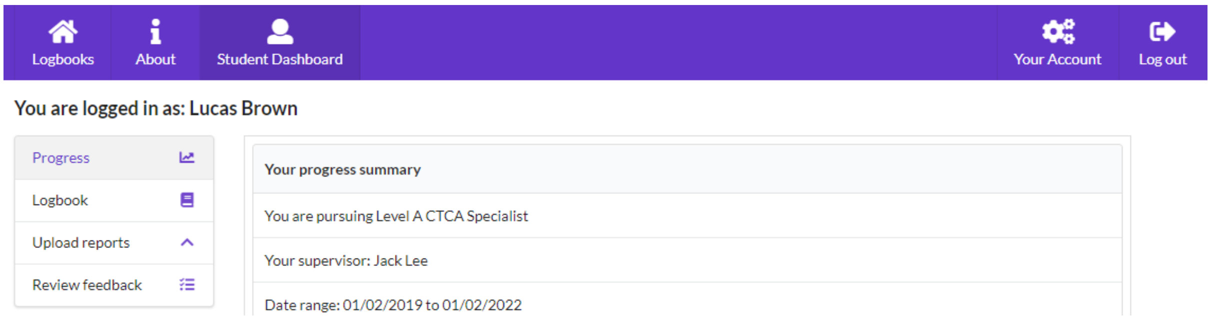
Password: ••••••••••••••

Keep me logged in

Forgot password?

Log in

After Log in, the webpage will look like this.



You are logged in as: Lucas Brown

Your progress summary

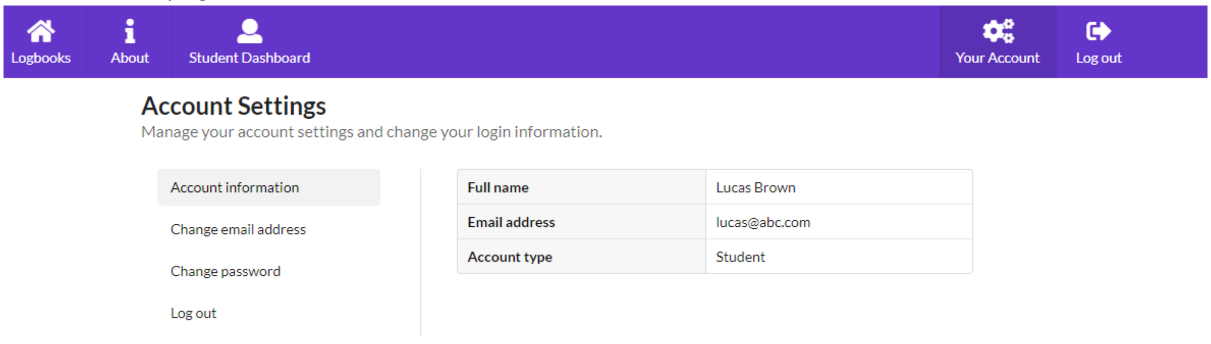
You are pursuing Level A CTCA Specialist

Your supervisor: Jack Lee

Date range: 01/02/2019 to 01/02/2022

Changing Account Details

User can change his/her email and password by clicking "Your Account" option at the top-right corner of the page.



Account Settings

Manage your account settings and change your login information.

Full name	Lucas Brown
Email address	lucas@abc.com
Account type	Student

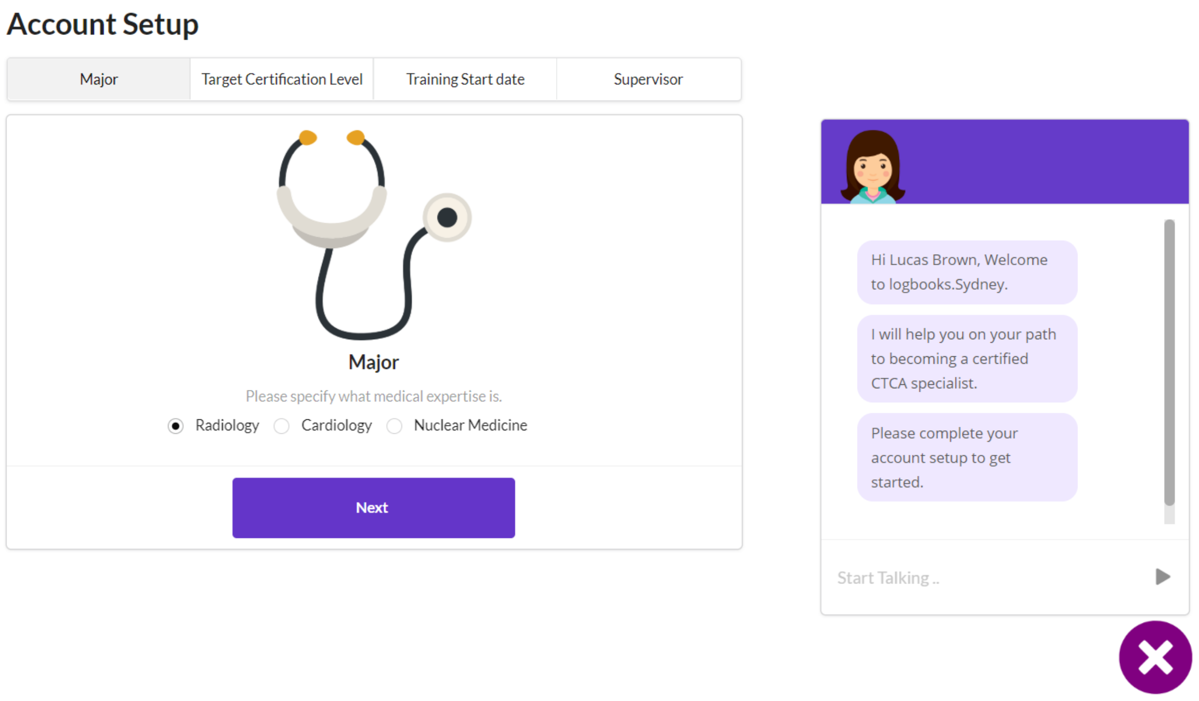
Log Out

User can log out from the application by clicking "Log out" option at the top-right corner of the webpage.

Student Navigation

Registration Wizard

If user is newly registered student, for the first time he/she will be prompted with a wizard to set up account details. There are four information (Major, Target Certification Level, Training Start date and Supervisor) need to be filled up. Chatbot will be there at right-bottom section of the page to help in case. Throughout the wizard, chatbot will keep informing what conditions student needs to fulfil to achieve the target certification.


Account Setup

Major | Target Certification Level | Training Start date | Supervisor

Major

Please specify what medical expertise is.

[● Radiology] ○ Cardiology ○ Nuclear Medicine

Next

Hi Lucas Brown, Welcome to logbooks.Sydney.

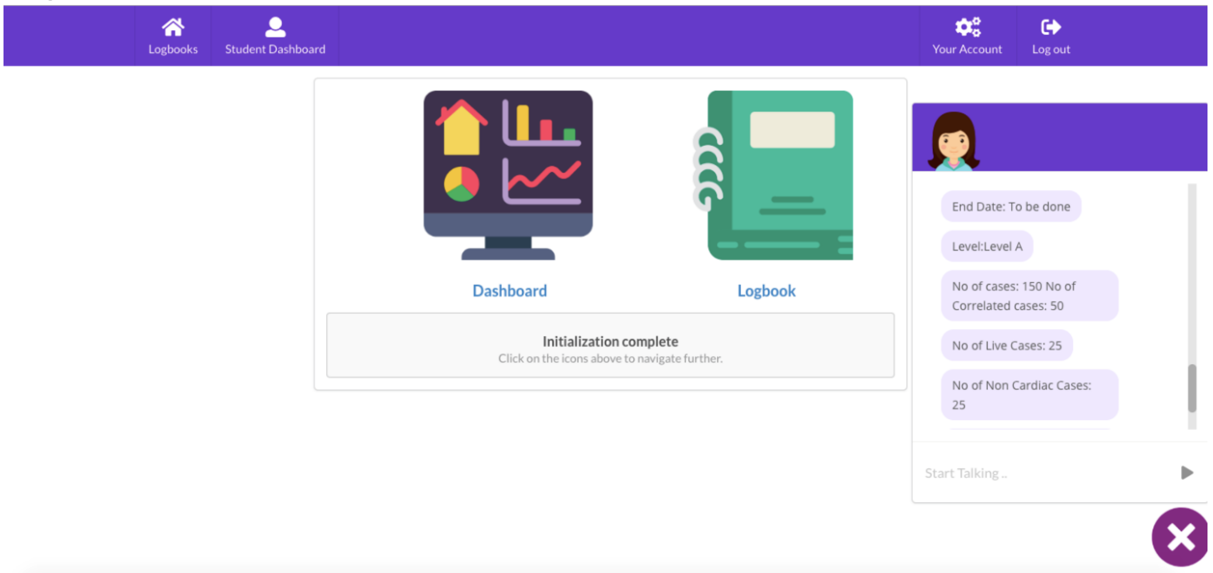
I will help you on your path to becoming a certified CTCA specialist.

Please complete your account setup to get started.

Start Talking ..

System Menu

After completing the wizard, user can directly navigate to Dashboard by clicking on the "Dashboard" link. If user directly wants to navigate to his/her logbook, user can do that by clicking on the "Logbook" link.



Initialization complete

Click on the icons above to navigate further.

End Date: To be done

Level:Level A

No of cases: 150 No of Correlated cases: 50

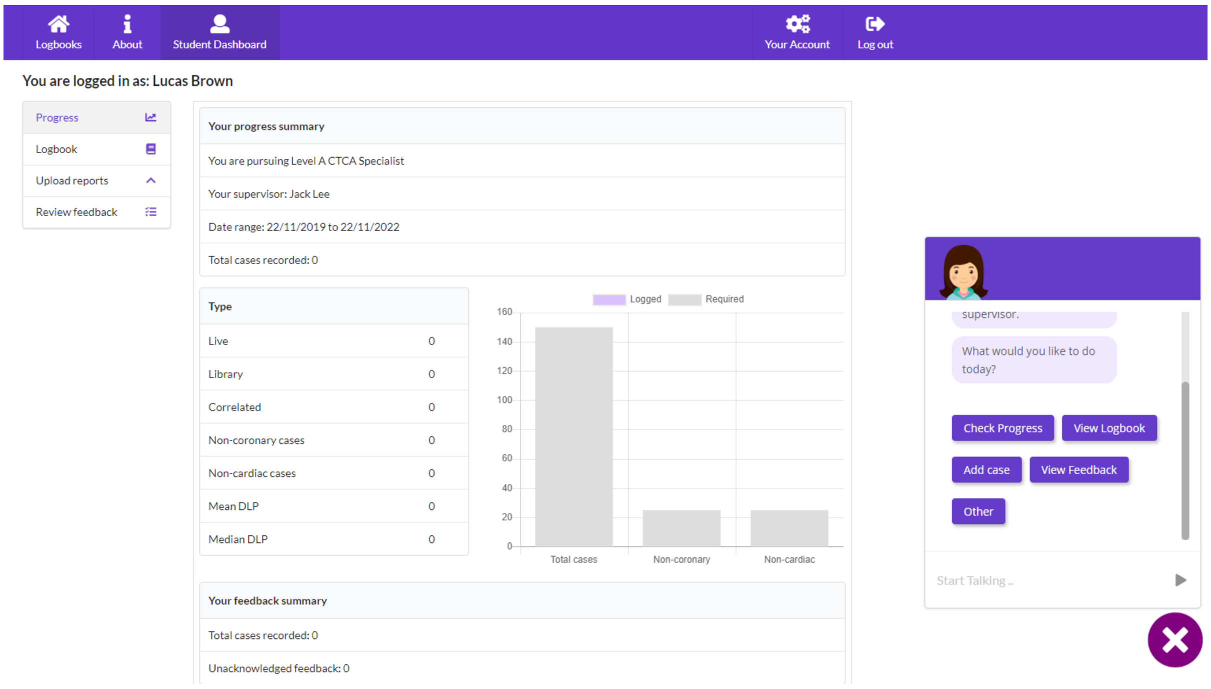
No of Live Cases: 25

No of Non Cardiac Cases: 25

Start Talking ..

Progress

Progress page is a dashboard for student to have an overview of his/her progress in the course required to achieve the target certification.
A newly registered student will not have any case logged yet. Therefore, the screen will appear as below, with all values to 0.



You are logged in as: Lucas Brown

Your progress summary

You are pursuing Level A CTCA Specialist

Your supervisor: Jack Lee

Date range: 22/11/2019 to 22/11/2022

Total cases recorded: 0

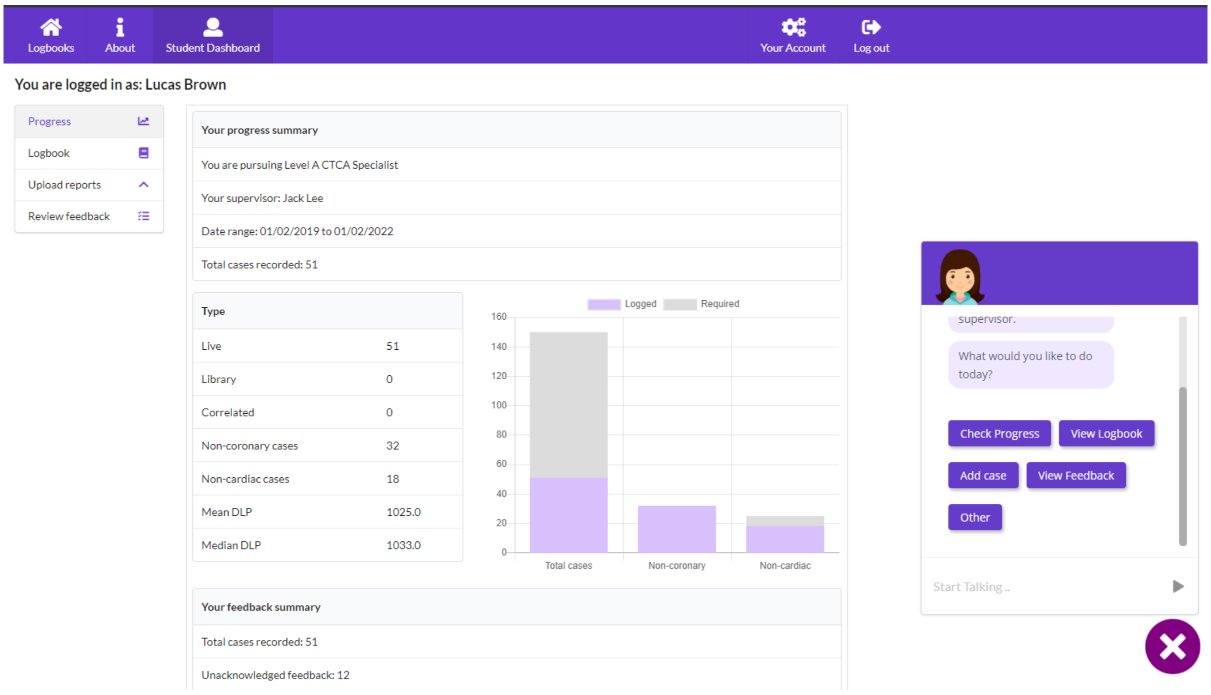
Type	
Live	0
Library	0
Correlated	0
Non-coronary cases	0
Non-cardiac cases	0
Mean DLP	0
Median DLP	0

Your feedback summary

Total cases recorded: 0

Unacknowledged feedback: 0

As the student progresses through his/her course and log more cases, the progress page will have values of his/her logbook summary.
Chatbot can answer progress related queries.



You are logged in as: Lucas Brown

Your progress summary

You are pursuing Level A CTCA Specialist

Your supervisor: Jack Lee

Date range: 01/02/2019 to 01/02/2022

Total cases recorded: 51

Type	
Live	51
Library	0
Correlated	0
Non-coronary cases	32
Non-cardiac cases	18
Mean DLP	1025.0
Median DLP	1033.0

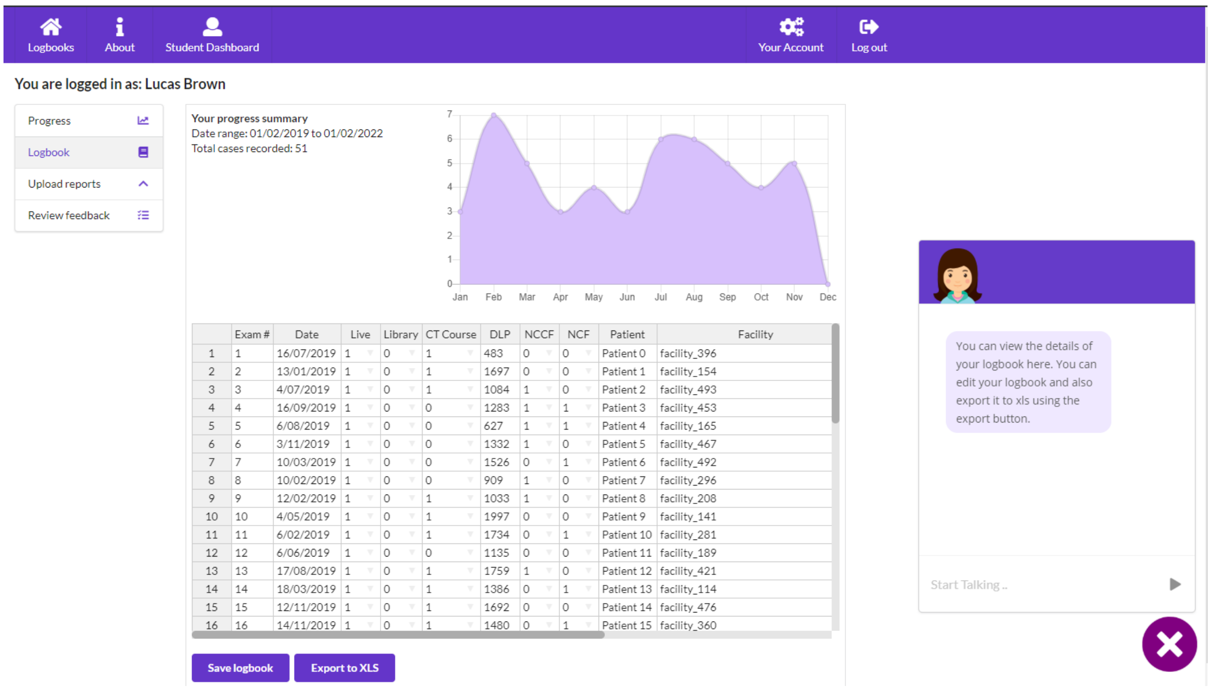
Your feedback summary

Total cases recorded: 51

Unacknowledged feedback: 12

Logbook

In this page, student can view his/her logbook. This has an excel-like interface and functionalities like edit-copy-paste, drag values etc. Once the student is ready to save the changes, he/she can do that by clicking on the "Save Logbook" button at the bottom.
Student can export the logbook by clicking on the "Export to XLS" button next to "Save Logbook" button.
This will generate an excel file in the format required to submit to have the certification.
Chatbot can help students with the navigation and queries related to logbook.



You are logged in as: Lucas Brown

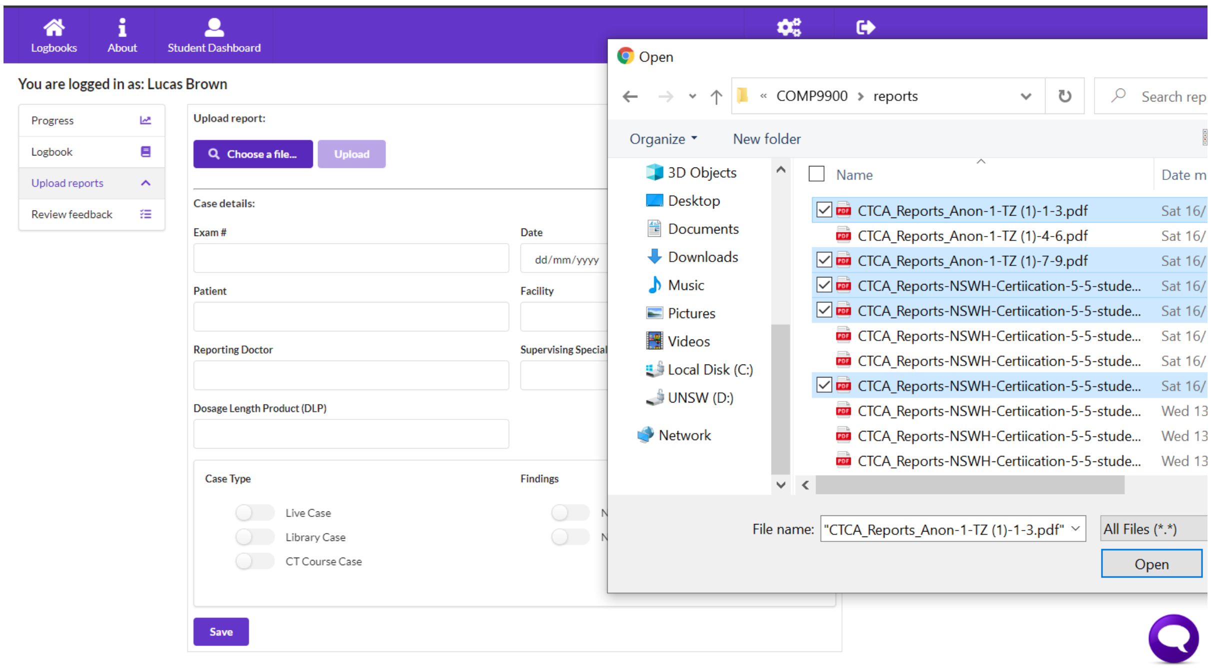
Your progress summary
Date range: 01/02/2019 to 01/02/2022
Total cases recorded: 51

	Exam #	Date	Live	Library	CT Course	DLP	NCCF	NCF	Patient	Facility
1	1	16/07/2019	1	0	1	483	0	0	Patient 0	facility_396
2	2	13/01/2019	1	0	1	1697	0	0	Patient 1	facility_154
3	3	4/07/2019	1	0	1	1084	1	0	Patient 2	facility_493
4	4	16/09/2019	1	0	0	1283	1	1	Patient 3	facility_453
5	5	6/08/2019	1	0	0	627	1	1	Patient 4	facility_165
6	6	3/11/2019	1	0	0	1332	1	0	Patient 5	facility_467
7	7	10/03/2019	1	0	0	1526	0	1	Patient 6	facility_492
8	8	10/02/2019	1	0	0	909	1	0	Patient 7	facility_296
9	9	12/02/2019	1	0	1	1033	1	0	Patient 8	facility_208
10	10	4/05/2019	1	0	1	1997	0	0	Patient 9	facility_141
11	11	6/02/2019	1	0	1	1734	0	1	Patient 10	facility_281
12	12	6/06/2019	1	0	0	1135	0	0	Patient 11	facility_189
13	13	17/08/2019	1	0	1	1759	1	0	Patient 12	facility_421
14	14	18/03/2019	1	0	1	1386	0	1	Patient 13	facility_114
15	15	12/11/2019	1	0	1	1692	0	0	Patient 14	facility_476
16	16	14/11/2019	1	0	1	1480	0	1	Patient 15	facility_360

You can view the details of your logbook here. You can edit your logbook and also export it to xls using the export button.

Upload Reports

In this page, student can upload single or multiple reports of the cases he/she has performed. Student can select files from his/her computer by clicking “Choose a file” button and then select intended files and then “Upload” button.



	Name	Date m
✓	CTCA_Reports_Anon-1-TZ (1)-1-3.pdf	Sat 16/
	CTCA_Reports_Anon-1-TZ (1)-4-6.pdf	Sat 16/
✓	CTCA_Reports_Anon-1-TZ (1)-7-9.pdf	Sat 16/
✓	CTCA_Reports-NSWH-Certiication-5-5-stude...	Sat 16/
✓	CTCA_Reports-NSWH-Certiication-5-5-stude...	Sat 16/
	CTCA_Reports-NSWH-Certiication-5-5-stude...	Sat 16/
	CTCA_Reports-NSWH-Certiication-5-5-stude...	Sat 16/
✓	CTCA_Reports-NSWH-Certiication-5-5-stude...	Sat 16/
	CTCA_Reports-NSWH-Certiication-5-5-stude...	Wed 13
	CTCA_Reports-NSWH-Certiication-5-5-stude...	Wed 13
	CTCA_Reports-NSWH-Certiication-5-5-stude...	Wed 13

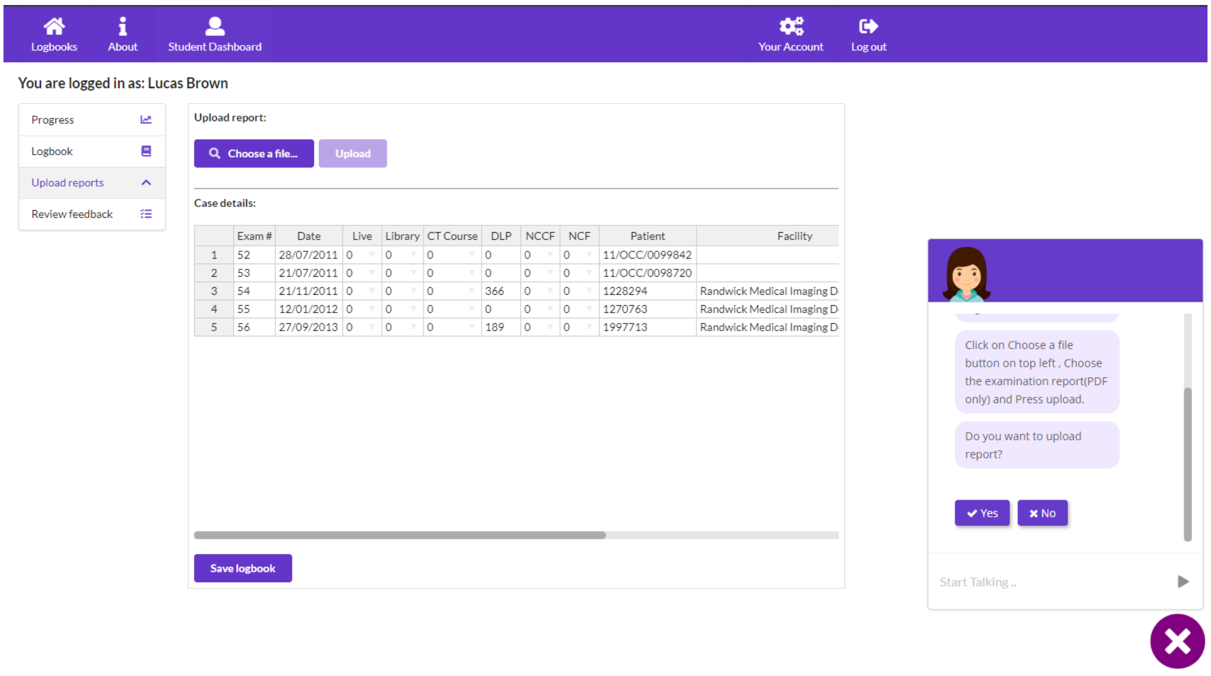
File name: "CTCA_Reports_Anon-1-TZ (1)-1-3.pdf"

All Files (*.*)

Open

Application will extract information from the pdf reports and create an excel-like interface for easy edit. Once student is ready to save the changes, He/she can do so by clicking “Save logbook” button at the bottom.

Chatbot will be helping the student by providing options to upload report. Clicking on the “Yes” button in the chatbot window will open the option to upload files.



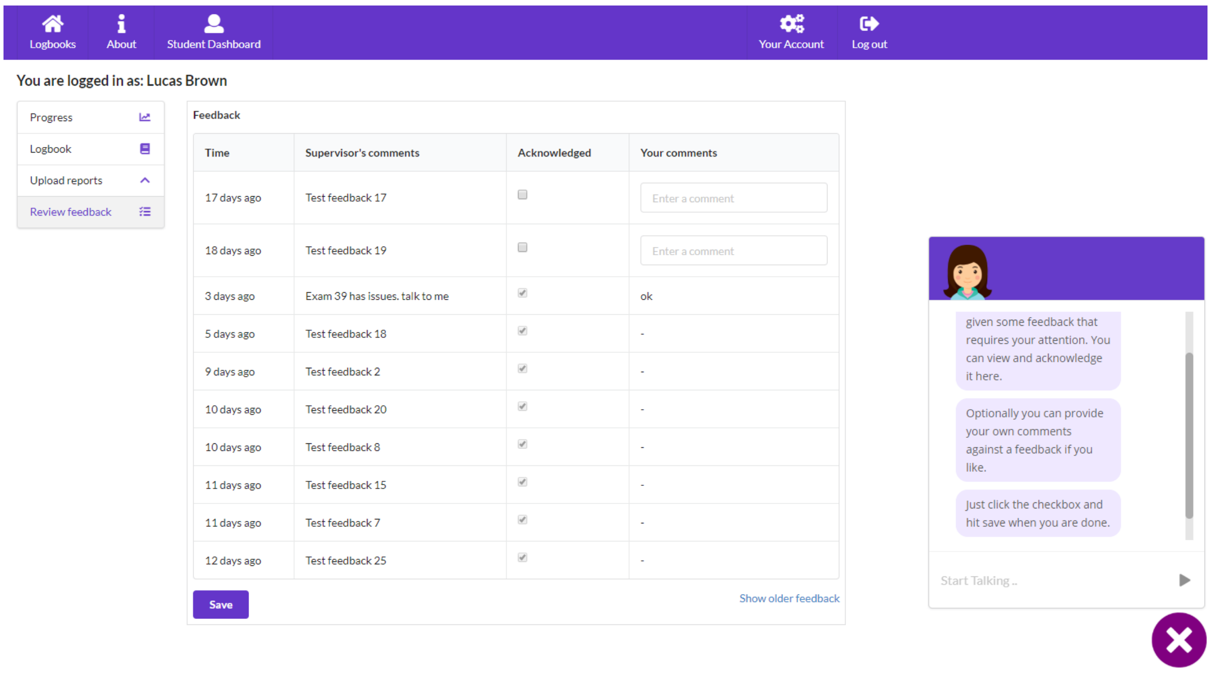
	Exam #	Date	Live	Library	CT Course	DLP	NCCF	NCF	Patient	Facility
1	52	28/07/2011	0	0	0	0	0	0	11/OCC/0099842	
2	53	21/07/2011	0	0	0	0	0	0	11/OCC/0098720	
3	54	21/11/2011	0	0	0	366	0	0	1228294	Randwick Medical Imaging D
4	55	12/01/2012	0	0	0	0	0	0	1270763	Randwick Medical Imaging D
5	56	27/09/2013	0	0	0	189	0	0	1997713	Randwick Medical Imaging D

Click on Choose a file button on top left , Choose the examination report(PDF only) and Press upload.

Do you want to upload report?

Review Feedback

In this page student can check the feedback provided by his/her supervisor. Unacknowledged feedbacks will appear on top, sorted by feedback date. Acknowledged feedbacks will appear after unacknowledged feedbacks, sorted by feedback date.
Student can acknowledge the feedback by checking off the “Acknowledged” checkboxes. Student can put his/her comment for each feedback.
Finally, student can save these changes by clicking the “Save” button at the bottom.
Chatbot will help the student by informing how to use this page.



You are logged in as: Lucas Brown

Feedback

Time	Supervisor's comments	Acknowledged	Your comments
17 days ago	Test feedback 17	☐	Enter a comment
18 days ago	Test feedback 19	☐	Enter a comment
3 days ago	Exam 39 has issues. talk to me	☑	ok
5 days ago	Test feedback 18	☑	-
9 days ago	Test feedback 2	☑	-
10 days ago	Test feedback 20	☑	-
10 days ago	Test feedback 8	☑	-
11 days ago	Test feedback 15	☑	-
11 days ago	Test feedback 7	☑	-
12 days ago	Test feedback 25	☑	-

given some feedback that requires your attention. You can view and acknowledge it here.

Optionally you can provide your own comments against a feedback if you like.

Just click the checkbox and hit save when you are done.

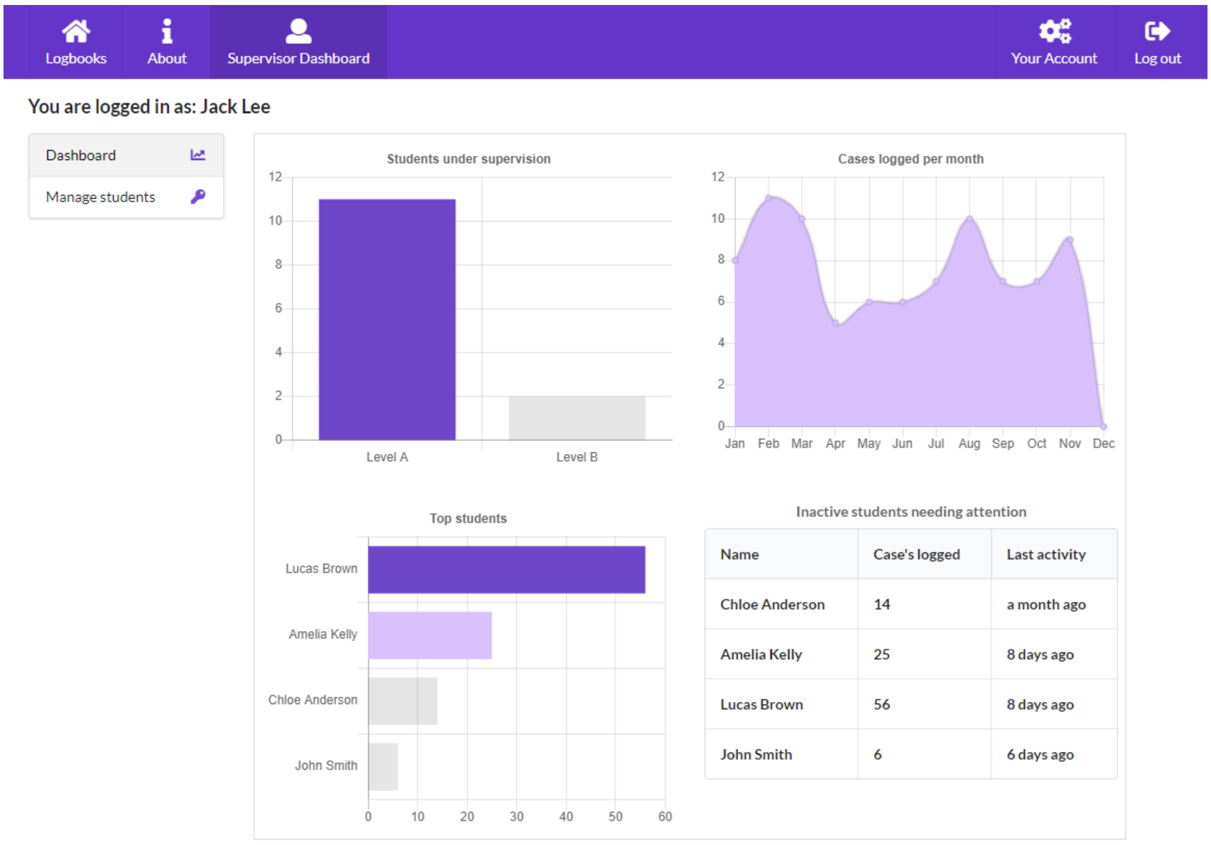
Supervisor Navigation

Supervisor accounts have two main pages – Dashboard and Manage Students.

Dashboard

Dashboard page is for proving an overall view of all students to the supervisor. This page has four sections.

- An overview of how many students and pursuing each level-A and level-B certifications
- Total cases per month
- Top students
- Inactive students who need attention



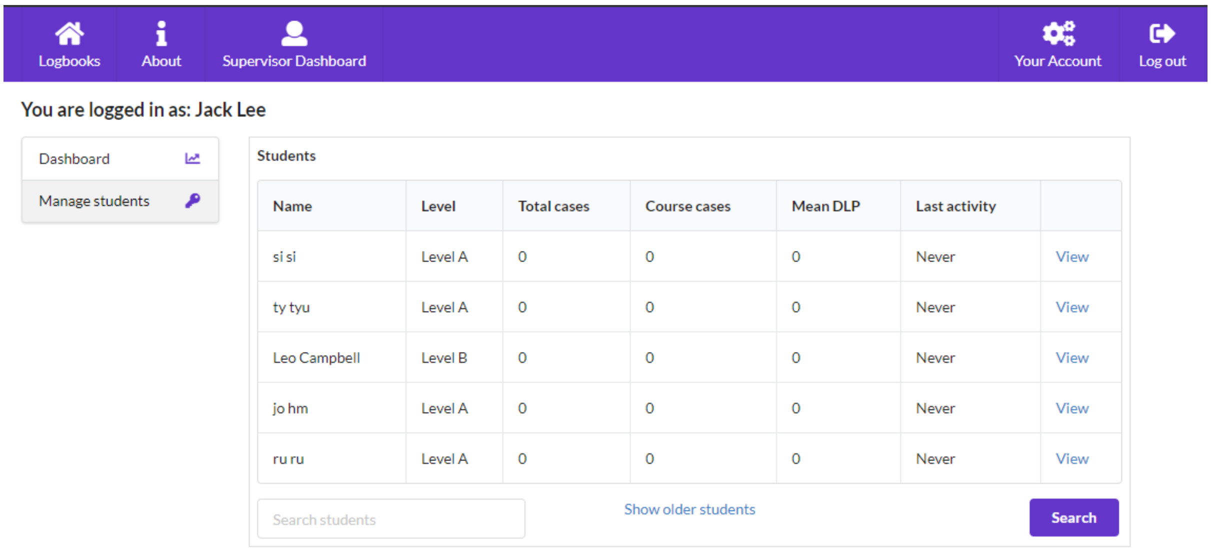
You are logged in as: Jack Lee

Inactive students needing attention

Name	Case's logged	Last activity
Chloe Anderson	14	a month ago
Amelia Kelly	25	8 days ago
Lucas Brown	56	8 days ago
John Smith	6	6 days ago

Manage Students

This page shows all the students under supervision. Supervisor can search individual student and see their details.



You are logged in as: Jack Lee

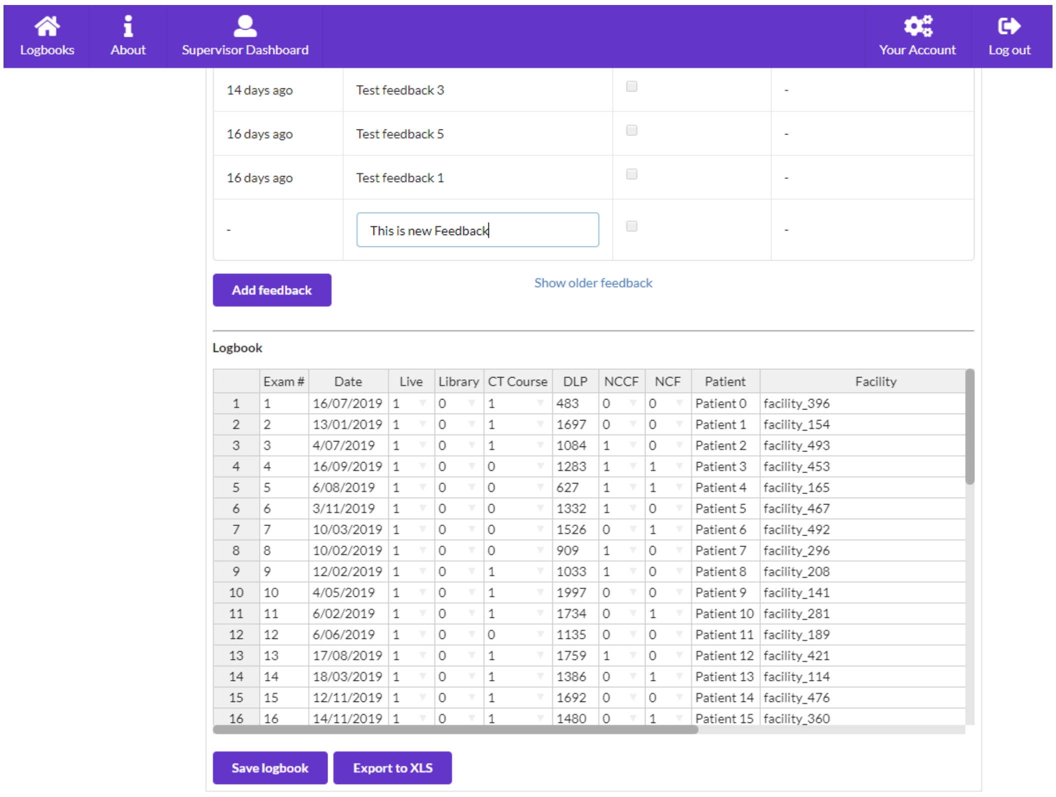
Students

Name	Level	Total cases	Course cases	Mean DLP	Last activity	
si si	Level A	0	0	0	Never	View
ty tyu	Level A	0	0	0	Never	View
Leo Campbell	Level B	0	0	0	Never	View
jo hm	Level A	0	0	0	Never	View
ru ru	Level A	0	0	0	Never	View

Supervisor can view each student's logbook by clicking the "view" option at the right of each student record.

This will open the student's logbook. Supervisor can also edit, save or export as excel if wants.

Supervisor can provide feedback to students which will appear in students' "Review Feedback" page.



14 days ago	Test feedback 3		-
16 days ago	Test feedback 5		-
16 days ago	Test feedback 1		-
-	This is new Feedback		-

Logbook

	Exam #	Date	Live	Library	CT Course	DLP	NCCF	NCF	Patient	Facility
1	1	16/07/2019	1	0	1	483	0	0	Patient 0	facility_396
2	2	13/01/2019	1	0	1	1697	0	0	Patient 1	facility_154
3	3	4/07/2019	1	0	1	1084	1	0	Patient 2	facility_493
4	4	16/09/2019	1	0	0	1283	1	1	Patient 3	facility_453
5	5	6/08/2019	1	0	0	627	1	1	Patient 4	facility_165
6	6	3/11/2019	1	0	0	1332	1	0	Patient 5	facility_467
7	7	10/03/2019	1	0	0	1526	0	1	Patient 6	facility_492
8	8	10/02/2019	1	0	0	909	1	0	Patient 7	facility_296
9	9	12/02/2019	1	0	1	1033	1	0	Patient 8	facility_208
10	10	4/05/2019	1	0	1	1997	0	0	Patient 9	facility_141
11	11	6/02/2019	1	0	1	1734	0	1	Patient 10	facility_281
12	12	6/06/2019	1	0	0	1135	0	0	Patient 11	facility_189
13	13	17/08/2019	1	0	1	1759	1	0	Patient 12	facility_421
14	14	18/03/2019	1	0	1	1386	0	1	Patient 13	facility_114
15	15	12/11/2019	1	0	1	1692	0	0	Patient 14	facility_476
16	16	14/11/2019	1	0	1	1480	0	1	Patient 15	facility_360

References:

[1] Kruchten, Philippe (1995, November). Architectural Blueprints — The “4+1” View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50
[2] https://cloud.google.com/solutions/document-ai/
[3] https://stanfordnlp.github.io/stanfordnlp/
[4] Molich, R., and Nielsen, J. (1990). Improving a human–computer dialogue, Communications of the ACM 33, 3 (March), 338–348