

1. Project - NDIS Positive Behaviour Support (code: ND)	2
1.1 Links	3
1.2 Development	4
1.2.1 Deployment Workflow	5
1.2.2 Implementation	10
1.2.2.1 Canvas connection and receive documents	11
1.2.2.2 Database Connection & Access	12
1.2.2.3 Database table reading	13
1.2.2.4 Data Insertion & Update	14
1.2.2.5 PDF Extraction	16
1.2.2.6 PDF Extraction2	20

Project - NDIS Positive Behaviour Support (code: ND)

This project is developed to enhance practitioners' ability to provide quality Behaviour Support Plans (BSPs) that are consistent with legislation, policy, and good clinical practice, i.e., report requirements of the National Disability Insurance Scheme (NDIS). The proposed methodology is to provide education and supporting resources through the Canvas LMS platform to upskill the relevant workforces and integrate artificial intelligence (AI) to allow the workforce to self-examine through the provision of AI-generated feedback.

For more detailed information on this project visit [Project Description](#), [Architecture](#), [Links](#), [Task Tracking](#) and [Quality Control](#).

Trello board link: <https://trello.com/b/5mjfxNPo/comp900822022s2ndboxjelly>

Github link: https://github.com/COMP90082-2022-SM2/ND_boxjelly

Development team (ND-BoxJelly)

Name	Photo	Email
Yuling Zheng		yulingz3@student.unimelb.edu.au
Sihao SHEN		sihaos@student.unimelb.edu.au
Hanyu ZHU		hazhu2@student.unimelb.edu.au
Yang Song		yangsong2@student.unimelb.edu.au

Client

Name	Contact
Micheal Kirley	

Project supervisor

Samodha pallewatta	samodha.pallewatta@unimelb.edu.au
--------------------	--

Recent space activity



Yuling Zheng

[Timeline](#) updated 21 minutes ago • [view change](#)

[Testing](#) updated 23 minutes ago • [view change](#)

[Quality Control](#) updated 23 minutes ago • [view change](#)

[Product Backlog](#) updated 24 minutes ago • [view change](#)

[Architecture](#) updated 25 minutes ago • [view change](#)

Space contributors

- [Yuling Zheng](#) (21 minutes ago)
- [Yang SONG](#) (an hour ago)
- [Sihao SHEN](#) (13 hours ago)
- [Hanyu ZHU](#) (18 hours ago)
- [Minyi Chen](#) (49 days ago)
- ...

Links

- [Trello Workspace](#)
- [Github repo](#)

Development

[Deployment Workflow](#)

[Implementation](#)

Deployment Workflow

Deployment Requirements

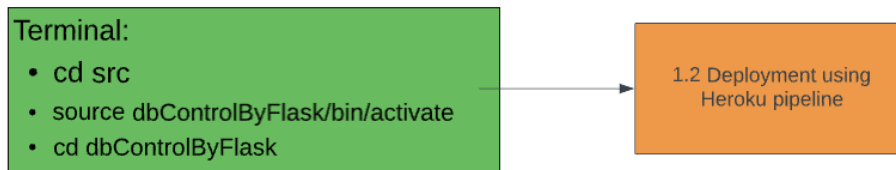
- Heroku account (Free)
- Heroku CLI (create and manage Heroku Applications from the terminal). Documentation: <https://devcenter.heroku.com/articles/heroku-cli>
- MySQL Workbench

Workflow

Initialization / Set-Up Stage



Development & Testing Stage



1. Python Flask App Deployment

1.1 Project Initialization

1.1.1 Environment & Repository Set-up

- `cd src`
- Set up a Python Virtual Environment: `python3 -m venv dbControlByFlask`
- Activate the Virtual Environment: `source dbControlByFlask/bin/activate`
- `cd dbControlByFlask`
- Install Dependencies: `pip3 install Flask`
- Install Heroku plugins
 - `heroku plugins:install buildpack-registry`
 - `heroku plugins:install buildpacks`
- Heroku Python & Java buildpacks required:
 - `heroku buildpacks:set heroku/python` (This detects your app as Python)
 - `heroku buildpacks:add heroku/java` ('Tabula' library requires Java runtime on the environment)
- Initialize the Git Repository: `git init`
- `heroku login`
- Clone the repo: `heroku git:clone -a db-control-by-flask2`

1.1.2 Heroku Deployment Set-up

- Create a text file listing project dependencies/packages: `pip3 freeze > requirements.txt`
- Create Procfile tell Heroku how the Python app will be running: `nano Procfile -use Gunicorn` (a Web Server Gateway Interface HTTP Server) as it's compatible with Flask

Project structure

dbControlByFlask/

|

|----- venv /

|----- main.py

|----- requirements.txt

|----- Procfile

Tutorial: <https://dev.to/techparida/how-to-deploy-a-flask-app-on-heroku-heb>

App Object – app = Flask(__name__) to represent the app

View to Route - @app.route("/process")

1.2 Deploy the app using Heroku pipelines

Use Git to manage changes)

- heroku login
- git add .
- git commit -am "make it better"
- git branch: to check which branch you are currently on

```
(dbControlByFlask) sophiezheng@Sophies-MacBook-Air dbControlByFlask % git branch
* Sophie
master
```

- git push heroku Sophie:master (Push the git repo to the Heroku remote 'master' branch)

How it looks in Terminal:

```
(dbControlByFlask) sophiezheng@Sophies-MacBook-Air dbControlByFlask % git add .
(dbControlByFlask) sophiezheng@Sophies-MacBook-Air dbControlByFlask % git commit -am "make it better"
[Sophie 26cccac] make it better
1 file changed, 2 insertions(+), 2 deletions(-)
(dbControlByFlask) sophiezheng@Sophies-MacBook-Air dbControlByFlask % git push heroku Sophie:master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
```

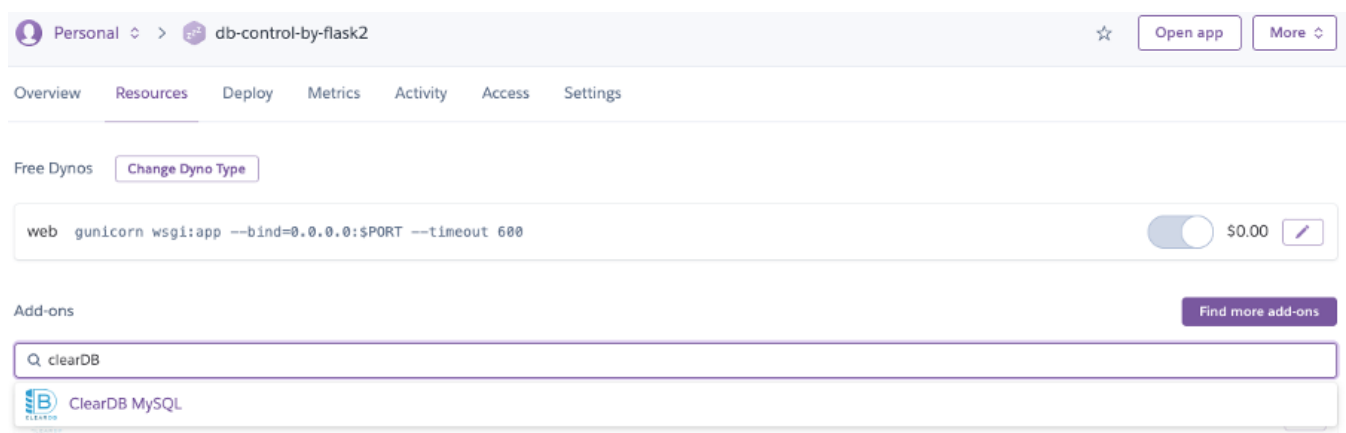
If builds successfully, it will show the URL: <https://db-control-by-flask2.herokuapp.com/process>

Debug

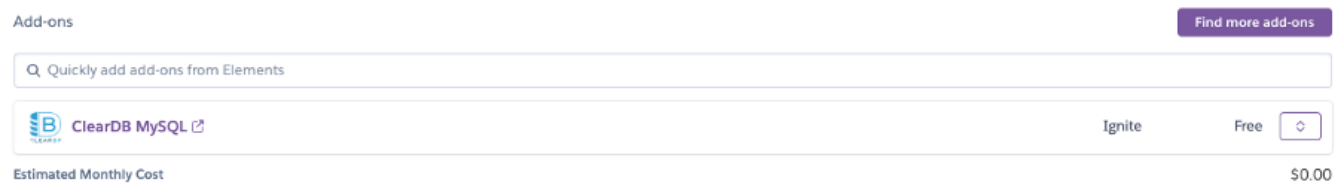
- heroku logs --tail

2. Database Deployment

2.1 Deploy MySQL database on Heroku with ClearDB MySQL Add-on



If successfully added ClearDB MySQL on the app:



Config Vars: **CLEARDB_DATABASE_URL** -to access the remote MySQL server

Heroku config | grep CLEARDB

```
(dbControlByFlask) sophiezheng@Sophies-MacBook-Air dbControlByFlask % heroku config | grep CLEARDB
> Warning: heroku update available from 7.59.2 to 7.63.4.
CLEARDB_DATABASE_URL: mysql://bcad2e6d9226f9:318b6978@us-cdbr-east-06.cleardb.net/heroku_f7c500f08d32a48?reconnect=true
```

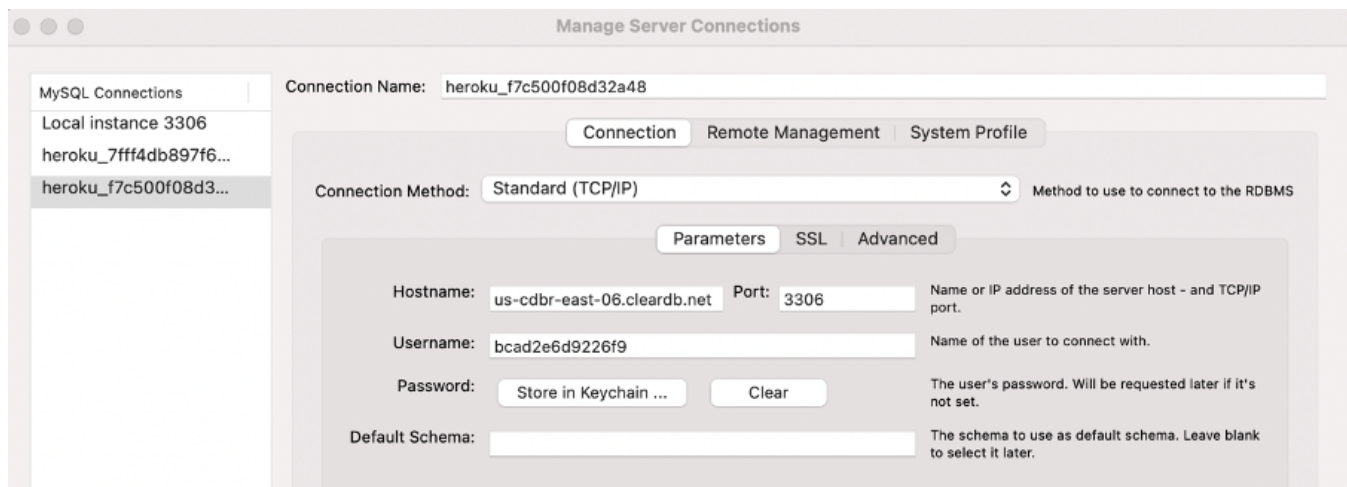
- user = 'bcad2e6d9226f9'
- host = 'us-cdbr-east-06.cleardb.net'
- password = '318b6978'
- database = 'heroku_f7c500f08d32a48'

Documentation: <https://devcenter.heroku.com/articles/cleardb>

The graphical tool we used to view data: **MySQL Workbench**. Other recommended graphical tools include Sequel Pro for Mac OS X or Navicat.

2.2 Connect MySQL on a local computer

Connect to the database on MySQL Workbench info as below.



3. Database Deployment(Alternative)

As mentioned above, we currently use clearDB as the main database for the application. In case of the application has a higher requirement for a database, we create an Azure Mysql database as an alternative.

The reason we use Azure Mysql as the alternative is due to the following features:

- It provides redundancy and high availability of data within the same zone. Moreover, it offers maximum control to the user and will allow you to select the time of the maintenance window.
- It provides high Data protection with its automatic backups and data restoration services that can store the data for up to a month.
- It automates the patching and maintenance of the Azure processing hardware and database engine. This way you will experience a secure and up-to-date service.
- It helps you in optimizing your spending by providing Cost controls and a low-cost expandable SKU that has the ability to start or stop the server anytime. Moreover, the Azure Database for MySQL comes in a pay-as-you-go pricing model.

- You will get Enterprise-grade security for your data, and its privacy feature will protect your sensitive data both when at rest and when in motion.
- It offers automatic data Monitoring that simplifies your task of managing large-scale deployments. Furthermore, it also offers Industry-leading support experience. to guide you through day-to-day tasks

3.1 Deploy MySQL database on Azure Mysql

Connect
 View process list
 Delete
 Reset password
 Restore
 Restart
 Stop
 Refresh
 Feedback

^ Essentials JSON View

Subscription (move)	: Azure subscription 1	Server name	: comp90082.mysql.database.azure.com
Subscription ID	: ab42405b-e188-4cda-a3b9-314e6cb7ecb4	Server admin login name	: project_admin
Resource group (move)	: COMP90082-project	Configuration	: Burstable, 81ms, 1 vCores, 2 GiB RAM, 20 GiB storage
Status	: Available	MySQL version	: 5.7
Location	: Australia East	Availability zone	: 3
		Created On	: 2022-10-02 11:10:49.5692172 UTC

Azure Mysql config

- host: 'comp90082.mysql.database.azure.com'
- user: 'project_admin'
- password: 'comp90082@@@'
- database: 'test'
- ssl-mode: 'require'

Documentation: <https://learn.microsoft.com/zh-cn/azure/mysql/flexible-server/>

3.2 Connect MySQL on a local computer

Connect to the database on MySQL Workbench info as below.


MySQL Workbench

To connect with MySQL workbench client, follow the steps below.

1. Click the + symbol in the **MySQL Connections** tab to add a new connection.
2. Enter a name for the connection in the **Connection name** field.
3. Select **Standard (TCP/IP)** as the Connection Type.
4. Enter **comp90082.mysql.database.azure.com** in hostname field.
5. Enter **project_admin** as username and then enter your **Password**.
6. Go to the **SSL tab** and update the Use SSL field to Require.
7. In the **SSL CA File** field, enter the file location of the **DigiCertGlobalRootCA.crt.pem** file.
8. Click **Test Connection** to test the connection.
9. If the connection is successful, click **OK** to save the connection.

Connection Name:

Connection Remote Management System Profile

Connection Method:  Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: Port: Name or IP address of the server host - and port.

Username: Name of the user to connect with.

Password: The user's password. Will be requested later not set.

Default Schema: The schema to use as default schema. Leave to select it later.

Implementation

[Canvas connection and receive documents](#)

[Database Connection & Access](#)

[Database table reading](#)

[Data Insertion & Update](#)

[PDF Extraction](#)

[PDF Extraction2](#)

Canvas connection and receive documents

- Through the communication with the client (9.16 zoom meeting), the client made it clear that the project did not need to be associated with the Canvas system (This project is more like a 'proof of concept' to illustrate that small components of the project are feasible).

1. Canvas API

How to access Canvas API: <https://github.com/agogear/canvasdashboards/blob/main/docs/CanvasAPI.md>

in Terminal (Mac) or Command Prompt (Windows) use this to connect to Canvas:

- **curl https://canvas.lms.unimelb.edu.au/api/v1/users/self -H "Authorization: Bearer \$token"**

2. Python

2.1 Connection

- **from canvasapi import Canvas**
- **canvas = Canvas(API_URL, API_KEY) # API_URL = https://canvas.lms.unimelb.edu.au**

2.2 Get the link to download all documents

- **zip_url = "https://canvas.lms.unimelb.edu.au/courses/"+course_id+"/assignments/"+asgmt_id+"/submissions?zip=1"**

2.3 Receive documents

When calling the zip_url in the website it gets redirected to a canvas session to login in.

Solution: 1. GET the URL and do not follow the redirection

2. Obtain the redirect location, but don't call it. Call most of its location so that it looks like location/view?theme=dark
3. Replace the location/view?theme=dark with location/annotated.pdf and then POST to that address.
4. GET location/annotated.pdf/is_ready until it returns { ready:true }.
5. GET location/annotated.pdf and save the file.

link: <https://community.canvaslms.com/t5/Canvas-Question-Forum/Download-annotated-submissions-via-API/m-p/153466>

Database Connection & Access

MySQL database access(clearDB)

- uses `mysql.connector` to access MySQL database.
- connection parameters can be found when deploying ClearDB.

```
mydb = mysql.connector.connect(  
    user = 'bcad2e6d9226f9',  
    host = 'us-cdbr-east-06.cleardb.net',  
    password = '318b6978',  
    database = 'heroku_f7c500f08d32a48',  
    connect_timeout=1000  
)
```

Check table existing in the database(clearDB)

- uses `create_engine`, `inspect` to check whether tables have existed in the database
- `engine = create_engine('mysql://bcad2e6d9226f9:318b6978@us-cdbr-east-06.cleardb.net/heroku_f7c500f08d32a48')`
`inspector=inspect(engine)`
`if inspector.has_table('user') == False: # if tables haven't created yet, create all the tables based on db.model`
`db.create_all()`

Connections for SQL query execution(clearDB)

- use `db cursor()` to make connections for executing SQL queries
- `cur = mydb.cursor()`

MySQL database connection test(Azure Mysql)

```
import mysql.connector  
from mysql.connector import errorcode  
  
config = {  
    'host': 'comp90082.mysql.database.azure.com',  
    'user': 'project_admin',  
    'password': 'comp90082@@',  
    'database': 'test'  
}  
  
try:  
    conn = mysql.connector.connect(**config)  
    print("Connection established")  
except mysql.connector.Error as err:  
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:  
        print("Something is wrong with the user name or password")  
    elif err.errno == errorcode.ER_BAD_DB_ERROR:  
        print("Database does not exist")  
    else:  
        print(err)  
else:  
    cursor = conn.cursor()  
  
cursor.execute("DROP TABLE IF EXISTS inventory;")  
print("Finished dropping table (if existed).")
```

Database table reading

Executes SQL SELECT query using the `cursor.execute()` method.

Uses Python's cursor class methods `fetchall()`/`fetchone()` to retrieve contents from the database tables

Considering that data scientists are not familiar with databases, we provide an API in `get_user_information()` to help them get the latest pdf of the specified user and return the content of the pdf in JSON format. Data scientists can obtain the specified user data by directly adding `getUserInfo/<user_id>` at the end of the URL. Then, the API will return a JSON object containing all the information in the pdf.

```
@app.route("/getUserInfo/<id>", methods=["POST", "GET"])
def get_user_information(id):
    user_id = int(id)
    return getAll(cur, mydb, user_id)
```

The data reading is in `read_db.py`. We divide a PDF into 6 parts according to the page marked in the pdf and use something like "page1", "page2" as the key. Each page internally uses the table name in the database as an index, which can view the related data association through the Domain Diagram in the Architecture. For tables that contain multiple rows, we convert each row into a dictionary and place these side-by-side contents in a list. Data scientists can use python to quickly convert such tabular content into dataframe format.

The following figure is an example of page2.

Data Insertion & Update

Each user has a unique user_id in the database to determine the identity. At the same time, we divided a pdf into some tables, and each table has a column of user_id attributes to determine the owner of the pdf. These tables are unique in the pdf, and for those that are not unique in the pdf, we will use these unique tables for the association.

Let's take the content of the page2 in the pdf as an example. We store the entire table in the figure below in the assessment table and assign it a unique id. We store the answers to the last two questions in the assessment table, while the answer to the first question has multiple lines. So we store these non-unique data in another table named persons_consulted, and the id in assessment is used as the foreign key named as assessment_id to associate the two tables.

Original table in page2:

PAGE 2 – Assessments and Data Gathering

- This page contains the relevant information regarding the assessments undertaken, and the data gathered to inform PBSP development. This information only needs to be entered once by the user.

NOTE: We will need to develop checklist items to assess the content in this section. This section will mainly be looking for the absence of content (i.e., information is not provided).

Persons consulted to prepare this PBSP (add/remove rows as required)	
Who are they?	How were they consulted?
Adult with disability	Direct observation
Family of adult	Interview – face to face
Guardian	Interview – face to face
Neuropsychiatrist	Interview – telephone
Speech and language pathologist	Interview – telephone
Graduate occupational therapist	Interview – telephone
General practitioner	Interview – telephone
Service manager	Interview – telephone
House coordinator	Interview – telephone
Direct support staff	Interview – telephone
Outline the behavioural assessment approaches implemented to develop this PBSP	
A functional behaviour assessment that included the use of the Contextual Assessment Inventory, the Functional Assessment Interview, scatterplots and ABC note cards, and semi-structured interviews.	
Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP	
Comprehensive Health Assessment Program (CHAP) by GP, Communication assessment by speech and language pathologist, mental health assessment by neuropsychiatrist, and occupational therapy assessment by occupational therapist.	

assessment:

	id	user_id	behaviouralAssessment	nonBehaviouralAssessment	score
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	A functional behaviour assessment that included ...	Comprehensive Health Assessment Program (CHAP) b...	NULL

persons_consulted

		id	who	how	assessment_id
<input type="checkbox"/>	Edit Copy Delete	1	Adult with disability	Direct observation	1
<input type="checkbox"/>	Edit Copy Delete	2	Family of adult	Interview – face to face	1
<input type="checkbox"/>	Edit Copy Delete	3	Guardian	Interview – face to face	1
<input type="checkbox"/>	Edit Copy Delete	4	Neuropsychiatrist	Interview – telephone	1
<input type="checkbox"/>	Edit Copy Delete	5	Speech and language pathologist	Interview – telephone	1
<input type="checkbox"/>	Edit Copy Delete	6	Graduate occupational therapist	Interview – telephone	1
<input type="checkbox"/>	Edit Copy Delete	7	General practitioner	Interview – telephone	1
<input type="checkbox"/>	Edit Copy Delete	8	Service manager	Interview – telephone	1
<input type="checkbox"/>	Edit Copy Delete	9	House coordinator	Interview – telephone	1
<input type="checkbox"/>	Edit Copy Delete	10	Direct support staff	Interview – telephone	1

When performing an insert operation, we will insert data in order. For example, we first obtain unique table data for the insert operation, then obtain non-unique data related to this table, use this table to associate with user_id and get the largest id to associate and insert into the database. In this process, we did not delete the previous data, so we need to obtain the maximum id of the unique table during the insertion process. This also means that if historical data needs to be queried in the future, it can be obtained by sorting these unique tables.

Also, use the second page as an example. We first extract the only information behavior_assessment and non_behavioural_assessment and store them in the assessment table. Then, we get the just updated assessment_id by getting the data with the largest id in the assessment of the current user, and use it to store the non-unique information persons_consulted.

```
# Page2
behavioural_assessment = extract.extract_paragraph(
    "Outline the behavioural assessment approaches implemented to develop this PBSP",
    "Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP",
    all_content)
non_behavioural_assessment = extract.extract_paragraph(
    "Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP",
    "PAGE 3 - Functional Behavioural Assessment", all_content)
#stor the unique information in assessment table
data_insert(
    mydb, cur, "assessment",
    ("user_id, " + "behaviouralAssessment, " + "nonBehaviouralAssessment"),
    (user_id, behavioural_assessment, non_behavioural_assessment))

#get the latest assessment_id by given user_id
last_id = data_get_last_id(mydb, cur, "assessment", user_id)[0][0]
current_table = extract.extract_table(
    'Persons consulted to prepare this PBSP (add/remove rows as required)',
    'Outline the behavioural assessment approaches implemented to develop this PBSP',
    tables, current_table_position)
```

PDF Extraction

1. PDF Reading (pdf_reader.py)

1.1 Constants

dictionary: 'page_info'

- require adjustments for different PDFs.

page_info: { \$page number\$: {"continuous": boolean, "next_page": boolean }}

- continuous: true if a section's answer is not completed on that page and more extraction from the next page is needed to continue with the previous page
- next_page: true if there are more sections/questions/attributes to fill in in that tables

```
page_info = {1: {"continuous": True, "next_page": False}, 3: {"continuous": False, "next_page": False},
4: {"continuous": False, "next_page": False}, 5: {"continuous": True, "next_page": True},
7: {"continuous": False, "next_page": True}, 9: {"continuous": False, "next_page": True},
10: {"continuous": False, "next_page": True}, 11: {"continuous": False, "next_page": True},
12: {"continuous": False, "next_page": False}, 13: {"continuous": False, "next_page": False},
14: {"continuous": False, "next_page": False}}
```

1.2 Python libraries

'pdfminer.six'

- Use `extract_text_to_fp` and set `output_type='html'` to convert the PDF file to an HTML format

'beautifulsoup'

- As selected options are in bold, use the library to parse the texts, and extract texts based on the 'bold' style in the HTML tags, named `bolded_lst`

1.3 Functions

table_extraction(page_number):

- use `read_pdf` from the 'tabula' library to read tables page by page and convert tables to nested lists for each page

extract_answers():

- Identify whether a table has single or multiple columns based on the length of lists:
 - If a section is comprised of a single column,
 - if the content is included in the `bolded_lst`, meaning the section is a select-option type, thus, append that single selected option to the 'answers_all' list
 - else: means that the section includes a list of contents, thus, append all to the 'answers' list
 - If a section is comprised of multiple columns, appends to the 'sub_col_answers_all' list

2. Extraction for tables with multiple columns (sub_column_data_inserter.py)

2.1 Constants

dictionary: ref_table

- Every multi-column-contained tables' foreign key reference to another table's primary key
- {\$multi-column-contained table name\$: \$referenced table name\$}


```
ref_table = {"persons_consulted": "assessment", "STC": "ba_function", "medication": "chemical_restraint",
"chemical_restraint": "intervention", "physical_restraint": "intervention",
"mechanical_restraint": "intervention", "environmental_restraint": "intervention",
"social_validity1": "chemical_restraint", "authorisation1": "chemical_restraint",
"social_validity2": "physical_restraint", "authorisation2": "physical_restraint",
"social_validity3": "mechanical_restraint", "authorisation3": "mechanical_restraint",
"social_validity4": "environmental_restraint", "authorisation4": "environmental_restraint",
"social_validity5": "seclusion_restraint", "authorisation5": "seclusion_restraint",
"how_implementer": "implementation", "implementation_plan": "implementation", "how_communicate": "implementation",
"how_implementation": "implementation"}
```

An example: {"persons_consulted": "assessment"}

PAGE 2 – Assessments and Data Gathering *table: assessment*

- This page contains the relevant information regarding the assessments undertaken, and the data gathered to inform PBSP development. This information only needs to be entered once by the user.

NOTE: We will need to develop checklist items to assess the content in this section. This section will mainly be looking for the absence of content (i.e., information is not provided).

Persons consulted to prepare this PBSP (add/remove rows as required) <i>table: persons_consulted</i>	
Who are they? <i>who</i>	How were they consulted? <i>they</i>
Mother	Face to face interview
Father	Telephone interview
Teacher	Face to face interview
Integration aide	Video conference interview
Sports coordinator	Face to face interview
Outline the behavioural assessment approaches implemented to develop this PBSP <i>behaviouralAssessment</i>	
Contextual Assessment Inventory completed by his teacher, observation (scatter plots) completed by his teacher and integration aide, and Functional Assessment Interview (completed by his mother and teacher)	
Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP <i>nonBehaviouralAssessment</i>	
None	

2.2 Algorithm (sub_column_data_insert(mydb, cur, page_num, sub_column_table_info, sub_col_answers_all))

```
for i in range( len(sub_column_table_info[page_num]) ): #for each multi-column table on that page
```

```
    for sub_attribute in sub_col_answers_all[i][0]: # Length is determined by the length of an element in extracted answers: sub_col_answers_all
```

```
        # concatenate all extracted sub_attribute names as a string
```

```
    if current sub-column table is in ref_table:
```

```
        # find the current table's foreign key -which is the primary key of its reference table: f_k
```

```
    # insert f_k and extracted text: sub_col_answers_all[i] into the db
```

3. Main text extraction algorithm (main.py)

3.1 Constants

3.1.1. dictionary: table_dict

- includes all the table names on each page of the PDF and all the attributes for each table.

```

table_dict = {1: [{"short_summary", "user_id, summary"}],
3: [{"assessment", "user_id, behaviouralAssessment, nonBehaviouralAssessment"}],
4: [{"ba_function", "user_id, functionName"}, {"ba_function", "user_id, description, summary, proposedAlternative"}],
5: [{"goal", "user_id, behaviour, life"}, {"strategies", "user_id, environment, teaching, others"}],
7: [{"reinforcement", "user_id, reinforcer, schedule, howIdentified"}, {"de_escalation", "user_id, howtoPrompt, strategies, postIncident"}],
9: [{"intervention", "user_id, ifProposed"}, {"intervention", "user_id, type"}, {"chemical_restraint", "positiveStrategy, circumstance, procedure,
howRestrainReduce, why"}],
10: [{"physical_restraint", "intervention_id, description, positiveStrategy, circumstance, procedure, how, why"}, {"mechanical_restraint", "descri
ption, positiveStrategy, circumstance, procedure, howKnow, howRestraint, why"}],
11: [{"environmental_restraint", "description, frequency, positive_strategy, circumstance, person, procedure, impact, howImpact,
howRestraint, why"}],
12: [{"seclusion_restraint", "frequency, positiveStrategy, circumstance, maxFrequency, procedure, how, why"}],
13: [{"implementation", "user_id, people, timeframe"}], 14: [{"socialv", "user_id, acceptability, who"}]
}

```

An example: {3: ("assessment", "user_id, behaviouralAssessment, nonBehaviouralAssessment")}

table:

Outline the behavioural assessment approaches implemented to develop this PBSP	<i>behaviouralAssessment</i>
Contextual Assessment Inventory completed by his teacher, observation (scatter plots) completed by his teacher and integration aide, and Functional Assessment Interview (completed by his mother and teacher)	
Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP	<i>nonBehaviouralAssessment</i>
None	

3.1.2. dictionary: sub_column_table_info

- includes table names of sections that have multiple columns.

```

sub_column_table_info = {3: ["persons_consulted"], 4: ["STC"], 9: ["medication", "social_validity1", "authorisation1"],
10: ["social_validity2", "authorisation2", "social_validity3", "authorisation3"],
11: ["social_validity4", "authorisation4"], 12: ["social_validity5", "authorisation5"],
13: ["how_implementer", "implementation_plan", "how_communicate", "how_implementation"],
14: ["socialv"]}

```

An example: {3: ["persons_consulted"]}

Persons consulted to prepare this PBSP (add/remove rows as required) <i>table: persons_consulted</i>	
Who are they? <i>who</i>	How were they consulted? <i>they</i>
Mother	Face to face interview
Father	Telephone interview
Teacher	Face to face interview
Integration aide	Video conference interview
Sports coordinator	Face to face interview

3.2 Algorithm

for *page_num* in *page_num_list*:

```

table_lists = table_extraction(page_num) #from the pdf_reader.py

```

```

if page_info[page_num-1]["next_page"] is True: # if there are texts that are continued from the previous page, ignore that

```

```

    extract_answers(table_lists[1:])

```

```

else:

```

```

    extract_answers(table_lists)

```

```

if continuous is True: #if a section's answer is split into two pages, extract the texts that are on the next page

```

```

    next_table_lists = table_extraction(page_num+1) #extract the continued texts that are on the next page

```

append **next_table_lists** to the **answers_all**

if the next page has more sections for the same table:

extract_answers(next_table_lists)

for db_table_name, current table's attributes in **table_dict[page_num]**:

if the current table is not in the ref_table: # meaning the current table is linked to the user info

records for insertion includes user_id

else:

#find the foreign key which is the primary key (id) of the reference table

concatenate all extracted values

if the current table has already had records:

update the row to db

else:

insert into the table in db

if **sub_col_answers_all**: #if there is/are sub-column tables on that page

if the sub-column tables are on the next page:

next_sub_table_lists = **table_extraction(page_num+1)**

extract_answers(next_sub_table_lists) **next_sub_answers**

append all **next_sub_answers** to **sub_col_answers_all**

sub_column_data_insert(mydb, cur, page_num, sub_column_table_info, sub_col_answers_all) # insert into the db

PDF Extraction2

1 pdfExtraction2.py

1.1 Python libraries

'pdfplumber '

- Use `extract_tables()` to get all text information on each page, including text in a table
- Use `filter()` to get all bold text on each page.

'fitz'

- Extract the text in PDF to generate a very long string containing all the text content of the pdf.

1.2 Functions

Through the `get_pdf_content` function, we get three variables.

- we can get a very long string containing all the text content of the pdf.
- A list whose items retain the table format to a certain extent
- A list of items that store all bold text and each of them represents a page.

The content extraction is in `process_pdf2()`, which extracts the information sequentially from PDF and store them in the database.

2 Extraction

2.1 Extraction for paragraphs

In fact, almost all data extraction uses the same principle, that is, to find the previous sentence and the next sentence of the fragment you want to extract, and use those two sentences to locate it. This process is written in `process_pdf2()`

Take `page2` as an example. As you can see, the target text, which is the text underlined in yellow, can be easily extracted by the title framed in red.

PAGE 2 – Assessments and Data Gathering

- This page contains the relevant information regarding the assessments undertaken, and the data gathered to inform PBSP development. This information only needs to be entered once by the user.

NOTE: We will need to develop checklist items to assess the content in this section. This section will mainly be looking for the absence of content (i.e., information is not provided).

Persons consulted to prepare this PBSP (add/remove rows as required)	
Who are they?	How were they consulted?
Mother	Face to face interview
Father	Telephone interview
Teacher	Face to face interview
Integration aide	Video conference interview
Sports coordinator	Face to face interview
Outline the behavioural assessment approaches implemented to develop this PBSP	
Contextual Assessment Inventory completed by his teacher, observation (scatter plots) completed by his teacher and integration aide, and Functional Assessment Interview (completed by his mother and teacher)	
Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP	
None	

nts initiate the interaction. \n\n\nPAGE 2 - Assessments and Data Gathering \n\n\This page contains the relevant information regarding the assessments undertaken, and the data gathered to \ninform PBSP development. This information only needs to be entered once by the user. \n\nNOTE: We will need to develop checklist items to assess the content in this section. This section will mainly \nbe looking for the absence of content (i.e., information is not provided). \n\nPersons consulted to prepare this PBSP (add/remove rows as required) \nWho are they? \nHow were they consulted? \nMother \nFace to face interview \nFather \nTelephone interview \nTeacher \nFace to face interview \nIntegration aide \nVideo conference interview \nSports coordinator \nFace to face interview \nOutline the behavioural assessment approaches implemented to develop this PBSP \nContextual Assessment Inventory completed by his teacher, observation (scatter plots) completed by his teacher \nand integration aide, and Functional Assessment Interview (completed by his mother and teacher) \nAdditional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP \nNone \n\n\nPAGE 3 - Functional Behavioural Assessment \n\n\We en

We can use the `extract_paragraph` function to achieve this by rendering the two positioning texts and the text format of the PDF.

```
# Page2
behavioural_assessment = extract.extract_paragraph(
    "Outline the behavioural assessment approaches implemented to develop this PBSP",
    "Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP",
    all_content)
```

There is also another method to extract paragraphs, which also includes a variable called `current_text_position` as input to indicate the last extraction position. This is because there are some repeated positioning statements in the pdf, we use a variable to record the end address of the previous excerpt during the sequential extraction process. This reduces the time of repeated scanning and avoids the problem that repeated statements cannot be located. The returned value from `extract_paragraph` given `start` is a tuple the first one is the position and the second one is the target text.

```
goals_life = extract.extract_paragraph_given_start(
    "Goals specific to enhancing the person's quality of life",
    "Strategies", all_content, current_text_position)
current_text_position = goals_life[1]
data_insert(mydb, cur, "goal", ("user_id, " + "behaviour, " + "life"),
            (user_id, goals_behaviours, goals_life[0]))
```

2.2 Extraction for tables

The process to extract a table is the same as a paragraph, which is also used the locating text to find the target table.

PAGE 2 – Assessments and Data Gathering

- This page contains the relevant information regarding the assessments undertaken, and the data gathered to inform PBSP development. This information only needs to be entered once by the user.

NOTE: We will need to develop checklist items to assess the content in this section. This section will mainly be looking for the absence of content (i.e., information is not provided).

Persons consulted to prepare this PBSP (add/remove rows as required)	
Who are they?	How were they consulted?
Mother	Face to face interview
Father	Telephone interview
Teacher	Face to face interview
Integration aide	Video conference interview
Sports coordinator	Face to face interview
Outline the behavioural assessment approaches implemented to develop this PBSP	
Contextual Assessment Inventory completed by his teacher, observation (scatter plots) completed by his teacher and integration aide, and Functional Assessment Interview (completed by his mother and teacher)	
Additional non-behavioural assessments undertaken or reviewed to inform the development of this PBSP	
None	

```
['Persons consulted to prepare this PBSP (add/remove rows as required)',
 None],
['Who are they?', 'How were they consulted?'],
['Mother', 'Face to face interview'],
['Father', 'Telephone interview'],
['Teacher', 'Face to face interview'],
['Integration aide', 'Video conference interview'],
['Sports coordinator', 'Face to face interview'],
['Outline the behavioural assessment approaches implemented to develop this PBSP',
 None],
```

```
#get the latest assessment_id by given user_id
last_id = data_get_last_id(mydb, cur, "assessment", user_id)[0][0]
current_table = extract.extract_table(
    'Persons consulted to prepare this PBSP (add/remove rows as required)',
    'Outline the behavioural assessment approaches implemented to develop this PBSP',
    tables, current_table_position)

current_table_position = current_table[1]
```

2.2 Extraction for bolds

The bolds are stored in a list called bolds and each item is represented as a page. When extracting bolds from the table, we use a predefined list and for each item in this list, we search whether it exists in the bolds. Also, there are other more accurate methods. For example, we can first locate the page that the target bolds in and then apply the search method, which can avoid the same text on other pages.

PAGE 3 – Functional Behavioural Assessment

- *We envision that this page will start with a drop-down menu that allows the user to choose one of the five functions of behaviour:*
 - *Avoidance/escape*
 - *Communication*
 - *Physical/sensory need*
 - *Seeking an object/activity*
 - *Other – please specify*
- *Once they choose a function, the text boxes below would come up for the user to input information.*
- *Following this, the user will be asked if they want to include another function of behaviour.*
 - *If YES, the drop-down menu that allows the user to choose one of the five functions behaviour appears again. Once a choice is made, these text boxes appear again.*
 - *Repeat this process until the user has no more functions to add.*

NOTE: The items on the existing BSPA-tool will be used to assess the content in this section. We need to make some additional modifications to the tool to assist in more precise scoring.

Function				
Avoidance/escape	Communication	Physical/sensory need	Seeking an object/activity	Other – please specify

```
: ['Positive Behaviour Support Plan (PBSP) PAGE 1 - About the Person with Disability ',
  'PAGE 2 - Assessments and Data Gathering ',
  'PAGE 3 - Functional Behavioural Assessment Avoidance/escape ',
  'PAGE 4 - Positive Behavioural Support Interventions To manage the triggers of Eddie\'s behaviour: To assist, school management wi
11: To assist, teachers will: ',
  'What: How: Who: When: Where: Materials:',
  ',
  ',
  'PAGE 5 - Restrictive Intervention No ',
  'PAGE 6 - PBSP Implementation ']
```

```
# Page3
```

```
list_function = [  
    'Avoidance/escape', 'Communication', 'Physical/sensory need',  
    'Access - person/object/activity', 'Other - please specify'  
]
```

```
function_name = []
```

```
for i in list_function:
```

```
    for j in bolds:
```

```
        if i in j:
```

```
            function_name.append(i)
```