

SOI ISLP Website

Purpose: Students can apply for ISLP trips and Staffs can approve/decline the student's application.

Main Page

The screenshot shows the homepage of the SOI ISLP website. At the top, there is a logo for "REPUBLIC POLYTECHNIC" and a "Sign In" button. Below the header is a large image of a group of people posing for a photo in an airport terminal. Underneath this image is a navigation bar with a "Present" dropdown menu and a "Past" dropdown menu. The "Present" menu is currently active and displays three sections: "Batam ISLP", "Brunei ISLP", and "YMCA Vietnam e-ISLP". Each section contains a thumbnail image, a title, a brief description, and two small text boxes for trip dates and registration deadlines.

- Batam ISLP**
Join the Batam International Service-Learning Programme (ISLP) 2025! Make a difference while exploring a new culture through meaningful community service.
Trip Dates: 24 - 28 March 2025
Registration Deadline: 31 October 2024
- Brunei ISLP**
Join Brunei ISLP today! Visit and explore different traditions and enjoy delicious food.
Trip Dates: 26 February 2025, 28 February 2025, 01 March 2025
Registration Deadline: 30 January 2025
- YMCA Vietnam e-ISLP**
This is the second instalment of SOI's virtual International Service-Learning Project (ISLP). The e-ISLP serves as an avenue to continue international relations virtually without travelling. For the coming e-ISLP in October, we partnered with YMCA Vietnam. Students engaged youths from Vietnam in areas of life skills, befriending, and various activities. e-ISLP Partnership with YMCA Vietnam.
Trip Dates: 15 - 16 November 2022

Contents

- [Main Features](#)
- [How to use](#)
- [Setting up mySQL database](#)
- [Setting up Amazon Cognito](#)
- [Setting up S3 Bucket](#)
- [Setting up AWS CLI](#)
- [Connecting to EC2](#)
- [Showing of Features](#)
- [Explanation of Codes](#)

Main Features:

Main Page (public.html without user) ([Example](#)) ([Code](#))

- Slideshow to show photos ([Example](#)) ([Code](#))
- Viewing of the Present and Past ISLP based on whether the Trip Dates are over ([Example](#)) ([Code](#))
- Signing in to the website via Amazon Cognito and Logging Out (login.html) ([Example](#)) ([Code](#))
 - For Authentication and Authorization ([Code](#))
 - Students (direct to public.html)
 - Staff (direct to staff.html)

Student Page (public.html with user) ([Example](#))

- Applying for ISLP Trips (For students and they can only apply once) ([Example](#)) ([Code](#))
 - Re-submission of applications
 - Able to view their previous submission and re-submit their submission if they need to change their details ([Example](#))
 - Receive an email confirmation of their details through their personal email account ([Example](#)) ([Code](#))
- Viewing ISLP full details and the schedule for the ISLP trip (For students that got accepted into the ISLP trip) ([Example](#)) ([Code](#))

Staff Page (staff.html) ([Example](#))

- Viewing ISLP details and the student's submission ([Example](#)) ([Code](#))
 - Viewing of Student's Document ([Example](#)) ([Code](#))
- Filter of Diploma and Searching student's name ([Example](#)) ([Code](#))
- Editing of ISLP details ([Example](#)) ([Code](#))
- Creating new ISLP trips ([Example](#)) ([Code](#))
- Approving and declining of students ([Example](#)) ([Code](#))
 - Able to select all or select the number of rows to approve and decline
 - Students will receive an email through their personal email indicating that they have been accepted or rejected for the trip ([Example](#))

How to use:

website.py

To connect to MySQL database in AWS (Line 24)

The screenshot shows the AWS RDS 'Databases' page for a 'primary-database'. The 'Configuration' tab is selected. Key details include:

- DB Identifier:** primary-database
- Status:** Available
- Engine version:** 8.0.39
- RDS Extended Support:** Disabled
- DB name:** primarydatabase
- License model:** General Public License
- Option groups:** defaultmysql-8-0 (In sync)
- Amazon Resource Name (ARN):** arn:aws:rds:ap-southeast-1:794038236090:db:primary-database
- Resource ID:** db-QJFHBO5FFHPJ4DKJQ5Y6ZTBRQ

```
#Database Credentials
host = "primary-database.cdou8g0ag1na.ap-southeast-1.rds.amazonaws.com"
port = 3306
database = "primarydatabase"
username = "admin"
password = "_____"
```

To retrieve or upload the documents/images from the S3 buckets (Line 55)

The screenshot shows the AWS S3 'Buckets' page for the 'submissionsbucketfyp' bucket. The 'Properties' tab is selected. Key details include:

- AWS Region:** Asia Pacific (Singapore) ap-southeast-1
- Amazon Resource Name (ARN):** arn:aws:s3:ap-southeast-1:794038236090:bucket/submissionsbucketfyp
- Creation date:** January 9, 2023
- Default encryption:** SSE-KMS
- Encryption key ARN:** arn:aws:kms:ap-southeast-1:794038236090:key/42702622-69d7-4c1b-be53-a6558f53595a
- Bucket Key:** When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS.

```
S3_BUCKET = "submissionsbucketfyp"
S3_REGION = "ap-southeast-1"
S3_FOLDER = "uploads"
S3_IMAGE_FOLDER = "images"
S3_SCHEDULE_FOLDER = "schedules"
S3_SLIDESHOW_FOLDER = "slideshow"
kms_key_id = 'arn:aws:kms:ap-southeast-1:794038236090:key/42702622-69d7-4c1b-be53-a6558f53595a'
```

Sending emails to student's personal account using your own email (line 137)

```
#Send email to the applicant
def send_email(recipient_email, subject, body):
    sender_email = "_____@gmail.com" <- Example
    sender_password = "_____"

    msg = MIMEText()
    msg['From'] = sender_email
    msg['To'] = recipient_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))
```

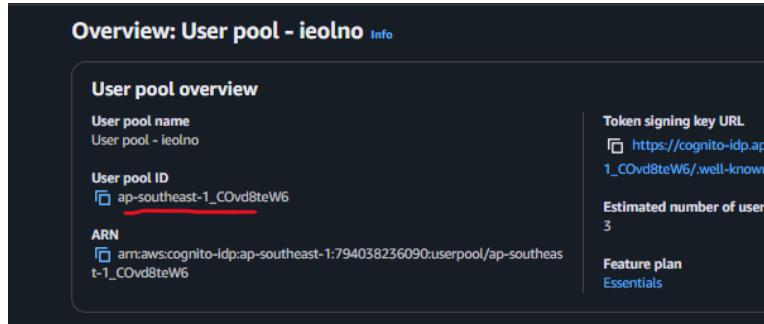
For Amazon Cognito authorization (line 209)

The screenshot shows the AWS Cognito console with the navigation path: Amazon Cognito > User pools > User pool - leisuo > App clients > App client: SOI ISLP. The 'App client information' section displays details like Client ID (5beatj5fm3dfd301dtm6g84181), Client secret (hidden), and Authentication flow session duration (3 minutes). Below this, the 'Quick setup guide' tab is selected, showing development platforms (Golang, Java, NodeJS, Python) and example code for Python:

```
1 oauth.register(
2     name='oidc',
3     authority='https://cognito-idp.ap-southeast-1.amazonaws.com/ap-southeast-1_C0vd8teW6',
4     client_id='5beatj5fm3dfd301dtm6g84181',
5     client_secret='<client secret>',
6     server_metadata_url='https://cognito-idp.ap-southeast-1.amazonaws.com/ap-southeast-1_C0vd8teW6/.well-known/openid-configuration',
7     client_kwargs={'scope': 'email openid phone'}
8 )
```

```
#Amazon Cognito
oauth.register(
    name='oidc',
    authority='https://cognito-idp.ap-southeast-1.amazonaws.com/ap-southeast-1_C0vd8teW6',
    client_id='5beatj5fm3dfd301dtm6g84181',
    client_secret='_____',
    server_metadata_url='https://cognito-idp.ap-southeast-1.amazonaws.com/ap-southeast-1_C0vd8teW6/.well-known/openid-configuration',
    client_kwargs={'scope': 'email openid phone'}
)
```

Checking the role of the user that log in using their groups that is tied to their email in Amazon Cognito user pool (line 182)



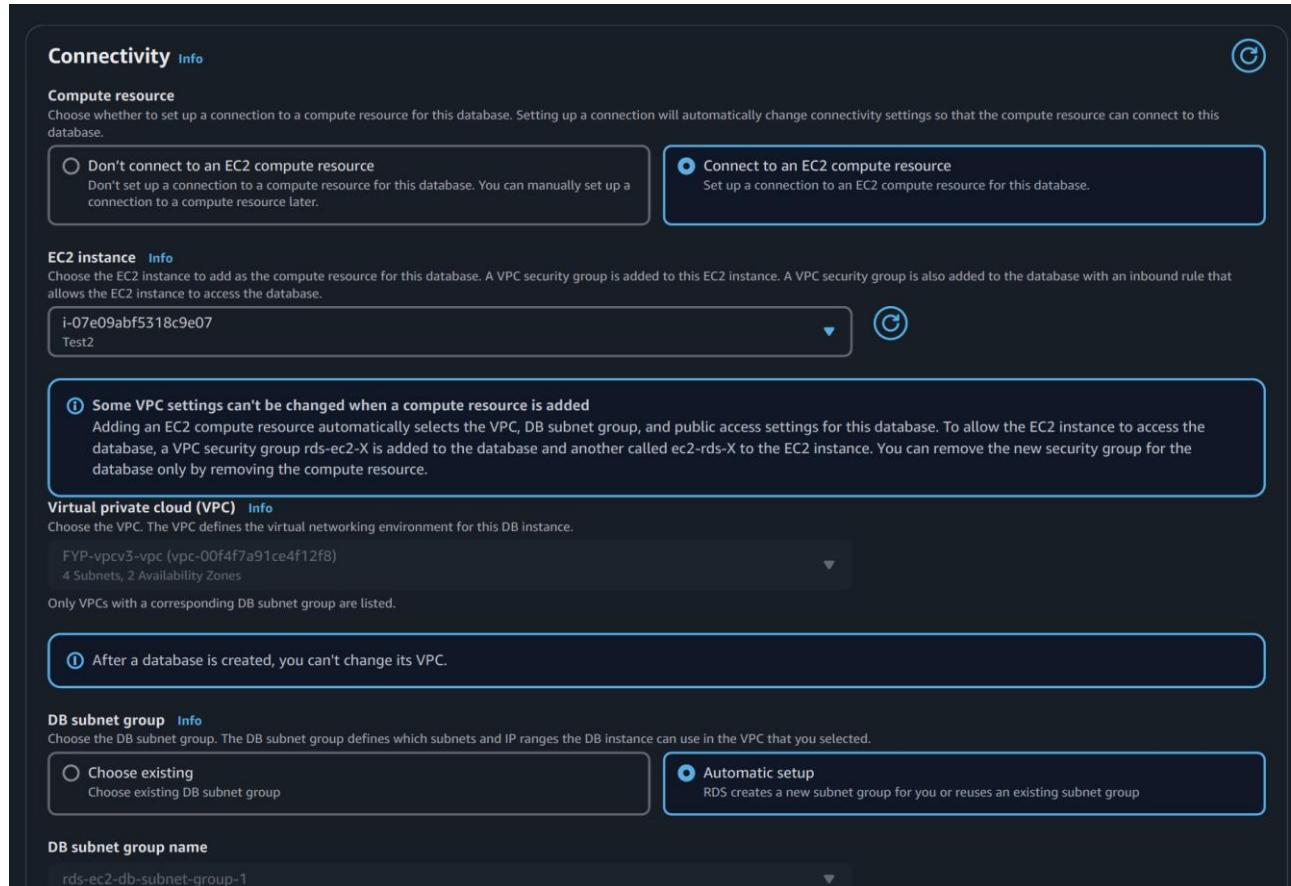
The screenshot shows the 'User pool overview' section of the AWS Cognito console. It displays the following details:

- User pool name: User pool - ieolno
- User pool ID: ap-southeast-1_C0vd8teW6 (highlighted with a red box)
- ARN: arn:aws:cognito-idp:ap-southeast-1:794038236090:userpool/ap-southeast-1_C0vd8teW6
- Token signing key URL: https://cognito-idp.ap-southeast-1.amazonaws.com/.well-known/jwks.json
- Estimated number of users: 3
- Feature plan: Essentials

```
#Checking if it is a student/staff
def check_role(student_email, role_name):
    try:
        #Get the user's attributes from Cognito
        response = cognito_client.admin_list_groups_for_user(
            UserPoolId='ap-southeast-1_C0vd8teW6',
            Username=student_email
        )
```

Setting up of MySQL Database

1. Create in Amazon RDS (MySQL)
2. Connect it to an EC2



The screenshot shows the 'Set compute and storage options' step of the RDS MySQL DB instance creation wizard. It includes the following sections:

- Connectivity**:
 - Compute resource**:
 - Don't connect to an EC2 compute resource
 - Connect to an EC2 compute resource
- EC2 instance**:
 - i-07e0abf5318c9e07
 - Test2
- Virtual private cloud (VPC)**:
 - Some VPC settings can't be changed when a compute resource is added**:

Adding an EC2 compute resource automatically selects the VPC, DB subnet group, and public access settings for this database. To allow the EC2 instance to access the database, a VPC security group rds-ec2-X is added to the database and another called ec2-rds-X to the EC2 instance. You can remove the new security group for the database only by removing the compute resource.
- DB subnet group**:
 - Choose existing**: Choose existing DB subnet group
 - Automatic setup**: RDS creates a new subnet group for you or reuses an existing subnet group
- DB subnet group name**: rds-ec2-db-subnet-group-1

3. After creating, turn on the database and modify it to allow publicly accessible so the code can access it.

Connectivity & security

Network type [Info](#)
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

IPv4
Your resources can communicate only over the IPv4 addressing protocol.

Dual-stack mode
Your resources can communicate over IPv4, IPv6, or both.

DB subnet group: rds-ec2-db-subnet-group-1

Security group: List of DB security groups to associate with this DB instance.
[Choose security groups](#)

rds-ec2-1 X

Certificate authority [Info](#)
Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rds-ca-rsa2048-g1 (default)
Expiry: May 22, 2061

Additional configuration

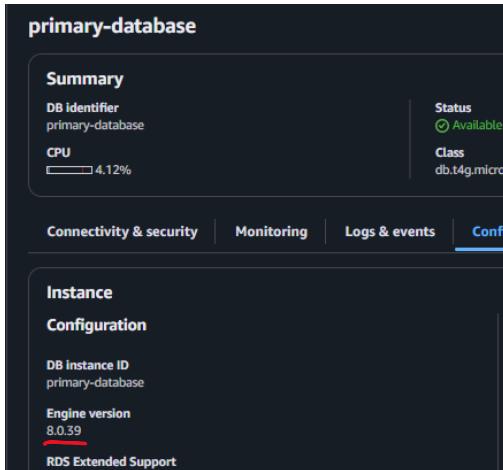
Public access
 Publicly accessible
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

Not publicly accessible
No IP address is assigned to the DB instance. EC2 instances and devices outside the VPC can't connect.

Database port: Specify the TCP/IP port that the DB instance will use for application connections. The application connection string must specify the port number. The DB security group and your firewall must allow connections to the port. [Learn more](#)

3306

4. Then download the MySQL Workbench must be the same version as the RDS in AWS
Download here (latest): <https://dev.mysql.com/downloads/installer/>
Older versions: <https://downloads.mysql.com/archives/workbench/>



5. Once installed, then connect to the Amazon RDS MySQL

primary-database

Summary

DB identifier: primary-database

Status: Available

CPU: 3.64%

Connectivity & security

Endpoint: primary-database.cdou8g0ag1na.ap-southeast-1.rds.amazonaws.com

Port: 3306

MySQL Connections +

Manage Server Connections

Connection Name: AWSDatabase

Connection Method: Standard (TCP/IP)

Parameters SSL Advanced

Hostname: primary-database.cdou8g0ag1na.ap-southeast-1.rds.amazonaws.com

Username: admin

Password: Store in Vault ... Clear

Default Schema:

6. Creating a database in MySQL workbench

```
CREATE DATABASE primarydatabase;
USE primarydatabase;
```

<pre>CREATE TABLE islodata (ISLP VARCHAR(255) PRIMARY KEY, deadline DATE, public_information TEXT, schedule_file VARCHAR(255), more_details TEXT, photo_file VARCHAR(255)); CREATE TABLE tripdates (id INT AUTO_INCREMENT PRIMARY KEY, ISLP VARCHAR(255), trip_date DATE, FOREIGN KEY (ISLP) REFERENCES islodata(ISLP) ON DELETE CASCADE);</pre>	<p>The screenshot shows the MySQL Workbench interface with three tables defined:</p> <ul style="list-style-type: none"> islodata (Primary Key: ISLP): <ul style="list-style-type: none"> ISLP VARCHAR(255) deadline DATE public_information TEXT schedule_file VARCHAR(255) more_details TEXT photo_file VARCHAR(255) tripdates (Primary Key: id): <ul style="list-style-type: none"> id INT ISLP VARCHAR(255) trip_date DATE submissions (Primary Key: id): <ul style="list-style-type: none"> id INT name VARCHAR(100) email VARCHAR(100) diploma VARCHAR(50) interest TEXT file_path VARCHAR(255) ISLP VARCHAR(50) status VARCHAR(50) file_name VARCHAR(255) personal_email VARCHAR(255) 	<pre>CREATE TABLE submissions (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100), diploma VARCHAR(50), interest TEXT, file_path VARCHAR(255), ISLP VARCHAR(50), status VARCHAR(50), file_name VARCHAR(255), personal_email VARCHAR(255), FOREIGN KEY (ISLP) REFERENCES islodata(ISLP) ON DELETE CASCADE);</pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Setting up Amazon Cognito

1. Create a user pool in Amazon Cognito

The screenshot shows the 'Define your application' section of the Cognito User Pool creation wizard. It includes fields for Application type (set to Traditional web application), Name (set to SOI_ISLP), and Configure options.

Configure options

You must make a few initial choices about the user pool that supports your application. To change these settings later, you must create a new user pool.

Options for sign-in identifiers

Choose sign-in attributes. Users can be an email address, phone number, or a user-selected username. When you select only email and phone, users must select either email or phone as their username type. When username is an option, users can sign in with any options you select if they have provided a value for that option.

Email

Phone number

Username

Want to set up social SAML or OIDC sign-in?

Required attributes for sign-up

Choose any attributes that you want to require users to provide. With username alone, you must set email address or phone number as a required attribute.

Select Attributes

email

2. Create an App clients to allow the users to do authorisation

The screenshot shows the 'Edit managed login pages configuration' section. It includes sections for Managed login pages, Allowed callback URLs, and Allowed sign-out URLs - optional.

Managed login pages

Configure the managed login pages for this app client.

Allowed callback URLs

Enter at least one callback URL to redirect the user back to after authentication. This is typically the URL for the app receiving the authorization code issued by Cognito. You may use HTTPS URLs, as well as custom URL schemes.

URL: https://127.0.0.1:5000/authorize

Allowed sign-out URLs - optional

Enter at least one sign-out URL. The sign-out URL is a redirect page sent by Cognito when your application signs users out. This is needed only if you want Cognito to direct signed-out users to a page other than the callback URL.

URL: https://127.0.0.1:5000/

3. Create Users like this

Users (3) Info				
View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.				
Property:	User name	Email address	Email verified	Confirmation status
<input type="radio"/>	294a553c-60e1-70f6-ea7b-ff79c...	22023942@myrp.edu.sg	No	Confirmed
<input type="radio"/>	d93a955c-4011-7085-6ee5-62b6...	staff1@rp.edu.sg	No	Confirmed
<input type="radio"/>	f90ae5fc-0091-7020-f39c-c1af7a...	22042810@myrp.edu.sg	No	Confirmed

4. Create Groups for staff and student

From the IAM Roles that are created

Staff

Group: staff

Group information

- Group name: staff
- IAM Role ARN: arn:aws:iam::794038236090:role/service-role/staff

Group members (1)

User name	Email address	Email verified	Confirmation status	Status
d93a955c-4011-7085-6ee5-62b6...	staff1@rp.edu.sg	No	Confirmed	Enabled

staff

Summary

- Creation date: January 09, 2025, 11:05 (UTC+08:00)
- Last activity: -

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Student

Student

Summary

- Creation date: January 12, 2025, 22:41 (UTC+08:00)
- Last activity: -

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Group: student

Group information

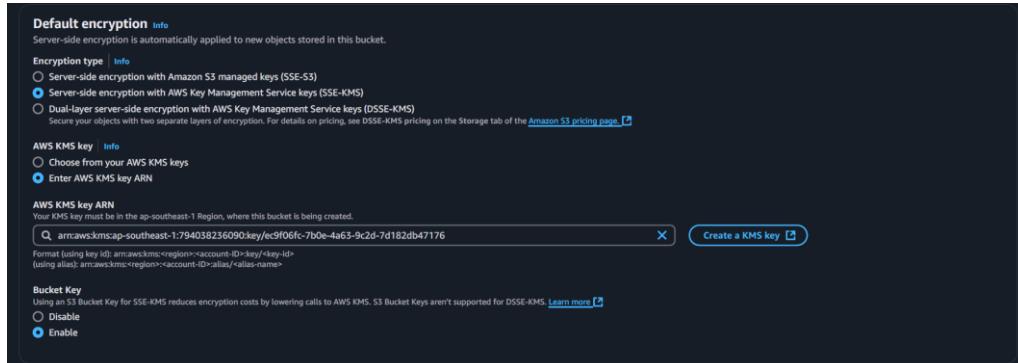
- Group name: student
- IAM Role ARN: arn:aws:iam::794038236090:role/Student

Group members (2)

User name	Email address	Email verified	Confirmation status	Status
f90ae5fc-0091-7020-f39c-c1af7a...	22042810@myrp.edu.sg	No	Confirmed	Enabled
294a553c-60e1-70f6-ea7b-ff79c...	22023942@myrp.edu.sg	No	Confirmed	Enabled

Setting up S3 Bucket

1. Create a S3 Bucket and create an encryption key. The encryption key should be for the IAM users to only use.



2. Create 4 folders

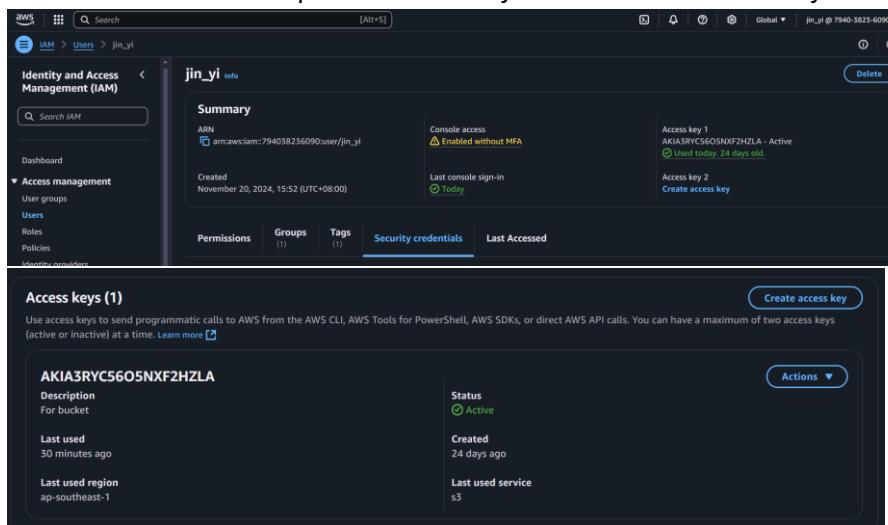
<p>The screenshot shows the 'Objects (4)' section of the S3 bucket 'submissionsbucketfy'. It lists four folder objects: 'images/' (Folder), 'schedules/' (Folder), 'slideshow/' (Folder), and 'uploads/' (Folder). Each folder has a small preview icon and a 'Copy S3 URI' button.</p>	<p>Images – for ISLP trip photos to display in the Present and Past section</p> <p>Schedules - to display the ISLP trip schedule for the trip dates (Must be in xlsx format)</p> <p>Slideshow – display the photos in the slideshow of the main page</p> <p>Uploads – to view the documents that the students</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Setting up AWS CLI

1. Download AWS CLI

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

2. To allow access to the S3 bucket, you need to create an access key for your own IAM user. Remember to save the password when you create the access key



3. Use the command, 'aws configure' in your command prompt

AWS Access Key ID [None]: YOUR_ACCESS_KEY

AWS Secret Access Key [None]: YOUR_SECRET_KEY

Default region name [None]: ap-southeast-1

Connecting to EC2 from RDS

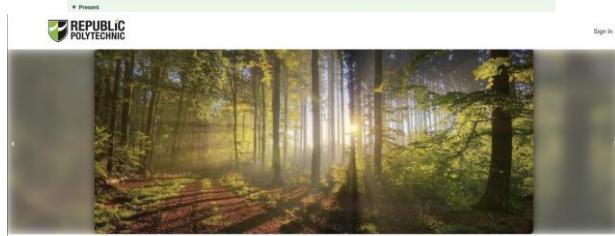
Connecting to two EC2. However, since this is a free tier account, I can only connect to one since if I do two it is a Cross-Availability Zone and it will charge.

The screenshot shows the Amazon RDS console under the 'Databases' section for a database named 'primary-database'. On the left, there's a sidebar with various navigation options like 'Dashboard', 'Databases', 'Query Editor', etc. The main area is divided into three sections: 'Connected compute resources (1)', 'Proxies (0)', and 'Security group rules (3)'. In the 'Connected compute resources' section, there's a single entry for an EC2 instance with identifier 'i-07a8b0ef4bad3d79e', located in 'ap-southeast-1a', associated with VPC security group 'rds-ec2-2' and compute resource security group 'ec2-rds-2'. The 'Proxies' section shows 'No proxies' with a 'Create proxy' button. The 'Security group rules' section lists three rules: one for 'rds-ec2-1' (CIDR/IP - Inbound, 0.0.0.0/0), one for 'rds-ec2-1' (EC2 Security Group - Inbound, sg-08c26cd910f84bb67), and one for 'rds-ec2-2' (EC2 Security Group - Inbound, sg-0bacc111cb9aa7734).

This screenshot shows the 'Review and confirm' step of the 'Set up EC2 connection' wizard. It's Step 2 of 2. The top navigation bar shows 'RDS > Databases > Set up EC2 connection'. The main content area is titled 'Review and confirm' and contains a 'Connection summary' section. It states: 'You are setting up a connection between RDS database primary-database and EC2 instance i-0f3f03313652db31d [?]. To set up a connection between the database and the EC2 instance, VPC security group is added to the database, and VPC security group ec2-rds-2 is added to the EC2 instance.' Below this is a diagram illustrating the connection setup. It shows two boxes: 'Security group: rds-ec2-2 (connection rule)' containing an 'RDS' icon and 'primary-database Port: 3306', and 'Security group: ec2-rds-2 (connection rule)' containing an 'EC2' icon and 'i-0f3f03313652db31d'. A blue arrow points from the RDS side to the EC2 side, and an orange arrow points from the EC2 side back to the RDS side. A note below the diagram says: 'Bold indicates an addition being made to set up a connection.' The next section is 'Changes to EC2 instance: i-0f3f03313652db31d', which shows a table of attribute changes. The 'Security group' attribute is listed with its current value as 'launch-wizard-3' and its new value as 'launch-wizard-3, ec2-rds-2'. At the bottom, a warning message states: '⚠ Cross-Availability Zone (AZ) charges might apply. The RDS database primary-database (ap-southeast-1a) and EC2 instance i-0f3f03313652db31d (ap-southeast-1b) are in different AZs. Cross AZ charges might apply. Data transfer within same Region [?]' with a 'Cancel', 'Previous', and 'Set up' button.

Showing of features

- Slideshow



- Viewing of Past and Present ISLP Trips



▼ Present

Batam ISLP

Join the Batam International Service-Learning Programme (ISLP) 2025! Make a difference while exploring a new culture through meaningful community service.

Trip Dates: 24 - 28 March 2025
Registration Deadline: 31 October 2024

▼ Past

Brunei ISLP

Join Brunei ISLP today! Visit and explore different traditions and enjoy delicious food.

Trip Dates: 26 February 2025, 28 February 2025, 01 March 2025
Registration Deadline: 30 January 2025

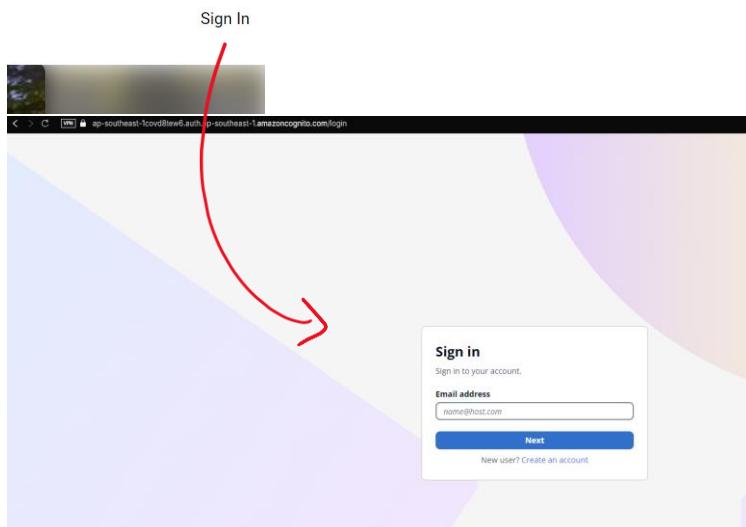
▼ Past

YMCA Vietnam e-ISLP

This is the second instalment of SOI's virtual International Service-Learning Project (ISLP). The e-ISLP serves as an avenue to continue international relations virtually without travelling. For the coming e-ISLP in October, we partnered with YMCA Vietnam. Students engaged youths from Vietnam in areas of life skills, befriending, and various activities. e-ISLP Partnership with YMCA Vietnam.

Trip Dates: 15 - 16 November 2022

- Signing into the website through Cognito



- Student Page

The student page features a banner photo of a large group of students posing in an airport terminal with a Republic Polytechnic banner. Below the banner are five small circular navigation dots. The main content area includes:

- BATAM ISLP**: A thumbnail image of a group at an event, with the text 'BATAM ISLP 2024'. Below it is a green 'Learn More' button.
- Brunei ISLP**: A thumbnail image of a mosque at night, with the text 'Join Brunei ISLP today! Visit and explore different traditions and enjoy delicious food.' Below it are two green buttons: 'Learn More' and 'Sign up'.

At the top right of the page, there is a user profile section with the email '22023942@myrp.edu.sg' and a 'Log Out' link.

- Applying for ISLP Trips

By pressing on to the sign-up button. It will lead to the form for the ISLP trip that they click (form.html).

The school email is already there to prevent students from using someone else's school email.

Once submitted then can re-submit again if they accidentally put the wrong information or document.

They would also receive an email that the submission is received

Submission Confirmation

22023942@myrp.edu.sg
To: You
Sun 02/02/2025 18:44

Dear Jin Yi,

Thank you for your submission! Here are the details we received:

- Name: Jin Yi
- Email: 22023942@myrp.edu.sg
- Personal Email: [REDACTED]
- Diploma: DIT
- Interest: I want to go to Brunei because I can help. :>
- ISLP: Brunei ISLP

Your uploaded file: Jin Yi Testimonial.docx

Best regards,
Republic Polytechnic

CONFIDENTIALITY CAUTION: This message is intended only for the use of the individual or entity to whom it is addressed and contains information that is privileged and confidential. If you, the reader of this message, are not the intended recipient, you should not disseminate, distribute or copy this communication. If you have received this communication in error, please notify us immediately by return email and delete the original message. Thank you.

- Viewing ISLP full details and the schedule for the ISLP trip (For students that got accepted into the ISLP trip)

The table below is retrieved from the excel file that contains the schedule for the specific ISLP and then displayed like this.

Date	Information
Day 1: Monday, March 24, 2025 – Arrival & Orientation	8:00 AM – 12:00 PM Arrival at Batam, transfer to accommodation. 12:30 PM – 1:30 PM Lunch at the accommodation. 2:00 PM – 3:30 PM Orientation session (Program overview, safety protocols, introductions). 3:30 PM – 4:30 PM Free time to settle in. 4:30 PM – 5:30 PM Free time to explore local cuisine. 5:30 PM – 8:30 PM Group discussion on expectations and goals. 9:00 PM: Free time.
Day 2: Tuesday, March 25, 2025 – Community Service & Local Immersion	7:00 AM – 8:30 AM Breakfast. 8:30 AM – 9:30 AM Community service project 1 (Assist in local schools, orphanages, or environmental clean-up). 12:30 PM – 1:30 PM Lunch at a local restaurant or provided by the community. 2:00 PM – 4:30 PM Visit to a cultural site or historical landmark. 4:30 PM – 5:30 PM Free time to explore local culture. 5:30 PM – 7:30 PM Dinner with local families or community members. 7:30 PM – 9:00 PM Group reflection on the day's experiences. 9:00 PM: Free time.
Day 3: Wednesday, March 26, 2025 – Community Service & Cultural Exploration	7:00 AM – 8:30 AM Breakfast. 8:30 AM – 12:30 PM Community service project 2 (Environmental sustainability, working with local organizations). 12:30 PM – 1:30 PM Lunch. 2:00 PM – 4:30 PM Cultural excursion (Visit local markets, villages, or religious sites). 4:00 PM – 5:30 PM Interactive session with a local cultural expert or historian. 5:00 PM – 7:30 PM Dinner at a local restaurant. 7:30 PM – 9:00 PM Group reflection and feedback session on the programme. 9:00 PM: Free time.
Day 4: Thursday, March 27, 2025 – Final Community Service & Local Experience	7:00 AM – 8:30 AM Breakfast. 8:30 AM – 12:30 PM Community service project 3 (Continue or complete the ongoing service projects). 12:30 PM – 1:30 PM Lunch with community members. 2:00 PM – 4:30 PM Free time for personal exploration (beaches, shopping, etc.). 4:00 PM – 5:30 PM Preparation for departure, packing. 5:00 PM – 6:30 PM Final group gathering, sharing of experiences. 7:30 PM – 9:30 PM Group reflection and feedback session on the programme. 9:00 PM: Free time.
Day 5: Friday, March 28, 2025 – Departure	7:00 AM – 8:30 AM Breakfast. 8:30 AM – 9:30 AM Final group gathering, sharing of experiences. 10:00 AM – 12:00 PM Depart Batam and travel back to respective locations.

- Staff page

The screenshot shows a list of submissions for different ISLP programs. The Brunei ISLP submission is selected, displaying its details. The submission table includes columns for Name, School Email, Personal Email, Diploma, Interest, Document, and Status.

NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA	INTEREST	DOCUMENT	STATUS
Juri Hao	22042810@myrp.edu.sg	q7junhao@gmail.com	DIT	Nice	View Document	Declined ✓ X
Jin Yi	22023942@myrp.edu.sg	jinbh0@gmail.com	DIT	I want to go to Brunei because I can help. :)	View Document	Pending ✓ X

- Viewing ISLP details and the student's submission

This screenshot shows a detailed view of a submission for the Brunei ISLP program. It includes a large image of the Brunei skyline at night, program details like deadline and public information, and a table of student information with their status.

NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA	INTEREST	DOCUMENT	STATUS
Juri Hao	22042810@myrp.edu.sg	q7junhao@gmail.com	DIT	Nice	View Document	Declined ✓ X
Jin Yi	22023942@myrp.edu.sg	jinbh0@gmail.com	DIT	I want to go to Brunei because I can help. :)	View Document	Pending ✓ X

- Viewing of Document on a html page

The screenshot shows a document viewer interface with a dark theme. It displays a warning message about the document being created with Spire.Doc for Python. Below the message is a section titled "Student Testimonial" featuring a small profile picture of a person.

- Filter of Diploma and Searching student's name

Filter

<input type="checkbox"/>	NAME	SCHOOL EMAIL
<input type="checkbox"/>	Jun Hao	22042810@rp.edu.sg
<input type="checkbox"/>	Jin Yi	22023942@rp.edu.sg

Select Diplomas

DAAA

DCDF

DECM

DFT

DIT

Cancel
Apply

Filter

<input type="checkbox"/>	NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA	INTEREST
<input type="checkbox"/>	Jin Yi	22023942@myrp.edu.sg	jiinbh6@gmail.com	DIT	I want to travel

- Editing of ISLP details

ISLP Submissions

staff1@rp.edu.sg
Log Out

Brunei ISLP



Deadline: 2/10/2025

Public Information: Join Brunei ISLP today! Visit and explore different traditions and enjoy delicious food.

More Details: Nice schedule.

Trip Dates: 2025-02-26, 2025-02-28, 2025-03-01

Schedule File: [View Schedule](#)

Trip Dates:

26/02/2025 Remove

28/02/2025 Remove

01/03/2025 Remove

[+ Add Trip Date](#)

Public Information:

Join Brunei ISLP today! Visit and explore different traditions and enjoy delicious food.

More Details:

Nice schedule.

Photo:

No file chosen

Schedule:

No file chosen

Save
Cancel

- Creating new ISLP trips

ISLP Submissions

P [+ Create](#)

ISLP (1 submissions)

ISLP (2 submissions)

Vietnam e-ISLP (0 submissions)

Create ISLP

ISLP Name:

Deadline: dd/mm/yyyy

Trip Dates: dd/mm/yyyy [+ Add Another Date](#)

Public Information:

Schedule File (Excel): Choose File | No file chosen

More Details:

Photo: Choose File | No file chosen

Create ISLP

[Back to Viewing Applications](#)

Add more trip dates

dd/mm/yyyy

Trip Dates:

dd/mm/yyyy [Remove](#)

dd/mm/yyyy [Remove](#)

dd/mm/yyyy [Remove](#)

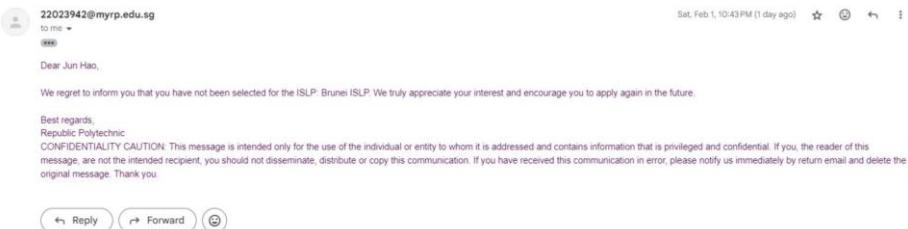
[+ Add Another Date](#)

Public Information:

- Approving and declining of students

Search Name Filter Approve (2) Decline (2)					Search Name Filter Approve (1) Decline (1)				
<input checked="" type="checkbox"/>	NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA	<input checked="" type="checkbox"/>	NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA
<input checked="" type="checkbox"/>	Jun Hao	[REDACTED]	cj7junhao@gmail.com	DIT	<input checked="" type="checkbox"/>	Jun Hao	22042810@myrp.edu.sg	cj7junhao@gmail.com	DIT
<input checked="" type="checkbox"/>	Jin Yi	[REDACTED]	jiinbh6@gmail.com	DIT	<input type="checkbox"/>	Jin Yi	22023942@myrp.edu.sg	jiinbh6@gmail.com	DIT
<input type="checkbox"/>	NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA	INTEREST	DOCUMENT	STATUS		
<input type="checkbox"/>	Jun Hao	22042810@myrp.edu.sg	[REDACTED]	DIT	Nice	View Document	Declined		
<input type="checkbox"/>	Jin Yi	22023942@myrp.edu.sg	[REDACTED]	DIT	I want to go to Brunei because I can help .:>	View Document	Pending		

- Students will receive an email through their personal email indicating that they have been accepted or rejected for the trip



Code Explanation

Functions

websity.py

connection_func() – is to connect to the mysql database to access the tables or insert new data or delete (Line 30)

```
def connection_func():
    connection = mysql.connector.connect(
        host=host,
        port=port,
        database=database,
        user=username,
        password=password
    )
    print("Connected to the database successfully!")
    return connection
```

login_required(f) – is to only allow people who logged in using their account can access certain pages (Line 84)

```
def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if 'user' not in session: #Check if user is logged in
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function
```

upload_to_s3(file, filename, folder) – is to upload pictures or documents to the folder in the S3 bucket so that the user can see the photos or the documents (Line 93)

```
#Upload folder to S3
def upload_to_s3(file, filename, folder):
    try:
        s3_file_path = f"{folder}/{filename}"
        #Upload the file with KMS encryption
        s3_client.upload_fileobj(
```

```

        file,
        S3_BUCKET,
        s3_file_path,
        ExtraArgs={
            'ServerSideEncryption': 'aws:kms',
            'SSEKMSKeyId': kms_key_id
        }
    )
    return s3_file_path
except Exception as e:
    print(f"Error: {e}")
    raise

```

delete_from_s3(file_path) – to delete the file in the folder of the S3. It is used for user who want to re-upload a new file which delete the existing file. (Line 115)

```

def delete_from_s3(file_path):
    try:
        s3_client.delete_object(Bucket=S3_BUCKET, Key=file_path)
        print(f"Deleted old file: {file_path}")
    except Exception as e:
        print(f"Error deleting file: {e}")

```

load_table(table) – to load the tables from the primarydatabase to let the user view the data within the tables (Line 123)

```

def load_table(table):
    connection = connection_func()
    cursor = connection.cursor(dictionary=True) #Use dictionary=True for dict
results
    cursor.execute("SELECT * FROM " +table+ ";")
    table = cursor.fetchall()

    #Close the connection
    cursor.close()
    connection.close()

    return table

```

send_email(recipient_email, subject, body) – it is to send the email to the students to see their confirmation of their submission or whether they are approved or rejected. (Line 136)

```

def send_email(recipient_email, subject, body):
    sender_email = "22023942@myrp.edu.sg"
    sender_password = "sikmo6-myqfyP-wujvik"

    msg = MIME Multipart()
    msg['From'] = sender_email

```

```

msg['To'] = recipient_email
msg['Subject'] = subject
msg.attach(MIMEText(body, 'plain'))

try:
    with smtplib.SMTP('smtp.office365.com', 587) as server:
        server.starttls()
        server.login(sender_email, sender_password)
        server.send_message(msg)
except Exception as e:
    print(f"Error sending email: {e}")

```

check_role(student_email, role_name) – since in the amazon cognito each user is in a group which is either staff or student. You can get the user's attribute from cognito and see which group they are in to allow them to have different access in the website (Line 182)

```

def check_role(student_email, role_name):
    try:
        #Get the user's attributes from Cognito
        response = cognito_client.admin_list_groups_for_user(
            UserPoolId='ap-southeast-1_C0vd8teW6', #Replace with your Cognito User
Pool ID
            Username=student_email
        )
        #Look for the group name in the user's group membership
        for group in response.get('Groups', []):
            if group['GroupName'] == role_name:
                return True

        #If group is not found, return False
        return False

    except ClientError as e:
        print(f"Error getting user from Cognito: {e}")
        return False

```

For Main Page (website.py) (Line 219) -> https://127.0.0.1:5000

Connect to the database to retrieve the tables islpdata to display

connect_func() function code

```

@app.route('/')
def base():
    user = session.get('user')
    connection = connection_func()
    cursor = connection.cursor(dictionary=True)

```

Fetching the ISLP trip data to display all the trips for public and users to see (website.py) (Line 233)

cursor.execute will execute a SQL query which in this case will select all the columns from the islodata table in the MySQL database.

	ISLP	deadline	public_information	schedule_file	more_details	photo_file
▶	Brunei ISLP	2025-02-10	Join Brunei ISLP today! Visit ...	schedules/Change.xlsx	Nice schedule.	images/batampic.jpg
	Batam ISLP	2024-10-31	Join the Batam International ...	schedules/Batam ISLP schedule.xlsx	This is the schedule.	images/batam.png
	YMCA Vietnam e-ISLP	2022-09-12	This is the second instalment ...	schedules/Vietnam schedule.xlsx	This is the schedule.	images/vietnam.png
✳	HULL	NULL	NULL	NULL	NULL	NULL

Then the fetchall() function fetches all the results and put it in a list of tuples where each tuples represent a row in the table.

The for loop will go through the islps and it will take the deadline from the row and format the deadline from 2025-02-10 to 10 February 2025.

Registration Deadline: 10 February 2025

It will check whether the deadline is over for the trip so that it will no longer let the students to apply for the trip.

```
#Fetch all ISLP data
    cursor.execute("SELECT * FROM islodata")
    islps = cursor.fetchall()
    for islp in islps:
        if islp.get('deadline'):
            deadline_date = islp['deadline']
            islp['formatted_deadline'] = deadline_date.strftime('%d %B %Y')
            islp['is_registration_open'] = deadline_date >= date.today()
```

Fetching the trip dates for each of the ISLP trip and putting them in past and present (website.py) (Line 241)

The cursor.execute will select the trip_date column from the tripdates table where the ISLP matches the ISLP from the islodata table when it goes through the for loop.

	id	ISLP	trip_date
▶	1	Batam ISLP	2025-03-24
	2	Batam ISLP	2025-03-25
	3	Batam ISLP	2025-03-26
	4	Batam ISLP	2025-03-27
	5	Batam ISLP	2025-03-28
	9	Brunei ISLP	2025-02-26
	11	Brunei ISLP	2025-02-28
	12	Brunei ISLP	2025-03-01
	13	YMCA Vietnam e-ISLP	2022-11-16
	14	YMCA Vietnam e-ISLP	2022-11-15

It will help to format the trip dates to make it easier for the user to read. For the first condition if there is only one trip date it will just be 01 March 2025.

But if it is not only one trip date, then it will check if the dates are consecutive. That means that it checks if the difference of the days is only one day apart then it will format in this manner 24 - 28 March 2025 if the month is the same. If the months are not the same, it will be put as 24 March – 1 April 2025. If the days are not consecutive it will be 26 February 2025, 28 February 2025, 01 March 2025. After that then it will store in the islp tuple the variable would be formatted_trip_dates to be

displayed on the public.html. Then to not let the users mix up which is active or not, the code will take the last trip date and see if it is not before today. Thus, putting the trips separate from past and present islp trip. ([Example](#))

```
present_islps = []
past_islps = []
for islp in islps:
    ...
        cursor.execute("SELECT trip_date FROM tripdates WHERE ISLP = %s ORDER BY trip_date", (islp['ISLP'],))
        trip_dates = [row['trip_date'] for row in cursor.fetchall()]
        if trip_dates:
            last_trip_date = trip_dates[-1] #Get the latest trip date
            if len(trip_dates) == 1:
                formatted_trip_dates = trip_dates[0].strftime('%d %B %Y')
            else:
                consecutive = all(
                    (trip_dates[i + 1] - trip_dates[i]).days == 1 for i in range(len(trip_dates) - 1)
                )
                if consecutive:
                    start_date, end_date = trip_dates[0], trip_dates[-1]
                    if start_date.month == end_date.month:
                        formatted_trip_dates = f"{start_date.day} - {end_date.day} {start_date.strftime('%B %Y')}"
                    else:
                        formatted_trip_dates = f"{start_date.strftime('%d %B')} - {end_date.strftime('%d %B %Y')}"
                else:
                    formatted_trip_dates = ', '.join(date.strftime('%d %B %Y'))
        for date in trip_dates)

        islp['formatted_trip_dates'] = formatted_trip_dates
        #Categorize into Present or Past
        if last_trip_date >= date.today():
            present_islps.append(islp)
        else:
            past_islps.append(islp)
    else:
        present_islps.append(islp)
```

In public.html, since it separated which ISLP trip is in the past and in the present it will be displayed in a for loop. (public.html) (Line 62)

For Present

```
<div class="content">
    <details open>
        <summary class="section-title">Present</summary>
```

```

<div class="content1">
    {% for islp in present_islps %}
        <h2 class="divheader">{{ islp.ISLP }}</h2>
        <div class="photo-button">
            <div class="photo-paragraph">
                
            </div>
            <div class="paragraph">
                <p class="roboto-regular">
                    {{ islp.public_information }}<br>
                    ☰ Trip Dates: {{ islp.formatted_trip_dates }} <br>
                    ✉ Registration Deadline: {{ islp.formatted_deadline }} <br>
                </p>
            </div>
        </div>
    ...

```

For past (public.html) (Line 92)

```

<details>
    <summary class="section-title">Past</summary>
    <div class="content2">
        {% for islp in past_islps %}
            <h2 class="divheader">{{ islp.ISLP }}</h2>
            <div class="photo-paragraph">
                
            </div>
            <div class="paragraph">
                <p class="roboto-regular">
                    {{ islp.public_information }}<br>
                    ☰ Trip Dates: {{ islp.formatted_trip_dates }} <br>
                </p>
            </div>
        ...

```

Check if the user is approved for the ISLP trip (website.py) (Line 274)

It takes the email and checks the submissions table in the primarydatabase and see if the student status is approved and it will send to the public.html to display the button to allow the user to see the form.html of the islp trip that they have been approved.

```

for islp in islps:
    ...
#Check if user is approved
    if user:
        email = user['email']
        cursor.execute("SELECT status FROM submissions WHERE email = %s AND
ISLP = %s", (email, islp['ISLP']))
        result = cursor.fetchone()

```

```

        islp['is_user_approved'] = result and result['status'] ==
'Approved'
    else:
        islp['is_user_approved'] = False

```

Since the user is approved, the public.html. The student will be able to click on the Learn More which will direct them to the learnmore.html (public.html) (Line 79) ([Example](#))

```

<div class="learnmore-signup">
    {% if islp.is_user_approved %}
        <a href="{{ url_for('learnmore', islp=islp.ISLP) }}" class="btn">Learn
More</a>
    {% endif %}

```

Fetch the slideshow images from S3 (website.py) (Line 283) ([Example](#))

It will fetch the images from the ‘slideshow’ folder in the S3 bucket and then store it in the slideshow_images to display it in the public.html

```

slideshow_images = []
try:
    response = s3_client.list_objects_v2(Bucket=S3_BUCKET,
Prefix='slideshow/')
    if 'Contents' in response:
        slideshow_images = [obj['Key'] for obj in response['Contents'] if
obj['Key'] != 'slideshow/']
    except Exception as e:
        print(f"Error fetching slideshow images: {e}")

```

It will retrieve image from the slideshow and do a for loop putting it into a slideshow. And using the /get-image/ route in the website.py to display the images in the slideshow (public.html) (Line 42)

```

<div class="slideshow">
    <div class="slideshow-container">
        {% for image in slideshow_images %}
            <div class="mySlides fade">
                
            </div>
        {% endfor %}

        <a class="prev" onclick="plusSlides(-1)"></a>
        <a class="next" onclick="plusSlides(1)">></a>
    </div>
    <div class="dotnumber">
        {% for image in slideshow_images %}
            <span class="dot" onclick="currentSlide(loop.index)"></span>
        {% endfor %}
    </div>
</div>

```

Getting images -> Example: <https://127.0.0.1:5000/get-image/?filename=images/batam.png>

It will take the argument in the link and the filename is images/batam.png and then retrieve the image from the S3 bucket and display it into the website. (website.py) (Line 305)

```
@app.route('/get-image/')
def get_image():
    image_key = request.args.get('filename')
    if not image_key:
        return "Filename parameter is required", 400
    try:
        response = s3_client.get_object(Bucket=S3_BUCKET, Key=image_key)
        image_content = response['Body'].read()
        mime_type, _ = mimetypes.guess_type(image_key)
        mime_type = mime_type or 'application/octet-stream'
        return Response(image_content, content_type=mime_type)
    except Exception as e:
        return str(e), 404
```

Redirecting the users to see different areas of the website (website.py) (Line 295)

It will check if the user email is a staff email or a student email by checking the end of the email and then direct to the website they should see. For the teachers, they will be directed to the staff.html where they will see all the submissions. For the students, they will see the public.html just like the users that are not signed in, but they have more features which is to sign up for the trips and be able to have access to the islp details if they are accepted to the trip.

```
if user:
    email = user['email']
    if email.endswith('@rp.edu.sg'):
        return redirect(url_for('staff'))
    else:
        return render_template('public.html', user=user,
present_islps=present_islps, past_islps=past_islps, user_email=email,
slideshow_images=slideshow_images)
    else:
        return render_template('public.html', present_islps=present_islps,
past_islps=past_islps, user=None, user_email="", slideshow_images=slideshow_images)
```

Once the student log in they will also be able to sign up for the ISLP Trips and see which account they are logged into.

Seeing the email they used to log in to (public.html) (Line 25)

```
<p class="sign-in">{{ user_email }}</p>
Being able to log out of the website (public.html) (Line 26)
```

```
{% if user %}
    <a href="{{ url_for('logout') }}" class="sign-in">← Log Out</a>
    {% else %}
```

When they press log out it will redirect them to this route. Which removes their session for the website and log out of their cognito account redirecting them back to the public.html and will not be able to access the functions that a student would see. (website.py) (Line 340)

```
@app.route('/logout')
def logout():
    #Clear the local session
    session.pop('user', None)

    #Construct Cognito logout URL
    cognito_logout_url = (
        "https://ap-southeast-1covid8tew6.auth.ap-southeast-
1.amazoncognito.com/logout?"
        "client_id=5beatj5fm3dfd30ldtm6g8418l&" 
        "logout_uri=https://127.0.0.1:5000/"
    )
    #Redirect to Cognito logout
    return redirect(cognito_logout_url)
```

Logging into the website

In the public.html there is a Sign in button on the upper right hand corner for the user to press to sign in (public.html) (Line 29)

```
<a href="{{ url_for('login') }}" class="sign-in">Sign In</a>
```

When they press it will direct the user to this page which is cognito's sign in page. It will initiate the login process by redirecting the user to the OAuth authorization endpoint. (website.py) (Line 323) ([Example](#))

```
#Login for staff and users
@app.route('/login')
def login():
    #Redirect to OAuth authorization endpoint (Cognito)
    return oauth.oidc.authorize_redirect("https://127.0.0.1:5000/authorize")
```

After successful authentication, the user will be redirected to the /authorize route where the access token is retrieved, the user information is stored inside the session and redirected back to the main page which is base() to determine the user role which is student or a staff or a guest. (website.py) (Line 329)

```
@app.route('/authorize')
def authorize():
    #Fetch the token after redirect
    token = oauth.oidc.authorize_access_token()
    user = token['userinfo']
    session['user'] = user
    #Redirect to the base route (public.html)
    return redirect(url_for('base'))
```

For Student

Applying for ISLP Trip submissions

In the main page section, it is mentioned how they can see the sign up button and the learn more button depending if the deadline has not exceed or they are accepted and the people who have not signed in cannot see.

```
<a href="{{ url_for('form', islp=islp.ISLP) }}>Sign up</a>
```

Hence, when the student press the button under the chosen ISLP they would be brought to the website -> <https://127.0.0.1:5000/form?islp=Brunei+ISLP>

The route uses the GET method to request data from the server to display the form. ([Example](#)) It will retrieve the user school email address based on the session, that stores the user. It will retrieve the islp from the button that belongs to the ISLP at the “a href” in public.html. Then, it will check if the user has already submitted the form before. If the user has submitted the form, it will take their previous submission and display it in the form. This is to prevent users from submitting duplicate form. Thus, they can resubmit the form if they want to change the details. (website.py) (Line 401)

```
#Form for ISLP
@app.route('/form', methods=['GET'])
@login_required
def form():
    user_email = session.get('user', {}).get('email', '')
    islp = request.args.get('islp')

    connection = connection_func()
    cursor = connection.cursor(dictionary=True)

    #Check if a submission exists for the user and ISLP
    cursor.execute(
        "SELECT * FROM submissions WHERE email = %s AND ISLP = %s",
        (user_email, islp)
    )
    submission = cursor.fetchone()

    cursor.close()
    connection.close()

    return render_template('form.html', user_email=user_email, islp=islp,
                           submission=submission)
```

Once it renders the template to the form.html putting in the information, it will first put in the school email that they sign in with and then put in the previous submission if they have submitted before if not it will be an empty form except for the school email portion.

The form will hold necessary information such as the islp they are applying, name, school email, personal email, diploma, reason of interest, upload file for their recommendation.

In form.html, the inputs will be in a form. Once the user press submit it will trigger the /upload action which directs them to the route and sends the data over to the database through the website.py. The POST method is to send data from the client to the server. The enctype attribute is to specify how the form data should be encoded which allows the form to include file uploads in this case is documents. It helps to ensure that the files are properly formatted and transmitted to the server along with the other form details. (form.html) (Line 95)

```
<form action="/upload" method="POST" enctype="multipart/form-data">
```

The islp is needed so that the data in the submissions table so that it knows that the student applied to which islp trip. The personal email is not changeable as it takes the email that they sign in with so it prevent users to put in other student's email. The rest of the inputs have the submission.____ if submission else "", which means it checks if the variable in the submission is empty, then if it is not empty, it will show the user's previous submission. If it is empty, the input would be blank for them. (form.html) (Line 99)

```
<input type="hidden" name="islp" value="{{ islp }}>

    <label for="name">Name *</label>
    <input type="text" id="name" name="name" placeholder="Your name"
           value="{{ submission.name if submission else '' }}" required>

    <label for="email">School Email *</label>
    <input type="email" id="email" name="email" placeholder="Your school email"
           value="{{ user_email }}" readonly
           required>

    <label for="personalemail">Personal Email *</label>
    <input type="email" id="personalemail" name="personalemail"
           placeholder="Your personal email"
           value="{{ submission.personal_email if submission else '' }}" required>

    <label for="diploma">Diploma *</label>
    <select id="diploma" name="diploma" required>
        <option value="DAAA" {% if submission and submission.diploma=='DAAA' %}selected{% endif %}>.....</option>
        <option value="DIT" {% if submission and submission.diploma=='DIT' %}selected{% endif %}>DIT</option>
    </select>

    <label for="interest">State Your Interest</label>
    <textarea id="interest" name="interest" rows="4">{{ submission.interest if submission else '' }}</textarea>

    <label for="testimonials">Upload File *</label>
    {% if submission and submission.file_name %}
    <p>
        Current file:
```

```

        <a href="/submissions/document?filename={{ submission.file_name }}">{{ submission.file_name }}</a>
    </p>
    {% endif %}
    <input type="file" id="testimonials" name="testimonials">
```

Once they submitted the form it goes to the route which is /upload. It will retrieve the data from the form. Then, it will check if the submission of the user sent is there and it will check if the file has changed if it has changed, it will delete the old file from the S3 bucket and upload a new file that they have recently uploaded into the S3 bucket. (website.py) (Line 423)

[delete_from_s3\(file_path\)](#), [upload_from_s3\(file, file.filename, S3_FOLDER\)](#)

```

@app.route('/upload', methods=['POST'])
@login_required
def upload():
    user_email = session.get('user', {}).get('email', '')
    islp = request.form.get('islp')
    form_data = request.form
    file = request.files.get('testimonials')

    connection = connection_func()
    cursor = connection.cursor(dictionary=True)

    try:
        #Check if a submission already exists
        cursor.execute(
            "SELECT * FROM submissions WHERE email = %s AND ISLP = %s",
            (user_email, islp)
        )
        submission = cursor.fetchone()

        #Handle file upload
        file_path = submission['file_path'] if submission else None
        file_name = submission['file_name'] if submission else None

        if file and file.filename:
            #Delete old file if it exists
            if file_path:
                delete_from_s3(file_path)

            #Upload new file
            file_path = upload_to_s3(file, file.filename, S3_FOLDER)
            file_name = file.filename
```

When the student resubmits the form it will update the student's data in the submissions table in primarydatabase in MySQL. Once it successfully submitted, it will send an email to the student's personal email address saying that they have successfully updated their submission. (website.py) (Line 455) ([Example](#))

```
send_email(recipient_email, subject, body)
```

```
if submission:
    #Update existing submission
    cursor.execute(
        """
        UPDATE submissions
        SET name = %s, diploma = %s, interest = %s, file_path = %s,
        file_name = %s, personal_email = %s
        WHERE email = %s AND ISLP = %s
        """,
        (
            form_data['name'],
            form_data['diploma'],
            form_data.get('interest', ''),
            file_path,
            file_name,
            form_data['personalemail'],
            user_email,
            islp
        )
    )

    body = f"""
Dear {form_data['name']},

You have successfully updated your submission. Here are the details we
received:
\t- Name: {form_data['name']}
\t- Email: {user_email}
\t- Personal Email: {form_data['personalemail']}
\t- Diploma: {form_data['diploma']}
\t- Interest: {form_data.get('interest', '')}
\t- ISLP: {islp}

Your uploaded file: {file.filename}

Best regards,
Republic Polytechnic
"""

    send_email(form_data['personalemail'], "Updated Submission
Confirmation", body)
```

It is the same as when the user applies the ISLP trip for the first time. However, when putting the data into the submissions table in primary data will be different (website.py) (Line 497)

```
cursor.execute(
    """
        INSERT INTO submissions (name, email, diploma, interest, ISLP,
status, file_path, file_name, personal_email)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """,
    (
        form_data['name'],
        user_email,
        form_data['diploma'],
        form_data.get('interest', ''),
        islp,
        'Pending',
        file_path,
        file_name,
        form_data['personalemail']
    )
)
```

After that it will refresh the data in the form for the user to see. (website.py) (Line 535)

```
return redirect(url_for('form', islp=islp))
```

Viewing of ISLP full details and schedule (For students who got approved for the ISLP Trips)

When the user presses this button it will lead them to

<https://127.0.0.1:5000/learnmore?islp=Batam+ISLP> (public.html) (Line 84)

```
<a href="{{ url_for('learnmore', islp=islp.ISLP) }}" class="btn">Learn More</a>
```

It will go to the /learnmore route which will retrieve the islp_name from islp=islp.ISLP and retrieve the ISLP data getting the more_details and schedule_file column from the islpdata table and only getting the islp that the user click on. With the schedule_path, it will use the file path and fetch the schedule excel file from the S3 bucket in the schedule folder. Then, it will read the excel sheet and convert it into a HTML table. (website.py) (Line 365)

```
@app.route('/learnmore')
@login_required
def learnmore():
    islp_name = request.args.get('islp')
    user_email = session.get('user', {}).get('email', '')
    connection = connection_func()
    cursor = connection.cursor(dictionary=True)
```

```

try:
    #Check if the student is approved for this ISLP
    cursor.execute("SELECT status FROM submissions WHERE email = %s AND ISLP = %s", (user_email, islp_name))
    submission_status = cursor.fetchone()

    if not submission_status or submission_status['status'] != 'Approved':
        return "You are not approved for this ISLP trip", 403 # Forbidden
access

    #Fetch the ISLP details
    cursor.execute("SELECT more_details, schedule_file FROM islpdata WHERE ISLP = %s", (islp_name,))
    islp_data = cursor.fetchone()

    if not islp_data:
        return "ISLP not found", 404

    #Fetch the schedule file from S3
    schedule_path = islp_data['schedule_file']
    response = s3_client.get_object(Bucket=S3_BUCKET, Key=schedule_path)
    schedule_content = response['Body'].read()

    #Parse the Excel file
    excel_data = pd.read_excel(BytesIO(schedule_content))
    #Replace '\n' with '<br>' in all columns and rows
    excel_data = excel_data.applymap(
        lambda x: x.replace('\n', '<br>') if isinstance(x, str) else x
    )
    #Convert to HTML table
    schedule_table = excel_data.to_html(index=False, classes='table table-bordered', escape=False)

finally:
    cursor.close()
    connection.close()

return render_template(
    'learnmore.html', more_details=islp_data['more_details'],
schedule_table=schedule_table, islp_name=islp_name)

```

Hence, displaying it nicely in the learnmore.html ([Example](#))

For Staff Page (website.py) (Line 545)

In the base() route, if the user email is a staff email then it will direct to this staff route which will let the staff see all the student submissions, ISLP details. They will be able to create, edit the ISLP details. ([Example](#))

```
@app.route('/staff', methods=['GET', 'POST'])
@login_required
def staff():
    user = session.get('user')
    user_email = user['email']

    if user_email.endswith('@rp.edu.sg'):
        connection = connection_func()
        cursor = connection.cursor(dictionary=True)

    try:
        #Fetch ISLP names and submission counts
        cursor.execute("""
            SELECT i.ISLP, COUNT(s.ISLP) AS submission_count
            FROM islpdata i
            LEFT JOIN submissions s ON i.ISLP = s.ISLP
            GROUP BY i.ISLP
        """)
        islps = cursor.fetchall()

    finally:
        cursor.close()
        connection.close()

    return render_template('staff.html', islps=islps, user_email=user_email)
else:
    return redirect(url_for('base'))
```

Sidebar in the staff page to View ISLP details and submissions

It is to let the staff to click on the ISLP trip allowing them to see the ISLP trips there are and their details and the student submissions. There is also a Create button to allow them to create new ISLP trips. The sidebar takes the ISLP name variable from the islp list of tuples and then see the submission count and display it like Batam ISLP (1 submissions). (staff.html) (Line 430)



```
<div class="sidebar">
    <div class="sidebar-header">
```

```

<h2 style="font-size: 20px;">ISLP</h2>
<a href="#" class="sidebar-button" onclick="switchToCreateISLP(); return false;">+ Create</a>
</div>
<ul>
    {% for islp in islps %}
    <li>
        <a href="#" data-location="{{ islp.ISLP }}>
            {{ islp.ISLP }} ({{ islp.submission_count }} submissions)
        </a>
    </li>
    {% endfor %}
</ul>
</div>

```

When the staff press onto the ISLP trip that they want to view, it will show the ISLP details and the student submission.

Once it senses the sidebar link in the list is clicked, it will fetch both ISLP details and student submissions. (staff.html) (Line 576)

```

document.querySelectorAll('.sidebar li a').forEach(link => {
    link.addEventListener('click', event => {
        switchToStaff()
        event.preventDefault();
        // Remove active class from all links
        document.querySelectorAll('.sidebar li a').forEach(a =>
a.classList.remove('active'));

        // Add active class to clicked link
        event.target.classList.add('active');
        islp = event.target.getAttribute('data-location');
        // Fetch both ISLP details and submissions
        fetchISLPDetails(islp);
        fetchSubmissions(islp);
    });
});

```

switchToStaff() (staff.html) (Line 561)

```

function switchToStaff() {
    document.getElementById('create-islp-container').style.display =
'none';
    document.getElementById('staff-container').style.display = 'block';
    document.getElementById('details-container').style.display = 'block';
}

```

fetchsubmissions(islp) function

It will fetch the route in website.py to show all the submissions data from the primarydatabase in MySQL. (staff.html) (Line 720)

```
function fetchSubmissions(islp) {
    console.log("Fetching submissions for:", islp);
    fetch(`/submissions/${islp}`)
```

The route /submission/<islp> will send the islp that the user wants to see over to the website.py and it will select the islp details for that specific islp from the islpdata table. Then, it will take the tripdates data for the specific islp. It will also generate the photo for the islp to let the user know what photo the islp is using. They also get to view what schedule is uploaded for that islp by using the url in the MySQL database and then generating the S3 bucket url for the staff to download the schedule file or view the photo. For viewing the students' submissions, it will retrieve all the student's submission for that ISLP and then put it into the response together with the islp details sending it back to the staff.html. (website.py) (Line 586)

```
@app.route('/submissions/<islp>')
@login_required
def view_submissions_by_islp(islp):
    user = session.get('user')
    user_email = user['email']

    if user_email.endswith('@rp.edu.sg'):
        connection = connection_func()
        cursor = connection.cursor(dictionary=True)

    try:
        #Fetch ISLP details
        cursor.execute("SELECT * FROM islpdata WHERE ISLP = %s", (islp,))
        islp_details = cursor.fetchone()

        if not islp_details:
            return jsonify({"error": "ISLP not found"}), 404

        #Fetch trip dates
        cursor.execute("SELECT trip_date FROM tripdates WHERE ISLP = %s",
        (islp,))
        trip_dates = [row['trip_date'].strftime('%Y-%m-%d') for row in
        cursor.fetchall()]
        islp_details['trip_dates'] = trip_dates

        #Generate S3 URLs for photo and schedule file
        islp_details['photo_url'] = s3_client.generate_presigned_url(
            'get_object', Params={'Bucket': S3_BUCKET, 'Key':
        islp_details['photo_file']}, ExpiresIn=3600
        )
        islp_details['schedule_url'] = s3_client.generate_presigned_url(
            'get_object', Params={'Bucket': S3_BUCKET, 'Key':
        islp_details['schedule_file']}, ExpiresIn=3600
```

```

        )

#Fetch submissions related to this ISLP
cursor.execute("SELECT * FROM submissions WHERE ISLP = %s", (islp,))
submissions = cursor.fetchall()

#Include submissions in the response
islp_details['submissions'] = submissions
print(islp_details)
return jsonify(islp_details)

finally:
    cursor.close()
    connection.close()

```

Once it sends the data by using jsonify. Back to staff.html, it will go to filterSubmissions() function (staff.html) (Line 722)

```

.then(response => response.json())
    .then(data => {
        if (data.submissions) {
            submissionsData = data.submissions; //Store data globally
            filterSubmissions(); //Apply filtering after fetching
        } else {
            console.error('No submissions found in response');
            submissionsData = [];
            filterSubmissions();
        }
    })
    .catch(error => {
        console.error('Error fetching submissions:', error);
        submissionsData = [];
        filterSubmissions();
    });
}

```

The filter submissions are to check if the search input for searching student's name has change or the staff filter the diploma. Then it will filter the data out. (staff.html) (Line 947) ([Example](#))

```

function filterSubmissions() {
    const searchQuery = document.getElementById('search-
name').value.toLowerCase();
    let filteredData = submissionsData; // Use the stored data

    // Filter by name
    if (searchQuery) {
        filteredData = filteredData.filter(submission =>
            submission.name.toLowerCase().includes(searchQuery)
    }
}

```

```

        );
    }

    // Filter by selected diplomas
    if (filteredDiplomas.length > 0) {
        filteredData = filteredData.filter(submission =>
            filteredDiplomas.includes(submission.diploma)
        );
    }
    renderTable(filteredData);
}

```

It will lead to `renderTable(submissions)` which will help to display the student's submissions (staff.html) (Line 759)

	NAME	SCHOOL EMAIL	PERSONAL EMAIL	DIPLOMA	INTEREST	DOCUMENT	STATUS	
□	Jun Hao	22042810@myrp.edu.sg	cj7junhao@gmail.com	DIT	Nice	View Document	Approved	✓ X
□	Jin Yi	22023942@myrp.edu.sg	jiinbh6@gmail.com	DIT	I want to go to Brunei because I can help. :>	View Document	Pending	✓ X

```

function renderTable(submissions) {
    const tableBody = document.getElementById('submissions-table').querySelector('tbody');
    tableBody.innerHTML = '';// Clear existing rows

    if (submissions.length === 0) {
        tableBody.innerHTML = '<tr><td colspan="8">No submissions found</td></tr>';
        return;
    }

    submissions.forEach(submission => {
        const row = `
<tr data-email="${submission.email}" data-islp="${submission.ISLP}">
    <td><input type="checkbox" class="select-row"></td>
    <td>${submission.name}</td>
    <td>${submission.email}</td>
    <td>${submission.personal_email}</td>
    <td>${submission.diploma}</td>
    <td>${submission.interest}</td>
    <td><a href="/submissions/document?filename=${submission.file_name}" target="_blank">View Document</a></td>
    <td>
        ${submission.status ? `<span style="color: ${submission.status === 'Approved' ? 'green' : 'red'}; font-weight: bold;">${submission.status}</span>` : `Pending`}
    </td>

```

```

        </td>
        <td>
            <button class="approve-btn"
                onclick="handleAction('${submission.email}', 'approve',
                '${submission.personal_email}', '${submission.ISLP}')">✓</button>
            <button class="decline-btn"
                onclick="handleAction('${submission.email}', 'decline',
                '${submission.personal_email}', '${submission.ISLP}')">X</button>
        </td>
    </tr>
    `;

    tableBody.innerHTML += row;
});
}

```

Then for the `fetchISLPDetails(islp)`, it will retrieve from the `/submission/<islp>`. Then it will take the islp details and display it on the website and there will be an edit button and a hidden edit container which allows the staff to edit the ISLP detail if they press the edit button. (staff.html) (Line 596)

The screenshot shows a web page titled "Brunei ISLP". It features a large image of a brightly lit mosque at night. Below the image, the page displays the following information:

- Deadline:** 2/10/2025
- Public Information:** Join Brunei ISLP today! Visit and explore different traditions and enjoy delicious food.
- More Details:** Nice schedule.
- Trip Dates:** 2025-02-26, 2025-02-28, 2025-03-01
- Schedule File:** [View Schedule](#)

At the bottom left, there is a green "Edit" button.

```

function fetchISLPDetails(islp) {
    fetch(`/submissions/${islp}`)
        .then(response => response.json())
        .then(data => {
            if (data.error) {
                document.getElementById('details-container').innerHTML =
`<p>${data.error}</p>`;
                return;
            }

            // Populate the details-container with ISLP details and edit
            functionality
            const detailsHTML = `
                <h2 style="text-align: center; margin-bottom: 20px; color:
                #4CAF50;">${data.ISLP}</h2>
                <div style="display: flex; align-items: flex-start; gap: 20px;">
                    <div style="flex: 1; max-width: 35%;">
                        .....

```

```

</div>
    ;
        document.getElementById('details-container').innerHTML =
detailsHTML;
....
```

View Document in the submission for the Staff to see

In the renderTable(submissions), when creating the html table at the View Document link. (staff.html) (Line 777)

```

<a href="/submissions/document?filename=${submission.file_name}"
target="_blank">View Document</a>
```

It will lead to this url which will have the argument of file name and it will direct to the route in the website.py.

<https://127.0.0.1:5000/submissions/document?filename=Jun%20Hao%20Testimonial.docx>

It will take the argument from the url which is the filename and find the data in the submissions table in the primarydatabase to find the file_path and the email. (website.py) (Line 633)

```

@app.route('/submissions/document')
@login_required
def view_document():
    filename = request.args.get('filename')
    user = session.get('user')
    user_email = user['email']

    if not user_email:
        return jsonify({"error": "Unauthorized access. Please log in."}), 401

    connection = connection_func()
    cursor = connection.cursor(dictionary=True, buffered=True)

    try:
        cursor.execute("""
            SELECT file_path, email
            FROM submissions
            WHERE file_name = %s
        """, (filename,))
        result = cursor.fetchone() #Ensure you fetch the result before closing the
cursor
```

If there results, it will check if the role of the user is a staff and if it is not it checks whether then user has the same email as the data itself. If not, they will be rejected but if they are they will be able to see the document. This will help to prevent people from seeing other people's document except for the staff. Instead of needing to download the document to view, they can access it in a temporary url. So, it takes the content from the S3 bucket and save he content to a temporary pdf file and display it. (website.py) (Line 658) ([Example](#))

```

if not result:
    return "Document not found", 404

#Check access rights
if not check_role(user_email, "staff") and result['email'] != user_email:
    return jsonify({"error": "You are not authorized to access this
document."}), 403

#Fetch the document from S3
response = s3_client.get_object(Bucket=S3_BUCKET, Key=result['file_path'])
document = Document()
with tempfile.NamedTemporaryFile(delete=False) as tmp_file:
    tmp_file.write(response['Body'].read())
    tmp_file_path = tmp_file.name
document.LoadFromFile(tmp_file_path)

#Convert to PDF
with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmp_pdf_file:
    tmp_pdf_file.close()
    document.SaveToFile(tmp_pdf_file.name, FileFormat.PDF)
    with open(tmp_pdf_file.name, 'rb') as pdf_file:
        pdf_content = pdf_file.read()

return Response(pdf_content, content_type='application/pdf')

except Exception as e:
    return jsonify({"error": str(e)}), 500

finally:
    cursor.close()
    connection.close()

```

Accepting and Declining students' submission

This is individual accepting and declining. When rendering the table using the `renderTable(submssions)` function that is activated by choosing the islp that the staff chose to view. At the part where the function creates a approve and decline button. (staff.html) (Line 784)



```

<button class="approve-btn" onclick="handleAction('${submission.email}', 'approve',
 '${submission.personal_email}', '${submission.ISLP}')">✓</button>
            <button class="decline-btn"
onclick="handleAction('${submission.email}', 'decline',
 '${submission.personal_email}', '${submission.ISLP}')">X</button>

```

When clicking on the approve button or the decline button, it will call the function which is handleAction(email, action, personal_email, islp). This will send the information of the student that the staff clicked on to the /submissions/action route. (staff.html) (Line 795)

```
async function handleAction(email, action, personal_email, islp) {
    const response = await fetch(`/submissions/action`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email, action, islp, personal_email })
    });
}
```

In the webssite.py, it will retrieve the data from the staff.html which is the student email, action (approve/decline), personal email and the islp trip that the student applied to. It will search the student submission in the submissions table in the primarydata and change the status column for the student submission to approve or decline. Then, it will send the email to the student to tell them if they are declined or approved. (website.py) (Line 698)

send_email(recipient_email, subject, body)

```
@app.route('/submissions/action', methods=['POST'])
@login_required
def handle_submission_action():
    user = session.get('user')
    user_email = user['email']

    if user_email.endswith('@rp.edu.sg'):
        data = request.get_json()
        email = data['email']
        action = data['action']
        islp = data['islp']
        personal_email = data['personal_email']

        #Load submissions and find the target submission
        connection = connection_func()
        cursor = connection.cursor(dictionary=True)

        #Fetch the submission from the database
        cursor.execute("SELECT * FROM submissions WHERE email = %s AND ISLP = %s;",
        (email, islp))
        submission = cursor.fetchone()

        if not submission:
            cursor.close()
            connection.close()
            return jsonify({"error": "Submission not found"}), 404

        #Update the status of the submission based on the action
        if action == 'approve':
            new_status = 'Approved'
```

```

        send_email(
            personal_email,
            "You have been selected!",
            f"Dear {submission['name']},\n\nCongratulations! You have been
selected for the ISLP: {islp}.\n\nBest regards,\nRepublic Polytechnic"
        )
    elif action == 'decline':
        new_status = 'Declined'
        send_email(
            personal_email,
            "Thank you for your application",
            f"Dear {submission['name']},\n\nWe regret to inform you that you
have not been selected for the ISLP: {islp}. "
            "We truly appreciate your interest and encourage you to apply again
in the future.\n\nBest regards,\nRepublic Polytechnic"
        )
    else:
        cursor.close()
        connection.close()
        return jsonify({"error": "Invalid action"}), 400
    #Update the submission's status in the database
    cursor.execute("UPDATE submissions SET status = %s WHERE email = %s AND
ISLP = %s;", (new_status, email, islp))
    connection.commit()
    #Close the connection
    cursor.close()
    connection.close()
    return jsonify({"success": True})

```

After the action is done, it will fetchSubmissions() to refresh the table for the staff to see if it has succeeded. (staff.html) (Line 802)

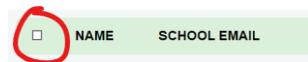
```

if (response.ok) {
    alert(`Action ${action} performed for ${email}`);
    fetchSubmissions();
} else {
    alert("Error performing the action");
}

```

When it comes to doing it in a bundle selection. The staff can use the tick boxes to select all or select the rows that they want to approve or decline together. ([Example](#))

At the top of the submission table there is a tick box that allows the staff to select all. (staff.html) (Line 483)



```
<th><input type="checkbox" id="select-all" class="select-all"></th>
```

Once clicked, it will trigger this and select all the tick boxes below to select all the rows. Then, it will show the approve and decline button. (staff.html) (Line 815)



```
let selectedRows = [];
document.getElementById('select-all').addEventListener('change', function () {
    const isChecked = this.checked;
    const checkboxes = document.querySelectorAll('.select-row');
    checkboxes.forEach(checkbox => checkbox.checked = isChecked);

    //Update selected rows
    selectedRows = isChecked ? [...checkboxes] : [];
    toggleActionButtons();
});
```

This will show the approve and decline button on the website and also show the number of rows the staff selected. (staff.html) (Line 847)

```
function toggleActionButtons() {
    const approveButtonContainer = document.querySelector('.approve-button');
    const declineButtonContainer = document.querySelector('.decline-button');
    const approveButton = approveButtonContainer.querySelector('button');
    const declineButton = declineButtonContainer.querySelector('button');

    if (selectedRows.length > 0) {
        //Enable buttons and show them
        approveButtonContainer.style.display = 'block';
        declineButtonContainer.style.display = 'block';

        //Optionally update the button text with the selected count
        approveButton.textContent = `Approve (${selectedRows.length})`;
        declineButton.textContent = `Decline (${selectedRows.length})`;
    } else {
        //Hide buttons if no rows selected
        approveButtonContainer.style.display = 'none';
        declineButtonContainer.style.display = 'none';
    }
}
```

Once the button is pressed it will call out the handleBulkAction(action) function. The action would be approved or decline. (staff.html) (Line 893)

```
document.querySelector('.approve-button button').addEventListener('click', () =>
handleBulkAction('approve'));
document.querySelector('.decline-button button').addEventListener('click',
() => handleBulkAction('decline'));
```

In the handleBulkAction function, it will get the rows and take the student's submission data one by one and execute the [handleAction\(email, action, personal_email, islp\)](#) function to send the email and to update the database. (staff.html) (Line 869)

```
function handleBulkAction(action) {
    const selectedCheckboxes = document.querySelectorAll('.select-
row:checked');

    selectedCheckboxes.forEach(checkbox => {
        const row = checkbox.closest('tr'); //Get the parent row
        const email = row.dataset.email;
        const personalEmail = row.querySelector('td:nth-
child(4)').textContent.trim();
        const islp = row.dataset.islp; //Get ISLP from dataset

        if (email && personalEmail && islp) {
            handleAction(email, action, personalEmail, islp);
        } else {
            console.error('Missing data for:', { email, personalEmail, islp });
        }
    });

    if (selectedCheckboxes.length === 0) {
        alert("Please select at least one row.");
    }
}
```

Editing ISLP Trips

At the [fetchISLPDetails\(islp\)](#) function, when it creates the ISLP details container. There is a hidden edit form so when the staff press on edit, it will show the editform below the details, allowing the staff to edit the details of the ISLP trip ([Example](#))

When pressing the edit button. (staff.html) (Line 620)

```
<button id="edit-button" style="margin-top: 20px; padding: 10px 20px; background-
color: #4CAF50; color: white; border: none; border-radius: 5px; cursor:
pointer;">Edit</button>
```

It will show the edit form. (staff.html) (Line 673)

```
document.getElementById('edit-button').addEventListener('click', function () {
    document.getElementById('edit-button').style.display =
'none';
    document.getElementById('edit-form').style.display =
'block';
});
```

Once it reveals the form, the staff can edit the ISLP details. They can only change, the trip dates, details, more details, photo and the schedule file. I didn't include the edit of deadline as if the staff frequent change it, it can confuse or discourage applicants, so ensure it's essential and communicate it clearly. However, if they need to change, they can access the MySQL and change it from there.

For trip date they can modify the dates and add new trip dates or remove the current ones. (staff.html) (Line 624)

Trip Dates:

26/02/2025	<input type="button" value="Remove"/>
28/02/2025	<input type="button" value="Remove"/>
01/03/2025	<input type="button" value="Remove"/>

+ Add Trip Date

```
<label for="trip-dates" style="display: block; margin-bottom: 10px; font-weight: bold;">>Trip Dates:</label>
    <div id="trip-dates-container" style="margin-bottom: 20px;">
        ${data.trip_dates.map((date, index) => `
            <div style="display: flex; align-items: center; margin-bottom: 10px;" id="trip-date-${index}">
                <input type="date" name="trip_dates" value="${date}" required style="flex: 1; padding: 10px; border: 1px solid #ddd; border-radius: 5px; margin-right: 10px;">
                <button type="button" onclick="removeTripDate(${index})" style="padding: 5px 10px; background-color: #f44336; color: white; border: none; border-radius: 5px; cursor: pointer;">Remove</button>
            </div>`).join('')}
        </div>
        <button type="button" onclick="addTripDateInput()" style="padding: 10px 20px; background-color: #4CAF50; color: white; border: none; border-radius: 5px; cursor: pointer; margin-bottom: 20px;">+ Add Trip Date</button>
```

The staff can modify the public detail and the more details (for accepted students) (staff.html) (Line 634)

```
<label for="details" style="display: block; margin-bottom: 10px; font-weight: bold;">>Public Information:</label>
    <textarea id="details" name="details" style="width: 97%; padding: 10px; border: 1px solid #ddd; border-radius: 5px; margin-bottom: 20px;">${data.public_information}</textarea>

    <label for="more-details" style="display: block; margin-bottom: 10px; font-weight: bold;">>More Details:</label>
    <textarea id="more-details" name="more_details" style="width: 97%; padding: 10px; border: 1px solid #ddd; border-radius: 5px; margin-bottom: 20px;">${data.more_details}</textarea>
```

Staff can upload new schedule file and photo for the ISLP trip. (staff.html) (Line 640)

```
<label for="photo-file" style="display: block; margin-bottom: 10px; font-weight: bold;">Photo:</label>
    <input type="file" id="photo-file" name="photo-file" accept="image/*" style="width: 97%; padding: 10px; border: 1px solid #ddd; border-radius: 5px; margin-bottom: 20px;">

    <label for="schedule-file" style="display: block; margin-bottom: 10px; font-weight: bold;">Schedule:</label>
    <input type="file" id="schedule-file" name="schedule-file" accept=".xls,.xlsx" style="width: 97%; padding: 10px; border: 1px solid #ddd; border-radius: 5px; margin-bottom: 20px;">
```

There are two buttons at the end which is save or cancel. (staff.html) (Line 647)

```
<button id="save-button" style="padding: 10px 20px; background-color: #4CAF50; color: white; border: none; border-radius: 5px; cursor: pointer;">Save</button>
    <button id="cancel-button" style="padding: 10px 20px; background-color: #f44336; color: white; border: none; border-radius: 5px; cursor: pointer;">Cancel</button>
```

The cancel button will hide the form away in the ISLP detail container which will not let the staff see the form again. (staff.html) (Line 678)

```
document.getElementById('cancel-button').addEventListener('click', function () {
    document.getElementById('edit-form').style.display = 'none';
    document.getElementById('edit-button').style.display = 'inline';
});
```

For the save button, it will retrieve all the information from the form and send it to /update-details route.

```
document.getElementById('save-button').addEventListener('click', async function () {
    const formData = new FormData();
    formData.append('islp', data.ISLP);
    formData.append('details',
    document.getElementById('details').value);
    formData.append('more_details',
    document.getElementById('more-details').value);
    formData.append('photo', document.getElementById('photo-file').files[0]);
    formData.append('schedule',
    document.getElementById('schedule-file').files[0]);
```

```

        const tripDates =
Array.from(document.querySelectorAll('input[name="trip_dates"]'))
    .map(input => input.value);
formData.append('trip_dates', JSON.stringify(tripDates));

try {
    const response = await fetch('/update-details', {
        method: 'POST',
        body: formData,
    });
}

```

In website.py, it will retrieve all the data from the form in staff.html, it will update all the details. For the photo and the schedule file, it will delete the existing file and upload a new one to save space in the S3 bucket. (website.py) (Line 799)

```

@app.route('/update-details', methods=['POST'])
@login_required
def update_details():
    user = session.get('user')
    user_email = user['email']

    if user_email.endswith('@rp.edu.sg'):
        islp = request.form.get('islp') #ISLP identifier
        details = request.form.get('details')
        more_details = request.form.get('more_details')
        photo_file = request.files.get('photo')
        schedule_file = request.files.get('schedule')

        trip_dates = json.loads(request.form.get('trip_dates', '[]'))
        trip_dates = [date for date in trip_dates if date.strip()]

        connection = connection_func()
        cursor = connection.cursor(dictionary=True) #Enable dictionary mode

        try:
            [Remove Existing Data]

            cursor.execute(
                "UPDATE islpdata SET public_information = %s, more_details = %s
WHERE ISLP = %s",
                (details, more_details, islp)
            )

            cursor.execute("DELETE FROM tripdates WHERE ISLP = %s", (islp,))
            for trip_date in trip_dates:
                cursor.execute(
                    "INSERT INTO tripdates (ISLP, trip_date) VALUES (%s, %s)",
                    (islp, trip_date)
                )
        
```

```

        )

        if photo_file and photo_file.filename:
            if existing_photo_file:
                delete_from_s3(existing_photo_file)

            photo_path = upload_to_s3(photo_file, photo_file.filename,
S3_IMAGE_FOLDER)
            cursor.execute(
                "UPDATE islpdata SET photo_file = %s WHERE ISLP = %s",
                (photo_path, islp)
            )

[schedule_file same function as photo_file]

connection.commit()
return jsonify({"success": True}), 200

```

Then the data will be reloaded with the new ISLP detail on the website. (staff.html) (Line 701)

```

if (response.ok) {
    alert('Details updated successfully!');
    fetchISLPDetails(data.ISLP); // Reload details
after saving
} else {
    alert('Failed to update details.');
}
} catch (error) {
    console.error('Error updating details:', error);
    alert('An error occurred.');
}
});
});
```

Creating ISLP trips

When the staff wants to create a new ISLP trip, they will have to press the create button and it will trigger the switchToCreateISLP() function without needing to go to another url page. (staff.html) (Line 433)

ISLP

+ Create

```

<a href="#" class="sidebar-button" onclick="switchToCreateISLP(); return false;">+
Create</a>
```

This will display the create container and switch off the other container to prevent the other container from overlaying the display. (staff.html) (Line 555)

```

function switchToCreateISLP() {
    document.getElementById('staff-container').style.display = 'none';
    document.getElementById('details-container').style.display = 'none';
    document.getElementById('create-islp-container').style.display =
'block';
}

```

The form will be displayed in this manner. The staff will be able to input the ISLP Name, deadline, trip dates, public information, schedule file, more details and the photo file. Once they submit it will go to the /createislp route in the website.py. (staff.html) (Line 499)

```

<div id="create-islp-container" style="display: none;">
    <h1>Create ISLP</h1>
    <form method="POST" action="/createislp" enctype="multipart/form-data">
        <label for="islp">ISLP Name:</label>
        <input type="text" id="islp" name="islp" required><br><br>

        <label for="deadline">Deadline:</label>
        <input type="date" id="deadline" name="deadline" required><br><br>

        <label for="trip_dates">Trip Dates:</label>
        <div id="trip-dates-container">
            <input type="date" name="trip_dates" required>
        </div>
        <button type="button" onclick="addTripDateInput()">+ Add Another Date</button><br><br>

        <label for="public_information">Public Information:</label>
        <textarea id="public_information" name="public_information"></textarea><br><br>

        <label for="schedule_file">Schedule File (Excel):</label>
        <input type="file" id="schedule_file" name="schedule_file" accept=".xls,.xlsx" required><br><br>

        <label for="more_details">More Details:</label>
        <textarea id="more_details" name="more_details"></textarea><br><br>

        <label for="photo_file">Photo:</label>
        <input type="file" id="photo_file" name="photo_file" accept="image/*" required><br><br>

        <button type="submit">Create ISLP</button>
    </form>
    <button onclick="switchToStaff()">Back to Viewing Applications</button>
</div>

```

In the /createislp route, it will take all the information from the form. It will upload the files to the S3 bucket and insert the islp data into the islpdata table and for the trip date will be separately put in the tripdates table. (website.py) (Line 755)

```

@app.route('/createislp', methods=['GET', 'POST'])
@login_required
def createislp():
    user = session.get('user')
    user_email = user['email']

    if user_email.endswith('@rp.edu.sg'):
        if request.method == 'POST':
            islp = request.form['islp']
            deadline = request.form['deadline']
            public_information = request.form['public_information']
            schedule_file = request.files['schedule_file']
            more_details = request.form['more_details']
            photo_file = request.files['photo_file']
            trip_dates = request.form.getlist('trip_dates') #Get list of trip
dates
            #Save files to a location (e.g., S3 or local folder)
            schedule_path = upload_to_s3(schedule_file, schedule_file.filename,
S3_SCHEDULE_FOLDER)
            photo_path = upload_to_s3(photo_file, photo_file.filename,
S3_IMAGE_FOLDER)

            #Save ISLP data to the database
            connection = connection_func()
            cursor = connection.cursor()
            try:
                cursor.execute("""
                    INSERT INTO islpdata (ISLP, deadline, public_information,
schedule_file, more_details, photo_file)
                    VALUES (%s, %s, %s, %s, %s, %s)
                """, (islp, deadline, public_information, schedule_path,
more_details, photo_path))
                #Save trip dates to the database
                for trip_date in trip_dates:
                    cursor.execute("""
                        INSERT INTO tripdates (ISLP, trip_date)
                        VALUES (%s, %s)
                    """, (islp, trip_date))

                connection.commit()
            finally:
                cursor.close()
                connection.close()
            flash("ISLP and trip dates created successfully!")
            return redirect(url_for('staff'))

```

Extra Information

Patching of MySQL can be modified in the primary database to do daily maintenance.

The screenshot shows the 'Modify DB instance: primary-database' page in the AWS RDS console. Under the 'Maintenance' section, there is a checkbox for 'Enable auto minor version upgrade'. Below it, a 'DB instance maintenance window' is set from Saturday 21:15 UTC to Sunday 00:15 UTC for a duration of 0.5 hours. Other settings like 'DB engine version' and 'Start day' are also visible.

However, if you want to do it in the AWS CLI. You can use this in your command prompt.

```
aws rds modify-db-instance \
--db-instance-identifier your-instance-id \
--auto-minor-version-upgrade \
--preferred-maintenance-window "Sun:03:00-Sun:06:00"
```

Or if you want to do it immediately use.

```
aws rds modify-db-instance --db-instance-identifier your-instance-id --engine-version new-version --
apply-immediately
```

If you want to change the photos in the slideshow, you can upload the photos in the slideshow folder in S3.

The screenshot shows the AWS S3 console with a bucket named 'submissionsbuckettyp'. Inside the 'submissionsbuckettyp' bucket, there is a folder named 'slideshow'. A large group photo of people holding a 'REPUBLIC POLYTECHNIC' banner is displayed. Below the photo, the S3 object list for 'slideshow/' is shown, containing six files named 'photo1.png' through 'photo8.png', all of which are PNG files with sizes ranging from 69.1 KB to 98.0 KB and were last modified on February 1, 2025.

Name	Type	Last modified	Size	Storage class
photo1.png	png	February 1, 2025, 16:53:10 (UTC+08:00)	69.1 KB	Standard
photo2.png	png	February 1, 2025, 16:53:11 (UTC+08:00)	58.0 KB	Standard
photo3.png	png	February 1, 2025, 16:53:11 (UTC+08:00)	61.5 KB	Standard
photo4.png	png	February 1, 2025, 16:53:11 (UTC+08:00)	384.6 KB	Standard
photo5.png	png	February 1, 2025, 16:53:11 (UTC+08:00)	98.0 KB	Standard
photo6.png	png	February 1, 2025, 16:53:12 (UTC+08:00)	80.8 KB	Standard
photo7.png	png	February 1, 2025, 16:53:10 (UTC+08:00)		
photo8.png	png	February 1, 2025, 16:53:10 (UTC+08:00)		