# Email OTP module task

## Task

1. To implement a secure email OTP module that can be used for our enterprise application. You are free to use any standard library from the language which you choose to implement the test.

2. Do document any assumptions that you make.

3. Describe how you would test your module.

## pseudo code

Class Email\_OTP\_Module {

/\* start can be called after an instance of Email OTP is constructed. Can be used to initialise variables.

\*/

function start() {

//optional to implement

}

/\* close can be called after an instance of Email OTP is to be remove from the application.

\*/

function close() {

//optional to implement

}

/\*

@func generate\_OTP\_email sends a new 6 digit random OTP code to the given email address input by the users. Only emails from the ".dso.org.sg" domain should be allowed to receive an OTP code.

You can assume a function send\_email(email\_address, email\_body) is implemented.

Email body to the user should be in this format "You OTP Code is 123456. The code is valid for 1 minute"

@param user\_email is an email address entered by the user.

@returns the following status code (assume implemented as constants)

STATUS\_EMAIL\_OK: email containing OTP has been sent successfully.

STATUS\_EMAIL\_FAIL: email address does not exist or sending to the email has failed.

STATUS\_EMAIL\_INVALID: email address is invalid.

\*/

function generate\_OTP\_email(string user\_email) {

let otp = '';

let email\_body = 'You OTP Code is '

let remainder\_email\_body = '. The code is valid for 1 minute'

for (let i = 0; i < 6; i++) {

otp += Math.floor(Math.random() \* 10);

}

//set variable otp to the one generated

this.otp = otp

//timeout otp by setting it to 'expired'

setTimeout(() => {

this.otp = 'expired'

}, 60000)

if(user\_email.trim().includes('.dso.org.sg')){

if (this.send\_email(user\_email.trim(), email\_body + otp + remainder\_email\_body)) {

return STATUS\_EMAIL\_OK

}

else {

return STATUS\_EMAIL\_FAIL

}

}

else {

return STATUS\_EMAIL\_INVALID

}

}

/\*

@func check\_OTP reads the input stream for user input of the OTP. The OTP to match is the current OTP generated by a send

allows user 10 tries to enter the valid OTP. check\_OTP should return after 1min if the user does not give a valid OTP.

@param input is a generic IOstream. It implements input.readOTP() which waits and returns the 6 digit entered by the user. this function call is blocking so you might need to wrap it in a timeout.

@returns the following status code (assume implemented as constants)

STATUS\_OTP\_OK: OTP is valid and checked

STATUS\_OTP\_FAIL: OTP is wrong after 10 tries

STATUS\_OTP\_TIMEOUT: timeout after 1 min

\*/

async function check\_OTP(iostream input) {

//not sure how to handle the blocking input.readOTP() but either stop the reading after 60 seconds and return STATUS\_OTP\_TIMEOUT or base it off otp variable being 'expired'

const otp = await input.readOTP()

if (otp === 'expired') {

return STATUS\_OTP\_TIMEOUT

}

if (otp === this.otp){

this.attempt = 0;

this.otp = null

return STATUS\_OTP\_OK

}

else {

this.attempt++

check\_OTP(input)

//not sure if input.readOTP() will finish upon sending the OTP, so will recall the method here

}

if(this.attempt === 10) {

this.attempt = 0;

this.otp = null

return STATUS\_OTP\_FAIL

}

}

}

Assumptions made: 1. send\_email will return a true/false response upon calling it to indicate whether the email was successfully sent

2. otp and attempts variable present in the module for it to remember and track the otp and attempts made by user

3. input.readOTP() will end upon receiving a 6 digit otp, hence requiring the check\_OTP method to be called again after every attempt

How to test module: For generate\_OTP\_email method, I will use multiple types of emails, 1 valid ( including '.dso.org.sg'), 1 invalid (not including '.dso.org.sg'), 1 doesn't exist ( to get the failure to send)

For check\_OTP, letting the time run to 60 seconds and testing it keying in the correct OTP, keying in the correct OTP in 60 seconds, keying the wrong OTP for multiple times, up to 10 times in 60 seconds.

These will cover all the cases of all the different returns required in the requirements