TwinCities Inventory Management System

Date: 15 January 2019

Version: 1.1

Authors:

Hilton Mthimunye (Hilton)

Keegan Kinnear (Monty)

Index

1.Technical Terms, Acronyms and Abbreviations

TERM	DEFINITION
API	Application Programming Interface
CMC	Case Management Console
IMS	Inventory Management System
CSP	Caché Server Page
EAO	Emoluments Attachment Order (Garnishee Order)
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
In scope	Functions to be tested
LAN	Local Area Network
Management console	The user interface belonging to any user logged into the IMS
Out of scope	Functions not to be tested
PC	Desktop Computer
SQL	Structured Query Language
System	Refers to the IMS in this document
Uniform Resource Identifier	Defines the location of a specific resource
UnitTest	A unit test is a test of the correctness of an individual code module, for example, a test of a method or class. A typical unit test is a method that executes the method that it tests and verifies that the method generates the correct output for a given set of inputs.
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Http status code	Description
200 OK	Standard response for successful HTTP requests.
400 Bad Request	The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing).
401 Unauthorized	Like 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. 401 semantically means "unauthenticated", i.e. the user does not have the necessary credentials.

403 Forbidden	The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort.
404 Not Found	The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.
500 Internal Server Error	A generic error message, given when an unexpected condition was encountered and no more specific message is suitable

2.Project Description

The IMS is a web-based application used for managing inventory stock within TwinCies.

3.Project Scope

1. The TwinCies EAO department will use this Web Application to utilise most of the functionality which will be provided by this system. TwinCies uses multiple client inventory systems to handle client related matters.
2. Some expectations from the system:
    2.1. Decrease TwinCies's dependency on hard copy paperwork;
    2.2. In the long term, eliminate TwinCies's dependency on physical interaction. This may take some time;
    2.3. Manage cases for every TwinCies product;
    2.4. Consolidate multiple client inventory systems of any TwinCies branch into a single system, especially for the EAO team.

4.Requirement Analysis

The purpose of the requirements analysis is to determine the user's expectations for the system. The objective of the requirements analysis is the identification and description of the system requirements.

The ability to manage **Users**:

- Type of users
    - Administrators: Users who can access the entire system. There must be at least one Administrator for accessing the entire data and features.
    - Standard Users: Users who can access data according to the defined permissions (profiles) and roles
- Setup of user roles
    - Every user is given a role and profile. The profile determines what features a user can access and the role determines what data the user can access.
    - Access should be configured for each action (Read, Add, Update, Delete, Export, Import)
- Manage Users (add, delete, amend)
- TwinCies users to queues

The ability to manage **Contacts**:

- Manage contacts (add, delete, amend)
- Categorise contacts
- Ability to communicate in multiple channels i.e. email, fax, Facebook, Twitter, WhatsApp, telephonically, etc.
- Centralized contacts
- Track everything in one place
- Know what you said last time
- Know when to follow up

The ability to manage **Campaigns**:

- Manage campaigns (add, delete, amend)
- Set campaign workflow
    - Steps to follow in workflow
    - Rules to apply in workflow
    - Rules to apply in steps
    - Timers on steps to escalate or re-delegate
    - Ability to set up custom/dynamic screens for display or capturing of pertinent info (can be *scripts* to be read by call agent, or fields to complete or tick boxes to tic during a step in a campaign)

The ability to manage **Queues**:

- Manage queues (add, delete, amend)
- Queued work distribution process

The ability to **Report** on different levels:

- Reporting on Campaigns
- Reporting on Queues
- Reporting on user productivity

5.Functional Specifications

5.1. Purpose of this Document

The purpose of this document is to provide brief information about the types of users who will be using the CRM and what they must do to gain access to the CRM. This document covers registration, verification, login and password reset for all system users.

5.2. Login Interface

Figure 1. Screenshot of login page

Business Logic

The login interface (*Figure 1*) does the following for all system users:

1. The user inputs their email address and password.
2. The user must click on the **Login** button which will verify and validate all the user's input:
   2.1. If the user input is invalid, the user must try again.
   2.2. If the user input is valid, the user is redirected to their relevant management console.
3. This is the process for a user inputting the incorrect password:
   3.1. If the user inputs the incorrect password, they will be given a maximum of four tries until they are blocked from the system.
   3.2. If the user inputs the correct password before the fourth try, they will gain access into their CMC.
   3.3. If he user reaches the limit of tries then they will be notified that their account has been blocked and they will receive an email that contains:
      3.3.1. A new reset code
      3.3.2. A TwinCies that takes them to the reset password interface
4. The login interface consists of hyperTwinCiess to the registration interface (*Register Account*) and the forgot password interface (*Forgot Password?*)

Front End (JavaScript)

Look and Feel

Element	CSS
"Welcome to TwinCies's IMS" Header	Color: Gray (#cccccc) Font-Family: Arial Rounded MT Bold
"Login" Header	Color: WhiteSmoke Font-Family: Arial Rounded MT Bold
Email Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when

	input requirements are met. Input requirements can be found below in the input requirements table.
Password Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Submit Button	
"Register account" TwinCies	Color: Lime Green (#bcff75)
"Forgot Password" TwinCies	Color: Lime Green (#bcff75)
Footer Text	Color: WhiteSmoke Font-Family: Arial

Input Requirements

Input	Purpose	Requirements
Email Address Input	To obtain the user's login credentials.	Email must contain '@'
Password Input	To obtain the user's login credentials.	Password length must be between 1-15 characters
Submit Button	The submit button will trigger backend functions to validate the user's credentials and redirect the user to the appropriate management console if there are no errors.	The button will be disabled until the requirements for the email and password input fields have been met.
"Register Account" TwinCies	To redirect the user to the registration page	N/A
"Forgot Password" TwinCies	To redirect the user to the forgot password page	N/A

Security layer (PHP)

The input is received from the front-end side is a JSON object which is then decoded into an array. The input is sent to validation methods, which should be successful, to allow the user to login into their management console.

The validation methods are as follows:

- *public function GetLoginInfo(Request $request)*
  This method receives input from the user interface and does the following:
  - Check if all the required fields are inputted

    - Calls the other validation functions and passes the array of input to them
- *public function CheckIfEmailIsCorrect(array $LoginJson)*

This method checks if the email address that was inputted is a valid TwinCies email account.

- *public function CheckIfUserExists(array $LoginJson)*

This method sends the email address to the database to check if the user exists in the database. If the user does exist, the database will return a JSON object containing of information that correlates to the user, including the hashed password and a token that will be used for the duration of their session.

- *public function CheckIfPasswordsMatch(array $LoginJson)*

This method checks if the password inputted and the hashed password received from the database match each other. This is what this method will return:

    - An error message if the password is incorrect, regardless of if the password meets the requirements.
    - An error message if the user has been blocked from the system.
    - A JSON consisting of the session token and user ID amongst other information needed for the user's CMC.

- *public function SendErrorMessage(array $LoginJson)*

This method sends an email to the user with a new reset code and a TwinCies to the reset password interface. The message states to the user that an unknown user is trying to access their account and that their account will be blocked until they reset their password.

Back-end (Laravel)

When a user logs in, the back-end side will return a password to the security layer to validate if the entered password matches with the database password or an http error code. This process happens as follows:

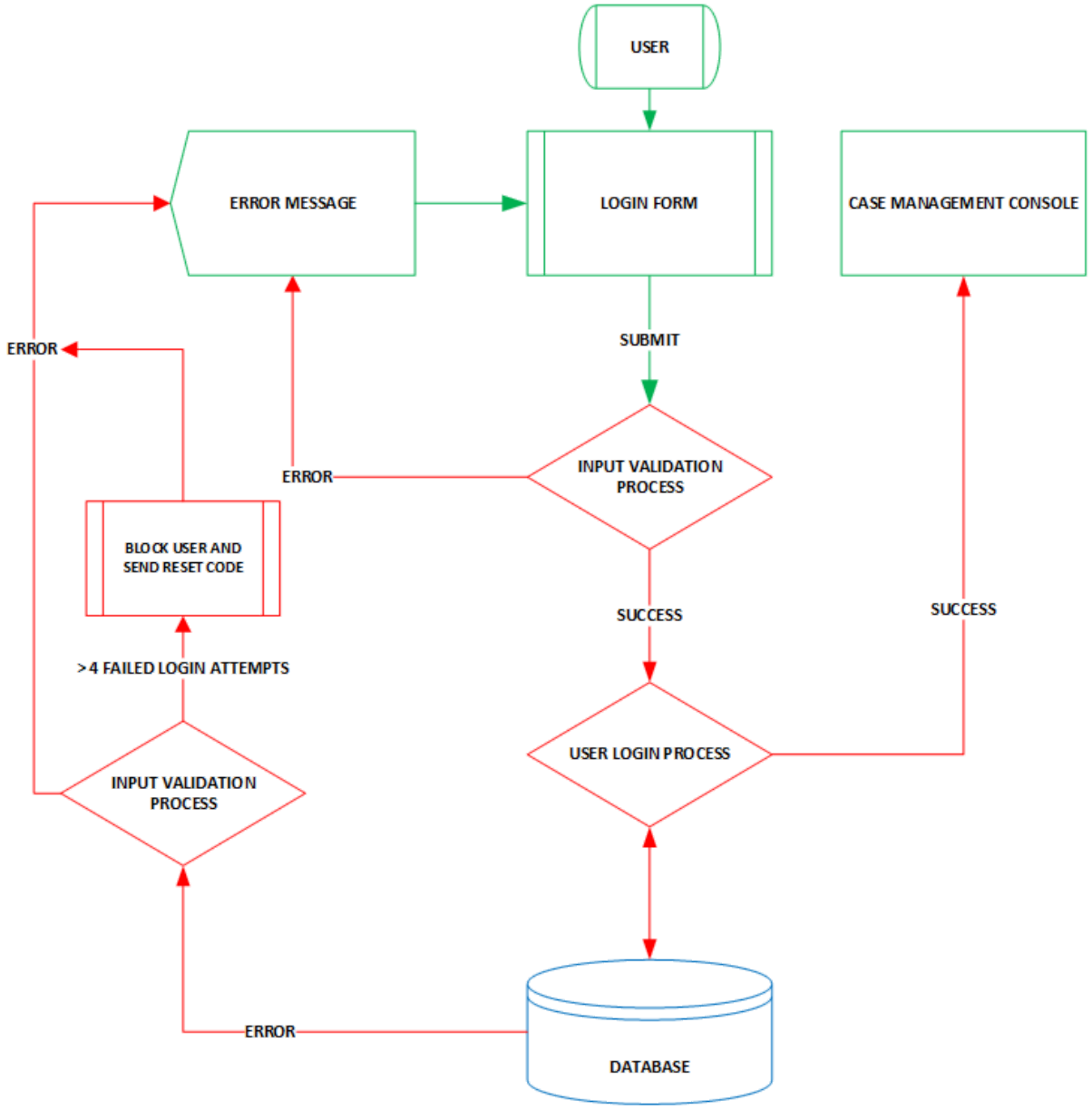
1. The system receives an email address and should check if the user exists in the database and returns on of the two options:
    a. An error message, stating that the user does not exist, if the email address was not found in the database.
    b. The system will return the password that corresponds with the email address.
2. The system will return a UserId, Token, Password and Job Title that is TwinCiesed to the email address, so that information can be easily accessed without having to keep receiving and using an email address to search for a user's information.

ClassMethods in the login process:

- *ClassMethod UserLogin(email As %String) As %Status*
    - This method will return a UserId, Token, Password and Job Title
    - Method will also check if a user is blocked from using the system, if blocked, return a 403 error
- *ClassMethod PasswordWordError(email As %String) As %Status*

- As part of the login process, this method will increase the Error Password Counter (checks failed login attempts)
- If the counter is more than 4 failed login attempts, the user will be blocked from logging in and will be sent a reset code to their email account.
- After successfully resetting the account, the limitation will be removed
- If user is blocked from logging in, this method will not send an email, but a Forbidden error until the user resets the password view the email sent earlier

Flow Diagram: Login Process



5.3. Registration Interface

Figure 2. Screenshot of registration page

Figure 3. Password requirements

Business Logic

The user registration interface (*Figure 2)* does the following for all users:

1. The user inputs their email address, verification code (which they received in an email) and their own personal password.
2. The user must click on the **Complete Registration** button which will verify and validate all the user's input:
   2.1. If the user input is invalid, the user must try again.
   2.2. If the user input is valid, the user is redirected to the login interface.
3. The registration interface contains a hyperTwinCies to the login interface (*Click here to login*)

Front End (JavaScript)

Look and Feel

Element	CSS
"Welcome to TwinCies's IMS" Header	Color: Gray (#cccccc) Font-Family: Arial Rounded MT Bold
"Registration" Header	Color: WhiteSmoke Font-Family: Arial Rounded MT Bold
Email Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Verification Code Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Password Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when input requirements are met. Input

	requirements can be found below in the input requirements table.
Password requirement text (Can be seen in *Figure 3*)	When a specific requirement is not met, the text color will be: Color: Bright Orange (#FF8300) When a specific requirement is met, the text color will be: Color: Lime Green (#bcff75)
Confirm Password Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Submit Button	
"Click here to Login" TwinCies	Color: Lime Green (#bcff75)
Footer Text	Color: WhiteSmoke Font-Family: Arial

Input Requirements

Input	Purpose	Requirements
Email Address Input	To obtain the user's login credentials.	Email must contain '@'
Verification Code	The user must input the verification code found in their TwinCies email.	The codes must have a length of 6 characters
Password Input	The user must set a password for their account	1. Password length must be between 8-15 characters. 2. Password must contain an uppercase character. 3. Password must contain a lowercase character. 4. Password must contain a number. 5. Password must contain a special character.

Confirm Password Input	The user must re-enter their password to make sure they did not make a mistake	Must match the password in the Password field.
Submit Button	The submit button will trigger backend functions to register the user and redirect the user to the login page if there are no errors.	The button will be disabled until the requirements for the input fields have been met.
“Click here to Login” TwinCies	To redirect the user to the login page	N/A

Security layer (PHP)

The input is received from the front-end side as a JSON object which is decoded into an array.

The input is sent to validation methods that check if it is valid.

The validation methods are as follows:

- *public function GetRegistrationInfo(Request $request)*

This method receives input from the front-end and does the following:

  - Check if all input has been inputted
  - Calls the other functions and passes the array of input

- *public function CheckIfEmailIsCorrect(array $RegJson)*

This method checks if the email address that was inputted is a valid TwinCies email account.

- *public function CheckIfPasswordIsCorrect(array $RegJson)*

This method validates that both passwords match all the requirements listed on the GUI.

- *public function CheckIfPasswordsMatch(array $RegJson)*

This method checks if the new password input and the confirm password input match each other. If they do match, the confirm password is hashed and stored to be sent to the database later.

- *public function CheckIfUserExists(array $RegJson)*

This method sends the email address to the database to check if the user exists in the database. If the user does exist in the database, the verification code for that user, which will be validated later, will be returned.

- *public function CheckIfUserIsAlreadyRegistered(array $RegJson)*

This method sends the email address along with the hashed password to the database to verify if the user is not already registered in the database.

- *public function CheckIfVerificationCodesMatch(array $RegJson)*

This method checks if the verification code inputted matches the verification code received from the database.

Back-end (Laravel)

The process for the back-end is as follows:

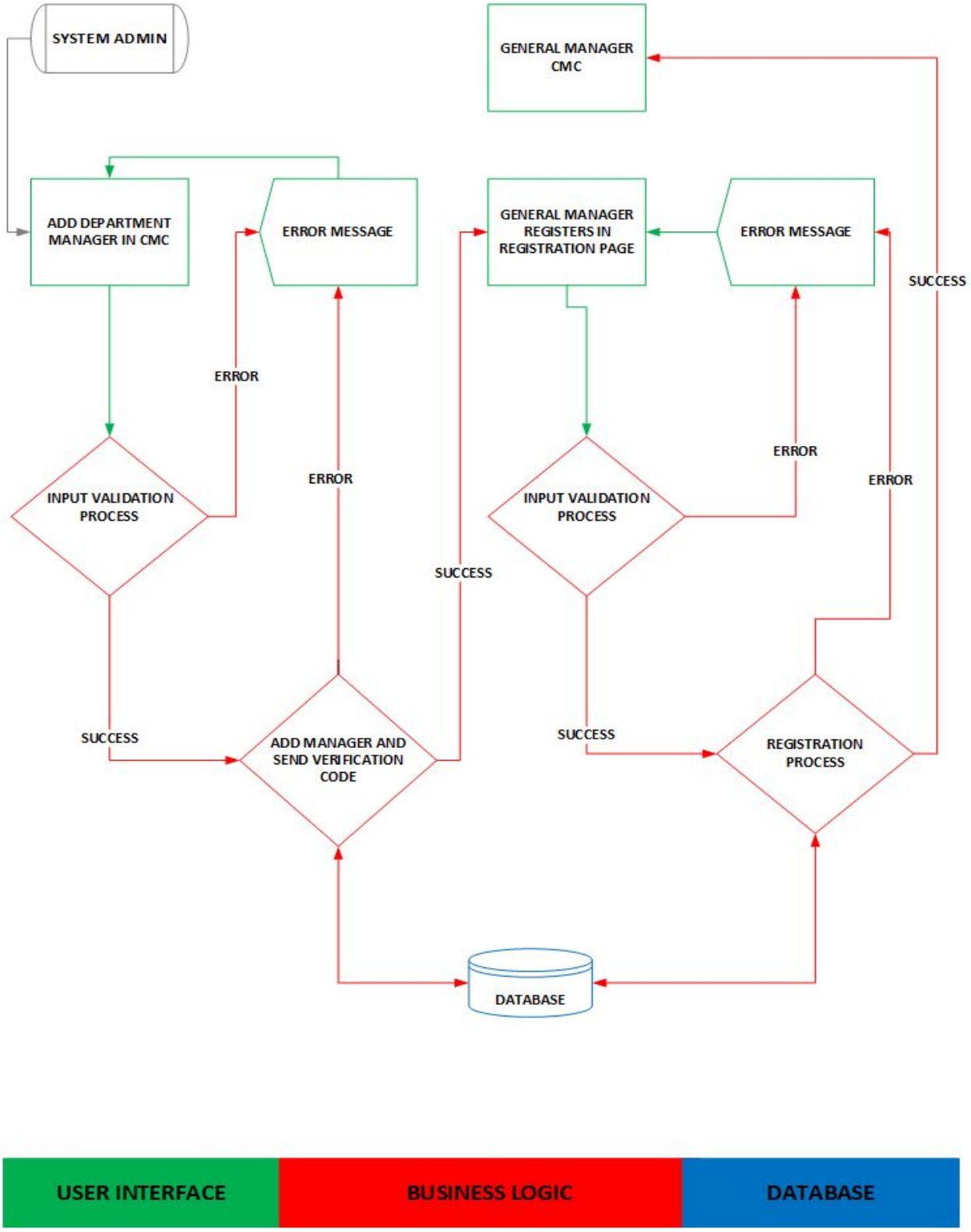
1. The system receives an email address and checks if the email address exists in the database then returns one of the two options:

    a. An error message, stating that the user does not exist, if the email address was not found in the database.
    b. The verification code that corresponds with the email address.
2. The system receives a password that has been hashed in a JSON Object - along with the email address - and does one of the two options:
    a. Return a 400 status code, stating that the user already has a password stored, if the system finds that a password has already been assigned for the user according to the database or 404 status code stating that the user does not exist in the database.
    b. The system stores the password in the database and changes the user's Active status to 1, which indicates that the user has been registered in the system.

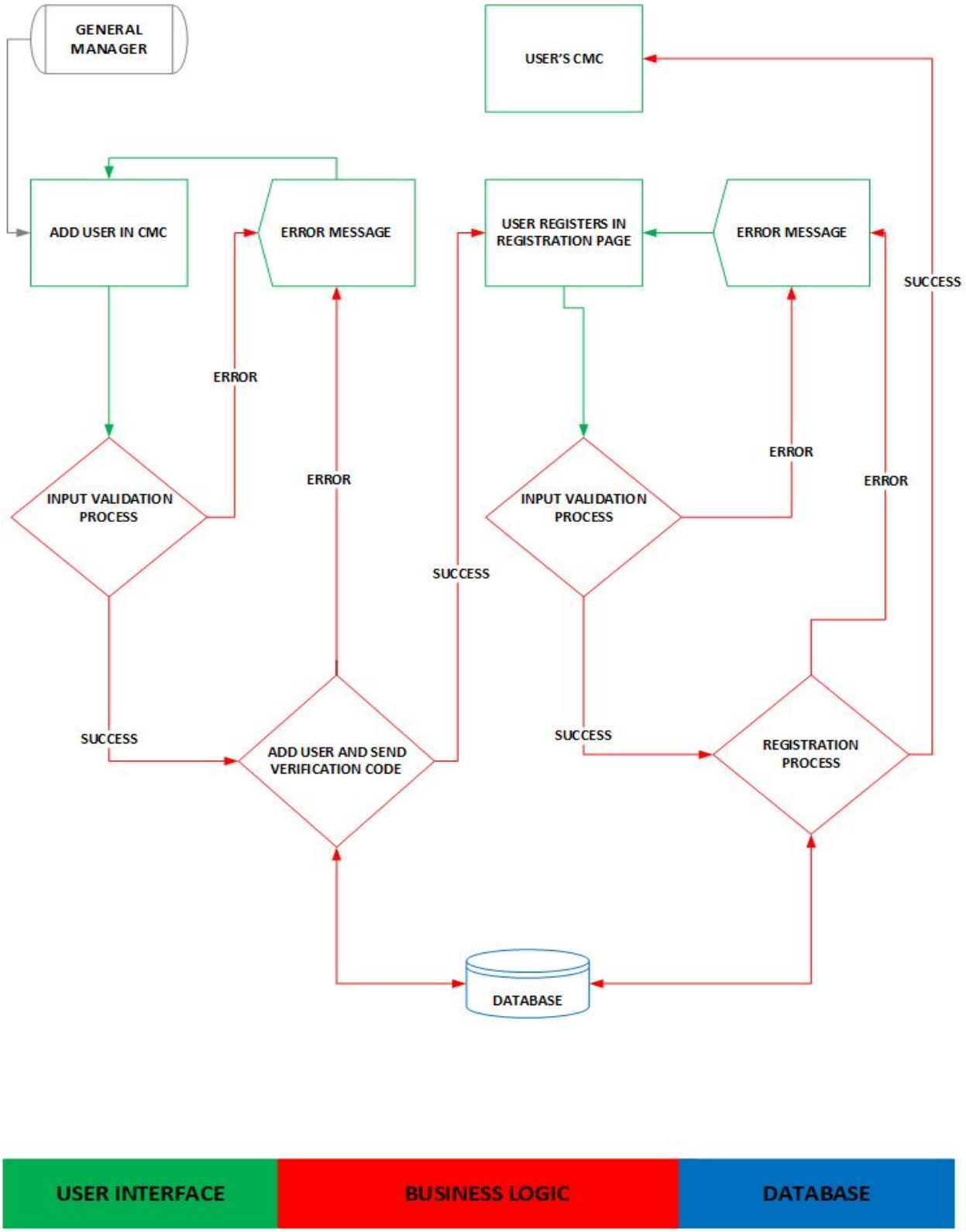
Methods in the registration process:

- *ClassMethod VerifyCodeUnderRegistration(email As %String) As %Status*
    - This method checks if a user's email address has been registered in the database
    - Verify the verification code
    - This method will return only the verification code of a user
- *ClassMethod RegisterUserPassword(email As %String) As %Status*
    - This method will register the password of any new user to the database
    - Get JSON data and check if JSON is a valid JSON Object
    - But the method can only be accessed after validating the verification code

Flow diagram: Registration process for General Manager



Flow diagram: Registration process for Standard User



5.4. Forgot Password Interface

Figure 4. Screenshot of forgot password page

Business Logic

The forgot password interface (*Figure 4)* is used to allow the user to input their email address to initiate the reset password process.

1. The user inputs their email address.
2. The user's email address will be validated and verified once the user clicks on the **Initiate Password Reset Process** button.
3. If the input is valid, the user is redirected to the reset code interface.

Front End (JavaScript)

Look and Feel

Element	CSS
"Welcome to TwinCies's IMS" Header	Color: Gray (#cccccc) Font-Family: Arial Rounded MT Bold
"Forgot Password" Header	Color: WhiteSmoke Font-Family: Arial Rounded MT Bold
Email Input Field	The input field will have a red outline when the input requirements aren't met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Submit Button	
"Click here to return to login" TwinCies	Color: Lime Green (#bcff75)
Footer Text	Color: WhiteSmoke Font-Family: Arial

Input Requirements

Input	Purpose	Requirements
Email Address Input	To obtain the user's email address.	Email must contain '@'.

Submit Button	The submit button will trigger backend functions to validate the user's email and redirect the user to the reset code page if there are no errors.	The button will be disabled until the requirements for the email input field have been met.
"Click to return to login" TwinCies	To redirect the user to the login page	N/A

Security layer (PHP)

The system receives the email address and implements the following validation methods:

- *public function GetEmailAddress(Request $request)*

This method receives the email address from the user interface and checks if the email address is valid.

- *public function CheckIfEmailIsCorrect(array $ForgotJson)*

This method checks if the email address that was inputted is a valid TwinCies email account.

- *public function CheckIfUserExists(array $ForgotJson)*

This method sends the email address to the back-end to check if the user exists in the database. The reset code for the user will be returned if the user does exist in the database.

- *public function SendResetCode(array $ForgotJson)*

This method sends an email to the user with their new reset code that they will use in the next phase of the reset password process.

Back-end (Laravel)

The forgot password process occurs as follows:

1. An email address is received and checked to see if it exists in the database:
    a. If the email address is not found in the database then 404 status code is sent stating that the email address does not exist.
    b. If the email address is found, an email is sent to the user consisting of a reset code for the reset password process.
    c. refer to the Back-end(Laravel) - Reset Password Section of this document for the reset password process.

Method in the forgot password process:

- *ClassMethod ResetCode(email As %String) As %Status*
    - This method will reset the reset code of user if they want to change their password and send the new reset code to the user's email
    - The first step is to check if the email address requesting to reset their password exists in the database, if not, the method will send a 404 status code.

5.5. Reset Code Interface

Figure 5. Screenshot of reset code page

Business Logic

The reset code interface (*Figure 5)* allows for:

1. The user to input the reset code that they received through email.
2. The user can click on the **Submit** button to validate and verify the input:

2.1. If the input is invalid, the user needs to try again.

2.2. If the input is valid, the user is redirected to the reset password interface.

Front End (JavaScript)

Look and Feel

Element	CSS
“Welcome to TwinCies’s IMS” Header	Color: Gray (#cccccc) Font-Family: Arial Rounded MT Bold
“Reset Code” Header	Color: WhiteSmoke Font-Family: Arial Rounded MT Bold
Reset code Input Field	The input field will have a red outline when the input requirements aren’t met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Submit Button	
“Click here to return to login” TwinCies	Color: Lime Green (#bcff75)
Footer Text	Color: WhiteSmoke Font-Family: Arial

Input Requirements

Input	Purpose	Requirements
Reset Code Input	The user must input the reset code found in their TwinCies email.	The reset code must have 6 characters.
Submit Button	The submit button will trigger backend functions to validate the user’s reset code and redirect the user to the reset password page if there are no errors.	The button will be disabled until the requirements for the reset code input field have been met.
“Click to return to login”	To redirect the user to the	N/A

TwinCies	login page	

Security layer (PHP)

The system will do the following once it receives the reset code from the GUI:

- *public function GetResetCode(Request $request)*

This method checks if the user inputted a reset code and sends that reset code to the other functions.

- *public function CheckIfResetCodeIsCorrect(array $userInfo)*

This method sends the user's email address to the database and receives a reset code in return.

- *public function CheckIfResetCodesMatch(array $userInfo)*

This method compares the reset code inputted by the user and the reset code from the database. If they match, the user will be redirected onto the final interface of the reset password process.

Back-end (Laravel)

The reset code process will be implemented if a user is blocked from the system or if a user forgot their password. And this back-end process is implemented as in the following manner:

1. The system receives an email address.
    a. If the system finds a reset code for the email address in the database then it will return that reset code.
    b. If the reset code is not found in the database, the system will return a 400 Bad Request status code stating that the user does not have a reset code.

Method in the reset code process:

- *ClassMethod GetResetCode(email As %String) As %Status*
    - When validating if a user is inputting a valid reset code, use this method. It will return a reset code of the user (email given)

5.6. Reset Password Interface

Figure 6. Screenshot of reset password page

Figure 7. Password requirements

Business Logic

The reset password interface (*Figure* 6) allows for:

1. The user to input their new personal password and to confirm the password.
2. The user can click on the **Set New Password** button to validate and verify the input:

2.1. If the input is invalid, the user needs to try again.
2.2. If the input is valid, the user is redirected to the login interface.

Front End (JavaScript)

Look and Feel

Element	CSS
“Welcome to TwinCies’s IMS” Header	Color: Gray (#cccccc) Font-Family: Arial Rounded MT Bold
“Reset Password” Header	Color: WhiteSmoke Font-Family: Arial Rounded MT Bold
Password Input Field	The input field will have a red outline when the input requirements aren’t met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Password requirement text (Can be seen in *Figure 7*)	When a specific requirement is not met, the text color will be: Color: bright orange (#FF8300) When a specific requirement is met, the text color will be: Color: Lime Green(#bcff75)
Confirm Password Input Field	The input field will have a red outline when the input requirements aren’t met. And the input field will have a green outline when input requirements are met. Input requirements can be found below in the input requirements table.
Submit Button	
“Click here to Login” TwinCies	Color: Lime Green(#bcff75)
Footer Text	Color: WhiteSmoke Font-Family: Arial

Input Requirements

Input	Purpose	Requirements
Password Input	The user must set a new password for their account	1. Password length must be between 8-15 characters. 2. Password must contain an uppercase character. 3. Password must contain a lowercase

		character. 4. Password must contain a number. 5. Password must contain a special character.
Confirm Password Input	The user must re-enter their password to make sure they did not make a mistake	Must match the password in the Password field.
Submit Button	The submit button will trigger backend functions to reset the user's password and redirect the user to the login page if there are no errors.	The button will be disabled until the requirements for the input fields have been met.
"Click here to Login" TwinCies	To redirect the user to the login page	N/A

Security layer (PHP)

The input received from the front-end, which is sent as a JSON object, will be decoded into an array and passed to validation methods to ensure that the user input is correct then the user's password can be reset.

The validation methods are as follows:

- *public function GetResetInfo(Request $request)*

This method receives input from the front-end and does the following:

  - Check if all input has been inputted
  - Calls the other functions and passes the array

- *public function CheckIfPasswordIsCorrect(array $ResetJson)*

This method verifies that both passwords meet all the requirements stated in the GUI.

- *public function CheckIfPasswordsMatch(array $ResetJson)*

This method checks if the new password input and the confirm password input match each other. If the passwords match, the confirm password is hashed and saved.

- *public function SendPassword(array $ResetJson)*

This method sends the email address of the user and the new hashed password to the database. If the password is stored successfully then the user will be notified of this.

Back-end (Laravel)

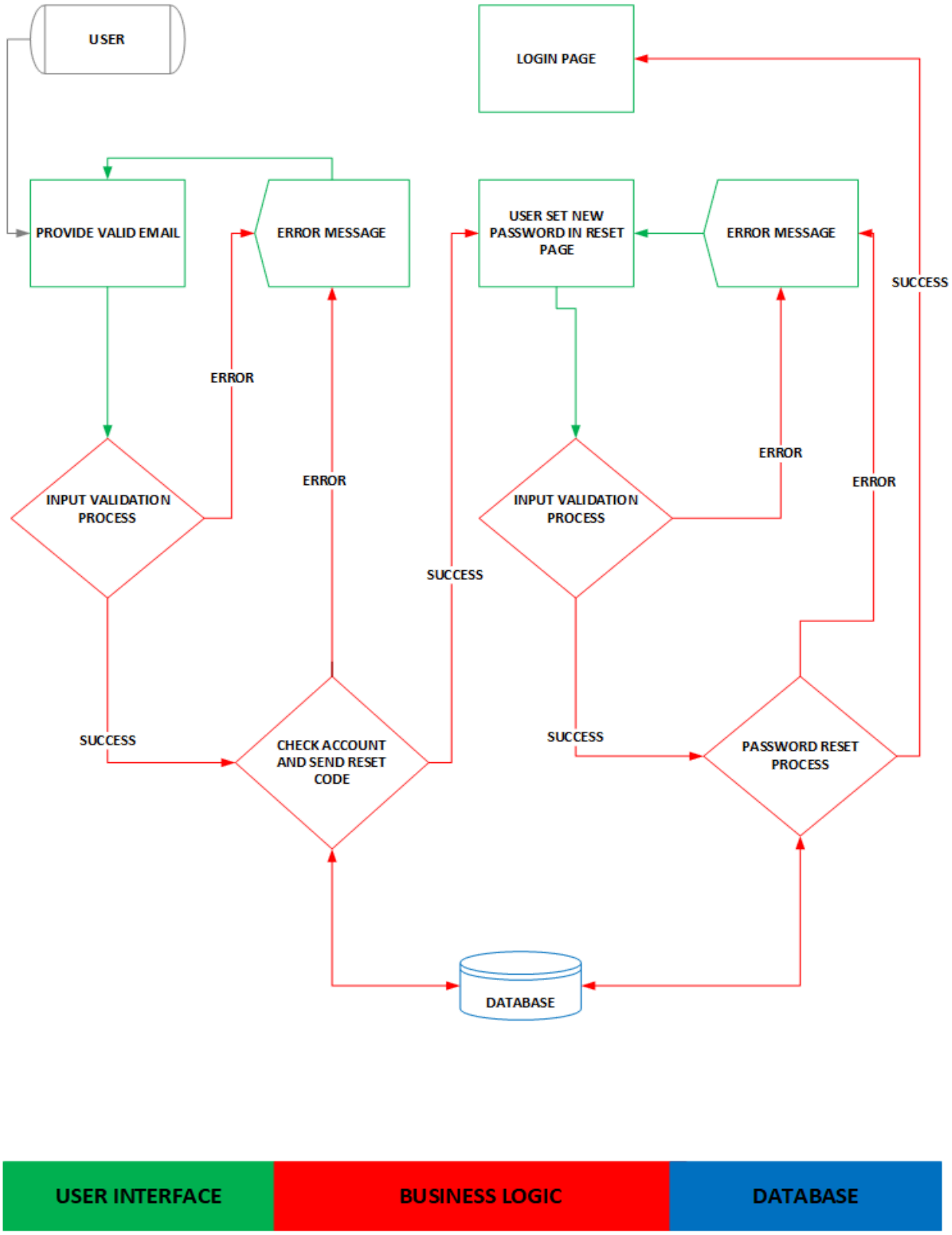
The reset password process will be implemented if a user inputs their new password. The back-end process is implemented in the following manner:

1. The system receives the new password and stores it into the database.

Method in the reset password process:

- *ClassMethod UpdateUserPassword(email As %String) As %Status*
    - This method will update the password of the user when resetting their password.
    - The first step is to check if the email address requesting to reset their password exists in the database, if not, the method will send a 404 status code.
    - And the method will send a 500 Internal Server Error if the hashed password is the same as the one in the database, this means we are updating the exact match in the database.

Flow diagram: Resetting password



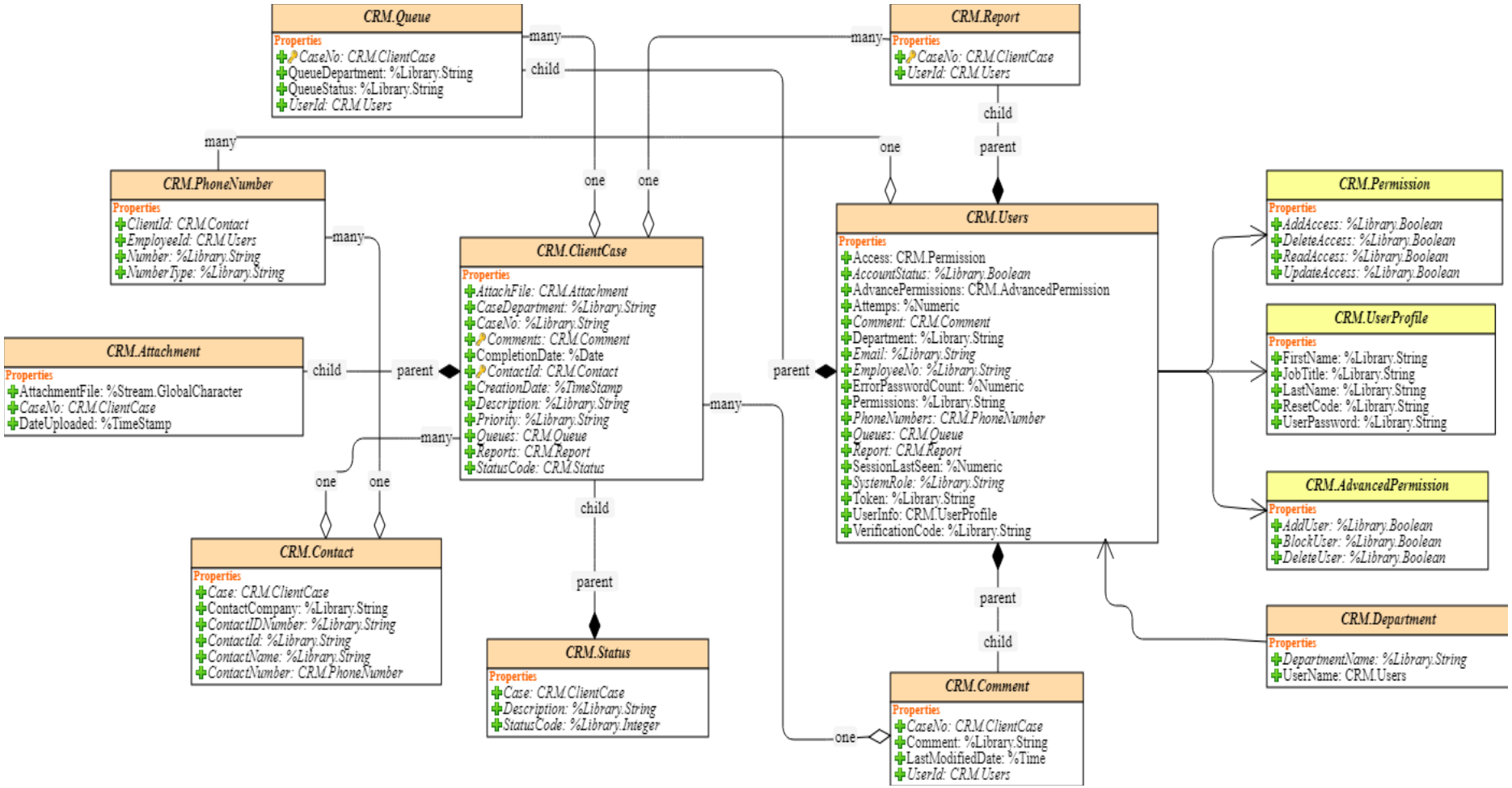
6. Database Design

6.1. Purpose of this Document

The purpose of this document is to illustrate an overview of the database and tables of the IMS.

6.2. ERD Overview

The IMS ERD was designed using Cache ClassExplorer and Cache Studio. The Cache ClassExplorer was used to create an overview picture of the ERD and Cache Studio was used in creating all the classes that create all the tables in the ERD. All the classes were created using InterSystems Laravel.


CRM.Queue
Properties
CaseNo: CRM.ClientCase
QueueDepartment: %Library.String
QueueStatus: %Library.String
UserId: CRM.Users
CRM.Report
Properties
CaseNo: CRM.ClientCase
UserId: CRM.Users
CRM.PhoneNumber
Properties
ClientId: CRM.Contact
EmployeeId: CRM.Users
Number: %Library.String
NumberType: %Library.String
CRM.ClientCase
Properties
AttachFile: CRM.Attachment
CaseDepartment: %Library.String
CaseNo: %Library.String
Comments: CRM.Comment
CompletionDate: %Date
ContactId: CRM.Contact
CreationDate: %TimeStamp
Description: %Library.String
Priority: %Library.String
Queues: CRM.Queue
Reports: CRM.Report
StatusCode: CRM.Status
CRM.Users
Properties
Access: CRM.Permission
AccountStatus: %Library.Boolean
AdvancePermissions: CRM.AdvancedPermission
Attemps: %Numeric
Comment: CRM.Comment
Department: %Library.String
Email: %Library.String
EmployeeNo: %Library.String
ErrorPasswordCount: %Numeric
Permissions: %Library.String
PhoneNumbers: CRM.PhoneNumber
Queues: CRM.Queue
Report: CRM.Report
SessionLastSeen: %Numeric
SystemRole: %Library.String
Token: %Library.String
UserInfo: CRM.UserProfile
VerificationCode: %Library.String
CRM.Permission
Properties
AddAccess: %Library.Boolean
DeleteAccess: %Library.Boolean
ReadAccess: %Library.Boolean
UpdateAccess: %Library.Boolean
CRM.UserProfile
Properties
FirstName: %Library.String
JobTitle: %Library.String
LastName: %Library.String
ResetCode: %Library.String
UserPassword: %Library.String
CRM.AdvancedPermission
Properties
AddUser: %Library.Boolean
BlockUser: %Library.Boolean
DeleteUser: %Library.Boolean
CRM.Attachment
Properties
AttachmentFile: %Stream.GlobalCharacter
CaseNo: CRM.ClientCase
DateUploaded: %TimeStamp
CRM.Contact
Properties
Case: CRM.ClientCase
ContactCompany: %Library.String
ContactIDNumber: %Library.String
ContactId: %Library.String
ContactName: %Library.String
ContactNumber: CRM.PhoneNumber
CRM.Status
Properties
Case: CRM.ClientCase
Description: %Library.String
StatusCode: %Library.Integer
CRM.Comment
Properties
CaseNo: CRM.ClientCase
Comment: %Library.String
LastModifiedDate: %Time
UserId: CRM.Users
CRM.Department
Properties
DepartmentName: %Library.String
UserName: CRM.Users
many
child
one
parent

6.3. Tables and Class Definitions

User Table

Stores basic information for all users of the system

Column Name	Database Type	Comment
UserId (Primary key)	varchar (128)	String to uniquely identify each user
UserEmail	varchar (64)	String used to store the user's email address
VerificationCode	varchar (6)	Code used to verify if the email address belongs to the user
AccountStatus	bit (1)	The status of the user's account – '0' being 'Inactive' and '1' being 'Active'
SystemRole	enum ("System Administrator", "General Manager", "Standard User")	The role that the user will have in the system
Department	varchar (64)	The department that the user works in
Token	varchar (125)	Token used for session handling
ErrorPasswordCount	varchar (64)	used to block user if exceeds 4
UserProfile %Serial object – stores personal information about the users		
FirstName	varchar (64)	String to store the name of the user
LastName	varchar (64)	String to store the surname of the user
UserPassword	varchar (128)	String to store the hashed password
ResetCode	varchar (6)	Code that user receives so that they reset their password
JobTitle	varchar (64)	Description of the user's job title
Permission %Serial object – stores data about the permission types		
ReadAccess	bit (1)	Shows whether the user has permission or not. '0' means the user doesn't and '1' means the user does
AddAccess	bit (1)	Shows whether the user has permission or not.

		‘0’ means the user doesn’t and ‘1’ means the user does
UpdateAccess	bit (1)	Shows whether the user has permission or not. ‘0’ means the user doesn’t and ‘1’ means the user does
DeleteAccess	bit (1)	Shows whether the user has permission or not. ‘0’ means the user doesn’t and ‘1’ means the user does

User Class Definition

Class Crm.User Extends %Persistent					
Property	**Datatype**	**Required**	**Unique**	**MaxLen**	**InitialExpression**
UserId	%String	Yes	Yes		
UserEmail	%String	Yes	No		
VerificationCode	%String	Yes	No	6	
AccountStatus	%Boolean	Yes	No		1
SystemRole	%String	Yes	No		
Department	%String	Yes	No		
UserInfo	Crm.UserProfile				
Access	Crm.Permission				

Relationship	**Class**	**Cardinality**	**Inverse**
Queue	Crm.Queue	Children	UserId
Report	Crm.Report	Children	UserId
PhoneNumber	Crm.PhoneNumber	Children	EmployeeId

UserProfile Class Definition

Class Crm.UserProfile Extends %Serial						
Property	**Datatype**	**Required**	**Unique**	**MinLen**	**MaxLen**	**InitialExpression**
FirstName	%String	No	No			
LastName	%String	No	No			
UserPassword	%String	Yes	No			
ResetCode	%String	No	No			
JobTitle	%String	No	No			

Permission Class Definition

Class Crm.Permission Extends %Serial					
Property	**Datatype**	**Required**	**Unique**	**MinLen**	**InitialExpression**

ReadAccess	%Boolean	Yes	No		1
AddAccess	%Boolean	Yes	No		0
UpdateAccess	%Boolean	Yes	No		0
DeleteAccess	%Boolean	Yes	No		0

PhoneNumber Table

Stores the various types of phone numbers

Column Name	Datatype	Comment
NumberType	enum(“Business”,”Mobile”,”Telephone”,’ Fax”)	The type of phone number that will be stored
Number	tinytext (12)	String to store the number
ClientId (Foreign Key)	varchar (128)	String to store the id of the User or the Contact

PhoneNumber Class Definition

Class Crm.PhoneNumber Extends %Persistent					
Property	**Datatype**	**Required**	**MinLen**	**MaxLen**	**InitialExpression**
NumberType	%String	Yes			“Mobile”
Number	%String	Yes	10	12	
ClientId	%String	Yes			

Relationship	Class	Cardinality	Inverse
ClientId	Crm.Contact	Parent	ContactPhone
EmployeeId	Crm.User	Parent	PhoneNumber

Contact Table

Stores information about the clients

Column Name	Datatype	Comment
ContactId (Primary key)	varchar (128)	String used to uniquely identify each client
ContactName	varchar (128)	String that will store the full name of a client
ContactIdNumber	varchar (13)	String that stores the contact’s identity number

ContactCompany	varchar (128)	String that will store the company name where the client works
ContactEmail	varchar (64)	String to store the client's email address

Contact Class Definition

Class Crm.Contact Extends %Persistent					
Property	**Datatype**	**Required**	**MinLen**	**MaxLen**	**InitialExpression**
ContactId	%String	Yes			
ContactName	%String	Yes			
ContactIdNumber	%String	Yes	13	13	
ContactCompany	%String	Yes			
ContactEmail	%String	No			

Relationship	**Class**	**Cardinality**	**Inverse**
ContactPhone	Crm.PhoneNumber	Children	ClientId
Case	Crm.ClientCase	Children	ContactId

ClientCase Table

Stores information about the inquires

Column Name	Datatype	Comment
CaseNo (Primary key)	varchar (128)	String used to uniquely identify a case
Description	longtext	The description of what the case is about
CaseReport	longtext	A feedback from the users about what has been with the case
Priority	enum ("Normal", "Medium", "High")	The level of importance that the case should be addressed in
CaseDepartment	varchar (64)	The department in which the case should be assigned to
CreationDate	timestamp	The date when the case was created
CompletionDate	date	The date when the case is finally completed
StatusCode (Foreign key)	int (2)	The status of the case

ContactId (Foreign key)	varchar (128)	String that shows which client the case belongs to

ClientCase Class Definition

Class Crm.ClientCase Extends %Persistent					
Property	**Datatype**	**Required**	**Unique**	**MaxLen**	**InitialExpression**
CaseNo	%String	Yes	Yes		
Description	%String	Yes	No		
CaseReport	%String	Yes	No	6	
Priority	%String	Yes	No		1
CaseDepartment	%String	Yes	No		
CreationDate	%TimeStamp	Yes	No		
CompletionDate	%Date	No	No		

Relationship	Class	Cardinality	Inverse
Queue	Crm.Queue	Children	CaseNo
Report	Crm.Report	Children	CaseNo
ContactId	Crm.Contact	Parent	Case
StatusCode	Crm.Status	Parent	Case

Status Table

Stores data for the status codes

Column Name	Datatype	Comment
StatusCode (Primary key)	int (2)	The status code number (ranges from 1 to 22)
StatusDescription	longtext	The description of a status code

Status Class Definition

Class Crm.Status Extends %Persistent						
Property	**Datatype**	**Required**	**Unique**	**MinVal**	**MaxVal**	**InitialExpression**
StatusCode	%Integer	Yes	Yes	1	22	
Description	%String	Yes	No			

Relationship	Class	Cardinality	Inverse
Case	Crm.ClientCase	Children	StatusCode

Queue Table

Stores data for the queues

Column Name	Datatype	Comment
QueueStatus	enum (“Pending”, “Completed”)	The status of the case in the queue
QueueDepartment	varchar (64)	The department in which the queue is placed
UserId (Foreign key)	varchar (128)	String that shows which user is working which case
CaseNo (Foreign key)	varchar (128)	String that shows which cases are currently being processed

Queue Class Definition

Class Crm.Queue Extends %Persistent					
Property	**Datatype**	**Required**	**Unique**	**MinLen**	**InitialExpression**
QueueStatus	%String	No	No		
QueueDepartment	%String	No	No		
Relationship	**Class**	**Cardinality**			**Inverse**
UserId	Crm.User	Parent			Queue
CaseNo	Crm.ClientCase	Parent			Queue

Report Table

Stores the comments and reports for each case

Column Name	Datatype	Comment
Comment	longtext	String that contains the comments that the user has for the case
LastModifiedDate	timestamp	The date for when the comment was added
CaseNo (Foreign key)	varchar (128)	String that shows which cases are currently being processed
UserId (Foreign key)	varchar (128)	String that shows which user is working which case

Report Class Definition

Class Crm.Report Extends %Persistent					
Property	**Datatype**	**Required**	**Unique**	**MinLen**	**InitialExpression**
Comment	%String	No	No		
LastModifiedDate	%TimeStamp	No	No		

Relationship	**Class**	**Cardinality**	**Inverse**
UserId	Crm.User	Parent	Report
CaseNo	Crm.ClientCase	Parent	Report

7.Test Plan

7.1. Purpose of this Document

The purpose of this document is to describe the testing approaches that will be used when testing the IMS system.

7.2. References

1. Functional Specifications for TwinCies IMS Access: Registration, Verification, Password Reset and Login (v0.3)

7.3. TwinCies IMS User Access - GUI Interface

Introduction

The purpose of this document is to describe the testing approaches that will be used when testing the GUI Interface for the User Access of the IMS system. This test plan will be used to check the functionality of the TwinCies CRM Case Management System, to inspect whether it works according to its specifications.

Objectives

The objectives for the tests are as follows:

- Listing the test requirements
- Describing the testing strategies that will be conducted
- Listing the deliverable elements of the tests
- Listing the positive and negative response the user may encounter.
- Testing the visible element design of interface.

Scope

The tests will be focused on the User Access GUI Interface of the IMS.

In Scope	Out of Scope
Graphical User Interface	CRM Access Security Layer
Registration form	
Login form	
Forgot Password form	
Reset Code form	

Reset Password form	

Test Strategy

Functional testing

The purpose of these tests is to find any unexpected behaviors within the IMS. The test aim to verify:

- Error messages
- The TwinCiess redirect the user to the correct pages.
- Accuracy of input validation
- Any security failures.
- All for elements produce the correct functionalities.

GUI testing

These tests must cover the look and feel, spelling mistakes, error messages and other elements that are specified in the functional specification document.

- Error messages will be tested with the human eye.

Testing Tools

Computer

1. VueJS
2. PC
3. LAN connection
4. Monitor
5. Keyboard
6. Mouse

Data

The data to be used for testing will be pre-loaded data, such as the email address and the verification code.

Getting Started

Testing environment

These browsers were selected for testing because they were found to be the most popular browsers.

1. Google Chrome
2. Internet Explorer

Assumptions and rules

- The tester will assume the role of either the system administrator, a general manager or standard user.
- All added users must have a TwinCies email address.
- Before the system administrator can add a general manager, the department they are being allocated to should already be created.
- The general manager of a department can only add standard users to their department.

Testing

Paths

Functions tested	URI
Registration	http://localhost:8080/#/Registration
Login	http://localhost:8080/#/
Forgot Password	http://localhost:8080/#/ForgotOne
Reset Code	http://localhost:8080/#/Reset
Reset Password	http://localhost:8080/#/ForgotTwo

Impacted tables

The following tables will be affected when using the TwinCies IMS Access system:

Process	Tables	SQL Command
Registration	User UserProfile	SELECT
Login		
Forgot password		
Reset code		
Reset password		

Test scenarios

The Input requirements are listed in Functional Specifications of TwinCies IMS Access.

Registration Form

Test Case	
Test Scenario Name	Registration
Description	These tests cover the functionality and visible design elements of the Registration Page

Test Scenario Page	Registration Page

Input Validation[1]				
Scenario No.	**Test Scenario**	**Input Data**	**Positive Results**	**Negative Results**
1	Verify that the registration form contains the correct input data that meets the requirements.	*Email *Verification Code *Password *Confirm Password	The user will be able to click the submit button.	The submit button will be disabled.
2	Verify if email address is a valid email address	*Email address	The input field will have a green outline.	The input field will have a red outline
3	Check if the password meets the requirements	*Password	The input field will have a green outline.	The input field will have a red outline
4	Check if the confirm password matches the password.	*Password *Confirm Password	The input field will have a green outline.	The input field will have a red outline
5	Check if verification code inputted meets the requirements.	*Verification code	The input field will have a green outline.	The input field will have a red outline
6	All page text		All the text on this page is readable.	All the text on this page is not readable.
7	Background image		The background image is clear.	The background image is not clear.

[1] Requirements as listed in Functional Specifications of TwinCies IMS Access.

8	“Return to Login” TwinCies		The user is redirected to the Login Page	The user is not redirected to the Login Page
9	The user clicks the submit button		The user is redirected to the Login Page	An error message will be displayed

Login Form

Test Case	
Test Scenario Name	Login
Description	These tests cover the functionality and visible design elements of the Login Page
Test Scenario Page	Login Page

Input Validation[2]				
Scenario No.	**Test Scenario**	**Input Data**	**Positive Results**	**Negative Results**
1	Verify that the login form contains the correct input data that meets the requirements.	*Email *Password	The user will be able to click the submit button.	The submit button will be disabled.
2	Verify if email address is a valid email address	*Email address	The input field will have a green outline.	The input field will have a red outline
3	Check if password meets the requirements	*Password	The input field will have a green outline.	The input field will have a red outline
4	All page text		All the text on this page is readable.	All the text on this page is not readable.

[2] Requirements as listed in Functional Specifications of TwinCies IMS Access.

5	Background image		The background image is clear.	The background image is not clear.
6	“Register Account” TwinCies		The user is redirected to the Registration Page	The user is not redirected to the Registration Page
7	“Forgot Password” TwinCies		The user is redirected to the Forgot Password page	The user is not redirected to the Forgot Password page
8	The user clicks the submit button		The user is redirected to the appropriate user console.	An error message will be displayed

Forgot Password Form

Test Case	
Test Scenario Name	Forgot Password
Description	These tests cover the functionality and visible design elements of the Forgot Password Page
Test Scenario Page	Forgot Password Page

Input Validation[3]				
Scenario No.	**Test Scenario**	**Input Data**	**Positive Results**	**Negative Results**
1	Verify that the forgot password form contains the correct input data that meets the requirements	*Email	The user will be able to click the submit button.	The submit button will be disabled.

[3] Requirements as listed in Functional Specifications of TwinCies IMS Access.

2	Verify if email address is a valid email address	*Email address	The input field will have a green outline.	The input field will have a red outline.
4	All page text		The text is readable.	The text is not readable.
5	Background image		The background image is clear.	The background image is not clear.
6	“Return to Login” TwinCies		The user is redirected to the Login Page	The user is not redirected to the Login Page
7	The user clicks the submit button		The user is redirected to the Reset Code Page	An error message will be displayed

Reset Code Form

Test Case	
Test Scenario Name	Reset Code
Description	These tests cover the functionality and visible design elements of the Reset Code Page
Test Scenario Page	Reset Code Page

Input Validation[4]				
Scenario No.	**Test Scenario**	**Input Data**	**Positive Results**	**Negative Results**
1	Verify that the Reset code form contains the correct input data that meets the	*Reset Code	The user will be able to click the submit button.	The submit button will be disabled.

[4] Requirements as listed in Functional Specifications of TwinCies IMS Access.

	requirements .			
2	Verify if Reset code is valid	*Reset Code	The input field will have a green outline.	The input field will have a red outline.
4	All page text		All the text on this page is readable.	All the text on this page is not readable.
5	Background image		The background image is clear.	The background image is not clear
6	“Return to Login” TwinCies		The user is redirected to the Login Page	The user is not redirected to the Login Page
7	The user clicks the submit button		The user is redirected to the Reset Password Page	An error message will be displayed

Reset Password Form

Test Case	
Test Scenario Name	Reset Password
Description	These tests cover the functionality and visible design elements of the Reset Password Page
Test Scenario Page	Reset Password Page

Input Validation[5]				
Scenario No.	**Test Scenario**	**Input Data**	**Positive Results**	**Negative Results**
1	Verify that the Rest Password form contains the correct input data	*Password *Confirm Password	The user will be able to click the submit button.	The submit button will be disabled.

[5] Requirements as listed in Functional Specifications of TwinCies IMS Access.

	that meets the requirements.			
2	Check if password meets the requirements	*Password	The input field will have a green outline.	The input field will have a red outline.
3	Check if the confirm password matches the password.	*Password *Confirm Password	The input field will have a green outline.	The input field will have a red outline.
4	All page text		The text is readable.	The text is not readable.
5	Background image		The background image is clear.	The background image is not clear.
6	“Return to Login” TwinCies		The user is redirected to the Login Page	The user is not redirected to the Login Page
7	The user clicks the submit button		The user is redirected to the Login Page	An error message will be displayed

7.4. TwinCies IMS User Access - Security Layer (PHP)

Introduction

This test plan will be used to check the functionality of the TwinCies CRM Case Management System in the security layer, to inspect whether it works according to its specifications.

Objectives

The objectives for the tests are as follows:

- Describing the testing strategies that will be conducted
- Listing the test requirements
- Listing the deliverable elements of the tests

Scope

The tests will be focused on the security layer of the IMS.

In Scope	Out of Scope
Registration process	GUI testing Database testing
Login process	
Forgot Password process	
Reset Password process	

Test Strategy

Functional testing

The purpose of the tests included in this document is to verify and validate user input. The system will not continue and error messages will be displayed if one of these tests fail.

The aim of these tests is to:

- Ensure that valid data is used to access the system
- Display a relevant error message for each error that occurs
- Test the functionality of the system and how it handles errors and/or invalid input

Testing Tools

Computer

1. Laravel
2. PC
3. LAN connection
4. Monitor
5. Keyboard
6. Mouse

Data

The data to be used for testing will be pre-loaded data, such as the email address, the verification code and the reset code. The password will be made up by the tester.

Getting Started

Testing environment

The testing can be done on the Restlet Client (REST API Testing), which can be added as an extension on Google Chrome, or by using Postman (API Development Environment).

Assumptions and rules

1. The tester must use a TwinCies email address and a verification code that is available on the database. For error testing, an invalid email address will work.
2. The tester must remember the password that they have stored or they will be blocked from the system - involuntarily - and made to reset their password.
3. The tester has access to the email address they will be using to view the emails that they receive from the system.

Testing

Impacted tables

The following tables will be affected when using the TwinCies IMS Access system:

Process	Tables	SL Command
Registration	User UserProfile	SELECT
Login		
Forgot password		
Reset code		
Reset password		

Test scenarios

Registration process

Test Case	
Test Scenario Name	Registration
Description	This test covers the functionality of a user registering their account
Test Scenario Page	Registration Page

Input Validation[6]			
Test Scenario	**Input Data**	**Expected Results**	**Comments**

[6] Requirements as listed in Functional Specifications of Q TwinCies IMS Access - Registration: Input Requirements

Verify that the registration form contains input data	*Email address *Verification Code *Password *Confirm Password	Display error message stating “Please fill out all the fields”	*Negative* - returns the error message *Positive* - the email address will be validated
Verify if email address is a valid email address	*Email address	Display error message stating “Not a valid email address”	*Negative* - returns the error message *Positive* - the next validation step occurs
Verify if email address is a valid TwinCies email address	*Email address	Display error message stating “Email address is not a TwinCies email account”	*Negative* - returns the error message *Positive* - the next validation step occurs
Check if passwords meet the requirements	*Password *Confirm Password	Display error message stating “Password must at least have one of the requirements”	*Negative* - returns the error message *Positive* - the next validation step occurs
Check if the passwords match	*Password *Confirm Password	Display error message stating “Passwords do not match”	*Negative* - returns the error message *Positive* - one of the passwords are hashed
Verify that the email address is registered on the database	*Email address	Display error message	*Negative* - returns the error message *Positive* - the next validation step occurs
Check if user has already registered on the database	*Email address	Display error message	*Negative* - returns the error message *Positive* - the next validation step occurs
Check if verification code inputted matches verification code in database	*Verification Code	Display error message stating “Verification codes do not match”	*Negative* - returns the error message *Positive* - returns an “OK” status

Login process

Test Case

Test Scenario Name	Login
Description	This test covers the functionality of a user logging into their case management console
Test Scenario Page	Login Page

Input Validation[7]			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify that the login form contains input data	*Email address *Password	Display error message stating "Please fill out all the fields"	*Negative* - returns the error message *Positive* - the email address will be validated
Verify if email address is a valid email address	*Email address	Display error message stating "Please enter a valid email address"	*Negative* - returns the error message *Positive* - the next validation step occurs
Verify if email address is a valid TwinCies email address	*Email address	Display error message stating "Email address is not a TwinCies email account"	*Negative* - returns the error message *Positive* - the next validation step occurs
Check if the email address inputted exists in the database	*Email address	Display error message	*Negative* - returns the error message *Positive* - the database returns a JSON object
Check if the password inputted matches the password in the database	*Password	Display error message	*Negative* - returns the error message *Positive* - the user has successfully been logged in and a JSON object will be sent to the front-end

Forgot password process

Test Case

[7] Requirements as listed in Functional Specifications of TwinCies IMS Access - Login: Input Requirements

Test Scenario Name	Forgot Password
Description	This test covers the functionality of a user initiating the reset password process
Test Scenario Page	Forgot Password Page

Input Validation[8]			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify that an email address has been inputted	*Email address	Display error message stating "Required field"	*Negative* - returns the error message *Positive* - the email address will be validated
Verify if email address is a valid email address	*Email address	Display error message stating "Not a valid email address"	*Negative* - returns the error message *Positive* - the email address will be validated
Verify if email address is a valid TwinCies email address	*Email address	Display error message stating "Email address is not a TwinCies email account"	*Negative* - returns the error message *Positive* - the email address will be validated
Check if the email inputted is found in the database	*Email address	Display error message	*Negative* - returns the error message *Positive* - an email is sent to the user

Reset password process

Test Case	
Test Scenario Name	Reset Password (Pt. I)
Description	This test covers the functionality of a user completing the reset password process
Test Scenario Page	Reset Code Page

[8] Requirements as listed in Functional Specifications of TwinCies IMS Access - Forgot Password: Input Requirements

Input Validation			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify that the reset code is inputted	*Reset code	Display error message stating “Please fill out the required field”	*Negative* - returns the error message *Positive* - the next validation step occurs
Verify if the reset code is found in the database	*Email address *Reset code	Display error message	*Negative* - returns the error message *Positive* - a reset code is returned from the database
Check if the reset code inputted matches the reset code in the database	*Reset code	Display error message stating “Incorrect reset code”	*Negative* - returns the error message *Positive* - returns an “OK” status

Test Case	
Test Scenario Name	Reset Password (Pt. II)
Description	This test covers the functionality of a user reset their password
Test Scenario Page	Reset Password Page

Input Validation[9]			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify if there is any input data	*New Password *Confirm Password	Display error message stating “Required field”	*Negative* - returns the error message *Positive* - the next validation step occurs
Verify if password meets	*New Password *Confirm Password	Display error message stating	*Negative* - returns the error message

[9] Requirements as listed in Functional Specifications of TwinCies IMS Access - Reset Password: Input Requirements

the requirements		“Password must at least have one of the requirements”	*Positive* - the next validation step occurs
Check if the passwords match	*New Password *Confirm Password	Display error message stating “Passwords do not match”	*Negative* - returns the error message *Positive* - the password is stored in the database

7.5. TwinCies IMS User Access - Database

Introduction

The purpose of this document is to describe the testing approaches that will be used when testing the IMS database. This test plan will be used to check the functionality of some classes and how they interact with the database, to inspect whether everything works per functional specifications.

Objectives

The objectives for the tests are as follows:

- Describing the testing strategies that will be conducted
- Listing the test requirements
- Listing the deliverable elements of the tests

Scope

In Scope	Out of Scope
Registration (back-end process)	Input Validation Sending emails
Login (back-end process)	
Forgot Password (back-end process)	
Reset Password (back-end process)	

Test Strategy

Functional testing: UnitTest

The **%UnitTest** package support test automation. When a unit test finishes executing it reports whether the test passed or failed. **%UnitTest** framework provide GUIs that summarize the test results. **%UnitTest** generates a CSP that displays the test results. It displays information concerning tests that passed in green and displays information concerning failed tests in red.

Here is a CSP test report generated by a **%UnitTest** unit test. Notice that it reports that the “Test Suite failed”. The user can drill down by clicking the hyperTwinCiess on the page and see pages that provide more detail about the test failure.

Here are the basic steps for executing a suite of unit tests:

1. Export the test class(es) to an XML file.
2. Open Terminal and point it at the namespace containing the class(es) that you are going to test. Assign to **^UnitTest** a string containing the path to the parent of the directory that contains the exported test class XML files.

```
USER> Set ^UnitTestRoot="c:\unittests"
```

3. In Terminal, run **%UnitTest.Manager.RunTest** passing it the name of the (child) directory that contains the test class XML files.

```
USER> Do ##class(%UnitTest.Manager).RunTest("mytests")
```

4. Review the test reports. The output in Terminal includes a URL for a CSP page that display the results in easy-to-read tables.

Before viewing the reports in a namespace, you must grant the Management Portal access to the UnitTest framework for a particular namespace. Do this by entering the following command at a terminal prompt in the **%SYS** namespace:

```
%SYS>Set ^SYS("Security","CSP","AllowPrefix","/csp/user/","%UnitTest.")=1
```

You can now navigate to the UnitTest Portal, using **System Explorer > Tools > UnitTest Portal**. If necessary, switch to the USER namespace.

The portal organizes the test results into a series of reports. Each test report organizes test results into a series of hyperTwinCiesed pages. Follow the TwinCiess to find increasingly specific information.

The first page provides a summary for all test suites. In this case, not all test suites passed.

Click **1** in the *ID* column.

The second page displays results by each test suite. In this example, **mytests**.

Click **mytests**.

The third page displays results by each test cast. In this example, the single test case *CASE.Tests* passed.

Click *CASE.Tests*

The fourth page shows the results broken out by test method. Here the single test method *TestAdd* passed.

Click **TestAdd**.

The final page displays results for each **AssertX** macro used in a test method. In this example, **AssertEquals** and **AssertNotEquals** both passed.

Testing Tools

Computer

1. Ensemble Terminal
2. CSP
3. Web browser
4. PC
5. Monitor
6. Keyboard
7. Mouse

Data

The data to be used for testing will be pre-loaded data, such as the System Administrator's email and verification code. The system administrator can register as normal and create other users. If the system administrator does not create any users, it is advisable to use automated UnitTest as they populate sample data for testing purposes.

Getting Started

Testing environment

Automated Testing can be done using Ensemble's Terminal. Google Chrome can be used to view CSP GUI of the results. Postman (API Development Environment) can also be used for manual testing, which can be added as an extension on Google Chrome.

Assumptions and rules

1. All data will be received as JSON if not using automated tests.
2. Only the email address of a user of the system will be received as a parameter, any data will be received through the above mentioned JSON Object.
3. For the system to work, a System Administrator will be created using the PostUser method found in the CRM.ServiceREST class.
4. Tests will work based on data available on the database
5. If any functionality fails, a status code will be sent back

Testing

Impacted tables

The following tables will be affected when using the TwinCies IMS User Access system:

Tables	Methods tested	Http Methods	SQL & Laravel Commands
User & UserInfo	Create User: PostUser	POST	%New(), %Save()
	Registration: VerifyCodeUnderRegistration, RegisterUserPassword	GET, PUT	SELECT, %OpenID(), %Save()
	Login: UserLogin, PasswordWordError	GET, PUT	SELECT
	Forgot Password: ResetCode	PUT	SELECT, %OpenID(), %Save()
	Reset Password: GetResetCode, UpdateUserPassword	GET, POST	SELECT, %OpenID(), %Save()

Test scenarios

Registration (back-end process)

Test Case	
Test Scenario Name	Registration
Description	This test covers the functionality of a user registering their account
Test Scenario Page	Registration Page

ClassMethod VerifyCodeUnderRegistration(email As %String) As %Status			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify if email address has been registered in the database	*Email address	Next validation step	*Negative* - 500 Internal Server Error or 404 Not found *Positive* - the email address will be validated
Verify the verification code	*Email address	Returns only the verification code of a user in a JSON Object	*Negative* - returns the error message *Positive* - the next validation step occurs
ClassMethod RegisterUserPassword(email As %String) As %Status			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Validate JSON Object	*Password	Next validation step	*Negative* - 400 Invalid JSON string *Positive* - the next validation step occurs
Register the password of any new user to the database	*Email address *Password	200 OK status code	*Negative* - 400 Bad Request, 400 Invalid JSON string or 404 Invalid email or 500 Internal Server Error *Positive* - the email address will be validated

Login (back-end process)

Test Case	
Test Scenario Name	Login
Description	This test covers the functionality of a user logging into their case management console
Test Scenario Page	Login Page

ClassMethod UserLogin(email As %String) As %Status

Test Scenario	Input Data	Expected Results	Comments
Verify email address	*Email address	Next validation step	*Negative* - 500 Internal Server Error or 404 Invalid email *Positive* - the email address will be validated
Verify if user registered a password in the database	*Email address	Next validation step	*Negative* - 400 Bad request *Positive* - the next validation step occurs
Verify if user is logged in	*Email address	Return a UserId, Token, Password and Job Title	*Negative* - 400 Bad Request, 404 Invalid email or 500 Internal Server Error *Positive* - the next validation step occurs
ClassMethod PasswordWordError(email As %String) As %Status			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify if user is blocked	*Email address	Laravel's MACRO Status	*Negative* - 401 Unauthorized or 403 Forbidden *Positive* - the email address will be validated

Forgot password (back-end process)

Test Case	
Test Scenario Name	Forgot Password
Description	This test covers the functionality of a user initiating the reset password process
Test Scenario Page	Forgot Password Page

ClassMethod VerifyCodeUnderRegistration(email As %String) As %Status			
Test Scenario	**Input Data**	**Expected Results**	**Comments**

Verify if system created a new reset code	*Email address	Return a new reset code in a JSON Object	*Negative* - 400 Bad Request *Positive* - return a reset code

Reset password (back-end process)

Test Case	
Test Scenario Name	Reset Password
Description	This test covers the functionality of a user completing the reset password process
Test Scenario Page	Reset Code Page

ClassMethod UpdateUserPassword(email As %String) As %Status			
Test Scenario	**Input Data**	**Expected Results**	**Comments**
Verify email address	*Email address	Next validation step	*Negative* - 500 Internal Server Error or 404 Invalid email *Positive* - the email address will be validated
Verify if password has been updated	*Email address *Password	Laravel's MACRO Status	*Negative* - 500 Internal Server Error or 404 Invalid email *Positive* - $$$OK status and an updated hashed password in the database