Logo, company name

Description automatically generated

Project PicMe

Version 1.12

Justin Storms, Andrian Stepanyuk, Wyatt Stohr

Central Washing University

CS380 – Final Project

June 2, 2023

Table of Contents

[Project Description 4](#_Toc136706547)

[User Guide 4](#_Toc136706548)

[UML Diagrams 5](#_Toc136706549)

[Use Case Diagrams 7](#_Toc136706550)

[Testing Overview 8](#_Toc136706551)

[Database Testing 8](#_Toc136706552)

[Repositories 8](#_Toc136706553)

[Services 9](#_Toc136706554)

[GUI Testing 9](#_Toc136706555)

[Entity 9](#_Toc136706556)

[Http 9](#_Toc136706557)

[State 9](#_Toc136706558)

[Known Bugs 10](#_Toc136706559)

[Possible Feature Additions 10](#_Toc136706560)

[Login Functionality 10](#_Toc136706561)

[Picture Upload 10](#_Toc136706562)

[Post Upload 10](#_Toc136706563)

[Comment Upload/View 11](#_Toc136706564)

[User Settings and Password Management 11](#_Toc136706565)

[Friend List Management 11](#_Toc136706566)

[Friend Profile Navigation 11](#_Toc136706567)

[Extra Security Measures 11](#_Toc136706568)

[Administrator Views 12](#_Toc136706569)

[Administrator Database Management 12](#_Toc136706570)

[Appendix 13](#_Toc136706571)

Table of Figures

[Figure 1: GUI and View Controllers 5](#_Toc136706572)

[Figure 2: Database Queries and Controller 6](#_Toc136706573)

[Figure 3: PicMe Database ERD 13](#_Toc136706574)

[Figure 4: PicMe UI Blueprint 14](#_Toc136706575)

[Figure 5: Admin Login UI 15](#_Toc136706576)

[Figure 6: New Admin User Creation UI 15](#_Toc136706577)

[Figure 7: Administrator Main UI 16](#_Toc136706578)

# Project Description

Project PicMe is designed as an application which can be used to network personal information, specifically photos and reactions to those photos. It is meant to connect people through their experiences, as the basis for a social media platform’s functionality. The current version is a prototype of the databasing, and Generic User Interface (GUI) event handling required to accomplish this goal.

PicMe consists of two distinct halves: the database back-end, and the GUI front end. This is connected by controller framework which handles the database queries as well as GUI controllers which manage scene and data preservation between views. Although the prototype does not have the full functionality to connect users it does demonstrate how these profiles can be built to retrieve data from the database to populate a user’s photos and associated comments. This version demonstrates the simplicity of navigation that an end-user can expect from future versions.

A third section of the project also exists in development; however, it could not be released for demonstration due to issues with development kits. This is the Administrator interface, designed to allow tickets on database-related manipulation, exporting, and reporting to be processed on the business-end. It also enforces accountability via access and activity logging due to the sensitive nature of the user data stored within the PicMe database.

# User Guide

To install PicMe, simply place the PicMe.zip file in the directory you wish and extract the files. The executable file will be labeled as PicMe. It is recommended not to manipulate the project files otherwise, as the prototype version does not remove them from the user directory and are accessible.

Once launched, PicMe will display a Demo home screen. Know that future versions of PicMe will have profile controls and security/authentication before arriving at the home screen for a user. On this screen there are a few locations of note:

Next, we have the Photos screen. All photos from the user are present as thumbnail views. Selecting a thumbnail will update the Home screen with the selected photo and post caption then navigate the user to the Home screen where it can be viewed.

Last is the Profile screen. Here the current picture selected for the Home screen will also be visible, but PicMe will display the user’s profile information.

# UML Diagrams

A picture containing text, screenshot, diagram, parallel

Description automatically generated

Figure : GUI and View Controllers

A picture containing text, screenshot, diagram, parallel

Description automatically generated

Figure : Database Queries and Controller

# Use Case Diagrams

# Testing Overview

The PicMe application was tested in its two zones: the back-end database, and the front-end GUI. In almost all cases Junit testing was utilized as the test method.

## Database Testing

The database consists of a SQL server which communicates via Http requests. Rather than direct SQL calls from the local database, it is instead designed to move the concept seamlessly to an online database which would process the requests in a similar fashion with minor changes to the methods to call that data. While not all functionality is present in the prototype, the database was tested as new data pull requests were necessary during development. These were designed to be expanded upon as the project continues in development. While not all testing listed are for processes the current prototype utilize, testing was still created for sections still being developed, such as the additional features which will be listed later and the Admin interface.

Testing was implemented to verify that the various necessary operations for interacting with a database were accounted for. These tests include:

### Repositories

Admin Repository Testing

These tests cover the Admin repository where the usernames and passwords for administrators are stored. Affected areas: SQL insertions of new users, username lookups by ID, and account deletions.

Comment Repository Testing

Comment repository testing covers

Friend Repository Testing

The friend repository testing

Person Repository Testing

Person, or the commercial end-user, repositories test for

Picture Repository Testing

Pictures underwent repository testing to ensure that their unique typing (Blob) as stored in the database was being properly requested.

PostPicture Repository Testing

Posts associated with pictures were tested to ensure the Picture entity’s primary keys could be properly referenced and returned as foreign keys in the PostPicture database relation. These post IDs could return post IDs which can be used with the Post Relation to return post text.

Post Repository Testing

Post repository testing checks for Person IDs to be able to reference the foreign keys in the Post database entity and return any post ID associated with that user.

### Services

Admin Service Test

Friend Service Test

Person Service Test

Picture Service Test

Post Service Test

## GUI Testing

Due to the conflicts caused by the modular nature of JavaFX, Junit testing was not able to be performed on the GUI directly. Instead, the Entity, Http, and State packages were checked to ensure that functionality was not lost when transitioning from the Junit test housing in the PicMe1.12 database application to the PicMe GUI directories.

### Entity

### Http

### State

# Known Bugs

# Possible Feature Additions

Given the result of this early prototype and an underestimation of the manpower and time required to complete the task, there are many future additions which could be added to project PicMe. However, each addition makes the next easier to implement. This is due to the main barrier of feature addition and implementation being translation of the database request to useable output, due to the Gson to Json conversions (and vice-versa) required to create the requests which interact with the database, along with the database data types for data such as images not being directly compatible as objects with GUI outputs such as ImageView objects. The list includes the following (See Figure 2 in appendix for a visual representation):

## Login Functionality

Currently, the demo version of PicMe only supports a single profile navigation, though the database framework exists which supports multiple user account information. Account credentials will eventually be stored upon login, and the user ID functions as the primary key relating directly to many entities in the database. It can also be used retrieve additional primary keys from indirect entities which connect all data associated with that user. The main development needed are the views within the GUI and additional controllers within the entity, http, and state packages within the GUI directory. Most database request code necessary to pull and validate a user already exist.

Additionally, account creation is necessity to this, which will use existing methods used to upload test accounts into the database. Instead of static values, these values will be replaced with references to textField objects the user can easily manipulate within GUI.

## Picture Upload

PicMe actually uses the necessary code to upload data to the database to build the tests and demo profile. As with login functionality, the majority of what is needed to implement this are views and scenes built into the GUI. A local directory where user photos can be stored would be used to determine photos to upload. This way a user will not have to figure out where it is stored, but instead save it to the known location of PicMe in an easily identifiable folder.

Due to the way the Blob data type pulled form the database is required to store pictures; extension validation is also needed. When a picture is pulled, it has to be written as byte data to a file, which will need to be named and appended with an extension to identify file type. The database already holds the necessary data for this, and the controllers are built with data fields needed to hold the String parsing to compare them and write output appropriately.

## Post Upload

As with the prior upload feature, post upload is a simplified version which only requires a string input to be stored as VarChar data in order to implement this feature. A text field capturing the post a user has made regarding a specific picture would run an insert as a new tuple into the Post entity within the database. The view in the GUI would represent the majority of the work needed to accomplish this.

## Comment Upload/View

A comment upload is in many ways similar to a post in terms of the database interaction. It can be associated with either a post or a picture and is retrievable using the person or post ID as a primary key for lookup.

This will allow for polymorphism to handle sorting out where a comment would be assigned within the GUI views for output once pulled. The ID used as a key will determine whether it is a post, or a picture and the same output method may be used in implementation.

## User Settings and Password Management

The user control of their own profile and settings closely resembles actions taken by another future implementation of the PicMe project, the Administrator interface. The view and associated controllers will need to be developed to form the skeleton of this interaction, but it would borrow the already tested database interaction methods for uploading and downloading changes. While obvious, the timing of developing this feature would have to come after the Account Login feature as it would require validation that a user is only able to manage their own data, including the option to change user password.

## Friend List Management

While the networking for the discovery of friends through adjacency via graph theory is complex, the structure of how that data can be stored within a profile is not. Friends are already tracked through a Friend-of-Friend-to entity relation in the database. When a profile is logged in, this list can already be populated creating an immediate friends list. Discoverability through graph calculation will have to be adjusted during beta testing, as too much discovery from “friend nodes” too far away from the initial friend list loses it meaning, while single step navigation though the graph would yield a limited number of results that could prove to be detrimental in nurturing new connections. Additionally, the “distance” between friends on the graph can be weighted by similarities compared upon between user profiles, allowing for nodes physically farther away to be considered as potentially returnable as a possible friend to request.

The user would interface with this idea using a list abstracted form it, populated by a view in the GUI. Selecting a profile and committing to sending a request would allow for verification of connection. The user themselves would be able to see their own request view, where they would approve or deny profiles who send requests.

## Friend Profile Navigation

Friend profile navigation is the resulting functionality of the list of person IDs stored as friends within their profile, retrieved at login. This feature actually would reuse the code for views of the current profile, with notable exclusions of access to account data management and personal data marked private if not verified as the actual user. This would allow the user to navigate a profile as though it were their own, resulting in less time learning the interface.

## Extra Security Measures

While a password itself can be a preventative measure in limiting unwanted access to accounts, both the Commercial and Administrative users will also require a security token input to verify the user. This will be required upon account login and password change. User-generated passwords may not follow good practices to prevent infiltration. A rotating key via token is necessary to reduce the event of brute-forcing login attempts. Up to one billion passwords per second (potential worst case) can be sent by computer as instructions and will try all intermediate values aside from “dictionary entries” commonly found in most passwords. The token’s rotation helps to invalidate brute for attempts up to a certain number of instructions per second. The administrator interface within the PicMe project already enforces a user password with a strength of 72 billion combinations. This is meant to be combined with authentication failure lockouts and prompts to update password with sufficient suspicious logins exist.

## Administrator Views

While mostly complete, the administrator interface still requires testing as the commercial end of the project aligned with the demo on how the prototype could be used as the basis of a social media platform. GUI and controllers are established, but without testing it cannot be released accompanying the rest of the demo. This feature will probably require the least time to bring online but is not a priority until features of the Commercial-user PicMe interfaces are complete. Admin GUI design can be seen below as Figure 3, Figure 4, and figure 5 in report Appendix.

## Administrator Database Management

This feature, while mostly complete being an extension of the Administrator Views, would need to undergo redevelopment if the structure of the Commercial-user interactions changes needs within the database. Therefore, it is better to redesign this in as few iterations as possible once the other aspects of the project are more established.

The administrator database management operates on a different scope than user database manipulation. While the goal of the Commercial-user is connection and sharing, the database admin would use this interface to process ticket requests on users. IDs can be used to pull various user information for export to a secure location. This is in anticipation of legal requirements for the data handling and requests such as law enforcement compliance. All admin user activities are tracked as logs describing the interaction for every execution, upholding accountability and would be output to a secure location for storage. These features already exist within the project structure, but changes may need to be made to implementations as the features on the Commercial end develop.

# Appendix

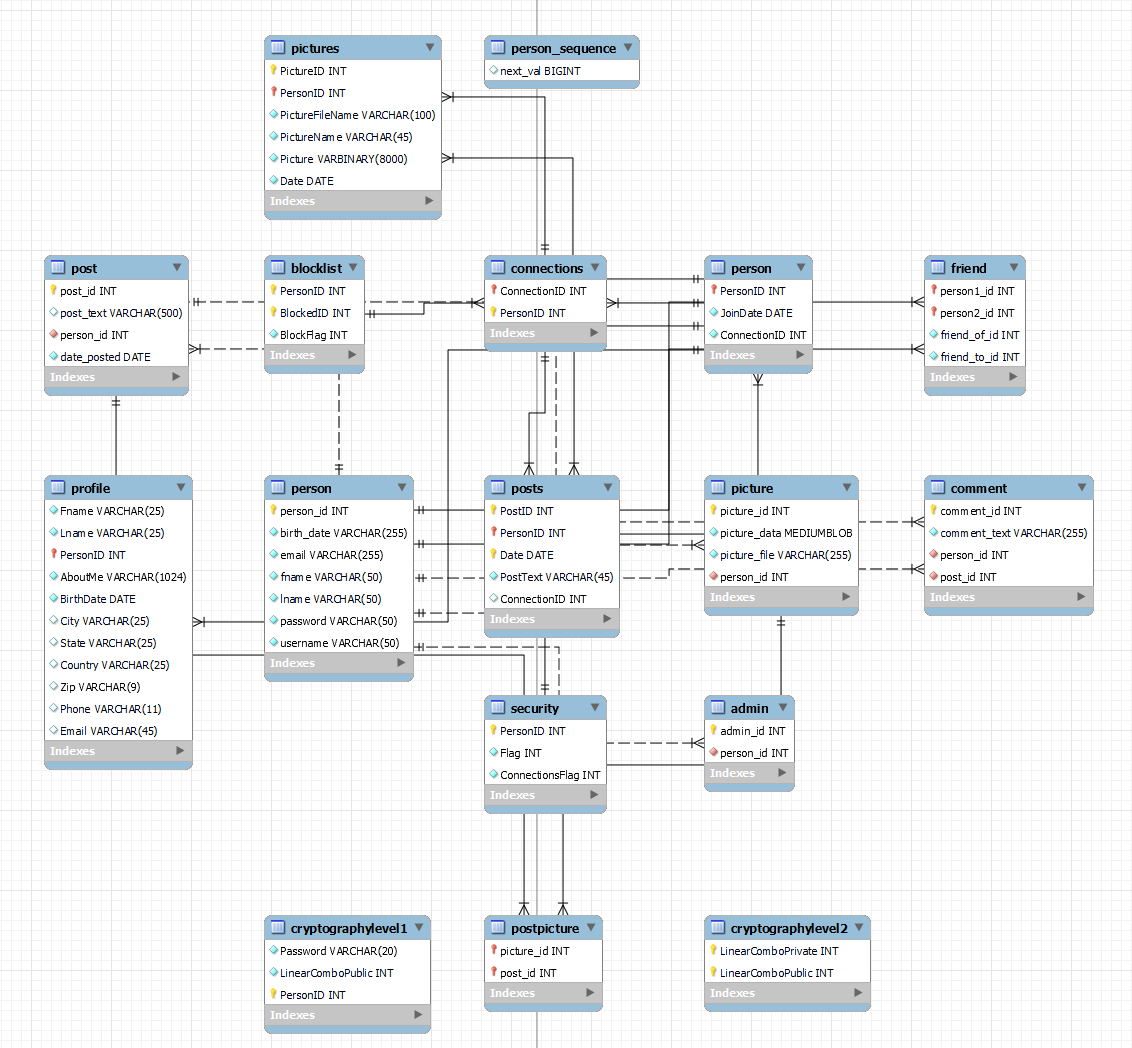


Figure : PicMe Database ERD

A picture containing text, diagram, map, plan

Description automatically generated

Figure : PicMe UI Blueprint

A screenshot of a computer

Description automatically generated

Figure : Admin Login UI

A screenshot of a computer screen

Description automatically generated with medium confidence

Figure : New Admin User Creation UI

A screenshot of a computer

Description automatically generated

Figure : Administrator Main UI