

|  |
| --- |
| Detailed Design Document  Fairfield Connect Application |
|  |
| December 16  Grad Student Team  Members: Tom Grimes, Hien Tran, Ram Tamanampudi, Likitha Varapana |

SWEG 5301

Table of Contents

[Introduction 4](#_Toc90560813)

[Architecture 5](#_Toc90560814)

[Module Design 6](#_Toc90560815)

[Users Module 6](#_Toc90560816)

[User Sign In 7](#_Toc90560817)

[User account retrieval 7](#_Toc90560818)

[Update user account 7](#_Toc90560819)

[Admin sign in 7](#_Toc90560820)

[Authentication Module 8](#_Toc90560821)

[User Sign In 9](#_Toc90560822)

[Admin sign in 10](#_Toc90560823)

[Database Module 10](#_Toc90560824)

[System Inputs module 10](#_Toc90560825)

[Validate Input Parameter Module: 11](#_Toc90560826)

[Data Transformation Module: 12](#_Toc90560827)

[DAC Data Access Control Module: 12](#_Toc90560828)

[High Level Module Design 12](#_Toc90560829)

[Database 13](#_Toc90560830)

[File Server 14](#_Toc90560831)

[Landlord Request Module Design 14](#_Toc90560832)

[Post Manager 15](#_Toc90560833)

[Poster 15](#_Toc90560834)

[Thumbnailer 15](#_Toc90560835)

[Abstract Data Types 17](#_Toc90560836)

[User 17](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560837)

[UserLog 17](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560838)

[Post 17](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560839)

[Landlord Review 17](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560840)

[Comments 18](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560841)

[Landlord DB 18](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560842)

[Requests 18](file:///C:\Users\e359043\Documents\Personal\DDDGradStudents.docx#_Toc90560843)

[Database Design 19](#_Toc90560844)

[Overview 19](#_Toc90560845)

[Entity Relationship Diagram 19](#_Toc90560846)

[Comprises Diagram 20](#_Toc90560847)

[Uses Diagram 21](#_Toc90560848)

[Integration Test Plan 22](#_Toc90560849)

[GUI Module: 22](#_Toc90560850)

[Login 22](#_Toc90560851)

[Incorrect Email ID 22](#_Toc90560852)

[Incorrect Password 22](#_Toc90560853)

[User Signup 23](#_Toc90560854)

[User signup 23](#_Toc90560855)

[Invalid Email ID 23](#_Toc90560856)

[Enter valid Email ID 24](#_Toc90560857)

[Email ID already in use 24](#_Toc90560858)

[Weak Password 24](#_Toc90560859)

[User leaves blanks in the fields 25](#_Toc90560860)

[Reset Password 25](#_Toc90560861)

[Invalid Email given on Reset password option 26](#_Toc90560862)

[GUI TESTING 29](#_Toc90560863)

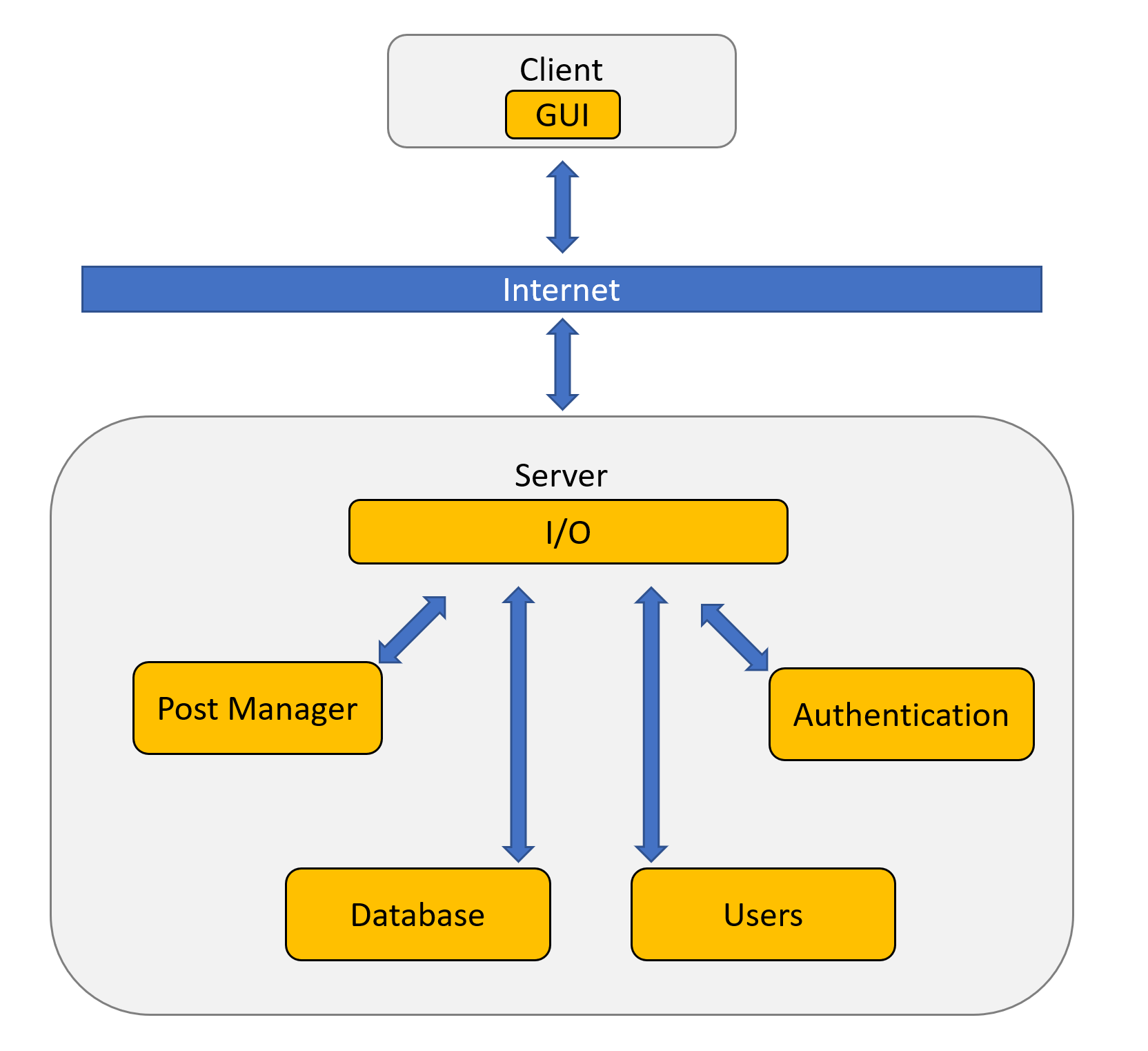
# Introduction

Preparing for and living life on a new campus is extremely complicated and can be stressful for any student. Students coming from out of town and especially international students have to rely on the internet to find people to socialize, and study with. Perhaps most importantly, they have to decide where to live as well. Currently, they’re using Facebook, Realator.com, and Zillow to find listings for these sorts of things. Unfortunately, it can be hard to find a site that’s hosting truly unbiased information or being written by someone from the same perspective. Someone without experience riding local public transit may see a nearby bus stop on a map and choose to live somewhere based on that information. However, once moved in, they realize their commute to school is well over an hour after transferring busses, and the bus breaks down frequently on this line. There’s also an incentive for those hosting property listings to scrub negative reviews from their platforms. These sites rely on the revenue from property managers listing on their services and often let them curate comments left on their listings. A site hosted by the property company themself has no real incentive to paint anything but the rosiest picture in order to rent the property. This leaves students without a way to know they’re getting relevant information from the resources available  
  
Fairfield Connect

Our application, Fairfield Connect, aims to solve the problems we’ve stated by building trust with peer generated content. Verified students will be able to leave reviews of their housing and find roommates using a web application built on established message board concepts. Students will want to use the sight because they know they can leave an honest review that wont get scrubbed by owner interference. Other students will gain confidence in the information they’re finding because they know it’s been generated by other real students with a similar perspective. To capitalize on the strengths of a student only environment. Other relevant features have been added to help students study, socialize, and participate in a student to student marketplace.

# Architecture

To meet both functional and non-functional requirements which were derived from careful analysis of the business needs, a server client architecture was chosen. This architecture best allows a large number of users to simultaneously interact with the system via a web browser. This approach makes the system scalable and maintainable, with a focus on reuse for when it comes time to expand the system and add new features as usage increases. The modules contained within this architecture have defined required and provided interfaces. The diagram below shows the overall architecture while the descriptions below provide descriptions of the sub-modules and definitions of their interfaces.



# Module Design

## Users Module

This module helps users and admins of the system to do various operations.

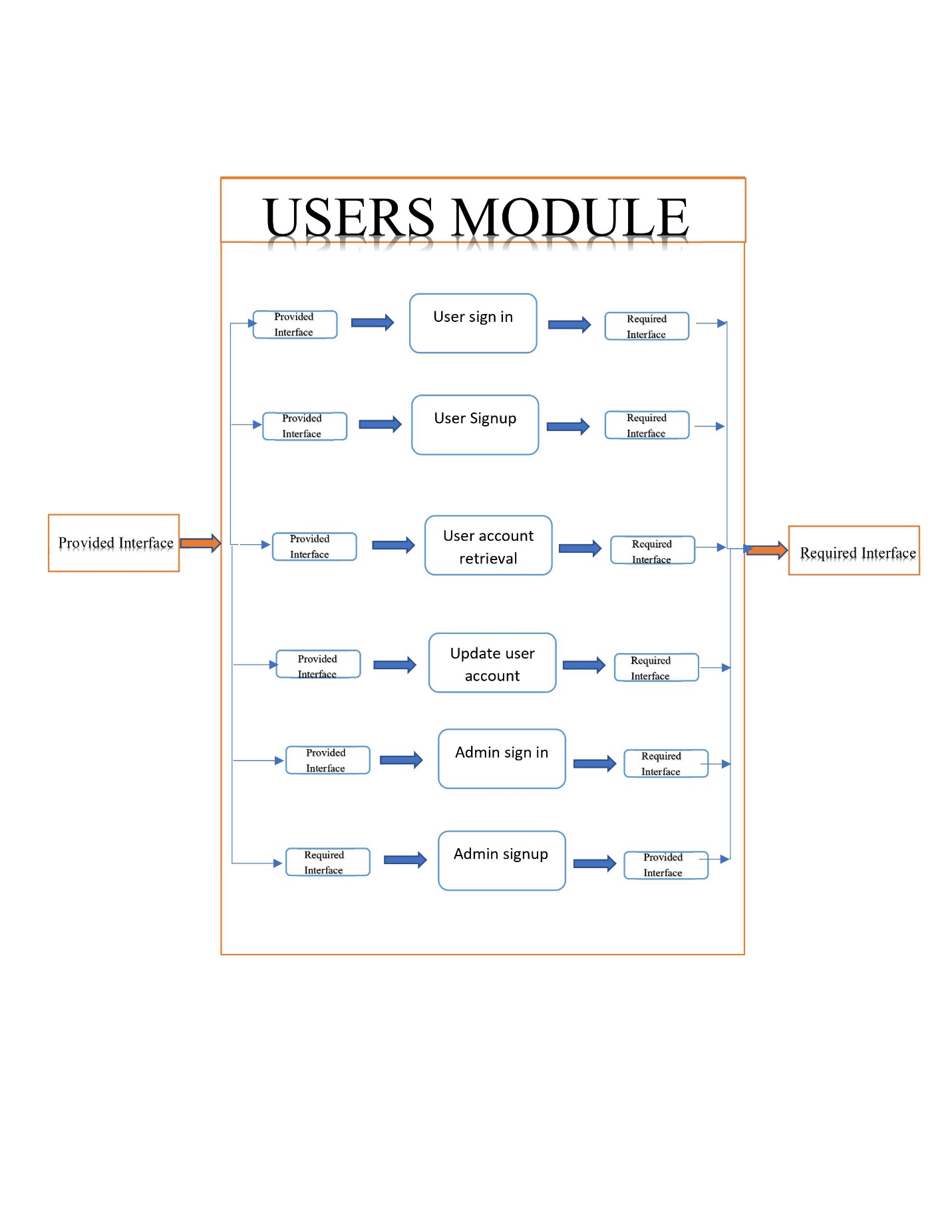
Provided interface:

This provided interface for user’s module is the combination of all provided interface of its sub components.

Required interface:

This Required interface for user’s module is the combination of all the required interface of sub components.

Subcomponents are the user to sign in or signup up for account, retrieves user account password if forgotten, updates users account, admin account login, admin account password recovery



Architecture:

### User Sign In

This component enables user to sign in into the system to use the services that being provided by the software.

Provided Interface:

The Provided interface for this module is email id and password

Email\_id Varchar2(20)

Password varchar2(30)

Required Interface:

The required interface enables the user to get into the system with their credentials, this requires user account created in the data base.

### User account retrieval

Whenever the user forgets the password for their account this component helps user in getting it back.

Provided Interface::

Email\_id Varchar2(20)

Stag\_id int

Verification\_code int

new\_password Varchar2(20)

reenter\_password varchar2(20)

Required Interface:

The required interface helps in resetting password for user account, it requires user account already created in the data base

### Update user account

User wants to update their personal information on their account this component helps in updating the previous user data.

Provided Interface:

The provided interface for this module is first name, last name, address & phone number

FirstName varchar (255)

LastName varchar (255)

Address varchar (255)

Phone\_Number int (10)

Required Interface:

The required interface helps in updating user’s data on user account, it requires user account already created in the data base

### Admin sign in

This component enables admin to sign in into the system to use the services that being provided by the software and also some extra features like maintaining control over the posts, user accounts.

Provided Interface:

The provided interface for this module is email id, password and admin id

Email\_id Varchar2(20)

Password varchar2(30)

Admin\_id int

Required Interface:

The required interface enables the admin to get into the system with their credentials, this requires admin account created in the data base.

## Authentication Module

This module verifies the users and admins when they are signing into the system and also at the time of user account retrieval.

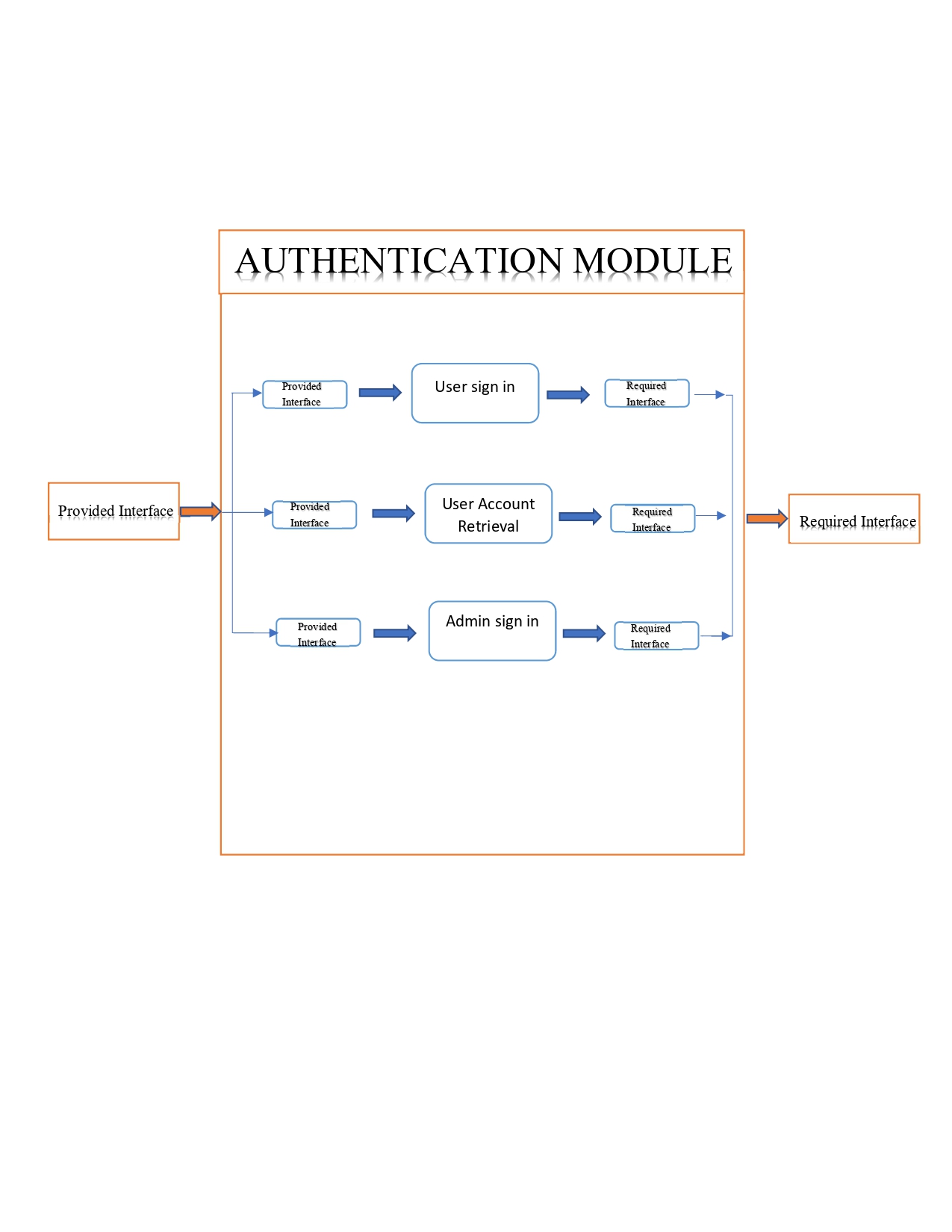
Provided interface:

This provided interface for user’s module is the combination of all provided interface of its sub components.

Required interface:

This Required interface for user’s module is the combination of all the required interface of sub components.

Subcomponents are the user to sign, retrieves user account password if forgotten, admin account login, admin account sign in



Architecture:

### User Sign In

This component authenticates the user when trying to sign in into the system to use the services that being provided by the software.

Provided Interface

The Provided interface for this module is email id and password

Validateuser.Email\_id

Validateuser.Password

Required Interface:

The required interface enables the admin to get into the system by verifying the entered credentials with their credentials, this requires admin account created in the data base.

User account retrieval:

This module helps in authentication or verifying the user data whenever user is trying to reset password for his account.

Provided Interface:

Validateuser.Email\_id

Validateuser.Stag\_id

Validateuser.Verification\_code

Required Interface:

The required interface helps in resetting password for user account, it requires user account already created in the data base

### Admin sign in

This component authenticates the admin when trying to sign in into the system to maintain the website and doing work on the site.

Provided Interface:

The provided interface for this module is email id, password and admin id

Validateuser.Email\_id

Validateuser.Password

Validateuser.Admin\_id

Required Interface:

The required interface enables the admin to get into the system by verifying the entered credentials with their credentials, this requires admin account created in the data base.

## Database Module

The module is further divided into lower-level modules namely

* System Inputs Module
* System Output Module

### System Inputs module

Purpose:

The purpose of this module is to validate input parameters and file formats of the input files of the system.

Provided Interface:

The provided interface for this module is the union of the provided interfaces of all its sub-modules.

Required Interface:

The required interface for this module is the union of the required interfaces of all its sub-modules.

High Level Module Design:

This module is further divided into lower-level modules like validate input parameter module, validate input parameter module, Validate User.

Diagram

Description automatically generated

### Validate Input Parameter Module:

Purpose:

The purpose of this module is to validate the input parameters provided by user to the system. If the validation fails, the file will not be processed.

Provided Interface: The required interface for this module is

boolean validInputParameter(File fileName)

Required Interface: The required interface for this module is

performDataPreProcessing(File fileName)

* System Output Module:

Purpose:

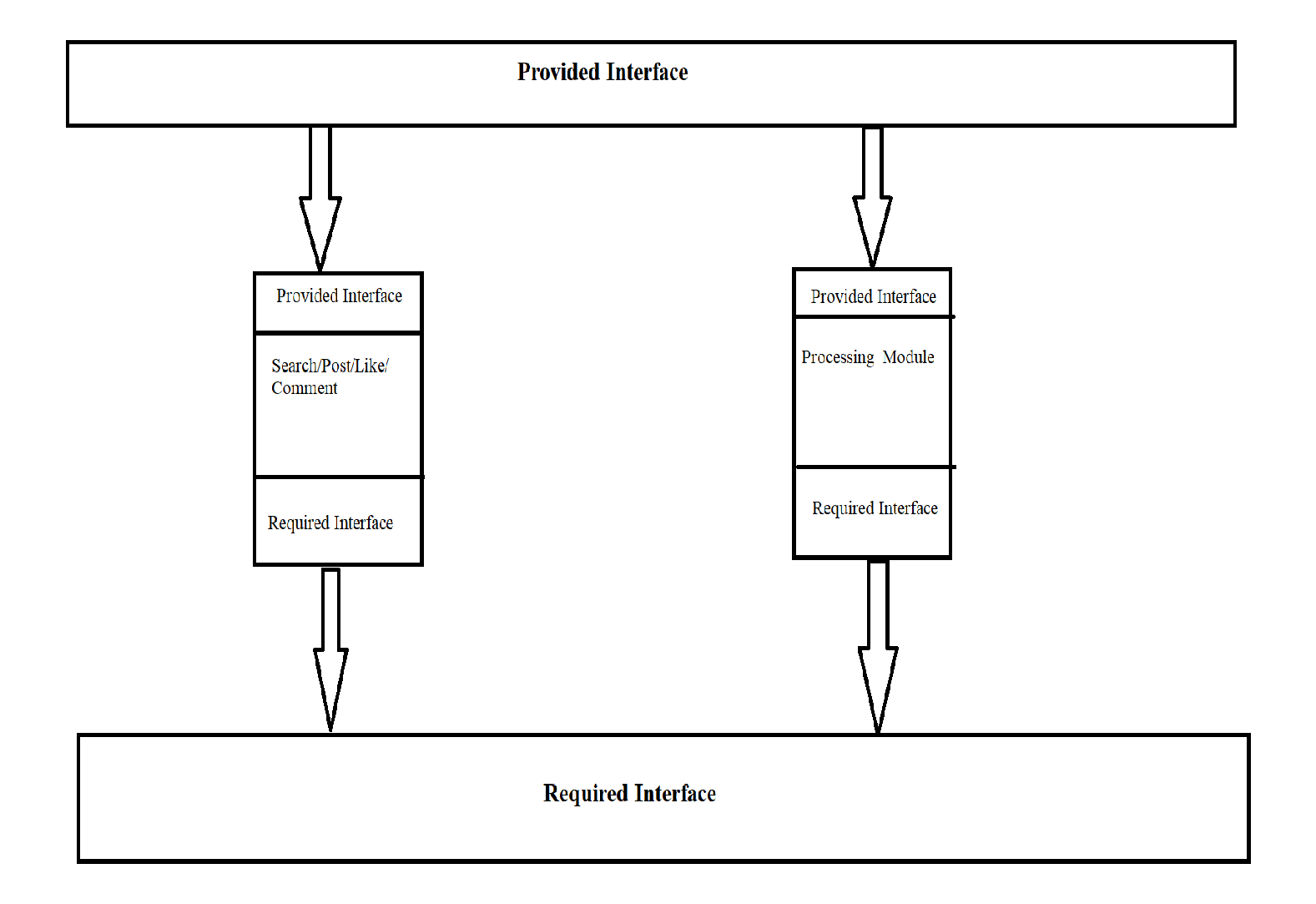
The purpose of this module is to generate the Output.

Provided Interface:

The provided interface for this module is the Search/Post/Like/Comment functionality with the application.

Required Interface:

The required interface for this module is the DAC module.



### Data Transformation Module:

Purpose:

This purpose of this module is to transform the processed data into the required format needed by the DAC module.

Provided Interface:

Controller transformData(List <Object> in User/Admin)

Required Interface:

Controller DAC (Function, details File)

### DAC Data Access Control Module:

Purpose:

The purpose of this module is to get the data transformed from system inputs based on the inputs and access the Database.

Provided Interface:

This module is the core data processing modules and will be called from all processing modules.

Required Interface:

This module interacts with the data base/File Server.

### High Level Module Design

This module receives the input request and the desired function to be performed along with the database collection. Below are the high-level operations:

When Create

Db.< Collection>. Insert one

{User schema as per the collection, Status}

Db.< Collection>. Insert many

{User schema as per the collection, Status}

When Read/Validate

Db.< Collection>Search

{User Query} validate

When Update

Db.< Collection>. Updateone

{User Update}{$set : (Status}

When UpdateAll

Db.< Collection>. Updatemany

{User Query}. limit(modifier)

When AdminCreate

Db.<Collection>.Adminone

{Admin schema as per the collection,Status}

When AdminValidate

Db.<Collection>.Search

{AdminQuery} validate

When AdminUpdate

Db.<Collection>.Adminone

{Admin Update} {$set : (status)}

When AdminUpdateAll

Db.<Collection>.Adminmany

{Admin Update}. limit(modifier)

When Post

Db.<Collection>.PostOne

{User Post} {$set: (status) successfully}

When Post

Db.<Collection>.PostMany

{User Post} {$set:(status) limit}

When Delete

Db.< Collection>. Deleteone

{User Delete} {$set : (Status)}

### Database

Purpose:

The purpose of this module is to allow the users to store the data and retrieve data from database.

Required Interface:

DAC Module is the one way to interact with database.

### File Server

Purpose:

This purpose of this module is to save the Document files into the database.

Provided Interface:

No provided interface is needed for this module.

Required Interface:

DAC Module is the one way to interact with database.

* Server:

Purpose:

The purpose of the server is a system that provides resources, data, services, or programs to other computers, known as clients, over a network. An individual system can provide resources and use them from another system at the same time. This means that a device could be both a server and a client at the same time.

Provided Interface:

No provided interface is needed for this module.

Required Interface:

No Required interface is needed for this module.

### Landlord Request Module Design

Purpose:

The purpose of this module is to provide the LandLord requested by User

Provide Interface:

The provided interface for this module is

Controller.LandLord(Email,UserName,Password)

Controller.CreatePost (Add Post,Update Post,Delete Post)

Required Interface:

UserLogin Page

* Review LandLord Module Design:

Purpose:

The purpose of this module is to review the LandLord.

Provide Interface:

The provided interface for this module is

Controller.RevuewLandLord(ReviewID,LandLordID,UserID)

Controller.Post (Title,Content,Rating)

Required Interface:

UserLogin Page

EditPost Page

## Post Manager

### Poster

Purpose: module Post is responsible for creating, editing, deleting and sorting posts

Attributes:

* Post ID
* User ID
* Category ID
* Title
* Description
* Date

Methods:

Required Interface:

* Post Management
* Post Data

Provided Interface:

Create

* GetID (int)
* SetID (int)
* GetCategoryID (int)
* SetCategoryID (int)
* GetTitle (String)
* SetTitle (String)
* GetDescription (String)
* SetDescription (String)
* GetDate (DateTime)
* SetDate (DateTime)

Edit ()

Delete ()

Sort ()

### Thumbnailer

Purpose: module Post is responsible for uploading, deleting thumbnails and checking thumbnails’s size

Attributes:

* Thumbnails ID
* Post ID
* Name
* Size

Methods:

Required Interface:

* Post Management
* Thumbnails Data

Provided Interface:

Upload

* GetID (int)
* SetID (int)
* GetPostID (int)
* SetPostID (int)
* GetName (String)
* SetName (String)
* GetSize (decimal)
* SetSize (decimal)
* CheckSize(bool)

Open ()

IsFound()

Delete ()

# Abstract Data Types

User Type

UserID ObjectID

StudentID ObjectID

FirstName String

LastName String

Password String

Email String

Validation(Y/N) Boolean

UserLog Type

UserID ObjectID

Type String

TimeStamp String

Success Boolean

Post Type

GenericID ObjectID

PosterID PostID

Title String

Category String

Rating String

Locked Boolean

Verified Boolean

Admin Boolean

Body String

Landlord Review Type

LandlordID ObjectID

PostID PostID

Address String

Unit Number

Rent Number

Review String

Image String

Comments Type

CommentID ObjectID Body String CreateTime Date UpdateTime Date Admin Boolean

Landlord DB Type

LandlordID ObjectID

Landlord Name String

Units Number

Rating Number

Hidden Boolean

Req ID String

Requests Type

RequestID ObjectID MessageID ObjectID

SenderID ObjectID

Type String

Approved Boolean

# Database Design

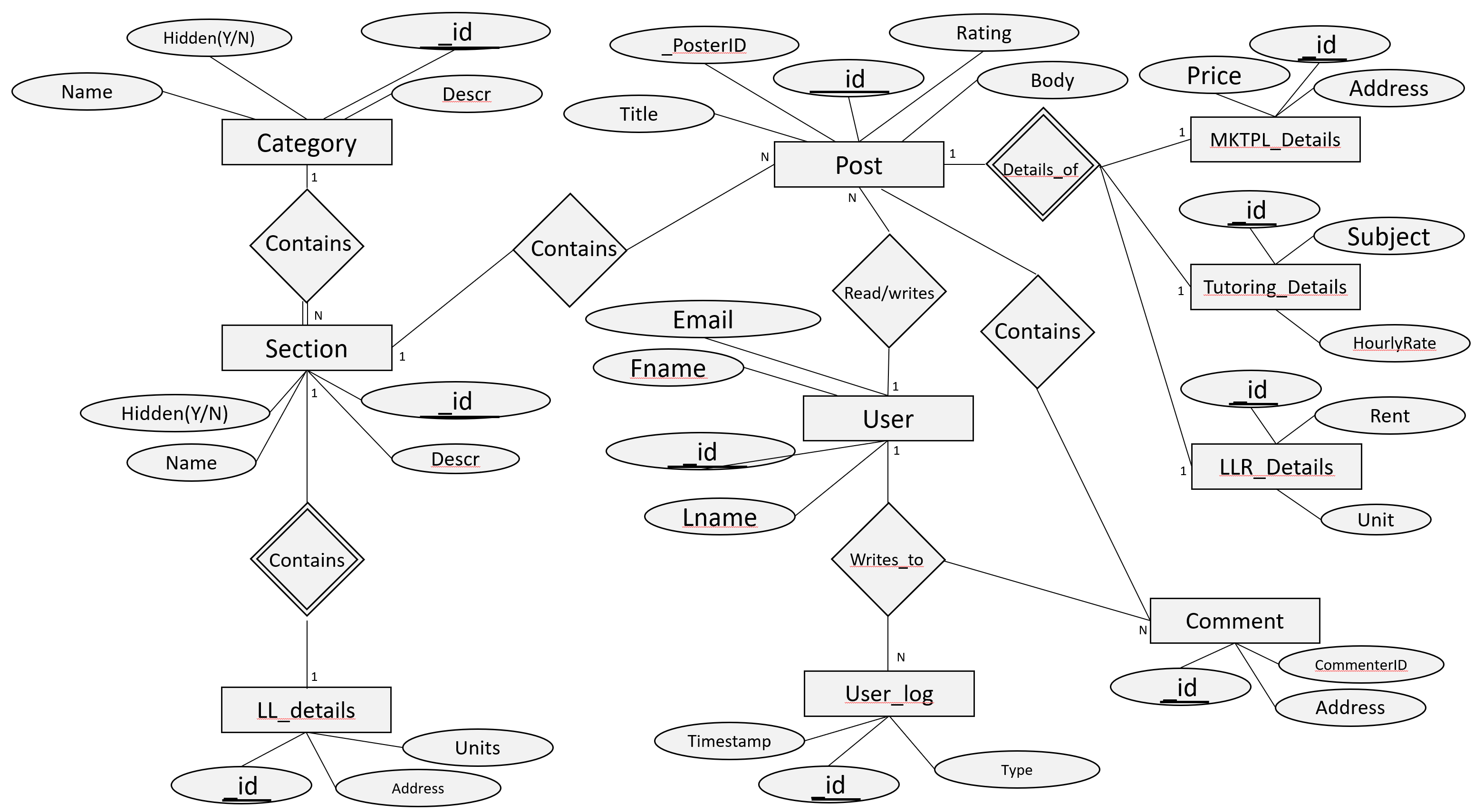
## Overview

The schema presented describes the database used to house the data used by the Fairfield Connect application and the relationships between the data. User data is contained in a way to maximize traceability. The data is stored in a way that facilitates searching and sorting the posts to achieve the business requirements of the application. The database is structured in such a way that new type specific attributes can be added to new kinds of posts without affecting the existing post database.

The database design for Fairfield Connect implements:

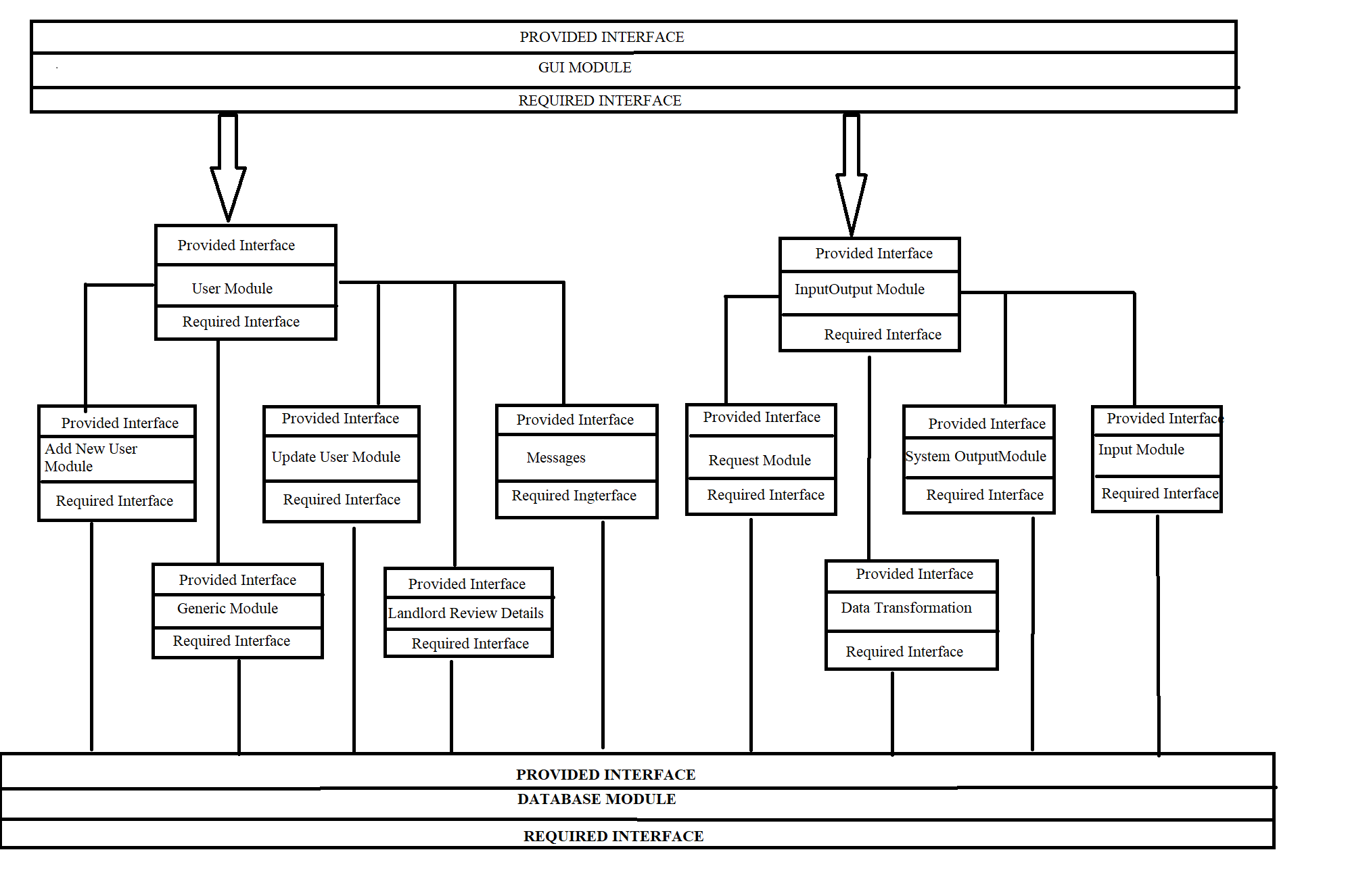
* + Information organized to support business logic
  + Stores user added data to the correct location
  + Contains metadata to help sort and search posts
  + Keeps logs of user actions

## Entity Relationship Diagram



# Comprises Diagram

# Uses Diagram



# Integration Test Plan

## GUI Module:

|  |  |
| --- | --- |
| Test Case | Login |
| Test case ID | 1 |
| Description | Check results on entering valid Email ID & Password  If no error User will be logged in successfully.  User has entered a valid Email ID and Password |
| Input | Enter email and password |
| Output | Logged in when input credentials are accepted |
| What is being tested? | When user enters a valid email id and password  Then Clicks on login. |
| Testing constraints | Email ID and password should be correct |

|  |  |
| --- | --- |
| Test Case | Incorrect Email ID |
| Test case id | 2 |
| Description | Check results on entering Invalid Email ID |
| Input | Enter username and password |
| Output | Returns Error message:  Enter valid Email ID or password,  if you hadn’t created an account, please create one |
| What is being tested? | User enters the email id then  Hits Login. |
| Testing credentials | Email ID should be Incorrect |

|  |  |
| --- | --- |
| Test Case | Incorrect Password |
| Test Case id | 3 |
| Description | Check results on entering Incorrect Password |
| Input | Enter username and password |
| Output | Returns Error message:  Enter valid Email ID or password,  if you hadn’t created an account, please create one |
| What is being tested? | When user enters the wrong password then  Hits Login. |
| Testing credentials | Password should be Incorrect |

## User Signup

|  |  |
| --- | --- |
| Test Case | User signup |
| Test Case id | 4 |
| Description | User selects create account  Next screen enters first name, last name, email address (user ID), password, phone number, and info about themselves |
| Input | Fill all the parameters and click on create Account. |
| Output | Account created successfully! |
| What is being tested? | When a user wants to login into website,  If he doesn’t have an account, then he will be asked for creating an account. |
| Testing credentials | Sign up |

|  |  |
| --- | --- |
| Test Case | Invalid Email ID |
| Test case id | 5 |
| Description | User selects to register and in the next screen enters first name, last name, email address, password, phone number, and info about themselves |
| Input | Enter valid Email ID |
| Output | Return email address is invalid. Try Again. |
| What is being tested? | User is trying to register account by using his email id.  System will not accept the email id if the domain name is wrong or the email id does not exist. |
| Testing credentials | Email ID should be correct with a valid domain name. |

|  |  |
| --- | --- |
| Test Case | Email ID already in use |
| Test case id | 6 |
| Description | User selects to register and in the next screen enters first name, last name, email address, password, phone number, and info about themselves |
| Input | Enter valid Email ID |
| Output | Return email address is already in use. Try Again. |
| What is being tested? | User is trying to register account by using his email id.  System will not accept the email id if the already in use. |
| Testing credentials | Email ID must be unique. |

|  |  |
| --- | --- |
| Test Case | Weak Password |
| Test case | 7 |
| Description | User selects to register in the next screen enters first name, last name, email address, password, phone number, and info about themselves |
| Input | Give valid Password. |
| Output | Return password is too weak, use at least 1 special character and number |
| What is being tested | User is creating the password for the account.  Should follow the constraints and create password accordingly. |
| Testing credentials | At least 1 special character and number must be used. |

|  |  |
| --- | --- |
| Test Case | User leaves blanks in the fields while registering an account |
| Test Case id | 8 |
| Description | User selects to register in the next screen, user clicks on register without entering first name, last name, email address , password, phone number, and info about themselves |
| Input | Enter details in all the required fields. |
| Output | Please fill out the mandatory fields |
| What is being tested? | User needs to enter data. |
| Testing credentials | Mandatory fields are being tested. |

Reset Password

|  |  |
| --- | --- |
| Test Case | Reset Password |
| Test Case | 9 |
| Description | User selects forgot password option, fills out their email, clicks forgot/reset password, and receives email to reset password. |
| Input | Give Valid Email address. |
| Output | User receives email with reset password |
| What is being tested? | Forgot/Reset Password of user unable to recollect. |
| Testing credentials | Email address should be valid. |

|  |  |
| --- | --- |
| Test Case | Invalid Email given on Reset password option. |
| Test case id | 10 |
| Description | User selects forgot password option, fills out their email, clicks forgot/reset password, and receives email to reset password. |
| Input | Give In Valid Email ID. |
| Output | Email ID is invalid. |
| What is being tested? | Email id when trying to reset password. |
| Testing credentials | Email ID must be valid. |

|  |  |
| --- | --- |
| Test Case | Comment |
| Test case id |  |
| Description | User Comment to a result Document |
| Input | Add comment |
| Output | Response error to login |
| What is being tested? | Comment functionality |
| Testing credentials | Add comments and hit save. |

|  |  |
| --- | --- |
| Test Case | Comment |
| Test case id |  |
| Description | User Comment to a result Document |
| Input | Add comment |
| Output | Comment save successfully |
| What is being tested? | Comment functionality |
| Testing credentials | Add comments and hit save. |

|  |  |
| --- | --- |
| Test Case | Post |
| Test case id |  |
| Description | Post that is created by User |
| Input | Add Post |
| Output | Post displayed |
| What is being tested? | Post functionality |
| Testing credentials | Add Post and hit save |

|  |  |
| --- | --- |
| Test Case | Notification |
| Test case id |  |
| Description | Notification that User receives when post is created successfully |
| Input | Add Notifications |
| Output | Generate Notification when post created |
| What is being tested? | Verifies the notification |
| Testing credentials | Verifies Notifications and hit save |

|  |  |
| --- | --- |
| Test Case | Thumbnail |
| Test case id |  |
| Description | Picture that User uploads |
| Input | Add Thumbnail |
| Output | Post Thumbnail |
| What is being tested? | Picture functionality |
| Testing credentials | Add Picture and hit save |

|  |  |
| --- | --- |
| Test Case | Landlord |
| Test case id |  |
| Description | Landlord that is reviewed by User |
| Input | Verifies the landlord |
| Output | Post by the landlord |
| What is being tested? | Landlord functionality |
| Testing credentials | Verifies landlord and hit save |

|  |  |
| --- | --- |
| Test Case | Landlord Request |
| Test case id |  |
| Description | New Landlord is requested by user |
| Input | Verifies the landlord |
| Output | New Landlord |
| What is being tested? | Landlord functionality |
| Testing credentials | Verifies landlord and hit save |

|  |  |
| --- | --- |
| Test Case | Category |
| Test case id |  |
| Description | Category in each section |
| Input | Add Category |
| Output | New category created |
| What is being tested? | Category functionality |
| Testing credentials | Verifies Category and hit save |

|  |  |
| --- | --- |
| Test Case | Section |
| Test case id |  |
| Description | Sections are like Marketing, Tutoring, Social, Housing |
| Input | Add sections |
| Output | Sections created |
| What is being tested? | Verifies the Sections |
| Testing credentials | Verifies Sections and hit save |

GUI TESTING

|  |  |
| --- | --- |
| Test Case | GUI Testing |
| Test case id |  |
| Description | User tests from Chrome |
| Input | Login and checks all functionality |
| Output | Results returned |
| What is being tested? | Browser compatibility |
| Testing credentials | Open Chrome and Check all functions of the website |

|  |  |
| --- | --- |
| Test Case | GUI Testing |
| Test case id |  |
| Description | User tests from Safari |
| Input | Login and checks all functionality |
| Output | Results returned |
| What is being tested? | Browser compatibility |
| Testing credentials | Open Safari and Check all functions of the website |

|  |  |
| --- | --- |
| Test Case | GUI Testing |
| Test case id |  |
| Description | User tests from Firefox |
| Input | Login and checks all functionality |
| Output | Results returned |
| What is being tested? | Browser compatibility |
| Testing credentials | Open Firefox and Check all functions of the website |

|  |  |
| --- | --- |
| Test Case | GUI Testing |
| Test case id |  |
| Description | User tests from Binge |
| Input | Login and checks all functionality |
| Output | Results returned |
| What is being tested? | Browser compatibility |
| Testing credentials | Open Binge and Check all functions of the website |