System Architecture Document

Kenan Karavoussanos
Shaylin Pillay
Preshen Goobiah
Marc Karp

October 8, 2018

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Overview . . . 3

2 Architectural Goals and Constraints **3**

3 Architectural Representation **4**
3.1 Architectural Views . . . 4
3.2 Architectural Design Patterns . . . 5

4 Architectural View Decomposition **6**
4.1 Use-Case View . . . 6
4.1.1 Use Case Narrative . . . 6
4.2 Design View . . . 8
4.2.1 Class Diagram . . . 8
4.2.2 Hardware Architecture Diagram . . . 9
4.2.3 Software Architecture Diagram . . . 10
4.3 Process View . . . 11
4.3.1 Create Application . . . 12
4.3.2 Update Application . . . 13
4.4 Component View . . . 13
4.4.1 Common User Screens . . . 14
4.4.2 Post Graduate Officer . . . 14
4.4.3 Supervisor . . . 17

4.4.4 Post Graduate Coordinator 18
4.5 Database View 19

5 Performance and Unit Testing **20**
5.1 Performance Test Details and Results 20
5.2 Unit Testing 21

1 Introduction

This introduction provides a brief overview of System Architecture Document for the current iteration of the Post Graduate Application Approval System. It consists of the purpose, scope, problem statement,project objectives, stakeholders and overview of the rest of the document.

1.1 Purpose

This document provides the reader with an architectural overview of the Post Graduate Application Approval System. The primary purpose of this project is to create a single integrated system that facilitates application review and decision making. For a full description of system requirements please see the Software Requirements Specification Document

This document is intended to elucidate the major architectural decisions that have been made when designing and implementing the system. This is achieved by viewing the system architecture from various perspectives, called views. These views are intended to explain the system architecture through all levels of the development stack, from front-end to back-end.

1.2 Scope

The scope of this document is the design and implementation of the software based Post Graduate Application Approval System which consists of the application upload by the Post Graduate Officer, the intermediary application review by the Supervisor and the final application review made by the Post Graduate Coordinator.

1.3 Overview

The rest of the document will briefly discuss the architectural goals and constraints of the environment followed by a brief description of the chosen architectural views and patterns. The architectural views will then be displayed in detail. Finally, the performance evaluation of the system and a description of the software testing will be discussed.

2 Architectural Goals and Constraints

The architecture of the system has been designed to achieve the following objectives:

1. To assist the application approval process by having all application documents and information in a single digital repository.
2. To assist the Post Graduate Officer with the upload of application documents.
3. To simplify application reviews by having a single integrated view of all important information for each decision maker.
4. To prevent human error by providing notifications and weekly reminders about pending applications.

The significant constraints kept in mind when developing the system were as follows:

1. Security
2. Ease of Use
3. Paperless

3 Architectural Representation

3.1 Architectural Views

The development of the system has various contributors each with their own priorities and tasks. As such, the system needs to be documented from various perspectives to aid, and eventually validate, the completion of a contributor's tasks. The system architecture shall be represented from the following views:

1. Use Case View: This defines the high-level interactions between various actors and the system.
2. Design View: This contains the class and architecture diagrams of the system.
3. Process View: This displays the processes within the system that combine to perform the various interactions defined in the Use Case View.
4. Component View: This displays the User Interface of the system.
5. Database View: This contains the Entity-Relationship Diagram for the system database.

3.2 Architectural Design Patterns

ASP.NET Core framework was used in the implementation of the system. This follows the Model-View-Controller(MVC) design pattern. This design pattern separates the project into three distinct layers:

1. Model: Defines the data structures of the system and directly handles all logic and data within the system. A model class communicates exclusively with its controller.

2. View: A visual representation of a model. Typically in the form of a web page or a component of the web page. The view communicates exclusively with its controller.

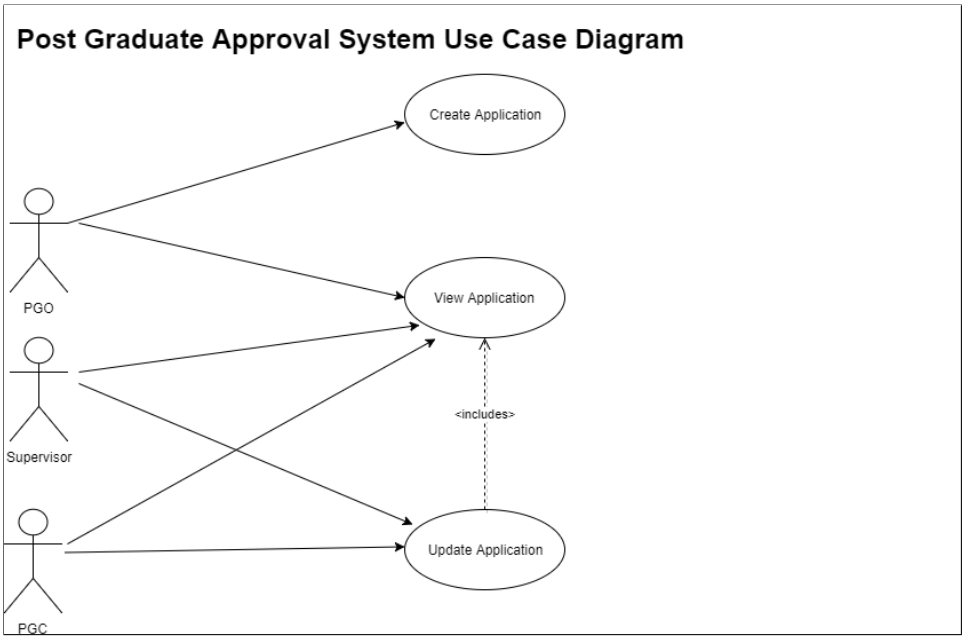
3. Controller: Accepts user input and maps it to instructions for models, views and potentially other controllers. Directly responsible for communication between components of the system.

This framework and design pattern was chosen to enhance modularity of the system. This allows for:

- Parallel development
- Efficient code reuse
- Faster bug detection and tracking.
- Greater unit testing coverage.

4 Architectural View Decomposition

4.1 Use-Case View



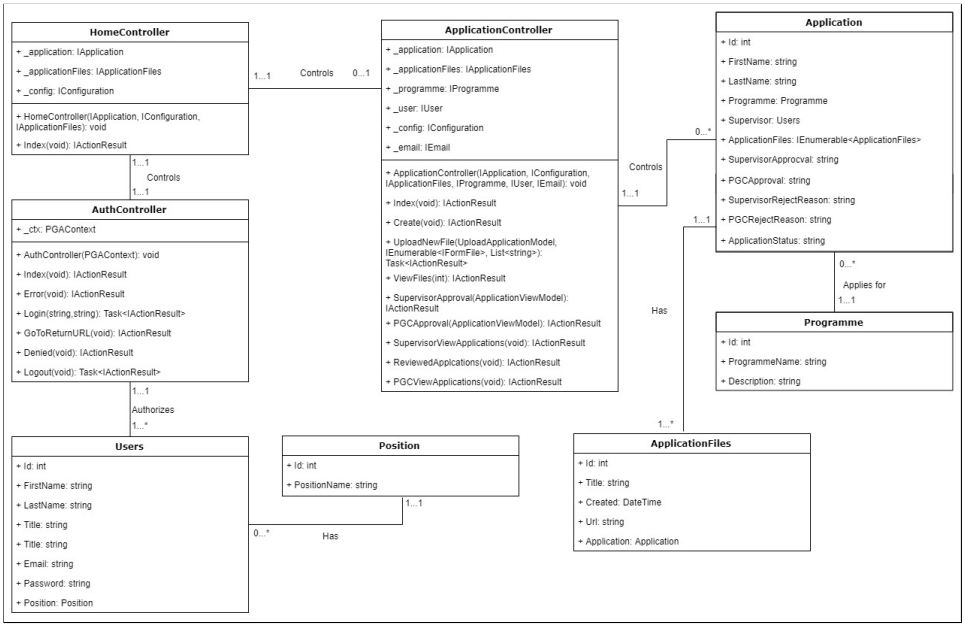
Use Case Diagram

4.1.1 Use Case Narrative

A PGO downloads an application from SIMS. The application with additional documents is then uploaded onto PGA System with summarized information entered into the system, creating an Application. Once uploaded, the PGO then verifies all documentation for the relevant Application. The PGO will additionally enter in Supervisors(s). An email will then be sent to Supervisor(s), notifying them to review the Application. The Supervisor will review the Application and make a decision on whether to Accept or Reject the Application with comments. The PGC will then be notfied that a Supervisor has made a decision and is pending their approval. The PGC notes the Supervisors decision, reviews the documentation and makes a PGC decision with comments.

4.2 Design View

4.2.1 Class Diagram

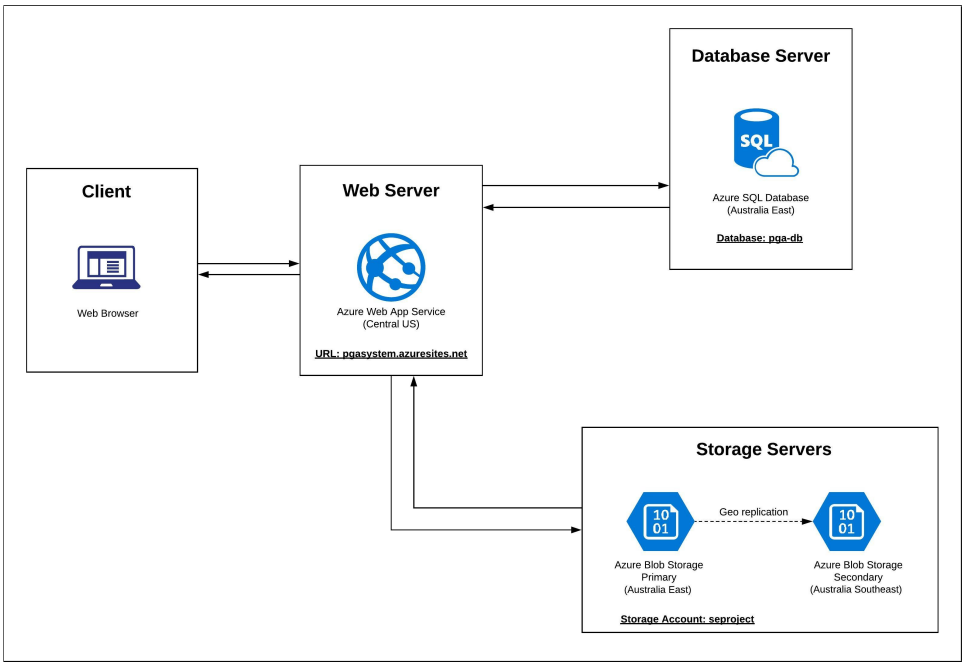


Class Diagram

Description The classes are described as follows:

- AuthController: Handles authentication and displays the login page.
- HomeController: The Controller for the home screens.
- ApplicationController: Handles all Application related operations i.e create, view, update.
- Application: The data structure that stores an application.
- Programme: Stores the application's course option.
- ApplicationFiles: Stores file metadata and the URL to the file on the Blob storage server.
- Users: Stores system user data.
- Position: Stores a user's position i.e PGO, Supervisor or PGC.

4.2.2 Hardware Architecture Diagram

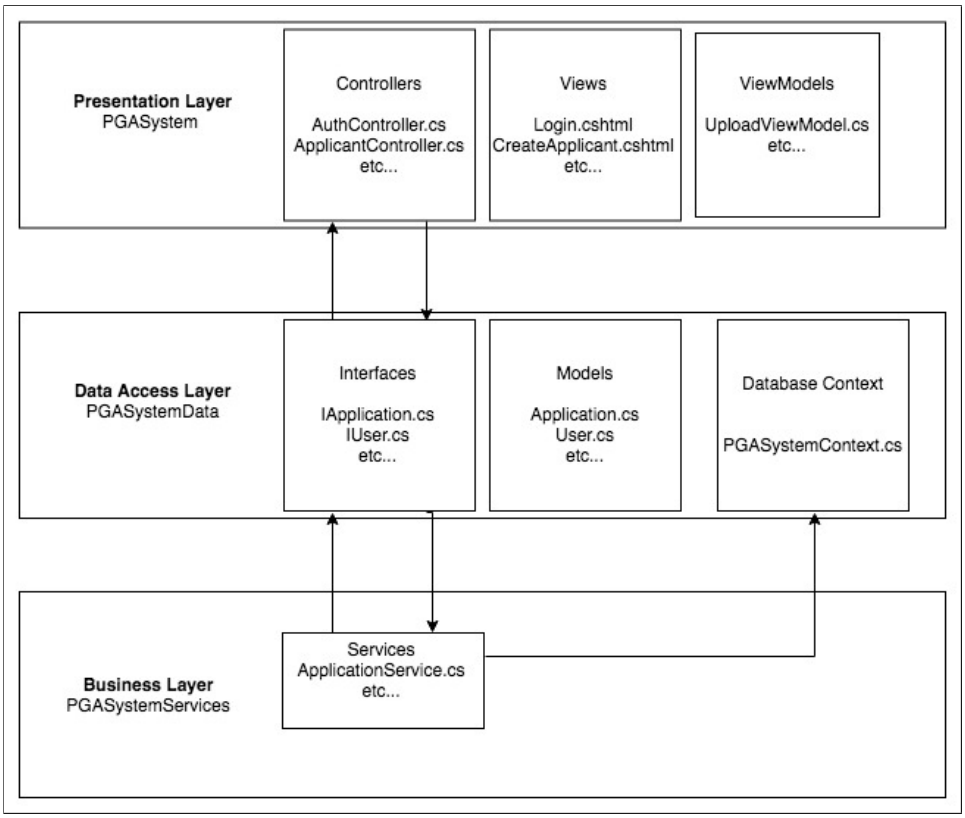


Hardware Architecture Diagram

Description The hardware architecture consists of:

- The clients web browser as the layer presenting the PGA System.
- An Azure web app service that hosts and executes the ASP.NET Core MVC web application (PGA System).
- An Azure Database server that contains a SQL database where application details are stored.
- An Azure Blob Storage server where application documents are stored.

4.2.3 Software Architecture Diagram



Software Architecture Diagram

Description The PGASystem is divided into 3 layers:

- Presentation Layer (PGASystem)
- Data Access Layer (PGASystemData)
- Business Layer (PGASystemServices)

The Presentation Layer contains:

- Views: Web pages in HTML/CSS
- Controllers: which handle user requests. The controllers utilize methods defined in the Interfaces of the Data Access Layer.

- View Models: which define what data is sent/received from respective web pages.

The Data Access Layer contains:

- Interfaces: which contain the definitions for methods that will be written in the Business layer
- Models: which define the structure of data that is stored in the SQL database and used by the application
- Database Context: which interacts directly with the SQL database.

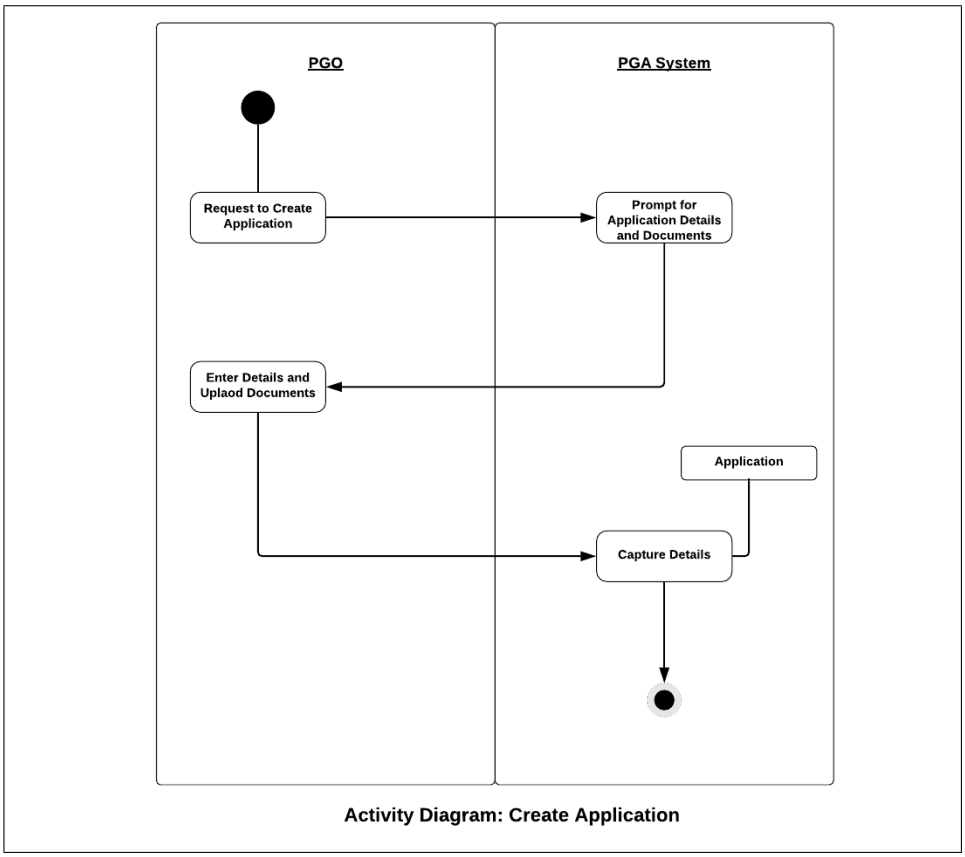
The Business Layer contains:

- Services in which all of the business logic is written, it utilizes the context of the Data Access Layer.

4.3 Process View

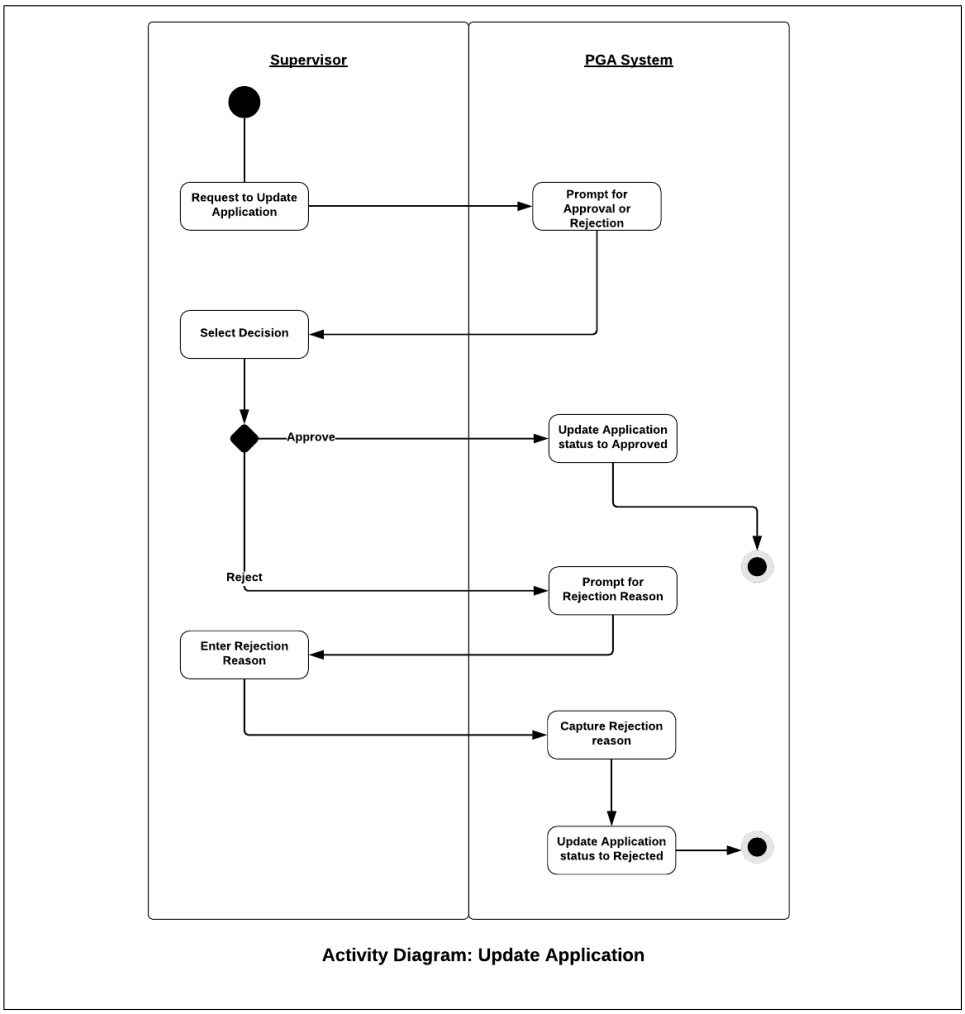
The following activity diagrams describe the dynamic interactions of the two most architecturally significant use cases, namely, Create Application and Update Application.

4.3.1 Create Application



Activity Diagram for the Create Application Use Case

4.3.2 Update Application

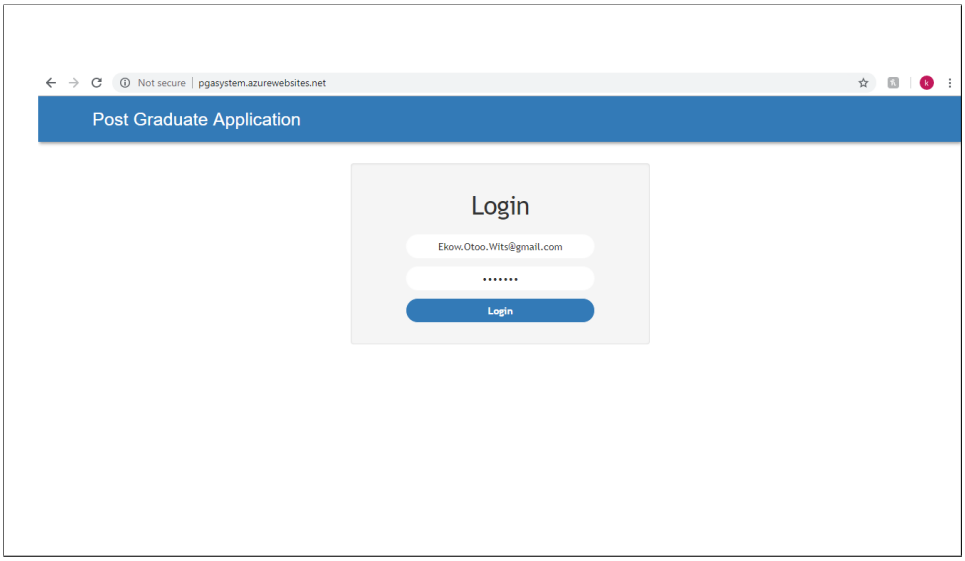


Activity Diagram for the Update Application Use Case

4.4 Component View

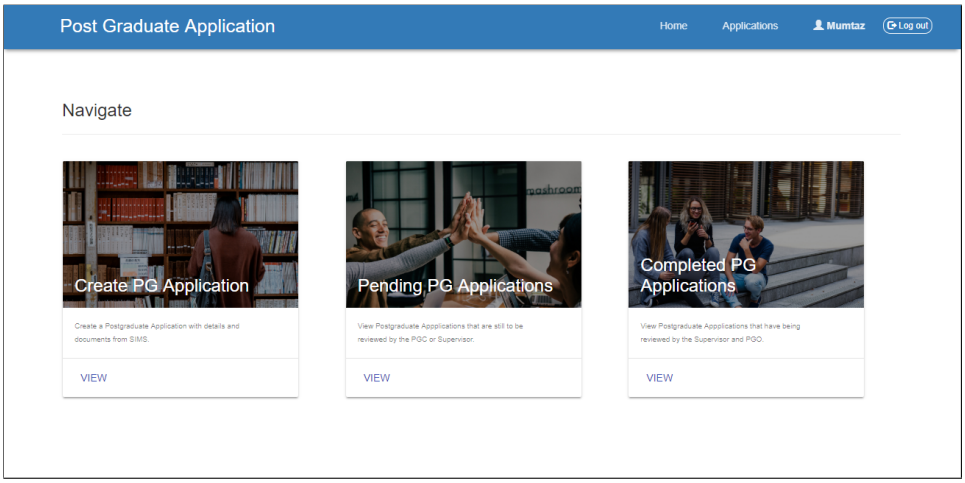
This section displays the User Interfaces for each user:

4.4.1 Common User Screens

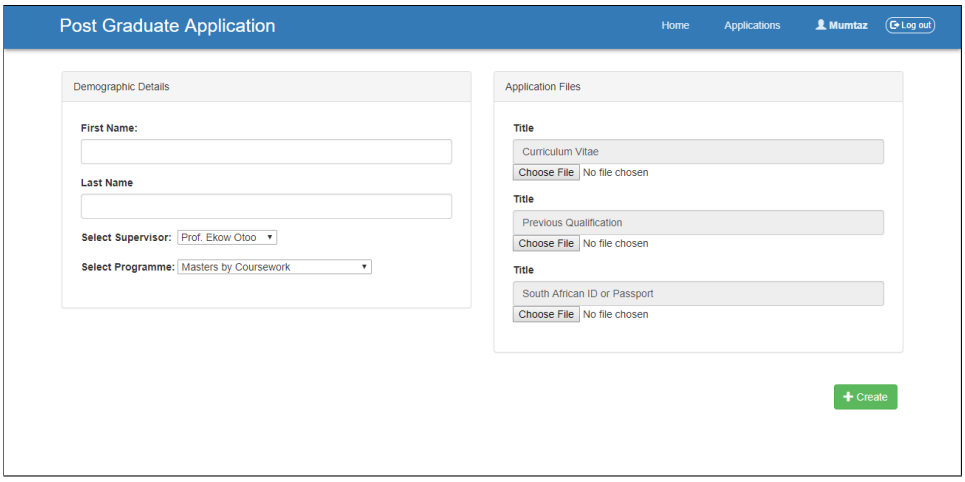


The Login Page

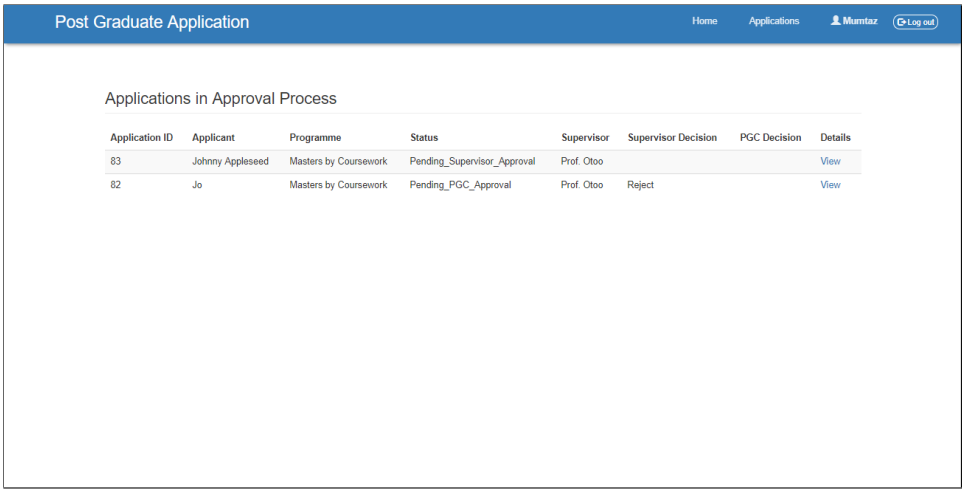
4.4.2 Post Graduate Officer



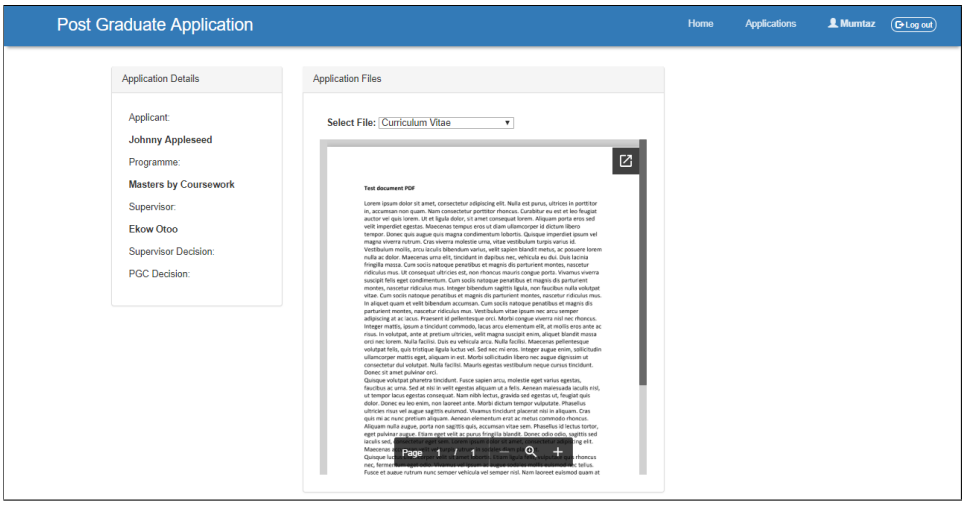
The PGO Logs in and lands on this screen.



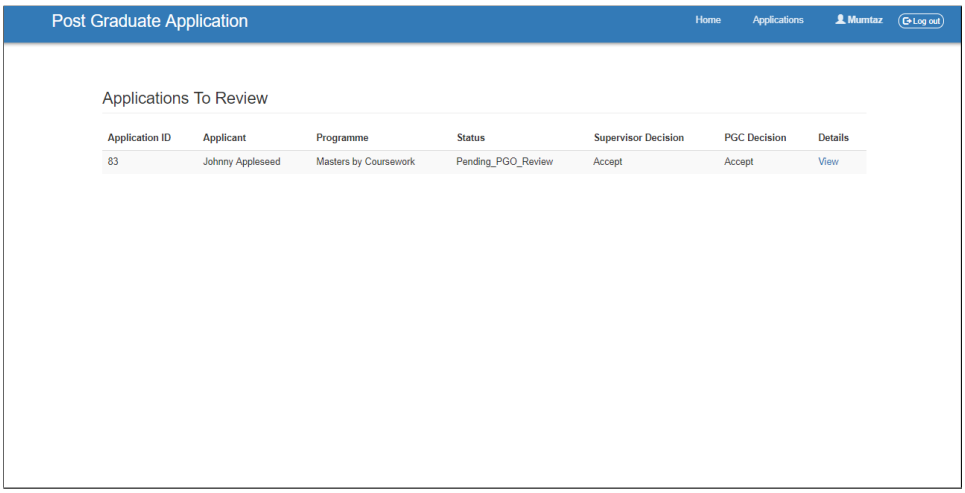
The PGO lands here when they click Create Application.



The PGO lands here when they click Pending Applications from the home page. Here a list of applications with pending decisions, from either a Supervisor or the PGC, is displayed.

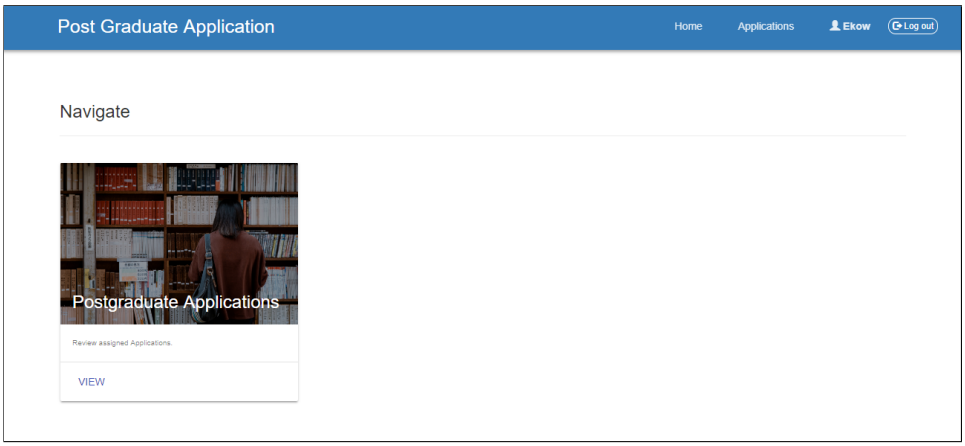


Once the PGO selects an application to view, this screen is displayed. It displays applicant and application information as well as a file viewer.

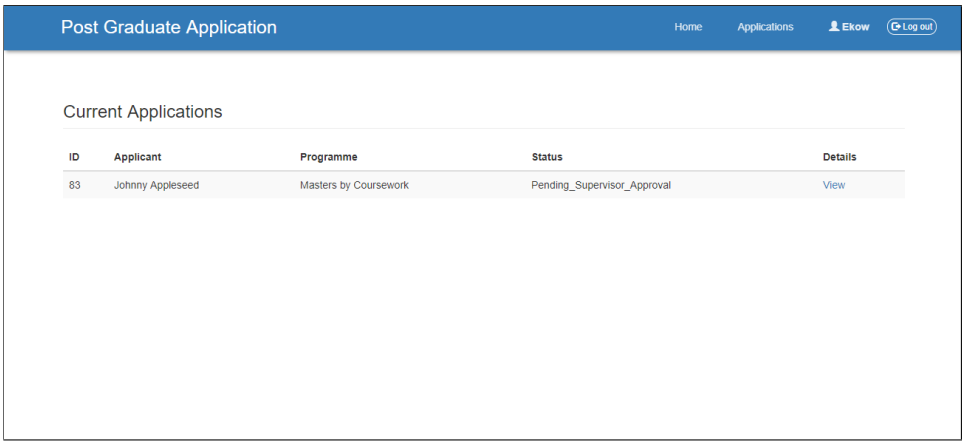


This page displays applications where the PGC has made a final decision.

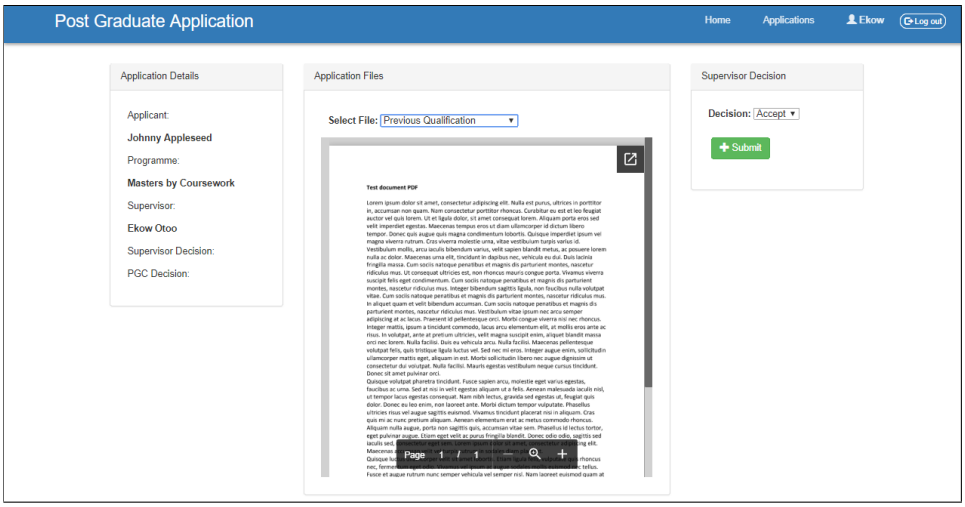
4.4.3 Supervisor



The Landing Page for a Supervisor

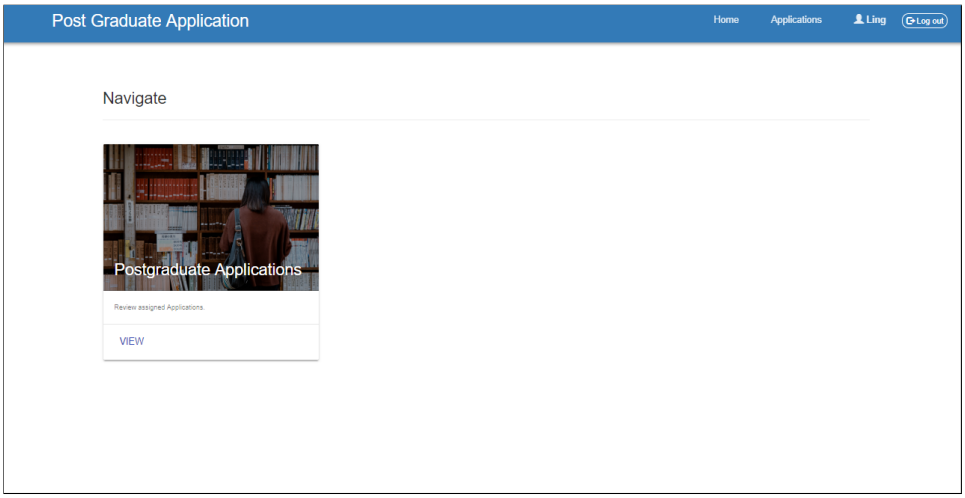


The Supervisor can see all pending applications that need their decision.

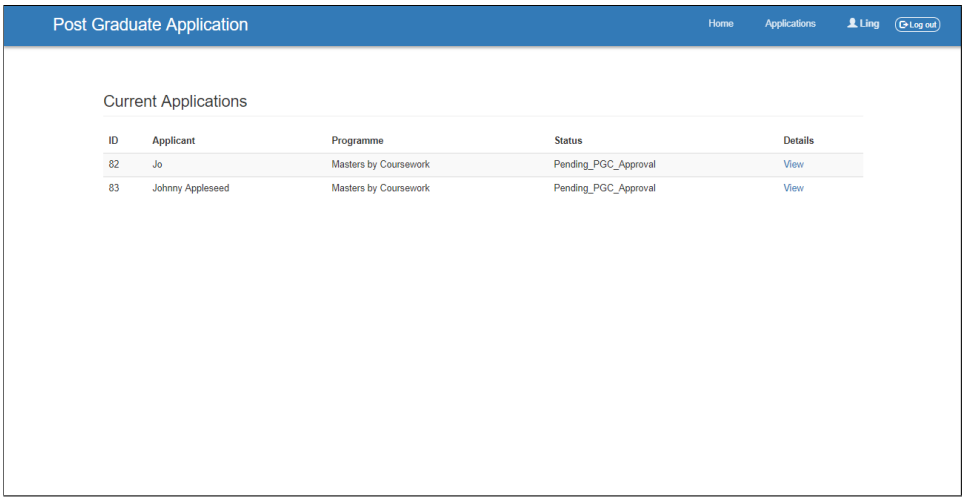


The view application page for the Supervisor. It displays applicant and application information, a file viewer and a decision picker.

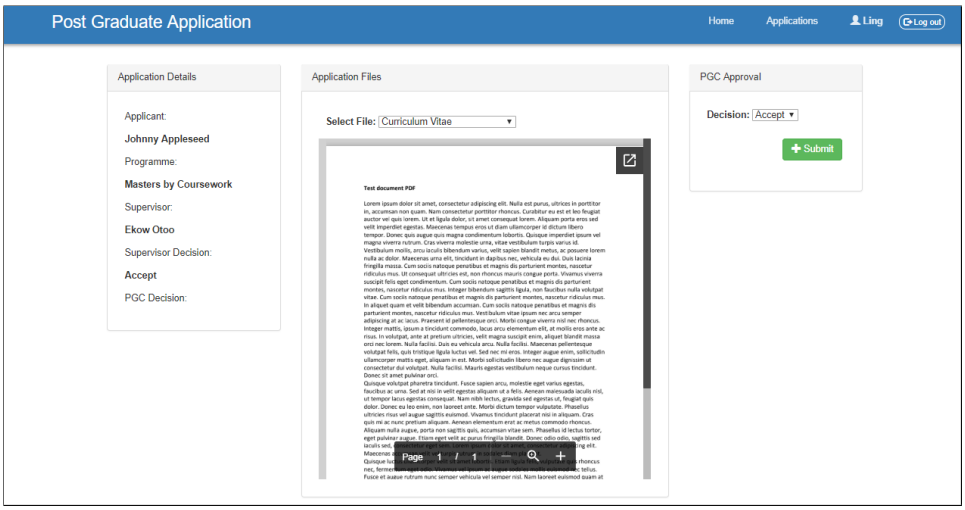
4.4.4 Post Graduate Coordinator



The Landing Page for the PGC



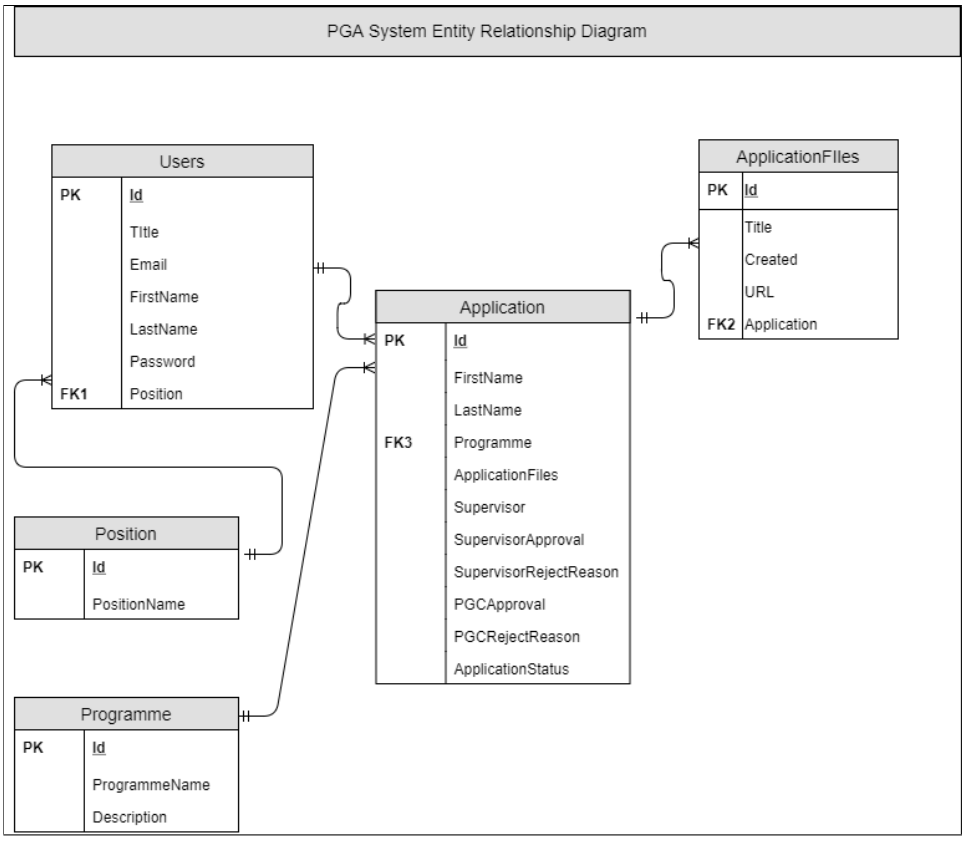
The PGC can see all pending applications that have been accepted by a Supervisor that need final approval.



The view application page for the PGC. It displays applicant and application information, a file viewer and a decision picker.

4.5 Database View

This section contains the Entity-Relationship Diagram for the System Database:



5 Performance and Unit Testing

5.1 Performance Test Details and Results

A stress test was conducted on the system. The Microsoft Azure performance testing framework was used. The stress test simulated 40 concurrent users, each sending an average of 5 requests per second for a full minute. The test results were as follows:

- Average response time: 0.08 seconds
- Average total requests per second: 194.67
- Total requests: 10927
- Successful requests: 10927
- Failed requests: 0

5.2 Unit Testing

A total of four test cases were implemented to show the feasibility of the testing framework. In future development of the system, it is expected that many more Unit tests will be implemented.

The following Unit Test Scenarios have been tested:

- Get Application:

  This scenario ensures that users can view applications seemlessly when required. Two test cases are used to achieve this. The first test checks if the system returns a valid applicant by passing an application Id. The second test, tests that system throws a valid exception with a description if an invalid (negative id or non-existent) applicationId is accessed. Both these tests ensure that the prototype functions correctly in every scenario that requires it to retrieve an application.

- Get Files:

  This scenario ensures that all corresponding applicant documents can be viewed by the user. Three test cases are used to achieve this. The first test tests if the correct number of files is returned for valid application Id. This is important to ensure that files from an applicant do not go missing. The second case tests if the correct file data is returned for valid application Id. This is important to ensure that documents from applicants do not get mixed up. Lastly, the third test tests if a valid exception is thrown if an application does not exist. All these tests in conjunction ensure that the prototype works correctly when required to retrieve application files.

- Add Application:

  This scenario tests whether a PGO can add an application to the system without any errors. One test case is used for this, which tests if the application has been added to the context. This is important as the test will fail if the application is not added and hence it ensures that applications do not go missing when added to the system.

- Get Supervisor:

  This scenario tests if a valid supervisor is returned when requested by passing a supervisor Id. This ensures that supervisors can be correctly allocated and notified when required. Two test cases are used to achieve this. The first tests if a valid supervisor is returned when

entering a supervisor id and the second tests if an error is thrown if an invalid supervisor Id is entered. This is important to ensure that supervisors are correctly identified by their Id in the system.