Table of Contents

[Table of Figures 1](#_Toc131081408)

[Introduction to project 2](#_Toc131081409)

[Methodology 2](#_Toc131081410)

[Project design 3](#_Toc131081411)

[Project requirements specification 3](#_Toc131081412)

[Functional Requirements 3](#_Toc131081413)

[Non-functional requirements 3](#_Toc131081414)

[On-screen requirements 4](#_Toc131081415)

[User Stores & Respective User Acceptance Tests 6](#_Toc131081416)

[Data dictionary 7](#_Toc131081417)

[UML diagrams 8](#_Toc131081418)

[Software Architectural Style for The System 10](#_Toc131081419)

# Table of Figures

[Figure 1 Login 4](#_Toc131080867)

[Figure 2 Signup 5](#_Toc131080868)

[Figure 3 Homepage 5](#_Toc131080869)

[Figure 4 Use case diagram. 8](#_Toc131080870)

[Figure 5 Sequence diagram 8](#_Toc131080871)

[Figure 6 Component diagram 9](#_Toc131080872)

[Figure 7 Activity diagram 9](#_Toc131080873)

# Introduction to project

In this project we will be designing and implementing a quiz platform on which users can create and answer quizzes, in the vein of Sporcle or Jetpunk. Users can share results of their quizzes with their friends and can create groups for competitions. Multiple types of questions should be possible, including simple question answer questions, image-based questions, matching questions, etc. Quizzes should be tagged by category and can be rated by users.

# Methodology

The methodology that will be used in this project includes repetition of design, code and implementation then testing and analysing the result. Further when repeating the phases again, new functionalities will be added to the project. This methodology is called agile methodology.

We will be using Scrum which is a subset of agile and through each sprint, our team will have five types of Scrum meetings.

**Sprint Planning Meeting**

The sprint-planning meeting is held on the first day of every sprint. The ScrumMaster, Product Owner, and Team are all in attendance.

**Daily Scrum**

Daily check in meetings to check progress.

**Sprint Review**

Main meeting to showcase the sprint.

**Sprint Retrospective**

Held after the sprint review and before sprint planning. The fundamental purpose of these meetings is to review what was done right or wrong during the sprint.

**Backlog refinement**

Reviewing backlog items and checking that they are appropriately prioritised.

# Project design

Firebase: Has been used as the database for the project.

JavaScript: Allows you to make web pages interactive.

GitHub: Version control repository

ReactJS: Programming Framework for Web UI

Vs code: Coding editor

CSS: Describing the presentation of Web pages, including colours, layout, and fonts.

Taiga: Track the progress of the project.

TravisCI: Supports the development process by automatically building and testing code changes, providing immediate feedback on the success of the change.

# Project requirements specification

By project requirements specifications we can analyse the tasks which going to be done by the system.

* System must store information about users, quizzes, questions, and results.
* System should do not allow unauthorized user to enter the system.
* System should keep and display the results of the users.
* System should send authentication to the users.

# Functional Requirements

* Users can create and answer quizzes.
* Users can see their result after submitting the quiz.
* Users can share results of their quizzes with their friends and can create groups for competitions.

# Non-functional requirements

**Reliability**

The application is bug free. The system runs for a long time without failure under predefined conditions.

**Ease of use**

The application should be user friendly and should require least effort to operate.

**Portability**

The application must be usable for most android phones and Windows users.

**Security**

To prevent unauthorized access to the Witsquiz app the system will require users to create accounts to gain access to the application. The registration process will require email address and their preferred password. This information will be stored in the database and will be use later to authenticate users and grant them access to the platform. The system requires user to create a strong password (**1 Upper case letter,1 symbol,1 lower case letter, and a minimum of 8 letters**) for the password to be acceptable. This is to ensure that a user’s password is not easy to guess, and someone else’s credentials cannot be used to gain access to the platform. The system will lock the user's account after 3 login failed attempts to protect the user's information. A user can choose to save password in order to grant access next time when using the app without entering details.

**Scalability**

The system should be able to handle more users.

# On-screen requirements

This is the login page for the users in the system. The users should to login in the system before used it. JavaScript validation was used to ensure that the login module is safe from the threat of attackers.

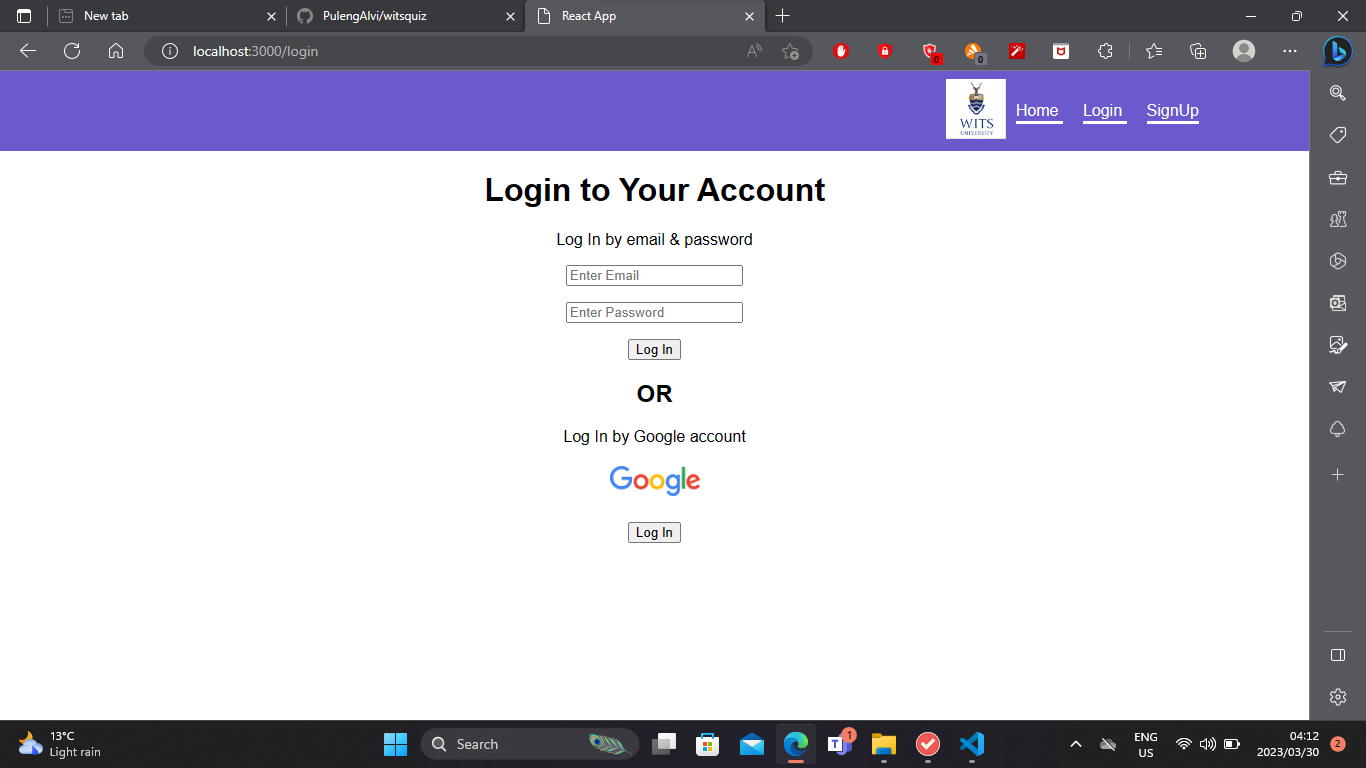


Figure Login

This is a signup page for users that have not already registered in the system. The system allows users to signup with using google account.

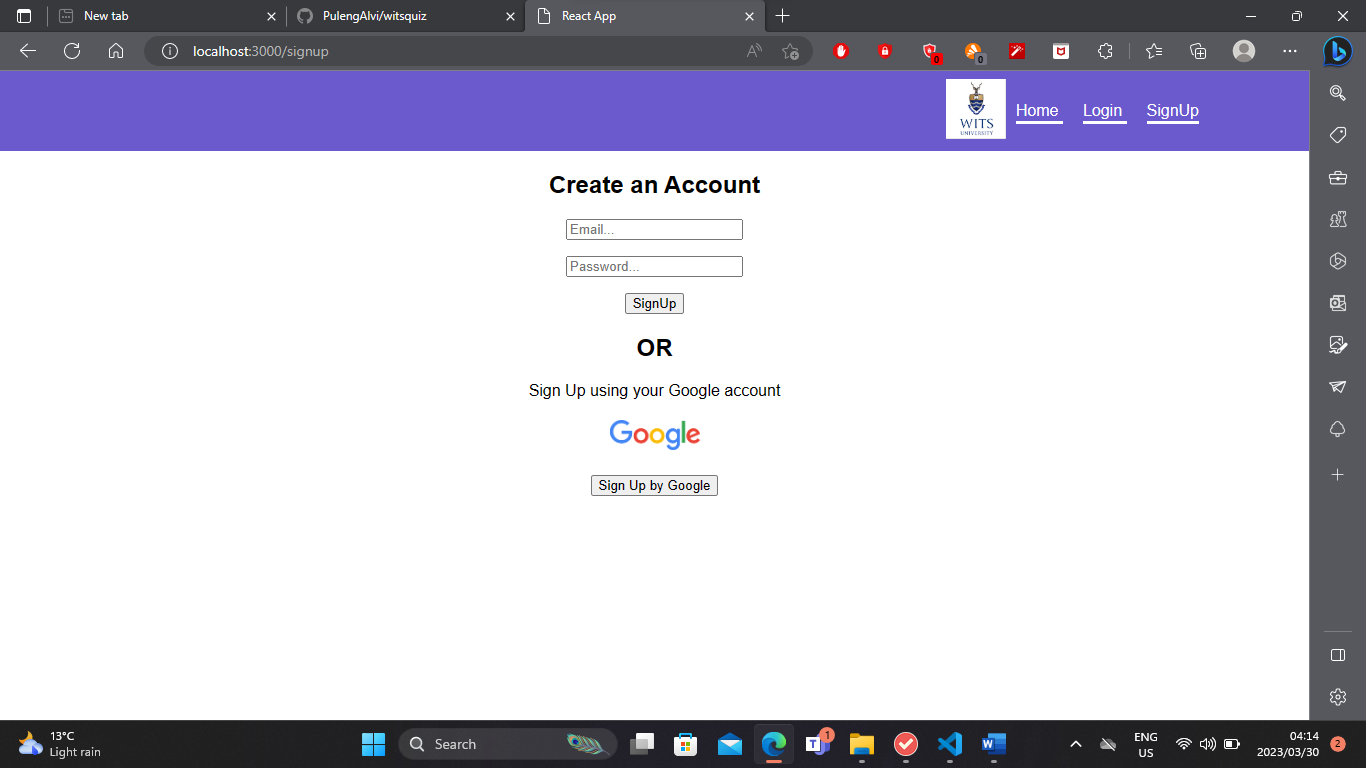


Figure Signup

This is the homepage.

Graphical user interface, text, application

Description automatically generated

Figure Homepage

# User Stores & Respective User Acceptance Tests

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | User stories | User Acceptance Tests | Functional requirements |
| Sign Up | As a new user, I want to be able to see the sign-up page and register an account, so that I can start using the application. | **Given** I am a new user without an account, **When** I fill in my correct registration details and press the sign-up button, T**hen** my account should be created, and I can start using the application. | The system must allow user to provide their details to register their account.  The system must be able to store the user’s account details in the users’ database. |
| Login | As a user who already has an account, I want to be directed to a login page and login, so that I can continue from where I left of without having to create an account again. | **Given** I am currently logged out from the application, **When** I fill in a correct username/email and password and press the login button, **Then** I should get logged into the application. | The system must check if there entered username and password are present in the database and belong to the same person. |
| Forgot password | As a user who already has an account, I want to be able to reset my password, so that I can login to the application. | **Given** that I am a registered user, **When** I press the forgot password hyperlink, I should be directed to forgot password page, **Then** I can restart my password. | The system must check if the entered email exists in the database.  If the email exists, then it must allow the user to reset password. |
| Homepage | As a logged in user, I want to be able to see the homepage,  so that I can navigate to different places of the application. | **G**iven that I am a registered user, When I login, Then I should be directed to the homepage. | The system must display the homepage after the user has logged in. |

# Data dictionary

# UML diagrams

Use case diagram

Use cases describe how the system-to-be from a user’s perspective. It used to show the functions to be supported by the system.

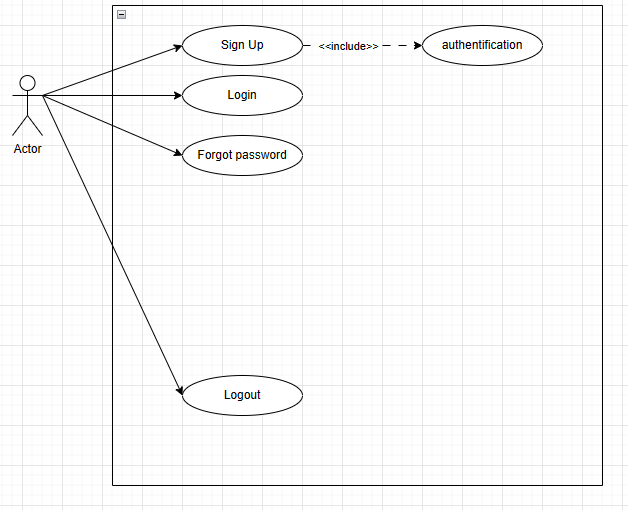


Figure Use case diagram.

Sequence diagram

To show the interactions between objects in the sequential order that those interactions occur.

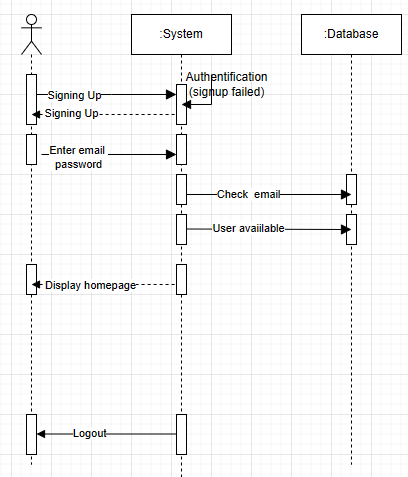


Figure Sequence diagram

Component diagram

To visualize the organization of system components and the dependency relationships between them.

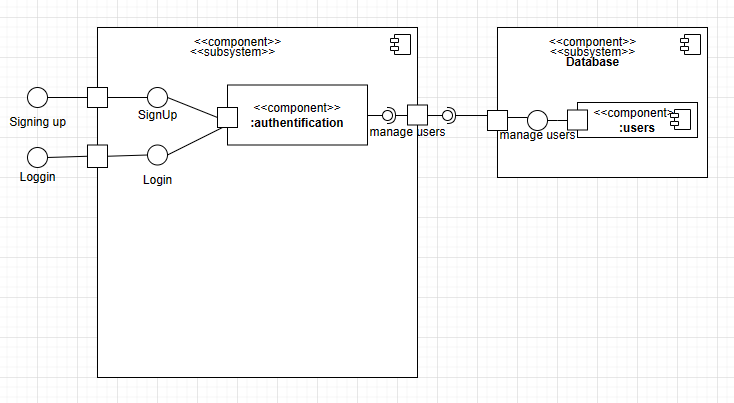


Figure Component diagram

Activity diagram

Shows the flow from one activity to another in a system.

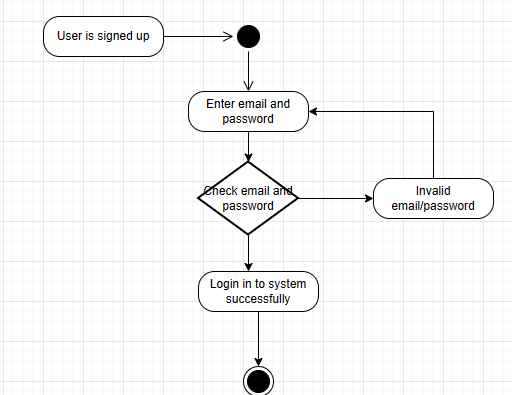


Figure Activity diagram

# Software Architectural Style for The System

A multi-tier (3-tier) architectural style will be used for the software-to-be. The 3-tier architectural style organises software components into 3 logical tiers. The 3 logical tiers will be:

1. The **presentation tier** – the user interface (UI)

2. The **application tier** – this is where the data is processed

3. The **data tier** – the database management system (DBMS) which is where data processed by the application is managed and stored.

Why this architectural style chosen?

1. Faster development: Each tier can be developed simultaneously; this will allow the team to meet the client’s time deadlines for a functional final product.

2. Improved Reliability: This style will satisfy the client’s needs for a functional and reliable system. The performance of a tier is less likely to be affected by the failure of a separate tier.

3. Scalability: Tiers can be scaled independently