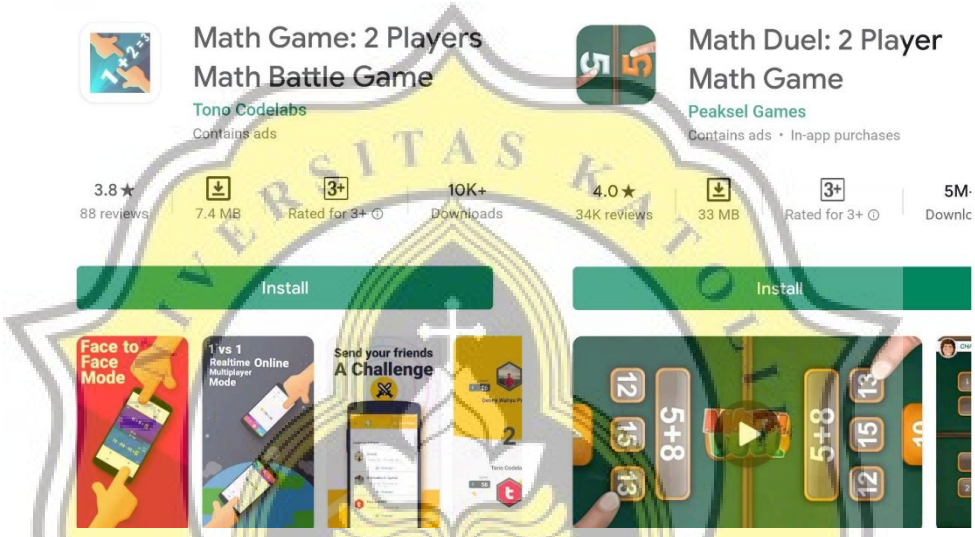
BAB IV PERANCANGAN GAME DAN PENGUJIAN

4.1 Perancangan Game

Perancangan *game* adalah langkah pertama dalam pembuatan *game*. Untuk merancang *game* Ayo Hitung dilakukan observasi pada *game* yang sejenis dengan Ayo Hitung.



Gambar 4.1 Observasi Game Seperti Ayo Hitung

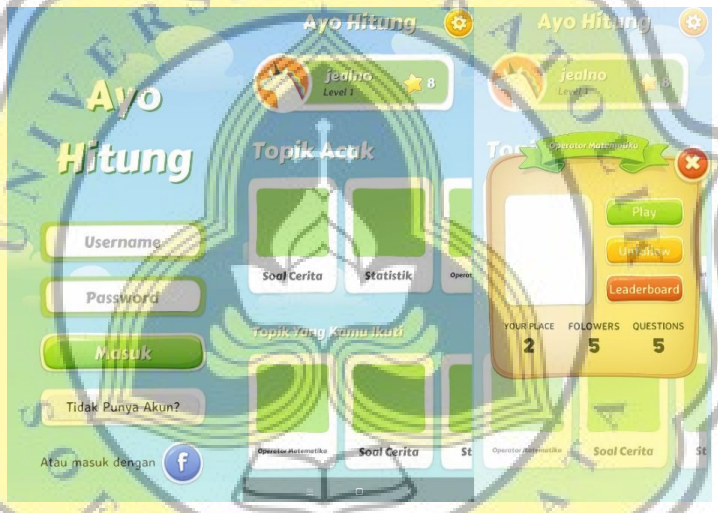
Setelah ditelusuri ditemukan 2 *game* sejenis dengan *game* Ayo Hitung yang angkat di dalam penelitian ini. Rancangan *game* Ayo Hitung akan dibuat berbasis 2 dimensi. Seperti yang sudah ditulis diatas, *game* Ayo Hitung berbasis daring atau biasa yang disebut dengan *multiplayer* yang mengharuskan dua pengguna bertanding dan menebak jawaban yang benar dengan batas waktu yang telah ditentukan lalu yang memiliki skor paling tinggi maka pengguna tersebut yang menang. Pada halaman pertama Ayo Hitung pengguna diminta untuk *login* dengan akunnya. Setelah melakukan proses *login*, pengguna akan diminta untuk memilih 3 kategori yang disukainya. Berikutnya pengguna akan dipindahkan ke halaman utama *game* Ayo Hitung yaitu memilih topik dan melakukan *searching* pengguna lainnya agar dapat bermain. Di dalam halaman

utama juga terdapat sistem *ranking* yang berisi pengguna yang telah memainkan kategori tersebut.

4.2 Konsep, Desain, Gameplay dan Alur Game

4.2.1 Desain, Gameplay dan Alur Game

Saat pengguna *login* ke dalam *game* Ayo Hitung akan muncul beberapa menu yaitu profile, topik – topik yang telah dibuat, dan roda gerigi. Ketika pengguna memilik salah satu topik akan muncul menu baru yaitu play, follow, dan leaderboard.



Gambar 4.2 Halaman Login, Utama, Topik

A. Play

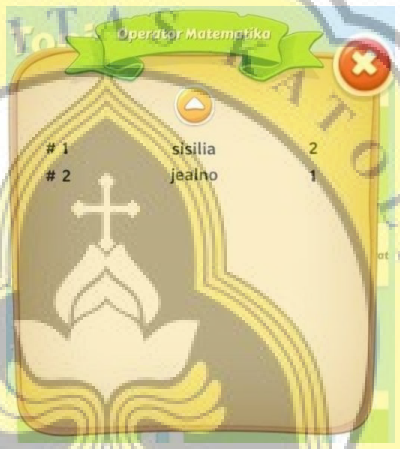
Saat menu ini dipilih oleh pengguna, maka *game* otomatis mencari pengguna lainnya yang sedang mencari. Ketika terhubung maka kedua pengguna akan bertanding dengan 5 pertanyaan. Yang mendapatkan skor lebih tinggi, itulah pemenangnya.

B. Follow/Unfollow

Saat menu ini dipilih, maka sistem akan menambahkan/mengeluarkan pengguna ke dalam daftar pengguna lainnya yang mengikuti topik tersebut.

C. Leaderboard

Begitu menu ini dipilih maka pengguna akan dapat melihat ranking berdasarkan banyaknya pertandingan yang dimenangkan.



Gambar 4.3 Halaman Leaderboard

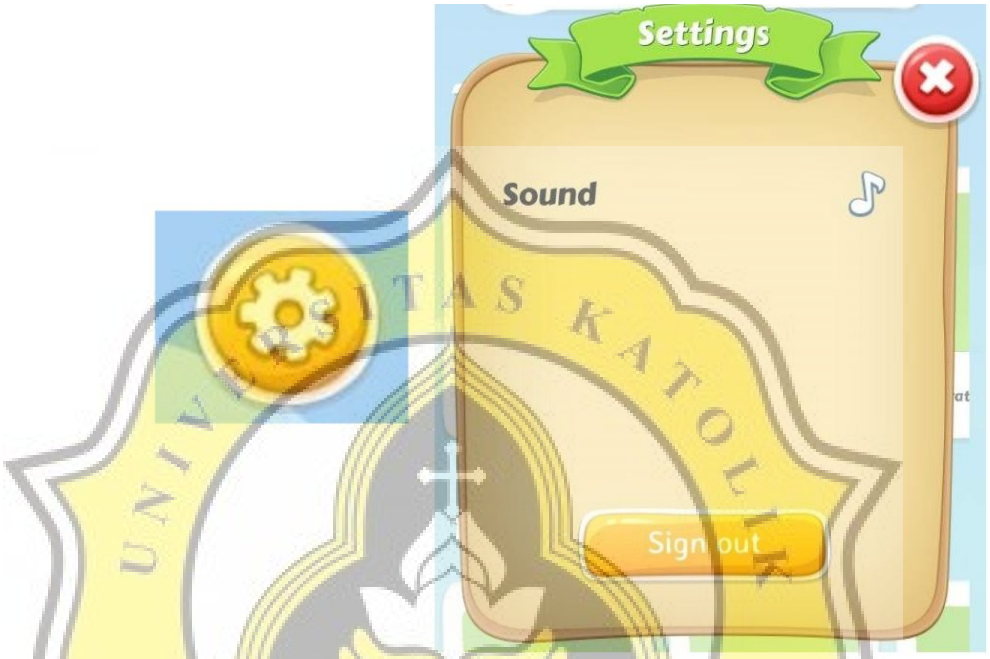
D. Profile

Terdapat info nama pengguna dan jumlah bintang yang dimenangkan dari pertandingan.

Gambar 4.4 Profile

E. Roda Gerigi (Settings)

Di dalam menu ini terdapat menu untuk mematikan suara dan opsi untuk *sign-out* atau keluar dari *game* Ayo Hitung.



Gambar 4.5 Settings

4.2.2 Konsep Game

Ayo Hitung merupakan *game* pembelajaran berbasis daring tentang mata pelajaran matematika untuk SD Kelas 6 di kota Cirebon yang dijalankan pada sistem operasi *android*. *Game* Ayo Hitung diciptakan guna merealisasikan media pembelajaran yang makin memukau dan mengasyikkan untuk penggunanya belajar matematika dengan bertanding dengan pengguna lainnya. *Game* Ayo Hitung memakai ide *multiplayer-game* yang mana dua pengguna saling bertanding untuk memenangkan pertandingan. Agar membuat *game* lebih bervariasi, maka ditambahkan topik yaitu operator matematika, soal cerita, statistik, dan FPB KPK.

4.3 Pembuatan dan Pemrograman Game

Pembuatan *game* "Ayo Hitung" menggunakan kombinasi 3 aplikasi yaitu Unity 3D, NodeJS dan MongoDB. Unity 3D adalah salah satu *game engine* untuk membuat *game* dengan bahasa pemrograman C#. NodeJS digunakan untuk membuat server dan mongoDB digunakan untuk melakukan penyimpanan data. Dalam pembuatan aplikasi ini menggunakan konsep OOP untuk mempermudah pengelompokkan fungsi – fungsi yang akan dibuat. Berikut ini adalah kumpulan *script* vital yang diperlukan agar *game* "Ayo Hitung" dapat bekerja dengan normal, sebagai berikut:

4.3.1 NodeJS dan MongoDB

1) Inisiasi Database

Script ini berfungsi untuk menginisiasi database ke mongoDB. MongoClient digunakan untuk melakukan koneksi ke databasenya.

```
var mongoDBHost = "@cluster0-shard-00-00-
nylok.mongodb.net:27017,cluster0-shard-00-01-
nylok.mongodb.net:27017,cluster0-shard-00-02-nylok.mongodb.net:27017";
var mongoDBName = "ayohitung?authSource=admin&replicaSet=Cluster0-
shard-0&ssl=true";
var mongoUserName = "jealno";
var mongoPass = "akunov94";

MongoClient.connect('mongodb://' + mongoUserName + ':' + mongoPass +
mongoDBHost + '/' + mongoDBName, function (err, db) {
  if (err)
    throw err;

  global.DB = db;

  updateV2.Run();
});
```

Script 4.1 Script Inisiasi Database

2) Penambahan Topik

Script ini berfungsi untuk menambahkan topik di dalam *game*. Cara kerjanya adalah membuka *collection* bernama topics lalu

menambahkan filter id (auto_increment), topic name, topic image, dan follower number dengan default value 0.

```
DB.collection("topics").findOne(
                {
                   id: parseInt(req.body.topicId)
                }, function(err, topic)
                {
                   if (err)
                      throw err;

                   if (topic)
                   {
                      DB.collection("questions").insert(
                      {
                         id: nextID,
                         body: req.body.body,
                         imageUrl: req.body.imageUrl,
                         topicId: topic.id,
                         answers: answers,
                         correctAnswer: req.body.correctAnswer
                      }, function(err, result)
                      {
                         res.send(
                         {
                            successStr: true
                         });
                      });
                   }
                   else
                   {
                      errorResponse(res, 1002);
                   }
                });

app.post('/add-topic', function(req, res)
   {
      if (Security.RequestIsValid(req, res))
      {
         if (req.body.title.length > 0 && req.body.imageUrl.length > 0)
         {
            DB.collection("topics").find().sort(
            {
               id: -1
            }).limit(1).toArray(function(err, maxTopic)
            {
               if (err)
                  throw err;
```

```
var nextID = 1;
if (maxTopic.length > 0)
   nextID = maxTopic[0].id + 1;

DB.collection("topics").insert(
{
   id: nextID,
   name: req.body.title,
   imageUrl: req.body.imageUrl,
   followersNumber: 0
}, function(err)
{
   if (err)
      throw err;

   res.send(
   {
      successStr: true
   });

   });
  });
 }
 else
 {
   errorResponse(res, 1002);
 }
 }
});
```

Script 4.2 Script Penambahan Topik

3) Penambahan Soal

Script ini berfungsi untuk menambahkan soal di dalam *game*. Cara kerjanya adalah menyimpan jawaban benar di array answers dan membuka *collection* topics untuk mengambil id topik guna mengkategorikan soal sesuai dengan topik. Setelah itu soal ditambahkan ke *collection* bernama questions yang berisi id (auto_increment), body (soal berbentuk teks), imageUrl (soal berbentuk gambar), topicId (yang diambil dari *collection* topics), answers (jawaban benar yang sebelumnya ditaruh pada array).

```
app.post('/add-question', function(req, res)
```

```
{
   if (Security.RequestIsValid(req, res))
   {
      if (validateQuestion(req.body))
      {
         DB.collection("questions").find().sort(
         {
            id: -1
         }).limit(1).toArray(function(err, maxQuestion)
         {
            if (err)
               throw err;

            var nextID = 1;
            if (maxQuestion.length > 0)
               nextID = maxQuestion[0].id + 1;

            var answers = [];
            var answersTexts = req.body.answersTexts.split(",");
            var answersImages = req.body.answersImages.split(",");

            for (var i = 0; i < answersTexts.length || i < answersImages.length;
i++)
            {
               var a = {
                  id: i
               };

               if (answersTexts.length > i)
                  a.body = answersTexts[i];

               if (answersImages.length > i)
                  a.imageUrl = answersImages[i];

               answers.push(a);
            }

            DB.collection("topics").findOne(
            {
               id: parseInt(req.body.topicId)
            }, function(err, topic)
            {
               if (err)
                  throw err;

               if (topic)
               {
                  DB.collection("questions").insert(
                  {
                     id: nextID,
                     body: req.body.body,
```

```
                imageUrl: req.body.imageUrl,
                topicId: topic.id,
                answers: answers,
                correctAnswer: req.body.correctAnswer
            }, function(err, result)
            {
                res.send(
                {
                    successStr: true
                });
            });
        }
        else
        {
            errorResponse(res, 1002);
        }
    });
  });
 }
 else
 {
    errorResponse(res, 1002);
 }
 }
});
```

Script 4.3 Script Penambahan Soal

4) Penghapusan Soal dan Topik

Script ini berfungsi untuk menghapus soal & topik yang sudah terbuat. Cara kerjanya adalah membuka *collection* questions dan topics lalu menghapus soal dan topik berdasarkan id yang ada di *collection* questions dan topics.

```
app.post('/remove-question', function(req, res)
 {
    if (Security.RequestIsValid(req, res))
    {
        DB.collection("questions").remove(
        {
            id: parseInt(req.body.questionId)
        }, function(err)
        {
            if (err)
                throw err;
```

```
            res.send(
            {
               removedItemId: req.body.questionId,
               successStr: true
            });
         });
      }
   });
app.post('/remove-topic', function(req, res)
   {
      if (Security.RequestIsValid(req, res))
      {
         DB.collection("topics").remove(
         {
            id: parseInt(req.body.topicId)
         }, function(err)
         {
            if (err)
               throw err;

            res.send(
            {
               removedItemId: req.body.topicId,
               successStr: true
            });
         });
      }
   });
```

Script 4.4 Script Penghapusan Soal dan Topik

5) Pengeditan Topik

Script ini berfungsi untuk mengedit topik yang sudah terbuat. Sebenarnya script ini sama dengan menambah topik, bedanya adalah mengedit topik berdasarkan id terpilih yang ada di *collection* topics.

```
app.post('/edit-topic', function(req, res)
   {
      if (Security.RequestIsValid(req, res))
      {
         DB.collection("topics").findOne(
         {
            id: parseInt(req.body.topicId)
         }, function(err, topic)
```

```
{
   if (err)
      throw err;

   if (topic)
   {
      DB.collection("topics").update(
      {
         id: parseInt(topic.id)
      },
      {
         $set:
         {
            name: req.body.name,
            imageUrl: req.body.imageUrl
         }
      }, function(err)
      {
         if (err)
            throw err;

         res.send(
         {
            successStr: true
         });
      });
   }
   else
   {
      errorResponse(res, 1003);
   }
});
   }
});
if (topic)
      {
      DB.collection("topics").update(
      {
         id: parseInt(topic.id)
      },
      {
         $set:
         {
            name: req.body.name,
            imageUrl: req.body.imageUrl
         }
      }, function(err)
      {
         if (err)
            throw err;
```

```
            res.send(
            {
               successStr: true
            });
         });
      }
      else
      {
         errorResponse(res, 1003);
      }
```

Script 4.5 Script Pengeditan Topik

6) Pengeditan Soal

Script ini berfungsi untuk mengedit soal yang sudah terbuat. Sebenarnya script ini sama dengan menambah soal, bedanya adalah mengedit soal berdasarkan id terpilih yang ada di *collection* questions.

```
app.post('/edit-question', function(req, res)
   {
      if (Security.RequestIsValid(req, res))
      {
         if (validateQuestion(req.body))
         {
            DB.collection("questions").findOne(
            {
               id: parseInt(req.body.questionId)
            }, function(err, question)
            {
               if (err)
                  throw err;

               if (question)
               {
                  var answers = [];
                  var answersTexts = req.body.answersTexts.split(",");
                  var answersImages = req.body.answersImages.split(",");

                  for (var i = 0; i < answersTexts.length || i <
answersImages.length; i++)
                  {
                     var a = {
                        id: i
```

```
    };

    if (answersTexts.length > i)
      a.body = answersTexts[i];

    if (answersImages.length > i)
      a.imageUrl = answersImages[i];

    answers.push(a);
  }

  DB.collection("topics").findOne(
  {
    id: parseInt(req.body.topicId)
  }, function(err, topic)
  {
    if (err)
      throw err;

    if (topic)
    {
      DB.collection("questions").update(
      {
        id: parseInt(question.id)
      },
      {
        $set:
        {
          body: req.body.body,
          imageUrl: req.body.imageUrl,
          topicId: topic.id,
          answers: answers,
          correctAnswer: req.body.correctAnswer
        }
      }, function(err)
      {
        if (err)
          throw err;

        res.send(
        {
          successStr: true
        });

      });
    }
    else
    {
      errorResponse(res, 1003);
    }
  });
```

```
                }
                else
                {
                    errorResponse(res, 1004);
                }
            });
        }
        else
        {
            errorResponse(res, 1002);
        }
    }
});

DB.collection("topics").findOne(
        {
            id: parseInt(req.body.topicId)
        }, function(err, topic)
        {
            if (err)
                throw err;

            if (topic)
            {
                DB.collection("questions").update(
                {
                    id: parseInt(question.id)
                },
                {
                    $set:
                    {
                        body: req.body.body,
                        imageUrl: req.body.imageUrl,
                        topicId: topic.id,
                        answers: answers,
                        correctAnswer: req.body.correctAnswer
                    }
                }, function(err)
                {
                    if (err)
                        throw err;

                    res.send(
                    {
                        successStr: true
                    });

                });
            }
            else
            {
```

```
            errorResponse(res, 1003);
        }
    });
```

Script 4.6 Script Pengeditan Soal

7) Mengaktifkan Server dan Status Server

Script ini berfungsi untuk mengaktifkan server agar pengguna dapat mengakses ke dalam *game*. Server diatur pada port 3000 agar tidak berpindah – pindah ke port lainnya saat dimulai ulang.

```
app.get("/", function (req, res) {
    res.send("Server Started, started on port: " + serverPort);
});

adminModule.Initialize(app);

playerModule.Initialize(app);

var serverPort = process.env.PORT || 3000;

app.listen(serverPort, function () {
    console.log('Server app is started [PORT ' + serverPort + ']');
});
```

Script 4.7 Script Mengaktifkan Server

8) Sign In

Script ini berfungsi untuk masuk ke dalam *game* dengan akun yang sudah dibuat sebelumnya. Cara kerjanya membandingkan inputan pengguna dengan database. Variable token ke depannya akan digunakan untuk berbagai macam hal seperti melakukan pencarian pengguna lainnya untuk bertanding.

```
app.post('/signin', function(req, res)
    {
        if (Security.RequestIsValid(req, res))
        {
            var requestedUserName = req.body.username;
            var requestedPasswordHash = req.body.password_hash;

            var player = new Player();
```

```
        player.SignIn(requestedUserName, requestedPasswordHash,
function(success, errorCode)
        {
           if (success)
           {
              var token = player.GenerateNewToken();

              player.Push(function()
              {
                 var response = utils.GetEmptyResponse();

                 response.token = token;

                 res.send(response);
              });
           }
           else
           {
              errorResponse(res, errorCode);
           }
        });
     }
  });
```

Script 4.8 Script Sign In

9) Sign Up

Script ini berfungsi untuk membuat akun baru untuk masuk ke dalam *game*. Script ini juga mengecek apakah sudah ada akun dengan username yang sama. Jika sudah, maka akan memunculkan error. Jika tidak, maka akun yang sudah diinput akan terbuat.

```
app.post('/signup', function(req, res)
   {
      if (Security.RequestIsValid(req, res))
      {
         var username = req.body.username;
         var passwordHash = req.body.password_hash;
         var email = req.body.email;

         DB.collection('players').findOne(
         {
            username: username
         }, function(err, existedPlayer)
         {
```

```
if (err)
   throw err;

if (existedPlayer)
{
   errorResponse(res, 3);
}
else
{
   var player = new Player();

   var token = player.GenerateNewToken();

   player.SetUsername(username);
   player.SetPasswordHash(passwordHash);
   player.SetEmail(email);

   player.Push(function()
   {
      var response = utils.GetEmptyResponse();

      response.token = token;

      res.send(response);
   });
  }
 });
}
});
```

Script 4.9 Script Sign Up

10) Menampilkan Rincian Data *Game*

Script ini berfungsi untuk menampilkan data topik, jumlah pertanyaan dalam 1 topik dan leaderboard yang ada pada database. Script ini berjalan saat pengguna berhasil masuk ke dalam *game* tepatnya saat pengguna masuk ke halaman utama. Cara kerjanya adalah berpatokan pada *collections* topics lalu menampilkan data seperti nama topik, jumlah pertanyaan dalam 1 topik serta mengkorelasikan *collections* topics dan leaderboard berdasarkan id topic guna menampilkan list leaderboard.

```
app.post('/get-general-data', function(req, res)
```

```
{
    if (Security.RequestIsValid(req, res))
    {
        DB.collection("topics").find().toArray(function(err, topics)
        {
            if (err)
                throw err;

            var topicsPublic = [];

            for (var i = 0; i < topics.length; i++)
            {
                var topic = new Topic();

                topic.SetDocument(topics[i]);

                topicsPublic.push(topic.GetPublicData())
            }

            var response = utils.GetEmptyResponse();

            response.data = {
                topics: topicsPublic
            };

            var appSettings = new AppSettings();

            appSettings.Pull(function()
            {
                response.data.settings = appSettings.GetPublicData();
                res.send(response);
            });
        });
    }
});

app.post('/get-questions-count-by-topic', function(req, res)
{
    if (Security.RequestIsValid(req, res))
    {
        var topic = new Topic();

        topic.SetID(req.body.topic);

        topic.GetQuestionsCount(function(count)
        {
            var response = utils.GetEmptyResponse();

            response.questionsNumber = count;
            response.topicID = topic.GetID();
```

```
            res.send(response);
        });
    }
});

app.post('/get-leaderboard-by-topic', function(req, res)
{
    if (Security.RequestIsValid(req, res))
    {
        var topic = new Topic();

        topic.SetID(req.body.topic);

        topic.GetLeaderboard(function(leaderboard)
        {
            var response = utils.GetEmptyResponse();

            response.leaderboard = leaderboard;
            response.topic = parseInt(topic.GetID());

            res.send(response);
        });
    }
});
```

Script 4.10 Script Menampilkan Rincian Data

11) Follow dan Unfollow Topik

Script ini berfungsi untuk mengikuti maupun tidak mengikuti topik. Cara kerjanya adalah mengkorelasikan *collections* topics dan players lalu menginput id topic yang di-follow/unfollow pengguna ke dalam *collections* players. Di dalam *collections* player terdapat satu filter berjenis array yang bernama topics guna menyimpan topik apa saja yang sudah di-follow oleh pengguna tersebut.

```
app.post('/follow-topics', function(req, res)
{
    if (Security.RequestIsValid(req, res))
    {
        var player = new Player();

        player.PullByToken(req.body.token, function()
        {
            if (player.IsInstantiated)
```

```
{
    var parsedTopics = req.body.ids.split(",").map(function(item)
    {
        return parseInt(item, 10);
    });

    player.FollowTopics(parsedTopics, function()
    {
        var response = utils.GetEmptyResponse();

        response.topics = player.GetPublicData().topics;

        res.send(response);
    });
  }
  else
  {
    errorResponse(res, 1);
  }
 });
 }
});

app.post('/unfollow-topics', function(req, res)
{
  if (Security.RequestIsValid(req, res))
  {
    var player = new Player();

    player.PullByToken(req.body.token, function()
    {
      if (player.IsInstantiated)
      {
        var parsedTopics = req.body.ids.split(",").map(function(item)
        {
          return parseInt(item, 10);
        });

        player.UnfollowTopics(parsedTopics, function()
        {
          var response = utils.GetEmptyResponse();

          response.topics = player.GetPublicData().topics;

          res.send(response);
        });
      }
      else
      {
        errorResponse(res, 1);
      }
```

```
        });
    }
})
```

Script 4.11 Script Follow dan Unfollow Topic

12) Mencari Pengguna Lain Untuk Bertanding

Script ini berfungsi untuk mencari pengguna lain untuk bertanding bersama. Cara kerjanya adalah menginput filter baru pada *collections* game_requests yang berisi id player pengguna dan id topik yang dipilih untuk dimainkan. Lalu saat ketemu, maka akan otomatis tergenerate id pertandingan dan soal yang nantinya akan dimasukkan ke dalam *collections* games.

```
function createGameRequest(playerID, topicID, res)
{
    DB.collection('game_requests').insert(
    {
        player_id: playerID,
        topic: parseInt(topicID),
        last_update: utils.GetTimestamp()
    }, function(err)
    {
        if (err)
            throw err;

        var response = utils.GetEmptyResponse();

        response.status = "searching";

        res.send(response);
    })
}

function readyGameResponse(game, me, res)
{
    var opponentID = game.player1.id == me.GetID() ? game.player2.id :
game.player1.id;

    var opponent = new Player();

    opponent.SetID(opponentID);

    opponent.Pull(function()
```

```
{
    if (opponent.IsInstantiated)
    {
        var response = utils.GetEmptyResponse();

        response.status = "ready";
        response.game = game;
        response.game.me = me.GetPublicData();
        response.game.opponent = opponent.GetPublicData();

        res.send(response);
    }
    else
    {
        errorResponse(res, 7);
    }
});
}
var game = new Game();

                    game.SetTopicByID(requestedTopicID,
function(success)
                    {
                        if (success)
                        {
                            game.GenerateQuestions(function(success)
                            {
                                if (success)
                                {
                                    game.AddPlayer(player.GetID());
                                    game.AddPlayer(requests[i].player_id);

                                    game.Push(function()
                                    {
                                        DB.collection('game_requests').update(
                                        {
                                            _id: requests[i]._id
                                        },
                                        {
                                            $set:
                                            {
                                                created_game_id: game.GetID()
                                            }
                                        }, function(err)
                                        {
                                            if (err)
                                                throw err;

readyGameResponse(game.GetPublicData(), player, res);
```

```
                    });
                });
            }
            else
            {
                errorResponse(res, 11);
            }
        });
    }
    else
    {
        errorResponse(res, 10);
    }
});
```

Script 4.12 Script Bertanding

13) Mengenerate Soal Yang Dimainkan

Script ini berfungsi untuk mengenerate soal sesuai topik yang dipilih. Script ini bekerja setelah pengguna saling bertemu untuk bertanding (poin no 12). Cara kerjanya adalah membuka *collections* questions lalu melakukan generate secara random lalu disimpan pada array untuk dikirimkan ke *collections* games.

```
this.GenerateQuestions = function(callback)
{
    if (In.Document.topic != undefined)
    {
        var appSettings = new AppSettings();

        appSettings.Pull(function(success)
        {
            if (success)
            {
                DB.collection('questions').find(
                {
                    topicId: In.Document.topic.id
                }).toArray(function(err, allQuestionsByTopic)
                {
                    if (err)
                        throw err;

                    var questions = [];

                    function isQuestionUnique(q, arr)
```

```
                {
                    for (var i = 0; i < arr.length; i++)
                    {
                        if (q.id == arr[i].id) return false;
                    }

                    return true;
                };

                while (questions.length <
appSettings.GetNumberOfQuestionsInGame() && questions.length <
allQuestionsByTopic.length)
                {
                    var randomIndex = utils.RandomIntInRange(0,
allQuestionsByTopic.length - 1);

                    if (isQuestionUnique(allQuestionsByTopic[randomIndex],
questions))
                        questions.push(allQuestionsByTopic[randomIndex]);
                }

                In.Document.questions = questions;

                In.Push(function()
                {
                    if (questions.length > 0)
                    {
                        callback(true);
                    }
                    else
                    {
                        callback(false);
                    }
                });
            });
        }
    });
  }
}
```

Script 4.13 Script Generate Soal

14) Saat Game Mulai

Script ini berfungsi saat pengguna 1 dan 2 bertanding. Dalam script ini, *collections* yang digunakan adalah games. Isi dari *collections* tersebut adalah id game, questions yang sudah di random (poin no 13), topic, data player 1 dan 2. Di dalam data player

tersebut, ada data jawaban masing - masing pengguna serta correctAnswers guna memvalidasi jawaban pengguna agar bisa menentukan pemenangnya.

```
player.PullByToken(requestedPlayerToken, function()
        {
           if (player.IsInstantiated)
           {
              var game = new Game();

              game.SetID(requestedGameID);

              game.Pull(function(success)
              {
                 if (success)
                 {
                    if (game.IsPlayerInTheGame(player.GetID()))
                    {
                       var question =
game.GetQuestionByID(requestedQuestionID);

                       if (question != undefined)
                       {
                          if (!game.IsQuestionAnswered(question.id,
player.GetID()))
                          {
                             game.Answer(question.id, requestedAnswerID,
player.GetID(), function(isAnswerCorrect)
                             {
                                if (isAnswerCorrect)
                                {
                                   player.AddCorrectAnswer(function()
                                   {
                                   });
                                }
```

Script 4.14 Script Saat Game Mulai

15) Saat Game Berakhir

Script ini berfungsi ketika pertandingan sudah berakhir guna menentukan pemenang sekaligus memperbaharui list leaderboard. Script ini masih menggunakan *collections* games yang mana diambil filter score untuk menentukan pemenangnya. Lalu

pemenangnya mendapatkan tambahan poin sesuai topik yang dimainkan dan data tersebut dimasukkan pada *collections* leaderboard dengan menginput id player, id topic, dan score topik tersebut.

```
if (game.IsFinished())
{
        var winner = game.GetWinner();
        if (winner != undefined)
        {
                var topic = game.GetTopic();
                var winnerPlayer = new Player();
                winnerPlayer.SetID(winner.id);

 winnerPlayer.AddPointToLeaderboard(topic.id, function()
 {
        successResponse();
});
}
        else
        {
                successResponse();
         }
}
```

Script 4.15 Script Saat Game Berakhir

4.3.2 Unity 3D

1) Remote Data Loader

Script ini digunakan untuk memanggil function guna menload beberapa data dari database. Data tersebut disebut dengan General Data. Ketika general data sudah dipastikan tidak kosong, maka data tersebut akan dibungkus dalam function onGeneralDataLoaded untuk digunakan ke depannya.

```
public class RemoteDataLoader : DataLoader
  {
    public override void loadData()
    {
```

```
        base.loadData();

        ServerAPI.getGeneralData(onLoadData);
    }

    void onLoadData(bool success, GeneralDataResponse res)
    {
        if (success)
        {
            StartCoroutine(loadTopicsImages(res.data));
        }
        else
        {
            if (onFailed != null)
            {
                onFailed();
            }
        }
    }
void generalDataIsReady(GeneralData data)
    {
        if (onGeneralDataLoaded != null)
            onGeneralDataLoaded(data);
    }
}
```

Script 4.16 Script Remote Data Loader

2) Data Manager

Script ini digunakan untuk mengatur beberapa data yang sudah dibungkus pada poin 1 lalu didistribusikan ke beberapa komponen seperti topik apa saja yang ada dalam database dan game settings yang berisi jumlah pertanyaan dalam 1 pertandingan dan minimal topik yang harus diikuti. Lalu data manager juga membuat request ke database untuk mengambil jumlah pertanyaan dalam 1 topik dan data leaderboard.

```
public class GeneralData
    {
        static GeneralData Instance;

        public Topic[] topics;
        public GameSettings settings;
```

```
        public GeneralData()
        {
            topics = new Topic[0];
            settings = new GameSettings();

            Instance = this;
        }
    }

    public static void GetQuestionsCount(int topicID,
    Callback<GetQuestionsCountByTopicResponse> callback)
        {
            ServerAPI.getQuestionsCountByTopic(topicID, callback);
        }

        public static void GetLeaderboardByTopic(int topicID,
    Callback<GetLeaderboardForTopicResponse> callback)
        {
            ServerAPI.getLeaderboardForTopic(topicID, callback);
        }
```

Script 4.17 Script Data Manager

3) Sign In, Sign Up, dan Auth

Script ini digunakan untuk melakukan proses authentication untuk masuk / mendaftar ke dalam *game*. Untuk menghindari inputan kosong dari pengguna, maka dilakukan pengecekan terhadap length username dan password.

```
    public static void signUp(string username, string password,
    Callback<LoginResponse> callback)
        {
            if (singleton != null)
                singleton._signUp(username, password, callback);
        }

        public void _signUp(string username, string password,
    Callback<LoginResponse> callback)
        {
            StartCoroutine(
                api_call(
                    "signup",
                    new string[]
                        {
                            "username", username,
```

```
                "password_hash", Util.Sha256(password)
            },
            callback
        )
    );
}

public static void signIn(string username, string password,
Callback<LoginResponse> callback)
    {
        if (singleton != null)
            singleton._signIn(username, password, callback);
    }

    public void _signIn(string username, string password,
Callback<LoginResponse> callback)
    {
        StartCoroutine(
            api_call(
                "signin",
                new string[]
                {
                    "username", username,
                    "password_hash", Util.Sha256(password)
                },
                callback
            )
        );
    }

public bool SignIn(string username, string password)
    {
        if (username.Length > 0 && password.Length > 0)
        {
            ServerAPI.signIn(username, password, loginCallback);

            return true;
        }

        return false;
    }

    public bool SignUp(string username, string password)
    {
        if (username.Length > 0 && password.Length > 0)
        {
            ServerAPI.signUp(username, password, loginCallback);

            return true;
        }
```

```
        return false;
    }
```

Script 4.18 Script Sign In, Sign Up, Auth

4) Profile Manager

Script ini kegunaannya sangat multifungsi yaitu adalah function follow atau unfollow topik, function guna menjawab pertanyaan saat pertandingan berlangsung, function untuk menyimpan token yang dijelaskan pada bagian nodejs poin ke 8.

```
private void onSceneLoaded(Scene scene, LoadSceneMode mode)
    {
        if (scene.name == "Auth")
        {
            authManager.CheckSavedToken();
        }
    }

public APICallAnswer(int answerID, string gameID, int questionID,
Action<bool> callback)
        {
            this.callback = callback;

            ServerAPI.answer(authManager.Token, answerID, gameID,
questionID, onLoad);
        }

        void onLoad(bool success, AnswerResponse res)
        {
            callback(success && res.success);
        }

public static void FollowTopics(Topic[] topics)
    {
        var ids = new int[topics.Length];

        for (int i = 0; i < ids.Length; i++)
        {
            ids[i] = topics[i].id;
        }

        ServerAPI.followTopics(authManager.Token, ids, onFollowTopics);
    }

    static void onFollowTopics(bool success, FollowTopicsResponse res)
```

```
{
    if (success)
    {
        Profile.topics = res.topics;

        if (OnFollowTopicsSuccessful != null)
            OnFollowTopicsSuccessful();

        if (OnChangeFollowedTopics != null)
            OnChangeFollowedTopics(Profile.topics);
    }
    else
    {
        if (OnFollowTopicsFailed != null)
            OnFollowTopicsFailed();
    }
}
```

Script 4.19 Script Profile Manager

5) Mencari Pengguna Lain Untuk Bertanding

Script ini digunakan untuk pencarian pengguna lain untuk bertanding. Script ini berjalan setelah pengguna mengklik tombol play di salah satu topik yang tersedia. Jika 2 pengguna saling bertemu, maka respon menjadi ready dan halaman dipindahkan ke scene InGameMultiplayerMode.

```
public void FindGame(Topic topic, Action<bool, FindGameResponse>
callback)
    {
        onFoundGameCallback = callback;

        ServerAPI.findGame(ProfileManager.authManager.Token, topic.id,
findGameCallback);
    }

    void findGameCallback(bool success, FindGameResponse res)
    {
        if (success)
        {
            if (res.status == "ready")
            {
                currentGame = res.game;

                setTopicSprite();
```

```
            }
        }

        if (onFoundGameCallback != null)
            onFoundGameCallback(success, res);
    }

    void setTopicSprite()
    {
        if (currentGame != null)
        {
            var topic = DataManager.GetTopicByID(currentGame.topic.id);

            if (topic != null)
            {
                currentGame.topic.Sprite = topic.Sprite;
            }
        }
    }

public static void findGame(string token, int topicID,
Callback<FindGameResponse> callback)
    {
        if (singleton != null)
            singleton._findGame(token, topicID, callback);
    }

    public void _findGame(string token, int topicID,
Callback<FindGameResponse> callback)
    {
        StartCoroutine(
            api_call(
                "find-game",
                new string[]
                    {
                        "token", token,
                        "topic", topicID.ToString()
                    },
                callback
            )
        );
    }

void onFound(bool success, FindGameResponse response)
    {
        if (isActive)
        {
            if (gameObject != null && isActiveAndEnabled)
            {
                if (!success || response.status == "searching")
                {
```

```
            StartCoroutine(pauseAndRequest());
        }
        else if (response.status == "ready")
        {
            StateMachine.ChangeState(State.InGameMultiplayerMode);
        }
    }
}
                                                                    }
```

Script 4.20 Script Mencari Pengguna Untuk Bertanding

6) Generate Pertanyaan dan Batas Waktu

Script ini digunakan untuk generate otomatis soal dari database dan disimpan ke dalam list array. Sementara untuk timer, waktu tersisa (timeLeft) akan terus menerus dikurangi sesuai dengan satuan Time.deltaTime.

```
public void start(float time)
    {
        totalTime = timeLeft = time;

        routine = StartCoroutine(countdown());

        Debug.Log("Total Time: "+ totalTime);
        Debug.Log("Time: "+ time);
        Debug.Log("Routine: " + routine);
    }

    public void stop()
    {
        if (routine != null)
            StopCoroutine(routine);
    }

    IEnumerator countdown()
    {
        yield return null;

        while (timeLeft > 0)
        {
            timeLeft -= Time.deltaTime;

            var normalizedTime = timeLeft / totalTime;

            mainImage.color = color.Evaluate(1f - normalizedTime);
```

```
            mainTransform.anchorMax = new
Vector2(Mathf.Clamp(normalizedTime, 0.05f, 1f), 1f);
            mainTransform.offsetMax = Vector2.zero;

            yield return null;
        }

        if (onFinished != null)
            onFinished();
    }
```

Script 4.21 Script Generate Soal dan Batas Waktu

7) Saat Game Mulai

Script ini berlaku untuk kedua pengguna yang sedang bertanding dan digunakan untuk menjawab serta mendapatkan update tentang jawaban lawan benar atau salah. Ketika list dari array poin ke 6 sudah habis, maka langsung akan menjalankan script EndGame untuk menentukan pemenangnya. Untuk ProfileManager.Answer sudah dibahas dalam poin 4 (Profile Manager) bahwa akan memvalidasi jawaban dari pengguna dengan jawaban benar yang ada di database.

```
public static void Answer(int answerID, int questionID, Action<bool> callback)
    {
        ProfileManager.Answer(answerID, singleton.game.id, questionID,
callback);
    }

if (success && info.readyBool)
        {
            var opponentLastAnswer =
info.opponent.gameInfo.answers[info.opponent.gameInfo.answers.Length - 1];

            var myLastAnswer =
info.me.gameInfo.answers[info.me.gameInfo.answers.Length - 1];

            var correctAnswer =
info.me.gameInfo.correctAnswers[info.me.gameInfo.correctAnswers.Length -
1];
```

```
QuestionManager_MM.ShowResultForCurrentQuestion(myLastAnswer,
opponentLastAnswer, correctAnswer);

        if (OnUpdateGameInfo != null)
            OnUpdateGameInfo(info);

        ExecWithDelay.Exec(1.5f, () =>
        {
            if (!QuestionManager_MM.NextQuestion())
            {
                EndGame(info);
            }
        });
    }
    else
    {
        ExecWithDelay.Exec(0.25f, () =>
        {
            WaitTheSecondPlayerAnswer();
        });
    }
```

Script 4.22 Script Game Dimulai

8) Saat Game Berakhir

Script ini digunakan untuk menentukan siapa pemenang dari pertandingan yang sudah dilakukan. Cara kerjanya adalah terus menerus melakukan update terhadap skor yang dimiliki oleh masing – masing pengguna dan akhirnya dilakukan perbandingan contoh jika skor pengguna 1 lebih besar dari pengguna 2 maka pengguna 1 pemenangnya.

```
public static void EndGame(GameInfoUpdate info)
    {
        Panel_GameResult_MM.Initialize(info);

        InGameUI_MM.setState(InGameUIState.Result);
    }

void initialize(GameInfoUpdate info)
    {
        player1Name.text = info.me.playerObject.username;
        player2Name.text = info.opponent.playerObject.username;
```

```
        player1Score.text = info.me.gameInfo.score.ToString();
        player2Score.text = info.opponent.gameInfo.score.ToString();

        player1Level.text = string.Format("Level {0}",
info.me.playerObject.level);
        player2Level.text = string.Format("Level {0}",
info.opponent.playerObject.level);

        GameStatusWin.SetActive(info.me.gameInfo.score >
info.opponent.gameInfo.score);
        GameStatusLose.SetActive(info.me.gameInfo.score <
info.opponent.gameInfo.score);
        GameStatusDraw.SetActive(info.me.gameInfo.score ==
info.opponent.gameInfo.score);
    }
```

Script 4.23 Script Game Berakhir

4.4 Pembuatan Server dan Database

4.4.1 Server

1) Instalasi Linux dengan varian Ubuntu 18.04

   Operating System (OS) yang digunakan untuk implementasi game ini adalah Ubuntu. Dikarenakan distro yang terkenal *developer-friendly*.

2) Penginstalan Apache

   Apache digunakan untuk mem-bypass port 3000 yang akan digunakan untuk membuat *game* dapat diakses secara daring.

3) Setting Apache

Settingan dibawah ini digunakan agar *game* dapat berjalan. Settingan ini dilakukan pada file 000-default.conf pada direktori /etc/apache2/sites-enabled. ServerName di-set ke IP server dikarenakan *game* Ayo Hitung hanya dapat diakses lewat IP bukan domain. Lalu penyimpanan file untuk menjalankan *game* disimpan di direktor /var/www/html. Dan yang terpenting adalah mem-bypass port ke 3000 agar *game* dapat berjalan.

Gambar 4.6 Setting Apache

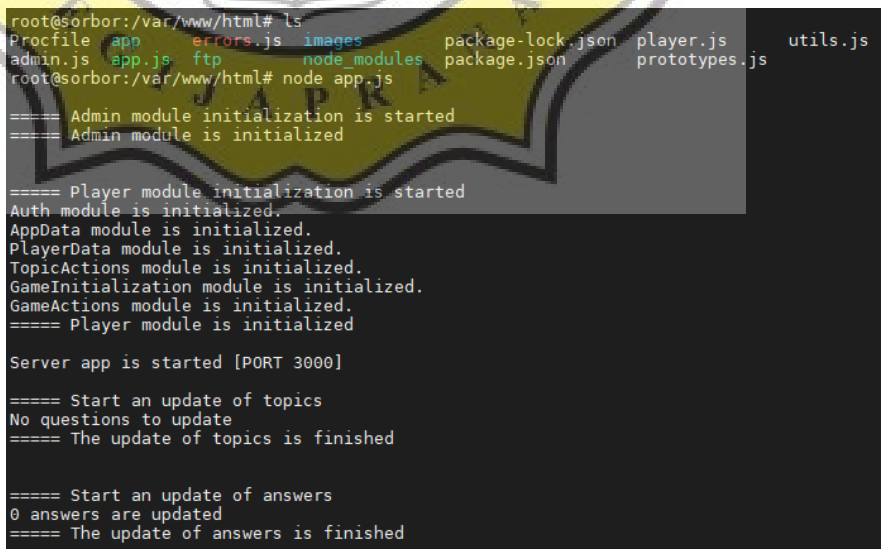
4) Penginstalan dan Setting NodeJS

Node adalah *javascript runtime* yang digunakan untuk server dari *game* Ayo Hitung. Seperti yang sudah ditulis pada poin 3, file yang digunakan untuk membuat *game* menjadi daring ditempatkan pada direktori /var/www/html. Setelah file dipastikan ada pada

direktori tersebut, langkah selanjutnya adalah melakukan penginstalan modul yang diperlukan agar *game* dapat daring dengan perintah *npm install*.

5) Penjalanan *Game* Menjadi *Online*

Setelah melakukan tahap – tahap sebelumnya dengan benar, maka langkah selanjutnya adalah membuat *game* menjadi daring. Untuk tahap pencegahan server tiba – tiba mati, maka digunakan *Screen*. *Screen* sebenarnya digunakan untuk saling membagikan apa yang sedang dilakukan pengguna A ke B, tetapi untuk studi kasus ini digunakan agar dapat menutup server tanpa mematikan server. Untuk membuat *screen*, maka digunakan *screen -S nama_screen*. Untuk melakukan pelepasan dari *screen* menggunakan command *Ctrl+A D*. Sedangkan untuk melakukan akses ulang ke *screen* yang telah dibuat adalah *screen -r id_screen*. Setelah melakukan setting dengan *screen* selanjutnya adalah melakukan proses daring di server yaitu dengan cara *node app.js*. Jika sukses, maka akan muncul info seperti gambar dibawah ini.



Gambar 4.7 Sukes Menjalankan Server

4.4.2 Database

1) Login Pada Website MongoDB

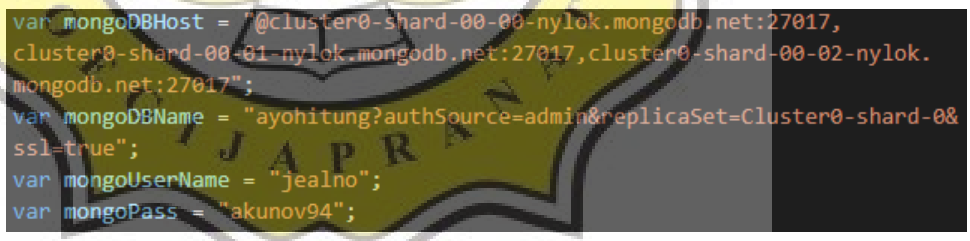
Database yang akan digunakan pada *game* Ayo Hitung adalah MongoDB. Pada implementasinya maka layanan yang digunakan adalah *cloud* dari mongodb.

2) Membuat Cluster

Setelah berhasil login, maka langkah selanjutnya adalah membuat cluster baru. Yang dimaksud dari cluster adalah suatu blok data yang disediakan oleh layanan mongo yang dikhususkan untuk penggunaan perseorangan.

3) Koneksi Database dengan Server

Setelah berhasil membuat cluster, pengguna akan diberikan domain khusus untuk mengakses databasenya. Lakukan pengisian domain, username, password, nama database pada file app.js yang ada di server seperti gambar dibawah ini.

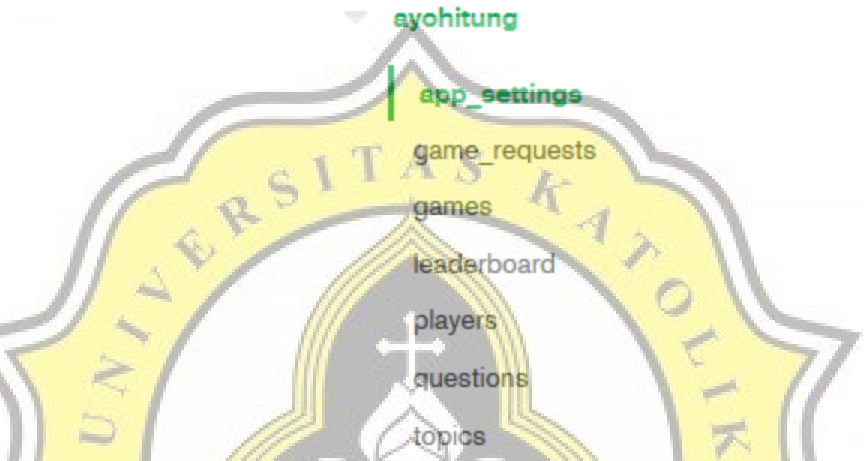


Gambar 4.8 Koneksi Database dengan Server

Setelah melakukan beberapa tahap diatas, proses pembuatan *database* sudah selesai. Istilah *table* yang digunakan dalam MongoDB adalah *collections*. Untuk isi dari *collections* sendiri akan terbuat otomatis saat kita menjalankan *game* via Administator Scene pada Unity.

4.4.3 Penjelasan Struktur Database

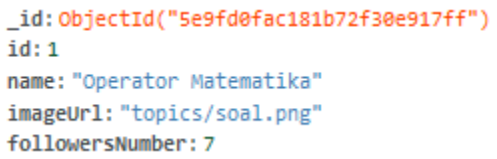
Ketika sudah menjalankan scene Administator pada Unity lalu membuat topik dan pertanyaan, maka *collections* app_settings, questions, dan topics akan terbuat otomatis. Sedangkan untuk *collections* sisanya akan terbuat saat *game* dimainkan oleh pengguna.



Gambar 4.9 Collections Ayo Hitung

1) Topics

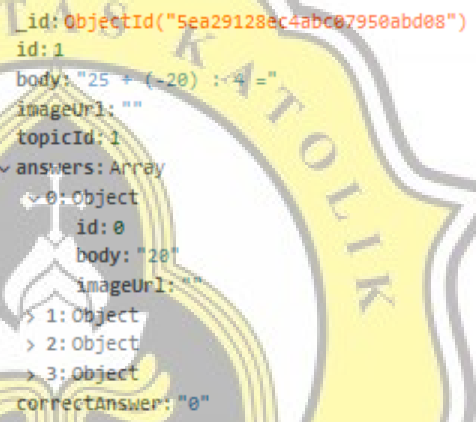
Terdapat id, name, imageUrl, dan followersNumber. ID akan digunakan untuk melakukan korelasi dengan *collections* lainnya. Name digunakan untuk mengisi nama topik tersebut. imageUrl digunakan untuk menyimpan logo dari topik tersebut. Dan followersNumber digunakan untuk mengetahui seberapa banyak topik ini diikuti.



Gambar 4.10 Collections Topics

2) Questions

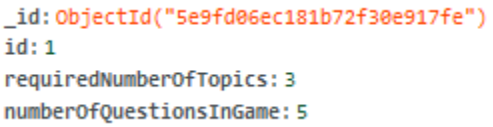
Terdapat id, body, imageUrl, topicId, answers (array), dan correctAnswer. ID akan digunakan untuk melakukan korelasi dengan *collections* lainnya. Body digunakan untuk mengisi pertanyaan. topicID digunakan untuk mengkorelasikan antara pertanyaan dan topik yang telah dibuat. Answers digunakan untuk menginput 4 jawaban yang mempunyai isi id dan body yang merupakan jawaban pilihan gandanya.



Gambar 4.11 Collections Questions

3) App Settings

Terdapat id, requiredNumberofTopics, dan numberOfQuestionsInGame. requiredNumberOfTopics digunakan untuk minimal jumlah topik yang harus diikuti. Sedangkan numberOfQuestionsInGame adalah jumlah pertanyaan dalam pertandingan.



Gambar 4.12 Collections App Settings

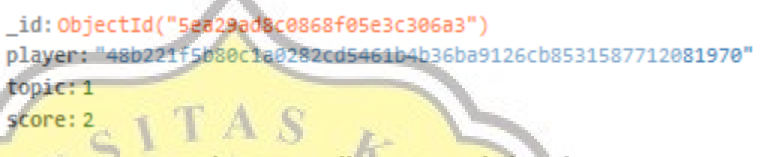
4) Players

Terdapat id, username, password_hash, email, tokens, fbID, topics, correctAnswers, dan level. ID digunakan untuk korelasi dengan *collections* lainnya. Username dan password_hash digunakan sebagai user dan password untuk login ke dalam *game*. Email digunakan menyimpan email dari fbID. Tokens tergenerate otomatis setiap pengguna dengan username tersebut login ke dalam *game* dan akan dikorelasikan dengan *collections* lainnya. Topics digunakan sebagai topik yang telah diikuti oleh pengguna tersebut yang dikorelasi dengan id yang ada di *collections* topics. CorrectAnswers digunakan sebagai jumlah berapa jawaban benar yang telah dijawab oleh pengguna. Level digunakan acuan sudah seberapa tinggi level yang diraih

Gambar 4.13 Collections Players

5) Leaderboard

Terdapat player, topic, dan score. Player disini dikaitkan dengan id yang ada pada *collections* players. Topic dikaitkan dengan id yang ada pada *collections* topics. Dan score adalah seberapa banyak pertandingan yang dimenangkan oleh pengguna pada topic yang dikorelasikan dengan *collections* players dan topics.



Gambar 4.14 Collections Leaderboard

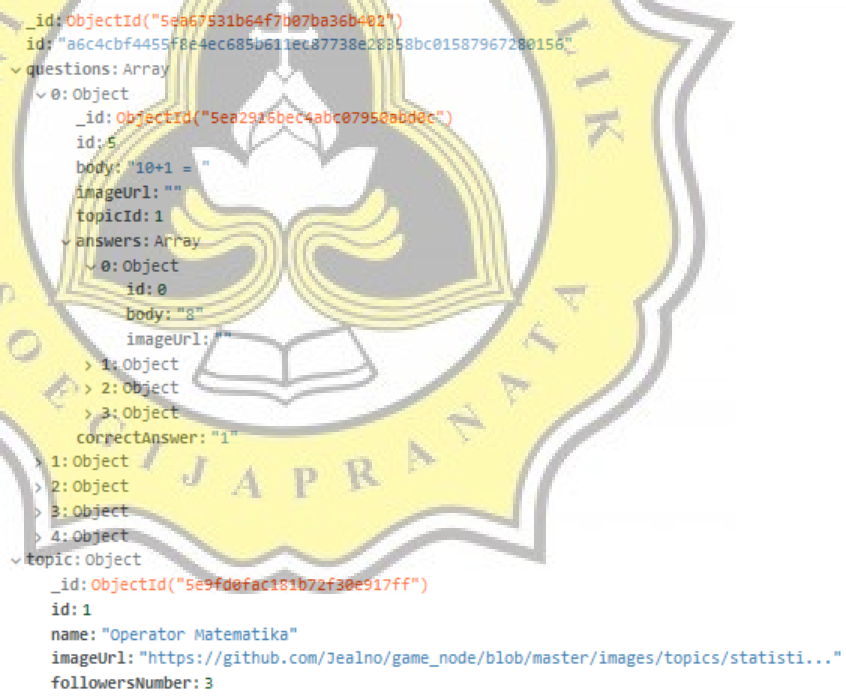
6) Game Request

Terdapat player_id, topic, last_update dan created_game_id. *Collections* ini digunakan ketika pengguna mencari pengguna lain untuk bertanding. Player_id sini dikaitkan dengan id yang ada pada *collections* players. Topic diambil dari id yang ada pada *collections* topics. Last_update adalah waktu ketika pengguna mencari pengguna lainnya. Jika ketemu, maka akan tergenerate otomatis game_id pada created_game_id.

Gambar 4.15 Collections Game Request

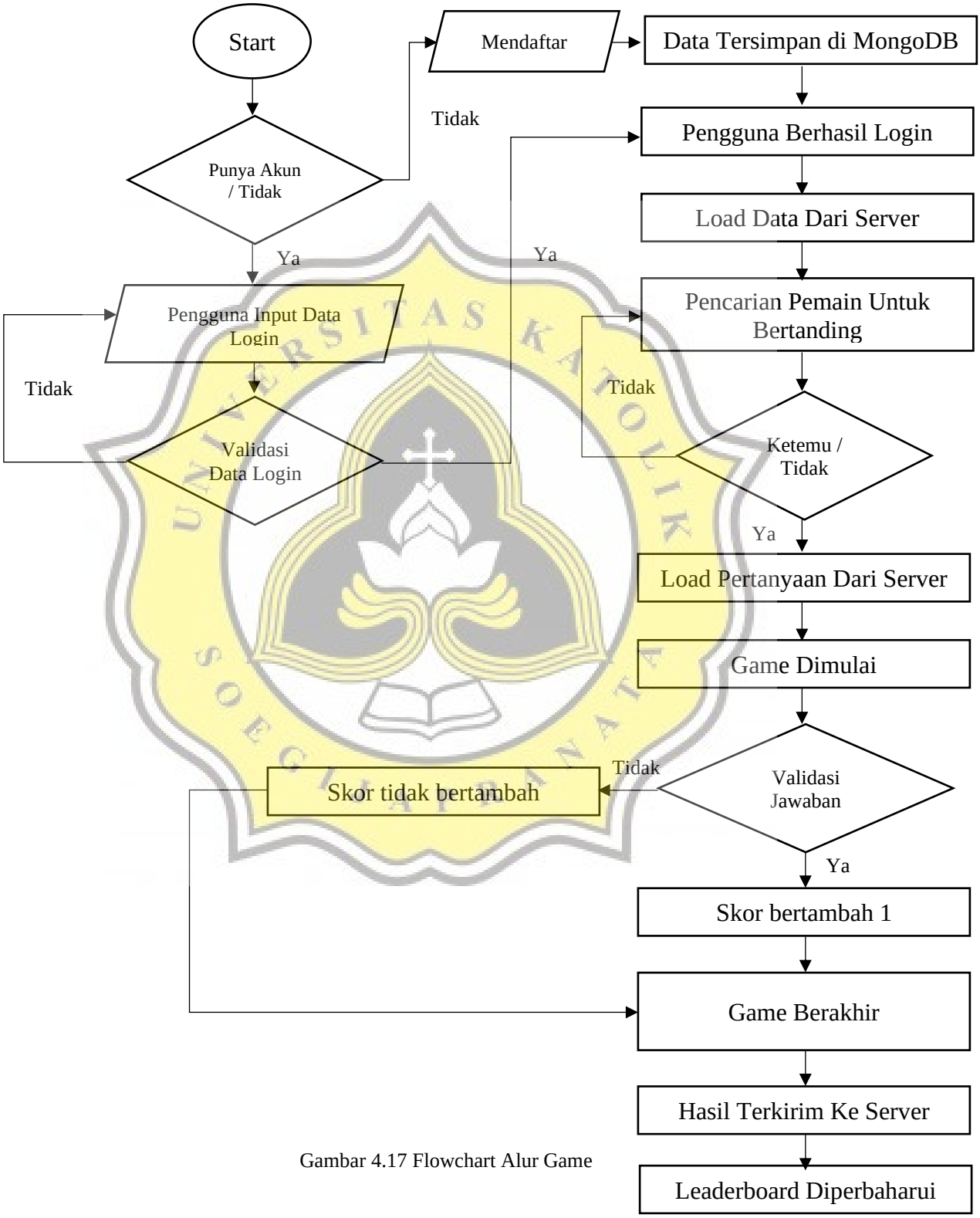
7) Games

Terdapat banyak sekali parameter yang ada di *collections* ini. ID diambil dari created_game_id yang ada pada *collections* game_request. Lalu questions diambil sesuai dengan topik yang dipilih pengguna sebelum mencari pertandingan. Lalu ada tambahan 2 object yaitu player 1 dan 2 yang digunakan memvalidasi jawaban benar atau tidak yang telah dipilih oleh pengguna. Untuk memvalidasinya adalah dengan cara mengambil object correctAnswer yang ada pada *collections* questions lalu membandingkan dengan jawaban pengguna pada object answers. Jika benar maka object score akan ditambah 1.



Gambar 4.16 Collections Games

4.5 Alur Game



Gambar 4.17 Flowchart Alur Game

Penjelasan flowchart diatas adalah sebagai berikut:

1. Punya Akun / Tidak

   Jika pengguna mempunyai akun, maka pengguna akan diminta menginput data. Jika tidak, maka akan diarahkan untuk mendaftar.

2. Validasi Data Login

   Setelah pengguna memasukan data login, dilakukan pengecekan ke database MongoDB untuk mengecek apakah valid / tidak data login tersebut pada *collections players*. Jika tidak valid maka akan dikembalikan ke menu input. Jika benar maka akan pengguna berhasil login.

3. Mendaftar

   Jika pengguna tidak mempunyai akun, maka harus melakukan input username dan password.

4. Data Tersimpan di MongoDB

   Setelah pengguna menginput di halaman pendaftaran, maka data akan dimasukan ke dalam database MongoDB di bagian *collections players.*

5. Pengguna Berhasil Login

   Setelah pengguna masuk melalui halaman login / mendaftar. Pengguna berhasil masuk ke halaman utama *game* Ayo Hitung.

6. Load Data Dari Server

   Setelah pengguna masuk ke halaman utama, di saat itu juga *game* men-load data dari server. Data yang di-load adalah profile, jumlah pertanyaan yang berhasil dijawab, topik, dan leaderboard.

7. Pencarian Pemain Untuk Bertanding

   Untuk pencarian pemain lainnya untuk bertanding, pengguna membuka salah satu topik lalu akan muncul sebuah popup yang berisi Play, Follow/Unfollow, dan Leaderboard. Untuk melakukan pencarian, maka pengguna memilih menu play.

8. Load Pertanyaan Dari Server

   Setelah 2 pengguna bertemu, maka *game* meminta *request* ke server untuk mendapatkan pertanyaan berdasarkan topik yang sudah sebelumnya dipilih oleh pengguna.

9. Game dimulai

   Pada tahap ini, 2 pengguna bertanding berlawanan dengan menjawab 5 soal dengan 4 pilihan jawaban dan waktu yang terbatas per pertanyaan.

10. Validasi Jawaban

    Validasi jawaban ini dilakukan ketika pengguna memilih salah satu jawaban maka langsung dilakukan pengecekan ke server apakah jawabannya valid / tidak. Jika salah, maka skor tidak bertambah. Jika benar, maka skor bertambah satu.

11. Game Berakhir

    Saat game berakhir, maka dilakukan penghitungan skor antara pemain 1 dan 2 yang sudah bertanding dengan 5 soal. Dengan tujuan menentukan siapa pemenangnya.

12. Hasil Terkirim Ke Server

Setelah perhitungan dilakukan secara lokal pada tahap nomor 11, maka data skor dan pengguna akan dikirimkan di server dengan tujuan untuk mengumumkan pemenangnya di layar masing – masing pengguna.

13. Leaderboard Diperbaharui

Setelah mendapatkan pemenangnya, maka leaderboard dengan pengguna yang barusan bermain dan topik yang dipilih akan ditambahkan 1 poin.

4.6 Pengujian Game

4.6.1 Deskripsi Responden

Setelah melaksanakan uji coba *game* dan pembagian kuesioner *post-test* kepada anak kelas 6 SD di kota Cirebon, didapatkan data yang valid dan tepat sasaran guna pengujian *game* Ayo Hitung. Data yang sudah didapatkan kemudian diolah menggunakan aplikasi IBM SPSS Statistics 24. Gambaran responden berdasarkan umur dan jenis kelamin disajikan pada tabel 4.1.

Tabel 4.1 Data Umur Responden

Umur		
Umur (tahun)	Frekuensi	Presentase
11	9	22,5 %
12	24	60 %
13	7	17,5 %
Total	**40**	**100%**

Dilihat dari tabel 4.1, dengan total 40 responden bersumber dari siswa siswi kelas 6 SD di kota Cirebon yang berumur 11 – 13 tahun. Responden yang berumur 11 tahun sebanyak 9 orang (22,5%), 24 responden berumur 12 tahun

(60%), 7 responden berumur 13 tahun (17,5%). Dari 40 responden yaitu siswa siswi kelas 6 yang memainkan *game* Ayo Hitung umur yang banyak bermain *game* Ayo Hitung adalah 12 tahun.

Tabel 4.2 menyajikan hasild data uji coba *game* Ayo Hitung berdasarkan jenis kelamin.

Tabel 4.2 Data Jenis Kelamin Responden

Jenis Kelamin		
Jenis Kelamin	Frekuensi	Presentase
Laki – laki	26	65 %
Perempuan	14	35 %
Total	40	100 %

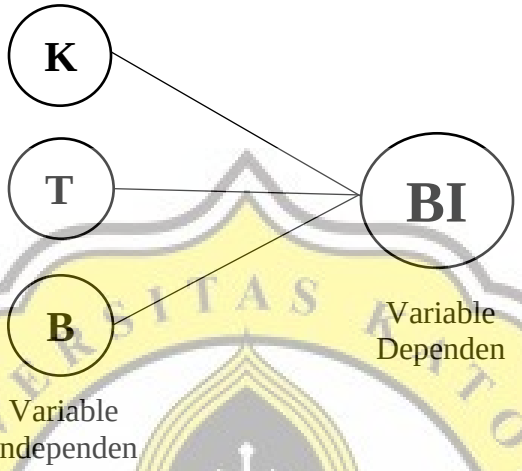
Data pada Tabel 4.2 merupakah jumlah responden sebanyak 40 responden yang dikelompokan berdasarkan jenis kelamin. Pada tabel 4.3 menunjukan jumlah responden laki-laki sebanyak 26 responden (65%), untuk responden perempuan berjumlah 14 responden (35%).

4.7 Analisa Variabel Penelitian

4.7.1 Model Pengujian

Penelitian ini memakai dua model pengujian yaitu Variabel *Dependen* dan Variabel *Independen*. Variabel Dependen adalah variabel yang sangat dipengaruhi oleh variabel lainnya. Variabel *dependen* yang dipakai dalam penelitian ini adalah Minat (BI). Variabel *Independen* adalah variabel yang bebas dan tidak terpengaruh oleh variabel lainnya. Variabel *Independen* yang dipakai adalah Kesenangan (K), Ketertarikan (T), dan Keterlibatan (B). Variabel *Dependen* dan *Independen* akan diuji dalam program IBM SPSS guna

memastikan korelasi antar variabel. Mengenai model pengujian yang akan dipakai dalam g*ame* Ayo Hitung, adalah sebagai berikut:



Gambar 4.18 Model Pengujian *Game* Ayo Hitung

Hipotesa pengujian :

1. Kesenangan (K) *Game* AyoHitung berpengaruh kuat terhadap Minat (BI).
2. Ketertarikan (T) *Game* AyoHitung berpengaruh kuat terhadap Minat (BI).
3. Keterlibatan (B) *Game* AyoHitung berpengaruh kuat terhadap Minat (BI)

4.7.2 Validitas Variabel Kuesioner

Penelitian ini memakai 5 variabel penelitian yaitu: Kesenangan (K), Ketertarikan (T), dan Keterlibatan (B), Minat (BI) untuk menguji game "AyoHitung". Semua variabel penelitian diharuskan untuk diuji keabsahannya guna menilai apakah pertanyaan di kuesioner dapat mewakili setiap variabel dalam memvalidasi pandangan dari responden. Untuk menguji keabsahan pada setiap variabel maka dilakukan pengujian dengan metode Principal Component Faktor Analysis.

Setelah dilakukan metode pengujian Principal Component Faktor Analysis dengan variabel K1, K2, K3, T1, T2, T3, B1, B2, B3, BI1, BI2, BI3

maka diperoleh hasil yang kurang sinkron. Dari tabel 4.3 terlihat untuk variabel K3 dan T1 tidak valid.

Tabel 4.3 Hasil Pengujian Validitas Metode PCA Ke-1

	Component			
	1	2	3	4
K1			.805	
K2			.821	
K3			.068	
T1	.159			.685
T2	.821			
T3	.748			
B1		.848		
B2		.836		
B3		.596		
BI1	.816			
BI2	.722			
BI3	.838			

Karena masih ada data yang tidak valid maka dilakukan pengujian *Principal Component Faktor Analysis* diulang dengan variable K3 dan T1 tidak dimasukan dalam pengujian. Dari tabel 4.4 dapat disimpulkan hasil sudah sinkron yaitu semua variabel sudah berada pada letak yang setara dan valid. Pengujian dengan metode ini bermaksud untuk mengenaskan bahwa seluruh indikator dalam tiap variabel diatas adalah *konvergen*.

Tabel 4.4 Hasil Pengujian Validitas Metode PCA Ke-2

	Component			
	1	2	3	4
K1				.796
K2				.833
T2	.802			
T3	.764			
B1			.836	
B2			.830	
B3			.646	
BI1	.823			
BI2	.709			
BI3	.840			

Setelah beberapa pengujian maka indikator yang dinyatakan valid dan dapat dipakai untuk metode pengujian berikutnya adalah K1, K2, T2, T3, B1, B2, B3, BI1, BI2, BI3.

4.7.3 Analisa Reabilitas Variabel Kuesioner

Koefisien Cronbach's Alpha dipakai untuk menguji tingkat kepercayaan pada kuesioner. Dari hasil pengujian realibilitas table 4.5 dapat disimpulkan sebagai berikut.

Tabel 4.5 Koefisien Cronbach's Alpha untuk masing-masing variabel

Variabel	Koefisien Cronbach's Alpha	Hasil
Kesenangan (K)	0,547	*Poor*
Ketertarikan (T)	0,611	*Questionable*
Keterlibatan (B)	0,754	*Acceptable*
Minat (BI)	0,872	*Good*

Variabel K masuk dalam kategori *Poor*, variabel T masuk dalam kategori *Questionable*, variabel B masuk dalam kategori *Acceptable*, dan variabel BI masuk dalam kategori *Good.* Hasil pengujian ini serasi dengan *Internal Consistency Crombach alpha* pada tabel 4.6

Tabel 4.6 Penilaian Realibilitas

Cronbach's alpha	Internal consistency
α ≥ 0.9	Excellent
0.9 > α ≥ 0.8	Good
0.8 > α ≥ 0.7	Acceptable
0.7 > α ≥ 0.6	Questionable
0.6 > α ≥ 0.5	Poor
0.5 > α	Unacceptable

4.7.4 Uji Korelasi

Metode yang dipakai untuk penelitian ini adalah memakai uji korelasi antar variabel untuk menguji apakah variabel Kesenangan (K), Ketertarikan (T), Keterlibatan (B) mempunyai korelasi dengan variabel Minat (BI). Dari Tabel 4.8 dapat disimpulkan bahwa T dan B mempunyai korelasi dengan Intensi (I). Dan variabel Kesenangan (K) tidak berelasi dengan Intensi (BI). Data lainnya yang ditemukan adalah ketertarikan (T) mempunyai relasi yang kuat dengan keterlibatan (B) pengguna saat memainkan *game* Ayo Hitung. Ini menunjukan bahwa ketertarikan (T), keterlibatan (B) dalam menggunakan *game* Ayo Hitung memiliki hubungan yang sangat erat dengan keinginan dari pengguna untuk belajar matematika dengan basis daring (BI).

Tabel 4.7 Korelasi Antar Variabel

		SK	ST	SB	SBI
SK	Pearson Correlation	1	.036	-.008	-.083
	Sig. (2-tailed)		.826	.959	.612
	N	40	40	40	40
ST	Pearson Correlation	.036	1	.476**	.843**
	Sig. (2-tailed)	.826		.002	.000
	N	40	40	40	40
SB	Pearson Correlation	-.008	.476**	1	.621**
	Sig. (2-tailed)	.959	.002		.000
	N	40	40	40	40
SBI	Pearson Correlation	-.083	.843**	.621**	1
	Sig. (2-tailed)	.612	.000	.000	
	N	40	40	40	40

**. Correlation is significant at the 0.01 level (2-tailed).