VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

PROJECT FINAL REPORT

<u>Course</u>: Mobile Application Development

<u>Lecturer</u>: Dr. Lê Duy Tân

THE BRAIN TRAIN GAME

By
Phạm Hồng Đăng – ITITIU19009
Dương Vĩ Quyền – ITITIU19045
Nguyễn Hoàng Duy – ITITIU19111

Ho Chi Minh City, Vietnam
2022

Phạm Hồng Đăng	Nguyễn Hoàng Duy	Dương Vĩ Quyền
1/3	1/3	1/3

Table 1: Contributions

GitHub repository: https://github.com/hongdangcseiu/DDQ-Brain-Train

TABLE OF CONTENTS

Table of Contents

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

1.1. Background

After the 4.0 revolution or the technological revolution, the world is currently advancing quickly. More social services are being offered, and the majority of them are delivered quickly. Taking advantage of this chance, many young businesspeople and entrepreneurs have aspired to create their own services in order to keep up with the current trend and contribute to theglobal development.

However, in order for a service to become popular and widely used by users, many factors must be considered, especially regarding social services especially support for patients. We know that MCI is the stage between the cognitive decline of normal aging and the deleterious cognitive decline of dementia but we can make it progress more slowly. And for a hospital to be able to make some games in order to help them improve their brain, managing patient accounts as well as their playing progress is an extremely important factor.

For patients, our team created a straightforward brain game. In addition to letting people play memory games in categories including memory, attention, language, and math, this game will help the hospital manage patients' information and game progress. With the hope that this app is somewhat helpful in improving the cognition of MCI patients.

1.2. Problem Statement

To prevent the development of MCI, the doctor must give and guide the brain exercise for the patient. However, patients may also forget the assigned exercise's content or take time to the therapy rooms. Besides, it will be challenging for the doctor to manage patient information and evaluate the development of the disease.

Along with the development of smartphones, creating an app can help the doctor to manage the patient as well as create simple memory games to improve the MCI disease is the best method. Help users save time and the doctor will know the disease situation through the game

1.3. Scope and Objectives

- Although it was originally planned that the application could let doctors manage patients remotely and create brain games for users, due to time reasons, our project can only allow users to play pre-made games and record their progress:

- The scope of this project will be limited to the following:

I. User login through the app
II. User personal profiles: Patient will have access to their profiles and their level of each game.
III. User Dashboard: Patient will be able to see their progress and information account.
IV. Homepage: User can access to list of games, choose level, and play game.

II. METHODOLOGY

2.1 Overview

In this project, Android Studio is the developer tool to create this mobile game app using the main language for coding in Java. Besides, for information user and game data management, we use SQL lite database engine to create the database easily and manage data conveniently.

By using DAO pattern, it's Easy to extend, maintain. All storage details are hidden from the rest of the application. Therefore, changes can be made by modifying only one implementation of the DAO while the rest of the application is left unaffected. The DAO acts as an intermediary between the application and the database.

2.2 User requirement analysis

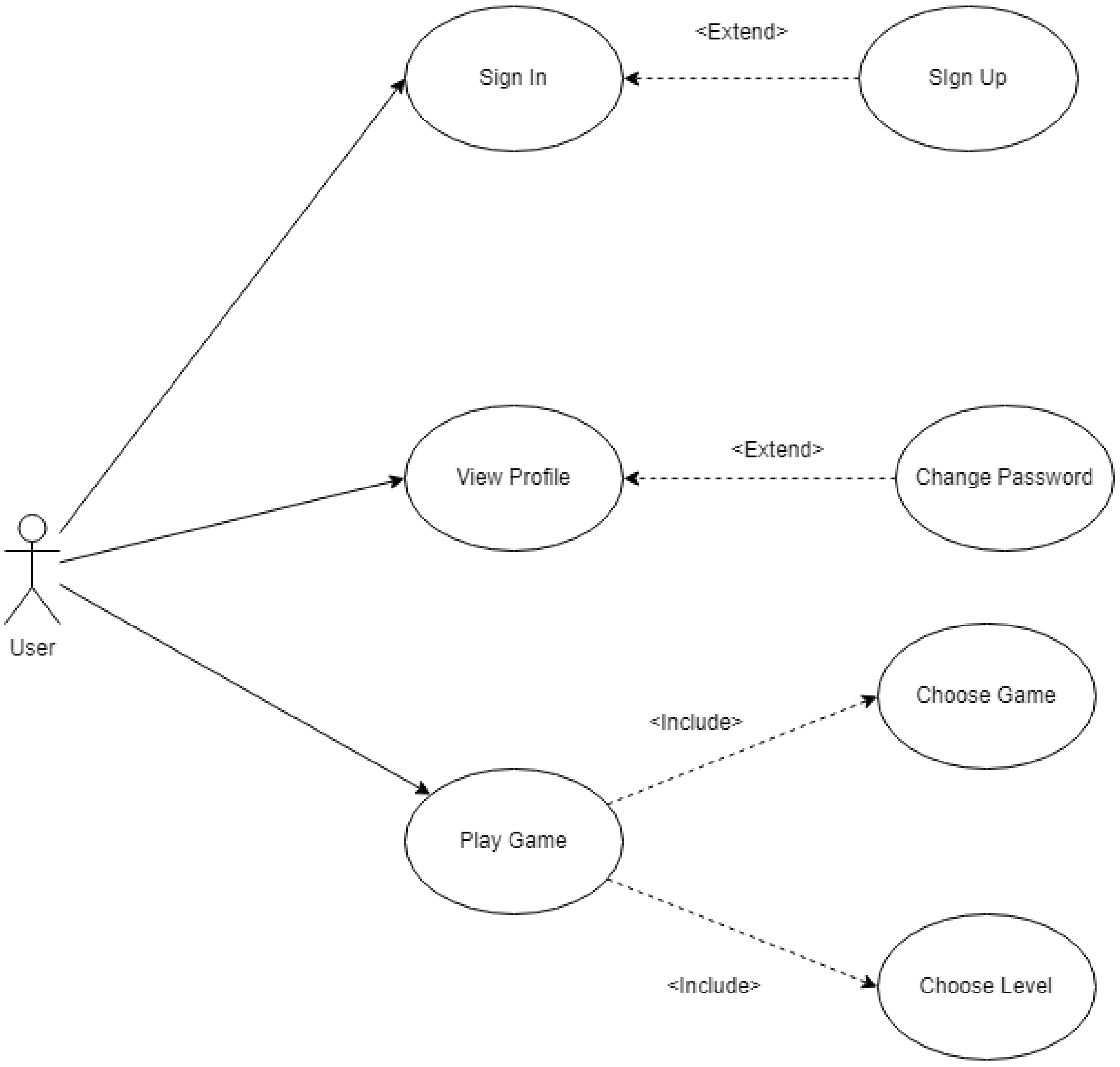


Figure II.1: Use case diagram

Description: The app is only now user-focused, so the diagram just have only one actor with some use cases:

- The users log in to application to use, if they do not have account, the user can sign up for new member before loging in.

- In the user dashboard, the users can access to profile to view the information and can change the password of your account.

- After choosing game and level, the users can play the game.

2.3 System Design

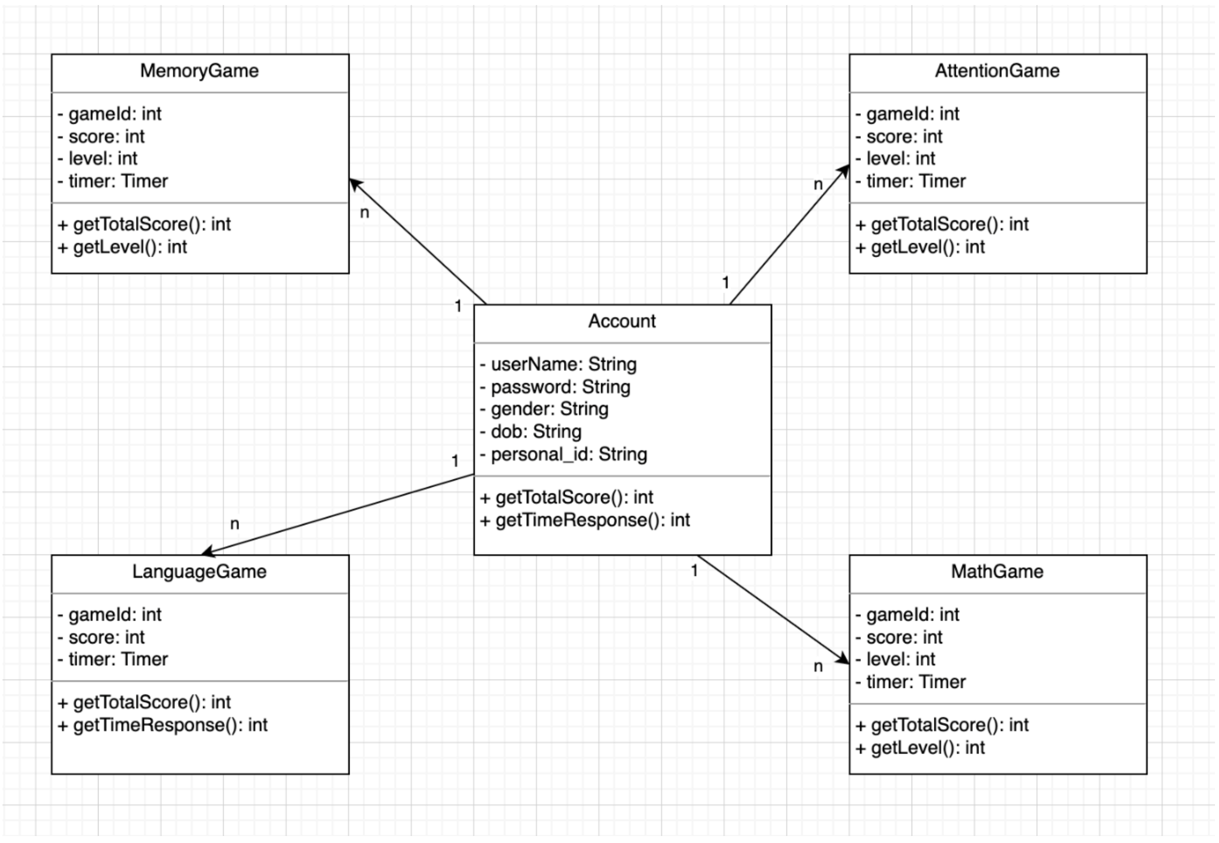


Figure II.2: Class diagram

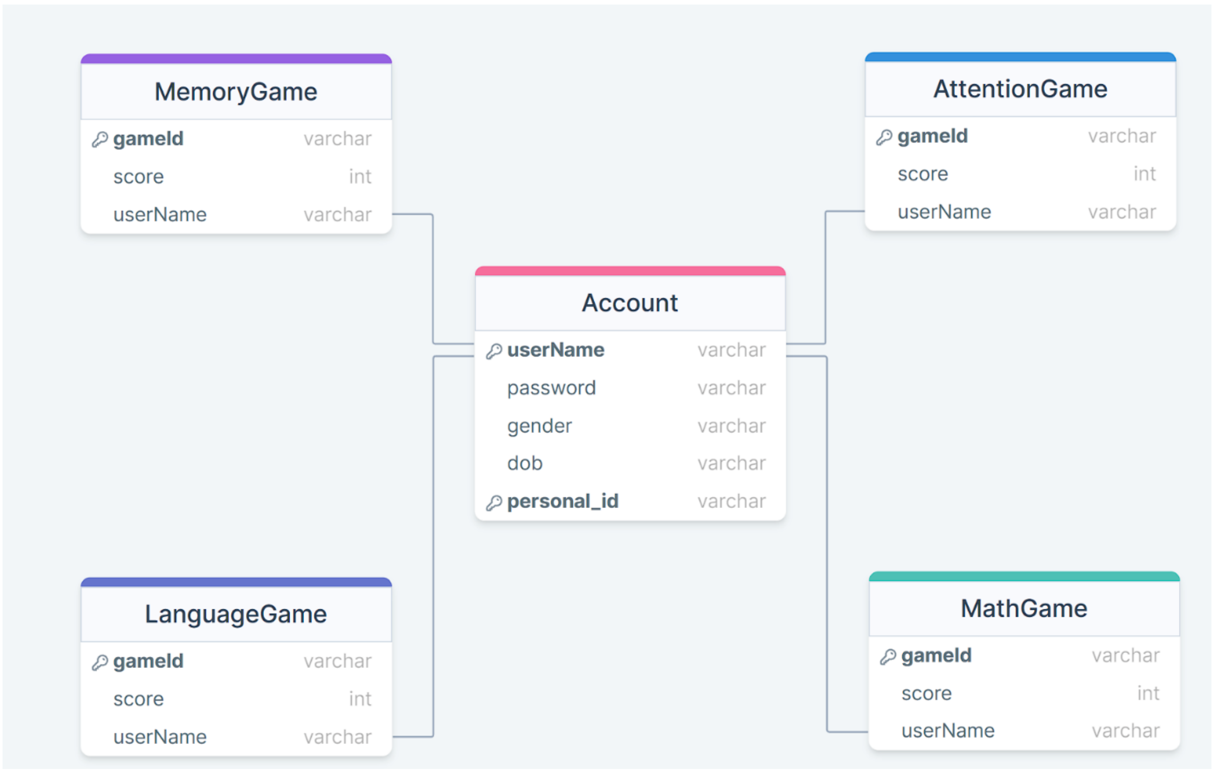


Figure II.3: Database Design

Description:

- Each user has one account that is determined by their personal ID. Each account can access four category games: Memory, Attention, Language, and Math by foreign key userName.

2.5 User Interface design

- Sign In, Sign Up

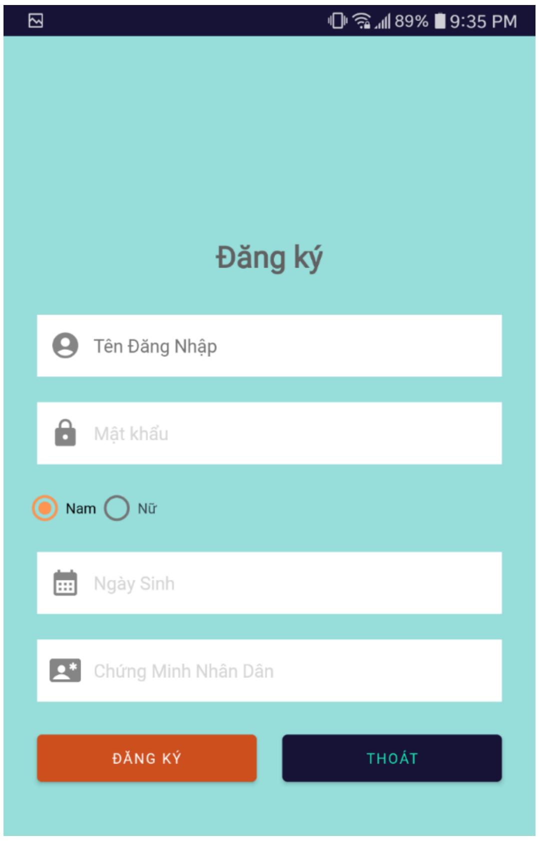


Figure II.4: Sign In

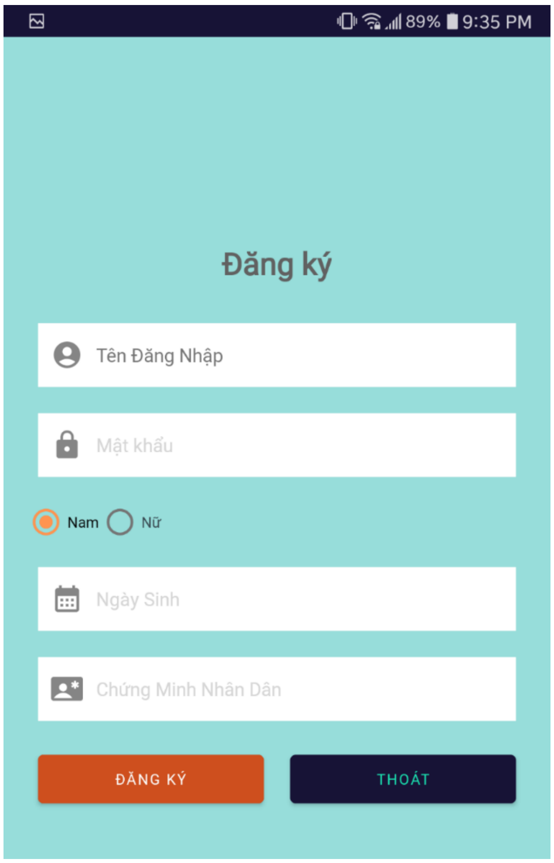


Figure II.5: Sign Up

Description: In order to login to the account user have to input the correct created username and password. If user doesn't have an account they can use the sign up feature.

- Main Dashboard

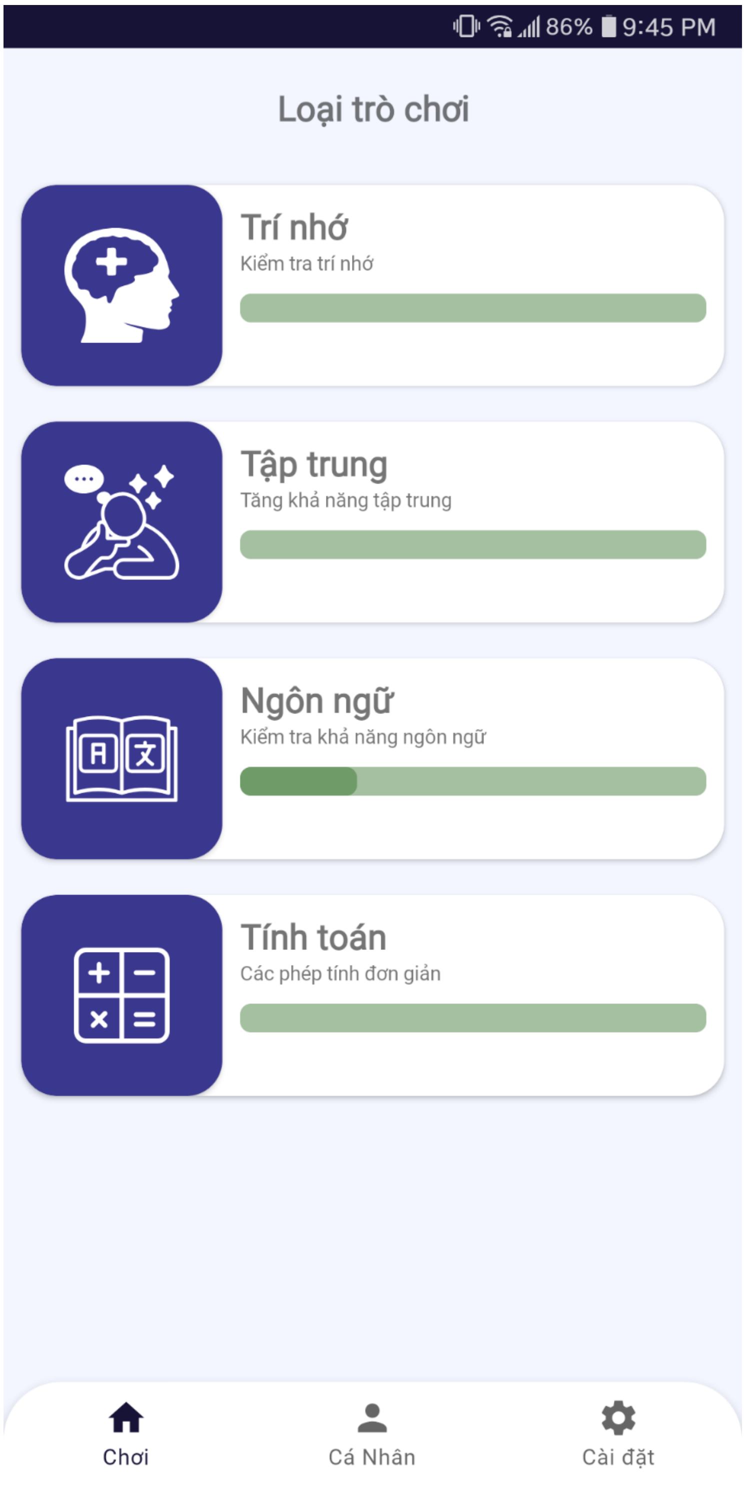


Figure II.6: Main dashboard Interface

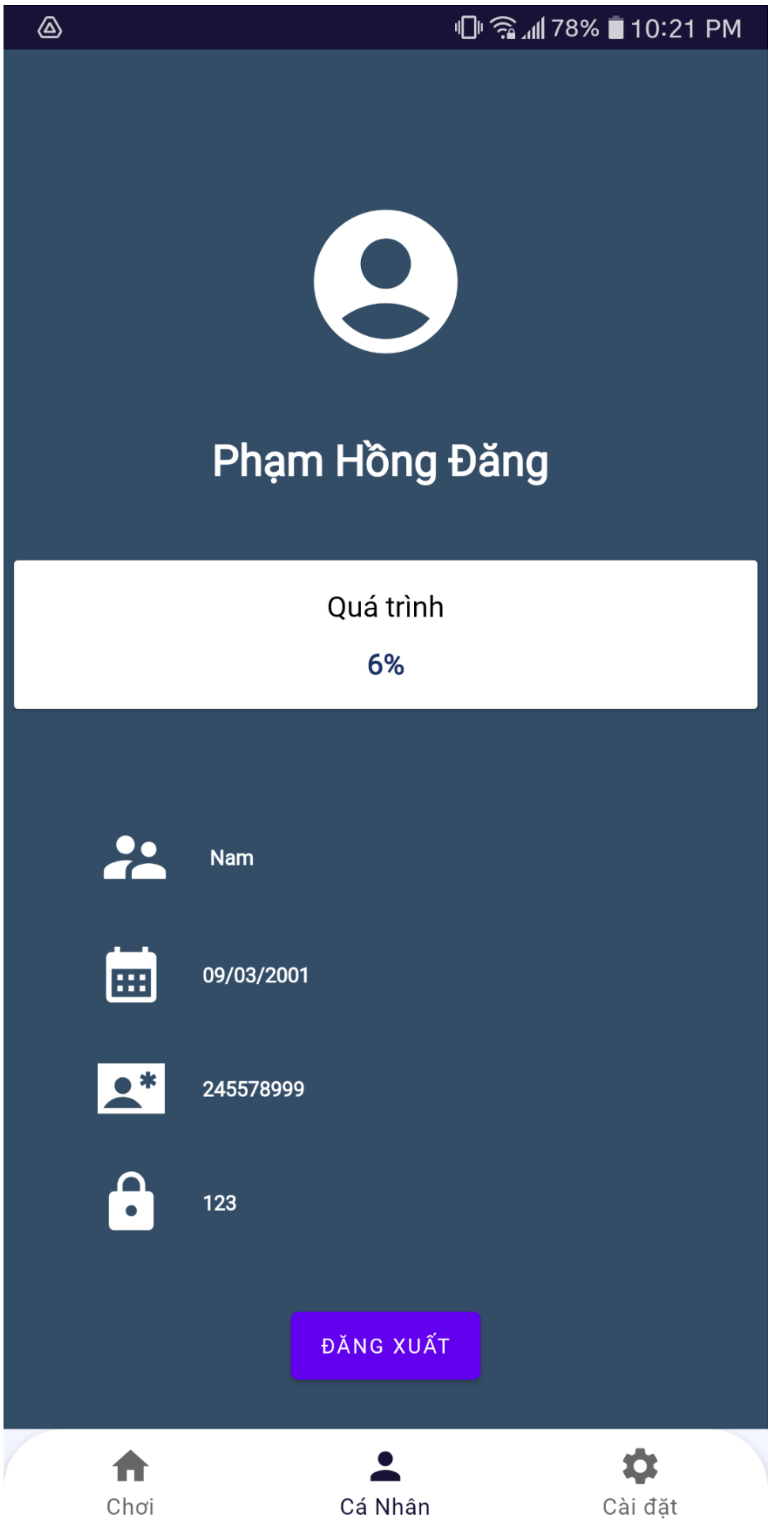
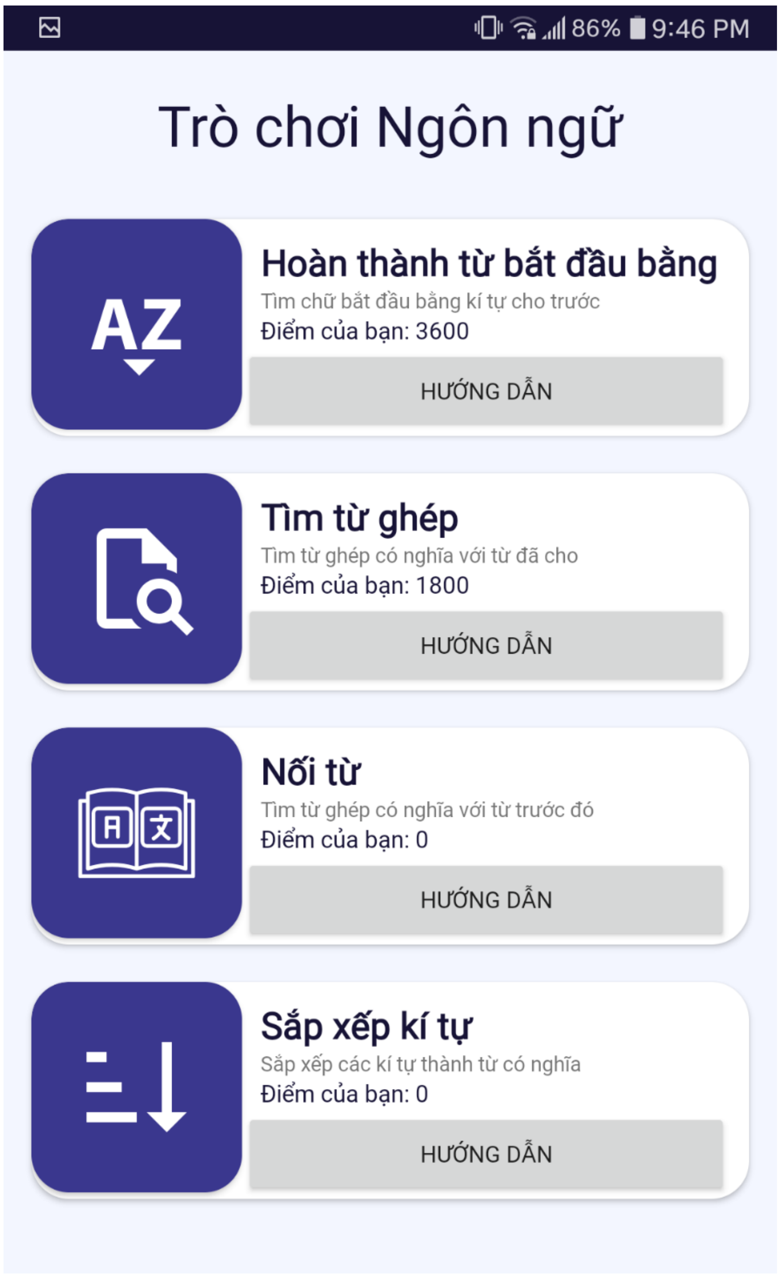


Figure II.7: Profile Interface

Description: In the main dashboard, users can access the game area, their profile, and setting. In the game area, users can choose the game they want to play and see the process of each category game. To see the user information, users can access their profile or access settings to change the function of the application.

- Choosing Game and Level



86% 9:46 PM

Trò chơi Ngôn ngữ

Hoàn thành từ bắt đầu bằng
Tìm chữ bắt đầu bằng kí tự cho trước
Điểm của bạn: 3600

Tìm từ ghép
Tìm từ ghép có nghĩa với từ đã cho
Điểm của bạn: 1800

Nối từ
Tìm từ ghép có nghĩa với từ trước đó
Điểm của bạn: 0

Sắp xếp kí tự
Sắp xếp các kí tự thành từ có nghĩa
Điểm của bạn: 0

Figure II.8: Choosing game Menu

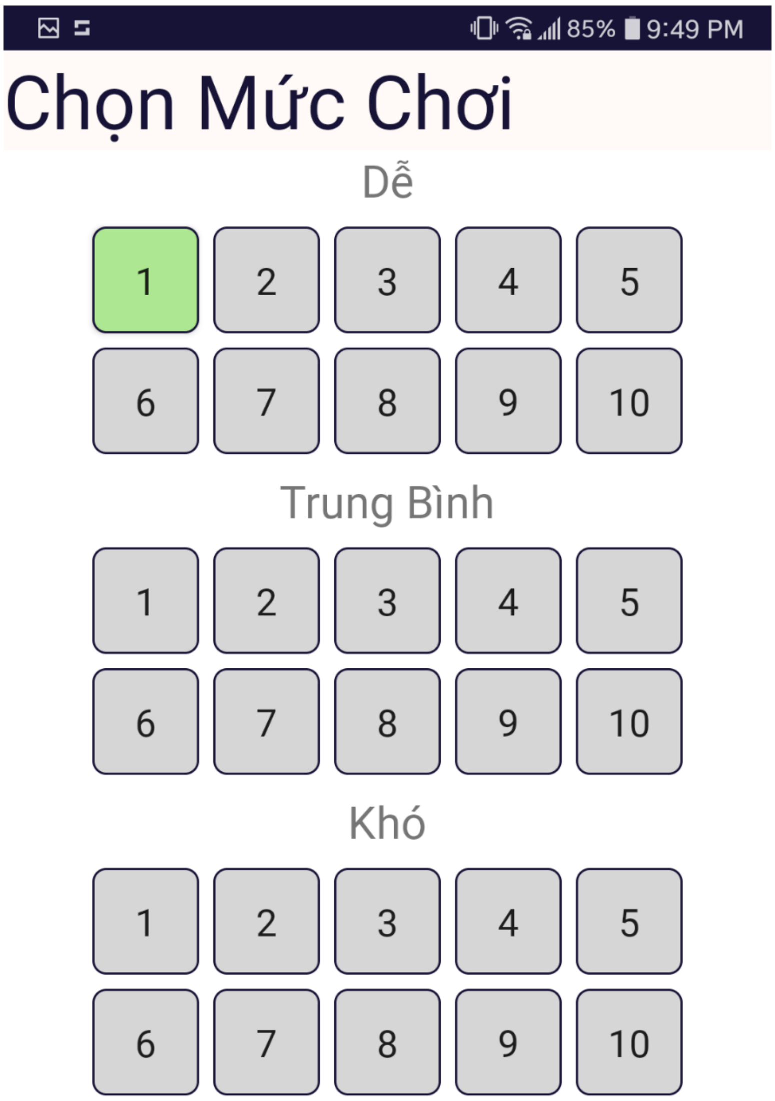


Figure II.9: Choosing Game Level

Description: Chosing game interface is the place that allow user can choose each game in each category, check the percentage of completion and their score. After chosing game, the level interface will show all available level that player can play.

- Game

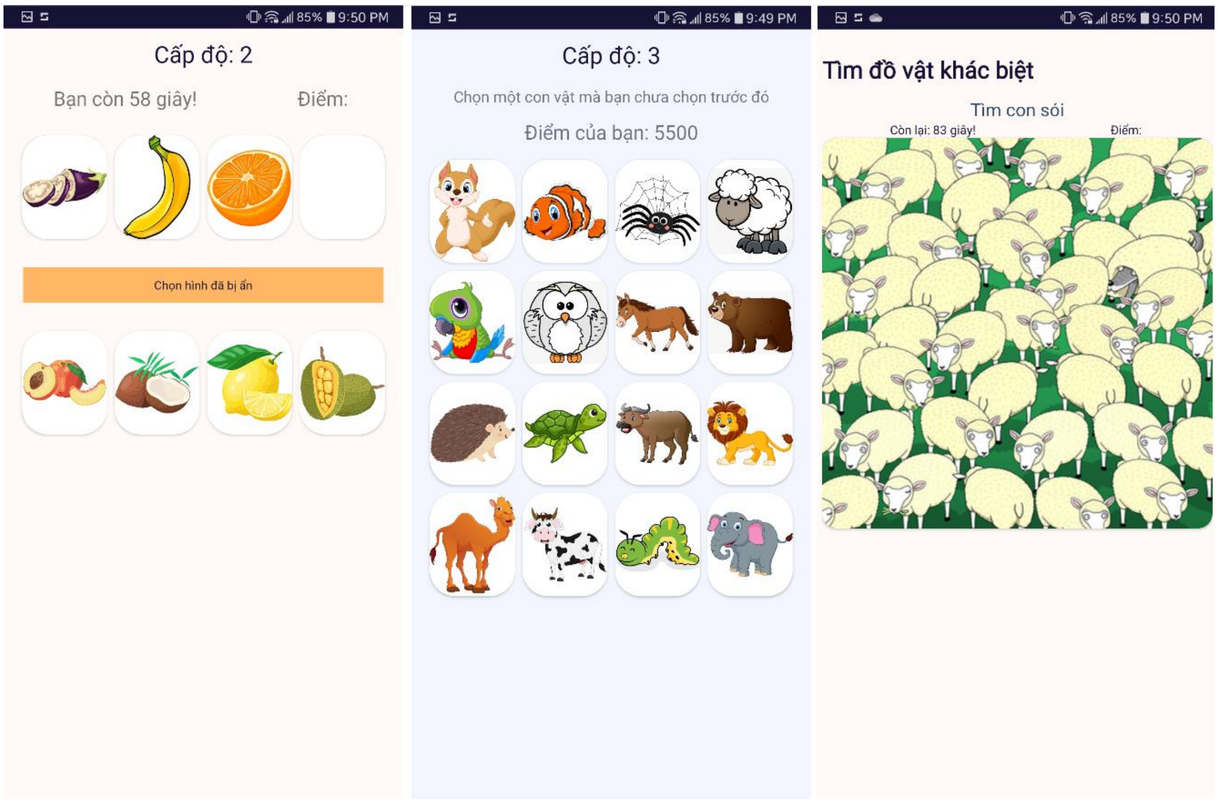
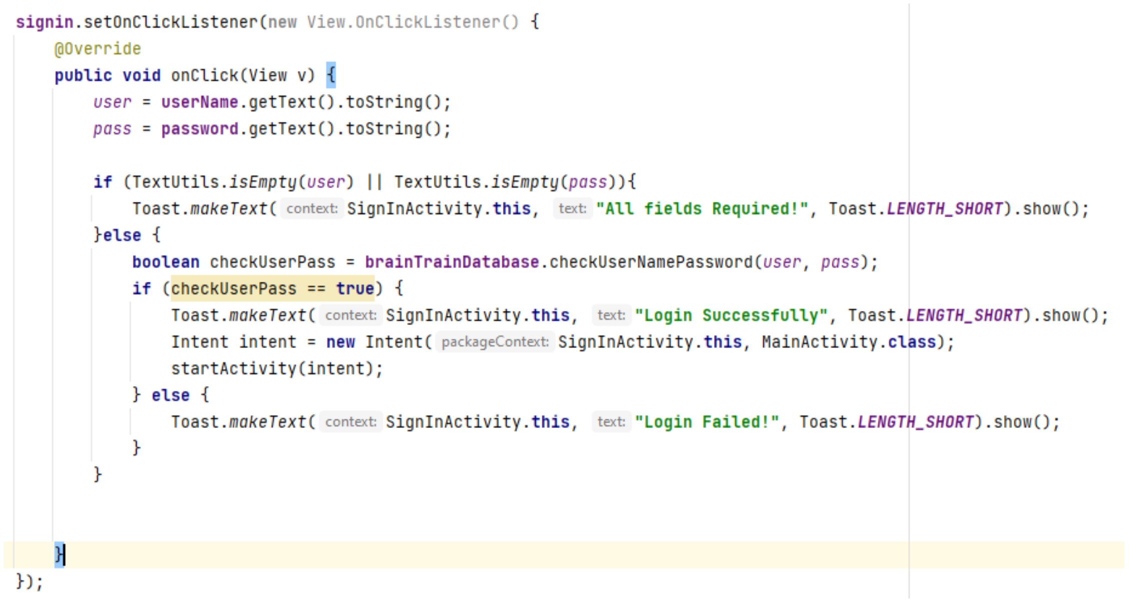


Figure II.10: Game UI

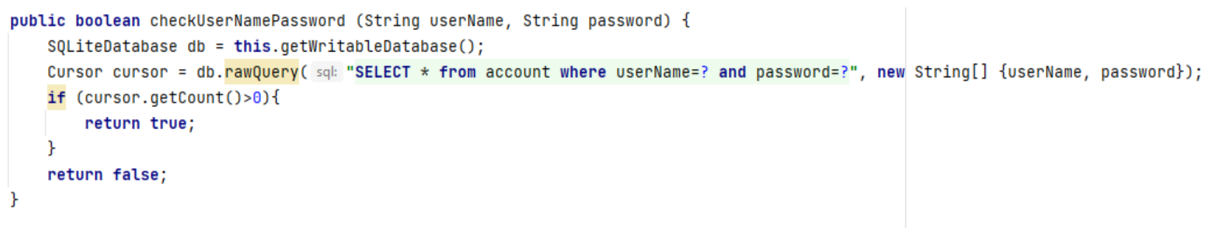
III. IMPLEMENT AND RESULTS

3.1. Implement



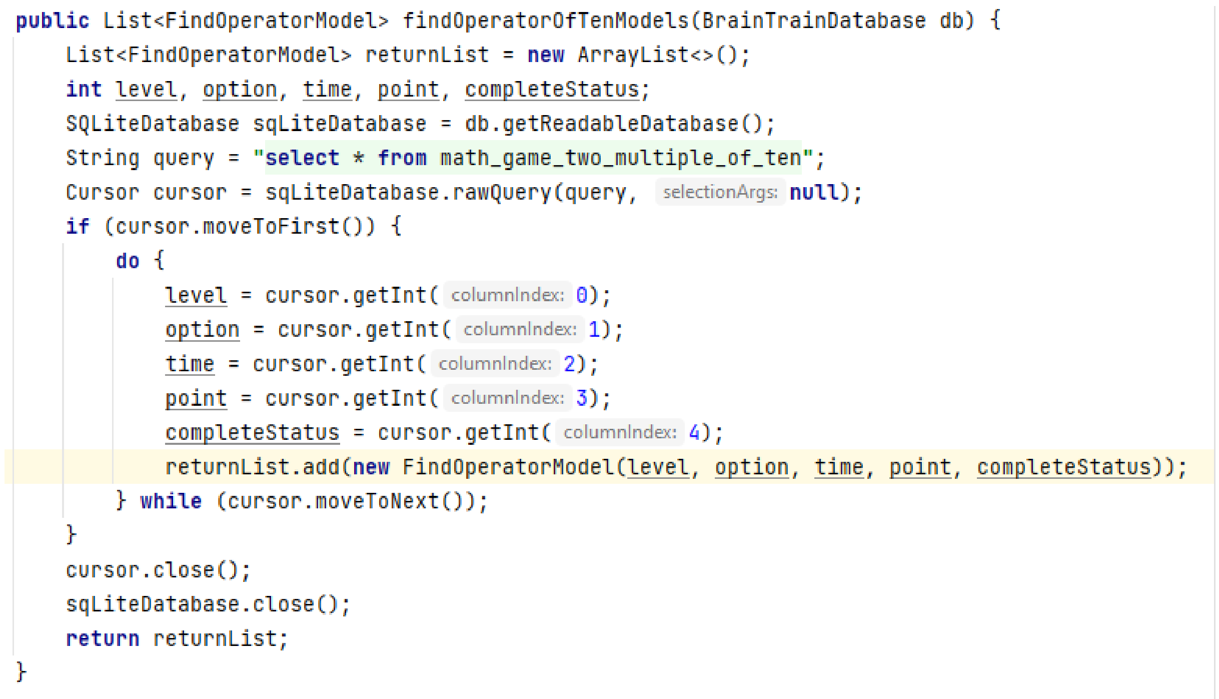
```
signin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        user = userName.getText().toString();
        pass = password.getText().toString();

        if (TextUtils.isEmpty(user) || TextUtils.isEmpty(pass)){
            Toast.makeText( context: SignInActivity.this,  text: "All fields Required!", Toast.LENGTH_SHORT).show();
        }else {
            boolean checkUserPass = brainTrainDatabase.checkUserNamePassword(user, pass);
            if (checkUserPass == true) {
                Toast.makeText( context: SignInActivity.this,  text: "Login Successfully", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent( packageContext: SignInActivity.this, MainActivity.class);
                startActivity(intent);
            } else {
                Toast.makeText( context: SignInActivity.this,  text: "Login Failed!", Toast.LENGTH_SHORT).show();
            }
        }


    }
});
```

Figure III.1: Login Function



```
public boolean checkUserNamePassword (String userName, String password) {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "SELECT * from account where userName=? and password=?", new String[] {userName, password});
    if (cursor.getCount()>0){
        return true;
    }
    return false;
}
```

Figure III.2: Check existing user from database

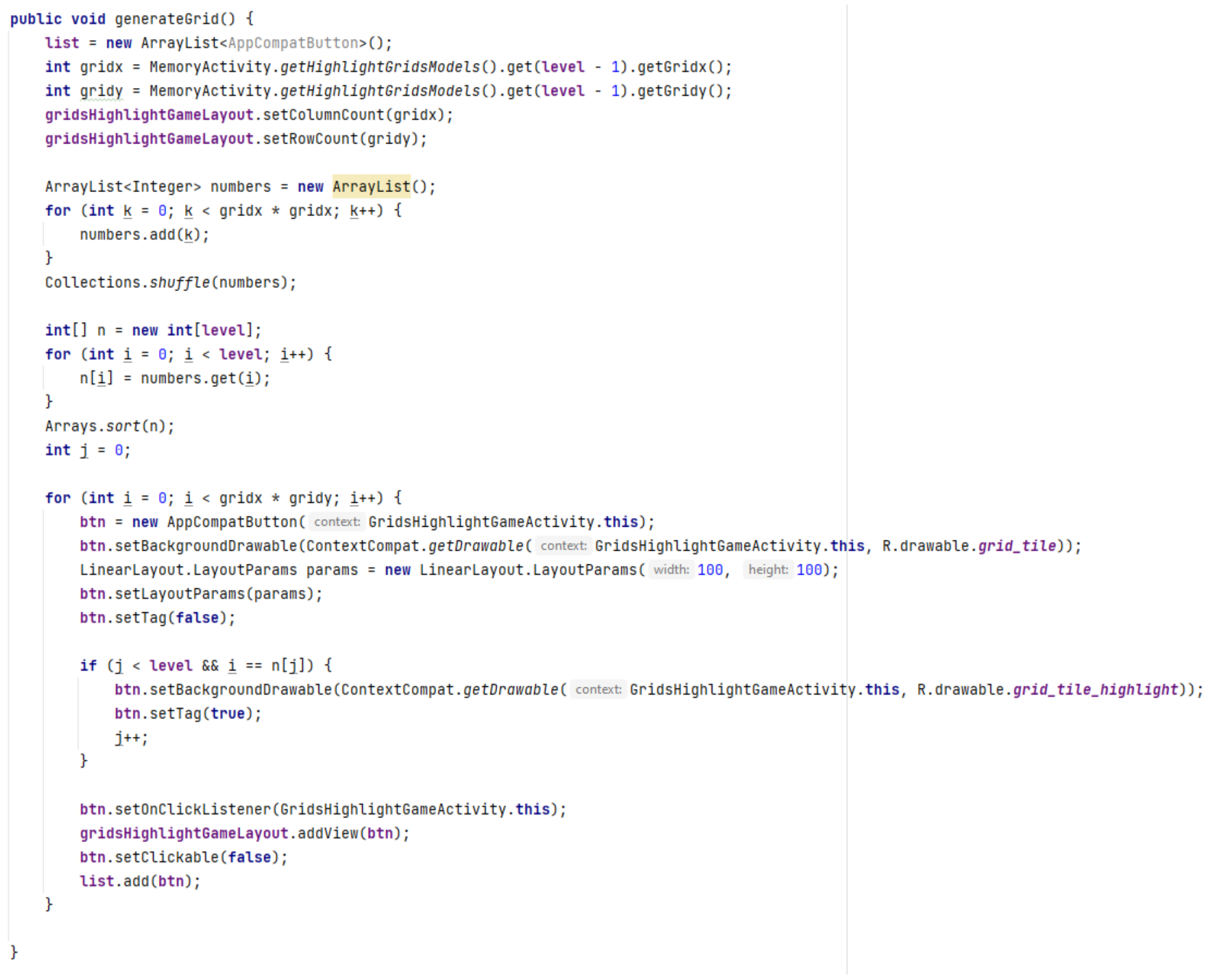


```
public List<FindOperatorModel> findOperatorOfTenModels(BrainTrainDatabase db) {
    List<FindOperatorModel> returnList = new ArrayList<>();
    int level, option, time, point, completeStatus;
    SQLiteDatabase sqLiteDatabase = db.getReadableDatabase();
    String query = "select * from math_game_two_multiple_of_ten";
    Cursor cursor = sqLiteDatabase.rawQuery(query,  selectionArgs: null);
    if (cursor.moveToFirst()) {
        do {
            level = cursor.getInt( columnIndex: 0);
            option = cursor.getInt( columnIndex: 1);
            time = cursor.getInt( columnIndex: 2);
            point = cursor.getInt( columnIndex: 3);
            completeStatus = cursor.getInt( columnIndex: 4);
            returnList.add(new FindOperatorModel(level, option, time, point, completeStatus));
        } while (cursor.moveToNext());
    }
    cursor.close();
    sqLiteDatabase.close();
    return returnList;
}
```

Figure III.3: Query data from database

3.1.1. Memory games:

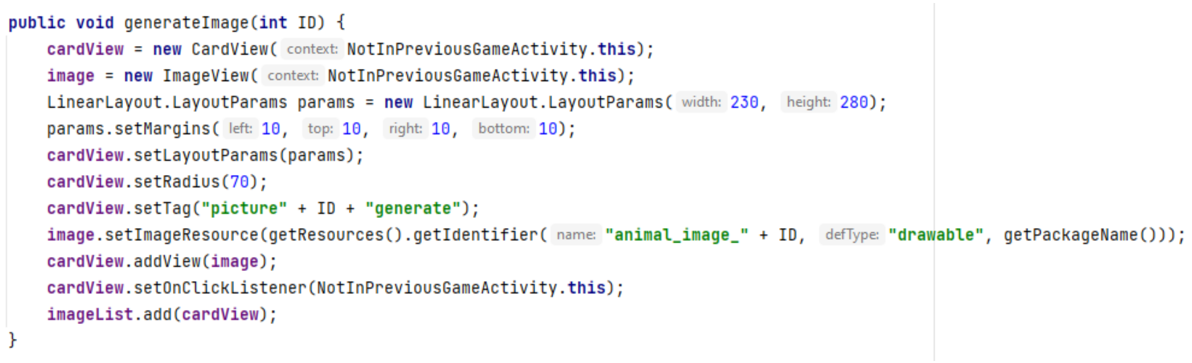
Memory game one:



```
public void generateGrid() {
    list = new ArrayList<AppCompatButton>();
    int gridx = MemoryActivity.getHighlightGridsModels().get(level - 1).getGridx();
    int gridy = MemoryActivity.getHighlightGridsModels().get(level - 1).getGridy();
    gridsHighlightGameLayout.setColumnCount(gridx);
    gridsHighlightGameLayout.setRowCount(gridy);

    ArrayList<Integer> numbers = new ArrayList();
    for (int k = 0; k < gridx * gridx; k++) {
        numbers.add(k);
    }
    Collections.shuffle(numbers);

    int[] n = new int[level];
    for (int i = 0; i < level; i++) {
        n[i] = numbers.get(i);
    }
    Arrays.sort(n);
    int j = 0;

    for (int i = 0; i < gridx * gridy; i++) {
        btn = new AppCompatButton( context: GridsHighlightGameActivity.this);
        btn.setBackgroundDrawable(ContextCompat.getDrawable( context: GridsHighlightGameActivity.this, R.drawable.grid_tile));
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams( width: 100,  height: 100);
        btn.setLayoutParams(params);
        btn.setTag(false);

        if (j < level && i == n[j]) {
            btn.setBackgroundDrawable(ContextCompat.getDrawable( context: GridsHighlightGameActivity.this, R.drawable.grid_tile_highlight));
            btn.setTag(true);
            j++;
        }

        btn.setOnClickListener(GridsHighlightGameActivity.this);
        gridsHighlightGameLayout.addView(btn);
        btn.setClickable(false);
        list.add(btn);
    }

}
```

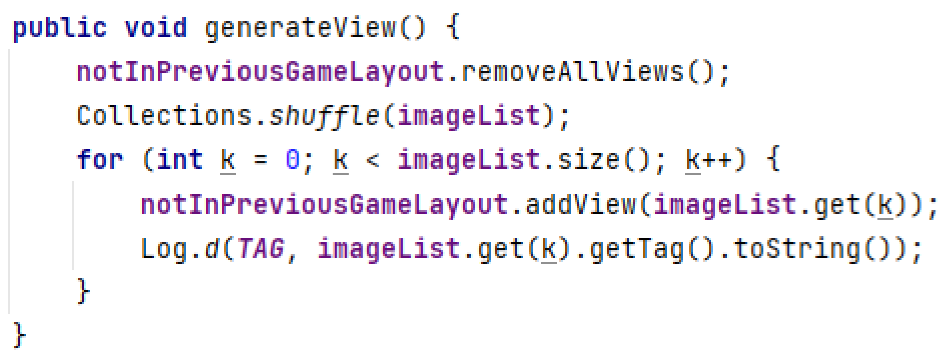
Figure III.4: Generate grid function of game 1 domain Memory

Memory game two



```
public void generateImage(int ID) {
    cardView = new CardView( context: NotInPreviousGameActivity.this);
    image = new ImageView( context: NotInPreviousGameActivity.this);
    LinearLayout.LayoutParams params = new LinearLayout.LayoutParams( width: 230,  height: 280);
    params.setMargins( left: 10,  top: 10,  right: 10,  bottom: 10);
    cardView.setLayoutParams(params);
    cardView.setRadius(70);
    cardView.setTag("picture" + ID + "generate");
    image.setImageResource(getResources().getIdentifier( name: "animal_image_" + ID,  defType: "drawable", getPackageName()));
    cardView.addView(image);
    cardView.setOnClickListener(NotInPreviousGameActivity.this);
    imageList.add(cardView);
}
```

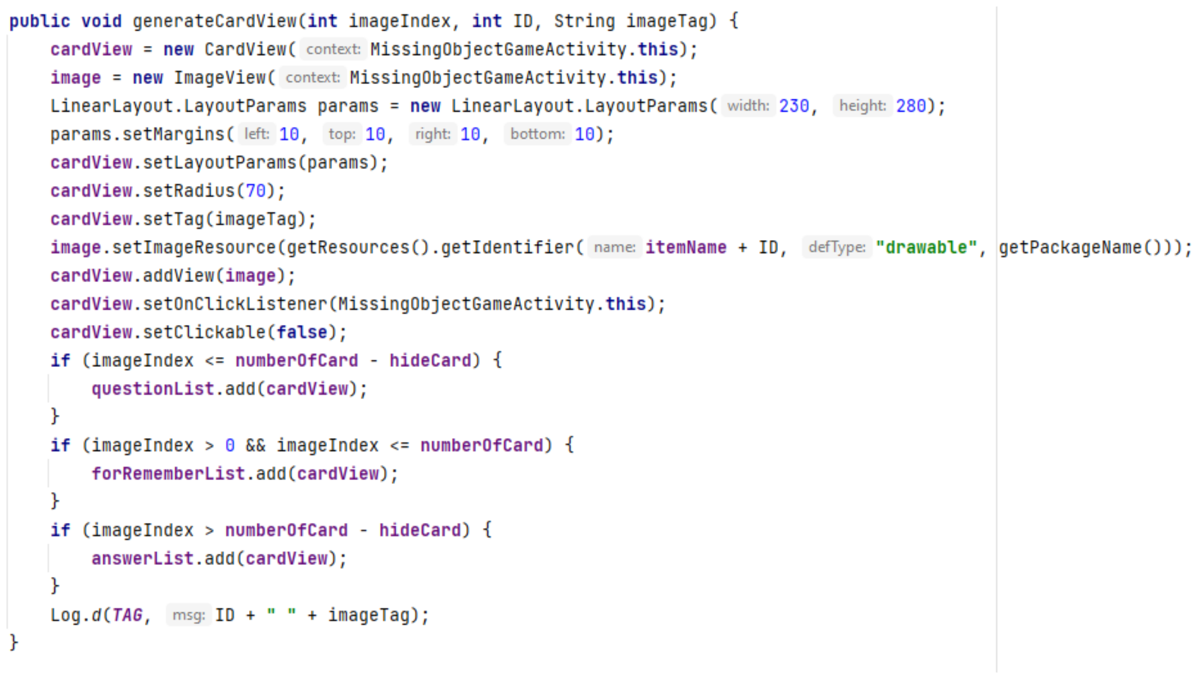
Figure III.5: Generate image for game



```java
public void generateView() {
    notInPreviousGameLayout.removeAllViews();
    Collections.shuffle(imageList);
    for (int k = 0; k < imageList.size(); k++) {
        notInPreviousGameLayout.addView(imageList.get(k));
        Log.d(TAG, imageList.get(k).getTag().toString());
    }
}
```

Figure III.6: Generate game view each time update UI

Memory game three

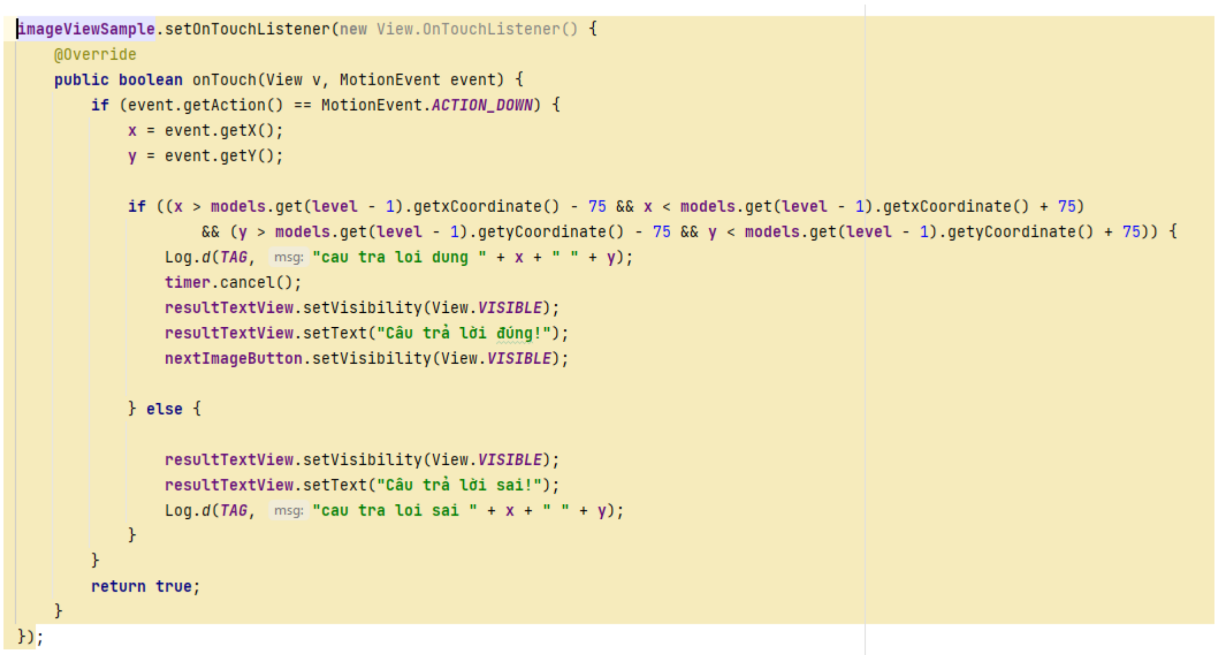


```java
public void generateCardView(int imageIndex, int ID, String imageTag) {
    cardView = new CardView( context: MissingObjectGameActivity.this);
    image = new ImageView( context: MissingObjectGameActivity.this);
    LinearLayout.LayoutParams params = new LinearLayout.LayoutParams( width: 230,  height: 280);
    params.setMargins( left: 10,  top: 10,  right: 10,  bottom: 10);
    cardView.setLayoutParams(params);
    cardView.setRadius(70);
    cardView.setTag(imageTag);
    image.setImageResource(getResources().getIdentifier( name: itemName + ID,  defType: "drawable", getPackageName()));
    cardView.addView(image);
    cardView.setOnClickListener(MissingObjectGameActivity.this);
    cardView.setClickable(false);
    if (imageIndex <= numberOfCard - hideCard) {
        questionList.add(cardView);
    }
    if (imageIndex > 0 && imageIndex <= numberOfCard) {
        forRememberList.add(cardView);
    }
    if (imageIndex > numberOfCard - hideCard) {
        answerList.add(cardView);
    }
    Log.d(TAG,  msg: ID + " " + imageTag);
}
```

Figure III.7: Generate Image for game

4.1.1. Attention games:

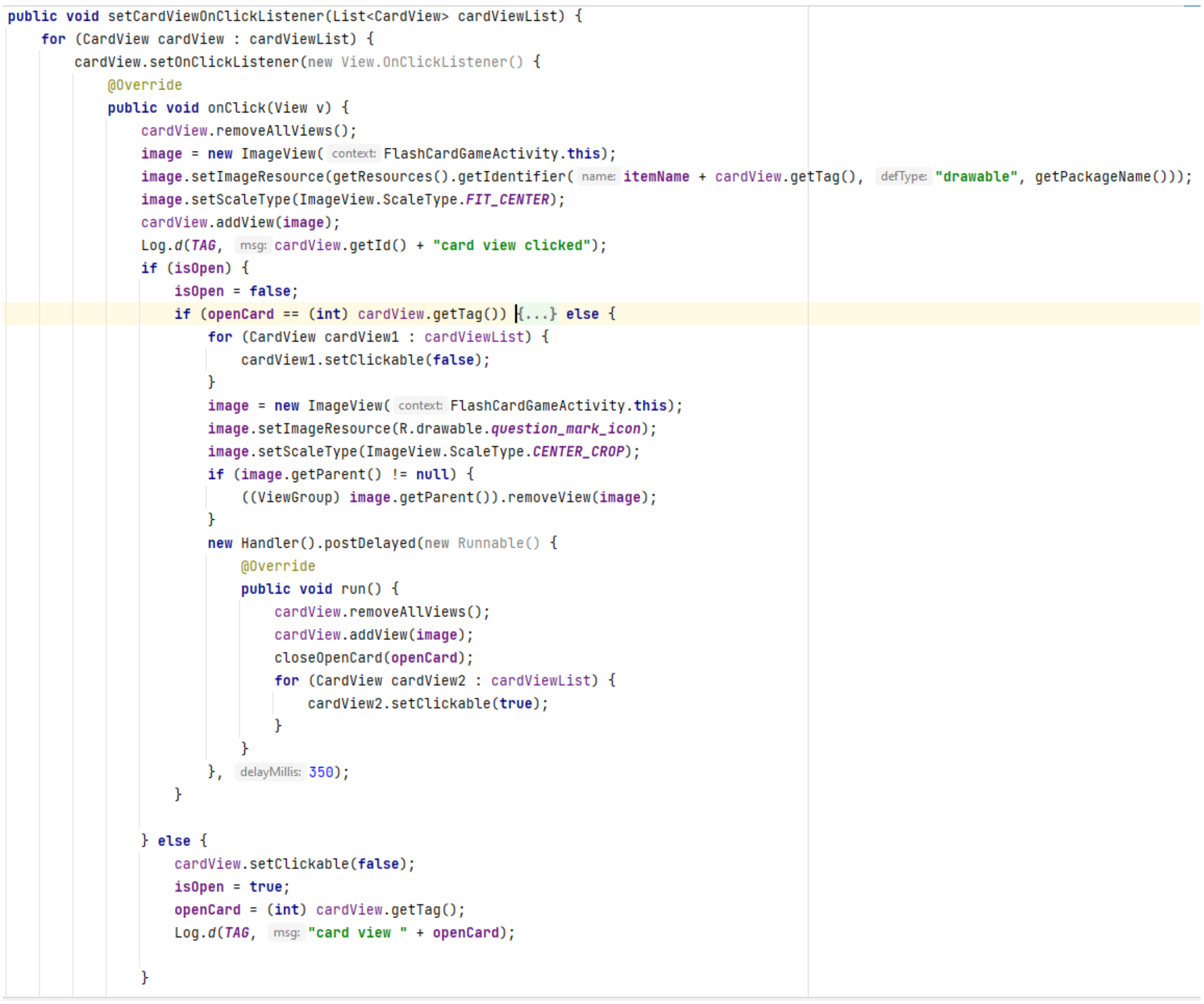
Attention game one



```
imageViewSample.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            x = event.getX();
            y = event.getY();

            if ((x > models.get(level - 1).getxCoordinate() - 75 && x < models.get(level - 1).getxCoordinate() + 75)
                    && (y > models.get(level - 1).getyCoordinate() - 75 && y < models.get(level - 1).getyCoordinate() + 75)) {
                Log.d(TAG,  msg: "cau tra loi dung " + x + " " + y);
                timer.cancel();
                resultTextView.setVisibility(View.VISIBLE);
                resultTextView.setText("Câu trả lời đúng!");
                nextImageButton.setVisibility(View.VISIBLE);

            } else {

                resultTextView.setVisibility(View.VISIBLE);
                resultTextView.setText("Câu trả lời sai!");
                Log.d(TAG,  msg: "cau tra loi sai " + x + " " + y);
            }
        }
        return true;
    }
});
```

Figure III.8: Check click coordinate

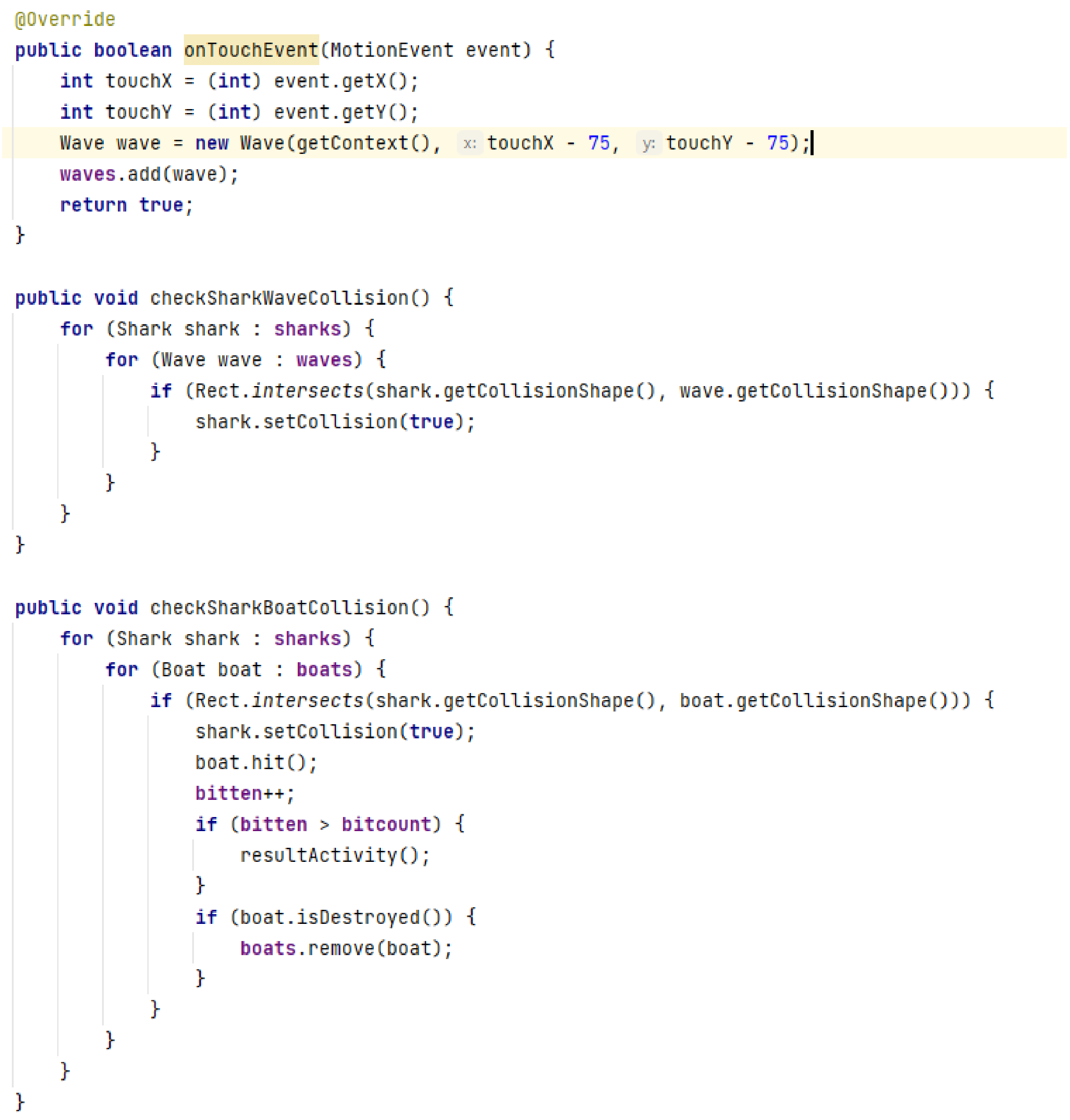
Attention game two



```
public void setCardViewOnClickListener(List<CardView> cardViewList) {
    for (CardView cardView : cardViewList) {
        cardView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                cardView.removeAllViews();
                image = new ImageView( context: FlashCardGameActivity.this);
                image.setImageResource(getResources().getIdentifier( name: itemName + cardView.getTag(),  defType: "drawable", getPackageName()));
                image.setScaleType(ImageView.ScaleType.FIT_CENTER);
                cardView.addView(image);
                Log.d(TAG,  msg: cardView.getId() + "card view clicked");
                if (isOpen) {
                    isOpen = false;
                    if (openCard == (int) cardView.getTag()) {...} else {
                        for (CardView cardView1 : cardViewList) {
                            cardView1.setClickable(false);
                        }
                        image = new ImageView( context: FlashCardGameActivity.this);
                        image.setImageResource(R.drawable.question_mark_icon);
                        image.setScaleType(ImageView.ScaleType.CENTER_CROP);
                        if (image.getParent() != null) {
                            ((ViewGroup) image.getParent()).removeView(image);
                        }
                        new Handler().postDelayed(new Runnable() {
                            @Override
                            public void run() {
                                cardView.removeAllViews();
                                cardView.addView(image);
                                closeOpenCard(openCard);
                                for (CardView cardView2 : cardViewList) {
                                    cardView2.setClickable(true);
                                }
                            }
                        },  delayMillis: 350);
                    }

                } else {
                    cardView.setClickable(false);
                    isOpen = true;
                    openCard = (int) cardView.getTag();
                    Log.d(TAG,  msg: "card view " + openCard);

                }
```

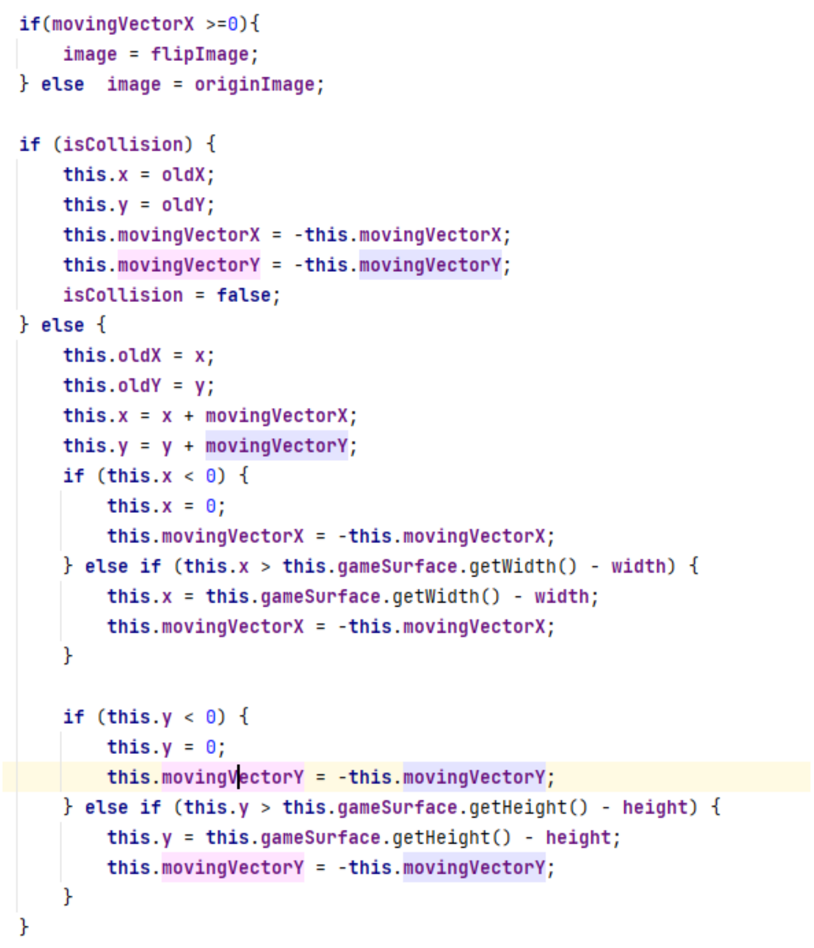
Figure III.9: Set image each time user click

Attention game three



```java
@Override
public boolean onTouchEvent(MotionEvent event) {
    int touchX = (int) event.getX();
    int touchY = (int) event.getY();
    Wave wave = new Wave(getContext(),  x: touchX - 75,  y: touchY - 75);
    waves.add(wave);
    return true;
}

public void checkSharkWaveCollision() {
    for (Shark shark : sharks) {
        for (Wave wave : waves) {
            if (Rect.intersects(shark.getCollisionShape(), wave.getCollisionShape())) {
                shark.setCollision(true);
            }
        }
    }
}

public void checkSharkBoatCollision() {
    for (Shark shark : sharks) {
        for (Boat boat : boats) {
            if (Rect.intersects(shark.getCollisionShape(), boat.getCollisionShape())) {
                shark.setCollision(true);
                boat.hit();
                bitten++;
                if (bitten > bitcount) {
                    resultActivity();
                }
                if (boat.isDestroyed()) {
                    boats.remove(boat);
                }
            }
        }
    }
}
```

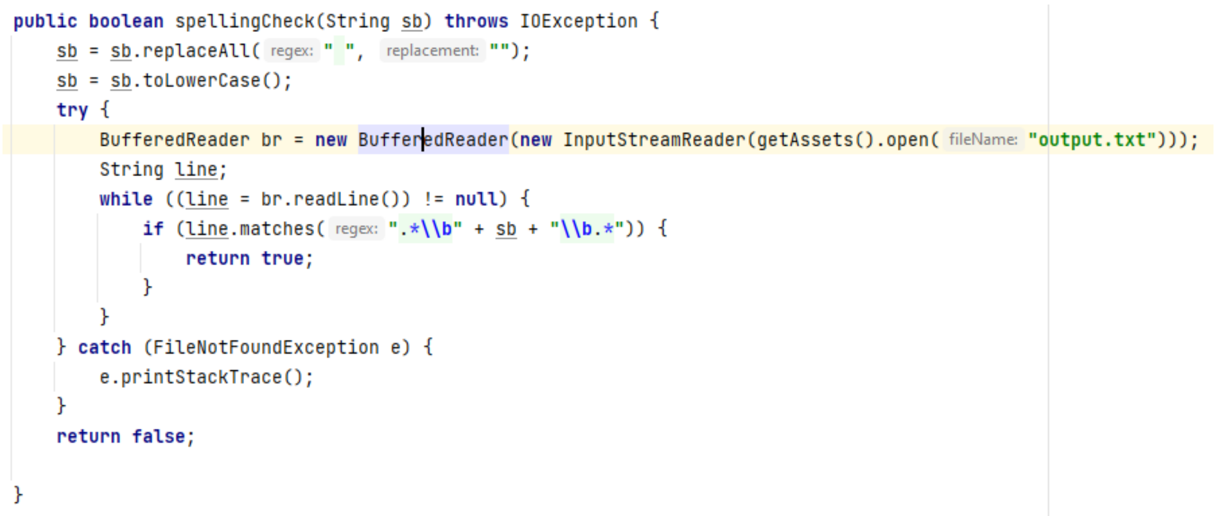
Figure III.10: Generate wave and check collision



```
if(movingVectorX >=0){
    image = flipImage;
} else  image = originImage;

if (isCollision) {
    this.x = oldX;
    this.y = oldY;
    this.movingVectorX = -this.movingVectorX;
    this.movingVectorY = -this.movingVectorY;
    isCollision = false;
} else {
    this.oldX = x;
    this.oldY = y;
    this.x = x + movingVectorX;
    this.y = y + movingVectorY;
    if (this.x < 0) {
        this.x = 0;
        this.movingVectorX = -this.movingVectorX;
    } else if (this.x > this.gameSurface.getWidth() - width) {
        this.x = this.gameSurface.getWidth() - width;
        this.movingVectorX = -this.movingVectorX;
    }

    if (this.y < 0) {
        this.y = 0;
        this.movingVectorY = -this.movingVectorY;
    } else if (this.y > this.gameSurface.getHeight() - height) {
        this.y = this.gameSurface.getHeight() - height;
        this.movingVectorY = -this.movingVectorY;
    }
}
```

Figure III.11: Update shark location function

4.1.2. Language games:

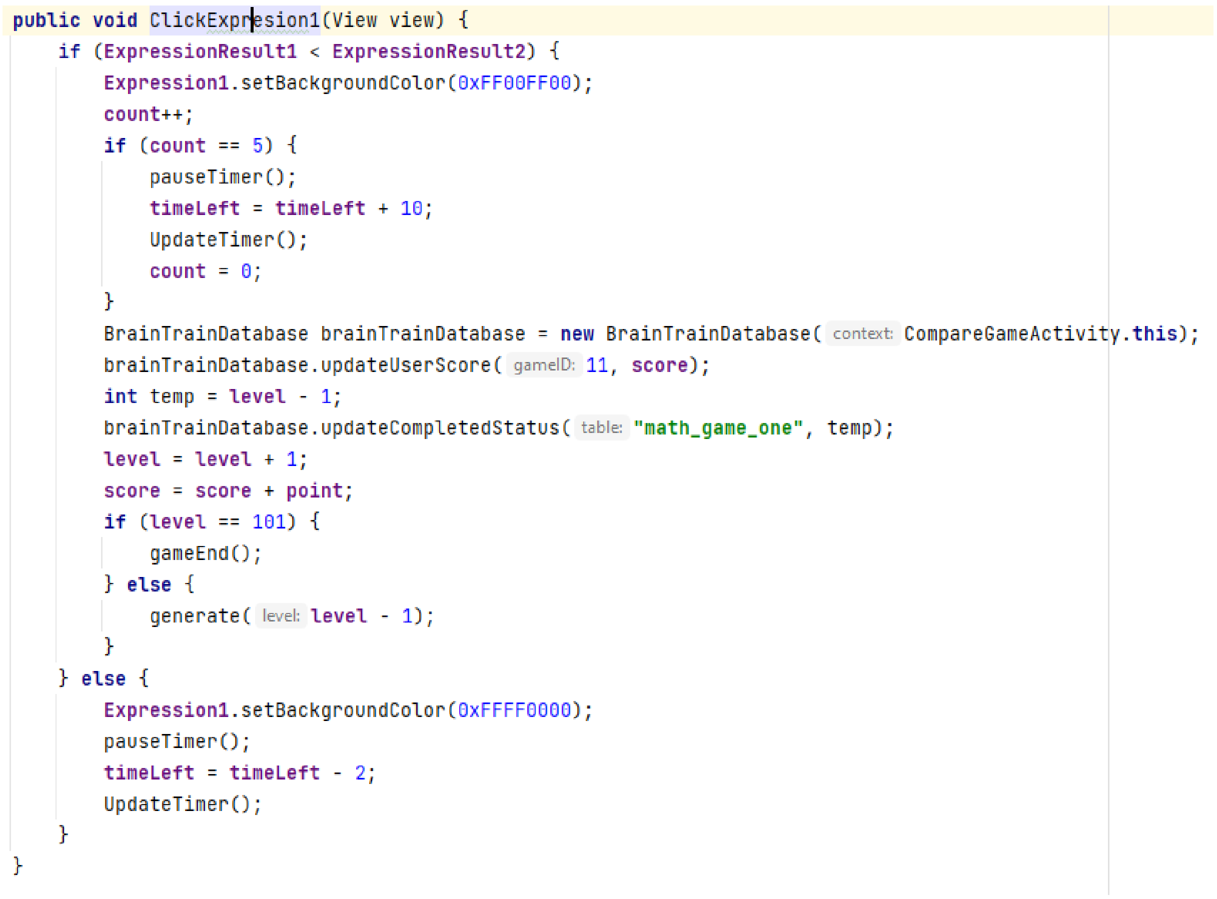


```
public boolean spellingCheck(String sb) throws IOException {
    sb = sb.replaceAll( regex: " ",  replacement: "");
    sb = sb.toLowerCase();
    try {
        BufferedReader br = new BufferedReader(new InputStreamReader(getAssets().open( fileName: "output.txt")));
        String line;
        while ((line = br.readLine()) != null) {
            if (line.matches( regex: ".*\\b" + sb + "\\b.*")) {
                return true;
            }
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    return false;

}
```

Figure III.12: Check spelling of user input word

4.1.3. Math games:

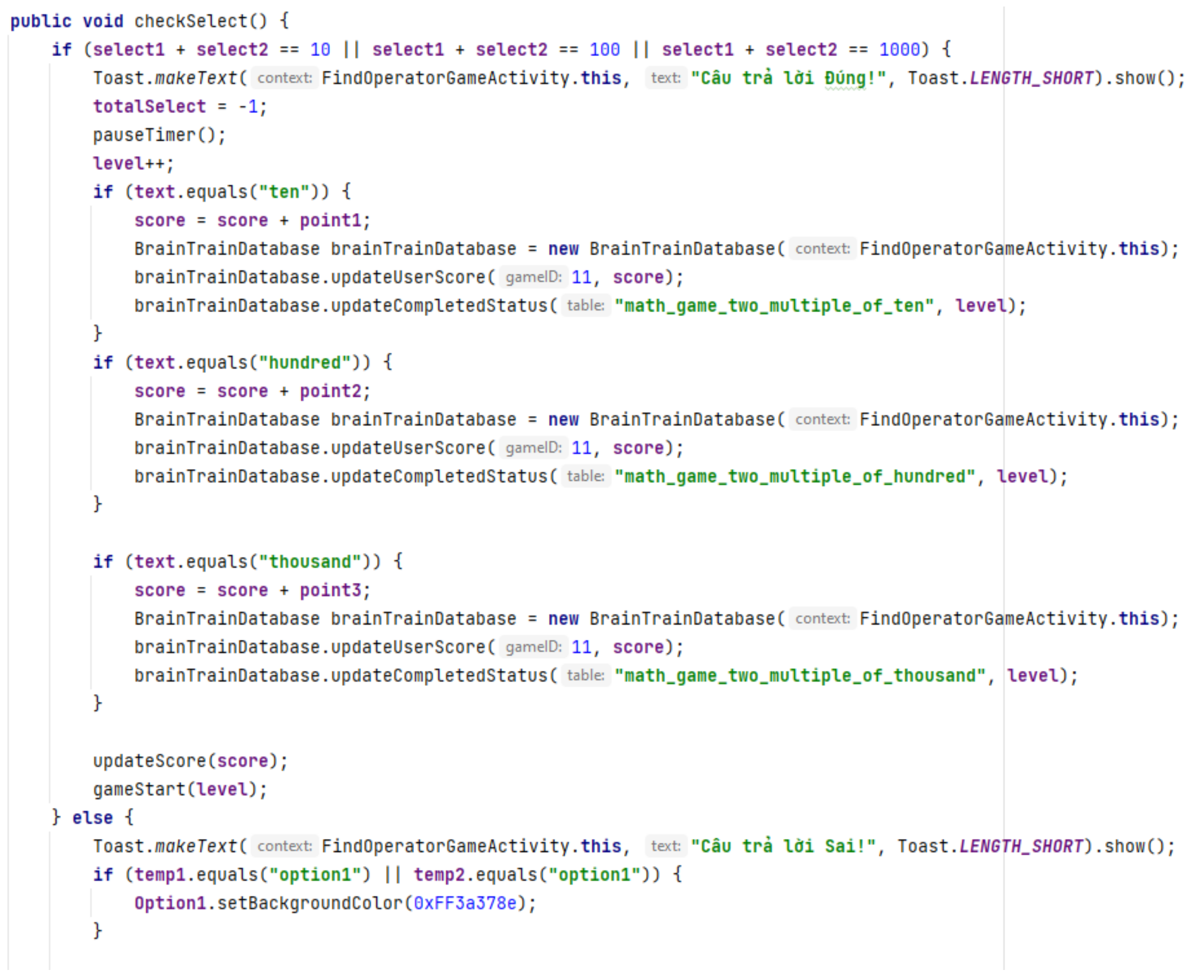
Math game one



```
public void ClickExpresion1(View view) {
    if (ExpressionResult1 < ExpressionResult2) {
        Expression1.setBackgroundColor(0xFF00FF00);
        count++;
        if (count == 5) {
            pauseTimer();
            timeLeft = timeLeft + 10;
            UpdateTimer();
            count = 0;
        }
        BrainTrainDatabase brainTrainDatabase = new BrainTrainDatabase( context: CompareGameActivity.this);
        brainTrainDatabase.updateUserScore( gameID: 11, score);
        int temp = level - 1;
        brainTrainDatabase.updateCompletedStatus( table: "math_game_one", temp);
        level = level + 1;
        score = score + point;
        if (level == 101) {
            gameEnd();
        } else {
            generate( level: level - 1);
        }
    } else {
        Expression1.setBackgroundColor(0xFFFF0000);
        pauseTimer();
        timeLeft = timeLeft - 2;
        UpdateTimer();
    }
}
```

Figure III.13: Check right expression

Math game two



```
public void checkSelect() {
    if (select1 + select2 == 10 || select1 + select2 == 100 || select1 + select2 == 1000) {
        Toast.makeText( context: FindOperatorGameActivity.this,  text: "Câu trả lời Đúng!", Toast.LENGTH_SHORT).show();
        totalSelect = -1;
        pauseTimer();
        level++;
        if (text.equals("ten")) {
            score = score + point1;
            BrainTrainDatabase brainTrainDatabase = new BrainTrainDatabase( context: FindOperatorGameActivity.this);
            brainTrainDatabase.updateUserScore( gameID: 11, score);
            brainTrainDatabase.updateCompletedStatus( table: "math_game_two_multiple_of_ten", level);
        }
        if (text.equals("hundred")) {
            score = score + point2;
            BrainTrainDatabase brainTrainDatabase = new BrainTrainDatabase( context: FindOperatorGameActivity.this);
            brainTrainDatabase.updateUserScore( gameID: 11, score);
            brainTrainDatabase.updateCompletedStatus( table: "math_game_two_multiple_of_hundred", level);
        }

        if (text.equals("thousand")) {
            score = score + point3;
            BrainTrainDatabase brainTrainDatabase = new BrainTrainDatabase( context: FindOperatorGameActivity.this);
            brainTrainDatabase.updateUserScore( gameID: 11, score);
            brainTrainDatabase.updateCompletedStatus( table: "math_game_two_multiple_of_thousand", level);
        }

        updateScore(score);
        gameStart(level);
    } else {
        Toast.makeText( context: FindOperatorGameActivity.this,  text: "Câu trả lời Sai!", Toast.LENGTH_SHORT).show();
        if (temp1.equals("option1") || temp2.equals("option1")) {
            Option1.setBackgroundColor(0xFF3a378e);
        }
```

Figure III.14: Check user seleck result

4.2. Results

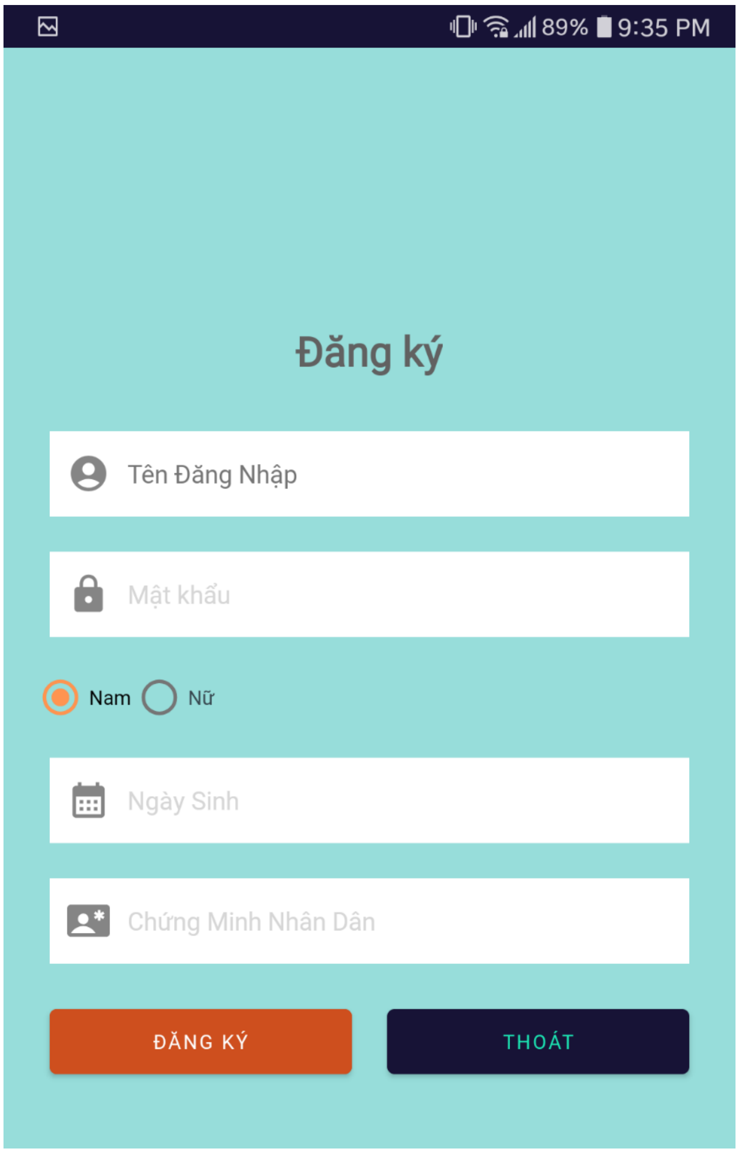


Figure III.15: Sign up page

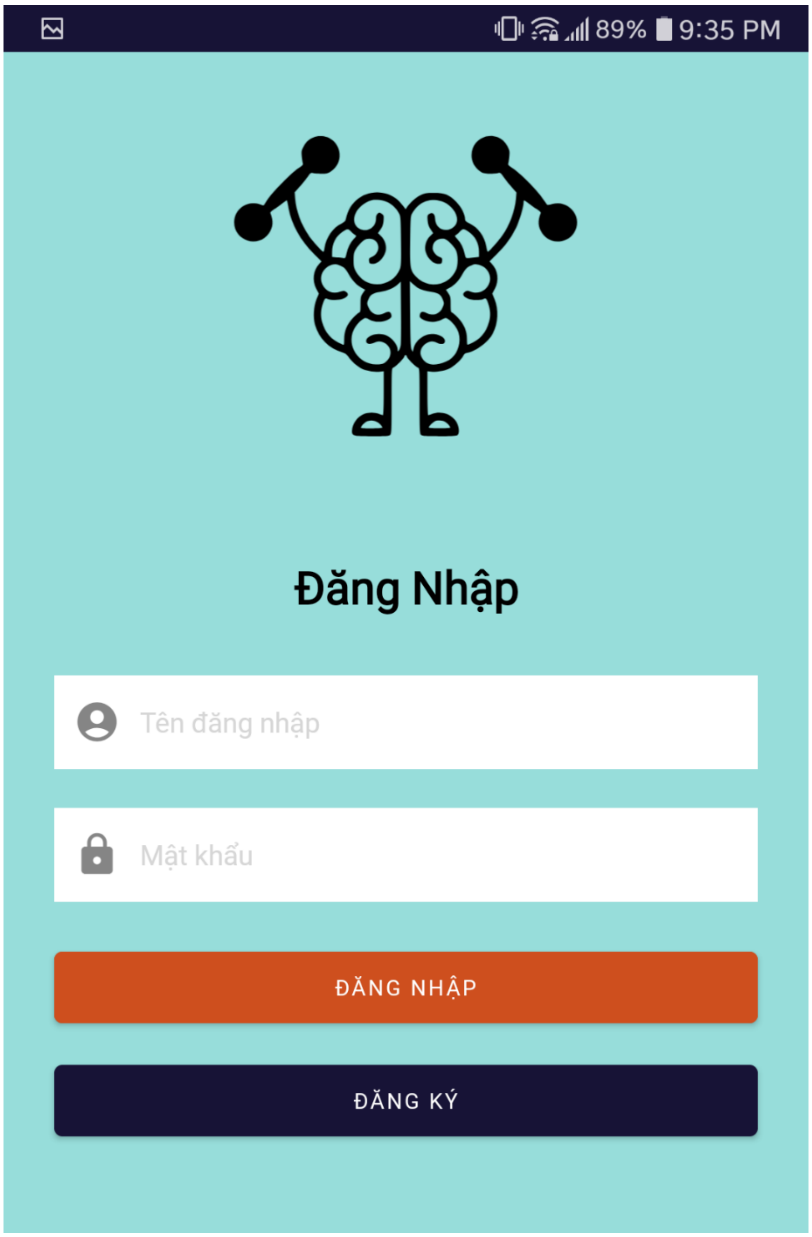


Figure III.16: Sign in page

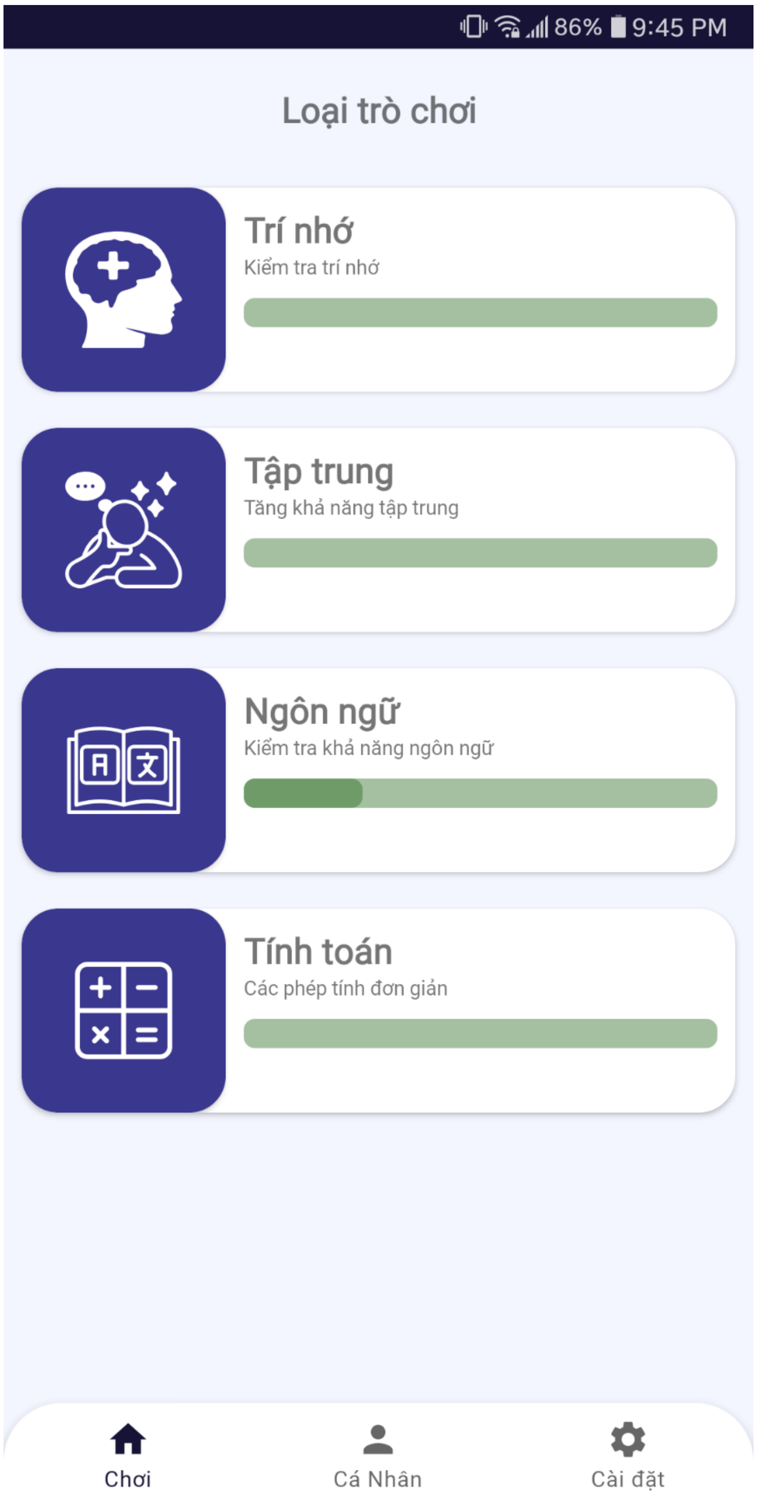


Figure III.17: Home page

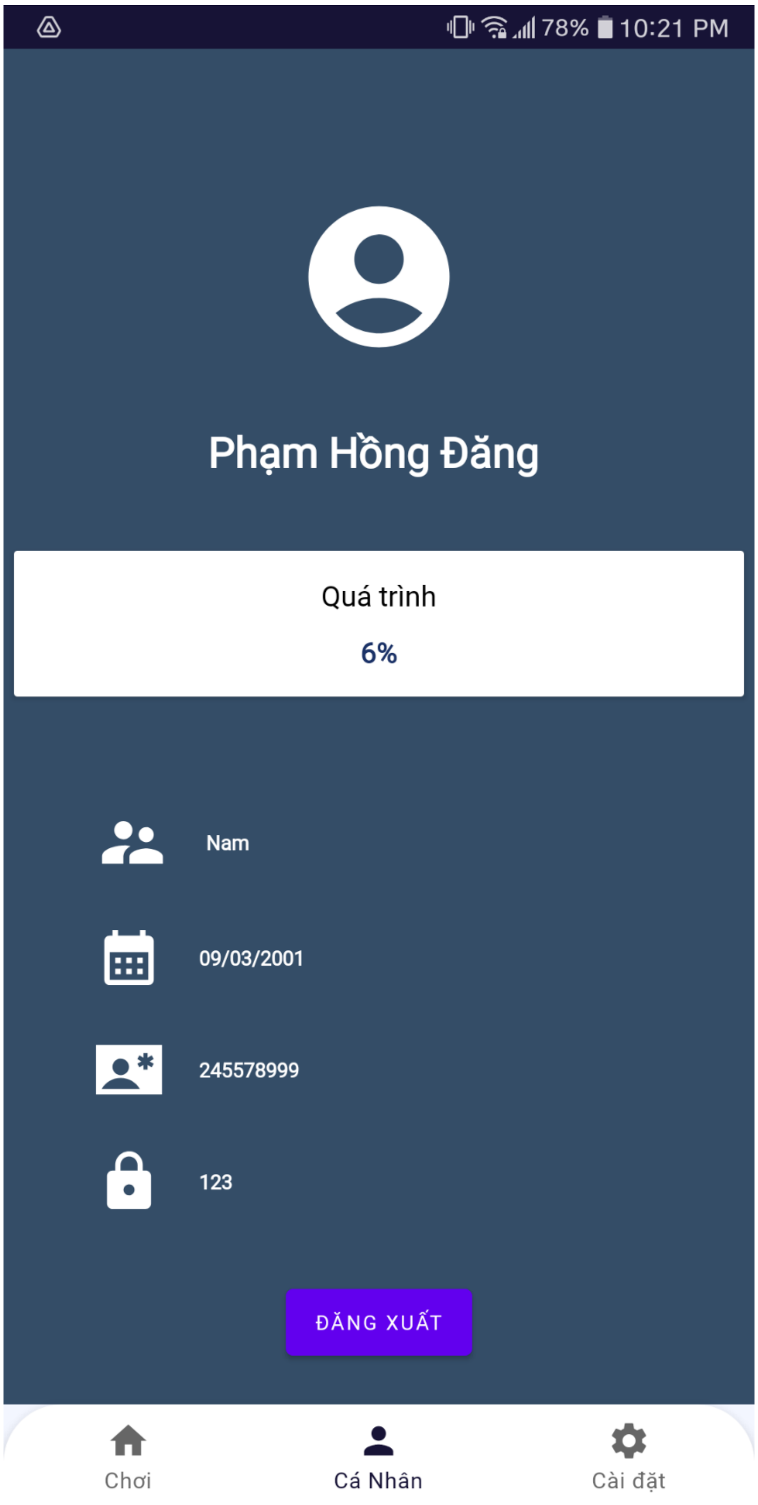


Figure III.18: Profile page

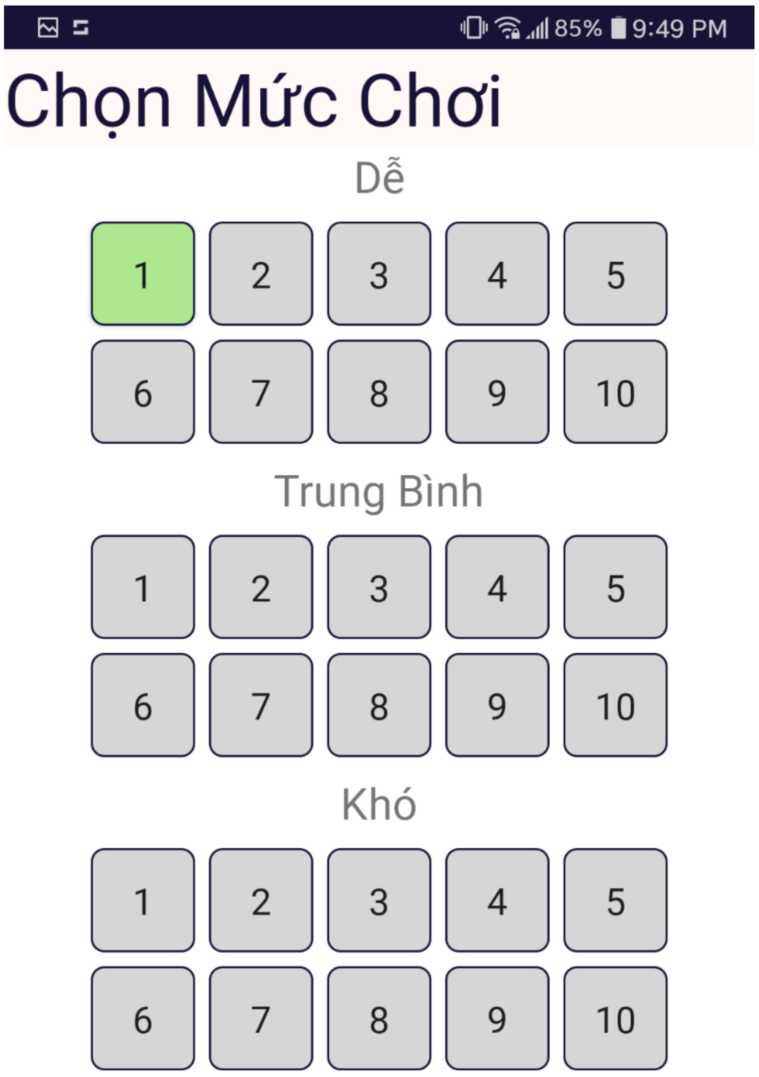
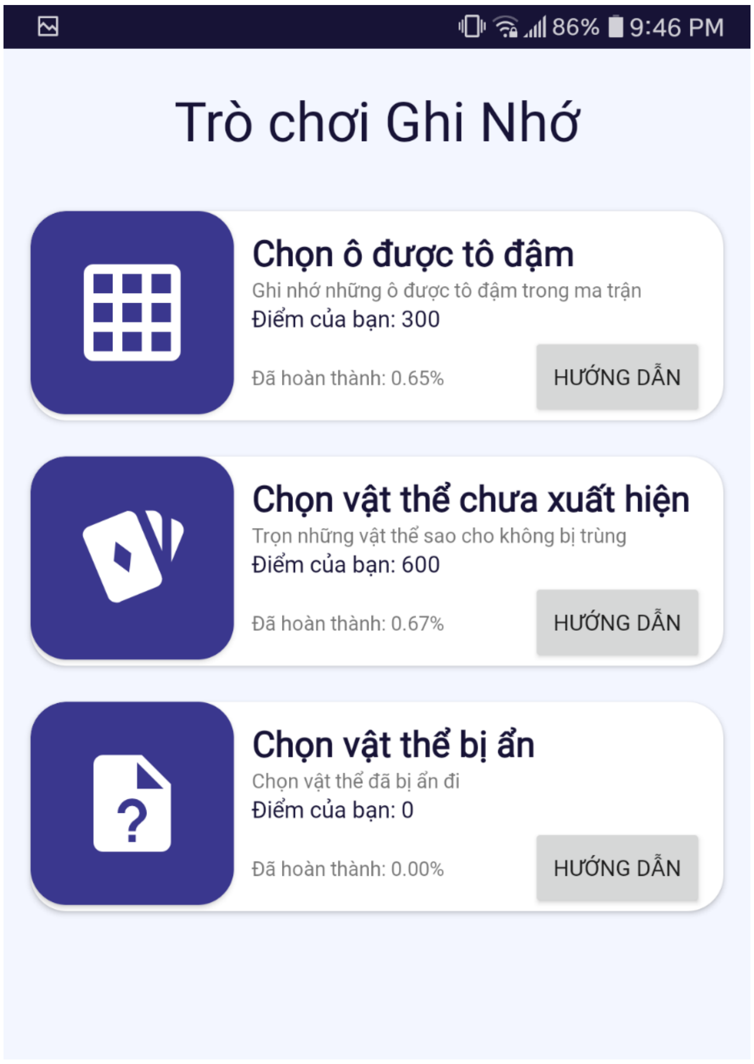


Figure III.19: Select level page

4.2.1. Memory games


9:46 PM 86%

Trò chơi Ghi Nhớ

Chọn ô được tô đậm
Ghi nhớ những ô được tô đậm trong ma trận
Điểm của bạn: 300
Đã hoàn thành: 0.65%

Chọn vật thể chưa xuất hiện
Trọn những vật thể sao cho không bị trùng
Điểm của bạn: 600
Đã hoàn thành: 0.67%

Chọn vật thể bị ẩn
Chọn vật thể đã bị ẩn đi
Điểm của bạn: 0
Đã hoàn thành: 0.00%

Figure III.20: Memory game menu

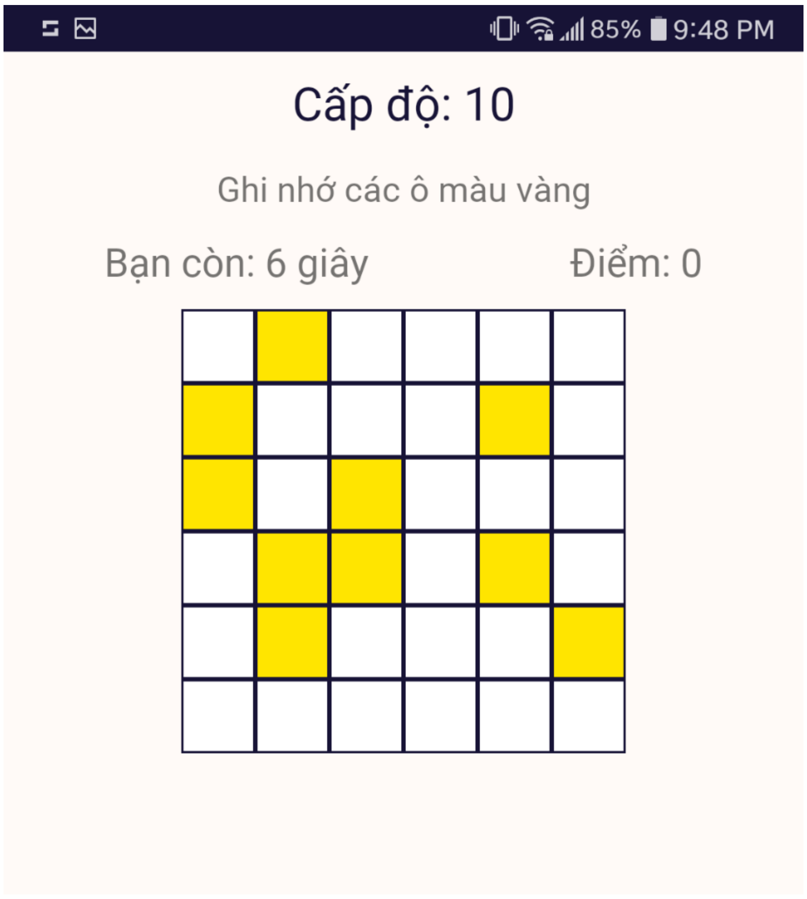


Figure III.21: Memory game one UI

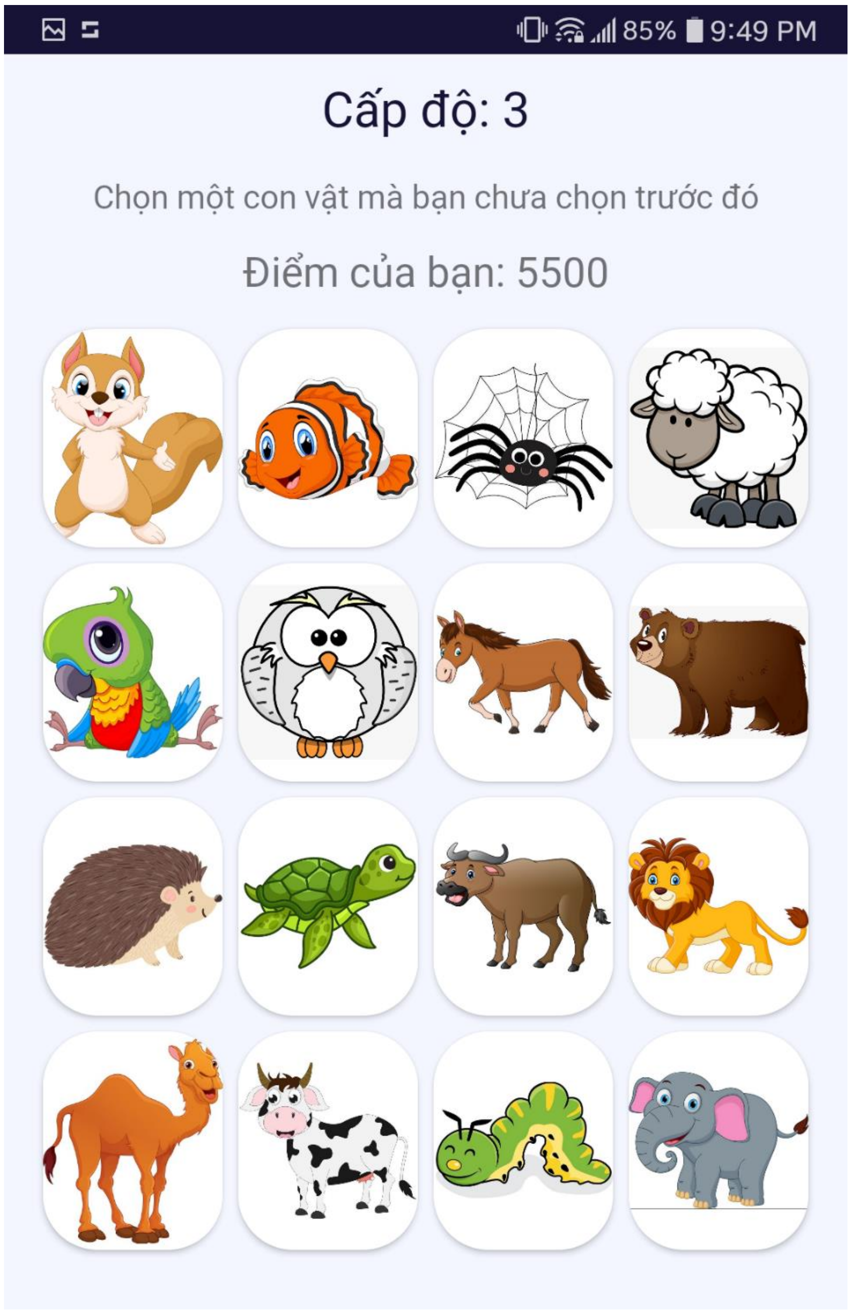


Figure III.22: Memory game two UI

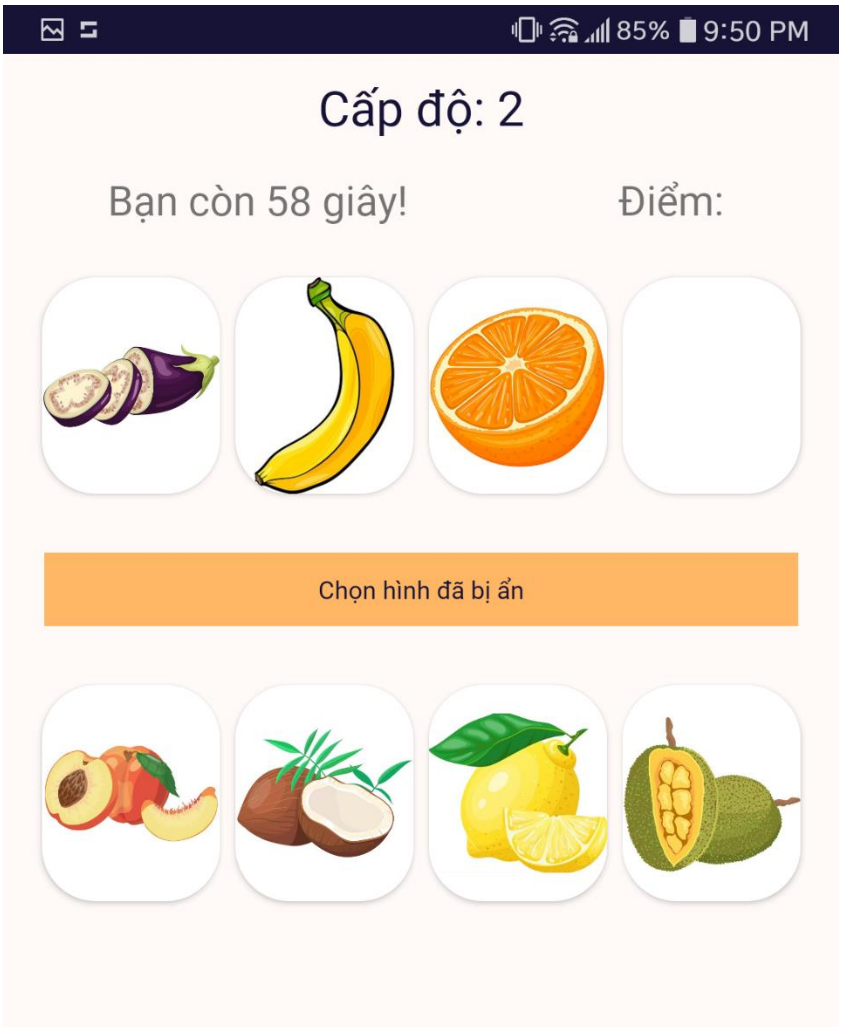


Figure III.23: Memory game three UI

4.2.2. Attention games

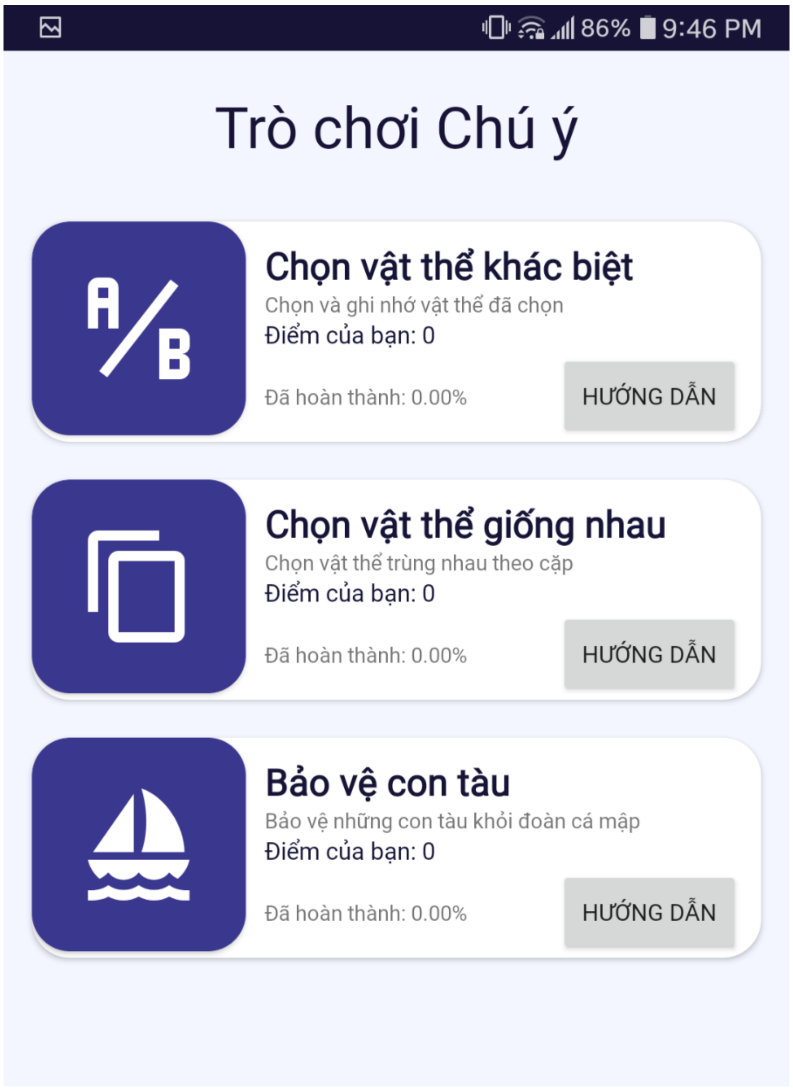


Figure III.24: Attention game menu

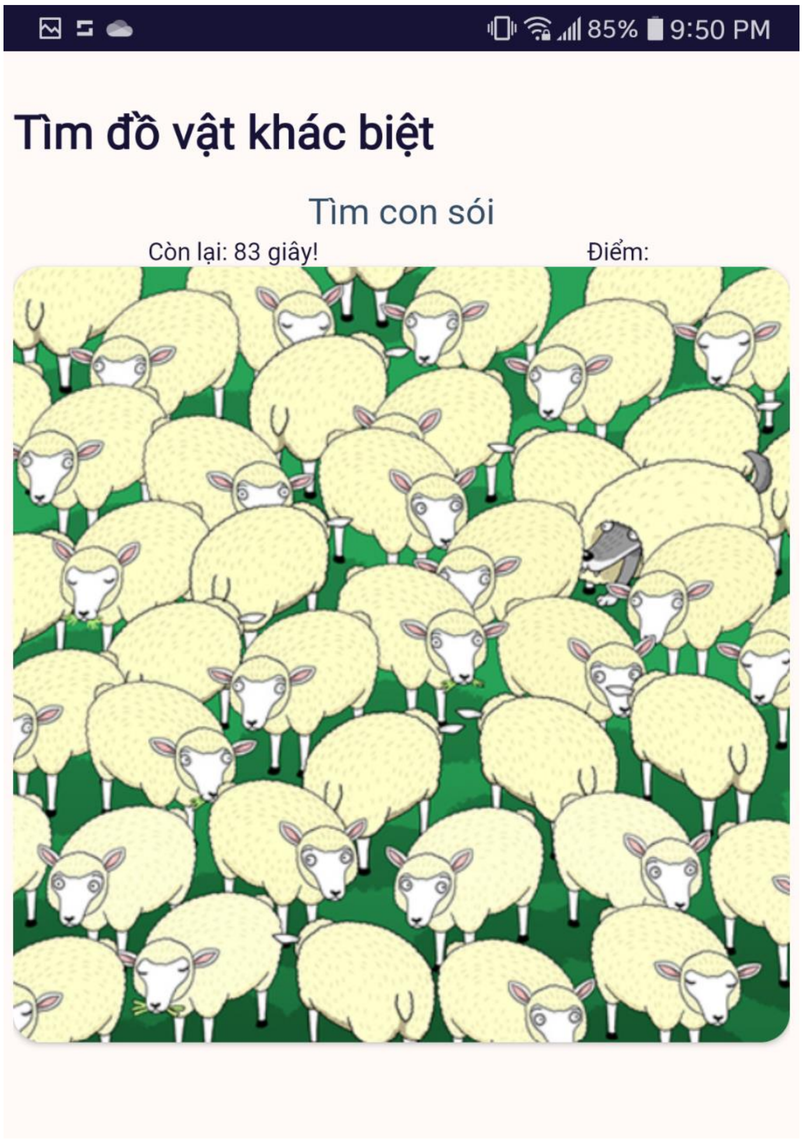
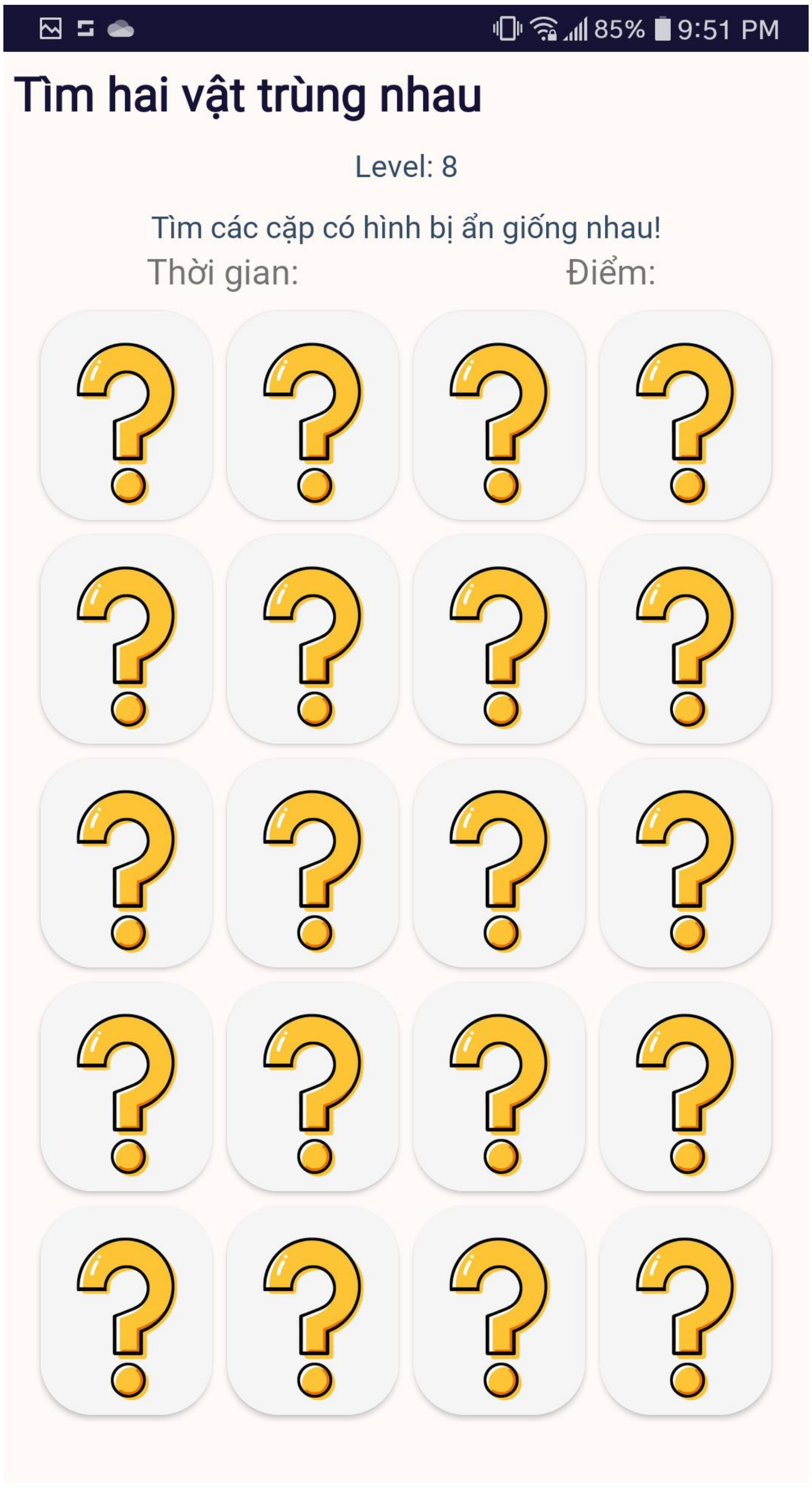


Figure III.25: Attention game one UI



Tìm hai vật trùng nhau

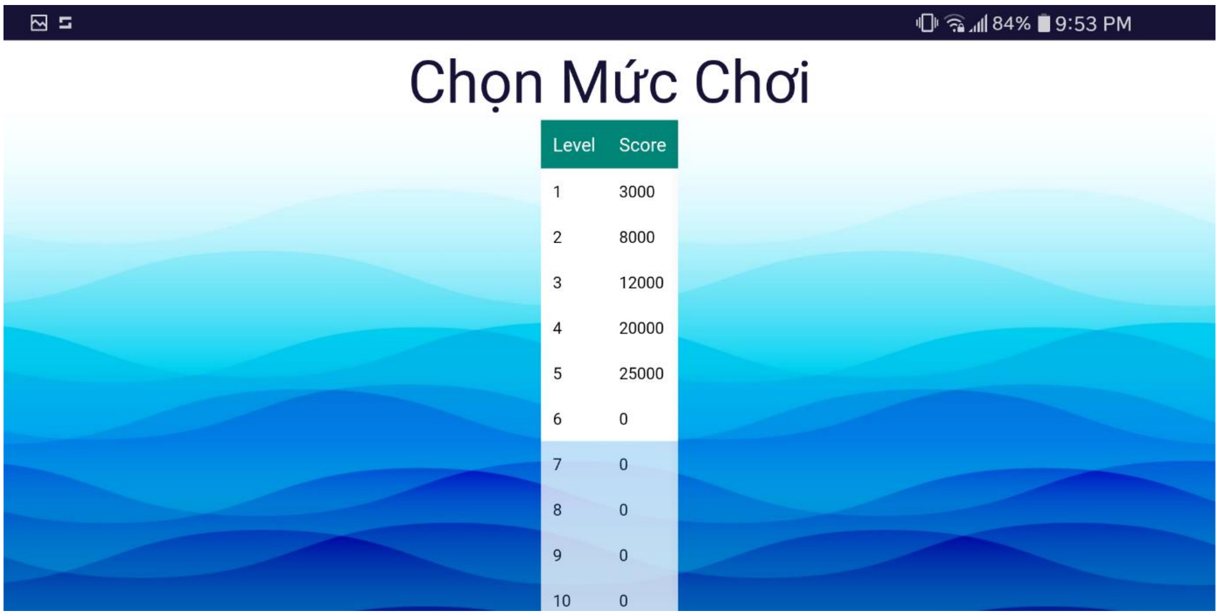
Level: 8

Tìm các cặp có hình bị ẩn giống nhau!

Thời gian:

Điểm:

Figure III.26: Attention game two UI



Chọn Mức Chơi

Level	Score
1	3000
2	8000
3	12000
4	20000
5	25000
6	0
7	0
8	0
9	0
10	0

Figure III.27: Attention game three Level menu

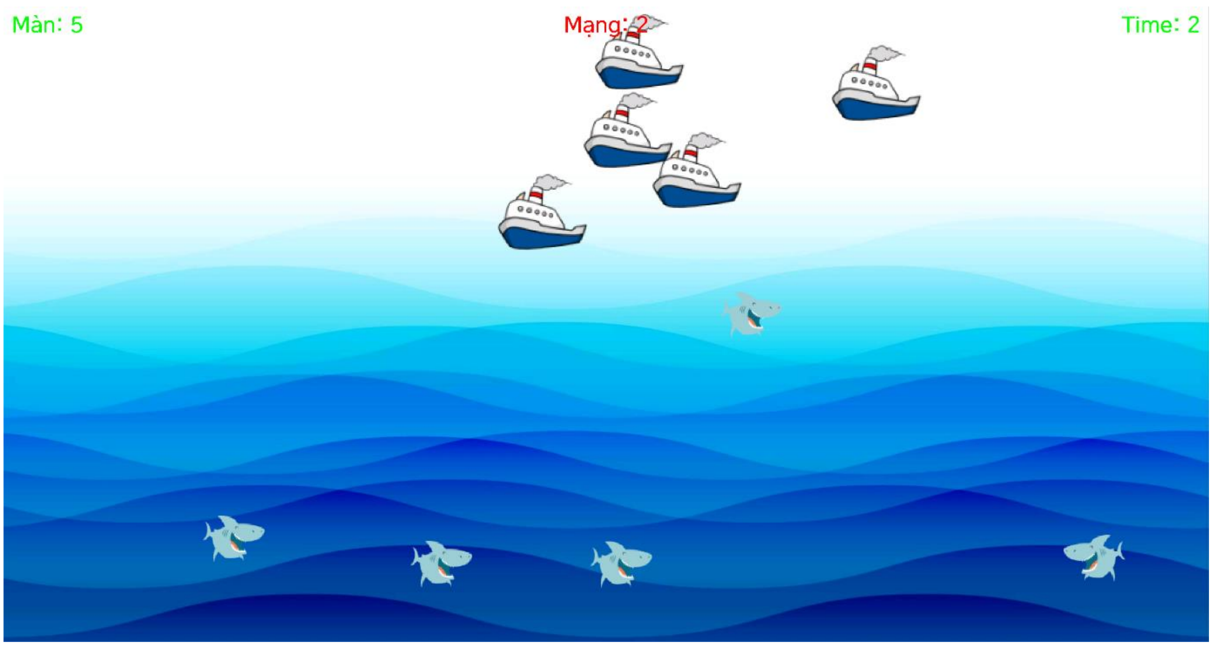


Figure III.28: Attention game three UI

4.2.3. Language games

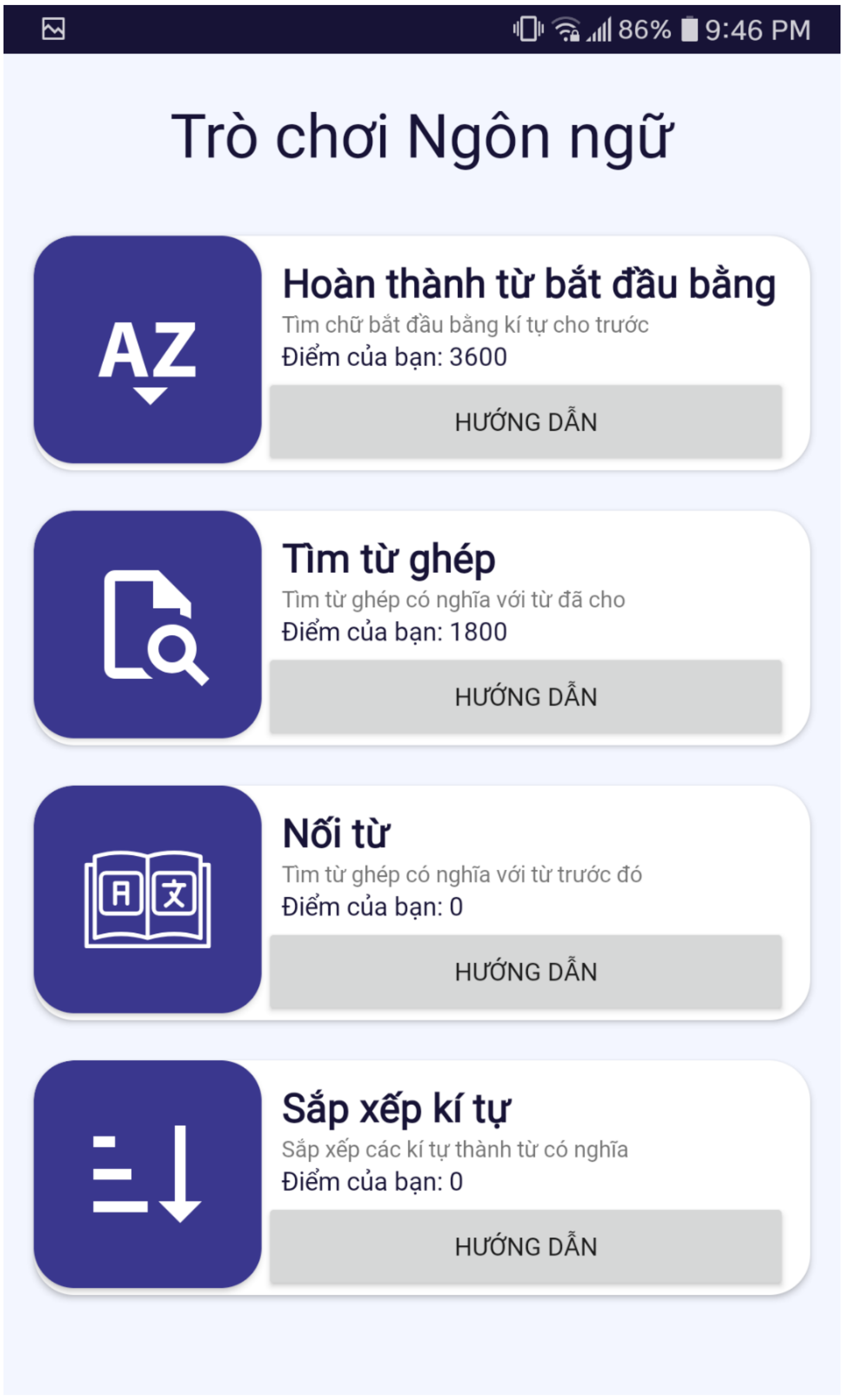
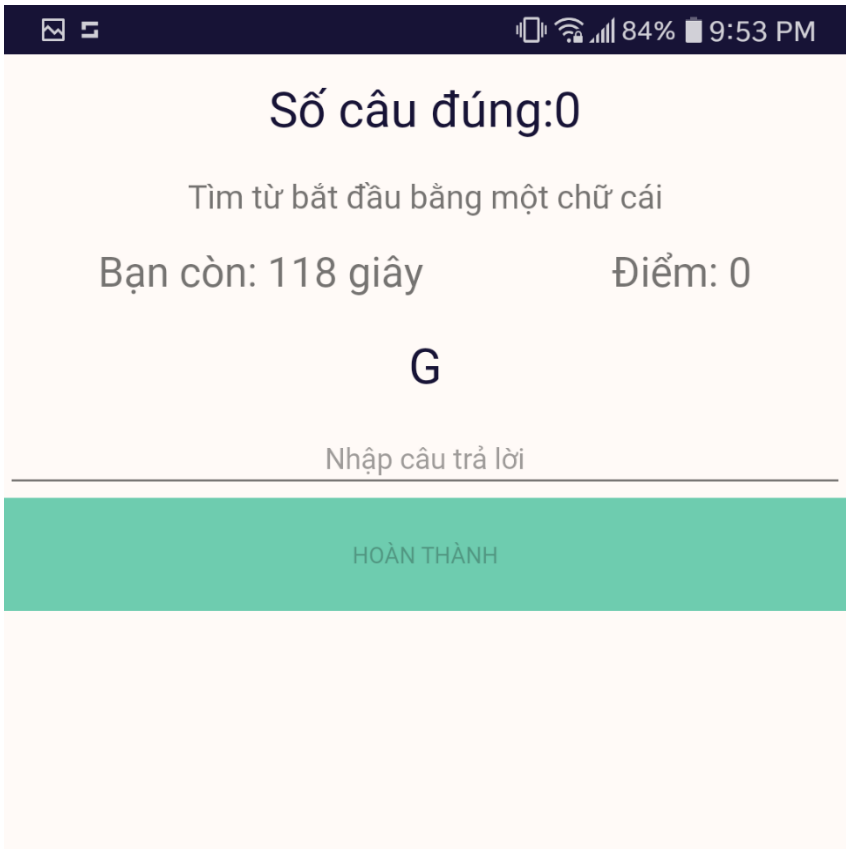

86% 9:46 PM
Trò chơi Ngôn ngữ
Hoàn thành từ bắt đầu bằng
Tìm chữ bắt đầu bằng kí tự cho trước
Điểm của bạn: 3600
HƯỚNG DẪN
Tìm từ ghép
Tìm từ ghép có nghĩa với từ đã cho
Điểm của bạn: 1800
HƯỚNG DẪN
Nối từ
Tìm từ ghép có nghĩa với từ trước đó
Điểm của bạn: 0
HƯỚNG DẪN
Sắp xếp kí tự
Sắp xếp các kí tự thành từ có nghĩa
Điểm của bạn: 0
HƯỚNG DẪN

Figure III.29: Language game menu


84% 9:53 PM

Số câu đúng:0

Tìm từ bắt đầu bằng một chữ cái

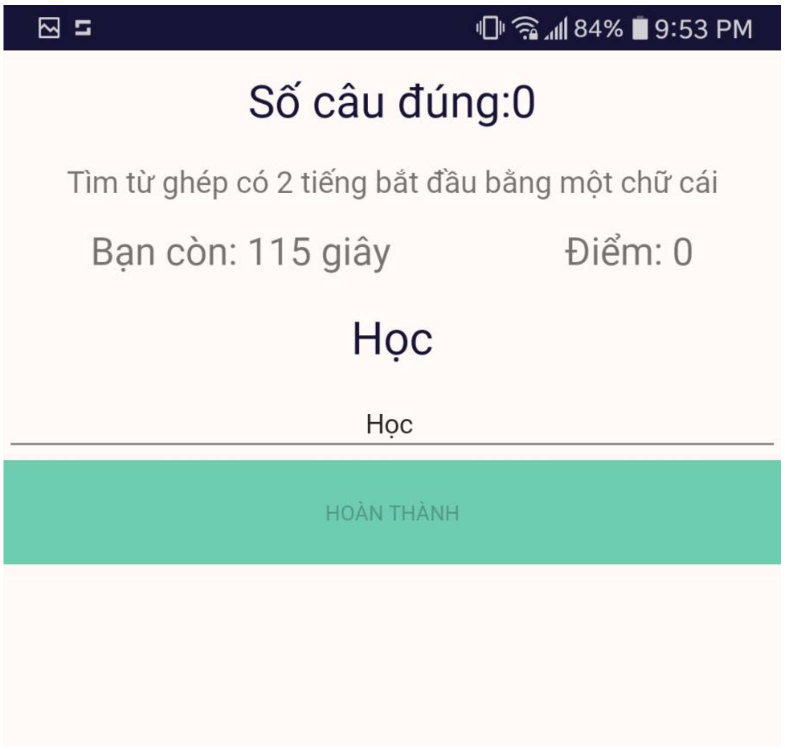
Bạn còn: 118 giây

Điểm: 0

G

Nhập câu trả lời

Figure III.30: Language game one UI


84% 9:53 PM

Số câu đúng:0

Tìm từ ghép có 2 tiếng bắt đầu bằng một chữ cái

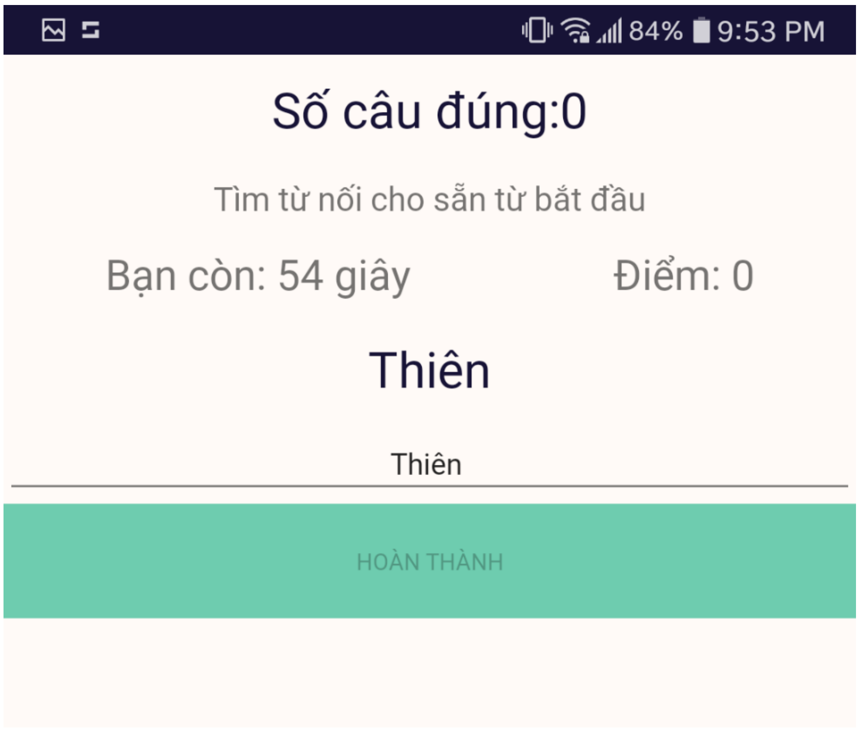
Bạn còn: 115 giây

Điểm: 0

Học

Học

Figure III.31: Language game two UI


84% 9:53 PM

Số câu đúng:0

Tìm từ nối cho sẵn từ bắt đầu

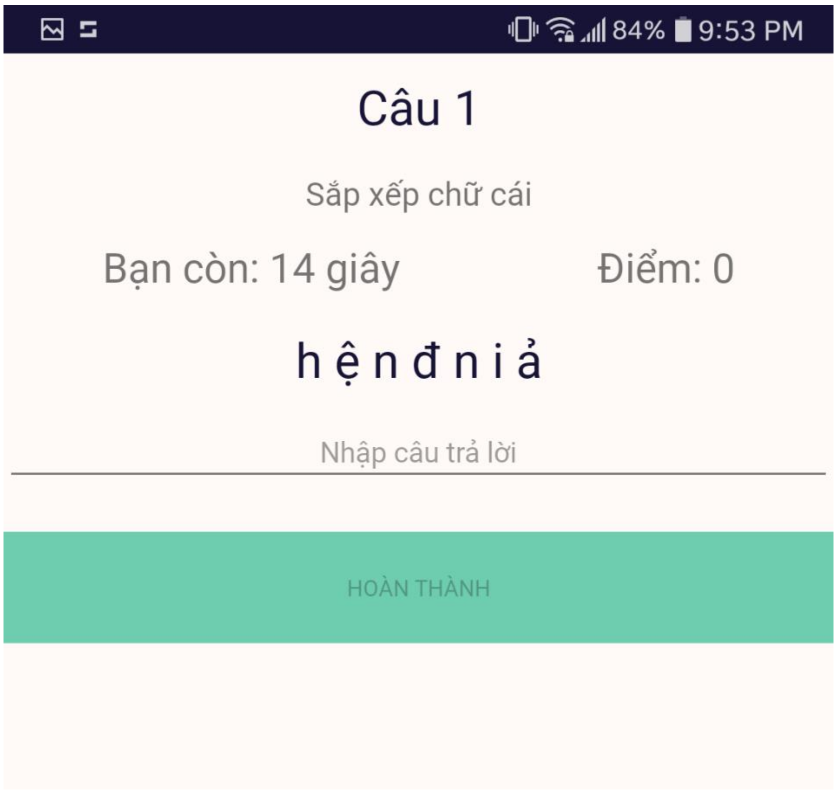
Bạn còn: 54 giây

Điểm: 0

Thiên

Thiên

Figure III.32: Language game three UI


84% 9:53 PM

Câu 1

Sắp xếp chữ cái

Bạn còn: 14 giây

Điểm: 0

h ệ n đ n i ả

Nhập câu trả lời

Figure III.33: Language game four UI

4.2.4. Math games

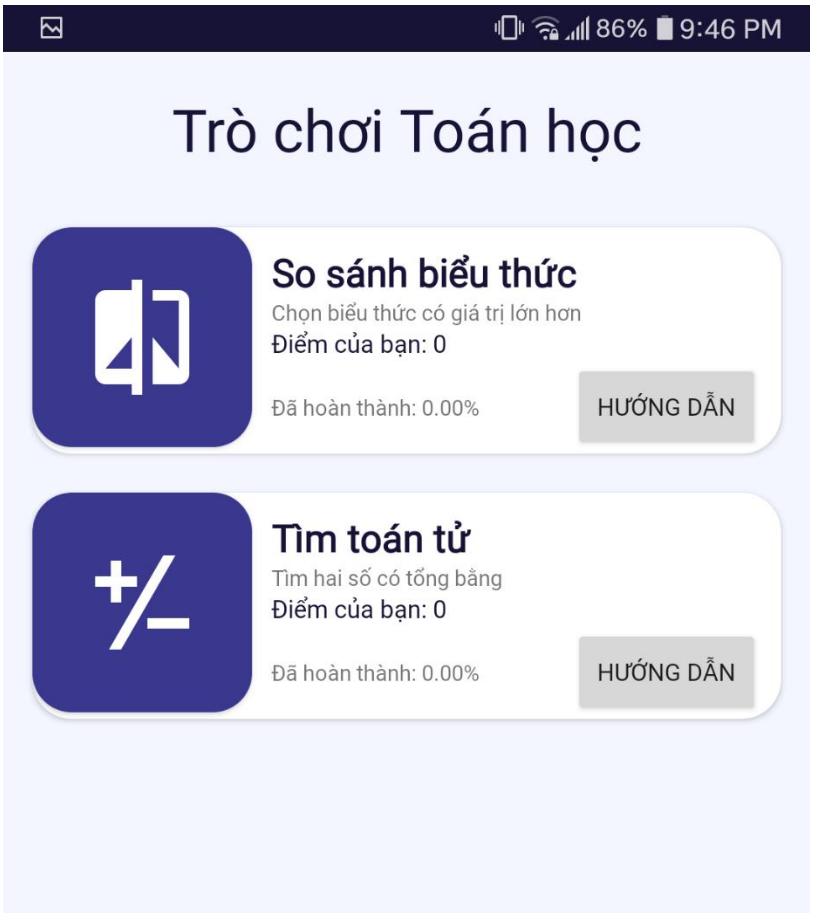


Figure III.34: Math game menu

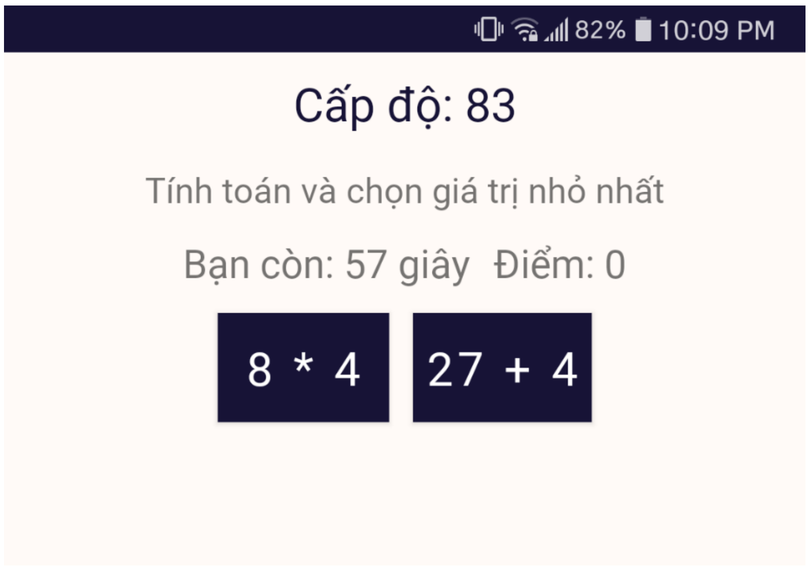


Figure III.35: Math game one UI

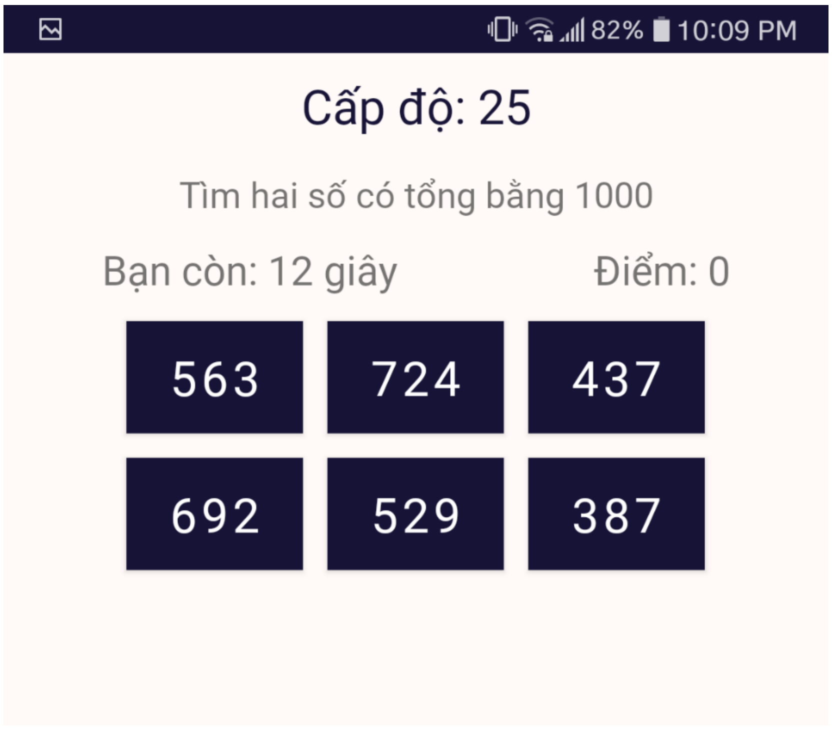


Figure III.36: Math game two UI

IV. CONCLUSION AND FUTURE WORK

6.1. Conclusion

❖ The outcome of the project:

1. The system allows Users to create, log in and change password
2. The system allows Users to play different games in 4 different parts memory, attention, language, and math
3. The system save the user score and level each game as well as show the their playing progress.

❖ The learning outcome after this project

1. Create the complete application for mobile device
2. Learn how to design a database
3. How to handle a database with programming language
4. How to connect database with Android Studio

6.2. Pros and Cons

Pros	Cons
The system allows users to change their password of account and have all function that users need.	UI quite simple, is not eye-catching and difficult to attract users
The system response quickly to the user’s request.	The system still dose not have functions for admin.
The system has friendly GUI, so it is easy to use	System doesn’t allow multimedia yet (images, videos, …)

Table 2: Pros and Cons

6.3. Future work

- ❖ Develop more functions for the system
- ❖ Allow the app for rotation
- ❖ Improve setting function
- ❖ Add music and sound to games
- ❖ Improve profile setting
- ❖ Improve UI to look more like a real system
- ❖ Develop the app to become a website and available for IOS to make it more convenient for many users

V. REFERENCES

1. John Horton “Android Programming for Beginners: Build in-depth, full-featured Android 9 Pie apps starting from zero programming experience” Second Edition, 2018.
2. O'Reilly “Android Cookbook”, Second Edition, 2017.
3. Stack Overflow - Where Developers Learn, Share, & Build Careers: https://stackoverflow.com/
4. Android Mobile App Developer Tools – Android Developers: https://developer.android.com/
5. Android 2D Game Tutorial for Beginners | o7planning.org: https://o7planning.org/10521/android-2d-game-tutorial-for-beginners