ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Integration of Serious Games to Implicitly Detect Personality in a Group Recommender System for Tourism

Kazem Peyvand

Dissertation submitted in partial fulfillment of the requirements for the Degree of master's in informatics engineering, Specialization in Games, Graphical and Interactive Systems

Supervisor : Patrícia Alves

Co-Supervisor: Goreti Marreiros

Porto, September 28 ,2025

Declaration of Integrity

I hereby affirm that I have carried out this academic work with the utmost importance.

I confirm that I have neither plagiarized nor engaged in any form of information misuse or falsification of results during the entire process of developing this work.

As such, the work presented in this document is entirely original, solely my own, and has not been previously submitted or used for any other purpose.

Additionally, I declare that I am fully aware of and adherent to the code of ethical conduct of P. Porto.

ISEP, Porto, September 28,2025

Resumo

A motivação do utilizador é essencial para a eficácia dos sistemas de recomendação em turismo. Estes enfrentam desafios como o problema do cold-start e a recolha intrusiva de dados através de questionários. A gamificação surge como uma solução eficaz, promovendo envolvimento e interação natural com a aplicação.

Esta dissertação apresenta a integração de técnicas de gamificação numa aplicação móvel para Android de um Sistema de Recomendação para Grupos (GRS) de turismo, no âmbito do projeto Accelerate & Transform Tourism (ATT), nomeadamente a integração de jogos sérios, previamente desenvolvidos em Unity, que recolhem traços de personalidade segundo o modelo Big Five de forma implícita. A principal contribuição reside no desenvolvimento de ecrãs dedicados em Android para o lançamento destes jogos, na recolha de dados de personalidade através de trocas em formato JSON e na apresentação detalhada dos resultados dos traços de personalidade por meio de janelas interativas.

Para abordar o problema do cold start e o desafio de recolher dados fiáveis dos utilizadores sem recorrer a questionários intrusivos, o sistema proposto utiliza jogos sérios como ferramenta implícita de perfilagem de personalidade. Os traços de personalidade captados são posteriormente utilizados para personalizar (agrupar) recomendações de atividades turísticas e pontos de interesse.

Os resultados demonstram a viabilidade de integrar jogos desenvolvidos em Unity num ambiente Android, gerir a troca de dados e proporcionar aos utilizadores um retorno transparente e gamificado sobre os seus perfis. O protótipo final evidencia como a gamificação pode aumentar a participação dos utilizadores, enriquecer a recolha de dados e melhorar a experiência global do utilizador em aplicações de turismo digital.

Palavras-chave: Gamificação, Sistemas de Recomendação, Traços de Personalidade, Big Five, Android, Unity, Jogos Sérios, Turismo

Abstract

User motivation is crucial to the effectiveness of tourism recommender systems. These systems face challenges such as the cold-start problem and the intrusive collection of data through questionnaires. Gamification emerges as an effective solution, fostering engagement and natural interaction with the application.

This dissertation presents the integration of gamification techniques into an Android mobile application of a Group Recommender System (GRS) for tourism, developed within the scope of the Accelerate & Transform Tourism (ATT) project, namely the integration of serious games previously developed in Unity, which implicitly collect personality traits based on the Big Five model. The core contribution lies in developing dedicated Android screens to launch these games, retrieving personality data through JSON exchanges, and presenting detailed trait breakdowns via interactive popups.

To address the cold-start problem and the challenge of collecting reliable user data without intrusive questionnaires, the proposed system uses serious games as an implicit personality profiling tool. The captured personality traits are then used to personalize (group) recommendations for tourism activities and points of interest.

The results demonstrate the feasibility of integrating Unity-based games within an Android environment, managing data exchange, and providing users with transparent, gamified feedback on their profiles. The final prototype shows how gamification can increase user participation, enrich user data collection, and enhance the overall user experience in digital tourism applications.

Keywords: Gamification, Recommender Systems, Personality Traits, Android, Unity, Serious Games, Tourism

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my family for their unwavering support throughout this journey. Their encouragement, love, and belief in me have been instrumental in shaping who I am today. This work would not have been possible without their constant presence by my side.

I am sincerely thankful to my supervisor, Patrícia Alves and my co-supervisor, Goreti Marreiros for granting me the invaluable opportunity to merge game development with research. Your expertise, guidance, and wisdom have played a critical role in the success of this project.

I would also like to acknowledge my fellow master's students for their collaboration, teamwork, and the shared experiences that enriched both the projects and classes during this academic journey.

Lastly, I wish to express my profound gratitude to all those who contributed to the body of knowledge on which this project is built. By drawing on the works of other researchers, teams, and colleagues who have explored gamification and its applications in various domains, I was able to contextualize and develop the ideas presented in this work. Your foundational studies and insights have been instrumental in shaping the direction and objectives of this project.

Index

List of Figures

List of Tables

Acronyms and symbols

3D	3 Dimensions
ACM	Association for Computing Machinery
API	Application Programming Interface
ATT	Agenda Acelerar e Transformar o Turismo (Accelerate & Transform Tourism)
BFI	Big-Five
CF	Collaborative-Filtering
FFM	Five-Factor Model
GECAD	Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development
GDPR	General Data Protection Regulation
GRASP	General Responsibility Assignment Principles
GRS	Group Recommender System
IEEE	Institute of Electrical and Electronics Engineers
IPIP	International Personality Item Pool
IPIP-NEO	International Personality Item Pool – Neuroticism, Extraversion, Openness
IPP	Polytechnic Institute of Porto
IR	Intrinsic Rewards
ISEP	Instituto Superior de Engenharia do Porto
JSON	JavaScript Object Notation
MAMS	Multi-Agent Microservices System
MAS	Multi-Agent Systems
MVC	Model-View-Controller
MVP	Model-View-Presenter

1 Introduction

1.1 Context and Motivation

This project was developed within the scope of the master's in informatics engineering at the Instituto Superior de Engenharia do Porto (ISEP), in the specialization area of Games, Graphical, and Interactive Systems. Conducted in collaboration with the Research Group on Intelligent En-gineering and Computing for Advanced Innovation and Development (GECAD), this research was integrated into the Accelerate & Transform Tourism (ATT) initiative[1], part of Portugal's Recovery and Resilience Plan (RRP) funded by the European Union. This work builds upon an existing Group Recommender System (GRS) for tourism, previously developed at GECAD and described in Expert Systems with Applications (Alves et al., 2025). The original prototype addressed the need to personalize recommendations for groups of tourists, using personality traits to provide initial recommendations and to solve the cold-start problem, while employing personality-based dynamic clustering to handle group heterogeneity and conflicting preferences. Within this context, the main contribution of this work was the design and implementation of an Android-based extension for the GRS prototype. The work focused on developing new Android application screens, integrating Unity-based serious games to implicitly retrieve personality traits, handling JSON-based data exchange, and linking this information to a backend database for storage and analysis. The traits collected through gameplay directly support the recommender logic by enabling automatic personality profiling, replacing static questionnaires with a gamified, engaging user experience. The resulting Android application demonstrates how mobile technology, gamification mechanics, and serious games can work together to enhance user motivation and engagement, while generating reliable personality data for better recommendations. The system bridges research with practical applications by providing a fully functional prototype, complete with documented architecture, UI design, code structure, and backend integration, supporting the Group Recommender System in the real-world tourism context.

[1] https://www.gecad.isep.ipp.pt/portfolio/_att/

1.2 Problem

Recommender systems for tourism often face significant challenges that limit their effectiveness and user acceptance (see Figure 1), especially when applied to groups of travelers. One well-known challenge is the cold-start problem: when a new user first uses the system, there is little or no interaction history, which makes it difficult to provide meaningful personalized suggestions (Alves et al., 2025). Another critical issue is the conflicting group preferences problem, where tourists travelling together have different interests, and the system must find a fair compromise that satisfies the group as a whole.

Figure 1: Recommendation quality Challenges

To help overcome these problems, previous research within the Grouplanner project and related prototypes used personality profiling — for example, by asking users to fill out explicit Big Five personality questionnaires — to provide initial recommendations and to create dynamic clusters of similar users (Alves et al., 2025). However, explicit questionnaires can be time-consuming, intrusive, and sometimes unreliable, because people may answer in socially desirable ways rather than truthfully (McCrae & Costa, 2004).

Therefore, the need to replace explicit personality questionnaires with a more engaging, non-intrusive alternative arose. The chosen approach was to integrate a suite of serious games that implicitly measure the same personality traits in a playful way. This solution avoids long forms and instead gathers the data while users play, improving the user experience and increasing engagement through gamification.

The problem this dissertation addresses, therefore, is how to integrate these personality measurement games into the existing tourism Group Recommender System, ensuring that the personality traits recorded in the Unity games replace or complement the broader personality scores originally obtained through traditional questionnaires. The work also needed to ensure that these traits could be reliably stored, transmitted, and used by the recommender logic, closing the loop between gameplay, trait extraction, and group recommendation.

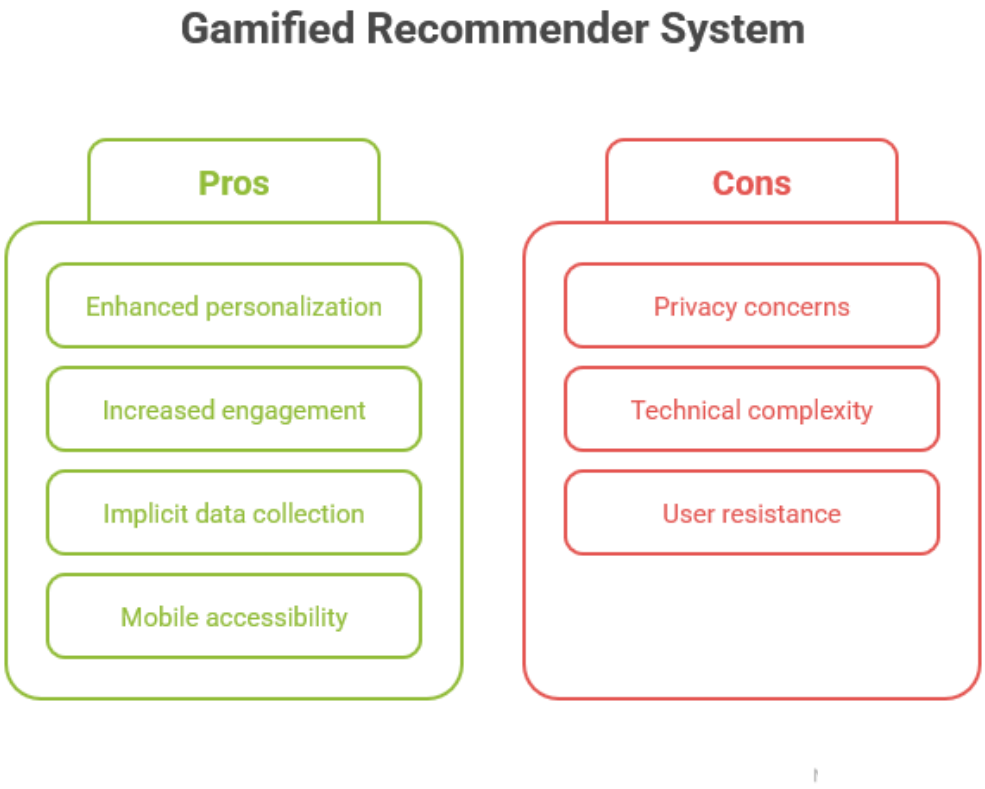


Figure 2 - Gamified recommender system

1.3 Specific Contributions

This dissertation focused on extending an existing group recommender system prototype for tourism by integrating a set of Unity-based serious games designed to measure user personality traits implicitly. The main contributions of this work are:

Modifying and preparing serious personality games: Ensuring each Unity game correctly records detailed sub-trait scores according to the Big Five model (e.g., trust, openness, conscientiousness facets).

Developing the Android integration module: Designing and implementing Android screens that allow users to launch the Unity games directly from the app, show game instructions and trait explanations through pop-ups, and retrieve the resulting personality scores in JSON format.

Implementing trait data handling and backend communication: Creating the logic to receive, process, and persist with the trait scores generated by the games, sending them to the backend database so they can be used to personalize recommendations within the group recommender system.

Replacing explicit personality surveys: Replacing the broader scores obtained from static BFI questionnaires with the more precise sub-traits collected implicitly during gameplay, helping to address the cold-start problem and improve the dynamic clustering of group members with diverse preferences (Alves et al., 2025).

Overall, this work demonstrates how serious games, and gamification can be practically embedded in a mobile context to improve user experience and enrich the personality data that powers group recommendation strategies for tourism.

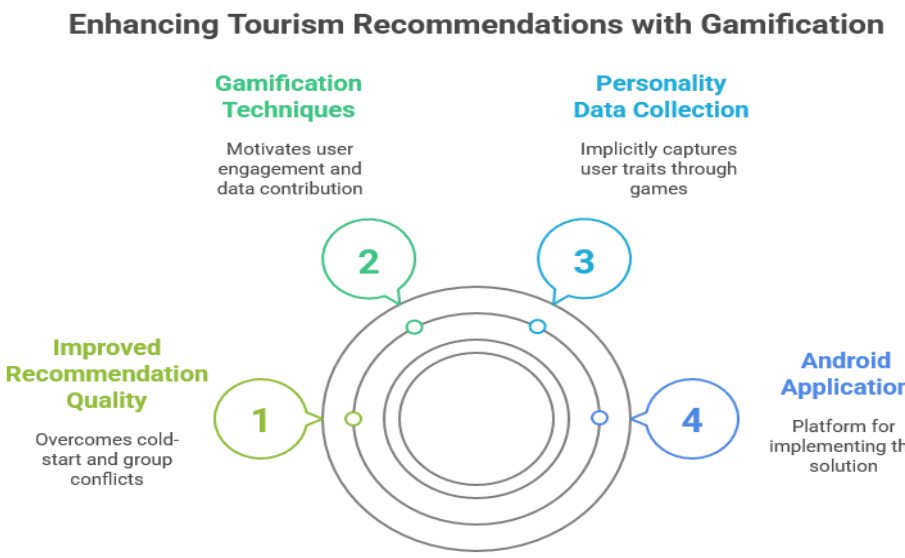


Figure 3: Tourism recommendation

1.4 Approach

The approach to this dissertation was divided into two phases:

Integration Preparation:

First, the Unity-based serious games previously developed by the research team were reviewed and updated to ensure that all relevant Big Five sub-traits could be recorded in a structured JSON format. Existing traits measurement logic was verified and adjusted where needed.

Android Integration Development:

The core implementation work consisted of building Android screens to launch the Unity games, handle the returned trait data, and show clear pop-ups explaining each game's purpose and the traits measured. This included connecting the Android app to the backend API, which stores the recorded traits in a PostgreSQL database. The integration used standard Android development practices (Activities, Fragments) and follows the MVVM pattern to ensure maintainability.

By focusing on this targeted integration and technical bridging, the project ensured that the recommender system could transition from static questionnaires to a more engaging, gamified, and implicit personality profiling approach.

1.5 Research strategy

The research strategy for this work was designed to ensure both technical relevance and practical integration with the broader GRS research.

Review of Relevant Literature: A targeted review was carried out using major academic databases (ACM Digital Library, Google Scholar, ScienceDirect) to understand best practices in:

- Implicit personality profiling through serious games
- Gamification methods for user motivation in tourism
- Mobile integration of Unity games into Android applications
- Data handling and backend communication for user trait storage

Integration Requirements Definition: Based on the reviewed literature and the architecture of the existing GRS prototype (Alves et al., 2025), integration requirements were defined. This included how the games should collect detailed sub-traits and how the Android interface would manage game launch, trait retrieval, and backend submission.

Technical Development and Validation: The practical work focused on:

- Modifying the Unity-based serious games to output structured JSON trait data.

- Developing Android screens to launch each game, handle returned scores and show clear instructions via pop-ups.
- Connecting the app to a backend API for storing user scores securely.
- Testing these integrations internally to ensure smooth communication between Unity, Android, and the backend server.

1.6 Data Handling and Ethical Compliance

In both games, all recorded metrics are serialized into a structured JSON file upon session completion. This JSON file serves as the primary data object for transmission to the Android application and subsequent analysis. It contains gameplay-related metrics, personality-related indicators, and optional demographic data provided by the user.

To comply with the General Data Protection Regulation (GDPR) and institutional ethical standards, the data collection pipeline incorporates the following principles:

Pseudonymization and Anonymity

All user identifiers (e.g., name, email) are pseudonymized before storage or transmission.

Only a unique participant code is retained, ensuring that no personally identifiable information (PII) is stored in raw data files.

Explicit Consent

Before starting gameplay, users are required to review and accept an informed consent form presented within the Android application.

Consent covers data collection, storage, and use for research purposes only.

Demographic Information

Demographic data is collected voluntarily to contextualize research findings.

This includes attributes such as age, gender, education level, and prior gaming experience.

The information is stored in a separate JSON section and linked only by pseudonymized participant ID.

Secure Storage and Transmission

JSON files are transmitted via secure HTTPS connections to the backend server.

Data is stored in an encrypted PostgreSQL database, accessible only to authorized research staff.

Data Minimization

Only essential metrics are collected, avoiding unnecessary personal information.

Raw gameplay logs are summarized to reduce storage overhead and minimize risk.

1.7 Document Structure

The rest of the dissertation is organized into the following chapters:

Chapter 2: Context and State of the Art — Reviews the literature on recommended systems for tourism, gamification principles, serious games for implicit personality profiling, and mobile app integration strategies.

Chapter 3: System Design — Describes the technical design of the integration, including system architecture, game launch flow, user interface structure, and backend communication logic.

Chapter 4: Implementation — Provides details of the actual implementation work: Unity game modifications, Android activities for launching games and pop-ups, JSON data handling, and backend API interaction. Includes relevant snippet code and screen examples.

Chapter 5: Conclusions and Future Work — Summarizes the main outcomes of the integration work, discusses limitations, and outlines potential directions for extending the system, including plans to make the app available publicly and expand trait profiling scope.

Chapter 6: References — Lists all academic sources cited in this dissertation.

2 Context and State of the Art

This chapter reviews the foundational concepts, technologies, and prior research relevant to this dissertation. It begins by exploring recommender systems in the context of tourism, gamification strategies for increasing user engagement, the role of personality acquisition in recommendation systems, and recent advances in Android-based serious games for personality profiling.

The following subsections provide the theoretical foundation necessary for the design and implementation of the Android application developed in this work.

2.1 Recommender Systems in Tourism

Recommender systems (RS) are software solutions designed to provide users with personalized content based on their preferences, behavior, or contextual data. In the tourism sector, recommender systems are used to suggest destinations, cultural attractions, restaurants, or even customized travel itineraries.

Despite their growing use, tourism RS still faces several challenges, including:

- Cold-start problem: New users have little or no interaction history, reducing recommendation quality.
- Diverse user interests: Tourists often travel in groups, making it difficult to reconcile conflicting preferences.
- Contextual needs: Recommendations should adapt to the user's context, such as mood, personality, or current location.

Previous work related to this dissertation involved the GroupPlanner project by GECAD, which addressed these challenges through a Group Recommender System (GRS) that proposed personalized routes based on group dynamics. However, this project builds upon that by

creating a dedicated Android application, designed specifically to engage users through gamification and implicit personality profiling.

2.2 Gamification for Motivation in Tourism Applications

Gamification introduces game-design elements—such as points, levels, challenges, and leaderboards—into non-game contexts to enhance user engagement. In tourism, gamification is being used increasingly to:

- Encourage exploration of lesser-known destinations.
- Foster competition or collaboration between travelers.
- Provide rewards (e.g., badges or achievements) for completing tourism-related activities.

By embedding serious games within the Android app, this work addresses one of the main problems in RS: engaging users to provide meaningful data while offering them fun and valuable interactions. Unlike static questionnaires, the games implicitly capture personality traits, making the experience non-intrusive and playful.

The Android app includes:

- Achievements and rewards systems.
- Progress tracking and scoring dashboards.
- Location-based challenges tied to tourism points of interest.

2.3 Android as a Platform for Tourism and Gamification

The choice of Android Studio as the development platform is driven by several factors:

- Widespread use of Android devices among global travelers.
- Access to powerful libraries for UI/UX design, geolocation, and real-time interaction.
- Flexibility to integrate advanced features like serious games, REST APIs for data exchange, and local data persistence.

Modern Android development also enables the use of architectural patterns like MVVM (Model-View-View Model), improving code maintainability and scalability. Integration with Firebase or other analytics platforms further allows real-time collection of user interaction metrics for later analysis.

By providing a native Android experience, this project ensures that recommendations and gamified elements are fully accessible on mobile devices, aligning with how tourists interact with digital tools during their travels.

2.4 Implicit Personality Profiling Through Serious Games

Traditional methods of collecting personality data, such as the IPIP-NEO-120 questionnaire, are explicit, lengthy, and prone to bias as users may answer in socially desirable ways rather than truthfully [McCrae & Costa, 2004]. To overcome these limitations, this work integrates seven Unity-based serious games within the Android application to implicitly capture user behaviors mapped to the Big Five personality traits.

Each game is designed with mechanics that elicit natural decision-making under varying cognitive, moral, or social conditions. Gameplay metrics are serialized into a unified JSON schema, ensuring a standardized data pipeline across all games. While the current implementation includes advanced personality modeling for two games (*Coin Catcher* and *Mindful Escape*), the architecture is extensible and ready for expansion to the remaining games in future work.

2.5 Covered Personality Traits

The games collectively target multiple Big Five traits (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism) and their facets. Each game is aligned with a subset of traits as described in Table 1.

2.5.1 The Agreeableness Domain

Agreeableness reflects individual differences in concern for cooperation and social harmony. Individuals high in Agreeableness are more trusting, helpful, and compassionate, while those low in this trait are more competitive, skeptical, and potentially antagonistic. This project focuses on five key facets of Agreeableness, as defined by the Five-Factor Model (FFM) [Costa & McCrae, 1992]:

Cooperation: The tendency to be accommodated and avoid conflict.

Morality: A preference for honesty and genuineness over manipulation.

Altruism: Active concern for the welfare of others.

Modesty: A tendency to be humble and self-effacing.

Anger: (A reversed-scored facet) The propensity to experience anger and frustration.

Table 1: Personality Traits

Game	**Target Traits (Big Five Facets)**	**Current Implementation Status**
Mindful Escape	Cooperation, Modesty, Altruism (Agreeableness)	Implemented, advanced metrics
Coin Catcher	Morality, Altruism, Anger (Agreeableness)	Implemented, advanced metrics
Other World	Openness (Creativity, Curiosity, Imagination)	Basic JSON logging, future expansion
UnboxIt	Conscientiousness (Organization, Order, Goal-Orientation)	Basic JSON logging, future expansion
Fishy Catcher	Extraversion (Social interaction, Engagement, Responsiveness)	Basic JSON logging, future expansion
Time Travel	Neuroticism (Stress management, Decision under pressure)	Basic JSON logging, future expansion
Which Way	Conscientiousness & Openness (Decision-making, Exploration)	Basic JSON logging, future expansion
Game	Target Traits (Big Five Facets)	Current Implementation Status
Mindful Escape	Cooperation, Modesty, Altruism (Agreeableness)	Implemented, advanced metrics
Coin Catcher	Morality, Altruism, Anger (Agreeableness)	Implemented, advanced metrics
Other World	Openness (Creativity, Curiosity, Imagination)	Basic JSON logging, future expansion

2.5.2 Coin Catcher: Assessing Morality and Related Traits

The Coin Catcher game immerses players in a first-person 3D environment featuring embedded moral dilemmas, designed to measure the Morality sub-trait.

Game Mechanics for Trait Capture:Players navigate a environment and collect coins. They encounter four unique moral dilemmas, each presenting multiple-choice options (e.g., help an injured Non-Player Character (NPC) or ignore them to collect more coins). Each choice is pre-mapped to a value on a Likert scale (1–5), reflecting its moral weight. Critically, choices cannot be undone, encouraging intuitive, emotionally driven decisions that better reflect the player's internal values rather than a calculated response.

Collected Metrics and Their Psychological Rationale:

• MoralDilemmaChosenOption & DecisionMorality: The specific action and its score form the primary measure of moral compass.

• CoinsCollected: A measure of self-interest, inversely correlated with Altruism and Morality.

• DecisionTimeTaken: Reaction time can indicate cognitive conflict or certainty, potentially relating to internal consistency of values.

• The final, normalized MoralityScore (range [0,1]) is derived from the aggregate of all decisions, providing a robust behavioral measure.

2.5.3 Mindful Escape: Assessing Cooperation and Related Traits

The Mindful Escape game frames trait collection within a 3D escape room puzzle based on social interaction and resource management, designed primarily to measure Cooperation.

Game Mechanics for Trait Capture:

Players must collaborate with three NPCs, each programmed with a distinct strategy from game theory (Always Cooperate, Always Defect, Tit-for-Tat). Through a trading interface, players decide to send genuine help, fake help, or not help. A limited pool of shared "special hints" introduces a "Tragedy of the Commons" dynamic, testing the player's willingness to cooperate for the common good versus acting in immediate self-interest.

Collected Metrics and Their Psychological Rationale:

• TotalTimesCooperated: The raw count of cooperative actions directly measures the propensity to cooperate.

• TotalTradesDoneRatio: Measures engagement in social interaction, a prerequisite for cooperative behavior.

• TotalSpecialHintsUsedRatio: Consumption of shared community resource is a key indicator of selfishness vs. altruism and modesty.

• GameCooperationScore: The final normalized score aggregates these behaviors into a primary metric.

3 System Design

This chapter presents the conceptual and technical design of the Android application, focusing on its architecture, user interface, gamification components, and the serious games integrated into the app.

3.1 Application Architecture

The application was designed using the Model-View-View Model (MVVM) architectural pattern. This separation of concerns improves code organization, readability, and scalability.

Model: Handles data-related logic. It includes the data classes representing user profiles, preferences, game scores, and demographic information.

View: Responsible for displaying information to the user (XML layouts, Jetpack Compose components).

View Model: Acts as a bridge between Model and View, containing the application's logic and handling UI-related data in a lifecycle-conscious way. It manages the flow of data, including user profiles and trait scores, between the UI and the backend repositories.

3.2 Main Features

The key features implemented in the Android application include:

User Registration and Demographic Profile Management

A foundational feature of the application is the collection and management of basic user demographic information. This data serves multiple purposes: it provides essential context for the recommender system, allows for segmenting user data in analysis (e.g., to study trait correlations across different age groups or nationalities), and helps in personalizing the user

experience. The demographic data is collected during the initial registration process and stored securely in the backend database alongside the personality trait scores.

The core demographic attributes stored for each user are defined in the following data model:

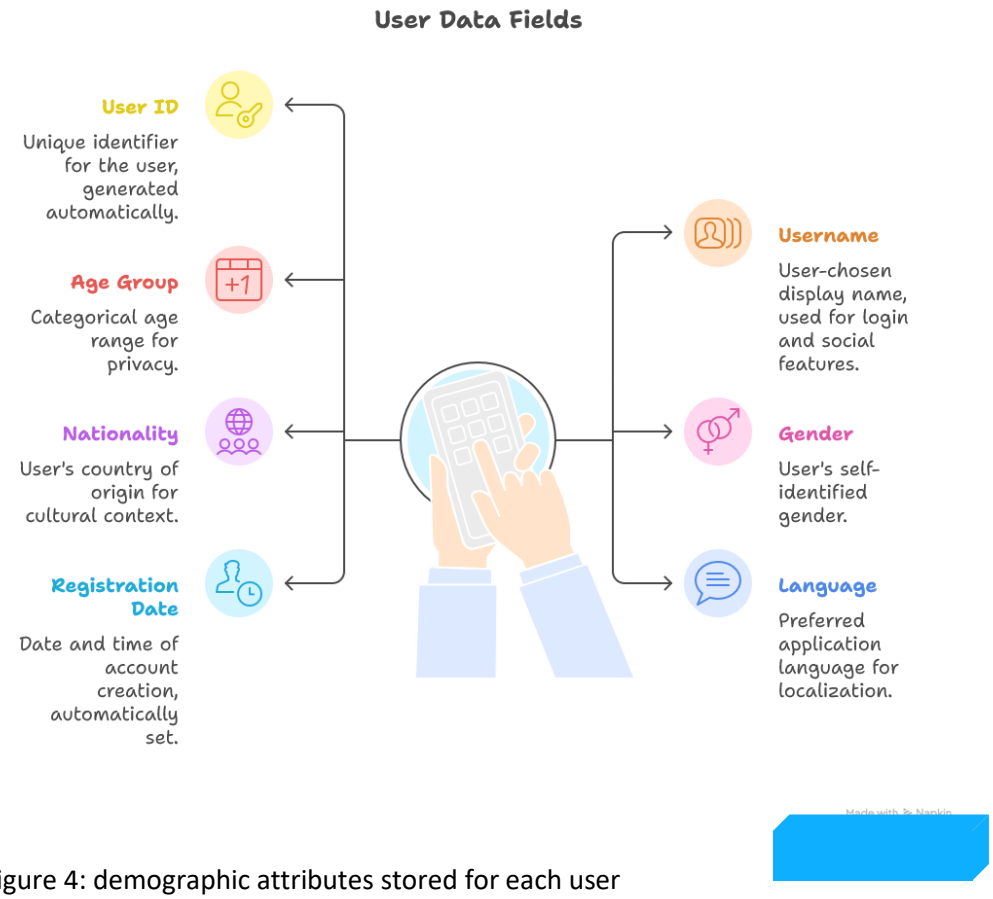


Figure 4: demographic attributes stored for each user

3.3 UI/UX Design

The application follows Android Material Design guidelines for consistent, intuitive, and visually appealing user experiences. Emphasis was placed on:

Clear navigation between modules.

Immediate visual feedback on completed actions (e.g., earning a badge).

Accessible game interfaces with clear objectives.

Code Organization

The project repository was structured as follows:

```
app/
└── src/
    └── main/
        ├── java/com/example/tourapp/
        │   ├── ui/           # Activities, Fragments, Adapters
        │   ├── viewmodel/    # ViewModels for UI logic
        │   ├── model/        # Data classes (User, PersonalityData, etc.)
        │   └── repository/   # Classes for handling data operations (API,
database)
        └── res/
            ├── layout/       # XML layout files for UI
            ├── drawable/     # Icons, images, and graphical assets
            └── values/       # Strings, colors, dimensions
```

3.3.1 System Architecture Diagram

The architecture of the Android application is represented in Figure 1. It highlights the flow of data between the user interface, the View Model, and the underlying data models and repositories.

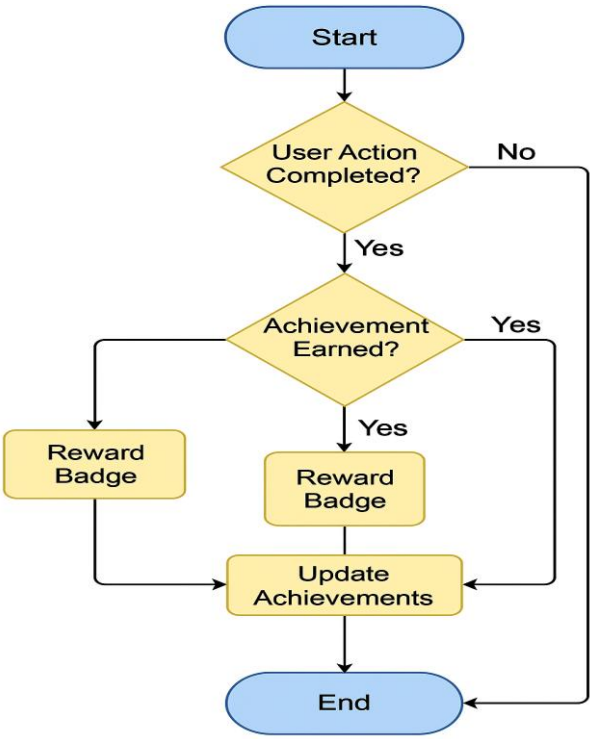


Figure 5: System Architecture Diagram

3.3.2 Class Diagram

A data class diagram is provided to illustrate the interaction between the main components of the application, specifically focusing on the process of collecting personality traits and generating recommendations.

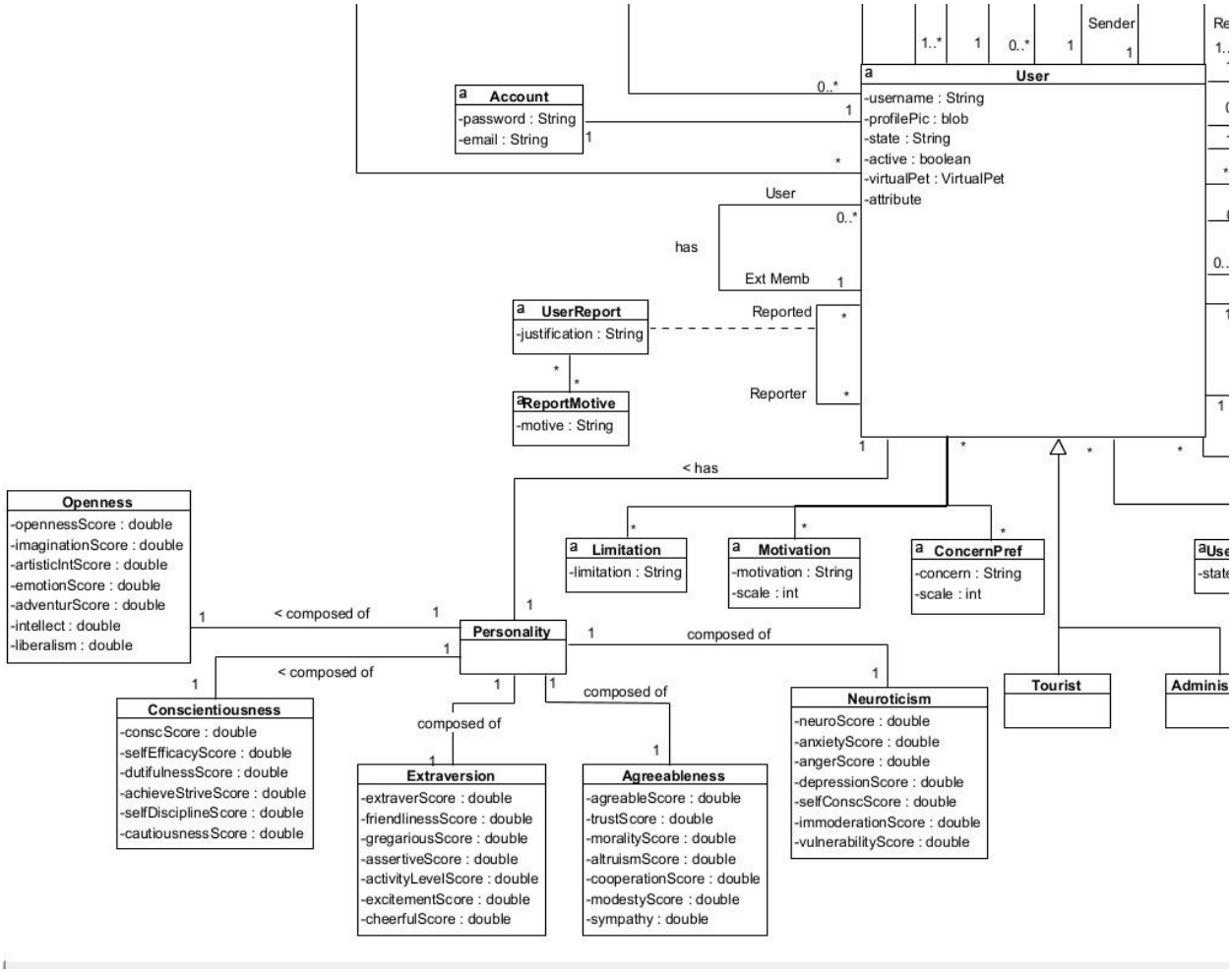


Figure 6: Data Flow Diagram for Personality Acquisition and Recommendation Process

4 Implementation

This chapter presents the practical work developed for integrating Unity-based serious games into an existing Group Recommender System (GRS) prototype for tourism. The main contribution lies in modifying the personality games to record detailed sub-traits, implementing Android activities to launch the Unity games, managing trait data exchange via JSON, and storing results in a PostgreSQL backend through a Flask API.

The work did not develop the entire GRS but rather extended its personality acquisition component by replacing generic questionnaire-based scores with richer implicit data from gameplay.

The following sections describe the tools used, the code structure, core modules, and technical challenges addressed during the integration.

4.1 Development Framework

The integration work was developed using Android Studio (Java/Kotlin), Unity for the serious games, and a Flask server connected to a PostgreSQL database for secure trait storage [UnityDocs, AndroidDocs].

The architecture follows the MVVM pattern, separating the user interface (View), business logic (ViewModel), and data management (Chart).

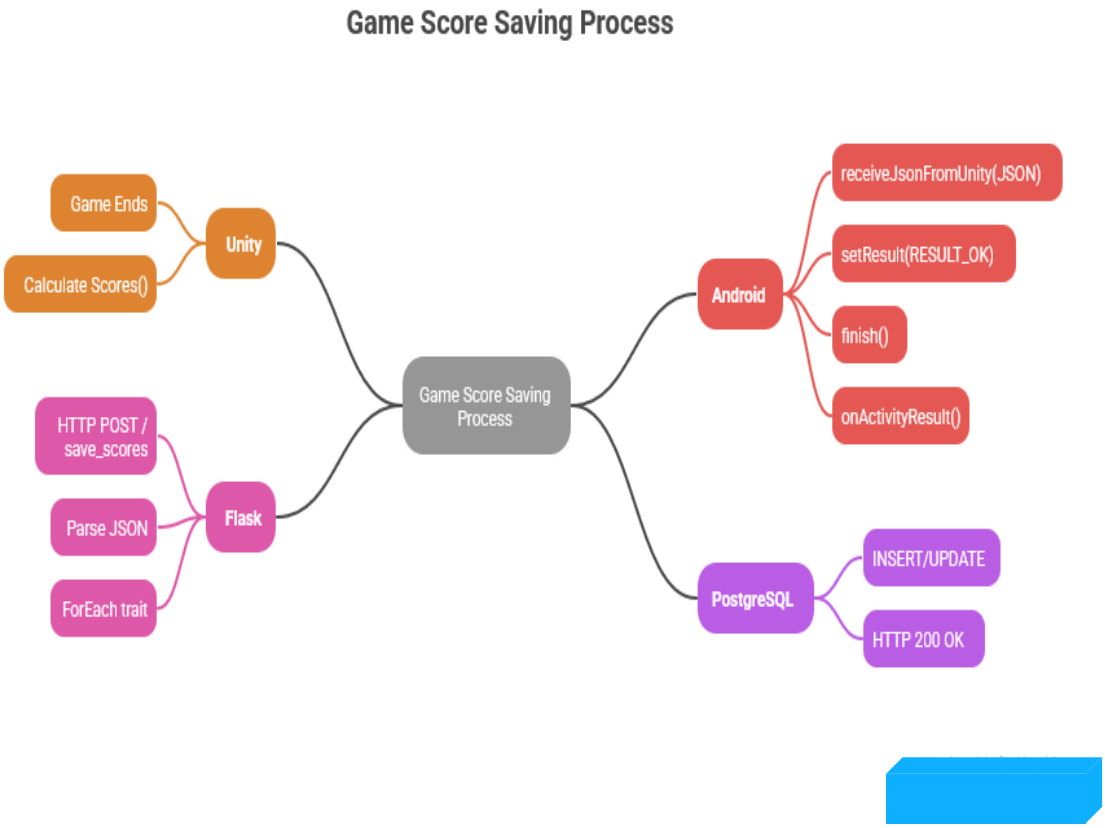


Figure 7 Sequence Diagram of the End-to-End Data Pipeline

4.2 PersonalityGamesActivity: Integration of Unity Games within the Android Ecosystem

4.2.1 User Interface and Game Integration:

The PersonalityGamesActivity class is responsible for managing the user interface where participants can select and launch the Unity-based serious games integrated into the Android application. This activity provides two primary functionalities:

Game Launching – enabling users to start Unity games directly from the Android interface.

Game Descriptions – presenting pop-up dialogs that describe each game's objectives, mechanics, and the Big Five sub-traits targeted.

Game Launching:

Each game is mapped to a dedicated button widget in the XML layout. When clicked, the button triggers the method launchGameByPackageName(), which attempts to open the game via its package name and Unity's UnityPlayerActivity.

The launch procedure follows a three-tier fallback strategy:

Primary attempt: Explicitly start the Unity activity by package name.

Secondary attempt: Query the Android package manager for the default launch intent.

Final fallback: Notify the user if the game is unavailable and redirect to the Play Store installation page.

This design ensures robustness: installed games launch seamlessly, while unavailable games trigger informative messages (e.g., *"Time Travel coming soon!"*), maintaining user engagement even with incomplete content.

Game Descriptions:

To improve transparency and user understanding, each game includes a description button. When clicked, a pop-up window (PopupWindow) is instantiated from the custom layout dialog_game_description.xml. The pop-up presents:

Game title (e.g., Coin Catcher, Mindful Escape).

Short description of gameplay mechanics.

Mapping to targeted Big Five sub-traits (e.g., Morality, Cooperation, Altruism).

This design choice directly supports the research objective of making personality profiling explicit and understandable to participants, thereby increasing trust and informed participation.

Trait Score Popups (Optional)

In addition to per-game descriptions, the activity also supports an optional trait feedback popup, which presents a tabular summary of personality scores across the Big Five dimensions. The table is dynamically updated using the updateTableScore() method, which replaces placeholder values with the actual normalized scores (range [0–1]) calculated after gameplay.

This optional feature enhances user awareness of how their behavior during gameplay translates into measurable personality traits, reinforcing the gamified transparency of the system.

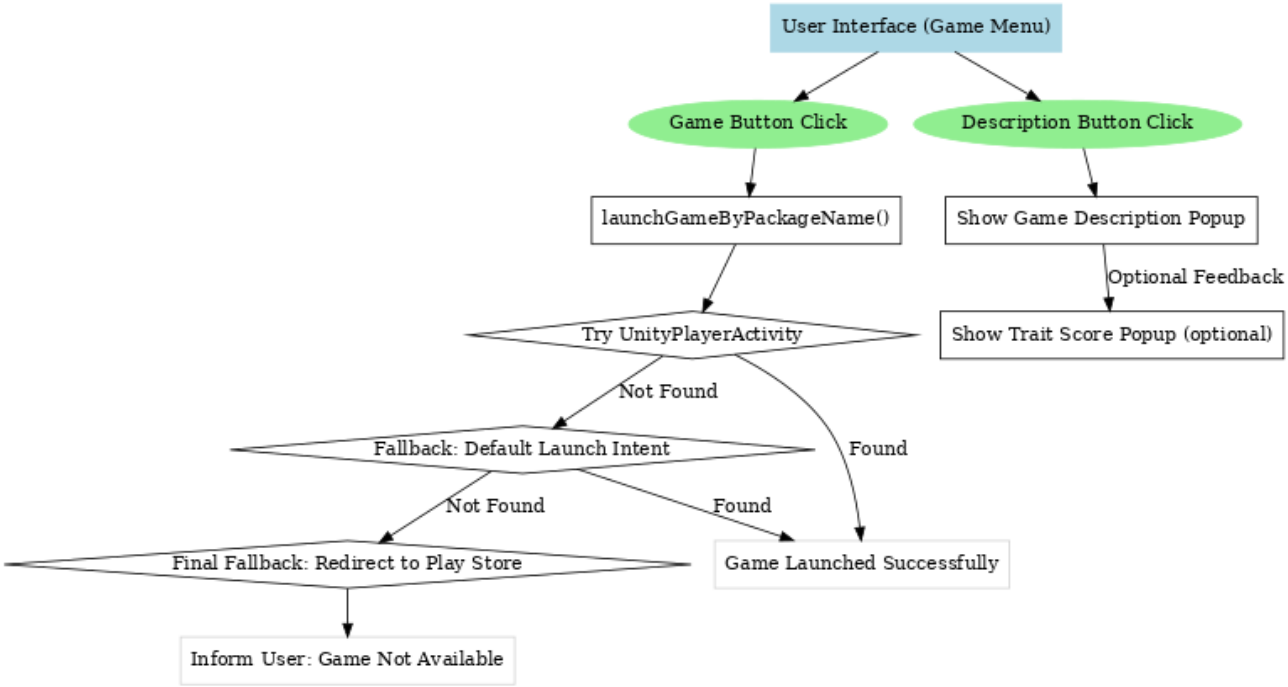


Figure 8 PersonalityGamesActivity_Flow.

4.2.2 Core Implementation for Gamified Personality Profiling

In summary, the PersonalityGamesActivity serves as the central hub connecting the Android application and the embedded Unity mini-games. It:

Provides a structured and visually engaging menu for all seven games.

Implements a robust and fail-safe launching mechanism for each Unity game.

Enhances user motivation and engagement through descriptive pop-ups and optional personality trait feedback.

Acts as a bridge between Unity game environments and the psychological Big Five trait framework, enabling real-time personality assessment within a gamified system.

This implementation demonstrates the technical feasibility of embedding Unity-based serious games into an Android ecosystem while maintaining usability, transparency, and research alignment in gamified personality profiling.

```
import android.content.ActivityNotFoundException;
import android.content.ComponentName;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Point;
import android.net.Uri;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.PopupWindow;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import com.example.umgrouplannerfrontend.R;

public class PersonalityGamesActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_personality_games);
        setupGameButtons();
        setupGameDescriptionButtons();
    }

    /** Configures game launchers for all available Unity-based mini-games.
*/
    private void setupGameButtons() {
        findViewById(R.id.btn_game1).setOnClickListener(v ->

launchGameByPackageName("com.GECAD_ATT.MindfulEscapeTimerEN"));  //
MindfulEscape
        findViewById(R.id.btn_game2).setOnClickListener(v ->
                launchGameByPackageName("com.GecadATT.OtherworldNT"));
// OtherWorld
        findViewById(R.id.btn_game3).setOnClickListener(v ->
                launchGameByPackageName("com.GECADNT.UnboxItNoT"));
// UnboxIt
        findViewById(R.id.btn_game4).setOnClickListener(v ->
                launchGameByPackageName("com.ISEP.CoinCatcher"));
// CoinCatcher
        findViewById(R.id.btn_game5).setOnClickListener(v ->
                launchGameByPackageName("com.GECADNT.FishyCatcherNT"));
// FishyCatcher

        // Placeholder games
        findViewById(R.id.btn_game6).setOnClickListener(v ->
                Toast.makeText(this, "Time Travel coming soon!",
Toast.LENGTH_SHORT).show());
        findViewById(R.id.btn_game7).setOnClickListener(v ->
                Toast.makeText(this, "Which Way coming soon!",
Toast.LENGTH_SHORT).show());
    }

    /** Safely launches a Unity-based game via its package name. */
    private void launchGameByPackageName(String packageName) {
        try {
            Intent unityIntent = new Intent(Intent.ACTION_MAIN);
            unityIntent.setComponent(new ComponentName(packageName,
"com.unity3d.player.UnityPlayerActivity"));
            unityIntent.addCategory(Intent.CATEGORY_LAUNCHER);
            startActivity(unityIntent);
        } catch (ActivityNotFoundException e) {
```

```
            try {
                PackageManager pm = getPackageManager();
                Intent launchIntent =
pm.getLaunchIntentForPackage(packageName);
                if (launchIntent != null) startActivity(launchIntent);
                else throw new ActivityNotFoundException();
            } catch (Exception ex) {
                Toast.makeText(this, "Game launch failed. Please ensure the
game is installed.",
                        Toast.LENGTH_LONG).show();
                redirectToPlayStore(packageName);
            }
        }
    }

    /** Redirects to Google Play if a game is not installed. */
    private void redirectToPlayStore(String packageName) {
        try {
            startActivity(new Intent(Intent.ACTION_VIEW,
                    Uri.parse("market://details?id=" + packageName)));
        } catch (ActivityNotFoundException e) {
            startActivity(new Intent(Intent.ACTION_VIEW,

Uri.parse("https://play.google.com/store/apps/details?id=" +
packageName)));
        }
    }

    /** Configures pop-ups for game descriptions. */
    private void setupGameDescriptionButtons() {
        setupGameDescriptionButton(R.id.btn_game1_desc,
R.string.game1_title, R.string.game1_desc);
        setupGameDescriptionButton(R.id.btn_game2_desc,
R.string.game2_title, R.string.game2_desc);
        setupGameDescriptionButton(R.id.btn_game3_desc,
R.string.game3_title, R.string.game3_desc);
        setupGameDescriptionButton(R.id.btn_game4_desc,
R.string.game4_title, R.string.game4_desc);
        setupGameDescriptionButton(R.id.btn_game5_desc,
R.string.game5_title, R.string.game5_desc);
        setupGameDescriptionButton(R.id.btn_game6_desc,
R.string.game6_title, R.string.game6_desc);
        setupGameDescriptionButton(R.id.btn_game7_desc,
R.string.game7_title, R.string.game7_desc);
    }

    /** Displays the description pop-up for a selected game. */
    private void setupGameDescriptionButton(int buttonId, int titleRes, int
descRes) {
        Button btn = findViewById(buttonId);
        btn.setOnClickListener(v ->
                showGameDescription(getString(titleRes),
getString(descRes)));
    }

    /** Builds and shows a descriptive pop-up window. */
    private void showGameDescription(String title, String description) {
        View popupView =
LayoutInflater.from(this).inflate(R.layout.dialog_game_description, null);
        PopupWindow popupWindow = new PopupWindow(popupView,
```

```
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT, true);

        ((TextView)
popupView.findViewById(R.id.dialog_title)).setText(title);
        ((TextView)
popupView.findViewById(R.id.dialog_description)).setText(description);
        popupView.findViewById(R.id.dialog_close).setOnClickListener(v ->
popupWindow.dismiss());

        Point size = new Point();
        getWindowManager().getDefaultDisplay().getSize(size);
        popupWindow.setWidth((int) (size.x * 0.8));
        popupWindow.showAtLocation(popupView, Gravity.CENTER, 0, 0);
    }
}
```

Code 1 – PersonalityGamesActivity.java

4.2.3 PersonalityTraitsActivity: Visualization of Personality Profiling Data

The PersonalityGamesActivity is responsible for launching Unity games, receiving their results, displaying personality scores to the user, and sending these results to the backend for persistence.

Core Implementation for the Big Five Trait Display:

The PersonalityTraitsActivity is responsible for presenting the results of the implicit personality profiling collected through the Unity-based serious games. It provides the user with an interactive interface displaying the Big Five personality traits — Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism — along with their six sub-traits (facets) for detailed exploration.

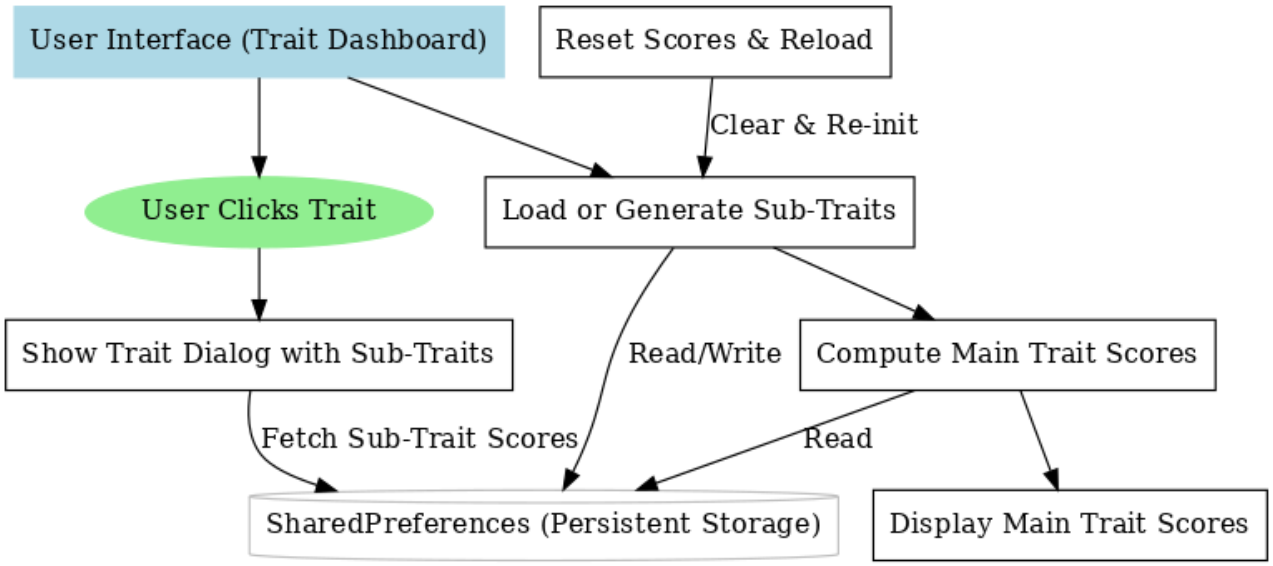


Figure 9 PersonalityTraitsActivity_Flow

4.2.4 Functional Overview

The implementation of `PersonalityTraitsActivity` performs three core functions:

Data Integration from Unity Games

Each Unity game produces a JSON file containing behavioral metrics and derived personality indicators.

These files are transmitted to the Android application, parsed, and stored locally using `SharedPreferences`.

Each of the five personality traits includes **six sub-traits** (e.g., *Conscientiousness → Self-Efficacy, Orderliness, Dutifulness,* etc.).

Sub-trait scores are extracted from JSON data and persisted for later visualization.

Calculation of Main Trait Scores

The main trait score is computed as the **average of its six sub-trait values**, providing a simplified yet representative measure.

The system maintains a **normalized range (0–1)** consistent with Unity's data model, ensuring interpretive uniformity.

User Interface and Trait Exploration

The activity layout (`activity_personality_traits.xml`) displays the five main trait scores.

Each trait button opens a **dedicated dialog** showing its six sub-traits and values (e.g., `dialog_trait_openness.xml`).

The dialog design ensures **visual consistency** and intuitive navigation across all traits.

4.2.5 Role in the System

`PersonalityTraitsActivity` functions as the **presentation layer** within the personality profiling pipeline:

Unity games collect behavioral data and derive sub-trait indicators.

The **Android application** parses and stores these values from JSON files.

The **activity** visualizes results through a gamified, interactive interface that allows users to explore both the **aggregate trait scores** and the **granular sub-trait data**.

```
package com.example.umgrouplannerfrontend.ui.activities;

import android.app.Dialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import com.example.umgrouplannerfrontend.R;

import java.util.HashMap;
import java.util.Locale;
import java.util.Map;
import java.util.Random;

public class PersonalityTraitsActivity extends AppCompatActivity {

    private static final String PREFS_NAME = "PersonalityPrefs";
    private Map<String, Float> mainTraitScores = new HashMap<>();

    private final Map<String, String[]> traitToSubTraits = new
HashMap<String, String[]>() {{
        put("Openness", new String[]{"Imagination", "Artistic Interests",
"Emotionality", "Adventurousness", "Intellect", "Liberalism"});
        put("Conscientiousness", new String[]{"Self-Efficacy",
"Orderliness", "Dutifulness", "Achievement-Striving", "Self-Discipline",
"Cautiousness"});
        put("Extraversion", new String[]{"Friendliness", "Gregariousness",
"Assertiveness", "Activity Level", "Excitement-Seeking", "Cheerfulness"});
        put("Agreeableness", new String[]{"Trust", "Morality", "Altruism",
"Cooperation", "Modesty", "Sympathy"});
```

```
        put("Neuroticism", new String[]{"Anxiety", "Anger", "Depression",
"Self-Consciousness", "Irritability", "Vulnerability"});
    }};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_personality_traits);

        loadOrGenerateSubTraits();
        computeMainTraitScores();
        displayMainScores();
    }

    /** Loads existing sub-trait data or generates random baseline values
if missing. */
    private void loadOrGenerateSubTraits() {
        SharedPreferences prefs = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        Random rand = new Random();

        for (String trait : traitToSubTraits.keySet()) {
            for (String subTrait : traitToSubTraits.get(trait)) {
                String key = trait + "_" + subTrait;
                if (!prefs.contains(key)) {
                    float value = 0.4f + rand.nextFloat() * 0.5f;
                    editor.putFloat(key, value);
                }
            }
        }
        editor.apply();
    }

    /** Aggregates sub-trait values to compute normalized Big Five scores.
*/
    private void computeMainTraitScores() {
        SharedPreferences prefs = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
        for (String trait : traitToSubTraits.keySet()) {
            float sum = 0;
            for (String subTrait : traitToSubTraits.get(trait)) {
                sum += prefs.getFloat(trait + "_" + subTrait, 0.5f);
            }
            mainTraitScores.put(trait, sum / 6f);
        }
    }

    /** Displays calculated trait scores in the main interface. */
    private void displayMainScores() {
        setTraitScore(R.id.openness_score, "Openness");
        setTraitScore(R.id.conscientiousness_score, "Conscientiousness");
        setTraitScore(R.id.extraversion_score, "Extraversion");
        setTraitScore(R.id.agreeableness_score, "Agreeableness");
        setTraitScore(R.id.neuroticism_score, "Neuroticism");
    }

    private void setTraitScore(int textViewId, String traitName) {
        TextView traitScore = findViewById(textViewId);
```

```
        traitScore.setText(String.format(Locale.getDefault(), "%.2f",
mainTraitScores.get(traitName)));

        // Hidden reset: long press on Openness
        if (traitName.equals("Openness")) {
            traitScore.setOnLongClickListener(v -> {
                resetScoresAndReload();
                return true;
            });
        }
    }

    /** Recomputes scores and refreshes the interface. */
    private void resetScoresAndReload() {
        SharedPreferences prefs = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
        prefs.edit().clear().apply();
        loadOrGenerateSubTraits();
        computeMainTraitScores();
        displayMainScores();
    }

    /** Opens a detailed pop-up dialog for sub-trait display. */
    public void onTraitClick(View view) {
        String traitName = view.getTag().toString();
        showTraitDialog(traitName);
    }

    private void showTraitDialog(String traitName) {
        final Dialog dialog = new Dialog(this);
        dialog.setContentView(getDialogLayout(traitName));

        TextView title = dialog.findViewById(R.id.trait_title);
        if (title != null) title.setText(traitName);

        SharedPreferences prefs = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
        TableLayout table = dialog.findViewById(R.id.trait_table);

        if (table != null) {
            String[] subTraits = traitToSubTraits.get(traitName);
            for (int i = 0; i < subTraits.length && i <
table.getChildCount(); i++) {
                TableRow row = (TableRow) table.getChildAt(i);
                if (row.getChildCount() >= 2) {
                    ((TextView) row.getChildAt(0)).setText(subTraits[i]);
                    float score = prefs.getFloat(traitName + "_" +
subTraits[i], 0.5f);
                    ((TextView)
row.getChildAt(1)).setText(String.format(Locale.getDefault(), "%.2f",
score));
                }
            }
        }

        dialog.show();
        Window window = dialog.getWindow();
        if (window != null) {
```

```
            window.setLayout((int)
(getResources().getDisplayMetrics().widthPixels * 0.9),
                    WindowManager.LayoutParams.WRAP_CONTENT);
        }
    }

    private int getDialogLayout(String trait) {
        switch (trait) {
            case "Openness": return R.layout.dialog_trait_openness;
            case "Conscientiousness": return
R.layout.dialog_trait_conscientiousness;
            case "Extraversion": return R.layout.dialog_trait_extraversion;
            case "Agreeableness": return
R.layout.dialog_trait_agreeableness;
            case "Neuroticism": return R.layout.dialog_trait_neuroticism;
            default: return R.layout.dialog_trait_openness;
        }
    }
}
```

Code 2 – PersonalityTraitsActivity.java

4.3 Gamepersonality.XML Layout

User Interface Layout for the Personality Games Module

The file Gamepersonality.xml defines the graphical user interface (GUI) for the *Personality Games* module within the Android application.
This layout serves as the central visual hub where all Unity-based serious games are presented to the user in a structured, grid-based format.
It integrates responsive design principles, consistent visual hierarchy, and flexible scalability to accommodate multiple game entries.

Structure Overview:

1. Root ConstraintLayout:

Acts as the main container for the entire interface.

Applies a custom background (@drawable/background) and uses constraint-based positioning for precise element alignment across different screen sizes.

2. Title Section:

A TextView positioned at the top of the layout displays the title “GAMES”.

Styled with a bold custom font, size 32sp, and thematic color #FF6D00, creating a visually engaging and thematic header.

3. Scrollable Games Grid:

Implemented using a ScrollView to ensure that all game items remain accessible on devices of various sizes and orientations.

Contains a nested ConstraintLayout organizing games into horizontal rows for a balanced, structured display.

4. Game Items:

Each game is represented by a vertical LinearLayout containing:

Icon (ImageView): Displayed within a circular frame (@drawable/glass_circle_background) for a polished, modern appearance.

Title (TextView): Displays the name of the Unity game (e.g., Mindful Escape, Other World, Coin Catcher).

Description Button (Button): Opens a popup dialog with information about the game's concept, objective, and its associated Big Five personality traits.

Each column's width is defined using percentage constraints (layout_constraintWidth_percent="0.33") ensuring exactly three games per row.

5. Rows of Games:

Row 1: Mindful Escape, Other World, UnboxIt.

Row 2: Coin Catcher, Fishy Catcher, Time Travel.

Row 3: Which Way.

Additional rows can be easily added by duplicating the existing structure to accommodate new Unity-based serious games.

4.3.1 Key Features

Responsive Design: The use of ConstraintLayout and percentage-based dimensions guarantees consistent alignment and scaling on both smartphones and tablets.

Visual Consistency: Each game item follows the same layout style (icon → title → button), ensuring a coherent and intuitive user experience.

Scalability: The modular row-based design allows seamless inclusion of additional games without altering the layout logic.

Integration with Activity Code: Each button and icon is bound to event handlers defined in PersonalityGamesActivity.java, enabling:

Direct launching of Unity games using their respective package names.

Triggering descriptive popups that link gameplay to specific Big Five personality constructs.

4.3.2 XML Implementation

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/root_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".ui.activities.PersonalityGamesActivity">

    <!-- Title -->
    <TextView
        android:id="@+id/title_games"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:fontFamily="@font/quicksand"
        android:gravity="center"
        android:text="GAMES"
        android:textColor="#FF6D00"
        android:textSize="32sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <!-- Games Grid -->
    <ScrollView
        android:id="@+id/games_grid"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:fillViewport="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/title_games">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp">

            <!-- [ Game rows and items as described above ] -->
        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Code 3 – XML definition of the Personality Games user interface.

4.4 Handling Game Results via JSON Integration

A critical feature of the application lies in the data exchange pipeline between the Unity-based games and the Android host system. Each Unity game captures player behavior and outputs a JSON object representing personality trait indicators consistent with the Big Five model.

When gameplay ends, this JSON payload is returned to the Android activity, validated, and processed for storage and analysis.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == GAME_RESULT_REQUEST && resultCode == RESULT_OK &&
data != null) {
        String jsonString = data.getStringExtra("trait_results");
        if (jsonString != null) {
            processTraitScoresFromJson(jsonString);
        }
    }
}
```

Code 4 – Java method for receiving and processing Unity game JSON results.

4.4.1 Explanation

Lifecycle Hook: The method onActivityResult() is automatically invoked whenever a launched Unity activity finishes and returns control to the Android application.

Request Filtering: It checks that the response corresponds to the expected request (GAME_RESULT_REQUEST), ensuring unrelated activities do not trigger the data pipeline.

Result Validation: The method verifies the successful result status (RESULT_OK) and that the Intent contains valid data.

JSON Extraction: The JSON string (trait_results) is retrieved and parsed. This object contains:

Main Big Five trait values (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism).

Sub-trait breakdowns (e.g., Trust, Altruism, Morality).

Optional gameplay metadata such as reaction time and decision count.

Delegation for Processing: Once validated, the JSON data is passed to processTraitScoresFromJson(), which handles:

JSON parsing and object creation.

Local or backend storage of personality results.

UI updates in PersonalityTraitsActivity to reflect the latest computed values.

4.4.2 Role in the System

This mechanism forms the bridge between gameplay and personality profiling:

Unity games → generate behavioral and psychological data.

Android application → receives, validates, and structures this data.

Backend (Flask + PostgreSQL) → persists the results for long-term research analysis.

Centralizing JSON handling within onActivityResult() ensures consistency, scalability, and robustness as additional games are added to the ecosystem.

4.5 The Integrated Data Pipeline: From Gameplay to Persistent Storage

The integration architecture follows a unidirectional data flow:

Unity → Android → Flask → PostgreSQL

4.5.1 System Components

Unity Engine – Generates raw behavioral and personality-related metrics during gameplay.

Android Application – Acts as an intermediary, capturing data and managing communication with the backend.

Flask Server with PostgreSQL Database – Provides persistent storage for user-specific trait profiles.

4.5.2 Pipeline Characteristics

Secure and Scalable: The system design prevents unauthorized data manipulation while allowing new games or metrics to be integrated easily.

Implicit Data Capture: Personality indicators are collected automatically during gameplay, without requiring additional user input.

Research-Oriented: This flow supports long-term behavioral tracking and user modeling within the gamified framework of the study.

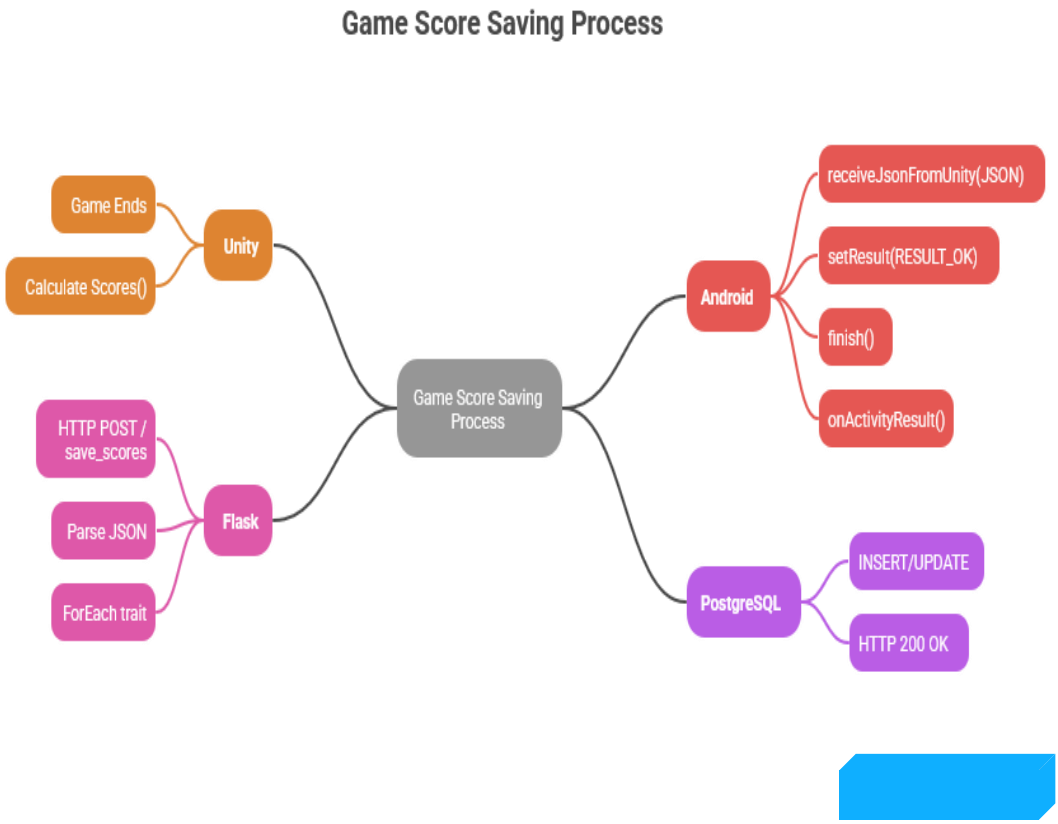


Figure 10: sequence diagram

4.6 Data Generation and Preparation in Unity

The Unity environment plays a fundamental role in generating and structuring personality-related gameplay data. Each Unity-based serious game is designed not only for entertainment but also to infer player traits through embedded behavioral metrics. Once a gameplay session concludes, Unity processes and transmits this data in a structured and secure manner to the Android host application.

4.6.1 Data Calculation and JSON Serialization in Unity

When the game ends:

A dedicated C# class named MetricsManager executes the method OnGameEnd().

This method calculates all personality trait scores (e.g., Cooperation, Anger).

The scores are then normalized—scaled into a consistent range (typically 0 to 1) to ensure comparability across games and users.

Serialization to JSON:

The results are stored in a structured data object.

Unity's built-in JsonUtility.ToJson() method converts this object into a JSON string.

JSON (JavaScript Object Notation) provides a lightweight, standardized format for transferring structured data between systems.

Sending data to Android:

Unity does not directly handle network requests on Android due to security and permission constraints.

Instead, Unity communicates with the Android application using the AndroidJavaClass and AndroidJavaObject APIs.

These APIs allow Unity to invoke Java methods directly within the Android environment—effectively bridging the two systems.

The call currentActivity.Call("receiveJsonFromUnity", jsonData) sends the JSON payload to the Android host app for further processing.

Key Technical Insight:

AndroidJavaClass provides access to Java classes within Unity's C# runtime.

currentActivity represents the currently running Android Activity, which receives the JSON string.

This design cleanly separates system responsibilities:

Unity → Gameplay and data generation.

Android → Communication and backend interfacing.

Flask → Data storage and analysis.

Key Implementation Detail: Unity delegates data transfer responsibilities to Android, avoiding complex permission handling and potential security issues. This separation ensures a modular architecture where Unity focuses on gameplay and data generation, while Android handles secure communication and persistence.

```
// File: MetricsManager.cs
public class MetricsManager : MonoBehaviour {
    public void OnGameEnd() {
        // Calculate all trait scores (e.g., Cooperation, Anger)
        CalculateFinalMetricsValues();

        // Serialize the data object into a JSON string
        string jsonData = JsonUtility.ToJson(personalityData);

        // Call the method in the parent Android application
        SendDataToAndroid(jsonData);
    }

    private void SendDataToAndroid(string jsonData) {
        // Use Unity's AndroidJavaClass to get the current Android Activity
```

```
        using (AndroidJavaClass unityPlayer = new
AndroidJavaClass("com.unity3d.player.UnityPlayer"))
        using (AndroidJavaObject currentActivity =
unityPlayer.GetStatic<AndroidJavaObject>("currentActivity")) {
            // Call a method named 'receiveJsonFromUnity' in the Android
activity
            currentActivity.Call("receiveJsonFromUnity", jsonData);
        }
    }
}
```

Code 5 – Initiating the Data Transfer

4.6.2 Data Bridging via the Android Native Layer

Once Unity finishes a game, the captured personality data must be transmitted to the main Android app. This occurs through a **bridging mechanism** implemented in the Android activity that launched the Unity game. The process is as follows:

Unity Callback: Unity invokes a Java method called receiveJsonFromUnity, passing the generated JSON payload.

UI Thread Handling: The Android app ensures thread safety by executing all UI-related operations inside runOnUiThread.

Debug Logging: The JSON payload is printed using:

```
Log.d("UNITY_TO_ANDROID", "Received JSON: $jsonData")
```

Optional Parsing: Using Gson, the JSON can be deserialized into a Kotlin data class:

```
val gson = Gson()
val personalityData = gson.fromJson(jsonData, PersonalityData::class.java)
```

Returning Data: The data is packed into an Intent and returned to the previous activity:

```
val resultIntent = Intent().apply {
    putExtra("trait_results", jsonData)
}
setResult(Activity.RESULT_OK, resultIntent)
finish()
```

AppKotlin Bridge Class: UnityHostActivity.ktroach:

Purpose: This activity serves as the communication bridge between Unity and the Android host.
It ensures that Unity-generated data is properly received, logged, optionally parsed, and safely returned to the main Android activity for further processing.

```
class UnityHostActivity : AppCompatActivity() {

    fun receiveJsonFromUnity(jsonData: String) {
        runOnUiThread {
            Log.d("UNITY_TO_ANDROID", "Received JSON: $jsonData")
            val gson = Gson()
```

```
            val personalityData = gson.fromJson(jsonData,
PersonalityData::class.java)

            val resultIntent = Intent().apply {
                putExtra("trait_results", jsonData)
            }
            setResult(Activity.RESULT_OK, resultIntent)
            finish()
        }
    }
}
```

4.6.3 Data Persistence via the Flask API and PostgreSQL

The final stage of the pipeline involves the Flask backend receiving and storing the personality data transmitted by the Android app.

```
from flask import Flask, request, jsonify
import psycopg2

app = Flask(__name__)

def connect_db():
    return psycopg2.connect(
        host="vsgate-s1.dei.isep.ipp.pt",
        port=10706,
        database="ISEP_DB",
        user="postgres",
        password="RWtvM9lsazUs"
    )

@app.route('/save_scores', methods=['POST'])
def save_scores():
    try:
        full_data = request.get_json()
        user_id = full_data.get("playerID")
        traits_data = full_data.get("traits", {})

        if not user_id or not traits_data:
            return jsonify({"error": "Missing user_id or traits data"}),
400

        conn = connect_db()
        cur = conn.cursor()

        for trait_name, score in traits_data.items():
            cur.execute("""
                INSERT INTO personality_scores (user_id, trait, score)
                VALUES (%s, %s, %s)
                ON CONFLICT (user_id, trait)
                DO UPDATE SET score = EXCLUDED.score;
            """, (user_id, trait_name, score))

        conn.commit()
        cur.close()
        conn.close()
```

```
            return jsonify({"status": "success"}), 200

        except Exception as e:
            return jsonify({"error": str(e)}), 500
```

Code 6 – Flask API Endpoint

Key Detail: The backend iterates over each trait dynamically, allowing easy addition of new metrics without code modification. This flexibility supports future expansions of personality dimensions.

4.7 Summary of the Architectural Choice

This multi-stage architecture provides a modular, secure, and scalable solution for implicit personality data acquisition.

Table 2: multi-stage architecture

Component	**Responsibility**
Unity	Generates and serializes gameplay data.
Android	Receives, validates, and forwards data securely.
Flask + PostgreSQL	Persists structured personality profiles.

Advantages:

Security: Android manages networking and permissions safely.

Reliability: Handles connection issues and retries gracefully.

Separation of Concerns: Each system performs a specialized role.

Scalability: New traits or games can be added without re-engineering the full pipeline.

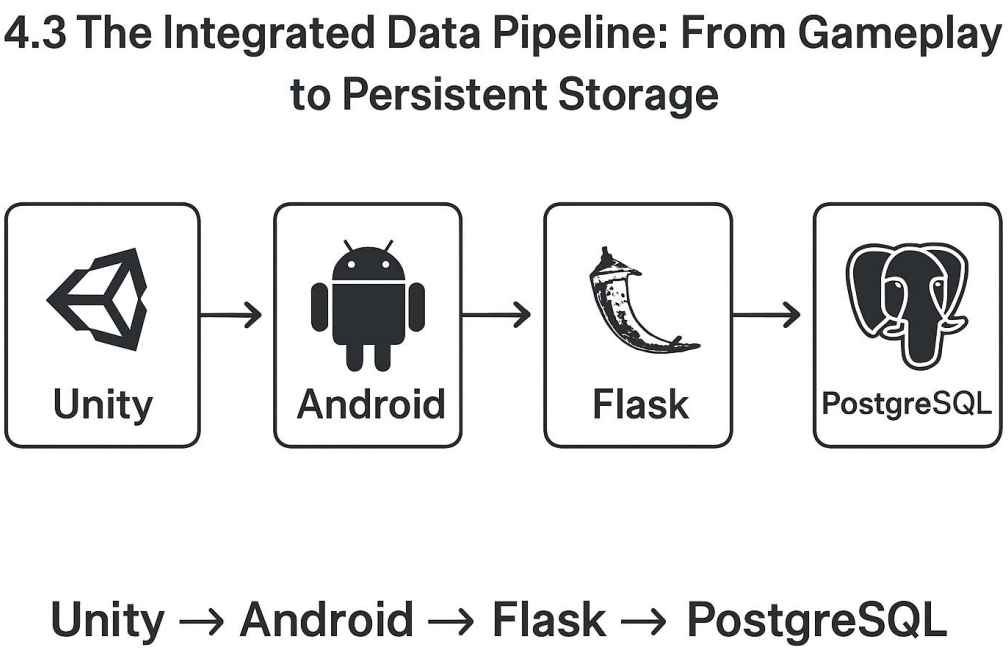


Figure 11: Pipeline integration

4.8 Algorithm Implementation for Trait Calculation

New algorithms were implemented within the **MetricsManager** class to compute each of the five personality traits based on specific in-game behaviors. Each algorithm converts raw behavioral metrics into a normalized score ranging from **0.0 to 1.0**.

4.8.1 Trait Calculation Process

The process follows three main steps:

1.Raw Metric Calculation: For each trait, raw scores are derived from in-game actions. Examples include:

Cooperation: Helping other players or NPCs.

Anger: Quick, aggressive interactions with NPCs.

Morality: Decisions in moral dilemmas (honest vs. deceptive trades).

Altruism: Assisting NPCs without expectation of reward.

Modesty: Avoiding self-praise or recognition.

2.Normalization: Raw scores are normalized to the [0, 1] range using a consistent formula:

$$\text{NormalizedScore} = \text{Clamp01}\left(\frac{\text{RawScore} - \text{Min}}{\text{Max} - \text{Min}}\right)$$

3.Serialization: The final scores are stored in a structured data model and serialized to JSON for downstream use.

Example JSON Output:

```
{
  "playerID": "pseudonymized_id_12345",
  "sessionID": "session_67890",
  "traits": {
    "Cooperation": 0.85,
    "Anger": 0.22,
    "Morality": 0.90,
    "Altruism": 0.78,
    "Modesty": 0.65
  },
  "generalMetrics": {
    "totalGameTime": 354.2,
    "totalTradesDone": 8
  }
}
```

This output allows the Android application and backend API to process the data without modifications, as they handle generic key-value pairs.

4.9 Implementation of a Multi-Trait Profiling Framework

The framework was extended from a single-trait implementation (Cooperation) to model the full **Agreeableness domain**, including five key facets defined by the **Five-Factor Model (FFM)** [Costa & McCrae, 1992]:

- Cooperation
- Morality
- Altruism
- Modesty
- Anger (reverse-scored)

This extension followed a **consistent algorithmic pattern** to ensure scalability and maintainability:

1. Capture in-game behaviors.
2. Compute raw scores for each trait.
3. Normalize scores to a [0, 1] range.

4. Store in a structured data model (**TraitScores**) for JSON serialization.

4.9.1 Unified Data Model

The **PersonalityData** class now contains a structured trait container:

```
[System.Serializable]
public class PersonalityData
{
    public string playerID;
    public string sessionID;
    public TraitScores traits;
    public GeneralGameMetrics generalMetrics;
}

[System.Serializable]
public class TraitScores
{
    public float Cooperation;
    public float Morality;
    public float Altruism;
    public float Modesty;
    public float Anger;
}
```

4.9.2 Examples of In-Game Behavior Mapping

Table 3: Game Behavior Mapping

Trait	Example Behavior	Correlation
Cooperation	Helping NPCs genuinely	Positive
Morality	Choosing honest trade options	Positive
Altruism	Giving hints without reward	Positive
Modesty	Downplaying own contribution	Positive
Anger	Quick aggressive dialogue selections	Positive

4.9.3 Justification for Multi-Trait Profiling

Relying on a single trait can lead to misinterpretation of user behavior. Multi-trait profiling provides a richer understanding of the user's psychological drivers:

- Users with the same Cooperation score may behave differently depending on their **Anger** or **Morality** scores.
- Incorporating multiple traits enables the **Group Recommender System (GRS)** to:
    - Perform more accurate clustering of users.
    - Resolve preference conflicts based on the underlying behavioral motivations.
    - Improve personalization and satisfaction in group recommendations.

Conclusion: Multi-trait profiling captures the complexity of human behavior, allowing for more precise, flexible, and human-aware group recommendations.

5 Conclusions and Future Directions

5.1 Overview

This dissertation focused on extending a tourism-oriented Group Recommender System (GRS) prototype through the integration of implicit personality data collection using serious games developed in Unity. The work addressed a key technical component within the larger ATT project — the Android-side implementation responsible for receiving, processing, and storing personality trait scores generated during gameplay.

The contributions primarily centered on Android integration, data flow management, and backend connectivity, enabling Unity-generated data to be seamlessly captured, displayed, and stored for later use in recommendation logic.

5.2 Implemented Contributions

The main technical achievements of this work include:

- **Design and Development of Android Interfaces:** Created responsive and user-friendly screens to display **Big Five** personality traits and sub-traits.
- **Unity–Android Integration via JSON:** Implemented a structured JSON communication protocol for transferring personality scores between Unity games and the Android application.
- **Backend Development:** Designed a simple yet robust **Flask + PostgreSQL** backend API for storing and managing personality scores.
- **Data Visualization:** Provided intuitive interfaces and popup components to present sub-trait details interactively.
- **Data Pipeline Completion:** Established a full end-to-end data path — *from gameplay → JSON → Android → database → visual display*.

It is important to note that the Unity serious games were not redesigned or redeveloped; this work focused exclusively on the Android integration layer and data management pipeline.

The successful integration demonstrates that even a small technical step can significantly enhance system robustness, enabling the transition from questionnaire-based personality profiling to dynamic, behavior-based analysis.

5.3 Technical Challenges and Resolutions

Table 4: Technical Challenges and Resolutions

Challenge	**Resolution**
Unity-to-Android data transfer	Implemented JSON wrapper classes in Unity and used Intents to return data to the Android activity.
Package name mismatches	Added fallback logic to redirect users to the Play Store if a game package was missing.
Trait storage consistency	Used ON CONFLICT UPSERT in PostgreSQL to update existing trait values reliably.
Visual clarity and responsiveness	Applied Material Design principles for modern and accessible layouts, icons, and trait pop-ups.

5.4 Limitations

While the results confirm the technical feasibility of the integration, this work has several limitations:

- The Unity games were reused in their original form; no modifications were made to gameplay or scoring logic.
- The Android app remains a **prototype**, tested only in local and controlled environments.
- There is currently **no live connection** with an operational Group Recommender System.
- User evaluation and large-scale testing were not conducted.
- Personality data is limited to a subset of traits, focusing primarily on the **Agreeableness** dimension.

These constraints define the scope of the dissertation while providing clear opportunities for future research and expansion.

5.5 Future Directions

Building upon the current implementation, several development paths can be pursued to evolve the system into a fully functional, adaptive tourism recommender platform:

1. **Public Deployment:** Finalize and publish the Android application on the **Google Play Store**, including proper user consent forms, privacy policies, and multilingual support.
2. **Full GRS Integration:** Connect personality trait data dynamically to the operational recommender system, replacing static questionnaire inputs.
3. **Enhanced Trait Modeling:** Collaborate with psychologists to refine sub-trait definitions and improve the behavioral accuracy of personality inference.
4. **Expanded Game Portfolio:** Introduce additional Unity mini-games to capture broader psychological dimensions (e.g., trust, leadership, curiosity).
5. **Real-Time Synchronization:** Implement direct, low-latency communication between Unity, Android, and the backend for immediate data persistence.
6. **Machine Learning Integration:** Employ adaptive algorithms to update recommendations dynamically based on gameplay-derived personality updates.
7. **Cross-Cultural Validation:** Extend testing to international and multicultural user groups to evaluate model generalizability.
8. **Ethical and Legal Compliance:** Reinforce GDPR compliance with secure data encryption, anonymization, and transparent user data handling.

5.6 Summary

Even as a prototype, this work represents a **meaningful step toward personality-aware and group-aware tourism recommender systems**.By combining Android integration, JSON-based communication, and backend data persistence, the project establishes a **technically sound foundation** for future advancements in digital tourism personalization.

The implemented system demonstrates that implicit data collection through **serious games** can replace traditional questionnaires, offering a more engaging, accurate, and user-centered approach to psychological profiling in tourism contexts.

6 References

(Alves et al. 2022) Alves, P., Gomes, D., Rodrigues, C., Carneiro, J., Novais, P. & Marreiros, G. 2022, *Grouplanner: A Group Recommender System for Tourism with Multi-Agent MicroServices*, in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation*, LNCS 13616, Springer, pp. xxx–xxx, viewed 23 September 2025, https://doi.org/10.1007/978-3-031-18192-4_37.

(Alves et al. 2023) Alves, P., Martins, H., Saraiva, P., Carneiro, J., Novais, P. & Marreiros, G. 2023, 'Group Recommender Systems for Tourism: How Does Personality Predict Preferences for Attractions, Travel Motivations, Preferences and Concerns?', *User Modeling and User-Adapted Interaction*, vol. 33, no. 5, pp. 1141–1210, viewed 23 September 2025, https://doi.org/10.1007/s11257-023-09361-2.

(Alves et al. 2024) Alves, P., Trindade, J., Monteiro, G., Saraiva, P., Campos, P., Marreiros, G. & Novais, P. 2024, 'You Want to Play a Game? Detecting Personality Traits with Mobile Minigames', *TechRxiv*, viewed 23 September 2025, https://doi.org/10.36227/techrxiv.171177596.69544930/v1.

(Alslaity & Tran 2021) Alslaity, A. & Tran, T. 2021, 'Users' Responsiveness to Persuasive Techniques in Recommender Systems', *Frontiers in Artificial Intelligence*, vol. 4, 679459, viewed 23 September 2025, https://doi.org/10.3389/frai.2021.679459.

(Android Developers 2024) Android Developers 2024, *Building MVVM Architecture Apps*, Google, viewed 23 September 2025, https://developer.android.com.

(Bartle 2003) Bartle, R. 2003, *Exploring Player Types: Implicit & Explicit Gaming Preferences*, viewed 23 September 2025, https://mud.co.uk/richard/.

(Deterding et al. 2011) Deterding, S., Dixon, D., Khaled, R. & Nacke, L. 2011, 'From Game Design Elements to Gamefulness: Defining "Gamification"', *Proceedings of the 15th International Academic MindTrek Conference*, pp. 9–15.

(Dhelim et al. 2022) Dhelim, S., et al. 2022, 'A Survey on Personality-Aware Recommendation Systems', *Artificial Intelligence Review*, vol. 55, no. 3, pp. 2409–2454, viewed 23 September 2025, https://doi.org/10.1007/s10462-021-10063-7.

(Gope & Jain 2017) Gope, J. & Jain, S.K. 2017, 'A Survey on Solving Cold Start Problem in Recommender Systems', *2017 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, viewed 23 September 2025, https://doi.org/10.1109/CCAA.2017.8229786.

(IEEE 2023) IEEE 2023, 'Gamification Standards for Mobile Systems', *IEEE Software*, vol. 40, no. 1, pp. 35–47.

(Marreiros 2024) Marreiros, G. 2024, *Gamification in Digital Tourism*, Springer.

(McCrae & Costa 2004) McCrae, R.R. & Costa, P.T. 2004, 'A Five-Factor Theory of Personality', in J.P. Oliver & R.W. Robins (eds), *Handbook of Personality Psychology*, pp. xxx–xxx.

(Pereira 2023) Pereira, F. 2023, 'Evaluating Gamification Methods for Personality Profiling in Mobile Contexts', *Proceedings of the ACM CHI Conference*, pp. xxx–xxx.

(PostgreSQL Global Development Group 2024) PostgreSQL Global Development Group 2024, *PostgreSQL Documentation: Database Design and Best Practices*, viewed 23 September 2025, https://www.postgresql.org.

(Rodrigues et al. 2021) Rodrigues, L., Toda, A.M., Oliveira, W., Palomino, P.T., Vassileva, J. & Isotani, S. 2021, 'Automating Gamification Personalization: To the User and Beyond', *arXiv preprint*, arXiv:2104.12092.

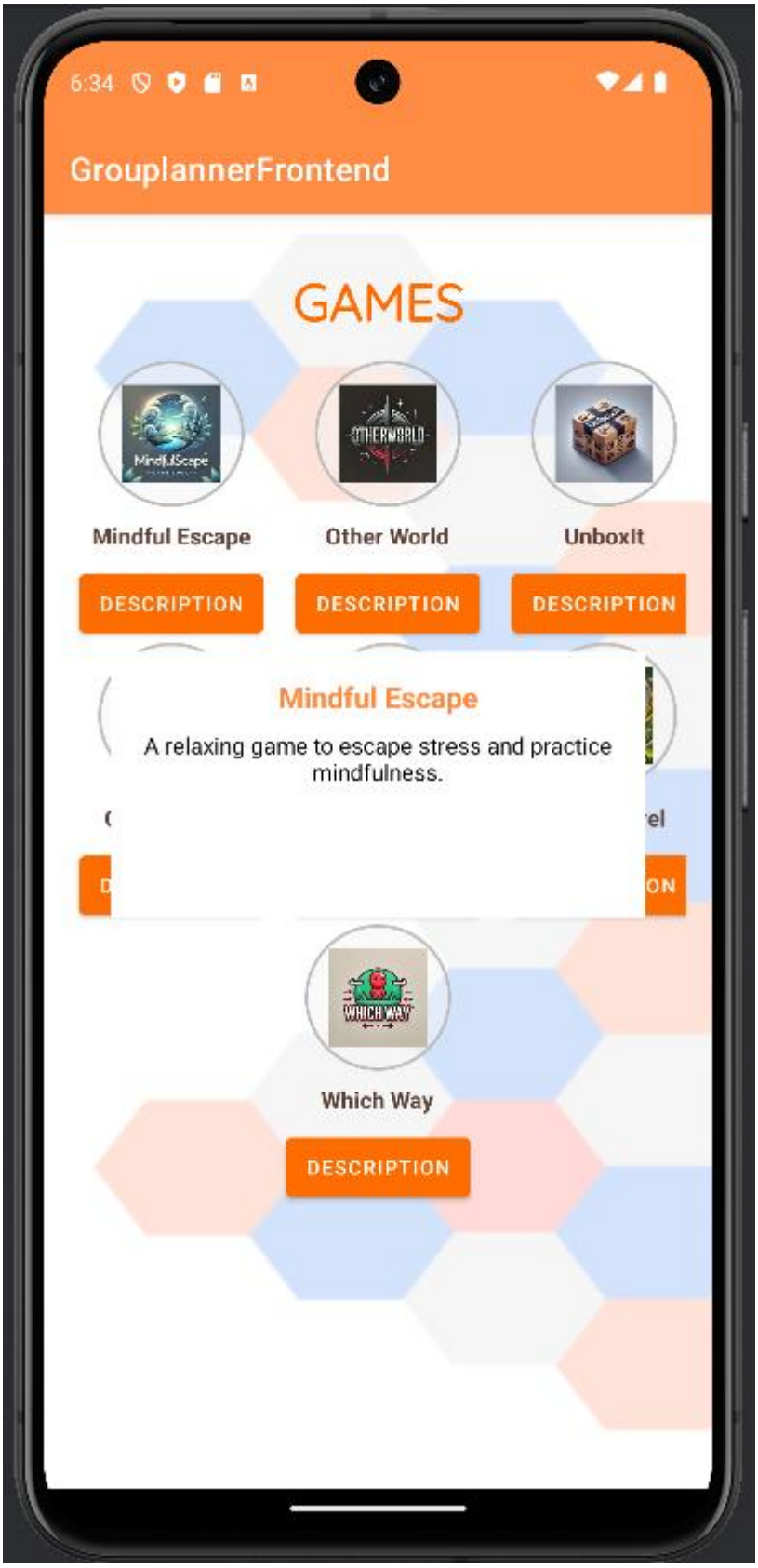
(Unity Technologies 2023) Unity Technologies 2023, *Unity Android Integration Documentation*, Unity Docs.

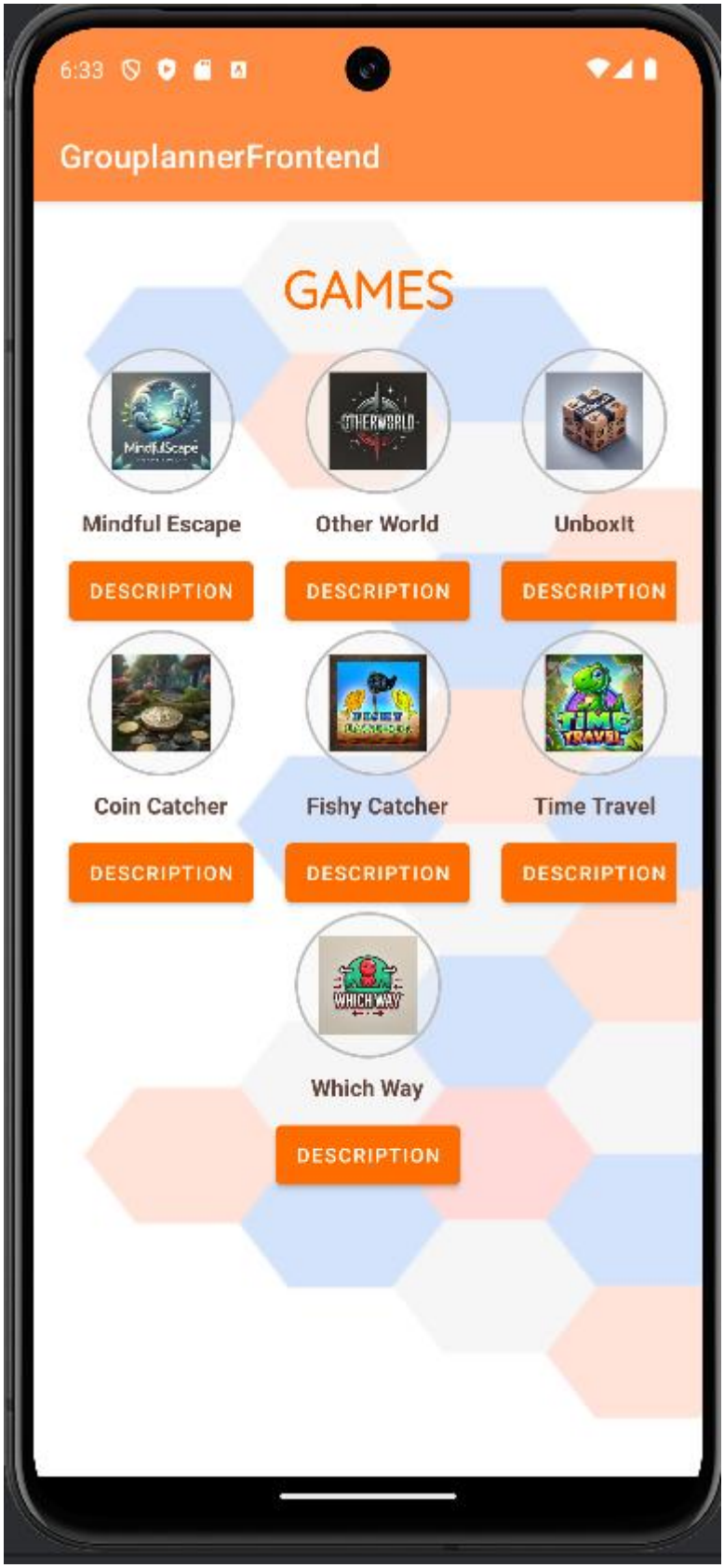
(Van Lankveld et al. 2011) Van Lankveld, G., Spronck, P., Van Den Herik, J. & Arntz, A. 2011, 'Games as Personality Profiling Tools', *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, IEEE, pp. 197–202, viewed 23 September 2025, https://doi.org/10.1109/CIG.2011.6032007.

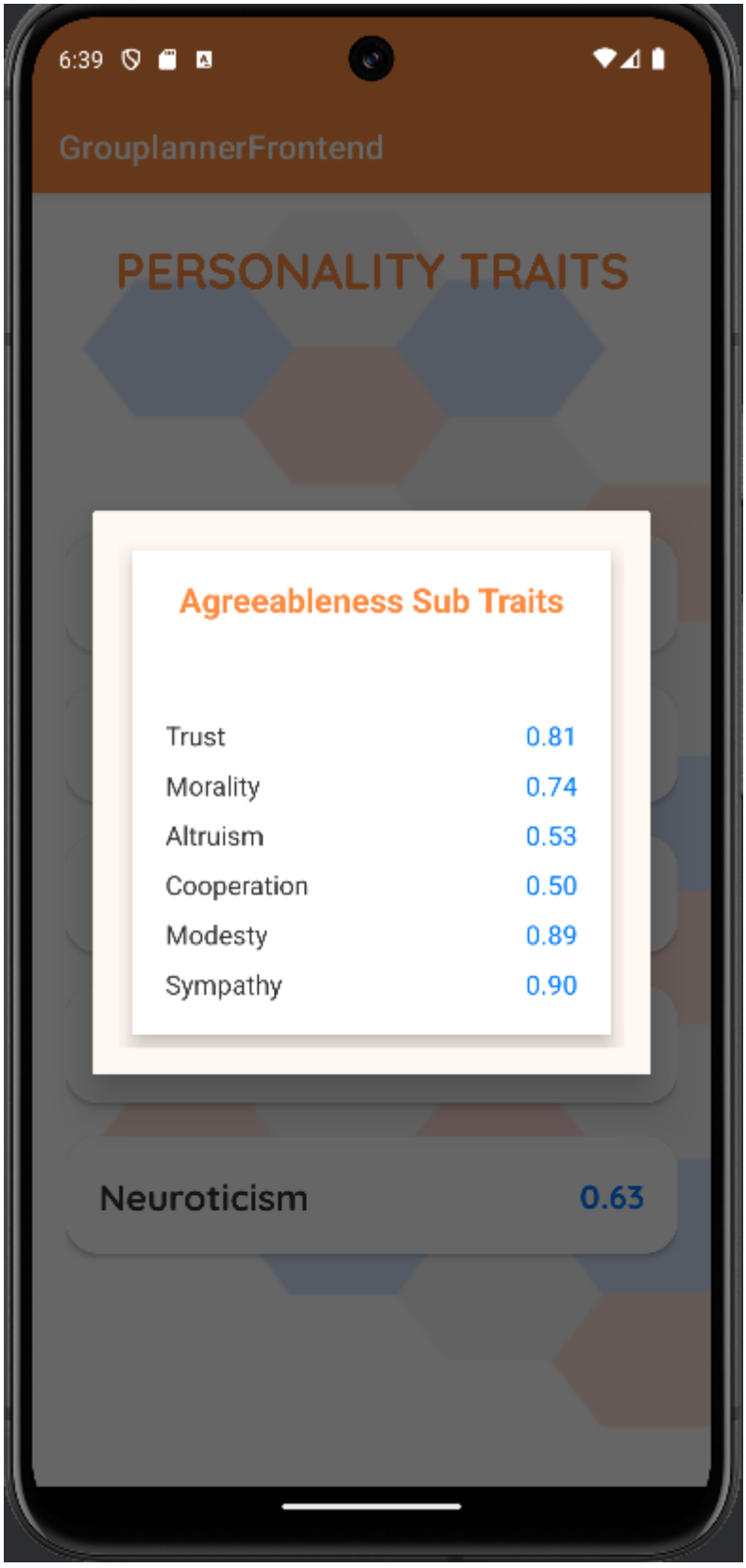
(Xu, Buhalis & Weber 2017) Xu, F., Buhalis, D. & Weber, J. 2017, 'Serious Games and the Gamification of Tourism', *Tourism Management*, vol. 60, pp. 244–256.

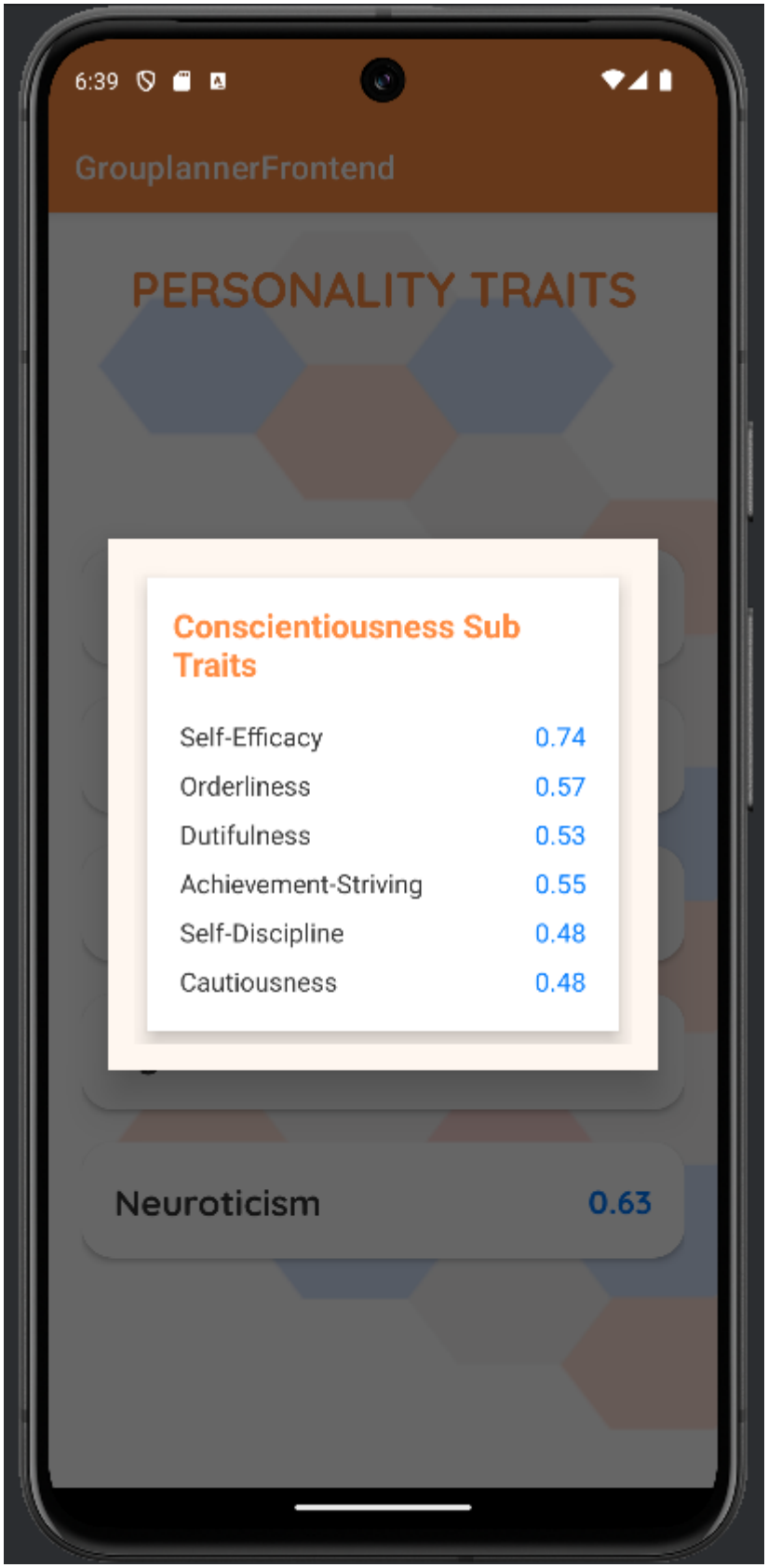
7 Appendix – GRS screens

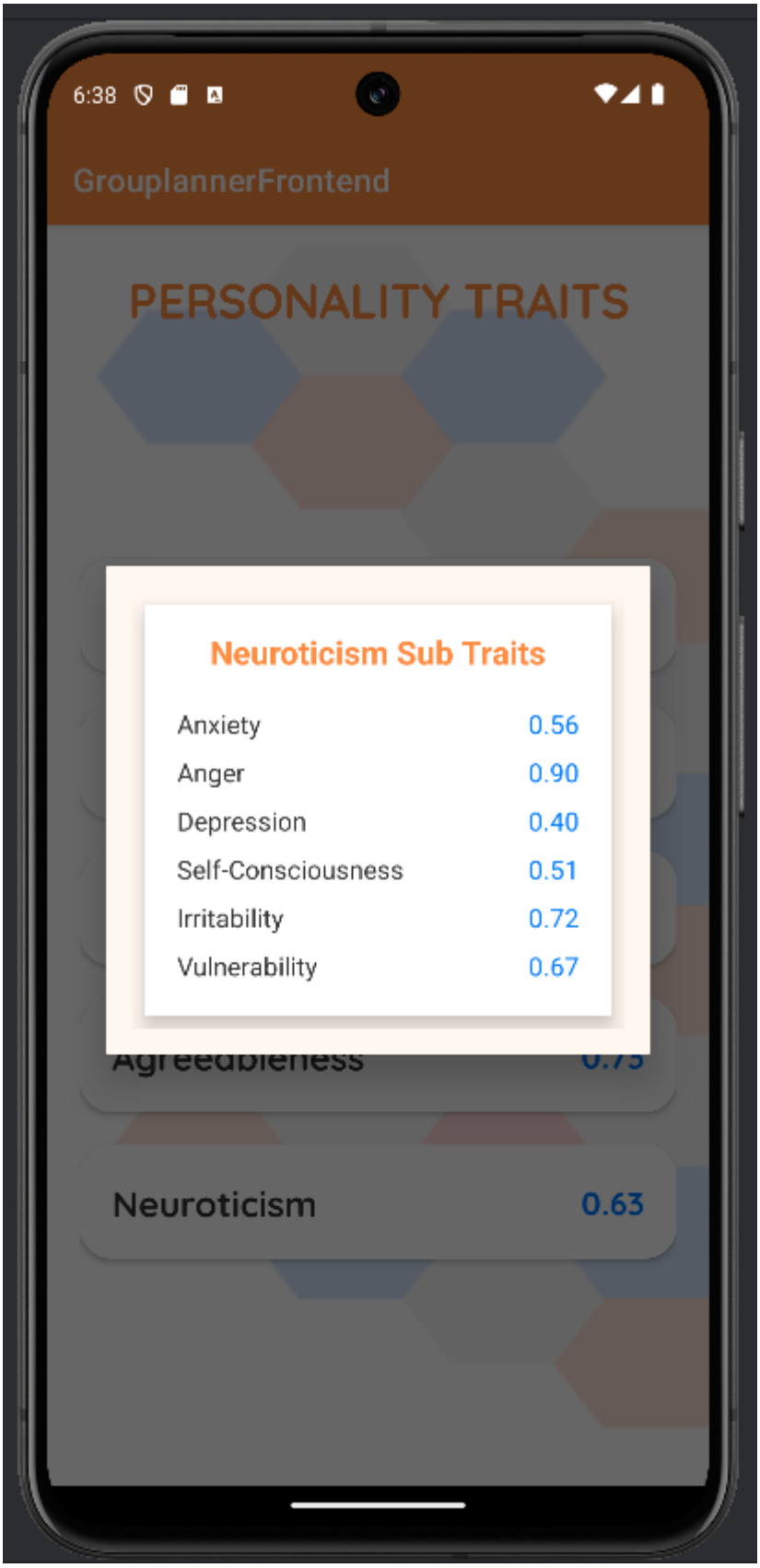

6:26
GrouplannerFrontend
Level
AllXp
TextView
TextView
TextView
TextView
Recommendations
Leaderboard
Trophies
Games
Visited POI
Friends
POI
User Search
Excursion G...
Private Mes...
Profile

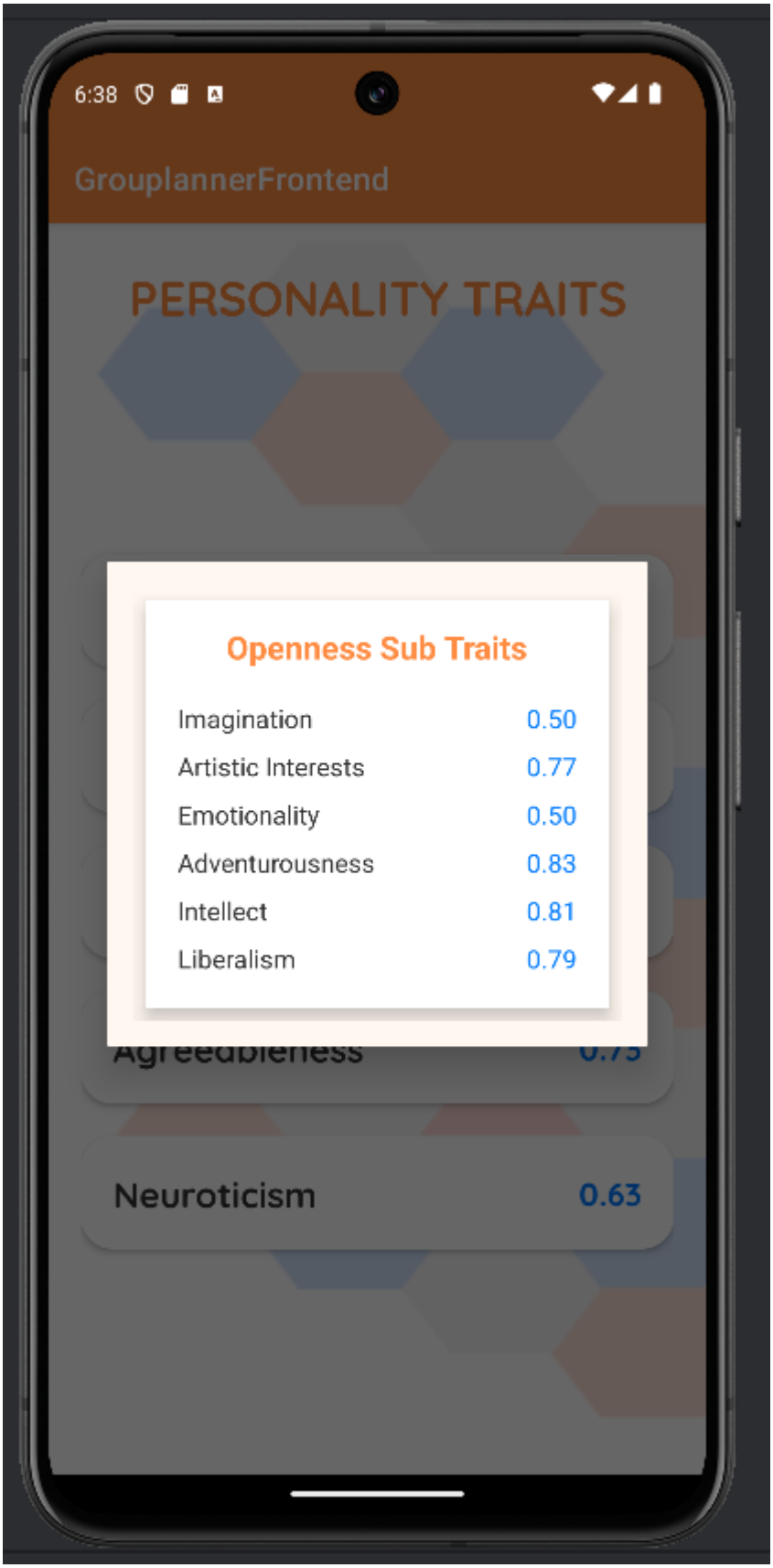

6:34
GrouplannerFrontend
GAMES
Mindful Escape
Other World
UnboxIt
DESCRIPTION
DESCRIPTION
DESCRIPTION
Mindful Escape
A relaxing game to escape stress and practice mindfulness.
Which Way
DESCRIPTION


6:33
GrouplannerFrontend
GAMES
Mindful Escape
DESCRIPTION
Other World
DESCRIPTION
UnboxIt
DESCRIPTION
Coin Catcher
DESCRIPTION
Fishy Catcher
DESCRIPTION
Time Travel
DESCRIPTION
Which Way
DESCRIPTION


6:39
GrouplannerFrontend
PERSONALITY TRAITS
Agreeableness Sub Traits
Trust 0.81
Morality 0.74
Altruism 0.53
Cooperation 0.50
Modesty 0.89
Sympathy 0.90
Neuroticism 0.63


6:39
GrouplannerFrontend
PERSONALITY TRAITS
Conscientiousness Sub Traits
Self-Efficacy 0.74
Orderliness 0.57
Dutifulness 0.53
Achievement-Striving 0.55
Self-Discipline 0.48
Cautiousness 0.48
Neuroticism 0.63


6:38
GrouplannerFrontend
PERSONALITY TRAITS
Neuroticism Sub Traits
Anxiety 0.56
Anger 0.90
Depression 0.40
Self-Consciousness 0.51
Irritability 0.72
Vulnerability 0.67
Neuroticism 0.63


6:38
GrouplannerFrontend
PERSONALITY TRAITS
Openness Sub Traits
Imagination 0.50
Artistic Interests 0.77
Emotionality 0.50
Adventurousness 0.83
Intellect 0.81
Liberalism 0.79
Neuroticism 0.63