School of Science, Engineering and Technology

Assignment 2: Group Project

COSC3062|COSC3063 – iPhone Software Engineering

Lecturer: Mr. Tom Huynh

Students: Bui Minh Duc, S4070921
Nguyen Tuan Minh Khoi, S3995060
Nguyen Huy Hoang, S4041847
Nguyen Danh Bao, S3978319
Nguyen Ngoc Hai, S3978281

ASSIGNMENT DETAILS

Course Name:	iPhone Software Engineering
Subject Code:	COSC2062|COSC2063
Campus:	RMIT Hanoi Campus
Type of the Assignment:	Group
Student Name:	Bui Minh Duc, S4070921 Nguyen Tuan Minh Khoi, S3995060 Nguyen Huy Hoang, S4041847 Nguyen Danh Bao, S3978319 Nguyen Ngoc Hai, S3978281
Assignment Due Date:	Sep 12th, 2025
Submission Date:	Sep 12th, 2025
Page Count:	# (Excluding Cover Pages, Table of Content, References, Appendix, and Graphics)
Word count:	# (Excluding Cover Pages, Table of Content, References, Appendix, and graphics)

TABLE OF CONTENTS

A. INTRODUCTION, OVERVIEW, AND CONTEXT

1. App Overview

"Chaotic Street (Phố Loạn)" is an educational iOS game that addresses one of Vietnam's most considerable social challenges: traffic safety. Inspired with a popular game called *"Crossy Road"*, the team aims to develop a Vietnamese-themed version, combining engaging gameplay with meaningful education about traffic rules and safety practices specific to Vietnam's unique transportation landscape.

2. Goals and Target Audience

Primary Goals:

The primary goal of this game is to reduce traffic accidents in Vietnam through gamified education, targeting Vietnamese smartphone users aged 16-36 who represent both the highest risk group for traffic accidents and the most active mobile gaming demographic.

Secondary Goals:

- Increase awareness of Vietnamese traffic laws
- Promote safer driving and pedestrian habits
- Create cultural awareness about Vietnam's transportation challenges
- Provide an entertaining platform for social impact education

3. Motivation and Inspiration

Thousands of traffic accidents occur in Vietnam annually, many of which are preventable and disproportionately affect young people. Despite government efforts to improve infrastructure and enforce traffic laws, a significant gap remains in public awareness and compliance—especially among youth. Traditional educational campaigns often fail to engage this demographic in meaningful or lasting ways.

This inspired us to approach this with gamified learning. By blending entertainment with practical education, we aim to reach users in a format they already enjoy—mobile games. The success of titles like ***'Crossy Road'*** demonstrated how simple, addictive gameplay can capture wide audiences. We saw an opportunity to take this engaging format and reimagine it in the context of Vietnam - bustling streets, chaotic traffic, and familiar cultural elements.

'Phố Loạn' is born from the idea that learning doesn't have to be boring. By immersing players in a stylized, exaggerated version of Vietnam's traffic environment, we can both entertain and educate. Players won't just be dodging cars for points - they'll be in for real-life lessons about crossing the street safely and understanding pedestrian behaviour within Vietnamese cities.

B. COMPETITOR ANALYSIS

Potential Competitors:

Table 1. *Identified competitors.*

Direct Competitors	Traffic Rush	Generic traffic crossing game without educational content

	Vietnamese Driving Test Apps	Highly educational but dry, following quiz-based format
	Original Crossy Road Frogger	Popular endless runner but no educational value
Indirect Competitors	Disney Crossy Road	Character variety but Western-focused themes
	Pac-Man 256	Similar grid movement but different theme entirely
Educational Game Competitors	Duolingo	Gamified learning but language-focused
	Dragon Box	Educational games but math-focused
	Monument Valley	Puzzle games with cultural themes but no social impact

Killer Features:

Feature: Real Vietnamese traffic laws and pedestrian rules are embedded as facts after the game session, allowing players to reflect on their past games and improve in the next.

It is learning while playing. Learning is in the game, making players internalize safety behaviours subconsciously.

C. IMPLEMENTATION DETAILS

1. Architecture Overview

The application follows the ***Model – View – ViewModel (MVVM)*** architecture pattern, chosen for its clean separation of concerns and excellent compatibility with SwiftUI's reactive programming model. Below is an example of most significant components of the game:

Table 2. *Some of the core components in the game.*

Models	Player.swift	Player position, state, and movement logic
	Vehicle.swift	Vehicle types, movement patterns, Vietnamese characteristics
	Lane.swift	Lane generation, traffic patterns, difficulty scaling
ViewModels	GameEngine.swift	Core game loop, collision detection, state management
	ScoreViewModel.swift	Score management, leaderboard integration
	UserViewModel.swift	User authentication and profile management
Views	GameView.swift	Main gameplay interface with responsive design
	SettingsView.swift	User authentication and profile management

	FetchLeaderBoardView.swift	Leaderboard with interactive charts.

2. Key Technical Components

2.1. Game Engine System

The ‘***GameEngine’*** layer acts as the central orchestrator of the entire gameplay experience, implementing a complex real-time game loop that manages multiple parallel systems while maintaining a stable 60 frames per second performance. Unlike traditional iOS apps, which primarily respond to user input, our ‘***GameEngine***’ creates a simulation environment that continuously processes vehicle movement, collision detection, lane creation, and scoring across multiple threads.

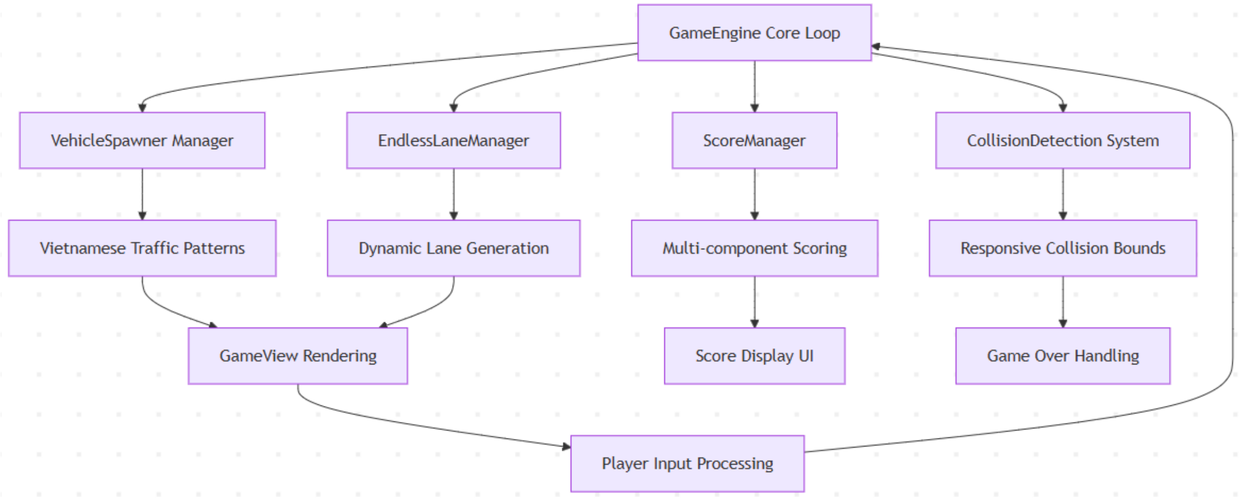

Figure 1. *GameEngine's flow.*

The engine's architecture follows a component-based development pattern, with individual managers responsible for specific aspects of the gameplay while maintaining loose coupling through clearly defined interfaces. This architectural approach allows each team member to develop their specialized components independently, while ensuring seamless integration through the central ‘***GameEngine***’ orchestrator.

2.2. Endless Lane Generation System

The ‘***VehicleSpawner’*** class implements an advanced traffic simulation system that generates realistic traffic patterns of Vietnam through carefully calibrated probability distributions and timing algorithms. Instead of using a simple random generation method, Spawner considers the individual characteristics of each lane, the current level of difficulty, and the actual traffic flow pattern to create a believable vehicle sequence that reflects the real-life urban traffic situation of Vietnam.

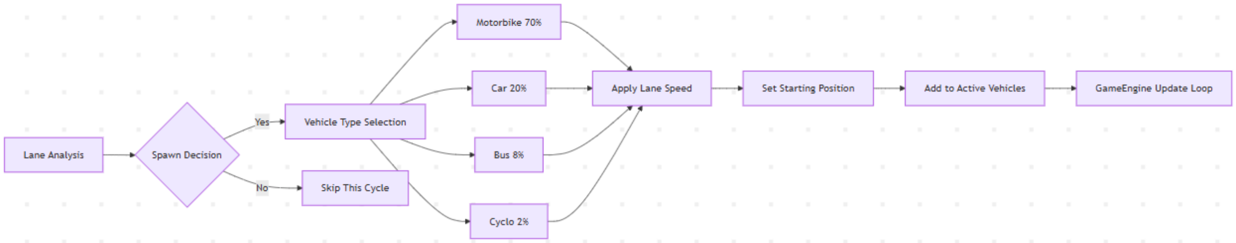

Figure 2. *Lane generation's flow.*

Spawner's decision-making process simultaneously evaluates multiple factors when determining vehicle placement. Lane capacity calculations help prevent overcrowding while maintaining traffic density appropriate to the current level of difficulty. The choice of vehicle type follows the original traffic distribution of Vietnam, with motorbikes accounting for 70% of the vehicles produced, cars accounting for 20%, buses accounting for 8%, and traditional bicycles accounting for the remaining 2%. These percentages reflect the actual urban traffic composition of Vietnam, giving players realistic access to the types of vehicles they might encounter in real-life situations.

2.3. Relative Positioning System

The game uses a sophisticated relative positioning system where the player remains stationary while the “world” moves. This creates the endless runner effect with the player fixed at 66% down the screen while maintaining smooth 60 FPS performance.

2.4. Collision Detection System

The collision detection system performs pixel-perfect collision calculations using a responsive bounding box that accounts for different screen sizes and device orientations. Unlike simple rectangular collision detection, our system uses a graduated collision boundary that provides accurate collision feedback while maintaining fair game mechanics across different device configurations.

2.5. Current User

The user in the app is represented by the AppUser struct, which wraps the Firebase User object and stores the user's UID and email. The main logic for managing the user is handled by the ***UserViewModel*** class. When a user signs up or logs in, UserViewModel calls setUser(from:) to create an AppUser instance from the Firebase user and assigns it to its @Published var user.

Then UserViewModel is instantiated as a @StateObject in the root app view, making it available throughout the app as SwiftUI's environment object. Views such as LoginView, SignupView, etc. access the user via @EnvironmentObject. Other classes, like ScoreViewModel, receive the user by @Published

2.6. Authentication

Authentication in this project is implemented using Firebase Authentication and primarily managed through the UserViewModel class. This class provides methods for user sign-up, login, password reset, and password update, along with published state variables for UI feedback.

Signup

Uses createUser(withEmail:password:) provided by FirebaseAuth to register new users. Upon successful registration in Firebase Authentication, it also creates a user document in Firestore and sends an email verification.

Login

Uses signIn(withEmail:password:) to authenticate users. It updates the state and sets the global user on success.

Password Reset

Uses sendPasswordReset(withEmail:) to send a reset link to the user's email.

Password Update

Re-authenticates the user using the credentials and then updates the password using updatePassword(to:).

2.7. Game Instruction

This view serves as an interactive guide to introduce players to the gameplay. Designed as a pop-up or sheet, it presents three instruction pages, each showcasing different types of content to explain the rules and mechanics in a clear and engaging way. Players can swipe through the pages using a **TabView** with **PageTabViewStyle**, while page indicators (dots) at the bottom provide a visual cue of their current position. Besides, navigation buttons labelled “Previous” and “Next” allow players to manually move between pages, making the tutorial both intuitive and easy to follow.

View link: https://imgflip.com/gif/a695bp

2.8. Leaderboard and User Scores

The **FetchLeaderboardView** provides players with a three-tab system to explore different aspects of their gameplay. The **Leaderboard tab** shows global rankings with medals for the top three players, player details such as email, score, and difficulty level, along with refresh and animated trophy features. The **Achievements tab** tracks milestones, displaying locked or unlocked achievements and its completion dates. The **My Scores tab** presents a player’s personal score history, including bar charts of score progression, up to 15 recent results, and clear messaging when no scores are available.

View link: https://imgflip.com/gif/a69ht9

D. DESIGN AND USER EXPERIENCE

3. Visual Appeal

The app features a Vietnamese-inspired design language, with a colour palette containing mostly green, red, and yellow Vietnamese flag colours, together with rounded and friendly fonts promoting safety awareness. Additionally, the game supports dark and light mode, fully support appropriate contrast ratios.

E. KNOWN BUGS AND PROBLEMS

- Changing the interface language of the application directly in the app is currently not possible, due to a limitation of the text renderer for apps that use the native Localizable string catalogue.
- The game does not scale 100% perfect in larger screens, owing to complex tiles organization.

F. CONCLUSION

1. Technical Achievement and Learning Outcomes

"Chaotic Road (Phố Loạn)" project demonstrated successful integration of complex iOS development concepts including real-time game loop implementation, responsive user interface design, multi-device compatibility, and cloud-based data persistence within a cohesive educational gaming platform. The MVVM architecture implementation provides a solid foundation for ongoing development while maintaining clear separation of concerns that facilitated effective collaboration.

The project's technical challenges, particularly the responsive design implementation for iPad compatibility and performance optimization across device generations, provided valuable learning experiences in iOS development best practices while delivering practical solutions applicable to future mobile application development projects.

2. Social Impact and Educational Value

Beyond technical achievement, ***"Chaotic Road (Phố Loạn)"*** project addresses genuine social challenges through culturally sensitive, contextually relevant educational content that acknowledges the unique characteristics of Vietnamese urban traffic environments. The application's combination of engaging gameplay mechanics with meaningful educational content creates a promising platform for ongoing traffic safety awareness development that extends beyond traditional approaches.

G. PROJECT RESPONSIBILITIES

Member name	SID	Responsibilities
Nguyen Ngoc Hai	S3978281	• Developed the functions for controlling the game state, player movement, vehicle spawning, collision detection and scoring. • Co-developed the road system and lane generation
Nguyen Danh Bao	S3978319	• Implemented the user interface design for authentication and leaderboard functions
Nguyen Tuan Minh Khoi	S3995060	• Developed the logic for user score and leaderboard + the achievement system • Developed user management and storage with Firebase and Cloudinary
Nguyen Huy Hoang	S4041847	• Designed the assets for vehicles: cars, buses, motorbikes etc. • Designed the background music and collision sound effect • Co-developed the road system and lane generation
Bui Minh Duc	S4070921	• Designed the splash screen, Settings, dark mode and app localisation • Co-developed AudioManager and user interface design with other team members

H. REFERENCES

Traffic Rush

Reference resources used in developing the application

https://github.com/gadirom/SwiftUI-Game

A sample arcade game developed using SwiftUI with manually managed @State and game loop using Timer.

https://github.com/kieranb662/CollisionDetectionExamples

Collision detection between Point and Circle, Circle and Rectangle, and between Rectangles

https://github.com/amosgyamfi/open-swiftui-animations

Animation using SwiftUI

I. APPENDICES

APPENDIX A: Link to GitHub Repository

https://github.com/RMIT-Vietnam-Teaching/cosc3062-cosc3063-ios-assignment-2-hanoi-group-1

APPENDIX B: Link to video presentation on YouTube

https://youtu.be/YKWQ9G4lviQ