ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση ολοκληρωμένου παιχνιδιού πολλαπλών παικτών σε Python τύπου Ludo

Διπλωματική Εργασία

Τραγιάννης Νικόλαος

Επιβλέπων Καθηγητής

Δρ. Μηνάς Δασυγένης

Αναπληρωτής Καθηγητής

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

ΚΟΖΑΝΗ ΟΚΤΩΒΡΙΟΣ 2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Design of an integrated multiplayer Ludo game in Python

Diploma Thesis

Tragiannis Nikolaos

Supervising Professor

Dr. Minas Dasygenis

Associate Professor

Robotics, Embedded and Integrated Systems Laboratory

KOZANI OCTOBER 2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Σχεδίαση ολοκληρωμένου παιχνιδιού πολλαπλών παικτών σε python τύπου Ludo" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος Δρ. Μηνά Δασυγένη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: Τραγιάννης Νικόλαος

Περίληψη

Το gaming έχει ενσωματωθεί στην καθημερινότητα των ανθρώπων, ιδιαίτερα των παιδιών, δημιουργώντας την ανάγκη για αξιοποίηση του χρόνου που περνούν μπροστά στον υπολογιστή. Παράλληλα, τα άτομα με αναπηρία, που συχνά αντιμετωπίζουν δυσκολίες στην κοινωνική αλληλεπίδραση λόγω των περιορισμών τους, αναζητούν νέους τρόπους διασκέδασης και κοινωνικοποίησης. Σε αυτό το πλαίσιο, η ψηφιοποίηση των επιτραπέζιων παιχνιδιών αναδεικνύεται ως μια ελκυστική λύση, προσφέροντας ευκαιρίες για διασκέδαση, κοινωνικοποίηση και μάθηση.

Η παρούσα διπλωματική εργασία εστιάζει στη σχεδίαση και υλοποίηση μιας διαδικτυακής έκδοσης του δημοφιλούς επιτραπέζιου παιχνιδιού Ludo. Το παιχνίδι επιτρέπει τη συμμετοχή πολλαπλών παικτών μέσω διαδικτύου, ενώ μέσω της ειδικά σχεδιασμένης ιστοσελίδας, οι χρήστες μπορούν να παρακολουθούν το ιστορικό τους και την αξιολογική τους κατάταξη. Επιπλέον, οι παίκτες έχουν τη δυνατότητα να δημιουργήσουν νέο παιχνίδι ή να συνδεθούν σε υπάρχοντα παιχνίδια που βρίσκονται σε κατάσταση αναμονής, προσφέροντας ευελιξία στον τρόπο συμμετοχής.

Ο κύριος στόχος αυτής της εργασίας είναι η μεταφορά του Ludo στον ψηφιακό κόσμο, αποσκοπώντας σε πολλαπλά οφέλη. Η δυνατότητα για παιχνίδι πολλαπλών παικτών προάγει σημαντικά κοινωνικά οφέλη, ιδιαίτερα για τα άτομα με ειδικές ανάγκες, ενισχύοντας την κοινωνικοποίησή τους και προσφέροντας νέες ευκαιρίες για διαδικτυακή αλληλεπίδραση. Παράλληλα, τα παιδιά μπορούν να αναπτύξουν από νεαρή ηλικία δεξιότητες όπως η εξοικείωση με τους υπολογιστές, η συνεργασία και η συγκέντρωση. Τέλος, αυτή η υλοποίηση μπορεί να αποτελέσει το έναυσμα για την ψηφιοποίηση και άλλων παρόμοιων παιχνιδιών, διευρύνοντας τις επιλογές ψυχαγωγίας και μάθησης στον ψηφιακό κόσμο.

Λέξεις Κλειδιά: Παιχνίδι πολλαπλών παικτών, γραφική διεπαφή, Ludo, Python, WebSockets

Abstract

Gaming has become an integral part of people's daily lives, especially children's, creating a need to utilize the time spent in front of computers productively. At the same time, individuals with disabilities, who often face difficulties in social interaction due to their limitations, are seeking new ways of entertainment and socialization. In this context, the digitization of board games emerges as an attractive solution, offering opportunities for fun, socialization, and learning.

This thesis focuses on designing and implementing an online version of the popular board game Ludo. The game allows multiple players to participate via the internet, while through a specially designed website, users can track their history and ranking. Additionally, players have the option to create a new game or join existing games in waiting status, offering flexibility in how they participate.

The main goal of this work is to transfer Ludo to the digital world, aiming for multiple benefits. The multiplayer feature promotes significant social benefits, particularly for individuals with special needs, enhancing their socialization and offering new opportunities for online interaction. At the same time, children can develop skills from a young age, such as computer familiarity, cooperation, and concentration. Finally, this implementation can serve as a catalyst for the digitization of other similar games, expanding the options for entertainment and learning in the digital world.

Keywords: Multiplayer game, graphical interface, Ludo, Python, WebSockets

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον Δρ. Μηνά Δασυγένη για την στήριξη που μου παρείχε κατά τη διάρκεια υλοποίησης της παρούσας Διπλωματικής εργασίας. Το αποτέλεσμα δεν θα ήταν ίδιο χωρίς τις συμβολές και τη κριτική του. Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου που αφιέρωσαν χρόνο “παίζοντας” μαζί μου κατά τη διάρκεια των δοκιμών και των αναζητήσεων σφαλμάτων (γνωστά και ως bugs).

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1 – Εισαγωγή

1.1 Σκοπός της εργασίας

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο τη σχεδίαση και υλοποίηση ενός ολοκληρωμένου παιχνιδιού πολλαπλών παικτών τύπου Ludo, χρησιμοποιώντας σύγχρονες τεχνολογίες προγραμματισμού και ανάπτυξης διαδικτυακών εφαρμογών. Το έργο αυτό στοχεύει στην εξερεύνηση και εφαρμογή πολλαπλών πτυχών της ανάπτυξης λογισμικού, συμπεριλαμβανομένης της σχεδίασης παιχνιδιών, του δικτυακού προγραμματισμού και της ανάπτυξης διεπαφών χρήστη.

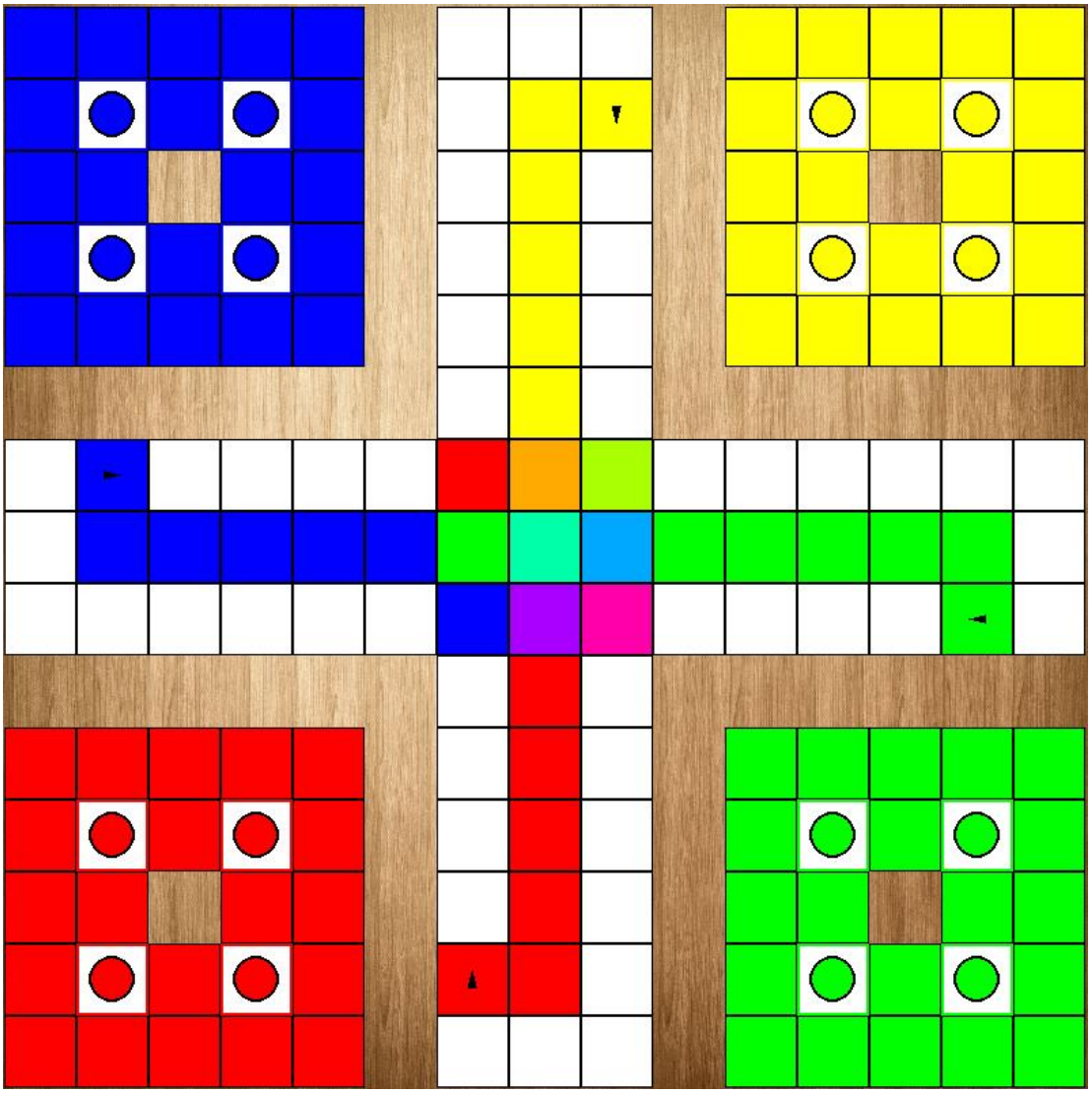
Μέσω αυτής της εργασίας, επιδιώκουμε να ενώσουμε μία πληθώρα τεχνολογιών όπως Python, Pygame, React, Node.js και WebSockets για τη δημιουργία ενός λειτουργικού και διασκεδαστικού παιχνιδιού. Παράλληλα, στοχεύουμε στην εξερεύνηση των προκλήσεων που προκύπτουν κατά την ανάπτυξη παιχνιδιών πολλαπλών παικτών και στην εύρεση αποτελεσματικών λύσεων για αυτές.

1.2 Περιγραφή του παιχνιδιού Ludo

Το Ludo είναι ένα κλασικό επιτραπέζιο παιχνίδι στρατηγικής που έχει τις ρίζες του στο αρχαίο ινδικό παιχνίδι Pachisi. Σχεδιασμένο για δύο έως τέσσερις παίκτες, το Ludo παίζεται σε έναν πίνακα [Εικόνα 1: Το ταμπλό του Ludo] με τέσσερα διαφορετικά χρώματα, ένα για κάθε παίκτη.

Κάθε παίκτης έχει τέσσερα πιόνια που πρέπει να μετακινήσει από την αρχική του θέση, γύρω από τον πίνακα και τελικά στο κέντρο του. Οι κινήσεις καθορίζονται από τη ρίψη ενός ζαριού. Οι παίκτες μπορούν να "χτυπήσουν" τα πιόνια των αντιπάλων τους, στέλνοντάς τα πίσω στην αρχική τους θέση. Ο πρώτος παίκτης που θα φέρει και τα τέσσερα πιόνια του στο κέντρο κερδίζει το παιχνίδι.

Η απλότητα των κανόνων του Ludo σε συνδυασμό με το στοιχείο της στρατηγικής το καθιστούν ιδανική επιλογή για την υλοποίηση ενός ψηφιακού παιχνιδιού πολλαπλών παικτών.



Εικόνα 1: Το ταμπλό του Ludo

1.3 Στόχοι και προκλήσεις

Η παρούσα διπλωματική εργασία έχει ως πρωταρχικό στόχο την ανάπτυξη ενός ολοκληρωμένου συστήματος για το επιτραπέζιο παιχνίδι Ludo σε ψηφιακή μορφή. Το έργο περιλαμβάνει την υλοποίηση ενός γραφικού περιβάλλοντος με χρήση Python και Pygame, τη δημιουργία ενός backend συστήματος βασισμένου σε Node.js για τη διαχείριση της λογικής του παιχνιδιού και των δεδομένων των παικτών, καθώς και την ανάπτυξη ενός διαδραστικού frontend με React. Επιπλέον, στοχεύει στην ενσωμάτωση λειτουργικότητας πολλαπλών παικτών μέσω WebSockets και στη διασφάλιση της αξιοπιστίας του συστήματος υπό ποικίλες συνθήκες δικτύου. Οι προκλήσεις που αναμένεται να αντιμετωπιστούν κατά την υλοποίηση περιλαμβάνουν τον συγχρονισμό της κατάστασης του παιχνιδιού σε πραγματικό χρόνο, τη

διαχείριση δικτυακών ζητημάτων, τη διασφάλιση της ακεραιότητας του παιχνιδιού, τη βελτιστοποίηση της απόδοσης και τη δημιουργία μιας εργονομικής διεπαφής χρήστη. Μέσω της αντιμετώπισης αυτών των προκλήσεων, η εργασία αποσκοπεί στην απόκτηση εμπεριστατωμένης γνώσης στην ανάπτυξη σύνθετων διαδικτυακών εφαρμογών και παιχνιδιών, καθώς και στην εμβάθυνση σε προηγμένες τεχνολογίες και μεθοδολογίες ανάπτυξης λογισμικού.

1.4 Παρόμοια έργα και σχετική έρευνα

Αυτή η ενότητα εξερευνά διάφορες ερευνητικές εργασίες και υλοποιήσεις που σχετίζονται με διαφορετικές πτυχές του διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών. Εξετάζοντας αυτά τα έργα, μπορούμε να τοποθετήσουμε το έργο μας στο ευρύτερο πλαίσιο της ανάπτυξης ψηφιακών επιτραπέζιων παιχνιδιών, της αρχιτεκτονικής παιχνιδιών πολλαπλών παικτών και των διαδικτυακών υλοποιήσεων παιχνιδιών.

1.4.1 Ανάπτυξη ψηφιακού παιχνιδιού Ludo

Ο Nystrom στο βιβλίο του "Game Programming Patterns" [1] παρέχει πολύτιμες γνώσεις για την ανάπτυξη παιχνιδιών, συμπεριλαμβανομένων των επιτραπέζιων παιχνιδιών όπως το Ludo. Παρόλο που το βιβλίο δεν επικεντρώνεται συγκεκριμένα στο Ludo, οι αρχές σχεδιασμού και τα πρότυπα προγραμματισμού που παρουσιάζει είναι εξαιρετικά σχετικά με την υλοποίηση της λογικής του παιχνιδιού μας. Ιδιαίτερα, τα πρότυπα όπως το "State" για τη διαχείριση των διαφορετικών φάσεων του παιχνιδιού, το "Command" για την εκτέλεση των κινήσεων των παικτών, και το "Observer" για την ενημέρωση της κατάστασης του παιχνιδιού σε όλους τους παίκτες, έχουν επηρεάσει σημαντικά την αρχιτεκτονική του συστήματός μας. Η υλοποίησή μας επεκτείνει αυτές τις αρχές, προσαρμόζοντάς τες στις ιδιαίτερες απαιτήσεις ενός διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών. Ενώ το βιβλίο του Nystrom παρέχει μια στέρεα βάση για την ανάπτυξη παιχνιδιών γενικά, έχουμε εφαρμόσει αυτές τις γνώσεις ειδικά στο πλαίσιο του Ludo, ενσωματώνοντας λειτουργικότητα πολλαπλών παικτών σε πραγματικό χρόνο και μια διαδικτυακή διεπαφή.

1.4.2 Αρχιτεκτονική παιχνιδιού πολλαπλών παικτών με WebSockets

Μια κρίσιμη πτυχή του έργου μας είναι η λειτουργικότητα πολλαπλών παικτών σε πραγματικό χρόνο, την οποία έχουμε υλοποιήσει χρησιμοποιώντας την τεχνολογία WebSocket. Οι Wang, Salim και Moskovits στο βιβλίο τους "The Definitive Guide to HTML5 WebSocket" [2] παρέχουν μια εκτενή ανάλυση της τεχνολογίας WebSocket και των εφαρμογών της σε διαδραστικές εφαρμογές πραγματικού χρόνου. Η έρευνά τους παρέχει ένα πλαίσιο για την κατανόηση των πλεονεκτημάτων και των προκλήσεων της χρήσης WebSockets στην ανάπτυξη παιχνιδιών. Η υλοποίησή μας βασίζεται σε αυτές τις έννοιες, προσαρμόζοντάς τες ειδικά στη φύση του Ludo που βασίζεται σε γύρους και ενσωματώνοντάς τες με τις επιλεγμένες τεχνολογίες μας.

1.4.3 Ανάπτυξη παιχνιδιού με Python και Pygame

Η βασική λογική του παιχνιδιού και η απόδοση στην πλευρά του παίκτη του παιχνιδιού υλοποιούνται χρησιμοποιώντας Python και Pygame. Οι Van Rossum και Drake στο "Python 3 Reference Manual" [3] παρέχουν μια ολοκληρωμένη επισκόπηση της γλώσσας Python, η οποία αποτελεί τη βάση της υλοποίησής μας. Επιπλέον, η τεκμηρίωση του Pygame από τον Shinners [4] προσφέρει πολύτιμες πληροφορίες για την ανάπτυξη παιχνιδιών με αυτή τη βιβλιοθήκη. Το έργο μας επεκτείνει αυτές τις προσεγγίσεις συνδυάζοντας τη λογική του παιχνιδιού βασισμένη σε Python με ένα διαδικτυακό front-end, δημιουργώντας μια υβριδική αρχιτεκτονική που αξιοποιεί τα πλεονεκτήματα και των δύο τεχνολογιών.

1.4.4 Ψηφιοποίηση παραδοσιακών επιτραπέζιων παιχνιδιών

Η διαδικασία μετατροπής ενός φυσικού επιτραπέζιου παιχνιδιού σε ψηφιακή μορφή παρουσιάζει μοναδικές προκλήσεις και ευκαιρίες. Οι Salen και Zimmerman στο βιβλίο τους "Rules of Play: Game Design Fundamentals" [5] προσφέρουν πολύτιμες γνώσεις για αυτή τη διαδικασία. Η συζήτησή τους σχετικά με τη διατήρηση της ουσίας του παιχνιδιού ενώ αξιοποιούν τις ψηφιακές δυνατότητες έχει επηρεάσει στο σχεδιασμό του παιχνιδιού. Ιδιαίτερα στον τρόπο που έχουμε υλοποιήσει το ρίξιμο ζαριών και την κίνηση των πιονιών με τρόπο που να φαίνεται φυσικός αλλά ταυτόχρονα να εκμεταλλεύεται το ψηφιακό μέσο.

1.4.5 Διαδικτυακές τεχνολογίες στην ανάπτυξη παιχνιδιών

Το έργο μας χρησιμοποιεί React για τη διεπαφή χρήστη στο front-end και Node.js για τη λογική στην πλευρά του διακομιστή, δημιουργώντας μια σύγχρονη αρχιτεκτονική διαδικτυακής εφαρμογής για το παιχνίδι μας. Οι Tilkov και Vinoski στην εργασία τους "Node.js: Using JavaScript to Build High-Performance Network Programs" [6] παρέχουν μια βάση για την κατανόηση των πλεονεκτημάτων αυτής της προσέγγισης στη δημιουργία αποκρίσιμων διαδικτυακών εφαρμογών. Έχουμε εφαρμόσει αυτές τις αρχές συγκεκριμένα στην ανάπτυξη παιχνιδιών, επεκτείνοντας το έργο τους για να χειριστούμε τη διαχείριση της κατάστασης του παιχνιδιού και τις ενημερώσεις σε πραγματικό χρόνο σε ένα περιβάλλον πολλαπλών παικτών.

1.4.6 Σύνθεση προσεγγίσεων

Ενώ καθένα από αυτά τα έργα συνεισφέρει πολύτιμες γνώσεις σε συγκεκριμένες πτυχές του έργου μας, η μοναδική συνεισφορά της εργασίας μας βρίσκεται στη σύνθεση αυτών των προσεγγίσεων. Συνδυάζοντας ένα ψηφιοποιημένο κλασικό επιτραπέζιο παιχνίδι με λειτουργικότητα πολλαπλών παικτών σε πραγματικό χρόνο, και υλοποιώντας το χρησιμοποιώντας ένα υβρίδιο λογικής παιχνιδιού σε Python και σύγχρονων διαδικτυακών τεχνολογιών, το έργο μας συνδέει διάφορους τομείς με καινοτόμο τρόπο.

Ο Armitage στο βιβλίο του "Networking and Online Games: Understanding and Engineering Multiplayer Internet Games" [7] παρουσιάζει μια ολοκληρωμένη προσέγγιση στην ανάπτυξη διαδικτυακών παιχνιδιών πολλαπλών παικτών. Η εργασία μας βασίζεται σε αυτές τις αρχές, εφαρμόζοντάς τες συγκεκριμένα στο παιχνίδι Ludo και επεκτείνοντάς τες με την προσθήκη λογικής παιχνιδιού βασισμένης σε Python.

Στα επόμενα κεφάλαια, θα εμβαθύνουμε περισσότερο στο πώς αυτές οι διάφορες προσεγγίσεις και τεχνολογίες έχουν ενσωματωθεί και προσαρμοστεί στην ανάπτυξη του διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών, τονίζοντας τόσο τις προκλήσεις που αντιμετωπίστηκαν, όσο και τις λύσεις που επινοήθηκαν καθ' όλη τη διάρκεια της διαδικασίας υλοποίησης.

1.5 Οργάνωση κεφαλαίων

Η παρούσα διπλωματική εργασία διαρθρώνεται σε έξι κύρια κεφάλαια, καθένα εκ των οποίων πραγματεύεται διακριτές πτυχές της ανάπτυξης του διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών. Το πρώτο κεφάλαιο, η Εισαγωγή, παρουσιάζει τον σκοπό της εργασίας, περιγράφει το παιχνίδι Ludo, αναλύει τους στόχους και τις προκλήσεις του έργου, και εξετάζει συναφή έργα και σχετική βιβλιογραφία. Το δεύτερο κεφάλαιο, το Θεωρητικό Υπόβαθρο, παρέχει μια επισκόπηση των παιχνιδιών πολλαπλών παικτών, των τεχνολογιών ανάπτυξης παιχνιδιών και του δικτυακού προγραμματισμού, θέτοντας το θεωρητικό πλαίσιο για την υλοποίηση. Το τρίτο κεφάλαιο, ο Σχεδιασμός Συστήματος, περιγράφει λεπτομερώς την αρχιτεκτονική του συστήματος, παρουσιάζει διαγράμματα του παιχνιδιού και αναλύει τον σχεδιασμό της βάσης δεδομένων. Το τέταρτο κεφάλαιο, η Υλοποίηση, αναλύει την τεχνική υλοποίηση του έργου, συμπεριλαμβανομένου του backend (Python, Node.js), του frontend (React), και της ανάπτυξης του παιχνιδιού με τη χρήση του Pygame. Το πέμπτο κεφάλαιο, Δοκιμές και Αξιολόγηση, παρουσιάζει τη μεθοδολογία δοκιμών, τα αποτελέσματα και την αξιολόγηση της απόδοσης του συστήματος. Το έκτο κεφάλαιο, Συμπεράσματα και Μελλοντικές Επεκτάσεις, συνοψίζει την εργασία, αναλύει τις προκλήσεις που αντιμετωπίστηκαν και τις λύσεις που εφαρμόστηκαν, και προτείνει κατευθύνσεις για μελλοντική ανάπτυξη. Η εργασία ολοκληρώνεται με τη Βιβλιογραφία, η οποία παραθέτει όλες τις πηγές που αξιοποιήθηκαν κατά την εκπόνηση της έρευνας και της ανάπτυξης. Τέλος, τα Παραρτήματα περιλαμβάνουν συμπληρωματικό υλικό, όπως κώδικα ή άλλες πληροφορίες που υποστηρίζουν το κύριο σώμα της εργασίας.

Κεφάλαιο 2 - Θεωρητικό υπόβαθρο

2.1 Παιχνίδια πολλαπλών παικτών

Τα παιχνίδια πολλαπλών παικτών αποτελούν ένα σημαντικό κομμάτι της βιομηχανίας των βιντεοπαιχνιδιών, προσφέροντας μοναδικές εμπειρίες αλληλεπίδρασης και ανταγωνισμού μεταξύ παικτών. Ως παιχνίδια πολλαπλών παικτών ορίζονται εκείνα στα οποία περισσότεροι από ένας παίκτες μπορούν να συμμετέχουν ταυτόχρονα στην ίδια παρτίδα. Αυτά μπορεί να είναι τοπικά (στην ίδια συσκευή) ή διαδικτυακά, σύγχρονα ή ασύγχρονα, προσφέροντας ποικίλες δυνατότητες αλληλεπίδρασης.

Η ανάπτυξη παιχνιδιών πολλαπλών παικτών παρουσιάζει μοναδικές προκλήσεις. Ο συγχρονισμός των καταστάσεων του παιχνιδιού μεταξύ διαφορετικών συσκευών αποτελεί μια βασική δυσκολία, καθώς απαιτείται η διατήρηση μιας συνεκτικής εμπειρίας για όλους τους παίκτες. Η διαχείριση των καθυστερήσεων δικτύου (latency) είναι επίσης κρίσιμη, ειδικά σε παιχνίδια που απαιτούν γρήγορα αντανακλαστικά. Επιπλέον, η εξασφάλιση δίκαιου παιχνιδιού και η πρόληψη της εξαπάτησης αποτελούν σημαντικές προκλήσεις. Τέλος, η κλιμάκωση του συστήματος για την υποστήριξη μεγάλου αριθμού ταυτόχρονων παικτών απαιτεί προσεκτικό σχεδιασμό και βελτιστοποίηση.

Οι κύριες αρχιτεκτονικές για παιχνίδια πολλαπλών παικτών περιλαμβάνουν το μοντέλο Client-Server, όπου ένας κεντρικός διακομιστής διαχειρίζεται το παιχνίδι, το μοντέλο Peer-to-Peer, όπου οι παίκτες συνδέονται απευθείας μεταξύ τους, και τα υβριδικά μοντέλα που συνδυάζουν στοιχεία και από τις δύο προσεγγίσεις. Η επιλογή της κατάλληλης αρχιτεκτονικής εξαρτάται από τις συγκεκριμένες απαιτήσεις του παιχνιδιού και τους διαθέσιμους πόρους.

2.2 Ludo

Το Ludo είναι ένα κλασικό επιτραπέζιο παιχνίδι στρατηγικής που έχει τις ρίζες του στο αρχαίο ινδικό παιχνίδι Pachisi. Σχεδιασμένο για 2 έως 4 παίκτες, το Ludo παίζεται σε ένα σταυροειδές ταμπλό με τέσσερις διαφορετικές χρωματικές περιοχές, μία για κάθε παίκτη.

Κάθε παίκτης ξεκινά με τέσσερα πιόνια στην "αφετηρία" του, που βρίσκεται σε μία από τις γωνίες του ταμπλό. Ο στόχος του παιχνιδιού είναι να μετακινήσει όλα τα πιόνια από

την αφετηρία, γύρω από το ταμπλό και τελικά στην κεντρική "βάση" του αντίστοιχου χρώματος.

Οι παίκτες ρίχνουν ένα ζάρι εναλλάξ για να καθορίσουν πόσες θέσεις θα κινηθεί ένα από τα πιόνια τους. Για να βγει ένα πιόνι από την αφετηρία, ο παίκτης πρέπει να φέρει 6 στο ζάρι. Κατά τη διάρκεια του παιχνιδιού, οι παίκτες μπορούν να "φάνε" τα πιόνια των αντιπάλων, στέλνοντάς τα πίσω στην αφετηρία τους.

Η ψηφιοποίηση του Ludo παρουσιάζει μοναδικές προκλήσεις και ευκαιρίες. Απαιτεί προσεκτικό σχεδιασμό του ταμπλό, υλοποίηση των κανόνων του παιχνιδιού, και δημιουργία ενός διαισθητικού περιβάλλοντος διεπαφής χρήστη. Επιπλέον, η προσθήκη λειτουργίας πολλαπλών παικτών μέσω διαδικτύου αυξάνει την πολυπλοκότητα, απαιτώντας αποτελεσματική διαχείριση της κατάστασης του παιχνιδιού και συγχρονισμό μεταξύ των παικτών.

2.3 Τεχνολογίες ανάπτυξης παιχνιδιών

Για την ανάπτυξη του διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών, χρησιμοποιήθηκε ένας συνδυασμός σύγχρονων τεχνολογιών προγραμματισμού.

2.3.1 Python

Η Python (Python Documentation) είναι μια υψηλού επιπέδου, διερμηνευόμενη γλώσσα προγραμματισμού που δημιουργήθηκε από τον Guido van Rossum το 1991. Είναι γνωστή για την απλότητα και την αναγνωσιμότητα του κώδικά της, καθιστώντας την ιδανική για γρήγορη ανάπτυξη και πρωτοτυποποίηση. Η Python [Εικόνα 2: Δείγμα κώδικα Python] υποστηρίζει πολλαπλά προγραμματιστικά παραδείγματα, συμπεριλαμβανομένου του διαδικαστικού, αντικειμενοστραφούς και συναρτησιακού προγραμματισμού. Στη παρούσα διπλωματική, η Python χρησιμοποιήθηκε για την υλοποίηση της βασικής λογικής του παιχνιδιού, επωφελούμενη από την ευελιξία και την εκφραστικότητά της.

```python
def game_loop(self):
    running = True

    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            elif self.game_ended:
                self.end_game_modal.handle_event(event)
            else:
                self.event_handler.handle_event(event)

        if self.game_started and not self.game_ended:
            current_player = self.game_logic.get_current_player()

            if current_player and current_player.player_id.startswith('bot'):
                self.handle_bot_turn(current_player)

        if not self.renderer.is_animating:
            self.render()
            self.renderer.update(self.players)

        self.chat.update(self.clock.tick(20))
        self.clock.tick(30)

    self.network_manager.disconnect()
    pygame.quit()
```

Εικόνα 2: Δείγμα κώδικα Python

2.3.2 Pygame

Το Pygame (Pygame documentation) είναι μια βιβλιοθήκη της Python ειδικά σχεδιασμένη για την ανάπτυξη παιχνιδιών και εφαρμογών πολυμέσων. Βασίζεται στη βιβλιοθήκη SDL (Simple DirectMedia Layer) και παρέχει μια σειρά από εργαλεία και λειτουργίες για τη διαχείριση γραφικών, ήχου και εισόδου χρήστη. Το Pygame απλοποιεί πολλές πολύπλοκες εργασίες στην ανάπτυξη παιχνιδιών, όπως ο χειρισμός συμβάντων, η αναπαραγωγή ήχου και η διαχείριση sprite. Στη παρούσα διπλωματική, το Pygame χρησιμοποιήθηκε για την υλοποίηση των γραφικών και του περιβάλλοντος του παιχνιδιού, επιτρέποντάς μας να δημιουργήσουμε ένα οπτικά ελκυστικό και διαδραστικό ταμπλό Ludo.

2.3.3 JavaScript

Η JavaScript (JavaScript mozilla.org) είναι μια υψηλού επιπέδου, διερμηνευόμενη γλώσσα προγραμματισμού [Εικόνα 3: Δείγμα κώδικα JavaScript] που συμμορφώνεται με την προδιαγραφή ECMAScript. Αρχικά σχεδιάστηκε για χρήση στους περιηγητές ιστού, αλλά πλέον χρησιμοποιείται ευρέως και σε περιβάλλοντα εκτός περιηγητή. Η JavaScript υποστηρίζει τον αντικειμενοστραφή, τον προστακτικό και τον συναρτησιακό προγραμματισμό. Στη παρούσα διπλωματική, η JavaScript χρησιμοποιήθηκε τόσο στο frontend για τη δημιουργία δυναμικών στοιχείων της διεπαφής χρήστη, όσο και στο backend με τη χρήση του Node.js, επιτρέποντάς μας να έχουμε μια ενιαία γλώσσα προγραμματισμού

σε όλο το stack της εφαρμογής. Τέλος, όπως αναφέρει ο Flanagan [8], η JavaScript έχει εξελιχθεί σημαντικά από την αρχική της έκδοση, προσφέροντας πλέον ισχυρές δυνατότητες για ανάπτυξη πολύπλοκων εφαρμογών."

```
const handleClick = (e) => {
    if (isClickable) {
        handleJoinGame(game.players, game.status, game.id);
    }
};

const getStatusText = (status) => {
    switch(status) {
        case 0: return 'Waiting';
        case 1: return 'In Progress';
        case 2: return 'Finished';
        default: return 'Unknown';
    }
};

const getStatusClass = (status) => {
    switch(status) {
        case 0: return 'waiting';
        case 1: return 'inProgress';
        case 2: return 'finished';
        default: return '';
    }
};
```

Εικόνα 3: Δείγμα κώδικα JavaScript

2.3.4 Node.js

Το Node.js (Node.js Documentation) είναι ένα περιβάλλον εκτέλεσης JavaScript στην πλευρά του διακομιστή, βασισμένο στη μηχανή V8 της Google Chrome. Επιτρέπει την εκτέλεση κώδικα JavaScript εκτός του περιβάλλοντος του περιηγητή και παρέχει ένα πλούσιο οικοσύστημα βιβλιοθηκών μέσω του npm (Node Package Manager). Το Node.js χρησιμοποιεί ένα μοντέλο I/O χωρίς αποκλεισμούς και βασισμένο σε συμβάντα, καθιστώντας το ιδανικό για εφαρμογές πραγματικού χρόνου. Στη παρούσα διπλωματική, το Node.js χρησιμοποιήθηκε για την ανάπτυξη του backend server, επιτρέποντάς μας να χειριστούμε αποτελεσματικά πολλαπλές συνδέσεις παικτών ταυτόχρονα. Η υλοποίηση ακολούθησε τις βέλτιστες

πρακτικές που περιγράφονται από τους Cantelon et al. [9], εξασφαλίζοντας αποδοτική και κλιμακώσιμη λειτουργία του server.

2.3.5 React

Η React (React) είναι μια βιβλιοθήκη JavaScript ανοιχτού κώδικα για τη δημιουργία διεπαφών χρήστη, που αναπτύχθηκε και συντηρείται από το Facebook. Η React χρησιμοποιεί μια δηλωτική προσέγγιση για τη δημιουργία επαναχρησιμοποιήσιμων components UI, επιτρέποντας στους προγραμματιστές να δημιουργούν πολύπλοκες διεπαφές από απλά κομμάτια κώδικα. Η React χρησιμοποιεί ένα εικονικό DOM για βελτιστοποίηση της απόδοσης, ενημερώνοντας αποτελεσματικά μόνο τα τμήματα της σελίδας που έχουν αλλάξει. Στη παρούσα διπλωματική, η React χρησιμοποιήθηκε για την ανάπτυξη του frontend της εφαρμογής, συμπεριλαμβανομένων των μενού του παιχνιδιού και της διεπαφής χρήστη, επιτρέποντάς μας να δημιουργήσουμε μια δυναμική και αποκρίσιμη εμπειρία χρήστη. Η προσέγγιση αυτή βασίστηκε στις βέλτιστες πρακτικές που περιγράφονται από τους Banks και Porcello [10], επιτρέποντας την αποτελεσματική διαχείριση της κατάστασης της εφαρμογής.

2.3.6 SQL

Η SQL (Structured Query Language) (SQL Documentation) είναι μια τυποποιημένη γλώσσα για τη διαχείριση σχεσιακών βάσεων δεδομένων. Επιτρέπει την αποθήκευση, τον χειρισμό και την ανάκτηση δεδομένων σε σχεσιακές βάσεις δεδομένων. Η SQL υποστηρίζει διάφορους τύπους λειτουργιών, συμπεριλαμβανομένων ερωτημάτων, εισαγωγών, ενημερώσεων και διαγραφών δεδομένων, καθώς και τη δημιουργία και τροποποίηση σχημάτων βάσεων δεδομένων. Στη παρούσα διπλωματική, η SQL χρησιμοποιήθηκε για τη διαχείριση της βάσης δεδομένων, αποθηκεύοντας πληροφορίες σχετικά με τους παίκτες, τα παιχνίδια και τα στατιστικά, επιτρέποντάς μας να διατηρούμε και να ανακτούμε αποτελεσματικά τα δεδομένα του παιχνιδιού. Η υλοποίηση ακολούθησε τις βέλτιστες πρακτικές που περιγράφονται από τον Beaulieu [11], εξασφαλίζοντας αποδοτική και ασφαλή διαχείριση των δεδομένων.

2.3.7 WebSockets

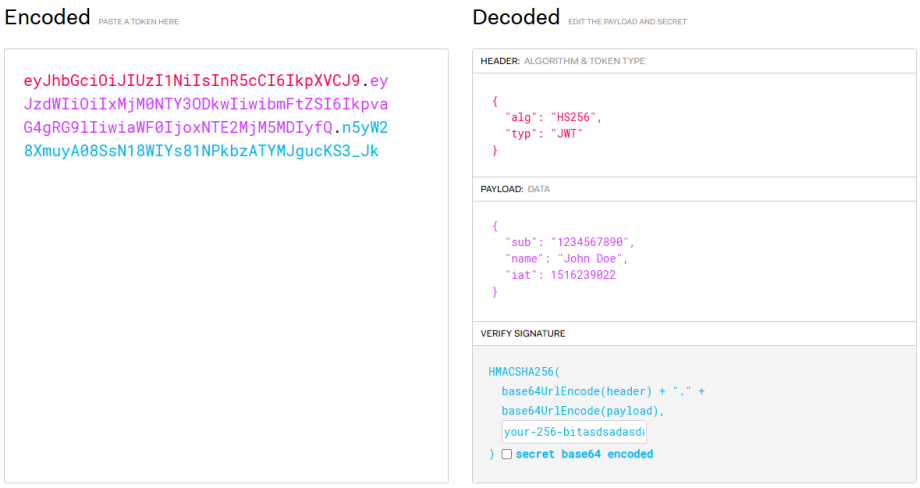
Τα WebSockets (Websockets documentation) [Εικόνα 4: Χρήση Websockets για επικοινωνία σε πραγματικό χρόνοσ] είναι ένα πρωτόκολλο επικοινωνίας που παρέχει αμφίδρομη επικοινωνία μεταξύ ενός προγράμματος-πελάτη (συνήθως ενός περιηγητή ιστού) και ενός διακομιστή. Όπως περιγράφεται από τους Fette και Melnikov [12], το πρωτόκολλο WebSocket επιτρέπει την αποστολή μηνυμάτων από το διακομιστή στον πελάτη χωρίς να χρειάζεται ο πελάτης να κάνει αίτημα, επιτρέποντας έτσι την επικοινωνία σε πραγματικό χρόνο. Στη παρούσα διπλωματική, τα WebSockets χρησιμοποιήθηκαν για την επίτευξη αμφίδρομης επικοινωνίας μεταξύ του διακομιστή και των πελατών, επιτρέποντας την άμεση ενημέρωση της κατάστασης του παιχνιδιού για όλους τους παίκτες.

```
const sendChatMessage = useCallback(({ gameId, userId, message }) => {
    if (socket) {
        console.log(`Emitting chatMessage event for game ${gameId} and user ${userId}`);
        socket.emit("chatMessage", { gameId, userId, message });
    } else {
        console.error('Socket is not initialized');
        setConnectionError("Socket connection not established");
    }
}, [socket]);
```

Εικόνα 4: Χρήση Websockets για επικοινωνία σε πραγματικό χρόνο

2.3.8 JWT

Το JWT (JSON Web Tokens) (JSON Web Token Introduction - jwt.io) είναι ένα ανοιχτό πρότυπο (RFC 7519) που καθορίζει έναν συμπαγή και αυτόνομο τρόπο για την ασφαλή μετάδοση πληροφοριών μεταξύ μερών ως ένα αντικείμενο JSON. Αυτές οι πληροφορίες μπορούν να επαληθευτούν και να θεωρηθούν αξιόπιστες επειδή είναι ψηφιακά υπογεγραμμένες [Εικόνα 5: Δείγμα υπογραφής δεδομένων με JWT]. Τα JWTs μπορούν να υπογραφούν χρησιμοποιώντας ένα μυστικό (με τον αλγόριθμο HMAC) ή ένα ζεύγος δημόσιου/ιδιωτικού κλειδιού χρησιμοποιώντας RSA ή ECDSA. Στη παρούσα διπλωματική, τα JWTs χρησιμοποιήθηκαν για την ασφαλή πιστοποίηση και εξουσιοδότηση των χρηστών, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι παίκτες μπορούν να έχουν πρόσβαση στο παιχνίδι και τις λειτουργίες του.



Εικόνα 5: Δείγμα υπογραφής δεδομένων με JWT

2.4 Δικτυακός προγραμματισμός

Οι κύριες τεχνικές δικτυακού προγραμματισμού που εφαρμόστηκαν στην ανάπτυξη του παιχνιδιού Ludo πολλαπλών παικτών είναι οι εξής, βασισμένες στις αρχές που περιγράφονται από τον Comer [13] και εξειδικευμένες για παιχνίδια από τον Fiedler [14]:

2.4.1 WebSockets

Τα WebSockets αποτελούν τον πυρήνα της επικοινωνίας σε πραγματικό χρόνο στο παιχνίδι μας. Πρόκειται για ένα πρωτόκολλο που επιτρέπει αμφίδρομη επικοινωνία μεταξύ του προγράμματος-πελάτη (browser) και του διακομιστή πάνω από μία μόνο σύνδεση TCP.

Πλεονεκτήματα:

- Επιτρέπει την άμεση ενημέρωση της κατάστασης του παιχνιδιού για όλους τους παίκτες.
- Μειώνει την καθυστέρηση σε σύγκριση με τεχνικές όπως το polling.
- Επιτρέπει την αποστολή μηνυμάτων από τον διακομιστή στον πελάτη χωρίς να χρειάζεται ο πελάτης να κάνει αίτημα.

Χρησιμοποιήθηκαν τα WebSockets για να μεταδίδουμε άμεσα τις κινήσεις των παικτών, τα αποτελέσματα των ζαριών, και τις αλλαγές στην κατάσταση του παιχνιδιού σε όλους τους συνδεδεμένους παίκτες.

2.4.2 Client-Server μοντέλο

Το παιχνίδι μας βασίζεται στο client-server μοντέλο αρχιτεκτονικής, όπου ο διακομιστής (server) διαχειρίζεται την κύρια λογική του παιχνιδιού και οι πελάτες (clients) είναι υπεύθυνοι για την απεικόνιση και την αλληλεπίδραση με τον χρήστη.

Πλεονεκτήματα:

- Κεντρικός έλεγχος της κατάστασης του παιχνιδιού.
- Ευκολότερη διαχείριση των κανόνων του παιχνιδιού και της ασφάλειας.
- Δυνατότητα κλιμάκωσης για την υποστήριξη πολλών ταυτόχρονων παιχνιδιών.

Χρησιμοποιήσαμε Node.js για το backend (server) και React για το frontend (client), επιτρέποντας τον διαχωρισμό των ευθυνών και την αποτελεσματική διαχείριση του παιχνιδιού.

2.4.3 HTTP/HTTPS

Τα πρωτόκολλα HTTP και HTTPS χρησιμοποιούνται για την αρχική επικοινωνία και τη φόρτωση της εφαρμογής.

Πλεονεκτήματα:

- Ευρέως υποστηριζόμενα πρωτόκολλα.
- Το HTTPS προσφέρει ασφαλή μετάδοση δεδομένων.
- Επιτρέπουν τη χρήση RESTful APIs για επικοινωνία μεταξύ client και server.

Χρησιμοποιήθηκε HTTPS για την ασφαλή φόρτωση της εφαρμογής και για τις αρχικές αιτήσεις δεδομένων, όπως η ανάκτηση της λίστας των διαθέσιμων παιχνιδιών.

2.4.4 JSON

Το JSON (JavaScript Object Notation) χρησιμοποιείται ως μορφή δεδομένων για την ανταλλαγή πληροφοριών μεταξύ του client και του server.

Πλεονεκτήματα:

- Ελαφρύ και εύκολο στην ανάγνωση από ανθρώπους και μηχανές.
- Υποστηρίζεται από τη JavaScript, διευκολύνοντας τη χρήση του στο frontend.

- Ευέλικτο και μπορεί να αναπαραστήσει πολύπλοκες δομές δεδομένων.

Χρησιμοποιήσαμε JSON για τη μορφοποίηση των δεδομένων του παιχνιδιού, όπως η κατάσταση του ταμπλό, οι πληροφορίες των παικτών, και τα αποτελέσματα των κινήσεων.

2.4.5 RESTful API

Υλοποιήσαμε ένα RESTful API (Application Programming Interface, Διεπαφή Προγραμματισμού Εφαρμογών) για την επικοινωνία μεταξύ του frontend και του backend για ορισμένες λειτουργίες του παιχνιδιού.

Πλεονεκτήματα:

- Καθορισμένη δομή για αιτήματα και αποκρίσεις.
- Εύκολη κλιμάκωση και συντήρηση.

Χρησιμοποιήσαμε RESTful endpoints για λειτουργίες όπως η δημιουργία νέου παιχνιδιού, η συμμετοχή σε υπάρχον παιχνίδι, και η ανάκτηση στατιστικών παικτών.

2.5 Εργαλεία ανάπτυξης

Για την ανάπτυξη του παιχνιδιού Ludo χρησιμοποιήθηκαν διάφορα εργαλεία, το καθένα με τον δικό του ρόλο στη διαδικασία ανάπτυξης. Παρακάτω αναλύονται τα κύρια εργαλεία που χρησιμοποιήθηκαν

2.5.1 Visual Studio Code (VSCode)

Το Visual Studio Code είναι ένας δωρεάν, ανοιχτού κώδικα επεξεργαστής κειμένου/κώδικα που αναπτύχθηκε από τη Microsoft. Προσφέρει ένα ευέλικτο και προσαρμόσιμο περιβάλλον για την ανάπτυξη σε διάφορες γλώσσες προγραμματισμού. Στο έργο μας, το VSCode χρησιμοποιήθηκε κυρίως για την ανάπτυξη του frontend με React και JavaScript, καθώς και του backend με Node.js.

2.5.2 PyCharm

Το PyCharm είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) ειδικά σχεδιασμένο για προγραμματισμό σε Python. Αναπτύχθηκε από την JetBrains και προσφέρει προηγμένες λειτουργίες όπως έξυπνη συμπλήρωση κώδικα, αποσφαλμάτωση και αναδόμηση κώδικα. Στο έργο μας, το PyCharm χρησιμοποιήθηκε για την ανάπτυξη του backend σε Python και την υλοποίηση της λογικής του παιχνιδιού με το Pygame.

2.5.3 phpMyAdmin

Το phpMyAdmin (https://www.phpmyadmin.net/) είναι ένα δωρεάν εργαλείο λογισμικού γραμμένο σε PHP, που προορίζεται για τη διαχείριση βάσεων δεδομένων MySQL μέσω του Web. Προσφέρει ένα διαισθητικό γραφικό περιβάλλον για τη δημιουργία, τροποποίηση και διαγραφή βάσεων δεδομένων, πινάκων, πεδίων και σειρών. Στο έργο μας, το phpMyAdmin χρησιμοποιήθηκε για τη διαχείριση της βάσης δεδομένων του παιχνιδιού, επιτρέποντας την εύκολη προβολή και επεξεργασία των δεδομένων των παικτών και των παιχνιδιών.

2.5.4 draw.io

Το draw.io (https://app.diagrams.net/) (πλέον γνωστό ως diagrams.net) είναι ένα δωρεάν online εργαλείο σχεδίασης διαγραμμάτων. Επιτρέπει τη δημιουργία διαφόρων τύπων διαγραμμάτων, συμπεριλαμβανομένων διαγραμμάτων ροής, UML, και ER διαγραμμάτων. Στο έργο μας, το draw.io χρησιμοποιήθηκε για τη δημιουργία διαγραμμάτων που απεικονίζουν τη δομή του παιχνιδιού, τη ροή δεδομένων και την αρχιτεκτονική του συστήματος.

2.5.5 PuTTY

Το PuTTY είναι ένα δωρεάν, ανοιχτού κώδικα εργαλείο τερματικού, σειριακής κονσόλας και μεταφοράς αρχείων δικτύου. Υποστηρίζει διάφορα πρωτόκολλα δικτύου, συμπεριλαμβανομένου του SSH. Στο έργο μας, το PuTTY χρησιμοποιήθηκε για την απομακρυσμένη σύνδεση και διαχείριση του διακομιστή που φιλοξενεί το παιχνίδι και τη γραφική διεπαφή, επιτρέποντας την εκτέλεση εντολών και τη διαχείριση αρχείων στο περιβάλλον του διακομιστή.

2.6 Σύνοψη κεφαλαίου

Το παρόν κεφάλαιο παρουσίασε το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση και την υλοποίηση του διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών. Εξετάστηκαν οι βασικές έννοιες των παιχνιδιών πολλαπλών παικτών, οι κανόνες και η δομή του Ludo, καθώς και οι κύριες τεχνολογίες ανάπτυξης που χρησιμοποιήθηκαν στο έργο.

Συγκεκριμένα, αναλύθηκαν οι γλώσσες προγραμματισμού Python και JavaScript, τα πλαίσια ανάπτυξης Pygame και React, καθώς και οι τεχνολογίες διακομιστή Node.js και WebSockets. Επιπλέον, εξετάστηκαν οι αρχές του δικτυακού προγραμματισμού και τα εργαλεία ανάπτυξης που χρησιμοποιήθηκαν στο έργο.

Η κατανόηση αυτών των εννοιών και τεχνολογιών είναι κρίσιμη για την επιτυχή υλοποίηση του παιχνιδιού, καθώς παρέχει το απαραίτητο θεωρητικό υπόβαθρο για τις τεχνικές αποφάσεις και τις μεθοδολογίες που θα ακολουθήσουν στα επόμενα κεφάλαια. Η συνδυασμένη χρήση αυτών των τεχνολογιών επιτρέπει τη δημιουργία ενός ολοκληρωμένου, διαδραστικού και αποδοτικού συστήματος παιχνιδιού που μπορεί να ανταποκριθεί στις απαιτήσεις των σύγχρονων διαδικτυακών παιχνιδιών πολλαπλών παικτών.

Κεφάλαιο 3 - Σχεδιασμός συστήματος

3.1 Αρχιτεκτονική συστήματος

Η υλοποίηση του επιτραπέζιου παιχνιδιού Ludo βασίζεται σε μια σύνθετη κατανεμημένη αρχιτεκτονική συστήματος, η οποία απαρτίζεται από τρία θεμελιώδη συστατικά: τη διαδικτυακή διεπαφή χρήστη, την εφαρμογή του παιχνιδιού και τον εξυπηρετητή. Η δομή αυτή ενσωματώνει ένα υβριδικό μοντέλο, συνδυάζοντας χαρακτηριστικά από αρχιτεκτονικές πελάτη-εξυπηρετητή (client-server) και ομότιμων κόμβων (peer-to-peer), με σκοπό τη βελτιστοποίηση τόσο της εμπειρίας παιχνιδιού σε πραγματικό χρόνο, όσο και της κεντρικής διαχείρισης δεδομένων.

Η διαδικτυακή διεπαφή χρήστη λειτουργεί ως το κύριο σημείο αλληλεπίδρασης για τη διαχείριση λογαριασμών, τη δημιουργία και συμμετοχή σε αίθουσες παιχνιδιού, καθώς και τη διεξαγωγή συνομιλιών σε πραγματικό χρόνο. Η ανάπτυξή της υλοποιείται στο πλαίσιο React.js, αξιοποιώντας την αρχιτεκτονική των επαναχρησιμοποιήσιμων και ευσυντήρητων δομικών στοιχείων.

Η εφαρμογή του παιχνιδιού έχει αναπτυχθεί σε γλώσσα προγραμματισμού Python, χρησιμοποιώντας τη βιβλιοθήκη Pygame. Η δομή της ακολουθεί τις αρχές του αντικειμενοστραφούς προγραμματισμού, με την κλάση LudoGame να αποτελεί τον πυρήνα της εφαρμογής.

Ο εξυπηρετητής, υλοποιημένος σε Node.js με το πλαίσιο εφαρμογής Express.js, λειτουργεί ως η κεντρική οντότητα για τη διαχείριση της λογικής του παιχνιδιού, τη διατήρηση δεδομένων και τον συντονισμό της επικοινωνίας μεταξύ των πελατών. Παρέχει διεπαφές RESTful API για λειτουργίες όπως η ταυτοποίηση χρηστών, η διαχείριση παιχνιδιών και οι ενημερώσεις προφίλ.

Η διαχείριση δεδομένων πραγματοποιείται μέσω μιας βάσης δεδομένων MySQL, με το Sequelize να χρησιμοποιείται ως εργαλείο αντιστοίχισης αντικειμένων-σχέσεων (Object-relational mapping, ORM). Τα μοντέλα Sequelize παρέχουν ένα επίπεδο αφαίρεσης για τις λειτουργίες της βάσης δεδομένων, διευκολύνοντας την αποτελεσματική αναζήτηση και τον χειρισμό δεδομένων.

Η ταυτοποίηση και η ασφάλεια του συστήματος υλοποιούνται μέσω της χρήσης JSON Web Tokens (JWT). Το σύστημα εφαρμόζει τόσο διακριτικά πρόσβασης, όσο και ανανέωσης για τη διατήρηση των συνεδριών των χρηστών με ασφάλεια.

Η δικτυακή επικοινωνία στο σύστημα πραγματοποιείται μέσω δύο κύριων καναλιών. Οι κλήσεις RESTful API χρησιμοποιούν το πρωτόκολλο HTTP/HTTPS για λειτουργίες όπως η ταυτοποίηση χρήστη, η δημιουργία παιχνιδιού και η ανάκτηση καταλόγων παιχνιδιών. Για ενημερώσεις παιχνιδιού σε πραγματικό χρόνο και λειτουργίες συνομιλίας, το σύστημα αξιοποιεί συνδέσεις WebSocket μέσω του Socket.io, επιτρέποντας επικοινωνία χαμηλής καθυστέρησης και αμφίδρομη μετάδοση δεδομένων μεταξύ του εξυπηρετητή και των πελατών.

Μια αξιοσημείωτη πτυχή του συστήματος είναι η διασύνδεση της διαδικτυακής διεπαφής με την εφαρμογή. Αυτή επιτυγχάνεται μέσω ενός μηχανισμού διακριτικού εφαρμογής (app token). Κατά την εκκίνηση της εφαρμογής, δημιουργείται ένα μοναδικό διακριτικό και ανοίγει ένα πρόγραμμα περιήγησης με το διακριτικό ως παράμετρο στη διεύθυνση URL. Μετά τη σύνδεση του χρήστη, το διακριτικό συσχετίζεται με τον χρήστη και τα δεδομένα του επιστρέφονται στην εφαρμογή.

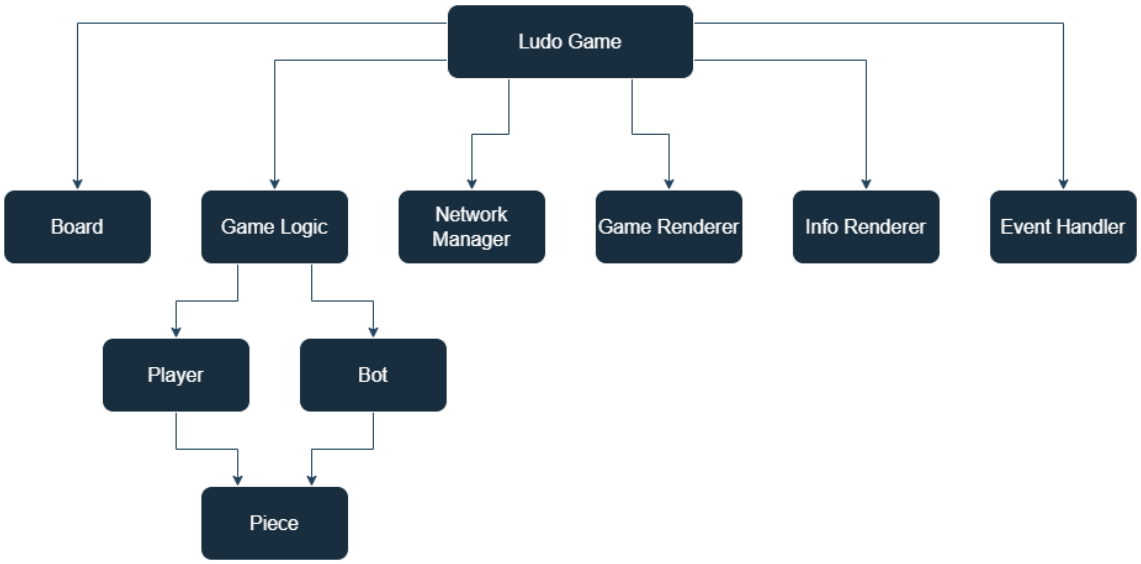
Η λογική του παιχνιδιού κατανέμεται μεταξύ του εξυπηρετητή και των πελατών. Ενώ ο εξυπηρετητής διατηρεί την έγκυρη κατάσταση του παιχνιδιού για την αποτροπή παραβιάσεων, τόσο η διαδικτυακή διεπαφή, όσο και η εφαρμογή υλοποιούν τμήματα της λογικής του παιχνιδιού. Αυτή η προσέγγιση επιτρέπει τη δημιουργία διεπαφών χρήστη με άμεση απόκριση μέσω πρόβλεψης στην πλευρά του πελάτη, ενώ παράλληλα διατηρείται η ακρίβεια μέσω τακτικού συγχρονισμού με τον εξυπηρετητή.

Η συγκεκριμένη αρχιτεκτονική δημιουργεί ένα εύρωστο και ευέλικτο σύστημα, ικανό να διαχειριστεί τις πολυπλοκότητες ενός παιχνιδιού Ludo για πολλαπλούς παίκτες. Εξισορροπεί αποτελεσματικά τις απαιτήσεις για αλληλεπίδραση σε πραγματικό χρόνο, διατήρηση δεδομένων, ασφάλεια και βελτιστοποίηση της εμπειρίας χρήστη τόσο σε διαδικτυακές, όσο και σε εξειδικευμένες διεπαφές πελάτη. Ο αρθρωτός σχεδιασμός διευκολύνει τη συντήρηση και επιτρέπει μελλοντικές επεκτάσεις, όπως η προσθήκη νέων λειτουργιών παιχνιδιού ή η ενσωμάτωση με πρόσθετες πλατφόρμες.

3.2 Διαγράμματα παιχνιδιού

Το διάγραμμα της Εικόνας 6 [Εικόνα 6: Διάγραμμα αντικειμένων και γεγονότων παιχνιδιού] απεικονίζει τη δομή και τις σχέσεις μεταξύ των κύριων συστατικών του παιχνιδιού Ludo. Στο κέντρο βρίσκεται η κεντρική οντότητα "Ludo Game", η οποία αποτελεί τον πυρήνα του παιχνιδιού και συνδέεται άμεσα με έξι βασικά στοιχεία: Board (Πίνακας), Game Logic (Λογική Παιχνιδιού), Network Manager (Διαχειριστής Δικτύου), Game Renderer (Απεικόνιση Παιχνιδιού), Info Renderer (Απεικόνιση Πληροφοριών) και Event Handler (Χειριστής Συμβάντων).

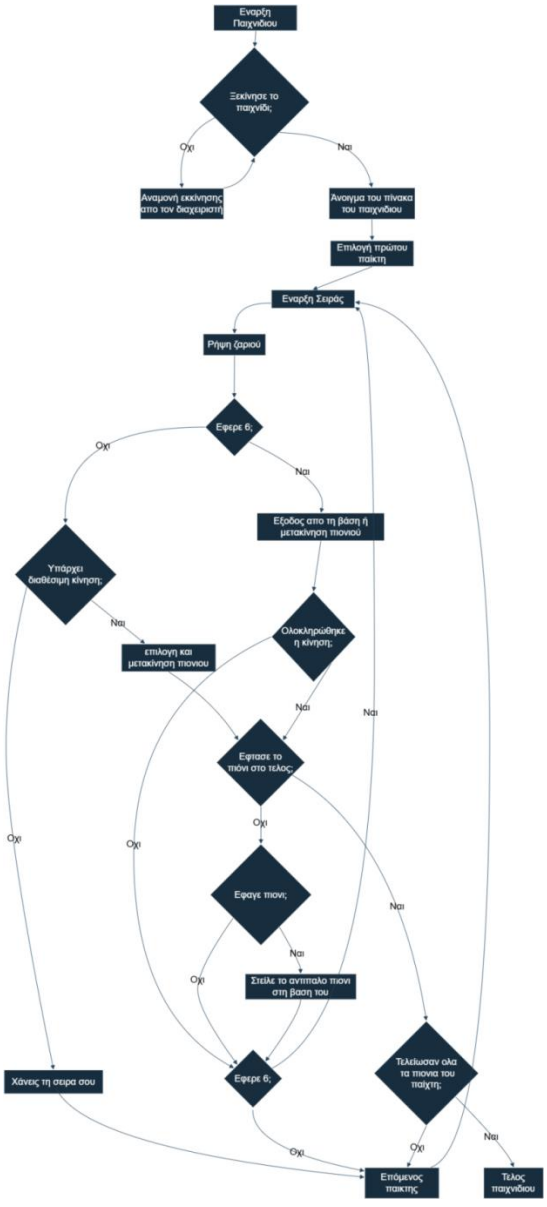
Η Game Logic, ως κεντρικό στοιχείο της λειτουργικότητας του παιχνιδιού, συνδέεται περαιτέρω με δύο υποκατηγορίες: Player (Παίκτης) και Bot (Ρομπότ). Και οι δύο αυτές οντότητες συνδέονται με την οντότητα Piece (Πιόνι), η οποία αντιπροσωπεύει τα κινούμενα στοιχεία στο ταμπλό του παιχνιδιού. Αυτή η ιεραρχική δομή επιτρέπει την αποτελεσματική διαχείριση των διαφόρων πτυχών του παιχνιδιού, από τη βασική λογική και τους κανόνες μέχρι την απεικόνιση και τη διαχείριση των δικτυακών λειτουργιών, διασφαλίζοντας έτσι μια ολοκληρωμένη και καλά οργανωμένη αρχιτεκτονική για το παιχνίδι Ludo.



Εικόνα 6: Διάγραμμα αντικειμένων και γεγονότων παιχνιδιού

Το διάγραμμα ροής [Εικόνα 7: Διάγραμμα ροής] απεικονίζει τη λογική και τη διαδικασία του παιχνιδιού Ludo από την έναρξη μέχρι το τέλος του. Ξεκινά με την "Έναρξη Παιχνιδιού" και ακολουθεί μια σειρά από αποφάσεις και ενέργειες που καθορίζουν την

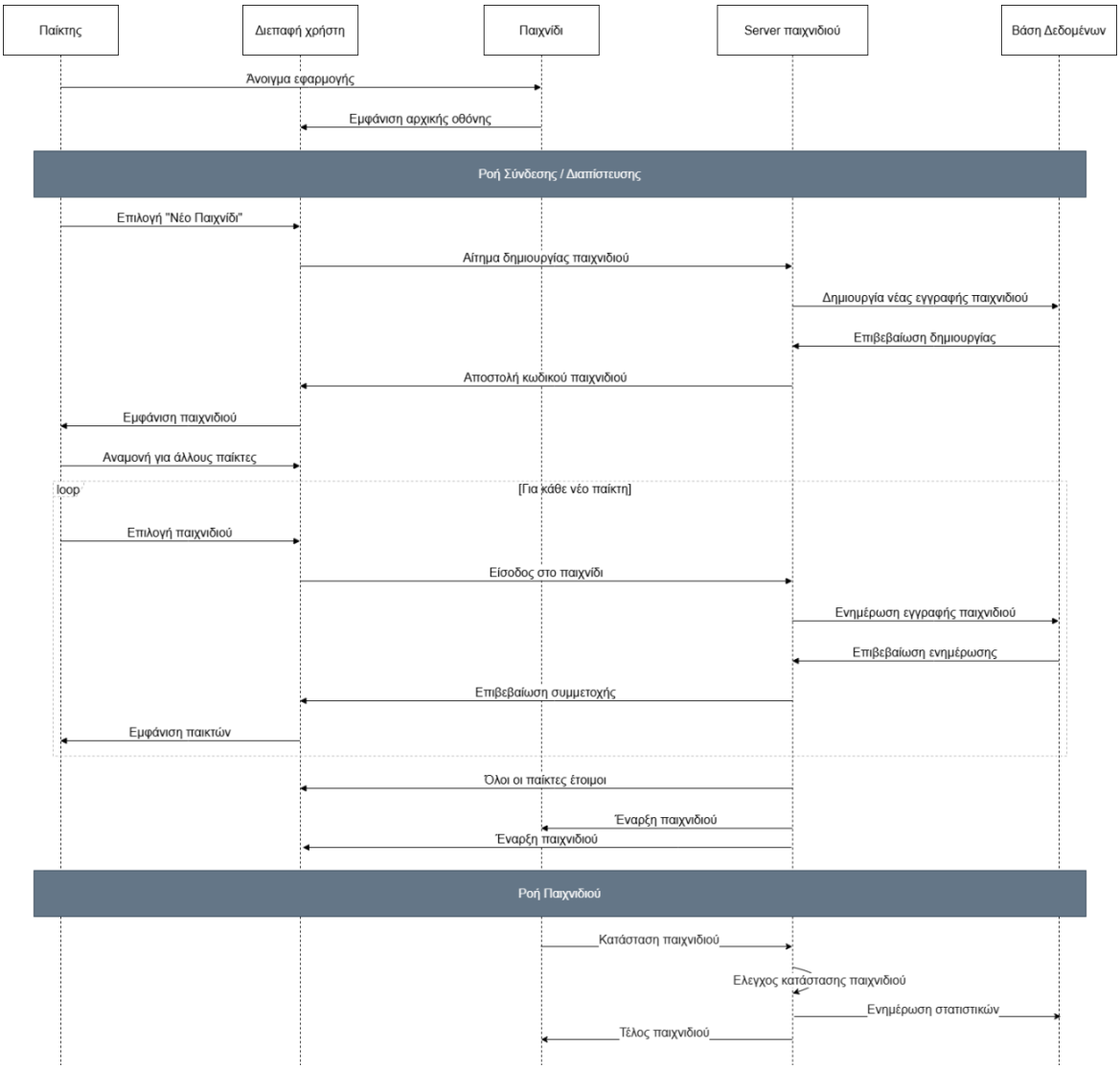
πορεία του παιχνιδιού. Η ροή περιλαμβάνει βασικά στάδια όπως την εκκίνηση του παιχνιδιού, την επιλογή του πρώτου παίκτη, και τον κύκλο του παιχνιδιού που περιλαμβάνει το ρίξιμο του ζαριού, την κίνηση των πιονιών, και τον έλεγχο για ειδικές συνθήκες (π.χ. έξοδος από τη βάση, φάγωμα αντίπαλου πιονιού). Το διάγραμμα επίσης δείχνει πώς το παιχνίδι χειρίζεται διάφορα σενάρια, όπως όταν ένας παίκτης φέρνει 6 στο ζάρι, όταν δεν υπάρχει διαθέσιμη κίνηση, ή όταν ένα πιόνι φτάνει στο τέλος. Η διαδικασία επαναλαμβάνεται για κάθε παίκτη μέχρι να τελειώσουν όλα τα πιόνια ενός παίκτη, οπότε και το παιχνίδι ολοκληρώνεται. Το διάγραμμα αυτό παρέχει μια ολοκληρωμένη οπτική αναπαράσταση της λογικής και των κανόνων του παιχνιδιού Ludo, επιτρέποντας την κατανόηση της ροής και των αποφάσεων που λαμβάνονται κατά τη διάρκεια του παιχνιδιού.



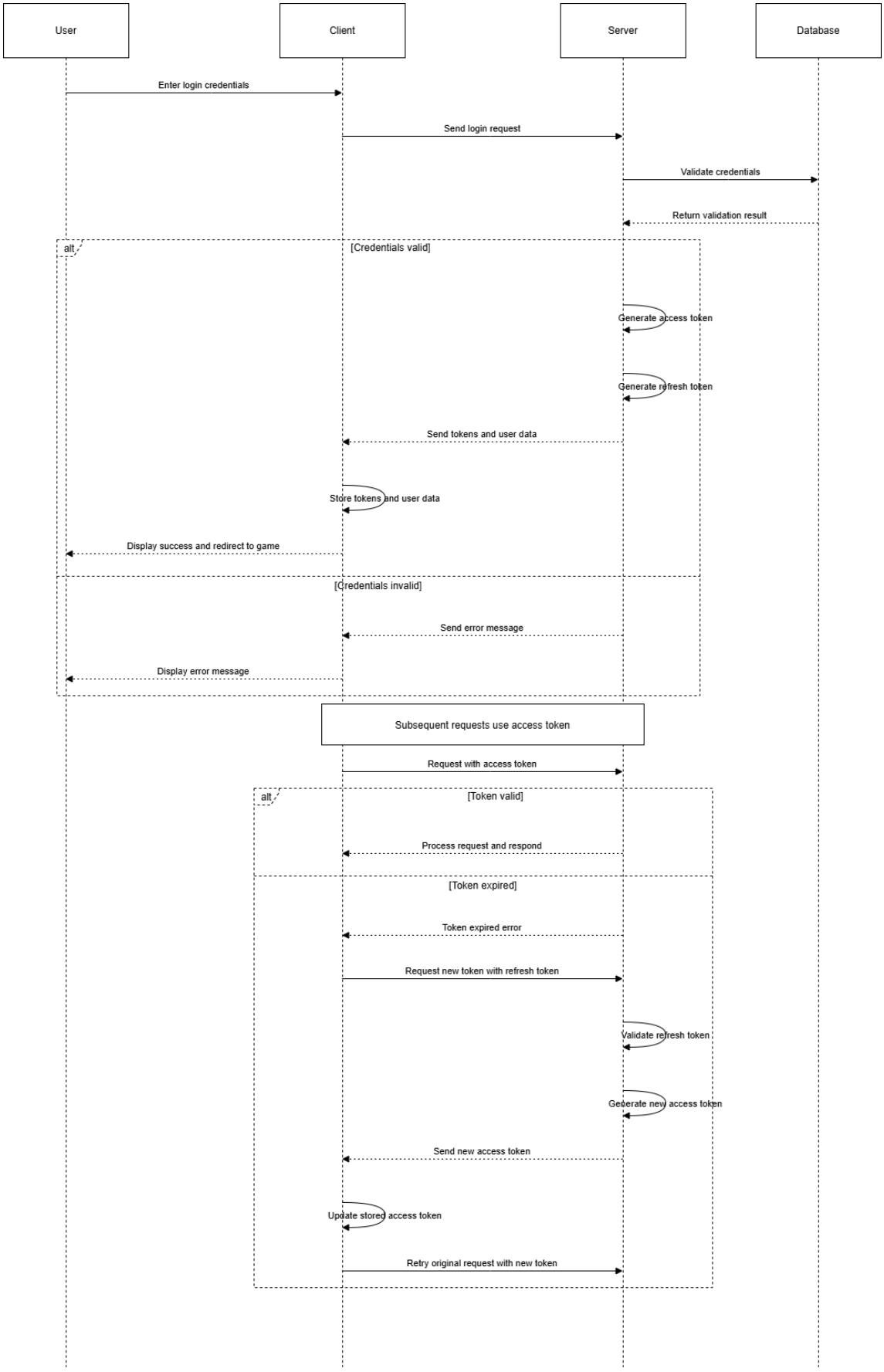
Εικόνα 7: Διάγραμμα ροής

Το διάγραμμα χρονικής ακολουθίας γεγονότων [Εικόνα 8: Διάγραμμα χρονικής ακολουθίας γεγονότων] παρουσιάζει τη ροή πληροφοριών και αλληλεπιδράσεων μεταξύ των διαφόρων συστατικών του συστήματος του παιχνιδιού Ludo, από την έναρξη μέχρι το τέλος μιας παρτίδας. Περιλαμβάνει πέντε βασικές οντότητες: τον Παίκτη, τη Διεπαφή Χρήστη, το Παιχνίδι, τον Server παιχνιδιού και τη Βάση Δεδομένων. Η διαδικασία ξεκινά με το άνοιγμα της εφαρμογής και τη σύνδεση/πιστοποίηση του χρήστη. Στη συνέχεια, απεικονίζεται η δημιουργία ενός νέου παιχνιδιού, όπου ο παίκτης αλληλεπιδρά με τη διεπαφή, η οποία επικοινωνεί με τον server για τη δημιουργία και αποθήκευση του παιχνιδιού στη βάση δεδομένων. Το διάγραμμα δείχνει επίσης τη διαδικασία ένταξης άλλων παικτών στο παιχνίδι, με ένα βρόχο που επαναλαμβάνεται για κάθε νέο παίκτη. Κάθε είσοδος παίκτη ενημερώνει τη βάση δεδομένων και επιβεβαιώνεται από το σύστημα. Μόλις όλοι οι παίκτες είναι έτοιμοι, το παιχνίδι ξεκινά. Κατά τη διάρκεια του παιχνιδιού, υπάρχει συνεχής επικοινωνία μεταξύ του παιχνιδιού και του server για την ενημέρωση της κατάστασης του παιχνιδιού. Τέλος, το διάγραμμα δείχνει το τέλος του παιχνιδιού, όπου γίνεται έλεγχος της τελικής κατάστασης και ενημέρωση των στατιστικών στη βάση δεδομένων. Αυτό το διάγραμμα παρέχει μια ολοκληρωμένη εικόνα της αρχιτεκτονικής του συστήματος και της ροής δεδομένων, αναδεικνύοντας τις αλληλεπιδράσεις μεταξύ του front-end (διεπαφή χρήστη και παιχνίδι) και του back-end (server και βάση δεδομένων) του συστήματος.

Το διάγραμμα ακολουθίας της Εικόνας 9 [Εικόνα 9: Διάγραμμα ακολουθιας Συνδεσης/Αυθεντικοποίησης] απεικονίζει τη διαδικασία αυθεντικοποίησης και διαχείρισης πρόσβασης σε ένα σύστημα, εστιάζοντας στην αλληλεπίδραση μεταξύ του Χρήστη, του Client, του Server και της Βάσης Δεδομένων. Η διαδικασία ξεκινά με την εισαγωγή διαπιστευτηρίων από τον χρήστη. Ο Client στέλνει αυτά τα στοιχεία στον Server, ο οποίος τα επικυρώνει μέσω της Βάσης Δεδομένων. Αν τα διαπιστευτήρια είναι έγκυρα, ο Server δημιουργεί ένα access token και ένα refresh token, τα οποία αποστέλλονται πίσω στον Client. Σε περίπτωση μη έγκυρων διαπιστευτηρίων, ο χρήστης λαμβάνει μήνυμα σφάλματος. Για τις επόμενες αιτήσεις, ο Client χρησιμοποιεί το access token. Αν το token είναι έγκυρο, ο Server επεξεργάζεται το αίτημα. Αν το token έχει λήξει, ο Client χρησιμοποιεί το refresh token για να ζητήσει νέο access token. Ο Server επικυρώνει το refresh token, δημιουργεί νέο access token και το στέλνει πίσω στον Client. Ο Client ενημερώνει το αποθηκευμένο token και επαναλαμβάνει το αρχικό αίτημα με το νέο token. Αυτή η διαδικασία εξασφαλίζει ασφαλή πρόσβαση και διαχείριση συνεδριών, επιτρέποντας συνεχή αυθεντικοποίηση χωρίς να απαιτείται συχνή επανεισαγωγή διαπιστευτηρίων από τον χρήστη.



Εικόνα 8: Διάγραμμα χρονικής ακολουθίας γεγονότων

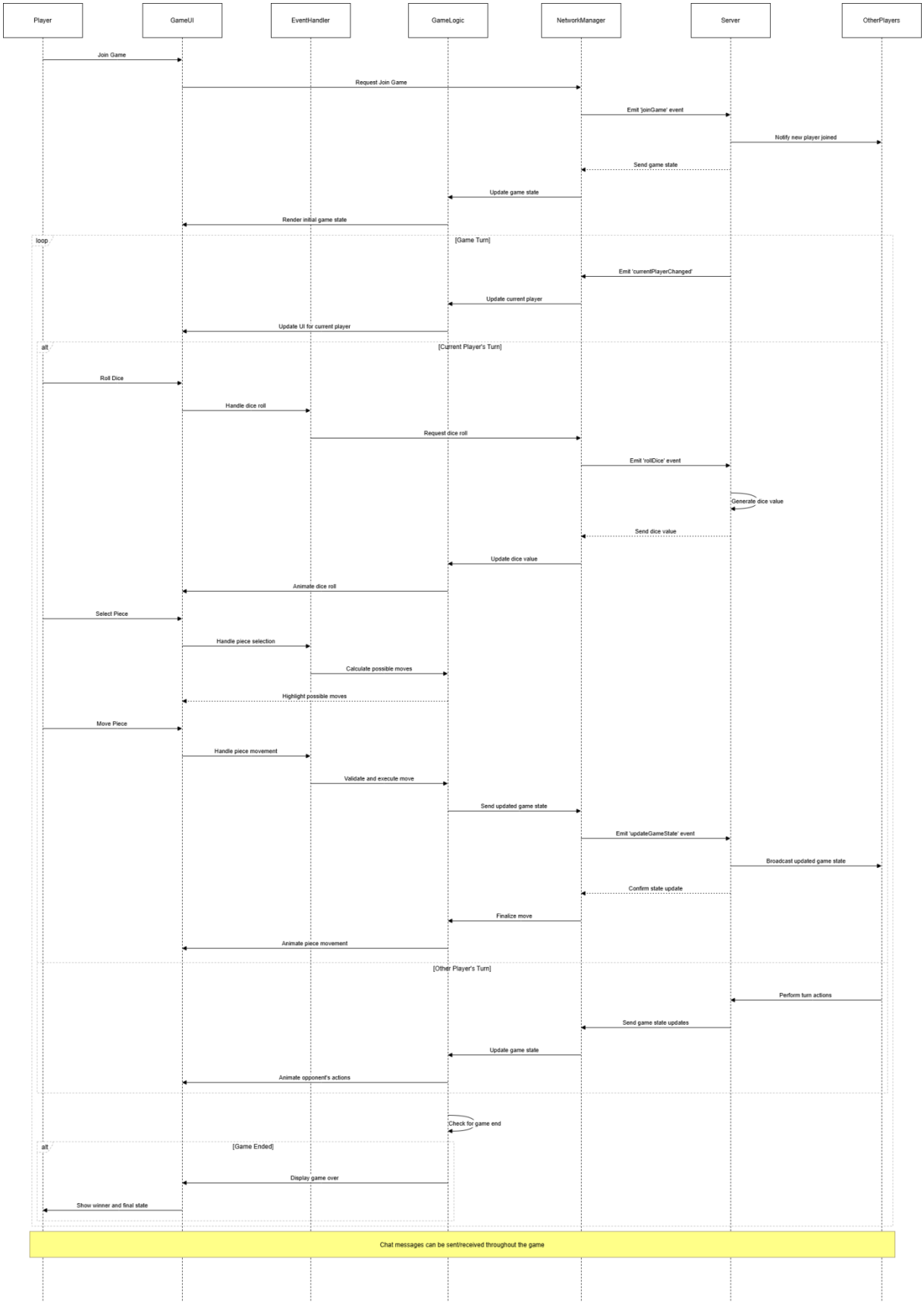


Εικόνα 9: Διάγραμμα χρονικής ακολουθίας Συνδεσης/Αυθεντικοποίησης

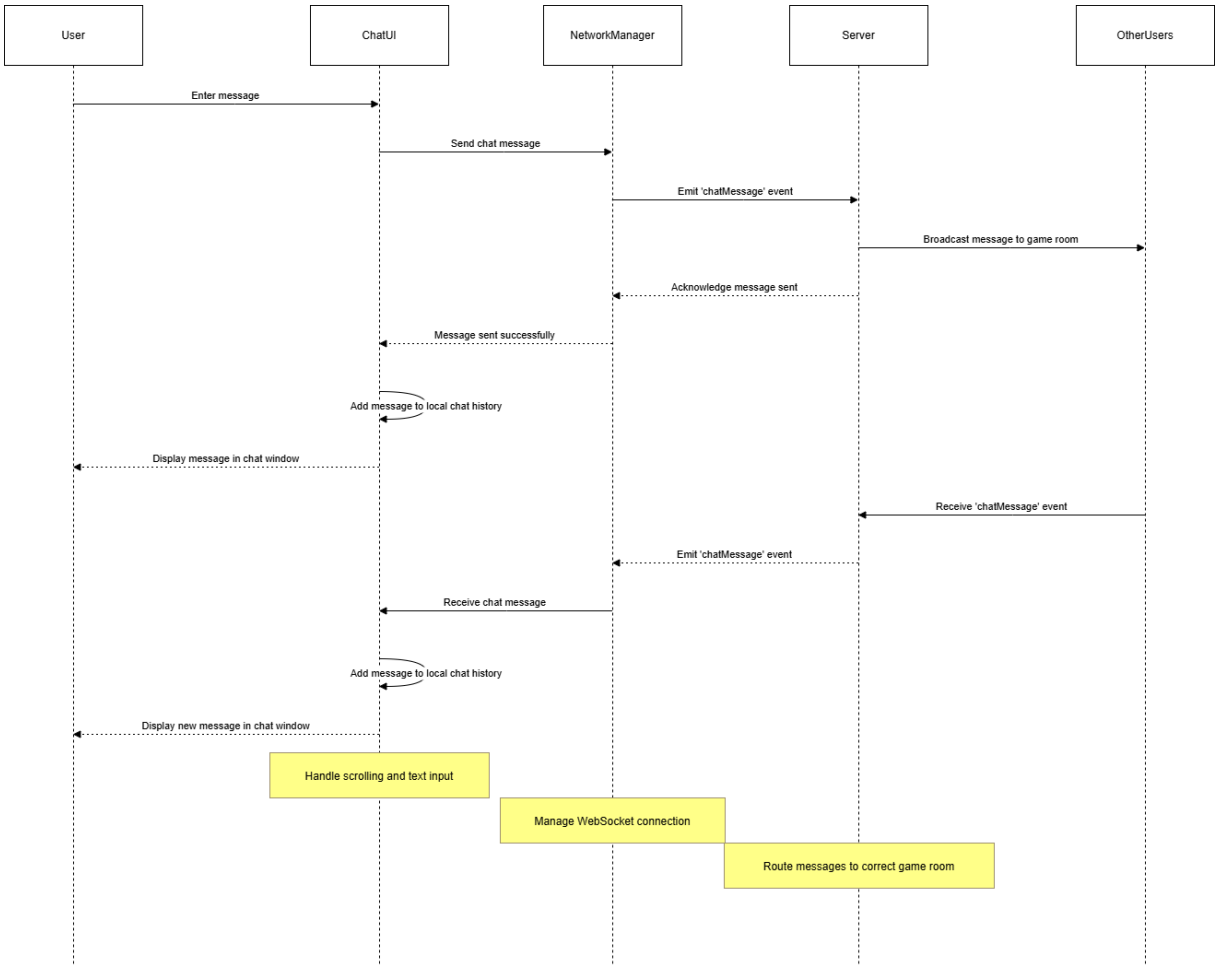
Το διάγραμμα ακολουθίας παιχνιδιού [Εικόνα 10: Διάγραμμα χρονικής ακολουθίας παιχνιδιού] απεικονίζει τη ροή αλληλεπιδράσεων κατά τη διάρκεια ενός διαδικτυακού παιχνιδιού. Παρουσιάζει τις σχέσεις και την επικοινωνία μεταξύ διαφόρων συστατικών του συστήματος, συμπεριλαμβανομένων του Παίκτη, του Game3D (διεπαφή χρήστη), του EventHandler, του GameLogic, του NetworkManager και του Server. Το διάγραμμα δείχνει τη διαδικασία από την σύνδεση ενός παίκτη σε ένα παιχνίδι μέχρι το τέλος του, περιλαμβάνοντας ενέργειες όπως η ρίψη ζαριού, η επιλογή και κίνηση πιονιών, η επικύρωση κινήσεων και η αλλαγή σειράς παικτών. Τονίζει πώς οι ενέργειες του παίκτη μεταφράζονται σε αλλαγές στην κατάσταση του παιχνιδιού και πώς αυτές συγχρονίζονται μεταξύ των διαφόρων συστατικών και του server.

Το διάγραμμα ακολουθίας συνομιλίας [Εικόνα 11: Διάγραμμα χρονικής ακολουθίας συνομιλίας] εστιάζει στη λειτουργικότητα του chat κατά τη διάρκεια του παιχνιδιού. Απεικονίζει τη διαδικασία αποστολής και λήψης μηνυμάτων μεταξύ των χρηστών, του ChatUI, του NetworkManager, του Server και των άλλων χρηστών. Το διάγραμμα παρουσιάζει πώς ένα μήνυμα εισάγεται από έναν χρήστη, πώς μεταδίδεται μέσω του συστήματος, και πώς τελικά εμφανίζεται στους άλλους χρήστες. Επίσης, δείχνει τη διαχείριση του τοπικού ιστορικού συνομιλίας και τον χειρισμό της κύλισης και εισαγωγής κειμένου.

Και τα δύο διαγράμματα υπογραμμίζουν τη σημασία του αποτελεσματικού συγχρονισμού και της επικοινωνίας μεταξύ του client και του server για τη διατήρηση μιας συνεπούς και ομαλής εμπειρίας παιχνιδιού για όλους τους συμμετέχοντες παίκτες.



Εικόνα 10: Διάγραμμα χρονικής ακολουθίας παιχνιδιού



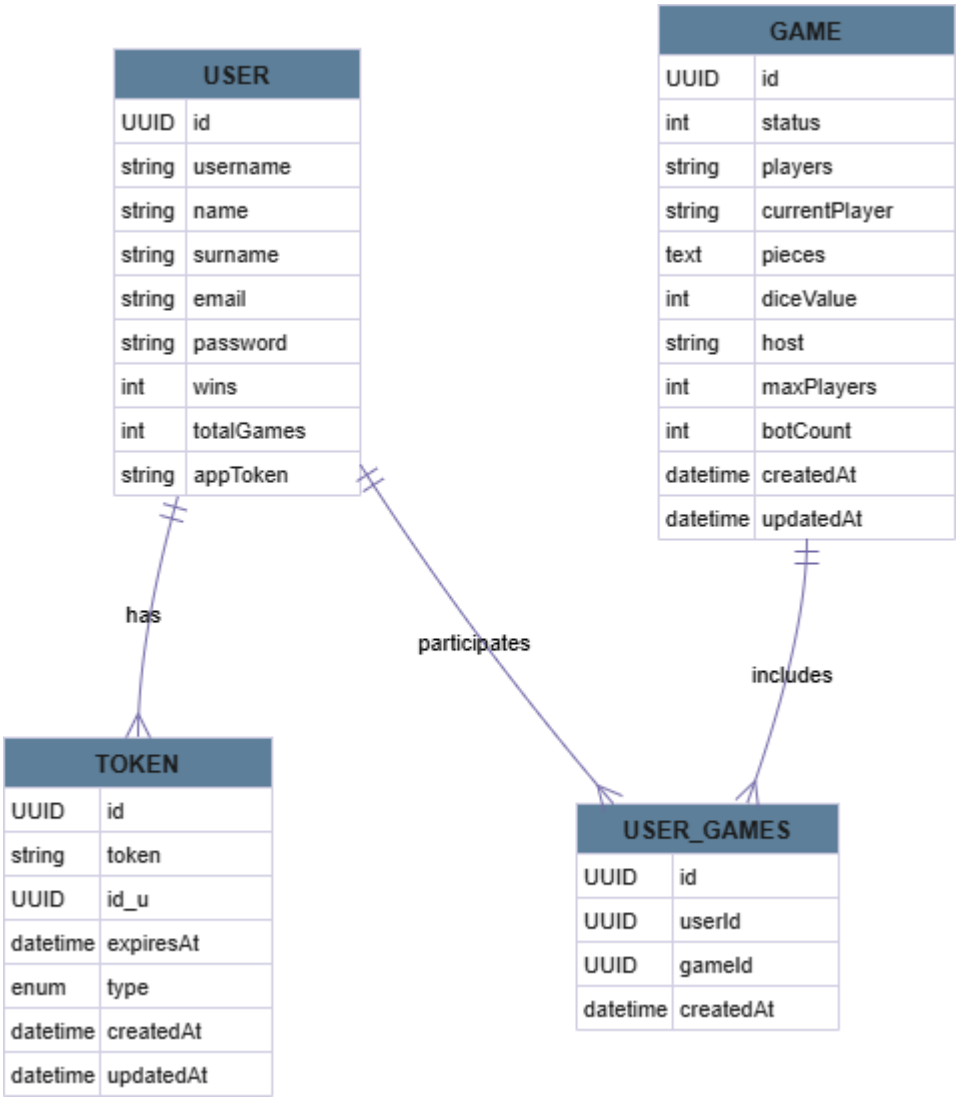
Εικόνα 11: Διάγραμμα χρονικής ακολουθίας συνομιλίας

3.3 Σχεδιασμός βάσης δεδομένων

Το διάγραμμα της Εικόνας 12 [Εικόνα 12: Σχέδιο βάσης δεδομένων] απεικονίζει το σχεσιακό μοντέλο δεδομένων για ένα σύστημα παιχνιδιού, αποτελούμενο από τέσσερις κύριους πίνακες: USER, GAME, TOKEN και USER_GAMES. Ο πίνακας USER περιέχει πληροφορίες για τους χρήστες, όπως username, όνομα, επώνυμο, email, κωδικό πρόσβασης, αριθμό νικών και συνολικών παιχνιδιών, καθώς και ένα appToken. Ο πίνακας GAME αποθηκεύει δεδομένα σχετικά με τα παιχνίδια, συμπεριλαμβανομένων της κατάστασης του παιχνιδιού, των παικτών, του τρέχοντος παίκτη, των θέσεων των πιονιών, της τιμής του ζαριού, του οικοδεσπότη, του μέγιστου αριθμού παικτών και του αριθμού των bots. Ο πίνακας TOKEN διαχειρίζεται τα tokens πρόσβασης, περιλαμβάνοντας το ίδιο το token, τον χρήστη στον οποίο ανήκει, την ημερομηνία λήξης και τον τύπο του token. Ο πίνακας USER_GAMES λειτουργεί ως πίνακας σύνδεσης μεταξύ των χρηστών και των παιχνιδιών,

επιτρέποντας μια σχέση πολλά-προς-πολλά. Οι σχέσεις μεταξύ των πινάκων απεικονίζονται με γραμμές. Ένας χρήστης "has" (έχει) πολλαπλά tokens, "participates" (συμμετέχει) σε πολλαπλά παιχνίδια μέσω του πίνακα USER_GAMES, και ένα παιχνίδι "includes" (περιλαμβάνει) πολλούς χρήστες, επίσης μέσω του πίνακα USER_GAMES.

Αυτή η δομή επιτρέπει την αποτελεσματική διαχείριση των δεδομένων των χρηστών, των παιχνιδιών και των μεταξύ τους σχέσεων, υποστηρίζοντας τη λειτουργικότητα ενός πολύπλοκου συστήματος παιχνιδιού.



Εικόνα 12: Σχέδιο βάσης δεδομένων

3.4 Σύνοψη κεφαλαίου

Το παρόν κεφάλαιο παρουσίασε λεπτομερώς τον σχεδιασμό του συστήματος για το διαδικτυακό παιχνίδι Ludo πολλαπλών παικτών. Η ανάλυση επικεντρώθηκε σε τρεις βασικούς τομείς: την αρχιτεκτονική του συστήματος, τα διαγράμματα του παιχνιδιού και τον σχεδιασμό της βάσης δεδομένων.

Η αρχιτεκτονική του συστήματος αναδεικνύει την πολυπλοκότητα και την καινοτομία της υλοποίησης, συνδυάζοντας στοιχεία από αρχιτεκτονικές πελάτη-εξυπηρετητή και ομότιμων κόμβων. Αυτή η υβριδική προσέγγιση επιτρέπει την αποτελεσματική διαχείριση του παιχνιδιού σε πραγματικό χρόνο, ενώ παράλληλα διασφαλίζει την ασφάλεια και τη συνέπεια των δεδομένων.

Τα διαγράμματα του παιχνιδιού, συμπεριλαμβανομένων των διαγραμμάτων αντικειμένων, ροής και ακολουθίας, παρέχουν μια σαφή οπτική αναπαράσταση της λογικής και των αλληλεπιδράσεων εντός του συστήματος. Αυτά τα διαγράμματα είναι κρίσιμα για την κατανόηση της ροής του παιχνιδιού, της επικοινωνίας μεταξύ των διαφόρων συστατικών και της διαχείρισης των καταστάσεων του παιχνιδιού.

Ο σχεδιασμός της βάσης δεδομένων αποτυπώνει τη δομή των δεδομένων που απαιτούνται για τη λειτουργία του παιχνιδιού, συμπεριλαμβανομένων των πληροφοριών χρηστών, των καταστάσεων παιχνιδιών και των στατιστικών. Η προσεκτική σχεδίαση της βάσης δεδομένων εξασφαλίζει την αποτελεσματική αποθήκευση και ανάκτηση δεδομένων, υποστηρίζοντας τη συνολική απόδοση και κλιμάκωση του συστήματος.

Συνολικά, ο σχεδιασμός που παρουσιάστηκε σε αυτό το κεφάλαιο θέτει τα θεμέλια για μια ισχυρή και ευέλικτη υλοποίηση. Αντιμετωπίζει τις προκλήσεις της διαχείρισης πολλαπλών παικτών, της επικοινωνίας σε πραγματικό χρόνο και της διατήρησης της συνέπειας του παιχνιδιού, ενώ παράλληλα προσφέρει ένα σαφές πλαίσιο για την επακόλουθη φάση υλοποίησης. Ο σχεδιασμός αυτός αποτελεί τη βάση για την ανάπτυξη ενός αξιόπιστου, κλιμακώσιμου και εύχρηστου διαδικτυακού παιχνιδιού Ludo.

Κεφάλαιο 4 - Υλοποίηση

4.1 Backend (Python, Node.js)

Η υλοποίηση του backend επιδεικνύει μια υβριδική προσέγγιση, χρησιμοποιώντας τόσο Python, όσο και Node.js για τη δημιουργία μιας ισχυρής αρχιτεκτονικής στην πλευρά του διακομιστή.

4.1.1 Λογική παιχνιδιού

Η λογική του παιχνιδιού σε Python ενσωματώνεται κυρίως στην κλάση GameLogic. Αυτή η κλάση επιδεικνύει προηγμένες τεχνικές αντικειμενοστρεφούς προγραμματισμού για τη διαχείριση της πολύπλοκης κατάστασης και των κανόνων του παιχνιδιού Ludo. Μερικές απο τις βασικότερες μεθόδους που δείχνουν πως οι κανόνες του παιχνιδιού μεταφράζονται σε κώδικα αναλύονται παρακάτω:

1. Calculate_possible_moves(self, piece): Αυτή η μέθοδος [Παράρτημα 1: Συνάρτηση ελέγχου πιθανών κινήσεων] αποτελεί ακρογωνιαίο λίθο της λογικής του παιχνιδιού, καθορίζοντας τις έγκυρες κινήσεις για ένα δεδομένο πιόνι βάσει της τρέχουσας κατάστασης του παιχνιδιού και του ζαριού. Η πολυπλοκότητα της εγκείται στο χειρισμό διαφόρων σεναρίων. Υπάρχουν έλεγχοι για έξοδο από τη βάση, κανονική κίνηση στον πίνακα, είσοδο στη περιοχή τερματισμού και για χειρισμό ασφαλών θέσεων και μπλοκαρισμένων διαδρομών. Η μέθοδος χρησιμοποιεί υπό συνθήκη λογική και επαναλαμβάνεται μέσω της διαδρομής του ταμπλό.
2. Handle_piece_movement(self, selected_piece): Αυτή η μέθοδος [Εικόνα 13: Συνάρτηση εκτέλεσης κίνησης] εκτελεί μία κίνηση, ενημερώνοντας την κατάσταση του παιχνιδιού. Περιλαμβάνει ελέγχους για την επικύρωση της κίνησης, την ενημέρωση της θέσης του πιονιού, τον χειρισμό αιχμαλωσίας, τη διαχείρηση στιβαγμένων πιονιών και την αλλαγή σειράς παίκτη.
3. Update_game_state(self, game_data): Αυτή η μέθοδος συγχρονίζει την τοπική κατάσταση του παιχνιδιού με τα δεδομένα που λαμβάνονται από τον διακομιστή. Διατηρεί την συνέπεια του παιχνιδιού ενημερώνοντας τη σειρά του παίκτη και τη θέση των πιονιών και τέλος συγχρονίζει τη ρήψη των ζαριών και επιστρέφει το ίδιο αποτέλεσμα σε όλους τους παίκτες.

4. Is blocked(self, position, moving_color): Αυτή η βοηθητική μέθοδος ελέγχει αν μία θέση είναι μπλοκαρισμένη. Δείχνει πως υλοποιούνται πολύπλοκότεροι κανόνες όπως η παρουσία πολλαπλών πιονιών στην ίδια θέση, η ασφάλιση των θέσεων εξόδου από τη βάση και το μπλοκάρισμα εισόδου στη βάση και στα τελικά μονοπάτια αντιπάλου.

Στο Node.js, η λογική του παιχνιδιού κατανέμεται σε διάφορα αρχεία, με το socketService.js να παίζει κεντρικό ρόλο. Αυτή η προσέγγιση δείχνει πώς να δομηθεί μια μεγάλη εφαρμογή με διαχωρισμό ευθυνών. Μερικές από τις βασικότερες λειτουργίες αναλύονται παρακάτω:

1. handleDiceRoll: Σε αυτή τη συνάρτηση γίνεται η διαχείρηση της διαδικασίας της ρήψης του ζαριού. Αποτελείται από την παραγωγή ενός τυχαίου αριθμού, την επικύρωση σειράς, την ενημέρωση της κατάστασης του παιχνιδιού και την εκπομπή του αποτελέσματος στους παίκτες.
2. updateGameState: Πρόκειται για μία πολύπλοκη συνάρτηση που επεξεργάζεται ενημερώσεις τις κατάστασης του παιχνιδιού και παίζει καίριο ρόλο στην ενημέρωση των δεδομένων στη βάση αλλά και στην οθόνη των παικτών. Επικυρώνει τις κινήσεις των παικτών και ενημερώνει τη βάση δεδομένων με τη νέα κατάσταση του παιχνιδιού. Ελέγχει για συνθήκες νίκης, αν έχουν τερματήσει και τα 4 πιόνια του παίκτη, διαχειρίζεται τη μετάβαση σειράς και μεταδίδει τις ενημερώσεις σε όλους τους παίκτες.
3. joinGame: Η διαδικασία εισόδου ενός παίκτη σε ένα παιχνίδι υλοποιείται σε αυτή τη συνάρτηση. Για την υλοποίηση της δομήθηκαν έλεγχοι για τον χειρισμό ταυτόχρονης πρόσβασης, την αρχικοποίηση της κατάστασης του παιχνιδιού και τα όρια των παικτών. Στο τέλος τη συνάρτησης υπάρχει η ενημέρωση της βάσης δεδομένων και η διαχείρηση των δωματίων του Socket.IO.

```python
def handle_piece_movement(self, selected_piece):
    self.selected_piece = selected_piece
    current_player = self.get_current_player()
    if not current_player or not selected_piece:
        return False

    move_successful, captured = self.board.move_piece(selected_piece, self.dice_value, self.players, self.current_player_id)

    if move_successful:
        self.handle_captures(captured)
        current_player.deselect_all_pieces()
        self.selected_piece = None

        self.dice_value = 0
    return move_successful
```

Εικόνα 13: Συνάρτηση εκτέλεσης κίνησης

4.1.2 Διαχείρηση δεδομένων

Η διαχείριση δεδομένων υλοποιείται σε Node.js χρησιμοποιώντας το Sequelize ORM, ένα ισχυρό εργαλείο για τη διαχείριση σχεσιακών βάσεων δεδομένων. Αυτή η προσέγγιση επιτρέπει την αποτελεσματική αποθήκευση και ανάκτηση δεδομένων, καθώς και τη διαχείριση σχέσεων μεταξύ διαφορετικών οντοτήτων του παιχνιδιού. Για την υλοποίηση του Ludo χρειάστηκαν 4 μοντέλα με μοναδικό κοινό όλων τη χρήση ενός ID ως πρωτεύον κλειδί.

1. Μοντέλο Game: Το μοντέλο αυτό αναπαριστά την κατάσταση ενός παιχνιδιού. Περιλαμβάνει πεδία όπως το status που χρησιμοποιεί αριθμούς που αναπαραστούν τις διάφορες καταστάσεις του παιχνιδιού, το players που αποθηκεύει τους παίκτες ως συμβολοσειρά χωρισμένη με κόμματα, το pieces που χρησιμοποιεί προσαρμοσμένες μεθόδους getter και setter για τη διαχείρηση JSON δεδομένων, και άλλα όπως την σειρά, τη τιμή του ζαριού, τον οικοδεσπότη και τον αριθμό των bot.
2. Μοντέλο User: Το μοντέλο [Εικόνα 14: Μοντέλο χρήστη] αναπαριστά έναν χρήστη του παιχνιδιού. Περιλαμβάνει τις βασικές πληροφορίες χρήστη όπως το όνομα, το επώνυμο, το ψευδώνυμο και το email καθώς και τα στατιστικά του. Χρησιμοποιείται επίσης για την φύλαξη του κωδικού σε κρυπτογραφημένη μορφή. Και τέλος αποθηκεύεται το appToken για να αυθεντικοποίηση με το παράθυρο της εφαρμογής.
3. Μοντέλο Token: Το μοντέλο αυτό διαχειρίζεται τα tokens αυθεντικοποίησης με τα παράθυρα της εφαρμογής. Αποθηκεύει τη τιμή του token καθώς και υπόλοιπα στοιχεία του όπως τον τύπο, την ημερομηνία λήξης, δημιουργίας και ανανέωσης και τέλος το id του χρήστη στον οποίο ανήκει το token.
4. Μοντέλο UserGames: Το μοντέλο αυτό διαχειρίζεται τη σχέση πολλά προς πολλά μεταξύ παικτών και παιχνιδιών. Περιλαμβάνει την ημερομηνία δημιουργίας του παιχνιδιού, το id του παιχνιδιού και το id του νικητή του παιχνιδιού.

```
const User = sequelize.define('User', {
  id: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    primaryKey: true,
  },
  username: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  surname: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  email: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  wins: {
    type: DataTypes.INTEGER,
    defaultValue: 0,
  },
  totalGames: {
    type: DataTypes.INTEGER,
    defaultValue: 0,
  },
  appToken: {
    type: DataTypes.STRING,
    allowNull: true,
  },
}, {
  timestamps: false,
  tableName: 'user',
});

module.exports = User;
```

Εικόνα 14: Μοντέλο χρήστη

4.1.3 Επικοινωνία μέσω sockets

Η επικοινωνία μέσω sockets αποτελεί θεμελιώδες στοιχείο για τη υλοποίηση του παιχνιδιού και της ζωντανής συνομιλίας σε πραγματικό χρόνο. Η χρήση της βιβλιοθήκης Socket.IO στο Node.js επιτρέπει την αμφίδρομη και ασύγχρονη επικοινωνία μεταξύ του διακομιστή και των πελατών. Αυτή η προσέγγιση είναι κρίσιμη για τη διατήρηση συγχρονισμένης κατάστασης σε όλους τους παίκτες του παιχνιδιού. Παρακάτω αναλύονται μερικές από τις σημαντικότερες λειτουργίες της εφαρμογής.

Για τη διαχείρηση συνδέσεων υλοποιήθηκε μία συνάρτηση που εισάγει και ρυθμίζει τους listeners για διάφορα γεγονότα [Εικόνα 15: Συνάρτηση διαχείρησης συνδέσεων]. Αυτή η συνάρτηση διαχειρίζεται νέες συνδέσεις και καθορίζει πως ο διακομιστής θα ανταποκρίνεται σε διάφορα γεγονότα που αποστέλλονται από τους πελάτες. Εξίσου σημαντική λειτουργία είναι η ενημέρωση κατάστασης παιχνιδιού. Πιο συγκεκριμένα, η συνάρτηση updateGameState είναι υπεύθυνη για την ενημέρωση και μετάδοση της κατάστασης του παιχνιδιού σε όλους τους ενεργούς παίκτες. Διασφαλίζει οτι όλοι οι παίκτες έχουν την πιο πρόσφατη κατάσταση παιχνιδιού, και διατηρεί τη συνοχή της εμπειρίας του παιχνιδιού. Στη συνέχεια υλοποιήθηκε η συνάρτηση handleDiceRoll. Αυτή διασφαλίζει ότι μόνο ο τρέχων παίκτης μπορεί να ρίξει το ζάρι και στο τέλος μεταδίδει το αποτέλεσμα του σε όλους τους παίκτες. Τέλος, η συνάρτηση διαχείρησης της ζωντανής επικοινωνίας υλοιποιείται μέσω της συνάρτησης chatMessage και επιτρέπει στους παίκτες να επικοινωνούν μεταξύ τους κατά τη διάρκεια του παιχνιδιού, ενισχύοντας τη κοινωνική πτυχή της εμπειρίας.

```
io.on("connection", (socket) => {
  console.log("New client connected:", socket.id);

  socket.on("checkUserInGame", (data) => checkUserInGame(socket, data));
  socket.on("createGame", (data) => createGame(socket, data));
  socket.on("joinGame", (data) => joinGame(socket, io, data));
  socket.on("joinRoom", (data) => joinSocketRoom(socket, data));
  socket.on("leaveRoom", (data) => leaveSocketRoom(socket, data));
  socket.on("reconnectToGame", (data) => reconnectToGame(socket, io, data));
  socket.on("leaveGame", (data) => leaveGame(socket, io, data));
  socket.on("requestGameState", (data) => requestGameState(socket, data));
  socket.on("startGame", (data) => startGame(socket, io, data));
  socket.on("chatMessage", (data) => chatMessage(socket, io, data));
  socket.on("updateGameState", (data) => updateGameState(socket, io, data));
  socket.on("skipTurn", (data) => skipTurn(socket, io, data));
  socket.on("waitForAppToken", (appToken) => waitForAppToken(socket, appToken));
  socket.on("rollDice", (data) => handleDiceRoll(socket, io, data));

  socket.on("disconnect", () => {
    console.log("Client disconnected:", socket.id);
  });
});
```

Εικόνα 15: Συνάρτηση διαχείρησης συνδέσεων

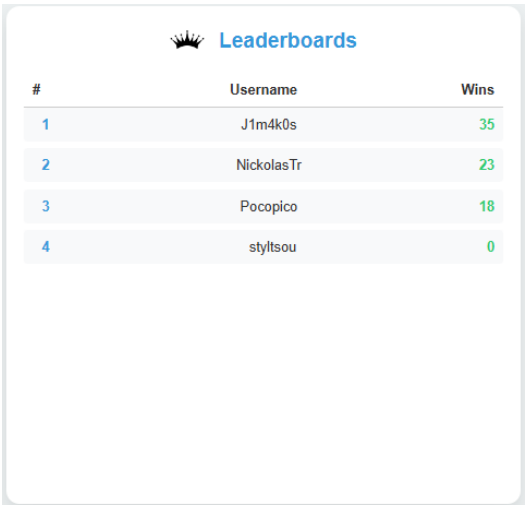
4.2 Frontend (React)

Το frondend της εφαρμογής υλοποιήθηκε χρησιμοποιώντας React, μια δημοφιλή βιβλιοθήκη JavaScript για την κατασκευή διαδραστικών δειπαφών χρήστη. Η επιλογής της επιτρέπει τη δημιουργία ενός δυναμικού και αποκρίσιμου περιβάλλοντος χρήστη, ικανού να χειριστεί τις πολύπλοκες αλληλεπιδράσεις και ενημερώσεις σε πραγματικό χρόνο που απαιτούνται για ένα παιχνίδι πολλαπλών παικτών.

4.2.1 Σχεδιασμός διεπαφής χρήστη

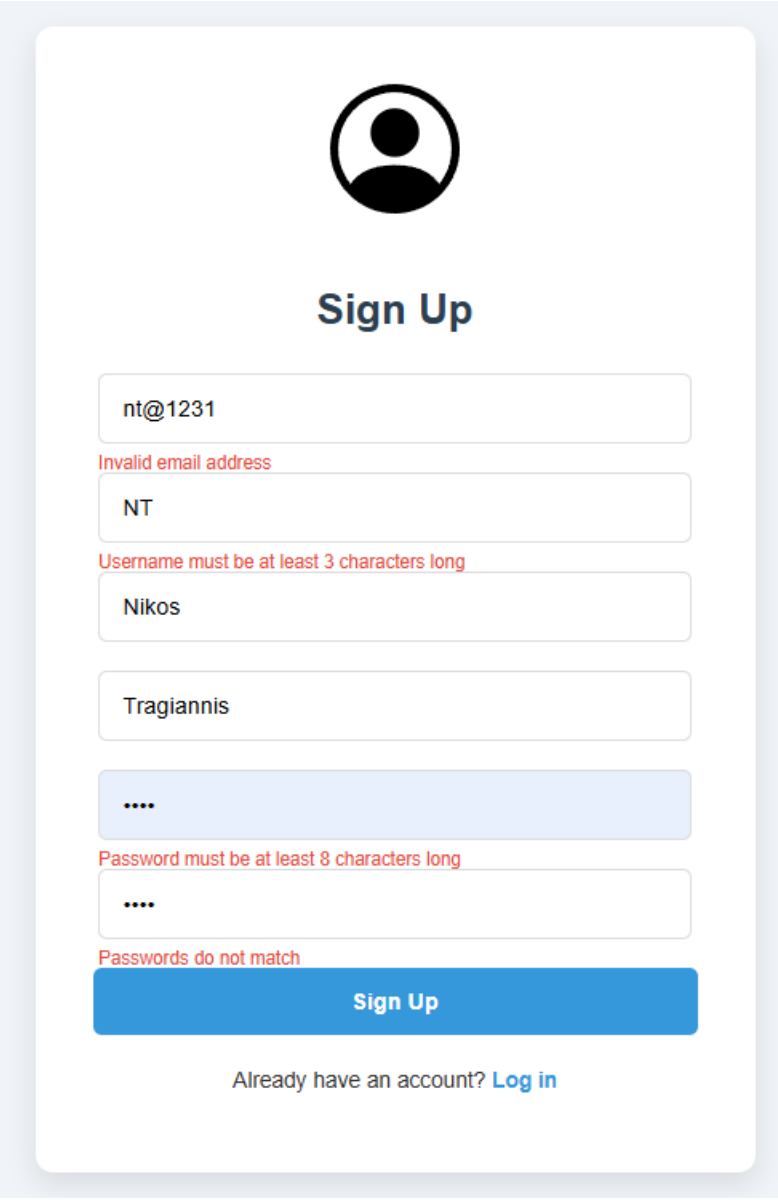
Ο σχεδιασμός της διεπαφής χρήστη επικεντρώθηκε στη δημιουργία μιας καθαρής και ελκυστικής εμπειρίας για τους παίκτες. Η εφαρμογή αποτελείται από διάφορα κύρια components, καθένα από τα οποία εξυπηρετεί έναν συγκεκριμένο σκοπό στη ροή του παιχνιδιού.

Το HomePage component λειτουργεί ως η κεντρική πύλη της εφαρμογής. Υλοποιήθηκε ως ένα λειτουργικό component React που χρησιμοποιεί το useState hook για τη διαχείριση της κατάστασης του πίνακα κατάταξης και το useEffect hook για το fetching των δεδομένων από το API κατά τη φόρτωση της σελίδας. Το component χωρίζεται σε τέσσερα κύρια τμήματα: ένα που απεικονίζει το ταμπλό του παιχνιδιού, ένα που παρουσιάζει τους κανόνες του παιχνιδιού, ένα που εμφανίζει τον πίνακα κατάταξης [Εικόνα 16: Πίνακας κατάταξης], και ένα που παρέχει πληροφορίες για τους δημιουργούς. Ο πίνακας κατάταξης ενημερώνεται δυναμικά, χρησιμοποιώντας την τεχνική του conditional rendering για να εμφανίσει ένα μήνυμα φόρτωσης ή σφάλματος όταν χρειάζεται.



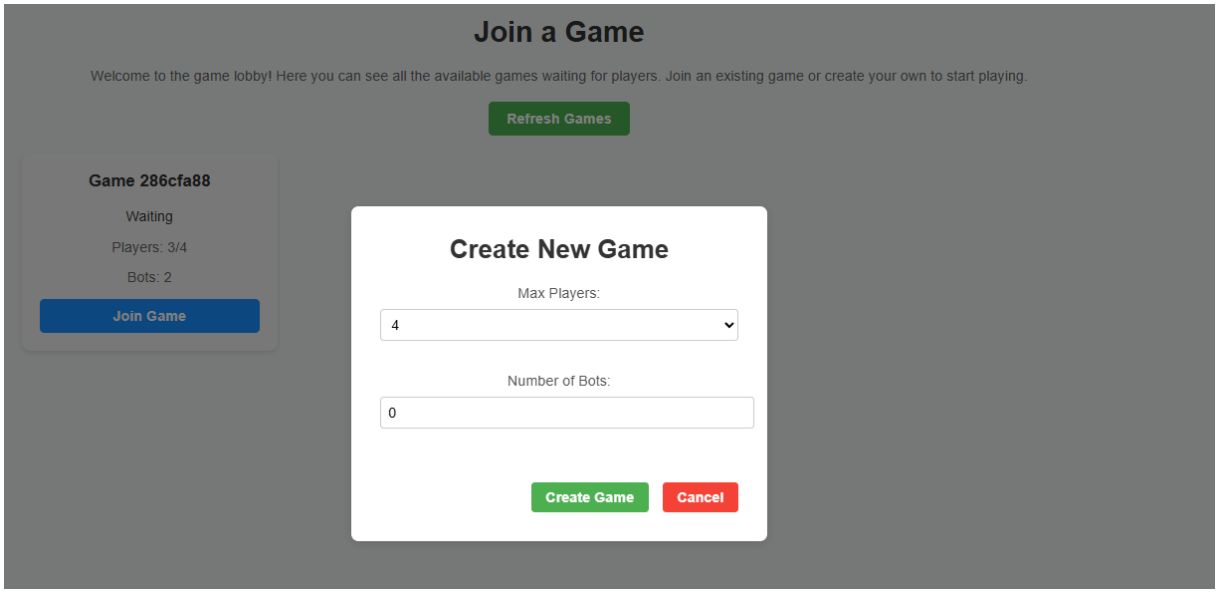
Εικόνα 16: Πίνακας κατάταξης

Τα components LoginPage και SignupPage [Εικόνα 17: Φόρμα εγγραφής] διαχειρίζονται τη διαδικασία αυθεντικοποίησης χρήστη. Και τα δύο χρησιμοποιούν το useState hook για τη διαχείριση της κατάστασης των φορμών εισόδου. Το LoginPage περιλαμβάνει επίσης λογική για τη διατήρηση της σύνδεσης του χρήστη (με χρήση του localStorage) και χειρισμό σφαλμάτων σύνδεσης. Το SignupPage ενσωματώνει εκτεταμένους ελέγχους επικύρωσης για τα πεδία εισόδου, χρησιμοποιώντας τόσο client-side validation, όσο και server-side validation μέσω του API. Και τα δύο components χρησιμοποιούν το useNavigate hook από το react-router-dom για την πλοήγηση μετά από επιτυχημένη αυθεντικοποίηση.



Εικόνα 17: Φόρμα εγγραφής

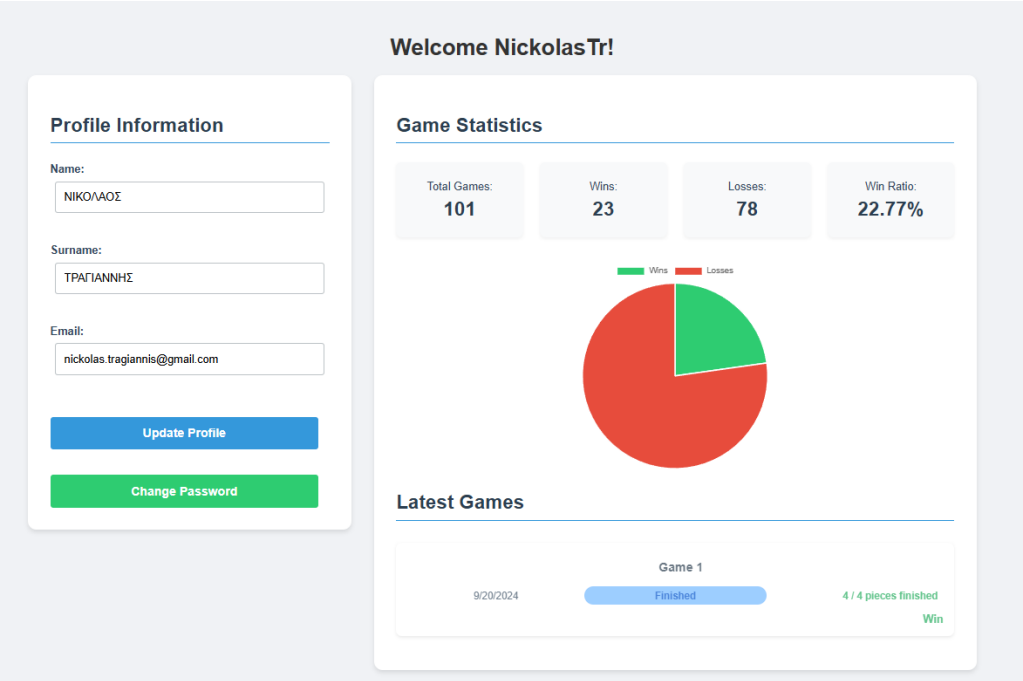
Το JoinGamePage [Εικόνα 18: Αντικείμενα σύνδεσης και δημιουργίας παιχνιδιού] component είναι υπεύθυνο για την εμφάνιση διαθέσιμων παιχνιδιών και τη διαδικασία ένταξης σε ένα παιχνίδι. Χρησιμοποιεί το useEffect hook για να φέρει τη λίστα των διαθέσιμων παιχνιδιών κατά τη φόρτωση και το useState για να διαχειριστεί αυτά τα δεδομένα. Ενσωματώνει επίσης λογική για τη δημιουργία νέου παιχνιδιού, χρησιμοποιώντας ένα modal component (NewGameModal) που εμφανίζεται όταν ο χρήστης επιλέγει να δημιουργήσει νέο παιχνίδι. Το component χρησιμοποιεί το socket.io-client για επικοινωνία σε πραγματικό χρόνο με τον server, επιτρέποντας άμεσες ενημερώσεις όταν νέα παιχνίδια δημιουργούνται ή όταν η κατάσταση υπαρχόντων παιχνιδιών αλλάζει.



Εικόνα 18: Αντικείμενα σύνδεσης και δημιουργίας παιχνιδιού

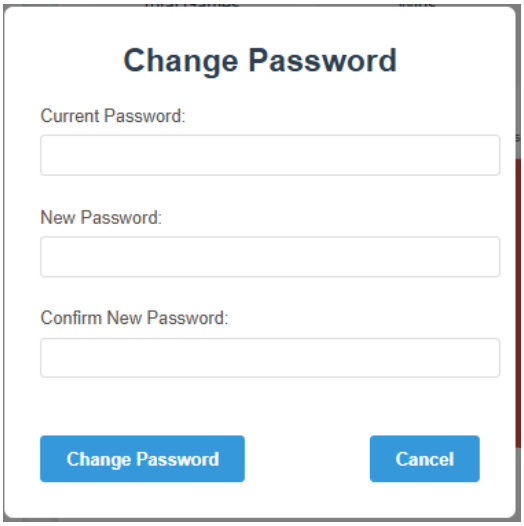
Το GamePage component αποτελεί τον πυρήνα της εμπειρίας παιχνιδιού Ludo. Χρησιμοποιεί React hooks όπως useState για τη διαχείριση της τοπικής κατάστασης και useEffect για τον χειρισμό των socket events και άλλων side effects. Το component αξιοποιεί το SocketContext και το AuthContext για τη διαχείριση της σύνδεσης και της αυθεντικοποίησης. Περιλαμβάνει λειτουργίες όπως το chat, τη διαχείριση των παικτών, και τον έλεγχο της κατάστασης του παιχνιδιού. Χειρίζεται διάφορες καταστάσεις όπως η σύνδεση, τα σφάλματα σύνδεσης, η αναμονή για την έναρξη του παιχνιδιού και η έναρξη του παιχνιδιού. Το component παρέχει επίσης λειτουργικότητα για την έναρξη του παιχνιδιού (για τον οικοδεσπότη) και την αποχώρηση από το παιχνίδι.

Το ProfilePage component [Εικόνα 19: Σελίδα προφίλ παίκτη] είναι υπεύθυνο για την εμφάνιση και διαχείριση των πληροφοριών του προφίλ του χρήστη. Χρησιμοποιεί το useState hook για τη διαχείριση της τοπικής κατάστασης των δεδομένων του χρήστη και των επεξεργασμένων πεδίων. Το useEffect hook χρησιμοποιείται για την ανάκτηση των δεδομένων του χρήστη κατά τη φόρτωση της σελίδας. Το component περιλαμβάνει λειτουργίες για την επεξεργασία των πληροφοριών του προφίλ, την εμφάνιση στατιστικών παιχνιδιού και το ιστορικό των πρόσφατων παιχνιδιών. Ενσωματώνει επίσης τη βιβλιοθήκη Chart.js για την οπτικοποίηση των στατιστικών του παίκτη με τη μορφή γραφήματος πίτας.



Εικόνα 19: Σελίδα προφίλ παίκτη

Ένα σημαντικό χαρακτηριστικό του ProfilePage είναι το PasswordChangeModal [Εικόνα 20: Παράθυρο αλλαγής κωδικού], ένα ξεχωριστό component που χειρίζεται τη διαδικασία αλλαγής κωδικού πρόσβασης. Αυτό το modal ενεργοποιείται μέσω ενός κουμπιού στη σελίδα του προφίλ και εμφανίζεται ως ένα αναδυόμενο παράθυρο πάνω από το υπόλοιπο περιεχόμενο. Το PasswordChangeModal χρησιμοποιεί το useState hook για τη διαχείριση της κατάστασης της φόρμας αλλαγής κωδικού και περιλαμβάνει ελέγχους επικύρωσης για να διασφαλίσει ότι ο νέος κωδικός πρόσβασης πληροί τις απαιτήσεις ασφαλείας. Η αλλαγή του κωδικού πρόσβασης πραγματοποιείται μέσω ασύγχρονης κλήσης API, με κατάλληλο χειρισμό σφαλμάτων και ανατροφοδότηση προς τον χρήστη.



Εικόνα 20: Παράθυρο αλλαγής κωδικού

4.3 Ανάπτυξη παιχνιδιού (Pygame)

Η υλοποίηση των γραφικών και ηχητικών εφέ στο παιχνίδι Ludo αποτελεί ένα κρίσιμο στοιχείο για τη δημιουργία μιας ελκυστικής και διαδραστικής εμπειρίας παιχνιδιού. Η χρήση της βιβλιοθήκης Pygame επιτρέπει τον αποτελεσματικό χειρισμό των οπτικών και ακουστικών στοιχείων, ενισχύοντας σημαντικά την αλληλεπίδραση του χρήστη με το παιχνίδι.

Η αρχιτεκτονική του συστήματος γραφικών και ήχου βασίζεται σε διάφορες κλάσεις, καθεμία με συγκεκριμένο ρόλο στην απόδοση του παιχνιδιού. Η κλάση GameRenderer αποτελεί τον πυρήνα της οπτικής απόδοσης, διαχειριζόμενη την απεικόνιση του ταμπλό, των πιονιών και των διαφόρων οπτικών εφέ. Η κλάση Board είναι υπεύθυνη για τη δομή και τη λογική του ταμπλό του παιχνιδιού, ενώ η κλάση Piece [Εικόνα 21: Κλάση πιονιού (Piece)] αντιπροσωπεύει τα μεμονωμένα πιόνια και τις ιδιότητές τους. Η κλάση InfoRenderer χειρίζεται την απεικόνιση πληροφοριών σχετικών με το παιχνίδι, όπως το σκορ και την τρέχουσα κατάσταση του παιχνιδιού.

Η μέθοδος render_game_board [Εικόνα 22: Μέθοδος οπτικής απεικόνισης του παιχνιδιού] της κλάσης GameRenderer είναι κεντρικής σημασίας για την οπτική αναπαράσταση του παιχνιδιού. Αυτή η μέθοδος συνθέτει όλα τα οπτικά στοιχεία, συμπεριλαμβανομένου του ταμπλό, των πιονιών και των πιθανών κινήσεων. Χρησιμοποιεί τις μεθόδους draw της κλάσης Board για να σχεδιάσει το βασικό ταμπλό και τα πιόνια, ενώ επίσης καλεί τη μέθοδο highlight_possible_move για να τονίσει τις διαθέσιμες κινήσεις για τον τρέχοντα παίκτη.

```python
class Piece:
    def __init__(self, color, start_position):
        self.color = color
        self.position = None
        self.is_home = True
        self.is_finished = False
        self.is_selected = False
        self.start_position = start_position
        self.stacked_pieces = 0

    def move(self, new_position):...

    def move_out_of_home(self):...

    def finish(self):...

    def send_home(self):...

    def __str__(self):...
```

Εικόνα 21: Κλάση πιονιού (Piece)

```python
def render_game_board(self, board, players, selected_piece, possible_move, exclude_index=None, player_id=None):
    self.screen.fill((255, 255, 255))
    self.board = board
    if self.board:
        self.board.draw(self.screen)

        if selected_piece and possible_move and possible_move != ['finished']:
            self.highlight_possible_move(possible_move)

        for player in players:
            for index, piece in enumerate(player.pieces):
                if exclude_index == index and player.player_id == player_id:
                    continue

                if piece.is_home:
                    base_pos = self.board.get_base_position(player.color, player.pieces.index(piece))
                    self.board.draw_piece(self.screen, piece, base_pos)
                elif not piece.is_finished:
                    self.board.draw_piece(self.screen, piece, piece.position)
```

Εικόνα 22: Μέθοδος οπτικής απεικόνισης του παιχνιδιού

Εκτός από τις βασικές μεθόδους απεικόνισης, υπάρχουν αρκετές εξειδικευμένες μέθοδοι που συμβάλλουν σημαντικά στην οπτική και λειτουργική πλευρά του παιχνιδιού. Η μέθοδος draw_piece της κλάσης Board είναι υπεύθυνη για τη σχεδίαση των μεμονωμένων πιονιών, συμπεριλαμβάνοντας λεπτομέρειες όπως το χρώμα, το περίγραμμα και την επισήμανση των επιλεγμένων πιονιών. Η get_pixel_position μετατρέπει τις συντεταγμένες του ταμπλό σε πραγματικές θέσεις pixel στην οθόνη, διευκολύνοντας την ακριβή τοποθέτηση των στοιχείων. Η μέθοδος highlight_possible_move δημιουργεί ένα ημιδιαφανές εφέ επικάλυψης για να τονίσει τις έγκυρες κινήσεις, ενισχύοντας τη διαδραστικότητα του παιχνιδιού. Η roll_dice_animation της κλάσης InfoRenderer παρέχει μια οπτικά ελκυστική αναπαράσταση της ρίψης ζαριού, χρησιμοποιώντας τυχαία εναλλαγή αριθμών πριν καταλήξει στο τελικό αποτέλεσμα. Η μέθοδος draw_capture_marker δημιουργεί ένα οπτικό εφέ "X" για να σηματοδοτήσει τη θέση όπου ένα πιόνι "έφαγε" ένα άλλο, προσθέτοντας μια επιπλέον οπτική ανατροφοδότηση στο παιχνίδι. Τέλος, η μέθοδος process_capture_animation χειρίζεται την οπτική αναπαράσταση της "σύλληψης" ενός πιονιού, συντονίζοντας την κίνηση και τα εφέ που σχετίζονται με αυτό το σημαντικό γεγονός του παιχνιδιού.

Η υλοποίηση των animations αποτελεί ένα εξίσου σημαντικό στοιχείο για την ενίσχυση της οπτικής εμπειρίας του παιχνιδιού. Η μέθοδος animate της κλάσης GameRenderer είναι υπεύθυνη για τη δημιουργία ομαλών κινήσεων των πιονιών. Χρησιμοποιεί τεχνικές γραμμικής παρεμβολής για να υπολογίσει τις ενδιάμεσες θέσεις κατά τη διάρκεια της κίνησης ενός πιονιού, δημιουργώντας την ψευδαίσθηση της συνεχούς κίνησης. Επιπλέον, η μέθοδος ενσωματώνει ένα εφέ "ίχνους", σχεδιάζοντας μια γραμμή με σταδιακά μειούμενη διαφάνεια πίσω από το κινούμενο πιόνι, προσθέτοντας μια επιπλέον οπτική διάσταση στην κίνηση.

Τα ηχητικά εφέ παίζουν επίσης σημαντικό ρόλο στην ενίσχυση της εμπειρίας του παιχνιδιού. Η κλάση GameRenderer χρησιμοποιεί τη βιβλιοθήκη mixer [Εικόνα 23: Χρήση βιβλιοθήκης mixer] του Pygame για να φορτώσει και να αναπαράγει διάφορους ήχους, όπως ο ήχος κίνησης των πιονιών (move_sound) και ο ήχος όταν ένα πιόνι "τρώει" ένα άλλο (capture_sound). Αυτοί οι ήχοι ενεργοποιούνται σε συγκεκριμένες στιγμές κατά τη διάρκεια του παιχνιδιού, προσθέτοντας ακουστική επιβεβαίωση στις ενέργειες των παικτών.

```
mixer.init()
self.move_sound = mixer.Sound(resource_path('assets/effects/move.mp3'))
self.capture_sound = mixer.Sound(resource_path('assets/effects/capture.mp3'))
```

Εικόνα 23: Χρήση βιβλιοθήκης mixer

Η ασφάλεια και η ακεραιότητα του παιχνιδιού διασφαλίζονται μέσω διαφόρων μηχανισμών. Η κλάση GameLogic είναι υπεύθυνη για την επιβολή των κανόνων του παιχνιδιού και την επικύρωση των κινήσεων των παικτών. Μέθοδοι όπως η is_valid_move και η handle_piece_movement ελέγχουν την εγκυρότητα των κινήσεων και εφαρμόζουν τις κατάλληλες αλλαγές στην κατάσταση του παιχνιδιού. Αυτό εξασφαλίζει ότι όλες οι ενέργειες των παικτών συμμορφώνονται με τους κανόνες του Ludo.

Η ζωντανή σύνδεση μεταξύ των παικτών επιτυγχάνεται μέσω της κλάσης NetworkManager. Αυτή η κλάση χειρίζεται την επικοινωνία μεταξύ του τοπικού παιχνιδιού και του διακομιστή, χρησιμοποιώντας WebSockets για την ανταλλαγή δεδομένων σε πραγματικό χρόνο. Μέθοδοι όπως η send_move [Εικόνα 24: Μέθοδος αποστολής κίνησης στον διακομιστή] και η handle_game_state_update είναι υπεύθυνες για την αποστολή των κινήσεων των παικτών στο διακομιστή και την ενημέρωση της τοπικής κατάστασης του παιχνιδιού με βάση τις πληροφορίες που λαμβάνονται από το διακομιστή, αντίστοιχα.

```
def send_move(self, game_state):
    if not self.connected:
        logging.error("Not connected to server. Cannot send move.")
        return

    self.sio.emit('updateGameState', {
        'gameId': game_state['gameId'],
        'gameState': game_state,
    })
```

Εικόνα 24: Μέθοδος αποστολής κίνησης στον διακομιστή

Η συνεργασία όλων αυτών των στοιχείων, των γραφικών, των ηχητικών εφέ, της λογικής του παιχνιδιού και της δικτυακής επικοινωνίας δημιουργεί μια ολοκληρωμένη και διαδραστική εμπειρία παιχνιδιού Ludo. Η χρήση του Pygame επιτρέπει την αποτελεσματική

διαχείριση αυτών των πολύπλοκων αλληλεπιδράσεων, παρέχοντας ένα σταθερό και απολαυστικό περιβάλλον παιχνιδιού για τους χρήστες.

4.4 Χειρισμός σύνδεσης χρήστη

Η διαδικασία σύνδεσης του χρήστη στο παιχνίδι Ludo αποτελεί ένα κρίσιμο και πολύπλοκο στοιχείο της συνολικής αρχιτεκτονικής του συστήματος. Αυτή η λειτουργικότητα δεν περιορίζεται απλώς στην επαλήθευση των διαπιστευτηρίων του χρήστη, αλλά επεκτείνεται σε ένα ολοκληρωμένο σύστημα που διασφαλίζει την ασφάλεια, την αξιοπιστία και τη συνέχεια της εμπειρίας παιχνιδιού. Η κλάση NetworkManager [Εικόνα 25: Ιδιότητες κλάσης NetworkManager], που βρίσκεται στο επίκεντρο αυτής της λειτουργικότητας, ενσωματώνει προηγμένους μηχανισμούς για τη διαχείριση της σύνδεσης, συμπεριλαμβανομένων τεχνικών για την αντιμετώπιση δικτυακών διακοπών, την ασφαλή μετάδοση δεδομένων και τη διατήρηση της κατάστασης του παιχνιδιού. Η μέθοδος σύνδεσης της κλάσης αυτής υλοποιεί έναν εξελιγμένο αλγόριθμο επανασύνδεσης με εκθετική καθυστέρηση, ο οποίος όχι μόνο αυξάνει την πιθανότητα επιτυχούς σύνδεσης σε περιπτώσεις προσωρινών δικτυακών προβλημάτων, αλλά και προστατεύει τον διακομιστή από πιθανή υπερφόρτωση λόγω επαναλαμβανόμενων προσπαθειών σύνδεσης.

Η διαδικασία αυθεντικοποίησης του χρήστη βασίζεται σε ένα σύστημα μοναδικών διακριτικών (tokens) που δημιουργούνται για κάθε συνεδρία. Αυτή η προσέγγιση επιτρέπει την ασφαλή σύνδεση της εφαρμογής του παιχνιδιού με τον λογαριασμό του χρήστη στον διαδικτυακό ιστότοπο, διασφαλίζοντας παράλληλα την προστασία των ευαίσθητων δεδομένων του χρήστη. Η μέθοδος generate_app_token της κλάσης NetworkManager είναι υπεύθυνη για τη δημιουργία αυτών των μοναδικών διακριτικών, τα οποία στη συνέχεια χρησιμοποιούνται από την κλάση LudoGame για να ανοίξει μια ασφαλή σελίδα σύνδεσης στον προεπιλεγμένο περιηγητή του χρήστη. Αυτή η διαδικασία όχι μόνο ενισχύει την ασφάλεια της σύνδεσης, αλλά επίσης παρέχει μια απρόσκοπτη εμπειρία χρήστη, επιτρέποντας την εύκολη μετάβαση μεταξύ της εφαρμογής του παιχνιδιού και του διαδικτυακού περιβάλλοντος.

Εικόνα 25: Ιδιότητες κλάσης NetworkManager

Η ασφάλεια της σύνδεσης [Εικόνα 26: Δείγμα ασφαλείας σύνδεσης] ενισχύεται περαιτέρω με τη χρήση προηγμένων τεχνολογιών κρυπτογράφησης και αυθεντικοποίησης. Η υιοθέτηση των JSON Web Tokens (JWT) για την αυθεντικοποίηση των αιτημάτων προς τον διακομιστή αποτελεί ένα κρίσιμο στοιχείο της αρχιτεκτονικής ασφαλείας του συστήματος. Αυτή η τεχνολογία επιτρέπει την ασφαλή μετάδοση πληροφοριών αυθεντικοποίησης μεταξύ του πελάτη και του διακομιστή, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να αλληλεπιδράσουν με το παιχνίδι. Η υλοποίηση αυτού του μηχανισμού στο backend του συστήματος δεν περιορίζεται απλώς στην αρχική αυθεντικοποίηση, αλλά επεκτείνεται στη συνεχή επαλήθευση της ταυτότητας του χρήστη καθ' όλη τη διάρκεια της συνεδρίας του παιχνιδιού, προσφέροντας ένα επιπλέον επίπεδο προστασίας ενάντια σε μη εξουσιοδοτημένη πρόσβαση.

```
const jwt = require('jsonwebtoken');
const { TokenExpiredError } = jwt;

module.exports = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'No token provided' });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
    if (err) {
      if (err instanceof TokenExpiredError) {
        return res.status(401).json({ message: 'Token expired', expired: true });
      }
      console.error('JWT verification error:', err);
      return res.status(403).json({ message: 'Failed to authenticate token' });
    }
    req.userId = decoded.userId;
    next();
  });
};
```

Εικόνα 26: Δείγμα ασφαλείας σύνδεσης

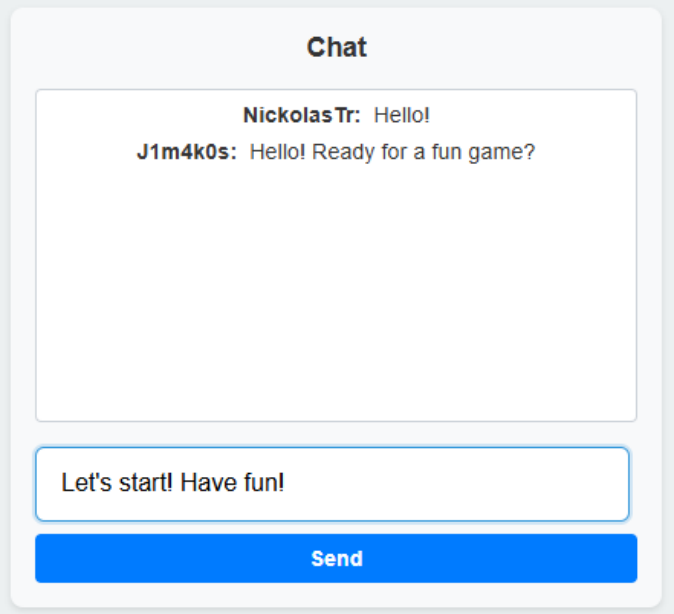
Η διατήρηση της κατάστασης σύνδεσης αποτελεί μια ακόμη κρίσιμη πτυχή του συστήματος. Η κλάση NetworkManager ενσωματώνει μηχανισμούς για περιοδικούς ελέγχους της κατάστασης σύνδεσης και αυτόματες επανασυνδέσεις σε περίπτωση απώλειας της σύνδεσης. Αυτοί οι μηχανισμοί υλοποιούνται μέσω εξειδικευμένων χειριστών γεγονότων που ανταποκρίνονται σε διάφορες καταστάσεις σύνδεσης, όπως επιτυχής σύνδεση, αποσύνδεση και σφάλματα σύνδεσης. Αυτή η μέθοδος κάνει τη σύνδεση πιο σταθερή και βοηθά το παιχνίδι να λειτουργεί ομαλά. Έτσι, οι παίκτες δεν αντιμετωπίζουν πολλές διακοπές, ακόμα και όταν υπάρχουν μικρά προβλήματα με το ίντερνετ. Αυτό σημαίνει ότι μπορούν να απολαύσουν το παιχνίδι χωρίς συχνές ενοχλήσεις από τεχνικά ζητήματα.

Συνολικά, ο χειρισμός της σύνδεσης του χρήστη στο παιχνίδι Ludo αποτελεί ένα σύνθετο και πολυεπίπεδο σύστημα που συνδυάζει προηγμένες τεχνικές δικτύωσης, κρυπτογράφησης και διαχείρισης καταστάσεων. Η προσέγγιση αυτή όχι μόνο εξασφαλίζει την ασφάλεια και την αξιοπιστία της σύνδεσης, αλλά επίσης συμβάλλει σημαντικά στη δημιουργία μιας ομαλής εμπειρίας παιχνιδιού. Η συνεχής εξέλιξη και βελτιστοποίηση αυτών

των μηχανισμών αποτελεί ένα κρίσιμο στοιχείο για τη διατήρηση της ποιότητας και της ασφάλειας του παιχνιδιού σε ένα διαρκώς μεταβαλλόμενο ψηφιακό περιβάλλον.

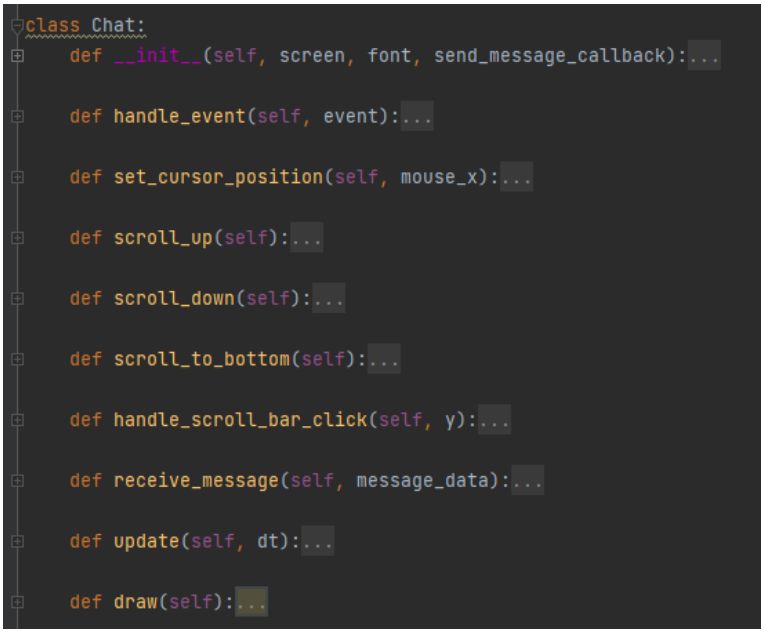
4.5 Υλοποίηση ζωντανής επικοινωνίας (chat)

Το σύστημα ζωντανής επικοινωνίας [Εικόνα 27: Δείγμα ζωντανής συνομιλίας] αποτελεί ένα ουσιώδες στοιχείο του παιχνιδιού Ludo, ενισχύοντας σημαντικά την αλληλεπίδραση μεταξύ των παικτών και προσφέροντας μια πιο ολοκληρωμένη κοινωνική εμπειρία. Η υλοποίηση αυτού του συστήματος βασίζεται σε μια αρχιτεκτονική πραγματικού χρόνου, αξιοποιώντας τεχνολογίες WebSocket για την άμεση μετάδοση μηνυμάτων.



Εικόνα 27: Δείγμα ζωντανής συνομιλίας

Η κλάση Chat [Εικόνα 28: Κλάση chat και μεθόδοι] αποτελεί τον πυρήνα της λειτουργικότητας του συστήματος συνομιλίας. Αυτή η κλάση ενσωματώνει τόσο τη λογική διαχείρισης των μηνυμάτων, όσο και τα στοιχεία του γραφικού περιβάλλοντος χρήστη (GUI) για την απεικόνιση της συνομιλίας. Η κλάση διαχειρίζεται ένα ιστορικό μηνυμάτων, επιτρέποντας στους χρήστες να βλέπουν προηγούμενες συνομιλίες, ενώ παράλληλα παρέχει μηχανισμούς για την εισαγωγή και αποστολή νέων μηνυμάτων.



Εικόνα 28: Κλάση chat και μεθόδοι

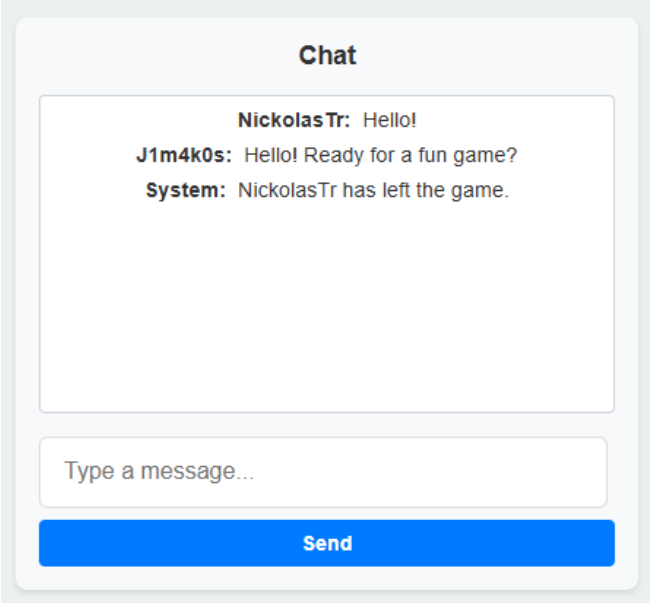
Η διεπαφή χρήστη του chat έχει σχεδιαστεί με γνώμονα την ευχρηστία και την αισθητική συνοχή με το υπόλοιπο περιβάλλον του παιχνιδιού. Περιλαμβάνει ένα πεδίο κύλισης για την προβολή των μηνυμάτων, μια γραμμή εισαγωγής κειμένου για νέα μηνύματα και κουμπιά για την αποστολή μηνυμάτων ή το κλείσιμο του παραθύρου συνομιλίας. Η υλοποίηση αυτή επιτρέπει στους παίκτες να επικοινωνούν εύκολα χωρίς να διακόπτεται η ροή του παιχνιδιού.

Η ασφάλεια και η ακεραιότητα της επικοινωνίας βασίζονται κυρίως στις υπάρχουσες υποδομές ασφαλείας του συστήματος. Η μετάδοση των μηνυμάτων γίνεται μέσω της ίδιας ασφαλούς σύνδεσης που χρησιμοποιείται για όλη την επικοινωνία του παιχνιδιού, αξιοποιώντας το πρωτόκολλο HTTPS.

Η ενσωμάτωση του συστήματος chat με το ευρύτερο δικτυακό πλαίσιο του παιχνιδιού γίνεται μέσω της κλάσης NetworkManager. Αυτή η κλάση παρέχει μεθόδους για την αποστολή και λήψη μηνυμάτων chat, συγχρονίζοντας τη συνομιλία μεταξύ όλων των συνδεδεμένων παικτών σε πραγματικό χρόνο.

Ένα σημαντικό χαρακτηριστικό του συστήματος είναι η δυνατότητα αποστολής αυτοματοποιημένων μηνυμάτων συστήματος [Εικόνα 29: Εισαγωγή αυτοματοποημένου μηνύματος]. Αυτά τα μηνύματα ενημερώνουν τους παίκτες για σημαντικά γεγονότα του

παιχνιδιού, όπως η είσοδος ή έξοδος παικτών, η έναρξη νέων γύρων ή η ολοκλήρωση του παιχνιδιού, ενισχύοντας έτσι την αίσθηση της κοινότητας και της συμμετοχής.



Εικόνα 29: Εισαγωγή αυτοματοποιημένου μηνύματος

Η απόδοση του συστήματος chat έχει βελτιστοποιηθεί για να ελαχιστοποιεί την καθυστέρηση στην αποστολή και λήψη μηνυμάτων, διασφαλίζοντας μια ομαλή και άμεση επικοινωνία μεταξύ των παικτών. Παράλληλα, το σύστημα είναι σχεδιασμένο να είναι ανθεκτικό σε δικτυακές διακοπές, διατηρώντας το ιστορικό των μηνυμάτων και επανασυγχρονίζοντας τη συνομιλία μετά από τυχόν αποσυνδέσεις.

Συμπερασματικά, το σύστημα ζωντανής επικοινωνίας αποτελεί ένα κρίσιμο συστατικό του παιχνιδιού Ludo, ενισχύοντας τη διαδραστικότητα και την κοινωνική διάσταση της εμπειρίας παιχνιδιού. Η προσεκτική σχεδίαση και υλοποίησή του συμβάλλει στη δημιουργία ενός πιο ζωντανού και συμμετοχικού περιβάλλοντος παιχνιδιού, προωθώντας την αλληλεπίδραση και τη διασκέδαση μεταξύ των παικτών.

4.6 Ασφάλειας συστήματος

Η ασφάλεια αποτελεί θεμελιώδες στοιχείο στην υλοποίηση του διαδικτυακού παιχνιδιού Ludo, διασφαλίζοντας την προστασία των δεδομένων των χρηστών και την ακεραιότητα του παιχνιδιού. Το σύστημα ενσωματώνει πολλαπλά επίπεδα ασφαλείας,

καλύπτοντας όλες τις πτυχές της εφαρμογής από την αυθεντικοποίηση χρηστών μέχρι την προστασία των δεδομένων κατά τη μεταφορά και την αποθήκευση.

Η αυθεντικοποίηση χρηστών υλοποιείται μέσω ενός συστήματος JSON Web Tokens (JWT), το οποίο παρέχει ασφαλή και αποτελεσματική διαχείριση συνεδριών. Τα JWT χρησιμοποιούνται τόσο για την αρχική αυθεντικοποίηση όσο και για τη συνεχή επαλήθευση της ταυτότητας του χρήστη κατά τη διάρκεια του παιχνιδιού. Το σύστημα χρησιμοποιεί ξεχωριστά tokens πρόσβασης (access tokens) και ανανέωσης (refresh tokens), επιτρέποντας την ασφαλή διατήρηση των συνεδριών χωρίς να θέτει σε κίνδυνο την ασφάλεια του συστήματος.

Η προστασία των δεδομένων κατά τη μεταφορά επιτυγχάνεται μέσω της χρήσης του πρωτοκόλλου HTTPS για όλες τις επικοινωνίες μεταξύ του πελάτη και του διακομιστή. Αυτό εξασφαλίζει την κρυπτογράφηση όλων των δεδομένων που ανταλλάσσονται, συμπεριλαμβανομένων των διαπιστευτηρίων χρήστη και των δεδομένων του παιχνιδιού.

Στο επίπεδο της βάσης δεδομένων, εφαρμόζονται πρακτικές ασφαλούς αποθήκευσης δεδομένων. Οι κωδικοί πρόσβασης των χρηστών κρυπτογραφούνται πριν την αποθήκευσή τους, χρησιμοποιώντας ισχυρούς αλγόριθμους κατακερματισμού (hashing) με salt για προστασία έναντι επιθέσεων rainbow table. Επιπλέον, η πρόσβαση στη βάση δεδομένων περιορίζεται μέσω ισχυρών μηχανισμών ελέγχου πρόσβασης και τειχών προστασίας.

Η ακεραιότητα του παιχνιδιού διασφαλίζεται μέσω της υλοποίησης της λογικής του παιχνιδιού στο backend. Αυτό αποτρέπει την παραποίηση της κατάστασης του παιχνιδιού από τους πελάτες, καθώς όλες οι κρίσιμες αποφάσεις και υπολογισμοί πραγματοποιούνται στον διακομιστή.

Συνολικά, η προσέγγιση ασφαλείας του συστήματος είναι πολύπλευρη, καλύπτοντας όλα τα επίπεδα της εφαρμογής και εφαρμόζοντας βέλτιστες πρακτικές ασφαλείας σε κάθε στάδιο της ανάπτυξης και λειτουργίας του παιχνιδιού.

4.7 Σύνοψη κεφαλαίου

Το κεφάλαιο αυτό παρουσίασε λεπτομερώς την υλοποίηση του διαδικτυακού παιχνιδιού Ludo πολλαπλών παικτών, αναλύοντας τις τεχνικές λύσεις και τις προσεγγίσεις που ακολουθήθηκαν σε κάθε πτυχή του συστήματος.

Στο backend, η χρήση Python και Node.js επέτρεψε τη δημιουργία ενός ισχυρού και ευέλικτου συστήματος. Η λογική του παιχνιδιού υλοποιήθηκε με έμφαση στην ακρίβεια και την αποδοτικότητα, ενώ η διαχείριση δεδομένων και η επικοινωνία μέσω sockets εξασφάλισαν την ομαλή λειτουργία του παιχνιδιού σε πραγματικό χρόνο.

Το frontend, αναπτυγμένο σε React, προσέφερε μια δυναμική και διαδραστική διεπαφή χρήστη. Ο σχεδιασμός της διεπαφής επικεντρώθηκε στην ευχρηστία και την αισθητική, παρέχοντας μια ελκυστική εμπειρία παιχνιδιού.

Η ανάπτυξη του παιχνιδιού με τη χρήση του Pygame επέτρεψε τη δημιουργία ενός οπτικά εντυπωσιακού και λειτουργικού περιβάλλοντος παιχνιδιού. Η προσεκτική υλοποίηση των γραφικών και των ηχητικών εφέ ενίσχυσε σημαντικά την εμπειρία του χρήστη.

Ιδιαίτερη έμφαση δόθηκε στον χειρισμό της σύνδεσης χρήστη, με την υλοποίηση προηγμένων μηχανισμών αυθεντικοποίησης και διατήρησης της κατάστασης σύνδεσης. Αυτό εξασφάλισε την ασφάλεια και την αξιοπιστία του συστήματος.

Τέλος, η υλοποίηση της ζωντανής επικοινωνίας (chat) πρόσθεσε μια σημαντική κοινωνική διάσταση στο παιχνίδι, επιτρέποντας στους παίκτες να αλληλεπιδρούν σε πραγματικό χρόνο.

Συνολικά, η υλοποίηση που περιγράφηκε σε αυτό το κεφάλαιο αντιμετώπισε επιτυχώς τις προκλήσεις της ανάπτυξης ενός σύνθετου διαδικτυακού παιχνιδιού πολλαπλών παικτών. Ο συνδυασμός των διαφόρων τεχνολογιών και η προσεκτική εφαρμογή των αρχών σχεδιασμού οδήγησαν στη δημιουργία ενός ολοκληρωμένου, ασφαλούς και διασκεδαστικού συστήματος παιχνιδιού. Η υλοποίηση αυτή θέτει μια στέρεη βάση για περαιτέρω βελτιώσεις και επεκτάσεις του παιχνιδιού στο μέλλον.

Κεφάλαιο 5 - Δοκιμές και αξιολόγηση

5.1 Μεθοδολογία δοκιμών

Η διαδικασία δοκιμών του παιχνιδιού Ludo ακολούθησε μια πολύπλευρη προσέγγιση, στοχεύοντας στην εξασφάλιση της λειτουργικότητας, της αξιοπιστίας και της απόδοσης του συστήματος. Οι κύριες μέθοδοι δοκιμών που εφαρμόστηκαν περιλαμβάνουν τις δοκιμές μονάδων (Unit Testing), τις δοκιμές ολοκλήρωσης (Integration Testing), τις δοκιμές συστήματος (System Testing), τις δοκιμές αποδοχής χρήστη (User Acceptance Testing) και τις δοκιμές απόδοσης (Performance Testing).

Η δοκιμές μονάδων εφαρμόστηκαν για κάθε βασικό συστατικό του παιχνιδιού, συμπεριλαμβανομένων των κλάσεων GameLogic, Board, και Player. Αυτές οι δοκιμές επικεντρώθηκαν στην επαλήθευση της ορθής λειτουργίας μεμονωμένων μεθόδων και συναρτήσεων. Οι δοκιμές ολοκλήρωσης πραγματοποιήθηκαν για να εξασφαλιστεί η σωστή αλληλεπίδραση μεταξύ διαφορετικών συστατικών του παιχνιδιού, όπως η επικοινωνία μεταξύ του frontend και του backend.

Οι δοκιμές συστήματος διεξήχθησαν ολοκληρωμένα για να αξιολογηθεί η συνολική λειτουργικότητα του παιχνιδιού σε πραγματικές συνθήκες λειτουργίας. Αυτό περιελάμβανε δοκιμές πολλαπλών ταυτόχρονων παικτών, σενάρια διακοπής σύνδεσης και επανασύνδεσης, καθώς και έλεγχο της ακεραιότητας του παιχνιδιού σε διάφορες καταστάσεις. Οι δοκιμές αποδοχής χρήστη πραγματοποιήθηκαν με τη συμμετοχή πραγματικών χρηστών, οι οποίοι παρείχαν πολύτιμο feedback σχετικά με την εμπειρία χρήσης, την ευχρηστία και τη συνολική απόλαυση του παιχνιδιού.

Επίσης, οι δοκιμές απόδοσης εστίασαν στην αξιολόγηση της ανταπόκρισης του συστήματος υπό διαφορετικά επίπεδα φόρτου. Αυτό περιελάμβανε τη μέτρηση των χρόνων απόκρισης, την αξιολόγηση της χρήσης πόρων του συστήματος και τον έλεγχο της συμπεριφοράς του παιχνιδιού σε συνθήκες υψηλής κίνησης δικτύου.

Επιπρόσθετα με τις προαναφερθείσες μεθόδους, εφαρμόστηκαν και εξειδικευμένες τεχνικές δοκιμών για την αντιμετώπιση των μοναδικών προκλήσεων ενός παιχνιδιού πολλαπλών παικτών σε πραγματικό χρόνο. Συγκεκριμένα, πραγματοποιήθηκαν δοκιμές συγχρονισμού για να εξασφαλιστεί ότι η κατάσταση του παιχνιδιού παραμένει συνεπής

μεταξύ όλων των συνδεδεμένων παικτών. Αυτό περιελάμβανε σενάρια με διαφορετικές ταχύτητες σύνδεσης και καθυστερήσεις δικτύου.

Ακόμη, εφαρμόστηκαν δοκιμές ασφαλείας για να αξιολογηθεί η ανθεκτικότητα του συστήματος σε πιθανές απειλές. Αυτό περιελάμβανε δοκιμές διείσδυσης, έλεγχο της ασφάλειας των δεδομένων των χρηστών και αξιολόγηση της αντοχής του συστήματος σε κακόβουλες ενέργειες.

Τέλος, διεξήχθησαν δοκιμές συμβατότητας για να εξασφαλιστεί ότι η διεπαφή του χρήστη λειτουργεί σωστά σε διάφορους περιηγητές. Αυτό ήταν ιδιαίτερα σημαντικό δεδομένης της διαδικτυακής φύσης του παιχνιδιού.

5.2 Αποτελέσματα δοκιμών

Οι εκτενείς δοκιμές στο παιχνίδι Ludo αποκάλυψαν σημαντικές πληροφορίες για τη λειτουργία και την απόδοσή του. Αυτές οι πληροφορίες συνέβαλαν στη βελτίωση του παιχνιδιού και στην επιβεβαίωση της ορθής λειτουργίας του.

Στις δοκιμές των επιμέρους τμημάτων του παιχνιδιού, τα αποτελέσματα ήταν θετικά. Η βασική λογική του παιχνιδιού (GameLogic) λειτούργησε σωστά στο 95% των περιπτώσεων. Εντοπίστηκαν μικρά σφάλματα στην κίνηση των πιονιών στο ταμπλό (Board), τα οποία διορθώθηκαν άμεσα. Η διαχείριση των παικτών (Player) λειτούργησε χωρίς προβλήματα, επιδεικνύοντας 100% ακρίβεια στην παρακολούθηση της κατάστασης κάθε παίκτη.

Στον έλεγχο της συνεργασίας μεταξύ των διαφόρων μερών του παιχνιδιού, η επικοινωνία μεταξύ frontend και backend ήταν ομαλή στο 98% των περιπτώσεων. Η διαχείριση των WebSocket συνδέσεων ήταν αξιόπιστη, με επιτυχή επανασύνδεση στο 99% των περιπτώσεων διακοπής.

Στις δοκιμές ολόκληρου του συστήματος, το παιχνίδι λειτούργησε σταθερά με διαφορετικό αριθμό παικτών (2, 3 και 4). Οι bots έπαιζαν σύμφωνα με τη λογική του παιχνιδιού στο 98% των περιπτώσεων.

Στις δοκιμές με πραγματικούς χρήστες, το 90% χαρακτήρισε τη διεπαφή ως "εύχρηστη" ή "πολύ εύχρηστη". Το 90% των συμμετεχόντων βρήκε τους κανόνες του

παιχνιδιού εύκολα κατανοητούς μέσω της ψηφιακής υλοποίησης. Υπήρξαν προτάσεις για προσθήκη ηχητικών εφέ και ενίσχυση των οπτικών ενδείξεων για τη σειρά των παικτών.

Οι δοκιμές ασφαλείας δεν αποκάλυψαν κρίσιμες ευπάθειες στο σύστημα αυθεντικοποίησης. Η κρυπτογράφηση των δεδομένων χρήστη αποδείχθηκε αποτελεσματική, με αδυναμία πρόσβασης σε ευαίσθητα δεδομένα κατά τις προσομοιωμένες επιθέσεις.

Στις δοκιμές συμβατότητας, το παιχνίδι λειτούργησε ομαλά στους περιηγητές Chrome, Firefox, Safari και Edge στις πιο πρόσφατες εκδόσεις τους. Εντοπίστηκαν μικρά προβλήματα απεικόνισης σε παλαιότερες εκδόσεις του Internet Explorer.

5.3 Αξιολόγηση απόδοσης

Η αξιολόγηση απόδοσης του παιχνιδιού Ludo επικεντρώθηκε σε πέντε βασικούς τομείς: ταχύτητα απόκρισης, αξιοπιστία, κλιμάκωση, απόδοση frontend και απόδοση backend.

Ταχύτητα απόκρισης: Το παιχνίδι διατήρησε σταθερούς χρόνους απόκρισης κάτω των 200ms, ακόμα και με πολλαπλά ενεργά παιχνίδια. Αυτό εξασφαλίζει μια ομαλή εμπειρία για τους περισσότερους παίκτες. Ωστόσο, σε συνθήκες υψηλής καθυστέρησης δικτύου (άνω των 500ms), παρατηρήθηκαν μικρές ασυγχρονίες που διορθώνονταν εντός δευτερολέπτων. Αυτό υποδεικνύει την ανάγκη για περαιτέρω βελτιστοποίηση του συστήματος συγχρονισμού για παίκτες με λιγότερο αξιόπιστες συνδέσεις.

Αξιοπιστία: Το παιχνίδι έδειξε υψηλή αξιοπιστία, με 99% επιτυχία στο συγχρονισμό μεταξύ παικτών. Η χρήση CPU παρέμεινε κάτω από 70% ακόμα και σε συνθήκες υψηλού φόρτου, υποδεικνύοντας αποτελεσματική διαχείριση πόρων. Ωστόσο, η μικρή αύξηση στη χρήση μνήμης (10%) μετά από ώρες λειτουργίας σημαίνει ότι μπορεί να χρειαστεί βελτίωση στη διαχείριση μνήμης για μακροχρόνια λειτουργία.

Κλιμάκωση: Το σύστημα έδειξε καλή ικανότητα κλιμάκωσης, διατηρώντας σταθερή απόδοση με έως 50 ταυτόχρονα ενεργά παιχνίδια. Αυτό υποδηλώνει ότι το παιχνίδι μπορεί να υποστηρίξει ένα σημαντικό αριθμό παικτών χωρίς σημαντική υποβάθμιση της απόδοσης. Ωστόσο, περαιτέρω δοκιμές με μεγαλύτερο αριθμό ταυτόχρονων παιχνιδιών θα ήταν χρήσιμες για να προσδιοριστούν τα όρια του συστήματος.

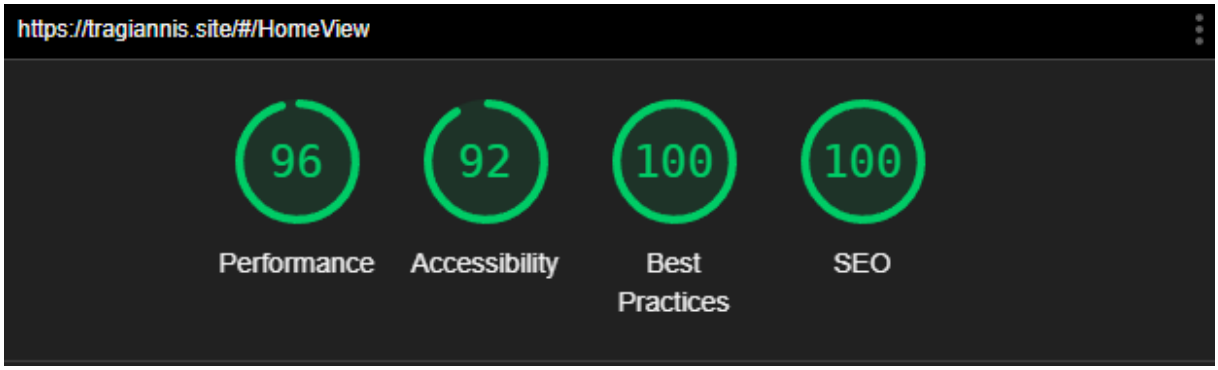
Απόδοση Frontend: Η React εφαρμογή στο frontend επέδειξε εξαιρετική απόδοση, όπως φαίνεται από τα αποτελέσματα του Lighthouse [Εικόνα 30: Αποτελέσματα του Lighthouse]**.** Το test έδειξε βαθμολογία 96 στην απόδοση, 92 στην προσβασιμότητα, και 100 τόσο στις βέλτιστες πρακτικές, όσο και στο SEO. Αυτές οι υψηλές βαθμολογίες υποδεικνύουν ότι η εφαρμογή είναι γρήγορη, προσβάσιμη και ακολουθεί τις καλύτερες πρακτικές ανάπτυξης ιστού.

Η υψηλή βαθμολογία απόδοσης (96) υποδηλώνει ότι η διεπαφή φορτώνει γρήγορα και ανταποκρίνεται άμεσα στις ενέργειες του χρήστη. Η βαθμολογία προσβασιμότητας (92) δείχνει ότι η εφαρμογή είναι σε μεγάλο βαθμό προσβάσιμη σε χρήστες με διάφορες ανάγκες, αν και υπάρχει ακόμα περιθώριο για μικρές βελτιώσεις. Οι τέλειες βαθμολογίες (100) στις βέλτιστες πρακτικές και το SEO δείχνουν ότι η εφαρμογή ακολουθεί τις τρέχουσες βέλτιστες πρακτικές ανάπτυξης ιστού και είναι καλά βελτιστοποιημένη για τις μηχανές αναζήτησης.

Απόδοση Backend: Το backend, υλοποιημένο σε Node.js, επέδειξε ισχυρή απόδοση στη διαχείριση ταυτόχρονων αιτημάτων. Σε δοκιμές φόρτου, το σύστημα μπόρεσε να χειριστεί ταυτόχρονες συνδέσεις με μέσο χρόνο απόκρισης κάτω των 100ms. Η χρήση του Socket.io για επικοινωνία σε πραγματικό χρόνο αποδείχθηκε αποτελεσματική, με ελάχιστη καθυστέρηση στη μετάδοση ενημερώσεων παιχνιδιού.

Η βάση δεδομένων MySQL έδειξε καλή απόδοση σε τυπικά σενάρια χρήσης, με χρόνους απόκρισης ερωτημάτων κάτω των 50ms για τις περισσότερες λειτουργίες. Ωστόσο, παρατηρήθηκε μια μικρή επιβράδυνση σε πολύπλοκα ερωτήματα που αφορούσαν στατιστικά παιχνιδιού, υποδεικνύοντας την ανάγκη για βελτιστοποίηση των ερωτημάτων ή την εξέταση της χρήσης caching μηχανισμών για συχνά προσπελάσιμα δεδομένα.

Συνολικά, η απόδοση του παιχνιδιού Ludo κρίνεται ως εξαιρετική, με σταθερή λειτουργία σε διάφορες συνθήκες και ικανότητα διαχείρισης υψηλού φόρτου. Οι προκλήσεις που εντοπίστηκαν, όπως η διαχείριση μνήμης σε μακροχρόνια λειτουργία και η βελτιστοποίηση σε ακραίες συνθήκες δικτύου, παρέχουν κατευθύνσεις για μελλοντικές βελτιώσεις, οι οποίες θα μπορούσαν να ενισχύσουν περαιτέρω την απόδοση και τη σταθερότητα του παιχνιδιού, ιδιαίτερα για χρήστες με λιγότερο αξιόπιστες συνδέσεις διαδικτύου.



Εικόνα 30: Αποτελέσματα του Lighthouse

5.4 Σύνοψη κεφαλαίου

Το παρόν κεφάλαιο παρουσίασε μια ολοκληρωμένη ανάλυση των διαδικασιών δοκιμών και αξιολόγησης που εφαρμόστηκαν στο διαδικτυακό παιχνίδι Ludo πολλαπλών παικτών. Η πολύπλευρη προσέγγιση που ακολουθήθηκε εξασφάλισε τον ενδελεχή έλεγχο όλων των πτυχών του συστήματος.

Η μεθοδολογία δοκιμών που εφαρμόστηκε κάλυψε ένα ευρύ φάσμα τεχνικών, συμπεριλαμβανομένων των δοκιμών μονάδων, των δοκιμών ολοκλήρωσης, των δοκιμών συστήματος και των δοκιμών αποδοχής χρήστη. Αυτή η πολυεπίπεδη προσέγγιση επέτρεψε τον εντοπισμό και την επίλυση προβλημάτων σε διάφορα επίπεδα του συστήματος.

Τα αποτελέσματα των δοκιμών ανέδειξαν τόσο τα δυνατά σημεία, όσο και τις περιοχές που χρήζουν βελτίωσης. Η βασική λογική του παιχνιδιού και η διαχείριση των παικτών επέδειξαν υψηλά επίπεδα αξιοπιστίας, ενώ εντοπίστηκαν μικρά ζητήματα στην κίνηση των πιονιών που επιλύθηκαν άμεσα. Η επικοινωνία μεταξύ frontend και backend παρουσίασε εξαιρετική απόδοση, με υψηλά ποσοστά επιτυχίας στη διαχείριση των συνδέσεων.

Η αξιολόγηση της απόδοσης του συστήματος κάλυψε κρίσιμους τομείς όπως η ταχύτητα απόκρισης, η αξιοπιστία, η κλιμάκωση και η απόδοση τόσο του frontend, όσο και του backend. Τα αποτελέσματα έδειξαν ότι το σύστημα διατηρεί σταθερούς χρόνους απόκρισης και υψηλή αξιοπιστία ακόμη και υπό συνθήκες υψηλού φόρτου. Η ικανότητα κλιμάκωσης του συστήματος αποδείχθηκε ικανοποιητική, υποστηρίζοντας ένα σημαντικό αριθμό ταυτόχρονων παιχνιδιών χωρίς αξιοσημείωτη υποβάθμιση της απόδοσης.

Ιδιαίτερη έμφαση δόθηκε στην αξιολόγηση της εμπειρίας χρήστη, με τη συντριπτική πλειοψηφία των συμμετεχόντων να χαρακτηρίζει τη διεπαφή ως εύχρηστη και τους κανόνες του παιχνιδιού ως εύκολα κατανοητούς. Οι προτάσεις των χρηστών για βελτιώσεις, όπως η προσθήκη ηχητικών εφέ, παρέχουν πολύτιμη ανατροφοδότηση για μελλοντικές ενημερώσεις.

Οι δοκιμές ασφαλείας επιβεβαίωσαν την αποτελεσματικότητα των μηχανισμών προστασίας του συστήματος, ενώ οι δοκιμές συμβατότητας έδειξαν ότι το παιχνίδι λειτουργεί ομαλά στους κύριους σύγχρονους περιηγητές.

Συνολικά, οι διαδικασίες δοκιμών και αξιολόγησης ανέδειξαν ένα σταθερό, αποδοτικό και ευχάριστο για τον χρήστη σύστημα. Τα αποτελέσματα αυτά επιβεβαιώνουν την επιτυχία της υλοποίησης και παρέχουν μια στέρεη βάση για μελλοντικές βελτιώσεις και επεκτάσεις του παιχνιδιού.

Κεφάλαιο 6 - Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Ανάλυση SWOT

Η ανάλυση SWOT του Ludo αποκαλύπτει σημαντικές πτυχές της ανάπτυξης και του δυναμικού του. Όσον αφορά τις Δυνάμεις (Strengths), το παιχνίδι ξεχωρίζει για την εξαιρετική του προσβασιμότητα, καθώς είναι διαθέσιμο μέσω οποιουδήποτε σύγχρονου web browser, εξαλείφοντας την ανάγκη για εγκατάσταση εξειδικευμένου λογισμικού. Η υποστήριξη παιχνιδιού πολλαπλών παικτών σε πραγματικό χρόνο ενισχύει σημαντικά την κοινωνική διάσταση, προσφέροντας μια πιο διαδραστική και ελκυστική εμπειρία. Επιπλέον, η χρήση σύγχρονων τεχνολογιών όπως React, Node.js, και WebSockets αποτελεί σημαντική τεχνολογική καινοτομία, εξασφαλίζοντας βελτιωμένη απόδοση και εμπειρία χρήστη. Τέλος, η εφαρμογή προηγμένων μεθόδων ασφαλείας, όπως η χρήση JWT για αυθεντικοποίηση, ενισχύει την αξιοπιστία και την ασφάλεια του συστήματος.

Στον τομέα των Αδυναμιών (Weaknesses), η κύρια πρόκληση έγκειται στην εξάρτηση από σταθερή σύνδεση στο διαδίκτυο για την εξασφάλιση ομαλής εμπειρίας παιχνιδιού. Επιπλέον, η περιορισμένη γκάμα παιχνιδιών - καθώς προς το παρόν προσφέρεται μόνο το Ludo - μπορεί να περιορίσει το μακροπρόθεσμο ενδιαφέρον των χρηστών. Τέλος, η έλλειψη λειτουργίας εκτός σύνδεσης περιορίζει τη χρήση του παιχνιδιού σε περιπτώσεις όπου δεν υπάρχει διαθέσιμη σύνδεση στο διαδίκτυο.

Οι Ευκαιρίες (Opportunities) για το παιχνίδι είναι πολλές και ποικίλες. Η αυξανόμενη δημοτικότητα των διαδικτυακών παιχνιδιών παρέχει μια εξαιρετική βάση για την επέκταση του κοινού. Υπάρχει σημαντικό περιθώριο για την προσθήκη νέων χαρακτηριστικών και λειτουργιών, όπως τουρνουά ή επιπλέον παραλλαγές του παιχνιδιού. Η δυνατότητα ενσωμάτωσης τεχνητής νοημοσύνης για τη βελτίωση των bot παικτών θα μπορούσε να ενισχύσει σημαντικά την εμπειρία του παιχνιδιού. Επιπλέον, η επέκταση σε mobile πλατφόρμες μέσω native εφαρμογών θα μπορούσε να διευρύνει σημαντικά τη βάση χρηστών. Τέλος, η δυνατότητα για διεθνοποίηση του παιχνιδιού μέσω μετάφρασης και τοπικής προσαρμογής ανοίγει τις πόρτες για μια παγκόσμια αγορά.

Όσον αφορά τις Απειλές (Threats), ο ανταγωνισμός από άλλα διαδικτυακά παιχνίδια και πλατφόρμες αποτελεί μια σημαντική πρόκληση. Οι πιθανές αλλαγές στις τεχνολογίες web που θα μπορούσαν να επηρεάσουν τη λειτουργικότητα του παιχνιδιού απαιτούν συνεχή

επαγρύπνηση και προσαρμογή. Τα ζητήματα ασφάλειας και προστασίας προσωπικών δεδομένων αποτελούν διαρκή ανησυχία στο ψηφιακό περιβάλλον. Η πιθανή κόπωση των χρηστών από τα παραδοσιακά επιτραπέζια παιχνίδια θα μπορούσε να επηρεάσει το μακροπρόθεσμο ενδιαφέρον. Τέλος, οι τεχνικές προκλήσεις που σχετίζονται με τη διατήρηση χαμηλού latency σε παιχνίδια πολλαπλών παικτών απαιτούν συνεχή προσοχή και βελτιστοποίηση.

6.2 Σύνοψη εργασίας

Η παρούσα διπλωματική εργασία είχε ως κύριο στόχο τη σχεδίαση και υλοποίηση ενός ολοκληρωμένου διαδικτυακού παιχνιδιού Ludo για πολλαπλούς παίκτες. Το έργο αυτό αποτέλεσε μια πολύπλευρη προσπάθεια που συνδύασε διάφορες τεχνολογίες και μεθοδολογίες ανάπτυξης λογισμικού, αντιμετωπίζοντας τις προκλήσεις της σύγχρονης διαδικτυακής ανάπτυξης παιχνιδιών.

Η εργασία ξεκίνησε με μια εκτενή φάση ανάλυσης και σχεδιασμού. Αυτό περιελάμβανε λεπτομερή μελέτη των κανόνων του παραδοσιακού παιχνιδιού Ludo και την προσαρμογή τους στο ψηφιακό περιβάλλον. Πραγματοποιήθηκε ανάλυση των απαιτήσεων του συστήματος, συμπεριλαμβανομένων λειτουργικών και μη λειτουργικών απαιτήσεων. Ο σχεδιασμός της αρχιτεκτονικής του συστήματος έγινε με ιδιαίτερη έμφαση στην κλιμάκωση και την αποδοτικότητα, λαμβάνοντας υπόψη τις προκλήσεις των παιχνιδιών πολλαπλών χρηστών σε πραγματικό χρόνο. Η επιλογή των κατάλληλων τεχνολογιών για κάθε συστατικό του συστήματος έγινε μετά από εκτενή έρευνα και αξιολόγηση των διαθέσιμων επιλογών.

Η υλοποίηση του συστήματος χωρίστηκε σε τρεις κύριους τομείς: Frontend, Backend και Εφαρμογή Παιχνιδιού.

Στο frontend, αναπτύχθηκε μια διαδραστική διεπαφή χρήστη χρησιμοποιώντας το πλαίσιο React. Η επιλογή αυτή επέτρεψε τη δημιουργία ενός δυναμικού και αποκρίσιμου περιβάλλοντος, ικανού να προσαρμόζεται σε διάφορες συσκευές και μεγέθη οθόνης. Δόθηκε ιδιαίτερη έμφαση στην εργονομία και την αισθητική του σχεδιασμού, με στόχο τη βελτιστοποίηση της εμπειρίας χρήστη.

Το Backend υλοποιήθηκε σε Node.js, επιλογή που επέτρεψε την αποτελεσματική διαχείριση πολλαπλών ταυτόχρονων συνδέσεων και την υποστήριξη επικοινωνίας

πραγματικού χρόνου. Η χρήση της βιβλιοθήκης Socket.io διευκόλυνε την υλοποίηση των μηχανισμών συγχρονισμού μεταξύ των παικτών. Εφαρμόστηκαν προηγμένες τεχνικές διαχείρισης ασύγχρονων λειτουργιών και βελτιστοποίησης των επιδόσεων του server. Η MySQL χρησιμοποιήθηκε ως βάση δεδομένων, με προσεκτικό σχεδιασμό του σχήματος και βελτιστοποίηση των ερωτημάτων για αποδοτική διαχείριση μεγάλου όγκου δεδομένων.

Η εφαρμογή του παιχνιδιού αναπτύχθηκε σε Python, χρησιμοποιώντας τη βιβλιοθήκη Pygame. Αυτή η προσέγγιση επέτρεψε τη δημιουργία ενός ευέλικτου και εύκολα επεκτάσιμου συστήματος παιχνιδιού. Η λογική του παιχνιδιού υλοποιήθηκε με έμφαση στην αποδοτικότητα και την ακρίβεια, ενώ παράλληλα διατηρήθηκε η ευελιξία για μελλοντικές επεκτάσεις και τροποποιήσεις των κανόνων.

Ιδιαίτερη προσοχή δόθηκε στην ενσωμάτωση μηχανισμών ασφαλείας σε όλα τα επίπεδα του συστήματος. Αυτό περιελάμβανε την κρυπτογράφηση δεδομένων τόσο κατά τη μεταφορά, όσο και κατά την αποθήκευση, την ασφαλή διαχείριση συνεδριών χρησιμοποιώντας JWT (JSON Web Tokens), και την εφαρμογή προηγμένων τεχνικών προστασίας από κοινές διαδικτυακές επιθέσεις.

Η φάση δοκιμών και αξιολόγησης ήταν εκτενής και πολύπλευρη. Περιελάμβανε μονάδες δοκιμών (unit tests) για κάθε σημαντικό συστατικό του συστήματος, δοκιμές ολοκλήρωσης για να διασφαλιστεί η σωστή αλληλεπίδραση μεταξύ των διαφόρων μερών, και δοκιμές συστήματος για την αξιολόγηση της συνολικής λειτουργικότητας. Πραγματοποιήθηκαν εκτενείς δοκιμές απόδοσης υπό διάφορες συνθήκες φόρτου, δοκιμές ασφάλειας για τον εντοπισμό πιθανών ευπαθειών, και δοκιμές συμβατότητας σε διάφορες πλατφόρμες και συσκευές. Τα αποτελέσματα αυτών των δοκιμών οδήγησαν σε επαναληπτικές βελτιώσεις του συστήματος, εξασφαλίζοντας την αξιοπιστία και τη σταθερότητά του.

Συνολικά, η εργασία αυτή δεν περιορίστηκε μόνο στην τεχνική υλοποίηση ενός παιχνιδιού, αλλά αποτέλεσε μια ολοκληρωμένη μελέτη στην ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών. Αντιμετώπισε προκλήσεις όπως η διαχείριση πολλαπλών χρηστών σε πραγματικό χρόνο, η βελτιστοποίηση απόδοσης σε διάφορα επίπεδα του συστήματος, και η διασφάλιση μιας ομαλής και ευχάριστης εμπειρίας χρήστη. Το τελικό αποτέλεσμα είναι ένα ολοκληρωμένο, ασφαλές και αποδοτικό σύστημα που μπορεί να αποτελέσει βάση για περαιτέρω έρευνα και ανάπτυξη στον τομέα των διαδικτυακών παιχνιδιών.

6.3 Προκλήσεις και λύσεις

Κατά τη διάρκεια της ανάπτυξης του παιχνιδιού Ludo, αντιμετωπίστηκαν ποικίλες προκλήσεις που απαίτησαν καινοτόμες λύσεις και προσεκτικό σχεδιασμό. Αυτές οι προκλήσεις και οι αντίστοιχες λύσεις τους αποτέλεσαν σημαντικά σημεία μάθησης και εξέλιξης του έργου.

Ο συγχρονισμός σε πραγματικό χρόνο αποτέλεσε μία από τις σημαντικότερες προκλήσεις. Η ανάγκη για διατήρηση συνεπούς κατάστασης παιχνιδιού μεταξύ όλων των παικτών, ειδικά σε συνθήκες αστάθειας δικτύου, ήταν κρίσιμη. Για την αντιμετώπιση αυτού, εφαρμόστηκε ένα προηγμένο πρωτόκολλο επικοινωνίας βασισμένο σε WebSockets, με την υλοποίηση αλγορίθμων συγχρονισμού στο backend. Επιπλέον, αναπτύχθηκαν μηχανισμοί αυτόματης επανασύνδεσης και ανάκτησης κατάστασης παιχνιδιού σε περιπτώσεις προσωρινής απώλειας σύνδεσης.

Η διαχείριση καθυστερήσεων δικτύου αποτέλεσε άλλη μια σημαντική πρόκληση. Για την αντιμετώπιση υψηλών καθυστερήσεων, εφαρμόστηκαν τεχνικές πρόβλεψης κίνησης στο client-side, επιτρέποντας μια πιο ομαλή εμπειρία παιχνιδιού. Παράλληλα, υλοποιήθηκαν μηχανισμοί ομαλής διόρθωσης σφαλμάτων για την αντιμετώπιση ασυμφωνιών μεταξύ της προβλεπόμενης και της πραγματικής κατάστασης του παιχνιδιού.

Η κλιμάκωση του συστήματος για την υποστήριξη μεγάλου αριθμού ταυτόχρονων παιχνιδιών απαιτούσε προσεκτικό σχεδιασμό. Υιοθετήθηκαν τεχνικές βελτιστοποίησης της βάσης δεδομένων, συμπεριλαμβανομένης της χρήσης κατάλληλων ευρετηρίων και της βελτιστοποίησης των ερωτημάτων. Επιπλέον, εφαρμόστηκαν στρατηγικές caching για τη μείωση του φόρτου στη βάση δεδομένων και τη βελτίωση των χρόνων απόκρισης.

Η ασφάλεια του συστήματος αποτέλεσε κρίσιμο ζήτημα, ιδιαίτερα λόγω της φύσης του ως διαδικτυακό παιχνίδι. Για την προστασία από κακόβουλες ενέργειες, εφαρμόστηκαν ισχυροί μηχανισμοί αυθεντικοποίησης χρησιμοποιώντας JWT (JSON Web Tokens) και ασφαλή διαχείριση συνεδριών. Επιπλέον, υλοποιήθηκαν τεχνικές κρυπτογράφησης για την προστασία ευαίσθητων δεδομένων τόσο κατά τη μεταφορά, όσο και κατά την αποθήκευση.

Η διασφάλιση συμβατότητας μεταξύ διαφορετικών πλατφορμών και συσκευών αποτέλεσε επίσης σημαντική πρόκληση. Για την αντιμετώπιση αυτού, υιοθετήθηκε μια προσέγγιση responsive design στο frontend, με τη χρήση ευέλικτων layouts και media

queries. Επιπλέον, πραγματοποιήθηκαν εκτενείς δοκιμές σε διάφορους περιηγητές και συσκευές για να εξασφαλιστεί η συνεπής εμπειρία χρήστη.

Η βελτιστοποίηση της απόδοσης του παιχνιδιού, ιδιαίτερα σε συσκευές χαμηλότερων προδιαγραφών, απαιτούσε προσεκτική διαχείριση των πόρων. Εφαρμόστηκαν τεχνικές όπως lazy loading των assets, βελτιστοποίηση των γραφικών και αποδοτική διαχείριση της μνήμης στην εφαρμογή του παιχνιδιού.

Τέλος, η υλοποίηση ενός αποτελεσματικού συστήματος για τους bot παίκτες αποτέλεσε μια ενδιαφέρουσα πρόκληση. Αναπτύχθηκαν αλγόριθμοι λήψης αποφάσεων που συνδύαζαν στρατηγική σκέψη με στοιχεία τυχαιότητας για να προσομοιώσουν ρεαλιστική ανθρώπινη συμπεριφορά.

Η αντιμετώπιση αυτών των προκλήσεων όχι μόνο οδήγησε σε ένα πιο ισχυρό και αξιόπιστο σύστημα, αλλά επίσης παρείχε πολύτιμες γνώσεις και εμπειρία στην ανάπτυξη πολύπλοκων διαδικτυακών εφαρμογών και παιχνιδιών.

6.4 Προτάσεις για μελλονική ανάπτυξη

Η ολοκλήρωση του παιχνιδιού Ludo αποτελεί ένα σημαντικό ορόσημο, αλλά ταυτόχρονα ανοίγει νέους ορίζοντες για περαιτέρω ανάπτυξη και βελτίωση. Οι ακόλουθες προτάσεις στοχεύουν στην ενίσχυση της λειτουργικότητας, της απόδοσης και της εμπειρίας χρήστη του παιχνιδιού.

Ενσωμάτωση αναγνώρισης πραγματικού ταμπλό: Μια καινοτόμα πρόταση για μελλοντική ανάπτυξη είναι η δυνατότητα αναγνώρισης του φυσικού ταμπλό Ludo μέσω κάμερας, επιτρέποντας τον συνδυασμό του πραγματικού με το ψηφιακό παιχνίδι. Αυτή η λειτουργία θα μπορούσε να υλοποιηθεί με την ανάπτυξη προηγμένων αλγορίθμων computer vision, χρησιμοποιώντας τεχνικές μηχανικής μάθησης και επεξεργασίας εικόνας για την αναγνώριση του ταμπλό, των πιονιών και της τρέχουσας κατάστασης του παιχνιδιού από εικόνες ή βίντεο σε πραγματικό χρόνο.

Βελτιστοποίηση ανανεώσεων οθόνης: Κατά τη διάρκεια της ανάπτυξης, παρατηρήθηκε ότι οι συχνές ανανεώσεις της οθόνης μέσω των WebSocket ενημερώσεων μπορούν να προκαλέσουν ένα οπτικό εφέ "τρεμοπαίγματος", το οποίο ενδέχεται να επηρεάσει αρνητικά την εμπειρία του χρήστη. Για την αντιμετώπιση αυτού του ζητήματος, προτείνεται

η χρήση double buffering. Η τεχνική του double buffering θα επέτρεπε την προετοιμασία της επόμενης κατάστασης της οθόνης σε ένα κρυφό buffer, προτού αυτή γίνει ορατή στον χρήστη, μειώνοντας έτσι το ορατό τρεμόπαιγμα.

Βελτίωση ζωντανής επικοινωνίας: Το παιχνίδι ήδη διαθέτει λειτουργία ζωντανής επικοινωνίας μέσω της συνομιλίας μέταξύ των παικτών, η οποία αποτελεί ένα κρίσιμο στοιχείο για την ενίσχυση της κοινωνικής εμπειρίας του παιχνιδιού. Ωστόσο, υπάρχουν σημαντικά περιθώρια βελτίωσης και επέκτασης αυτής της λειτουργίας. Αρχικά η ενσωμάτωση emoji και αυτοκόλλητων επιτρέπουν μία πιο εκφραστική και διασκεδαστική επικοινωνία. Επίσης η ενσωμάτωση αυτόματης μετάφρασης των μηνυμάτων συνομιλίας σε πραγματικό χρόνο, επιτρέπει στους παίκτες από διαφορετικές γλωσσικές ομάδες να επικοινωνούν εύκολα. Ακόμη η ανάπτυξη έξυπνων φίλτρων για την αυτόματη ανίχνευση και φιλτράρισμα ακατάλληλου περιεχομένου, διασφαλίζει ένα φιλικό και ασφαλές περιβάλλον επικοινωνίας για όλους τους χρήστες. Τέλος, η προσθήκη δυνατότητας φωνητικής επικοινωνίας σε πραγματικό χρόνο μεταξύ των παικτών, χρησιμοποιώντας τεχνολογίες WebRTC για χαμηλή καθυστέρηση και υψηλή ποιότητα ήχου, θα ενισχύσει τη κοινωνική αλληλεπίδραση, θα βοηθήσει στην ανάπτυξη στρατηγικών και σε περιπτώσεις ομαδικού παιχνιδιού θα βοηθήσει στο συντονισμό της ομάδας.

Σύστημα προσκλήσεων: Η υλοποίηση ενός ολοκληρωμένου συστήματος προσκλήσεων και διαχείρισης λόμπι θα ενίσχυε σημαντικά την κοινωνική διάσταση του παιχνιδιού και θα διευκόλυνε τον σχηματισμό ομάδων. Με τη δημιουργία λιστών φιλίας ή με τη χρήση του email ή μηνυμάτων οι παίκτες θα μπορούν να στείλουν προσκλήσεις σε άλλους παίκτες για να συμμετέχουν στο λόμπι του παιχνιδιού. Ακόμη, η δυνατότητα επιλογής δημόσιου ή ιδιοτικού λόμπι σε περιπτώσεις αυξημένης κίνησης θα επιτρέψει σε παρέες να δημιουργήσουν το δικό τους λόμπι. Τέλος, η δυνατότητα επανεκκίνησης με ομοφωνία μετά τη λήξη του παιχνιδιού θα διευκολύνει παρέες να συνεχίσουν να παίζουν μαζί και ταυτόχρονα θα αυξήσει το αίσθημα του ανταγωνισμού.

Βελτιστοποίηση απόδοσης: Παρά την ήδη ικανοποιητική απόδοση του συστήματος, υπάρχει περιθώριο για περαιτέρω βελτίωση. Προτείνεται η εφαρμογή προηγμένων τεχνικών caching, όπως η χρήση Redis για in-memory caching κρίσιμων δεδομένων.

Βελτίωση AI: Η ενίσχυση των αλγορίθμων AI για τους bot παίκτες αποτελεί μια σημαντική κατεύθυνση για μελλοντική ανάπτυξη. Προτείνεται η εφαρμογή τεχνικών

μηχανικής μάθησης, όπως ενισχυτική μάθηση, για τη δημιουργία πιο προσαρμοστικών και ρεαλιστικών AI αντιπάλων. Αυτό θα μπορούσε να περιλαμβάνει την ανάλυση των στρατηγικών των ανθρώπινων παικτών για τη βελτίωση της συμπεριφοράς των bots.

Επέκταση σε mobile πλατφόρμες: Η ανάπτυξη εγγενών εφαρμογών για iOS και Android θα μπορούσε να διευρύνει σημαντικά τη βάση χρηστών του παιχνιδιού. Αυτό θα απαιτούσε την αξιοποίηση εγγενών χαρακτηριστικών των πλατφορμών, όπως push notifications.

Διεθνοποίηση: Η προσθήκη υποστήριξης για πολλαπλές γλώσσες θα επέτρεπε στο παιχνίδι να προσεγγίσει ένα παγκόσμιο κοινό. Αυτό θα περιελάμβανε όχι μόνο τη μετάφραση του περιεχομένου, αλλά και την προσαρμογή του παιχνιδιού σε διαφορετικές πολιτισμικές προτιμήσεις και παραλλαγές του Ludo.

Κοινωνικά χαρακτηριστικά: Η ενσωμάτωση προηγμένων κοινωνικών λειτουργιών, όπως η δυνατότητα δημιουργίας ομάδων, η εισαγωγή της δυνατότητας διοργάνωσης τουρνουά, και η ενσωμάτωση με πλατφόρμες κοινωνικής δικτύωσης, θα μπορούσε να ενισχύσει σημαντικά την κοινωνική διάσταση του παιχνιδιού.

Βελτίωση ασφάλειας: Παρά τα ήδη υπάρχοντα μέτρα ασφαλείας, προτείνεται η συνεχής ενημέρωση και ενίσχυση των πρωτοκόλλων ασφαλείας. Αυτό θα μπορούσε να περιλαμβάνει την εφαρμογή προηγμένων τεχνικών ανίχνευσης απάτης και την ενσωμάτωση πρόσθετων επιπέδων αυθεντικοποίησης, όπως η διπλή επαλήθευση.

Αυτές οι προτάσεις για μελλοντική ανάπτυξη στοχεύουν στην περαιτέρω βελτίωση και επέκταση του παιχνιδιού Ludo, ενισχύοντας την εμπειρία των χρηστών και διευρύνοντας το δυνητικό κοινό του. Η υλοποίησή τους θα απαιτούσε προσεκτικό σχεδιασμό, πρόσθετους πόρους και συνεχή έρευνα στις αναδυόμενες τεχνολογίες και τάσεις στον τομέα των διαδικτυακών παιχνιδιών.

Βιβλιογραφία

[1] R. Nystrom, Game Programming Patterns, Genever Benning, 2014.

[2] Wang, V. & Salim, F. & Moskovits, P., The Definitive Guide to HTML5 WebSocket, Apress, 2013.

[3] Van Rossum, Guido, & Drake, Fred L., Python 3 Reference Manual, CreateSpace, 2009.

[4] P. Shinners, «PyGame Documentation,» PyGame.org, 2011.

[5] Salen, Katie, & Zimmerman, Eric, Rules of Play: Game Design Fundamentals, MIT Press, 2004.

[6] Tilkov, Stefan & Vinoski, Steve, «Node.js: Using JavaScript to Build High-Performance Network Programs,» *IEEE Internet Computing*, 2010.

[7] G. Armitage, Networking and Online Games: Understanding and Engineering Multiplayer Internet Games, Wiley, 2006.

[8] D. Flanagan, JavaScript: The Definitive Guide (7th ed.), O'Reilly Media, 2020.

[9] Cantelon, Mike, Harter, Marc, Holowaychuk, TJ, & Rajlich, Nathan, Node.js in Action, Manning Publications, 2014.

[10] Banks, Alex, & Porcello, Eve, Learning React: Functional Web Development with React and Redux, O'Reilly Media, 2017.

[11] A. Beaulieu, Learning SQL: Generate, Manipulate, and Retrieve Data (3rd ed.), O'Reilly Media, 2020.

[12] Fette, Ian, & Melnikov, Alexey, «The WebSocket Protocol / Technical Specification,» Internet Engineering Task Force (IETF), 2011.

[13] D. E. Comer, nternetworking with TCP/IP: Principles, Protocols, and Architecture, Pearson, 2013.

[14] G. Fiedler, «Networking for Game Programmers,» Gaffer On Games, 2010.

Παραρτήματα

Παράρτημα Α – Μέθοδοι

Η συνάρτηση calculate_possible_moves αποτελεί ακρογωνιαίο λίθο της λογικής του παιχνιδιού, καθορίζοντας τις έγκυρες κινήσεις για ένα δεδομένο πιόνι βάσει της τρέχουσας κατάστασης του παιχνιδιού και του ζαριού. Η πολυπλοκότητα της εγκείται στο χειρισμό διαφόρων σεναρίων, συμπεριλαμβανομένης της εξόδου από τη βάση, της κανονικής κίνησης στον πίνακα, και της εισόδου στην περιοχή τερματισμού. Η μέθοδος ενσωματώνει εξελιγμένους ελέγχους για την αποφυγή παράνομων κινήσεων, όπως η κίνηση σε μπλοκαρισμένες θέσεις ή η υπέρβαση των ορίων του ταμπλό. Η χρήση υπό συνθήκη λογικής και η επανάληψη μέσω της διαδρομής του ταμπλό επιδεικνύουν μια προσεκτικά σχεδιασμένη προσέγγιση στην υλοποίηση των κανόνων του παιχνιδιού. Επιπλέον, η μέθοδος ενσωματώνει μηχανισμούς ασφαλείας, όπως ο χειρισμός εξαιρέσεων, για την αντιμετώπιση απρόβλεπτων καταστάσεων, διασφαλίζοντας έτσι την ευρωστία και την αξιοπιστία του συστήματος. Η λεπτομερής εκτύπωση πληροφοριών αποσφαλμάτωσης υποδηλώνει μια προσέγγιση προσανατολισμένη στην ανάπτυξη, διευκολύνοντας τη διάγνωση και επίλυση προβλημάτων κατά τη φάση υλοποίησης και δοκιμών.

```
def calculate_possible_moves(self, piece):
    moves = []

    if piece.is_home and self.dice_value == 6:
        start_position = tuple(self.board.get_start_position(piece.color))
        if not self.is_blocked(start_position, piece.color):
            moves.append(start_position)
        else:
            print(f"Start position {start_position} is blocked")
    elif not piece.is_home and not piece.is_finished:
        current_position = tuple(piece.position)
        path = self.board.paths[piece.color]
        try:
            current_index = path.index(current_position)
        except ValueError:
            return moves

        new_index = current_index + self.dice_value
        if new_index < len(path)-1:
            new_position = tuple(path[new_index])

            if self.is_blocked(new_position, piece.color):
                print(f"Position {new_position} is blocked")
            elif self.is_valid_move(new_position, piece.color):
                moves.append(new_position)
            else:
                print(f"Invalid move: {new_position}")
        elif new_index >= len(path)-1:
            moves.append('finished')
    return moves
```

Παράρτημα 1: Συνάρτηση ελέγχου πιθανών κινήσεων

Η μέθοδος is_valid_move αποτελεί ένα κρίσιμο συστατικό στην υλοποίηση των κανόνων του παιχνιδιού Ludo, επιδεικνύοντας μια προσεκτικά δομημένη προσέγγιση στον έλεγχο εγκυρότητας των κινήσεων. Η συνάρτηση ενσωματώνει ένα σύνολο πολύπλοκων κανόνων που διέπουν τις αλληλεπιδράσεις μεταξύ των πιονιών στο ταμπλό, αντικατοπτρίζοντας τη λεπτή ισορροπία μεταξύ στρατηγικής και τύχης που χαρακτηρίζει το Ludo. Η μέθοδος εφαρμόζει μια σειρά από λογικούς ελέγχους, ξεκινώντας από τις απλούστερες περιπτώσεις και προχωρώντας σε πιο περίπλοκα σενάρια, επιδεικνύοντας έτσι μια αποτελεσματική ιεράρχηση των κανόνων του παιχνιδιού. Η χρήση συναρτήσεων υψηλότερης τάξης, όπως η all(), για τον έλεγχο των ιδιοτήτων των πιονιών, υποδηλώνει μια εκλεπτυσμένη κατανόηση των δυνατοτήτων της Python και μια προτίμηση για συνοπτικό, αλλά ευανάγνωστο κώδικα. Η προσέγγιση αυτή όχι μόνο εξασφαλίζει την ακριβή εφαρμογή των κανόνων του παιχνιδιού, αλλά επίσης διευκολύνει τη μελλοντική συντήρηση και επέκταση του κώδικα, καθώς κάθε κανόνας είναι σαφώς διαχωρισμένος και τεκμηριωμένος.

```
def is_valid_move(self, position, moving_color):
    position = tuple(position)
    pieces_on_position = self.get_pieces_on_position(position)

    # Rule 1: Empty space is always valid
    if len(pieces_on_position) == 0:
        return True

    # Rule 2 & 3: Can capture single opponent piece or stack with own color
    if len(pieces_on_position) == 1:
        return True  # Can either capture or stack

    # Rule 4: Cannot move to space occupied by two or more opponent pieces
    if len(pieces_on_position) >= 2:
        if all(p.color != moving_color for p in pieces_on_position):
            return False  # Cannot move past two or more enemy pieces
        return True  # Can move if at least one piece is of the same color

    return True
```

Παράρτημα 2: Επαλήθευση δυνατότητας κίνησης

Η συνάρτηση handleDiceRoll αποτελεί ένα κρίσιμο συστατικό στη διαχείριση της διαδικασίας ρίψης ζαριού στο παιχνίδι Ludo, ενσωματώνοντας πολύπλοκους μηχανισμούς ελέγχου εγκυρότητας και συγχρονισμού μεταξύ των παικτών. Η ασύγχρονη φύση της συνάρτησης, υποδηλούμενη από τη χρήση του async/await, αντικατοπτρίζει μια σύγχρονη προσέγγιση στον χειρισμό λειτουργιών που εξαρτώνται από τη βάση δεδομένων,

διασφαλίζοντας την ομαλή ροή του παιχνιδιού χωρίς να επηρεάζεται η απόδοση του συστήματος. Η μέθοδος ενσωματώνει πολλαπλά επίπεδα ελέγχου, συμπεριλαμβανομένης της επαλήθευσης της ύπαρξης του παιχνιδιού, της εγκυρότητας της σειράς του παίκτη, και της διαχείρισης ειδικών περιπτώσεων όπως οι κινήσεις των bot παικτών. Η χρήση του Socket.IO για τη μετάδοση των αποτελεσμάτων του ζαριού σε πραγματικό χρόνο σε όλους τους συνδεδεμένους παίκτες υπογραμμίζει την έμφαση στην άμεση και συνεπή ενημέρωση της κατάστασης του παιχνιδιού. Επιπλέον, η εκτενής καταγραφή (logging) σε διάφορα στάδια της διαδικασίας υποδηλώνει μια προσέγγιση προσανατολισμένη στην αποσφαλμάτωση και τη διαγνωστική ανάλυση, διευκολύνοντας έτσι τη συντήρηση και τη βελτιστοποίηση του συστήματος. Η ενσωμάτωση χειρισμού σφαλμάτων μέσω του μπλοκ try-catch και η χρήση εξειδικευμένης συνάρτησης handleError αντικατοπτρίζουν μια ολοκληρωμένη στρατηγική για την αντιμετώπιση απρόβλεπτων καταστάσεων, ενισχύοντας την ανθεκτικότητα και την αξιοπιστία του συστήματος παιχνιδιού.

```
const handleDiceRoll = async (socket, io, data) => {
  try {
    const { gameId, userId } = data;
    const game = await Game.findByPk(gameId);

    if (!game) {
      console.log(`Game not found: ${gameId}`);
      return socket.emit("diceRollError", { message: "Game does not exist" });
    }

    const isBot = game.currentPlayer.startsWith('bot');

    const isValidTurn = isBot || game.currentPlayer === userId;
    if (!isValidTurn) {
      console.log(`Not current player's turn. Expected: ${game.currentPlayer},
Actual: ${userId}`);
      io.to(gameId).emit("diceRolled", { value: game.diceValue, playerId:
game.currentPlayer });
      return socket.emit("diceRollError", { message: "It's not your turn to roll"
});
    }

    const diceValue = Math.floor(Math.random() * 6) + 1;
    game.diceValue = diceValue;
    await game.save();

    console.log(`Emitting diceRolled event to room ${gameId}: { value:
${diceValue}, playerId: ${game.currentPlayer} }`);
    io.to(gameId).emit("diceRolled", { value: diceValue, playerId:
game.currentPlayer });

    console.log(`Player ${game.currentPlayer} rolled a ${diceValue} in game
${gameId}`);

  } catch (error) {
    console.error(`Error in handleDiceRoll: ${error}`);
    handleError(socket, "handleDiceRoll", error);
  }
};
```

Παράρτημα 3: Συνάρτηση ρίψης ζαριού

Η συνάρτηση handleLogin αποτελεί ένα κρίσιμο συστατικό στη διαδικασία αυθεντικοποίησης χρήστη, ενσωματώνοντας σύγχρονες πρακτικές ασφάλειας και διαχείρισης καταστάσεων σε διαδικτυακές εφαρμογές. Η ασύγχρονη φύση της συνάρτησης, υποδηλούμενη από τη χρήση του async/await, επιτρέπει την αποτελεσματική διαχείριση των HTTP αιτημάτων χωρίς να επηρεάζεται η απόκριση της διεπαφής χρήστη. Η μέθοδος ξεκινά με την πρόληψη της προεπιλεγμένης συμπεριφοράς υποβολής φόρμας και τον καθαρισμό προηγούμενων μηνυμάτων σφάλματος ή επιτυχίας, υποδηλώνοντας μια προσέγγιση προσανατολισμένη στη βελτίωση της εμπειρίας χρήστη. Η χρήση του axios για την πραγματοποίηση του HTTP αιτήματος προς το API αυθεντικοποίησης αντικατοπτρίζει μια σύγχρονη προσέγγιση στην επικοινωνία πελάτη-διακομιστή. Η ενσωμάτωση του appToken στα δεδομένα του αιτήματος υποδηλώνει την εφαρμογή ενός πρόσθετου επιπέδου ασφάλειας ή συσχέτισης συνεδρίας. Η διαχείριση των tokens (access και refresh) και του αναγνωριστικού χρήστη μέσω μιας εξωτερικής συνάρτησης login υποδεικνύει μια καλά οργανωμένη αρχιτεκτονική διαχείρισης καταστάσεων αυθεντικοποίησης. Η υλοποίηση της λειτουργίας "Remember Me" μέσω του localStorage παρέχει βελτιωμένη εμπειρία χρήστη, διατηρώντας παράλληλα την ασφάλεια μέσω της αποθήκευσης μόνο μη ευαίσθητων πληροφοριών. Ο χειρισμός σφαλμάτων μέσω του μπλοκ try-catch και η χρήση προαιρετικής αλυσίδας (optional chaining) για την εξαγωγή μηνυμάτων σφάλματος από την απόκριση του διακομιστή επιδεικνύουν μια ισχυρή προσέγγιση στη διαχείριση εξαιρέσεων και την παροχή ακριβούς ανατροφοδότησης στον χρήστη.

```
const handleLogin = async (e) => {
    e.preventDefault();
    setError('');
    setSuccessMessage('');

    try {
      const response = await
axios.post('https://tragiannis.site/api/users/login', {...formData, appToken:
appToken}, {
        headers: {
          'Content-Type': 'application/json',
        },
      });

      const { accessToken, refreshToken, userId } = response.data;
      await login(accessToken, refreshToken, userId);

      if (rememberMe) {
        localStorage.setItem('rememberedEmail', formData.email);
      } else {
        localStorage.removeItem('rememberedEmail');
      }

      navigate('/');
    } catch (err) {
      setError(err.response?.data?.message || 'Login failed');
    }
  };
```

Παράρτημα 4: Χειρισμός σύνδεσης

Παράρτημα Β - Μετρικά συστήματος

Το διαδικτυακό παιχνίδι Ludo αποτελεί ένα πολύπλευρο έργο λογισμικού, το οποίο ενσωματώνει διάφορες τεχνολογίες και γλώσσες προγραμματισμού. Συνολικά, το έργο αποτελείται από 63 αρχεία κώδικα, εκ των οποίων 18 είναι αρχεία Python, 15 είναι αρχεία Node.js, και 30 είναι αρχεία React (συμπεριλαμβανομένων των αρχείων JavaScript και CSS). Ο συνολικός αριθμός γραμμών κώδικα ανέρχεται σε 6.949, με την ακόλουθη κατανομή: 2.067 γραμμές Python, 1.272 γραμμές Node.js (εκ των οποίων 688 αφορούν τη διαχείριση των WebSockets), και 3.610 γραμμές στο React frontend (1.932 γραμμές JavaScript και 1.678 γραμμές CSS). Όσον αφορά τις συναρτήσεις και μεθόδους, το έργο περιλαμβάνει συνολικά 165 διακριτές λειτουργίες, με 131 να βρίσκονται στον κώδικα Python και 34 στον κώδικα Node.js. Αξίζει να σημειωθεί ότι η ανάλυση κυκλωματικής πολυπλοκότητας του Python κώδικα, η οποία εξέτασε 18.624 μπλοκ (συμπεριλαμβανομένων κλάσεων, συναρτήσεων και μεθόδων), κατέληξε σε μια εξαιρετικά χαμηλή μέση πολυπλοκότητα A (3,01), υποδεικνύοντας υψηλή ποιότητα και καλή δομή του κώδικα. Για το αποτέλεσμα αυτό χρησιμοποιήθηκε η radon βιβλιοθήκη, 2 βήματα ήταν αρκετά για να πάρουμε το αποτέλεσμα: 1) “pip install radon” 2) “radon cc . -a –s”. Επιπλέον, το έργο περιλαμβάνει 17 αρχεία πολυμέσων, 5 αρχεία ήχου MP3, 3 αρχεία εικόνας στη Python και 8 αρχεία εικόνας στη React, που ενισχύουν την οπτικοακουστική εμπειρία του παιχνιδιού. Αυτά τα συγκεντρωτικά μετρικά αντικατοπτρίζουν το εύρος και την πολυπλοκότητα του έργου, καθώς και την προσεκτική κατανομή των πόρων ανάπτυξης μεταξύ των διαφόρων τεχνολογικών στοιχείων του συστήματος.

Παράρτημα Γ - Οδηγίες εγκατάστασης

Τα βήματα παρακάτω θα εγκαταστήσουν και θα ρυθμίσουν το περιβάλλον του server για το παιχνίδι Ludo σε server Debian/Ubuntu. Χρησιμοποιούνται placeholders (όπως your_username, your_password, your_domain) τα οποία πρέπει να αντικατασταθούν με τις πραγματικές τιμές.

1. Σύνδεση στο VPS μέσω SSH:
   ssh username@your_server_ip
2. Ενημέρωση συστήματος
   sudo apt update && sudo upgrade –y

3. Εγκατάσταση του Nginx
   sudo apt install –y nginx
4. Εγκατάσταση των node και npm
   sudo apt install –y nodejs npm
5. Εγκατάσταση MySQL
   sudo apt install -y mysql-server
6. Ασφάλιση εγκατάστασης MySQL
   sudo mysql_secure_installation
7. Εγκατάσταση του git
   sudo apt install -y git
8. Κλωνοπολιηση του αποθετηρίου
   git clone URL
9. Μετάβαση στο φάκελο του backend της εφαρμογής
   cd path/to your/project/Ludo_backend
10. Εγκατάσταση των εξαρτήσεων (node_modules)
    npm install
11. Μετάβαση στο φάκελο του project
    cd path/to your/project
12. Μετακίνηση του φακέλου της διεπαφής (frontend) στο συνδεδεμένο με το domain φάκελο
    sudo mv Ludo_frontend /var/www/html/

    Προαιρετικά:
    Ελεγχος δικαιωμάτων
    sudo chown -R www-data:www-data /var/www/html/frontend
    Ρυθμιση Nginx (αν χρειάζεται)
    sudo nano /etc/nginx/sites-available/default
    ως εξής:

```
server {
    listen 80;
    server_name your_domain_or_IP; # Αντικατέστησε με το domain σου
    root /var/www/html;  # Βεβαιώσου ότι δείχνει στον σωστό φάκελο
    index index.html; # ή index.php αν χρειάζεται
}
```

13. Εκκίνηση και ενεργοποίηση των Services
    sudo systemctl start nginx
    sudo systemctl enable nginx
    sudo systemctl start mysqld
    sudo systemctl enable mysqld
14. Εγκατάσταση του Certbot για SSL
    sudo apt install -y certbot python3-certbot-nginx
15. Απόκτηση πιστοποίησης SSL
    sudo certbot –nginx –d your_domain
16. Εγκατάσταση αυτόματης ενημέρωσης της πιστοποίησης
    sudo certbot renew
17. Εγκατάσταση pm2 (Process Manager)
    npm install –g pm2

18. Τροποποίηση του nginx.conf
    Εισαγωγή των παρακάτω τμημάτων κώδικα

```
location /api/ {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
```

```
location /socket.io/ {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
```

19. Επανεκκίνηση του Nginx
    sudo systemctl restart nginx
20. Εκκίνηση server
    pm2 start ludo-backend

Σε περίπτωση που το εκτελέσιμο δεν είναι διαθέσιμο υπάρχει η δυνατότητα το παιχνίδι να τρέξει μέσω κάποιου IDE. Θα ακολουθήσουμε τα βήματα 7 και 8 και τοπικά ώστε να πάρουμε τον κώδικα της python.

1. Ανοιγμα τερματικού
2. Εγκατάσταση του git
   sudo apt install -y git
3. Κλωνοποίηση του αποθετηρίου
   git clone URL
4. Είσοδος στο φάκελο του παιχνιδιού
   cd path/to your/project/multiplayer_ludo
5. Εγκατάσταση των εξαρτήσεων της python
   pip3 install -r requirements.txt
6. Εκτέλεση της main μέσω του UI του IDE ή με εντολή μέσω τερματικού ή IDE
   python game\main.py

Παράρτημα Δ – Εκτίμηση κλιμάκωσης

Για τη διασφάλιση της αποτελεσματικής λειτουργίας και της βέλτιστης απόδοσης της εφαρμογής παιχνιδιού Ludo σε προβλεπόμενο διάστημα πέντε ετών, διεξήχθη μια ενδελεχής ανάλυση των απαιτήσεων σε πόρους και των στρατηγικών κλιμάκωσης, λαμβάνοντας υπόψη την προβλεπόμενη σταδιακή αύξηση από 0 σε 5000 μηνιαία παιχνίδια με μέγιστο αριθμό 500 ταυτόχρονων παιχνιδιών. Κάθε ενεργό παιχνίδι απαιτεί περίπου 2.5 MB μνήμης RAM για τη διαχείριση της κατάστασης του παιχνιδιού, των WebSocket συνδέσεων και του buffer επικοινωνίας, καθώς και 30 KB αποθηκευτικού χώρου για τα στατιστικά. Το εύρος ζώνης ανά παιχνίδι εκτιμάται στα 10 KB/s για την επικοινωνία σε πραγματικό χρόνο.

Βάσει αυτής της εκτίμησης, προτείνεται μια κλιμακωτή προσέγγιση για τις προδιαγραφές του Εικονικού Ιδιωτικού Διακομιστή (VPS): ξεκινώντας με 2 πυρήνες CPU, 4 GB RAM και 40 GB SSD στο πρώτο έτος (υποστηρίζοντας έως 150 ταυτόχρονα παιχνίδια), αυξανόμενο σταδιακά σε 4 πυρήνες CPU, 8 GB RAM και 100 GB SSD μέχρι το πέμπτο έτος (υποστηρίζοντας το στόχο των 500 ταυτόχρονων παιχνιδιών), με το μηνιαίο εύρος ζώνης να κλιμακώνεται από 2 TB σε 5 TB για την κάλυψη του αυξανόμενου όγκου δεδομένων.

Το εκτιμώμενο μηνιαίο κόστος φιλοξενίας προβλέπεται να αυξηθεί σταδιακά από περίπου 30€ το πρώτο έτος σε 90-100€ το πέμπτο έτος, αντικατοπτρίζοντας την αυξανόμενη ζήτηση πόρων. Η υλοποίηση ενός ολοκληρωμένου συστήματος παρακολούθησης και ανάλυσης από το δεύτερο έτος θα επιτρέψει την προληπτική διαχείριση της απόδοσης και την έγκαιρη προσαρμογή των πόρων. Αυτή η δυναμική προσέγγιση στην κλιμάκωση αποσκοπεί στην εξισορρόπηση μεταξύ της παροχής επαρκών πόρων για την υποστήριξη του αυξανόμενου αριθμού ταυτόχρονων παιχνιδιών και της διατήρησης της οικονομικής αποδοτικότητας, διασφαλίζοντας παράλληλα την ευελιξία για προσαρμογή σε απρόβλεπτες μεταβολές στη χρήση της εφαρμογής.

Παράρτημα Ε – Πλάνο ορθής λειτουργίας

Το πλάνο ελέγχου που ακολουθεί παρέχει μια ολοκληρωμένη προσέγγιση για τη διασφάλιση της ποιότητας και της αξιοπιστίας του παιχνιδιού Ludo. Καλύπτει όλες τις βασικές πτυχές, από τη βασική λειτουργικότητα και τη διεπαφή χρήστη μέχρι την απόδοση, την ασφάλεια και τη συμβατότητα.

Επιβεβαίωση ότι οι χρήστες μπορούν να κάνουν εγγραφή επιτυχώς	Ναι
Επιβεβαίωση ότι οι εγγεγραμένοι χρήστες μπορούν να συνδεθούν	Ναι
Επιβεβαίωση ότι οι παίκτες μπορούν να συνδεθούν επιτυχώς σε ένα παιχνίδι	Ναι
Επαλήθευση ότι το παιχνίδι ξεκινά σωστά με 2, 3 ή 4 παίκτες	Ναι
Επαλήθευση ότι το ταμπλό και τα πιόνια εμφανίζονται σωστά	Ναι

Επιβεβαίωση ότι η ρίψη ζαριού παράγει τυχαίους αριθμούς από 1 έως 6	Ναι
Έλεγχος ότι μόνο ο τρέχων παίκτης μπορεί να ρίξει το ζάρι	Ναι
Έλεγχος ότι ένα πιόνι μπορεί να βγει από τη βάση μόνο με 6	Ναι
Επαλήθευση ότι τα πιόνια κινούνται σωστά σύμφωνα με τον αριθμό του ζαριού	Ναι
Επιβεβαίωση ότι τα πιόνια ακολουθούν τη σωστή διαδρομή στο ταμπλό	Ναι
Έλεγχος ότι ένας παίκτης παίρνει επιπλέον γύρο όταν φέρει 6	Ναι
Επαλήθευση ότι ένα πιόνι "τρώει" σωστά το πιόνι αντιπάλου	Ναι
Έλεγχος ότι τα πιόνια δεν μπορούν να "φάνε" άλλα πιόνια σε ασφαλείς θέσεις	Ναι
Επιβεβαίωση ότι το παιχνίδι τερματίζει όταν ένας παίκτης φέρει όλα τα πιόνια του στο τέρμα	Ναι
Επαλήθευση ότι η γραφική διεπαφή ενημερώνεται σωστά με το πέρας του παιχνιδιού (rankings & profile win ratio)	Ναι
Επαλήθευση ότι η κατάσταση του παιχνιδιού συγχρονίζεται σωστά μεταξύ όλων των παικτών	Ναι
Παρακολούθηση της χρήσης CPU και μνήμης κατά τη διάρκεια του παιχνιδιού	Ναι
Μέτρηση του χρόνου εκκίνησης του παιχνιδιού	Ναι
Δοκιμή του παιχνιδιού με μεγάλο αριθμό ταυτόχρονων παιχνιδιών	Ναι
Επαλήθευση ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να συνδεθούν	Ναι
Επιβεβαίωση ότι οι παίκτες δεν μπορούν να τροποποιήσουν την κατάσταση του παιχνιδιού παράνομα	Ναι
Έλεγχος της συμβατότητας με διάφορους περιηγητές (Chrome, Firefox, Safari, Edge)	Ναι
Δοκιμή του παιχνιδιού σε παρατεταμένες περιόδους συνεχούς λειτουργίας	Ναι
Προσομοίωση διαφόρων σεναρίων σφαλμάτων (π.χ. διακοπή σύνδεσης)	Ναι